Milestones in Matrix Computation

The Selected Works of Gene H. Golub With Commentaries

RAYMOND H. CHAN, CHEN GREIF,
AND DIANNE P. O'LEARY

OXFORD SCIENCE PUBLICATIONS

Milestones in Matrix Computation

This page intentionally left blank

MILESTONES IN MATRIX COMPUTATION: SELECTED WORKS OF GENE H. GOLUB, WITH COMMENTARIES

Raymond H. Chan
Department of Mathematics, The Chinese University of Hong Kong

Chen Greif
Department of Computer Science, The University of British Columbia

Dianne P. O'Leary
Computer Science Department and Institute for Advanced Computer Studies,
University of Maryland

OXFORD
UNIVERSITY PRESS

OXFORD
UNIVERSITY PRESS

Great Clarendon Street, Oxford OX2 6DP

Oxford University Press is a department of the University of Oxford.
It furthers the University's objective of excellence in research, scholarship,
and education by publishing worldwide in

Oxford New York

Auckland Cape Town Dar es Salaam Hong Kong Karachi
Kuala Lumpur Madrid Melbourne Mexico City Nairobi
New Delhi Shanghai Taipei Toronto

With offices in

Argentina Austria Brazil Chile Czech Republic France Greece
Guatemala Hungary Italy Japan Poland Portugal Singapore
South Korea Switzerland Thailand Turkey Ukraine Vietnam

Published in the United States
by Oxford University Press Inc., New York

First published 2007

British Library Cataloguing in Publication Data
Data available

Library of Congress Cataloging in Publication Data
Data available

Typeset by Newgen Imaging Systems (P) Ltd., Chennai, India
Printed in Great Britain
on acid-free paper by
Biddles Ltd., King's Lynn, Norfolk

ISBN 978–0–19–920681–0

1 3 5 7 9 10 8 6 4 2

PREFACE

At a meeting in Moscow in June 2005, Gil Strang suggested that there be a collection of Gene Golub's work to highlight his many important contributions to numerical analysis. The three of us were honored to undertake this pleasant task, with publication timed for February "29", 2007, the 75th anniversary of Gene's birth.

Gene chose 21 papers to include here, and we are grateful to the publishers for permission to reprint these works. We asked each of the coauthors to write about how the paper came to be written. These short essays reveal a lot about Gene's working style – his quickness and creativity, his ability to draw together threads from diverse areas, and the beauty of his ideas. They also illustrate the serendipity of mathematical discovery and demonstrate that mathematics research can be done anywhere from an office to an amusement park.

Gene's work is broad as well as deep, and we have divided the papers into five groups: iterative methods for linear systems, solution of least squares problems, matrix factorizations and applications, orthogonal polynomials and quadrature, and eigenvalue problems. To put the work in context, we asked a leading expert to write a commentary on each group of papers, putting them into historical perspective. It is a testimony to the high regard in which Gene is held by his colleagues, that the first five people we contacted agreed to write these commentaries. We are very grateful to Anne Greenbaum, Åke Björck, Nicholas Higham, Walter Gautschi, and G. W. (Pete) Stewart; their careful work will be a great aid to numerical researchers now and in the future.

We are also pleased to be able to include a biography of Gene, drawn from conversations with him, as well as photos collected from Gene and his friends.

And so we present this volume as a gift to Gene, gathering some of his many important gifts to the community. We treasure his friendship, look forward to his 19th birthday in 2008, and wish him many more happy and productive years.

Raymond H. Chan
Chen Greif
Dianne P. O'Leary

This page intentionally left blank

CONTENTS

IV MATRIX FACTORIZATIONS AND APPLICATIONS

V ORTHOGONAL POLYNOMIALS AND QUADRATURE

VI EIGENVALUE PROBLEMS

LIST OF PLATES

(Plates 10–11 between pp. 438–439)

This page intentionally left blank

PART I

GENE H. GOLUB

This page intentionally left blank

1

GENE H. GOLUB BIOGRAPHY

Chen Greif

The Early Years, 1932–1953

Gene Howard Golub was born on February 29th, 1932 to Bernice and Nathan Golub. His mother was from Latvia and his father from Ukraine. They both came to the United States independently of one another in 1923, and both settled in Chicago for family reasons: they each had an older sibling in the city.

Gene was born in the depth of the Great Depression. He has one brother, Alvin, who is three years older and currently lives in Chicago. Gene's father worked as a "bread man". His mother stayed home the first few years. At the age of $4\frac{1}{2}$ his mother needed to work so she took a job in a shop, sewing baseball caps, and Gene was admitted to kindergarten and spent $1\frac{1}{2}$ years there. Quite a solid kindergarten education! Gene was a student at the Haugan Elementary School for nine years. He skipped a grade, but makes a point of saying that he was not an exceptional student. At the age of 12 he started working at his cousin Sidney's pharmacy as a delivery boy, and later as a soda jerk. It was not unusual in those years for children to have to work, although he started working a little earlier than his other friends. At that point, says Gene, he was sure he was headed in the pharmacy direction. Little did he know...

Gene had a fairly well developed Jewish identity as a child. He went to Hebrew school ("Heder") from 3:15pm to 5:30pm almost every day. He learned the Hebrew alphabet and a few things about Jewish culture. His parents were very Jewish-centric. They were not religious and did not keep kosher, but Jewish holidays were observed. Gene would not go to school on Rosh Hashana and Yom Kippur, and special holiday events like the Seder, the Passover dinner, were celebrated every year. Gene's mother, whose maiden name was Gelman, had a large family in Chicago, and there were many get-togethers which kept the family close.

Gene had his Bar Mitzvah in February 1945. He did not have a sense of what was going on immediately after the war in Europe, although later on, as an adult, the holocaust greatly influenced him, his view of the world, and his personal identity. But at the age of 13, life just went on. He attended the Theodore Roosevelt High School from 1945 to 1949. The school had quite a rigid course program, without much in the way of extra-curricular activities. Three years of mathematics included algebra and geometry but no calculus. Gene remembers this period as an unremarkable one. He just "went along" as a student. He played baseball, basketball, and football but was not passionate about them.

During Gene's high school period his mother and father divorced, and after the divorce Gene saw his father only a couple of times. In December 1948 his father died, a year after the divorce. Gene continued to work in the pharmacy, and also had a job at a large department store. Following high school, he attended a community college for two years: Wright Junior College. Gene fondly remembers this period in his life. He took a variety of courses and was pretty happy. He wanted to be a chemist; and he loved political science. Then came analytic geometry and calculus, and the fun began! The teachers were good, and he made some good friends. After the two years passed he decided to go to the University of Chicago. He was admitted as a junior, and worked on a degree in mathematics. It was a big change, and an hour and a half of commuting each way did not make it easier. It was then that Gene decided to go to the University of Illinois in Urbana-Champaign for his final undergraduate year: a life-changing decision.

The University of Illinois, 1953–1959

Gene enjoyed the school, and living in a small town. He took the usual required courses such as biology and French, along with a few other courses that changed his life. Among those, one was a course on matrix theory from Franz Hohn, a very good teacher and kind man. In his first year as a graduate student, Gene took a course from the famous statistician C.R. Rao, who was at Illinois for a year. It was an advanced course in multivariate statistics, but in fact Gene learned in that course more about matrices than about statistics. Block Gaussian elimination and other matrix algorithms were introduced, and the course helped Gene gain a knowledge of matrix manipulations.

Gene had a part-time job working for a physicist at the accelerator center. In the final semester he took a programming course in the mathematics department, and learned how to program for the ILLIAC. Professor John Nash offered him a position as an assistant at the computing lab. It was 1953 and times were a little different than today: not everything revolved around computing.

Gene's first task was to program Milne's method. Given the primitive computing environment, it was hard! And given the method's weak stability properties, the program Gene had written was not used extensively later on. Gene went on to write a lot of statistical applications. This included a variety of matrix algorithms, and he became very familiar with the library and started feeling very comfortable around matrices. Many of the computer programs were in "half precision" arithmetic: 20 bits, and 1,024 words of memory.

Gene remembers very fondly his days at the University of Illinois and the many friendships formed. Several of the people he met, such as Bill Gear who was also a student at the same time and ended up having the same PhD advisor as Gene, became lifelong friends. People were sociable, cultured, and liked music and books. There were superb people around in terms of academic ability: David Wheeler from Cambridge (the inventor of the subroutine) came from the UK and developed the basic libraries. Gene in fact never took a course in numerical analysis per se, but studied a great deal from the elegant programs of Wheeler simply by looking at the code and trying to understand it. Stanley Gill of the famed Runge–Kutta–Gill method then came. David Muller was a faculty member of the computing lab, and his method for solving nonlinear equations was very well known and a source of interest. In addition, Gene met Charles Wrigley, who was born in New Zealand and received his PhD in London. Wrigley was a psychometrician who was tremendously interested in computing and taught Gene about factor analysis. Through Wrigley Gene met Harry Harman, Louis Guttman, and many other distinguished psychometricians. It was a stimulating environment.

Gene was going to study statistics, but it was initially unclear under whose supervision. Bill Madow, whom he was considering, was on sabbatical, and Rao was around for a year but then left. Madow came back from California (where he had his sabbatical), but he eventually decided to go back to California. Abe Taub took Gene as a student. He was an applied mathematician, and had a close connection with John von Neumann. He gave Gene a paper written by von Neumann and others, about the use of Chebyshev polynomials in solving linear systems. As it turned out later, that paper had a decisive effect on Gene's research direction. So, even though he was working on a degree with a specialty in statistics, he ended up doing numerical analysis.

Gene's relationship with Taub was complex, but clearly Taub's influence on the direction Gene's career took was instrumental. Gene got financial support, attended conferences, and worked in the summers in various places. In summer 1955 he worked at the RAND corporation in California. He met George Dantzig and worked on the simplex method. Many other prominent people were around: Ken Arrow, Richard Bellman, David Blackwell, Herbert Scarf, and so on. The following summer he went to work for TRW in Los Angeles. His boss there was David Young, and George Forsythe as well as other prominent numerical analysts were around.

For his "6th" birthday in 1956 Gene had a special surprise: a few of his friends bought him a 1940 Chrysler with automatic transmission. We will reveal here that they paid the whopping amount of $50 for the car. Gene learned to drive and in summer 1956, while in California, he bought a Plymouth and drove it back to Illinois. In summer 1957 he worked at Bell Labs in New Jersey.

During his work on his thesis, Gene programmed the Chebyshev method and noticed that one of the parameters was converging, and then discovered it converged to the SOR parameter. The work of James Riley, whom he had met at TRW and who showed how the Richardson second order method was related to SOR, was helpful for Gene in simplifying the Chebyshev method for matrices with Property A.

Interestingly, while working on his thesis, Gene noticed that red/black ordering applied to a tridiagonal matrix had the property that the reduction leaves you with a tridiagonal matrix. It never appeared in his thesis. But it was an important idea which also prepared him for his singular value decomposition (SVD) work and the Fast Poisson Solver.

In 1959 Gene's advisor, Taub, invited Richard Varga to visit Illinois. Varga was a rising star, and when Gene talks about this he laughs and says that he suspects that Taub invited Varga to check him out! Gene and Varga discovered that they were working on similar things and Varga invited him to write a paper together. Collaboration started later on that year when the two met in Paris at a meeting. This led to what later turned out to be the first influential and major paper in Gene's career.

After the PhD, 1959–1962

Towards the end of his PhD studies, Gene applied for fellowships. He was awarded an NSF fellowship, and decided to go to Cambridge, England. He was at Cambridge for 15 months, from the spring of 1959 until July 1960. He renewed his acquaintance with Velvel Kahan who was also there as a postdoc. They were in a small office together for a while. It was a very nice experience: a period of "chilling out" after an intense term at Illinois. Velvel had a car and they drove around. They would often go to the National Physical Laboratory in London, where Jim Wilkinson was. One of the lectures was given by Cornelius Lanczos. It was there when Gene took note of the singular value decomposition. He may have heard about it earlier, but Lanczos was a great lecturer, and it stuck into Gene's head that you could use this decomposition. A few years later, the memory of this lecture would play a pivotal role in Gene's seminal work on computing the SVD.

While at Cambridge, Jim Snyder (a physicist who later became the head of the Computer Science Department at Illinois) mentioned that he was consulting at Berkeley and asked if Gene wanted to apply to the Berkeley National Laboratory. Gene did. He returned to the USA, bought a new car and drove off to Berkeley, to start his job in July 1960. It was a data analysis type of job, and Gene did not like it. But he did meet Paul Concus, and that began a long friendship and a collaboration. In December 1960, Gene decided to quit his job. In January 1961 he went down to Los Angeles to work again for TRW, which had become STL: Space Technology Laboratories. There were approximately a dozen mathematicians around, and consultants from Berkeley, UCLA, and other places were coming and going. But despite enjoying his job, Gene felt that he wanted eventually to be in a university.

The Stanford Years, 1962–

In the spring of 1962, STL sent Gene to a few places for recruiting. He went to Michigan, Wisconsin, Case, and in each of these places he himself was offered a job! In the meantime, he wrote to Forsythe and inquired about a position at Stanford. Forsythe wrote back, offering either a visiting assistant professorship, or a research associate position. Gene never received the letter. At some point Forsythe called him, to ask if Gene had the habit of answering his mail. Gene took on the visiting assistant professor position and started in the math department at Stanford in August 1962. Later on, Forsythe converted his position into a permanent one.

Gene fondly remembers the early years at Stanford: Forsythe was a magnet to other people, and there was an influx of visitors. Forsythe had wonderful students: Cleve Moler, James Ortega, Beresford Parlett, Jim Varah, and others. As Forsythe took on more administrative responsibilities, Gene took a lot of his responsibilities in the numerical analysis area. Gene acknowledges how much he learned from Forsythe about how to run things at a place like Stanford and how to aspire to be a good colleague and member in the community. Unfortunately, Forsythe died in 1972, after 15 years at Stanford. Gene describes Forsythe as an early founder of the numerical analysis community and praises his vision, wisdom and integrity.

The first years at Stanford marked Gene's rise to prominence. Ideas and papers were generated, collaborations and friendships formed. Gene graciously gives much of the credit for his success to his collaborators, their abilities, their collegiality, and their friendship. He describes the great work of Forsythe in forming the Computer Science Department of Stanford in 1966 (from the Computer Science Division of the Mathematics Department): Stanford was one of the first places to form such a department. People like John Herriot, John McCarthy and Donald Knuth came early on. Visitors like Jim Wilkinson, Peter Henrici, Germund Dahlquist and many others would come often, made life interesting and formed collaborations.

Gene went on to have a remarkable career. In three separate interviews and several hours of face to face and phone conversations, he vividly recalled many milestones. The success of his work on semi-iterative methods with Varga in the early 1960s, the computation of the SVD with Kahan in the mid-1960s and the fast Poisson solver in the early 1970s, followed with several milestone papers that have made an impact not only on the field of numerical linear algebra but on the broad areas of science and engineering in a variety of disciplines. His work on the preconditioned conjugate gradient method in the late 1970s (joint work with Paul Concus and Dianne O'Leary) helped popularize the method among large circles of scientists and practitioners. He put the total least squares problem on the map (joint work with Charlie Van Loan, after introducing the problem earlier in his work with Christian Reinsch). He worked on moments and quadrature rules with a variety of collaborators, work of great mathematical beauty. One of his latest contributions is his work on Google's PageRank algorithm; a technique for accelerating the convergence of the algorithm (joint work with Sep Kamvar, Taher Haveliwala and Christopher Manning) has received much attention.

The Birth of Papers

Gene has many anecdotes to offer on how some of his strongest papers came to life, and makes interesting connections that illustrate how some of his most important work started almost accidentally, just by way of paying attention to a comment, or resurrecting ideas that he had earlier in a different context.

An intriguing story is how the project on the computation of SVD came to life. We mentioned earlier that while in England Gene heard Lanczos speak on the SVD and kept it at a corner of his mind. Much later, in 1963, Ben Rosen talked at Stanford about computing pseudo-inverses via projections. At the end of the talk Forsythe got up and said, "Well, will somebody please figure out how to compute the pseudo-inverse of a matrix?!" Gene remembered Lanczos's lecture in this context. And it got him interested in the SVD. The combination of having heard Lanczos years earlier, and those stirring words, *marching orders*, by Forsythe, began an important component of Gene's career. He worked with Peter Businger, who was a research assistant, and he asked Peter to compute the eigenvalues of

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}.$$

The absolute values of the eigenvalues of this matrix are indeed the singular values of A. Peter put that into an eigenvalue routine and they saw zeros on the diagonal of the tridiagonal matrix that was generated. From studying David Young's work on Property A and from his own work on cyclic reduction Gene knew that one could reorder the matrix, and get a bidiagonal matrix. So Gene started thinking hard about ways to bidiagonalize the matrix, and figured out how to do it using left and right orthogonal transformations, while he visited Boeing in Seattle in the summer of 1963. Perhaps the fresh Pacific Northwest air helped with it.

In October 1963 there was a meeting at the University of Wisconsin. Gene saw Kahan and told him about the work he had been doing. Kahan was working on similar ideas, and they decided to collaborate. Kahan came to visit Stanford with Forsythe's support, and the famous Golub and Kahan paper was written.

Computing the singular values of a bidiagonal matrix efficiently came a little later. Gene thought that Householder transformations could do the trick. He and Christian Reinsch worked on this problem independently around the same time, and eventually an algorithm using QR with double shifts was published under joint authorship. In their paper the total least squares problem was also introduced. (The catchy name was given to it much later, by Charlie Van Loan.)

Years after the seminal work on the SVD, Paul Van Dooren was at Stanford and was scheduled to give a talk. In the audience were Wilkinson, Dahlquist, and Gene. At some point during the talk Paul asked, a little nervously: "Do you know what the SVD is?" The immediate answer came: "You are at the SVD Headquarters!" Nick Trefethen later designed a T-shirt with "SVD HQ" written on it.

The story of how the Fast Poisson Solver was born is also fascinating. Roger Hockney came as a research fellow to Stanford. He was working for Forsythe and the plasma physicist and engineer Oscar Buneman. He told Gene about the problems he was working on, that required solving a sequence of tridiagonal matrices. Gene remembered his playing with cyclic reduction as a graduate student. It was possible to apply cyclic reduction to a tridiagonal matrix, to get another tridiagonal matrix. Gene and Hockney realized that in fact a block version could be derived, as long as the blocks commute. Hockney programmed it but it seemed unstable if more than one step of cyclic reduction was carried out. Later on, Buneman gave a talk at Los Alamos and showed how to form the full procedure of cyclic reduction. He provided a two-page long program for doing it. This caused excitement. When Gene visited a little later, he and Buzbee were discussing it, when Clair Nielson came in and said he would like to use the method, but with different boundary conditions. This forced the three to sit down and deeply understand the method. Nielson came up with a way to solve the resulting difference equations. Later on, Alan George was very helpful and showed how to modify the right-hand side so that the method is stable. The paper of Buzbee, Golub and Nielson was at one point the most cited mathematics paper in the *SIAM Journal on Numerical Analysis.*

Buzbee, Dorr, George, and Golub went on to write their well-known paper on applying the solver to irregular domains. A package called FISHPACK, based on these ideas, was written by Paul Schwarztrauber and Roland Sweet. (Why "FISHPACK"? Translating the word "Poisson" from French might shed light on this mystery!)

The above mentioned work, along with an earlier paper of Gene with David Mayers at a conference in INRIA, have been an important part of the early advances of domain decomposition, and embedding techniques (fictitious domains). The rise of parallel computing at the time, and the attractive concept of subdividing a complex domain into simple subdomains, caught on thanks to work of Gene and many other people who played a pivotal role, such as Olof Widlund.

Another important work that followed was Gene's joint work with Concus and O'Leary on the preconditioned conjugate gradient (CG) method. In an earlier paper, Gene and Concus used the Chebyshev method for solving the Poisson equation as a means to solve the Helmholtz equation on a rectangular domain. Paul, Gene and Dianne presented the idea of using CG in its preconditioned version, and derived an elegant short algorithm that worked extremely well. In their paper the term "generalized CG" is actually used. The term "preconditioned CG" caught on later.

The Fast Poisson Solver and what followed it is a nice example of the serendipity of science: you never know where your research may take you. This work made a contribution to advances in domain decomposition techniques and in preconditioned iterative solvers, merely due to the need to find ways to overcome difficulties that arose in applying the solver, either due to a complicated computational domain, or due to a difficult underlying partial differential equation.

Finally, we mention the "accidental birth" of the work on moments and quadrature. Gene spent a year in 1965–66 at the Courant Institute. He arrived there shortly after meeting Dahlquist in Sweden and learning from him about the topic. Interestingly, a talk of Hans Weinberger given at Maryland that Gene had missed (and heard about from colleagues) about error bounds using residuals, stimulated his interest in error estimates, and he thought of ways to put it in the framework of moments. One day, he was sitting in Peter Lax's office. His mind started wandering. He picked up a book of Herbert Wilf, and started reading the section about Gaussian Quadrature. It was then and there that Gene realized that the weights were the squares of the first elements of the eigenvectors of the Jacobi matrix, obtained from orthogonal polynomials. (The eigenvalues were the nodes.) These mathematical facts may have been known to some experts, but Gene figured out how to do the computation: compute the weights by reorganizing the QR method. When recalling this story, Gene smiles and says that sometimes being "semibored" is all that it takes to make a discovery...

Service and the Book

Gene has a tremendous record of service to the scientific computing community. He served as President of SIAM, and played a central role in forming the International Council for Industrial and Applied Mathematics. He is the founding editor of two important SIAM journals – *SIAM Journal on Scientific Computing* and *SIAM Journal on Matrix Analysis* and *Applications*, and has been on a large number of advisory boards and committees. It's hard to imagine the community without the journals he founded, but their formation was not trivial and came after careful thought, taking into account the nature of existing journals in the field. Gene mentions a few little-known anecdotes. For example, he was influenced by Bellman when making the choice "matrix analysis." We should also mention that Gene founded the NA-NET and the NA-Digest, indispensable working and networking tools for many in the community for years to come.

Gene's devotion to his community has been demonstrated many times throughout the years in terms of time, energy and financial support. For his work on acceleration of the PageRank algorithm Gene received Google stock; he donated most of these funds to found the Paul and Cindy Saylor Chair at the University of Illinois. Gene says his gift is a way for him to acknowledge the important part that the university played in his life, and at the same time it was an opportunity to give tribute to Paul and Cindy for being such supportive friends that do so much for the academic community. Previous to this, Gene funded the Hohn/Nash student fellowship at the University of Illinois to honor two of his early mentors.

And how did the Golub and Van Loan book come about? Roger Horn was the founder of the Department of Mathematical Sciences at Johns Hopkins. In cooperation with Johns Hopkins Press, he held a series of short courses each documented in a monograph. He had invited Gene to teach one of these courses. Charlie Van Loan was there. They decided to try to write a monograph. As the book was being written, they wrote several papers together. The book now has three editions, has sold over 50,000 copies and has been cited over 10,000 times. According to Gene, Charlie Van Loan was the principal writer of the book and the force behind it, and Gene remembers the period of working with Van Loan on the book as a wonderful one.

People

When asked what has defined his career, Gene does not point out this or that paper, but rather he talks about people. He praises his students and says they have made him a better person. He admires their personalities, behavior, scholarship, and integrity, and says he feels fortunate to have met many of them. He attributes his hospitality and the endless number of parties he hosted at his home and back yard to the welcoming and open culture he had experienced at Illinois as a graduate student decades earlier. He also says that part of his success has to do with geography: people like coming to Stanford, they like the climate and the food and all the small pleasures that the area offers, and as a result the traffic of visitors has never stopped for over 40 years. There has been a lot of collaboration and a lot of scientific matchmaking at Stanford.

Gene has many anecdotes about his interaction with his colleagues and his students. Some of them are not necessarily related to inverting a matrix or computing eigenvalues but have made a great impact on Gene's life. Take email, for example. As the Computer Science Department grew, the numerical analysis group moved to the ground floor of Serra House, a building that had served as the home of the University President. The terminals were in the kitchen. "The person who shamed me into using email was Dan Boley," recalls Gene with a laugh. He came by and said "You are the only professor in the department who doesn't use email actively." Gene smiles and says that this statement was slightly exaggerated, but the seed was planted and there was no way back. Email became a big part of Gene's life.

Many of the first students who took Gene's courses are now very familiar names and great forces in the numerical community: Richard Bartels, Richard Brent, Jim Daniel, Alan George, Roger Horn, Victor Pereyra, Michael Saunders, Jim Varah, Margaret Wright, and others. He gives them credit for forcing him to "dot the i's and cross the t's" when he taught his first advanced numerical analysis course: this later helped in setting the stage for the Golub and Van Loan book.

Gene was married for a few years in the 1990s to Barbara Morris, whom he had met in England approximately 40 years earlier. His brother lives in Chicago (with a convenient escape route to Phoenix when winter hits). Most of Gene's waking hours in the last 50 or so years have been devoted to being with members of his extended family of the numerical linear algebra community.

Conclusion

This book's theme is about the impact that Gene's work has made in our field. Many more pages would be necessary to describe in full Gene's contributions. He has written many excellent papers, has become a member of the National Academy of Sciences and the National Academy of Engineers, founded *SIAM Journal on Scientific Computing* and *SIAM Journal on Matrix Analysis and*

Applications, was President of SIAM, founded the Scientific Computation and Computational Mathematics Program at Stanford, and has received ten honorary doctorates. But beyond all his honors, what makes Gene special is his dedication and his commitment to promoting numerical analysis and scientific computing, and his great support of young people.

In the last few years Gene has traveled a lot, being in many ways an ambassador of our community. We cherish the impact that Gene has had on our careers in so many ways: by hearing him give a talk that has made an impact in terms of selecting a research direction (or even a research career!), by accepting Gene's cordial invitation to visit at Stanford and experiencing that feeling of being at the "headquarters", or by meeting via Gene a colleague who has later become an important collaborator.

It is appropriate to close by going back to the roots: the "founding fathers" of numerical linear algebra. Gene has a lot of kind words to say about them. He singles out two seminal figures: Wilkinson and Householder. In particular, he takes time to talk about the role Jim Wilkinson had played. Wilkinson laid the foundations for pure numerical linear algebra, says Gene. He extracted the basic numerical problems and showed how to construct good numerical algorithms. Gene always felt that his own area of interest was in applied numerical linear algebra and that he was trying to take the lessons taught by Wilkinson and use them in different applications. That would include, for example, working out the stabilized version of the LU factorization for use in the simplex method (joint work with Richard Bartels). This work showed that one can get a reliable solution, and it is based on principles worked out by Wilkinson. "I see myself as an *applied Wilkinsonian*," says Gene. "Wilkinson is the one who really led the way for many in terms of error analysis, and by pointing out the important issues in matrix computations."

Acknowledgments

This biography summarizes several hours of interviews with Gene throughout 2006. I am grateful to Gene for sharing his time and memories with me. Many thanks also to Paul Saylor for his thorough reading of this biography and his constructive input, and to my co-editors for their helpful comments and suggestions.

2

PUBLICATIONS OF GENE H. GOLUB

Books

[B1] *Recent Advances In Numerical Analysis*, edited by Carl de Boor and Gene H. Golub, Academic Press (1978).

[B2] *Proceedings of Symposia In Applied Mathematics, Volume XXII, Numerical Analysis*, edited by Gene H. Golub and Joseph Oliger, American Mathematical Society (1978).

[B3] *Matrix Computations*, Gene H. Golub and Charles Van Loan, Johns Hopkins University Press (1983).

[B4] *Résolution Numérique Des Grands Systèmes Linéaires*, Gene H. Golub and Gérard A. Meurant, Editions Eyrolles (1983).

[B5] *Studies In Numerical Analysis*, edited by Gene H. Golub, The Mathematical Association of America (1984).

[B6] *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, edited by Roland Glowinski, Gene H. Golub, Gérard Meurant, and Jacques Périaux, SIAM (1988).

[B7] *Matrix Computations*, Gene H. Golub and Charles Van Loan, Johns Hopkins University Press, 2nd edition (1989).

[B8] *Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms*, edited by Gene H. Golub and Paul Van Dooren, NATO ASI Series, Series F: Computer and Systems Sciences, Vol. 70, Springer Verlag (1991).

[B9] *Scientific Computing and Differential Equations. An Introduction to Numerical Methods*, Gene H. Golub and James M. Ortega, Academic Press, Harcourt Brace Jovanovich (1992).

[B10] *Linear Algebra for Large Scale and Real-Time Applications*, edited by Marc S. Moonen, Gene H. Golub and Bart L. R. De Moor, NATO ASI Series, Kluwer Academic Publishers, Norwell, MA 02061 (1993).

[B11] *Scientific Computing. An Introduction with Parallel Computing*, Gene Golub and James M. Ortega, Academic Press (1993).

[B12] *Recent Advances in Iterative Methods*, edited by Gene H. Golub, A. Greenbaum and M. Luskin, Springer-Verlag (1993).

[B13] *Iterative Methods in Scientific Computing*, edited by Raymond H. Chan, Tony F. Chan, and Gene H. Golub, Springer-Verlag (1997).

[B14] *Matrix Computations*, Gene H. Golub and Charles Van Loan, Johns Hopkins University Press, 3rd edition (1996).

[B15] *Scientific Computing*, Editor-in-Chief Gene H. Golub, Springer (1997).

[B16] *Applications and Computation of Orthogonal Polynomials*, edited by W. Gautschi, Gene H. Golub and G. Opfer, Birkhäuser (1999).

[B17] *Iterative Methods and Matrix Computations*, Lectures of Instructor and Abstracts of Young Scientists. edited by Gene H. Golub and Lev A. Krukier. Rostov State University, Rostov-on-Don, Russia. (2–9 June 2002).

Thesis

[T1] *The use of Chebyshev matrix polynomials in the iterative solution of linear equations compared to the method of successive over-relaxation*, PhD Dissertation, University of Illinois (1959).

Conference papers and book chapters

[C1] *Matrix decompositions and statistical calculations*, in *Statistical Computation*, Academic Press, 365–397, 1969.

[C2] (with M. A. Saunders), *Linear least squares and quadratic programming*, in *Integer and Nonlinear Programming*, J. Abadie (ed.), North-Holland, 229–256, 1970.

[C3] (with R. H. Bartels and M. A. Saunders), *Numerical techniques in mathematical programming*, in *Nonlinear Programming*, Academic Press, 123–176, 1970.

[C4] *Direct methods for solving elliptic difference equations*, in *Symposium on the Theory of Numerical Analysis*, John L. Morris (ed.), Springer-Verlag, 1–19, 1971.

[C5] (with W. Dent), *Computation of the limited information maximum likelihood estimator*, in *Proceedings of the Computer Science and Statistics 6th Annual Symposium on the Interface*, M. Tarter (ed.), Western Periodicals, 60–65, 1973.

[C6] *Some uses of the Lanczos algorithm in numerical algebra*, in *Topics In Numerical Analysis*, John J. H. Miller (ed.), Academic Press, 173–184, 1973.

[C7] (with G. B. Dantzig, R. W. Cottle, B. C. Eaves, F. S. Hillier, A. S. Manne, D. J. Wilde and R. B. Wilson), *On the need for a System Optimization Laboratory*, in *Mathematical Programming*, Academic Press, New York, 1–32, 1973.

[C8] (with George P. H. Styan), *Some aspects of numerical computations for linear models*, in *Proceedings of the Computer Science and Statistics Seventh Annual Symposium on the Interface*, Iowa State University, 189–192, 1973.

[C9] (with Irwin Guttman and Rudolf Dutter), *Examination of pseudo-residuals of outliers for detecting spurosity in the general univariate linear model*, in *Multivariate Statistical Inference; Proceedings of The Research Seminar Dalhousie University, Halifax*, March 23–25, 1972, D. J. Kabe and R. P. Gupta (eds.), North-Holland, 63–108, 1973.

[C10] *Methods for computing eigenvalues of sparse matrix equations*, *Actas Del Seminario Sobre Metodos Numericos Moderno*, Volume I, V. Pereyra (ed.), Universidad Central de Venezuela, 127–148, 1974.

[C11*] (with Paul Concus), *A generalized conjugate gradient method for non-symmetric systems of linear equations*, *Second International Symposium on Computing Methods in Applied Sciences and Engineering*, Versailles, France, December 15–19; Lecture Notes in Economics and Mathematical Systems, vol. 134, Springer-Verlag, 56–65, (1976).

[C12] *Sparse matrix computations: Eigenvalues and linear equations*, *Seminaries Iria*, Inst. Rech. D'Informatique et d'Automatique, Rocquencourt, 117–140, 1975.

[C13] (with V. Pereyra), *Differentiation of pseudoinverses, separable nonlinear least squares problems, and other tales*, in *Generalized Inverses and Applications*, M. Nashed (ed.), Academic Press, 1976.

[C14*] (with Paul Concus and Dianne O'Leary), *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in *Proceedings of Sparse Matrix Conference, Sparse Matrix Computation*, J. R. Bunch and D. J. Rose, eds., Academic Press, 309–332, 1976.

[C15] (with Alan K. Cline and George W. Platzman), *Calculation of normal modes of oceans using a Lanczos method*, in *Proceedings of Sparse Matrix Conference, Sparse Matrix Computations*, Academic Press, 409–426, 1976.

[C16] (with F. Luk), *Singular value decomposition: applications and computations*, in *Proceedings of 1976 Army Numerical Analysis and Computers Conference.*

[C17*] (with R. Underwood), *The block Lanczos method for computing eigenvalues*, in *Proceedings of the Symposium on Mathematical Software*, University of Wisconsin, Madison, March 1977.

[C18] (with B. L. Buzbee and J. A. Howell), *Vectorization for the CRAY-1 of some methods for solving elliptic difference equations*, in *Proceedings of the Symposium on High Speed Computer and Algorithm Organization*, University of Illinois at Urbana-Champaign, April 1977.

[C19] (with D. Boley), *Inverse eigenvalue problems for band matrices*, in *Proceedings of Conference on Numerical Analysis*, Dundee, Scotland, June 1977.

[C20] (with D. Boley), *The matrix inverse eigenvalue problem for periodic Jacobi matrices*, in *Proceedings of the Fourth Conference on Basic Problems of Numerical Analysis*, (LIBLICE IV), Pilsen, Czechoslovakia, September 1978.

[C21] (with G. Dahlquist and S. Nash), *Bounds for the error in linear systems*, in *Proceedings of the Workshop on Semi-Infinite Programming*, Bad Honnef, West Germany, August 30–September 1, 1978.

[C22] (with C. Van Loan), *Total least squares*, in *Smoothing Techniques For Curve Estimation*, Proceedings, Heidelberg, 1979.

[C23] (with R. LeVeque), *Extensions and uses of the variable projection algorithm for solving nonlinear least squares problems*, in *Proceedings of the 1979 Army Numerical Analysis and Computers Conference, 1979.*

[C24] (with F. Luk and M. Pagano), *A large sparse least squares problem in photogrammetry*, in *Proceedings of the 12th Annual Symposium on the Interface*, University of Waterloo, May 1979.

[C25] (with R. Plemmons), *Sparse least squares problems*, in *Proceedings of the Iria Fourth International Symposium on Computing Methods in Applied Science and Engineering*, Versailles, France, December 1979.

[C26] (with M. Overton), *Convergence of a two-stage Richardson iterative procedure for solving systems of linear equations*, presented at the Dundee Biennial Conference on Numerical Analysis, University of Dundee, Scotland, June 23–26, 1981; in the Proceedings of the Conference, Springer-Verlag.

[C27] (with T. Chan and R. LeVeque), *Updating formulae and a pairwise algorithm for computing sample variances*, in the *Proceedings of the Computational Statistics 82 Conference*, Toulouse, France, September 1982.

[C28] (with D. Mayers), *The use of pre-conditioning over irregular regions*, presented at the *INRIA Sixth International Conference on Computing Methods in Applied Sciences and Engineering*, Versailles, France, 1983.

[C29] (with J. H. Ferziger and M. C. Thompson), *Block SOR applied to the cyclically-reduced equations as an efficient solution technique for convection–diffusion equations*, in Proceedings of *Computational Techniques and Applications: CTAC-87*, Elsevier Science and North Holland, 1987.

[C30] *Large scale least squares problems*, in *Computer Science and Statistics: Proceedings of the 19th Symposium on the Interface*, Philadelphia, PA, USA, 1987.

[C31] (with Robert J. Plemmons and Ahmed Sameh), *Parallel block schemes for large-scale least-squares computations*, in *High-Speed Computing: Scientific Applications and Algorithm Design*, 171–179, University of Illinois Press, 1988.

[C32] (with W. Gander, U. von Matt), *A constrained eigenvalue problem*, in *Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms. Proceedings of the NATO Advanced Study Institute.*, Leuven, Belgium, Aug. 1–12, 1988.

[C33] (with Howard Elman), *Line iterative methods for cyclically reduced non-self-adjoint elliptic problems*, *Proceedings of Seventh Army Conference on Applied Mathematics and Computing*, 1989.

[C34] (with Walter Gander and Dominik Gruntz) *Solving linear equations by extrapolation*, *Supercomputing*, J. S. Kovalik (ed.), Nato ASI Series, Springer, 1990.

[C35] (with John E. De Pillis), *Toward an effective two-parameter SOR method*, in *Proceedings of Iterative Methods for Large Linear Systems*, 107–119, Academic Press, 1990.

[C36] (with W. R. Ferng, R. J. Plemmons), *Adaptive Lanczos Methods for recursive condition estimation*, in *Proceedings of the SPIE — The International Society for Optical Engineering*, Vol. 1348, 326–37, 1990.

[C37] (with D. Boley), *A modified non-symmetric Lanczos algorithm and applications*, in *SVD and Signal Processing, II. Algorithms, Analysis and Applications.* Kington, RI, USA, 1990.

[C38] (with Sylvan Elhay and Jaroslav Kautsky), *Updating and downdating of orthogonal polynomials with data fitting applications*, *Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms*, edited by G.H. Golub and P. Van Dooren, NATO ASI Series, Vol. F. 70, 149–172, Springer, 1991.

[C39] (with A. Bojanczyk and P. Van Dooren), *The periodic Schur decomposition: Algorithms and applications*, in *SPIE*, 1770, 31–42, 1992.

[C40] (with D. Calvetti and L. Reichel), *Gaussian quadrature applied to adaptive Chebyshev Iteration*, in *Recent Advances in Iterative Methods*, G. Golub, A. Greenbaum, and M. Luskin (eds.), Springer-Verlag, 1993.

[C41] (with Bernd Fischer), *On the error computation for polynomial based iteration methods*, in *Recent Advances in Iterative Methods*, G. Golub, A. Greenbaum, and M. Luskin (eds.), Springer-Verlag, 1993.

[C42] (with R. W. Freund and N. M. Nachtigal), *Recent advances in Lanczos-based iterative methods for nonsymmetric linear systems*, in *Algorithmic Trends in Computational Fluid Dynamics*, M. Y. Hussaini, A. Kumar and M. D. Salas (eds.), Springer-Verlag, 1993.

[C43*] (with Gérard Meurant), *Matrices, moments and quadrature*, in *Proceedings of the 15th Dundee Conference, June-July 1993*, D. F. Griffiths and G. A. Watson (eds.), Longman Scientific & Technical, 1994.

[C44] (with Oliver Ernst), *A domain decomposition approach to solving the Helmholtz equation with a radiation boundary condition*, in *Contemporary Mathematics*, 157, of *Proceedings of the 6th International Domain Decomposition Meeting*, 177–192 (1994).

[C45] (with Hongyuan Zha), *The canonical correlations of matrix pairs and their numerical computation*, in *Linear algebra for signal processing* (Minneapolis, MN, 1992), 27–49, IMA Volumes in Mathematics and its Applications, 69, Springer, New York, 1995.

[C46] (with David Silvester and Andy Wathen), *Diagonal dominance and positive definiteness of upwind approximations for advection diffusion problems*, in *Numerical Analysis*, DF Griffiths and GA Watson (eds.), World Scientific, 125–131 (1996).

[C47] (with Urs Von Matt) *Generalized cross-validation for large scale problems*, in *Recent Advances in Total Least Squares Techniques and Errors-in-Variables Modeling*, Sabine Van Huffel (ed), SIAM, 139–148 (1997).

[C48] (with Henk A. van der Vorst), *150 years old and still alive: Eigenproblems, The State of the Art in Numerical Analysis*, I.S. Duff and G.A. Watson (eds), Clarendon Press, Oxford, 93–119 (1997).

[C49] (with Henk A. van der Vorst), *Closer to the solution: Iterative linear solvers, The State of the Art in Numerical Analysis*, I.S. Duff and G.A.Watson (eds), Clarendon Press, Oxford, 63–92 (1997).

[C50] (with Shivkumar Chandrasekaran, Ming Gu, Ali. H. Sayed) *Efficient algorithms for least squares type problems with bounded uncertainties*, in *Recent Advances in Total Least Squares Techniques and Errors-in-Variables Modeling*, Sabine Van Huffel (ed), SIAM, 171–182 (1997).

[C51] (with Urs von Matt), *Tikhonov regularization for large scale problems*, in *Scientific Computing*, Gene Howard Golub(ed), Springer, 3–26 (1997).

[C52] (with Knut Solna and Paul van Dooren), *Computing the SVD of a general matrix product/quotient*, *SIAM J. Matrix Anal. Appl.* 22 (2000), no. 1, 1–19.

[C53] (with Walter Gander), *Cyclic reduction – history and applications*, in *Scientific Computing*, Gene Howard Golub (ed), Springer, 73–85 (1997).

[C54] (with Michael Drexler), *Efficient solution of the Jacobian system in Newton's method close to a root*, in *Scientific Computing*, Gene Howard Golub (ed), Springer, 202–209 (1997).

[C55] (with N. Nguyen and P. Milanfar), *Preconditioners for regularized image superresolution*, in ICASSP '99, Vol. VI, pp. 3249–3252, Phoenix, AZ.

[C56] (with N. Nguyen and P. Milanfar), *Blind restoration/superresolution with generalized cross-validation using Gauss-type quadrature rules*, in *Proceedings of the 33rd Asilomar Conference on Signals, Systems, and Computers*, Monterey, CA, Oct. 24–27, 1999.

[C57] (with Zhaojun Bai), *Some unusual matrix eigenvalue problems*, in *Vector and Parallel Processing - VECPAR'98 Third International Conference*, Porto, Portugal, June 21–23, 1998. *Lecture Notes in Computer Science*, 1573, pp. 4–19, 1999.

[C58] (with Peyman Milanfar, Mihai Putinar, James Varah and Björn Gustafsson), *Shape reconstruction from moments: Theory, algorithms, and applications*, in *Proceedings of SPIE Vol.4116 on Advanced Signal Processing Algorithms, Architectures, and Implementations X*, San Diego, USA, August 2–4, 2000.

[C59] (with D. Vanderstraeten), *On the preconditioning of matrices with skew-symmetric splittings*, in *Numerical Algorithms*, Vol. 25, No. 1–4, p. 223–39, 2001.

[C60] (with B. Engquist), *From numerical analysis to computational science*, *Mathematics Unlimited*, 433–448, B. Enquist and W. Schmid (eds.), Springer, 2001.

[C61] (with Zhaojun Bai), *Computation of large-scale quadratic forms and transfer functions using the theory of moments, quadrature and Pade approximation* in *Modern Methods in Scientific Computing and Applications*, edited by A. Bourlioux and M. J. Gander, NATO Science Series, II. Mathematics, Physics and Chemistry, Vol. 75, 2002.

[C62] (with Yong Sun, Nate Folwell and Zenghai Li), *High precision accelerator cavity design using the parallel eigensolver Omega3P.* Winner of the 2002 ACES Best Paper Award.

[C63] (with M. J. Gander) *A non-overlapping optimized Schwarz method which converges with arbitrarily weak dependence on h. Domain decomposition methods in science and engineering*, 281–288 Natl. Auton. Univ. Mex., Mexico, 2003.

[C64] (with S. D. Kamvar, T. H. Haveliwala, and C. D. Manning), *Extrapolation methods for accelerating PageRank computations.* International World Wide Web Conference Proceedings of the 12th international conference on World Wide Web, ACM, Budapest, Hungary, 261–270, 2003.

[C65] (with D. B. Clayton, S. Skare, M. Modarresi, R. Bammer), *Augmented SENSE reconstruction by an improved regularization approach*, *Proceedings of the International Society for Magnetic Resonance in Medicine* 13th Annual Meeting, 2440, (2005).

Journal Publications

[J1*] (with R. S. Varga), *Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second-order Richardson iterative methods, Parts I and II*, *Numer. Math.* 3, 147–168 (1961).

[J2] *Bounds for the round-off errors in the Richardson second order method*, *BIT* 2, 212–223 (1962).

[J3] *Bounds for eigenvalues of tridiagonal symmetric matrices computed by the LR method*, *Math. Comp.* 16, 438–445 (1962).

[J4] *On the lower bound for the rank of a partitioned square matrix*, *Math. Comp.* 17, 186–188 (1963).

[J5] *Comparison of the variance of minimum variance and weighted least squares regression coefficients*, *Ann. Math. Statist.* 34, 984–991 (1963).

[J6*] *Numerical methods for solving linear least squares problems*, *Numer. Math.* 7, 206–216 (1965).

[J7] (with P. Businger), *Linear least squares solutions by Householder transformations*, *Numer. Math.* 7, 269–276 (1965).

[J8*] (with W. Kahan), *Calculating the singular values and pseudo-inverse of a matrix*, *SIAM J. Numer. Anal.* 2, Ser. B., 205–224 (1965).

[J9] (with G. E. Forsythe), *On the stationary values of a second-degree polynomial on the unit sphere*, *J. Soc. Indust. Appl. Math.* 13, 1050–1068 (1965).

[J10] (with J. H. Wilkinson), *Note on the iterative refinement of least squares solution*, *Numer. Math.* 9, 139–148 (1966).

[J11] (with A. Björck), *Iterative refinement of linear least squares solution by Householder transformation*, *BIT* 7, 322–337 (1967).

[J12] (with T. N. Robertson), *A generalized Bairstow algorithm*, *Comm. ACM* 10, 371–373 (1967).

[J13] *Least squares, singular values and matrix approximations*, *Appl. Mat.* 13, 44–51 (1968).

[J14] (with R. H. Bartels), *Stable numerical methods for obtaining the Chebyshev solution to an overdetermined system of equations*, *Comm. ACM* 11, 401–406 (1968).

[J15] (with R. H. Bartels), *Chebyshev solution to an overdetermined linear system*, An ALGOL algorithm, *Comm. ACM* 11, 428–430 (1968).

[J16*] (with R. H. Bartels), *The simplex method of linear programming using LU decomposition*, *Comm. ACM* 12, 266–268 (1969).

[J17] (with R. H. Bartels), *Simplex method procedure employing LU decomposition*, an ALGOL algorithm, *Comm. ACM* 12, 275–278 (1969).

[J18*] (with J. H. Welsch), *Calculation of Gauss quadrature rules*, *Math. Comp.* 23, 221–230 (1969).

[J19] (with P. A. Businger), *Singular value decomposition of a complex matrix*, Algorithm 358, *Comm. ACM* 12, 564–565 (1969).

[J20*] (with C. Reinsch), *Singular value decomposition and least squares solutions*, *Numer. Math.* 14, 403–420 (1970).

[J21*] (with B. L. Buzbee and C. W. Nielson), *On direct methods for solving Poisson's equation*, *SIAM J. Numer. Anal.* 7, 627–656 (1970).

[J22] (with B. L. Buzbee, F. W. Dorr, J. A. George), *The direct solution of the discrete Poisson equation on irregular regions*, *SIAM J. Numer. Anal.* 8, 722–736 (1971).

[J23] (with L. Smith), *Chebyshev approximation of continuous functions by a Chebyshev system of functions*, *Comm. ACM* 14, 737–746 (1971).

[J24] (with G. Dahlquist and S. Eisenstat), *Bounds for the error of linear systems of equations using the theory of moments*, *J. Math. Anal. Appl.* 37, 151–166 (1972).

[J25*] (with A. Björck), *Numerical methods for computing angles between linear subspaces*, *Math. Comp.* 27, 579–594 (1973).

[J26*] *Some modified matrix eigenvalue problems*, *SIAM Review* 15, 318–335 (1973).

[J27] (with G. P. H. Styan), *Numerical computations for univariate linear models*, *J. Statist. Comput. Simul.* 2, 253–274 (1973).

[J28*] (with V. Pereyra), *The differentiation of pseudo-inverses and non-linear least squares problems whose variables separate*, *SIAM J. Numer. Anal.* 10, 413–432 (1973).

[J29] (with P. Concus), *Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations*, *SIAM J. Numer. Anal.* 10, 1103–1120 (1973).

[J30] (with E. Seneta), *Computation of the stationary distribution of an infinite Markov matrix*, *Bull. Austral. Math. Soc.* 8, 333–341 (1973).

[J31] (with J. Varah), *On a characterization of the best ℓ_2 scaling of a matrix*, *SIAM J. Numer. Anal.* 11, 472–479 (1974).

[J32*] (with P. E. Gill, W. Murray and M. A. Saunders), *Methods for modifying matrix factorizations*, *Math. Comp.* 28, 505–535 (1974).

[J32] (with D. Fischer, O. Hald, C. Leiva and O. Widlund), *On Fourier–Toeplitz methods for separable elliptic problems*, *Math. Comp.* 28, 349–368 (1974).

[J33] (with E. Seneta), *Computation of the stationary distribution of an infinite stochastic matrix of special form*, *Bull. Austral. Math. Soc.* 10, 255–261 (1974).

[J34] *Bounds for matrix moments*, *Rocky Mountain J. Math.* 4, 207–211 (1974).

[J35] (with L. Jennings and W. Yang), *Waves in periodically structured media*, *J. Comp. Phys.* 17, 349–357 (1975).

[J37*] (with J. H. Wilkinson), *Ill-conditioned eigensystems and the computation of the Jordan canonical form*, *SIAM Review* 18, 578–619 (1976).

[J38] (with A. Björck), *Eigenproblems for matrices associated with periodic boundary conditions*, *SIAM Review* 19, 5–16 (1977).

[J39] (with P. Concus and D. O'Leary), *Numerical solution of nonlinear elliptic partial differential equations by generalized conjugate gradient method*, *Computing* 19, 321–339 (1978).

[J40*] (with C. de Boor), *The numerically stable reconstruction of a Jacobi matrix from spectral data*, *Linear Algebra and its Appl.* 21, 245–260 (1978).

[J41] (with R. W. Cottle and R. S. Sacher), *On the solution of large structured linear complementarity problems: the block partitioned case*, *Appl. Math. Optim.* 4, 347–363 (1978).

[J42] (with W. Langlois), *Direct solution of the equation for the Stokes stream function*, *Comp. Methods Appl. Mech. Engrg.* 19, 391–399 (1979).

[J43*] (with M. Heath and G. Wahba), *Generalized cross-validation as a method for choosing a good ridge parameter*, *Technometrics* 21, 2 215–223 (1979).

[J44] (with S. Nash and C. Van Loan), *A Hessenberg–Schur method for the problem $AX + XB = C$*, *IEEE Journal* 24, 6, 909–913 (1979).

[J45] (with C. Van Loan), *Unsymmetric positive definite linear systems*, *Linear Algebra and its Appl.* 28, 85–97 (1979).

[J46] (with R. Plemmons), *Large scale geodetic least squares adjustment by dissection and orthogonal decomposition*, *Linear Algebra and its Appl.* 34, 3–27 (1980).

[J47*] (with C. Van Loan), *An analysis of the total least squares problem*, *SIAM J. Numer. Anal.* 17, 6, 883–893 (1980).

[J48] (with R.C. Ward), *Numerical linear algebra*, *SIAM J. Numer. Anal.* 15, 3, 9–26 (1980).

[J49] (with F. Luk and M. Overton), *A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix*, *ACM Trans. Math. Software* 7, 149–169 (1981).

[J50] (with W. P. Tang), *The block decomposition of a Vandermonde matrix and its applications*, *BIT* 21, 505–517 (1981).

[J51] (with S. Nash), *Nonorthogonal analysis of variance using a generalized conjugate-gradient algorithm*, *J. Amer. Statist. Assoc.* 77, 109–116 (1982).

[J52] (with T. Chan and R. LeVeque), *Algorithms for computing the sample variance: analysis and recommendations*, *Statist. Comput.* 37, 242–247 (1983).

[J53] (with J. Kautsky), *Calculation of Gauss quadratures with multiple and free and fixed knots*, *Numer. Math.* 41, 147–163 (1983).

[J54] (with J. Kautsky), *On the calculation of Jacobi matrices*, *Linear Algebra and its Appl.* 52, 439–455 (1983).

[J55] (with D. Boley), *A modified method for reconstructing periodic Jacobi matrices*, *Math. Comp.* 42, 165, 143–150 (1984).

[J56] (with R. Gallant), *Imposing curvature restrictions on flexible functional forms*, *J. Econometrics* 26, 295–321 (1984).

[J57] (with D. L. Boley), *The Lanczos Arnoldi algorithm and controllability*, *Systems Control Lett.* 4, 317–324 (1984).

[J58] (with P. Concus and G. Meurant), *Block preconditioning for the conjugate gradient method*, *SIAM J. Scientific Statist. Comp.* 6, 220–252 (1985).

[J59] (with P. Manneback and Ph. L. Toint), *A comparison between some direct and iterative methods for large scale geodetic least squares problems*, *SIAM J. Scientific Statist. Comp.* 7, 799–816 (1986).

[J60] (with Carl D. Meyer, Jr.), *Using the QR factorization and group inversion to compute, differentiate, and estimate the sensitivity of stationary probabilities for Markov chains*, *SIAM J. Alg. Disc. Meth.* 7:2, 273–281 (1986).

[J61] (with R. Kannan), *Convergence of a two-stage Richardson process for nonlinear equations*, *BIT* 26, 209–216 (1986).

[J62] (with Alan Hoffman and G. W. Stewart), *A generalization of the Eckart–Young–Mirsky matrix approximation theorem*, *Linear Algebra and its Appl.* 88/89, 317–327 (1987).

[J63] (with D. Boley), *A survey of matrix inverse eigenvalue problems*, *Inverse Problems* 3, 595–622 (1987).

[J64] (with P. Arbenz), *On the spectral decomposition of Hermitian matrices modified by low rank perturbations with applications*, *SIAM J. Matrix Anal. Appl.* 9, 40–58 (1988).

[J65] (with P. Arbenz and W. Gander), *Restricted rank modification of the symmetric eigenvalue problem: theoretical considerations*, *Linear Algebra and its Appl.* 104, 75–95 (1988).

[J66] (with Michael L. Overton), *The convergence of inexact Chebyshev and Richardson iterative methods for solving linear systems*, *Numer. Math.* 53, 571–593 (1988).

[J67] (with Dianne P. O'Leary), *Some history of the conjugate gradient and Lanczos algorithms: 1948–1976*, *SIAM Review* 31, No. 1. 50–102 (1989).

[J68] (with W. Gander and U. Von Matt), *A constrained eigenvalue problem*, *Linear Algebra and its Appl.* 114, 815–839 (1989).

[J69] (with Mark Kent), *Estimates of eigenvalues for iterative methods*, *Math. Comp.* 53, 188, 619–626 (1989).

[J70] (with M. Gutknecht), *Modified moments for indefinite weight functions*, *Numer. Math.* 57, 607–624 (1989).

[J71] (with P. Comon), *Tracking a few extreme singular values and vectors in signal processing*, *IEEE Journal* 78, No. 8, 1327–1343 (1990).

[J72] (with H. Elman), *Iterative methods for cyclically reduced non-self-adjoint linear systems*, *Math. Comp.* 54:190, 671–700 (1990).

[J73] (with P. Arbenz), *QR-like algorithms for symmetric arrow matrices*, *SIAM J. Matrix Anal. Appl.* 13, 655–658 (1992).

[J74] (with Michael Berry), *Estimating the largest singular values of large sparse matrices via modified moments*, *Numer. Algorithms* (1), 353–374 (1991).

[J75] (with Daniel L. Boley, Sylvan Elhay and Martin H. Gutknecht), *Nonsymmetric Lanczos and finding orthogonal polynomials associated with indefinite weights*, *Numer. Algorithms*(1), No. 1, 21–43 (1991).

[J76] (with B. De Moor), *The Restricted singular value decomposition: properties and applications*, *SIAM J. Matrix Anal. Appl.* 12, 401–425 (1991).

[J77] (with S. Elhay and J. Kautsky), *Updating and downdating of orthogonal polynomials with data fitting applications*, *SIAM J. Matrix Anal. Appl.* 12, No. 2, 327–353 (1991).

[J78] (with H. Elman), *Iterative methods for cyclically reduced non-self-adjoint linear systems II*, *Math. Comp.* 56, No. 193, 215–242 (1991).

[J79] (with William R. Ferng and Robert J. Plemmons), *Adaptive Lanczos methods for recursive condition estimation*, *Numer. Algorithms*(1), No. 1, 1–20 (1991).

[J80] (with B. Fischer), *On generating polynomials which are orthogonal over several intervals*, *Math. Comp.* 56, No. 194, 711–730 (1991).

[J81] (with Roland W. Freund and Noël M. Nachtigal), *Iterative solutions of linear systems*, *Acta Numerica* , 1–44 (1991).

[J82] (with Urs Von Matt), *Quadratically constrained least squares and quadratic problems*, *Numer. Math.* 59, 561–580 (1991).

[J83] (with Daniel L. Boley, Richard P. Brent and Franklin T. Luk), *Algorithmic fault tolerance using the Lanczos process*, *SIAM J. Matrix Anal. Appl.* 13, 312–332 (1992).

[J84] (with Cheryl Carey, Hsin-Chu Chen and Ahmed Sameh), *A new approach for solving symmetric eigenvalue plems*, *Comput. Systems Enging*, 3:6, 671–679 (1992).

[J85] (with Sylvan Elhay and Jaroslav Kautsky), *Jacobi matrices for sums of weight functions*, *BIT* 32, 143–166 (1992).

[J86] (with H. C. Elman) *Line iterative methods for cyclically reduced discrete convection-diffusion problems*, *SIAM J. Scientific Statist. Comp.* 13, 339–363 (1992).

[J87] (with Bernd Fischer), *How to generate unknown orthogonal polynomials out of known orthogonal polynomials*, *J. Comput. Appl. Math.* 43, 99–115 (1992).

[J88] (with Bo Kagström and Paul Van Dooren), *Direct block tridiagonalization of single-input single-output systems*, in *Systems Control Lett.*, 18, 109–120 (1992).

[J89] (with Daniel Boley, Samy Makar, Nirmal Saxena, and Edward J. McCluskey), *Floating point fault tolerance with backward error assertions*, submitted for publication in *IEEE Trans. Comp.* 44, 302–311 (1993).

[J90] (with Zdeněk Strakoš), *Estimates in quadratic formulas*, in *Numer. Algorithms* 8, 241–268 (1994).

[J91] (with D. Calvetti and L. Reichel), *Adaptive Chebyshev iterative methods for nonsymmetric linear systems based on modified moments*, in *Numer. Math.* 67, 21–40 (1994).

[J92] (with H. C. Elman and G. Starke), *On the convergence of line iterative methods for cyclically reduced non-symmetrizable linear systems*, in *Numer. Math.* 67:2, 177–190 (1994).

[J93] (with Augustin A. Dubrulle), *A Multishift QR iteration without computation of the shifts*, *Numer. Algorithms* 7, 173–181 (1994).

[J94] (with Howard Elman), *Inexact and preconditioned Uzawa algorithms for saddle point problems*, *SIAM J. Numer. Anal.* 31 (1994), 1645–1661.

[J95] (with Walter Gander and Rolf Strebel), *Fitting of circles and ellipses least squares solution*, *BIT* 34, 558–578 (1994).

[J96] (with Guanghan Xu, Hongyuan Zha, and Thomas Kailath), *Fast algorithms for updating signal subspaces*, *IEEE Trans. Circuits Systems*, 41, 537–549 (1994).

[J97] (with Hongyuan Zha), *Perturbation analysis of the canonical correlations of matrix pairs*, *Linear Algebra Appl.*, 210, 3–28 (1994).

[J98] (with Cheryl M. M. Carey and Kincho H. Law), *A Lanczos-based method for structural dynamic re-analysis problems*, *Intl J. Numer. Methods Engr.*, 37, 16, 2857–2883 (1994).

[J99] (with Peter Arbenz), *Matrix shapes invariant under the symmetric QR algorithm*, *Numer. Linear Algebra Appl.* 2:2, 87–93 (1995).

[J100] (with S.H. Lui), *Homotopy method for the numerical solution of the eigenvalue problem of self-adjoint partial differential equations*, *Numer. Algorithms*, 10, 363–378 (1995).

[J101] (with Moody T. Chu and Robert E. Funderlic), *A rank-one reduction formula and its application to matrix factorizations*, *SIAM Review* 37, 512–530 (1995).

[J102] (with Zhaojun Bai and Mark Fahey), *Some large-scale matrix computation problems*, *J. Comput. Appl. Math.*, 74, 71–89 (1996).

[J103] (with Sowmini Varadhan and Michael Berry), *Approximating Dominant Singular Triplets of Large Sparse Matices via Modified Moments*, *Numer. Algorithms*, 13:1–2, 123–152 (1996).

[J104] (with Zhaojun Bai), *Bounds for the trace of the inverse and the determinant of symmetric positive definite matrices*, *Annals Numer. Math.*, 4, 29–38 (1997).

[J105] (with Urs von Matt), *Generalized cross-validation for large scale problems*, *J. Comput. Graph. Statist.*, 6:1, 1–34 (1997).

[J106] (with R. D. Fierro, P. C. Hansen, and D. P. O'Leary), *Regularization by truncated total least squares*, *SIAM J. Sci. Comput.* 18, 1223–1241 (1997).

[J107] (with S. Chandrasekaran, M. Gu, A. H. Sayed), *Parameter estimation in the presence of bounded modeling errors*, *IEEE Signal Process. Lett.*, 4, 195–197 (1997).

[J108] (with Moody T. Chu and Robert E. Funderlic), *On a variational formulation of the generalized singular value decomposition*, *SIAM J. Matrix Anal. Appl.* 18:4, 1082–1092 (1997).

[J109] (with G. Meurant), *Matrices, moments and quadrature II; how to compute the norm of the error in iterative methods*, *BIT* 37:3, 687–705 (1997).

[J110] (with T. Zhang and K.H. Law), *On the homotopy method for symmetric modified generalized eigenvalue problems*, *SIAM J. Sci. Comput.* 19 (1998), no. 5, 1625–1645.

[J111] (with S. Chandrasekaran, M. Gu and A. H. Sayed), *Parameter estimation in the presence of bounded data uncertainties*, *SIAM J. Matrix Anal. Appl.* 19, 235–252, (1998).

[J112] (with T. Zhang, and K.H. Law), *Eigenvalue perturbation and generalized Krylov subspace method*, *IMACS J. Appl. Numer. Math.*, 27 (1998) 185–202.

[J113] (with Andrew J. Wathen), *An iteration for indefinite systems and its application to the Navier–Stokes equations*, *SIAM J. Scientific Comput.*, 19:2, 530–539 (1998).

[J114] (with Lan Chieh Huang, Horst Simon and Wei-Pai Tang), *A fast Poisson solver for the finite difference solution of the incompressible Navier–Stokes equations*, *SIAM J. Scientific Comput.*, 19:5, 1606–1624 (1998).

[J115] (with K. H. Law and T. Zhang), *On the homotopy method for perturbed symmetric generalized eigenvalue problems*, *SIAM J. Scientific Comput.*, 19:5, 1625–1645 (1998).

[J116] (with Lars-Erik Andersson and Tommy Elfving), *Solution of biharmonic equations with application to radar imaging*, *J. Comp. Appl. Math.*, 94, 153–180 (1998).

[J117] (with Eldar Giladi and Joseph B. Keller), *Inner and outer iterations for the Chebyshev algorithm*, *SIAM J. Numer. Anal.* 35, 300–319 (1998).

[J118] (with S. Chandrasekaran, M. Gu, and A. H. Sayed), *Parameter estimation in the presence of bounded data uncertainties*, *SIAM J. Matrix Anal. Appl.* 19, 235–252 (1998).

[J119] (with J. Baglama, D. Calvetti, and L. Reichel), *Adaptively preconditioned GMRES algorithms*, *SIAM J. Scientific Comput.*, 20:1, 243–269 (1998).

[J120] (with Tony F. Chan and Pep Mulet), *A nonlinear primal-dual method for total variation-based image restoration*, *SIAM J. Scientific Comput.*, 20:6, 1964–1977 (1999).

[J121] (with D. Calvetti and L. Reichel), *A computable error bound for matrix functionals*, *J. Comput. Appl. Math.*, 103, pp. 301–306, (1999).

[J122] (with Moody Chu and Robert Funderlic), *Rank modifications of semi-definite matrices with applications to secant updates*, *SIAM J. Matrix Anal. Appl.* 20, 428–436, (1999).

[J123] (with T.Zhang and K.H. Law), *Subspace iterative methods for eigenvalue problems*, *Linear Algebra and its Appl.* 294, 239–258, Elsevier (1999).

[J124] (with M. Benzi), *Bounds for the entries of matrix functions with applications to preconditioning*, *BIT* 39:3, 417–438 (1999).

[J125] (with D. Calvetti and L. Reichel), *A computable error bound for matrix functionals*, *J. Comput. Appl. Math.* 103, no. 2, 301–306 (1999) (Reviewer: Wasin So).

[J126] (with D. Calvetti and L. Reichel), *Estimation of the L-curve via Lanczos bidiagonalization*, *BIT* 39:4, 603–619 (1999).

[J127] (with P.C. Hansen and D.P. O'Leary), *Tikhonov regularization and total least squares*, *SIAM J. Matrix Anal. Appl.* 21, 185–194 (1999).

[J128] (with S. Elhay, G. M. L. Gladwell, Y. M. Ram), *On some eigenvector–eigenvalue relations*, *SIAM J. Matrix Anal. Appl.* 20, no. 3, 563–574 (1999).

[J129] (with S. Chandrasekaran, M. Gu, A. H. Sayed), *An efficient algorithm for a bounded errors-in-variables model*, *SIAM J. Matrix Anal. Appl.* 20, no. 4, 839–859 (1999).

[J130] (with Qiang Ye), *Inexact preconditioned conjugate gradient method with inner–outer iteration*, *SIAM J. Sci. Comput.* 21, no. 4, 1305–1320, (1999/00).

[J131] (with Peyman Milanfar and James Varah), *A stable numerical method for inverting shape from moments*, *SIAM J. Sci. Comput.* 21, no. 4, 1222–1243 (1999/00).

[J132*] (with D. Calvetti, W.B. Gragg and L. Reichel), *Computation of Gauss–Kronrod quadrature rules*, *Math. Comp.* 69, no. 231, 1035–1052 (2000).

[J133] (with Henk A. van der Vorst), *Numerical progress in eigenvalue computation in the 20th century*, *J. Comput. Appl. Math.* 123, 35–65 (2000).

[J134] (with Malcolm F. Murphy and Andrew J. Wathen), *A note on preconditioning for indefinite linear systems*, *SIAM J. Sci. Comput.* 21, no. 6, 1969–1972 (2000).

[J135] (with Q. Ye), *Inexact inverse iteration for generalized eigenvalue problems*, *BIT* 40, Issue 4, pp. 671–684 (2000).

[J136] (with Denis Vanderstraeten), *On the preconditioning of matrices with skew-symmetric component*, *Numer. Algorithms* 25: 223–239 (2000).

[J137] (with Z. Zhang and H. Zha), *Large sparse symmetric eigenvalue problems with homogeneous linear constraints: the Lanczos process with inner–outer iterations*, *Linear Algebra and its Appl.* 309, 289–306 (2000).

[J138] (with K. Solna, P. van Dooren), *Computing the SVD of a general matrix product/quotient*, *SIAM J. Matrix Anal. Appl.* 22, 1–19, (2001).

[J139] (with Ahmed Sameh and Vivek Sarin), *A parallel balance scheme for banded linear systems Numer. Linear Algebra Appl.*, 5, 297–316, (2001).

[J140] (with Nhat Nguyen, P. Milanfar), *A computationally efficient superresolution image reconstruction algorithm.*, *IEEE Trans. Image Process.*, 10, 573–583 (2001).

[J141] (with N. Nguyen, P. Milanfar), *Efficient generalized cross-validation with application to parametric image restoration and resolution enhancement*, *IEEE Trans. Image Process.*, 10, pp. 1299–1308 (2001).

[J142] (with H. Melboe), *A stochastic approach to error estimates for iterative linear solvers, Part I*, *BIT* 41, (*Suppl*), 977–985 (2001).

[J143] (with X. Wu and Jin-Yun Yuan), *SOR-like methods for augmented systems*, *BIT*, 41(1), 71–85 (2001).

[J144] (with T. Zhang), *Rank-one approximation to higher order tensors*, *SIAM J. Matrix Anal. Appl.* 23, 534–550 (2001).

[J145] (with M. T. Chu), *Structured inverse eigenvalue problems*, *Acta Numerica*, 1–71 (2002).

[J146] (with Paul Concus and Yong Sun), *Object-oriented parallel algorithms for computing three-dimensional isopyncal flow*, *Inter. J. Numer. Meth. Fluids*, 39: 585–605 (2002).

[J147] (with C. Greif and J.M. Varah), *Block orderings for tensor-product grids in two and three dimensions*, *Numer. Algorithms*, 30(2), 93–111 (2002).

[J148] (with Jin-Yun Yuan), *Symmetric-triangular decomposition and its applications, Part I: Theorems and algorithms*, *BIT* 42, 4, 814–822 (2002).

[J149] (with S. Elhay and Y. M. Ram), *The spectrum of a modified linear pencil*, *Comput. Math. Appl.* 46, 1413–1426 (2003).

[J150] (with Victor Pereyra), *Separable nonlinear least squares: The variable projection method and its applications*, *Inverse Problems 19(2)*, R1-R26 (2003).

[J151] (with C. Greif), *On solving block-structured indefinite linear systems*, *SIAM J. Sci. Comput.* 24(6), 2076–2092 (2003).

[J152] (with J. Orchard, C. Greif, B. Bjornson, M. S. Atkins), *Simultaneous registration and activation detection for fMRI*, *IEEE Trans. Medical Imaging* 22(11), 1427–1435 (2003).

[J153] (with M. Benzi and M. J. Gander), *Optimization of the Hermitian and skew-Hermitian splitting iteration for saddle-point problems*, *BIT Numerical Mathematics* 43, 881–900 (2003).

[J154*] (with Zhong-Zhi Bai and Michael K. Ng), *Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems*, *SIAM J. Matrix Anal. Appl.* 24, 603–626 (2003).

[J155] (with Orly Alter), *Integrative analysis of genome-scale data by using pseudo-inverse projection predicts novel correlation between DNA replication and RNA transcription*, in *Proc. National Acad. Sciences*, 101, 16577–16582 (2004).

[J156] (with Michele Benzi), *A preconditioner for generalized saddle point problems*, *SIAM J. Matrix Anal. Appl.* 26(1) 20–41 (2004).

[J157] (with Diana M. Sima and Sabine Van Huffel), *Regularized total least squares based on quadratic eigenvalue problem solvers*, *BIT* 44, 793–812 (2004).

[J158] (with Michael Elad and Peyman Milanfar), *Shape from moments — an estimation theory perspective*, *IEEE Trans. Signal Process.* 52, 1814–1829 (2004).

[J159] (with Zhong-Zhi Bai and Jian-Yu Pan), *Preconditioned Hermitian and skew-Hermitian splitting methods for non-Hermitian positive semidefinite linear systems*, *Numer. Math.* 98, 1–32 (2004).

[J160] (with Sepandar Kamvar and Taher Haveliwala), *Adaptive methods for the computation of PageRank*, *Linear Algebra and its Appl.* 386, 51–65 (2004).

[J161] (with J. Y. Yuan, R. J. Plemmons, and W. A. G. CecÃlio), *Semi-conjugate direction methods for real positive definite systems*, BIT 44 189–207 (2004).

[J162] (with Oren E. Livne), *Scaling by binormalization*, *Numer. Algorithms* 35, 97–120 (2004).

[J163] (with R. Vandebril, M. Barel, and N. Mastronardi), *A bibliography on semiseparable matrices*, *Calcolo 43*, 249–270 (2005).

[J164] (with G. Boutry, M. Elad, and P. Milanfar), *The generalized eigenvalue problem for nonsquare pencils using a minimal perturbation approach*, *SIAM J. Matrix Anal. Appl.* 27, 582–601 (2005).

[J165] (with Y. Tsaig, M. Elad, P. Milanfar), *Variable projection for near-optimal filtering in low bit-rate block coders*, *IEEE Trans. Circuits Systems Video Technol.* 15(1) 154–160 (2005).

[J166] (with U. B. Holz, K. H. Law), *A subspace approximation method for the quadratic eigenvalue problem*, *SIAM J. Matrix Anal. Appl.* 26(2), 498–521 (2005).

[J167] (with Daniele Bertaccini, Stefano Serra Capizzano and Cristina Tablino Possio), *Preconditioned HSS methods for the solution of non-Hermitian positive definite linear systems and applications to the discrete convection–diffusion equation*, *Numer. Math.* 99(3), 441–484 (2005).

[J168] (with Stefano Serra-Capizzano and Daniele Bertaccini), *How to deduce a proper eigenvalue cluster from a proper singular value cluster in the non-normal case*, *SIAM J. Matrix Anal. Appl.* 27, 82–86 (2005).

[J169] (with Chen Greif and James M. Varah), *An algebraic analysis of block diagonal preconditioner for saddle point systems*, *SIAM J. Matrix Anal. Appl.* 27, 779–792 (2005).

[J170] (with Michele Benzi and Jrg Liesen), *Numerical solution of saddle point problems*, *Acta Numer.* 14, 1–137 (2005).

[J171] (with Zhong-Zhi Bai, Lin-Zhang Lu, and Jun-Feng Yin), *Block triangular and skew-Hermitian splitting methods for positive-definite linear systems*, *SIAM J. Scientific Statist. Comp.* 26, 844–863 (2005).

[J172] (with Annie Cuyt, Peyman Milanfar, and Brigitte Verdonk), *Multidimensional integral inversion, with applications in shape reconstruction*, *SIAM J. Scientific Statist. Comp.* 27, 1058–1070 (2005).

[J173] (with Orly Alter), *Reconstructing the pathways of a cellular system from genome-scale signals by using matrix and tensor computations*, *Proc. National Acad. Sciences*, 102, p. 17559–17564 (2005).

[J174] (with Li-Zhi Liao), *Continuous methods for extreme and interior eigenvalue problems*, *Linear Algebra and its Appl.* 415, 31–51 (2006).

[J175] (with Mike A. Botchev), *A class of nonsymmetric preconditioners for saddle point problems*, *SIAM J. Matrix Anal. Appl.* 27, 1125–1149 (2006).

An asterisk denotes a paper reprinted in this volume.

3

MAJOR AWARDS

[A1] March, 1958, National Science Foundation Postdoctoral Fellowship for study at Cambridge University, Cambridge, England.

[A2] Fellow of AAAS, 1981.

[A3] Award for Leadership in Numerical Analysis (Forsythe Lecturer), 1978.

[A4] Honorary Fellow, St. Catherine's College, Oxford, 1983.

[A5] Alumni Honor Award for Distinguished Service, University of Illinois at Urbana-Champaign, 1983.

[A6] Honorary Degree: Teknologie Doctor, Linköping University, 1984.

[A7] Foreign Member, Royal Swedish Academy of Engineering Sciences, Stockholm, Sweden, 1986.

[A8] Honorary Degree: Docteur Honoris Causa, Counsel of the Scientific, Technological and Medical University of Grenoble, 1986.

[A9] Honorary Degree: University of Waterloo, 1987.

[A10] Honorary Member, Institute of Electrical and Electronics Engineering, 1987.

[A11] Honorary Degree: Doctor of Laws, The University of Dundee, 1987.

[A12] Guggenheim Fellowship, 1987–1988 Academic Year.

[A13] Award for Distinguished Service to the Profession, Society for Industrial and Applied Mathematics, 1988.

[A14] Member, National Academy of Engineering, February 1990.

[A15] Honorary Degree: Doctor of Science, University of Illinois, 1991.

[A16] Award from the Committee for International Conferences on Applied and Industrial Mathematics to recognize "outstanding contributions to the international industrial and applied mathematics community", 1991.

[A17] Fletcher Jones Professor of Computer Science, 1991.

[A18] Honorary Degree, Universitat Louvain, Belgium, 1992.

[A19] Honorary Member, International Statistical Institute, Netherlands, 1992.

[A20] Member, National Academy of Sciences, April 1993.

[A21] Member, American Academy of Arts and Sciences, March 1994.

[A22] B. Bolzano Gold Medal for Merit in the Field of Mathematical Sciences, Academy of Sciences of the Czech Republic, March 1994.

[A23] Honorary Degree, University of Umea, Sweden, September 1995.

[A24] Honorary Degree, Australian National University, 1996.

[A25] Honorary Degree, Rostov State University, Rostov, Russia, May 2002.

[A26] Elected to Hall of Fame for Engineering, Science and Technology, July 2002.

[A27] Honorary Editor, BIT, November 2002.

[A28] Honorary Degree, Hong Kong Baptist University, December 2002.

[A29] Distinguished Israel Pollak Lecturer, 2004.

[A30] Memorial Medal of the Faculty, Charles University Prague, August 2004.

4

STUDENTS OF GENE H. GOLUB

Roger Hockney	1966
Richard Bartels	1968
Michael Jenkins	1969
Lyle Smith	1969
George Ramos	1970
Richard Brent	1971
Michael Saunders	1972
John Palmer	1974
Richard Underwood	1975
Dianne Prost O'Leary	1976
John Lewis	1976
Margaret Wright	1976
Michael Heath	1978
Franklin Luk	1978
Michael Overton	1979
Petter Bjørstad	1980
Daniel Boley	1981
Eric Grosse	1981
Stephen Nash	1982
Mark Kent	1989
Ray Tuminaro	1990
Hongyuan Zha	1993
Oliver Ernst	1994
Xiaowei Zhan	1997
Tong Zhang	1998
Nhat Nguyen	2000
Urmi Holz	2002
Yong Sun	2003
James Lambers	2003

PART II

ITERATIVE METHODS FOR LINEAR SYSTEMS

This page intentionally left blank

5

COMMENTARY, BY ANNE GREENBAUM

From the very early days, Gene Golub has been a driving force in the development and analysis of iterative methods for solving large sparse linear systems – problems for which Gaussian elimination is often prohibitive in terms of both storage and computation time. We review five of his seminal papers in this field.

Chebyshev semi-iterative methods, successive over-relaxation iterative methods, and second-order Richardson iterative methods, Parts I and II, by Golub and Varga [10]

This paper is probably less well-known today than it should be. In it the authors show the remarkable similarity between the Chebyshev semi-iterative method, the successive overrelaxation (SOR) method applied to an expanded matrix equation, and the second-order Richardson iterative method. They conclude that the Chebyshev semi-iterative method is to be preferred over the other two, since its iteration matrix has the smallest spectral norm, while the work per iteration is the same as that for the other methods. They present numerical results with the different methods used to solve elliptic difference equations.

The authors start with the simple iterative method

$$\vec{\alpha}^{(i+1)} = B\vec{\alpha}^{(i)} + \vec{g},$$

which has been derived from the linear system $A\vec{x} = \vec{k}$ through a matrix splitting, giving rise to a convergent iteration matrix B; that is $\rho(B) < 1$, where $\rho(\cdot)$ denotes the spectral radius. They assume that the iteration matrix B (which today might be called the *preconditioned* matrix) is real symmetric and positive definite. From this they derive the Chebyshev semi-iterative method by taking a linear combination of the iterates $\vec{\alpha}^{(i)}$ and choosing the coefficients so that the error in the linear combination is

$$\vec{\epsilon^*}^{(i)} = \tilde{p}_i(B)\vec{\epsilon}^{(0)},$$

where $\tilde{p}_i$ is the Chebyshev polynomial for the interval $[-\rho(B), \rho(B)]$, normalized so that $\tilde{p}_i(1) = 1$.

Next they consider the SOR method applied to the expanded linear system

$$\begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} = \begin{pmatrix} 0 & B \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} + \begin{pmatrix} \vec{g} \\ \vec{g} \end{pmatrix},$$

whose solution for $\vec{x}$ and $\vec{y}$ is the same as that of the original linear system, and whose iteration matrix,

$$J \equiv \begin{pmatrix} 0 & B \\ B & 0 \end{pmatrix},$$

has the same spectral radius as B. Since this matrix is real, symmetric, cyclic, and consistently ordered, the standard SOR theory due to Young [19] is applied to determine the optimal relaxation parameter: $\omega_b = 1/\sqrt{1-\rho^2(B)}$. Defining the vectors $\vec{\zeta}^{(2l)} \equiv \vec{x}^{(l)}$ and $\vec{\zeta}^{(2l+1)} \equiv \vec{y}^{(l)}$, the authors show that these vectors satisfy a recurrence of the same form as that for the Chebyshev iterates, except that in this case the parameter ω is fixed while in the Chebyshev method the parameter

$$\omega_{i+1} = \frac{1}{1-\rho^2\omega_i/4}, \quad i \geq 2, \quad \omega_1 = 1, \quad \omega_2 = \frac{2}{2-\rho^2},$$

changes at each iteration. Even more interestingly, they demonstrate that the limit of the Chebyshev parameters is equal to the optimal SOR parameter ω_b:

$$\lim_{m\to\infty} \omega_m = \omega_b.$$

Finally the authors consider the Richardson iterative method:

$$\vec{\eta}^{(m+1)} = \vec{\eta}^{(m)} + \alpha[B\vec{\eta}^{(m)} + \vec{g} - \vec{\eta}^{(m)}] + \beta[\vec{\eta}^{(m)} - \vec{\eta}^{(m-1)}], \quad m \geq 1,$$

where $\vec{\eta}^{(0)}$ and $\vec{\eta}^{(1)}$ are given initial guesses and α and β are fixed acceleration parameters. Using the analysis of Frankel and Riley [6,15,19], they demonstrate that with the best choice for the parameters α and β, this method is also equivalent to the SOR method applied to the extended linear system, with $\omega = \omega_b$. Thus, they have established the high degree of similarity among the three methods.

Writing the error at step m of each method in the form $q_m(B)\vec{\epsilon}^{(0)}$ for a certain polynomial q_m, the authors show that the Chebyshev polynomial $\tilde{p}_m(B)$ has smaller spectral norm than the polynomial $r_m(B)$ associated with the SOR method (or equivalently the second-order Richardson iteration), when $\vec{\zeta}^{(0)}$ is arbitrary and $\vec{\zeta}^{(1)} = B\vec{\zeta}^{(0)} + \vec{g}$, as well as with several other starting strategies. This establishes the superiority of the Chebyshev method in reducing the error at each step, assuming the worst-case initial error. Numerical experiments in part II of the paper illustrate this result in practice for cyclic matrices. Since the method requires the same amount of work per iteration as the others, the authors conclude that it is to be preferred.

A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations, by Concus, Golub, and O'Leary [5]

In this paper and in a paper with Concus [4], Golub turns his attention from the Chebyshev method to the conjugate gradient algorithm. In fact, these were two very influential papers helping to revive interest in the conjugate gradient algorithm, long after its invention in the early 1950s [12]. Another early paper showing the advantages of the conjugate gradient algorithm was written by Reid [14]. Never content to simply develop iterative methods and prove theorems about their convergence, Golub made it a personal mission to see that the engineers, physicists, statisticians, and others who were actually solving large linear systems learned about this new method and implemented it in their codes. The result was a revolution in the way large linear systems were being solved. I was a beginning graduate student, working at Lawrence Livermore National Laboratory at the time, and I saw the physics codes at the Lab quickly change from an ADI (alternating direction implicit) philosophy of iterating with PDEs (partial differential equations) to rigorously solving the large systems of difference equations using the conjugate gradient algorithm. That change (along with the implementation of more advanced preconditioners) remains in effect today.

Although [4] is dated slightly earlier, I will begin with [5] since it describes the basic *preconditioned conjugate gradient* (CG) algorithm and its properties (although the authors did not use the word "preconditioned" at the time but referred to it as the "generalized conjugate gradient algorithm"). In this paper it is assumed that the coefficient matrix A of the linear system $Ax = b$ is real symmetric and positive definite. The authors – Paul Concus, Gene Golub, and Dianne O'Leary – first point out the advantages of the CG algorithm when used as an iterative method:

- It does not require estimation of parameters.
- It takes advantage of the distribution of the eigenvalues of the iteration matrix.
- It requires fewer restrictions on the matrix A for optimal behavior than do such methods as SOR.

The paper includes a derivation of the preconditioned CG algorithm and its optimality property: The A-norm of the error at step $\ell + 1$ satisfies

$$\|e^{(\ell+1)}\|_A = \min_{p_\ell} \|(I - Kp_\ell(K))e^{(0)}\|_A,$$

where K is the preconditioned matrix $M^{-1}A$, and the minimum is over all polynomials p_ℓ of degree ℓ. The authors note that comparison with the Chebyshev polynomial gives the upper bound

$$\frac{\|e^{(\ell+1)}\|_A}{\|e^{(0)}\|_A} \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{\ell+1}, \quad \kappa = \frac{\lambda_{\max}(K)}{\lambda_{\min}(K)},$$

but they also note that this bound may be pessimistic.

The paper also mentions the relation between the CG algorithm and the Lanczos algorithm and how eigenvalue estimates can be obtained from a tridiagonal matrix derived from the CG recurrence coefficients. A hybrid algorithm is suggested, where one initially uses CG and obtains approximations to the extreme eigenvalues of the iteration matrix, at which point one could switch to the Chebyshev semi-iterative method with these eigenvalue estimates as parameters.

The paper also discusses various options for the preconditioner M (still referred to here as a *matrix splitting*, since A is written in the form $A = M - N$). For example, if A has the form

$$A = \begin{pmatrix} M_1 & F \\ F^T & M_2 \end{pmatrix},$$

where linear systems with coefficient matrices M_1 and M_2 are easy to solve, then the authors suggest taking

$$M = \begin{pmatrix} M_1 & 0 \\ 0 & M_2 \end{pmatrix}.$$

They note that for certain elliptic boundary value problems the iteration matrix K then has only a few distinct eigenvalues, even though $\lambda_{\max}(K)/\lambda_{\min}(K)$ is not especially small. They also mention the possibility of fast direct solvers for separable operators as preconditioners for nonseparable operators.

They further discuss the use of CG in combination with SSOR; that is, taking

$$M = (D + \omega L)D^{-1}(D + \omega L^T),$$

where $A = D + L + L^T$ and D is diagonal or block diagonal. Finally, they discuss the incomplete Cholesky decomposition of Meijerink and van der Vorst [13] as a preconditioner for CG.

A number of numerical examples are presented where the CG and Chebyshev algorithms are used to solve elliptic difference equations.

A generalized conjugate gradient method for non-symmetric systems of linear equations, by Concus and Golub [4]

In this paper, Concus and Golub deal with the case of nonsymmetric coefficient matrices. They make the ingenius observation that if one takes the symmetric part of the matrix $(A + A^T)/2$ as the preconditioner M for a nonsymmetric problem, then one can develop an algorithm very much like the symmetric CG method. This algorithm was later expressed more directly in terms of the Lanczos process by Widlund [18], and it is sometimes referred to as the CGW method.

To derive the algorithm, we write the iterates x_j as they are written in the conjugate gradient algorithm (although Concus and Golub derived the algorithm in a slightly different form). Starting with an initial guess x_0, an initial residual $r_0 = b - Ax_0$, a preconditioned residual $z_0 = M^{-1}r_0$, and an initial search direction $p_0 = z_0$, new approximations x_{j+1} can be written in the form

$$x_{j+1} = x_j + \alpha_j p_j,$$

where α_j is chosen to force M-orthogonality among the preconditioned residuals z_j. Since

$$z_{j+1} = z_j - \alpha_j M^{-1} A p_j,$$

it follows that

$$\alpha_j = \frac{\langle r_j, z_j \rangle}{\langle A p_j, z_j \rangle}.$$

The next search direction p_{j+1} is written in the form

$$p_{j+1} = z_{j+1} + \beta_j p_j,$$

where β_j is chosen to make p_{j+1} M-orthogonal to $M^{-1}Ap_j$:

$$\beta_j = -\frac{\langle z_{j+1}, A p_j \rangle}{\langle p_j, A p_j \rangle} = \frac{\langle z_{j+1}, r_{j+1} \rangle}{\langle z_j, r_j \rangle}.$$

Assuming that the symmetric part of A is positive definite, this algorithm is guaranteed to find the exact solution in at most n steps (assuming exact arithmetic), since $z_j^T M z_k = 0$ for $k \neq j$ implies that z_n (or an earlier z_k) must be zero. Moreover, if A has only $p < n$ distinct eigenvalues, then it finds the exact solution in p steps.

Concus and Golub consider expressions for the M-norm of the error, $\langle e_k, M e_k \rangle^{1/2}$. They show that, as for the symmetric case, the error at step k can be written in the form $e_k = [I - K p_{k-1}(K)] e_0$, where K is the preconditioned matrix $M^{-1}A$, and p_{k-1} is a certain $(k-1)$st degree polynomial. Unlike the symmetric case, however, p_{k-1} is not the polynomial that minimizes the M-norm of the error, although it can be shown to come fairly close to this. Concus and Golub derive a bound on the error in terms of that in the optimal second-order Richardson iterate.

Inner and outer iterations for the Chebyshev algorithm, by Giladi, Golub, and Keller [7]

In this paper, Giladi, Golub, and Keller return to the Chebyshev algorithm and ask the question: Suppose the preconditioning system $Mz = r$ is solved inexactly, perhaps using an iterative method inside the outer Chebyshev iteration. How accurately should it be solved in order to minimize the total work of the algorithm? They conclude that among all sequences of slowly varying tolerances for the inner iteration, a constant one is best.

This paper is among a group of papers dealing with inner and outer iterations. An earlier significant paper is by Golub and Overton [9]. A more recent outgrowth has been the study of the question of tolerances for inner iterations in the CG algorithm, where it has been shown, surprisingly, that while the initial tolerance should be at the level of accuracy desired in the final approximate solution, the tolerance at later iterations can be relaxed [2, 3, 16, 17].

The Giladi, Golub, and Keller paper stands out in that it precisely defines the cost function that they wish to minimize and then establishes a strong result about how that cost function can be minimized. They start by presenting an error bound for the inexact Chebyshev iteration. They then consider a decreasing sequence of tolerance values $\delta_k \to \eta$ and show that the asymptotic convergence rate for the algorithm with these inner tolerances is the same as that for the scheme with fixed tolerance $\delta_k = \eta$. They then use this result to determine the optimal sequence of tolerance values.

Let $\delta = \{\delta_j\}_{j=0}^{\infty}$ be a decreasing sequence of tolerance values with $0 < \eta < \delta_j < 1$. The number of inner iterations at outer iteration j is $\lceil \log \delta_j / \log r \rceil$, where r is the convergence factor of the method used for the inner iteration. If $N(\epsilon, \delta)$ denotes the number of outer iterations needed to reduce the error to ϵ using the strategy defined by δ, then the goal is to minimize the total number of inner iterations,

$$\sum_{j=0}^{N(\epsilon,\delta)-1} (-\log \delta_j).$$

Approximating this sum by an integral, and defining the set S of slowly varying sequences by

$$S = \{\delta : \quad \delta_k = \delta(\beta k), \quad \delta(x) \geq \eta > 0\},$$

where $\delta \in C^2$ and $\beta > 0$ is a small positive parameter, the authors attempt to find the sequence $\delta^* \in S$ that minimizes

$$C(\epsilon, \delta) \equiv - \int_0^{N(\epsilon,\delta)} \log \delta(\beta t)\, dt.$$

Since the number $N(\epsilon, \delta)$ cannot be determined precisely, they replace it by an upper bound $N_B(\epsilon, \delta)$ and instead minimize the associated cost $C_B(\epsilon, \delta)$. They show that if δ is any sequence in S, then there is a constant tolerance $\hat{\delta}$ for which $C_B(\epsilon, \hat{\delta}) \leq C_B(\epsilon, \delta)$. The constant $\hat{\delta}$ can be determined adaptively while solving the linear system.

The authors generalize their analysis to other iterative schemes, and finally they present numerical results demonstrating that a constant tolerance results in a smaller total number of inner iterations.

Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems, by Bai, Golub, and Ng [1]

This is a more recent paper on nonsymmetric problems. Here the authors again consider a splitting involving the Hermitian and skew-Hermitian part of the matrix, and, under the assumption that the Hermitian part is positive definite, they are able to prove a bound on the contraction factor of the method that depends only on the spectrum of the Hermitian part. Much recent work has focused on the effect of eigenvectors as well as eigenvalues on the convergence of nonsymmetric iterative methods such as the GMRES algorithm, so it is quite exciting to have a method whose convergence rate depends on the eigenvalues of the Hermitian part and not on the eigenvalues of A^*A (as with solving the normal equations) or on the possibly ill-conditioned eigenvectors of A (as with the GMRES algorithm).

The basic HSS (Hermitian/skew-Hermitian splitting) iteration is very simple: Let $H = (A + A^*)/2$ and $S = (A - A^*)/2$. Given an initial guess x_0, for $k = 0, 1, \ldots$, compute

$$\begin{aligned}(\alpha I + H)x_{k+1/2} &= (\alpha I - S)x_k + b\\ (\alpha I + S)x_{k+1} &= (\alpha I - H)x_{k+1/2} + b,\end{aligned}$$

where α is a given positive constant. Looking at the equation for the error $e_k \equiv A^{-1}b - x_k$, it can then be seen that

$$e_{k+1} = (\alpha I + S)^{-1}(\alpha I - H)(\alpha I + H)^{-1}(\alpha I - S)e_k.$$

The method converges provided the spectral radius of the iteration matrix, $M(\alpha) \equiv (\alpha I + S)^{-1}(\alpha I - H)(\alpha I + H)^{-1}(\alpha I - S)$, is less than 1, and the amount by which the error is reduced at each iteration is bounded by the norm of the iteration matrix. The authors first note that the spectral radius $\rho(M(\alpha))$ is the same as the spectral radius of $(\alpha I - H)(\alpha I + H)^{-1}(\alpha I - S)(\alpha I + S)^{-1}$, and since the spectral radius is less than or equal to the 2-norm of the matrix:

$$\begin{aligned}\rho(M(\alpha)) &\le \|(\alpha I - H)(\alpha I + H)^{-1}(\alpha I - S)(\alpha I + S)^{-1}\|_2\\ &\le \|(\alpha I - H)(\alpha I + H)^{-1}\|_2 \cdot \|(\alpha I - S)(\alpha I + S)^{-1}\|_2.\end{aligned}$$

Next the authors note that the 2-norm of the second matrix is 1, since $(\alpha I - S)$ $(\alpha I + S)^{-1}$ is a unitary matrix. Since the first matrix is Hermitian, its 2-norm is the largest absolute value of an eigenvalue:

$$\max_{\lambda_i \in \lambda(H)} \left| \frac{\alpha - \lambda_i}{\alpha + \lambda_i} \right|.$$

Since this quantity is always less than 1 when H is positive definite and $\alpha > 0$, the method is unconditionally convergent.

The contraction factor, or the amount by which the error norm is reduced at each iteration, depends on the norm being used. While it cannot be guaranteed that the 2-norm of the error decreases at each iteration, the authors define a new norm: $|||v||| \equiv \|(\alpha I + S)v\|_2$, for all $v \in \mathbf{C}^n$. Then since

$$(\alpha I + S)e_{k+1} = [(\alpha I - H)(\alpha I + H)^{-1}](\alpha I + S)e_k,$$

it follows that

$$|||e_{k+1}||| \le \|(\alpha I - H)(\alpha I + H)^{-1}\|_2 \cdot |||e_k|||.$$

Hence in this norm, the contraction factor is bounded in the same way as the spectral radius of the iteration matrix.

Next it is shown that the optimal α can be expressed in terms of the largest and smallest eigenvalues of H:

$$\alpha^* = \sqrt{\lambda_{\min}(H)\lambda_{\max}(H)},$$

and for this α, the spectral radius and the contraction number in the new norm are

$$\frac{\sqrt{\kappa(H)}-1}{\sqrt{\kappa(H)}+1}.$$

While the theoretical analysis of the HSS iteration is impressive, it does require solution of linear systems with coefficient matrices $\alpha I + H$ and $\alpha I + S$. This can be costly. The remainder of the paper is devoted to a discussion of the IHSS (inexact Hermitian/skew-Hermitian splitting) iteration. Here the linear systems are solved inexactly using the conjugate gradient method for the Hermitian positive definite coefficient matrix $\alpha I + H$ and using any of several Krylov space methods for the system with coefficient matrix $\alpha I + S$. Analysis of this inexact iteration is followed by some practical numerical examples illustrating the efficiency of the method.

Summary

This group of five papers, while far from a complete list of contributions, is indicative of the breadth and depth of Golub's work in the area of iterative methods. From early work on SOR and Chebyshev iteration to more recent contributions on the conjugate gradient algorithm, preconditioning, error estimation and its relation to Gauss quadrature [8, 11], inner and outer iterations, and methods for solving nonsymmetric linear systems, Golub has played a leading role in the development and advancement of iterative solution methods for large linear systems.

REFERENCES

1*. Zhong-Zhi Bai, Gene H. Golub, and Michael K. Ng. Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems. *SIAM J. Matrix Analysis and Applications*, **24**(3), 603–626 (2003).

2. A. Bouras and V. Frayssé, *A relaxation strategy for inexact matrix-vector products for Krylov methods.* Tech. Rep. 15, CERFACS, Toulouse, France (2000).

3. A. Bouras and V. Frayssé. Inexact matrix-vector products in Krylov methods for solving linear systems: A relaxation strategy. *SIAM J. Matrix Anal. Appl.*, **26**, 660–678 (2005).

4*. Paul Concus and Gene H. Golub, *A generalized conjugate gradient method for non-symmetric systems of linear equations*, presented at the Second International Symposium on Computing Methods in Applied Sciences and Engineering, Versailles, France, Dec. pp. 15–19 (1975).

5*. Paul Concus, Gene H. Golub, and Dianne O'Leary. A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations. In *Proceedings of Sparse Matrix Conference, Sparse Matrix Computation*, Academic Press, Inc., pp. 309–332 (1976).

6. S. P. Frankel, Convergence rates of iterative treatments of partial differential equations. *Math. Tables Aids Comput.*, **4**, 65–75 (1950).

7. E. Giladi, G. Golub, and J. B. Keller, Inner and outer iterations for the Chebyshev algorithm. *SIAM J. Numer. Anal.*, **35**(1), 300–319 (1998).

8. G. Golub and G. Meurant, Matrices, moments, and quadrature II; how to compute the norm of the error in iterative methods. *BIT*, **37**(3), 687–705 (1997).

9. G. Golub and M. Overton, The convergence of inexact Chebyshev and Richardson iterative methods for solving linear systems. *Numer. Math.*, **53**, 571–593 (1988).

10*. G. H. Golub and R. S. Varga, Chebyshev semi-iterative methods, successive over-relaxation iterative methods, and second-order Richardson iterative methods, Parts I and II. *Numer. Math.*, **3**, 147–168 (1961).

11. G. Golub and Z. Strakoš, Estimates in quadratic formulas. *Numerical Algorithms*, **8**(2–4), 241–268 (1994).

12. M. R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Standards*, **49**, 409–435 (1952).

13. J. A. Meijerink and H. A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, **31**, 148–162 (1977).

14. J. K. Reid, On the method of conjugate gradients for the solution of large sparse linear systems. In *Large Sparse Sets of Linear Equations*, J. K. Reid (ed.), Academic Press, New York pp. 231–254 (1971).
15. J. D. Riley, Iteration procedures for the Dirichlet difference problem, *Math. Tables Aids Comput.*, **8**, 125–131 (1954).
16. V. Simoncini and D. B. Szyld, Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, **28**, 454–477 (2003).
17. G. L. G. Sleijpen and J. van den Eshof, Inexact Krylov subspace methods for linear systems. *SIAM J. Matrix Anal. Appl.*, **26**, 125–153 (2004).
18. O. Widlund, A Lanczos method for a class of non-symmetric systems of linear equations. *SIAM Jour. Numer. Anal.*, **15**, 801–812 (1978).
19. D. M. Young, Iterative methods for solving partial difference equations of elliptic type. *Trans. Amer. Math. Soc.*, **76**, 92–111 (1954).

An asterisk denotes a paper reprinted in this volume.

6

CHEBYSHEV SEMI-ITERATIVE METHODS, SUCCESSIVE OVER-RELAXATION ITERATIVE METHODS, AND SECOND-ORDER RICHARDSON ITERATIVE METHODS, PARTS I AND II (WITH R. S. VARGA)

NUMER. MATH. (3), 147–168 (1961).

I was invited, in 1960 I believe, by Abe Taub at the University of Illinois, to a colloquium lecture there, and before the lecture, I was asked by Professor Taub what my current research was. I mentioned two areas, one being the use of Chebyshev polynomials in the theory of iterative methods in linear algebra. Taub mentioned then that he had a student (who turned out to be Gene Golub) working on similar ideas. What I learned later was that Taub told Gene that "if Varga publishes first, you will have to write a new thesis". During that visit, I later met Gene, who was visibly shaken about all of this, and I suggested that we discuss this further over coffee. There was indeed overlap in our results, but Gene did things that I hadn't done, and conversely. We then agreed to write a paper, in two parts, which appeared in *Numerical Mathematics* in 1961. This paper was surely better than either of us could have done alone, and it was quite a successful research paper, which was highly referenced. Adding to all of this is that Gene has often said that "Varga saved my life," meaning that he didn't have to write a second thesis! I have often been asked why Gene and I didn't write more papers together. Sincerely, it would have been nice, but I wandered, as time went by, more into approximation theory and complex function theory, where my PhD thesis arose, and Gene had chosen other interesting areas in which to work. He had been a stellar figure in the area of numerical analysis, and this field owes him much for his deep and useful research results.

Dick Varga
Kent, Ohio, USA

Numerische Mathematik 3, 147—156 (1961)

Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods

Part I

By

GENE H. GOLUB* and RICHARD S. VARGA

§ 1. Introduction

One of the major areas of interest in numerical analysis is concerned with the numerically efficient solution of the matrix equation

$$(1.1)\qquad A\vec{x} = \vec{k},$$

where A is a given $N\times N$ real symmetric and positive definite matrix, and $\vec{k}$ is a given column vector. The matrix equation (1.1) can be readily reduced to the analogous matrix equation

$$(1.2)\qquad \vec{x} = B\vec{x} + \vec{g},$$

where B is an $N\times N$ real symmetric matrix which is *convergent*, i.e., if the eigenvalues of the matrix B are μ_j, $1\leq j\leq N$, then the *spectral radius* [*9*] $\varrho(B)$ of B satisfies

$$(1.3)\qquad \varrho(B) \equiv \max_j |\mu_j| < 1.$$

At this point, one can consider the different convergent systematic iterative methods in the title of this paper, and basically, the literature on the analysis of these methods can be conveniently separated on the following classification of the matrix B. With B symmetric, we say [*20*] that B is *cyclic* (of index 2) if there exists an $N\times N$ permutation matrix Λ such that

$$(1.4)\qquad \Lambda B \Lambda^T = \begin{pmatrix} 0 & F \\ F^T & 0 \end{pmatrix} = \tilde{B},$$

where the non-vacuous diagonal blocks of $\tilde{B}$ are square, with zero entries. In the more familiar notation of YOUNG [*24, 26*], B satisfies property A, and $\tilde{B}$ is consistently ordered with the σ_1 ordering. If no such permutation matrix Λ exists, we say that B is *primitive***.

* This paper includes work from the doctoral dissertation [*7*] of the first author, who wishes to thank Professor A. H. TAUB of the University of Illinois for guidance and encouragement in the preparation of that dissertation.

** Usually, the terms primitive and cyclic are reserved (see [*23*]) for irreducible matrices with non-negative entries. In the case that the matrix B of (1.2) is symmetric and irreducible, with non-negative entries, these definitions agree with the classical terminology.

If B is primitive, then the only systematic iterative methods of the title which have been analysed* and used in large scale digital machine codes are the Chebyshev semi-iterative method [*5', 11, 16, 18, 25*], and the second order Richardson iterative method [*6, 13*]. Actually, one can also define the successive overrelaxation iterative method [*6, 26*] for an expanded matrix equation of the form (1.2), and in §2, our first result is that all three methods, when optimized with respect to acceleration parameters, are from a certain point of view remarkably similar**. In §3, we shall compare these three systematic iterative methods, using the spectral norms of the respective matrix operators as a basis for comparison, and we shall show that the matrix operator for the Chebyshev semi-iterative method possesses the smallest spectral norm. Since the practical application of the Chebyshev semi-iterative method in the primitive case requires effectively no additional arithmetic operations or vector storage over the other iterative methods, it would appear that of these three systematic iterative methods, there is no practical or theoretical reason for not always using the Chebyshev semi-iterative method for the primitive case.

If B is cyclic, then several results already exist in the literature [*13, 18, 25, 27*] comparing the three basic systematic iterative methods of the title. In §4, we shall define a new systematic iterative method, called the *cyclic Chebyshev semi-iterative method* for cyclic matrices B, which again requires effectively no more additional arithmetic operations or vector storage over the other iterative methods. This new systematic iterative method, which has combined the observations of several others, will be shown in §5, using spectral norms of matrix operators as a basic for comparison, to have the smallest spectral norm. Again, of the three systematic iterative methods of the title, it would appear that the modified Chebyshev semi-iterative method is the best choice in the cyclic case.

In §6, we shall show how these results can be used in conjunction with various block methods [*1, 8, 12, 21*] to numerically solve elliptic difference equations, and finally in §7, we shall give some of the results of recent numerical experiments on the comparison between the systematic iterative methods of the title.

§2. Primitive Matrices

We assume in this section that the matrix B of (1.2) is primitive. If $\vec{\alpha}^{(0)}$ is a given vector guess of the unique solution $\vec{x}$ of (1.2), then we iteratively form the vectors $\vec{\alpha}^{(i+1)}$, defined by

$$\vec{\alpha}^{(i+1)} = B\vec{\alpha}^{(i)} + \vec{g}, \qquad i \geqq 0. \tag{2.1}$$

Since $\varrho(B) < 1$, the sequence vectors $\vec{\alpha}^{(i)}$ converges [*9*] to the solution vector $\vec{x}$. If $\vec{\varepsilon}^{(i)} \equiv \vec{x} - \vec{\alpha}^{(i)}$, $i \geqq 0$, is the error vector associated with the iterate $\vec{\alpha}^{(i)}$, then

$$\vec{\varepsilon}^{(i+1)} = B\vec{\varepsilon}^{(i)}, \qquad i \geqq 0, \tag{2.2}$$

and thus, by induction

$$\vec{\varepsilon}^{(i)} = B^i\vec{\varepsilon}^{(0)}, \qquad i \geqq 0. \tag{2.2'}$$

* A notable exception to this is KAHAN's theoretical extension [*10*] of the successive overrelaxation iterative method to the case where B is primitive, and has non-negative entries.

** It has been generally assumed that the successive overrelaxation iterative method could not be applied in as general cases as could the Chebyshev semiiterative method. See [*27*, p. 291].

We now considcr forming lincar combinations* of the vectors $\vec{\alpha}^{(i)}$ in order to accelerate the convergence of (2.1). Let

$$\vec{\beta}^{(i)} = \sum_{k-0}^{i} a_{i,k}\,\vec{\alpha}^{(k)}, \qquad i \geqq 0. \tag{2.3}$$

As in [18], we impose the natural condition that $\sum_{k=0}^{i} a_{i,k} = 1$. Thus, if $\vec{\overset{*}{\varepsilon}}^{(i)} = \vec{x} - \vec{\beta}^{(i)}$, $i \geqq 0$, then

$$\vec{\overset{*}{\varepsilon}}^{(i)} = \left(\sum_{k=0}^{i} a_{i,k} B^k\right)\vec{\varepsilon}^{(0)}. \tag{2.4}$$

If $p_i(t) = \sum_{k=0}^{i} a_{i,k} t^k$, then (2.4) becomes formally

$$\vec{\overset{*}{\varepsilon}}^{(i)} = p_i(B)\,\vec{\varepsilon}^{(0)}, \tag{2.4'}$$

where $p_i(1) = 1$. Let $\{\vec{y}_i\}_{i=1}^{N}$ be an orthonormal set of eigenvectors of B, where $B\vec{y}_i = \mu_i \vec{y}_i$, $1 \leqq i \leqq N$. If $\vec{\varepsilon}^{(0)} = \sum_{k=1}^{N} c_k \vec{y}_k$, then it follows that

$$\vec{\overset{*}{\varepsilon}}^{(m)} = \sum_{k=1}^{N} c_k\, p_m(\mu_k)\,\vec{y}_k. \tag{2.5}$$

If all the eigenvalues μ_i of B were known a priori, then we could determine a polynomial $p_N(x)$ such that $p_N(\mu_k) = 0$ for $1 \leqq k \leqq N$. Since this is seldom the case, let S_m be the set of all polynomials $p_m(x)$ of degree m, normalized so that $p_m(1) = 1$. Since B is symmetric and convergent, all its eigenvalues μ_i satisfy $-1 < -\varrho(B) \leqq \mu_i \leqq \varrho(B) < 1$, and we seek** the polynomial $\tilde{p}_m(x)$ for which

$$\min_{p_m \in S_m}\left\{\max_{-\varrho \leqq x \leqq \varrho} |p_m(x)|\right\} = \max_{-\varrho \leqq x \leqq \varrho} |\tilde{p}_m(x)|, \tag{2.6}$$

wherc $\varrho = \varrho(B)$. As is wcll known [4], the *unique* solution of this problem is givcn explicitly by

$$\tilde{p}_m(x) = \frac{C_m(x/\varrho(B))}{C_m(1/\varrho(B))}, \quad m \geqq 0, \tag{2.7}$$

where

$$C_m(x) = \begin{cases} \cos(m \cos^{-1} x), & x \leqq 1, \quad m \geqq 0, \\ \cosh(m \cosh^{-1} x), & x \geqq 1, \quad m \geqq 0, \end{cases} \tag{2.8}$$

is the Chebyshev polynomial of degree m. Since the Chebyshev polynomials satisfy the well-known recurrence relation

$$C_{m+1}(x) = 2x\,C_m(x) - C_{m-1}(x), \qquad m \geqq 1, \tag{2.8'}$$

where $C_0(x) = 1$, $C_1(x) = x$, we can use (2.8') to deduce a recurrence relation for the polynomials $\tilde{p}_m(x)$ which, when inserted into (2.4'), leads to the following

* This is called "linear acceleration" by FORSYTHE [5]. Professor A. H. TAUB has kindly pointed out to us that these results were known much earlier to VON NEUMANN. See [2].

** If B is known to be non-negative, irreducible, and primitive, then the smallest interval $a \leqq x \leqq b$ which contains the eigenvalues of B is such [23] that $|a| < b = \varrho(B)$. While this change in the problem of (2.6) would result in improved convergence rates, it is in general difficult to obtain the lower bound in practical problems.

relationship* for the vectors $\vec{\beta}^{(i)}$:

$$(2.9)\qquad \vec{\beta}^{(i+1)} = \omega_{i+1}\{B\vec{\beta}^{(i)} + \vec{g} - \vec{\beta}^{(i-1)}\} + \vec{\beta}^{(i-1)}, \qquad i \geqq 1,$$

where

$$(2.10)\qquad \omega_{i+1} = \frac{2C_i(1/\varrho(B))}{\varrho(B)\, C_{i+1}(1/\varrho(B))}, \qquad i \geqq 1, \quad \omega_1 = 1.$$

With $\omega_1 = 1$, (2.9) reduces to $\vec{\beta}^{(1)} = B\vec{\beta}^{(0)} + \vec{g} = B\vec{\alpha}^{(0)} + \vec{g}$, since $\vec{\beta}^{(0)} = \vec{\alpha}^{(0)}$. Using (2.8'), we can also express the parameters ω_{i+1} as

$$(2.11)\qquad \omega_{i+1} = \frac{1}{1 - \left(\frac{\varrho^2\,\omega_i}{4}\right)}, \quad i \geqq 2, \quad \omega_1 = 1, \quad \omega_2 = \frac{2}{2-\varrho^2},$$

which is more convenient for actual computations. From (2.9), we notice that the determination of vector iterates $\vec{\beta}^{(i)}$ does not require the computation or storage of the auxiliary vector iterates $\vec{\alpha}^{(i)}$ of (2.1).

Having described the Chebyshev semi-iterative method, we now consider the successive overrelaxation iterative method of YOUNG and FRANKEL [*6*, *26*], applied to the matrix equation (1.2) where B is primitive. Without making further assumptions on the matrix B, such as B having entries only of one sign [*10*], successive overrelaxation applied directly to (2.1) has not as yet been completely rigorously analysed. We now show that by considering matrix equations with twice as many components, successive overrelaxation can be rigorously applied to a system of equations derived from (1.2). From (1.2), we consider the coupled pair of matrix equations

$$(2.12)\qquad \begin{cases} \vec{x} = B\vec{y} + \vec{g} \\ \vec{y} = B\vec{x} + \vec{g}, \end{cases}$$

which in matrix notation becomes

$$(2.12')\qquad \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} = \begin{pmatrix} 0 & B \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} + \begin{pmatrix} \vec{g} \\ \vec{g} \end{pmatrix}.$$

If

$$(2.13)\qquad J = \begin{pmatrix} 0 & B \\ B & 0 \end{pmatrix},$$

then the matrix J is also convergent, and $\varrho(J) = \varrho(B)$. Since $\varrho(B) < 1$, there is a unique solution of (2.12'), and evidently $\vec{x} = \vec{y}$.

The successive overrelaxation iterative method applied to (2.12') is defined by

$$(2.14)\qquad \begin{cases} \vec{x}^{(m+1)} = \vec{x}^{(m)} + \omega\{B\vec{y}^{(m)} + \vec{g} - \vec{x}^{(m)}\}, \\ \vec{y}^{(m+1)} = \vec{y}^{(m)} + \omega\{B\vec{x}^{(m+1)} + \vec{g} - \vec{y}^{(m)}\}, \qquad m \geqq 0, \end{cases}$$

where ω is the *relaxation factor*, and $\vec{x}^{(0)}$, $\vec{y}^{(0)}$ are initial guess vectors. Since the matrix J of (2.13) is evidently real, symmetric, cyclic, and consistently ordered in the sense of YOUNG [*26*], then we can apply the general theory of successive

[*] This is a somewhat simpler computational form of the recurrence relationship than is found, say, in [*5'*, *16*, *18*].

overrelaxation due to YOUNG [*26*] to (2.12′), and the optimum value of ω is given by

$$\omega_b = \frac{2}{1+\sqrt{1-\varrho^2(J)}} = \frac{2}{1+\sqrt{1-\varrho^2(B)}}\,. \tag{2.15}$$

To show the similarity of (2.14) to (2.9), we now define a sequence of vectors $\vec{\zeta}^{(l)}$, where

$$\begin{cases} \vec{\zeta}^{(2l)} = \vec{x}^{(l)}, \\ \vec{\zeta}^{(2l+1)} = \vec{y}^{(l)}, \quad l \geqq 0. \end{cases} \tag{2.16}$$

In terms of the vectors $\vec{\zeta}^{(l)}$, we can write (2.14) in the compact form

$$\vec{\zeta}^{(m+1)} = \omega\{B\vec{\zeta}^{(m)} + \vec{g} - \vec{\zeta}^{(m-1)}\} + \vec{\zeta}^{(m-1)}, \qquad m \geqq 1, \tag{2.17}$$

where $\vec{\zeta}^{(0)}$, and $\vec{\zeta}^{(1)}$ are given vectors guesses. Thus, we conclude that the successive overrelaxation iterative method applied to (2.12′) is in the same form as the Chebyshev semi-iterative method of (2.9), except that in (2.9) the relaxation factors vary with iteration, whereas in (2.17) the relaxation factor ω is fixed. Even more interesting is the fact that the numbers ω_i of (2.11) are strictly decreasing for $i \geqq 2$ $(0<\varrho(B)<1)$, and, as can be readily exhibited,

$$\lim_{m\to\infty} \omega_m = \frac{2}{1+\sqrt{1-\varrho^2(B)}} = \omega_b, \tag{2.18}$$

where ω_b is defined in (2.15).

We now consider the second order Richardson iterative method [*6*, *13*], which is defined from (1.2) by

$$\vec{\eta}^{(m+1)} = \vec{\eta}^{(m)} + \alpha\{B\vec{\eta}^{(m)} + \vec{g} - \vec{\eta}^{(m)}\} + \beta\{\vec{\eta}^{(m)} - \vec{\eta}^{(m-1)}\}, \qquad m \geqq 1, \tag{2.19}$$

where $\vec{\eta}^{(0)}$, $\vec{\eta}^{(1)}$ are given initial vector guesses to the unique solution $\vec{x}$ of (1.2), and α and β are fixed acceleration parameters. If $\beta' = \beta - \alpha$, this is equivalent to

$$\vec{\eta}^{(m+1)} = \vec{\eta}^{(m)} + \alpha\{B\vec{\eta}^{(m)} + \vec{g} - \vec{\eta}^{(m-1)}\} + \beta'\{\vec{\eta}^{(m)} - \vec{\eta}^{(m-1)}\}, \qquad m \geqq 1. \tag{2.19′}$$

One can extend the analysis of FRANKEL and RILEY [*6*, *26*], and the best acceleration parameters, those giving the fastest asymptotic convergence, are given* by

$$\alpha = \frac{2}{1+\sqrt{1-\varrho^2(B)}}, \qquad \beta' = -1. \tag{2.20}$$

With this choice of parameters α and β', we see that the second order Richardson iterative method of (2.19′) is identical with the successive overrelaxation iterative method of (2.17), with $\omega = \omega_b$. Of course, RILEY [*13*] pointed out this correspondence in the case of the numerical solution of the Dirichlet problem.

Having compared three systematic iterative methods for solving (1.2) when B is primitive, we see that each method, when optimized, bears a strong resemblance to the other methods. In the next section, we shall compare these three iterative methods using the spectral norms of the corresponding matrix operators as a basis for comparison.

* See also [*22*, p. 485].

§ 3. Primitive Matrices: Comparison of Methods

As in the previous section, we assume that B is real, primitive, and symmetric, with $\{\vec{y}_i\}_{i=1}^{N}$ as an orthonormal set of eigenvectors of B. If, as in § 2, $\vec{\varepsilon}^{(0)}=\sum_{k=1}^{N} c_k \vec{y}_k$, then Euclidean norm of $\vec{\varepsilon}^{(0)}$ is defined by

$$\|\vec{\varepsilon}^{(0)}\| = \left(\sum_{k=1}^{N} |c_k|^2\right)^{\frac{1}{2}}. \tag{3.1}$$

To review some facts [9] concerning norms of matrices and vectors, let M be any real $N\times N$ matrix. For any real vector $\vec{x}$ with N components, then from $\vec{y}=M\vec{x}$, we have

$$\|\vec{y}\| = \|M\vec{x}\| \leqq \tau(M)\,\|\vec{x}\|, \tag{3.2}$$

where

$$\tau(M) \equiv [\varrho(M^T M)]^{\frac{1}{2}}. \tag{3.3}$$

The quantity $\tau(M)$ is called the *spectral norm* of the matrix M. Characterized in a different manner, we have that

$$\tau(M) = \max_{\|\vec{x}\|\neq 0} \frac{\|M\vec{x}\|}{\|\vec{x}\|}. \tag{3.4}$$

It is clear that if M is symmetric, then the spectral norm $\tau(M)$ of M coincides with the spectral radius $\varrho(M)$ of M.

For the Chebyshev semi-iterative method of (2.9), we have that $\vec{\overset{*}{\varepsilon}}^{(0)}=\vec{\varepsilon}^{(0)}$, and $\vec{\overset{*}{\varepsilon}}^{(m)}=\tilde{p}_m(B)\,\vec{\varepsilon}^{(0)}$. Thus,

$$\|\vec{\overset{*}{\varepsilon}}^{(m)}\| \leqq \tau\left(\tilde{p}_m(B)\right)\|\vec{\varepsilon}^{(0)}\|, \qquad m\geqq 0. \tag{3.5}$$

Since the matrix B is symmetric, so is the matrix $\tilde{p}_m(B)$, and we can express $\tau\left(\tilde{p}_m(B)\right)$ as

$$\tau\left(\tilde{p}_m(B)\right) = \varrho\left(\tilde{p}_m(B)\right) = \max_{1\leqq i\leqq N} |\tilde{p}_m(\mu_i)|. \tag{3.6}$$

From (2.7) and (2.8), it follows that

$$\tau\left(\tilde{p}_m(B)\right) = \frac{\max_{1\leqq i\leqq N} |C_m(\mu_i/\varrho(B))|}{C_m(1/\varrho(B))}, \qquad m\geqq 0, \tag{3.7}$$

and from (1.3) and (2.8),

$$\tau\left(\tilde{p}_m(B)\right) = \frac{1}{C_m(1/\varrho(B))}, \qquad m\geqq 0. \tag{3.7'}$$

To simplify the expression in (3.7′), we recall from (2.8) that $C_m\left(1/\varrho(B)\right)=\cosh(m\sigma)$, where $\cosh\sigma=\frac{1}{\varrho(B)}$, and $\varrho(B)<1$. Thus, $C_m\left(1/\varrho(B)\right)=e^{m\sigma}\left(\frac{1+e^{-2m\sigma}}{2}\right)$, and since $\sigma=\ln\left\{\frac{1}{\varrho(B)}+\sqrt{\frac{1}{\varrho^2(B)}-1}\right\}$, we have that $e^{-m\sigma}=\left\{\frac{\varrho(B)}{1+\sqrt{1-\varrho^2(B)}}\right\}^m$. From YOUNG's basic formula [26] we can relate $e^{-m\sigma}$ to the quantity ω_b in (2.15), and we have that $e^{-m\sigma}=(\omega_b-1)^{m/2}$. Combining, we now write (3.7′) as

$$\tau\left(\tilde{p}_m(B)\right) = (\omega_b-1)^{m/2}\cdot\left\{\frac{2}{1+(\omega_b-1)^m}\right\}, \qquad m\geqq 0. \tag{3.8}$$

Since $\frac{2x}{1+x^2}<1$ for $0\leqq x<1$, it follows that the right side of (3.8) is less than unity, and is strictly decreasing with increasing m. Thus, we conclude that the matrix operator $\tilde{p}_m(B)$ for the Chebyshev semi-iterative method is *norm reducing* for all $m\geqq 1$.

For the successive overrelaxation iterative method, or equivalently for the second order Richardson iterative methods with $\alpha=\omega$ and $\beta=-1$, we have the following recurrence relation for the error vectors of the iterates of (2.17):

$$(3.9) \qquad \vec{\varepsilon}^{(m+1)}=\omega\, B\, \vec{\varepsilon}^{(m)}+(1-\omega)\, \vec{\varepsilon}^{(m-1)}, \qquad m\geqq 1,$$

where $\vec{\varepsilon}^{(m)}=\vec{x}-\vec{\zeta}^{(m)}$, $m\geqq 0$, so that $\vec{\varepsilon}^{(0)}$ and $\vec{\varepsilon}^{(1)}$ are dependent on the given vectors $\vec{\zeta}^{(0)}$ and $\vec{\zeta}^{(1'}$. If $\alpha_0(B)=I$, and $\alpha_1(B)=\omega B$, we define now the polynomials $\alpha_m(B)$ from the recurrence relation

$$(3.10) \qquad \alpha_{m+1}(B)=\omega\, B\, \alpha_m(B)+(1-\omega)\, \alpha_{m-1}(B), \qquad m\geqq 1.$$

By induction, $\alpha_m(B)$ is, for $\omega\neq 0$, a polynomial of degree m in B, and it is easily verified that

$$(3.11) \qquad \vec{\varepsilon}^{(m)}=\alpha_{m-1}(B)\, \vec{\varepsilon}^{(1)}+(1-\omega)\, \alpha_{m-2}(B)\, \vec{\varepsilon}^{(0)}, \qquad m\geqq 2.$$

Upon replacing the matrix B by the variable x in (3.10), the linear difference equation of (3.10) can be solved, and $\alpha_m(x)$ can be explicitly represented by

$$(3.12) \qquad \alpha_m(x)=\left\{\begin{array}{ll} \dfrac{\varphi_1^{m+1}(x)-\varphi_2^{m+1}(x)}{\varphi_1(x)-\varphi_2(x)}, & \varphi_1(x)\neq\varphi_2(x) \\ (m+1)\,\varphi_1^m(x), & \varphi_1(x)=\varphi_2(x) \end{array}\right\}, \qquad m\geqq 0,$$

where $\varphi_1(x)$ and $\varphi_2(x)$ are the roots of the equation

$$(3.13) \qquad \varphi^2(x)-\omega\, x\, \varphi(x)+(\omega-1)=0.$$

If $\omega=\omega_b$ of (2.15), and $-\varrho(B)\leqq x\leqq+\varrho(B)$, then as shown in [*6*], all the roots of (3.13) are complex conjugates of the form $(\omega_b-1)^{\frac{1}{2}}\, e^{\pm i\vartheta}$, where $\cos\vartheta=x/\varrho(B)$. Consequently,

$$(3.14) \qquad \alpha_m(x)=(\omega_b-1)^{m/2}\cdot\left\{\begin{array}{ll} \dfrac{\sin(m+1)\,\vartheta}{\sin\vartheta}, & \vartheta\neq 0, \pi, \\ m+1, & \vartheta=0, \\ (-1)^m(m+1), & \vartheta=\pi, \end{array}\right\}, \qquad m\geqq 0.$$

It is clear from (3.11) that $\vec{\varepsilon}^{(m)}$ depends on the relationship between $\vec{\varepsilon}^{(0)}$ and $\vec{\varepsilon}^{(1)}$. For example, if $\vec{\varepsilon}^{(1)}=-\vec{\varepsilon}^{(0)}$ and $\omega=\omega_b$, then

$$(3.15) \qquad \vec{\varepsilon}^{(m)}=q_{m-1}(B)\, \vec{\varepsilon}^{(0,}, \qquad m\geqq 2,$$

where

$$(3.16) \qquad q_{m-1}(B)=-\left[\alpha_{m-1}(B)+(\omega_b-1)\, \alpha_{m-2}(B)\right], \qquad m\geqq 2.$$

From the symmetry of the matrix B, we have that

$$(3.17) \qquad \tau[q_{m-1}(B)]=\varrho[q_{m-1}(B)]=\max_{1\leq i\leq N}|q_{m-1}(\mu)|,$$

but from (3.14), we can directly express $\tau[q_{m-1}(B)]$ as

$$(3.17') \qquad \tau[q_{m-1}(B)]=(\omega_b-1)^{\frac{m-1}{2}}\{|m|+|m-1|\,(\omega_b-1)^{\frac{1}{2}}\}, \qquad m\geqq 0.$$

We note that this spectral norm can actually initially increase with m, if ω_b is sufficiently large*.

* An analogous observation was made by SHELDON [*15*] in the cyclic case. See also § 5.

We shall now show that the situation of (3.17′) can be considerably improved. Let

$$\vec{\zeta}^{(1)} = B\vec{\zeta}^{(0)} + \vec{g}, \tag{3.18}$$

so that

$$\vec{\varepsilon}^{(1)} = B\vec{\varepsilon}^{(0)}. \tag{3.19}$$

For $\vec{\zeta}^{(1)}$ selected in this way, then

$$\vec{\varepsilon}^{(m)} = r_m(B)\,\vec{\varepsilon}^{(0)}, \qquad m \geqq 2, \tag{3.20}$$

where $r_m(B)$ is a polynomial of degree m in B defined recursively, using (3.11) and (3.19), by

$$r_m(B) = B\,\alpha_{m-1}(B) + (1-\omega)\,\alpha_{m-2}(B), \qquad m \geqq 2, \tag{3.21}$$

with $r_0(B) = I$, and $r_1(B) = B$.

Again,

$$\tau[r_m(B)] = \varrho[r_m(B)] = \max_{1\leqq i\leqq N} |r_m(\mu_i)|, \qquad m \geqq 0, \tag{3.22}$$

and, for the case $\omega = \omega_b$, a short calculation [7, pp. 22—23] based on (3.14) shows that

$$\tau[r_m(B)] = |r_m\,\varrho(B)| = (\omega_b - 1)^{m/2}\left\{1 + m\sqrt{1-\varrho^2(B)}\right\}, \qquad m \geqq 0. \tag{3.23}$$

It is readily verified [7, pp. 23—24] that the right side of (3.23) is monotone decreasing for all $m \geqq 0$, showing that the matrix operator for the successive overrelaxation iterative method of (2.17) is also norm reducing for $m \geqq 1$ with $\vec{\zeta}^{(1)}$ chosen according to (3.18).

While the Chebyshev semi-iterative method of (2.9) requires but one vector guess $\vec{\beta}^{(0)} = \vec{\zeta}^{(0)}$, (2.9) shows that $\vec{\beta}^{(1)}$ also satisfies (3.18), so that we can directly compare the spectral norm (3.8) of the Chebyshev semi-iterative method with the spectral norm (3.23) of the primitive successive overrelaxation iterative with $\vec{\zeta}^{(1)}$ chosen according to (3.18) method. Now, since $r_0(x) = 1$, and $r_1(x) = x$, it follows easily from (3.21) and (3.10) that $r_m(1) = 1$ for all $m \geqq 0$. But from (2.7), the same is true of the polynomials $\tilde{p}_m(x)$. Moreover, since $\tau[r_m(B)] = |r_m(\varrho(B))|$, and similarly $\tau[\tilde{p}_m(B)] = |\tilde{p}_m(\varrho(B))|$, we can use, as in (2.6), the well known property of the Chebyshev polynomials that among all polynomials $g_m(x)$ of degree m with $g_m(1) = 1$, $\tilde{p}_m(x)$ is the *unique* polynomial whose maximum absolute value on the interval $-\varrho(B) \leqq x \leqq +\varrho(B)$ is minimal. This gives us

Lemma 1. In the primitive case where $\vec{\zeta}^{(0)}$ is arbitrary and $\vec{\zeta}^{(1)} = B\vec{\zeta}^{(0)} + \vec{g}$,

$$\tau(\tilde{p}_m(B)) < \tau(r_m(B)), \qquad m > 1. \tag{3.24}$$

We shall now consider the successive overrelaxation iterative method with another starting procedure*. Let

$$\begin{aligned} \vec{\zeta}^{(1)} &= B\vec{\zeta}^{(0)} + \vec{g} \\ \vec{\zeta}^{(2)} &= B\vec{\zeta}^{(1)} + \vec{g}, \end{aligned} \tag{3.25}$$

* Such a starting procedure is suggested for the primitive case from results in the cyclic case by SHELDON [12]. See also § 5.

and $\vec{\zeta}^{(m)}$ $(m \geqq 3)$ we generated by (2.17). Then

$$(3.26) \qquad \vec{\varepsilon}^{(1)} = B\,\vec{\varepsilon}^{(0)}, \qquad \vec{\varepsilon}^{(2)} = B^2\,\vec{\varepsilon}^{(0)},$$

and in general

$$(3.27) \qquad \vec{\varepsilon}^{(m)} = t_m(B)\,\vec{\varepsilon}^{(0)}, \qquad m \geqq 0,$$

where the matrix $t_m(B)$ is a polynomial of degree m in the matrix B, and is defined recursively by

$$(3.28) \qquad t_m(B) = B^2 \alpha_{m-2}(B) + (1-\omega)\, B\, \alpha_{m-3}(B), \qquad m \geqq 3,$$

where $t_j(B) = B^j$ for $0 \leqq j \leqq 2$. Again,

$$(3.29) \qquad \tau[t_m(B)] = \varrho[t_m(B)] = \max_{1 \leqq i \leqq N} |t_m(\mu_i)|, \qquad m \geqq 0,$$

and for $\omega = \omega_b$, a short calculation based on (3.14) shows that

$$(3.30) \quad \tau[t_m(B)] = |t_m(\varrho(B))| = (\omega_b - 1)^{\frac{m-1}{2}}\, \varrho(B)\left\{1 + (m-1)\sqrt{1-\varrho^2(B)}\right\}, \; m \geqq 1.$$

With $s_m(t) \equiv t^m$, corresponding to the basic iterative method of (2.1), it is not difficult to show that

$$(3.31) \qquad \tau\left(r_m(B)\right) < \tau\left(t_m(B)\right) < \tau\left(s_m(B)\right) = \varrho^m(B) \quad \text{for} \quad m > 1.$$

Consequently, we have

Theorem 1. In the primitive case where $\vec{\zeta}^{(0)}$ is arbitrary and $\vec{\zeta}^{(1)} = B\vec{\zeta}^{(0)} + \vec{g}$, then for $0 < \varrho(B) < 1$, and $m > 1$,

$$(3.32) \qquad \tau\left(\tilde{p}_m(B)\right) < \tau\left(r_m(B)\right) < \tau\left(t_m(B)\right) < \tau\left(s_m(B)\right) = \varrho^m(B).$$

Thus, the spectral norm of the matrix operator for $m > 1$ iterations of the Chebyshev semi-iterative method is less than the spectral norms of the matrix operators for m iterations of the two variants (3.18) and (3.25) of the successive overrelaxation iterative method, as well as the spectral norms for m iterations of the iterative method of (2.1).

References

[1] ARMS, R. J., L. D. GATES and B. ZONDEK: A method of block iteration. Journal Soc. Indust. Appl. Math. **4**, 220—229 (1956).

[2] BLAIR, A., N. METROPOLIS, J. v. NEUMANN, A. H. TAUB and M. TSINGOU: A study of a numerical solution of a two-dimensional hydrodynamical problem. Math. Tables and Other Aids to Computation **13**, 145—184 (1959).

[3] CUTHILL, ELIZABETH H., and RICHARD S. VARGA: A method of normalized block iteration. Journal Assoc. Computing Mach. **6**, 236—244 (1959).

[4] FLANDERS, DONALD A., and GEORGE SHORTLEY: Numerical determination of fundamental modes. Journal of Applied Physics **21**, 1326—1332 (1950).

[5] FORSYTHE, G. E.: Solving linear algebraic equations can be interesting. Bull. Amer. Math. Soc. **59**, 299—329 (1953).

[5'] FRANK, WERNER: Solution of linear systems by RICHARDSON's method. Journal Assoc. Computing Mach. **7**, 274—286 (1960).

[6] FRANKEL, STANLEY P.: Convergence rates of iterative treatments of partial differential equations. Math. Tables Aids Comput. **4**, 65—75 (1950).

[7] GOLUB, GENE H.: The use of Chebyshev matrix polynomials in the iterative solution of linear equations compared with the method of successive overrelaxation. Doctoral Thesis, University of Illinois, 1959.

[8] HELLER, J.: Simultaneous, successive and alternating direction iterative methods. Journal Soc. Indust. Appl. Math. **8**, 150—173 (1960).

[9] HOUSEHOLDER, A. S.: The approximate solution of matrix problems. Journal Assoc. Computing Mach. **5**, 205—243 (1958).

[10] KAHAN, W.: Gauss-Seidel methods of solving large systems of linear equations. Doctoral Thesis, University of Toronto, 1958.

[11] LANCZOS, CORNELIUS: Solution of systems of linear equations by minimized iterations. Journal Research Nat. Bureau of Standards **49**, 33—53 (1952).

[12] PARTER, SEYMOUR V.: On two-line iterative methods for the Laplace and biharmonic difference equations. Numerische Mathematik **1**, 240—252 (1959).

[13] RILEY, JAMES D.: Iteration procedures for the dirichlet Difference problem. Math. Tables Aids Comput. **8**, 125—131 (1954).

[14] ROMANOVSKY, V.: Recherches sur les chaînes de MARKOFF. Acta Math. **66**, 147—251 (1936).

[15] SHELDON, J. W.: On the spectral norms of several iterative processes. Journal Assoc. Computing Mach. **6**, 494—505 (1959).

[16] STIEFEL, EDUARD L.: Kernel polynomials in linear algebra and their numerical applications. National Bureau of Standards Applied Math. Series 49, U.S. Government Printing Office, Washington, D. C. 1958, p. 1—22.

[17] TODD, JOHN: The condition of a certain matrix. Proc. Cambridge Philos. Soc. **46**, 116—118 (1950).

[18] VARGA, RICHARD S.: A comparison of the successive overrelaxation method and semi-iterative methods using Chebyshev polynomials. Journal Soc. Indust. Appl. Math. **5**, 39—46 (1957).

[19] VARGA, RICHARD S.: Numerical solution of the two-group diffusion equation in $x-y$ geometry. IRE Trans. of the Professional Group on Nuclear Science, N.S. **4**, 52—62 (1957).

[20] VARGA, RICHARD S.: p-cyclic matrices: a generalization of the Young-Frankel successive overrelaxation scheme. Pacific Journal of Math. **9**, 617—628 (1959).

[21] VARGA, RICHARD S.: Factorization and normalized iterative methods. Boundary Problems in Differential Equations, the University of Wisconsin Press, Madison 1960, p. 121—142.

[22] WACHSPRESS, E. L., P. M. STONE and C. E. LEE: Mathematical techniques in two-space-dimension multigroup calculations. Proceedings of the Second United Nations International Conference on the Peaceful Uses of Atomic Energy, United Nations, Geneva, 1958, volume 16, p. 483—488.

[23] WIELANDT, HELMUT: Unzerlegbare, nicht negative Matrizen. Math. Z. **52**, 642—648 (1950).

[24] YOUNG, DAVID: Iterative methods for solving partial differences equations of elliptic type. Doctoral Thesis, Harvard University, 1950.

[25] YOUNG, DAVID: On RICHARDSON's method for solving linear systems with positive definite matrices. Journal Math. and Physics **32**, 243—255 (1953).

[26] YOUNG, DAVID: Iterative methods for solving partial difference equations of elliptic type. Trans. Amer. Math. Soc. **76**, 92—111 (1954).

[27] YOUNG, DAVID: On the solution of linear systems by iteration. Proceedings of the Sixth Symposium in Applied Math., p. 283—298. New York: McGraw-Hill 1956.

Space Technology Laboratories, Inc.
Los Angeles, California
and
Case Institute of Technology
Cleveland 6, Ohio
10900 Euclid Avenue

(Received June 17, 1960)

Numerische Mathematik 3, 157—168 (1961)

Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods

Part II

By

GENE H. GOLUB and RICHARD S. VARGA

§4. Cyclic Matrices: The Cyclic Chebyshev Semi-Iterative Method

We now suppose that the $N \times N$ matrix B is cyclic, and in the form of (1.4). As we have already pointed out, the matrix B in this form satisfies YOUNG's property A, and is consistently ordered. Because B is real and symmetric, YOUNG's theory [26] can be applied to the solution of the matrix equation of (1.2). With B in the form (1.4), we partition the vectors $\vec{x}$ and $\vec{g}$ of (1.2) in a manner compatible with the partitioning in (1.4), and (1.2) is equivalent to

$$\begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & F \\ F^T & 0 \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \end{pmatrix} + \begin{pmatrix} \vec{g}_1 \\ \vec{g}_2 \end{pmatrix}. \tag{4.1}$$

Without using vectors with twice as many components, as was the case in §2, the successive overrelaxation iterative method can be rigorously applied directly to (4.1), giving

$$\begin{cases} \vec{x}_1^{(m+1)} = \omega \{F \vec{x}_2^{(m)} + \vec{g}_1 - \vec{x}_1^{(m)}\} + \vec{x}_1^{(m)} \\ \vec{x}_2^{(m+1)} = \omega \{F^T \vec{x}_1^{(m+1)} + \vec{g}_2 - \vec{x}_2^{(m)}\} + \vec{x}_2^{(m)}, \qquad m \geqq 0, \end{cases} \tag{4.2}$$

where $\vec{x}_1^{(0)}$, $\vec{x}_2^{(0)}$ are arbitrary guesses. The best choice of ω is given by

$$\omega_b = \frac{2}{1 + \sqrt{1 - \varrho^2(B)}} = \frac{2}{1 + \sqrt{1 - \varrho(F F^T)}}. \tag{4.3}$$

We can also apply to (4.1) the Chebyshev semi-iterative method of (2.9), which gives, by vector components,

$$\begin{cases} \vec{x}_1^{(m+1)} = \omega_{m+1} \{F \vec{x}_2^{(m)} + \vec{g}_1 - \vec{x}_1^{(m-1)}\} + \vec{x}_1^{(m-1)}, \\ \vec{x}_2^{(m+1)} = \omega_{m+1} \{F^T \vec{x}_1^{(m)} + \vec{g}_2 - \vec{x}_2^{(m-1)}\} + \vec{x}_2^{(m-1)}, \qquad m \geqq 1, \end{cases} \tag{4.4}$$

where $\vec{x}_1^{(1)} = F \vec{x}_2^{(0)} + \vec{g}_1$, and $\vec{x}_2^{(1)} = F^T \vec{x}_1^{(0)} + \vec{g}_2$, and these equations determine the vector sequences $\{\vec{x}_1^{(m)}\}_{m=0}^{\infty}$, and $\{\vec{x}_2^{(m)}\}_{m=0}^{\infty}$. It is interesting to observe that the proper subsequences $\{\vec{x}_1^{(2m+1)}\}_{m=0}^{\infty}$, and $\{\vec{x}_2^{(2m)}\}_{m=0}^{\infty}$ can be iteratively determined from

$$\begin{cases} \vec{x}_1^{(2m+1)} = \omega_{2m+1} \{F \vec{x}_2^{(2m)} + \vec{g}_1 - \vec{x}_1^{(2m-1)}\} + \vec{x}_1^{(2m-1)}, \qquad m \geqq 1, \\ \vec{x}_2^{(2m+2)} = \omega_{2m+2} \{F^T \vec{x}_1^{(2m+1)} + \vec{g}_2 - \vec{x}_2^{(2m)}\} + \vec{x}_2^{(2m)}, \qquad m \geqq 0, \end{cases} \tag{4.5}$$

where again $\vec{x}_1^{(1)}=F\vec{x}_2^{(0)}+\vec{g}_1$. Thus, this iterative method requires no additional vector storage over the successive overrelaxation iterative method*, and requires but the single vector guess $\vec{x}_2^{(0)}$.

We shall call this iterative method, obtained by selecting appropriate subsequences of Chebyshev semi-iterative method, the *cyclic Chebyshev semi-iterative method* for the matrix equation (4.1).

In the primitive case of §3, we considered the (primitive) successive overrelaxation iterative method, or equivalently the second order Richardson method $\alpha=\omega$ and $\beta=-1$, with the starting procedures

$$\vec{x}^{(1)}=B\vec{x}^{(0)}+\vec{g} \tag{4.6}$$

and

$$\left\{\begin{aligned}\vec{x}^{(1)}&=B\vec{x}^{(0)}+\vec{g}\\ \vec{x}^{(2)}&=B\vec{x}^{(1)}+\vec{g}.\end{aligned}\right. \tag{4.6'}$$

Here again, it is only necessary in the cyclic case to compute the proper subsequences $\{\vec{x}_1^{(2m+1)}\}_{m=0}^{\infty}$ and $\{\vec{x}_2^{(2m)}\}_{m=0}^{\infty}$, and the starting procedures (4.6) and (4.6′) become in this case

$$\vec{x}_1^{(1)}=F\vec{x}_2^{(0)}+\vec{g}_1 \tag{4.7}$$

and

$$\left\{\begin{aligned}\vec{x}_1^{(1)}&=F\vec{x}_2^{(0)}+\vec{g}_1\\ \vec{x}_2^{(2)}&=F^T\vec{x}_1^{(1)}+\vec{g}_2.\end{aligned}\right. \tag{4.7'}$$

If $\omega_m\equiv\omega$ then we see that (4.5) reduces to (4.2). Thus, for the cyclic Chebyshev semi-iterative method, a sequence of parameters ω_m is necessary whereas for the successive overrelaxation method, only one parameter is necessary. The variant of the successive overrelaxation method with the starting procedure (4.7′) has been studied by SHELDON [*15*] and the corresponding matrix operator for m iterative is denoted by $\mathfrak{L}_{\omega_b}^{m-1}\,\mathfrak{L}_1$. The relationship between the cyclic Chebyshev semi-iterative method and the successive overrelaxation method is quite close. Indeed, as given by (2.18), $\lim\limits_{m\to\infty}\omega_m=\omega_b$, and it is in fact shown in [*7*], under simple assumptions, that the cyclic Chebyshev semi-iterative method must degenerate *numerically* into the successive overrelaxation iterative method.

As in §3, we will compare the successive overrelaxation iterative method of (4.2) for the starting procedures of (4.7) and (4.7′) with the cyclic Chebyshev semi-iterative method of (4.5), and as we shall see, using spectral norms as a basis for comparison, the cyclic Chebyshev semi-iterative method is superior** to the successive overrelaxation iterative method.

* This idea has already been used by RILEY [*13*] to make the second order Richardson iterative method competitive in storage with the successive overrelaxation iterative method.

** In relationship to [*18*], Theorem 1 of [*18*] shows with spectral radii as a basis for comparison, that the iterative method of (4.2) with $\omega=\omega_b$ is at least twice as fast as the iterative method of (4.4). Using the cyclic Chebyshev semi-iterative method of (4.5) eliminate this factor of 2 since, from (4.5), each *complete* iteration of (4.5) increases the iteration indices of the vectors $\vec{x}_2$ and $\vec{x}_1$ by *two*.

§ 5. Cyclic matrices. Comparison of methods

The results in this section depend strongly upon the methods and results of §3, as well as the recent works of SHELDON [*15*]. For the Chebyshev semi-iterative method, the successive overrelaxation iterative method, and the second order Richardson iterative method of §2, we partition the error vector $\vec{\overset{*}{\varepsilon}}{}^{(m)}$ in a manner compatible with the form of the matrix B in (4.1), and we define

$$\vec{\overset{*}{\varepsilon}}{}^{(m)} \equiv \begin{pmatrix} \vec{\overset{*}{\varepsilon}}{}_1^{(m)} \\ \vec{\overset{*}{\varepsilon}}{}_2^{(m)} \end{pmatrix}, \qquad m>0, \tag{5.1}$$

where $\vec{\overset{*}{\varepsilon}}{}_1^{(0)} = \vec{\varepsilon}_1^{(0)}$ and $\vec{\overset{*}{\varepsilon}}{}_2^{(0)} = \vec{\varepsilon}_2^{(0)}$ are the vector components of the initial error vector. For these methods, we have that

$$\vec{\overset{*}{\varepsilon}}{}^{(m)} = p_m(B)\,\vec{\varepsilon}^{(0)}, \qquad m>0, \tag{5.2}$$

where the matrix operator $p_m(B)$ corresponds respectively to the matrix operators $\tilde{p}_m(B)$, $r_m(B)$, $t_m(B)$ and $s_m(B)$ of §3. For the cyclic Chebyshev semi-iterative method, and the (cyclic) successive overrelaxation iterative method with the starting procedures of (4.7) and (4.7′), the corresponding error vector for the m-th complete iteration of these methods is defined by

$$\vec{\delta}^{(m)} \equiv \begin{pmatrix} \vec{\overset{*}{\varepsilon}}{}_1^{(2m-1)} \\ \vec{\overset{*}{\varepsilon}}{}_2^{(2m)} \end{pmatrix}, \qquad m>0. \tag{5.3}$$

From (2.8′), (3.21), and (3.28), it follows that the polynomials $p_m(x)$ of odd degree contain only odd powers of x, while the polynomials of even degree contain only even powers of x. Thus, we define polynomials U_m and V_m through

$$\begin{cases} p_{2m+1}(x) = x\,U_m(x^2), & m \geqq 0, \\ p_{2m}(x) = V_m(x^2), & m \geqq 0. \end{cases} \tag{5.4}$$

Since the matrix has the form (4.1), then

$$B^{2m} = \left(\begin{array}{c|c} (FF^T)^m & 0 \\ \hline 0 & (F^TF)^m \end{array}\right) \quad \text{and} \quad B^{2m+1} = \left(\begin{array}{c|c} 0 & (FF^T)^m F \\ \hline (F^TF)^m F^T & 0 \end{array}\right), \tag{5.5}$$

and the definitions of (5.4) and the properties of the powers of the matrix B allow us to express $\vec{\delta}^{(m)}$ in the simple form

$$\vec{\delta}^{(m)} = \left(\begin{array}{c|c} 0 & U_{m-1}(FF^T)\cdot F \\ \hline 0 & V_m(F^TF) \end{array}\right) \vec{\varepsilon}^{(0)}, \qquad m>0. \tag{5.6}$$

Defining the matrix above as $P_m(B)$, this becomes

$$\vec{\delta}^{(m)} = P_m(B)\,\vec{\varepsilon}^{(0)}, \qquad m>0. \tag{5.6′}$$

We analogously define the 2×2 matrix $Q_m(\mu)$ as

$$Q_m(\mu)=\left(\begin{array}{c|c} 0 & U_{m-1}(\mu^2)\,\mu \\ \hline 0 & V_m(\mu^2)\end{array}\right),\qquad m>1, \tag{5.7}$$

whose spectral norm is easily seen to be

$$\tau[Q_m(\mu)]=\{\mu^2 U_{m-1}^2(\mu^2)+V_m^2(\mu^2)\}^{\frac{1}{2}},\qquad m\geqq 1. \tag{5.8}$$

From (5.4), this becomes

$$\tau[Q_m(\mu)]=\{p_{2m-1}^2(\mu)+p_{2m}^2(\mu)\}^{\frac{1}{2}},\qquad m\geqq 1. \tag{5.8'}$$

We now employ what is essentially a converse of Theorem 2 of the recent work of SHELDON [*15*]*. Denoting the eigenvalues of the matrix B by μ_i, $1\leqq i\leqq N$, then

$$\tau[P_m(B)]=\max_{1\leqq i\leqq N}\{p_{2m-1}^2(\mu_i)+p_{2m}^2(\mu_i)\}^{\frac{1}{2}},\qquad m\geqq 1. \tag{5.9}$$

Let us now denote the matrix operator of (5.6') associated with the polynomials $\tilde{p}_m(B)$, $r_m(B)$, $t_m(B)$, and $s_m(B)$ of §3 as $\tilde{P}_m(B)$, $R_m(B)$, $T_m(B)$, and $S_m(B)$, respectively. Then it follows immediately from the results of §3 that

$$\left\{\begin{aligned} \tau[\tilde{P}_m(B)]&=\{\tau^2(\tilde{p}_{2m-1}(B))+\tau^2(\tilde{p}_{2m}(B))\}^{\frac{1}{2}}\\ \tau[R_m(B)]&=\{\tau^2(r_{2m-1}(B))+\tau^2(r_{2m}(B))\}^{\frac{1}{2}}\\ \tau[T_m(B)]&=\{\tau^2(t_{2m-1}(B))+\tau^2(t_{2m}(B))\}^{\frac{1}{2}}\\ \tau[S_m(B)]&=\{\tau^2(s_{2m-1}(B))+\tau^2(s_{2m}(B))\}^{\frac{1}{2}}.\end{aligned}\right. \tag{5.10}$$

Since $\tau(\tilde{p}_m(B))$, $\tau(r_m(B))$, $\tau(t_m(B))$ and $\tau(s_m(B))$ decrease monotonically with m, so do $\tau[\tilde{P}_m(B)]$, $\tau[R_m(B)]$, $\tau[T_m(B)]$**, and $\tau[S_m(B)[$. Furthermore, by Theorem 1, for $m>1$ and $0<\varrho<1$,

$$\tau(\tilde{p}_m(B))<\tau(r_m(B))<\tau(t_m(B))<\tau(s_m(B)), \tag{5.11}$$

so that

Lemma 2. For all $m>1$ and $0<\varrho<1$,

$$\tau[\tilde{P}_m(B)]<\tau[R_m(B)]<\tau[T_m(B)]<\tau[S_m(B)].$$

The spectral norm of the successive overrelaxation iterative method of (4.2) for the case when ω is fixed equal to ω_b has been recently calculated by SHELDON

* Specifically, in the notation of SHELDON [*15*], the result we are using is given in the following

Theorem. If λ is a non-zero eigenvalue of L, then λ is also an eigenvalue of some $T(\mu_i)$ where μ_i is an eigenvalue of the matrix B.

This result is tacitly assumed in [*15*], and we are indebted to Dr. SHELDON for supplying us with a proof of this result.

** The quantity $\tau[T_m(B)]$ in (5.10) is algebraically equivalent to the expression for $\tau[\mathfrak{L}_{\omega_b}^{m-1}\mathfrak{L}_1]$ in [*15*]. Thus, the monotonicity noted above strengthens SHELDON's Theorem 4 in [*15*].

[*15*], and if $\mathfrak{L}^m_{\omega_b}$ represents the corresponding matrix operator for m iterations, then*

$$(5.12) \qquad \tau[\mathfrak{L}^m_{\omega_b}] = l_m^{\frac{1}{2}}(\omega_b - 1)^m, \qquad m \geqq 0,$$

where l_m is the larger root of

$$(5.13) \qquad l^2 - \left[8m^2 + 4m^2\left(r^2 + \frac{1}{r^2}\right) + 2\right] l + 1 = 0,$$

and $r^2 = \omega_b - 1$, so that

$$(5.12') \qquad \tau[\mathfrak{L}^m_{\omega_b}] = \left(\frac{2m}{\varrho} + \sqrt{\frac{4m^2}{\varrho^2} + 1}\right) \cdot (\omega_b - 1)^m, \qquad m \geqq 0.$$

We observe that in obtaining the spectral norms for the four iterative methods just considered, no assumption has been made about a special form of the initial error $\vec{\varepsilon}^{(0)}$, and thus the four iterative methods can be directly compared.

Then we have

Theorem 2. In the cyclic case for all $m > 1$ and $0 < \varrho < 1$, with no special assumption on the form of the initial error vector $\vec{\varepsilon}^{(0)}$,

$$(5.14) \qquad \begin{cases} \tau[\widetilde{P}_m(B)] < \tau[R_m(B)] < \tau[T_m(B)] < \tau[S_m(B)], \quad \text{and} \\ \tau[\widetilde{P}_m(B)] < \tau[\mathfrak{L}^m_{\omega_b}]. \end{cases}$$

Thus, the spectral norm of the matrix operator for the cyclic Chebyshev semi-iterative method is less than the spectral norm of the matrix operators for the successive overrelaxation iterative method and its modification by SHELDON.

Proof. From Lemma 2, it suffices to show that $\tau[\widetilde{P}_m(B)] < \tau[\mathfrak{L}^m_{\omega_b}]$ for all $m > 1$ and $0 < \varrho < 1$. By using the expressions of (3.8), (5.10), and (5.12'), this inequality reduces to

$$(5.15) \qquad \left\{r^{-2}\left(\frac{2}{1 + r^{4m-2}}\right)^2 + \left(\frac{2}{1 + r^{4m}}\right)^2\right\}^{\frac{1}{2}} < \frac{2m}{\varrho} + \sqrt{\frac{4m^2}{\varrho^2} + 1},$$

which is easily shown to be true for all $m > 1$, and $0 < \varrho < 1$. In fact, the proof of the above inequality shows that the ratio $\tau[\mathfrak{L}^m_{\omega_b}]/\tau[\widetilde{P}_m(B)]$ is a strictly increasing function of m, $m > 1$, for all $0 < \varrho < 1$. We strengthen the inequalities of (5.14) by including

Theorem 3. In the cyclic case with $0 < \varrho < 1$, and no special assumptions on the form of the initial error vector $\vec{\varepsilon}^{(0)}$, then the ratios

$$(5.16) \qquad \frac{\tau[R_m(B)]}{\tau[\widetilde{P}_m(B)]} \equiv \alpha_m; \qquad \frac{\tau[L^m_{\omega_b}]}{\tau[\widetilde{P}_m(B)]} \equiv \beta_m$$

are strictly increasing for $m > 1$, and

$$(5.17) \qquad \alpha_m = O(m), \qquad \beta_m = O(m), \qquad m \to \infty.$$

* Theorem 3 of [*15*] contains minor misprints, which we are now correcting.

Proof. It is an easy computation to show that $\tau[\tilde{P}_m(B)] < 2r^{2m}(1+r^{-2})$, and that $2r^{2m}(1+r^{-2})$ is smaller than either $\tau[R_m(B)]$ or $\tau[\mathfrak{L}^m_{\omega_b}]$. The statements of (5.16) and (5.17) then follow immediately*.

§ 6. Applications

A great many physical and engineering problems lead to the numerical solution of matrix equations of the form

$$A\vec{x} = \vec{k}, \tag{6.1}$$

where A is an $N \times N$ real symmetric and positive definite matrix which can after a suitable permutation of indices, be partitioned so that

$$A = \left[\begin{array}{cccc|cccc} A_{1,1} & 0 & \dots & 0 & A_{1,p+1} & & \dots & A_{1,s} \\ 0 & A_{2,2} & & 0 & \vdots & & & \vdots \\ \vdots & & \ddots & \vdots & \vdots & & & \vdots \\ 0 & 0 & \dots & A_{p,p} & A_{p,p+1} & & \dots & A_{p,s} \\ \hline A^T_{1,p+1} & \dots & & A^T_{p,p+1} & A_{p+1,p+1} & 0 & \dots & 0 \\ \vdots & & & \vdots & 0 & A_{p+2,p+2} & & 0 \\ \vdots & & & \vdots & \vdots & & \ddots & \vdots \\ A^T_{1,s} & \dots & & A^T_{p,s} & 0 & 0 & \dots & A_{s,s} \end{array}\right] \tag{6.2}$$

where the diagonal blocks $A_{j,j}$ are $n_j \times n_j$ matrices, $n_j \geqq 1$ for $1 \leqq j \leqq s$, and $\sum_{j=1}^{s} n_j = N$. ARMS, GATES, and ZONDEK [1] extended the original analysis of YOUNG [26] and FRANKEL [6] to what is called the *successive block overrelaxation iterative method*, and it can be verified that the assumptions on the matrix A above are sufficient for the application of their theory. Let the vectors $\vec{x}$ and $\vec{k}$ of (6.1) be partitioned in a manner compatible with (6.2). Then, we can write (6.1) as

$$\left\{\begin{array}{ll} A_{j,j} X_j + \sum\limits_{k=1}^{s-p} A_{j,p+k} X_{p+k} = K_j, & 1 \leqq j \leqq p, \\ A_{p+j,p+j} X_{p+j} + \sum\limits_{k=1}^{p} A^T_{k,p+j} X_k = K_{p+j}, & 1 \leqq j \leqq s-p. \end{array}\right. \tag{6.3}$$

The square submatrices $A_{j,j}$, $1 \leqq j \leqq s$, are evidently non-singular, so that if the block diagonal matrix C is defined by

$$C \equiv \begin{bmatrix} A_{1,1} & 0 & \dots & 0 \\ 0 & A_{2,2} & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & A_{s,s} \end{bmatrix}, \tag{6.4}$$

* Mr. DAVID FEINGOLD of Electricité de France (Paris) has recently proved (private communication) that the ratio $\{\tau[\mathfrak{L}^m_{\omega_b}]/\tau[R_m(B)]\}$ is strictly increasing for $m > 1$, $0 < \varrho < 1$, which strengthens Theorems 2 and 3.

then C is also non-singular. Now, $C^{-1}A$ has unit diagonal entries, and we define the matrix B as

$$C^{-1}A \equiv I - B, \tag{6.5}$$

so that the matrix B has zero diagonal entries. More precisely, B has the form

$$B = \left[\begin{array}{ccc|ccc} 0 & \cdots & 0 & B_{1,p+1} & \cdots & B_{1,s} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & B_{p,p+1} & \cdots & B_{p,s} \\ \hline B_{p+1,1} & \cdots & B_{p+1,p} & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ B_{s,1} & \cdots & B_{s,p} & 0 & \cdots & 0 \end{array}\right]. \tag{6.6}$$

With the definition of the matrix B in (6.5), (6.1) becomes

$$\vec{x} = B\vec{x} + C^{-1}\vec{k}, \tag{6.7}$$

The successive block overrelaxation iterative method applied to (6.7) is defined by

$$\begin{cases} X_j^{(m+1)} = \omega\left[\sum\limits_{k=1}^{s-p} B_{j,p+k} X_{p+k}^{(m)} + A_{j,j}^{-1} K_j - X_j^{(m)}\right] + X_j^{(m)}, \quad 1 \leqq j \leqq p, \\ X_{p+j}^{(m+1)} = \omega\left[\sum\limits_{k=1}^{p} B_{p+j,k} X_k^{(m+1)} + A_{p+j,p+j}^{-1} K_{p+j} - X_{p+j}^{(m)}\right] + X_{p+j}^{(m)}, \\ \qquad\qquad 1 \leqq j \leqq s - p, \end{cases} \tag{6.8}$$

where the $X_j^{(0)}$, $1 \leqq j \leqq s$, are given vector components of the given initial vector guess $\vec{x}^{(0)}$. The optimum value of ω is computed from (4.3), where the $N \times N$ matrix B is defined in (6.5). Equivalently, the iterations of (6.8) can be defined also from

$$X_j^{(m+1)} = \omega\left[\overset{*}{X}{}_j^{(m+1)} - X_j^{(m)}\right] + X_j^{(m)}, \qquad 1 \leqq j \leqq s, \tag{6.9}$$

where

$$\begin{cases} A_{j,j} \overset{*}{X}{}_j^{(m+1)} \equiv -\sum\limits_{k=1}^{s-p} A_{j,p+k} X_{p+k}^{(m)} + K_j, \quad 1 \leqq j \leqq p, \\ A_{p+j,p+j} \overset{*}{X}{}_{p+j}^{(m+1)} \equiv -\sum\limits_{k=1}^{p} A_{k,p+j}^{T} X_k^{(m+1)} + K_{p+j}, \quad 1 \leqq j \leqq s - p. \end{cases} \tag{6.9'}$$

Equation (6.9′) shows that, in order to carry out the successive block overrelaxation iterative method, we have assumed that matrix equations of the form

$$A_{j,j} X_j = G_j, \qquad 1 \leqq j \leqq s \tag{6.10}$$

can be solved directly for X_j, given G_j.

The matrix C defined in (6.4) is symmetric and positive definite, so that the matrices $C^{\frac{1}{2}}$ and $C^{-\frac{1}{2}}$ are uniquely defined. Forming the product $C^{-\frac{1}{2}}AC^{-\frac{1}{2}}$, we see that this product matrix also has unit diagonal entries, and in analogy with (6.5), we define the matrix $\tilde{B}$ by

$$C^{-\frac{1}{2}} A C^{-\frac{1}{2}} \equiv I - \hat{B}. \tag{6.11}$$

The matrix $\widetilde{B}$ has the same cyclic form as does B of (6.7), and since $C^{-\frac{1}{2}}AC^{-\frac{1}{2}}$ is a definite and symmetric matrix, it follows from (6.11) that $\widetilde{B}$ is symmetric and convergent. Defining

$$C^{\frac{1}{2}}\vec{x}=\vec{y}, \qquad C^{-\frac{1}{2}}\vec{k}=\vec{l} \tag{6.12}$$

and using (6.11), (6.1) reduces to

$$\vec{y}=\widetilde{B}\vec{y}+\vec{l}. \tag{6.13}$$

The matrix $\widetilde{B}$ is similar to B, with

$$\widetilde{B}=C^{\frac{1}{2}}BC^{-\frac{1}{2}}. \tag{6.14}$$

Summarizing, we have reduced our original problem (6.1) by means of a change of variables to the form (6.13), where $\widetilde{B}$ is symmetric, cyclic, and convergent.

We now apply the cyclic Chebyshev semi-iterative method to the numerical solution of (6.13). If the vector components $Y_j^{(0)}$, $1\leqq j\leqq p$, are given, then

$$\left\{\begin{aligned} Y_{p+j}^{(2m+1)} &= \omega_{2m+1}\left\{\sum_{k=1}^{p}\widetilde{B}_{p+j,k}Y_k^{(2m)}+L_{p+j}-Y_{p+j}^{(2m-1)}\right\}+Y_{p+j}^{(2m-1)}, \\ &\qquad\qquad 1\leqq j\leqq s-p, \\ Y_j^{(2m+2)} &= \omega_{2m+2}\left\{\sum_{k=1}^{s-p}\widetilde{B}_{j,p+k}Y_{p+k}^{(2m+1)}+L_j-Y_j^{(2m)}\right\}+Y_j^{(2m)}, \\ &\qquad\qquad 1\leqq j\leqq p, \qquad m\geqq 0, \end{aligned}\right. \tag{6.15}$$

defines the cyclic Chebyshev semi-iterative method. The ω's are calculated from (2.10), where $\varrho(B)=\varrho(\widetilde{B})$, since $\widetilde{B}$ is similar to B. To show now the relationship of this method to the successive block overrelaxation iterative method of (6.9)—(6.9′), we write (6.15) equivalently as

$$\left\{\begin{aligned} Y_{p+j}^{(2m+1)} &= \omega_{2m+1}\left(\overset{*}{Y}{}_{p+j}^{(2m+1)}-Y_{p+j}^{(2m-1)}\right)+Y_{p+j}^{(2m-1)}, \quad 1\leqq j\leqq s-p, \quad m\geqq 0, \\ Y_j^{(2m+2)} &= \omega_{2m+2}\left(\overset{*}{Y}{}_j^{(2m+2)}-Y_j^{(2m)}\right)+Y_j^{(2m)}, \quad 1\leqq j\geqq p, \quad m\geqq 0, \end{aligned}\right. \tag{6.16}$$

where

$$\left\{\begin{aligned} \overset{*}{Y}{}_{p+j}^{(2m+1)} &= \sum_{k=1}^{p}\widetilde{B}_{p+j,k}Y_k^{(2m)}+L_{p+j}, \quad 1\leqq j\leqq s-p, \quad m\geqq 0, \\ \overset{*}{Y}{}_j^{(2m+2)} &= \sum_{k=1}^{s-p}\widetilde{B}_{j,p+k}Y_{p+k}^{(2m+1)}+L_j, \quad 1\leqq j\leqq p, \quad m\geqq 0. \end{aligned}\right. \tag{6.16′}$$

By using the definitions of (6.11) and (6.12), it follows that (6.15) is *equivalent* to (6.9)—(6.9′), provided the proper ω's are used in each step. In essence then, we can *indirectly* carry out the modified Chebyshev semi-iterative method of (6.15) by performing the iterations

$$\left\{\begin{aligned} X_{p+j}^{(m+1)} &= \omega_{2m+1}\left(\overset{*}{X}{}_{p+j}^{(m+1)}-X_{p+j}^{(m)}\right)+X_{p+j}^{(m)}, \quad 1\leqq j\leqq s-p, \quad m\geqq 0, \\ X_j^{(m+1)} &= \omega_{2m+2}\left(\overset{*}{X}{}_j^{(m+1)}-X_j^{(m)}\right)+X_j^{(m)}, \quad 1\leqq j\leqq p, \quad m\geqq 0, \end{aligned}\right. \tag{6.9″}$$

where $\overset{*}{X}{}_j^{(m)}$, $1\leqq j\leqq s$, is defined in (6.9′).

In terms of spectral norms, let $\vec{\delta}^{(m)} = \begin{pmatrix} \vec{\varepsilon}_1^{(2m)} \\ \vec{\varepsilon}_2^{(2m+1)} \end{pmatrix}$ denote the error vector for the m-th complete iterate of (6.15), relative to the matrix $\tilde{B}$. From §5, we can state that

$$\|\vec{\delta}^{(m)}\| \leqq \tau[\tilde{P}_m(\tilde{B})] \cdot \|\delta^{(0)}\|, \qquad m \geqq 0. \tag{6.17}$$

If $\vec{\alpha}^{(m)} \equiv \begin{pmatrix} \vec{\alpha}_1^{(2m)} \\ \vec{\alpha}_2^{(2m+1)} \end{pmatrix}$ is the error for the m-th complete iteration of (6.9″), relative to the matrix B, then from $C^{\frac{1}{2}}\vec{x} = \vec{y}$, we have

$$\|C^{\frac{1}{2}}\vec{\alpha}^{(m)}\| \leqq \tau[\tilde{P}_m(\tilde{B})] \cdot \|C^{\frac{1}{2}}\vec{\alpha}^{(0)}\|, \qquad m \geqq 0. \tag{6.18}$$

Since both $C^{\frac{1}{2}}$ and $C^{-\frac{1}{2}}$ are symmetric and positive definite, their spectral radii coincide with their spectral norms, so that

$$\|C^{\frac{1}{2}} z\| \leqq \varrho(C^{\frac{1}{2}}) \, \|z\|, \tag{6.19}$$

and

$$\|C^{\frac{1}{2}} z\| \geqq \frac{\|z\|}{\varrho(C^{-\frac{1}{2}})}, \tag{6.19'}$$

where equality is possible in both (6.19) and (6.19′). Combining these inequalities, we have*

$$\|\vec{\alpha}^{(m)}\| \leqq \tau[\tilde{P}_m(\tilde{B})] \, [\varrho(C^{-\frac{1}{2}}) \cdot \varrho(C^{\frac{1}{2}})] \, \|\vec{\alpha}^{(0)}\|, \qquad m \geqq 0. \tag{6.20}$$

From the results of §5, of the iterative methods studied, the cyclic Chebyshev semi-iterative method of (6.16)—(6.16′) gives the smallest spectral norm relative to the matrix equation of (6.13). Since actually iterating by means of (6.9′)—(6.9″) is equivalent to iterating by means of (6.16)—(6.16′), we arrive at the conclusion that the iterations of (6.9′)—(6.9″) are quite efficient.

We now list some well known problems which numerically give rise to matrix equations of the form (6.1), where the matrix A can be written as in (6.2). Clearly, such a list would include all problems which have been previously rigorously attacked by the successive overrelaxation iterative method, and its extensions.

A. Dirichlet problem in a plane bounded region, using a five point approximation to LAPLACE's equation. Here, one can use successive point overrelaxation [*6, 19, 26*], successive line relaxation [*1, 3, 8*], or successive two line overrelaxation [*12, 21*], all these methods corresponding to different partitionings of the matrix A.

B. Dirichlet problem in a plane bounded region, using a ninepoint approximation to LAPLACE's equation. Here, one can use successive line overrelaxation [*1, 21*], or successive two line overrelaxation [*8, 12, 21*].

C. Biharmonic problem in a plane bounded region, using a thirteen point approximation to the biharmonic equation. Here, one can use successive two line overrelaxation [*8, 12, 21*].

* The quantity $(\varrho(M^{-1})) \cdot \varrho(M)$ is also called the *P-condition number* [*17*] for a non-singular matrix M, and is denoted by $P(M)$.

In all these problems, the cyclic Chebyshev semi-iterative method can be used, and from the results of §5, this iterative method gives the smaller spectral norm than the successive overrelaxation iterative methods.

Finally, matrix equations (6.1) do arise in which the matrix A cannot, after a permutation of indices, be put into the form of (6.2), even with proper partitioning. For example, in [*21*], a class of iterative methods called *primitive* iterative methods are studied, and for this class the results of § 2—3 are pertinent. It should also be said that even though the matrix A of (6.1) can be partitioned so that (6.2) holds, it can very well be the case that the diagonal blocks $A_{i,i}$, which must be directly inverted, as in (6.10), in order to apply the cyclic theory, are either too large in size or too complicated to permit such direct inversion. Thus, in solving the Dirichlet problem in a plane bounded region, if one chooses to use a nine point approximation to LAPLACE'S equation, but is unwilling to directly invert more than one equation in one unknown, a primitive iterative method results. Here too the results of § 2—3 are pertinent.

§ 7. Numerical Results

We will now give results from both algebraic and numerical investigations, comparing the Chebyshev semi-iterative method with variants of the successive overrelaxation iterative method in the cyclic case. First, if $\vec{\varepsilon}^{(0)}$ is the vector error of our initial estimate $\vec{x}_0$ of the unique solution of $A\vec{x}=\vec{k}$, and $\vec{\delta}^{(m)}$ is the error vector for the m-th complete iteration, then from (5.6'),

$$(7.1) \qquad \frac{\|\vec{\delta}^{(m)}\|}{\|\vec{\varepsilon}^{(0)}\|} \leqq \tau[P_m(B)], \qquad m>0.$$

Thus, if $m(\delta)$ is the least positive integer for which

$$(7.2) \qquad \tau[P_m(B)] \leqq \delta, \qquad 0<\delta<1,$$

then $m(\delta)$ is an *upper bound* for the number of iterations necessary to reduce the Euclidean length of the initial error by the factor δ. Let $m_1(\delta)$, $m_2(\delta)$, $m_3(\delta)$, and $m_4(\delta)$ denote $m(\delta)$ when $P_m(B)$ is taken to be respectively $\tilde{P}_m(B)$, $R_m(B)$, $T_m(B)$ and $\mathfrak{L}_{\omega_b}^m$. The tables 1—4 give $m_i(\delta)$ for various values of δ and $\varrho(B)$.

Table 1. $\omega_b=1.8195$; $\varrho=0.99507$

	$\delta=0.1$	$\delta=0.05$	$\delta=0.01$	$\delta=0.005$	$\delta=0.001$
$m_1(\delta)$	18	21	29	33	41
$m_2(\delta)$	22	27	36	40	49
$m_3(\delta)$	23	27	37	41	50
$m_4(\delta)$	37	41	50	54	63

Table 2. $\omega_b=1.93419$; $\varrho=0.999421$

	$\delta=0.1$	$\delta=0.05$	$\delta=0.01$	$\delta=0.005$	$\delta=0.001$
$m_1(\delta)$	50	60	84	94	117
$m_2(\delta)$	64	77	104	116	142
$m_3(\delta)$	65	77	105	116	143
$m_4(\delta)$	126	137	163	174	200

Table 3. $\omega_b=1.95218$; $\varrho(B)=0.9997$

	$\delta=0.1$	$\delta=0.05$	$\delta=0.01$	$\delta=0.005$	$\delta=0.001$
$m_1(\delta)$	69	93	116	130	163
$m_2(\delta)$	89	106	144	160	197
$m_3(\delta)$	89	107	145	161	198
$m_4(\delta)$	182	198	234	250	285

Table 4. $\omega_b=1.97211$; $\varrho(B)=0{,}9999$

	$\delta=0.1$	$\delta=0.05$	$\delta=0.01$	$\delta=0.005$	$\delta=0.001$
$m_1(\delta)$	119	143	200	225	282
$m_2(\delta)$	154	183	249	277	341
$m_3(\delta)$	154	184	250	278	341
$m_4(\delta)$	337	364	426	453	514

It is interesting to point out that the following

$$\lim_{\delta \to 0} \frac{m_i^{(\delta)}}{m_j^{(\delta)}} = 1 \tag{7.3}$$

can be proved* for all i, j. Thus, the cyclic Chebyshev semi-iterative method cannot require, for very small $\delta > 0$, percentagewise substantially different numbers of iterations than those required by the successive overrelaxation method. However, for slowly convergent problems, $\varrho(B)$ close to unity, there is a considerable advantage in using the cyclic Chebyshev in practical problems where δ is approximately 10^{-2}.

The above, while constituting an algebraic study of the various methods, does not give a complete picture of the comparison between these methods, because of the inequalities in (7.1) and (7.2). Although equality is attainable in (7.1) and (7.2), so that the numbers of iterations in Tables 1—4 are also attainable, we include results of numerical experiments in the cyclic case. In an effort to make the numerical experiments as up-to-date and practical as possible, we have compared the successive two line over-relaxation iterative method [*8*, *12*, *21*] with the cyclic Chebyshev semi-iterative method for the same partitioning of the matrix A of (6.2), in the numerical solution of self-adjoint partial differential equation

$$-\operatorname{div}\{D(x, y) \operatorname{grad} u(x, y)\} + \sigma(x, y)\, u(x, y) = S(x, y), \tag{7.4}$$

in a plane bounded region Ω, where D and σ are positive in Ω, with boundary conditions

$$\frac{\partial u(x, y)}{\partial n} = 0 \tag{7.5}$$

on the boundary Γ of Ω. These numerical problems involved non-constant mesh spacings. In part 1 of each problem, $S(x, y) \equiv 0$, so that the unique solution of the matrix problem of (6.1) is the null vector. With all the components of the initial vector $\vec{x}^{(0)}$ taken as 10^3, the iterations were continued until the maximum component of $\vec{x}^{(m)}$ was less than or equal to δ. In part 2 of each problem, $S(x, y) \equiv 1$ and with the same initial vector $\vec{x}^{(0)}$ as in part 1, the iterations were continued until

$$R^{(m+1)} = \sum |x_j^{(m+1)} - x_j^{(m)}| \tag{7.6}$$

satisfied $R^{(m+1)} \leqq \delta R^{(0)}$.

Because the norms of both parts of the experiment are convenient in computation, but not the spectral norms of the comparison, the following comparisons are of interest in connection with the relationships exhibited in §6. The successive overrelaxation method is applied to two different orderings of the matrix A: the first, the σ_1 ordering, is the ordering of (6.2); the second is the "normal" ordering in which the double lines of mesh points are swept serially through the mesh.

* See [*7*] for details.

Table 5. *Problem A 121 interior mesh points,* $\omega_b = 1.8195$

Part 1

Method	$\delta=0.1$	$\delta=0.01$	$\delta=0.005$	$\delta=0.001$
Cyclic Chebyshev	17	28	31	39
Sheldon's Modified SOR . . .	21	35	39	48
SOR with ω_b, σ_1 Ordering . . .	20	34	37	46
SOR with ω_b, Normal Ordering .	17	30	34	43

Part 2

Method	$\delta=0.1$	$\delta=0.01$	$\delta=0.005$
Cyclic Chebyshev	30	41	44
Sheldon's Modified SOR . . .	39	52	55
SOR with ω_b, σ_1 Ordering . . .	33	46	50
SOR with ω_b, Normal Ordering .	32	45	49

Table 6. *Problem B 667 interior mesh points,* $\omega_b = 1.93419$

Part 1

Method	$\delta=0.1$	$\delta=0.01$	$\delta=0.005$	$\delta=0.001$
Cyclic Chebyshev	71	106	110	133
Sheldon's Modified SOR . . .	88	123	134	157
SOR with ω_b, σ_1 Ordering . . .	93	127	137	160
SOR with ω_b, Normal Ordering .	81	121	133	155

Part 2

Method	$\delta=0.1$	$\delta=0.01$	$\delta=0.005$
Cyclic Chebyshev	83	113	119
Sheldon's Modified SOR . . .	113	147	157
SOR with ω_b, σ_1 Ordering . . .	97	133	143
SOR with ω_b, Normal Ordering .	91	127	137

For references, see Part I **3**, 147 (1961).

Space Technology Laboratories, Inc.
Los Angeles 45, California
and
Case Institute of Technology
Cleveland 6, Ohio
10900 Euclid Avenue

(Received June 17, 1960)

7

A GENERALIZED CONJUGATE GRADIENT METHOD FOR NON-SYMMETRIC SYSTEMS OF LINEAR EQUATIONS (WITH PAUL CONCUS)

SECOND INTERNATIONAL SYMPOSIUM ON COMPUTING METHODS IN APPLIED SCIENCES AND ENGINEERING, Versailles, France, December 15–19 1975;
Lecture Notes in Economics and Mathematical Systems, vol. 134, R. Glowinski and R. L. Lions, eds. Springer-Verlag, 56–65 (1976).

Gene Golub and I prepared this paper immediately after completion of the paper "A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations" with Dianne O'Leary (1976). Please see my background comments for that paper, which follows as Chapter 8.

Paul Concus
Berkeley, California, USA

A GENERALIZED CONJUGATE GRADIENT METHOD FOR NONSYMMETRIC SYSTEMS OF LINEAR EQUATIONS*

by

Paul Concus
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

and

Gene H. Golub
Computer Science Department
Stanford University
Stanford, CA 94305

0. Introduction

In a related paper [3] we discuss a generalized conjugate gradient (CG) iterative method for solving a system of real, linear, algebraic equations

$$Ax = b \quad , \tag{1.1}$$

where A is symmetric and positive definite. The method is based on splitting off from A an approximating symmetric, positive-definite matrix M that corresponds to a system of equations more easily solvable than is (1.1), and then accelerating the associated iteration using CG. The method appears to be especially effective for sparse matrices A arising from the discretization of boundary-value problems for elliptic partial differential equations. For these cases, naturally arising selections for M often result in iteration matrices possessing eigenvalue distributions for which CG acceleration is effective.

The CG method has a number of attractive properties when used as an iterative procedure:

(i) It does not require an estimation of parameters.

(ii) It takes advantage of the distribution of the eigenvalues of the iteration operator.

(iii) It requires fewer restrictions on the matrix A for optimal behavior than do such methods and successive overrelaxation.

In this paper we remove the restriction that A be symmetric, and require only that its symmetric part $(A + A^T)/2$ be positive definite. We derive the generalized CG method for this case, taking for the approximating matrix - M the symmetric part of A. We find that the method then simplifies, in that the computation of only one of the two CG parameters is required.

1. Derivation of the Method

We consider the system of linear equations

$$Ax = b \quad , \tag{1.1}$$

*Note added in electronic transfer: The *generalized conjugate gradient method* discussed here has become known more popularly as the *preconditioned conjugate gradient method.*

where A is a given n × n real matrix and b is a given real n-vector. We rewrite (1.1) as the system

$$Mx = Nx + b \ , \tag{1.2}$$

where $M = M^T = (A + A^T)/2$ is the symmetric part of A, and $N = -N^T = -(A - A^T)/2$ is the negative of its skew-symmetric part. We assume that M is positive definite. In [3],we discuss the solution of equations of the form (1.2) by a generalized CG method, for the case in which M is symmetric and positive definite and N is symmetric. In this paper, we derive the corresponding algorithm for the case in which N is skew-symmetric.

Our interest is in those situations for which it is a simpler computational task to solve

$$Mz = d \tag{1.3}$$

than it is to solve (1.1), and for which, in a sense to be described later, $M^{-1}N$ is not too large.

Consider an iteration of the form

$$x^{(k+1)} = x^{(k-1)} + \omega_{k+1}(\alpha_k z^{(k)} + x^{(k)} - x^{(k-1)}) \ , \tag{1.4}$$

where

$$Mz^{(k)} = r^{(k)} \ , \tag{1.5}$$

with

$$r^{(k)} = b - (M-N)x^{(k)} = b - Ax^{(k)} \ ,$$

the residual at the kth step. The quantities α_k and ω_{k+1} are scalar parameters.

Many iterative methods can be described by (1.4), e.g., if N were symmetric, the Chebyshev semi-iterative method and Richardson second order method would be of this form (cf. [5]). The generalized conjugate gradient method described below, which is also of this form, has the advantage over those two methods that no a priori information about the spectral radius of $M^{-1}N$ is needed for estimating parameters. Furthermore, it takes advantage of the actual distribution of the eigenvalues of $M^{-1}N$.

From (1.4) and (1.5), we obtain

$$Mz^{(k+1)} = Mz^{(k-1)} - \omega_{k+1}(\alpha_k Az^{(k)} + M(z^{(k-1)} - z^{(k)})) \ . \tag{1.6}$$

For the generalized CG method, the parameters $\{\alpha_k, \omega_{k+1}\}$ are computed so that

$$z^{(p)^T} Mz^{(q)} = 0 \qquad \text{for } p \neq q \text{ and } p, q = 0,1,\ldots,n-1 \ . \tag{1.7}$$

Since M is an n × n, symmetric,positive-definite matrix, (1.7) implies that for some $k \leq n$

$$z^{(k)} = \emptyset$$

and hence

$$x^{(k)} = x .$$

That is, the iteration converges in no more than n steps.

We derive the above result by induction. Assume

$$z^{(p)^T} M z^{(q)} = 0 \qquad \text{for } p \neq q \text{ and } p, q = 0, 1, \dots , k. \tag{1.8}$$

Since N is skew-symmetric, there holds that for any real n-vector w

$$w^T N w = 0 . \tag{1.9}$$

From (1.6), we have

$$z^{(k)^T} M z^{(k+1)} = z^{(k)^T} M z^{(k-1)} - \omega_{k+1}(\alpha_k z^{(k)^T} A z^{(k)} + z^{(k)^T} M(z^{(k-1)} - z^{(k)})),$$

and thus by (1.8) and (1.9),

$$z^{(k)^T} M z^{(k+1)} = -\omega_{k+1}(\alpha_k z^{(k)^T} M z^{(k)} - z^{(k)^T} M z^{(k)}) .$$

Hence by choosing $\alpha_k \equiv 1$, we obtain $z^{(k)^T} M z^{(k+1)} = 0$. Similarly, $z^{(k-1)^T} M z^{(k+1)} = 0$ for the choice

$$\omega_{k+1} = \frac{z^{(k-1)^T} M z^{(k-1)}}{z^{(k-1)^T} M z^{(k-1)} - z^{(k-1)^T} N z^{(k)}} . \tag{1.10}$$

We can simplify (1.10) by noting from (1.6), with (k+1) replaced with (k), that

$$z^{(k)^T} M z^{(k)} = \omega_k z^{(k)^T} N z^{(k-1)} ,$$

so that

$$-z^{(k-1)^T} N z^{(k)} = z^{(k)^T} M z^{(k)} / \omega_k .$$

We obtain

$$\omega_{k+1} = \left(1 + \frac{z^{(k)^T} M z^{(k)}}{z^{(k-1)^T} M z^{(k-1)}} \times \frac{1}{\omega_k} \right)^{-1} .$$

Then for $j \leq k-2$, we obtain from (1.6), (1.8), and (1.9) that

$$\begin{aligned} z^{(j)^T} M z^{(k+1)} &= z^{(j)^T} M z^{(k-1)} - \omega_{k+1}(z^{(j)^T}(M-N) z^{(k)} - z^{(j)^T} M(z^{(k-1)} - z^{(k)})) \\ &= \omega_{k+1} z^{(j)^T} N z^{(k)} . \end{aligned} \tag{1.11}$$

But, since for $\alpha_j = 1$,

$$Mz^{(j+1)} = Mz^{(j-1)} - \omega_{j+1}(-Nz^{(j)} + Mz^{(j-1)}) ,$$

there holds

$$z^{(k)^T} M\, z^{(j+1)} = \omega_{j+1}\, z^{(k)^T} N\, z^{(j)} \; . \qquad (1.12)$$

Thus, since from (1.8) the l.h.s. of (1.12) is zero, we have for $j \leq k-2$

$$z^{(j)^T} N\, z^{(k)} = 0 \; , \qquad (1.13)$$

which implies

$$z^{(j)^T} M\, z^{(k+1)} = 0 \qquad \text{for} \quad j \leq k-2 \; .$$

The desired result (1.7) then follows by induction.

The generalized CG method for the splitting $M = (A + A^T)/2$ is summarized as follows:

Algorithm

Let $x^{(0)}$ be a given vector and arbitrarily define $x^{(-1)}$. For $k = 0,1,\ldots$

(1) Solve $Mz^{(k)} = r^{(k)}$, where $r^{(k)} = b - Ax^{(k)}$.

(2) Compute

$$\omega_{k+1} = \left(1 + \frac{z^{(k)^T} M\, z^{(k)}}{z^{(k-1)^T} M\, z^{(k-1)}} \, \frac{1}{\omega_k} \right)^{-1} , \qquad k \geq 1$$

$$\omega_1 = 1 \; .$$

(3) Compute

$$x^{(k+1)} = x^{(k-1)} + \omega_{k+1}(z^{(k)} + x^{(k)} - x^{(k-1)}) \; .$$

In the computation of ω_{k+1}, one need not recompute $Mz^{(k)}$ since $r^{(k)}$ can be saved from step (1).

A simple induction argument shows that for all k, there holds

$$0 < \omega_{k+1} \leq 1 \; ,$$

unlike the case $N = N^T$, for which $\omega_{k+1} \geq 1$.

Note that since $z^{(p)^T} Mz^{(q)} = 0$ for $p \neq q$ and since by (1.13), $z^{(p)^T} Nz^{(q)} = 0$ for $|p-q| \neq 1$, there holds

$$z^{(p)^T} Az^{(q)} = 0 \qquad \text{for} \quad |p-q| > 1 \; .$$

Remarks concerning alternative forms of the generalized CG algorithm, which can be more efficient for actual computation, can be found in [3].

The calculated vectors $\{z^{(k)}\}_{k=0}^{n-1}$ will not generally be M-orthogonal in practice because of roundoff errors. One might consider forcing the newly calculated vectors to be M orthogonal by a procedure such as Gram-Schmidt. However, this would require the storage of all previously obtained vectors.

Our basic approach is to permit the gradual loss of orthogonality and with it the finite termination property of CG. We consider primarily the iterative aspects of the algorithm. In fact for solving large sparse systems arising from the discretization of elliptic partial differential equations,the application of principal interest for us and for which the generalized CG method seems particularly effective, convergence to desired accuracy often occurs within a number of iterations small compared with n.

2. Some Properties of the Method

In [3], there are presented some optimality properties, convergence properties, and eigenvalue relationships for the case in which A is symmetric. We discuss in this section related matters for the case in which M is symmetric and positive-definite and N is skew-symmetric.

2.1. From (1.6) with $\alpha_k = 1$ we obtain

$$z^{(k+1)} = z^{(k-1)} - \omega_{k+1}(-M^{-1}Nz^{(k)} + z^{(k-1)}) = (1-\omega_{k+1})\, z^{(k-1)} + \omega_{k+1}M^{-1}Nz^{(k)}, \quad (2.1)$$

which may be viewed as a relaxation of an iteration with iteration matrix

$$L = M^{-1}N \ .$$

We note that L is similar to a skew-symmetric matrix and hence that all the eigenvalues of L are either pure imaginary and occur in conjugate pairs, or are zero.

The eigenvalues of L can be determined directly from the generalized CG method in the same manner as for the symmetric case. We write (2.1) as

$$Lz^{(k)} = \left(1 - \frac{1}{\omega_{k+1}}\right) z^{(k-1)} + \frac{1}{\omega_{k+1}} z^{(k+1)}$$

or

$$L[z^{(0)}, z^{(1)}, \dots, z^{(n-1)}] =$$

$$= [z^{(0)}, z^{(1)}, \dots, z^{(n-1)})] \begin{bmatrix} 0 & 1-\frac{1}{\omega_2} & & & & \\ 1 & 0 & 1-\frac{1}{\omega_3} & & \bigcirc & \\ & \frac{1}{\omega_2} & 0 & \cdot & & \\ & & \frac{1}{\omega_3} & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & \bigcirc & & & \cdot & \cdot & 1-\frac{1}{\omega_n} \\ & & & & & \frac{1}{\omega_{n-1}} & 0 \end{bmatrix}$$

In matrix notation, the above equation can be written as

$$LZ = ZJ \ .$$

Assuming the columns of Z are linearily independent, it follows that

$$J = Z^{-1}LZ \ .$$

It can be shown that the kth principal minor of J yields very good estimates of the extreme eigenvalues of L, even in the presence of rounding errors. Note that although the matrix J is not skew-symmetric it is diagonally similar to such a matrix.

2.2 As in §2 of [3], define

$$K = I - M^{-1}N = I - L \ .$$

Then we have, as for the symmetric case,

$$z^{(k)} = [I - KP_{k-1}(K)]z^{(0)},$$

where

$$P_{k-1}(K) = \sum_{j=0}^{k-1} \beta_j^{(k-1)} K^j$$

is a polynomial in K of degree k-1. Correspondingly, we have

$$x^{(k)} = x^{(0)} + P_{k-1}(K)\ z^{(0)} \ .$$

As for the symmetric case, we define the weighted error function

$$E(x^{(k)}) = \frac{1}{2}\ e^{(k)^T} (M-N)\ e^{(k)} \ , \qquad (2.2)$$

where

$$e^{(k)} = x - x^{(k)} \ .$$

For the present case, (2.2) becomes

$$E(x^{(k)}) = \frac{1}{2}\ e^{(k)^T} Me^{(k)} \ .$$

Assuming that $(M-N)$ is nonsingular, we obtain, using

$$z^{(0)} = Ke^{(0)}$$

and

$$e^{(k)} = [I - KP_{k-1}(K)]\ e^{(0)} \ ,$$

the expression

$$E(x^{(k)}) = \frac{1}{2}\, e^{(0)^T}[I - KP_{k-1}(K)]^T M[I - KP_{k-1}(K)]\, e^{(0)} . \qquad (2.3)$$

The result for the symmetric case, that the polynomials $P_{k-1}(K)$ generated by CG minimize $E(x^{(k)})$ over the choice of all polynomials of degree k-1, does not hold here in general. Widlund [7] has shown, however, that there does hold

$$E(x^{(k)}) \leq \max_j (1 + \lambda_j^2)\, E(y) \qquad (2.4)$$

for any y of the form

$$y = x^{(0)} + S_{k-1}(K)\, z^{(0)} ,$$

where $S_{k-1}(K)$ is a polynomial in K of degree k-1. Here $i\lambda_j$, $j = 1,2,\ldots,n$, are the eigenvalues of L.

We remark that, as for the symmetric case, the generalized CG method converges in only p steps if K has only $p < n$ distinct eigenvalues. This same result holds also if K has a larger number of distinct eigenvalues but $e^{(0)}$ lies in a subspace generated by the eigenvectors associated with only p of these eigenvalues.

2.3. Let us consider the polynomials $S_{k-1}(K)$ generated by the Richardson second order method, for which $\omega_1 = 1$ and $\omega_{k+1} \equiv \omega$, a fixed parameter, for $k \geq 1$. For this case, (1.4) with $\alpha_k \equiv 1$ becomes

$$x^{(k+1)} = x^{(k-1)} + \omega(z^{(k)} + x^{(k)} - x^{(k-1)}) , \qquad k \geq 1 ,$$

and we have

$$e^{(k)} = [I - KS_{k-1}(K)]\, e^{(0)} \equiv T_{k,\omega}(L)\, e^{(0)} .$$

We seek a value of ω for which the spectral radius of $T_{k,\omega}(L)$ is a minimum.

Denote by $\rho(X)$ the spectral radius of a matrix X. By using an argument similar to that given in [4, pp. 18-24], it can be shown that for

$$\hat{\omega} = \frac{2}{1 + \sqrt{1 + \rho^2(L)}}$$

there holds

$$\rho(T_{k,\omega}(L)) \geq \rho(T_{k,\hat{\omega}}(L)) ,$$

where

$$\rho(T_{k,\hat{\omega}}(L)) = \theta^k \left(1 + \frac{1-\theta^2}{1+\theta^2}\, k \right) \qquad (2.5)$$

and

$$\theta = \frac{\rho(L)}{1 + \sqrt{1 + \rho^2(L)}} = \sqrt{(1 - \hat{\omega})} \quad . \tag{2.6}$$

To carry out the Richardson second order method we would need to have an estimate of $\rho(L)$. It is interesting to note that here also $0 < \omega \leq 1$. As for CG, underrelaxation is preferred for the case of skew-symmetric N.

2.4. One can use for y in (2.4) the optimal kth Richardson second-order iterate to obtain an asymptotic error estimate for the generalized CG method. Doing so yields, with the use of (2.5) and (2.6),

$$E(x^{(k)}) \leq C\theta^{2k} \left[1 + \frac{1 - \theta^2}{1 + \theta^2} k \right]^2 E(x^{(0)}) \ ,$$

where C is a constant independent of k.

3. An Example

To illustrate the method, we give here a simple example for which one can easily estimate the spectral radius of L. Consider the problem

$$\begin{aligned} -\Delta u + \sigma u_x &= f(x,y) \qquad & (x,y) &\in R \\ u &= g(x,y) \qquad & (x,y) &\in \partial R \ , \end{aligned}$$

where σ is a constant and R is the unit square $0 < x,y < 1$. We discretize on a uniform mesh of width h, using for Δ the standard five-point approximation Δ_h and for u_x at the point i,j the approximation $(U_{i+1,j} - U_{i-1,j})/(2h)$, where U_{ij} corresponds to $u(x,y)$ at $x = ih$, $y = jh$.

We consider solving the discrete problem by the algorithm of §1, for which

$$M = -\Delta_h$$

and

$$N = \begin{bmatrix} D & & & & O \\ & D & & & \\ & & \cdot & & \\ & & & \cdot & \\ O & & & & D \end{bmatrix} \ , \text{ where } D = \frac{-\sigma}{2h} \begin{bmatrix} 0 & 1 & & & & O \\ -1 & 0 & 1 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & 1 \\ O & & & & -1 & 0 \end{bmatrix} \quad .$$

A fast direct method (cf. [1]) can be used in this case for the solution of the system of equations $Mz^{(k)} = r^{(k)}$. (Of course, a fast direct method could be used, without iteration, to solve the entire problem for this simple example.)

To estimate the rate of convergence, we wish to determine the extremal eigenvalues of $L = M^{-1}N$, that is

$$N\varphi = i\lambda M\varphi \ . \tag{3.1}$$

For the corresponding differential operators, the equivalent eigenproblem is

$$\begin{aligned} \sigma\varphi_x &= i\lambda(\varphi_{xx} + \varphi_{yy}) && (x,y) \in R \\ \varphi &= 0 && (x,y) \in \partial R \ , \end{aligned} \tag{3.2}$$

for which one readily finds, by separation of variables, the eigenvalues to be

$$\lambda_{j,\ell} = \pm \frac{\sigma}{2\pi\sqrt{j^2 + \ell^2}} \ , \qquad j = 1, 2, \ldots; \quad \ell = 1, 2, \ldots \ .$$

The first eigenvalues $\lambda_{1,1}$ provide the uniform estimate for the spectral radius $\rho(L)$,

$$\rho(L) = |\lambda|_{\max} \cong \frac{\sqrt{2}}{4\pi}|\sigma|, \tag{3.3}$$

for which

$$\theta \cong \frac{\sqrt{2}|\sigma|}{4\pi}\left[1 + \sqrt{1 + \frac{\sigma^2}{8\pi^2}}\right]^{-1} .$$

Direct computation of the eigenvalues of (3.1), which is somewhat more cumbersome than for (3.2), shows (3.3) to be good asymptotically to within $O(h^2)$ as $h \to 0$.

We remark that for the symmetric problem with σu_x replaced by σu, and the splitting $M = -\Delta_h$ and $N = -\sigma I$, the estimate corresponding to (3.3) is [2] $|\lambda|_{\max} \cong |\sigma|/(2\pi^2)$. Numerical experiments illustrating the behavior of the modified CG method on related examples can be found in [7].

10

The possibility of using CG on nonsymmetric matrices in the manner presented here first occurred to us while listening to a presentation by T. Manteuffel of his dissertation research [6]. We wish to thank O. Widlund for making available to us his results to appear in [7] and to thank both O. Widlund and I. Karasalo for their helpful comments. This work was supported in part by the Energy Research and Development Administration and by the National Science Foundation.

REFERENCES

[1] B. L. Buzbee, G. H. Golub, and C.W. Nielson, "On direct methods for solving Poisson's equation," SIAM J. Numer. Anal. 7 (1970), pp. 627-656.

[2] P. Concus and G.H. Golub, "Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations," SIAM J. Numer. Anal. 10 (1973), pp. 1103-1120.

[3] P. Concus, G.H. Golub, and D.P. O'Leary, "A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations," Proc. Symposium on Sparse Matrix Computations, Argonne National Lab., Sept. 1975, Academic Press (to appear).

[4] G. H. Golub, "The use of Chebyshev matrix polynomials in the iterative solution of linear equations compared with the method of successive over-relaxation," Ph.D. Thesis, University of Illinois. 1959.

[5] G.H. Golub and R.S. Varga, "Chebyshev semi-iterative methods, successive over-relaxation iterative methods, and second order Richardson iterative methods," Numer. Math. 3 (1961), pp. 147-168.

[6] T. A. Manteuffel,"An iterative method for solving nonsymmetric linear systems with dynamic estimation of parameters," Ph.D. Thesis, University of Illinois, 1975.

[7] O. Widlund, Tech. Rept., Courant Institute, New York University (to appear).

8

A GENERALIZED CONJUGATE GRADIENT METHOD FOR THE NUMERICAL SOLUTION OF ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS (WITH PAUL CONCUS AND DIANNE P. O'LEARY)

in Proceedings of Sparse Matrix Conference,
Sparse Matrix Computation, J. R. Bunch and D. J. Rose, eds.,
Academic Press, Inc., 309–332, (1976).

Reprinted in Studies in Numerical Analysis,
G. H. Golub, ed., MAA Studies in Mathematics, vol. 24 178–198 (1984).

Although Gene Golub and I first met in 1960, when he came to California, and have been friends since, it was only in the early 1970s that we discovered that our scientific interests had overlapped. The catalyst was the appearance of fast Poisson solvers. I was interested in using them (or related fast Helmholtz solvers) to develop efficient methods for obtaining numerical solutions of certain nonseparable elliptic partial differential equations, and Gene Golub, from a numerical linear algebra point of view, was interested in iterative methods that might exploit such solvers. Shortly after our first joint results, which utilized Chebyshev acceleration, the conjugate gradient method was beginning to receive fresh interest as an iterative method. Our attention turned toward using that method as the convergence accelerator. Such a use was much in the air then and was being looked at by others for particular underlying iteration splittings. At about the same time Dianne O'Leary appeared at Stanford as a graduate student, and she was becoming interested in these subjects also. The following paper ensued. We were very enthusiastic over the results, but as sometimes happens when naming offspring, we called the method by a name that didn't stick. What is labeled here as the generalized conjugate gradient method is designated more popularly, and more descriptively, as the preconditioned conjugate gradient method.

Paul Concus
Berkeley, California, USA

Whenever I feel impatient with a graduate student, I think back to this project. Sometimes students need to hear the question explained multiple times. Sometimes they are unable to separate the wheat from the chaff in their work. And sometimes the published version bears little relation to their thesis writeup.

All of this applies to this work, done when I was a graduate student in my early twenties. Thankfully, Gene and Paul were exceptionally patient mentors.

Iterative methods for solving linear systems has been a recurring theme in my work, and I always return to it with new appreciation for the revolutionary paper of Hestenes and Stiefel upon which the subject of Krylov subspace methods was built, and for the privilege of having Gene as my PhD advisor.

Dianne O'Leary
College Park, Maryland, USA

A GENERALIZED CONJUGATE GRADIENT METHOD FOR THE NUMERICAL SOLUTION OF ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS*

Paul Concus, Gene H. Golub, and Dianne P. O'Leary

Abstract

We consider a generalized conjugate gradient method for solving sparse, symmetric, positive-definite systems of linear equations. principally those arising from the discretization of boundary value problems for elliptic partial differential equations. The method is based on splitting off from the original coefficient matrix a symmetric, positive-definite one that corresponds to a more easily solvable system of equations, and then accelerating the associated iteration using conjugate gradients. Optimality and convergence

*Note added in reprinting: *The generalized conjugate gradient method* discussed here has been become known more popularly as the *preconditioned conjugate gradient method*.

properties are presented, and the relation to other methods is discussed. Several splittings for which the method seems particularly effective are also discussed and, for some, numerical examples are given.

0. INTRODUCTION

In 1952, Hestenes and Stiefel [0] proposed the conjugate gradient method (CG) for solving the system of linear algebraic equations

$$A\mathbf{x} = \mathbf{b},$$

where A is an $n \times n$, symmetric, positive-definite matrix. This elegant method has as one of its important properties that in the absence of round-off error the solution is obtained in at most n iteration steps. Furthermore, the entire matrix A need not be stored as an array in memory; at each stage of the iteration it is necessary to compute only the product $A\mathbf{z}$ for a given vector $\mathbf{z}$.

Unfortunately the initial interest and excitement in CG was dissipated, because in practice the numerical properties of the algorithm differed from the theoretical ones; viz., even for small systems of equations ($n \leqslant 100$) the algorithm did not necessarily terminate in n iterations. In addition, for large systems of equations arising from the discretization of two-dimensional elliptic partial differential equations, competing methods such as successive overrelaxation (SOR) required only $O(\sqrt{n})$ iterations to achieve a prescribed accuracy [1]. It is interesting to note that in the proceedings of the Conference on Sparse Matrices and Their Applications held in 1971 [2] there is hardly any mention of the CG method.

In 1970, Reid [3] renewed interest in CG by giving evidence that the method could be used in a highly effective manner as an iterative procedure for solving large sparse systems of linear equations. Since then a number of authors have described the use of CG for solving a variety of problems (cf. [4], [5], [6], [7], [8]). Curiously enough, although CG was generally discarded during the sixties as a useful method for solving linear equations, except in conjunction with other methods [9], there was considerable interest in it for solving nonlinear equations (cf. [10]).

The conjugate gradient method has a number of attractive properties when used as an iterative method:

(i) It does not require an estimation of parameters.
(ii) It takes advantage of the distribution of the eigenvalues of the iteration operator.
(iii) It requires fewer restrictions on the matrix A for optimal behavior than do such methods as SOR.

Our basic view is that CG is most effective when used as an iteration acceleration technique.

In this paper, we derive and show how to apply a generalization of the CG method and illustrate it with numerical examples. Based on our investigations, we feel that the generalized CG method has the potential for widespread application in the numerical solution of boundary value problems for elliptic partial differential equations. Additional experience should further indicate how best to take full advantage of the method's inherent possibilities.

1. DERIVATION OF THE METHOD

Consider the system of equations

$$A\mathbf{x} = \mathbf{b}, \tag{1.1}$$

where A is an $n \times n$, symmetric, positive-definite matrix and $\mathbf{b}$ is a given vector. It is frequently desirable to rewrite (1.1) as

$$M\mathbf{x} = N\mathbf{x} + \mathbf{c}, \tag{1.2}$$

where M is positive-definite and symmetric and N is symmetric. In section 4 we describe several decompositions of the form (1.2). We are interested in those situations for which it is a much simpler computational task to solve the system

$$M\mathbf{z} = \mathbf{d} \tag{1.3}$$

than it is to solve (1.1).

We consider an iteration of the form

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k-1)} + \omega_{k+1}\left(\alpha_k \mathbf{z}^{(k)} + \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right), \tag{1.4}$$

where

$$M\mathbf{z}^{(k)} = \mathbf{c} - (M-N)\mathbf{x}^{(k)}. \tag{1.5}$$

Many iterative methods can be described by (1.4); e.g. the Chebyshev semi-iterative method and the Richardson second-order method (cf. [11]). The generalized CG method is also of this form.

For the Richardson or Chebyshev methods, the optimal parameters (ω_{k+1}, α_k) are given as simple, easy-to-compute functions of the smallest and largest eigenvalues of the iteration matrix $M^{-1}N$ [11]; thus good estimates of these eigenvalues are required for the methods to be efficient. The methods do not take into account the values of any of the interior eigenvalues of the iteration matrix.

The CG method, on the other hand, needs no *a priori* information on the extremal eigenvalues and does take into account the interior ones, but at a cost of increased computational requirements for evaluating ω_{k+1} and α_k. In section 3, we describe a technique to provide directly from the CG method good estimates for the extreme eigenvalues of the iteration matrix.

From equations (1.4) and (1.5), we obtain the relation

$$M\mathbf{z}^{(k+1)} = M\mathbf{z}^{(k-1)} - \omega_{k+1}\left(\alpha_k(M-N)\mathbf{z}^{(k)} + M(\mathbf{z}^{(k-1)} - \mathbf{z}^{(k)})\right). \tag{1.6}$$

For the generalized CG method the parameters $\{\alpha_k, \omega_{k+1}\}$ are computed so that

$$\mathbf{z}^{(p)^T} M \mathbf{z}^{(q)} = 0 \qquad \text{for } p \neq q \text{ and } p, q = 0, 1, \ldots, n-1. \tag{1.7}$$

Since M is $n \times n$ positive-definite, (1.7) implies that for some $k \leqslant n$

$$\mathbf{z}^{(k)} = \mathbf{0}$$

and, hence,

$$\mathbf{x}^{(k)} = \mathbf{x}. \tag{1.8}$$

That is, the iteration converges in no more than n steps.

We derive the above result by induction. Assume

$$\mathbf{z}^{(p)^T} M \mathbf{z}^{(q)} = 0 \quad \text{for } p \neq q \quad \text{and} \quad p, q = 0, 1, \ldots, k. \tag{1.9}$$

Then if

$$\alpha_k = \mathbf{z}^{(k)^T} M \mathbf{z}^{(k)} / \mathbf{z}^{(k)^T} (M - N) \mathbf{z}^{(k)}, \tag{1.10}$$

there holds

$$\mathbf{z}^{(k)^T} M \mathbf{z}^{(k+1)} = 0,$$

and if

$$\omega_{k+1} = \left(1 - \alpha_k \frac{\mathbf{z}^{(k-1)^T} N \mathbf{z}^{(k)}}{\mathbf{z}^{(k-1)^T} M \mathbf{z}^{(k-1)}} \right)^{-1} \tag{1.11}$$

then

$$\mathbf{z}^{(k-1)^T} M \mathbf{z}^{(k+1)} = 0.$$

We can simplify the above expression for ω_{k+1} as follows. From (1.6) we obtain

$$M\mathbf{z}^{(k)} = M\mathbf{z}^{(k-2)} - \omega_k \left(\alpha_{k-1} (M - N) \mathbf{z}^{(k-1)} + M (\mathbf{z}^{(k-2)} - \mathbf{z}^{(k-1)}) \right),$$

and then from (1.9)

$$\mathbf{z}^{(k)^T} N \mathbf{z}^{(k-1)} = \mathbf{z}^{(k)^T} M \mathbf{z}^{(k)} / (\omega_k \alpha_{k-1}).$$

Since

$$\mathbf{z}^{(k-1)^T} N \mathbf{z}^{(k)} = \mathbf{z}^{(k)^T} N \mathbf{z}^{(k-1)},$$

it follows

$$\omega_{k+1}=\left(1-\frac{\alpha_k}{\alpha_{k-1}}\frac{\mathbf{z}^{(k)^T}M\mathbf{z}^{(k)}}{\mathbf{z}^{(k-1)^T}M\mathbf{z}^{(k-1)}}\frac{1}{\omega_k}\right)^{-1}.$$

From (1.6), for $j<k-1$

$$\mathbf{z}^{(j)^T}M\mathbf{z}^{(k+1)}=\alpha_k\omega_{k+1}\mathbf{z}^{(j)^T}N\mathbf{z}^{(k)}.$$

But

$$M\mathbf{z}^{(j+1)}=M\mathbf{z}^{(j-1)}-\omega_{j+1}\big(\alpha_j(M-N)\mathbf{z}^{(j)}+M(\mathbf{z}^{(j-1)}-\mathbf{z}^{(j)})\big),$$

so that

$$\mathbf{z}^{(k)^T}N\mathbf{z}^{(j)}=0.$$

Thus, since $N=N^T$,

$$\mathbf{z}^{(j)^T}M\mathbf{z}^{(k+1)}=0 \qquad \text{for} \quad j<k-1.$$

Hence by induction we obtain (1.7) and (1.8).

The *generalized CG method* is summarized as follows.

ALGORITHM.

Let $\mathbf{x}^{(0)}$ be a given vector and arbitrarily define $\mathbf{x}^{(-1)}$. For $k=0,1,\ldots$

(1) Solve $M\mathbf{z}^{(k)}=\mathbf{c}-(M-N)\mathbf{x}^{(k)}$.

(2) Compute

$$\alpha_k=\frac{\mathbf{z}^{(k)^T}M\mathbf{z}^{(k)}}{\mathbf{z}^{(k)^T}(M-N)\mathbf{z}^{(k)}},$$

$$\omega_{k+1}=\left(1-\frac{\alpha_k}{\alpha_{k-1}}\frac{\mathbf{z}^{(k)^T}M\mathbf{z}^{(k)}}{\mathbf{z}^{(k-1)^T}M\mathbf{z}^{(k-1)}}\cdot\frac{1}{\omega_k}\right)^{-1} \qquad (k\geqslant 1),$$

$$\omega_1=1.$$

(3) Compute

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k-1)} + \omega_{k+1}\left(\alpha_k \mathbf{z}^{(k)} + \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right).$$

Note that the algorithm can be viewed as an acceleration of the underlying first-order iteration ($\omega_{k+1} \equiv 1$), $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{z}^{(k)}$. As with other higher order methods, the storage requirements of the algorithm are greater than those of the underlying first-order iteration being accelerated.

The algorithm presented above is given primarily for expository purposes. For actual computation, the following equivalent form can be more efficient in terms of storage [3].

ALGORITHM (alternative form).

Let $\mathbf{x}^{(0)}$ be a given vector and arbitrarily define $\mathbf{p}^{(-1)}$. For $k = 0, 1, \ldots$

(1) Solve $M\mathbf{z}^{(k)} = \mathbf{c} - (M - N)\mathbf{x}^{(k)}$.

(2) Compute

$$b_k = \frac{\mathbf{z}^{(k)^T} M \mathbf{z}^{(k)}}{\mathbf{z}^{(k-1)^T} M \mathbf{z}^{(k-1)}}, \qquad k \geqslant 1,$$

$$b_0 = 0,$$

$$\mathbf{p}^{(k)} = \mathbf{z}^{(k)} + b_k \mathbf{p}^{(k-1)}.$$

(3) Compute

$$a_k = \frac{\mathbf{z}^{(k)^T} M \mathbf{z}^{(k)}}{\mathbf{p}^{(k)^T}(M - N)\mathbf{p}^{(k)}},$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + a_k \mathbf{p}^{(k)}.$$

In the computation of the numerators of a_k and b_k one need not recompute $M\mathbf{z}^{(k)}$, since it can be saved from step (1). Also, instead of computing the right-hand side of step (1) explicitly at each iteration, it is often advantageous to compute it recursively from

$$[\mathbf{c}-(M-N)\mathbf{x}^{(k+1)}]=[\mathbf{c}-(M-N)\mathbf{x}^{(k)}]-a_k(M-N)\mathbf{p}^{(k)}, \tag{1.12}$$

which equation is obtained from step (3). The vector $(M-N)\mathbf{p}^{(k)}$ appearing in (1.12) may be saved from the computation of a_k. Similar remarks hold for the algorithm in its first form as well. There is evidence that the use of (1.12) is no less accurate than use of the explicit computation (see [18], [3] for particular examples).

The calculated vectors $\{\mathbf{z}^{(k)}\}_{k=0}^{n-1}$ will not generally be M-orthogonal in practice because of rounding errors. One might consider forcing the newly calculated vectors to be M-orthogonal by a procedure such as Gram-Schmidt. However, this would require the storage of all the previously obtained vectors.

Our basic approach is to permit the gradual loss of orthogonality and with it the finite termination property of CG. We consider primarily the iterative aspects of the algorithm. In fact, for solving large sparse systems arising from the discretization of elliptic partial differential equations, the application of principal interest for us and for which the generalized CG method seems particularly effective, convergence to desired accuracy often occurs within a number of iterations small compared with n.

2. OPTIMALITY PROPERTIES

From (1.6), we obtain

$$\mathbf{z}^{(k+1)}=\mathbf{z}^{(k-1)}-\omega_{k+1}\left(\alpha_k(I-M^{-1}N)\mathbf{z}^{(k)}+\mathbf{z}^{(k-1)}-\mathbf{z}^{(k)}\right). \tag{2.1}$$

Define

$$K=I-M^{-1}N. \tag{2.2}$$

We have $\mathbf{z}^{(1)} = (I - \alpha_0 K)\mathbf{z}^{(0)}$, and there follows by induction that

$$\mathbf{z}^{(l+1)} = \left[I - KP_l(K)\right]\mathbf{z}^{(0)} \tag{2.3}$$

where

$$P_l(K) = \sum_{j=0}^{l} \beta_j^{(l)} K^j. \tag{2.4}$$

We denote

$$p_l(\lambda) = \sum_{j=0}^{l} \beta_j^{(l)} \lambda^j \tag{2.5}$$

and from (2.1) we have for $k = 2,3,\ldots,l$

$$p_k(\lambda) = \omega_{k+1}(1-\alpha_k\lambda)p_{k-1}(\lambda) - (\omega_{k+1}-1)p_{k-2}(\lambda) + \alpha_k\omega_{k+1},$$

and

$$p_0(\lambda) = \alpha_0, \qquad p_1(\lambda) = \omega_2(\alpha_0 + \alpha_1 - \alpha_0\alpha_1\lambda).$$

The coefficients $\{\beta_j^{(l)}\}_{j=0}^{l}$ can be generated directly. From (2.3) and the relation $\mathbf{z}^{(l+1)} = \mathbf{z}^{(0)} + K(\mathbf{x}^{(l+1)} - \mathbf{x}^{(0)})$, there follows

$$\mathbf{x}^{(l+1)} = \mathbf{x}^{(0)} + P_l(K)\mathbf{z}^{(0)}.$$

Then if

$$Z = [\mathbf{z}^{(0)}, K\mathbf{z}^{(0)}, \ldots, K^l\mathbf{z}^{(0)}], \tag{2.6}$$

$$\mathbf{x}^{(l+1)} = \mathbf{x}^{(0)} + Z\boldsymbol{\beta}^{(l)}. \tag{2.7}$$

Consider the *weighted error function*:

$$E(\mathbf{x}^{(l+1)}) = \tfrac{1}{2}(\mathbf{x} - \mathbf{x}^{(l+1)})^T(M-N)(\mathbf{x} - \mathbf{x}^{(l+1)}). \tag{2.8}$$

Assuming that $(M-N)$ is nonsingular, we obtain, using

$$\mathbf{z}^{(0)} = K(\mathbf{x} - \mathbf{x}^{(0)}),$$

the relations

$$E(\mathbf{x}^{(l+1)}) = \tfrac{1}{2}\mathbf{z}^{(0)T}(I - KP_l(K))^T M(M-N)^{-1}M(I - KP_l(K))\mathbf{z}^{(0)}$$
$$= \tfrac{1}{2}\mathbf{e}^{(0)T}(I - KP_l(K))^T(M-N)(I - KP_l(K))\mathbf{e}^{(0)}, \tag{2.9}$$

where

$$\mathbf{e}^{(0)} = \mathbf{x} - \mathbf{x}^{(0)}.$$

Equivalently, we can use (2.7) and re-write (2.8) as

$$E(\mathbf{x}^{(l+1)}) = \tfrac{1}{2}(K^{-1}\mathbf{z}^{(0)} - Z\boldsymbol{\beta}^{(l)})^T(M-N)(K^{-1}\mathbf{z}^{(0)} - Z\boldsymbol{\beta}^{(l)}). \tag{2.10}$$

The quantity $E(\mathbf{x}^{(l+1)})$ is minimized when we choose $\boldsymbol{\beta}^{(l)}$ so that

$$G\boldsymbol{\beta}^{(l)} = \mathbf{h},$$

where

$$G = Z^T(M-N)Z, \qquad \mathbf{h} = Z^T M\mathbf{z}^{(0)}.$$

Let

$$\kappa = \lambda_{\max}(K)/\lambda_{\min}(K).$$

Then using arguments similar to those given in [12], the following can be shown:

(A) $$\frac{E(\mathbf{x}^{(l+1)})}{E(\mathbf{x}^{(0)})} \leqslant 4\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2(l+1)}. \tag{2.11}$$

(B) The generalized CG method is optimal in the class of all algorithms for which

$$\mathbf{x}^{(l+1)} = \mathbf{x}^{(0)} + P_l(K)\mathbf{z}^{(0)}.$$

That is, the approximation $\mathbf{x}^{(l+1)}$ generated by the generalized

CG method satisfies

$$E(\mathbf{x}^{(l+1)}) = \min_{P_l} \tfrac{1}{2}\mathbf{e}^{(0)T}(I - KP_l(K))^T \times (M-N)(I-KP_l(K))\mathbf{e}^{(0)},$$

where the minimum is taken with respect to all polynomials P_l of degree l.

Recall that we have assumed that M and $(M-N)$ are positive definite and symmetric. Thus the eigenvalues of $K=(I-M^{-1}N)$ are all real and K is similar to a diagonal matrix. Hence, if K has only $p<n$ distinct eigenvalues, there exists a matrix polynomial $Q_p(K)$ so that

$$Q_p(K)=0.$$

In this case, $E(\mathbf{x}^{(p)})=0$ and hence

$$\mathbf{x}^{(p)}=\mathbf{x},$$

so that the iteration converges in only p steps. The same result also holds if K has a larger number of distinct eigenvalues but $\mathbf{e}^{(0)}$ lies in a subspace generated by the eigenvectors associated with only p of these eigenvalues.

We remark also that statement (B) implies CG is optimal for the particular eigenvector mix of the initial error $\mathbf{e}^{(0)}$, taking into account interior as well as extremal eigenvalues. As will be discussed in the next section, the extremal eigenvalues are approximated especially well as CG proceeds, the iteration then behaving as if the corresponding vectors are not present. Thus the error estimate (2.11), which is based on the extremal eigenvalues, tends to be pessimistic asymptotically. One often observes, in practice (see section 5), a superlinear rate of convergence for the CG method.

3. EIGENVALUE COMPUTATIONS

The CG method can be used in a very effective manner for computing the extreme eigenvalues of the matrix $K=I-M^{-1}N$.

We write (see (2.1))

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k-1)} - \omega_{k+1}\left(\alpha_k K\mathbf{z}^{(k)} + \mathbf{z}^{(k-1)} - \mathbf{z}^{(k)}\right) \qquad (3.1)$$

as

$$K\mathbf{z}^{(k)} = c_{k-1}\mathbf{z}^{(k-1)} + a_k\mathbf{z}^{(k)} + b_{k+1}\mathbf{z}^{(k+1)},$$

or

$$K\left[\mathbf{z}^{(0)},\mathbf{z}^{(1)},\ldots,\mathbf{z}^{(n-1)}\right]$$

$$= \left[\mathbf{z}^{(0)},\mathbf{z}^{(1)},\ldots,\mathbf{z}^{(n-1)}\right]\begin{bmatrix} a_0 & c_0 & & & \bigcirc \\ b_1 & a_1 & c_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-2} \\ \bigcirc & & & b_{n-1} & a_{n-1} \end{bmatrix}$$

thus defining a_k, b_k, and c_k. In matrix notation, the above equation can be written as

$$KZ = ZJ. \qquad (3.2)$$

Assuming that the columns of Z are linearly independent, there follows from (3.2) that

$$K = ZJZ^{-1},$$

hence the eigenvalues of K are equal to those of J. As pointed out in section 2, if K has repeated eigenvalues or if the vector $\mathbf{z}^{(0)}$ is deficient in the direction of some eigenvectors of K, iteration (3.1) will terminate in $k < n$ steps.

The process described by (3.1) is essentially the Lanczos algorithm [13]. It has been shown by Kaniel [14] and by Paige [15] that good estimates of the extreme eigenvalues of K often can be obtained from the truncated matrix

$$J_k = \begin{bmatrix} a_0 & c_0 & & \bigcirc \\ b_1 & a_1 & c_1 & \\ & \ddots & \ddots & \ddots \\ \bigcirc & & \ddots & c_{k-2} \\ & & b_{k-1} & a_{k-1} \end{bmatrix}$$

where k is considerably less than n. This result holds even in the presence of round-off error [16].

It was pointed out in section 1 that the equation describing the CG method is of the same form as that describing the Chebyshev semi-iterative method and Richardson second order method, but that a knowledge of the extreme eigenvalues of K is required for obtaining parameters for the latter two methods. Thus one could construct a polyalgorithm in which the CG method is used initially to obtain good approximations to the solution and to the extreme eigenvalues of K, after which the Chebyshev semi-iterative method, say, is used, thereby avoiding the additional work of repeatedly calculating CG parameters. This technique has been used in an effective manner by O'Leary [17].

4. CHOICE OF M

For the splitting $M=I$, $N=I-A$ one obtains the basic, unmodified CG algorithm, for which

$$\mathbf{z}^{(k)}=\mathbf{r}^{(k)}=\mathbf{b}-A\mathbf{x}^{(k)}$$

is simply the residual at the kth step. Since the rate of convergence of the generalized CG method, as given by the estimate (2.11), decreases with increasing

$$\kappa=\lambda_{\max}(K)/\lambda_{\min}(K),$$

it is desirable to choose a splitting for which κ is as small as possible. If $A=L+D+U$, where D consists of the diagonal elements of A and $L(U)$ is a strictly lower (upper) triangular matrix, then it is reasonable to consider the choice

$$M=D, \qquad N=-(L+U).$$

This M, which is equivalent to a rescaling of the problem, is one for which (1.3) can be solved very simply for $\mathbf{z}$. It has been shown by Forsythe and Straus [19] that if A is two-cyclic then among all diagonal matrices this choice of M will minimize κ.

In many cases, the matrix A can be written in the form

$$A = \left(\begin{array}{c|c} M_1 & F \\ \hline F^T & M_2 \end{array}\right), \tag{4.3}$$

where the systems

$$M_1 \mathbf{z}_1 = \mathbf{d}_1 \qquad \text{and} \qquad M_2 \mathbf{z}_2 = \mathbf{d}_2$$

are easy to solve, and for such matrices, it is convenient to choose

$$M = \left(\begin{array}{c|c} M_1 & 0 \\ \hline 0 & M_2 \end{array}\right) \qquad \text{and} \qquad N = \left(\begin{array}{c|c} 0 & -F \\ \hline -F^T & 0 \end{array}\right).$$

Using (4.3), we can write the system (1.1) in the form

$$M_1 \mathbf{x}_1 + F \mathbf{x}_2 = \mathbf{b}_1 \tag{4.4a}$$

$$F^T \mathbf{x}_1 + M_2 \mathbf{x}_2 = \mathbf{b}_2. \tag{4.4b}$$

Let the initial approximation for $\mathbf{x}_1$ be $\mathbf{x}_1^{(0)}$, and obtain $\mathbf{x}_2^{(0)}$ as the solution to (4.4b) so that

$$M_2 \mathbf{x}_2^{(0)} = \mathbf{b}_2 - F^T \mathbf{x}_1^{(0)}.$$

This implies that

$$\mathbf{z}_2^{(0)} = \mathbf{0},$$

and, hence, by (1.10)

$$\alpha_0 = 1,$$

and thus

$$\mathbf{x}^{(1)} = \begin{pmatrix} \mathbf{x}_1^{(0)} + \mathbf{z}_1^{(0)} \\ \mathbf{x}_2^{(0)} \end{pmatrix}.$$

A short calculation shows that $\mathbf{z}_1^{(1)}=\mathbf{0}$ and hence $\alpha_1=1$. Using (1.6), a simple inductive argument then yields that for $j=0,1,2,\ldots$

$$\alpha_j\equiv 1, \qquad \mathbf{z}_1^{(2j+1)}=\mathbf{0}, \qquad \mathbf{z}_2^{(2j)}=\mathbf{0}. \tag{4.5}$$

This result was first observed by Reid [8] for the case in which M_1 and M_2 are diagonal, i.e., in which the matrix A has "Property A" and is suitably ordered. Other cases for elliptic boundary value problems in which matrices of the form (4.3) arise will be discussed in section 5. For these cases convergence can be rapid because K has only a few distinct eigenvalues, even though κ is not especially small.

Various other splittings of the matrix A can occur quite naturally in the solution of elliptic partial differential equations. For example, if one wishes to solve

$$\begin{aligned} -\Delta u+\sigma(x,y)u&=f \qquad (x,y)\in R\\ u&=g \qquad (x,y)\in \partial R, \end{aligned}$$

where R is a rectangular region and $\Delta u=\partial^2u/\partial x^2+\partial^2u/\partial y^2$, it is convenient to choose M as the finite difference approximation to a separable operator, such as the Helmholtz operator $-\Delta+C$, for which fast direct methods can be used [23]. A numerical example for this case is discussed in section 5. If one wishes to solve a separable equation, but on a nonrectangular region S, then by extending the problem to one on a rectangle R in which S is embedded, M can be chosen as the discrete approximation to the separable operator on R, for which fast direct methods can be used. Such a technique provides an alternative to the related capacitance matrix method [25] for handling such problems. Forms of this method utilizing CG, but in a different manner than here, are described in [26] and [27].

Several authors [4], [20], [21] have used CG in combination with symmetric successive overrelaxation (SSOR). For this method the solution of the equation $M\mathbf{z}^{(k)}=\mathbf{c}-(M-N)\mathbf{x}^{(k)}$ reduces to the solution of

$$(D+\omega L)D^{-1}(D+\omega U)\mathbf{z}^{(k)}=\omega(2-\omega)\mathbf{r}^{(k)}$$

where D, L, and U are as described previously in this section (although D may be block diagonal), $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$, and ω is a parameter in the open interval (0,2). SSOR is particularly effective in combination with CG because of the distribution of the eigenvalues of K (cf. [22]).

Meijerink and van der Vorst [7] have proposed that the following factorization of A be used:

$$A = FF^T + E,$$

so that

$$M = FF^T, \qquad N = -E.$$

The matrix F is chosen with a sparsity pattern resembling that of A. This splitting appears to yield a matrix K with eigenvalues that also are favorably distributed for CG. A block form of this technique recently developed by Underwood [24] achieves a more accurate approximate factorization of A with less computer storage and about the same number of arithmetic operations per iteration.

Generally, in addition to the requirement that (1.5) be "easy" to solve, M should have the following features if the generalized CG algorithm is to be computationally efficient. For rapid convergence one seeks a splitting so that

(i) $M^{-1}N$ has small or nearly equal eigenvalues

or (ii) $M^{-1}N$ has small rank.

Often a choice for M satisfying these restrictions comes about naturally from the inherent features of a given problem.

5. NUMERICAL EXAMPLES

For the first example, we consider the test problem discussed in [23]

$$-\operatorname{div}(a(x,y)\nabla u) = f \qquad (x,y) \in R$$
$$u = g \qquad (x,y) \in \partial R,$$

where $a(x,y) = [1 + \frac{1}{2}(x^4 + y^4)]^2$ and R is the unit square $0 < x$,

$y < 1$. After a transformation the problem becomes

$$-\Delta w + \sigma(x, y)w = a^{-1/2}f \qquad (x, y) \in R$$

$$w = a^{1/2}g \qquad (x, y) \in \partial R, \qquad (5.1)$$

where $\sigma(x, y) = 6(x^2 + y^2)/a^{1/2}$. As in [23] we discretize (5.1) on a uniform mesh of width h, using for Δ the standard five-point approximation Δ_h (see Henrici's paper in this study for approximations to Δ), and we choose the splitting

$$M = A + N = -\Delta_h + CI$$

with $C = 0 = \sigma_{\min}$ or $C = 3 = \frac{1}{2}(\sigma_{\max} + \sigma_{\min})$.

In [23] Chebyshev acceleration was used, which requires an estimate of the ratio of the extremal eigenvalues of the iteration matrix. Here we use the modified CG algorithm of section 1. For an initial guess $\mathbf{W}^{(0)} \equiv \mathbf{0}$ and choice of f and g corresponding to the solution $w = 2[(x - 1/2)^2 + (y - 1/2)^2]$, the results are given in Table 1 for $h = 1/64$. The results obtained for $h = 1/32$ were essentially identical, as the iteration is basically independent of h for this problem (see [23]).

Note that the Chebyshev method is sensitive both to the value of C and to the accuracy of the eigenvalues from which the parameters are calculated. The parameters used for the middle column were based on Gerschgorin estimates from the Rayleigh quotient,

TABLE 1
Maximum error vs. iteration number for first example.

	Chebyshev (from [23])			CG	
	$C = 0$ exact	$C = 3$ approximate	$C = 3$ exact		
iteration	eigenvalues	eigenvalues	eigenvalues	$C = 0$	$C = 3$
1			1.6(−2)	4.5(−2)	1.6(−2)
2			7.4(−4)	2.6(−3)	6.7(−4)
3			1.1(−5)	3.0(−5)	1.0(−5)
4			2.7(−7)	5.7(−7)	1.1(−7)
5	2.4(−6)	1.1(−6)	4.3(−9)	5.1(−9)	8.2(−10)
6			1.2(−10)	4.4(−11)	5.7(−12)

which gave a ratio of largest to smallest eigenvalue about three times too large. The CG method appears to be less sensitive to the value of C. After several iterations CG begins to converge more rapidly than does the optimal Chebyshev method, which behavior is typical of the CG superlinear convergence property discussed in section 2. This example is one for which rapid convergence results because the eigenvalues of $M^{-1}N$ are small.

We give as the second example

$$\begin{aligned} -\Delta u &= f \qquad (x, y) \in T \\ u &= g \qquad (x, y) \in \partial T \end{aligned}$$

where T is the domain shown in Figure 1. For a uniform square mesh of width h, and $0 < l < (2h)^{-1}$ a whole number, so that all boundary segments are mesh lines, the coefficient matrix A for the standard five-point discretization and natural ordering has the form (4.3). M_1 and M_2 correspond to the mesh points in each of the two squares, T_1 and T_2, and F to the coupling between them. F has non-zero entries in only $p = 2l - 1$ of its rows.

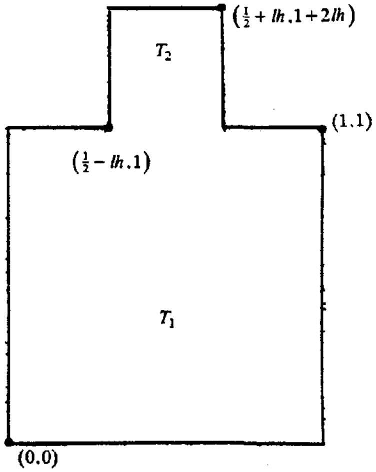


FIG. 1. T-shaped domain.

According to the discussion following (4.3) we choose

$$M=\left(\begin{array}{c|c} M_1 & 0 \\ \hline 0 & M_2 \end{array}\right)$$

and for initial approximation

$$\mathbf{U}^{(0)}=\left(\begin{array}{c} \mathbf{U}_1^{(0)} \\ \hline M_2^{-1}\left(\mathbf{b}_2-F^T\mathbf{U}_1^{(0)}\right) \end{array}\right).$$

Then for the generalized CG algorithm, there holds $\alpha_k \equiv 1$ and that $\mathbf{z}_1$ and $\mathbf{z}_2$ are alternately zero, thereby reducing computational and storage requirements. We use a fast direct Poisson solver for the systems involving M_1 and M_2.

The results for $\mathbf{U}_1^{(0)}$ uniformly distributed random numbers in $(0,2)$ and $f(x,y)$ and $g(x,y)$ such that $u=x^2+y^2$ is the solution are given in Table 2. Here the average error per point, the two norm of the error divided by the square root of the number of interior mesh points, is given for each of the test problems.

For this example, the eigenvalues of $M^{-1}N$ are not especially small in magnitude; however, since $M^{-1}N$ has rank of only $2p$,

TABLE 2
Average error per point vs. iteration number.

	Case I	Case II	Case III
h	1/32	1/64	1/64
l	4	4	8
p	7	7	15
iteration	ave. error/pt	ave. error/pt	ave. error/pt
1	8.58(−2)	3.70(−2)	1.08(−1)
2	7.05(−2)	3.13(−2)	9.82(−2)
3	1.30(−2)	6.66(−3)	4.94(−2)
4	3.35(−3)	2.53(−3)	1.80(−2)
5	2.71(−4)	6.03(−4)	4.28(−3)
10	2.65(−7)	5.13(−8)	7.35(−5)
15	1.14(−13)	5.60(−13)	4.71(−8)

convergence is obtained in only a moderate number of iterations. For Case I and Case II the last row represents full convergence to machine accuracy subject to rounding errors, as would be expected since $2p = 14$ for these cases.

We wish to thank Myron Stein of the Los Alamos Scientific Laboratory for his careful computer programming of the second test problem. This work was supported in part by the Energy Research and Development Administration, by the Hertz Foundation, and by the National Science Foundation.

REFERENCES

0. M. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Research NBS* **49** (1952), 409–436.*
1. D. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.
2. D. J. Rose and R. A. Willoughby (ed.), *Sparse Matrices and Their Applications*, Plenum Press, New York-London, 1972.
3. J. K. Reid, "On the method of conjugate gradients for the solution of large sparse systems of linear equations," Proc. Conference on *Large Sparse Sets of Linear Equations*, Academic Press, New York, 1971.
4. O. Axelsson, "On preconditioning and convergence acceleration in sparse matrix problems," Report CERN 74-10 of the CERN European Organization for Nuclear Research, Data Handling Division, Laboratory I, 8 May 1974.
5. R. Bartels and J. W. Daniel, "A conjugate gradient approach to nonlinear elliptic boundary value problems in irregular regions," Proc. Conf. on *Numerical Solution of Differential Equations*, Springer-Verlag, Berlin, 1974.
6. R. Chandra, S. C. Eisenstat, and M. H. Schultz, "Conjugate gradient methods for partial differential equations," *Advances in Computer Methods for Partial Differential Equations*, R. Vichnevtsky (ed), Publ. A.I. C.A.-1975.
7. J. A. Meijerink and H. A. van der Vorst, "Iterative solution of linear systems arising from discrete approximations to partial differential equations," Academisch Computer Centrum, Utrecht, The Netherlands, 1974.
8. J. Reid, "The use of conjugate gradients for systems of linear equations possessing 'Property A'," *SIAM J. Numer. Anal.* **9** (1972), 325–332.
9. E. L. Wachspress, "Extended applications of alternating direction implicit iteration model problem theory," *SIAM J.* **11** (1963), 994–1016.

*Note added in reprinting: See also M. Hestenes, "The conjugate-gradient method for solving linear systems," Proc. Symposia in Applied Math VI, McGraw-Hill, New York, 1956, 83–102.

10. R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *Comput. J.*, 7 (1964), 145–154.
11. G. H. Golub and R. S. Varga, "Chebyshev semi-iterative methods, successive over-relaxation iterative methods, and second order Richardson iterative methods," *Numer. Math.* **3** (1961), 147–168.
12. D. K. Faddeev and V. N. Faddeeva, *Computational Methods of Linear Algebra*, W. H. Freeman and Co., San Francisco, and London, 1963.
13. C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," *J. Research NBS* **45** (1950), 255–282.
14. S. Kaniel, "Estimates for some computational techniques in linear algebra," *Math. Comp.* **20** (1966), 369–378.
15. C. C. Paige, "The computation of eigenvalues and eigenvectors of very large sparse matrices," Ph.D. Thesis, London Univ., Institute of Computer Science, 1971.
16. C. C. Paige, "Computational variants of the Lanczos method for the eigenproblem," *J. Inst. Math. Appl.* **10** (1972), 373–381.
17. D. O'Leary, "Hybrid conjugate gradient algorithms," Ph.D. Thesis, Computer Science Dept., Stanford Univ., 1975.
18. M. Engeli, T. Ginsburg, H. Rutishauser and E. Stiefel, *Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems*, Birkhäuser Verlag, Basel/Stuttgart, 1959.
19. G. Forsythe and E. G. Straus, "On best conditioned matrices," *Proc. Amer. Math. Soc.* **6** (1955), 340–345.
20. D. M. Young, L. Hayes, and E. Schleicher, "The use of the accelerated SSOR method to solve large linear systems," Abstract, 1975 SIAM Fall Meeting, San Francisco.
21. L. W. Ehrlich, "On some experience using matrix splitting and conjugate gradient," Abstract, 1975 SIAM Fall Meeting, San Francisco.
22. L. W. Ehrlich, "The block symmetric successive overrelaxation method," *J. SIAM* **12** (1964), 807–826.
23. P. Concus and G. H. Golub, "Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations," *SIAM J. Numer. Anal.* **10** (1973), 1103–1120.
24. R. R. Underwood, "An approximate factorization procedure based on the block Cholesky factorization and its use with the conjugate gradient method," Tech. Rept., General Electric Nuclear Energy Division, San Jose, CA (to appear).
25. B. L. Buzbee, F. W. Dorr, J. A. George, and G. H. Golub, "The direct solution of the discrete Poisson equation in irregular regions," *SIAM J. Num. Anal.* **8** (1971), 722–736.
26. J. A. George, "The use of direct methods for the solution of the discrete Poisson equation on non-rectangular regions," Report CS-70-159, Computer Science Dept., Stanford Univ., Stanford, CA, (1970).
27. W. Proskurowski and O. Widlund, "On the numerical solution of Laplace's and Helmholtz's equations by the capacitance matrix method," Tech. Rept., Courant Institute, NYU (to appear).

9

HERMITIAN AND SKEW-HERMITIAN SPLITTING METHODS FOR NON-HERMITIAN POSITIVE DEFINITE LINEAR SYSTEMS (WITH ZHONG-ZHI BAI AND MICHAEL K. NG)

SIAM J. Matrix Anal. Appl. 24, 603–626 (2003).

It was in 1997 when I first met Gene at Hong Kong on the Workshop on Scientific Computing dedicated to his 65th birthday. Then Gene introduced me to my first two foreign teachers, Iain S. Duff and Andrew J. Wathen, whom I visited for one year from August 1997 to August 1998. The communication and discussion with Gene, Iain, and Andy on solving nonsymmetric linear systems arising from discretization of the convection–diffusion equations by making use of efficient preconditioning techniques changed my research interest and also my academic life, as I then began to study more application-oriented problems and computation-based methods.

Gene said to me that the numerical linear algebra community is a very friendly one; and the longer one is in this community, the more one will get this feeling. Later, I find that what he told me is exactly right. Gene is always very kind, helpful, encouraging, and concerned about young researchers.

About the Hermitian and skew-Hermitian splitting (HSS) iteration method coauthored with Gene and Michael, its embryonic form is a two-step iteration scheme including two iteration parameters, like the classical alternating direction implicit (ADI) iteration scheme for solving partial differential equations. Once this idea came out, we immediately realized that this two-parameter iteration scheme lacks mathematical beauty and its convergence demands complicated conditions. We observed that its equal-parameter case can completely avoid these shortcomings. This finally resulted in the present version of the HSS iteration method.

In fact, there are many good words to say about Gene at this moment, but let me just end this short essay with the following words using a Chinese idiom: Happy Birthday to Gene, my good teacher and helpful friend!

Zhong-Zhi Bai
Beijing, P.R. China

I would like to thank Gene for his inspired guidance to me in studying iterative methods, nonsymmetric systems, inverse problems and total least squares problems and other numerical linear algebra problems. I thank him for his enlightening suggestions and comments from which I benefit a lot in my career.

Michael K. Ng
Hong Kong

SIAM J. MATRIX ANAL. APPL.
Vol. 24, No. 3, pp. 603–626

HERMITIAN AND SKEW-HERMITIAN SPLITTING METHODS FOR NON-HERMITIAN POSITIVE DEFINITE LINEAR SYSTEMS*

ZHONG-ZHI BAI[†], GENE H. GOLUB[‡], AND MICHAEL K. NG[§]

Abstract. We study efficient iterative methods for the large sparse non-Hermitian positive definite system of linear equations based on the Hermitian and skew-Hermitian splitting of the coefficient matrix. These methods include a Hermitian/skew-Hermitian splitting (HSS) iteration and its inexact variant, the inexact Hermitian/skew-Hermitian splitting (IHSS) iteration, which employs some Krylov subspace methods as its inner iteration processes at each step of the outer HSS iteration. Theoretical analyses show that the HSS method converges unconditionally to the unique solution of the system of linear equations. Moreover, we derive an upper bound of the contraction factor of the HSS iteration which is dependent solely on the spectrum of the Hermitian part and is independent of the eigenvectors of the matrices involved. Numerical examples are presented to illustrate the effectiveness of both HSS and IHSS iterations. In addition, a model problem of a three-dimensional convection-diffusion equation is used to illustrate the advantages of our methods.

Key words. non-Hermitian matrix, splitting, Hermitian matrix, skew-Hermitian matrix, iterative methods

AMS subject classifications. 65F10, 65F15, 65T10

PII. S0895479801395458

1. Introduction. Many problems in scientific computing give rise to a system of linear equations

$$Ax = b, \quad A \in \mathbb{C}^{n\times n} \text{ nonsingular}, \quad \text{and } x, b \in \mathbb{C}^n, \tag{1.1}$$

with A a large sparse non-Hermitian and positive definite matrix.

Iterative methods for the system of linear equations (1.1) require efficient splittings of the coefficient matrix A. For example, the Jacobi and the Gauss–Seidel iterations [16] split the matrix A into its diagonal and off-diagonal (respectively, strictly lower and upper triangular) parts, and the generalized conjugate gradient (CG) method [7] and the generalized Lanczos method [27] split the matrix A into its Hermitian and skew-Hermitian parts; see also [11, 17, 26, 1] and [2], respectively. Because the matrix A naturally possesses a Hermitian/skew-Hermitian splitting (HSS) [7]

$$A = H + S, \tag{1.2}$$

*Received by the editors September 24, 2001; accepted for publication (in revised form) by D. P. O'Leary July 23, 2002; published electronically January 17, 2003.
http://www.siam.org/journals/simax/24-3/39545.html

[†]State Key Laboratory of Scientific/Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, P.O. Box 2719, Beijing 100080, The People's Republic of China (bzz@lsec.cc.ac.cn). The research of this author was subsidized by The Special Funds for Major State Basic Research Projects G1999032803.

[‡]Department of Computer Science, Stanford University, Stanford, CA 94305 (golub@sccm.stanford.edu). The work of this author was supported in part by DOE-FC02-01ER4177.

[§]Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong (mng@maths.hku.hk). The research of this author was supported in part by Hong Kong Research Grants Council grants HKU 7132/00P and 7130/02P, and by HKU CRCG grants 10203501, 10203907, and 10203408.

where

$$H = \frac{1}{2}(A + A^*) \quad \text{and} \quad S = \frac{1}{2}(A - A^*), \tag{1.3}$$

we will study in this paper efficient iterative methods based on this particular matrix splitting for solving the system of linear equations (1.1).

Now $A = H(I + H^{-1}S)$, and thus $A^{-1} = (I + H^{-1}S)^{-1}H^{-1}$. Thus, if we replace $(I + H^{-1}S)^{-1}$ by its first order approximation $I - H^{-1}S$, then $(I - H^{-1}S)H^{-1}$ could be employed as a preconditioner to the matrix A. Of course, the preconditioning effect is completely determined by the spectral distribution of the matrix $H^{-1}S$, and it is satisfactory if the Hermitian part H is dominant [1]. On the other hand, if the skew-Hermitian part S is dominant, we can use an alternative preconditioning strategy recently presented by Golub and Vanderstraeten in [15]. Their basic idea is to invert the shifted skew-Hermitian matrix $\alpha I + S$ and then employ $(I - (S + \alpha I)^{-1}(H - \alpha I))(S + \alpha I)^{-1}$ as a preconditioner to the matrix A. In fact, the preconditioning effect for this preconditioner depends not only on the spectrum but also on the eigenvectors of the matrix $(S + \alpha I)^{-1}(H - \alpha I)$, which is, however, closely related to the shift α. For a nearly optimal α, numerical experiments in [15] on a variety of problems from real-world applications have shown that the reductions in terms of iteration count largely compensate for the additional work per iteration when compared to standard preconditioners. We remark that, for both preconditioners, exact inverses of the matrices H and $\alpha I + S$ are quite expensive, and, therefore, some further approximations, e.g., the incomplete Cholesky (IC) factorization [21, 20] and the incomplete orthogonal-triangular (IQR) factorization [3], to these two matrices may be respectively adopted in actual applications. However, theoretical analysis about existence, stability, and accuracy of the resulting iterative method are considerably difficult.

Moreover, based on the HSS (1.2)–(1.3), in this paper we present a different approach to solve the system of linear equations (1.1), called the HSS iteration, and it is as follows.

The HSS iteration method. *Given an initial guess $x^{(0)}$, for $k = 0, 1, 2, \dots$, until $\{x^{(k)}\}$ converges, compute*

$$\begin{cases} (\alpha I + H)x^{(k+\frac{1}{2})} & = & (\alpha I - S)x^{(k)} + b, \\ (\alpha I + S)x^{(k+1)} & = & (\alpha I - H)x^{(k+\frac{1}{2})} + b, \end{cases} \tag{1.4}$$

where α is a given positive constant.

Evidently, each iterate of the HSS iteration alternates between the Hermitian part H and the skew-Hermitian part S of the matrix A, analogously to the classical alternating direction implicit (ADI) iteration for solving partial differential equations; see Peaceman and Rachford [23] and Douglas and Rachford [8]. Results associated to the stationary iterative method with alternation can be also found in Benzi and Szyld [4]. Theoretical analysis shows that the HSS iteration (1.4) converges unconditionally to the unique solution of the system of linear equations (1.1). The upper bound of the contraction factor of the HSS iteration is dependent on the spectrum of the Hermitian part H but is independent of the spectrum of the skew-Hermitian part S as well as the eigenvectors of the matrices H, S, and A. In addition, the optimal value of the parameter α for the upper bound of the contraction factor of the HSS iteration can be determined by the lower and the upper eigenvalue bounds of the matrix H.

Note that we can reverse the roles of the matrices H and S in the above HSS iteration method so that we may first solve the system of linear equations with coefficient matrix $\alpha I + S$ and then solve the system of linear equations with coefficient matrix $\alpha I + H$.

The two half-steps at each HSS iterate require exact solutions with the n-by-n matrices $\alpha I + H$ and $\alpha I + S$. However, this is very costly and impractical in actual implementations. To further improve the computing efficiency of the HSS iteration, we can employ, for example, the CG method to solve the system of linear equations with coefficient matrix $\alpha I + H$ and some Krylov subspace method to solve the system of linear equations with coefficient matrix $\alpha I + S$ to some prescribed accuracy at each step of the HSS iteration. Other possible choices of inner iteration solvers are classical relaxation methods, multigrid methods or multilevel methods, etc. This results in an inexact Hermitian/skew-Hermitian splitting (IHSS) iteration. The tolerances (or numbers of inner iteration steps) for inner iterative methods may be different and may be changed according to the outer iteration scheme. Therefore, the IHSS iteration is actually a nonstationary iterative method for solving the system of linear equations (1.1).

Model problem analysis for a three-dimensional convection-diffusion equation and numerical implementations show that both HSS and IHSS iterations are feasible and efficient for solving the non-Hermitian positive definite system of linear equations (1.1).

The organization of this paper is as follows. In section 2, we study the convergence properties and analyze the convergence rate of the HSS iteration. In section 3, we establish the IHSS iteration and study its convergence property. The three-dimensional convection-diffusion equation is employed as a model problem to give intuitive illustration for the convergence theory for the HSS iteration in section 4. Numerical experiments are presented in section 5 to show the effectiveness of our methods. And, finally, in section 6, we draw a brief conclusion and include some remarks. Moreover, the basic lemma used in the model problem analysis in section 4 and some illustrative remarks can be found in the appendix.

2. Convergence analysis of the HSS iteration.

In this section, we study the convergence rate of the HSS iteration. We first note that the HSS iteration method can be generalized to the two-step splitting iteration framework, and the following lemma describes a general convergence criterion for a two-step splitting iteration.

LEMMA 2.1. *Let $A \in \mathbb{C}^{n\times n}$, $A = M_i - N_i$ $(i = 1, 2)$ be two splittings*[1] *of the matrix A, and let $x^{(0)} \in \mathbb{C}^n$ be a given initial vector. If $\{x^{(k)}\}$ is a two-step iteration sequence defined by*

$$\begin{cases} M_1 x^{(k+\frac{1}{2})} & = & N_1 x^{(k)} + b, \\ M_2 x^{(k+1)} & = & N_2 x^{(k+\frac{1}{2})} + b, \end{cases}$$

$k = 0, 1, 2, \dots$, *then*

$$x^{(k+1)} = M_2^{-1} N_2 M_1^{-1} N_1 x^{(k)} + M_2^{-1}(I + N_2 M_1^{-1}) b, \quad k = 0, 1, 2, \dots .$$

Moreover, if the spectral radius $\rho(M_2^{-1}N_2M_1^{-1}N_1)$ of the iteration matrix $M_2^{-1}N_2M_1^{-1}N_1$ is less than 1, then the iterative sequence $\{x^{(k)}\}$ converges to the unique solution $x^* \in \mathbb{C}^n$ of the system of linear equations (1.1) for all initial vectors $x^{(0)} \in \mathbb{C}^n$.

For the convergence property of the HSS iteration, we apply the above results to obtain the following main theorem.

[1] Here and in what follows, $A = M - N$ is called a splitting of the matrix A if M is a nonsingular matrix.

THEOREM 2.2. *Let $A \in \mathbb{C}^{n\times n}$ be a positive definite matrix, let $H = \frac{1}{2}(A + A^*)$ and $S = \frac{1}{2}(A - A^*)$ be its Hermitian and skew-Hermitian parts, and let α be a positive constant. Then the iteration matrix $M(\alpha)$ of the HSS iteration is given by*

$$M(\alpha) = (\alpha I + S)^{-1}(\alpha I - H)(\alpha I + H)^{-1}(\alpha I - S), \tag{2.1}$$

and its spectral radius $\rho(M(\alpha))$ is bounded by

$$\sigma(\alpha) \equiv \max_{\lambda_i \in \lambda(H)} \left| \frac{\alpha - \lambda_i}{\alpha + \lambda_i} \right|,$$

where $\lambda(H)$ is the spectral set of the matrix H. Therefore, it holds that

$$\rho(M(\alpha)) \le \sigma(\alpha) < 1 \quad \forall \alpha > 0;$$

i.e., the HSS iteration converges to the unique solution $x^* \in \mathbb{C}^n$ of the system of linear equations (1.1).

Proof. By putting

$$M_1 = \alpha I + H, \quad N_1 = \alpha I - S, \quad M_2 = \alpha I + S, \quad \text{and} \quad N_2 = \alpha I - H$$

in Lemma 2.1 and noting that $\alpha I + H$ and $\alpha I + S$ are nonsingular for any positive constant α, we obtain (2.1).

By the similarity invariance of the matrix spectrum, we have

$$\begin{aligned}\rho(M(\alpha)) &= \rho((\alpha I - H)(\alpha I + H)^{-1}(\alpha I - S)(\alpha I + S)^{-1}) \\ &\le \|(\alpha I - H)(\alpha I + H)^{-1}(\alpha I - S)(\alpha I + S)^{-1}\|_2 \\ &\le \|(\alpha I - H)(\alpha I + H)^{-1}\|_2 \|(\alpha I - S)(\alpha I + S)^{-1}\|_2.\end{aligned}$$

Letting $Q(\alpha) = (\alpha I - S)(\alpha I + S)^{-1}$ and noting that $S^* = -S$, we see that

$$\begin{aligned}Q(\alpha)^*Q(\alpha) &= (\alpha I - S)^{-1}(\alpha I + S)(\alpha I - S)(\alpha I + S)^{-1} \\ &= (\alpha I - S)^{-1}(\alpha I - S)(\alpha I + S)(\alpha I + S)^{-1} \ = \ I.\end{aligned}$$

That is, $Q(\alpha)$ is a unitary matrix. ($Q(\alpha)$ is also called the Cayley transform of S.) Therefore, $\|Q(\alpha)\|_2 = 1$. It then follows that

$$\rho(M(\alpha)) \le \|(\alpha I - H)(\alpha I + H)^{-1}\|_2 = \max_{\lambda_i \in \lambda(H)} \left| \frac{\alpha - \lambda_i}{\alpha + \lambda_i} \right|.$$

Since $\lambda_i > 0 (i = 1, 2, \dots, n)$ and α is a positive constant, it is easy to see that $\rho(M(\alpha)) \le \sigma(\alpha) < 1$. □

Theorem 2.2 shows that the convergence speed of the HSS iteration is bounded by $\sigma(\alpha)$, which depends only on the spectrum of the Hermitian part H but does not depend on the spectrum of the skew-Hermitian part S, on the spectrum of the coefficient matrix A, or on the eigenvectors of the matrices H, S, and A.

Now, if we introduce a vector norm $|||x||| = \|(\alpha I + S)x\|_2$ (for all $x \in \mathbb{C}^n$) and represent the induced matrix norm by $|||X||| = \|(\alpha I + S)X(\alpha I + S)^{-1}\|_2$ (for all $X \in \mathbb{C}^{n\times n}$), then, from the proof of Theorem 2.2, we see that

$$|||M(\alpha)||| = \|(\alpha I - H)(\alpha I + H)^{-1}(\alpha I - S)(\alpha I + S)^{-1}\|_2 \le \sigma(\alpha),$$

and it follows that

$$|||x^{(k+1)} - x^*||| \le \sigma(\alpha)|||x^{(k)} - x^*|||, \qquad k = 0, 1, 2, \dots .$$

Therefore, $\sigma(\alpha)$ is also an upper bound of the contraction factor of the HSS iteration in the sense of the $|||\ \ |||$-norm.

We remark that if the minimum and the maximum eigenvalues of the Hermitian part H are known, then the optimal parameter α for $\sigma(\alpha)$ (or the upper bound of $\rho(M(\alpha))$ or $|||M(\alpha)|||$) can be obtained. This fact is precisely stated as the following corollary.

COROLLARY 2.3. *Let $A \in \mathbb{C}^{n\times n}$ be a positive definite matrix, let $H = \frac{1}{2}(A + A^*)$ and $S = \frac{1}{2}(A - A^*)$ be its Hermitian and skew-Hermitian parts, and let $\gamma_{\min}$ and $\gamma_{\max}$ be the minimum and the maximum eigenvalues of the matrix H, respectively, and let α be a positive constant. Then*

$$\alpha^* \equiv \arg \min_{\alpha} \left\{ \max_{\gamma_{\min}\le\lambda\le\gamma_{\max}} \left| \frac{\alpha - \lambda}{\alpha + \lambda} \right| \right\} = \sqrt{\gamma_{\min}\gamma_{\max}},$$

and

$$\sigma(\alpha^*) = \frac{\sqrt{\gamma_{\max}} - \sqrt{\gamma_{\min}}}{\sqrt{\gamma_{\max}} + \sqrt{\gamma_{\min}}} = \frac{\sqrt{\kappa(H)} - 1}{\sqrt{\kappa(H)} + 1},$$

where $\kappa(H)$ is the spectral condition number of H.

Proof. Now,

$$\sigma(\alpha) = \max \left\{ \left| \frac{\alpha - \gamma_{\min}}{\alpha + \gamma_{\min}} \right|, \left| \frac{\alpha - \gamma_{\max}}{\alpha + \gamma_{\max}} \right| \right\}. \tag{2.2}$$

To compute an approximate optimal $\alpha > 0$ such that the convergence factor $\rho(M(\alpha))$ of the HSS iteration is minimized, we can minimize the upper bound $\sigma(\alpha)$ of $\rho(M(\alpha))$ instead. If α^* is such a minimum point, then it must satisfy $\alpha^* - \gamma_{\min} > 0$, $\alpha^* - \gamma_{\max} < 0$, and

$$\frac{\alpha^* - \gamma_{\min}}{\alpha^* + \gamma_{\min}} = \frac{\gamma_{\max} - \alpha^*}{\gamma_{\max} + \alpha^*}.$$

Therefore,

$$\alpha^* = \sqrt{\gamma_{\min}\gamma_{\max}},$$

and the result follows. □

We emphasize that, in Corollary 2.3, the optimal parameter α^* minimizes only the upper bound $\sigma(\alpha)$ of the spectral radius of the iteration matrix but does not minimize the spectral radius itself; for an illustration of this phenomenon, see, e.g., Table 2.

Corollary 2.3 shows that, when the so-called optimal parameter α^* is employed, the upper bound of the convergence rate of the HSS iteration is about the same as that of the CG method, and it does become the same when, in particular, the coefficient matrix A is Hermitian. It should be mentioned that, when the coefficient matrix A is normal, we have $HS = SH$, and, therefore, $\rho(M(\alpha)) = |||M(\alpha)||| = \sigma(\alpha)$. The optimal parameter α^* then minimizes all of these three quantities.

3. The IHSS iteration. The two half-steps at each step of the HSS iteration require finding solutions with the n-by-n matrices $\alpha I + H$ and $\alpha I + S$, which is, however, very costly and impractical in actual implementations. To overcome this disadvantage and further improve the efficiency of the HSS iteration, we can solve the two subproblems iteratively. More specifically, we may employ the CG method to solve the system of linear equations with coefficient matrix $\alpha I + H$, because $\alpha I + H$ is Hermitian positive definite, and some Krylov subspace method [7, 24, 18] to solve the system of linear equations with coefficient matrix $\alpha I + S$. This results in the following IHSS iteration for solving the system of linear equations (1.1).

The IHSS iteration method. *Given an initial guess $\bar{x}^{(0)}$, for $k = 0, 1, 2, \dots$, until $\{\bar{x}^{(k)}\}$ converges, solve $\bar{x}^{(k+\frac{1}{2})}$ approximately from*

$$(\alpha I + H)\bar{x}^{(k+\frac{1}{2})} \approx (\alpha I - S)\bar{x}^{(k)} + b$$

by employing an inner iteration (e.g., the CG method) with $\bar{x}^{(k)}$ as the initial guess; then solve $\bar{x}^{(k+1)}$ approximately from

$$(\alpha I + S)\bar{x}^{(k+1)} \approx (\alpha I - H)\bar{x}^{(k+\frac{1}{2})} + b$$

by employing an inner iteration (e.g., some Krylov subspace method) with $\bar{x}^{(k+\frac{1}{2})}$ as the initial guess, where α is a given positive constant.

To simplify numerical implementation and convergence analysis, we may rewrite the above IHSS iteration method as the following equivalent scheme.

Given an initial guess $\bar{x}^{(0)}$, for $k = 0, 1, 2, \dots$, until $\{\bar{x}^{(k)}\}$ converges,

1. *approximate the solution of $(\alpha I + H)\bar{z}^{(k)} = \bar{r}^{(k)}$ ($\bar{r}^{(k)} = b - A\bar{x}^{(k)}$) by iterating until $\bar{z}^{(k)}$ is such that the residual*

$$\bar{p}^{(k)} = \bar{r}^{(k)} - (\alpha I + H)\bar{z}^{(k)} \tag{3.1}$$

satisfies

$$\|\bar{p}^{(k)}\| \le \varepsilon_k \|\bar{r}^{(k)}\|,$$

and then compute $\bar{x}^{(k+\frac{1}{2})} = \bar{x}^{(k)} + \bar{z}^{(k)}$;

2. *approximate the solution of $(\alpha I + S)\bar{z}^{(k+\frac{1}{2})} = \bar{r}^{(k+\frac{1}{2})}$ ($\bar{r}^{(k+\frac{1}{2})} = b - A\bar{x}^{(k+\frac{1}{2})}$) by iterating until $\bar{z}^{(k+\frac{1}{2})}$ is such that the residual*

$$\bar{q}^{(k+\frac{1}{2})} = \bar{r}^{(k+\frac{1}{2})} - (\alpha I + S)\bar{z}^{(k+\frac{1}{2})} \tag{3.2}$$

satisfies

$$\|\bar{q}^{(k+\frac{1}{2})}\| \le \eta_k \|\bar{r}^{(k+\frac{1}{2})}\|,$$

and then compute $\bar{x}^{(k+1)} = \bar{x}^{(k+\frac{1}{2})} + \bar{z}^{(k+\frac{1}{2})}$. Here $\|\cdot\|$ is a norm of a vector.

In the following theorem, we analyze the above IHSS iteration method in slightly more general terms. In particular, we consider inexact iterations for the two-step splitting technique (cf. Lemma 2.1). To this end, we generalize the norm $|||\cdot|||$ to $|||\cdot|||_{M_2}$, which is defined by $|||x|||_{M_2} = \|M_2 x\|$ (for all $x \in \mathbb{C}^n$), which immediately induces the matrix norm $|||X|||_{M_2} = \|M_2 X M_2^{-1}\|$ (for all $X \in \mathbb{C}^{n\times n}$).

THEOREM 3.1. *Let $A \in \mathbb{C}^{n\times n}$ and $A = M_i - N_i$ ($i = 1, 2$) be two splittings of the matrix A. If $\{\bar{x}^{(k)}\}$ is an iterative sequence defined as*

$$\bar{x}^{(k+\frac{1}{2})} = \bar{x}^{(k)} + \bar{z}^{(k)}, \quad \textit{with} \quad M_1\bar{z}^{(k)} = \bar{r}^{(k)} + \bar{p}^{(k)}, \tag{3.3}$$

satisfying $\frac{\|\bar{p}^{(k)}\|}{\|\bar{r}^{(k)}\|} \le \varepsilon_k$, *where* $\bar{r}^{(k)} = b - A\bar{x}^{(k)}$, *and*

$$\bar{x}^{(k+1)} = \bar{x}^{(k+\frac{1}{2})} + \bar{z}^{(k+\frac{1}{2})}, \quad \textit{with} \quad M_2\bar{z}^{(k+\frac{1}{2})} = \bar{r}^{(k+\frac{1}{2})} + \bar{q}^{(k+\frac{1}{2})}, \tag{3.4}$$

satisfying $\frac{\|\bar{q}^{(k+\frac{1}{2})}\|}{\|\bar{r}^{(k+\frac{1}{2})}\|} \le \eta_k$, *where* $\bar{r}^{(k+\frac{1}{2})} = b - A\bar{x}^{(k+\frac{1}{2})}$, *then* $\{\bar{x}^{(k)}\}$ *is of the form*

$$\begin{aligned} \bar{x}^{(k+1)} &= M_2^{-1}N_2M_1^{-1}N_1\bar{x}^{(k)} + M_2^{-1}(I + N_2M_1^{-1})b \\ &\quad + M_2^{-1}(N_2M_1^{-1}\bar{p}^{(k)} + \bar{q}^{(k+\frac{1}{2})}). \end{aligned} \tag{3.5}$$

Moreover, if $x^* \in \mathbb{C}^n$ *is the exact solution of the system of linear equations* (1.1), *then we have*

(3.6)
$$|||\bar{x}^{(k+1)} - x^*|||_{M_2} \le \big(\bar{\sigma} + \bar{\mu}\bar{\theta}\varepsilon_k + \bar{\theta}(\bar{\rho} + \bar{\theta}\bar{\nu}\varepsilon_k)\eta_k\big)\, |||\bar{x}^{(k)} - x^*|||_{M_2}, \quad k = 0, 1, 2, \dots ,$$

where

$$\begin{array}{lllllllll} \bar{\sigma} &=& \|N_2M_1^{-1}N_1M_2^{-1}\|, & \bar{\rho} &=& \|M_2M_1^{-1}N_1M_2^{-1}\|, & \bar{\mu} &=& \|N_2M_1^{-1}\|, \\ \bar{\theta} &=& \|AM_2^{-1}\|, & \bar{\nu} &=& \|M_2M_1^{-1}\|. \end{array}$$

In particular, if

$$\bar{\sigma} + \bar{\mu}\bar{\theta}\varepsilon_{\max} + \bar{\theta}(\bar{\rho} + \bar{\theta}\bar{\nu}\varepsilon_{\max})\eta_{\max} < 1, \tag{3.7}$$

then the iterative sequence $\{\bar{x}^{(k)}\}$ *converges to* $x^* \in \mathbb{C}^n$, *where* $\varepsilon_{\max} = \max_k\{\varepsilon_k\}$ *and* $\eta_{\max} = \max_k\{\eta_k\}$.

Proof. From (3.3), we obtain

$$\begin{aligned} \bar{x}^{(k+\frac{1}{2})} &= \bar{x}^{(k)} + M_1^{-1}(\bar{r}^{(k)} + \bar{p}^{(k)}) \\ &= (I - M_1^{-1}A)\bar{x}^{(k)} + M_1^{-1}b + M_1^{-1}\bar{p}^{(k)} \\ &= M_1^{-1}N_1\bar{x}^{(k)} + M_1^{-1}b + M_1^{-1}\bar{p}^{(k)}. \end{aligned} \tag{3.8}$$

Similarly, from (3.4), we get

$$\begin{aligned} \bar{x}^{(k+1)} &= \bar{x}^{(k+\frac{1}{2})} + M_2^{-1}(\bar{r}^{(k+\frac{1}{2})} + \bar{q}^{(k+\frac{1}{2})}) \\ &= (I - M_2^{-1}A)\bar{x}^{(k+\frac{1}{2})} + M_2^{-1}b + M_2^{-1}\bar{q}^{(k+\frac{1}{2})} \\ &= M_2^{-1}N_2\bar{x}^{(k+\frac{1}{2})} + M_2^{-1}b + M_2^{-1}\bar{q}^{(k+\frac{1}{2})}. \end{aligned}$$

Therefore, we have

$$\begin{aligned} \bar{x}^{(k+1)} &= M_2^{-1}N_2(M_1^{-1}N_1\bar{x}^{(k)} + M_1^{-1}b + M_1^{-1}\bar{p}^{(k)}) \\ &\quad + M_2^{-1}b + M_2^{-1}\bar{q}^{(k+\frac{1}{2})} \\ &= M_2^{-1}N_2M_1^{-1}N_1\bar{x}^{(k)} + M_2^{-1}(I + N_2M_1^{-1})b \\ &\quad + M_2^{-1}(N_2M_1^{-1}\bar{p}^{(k)} + \bar{q}^{(k+\frac{1}{2})}), \end{aligned} \tag{3.9}$$

which is exactly (3.5).

Because $x^* \in \mathbb{C}^n$ is the exact solution of the system of linear equations (1.1), it must satisfy

$$x^* = M_1^{-1}N_1x^* + M_1^{-1}b \tag{3.10}$$

and

$$x^* = M_2^{-1}N_2M_1^{-1}N_1x^* + M_2^{-1}(I + N_2M_1^{-1})b. \tag{3.11}$$

By subtracting (3.10) from (3.8) and (3.11) from (3.9), respectively, we have

$$\bar{x}^{(k+\frac{1}{2})} - x^* = M_1^{-1}N_1(\bar{x}^{(k)} - x^*) + M_1^{-1}\bar{p}^{(k)} \tag{3.12}$$

and

$$\bar{x}^{(k+1)} - x^* = M_2^{-1}N_2M_1^{-1}N_1(\bar{x}^{(k)} - x^*) + M_2^{-1}(N_2M_1^{-1}\bar{p}^{(k)} + \bar{q}^{(k+\frac{1}{2})}). \tag{3.13}$$

Taking norms on both sides of the identities (3.12) and (3.13), we can obtain

$$\begin{aligned} |||\bar{x}^{(k+\frac{1}{2})} - x^*|||_{M_2} &\leq |||M_1^{-1}N_1(\bar{x}^{(k)} - x^*)|||_{M_2} + |||M_1^{-1}\bar{p}^{(k)}|||_{M_2} \\ &\leq |||M_1^{-1}N_1|||_{M_2}|||\bar{x}^{(k)} - x^*|||_{M_2} + |||M_1^{-1}\bar{p}^{(k)}|||_{M_2} \\ &\leq \|M_2M_1^{-1}N_1M_2^{-1}\| \cdot |||\bar{x}^{(k)} - x^*|||_{M_2} \\ &\quad +\|M_2M_1^{-1}\| \cdot \|\bar{p}^{(k)}\| \end{aligned} \tag{3.14}$$

and

$$\begin{aligned} |||\bar{x}^{(k+1)} - x^*|||_{M_2} &\leq |||M_2^{-1}N_2M_1^{-1}N_1|||_{M_2}|||\bar{x}^{(k)} - x^*|||_{M_2} \\ &\quad +|||M_2^{-1}(N_2M_1^{-1}\bar{p}^{(k)} + \bar{q}^{(k+\frac{1}{2})})|||_{M_2} \\ &= \|N_2M_1^{-1}N_1M_2^{-1}\| \cdot |||\bar{x}^{(k)} - x^*|||_{M_2} \\ &\quad +\|N_2M_1^{-1}\bar{p}^{(k)} + \bar{q}^{(k+\frac{1}{2})}\| \\ &\leq \|N_2M_1^{-1}N_1M_2^{-1}\| \cdot |||\bar{x}^{(k)} - x^*|||_{M_2} \\ &\quad +\|N_2M_1^{-1}\| \cdot \|\bar{p}^{(k)}\| + \|\bar{q}^{(k+\frac{1}{2})}\|. \end{aligned} \tag{3.15}$$

Noticing that

$$\|\bar{r}^{(k)}\| = \|b - A\bar{x}^{(k)}\| = \|A(x^* - \bar{x}^{(k)})\| \leq \|AM_2^{-1}\| \cdot |||x^* - \bar{x}^{(k)}|||_{M_2}$$

and

$$\|\bar{r}^{(k+\frac{1}{2})}\| = \|b - A\bar{x}^{(k+\frac{1}{2})}\| = \|A(x^* - \bar{x}^{(k+\frac{1}{2})})\| \leq \|AM_2^{-1}\| \cdot |||x^* - \bar{x}^{(k+\frac{1}{2})}|||_{M_2},$$

by (3.12), (3.14), and the definitions of the sequences $\{\bar{p}^{(k)}\}$ and $\{\bar{q}^{(k+\frac{1}{2})}\}$, we have

$$\|\bar{p}^{(k)}\| \leq \varepsilon_k\|\bar{r}^{(k)}\| \leq \varepsilon_k\|AM_2^{-1}\| \cdot |||\bar{x}^{(k)} - x^*|||_{M_2} \tag{3.16}$$

and

$$\begin{aligned} \|\bar{q}^{(k+\frac{1}{2})}\| &\leq \eta_k\|\bar{r}^{(k+\frac{1}{2})}\| \\ &\leq \eta_k\|AM_2^{-1}\|(\|M_2M_1^{-1}N_1M_2^{-1}\| \cdot |||\bar{x}^{(k)} - x^*|||_{M_2} \\ &\quad +\|M_2M_1^{-1}\| \cdot \|\bar{p}^{(k)}\|) \\ &\leq \eta_k\|AM_2^{-1}\|(\|M_2M_1^{-1}N_1M_2^{-1}\| \\ &\quad +\varepsilon_k\|M_2M_1^{-1}\| \cdot \|AM_2^{-1}\|)|||\bar{x}^{(k)} - x^*|||_{M_2}. \end{aligned} \tag{3.17}$$

Through substituting (3.16) and (3.17) into (3.15), we finally obtain

$$\begin{aligned} |||\bar{x}^{(k+1)} - x^*|||_{M_2} &\leq \|N_2M_1^{-1}N_1M_2^{-1}\| \cdot |||\bar{x}^{(k)} - x^*|||_{M_2} \\ &\quad +\|N_2M_1^{-1}\| \cdot \varepsilon_k\|AM_2^{-1}\| \cdot |||\bar{x}^{(k)} - x^*|||_{M_2} \\ &\quad +\eta_k\|AM_2^{-1}\|(\|M_2M_1^{-1}N_1M_2^{-1}\| \\ &\quad +\varepsilon_k\|M_2M_1^{-1}\| \cdot \|AM_2^{-1}\|)|||\bar{x}^{(k)} - x^*|||_{M_2} \\ &\leq \big(\bar{\sigma} + \bar{\mu}\bar{\theta}\varepsilon_k + \bar{\theta}(\bar{\rho} + \bar{\theta}\bar{\nu}\varepsilon_k)\eta_k\big)\,|||\bar{x}^{(k)} - x^*|||_{M_2}. \qquad \square \end{aligned}$$

We remark that, if the inner systems can be solved exactly in some applications, the corresponding quantities $\{\varepsilon_k\}$ and $\{\eta_k\}$, and hence $\varepsilon_{\max}$ and $\eta_{\max}$, are equal to zero. It then follows that the convergence rate of the IHSS iteration reduces to the same as that of the HSS iteration. In general, Theorem 3.1 shows that, in order to guarantee the convergence of the IHSS iteration, it is not necessary for $\{\varepsilon_k\}$ and $\{\eta_k\}$ to approach zero as k is increasing. All we need is that the condition (3.7) is satisfied.

By specializing Theorem 3.1 to the shifted Hermitian and skew-Hermitian splittings

$$\begin{aligned} A &= M_1 - N_1 \equiv (\alpha I + H) - (\alpha I - S) \\ &= M_2 - N_2 \equiv (\alpha I + S) - (\alpha I - H), \end{aligned}$$

we straightforwardly obtain the following convergence theorem about the IHSS iteration method.

THEOREM 3.2. *Let $A \in \mathbb{C}^{n\times n}$ be a positive definite matrix, let $H = \frac{1}{2}(A + A^*)$ and $S = \frac{1}{2}(A - A^*)$ be its Hermitian and skew-Hermitian parts, and let α be a positive constant. If $\{\bar{x}^{(k)}\}$ is an iterative sequence generated by the IHSS iteration method (cf.* (3.1) *and* (3.2)*) and if $x^* \in \mathbb{C}^n$ is the exact solution of the system of linear equations* (1.1)*, then it holds that*

$$|||\bar{x}^{(k+1)} - x^*||| \le (\sigma(\alpha) + \theta\rho\eta_k)(1 + \theta\varepsilon_k)|||\bar{x}^{(k)} - x^*|||, \quad k = 0, 1, 2, \dots ,$$

where

$$\rho = \|(\alpha I + S)(\alpha I + H)^{-1}\|_2, \quad \theta = \|A(\alpha I + S)^{-1}\|_2. \tag{3.18}$$

In particular, if $(\sigma(\alpha) + \theta\rho\eta_{\max})(1 + \theta\varepsilon_{\max}) < 1$, then the iterative sequence $\{\bar{x}^{(k)}\}$ converges to $x^* \in \mathbb{C}^n$, where $\varepsilon_{\max} = \max_k\{\varepsilon_k\}$ and $\eta_{\max} = \max_k\{\eta_k\}$.

According to Theorem 3.1, we want to choose tolerances so that the computational work of the two-step splitting iteration method is minimized. In fact, as we have remarked previously, the tolerances $\{\varepsilon_k\}$ and $\{\eta_k\}$ are not required to approach zero as k increases in order to get the convergence of the IHSS iteration but are required to approach zero in order to asymptotically recover the original convergence rate (cf. Theorem 2.2) of the HSS iteration.

The following theorem presents one possible way of choosing the tolerances $\{\varepsilon_k\}$ and $\{\eta_k\}$ such that the original convergence rate (cf. Lemma 2.1) of the two-step splitting iterative scheme can be asymptotically recovered.

THEOREM 3.3. *Let the assumptions in Theorem* 3.1 *be satisfied. Suppose that both $\{\tau_1(k)\}$ and $\{\tau_2(k)\}$ are nondecreasing and positive sequences satisfying $\tau_1(k) \ge 1$, $\tau_2(k) \ge 1$, and $\lim_{k\to\infty}\sup \tau_1(k) = \lim_{k\to\infty}\sup \tau_2(k) = +\infty$, and that both δ_1 and δ_2 are real constants in the interval* $(0, 1)$ *satisfying*

$$\varepsilon_k \le c_1\delta_1^{\tau_1(k)} \quad \text{and} \quad \eta_k \le c_2\delta_2^{\tau_2(k)}, \qquad k = 0, 1, 2, \dots , \tag{3.19}$$

where c_1 and c_2 are nonnegative constants. Then we have

$$|||\bar{x}^{(k+1)} - x^*|||_{M_2} \le \left(\sqrt{\bar{\sigma}} + \bar{\omega}\bar{\theta}\delta^{\tau(k)}\right)^2|||\bar{x}^{(k)} - x^*|||_{M_2}, \quad k = 0, 1, 2, \dots ,$$

where

$$\tau(k) = \min\{\tau_1(k), \tau_2(k)\}, \quad \delta = \max\{\delta_1, \delta_2\}, \tag{3.20}$$

and

$$\bar{\omega} = \max\left\{\sqrt{c_1 c_2 \bar{\nu}}, \quad \frac{1}{2\sqrt{\bar{\sigma}}}(c_1\bar{\mu} + c_2\bar{\rho})\right\}.$$

In particular, we have

$$\lim_{k\to\infty} \sup \frac{|||\bar{x}^{(k+1)} - x^*|||_{M_2}}{|||\bar{x}^{(k)} - x^*|||_{M_2}} = \bar{\sigma};$$

i.e., the convergence rate of the inexact two-step splitting iterative scheme is asymptotically the same as that of the exact two-step splitting iterative scheme.

Proof. From (3.6) and (3.19), we obtain, for $k = 0, 1, 2, \dots$, that

$$\begin{aligned} |||\bar{x}^{(k+1)} - x^*|||_{M_2} &\leq \left(\bar{\sigma} + \bar{\mu}\bar{\theta}\varepsilon_k + \bar{\theta}(\bar{\rho} + \bar{\theta}\bar{\nu}\varepsilon_k)\eta_k\right) |||\bar{x}^{(k)} - x^*|||_{M_2} \\ &\leq \left(\bar{\sigma} + \bar{\mu}\bar{\theta}c_1\delta_1^{\tau_1(k)} + \bar{\theta}(\bar{\rho} + \bar{\theta}\bar{\nu}c_1\delta_1^{\tau_1(k)})c_2\delta_2^{\tau_2(k)}\right)|||\bar{x}^{(k)} - x^*|||_{M_2} \\ &\leq \left(\bar{\sigma} + \bar{\mu}\bar{\theta}c_1\delta^{\tau(k)} + \bar{\theta}(\bar{\rho} + \bar{\theta}\bar{\nu}c_1\delta^{\tau(k)})c_2\delta^{\tau(k)}\right)|||\bar{x}^{(k)} - x^*|||_{M_2} \\ &= \left(\bar{\sigma} + (c_1\bar{\mu} + c_2\bar{\rho})\bar{\theta}\delta^{\tau(k)} + c_1c_2\bar{\nu}\bar{\theta}^2\delta^{2\tau(k)}\right)|||\bar{x}^{(k)} - x^*|||_{M_2} \\ &\leq \left(\bar{\sigma} + 2\bar{\omega}\sqrt{\bar{\sigma}}\bar{\theta}\delta^{\tau(k)} + \bar{\omega}^2\bar{\theta}^2\delta^{2\tau(k)}\right)|||\bar{x}^{(k)} - x^*|||_{M_2} \\ &= \left(\sqrt{\bar{\sigma}} + \bar{\omega}\bar{\theta}\delta^{\tau(k)}\right)^2|||\bar{x}^{(k)} - x^*|||_{M_2}. \end{aligned}$$

The result follows straightforwardly. □

Theorems 3.2 and 3.3 immediately result in the following convergence result of the IHSS iteration method.

THEOREM 3.4. *Let the assumptions in Theorem* 3.2 *be satisfied. Suppose that both* $\{\tau_1(k)\}$ *and* $\{\tau_2(k)\}$ *are nondecreasing and positive sequences satisfying* $\tau_1(k) \geq 1$, $\tau_2(k) \geq 1$, *and* $\lim_{k\to\infty}\sup\tau_1(k) = \lim_{k\to\infty}\sup\tau_2(k) = +\infty$, *and that both* δ_1 *and* δ_2 *are real constants in the interval* $(0, 1)$ *satisfying* (3.19). *Then it holds that*

$$|||\bar{x}^{(k+1)} - x^*||| \leq \left(\sqrt{\sigma(\alpha)} + \omega\theta\delta^{\tau(k)}\right)^2|||\bar{x}^{(k)} - x^*|||, \quad k = 0, 1, 2, \dots ,$$

where ρ *and* θ *are defined by* (3.18), $\tau(k)$ *and* δ *are defined by* (3.20), *and*

$$\omega = \max\left\{\sqrt{c_1 c_2 \rho}, \quad \frac{1}{2\sqrt{\sigma(\alpha)}}(c_1\sigma(\alpha) + c_2\rho)\right\}.$$

In particular, we have

$$\lim_{k\to\infty} \sup \frac{|||\bar{x}^{(k+1)} - x^*|||}{|||\bar{x}^{(k)} - x^*|||} = \sigma(\alpha);$$

i.e., the convergence rate of the IHSS iteration method is asymptotically the same as that of the HSS iteration method.

According to Theorem 3.4, we show that, if the tolerances $\{\varepsilon_k\}$ and $\{\eta_k\}$ are chosen as in (3.19), then the IHSS iteration converges to the unique solution $x^* \in \mathbb{C}^n$ of the system of linear equations (1.1), and the upper bound of the asymptotic convergence factor of the IHSS iteration tends to $\sigma(\alpha)$ of that of the HSS iteration (cf. Theorem 2.2). Moreover, we remark that we may replace (3.19) by other rules for which $\{\varepsilon_k\}$ and $\{\eta_k\}$ approach zero. See [14].

TABLE 1
Work to compute a sweep of the IHSS method.

Operation	Work
$\bar{r}^{(k)} = b - A\bar{x}^{(k)}$	$n + a$
$(\alpha I + H)\bar{z}^{(k+\frac{1}{2})} = \bar{r}^{(k)}$	$\chi_k(H)$
$\bar{x}^{(k+\frac{1}{2})} = \bar{x}^{(k)} + \bar{z}^{(k+\frac{1}{2})}$	n
$\bar{r}^{(k+\frac{1}{2})} = b - A\bar{x}^{(k+\frac{1}{2})}$	$n + a$
$(\alpha I + S)\bar{z}^{(k+1)} = \bar{r}^{(k+\frac{1}{2})}$	$\chi_k(S)$
$\bar{x}^{(k+1)} = \bar{x}^{(k+\frac{1}{2})} + \bar{z}^{(k+1)}$	n

Computational complexity. To analyze the computational complexity of the HSS and the IHSS iterations, we need to estimate their computer times (via operation counts) and computer memories. Assume that a is the number of operations required to compute Ay for a given vector $y \in \mathbb{C}^n$ and $\chi_k(H)$ and $\chi_k(S)$ are the numbers of operations required to solve inner systems (3.1) and (3.2) inexactly with the tolerances $\{\varepsilon_k\}$ and $\{\eta_k\}$, respectively. Then the work to compute a sweep of the IHSS iteration is estimated using the results of Table 1. Straightforward calculations show that the total work to compute each step of the IHSS iteration is $\mathcal{O}(4n + 2a + \chi_k(H) + \chi_k(S))$.

In addition, a simple calculation shows that the memory is required to store $\bar{x}^{(k)}$, b, $\bar{r}^{(k)}$, $\bar{z}^{(k)}$. For the inexact solvers for inner systems (3.1) and (3.2), we require only some auxiliary vectors; for instance, CG-type methods need about five vectors [24]. Moreover, it is not necessary to store H and S explicitly as matrices, as all we need are two subroutines that perform the matrix-vector multiplications with respect to these two matrices. Therefore, the total amount of computer memory required is $\mathcal{O}(n)$, which has the same order of magnitude as the number of unknowns.

4. Application to the model convection-diffusion equation. We consider the three-dimensional convection-diffusion equation

$$-(u_{xx} + u_{yy} + u_{zz}) + q(u_x + u_y + u_z) = f(x, y, z) \tag{4.1}$$

on the unit cube $\Omega = [0,1] \times [0,1] \times [0,1]$, with constant coefficient q and subject to Dirichlet-type boundary conditions. When the seven-point finite difference discretization, for example, the centered differences to the diffusive terms, and the centered differences or the first order upwind approximations to the convective terms are applied to the above model convection-diffusion equation, we get the system of linear equations (1.1) with the coefficient matrix

$$A = T_x \otimes I \otimes I + I \otimes T_y \otimes I + I \otimes I \otimes T_z, \tag{4.2}$$

where the equidistant step-size $h = \frac{1}{n+1}$ is used in the discretization on all of the three directions and the natural lexicographic ordering is employed to the unknowns. In addition, $\otimes$ denotes the Kronecker product, and T_x, T_y, and T_z are tridiagonal matrices given by

$$T_x = \text{tridiag}(t_2, t_1, t_3), \quad T_y = \text{tridiag}(t_2, 0, t_3), \quad \text{and} \quad T_z = \text{tridiag}(t_2, 0, t_3),$$

with

$$t_1 = 6, \quad t_2 = -1 - r, \quad t_3 = -1 + r$$

if the first order derivatives are approximated by the centered difference scheme and

$$t_1 = 6 + 6r, \quad t_2 = -1 - 2r, \quad t_3 = -1$$

if the first order derivatives are approximated by the upwind difference scheme. Here

$$r = \frac{qh}{2}$$

is the mesh Reynolds number. For details, we refer to [9, 10] and [12, 13].

From (4.2), we know that the Hermitian part H and the skew-Hermitian part S of the matrix A are

$$H = H_x \otimes I \otimes I + I \otimes H_y \otimes I + I \otimes I \otimes H_z \tag{4.3}$$

and

$$S = S_x \otimes I \otimes I + I \otimes S_y \otimes I + I \otimes I \otimes S_z, \tag{4.4}$$

where

$$H_x = \text{ tridiag}\left(\frac{t_2+t_3}{2}, t_1, \frac{t_2+t_3}{2}\right), \quad H_y = H_z = \text{ tridiag}\left(\frac{t_2+t_3}{2}, 0, \frac{t_2+t_3}{2}\right),$$

$$S_\xi = \text{ tridiag}\left(\frac{t_2-t_3}{2}, 0, -\frac{t_2-t_3}{2}\right), \quad \xi \in \{x, y, z\}.$$

From Lemma A.1, we know, for the centered difference scheme, that

$$\begin{aligned} \min_{1\le j,k,l\le n} \lambda_{j,k,l}(H) &= 6(1-\cos(\pi h)), & \max_{1\le j,k,l\le n} \lambda_{j,k,l}(H) &= 6(1+\cos(\pi h)), \\ \min_{1\le j,k,l\le n} |\lambda_{j,k,l}(S)| &= 0, & \max_{1\le j,k,l\le n} |\lambda_{j,k,l}(S)| &= 6r\cos(\pi h). \end{aligned}$$

Therefore, the quantities in Theorem 2.2 can be obtained by concrete computations.

THEOREM 4.1. *For the system of linear equations* (1.1) *with the coefficient matrix* (4.2) *arising from the centered difference scheme for the three-dimensional model convection-diffusion equation* (4.1) *with the homogeneous Dirichlet boundary condition, the iteration sequence* $\{x^{(k)}\}$ *generated by the HSS iteration from an initial guess* $x^{(0)} \in \mathbb{C}^n$ *converges to its unique solution* $x^* \in \mathbb{C}^n$ *and satisfies*

$$|||x^{(k+1)} - x^*||| \le \left[1 - \pi h + \frac{1}{2}\pi^2 h^2 + O(h^3)\right] \cdot |||x^{(k)} - x^*|||, \qquad k = 0, 1, 2, \ldots .$$

We note that this bound is independent of q and the mesh Reynolds number. The results for the upwind difference scheme can be obtained in an analogous fashion. Since H and S in (4.3) and (4.4) can be diagonalized by sine transforms, the number of operations required at each HSS iteration is about $O(n^3 \log n)$. It follows that the total complexity of the HSS iteration is about $O(n^4 \log n)$ operations. Here n is the number of grid points in all three directions. Here the model problem is used as an example to illustrate the convergence rate of the HSS iteration. We remark that there may be other efficient methods for solving the model convection-diffusion equation (see [12, 13, 6]).

For a three-dimensional convection-diffusion system of linear equations arising from performing one step of cyclic reduction on an equidistant mesh, discretized by the centered and the upwind difference schemes, Greif and Varah [9, 10] considered two ordering strategies, analyzed block splittings of the coefficient matrices, and showed that the associated block Jacobi iterations converge for both the one-dimensional and the two-dimensional splittings with their spectral radii bounded by

$$1-\left(\frac{10}{9}\pi^2+\frac{1}{6}q^2\right)h^2+O(h^3) \quad \text{and} \quad 1-\left(2\pi^2+\frac{9}{10}q^2\right)h^2+O(h^3),$$

respectively. It is clear that these two bounds are larger than those of the HSS and the IHSS methods. Moreover, for the three-dimensional convection-diffusion model equation, the number of operations required for each step of the block Jacobi iteration is about $O(n^3)$ operations, and hence its total complexity is about $O(n^5)$ operations. We remark that their methods can provide an ordering for block Jacobi which can be used for preconditioning.

5. Numerical examples. In this section, we perform some numerical examples to demonstrate the effectiveness of both HSS and IHSS iterations.

5.1. Spectral radius. In this subsection, we first show in Figures 1 and 2 the spectral radius $\rho(M(\alpha))$ of the iteration matrix $M(\alpha)$ and its upper bound $\sigma(\alpha)$ for different α. Here the coefficient matrices A arise from the discretization of the differential equation

$$-u''+qu'=0$$

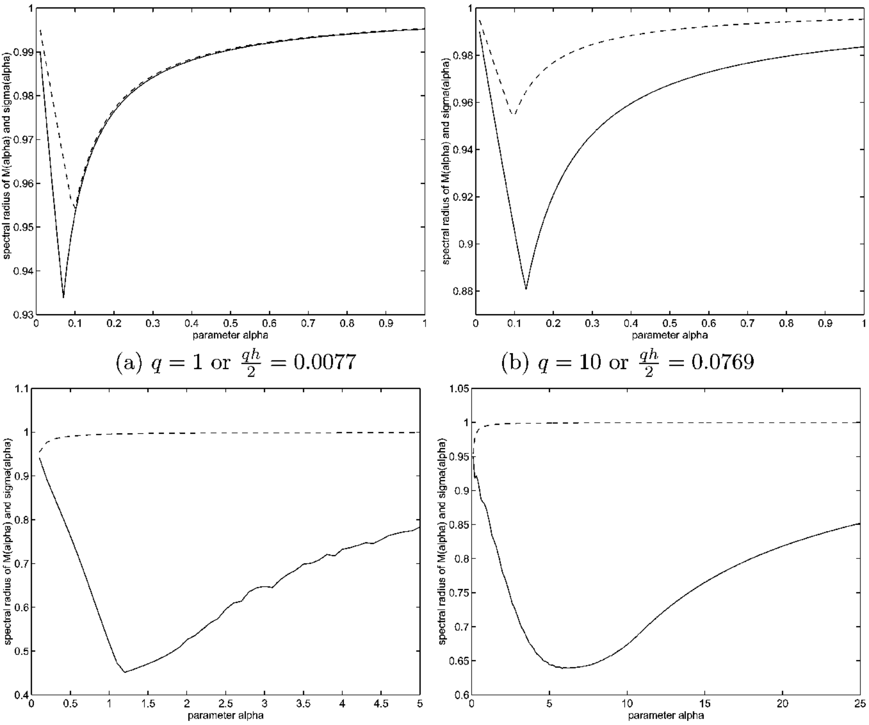
with the homogeneous boundary condition using the centered and the upwind difference schemes. In the tests, the size of the matrix A is 64-by-64. We see from the figures that both $\rho(M(\alpha))$ and $\sigma(\alpha)$ are always less than 1 for $\alpha > 0$. These results show that the HSS iteration always converges. Moreover, when q (or $qh/2$) is small, $\sigma(\alpha)$ is close to $\rho(M(\alpha))$, i.e., $\sigma(\alpha)$ is a good approximation to $\rho(M(\alpha))$. However, when q (or $qh/2$) is large (the skew-Hermitian part is dominant), $\sigma(\alpha)$ deviates from $\rho(M(\alpha))$ very much. From Figures 1 and 2, we see that the optimal parameter α_t,

$$\alpha_t \equiv \arg \min_{\alpha}\{\rho(M(\alpha))\},$$

is roughly equal to $qh/2$. To further investigate $\sigma(\alpha)$, we examine the parameter α in the HSS iteration in Figure 3. In the figure, we depict the spectral radii of the iteration matrices for different q (or $qh/2$) by using α^* in Corollary 2.3, $\tilde{\alpha}=qh/2$, and the optimal parameter α_t. It is clear that, when q (or $qh/2$) is small (i.e., the skew-Hermitian part is not dominant), α^* is close to α_t, and $M(\alpha^*)$ is a good estimate of $M(\alpha_t)$. However, when q (or $qh/2$) is large, α^* is not very useful; see Table 2. In contrast to α^*, we observe that $\tilde{\alpha}$ is close to α_t when q (or $qh/2$) is large. In the appendix, we give a remark to further explain why the spectral radius of $M(\tilde{\alpha})$ is less than $\sigma(\alpha^*)$ by using a 2-by-2 matrix example.

In Figure 4, we depict the eigenvalue distributions of the iteration matrices using α_t when $q=1,10,100,1000$. We see that the spectral radius of the iteration matrix for large q is less than that of the iteration matrix for small q.

5.2. Results for the HSS iteration. In this subsection, we test the HSS iteration by numerical experiments. All tests are started from the zero vector, performed



(a) $q = 1$ or $\frac{qh}{2} = 0.0077$

(b) $q = 10$ or $\frac{qh}{2} = 0.0769$

(c) $q = 100$ or $\frac{qh}{2} = 0.7692$

(d) $q = 1000$ or $\frac{qh}{2} = 7.6923$

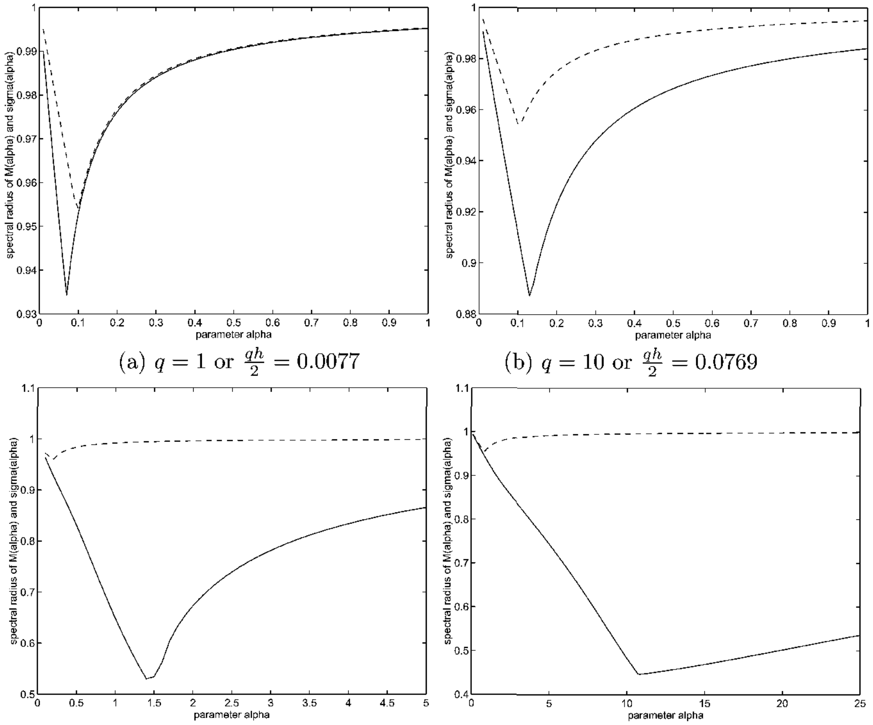
FIG. 1. *The spectral radius $\rho(M(\alpha))$ of the iteration matrices for different α: "——" and the upper bound $\sigma(\alpha)$ for different α: "- - - -" (centered difference scheme).*

TABLE 2
The spectral radii of the iteration matrices $M(\alpha^*)$, $M(\tilde{\alpha})$, and $M(\alpha_t)$ when $n = 64$.

Difference scheme	q	α^*	$\rho(M(\alpha^*))$	$\tilde{\alpha}$	$\rho(M(\tilde{\alpha}))$	α_t	$\rho(M(\alpha_t))$
centered	1	0.0966	0.9516	0.0077	0.9923	0.0700	0.9339
centered	10	0.0966	0.9086	0.0769	0.9264	0.1300	0.8807
centered	100	0.0966	0.9438	0.7692	0.6339	1.160	0.4487
centered	1000	0.0966	0.9511	7.6923	0.6445	5.800	0.6389
upwind	1	0.0974	0.9517	0.0077	0.9924	0.0700	0.9342
upwind	10	0.1041	0.9085	0.0769	0.9314	0.1300	0.8874
upwind	100	0.1710	0.9388	0.7692	0.7321	1.450	0.5237
upwind	1000	0.8399	0.9447	7.6923	0.6092	10.75	0.4466

in MATLAB with machine precision 10^{-16}, and terminated when the current iterate satisfies $\|r^{(k)}\|_2/\|r^{(0)}\|_2 < 10^{-6}$, where $r^{(k)}$ is the residual of the kth HSS iteration.

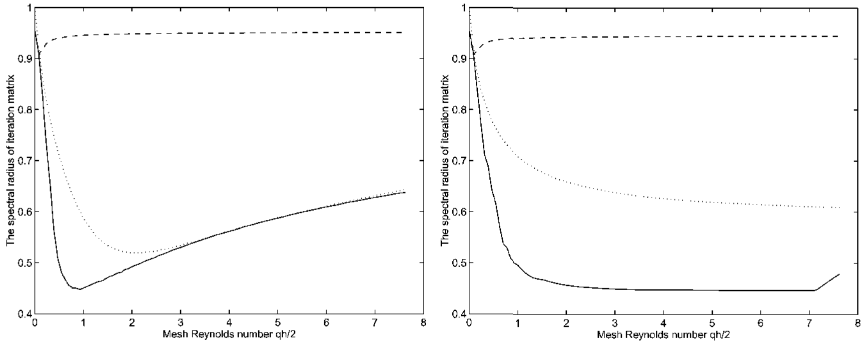
We solve the three-dimensional convection-diffusion equation (4.1) with the homogeneous Dirichlet boundary condition by the HSS iteration. The number n of grid points in all three directions is the same, and the n^3-by-n^3 linear systems with respect to the coefficient matrices $\alpha I + H$ and $\alpha I + S$ are solved efficiently by the sine and



(a) $q = 1$ or $\frac{qh}{2} = 0.0077$ (b) $q = 10$ or $\frac{qh}{2} = 0.0769$

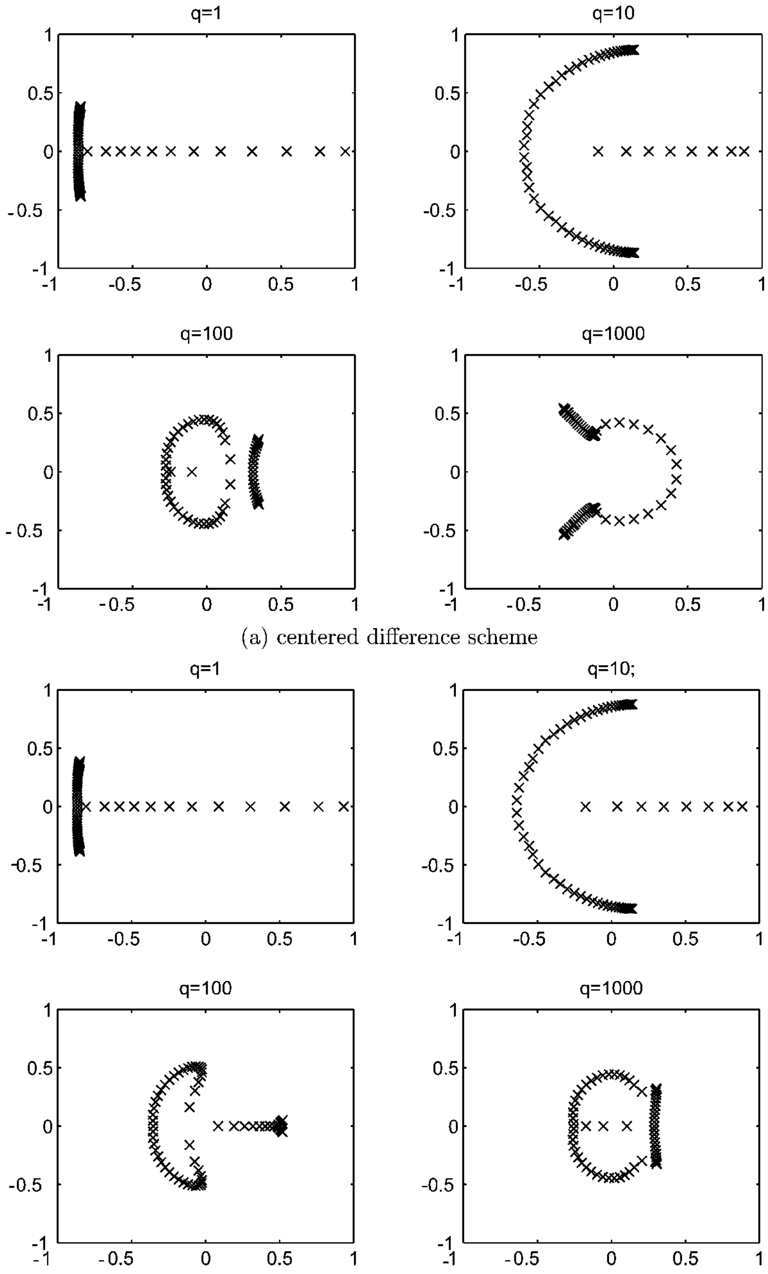
(c) $q = 100$ or $\frac{qh}{2} = 0.7692$ (d) $q = 1000$ or $\frac{qh}{2} = 7.6923$

FIG. 2. *The spectral radius $\rho(M(\alpha))$ of the iteration matrices for different α: "——" and the upper bound $\sigma(\alpha)$ for different α: "- - - -" (upwind difference scheme).*



centered difference scheme upwind difference scheme

FIG. 3. *The spectral radius of the iteration matrices for different q: using α_t "——," α^* in Corollary 2.3 "- - - -," and $\tilde{\alpha} = qh/2$ "........"*



(a) centered difference scheme

(b) upwind difference scheme

FIG. 4. *The eigenvalue distributions of the iteration matrices when $\alpha = \alpha_t$.*

TABLE 3
Number of HSS iterations for the centered (left) and the upwind (right) difference schemes using α^* in Corollary 2.3.

	q			
n	1	10	100	1000
8	34	23	34	35
16	61	42	59	62
32	116	83	117	123
64	234	169	231	244

	q			
n	1	10	100	1000
8	33	22	27	28
16	59	42	52	53
32	114	82	102	109
64	226	158	205	228

TABLE 4
Number of HSS iterations for the centered (left) and the upwind (right) difference schemes using $\tilde{\alpha} = qh/2$.

	q			
n	1	10	100	1000
8	208	28	25	193
16	433	52	22	106
32	844	102	25	76
64	>1000	195	33	66

	q			
n	1	10	100	1000
8	220	40	22	20
16	446	63	26	22
32	852	115	33	25
64	>1000	208	48	33

TABLE 5
Number of HSS iterations for the centered (left) and the upwind (right) difference schemes using the optimal α_t.

	q			
n	1	10	100	1000
8	33	16	20	37
16	58	31	21	48
32	113	57	25	46
64	221	105	33	51

	q			
n	1	10	100	1000
8	33	22	15	15
16	59	35	18	18
32	114	63	26	23
64	204	109	40	33

the modified sine transforms, respectively (cf. Lemma A.1). In Table 3, we list the numerical results for the centered difference and the upwind difference schemes when $q = 1, 10, 100, 1000$. Evidently, when q is large, the cell Reynolds number is also large for each fixed n. Since the eigenvalues of H are known, the parameter α^* can be computed according to Corollary 2.3. We observe that the number of iterations is not only increasing linearly with n but also roughly independent of q as predicted from the convergence analysis in Corollary 2.3. We also test $\tilde{\alpha}$ and the optimal α given in Table 2. In Tables 4 and 5, we present their numbers of HSS iterations. We see from the tables that the number of iterations using the optimal α is less than that using α^* especially when q is large. Moreover, when q is large, the numbers of iterations using the optimal α and $\tilde{\alpha}$ are about the same.

5.3. Results for IHSS iterations. The second test is for the three-dimensional convection-diffusion equation

$$-(u_{xx} + u_{yy} + u_{zz}) + q \exp(x + y + z)(xu_x + yu_y + zu_z) = f(x, y, z)$$

on the unit cube $\Omega = [0, 1] \times [0, 1] \times [0, 1]$, with the homogeneous Dirichlet boundary conditions. For this problem, the n^3-by-n^3 linear systems with respect to the coefficient matrices $\alpha I + H$ and $\alpha I + S$ cannot be solved efficiently by the sine and the modified sine transforms. Therefore, we solve the linear systems with coefficient matrices $\alpha I + H$ iteratively by the preconditioned CG (PCG) method with the sine transform based preconditioner presented in [22], and we solve the linear systems

TABLE 6
Number of IHSS iterations for the centered difference scheme using α^* in Table 3.

	$q = 1$			$q = 10$			$q = 100$			$q = 1000$		
	δ			δ			δ			δ		
n	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7
8	38	37	36	25	21	21	28	28	28	35	39	35
16	72	65	60	45	45	38	55	55	54	59	59	59
32	171	160	142	91	86	84	105	104	103	114	114	114
64	462	339	298	249	210	172	205	202	202	237	233	233

TABLE 7
Number of IHSS iterations for the centered difference scheme using the optimal α_t in Table 5.

	$q = 1$			$q = 10$			$q = 100$			$q = 1000$		
	δ			δ			δ			δ		
n	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7
8	42	41	41	24	24	24	17	17	17	35	35	35
16	78	71	68	42	38	38	34	34	34	43	41	41
32	167	146	136	81	75	73	60	60	60	44	44	44
64	453	355	292	161	150	137	116	116	116	54	54	54

with the coefficient matrix $\alpha I + S$ iteratively by the preconditioned CG for normal equations (PCGNE) method with the modified sine transform based preconditioner given in [19]. This results in the IHSS iteration discussed in section 3. We choose CGNE as the inner solver because it is quite stable, convergent monotonically, and transpose-free. Therefore, as an inner iteration, it could produce an approximate solution satisfying a prescribed rough accuracy in a few iteration steps.

In our computations, the inner PCG and PCGNE iterates are terminated if the current residuals of the inner iterations satisfy

$$(5.1)\qquad \frac{\|p^{(j)}\|_2}{\|r^{(k)}\|_2} \le \max\{0.1\delta^k, 1\times 10^{-7}\} \quad \text{and} \quad \frac{\|q^{(j)}\|_2}{\|r^{(k)}\|_2} \le \max\{0.1\delta^k, 1\times 10^{-6}\}$$

(cf. (3.19) and (3.20) in Theorem 3.3), where $p^{(j)}$ and $q^{(j)}$ are, respectively, the residuals of the jth inner PCG and iterates at the $(k+1)$st outer IHSS iterate, $r^{(k)}$ is the residual of the kth outer IHSS iterate, and δ is a control tolerance. In Tables 6–9, we list numerical results for the centered difference and the upwind difference schemes when $q = 1, 10, 100, 1000$. Since the eigenvalues of H cannot be explicitly obtained, the parameter α^* is not exactly known, and we employ the corresponding parameters used in HSS iterations in Tables 3 and 5 instead.

According to Tables 6–9, the number of IHSS iterations generally increases when δ increases. We see that these increases in IHSS iterations for small q are more significant than those for large q. We also observe that the number of IHSS iterations again increases linearly with n and roughly independent of q. In the tables, the number of iterations using the optimal α is again less than that using α^*, especially when q is large. Moreover, when the optimal α is used, the number of IHSS iterations is about the same for $q = 1, 10, 100, 1000$.

In Table 10, we list the average number of inner PCGNE iterations corresponding to the centered difference scheme. In this case, the Hermitian linear systems with the coefficient matrix $\alpha I + H$ can be solved efficiently by the sine transform. Therefore, we report only the average number of inner PCGNE iterations. In Tables 11 and 12, we report the average number of inner PCG and inner PCGNE iterations corresponding to the upwind difference scheme. It is obvious that, when the control parameter

TABLE 8
Number of IHSS iterations for the upwind difference scheme using α^* *in Table* 3.

	$q=1$			$q=10$			$q=100$			$q=1000$		
	δ			δ			δ			δ		
n	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7
8	33	33	32	24	23	23	27	26	26	28	28	28
16	78	68	68	46	42	42	63	61	60	70	70	69
32	171	155	129	103	87	82	131	127	127	166	164	164
64	460	348	306	263	180	164	248	248	246	370	367	366

TABLE 9
Number of IHSS iterations for the upwind difference scheme using the optimal α_t *in Table* 5.

	$q=1$			$q=10$			$q=100$			$q=1000$		
	δ			δ			δ			δ		
n	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7
8	42	35	35	24	24	22	30	30	30	29	29	29
16	80	70	70	44	44	44	48	48	48	59	59	59
32	165	144	131	83	82	80	85	85	85	95	95	95
64	316	258	239	179	143	141	137	137	137	143	143	143

TABLE 10
Average number of PCGNE iterations for the centered difference scheme using (a) α^* *in Table* 3 *and* (b) *the optimal* α_t *in Table* 5.

	$q=1$			$q=10$			$q=100$			$q=1000$		
	δ			δ			δ			δ		
n	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7
8	1.4	1.8	2.3	3.4	4.0	4.5	6.8	7.1	7.4	6.9	7.0	7.4
16	2.0	2.8	3.5	5.9	7.4	8.6	13.9	14.6	14.9	15.1	15.2	15.2
32	3.5	5.6	6.9	10.2	14.1	17.6	29.0	30.0	30.2	31.7	31.7	31.7
64	7.3	8.5	9.1	22.4	31.2	34.1	60.1	61.9	62.5	55.0	56.8	57.5

(a)

	$q=1$			$q=10$			$q=100$			$q=1000$		
	δ			δ			δ			δ		
n	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7
8	1.5	2.0	2.6	2.8	3.4	4.1	4.9	4.9	5.2	6.7	7.2	7.6
16	2.1	3.0	3.8	4.4	5.8	7.1	7.0	8.4	9.6	12.0	13.7	14.5
32	3.3	5.1	6.6	6.7	9.7	12.6	10.7	14.7	17.7	21.1	23.7	27.0
64	7.1	8.5	9.1	10.8	17.1	21.1	16.7	25.8	33.7	29.1	38.4	45.4

(b)

δ becomes small, the average number of inner PCG and inner PCGNE iterations becomes large. We observe from the tables that the average number of inner PCGNE iterations increases with q, but the average number of inner PCG iterations required is almost nonchanging. The reason is that the parameter q in the convection part does not affect the convergence rate of the Hermitian linear system but does affect the convergence rate of the shifted skew-Hermitian linear system. Moreover, the average number of inner PCGNE iterations using the optimal α_t is less than that of those using α^*, especially when q is large.

Moreover, we find that when δ decreases, the number of inner (PCG or PCGNE) iterations required increases in the numerical tests. In Figure 5, we show an example of this general phenomenon. This is mainly because the inner PCG and the inner PCGNE iterates are terminated if the current residuals of the inner iterations satisfy (5.1). When δ is small, more iterations are required to satisfy the stopping criterion.

Furthermore, instead of PCGNE, we solve the linear systems with the coefficient

TABLE 11

Average number of PCG iterations for the upwind difference scheme using (a) α^* *in Table* 3 *and* (b) *the optimal* α_t *in Table* 5.

	$q = 1$			$q = 10$			$q = 100$			$q = 1000$		
	δ			δ			δ			δ		
n	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7
8	1.5	2.3	3.1	1.3	1.9	2.5	1.4	2.0	2.7	1.4	2.1	2.8
16	2.8	4.3	5.3	2.1	3.1	4.3	2.5	4.0	5.1	2.6	4.4	5.3
32	5.4	6.6	7.0	3.8	5.6	6.4	4.6	6.3	7.0	5.3	6.7	7.2
64	7.9	8.3	8.5	7.1	7.7	8.1	7.0	8.1	8.4	7.6	8.4	8.6

(a)

	$q = 1$			$q = 10$			$q = 100$			$q = 1000$		
	δ			δ			δ			δ		
n	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7
8	1.3	1.6	2.1	1.0	1.5	1.6	1.0	1.6	2.0	1.0	1.6	1.9
16	1.6	2.3	2.6	1.3	1.9	2.3	1.4	2.0	2.4	1.5	2.2	2.5
32	3.0	3.5	3.6	2.2	3.1	3.4	2.2	3.1	3.4	2.3	3.2	3.5
64	4.3	4.6	4.7	3.8	4.3	4.6	3.4	4.3	4.5	3.5	4.3	4.6

(b)

TABLE 12

Average number of PCGNE iterations for the upwind difference scheme using (a) α^* *in Table* 3 *and* (b) *the optimal* α_t *in Table* 5.

	$q = 1$			$q = 10$			$q = 100$			$q = 1000$		
	δ			δ			δ			δ		
n	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7
8	1.4	1.7	2.1	2.7	3.2	3.8	4.6	4.9	5.5	4.7	5.2	6.0
16	2.0	2.8	3.8	4.9	6.3	7.7	10.4	12.5	13.9	11.6	13.8	14.6
32	3.4	5.4	6.5	9.8	13.0	16.0	24.6	28.6	29.8	28.4	29.9	30.5
64	7.3	8.6	9.2	21.9	27.6	32.3	55.8	59.9	61.2	62.5	63.2	63.5

(a)

	$q = 1$			$q = 10$			$q = 100$			$q = 1000$		
	δ			δ			δ			δ		
n	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7
8	1.5	1.9	2.3	2.9	3.4	4.0	3.1	3.9	4.4	3.6	4.2	4.9
16	2.1	2.9	3.9	3.9	5.4	6.7	5.5	7.4	8.8	6.1	7.9	9.9
32	3.3	5.0	6.5	5.9	9.2	11.9	9.1	13.2	16.8	10.8	15.3	19.8
64	5.1	7.1	8.0	11.3	16.1	20.9	15.3	24.0	31.0	19.7	30.5	38.3

(b)

matrix $\alpha I + S$ iteratively by the preconditioned GMRES method (PGMRES [25, 24]) with the modified sine transform based preconditioner given in [19]. Using the same stopping criterion as for the PCGNE, we report the average number of inner PGMRES iterations in Table 13. We see from Tables 10 and 13 that, when q is small, the average number of inner PCGNE iterations is slightly less than that of inner PGMRES iterations. However, when q is large, the average number of inner PGMRES iterations is less than that of inner PCGNE iterations.

6. Conclusion and remarks. For the non-Hermitian positive definite system of linear equations, we present a class of (inexact) splitting iteration methods based on the HSS of the coefficient matrix and the Krylov subspace iterations such as CG and CGNE, and we demonstrate that these methods converge unconditionally to the unique solution of the linear system. In fact, this work presents a general framework of iteration methods for solving this class of system of linear equations. There are several combinations in the framework of iterations. We can solve the Hermitian

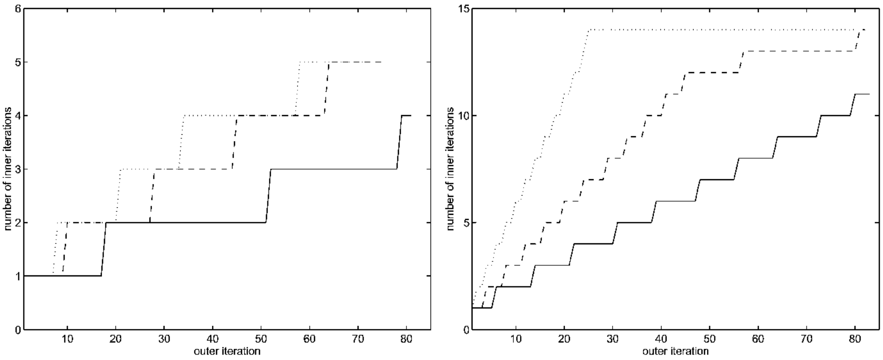


FIG. 5. *The number of inner iterations required for each outer iteration when* $n = 32$ *and* $q = 10$ *in the upwind difference scheme using the optimal* α_t*: (left) PCG inner iterations and (right) PCGNE inner iterations.* —— $(\delta = 0.9)$, - - - - $(\delta = 0.8)$, ... $(\delta = 0.7)$.

TABLE 13

Average number of PGMRES iterations for the centered difference scheme using (a) α^* *in Table* 3 *and* (b) *the optimal* α_t *in Table* 5.

	$q = 1$			$q = 10$			$q = 100$			$q = 1000$		
	δ			δ			δ			δ		
n	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7
8	1.6	1.9	2.3	4.0	4.3	4.9	6.9	7.1	7.4	6.9	7.1	7.5
16	2.2	2.9	3.6	6.4	7.5	8.6	14.5	14.6	14.9	15.2	15.3	15.5
32	3.7	5.7	7.0	10.9	14.5	17.6	27.8	28.1	30.2	28.5	29.7	30.7
64	7.5	9.1	9.2	23.6	31.3	34.1	48.1	51.3	54.5	48.5	51.8	52.5

(a)

	$q = 1$			$q = 10$			$q = 100$			$q = 1000$		
	δ			δ			δ			δ		
n	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7	0.9	0.8	0.7
8	1.8	2.1	2.5	2.8	3.5	4.1	4.9	5.0	5.2	6.8	7.4	7.7
16	2.3	3.0	3.9	4.4	5.8	7.2	7.0	8.5	9.5	12.1	13.9	14.5
32	3.3	5.2	6.7	6.8	9.7	12.7	9.1	12.5	15.1	16.5	18.3	21.0
64	7.3	8.5	9.2	10.8	17.2	21.2	13.3	19.4	24.9	19.9	23.8	33.5

(b)

part exactly or inexactly and the skew-Hermitian part exactly or inexactly. The best choice depends on the structures of the Hermitian and the skew-Hermitian matrices. Convergence theories for the correspondingly resulted exact HSS or IHSS iterations can be established following an analogous analysis to this paper with slight technical modifications.

Moreover, instead of CG and CGNE, we can employ other efficient iterative methods of types of Krylov subspace [24, 7], multigrid, multilevel, classical relaxation, etc. to solve the systems of linear equations with coefficient matrices $\alpha I + H$ and $\alpha I + S$ involved at each step of the HSS iteration. In particular, we mention that, when GMRES is applied to the linear system with coefficient matrix $\alpha I + S$, it automatically reduces to a two-term recurrence process, and its convergence property is dependent only on the eigenvalues, but independent of the eigenvectors, of the matrix $\alpha I + S$.

Appendix. The basic lemma used in the model problem analysis in section 4 is shown in this section.

LEMMA A.1 (see [5, 19]). *The matrix H in (4.3) can be diagonalized by the matrix $F^{(1)} \otimes F^{(1)} \otimes F^{(1)}$. Here $F^{(1)} = ([F^{(1)}]_{j,k})$ is the sine transform matrix defined by*

$$[F^{(1)}]_{j,k} = \sqrt{\frac{2}{n+1}} \sin\left(\frac{jk\pi}{n+1}\right), \quad j,k = 1,2,\dots,n.$$

The corresponding eigenvalues of H are given by

$$\lambda_{j,k,l}(H) = t_1 + (t_2+t_3) \cdot \left[\cos\left(\frac{j\pi}{n+1}\right) + \cos\left(\frac{k\pi}{n+1}\right) + \cos\left(\frac{l\pi}{n+1}\right)\right], \quad j,k,l = 1,2,\dots,n.$$

The matrix S in (4.4) can be diagonalized by the matrix $F^{(2)} \otimes F^{(2)} \otimes F^{(2)}$. Here $F^{(2)} = ([F^{(2)}]_{j,k})$ is the modified sine transform matrix defined by

$$[F^{(2)}]_{j,k} = \sqrt{\frac{2}{n+1}} i^{j+k+1} \sin\left(\frac{jk\pi}{n+1}\right), \quad j,k = 1,2,\dots,n.$$

The corresponding eigenvalues of S are given by

$$\lambda_{j,k,l}(S) = i(t_2-t_3) \cdot \left[\cos\left(\frac{j\pi}{n+1}\right) + \cos\left(\frac{k\pi}{n+1}\right) + \cos\left(\frac{l\pi}{n+1}\right)\right], \quad j,k,l = 1,2,\dots,n.$$

Here i is used to represent the imaginary unit.

Remark. We consider the 2-by-2 matrix

$$A = \begin{pmatrix} 2+2\cos(\pi h) & -qh/2 \\ qh/2 & 2-2\cos(\pi h) \end{pmatrix}$$

as an example to illustrate the use of the iteration parameter $\alpha = \tilde{\alpha} = qh/2$. It is clear that

$$H = \begin{pmatrix} 2+2\cos(\pi h) & 0 \\ 0 & 2-2\cos(\pi h) \end{pmatrix} \quad \text{and} \quad S = \begin{pmatrix} 0 & -qh/2 \\ qh/2 & 0 \end{pmatrix}.$$

We note that $2+2\cos(\pi h)$ and $2-2\cos(\pi h)$ are the largest and the smallest eigenvalues, respectively, of the Hermitian part of the discretization matrix of the differential equation $-u'' + qu' = 0$. In this case, the iteration matrix $M(\alpha)$ of the HSS iteration is similar to the matrix

$$\begin{aligned}\widetilde{M}(\alpha) = &\begin{pmatrix} \alpha-2-2\cos(\pi h) & 0 \\ 0 & \alpha-2+2\cos(\pi h) \end{pmatrix} \\ &\times \begin{pmatrix} \alpha+2+2\cos(\pi h) & 0 \\ 0 & \alpha+2-2\cos(\pi h) \end{pmatrix}^{-1} \times \begin{pmatrix} \alpha & qh/2 \\ -qh/2 & \alpha \end{pmatrix} \\ &\times \begin{pmatrix} \alpha & -qh/2 \\ qh/2 & \alpha \end{pmatrix}^{-1}.\end{aligned}$$

When $\alpha = \tilde{\alpha} = qh/2$, we have

$$\begin{pmatrix} \alpha & qh/2 \\ -qh/2 & \alpha \end{pmatrix} \times \begin{pmatrix} \alpha & -qh/2 \\ qh/2 & \alpha \end{pmatrix}^{-1} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

Then we compute the eigenvalues λ of $\widetilde{M}(\tilde{\alpha})$, and they are given by

$$\pm\sqrt{\frac{(2+2\cos(\pi h)-\frac{qh}{2})(-2+2\cos(\pi h)+\frac{qh}{2})}{(2+2\cos(\pi h)+\frac{qh}{2})(2-2\cos(\pi h)+\frac{qh}{2})}}.$$

By using the series expansion of the above expression in terms of h, we obtain

$$\lambda = \pm\sqrt{\frac{-\pi+\frac{q}{2}}{\pi+\frac{q}{2}}}\cdot\left(1-\frac{qh}{4}+\mathcal{O}(h^2)\right).$$

However, if we use α^* as the iteration parameter, the upper bound $\sigma(\alpha^*)$ of the spectral radius $\rho(M(\alpha^*))$ of the iteration matrix $M(\alpha^*)$ is given by

$$\frac{\sqrt{2+2\cos(\pi h)}-\sqrt{2-2\cos(\pi h)}}{\sqrt{2+2\cos(\pi h)}+\sqrt{2-2\cos(\pi h)}} = 1-\pi h+\mathcal{O}(h^2);$$

see Corollary 2.3. Hence, when $q > 4\pi$, $\rho(M(\tilde{\alpha}))$ is less than $\sigma(\alpha^*)$. From this example, we see that $\tilde{\alpha}$ is a good iteration parameter when q is large. Figure 3 indeed shows that $\tilde{\alpha}$ is close to α_t.

REFERENCES

[1] O. Axelsson, Z.-Z. Bai, and S.-X. Qiu, *A class of nested iteration schemes for linear systems with a coefficient matrix with a dominant positive definite symmetric part*, Numer. Algorithms, to appear.

[2] Z.-Z. Bai, *Sharp error bounds of some Krylov subspace methods for non-Hermitian linear systems*, Appl. Math. Comput., 109 (2000), pp. 273–285.

[3] Z.-Z. Bai, I. Duff, and A. J. Wathen, *A class of incomplete orthogonal factorization methods I: Methods and theories*, BIT, 41 (2001), pp. 53–70.

[4] M. Benzi and D. Szyld, *Existence and uniqueness of splittings of stationary iterative methods with applications to alternating methods*, Numer. Math., 76 (1997), pp. 309–321.

[5] R. H. Chan and M. K. Ng, *Conjugate gradient methods for Toeplitz systems*, SIAM Rev., 38 (1996), pp. 427–482.

[6] W. Cheung and M. K. Ng, *Block-circulant preconditioners for systems arising from discretization of the three-dimensional convection-diffusion equation*, J. Comput. Appl. Math., 140 (2002), pp. 143–158.

[7] P. Concus and G. H. Golub, *A generalized conjugate gradient method for non-symmetric systems of linear equations*, in Computing Methods in Applied Sciences and Engineering, Lecture Notes in Econom. and Math. Systems 134, R. Glowinski and J.R. Lions, eds., Springer-Verlag, Berlin, 1976, pp. 56–65; also available online from http://www-sccm.stanford.edu.

[8] J. Douglas, Jr. and H. H. Rachford, Jr., *Alternating direction methods for three space variables*, Numer. Math., 4 (1956), pp. 41–63.

[9] C. Greif and J. Varah, *Iterative solution of cyclically reduced systems arising from discretization of the three-dimensional convection-diffusion equation*, SIAM J. Sci. Comput., 19 (1998), pp. 1918–1940.

[10] C. Greif and J. Varah, *Block stationary methods for nonsymmetric cyclically reduced systems arising from three-dimensional elliptic equations*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 1038–1059.

[11] M. Eiermann, W. Niethammer, and R. S. Varga, *Acceleration of relaxation methods for non-Hermitian linear systems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 979–991.

[12] H. Elman and G. H. Golub, *Iterative methods for cyclically reduced non-self-adjoint linear systems*, Math. Comput., 54 (1990), pp. 671–700.

[13] H. Elman and G. H. Golub, *Iterative methods for cyclically reduced non-self-adjoint linear systems* II, Math. Comput., 56 (1991), pp. 215–242.

[14] E. Giladi, G. H. Golub, and J. B. Keller, *Inner and outer iterations for the Chebyshev algorithm*, SIAM J. Numer. Anal., 35 (1998), pp. 300–319.

[15] G. H. Golub and D. Vanderstraeten, *On the preconditioning of matrices with a dominant skew-symmetric component*, Numer. Algorithms, 25 (2000), pp. 223–239.

[16] G. H. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, 1996.

[17] G. H. Golub and A. J. Wathen, *An iteration for indefinite systems and its application to the Navier–Stokes equations*, SIAM J. Sci. Comput., 19 (1998), pp. 530–539.

[18] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia, 1997.

[19] L. Hemmingsson and K. Otto, *Analysis of semi-Toeplitz preconditioners for first-order PDEs*, SIAM J. Sci. Comput., 17 (1996), pp. 47–64.

[20] T. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math. Comput., 34 (1980), pp. 473–497.

[21] J. Meijerink and H. Van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comput., 31 (1977), pp. 148–162.

[22] M. K. Ng, *Preconditioning of elliptic problems by approximation in the transform domain*, BIT, 37 (1997), pp. 885–900.

[23] D. W. Peaceman and H. H. Rachford, Jr., *The numerical solution of parabolic and elliptic differential equations*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 28 41.

[24] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.

[25] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[26] C.-L. Wang and Z.-Z. Bai, *Sufficient conditions for the convergent splittings of non-Hermitian positive definite matrices*, Linear Algebra Appl., 330 (2001), pp. 215–218.

[27] O. Widlund, *A Lanczos method for a class of nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 15 (1978), pp. 801–812.

PART III

SOLUTION OF LEAST SQUARES PROBLEMS

This page intentionally left blank

10

COMMENTARY, BY ÅKE BJÖRCK

The method of least squares has been the standard procedure for the analysis of data from the beginning of the 1800s. A famous example of its use is when Gauss successfully predicted the orbit of the asteroid Ceres in 1801. In the simplest case the problem is: given $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$, find a vector $x \in \mathbf{R}^n$ which solves

$$\min_x \|b - Ax\|_2, \tag{10.1}$$

where $\|\cdot\|_2$ denotes the Euclidean norm. A least squares solution x is characterized by $r \perp \mathcal{R}(A)$, where $r = b - Ax$ is the residual and $\mathcal{R}(A)$ the range space of A.

The basic computational tool for solving (10.1) introduced by Gauss was to form and solve the normal equations $A^TAx = A^Tb$ by symmetric Gaussian elimination. After 1924 the Cholesky factorization [4] became the standard solution algorithm. The drawback with these methods is that forming the normal equations may square the condition number of the original problem. In hand computation this can be detected and compensated for by using variable precision. With the start of the computer age in the 1950s it soon became clear that a more stable algorithm was desirable.

Gram–Schmidt orthogonalization methods were used in many early computing centers. The great difference in behavior of the classical and modified Gram–Schmidt was noted by Rice [82], but the numerical properties were not fully understood at the time.

The five papers collected here are all related to various least squares problems. The new, more accurate and efficient algorithms developed by Golub and his coauthors in these papers meant a great leap forward in the ability to treat these problems, and they have found widespread applications.

Numerical methods for solving least squares problems, by Golub [27]

The systematic use of orthogonal transformations to reduce matrices to simpler form was initiated (in the same year!) by Givens [26] and Householder [54]. They had mainly eigenvalue computations in mind, but Householder also briefly discussed least squares problems. In this seminal paper Golub [27] showed how to compute the QR factorization of a rectangular $m \times n$ matrix A by Householder transformations and use it to solve a linear least squares problem. A careful error analysis of these transformations was published by Wilkinson [105] in 1965.

Householder QR has been aptly characterized by Trefethen and Bau [90] as being orthogonal triangularization, whereas Gram–Schmidt is triangular orthogonalization. Golub's paper came as a revelation at a time when the stability of Gram–Schmidt and the need to reorthogonalize were current hot topics. Golub's algorithm was elegant and provenly backward stable. The operation count and storage requirement was lower than for MGS (the difference is small if $m \gg n$). A factored representation of the full square orthogonal matrix Q and the upper triangular factor R could both be stored in an $m \times n$ array.

An Algol procedure implementing the Golub–Householder least squares algorithm, and incorporating simple iterative refinement, was published in a companion paper [12]. Later this appeared as Contribution I/8 in the famous Handbook [107]. This was adopted in the LINPACK and later in the LAPACK libraries.

Two years after the Golub paper appeared, Björck [6] proved that the MGS algorithm is stable for solving the least squares problem, even without reorthogonalization. The final rigorous proof of backward stability of MGS both for least squares and minimum norm problems came even later; see [10].

Several comparisons between between the accuracy of the MGS and Golub algorithms for the least squares problem were published in late 1960s and early 1970s. According to evaluations by Jordan [58] at the Los Alamos Scientific Laboratory and Wampler [103] at the National Bureau of Standards, MGS seemed to be slightly more accurate than the Golub–Householder algorithm. In 1970 Peters and Wilkinson [79] wrote:

> *Evidence is accumulating that the modified Gram–Schmidt method gives better results than Householder.*

However, the Golub–Householder procedure has the edge over Gram–Schmidt with respect to storage and operation count. The greater flexibility and simpler stability analysis of this method has also contributed to the fact that it has prevailed as the method of choice for solving linear least squares problems. Today the MGS algorithm for least squares problems is much less known and used.

Although Golub's paper [27] is only 11 pages long, it outlines several further developments, which we now briefly discuss.

- The implementation of two different *column pivoting strategies* is described. Equivalent schemes had previously been used before with the Cholesky and modified Gram–Schmidt algorithms. In the first strategy successive diagonal elements are maximized at each step. This is appropriate when treating (nearly) rank-deficient problems. The second strategy is appropriate when there is one vector b and one wishes to express it in as few columns of A as possible. This is related to stepwise multivariate regression in statistics, A detailed solution of this problem based on Golub's paper and including an Algol program is given in an unpublished report by Eldén [20].
- *Iterative refinement* is suggested as a way to increase the accuracy of the computed solution. This topic was further investigated by Golub and Wilkinson in [45] and [46]. Their scheme did not work as well as expected

for problems with a large residual. The authors made the (at the time) surprising discovery:

> *We conclude that although the use of the orthogonal transformations avoid some of the ill effects inherent in the use of the normal equations the value of $\kappa^2(A)$ is still relevant to some extent.*

It was soon discovered that a more satisfactory refinement scheme could be obtained by refining the solution to the augmented system satisfied by the solution and its residual. This idea was communicated to the author during a visit by Golub in 1966 to KTH, Stockholm. The modified refinement scheme converges with a rate that is always proportional to $\kappa(A)$; Björck [5]. An Algol procedure implementing the Golub–Householder algorithm for the constrained linear least squares problem

$$\min_x \|b_2 - A_2 x\|, \quad \text{subject to} \quad A_1 x = b_1$$

and incorporating the modified iterative refinement appeared in the joint paper [8].

- An implementation of a scheme for *iterative regularization* by Riley [83] (also known as proximal point method) using QR factorization is described and analyzed. It has the important property that the set of solutions that can be reconstructed with optimal accuracy increases with each iteration; see [50, Section 5.2]. A similar procedure is later discussed from a different point of view by Rutishauser [85].
- Finally, the *updating of the QR factorization* when rows are added was treated and algorithms for least squares *problems with linear equality constraints* discussed. Methods for modifying matrix factorizations were later treated in great detail and generality by Golub *et al.* in [25].

The applications of the Golub–Householder QR algorithm since it appeared more than 40 years ago have been numerous. Much of the algorithmic development has been driven by needs in different application areas such as statistics, optimization, signal processing, and control theory. Two fields of applications treated early by Golub and coauthors are statistical computations [29; 42] and quadratic programming [41].

A remarkable fact, shown by by Powell and Reid [80], is that with row and column pivoting the Golub–Householder method is stable also for *weighted least squares problems.* Cox and Higham [17] prove the important result that this remains true using column pivoting only, provided that the rows of (A, b) are pre-sorted by decreasing row norms. This allows for the use of standard QR subroutines which give much faster implementations.

Extensions of the Golub–Householder algorithm to rank deficient systems were given by Hanson and Lawson in [53] and later in the monograph [65]. Among these can be mentioned the complete orthogonal decomposition

$$A = (U_1 \; U_2) \begin{pmatrix} T & 0 \\ 0 & 0 \end{pmatrix} (V_1 \; V_2)^T,$$

where U and V are orthogonal and T upper (or lower) triangular. This decomposition can be updated much more cheaply than the singular value decomposition; see Stewart [88]. It has therefore become an important tool in applications where new data are added with time, e.g., in signal processing and control.

Kahan [59] gave an example showing that standard column pivoting will not always reveal the numerical rank of a matrix. This started efforts to develop algorithms for rank-revealing QR factorizations and subset selection. Heuristic techniques are necessary since finding the optimal selection is an NP-complete problem. The first improvement on the column pivoting scheme was given in the report by Golub, Klema, and Stewart [33]. This work was refined and extended by Tony Chan [14]. Important theoretical results were given by C. T. Pan and P. T. P. Tang [77]. Chandrasekaran and Ipsen [16] derive a selection of heuristic schemes that are shown to work well almost always. A survey of applications is found in [15].

Applying the Golub–Householder QR factorization to a banded least squares problem of size $m \times n$ with bandwidth w can take up to $O(mnw)$ flops. Reid [81] showed that by ordering and blocking the rows the operation count can be reduced to at most $2(m + 3n/2)w(w + 1)$ flops. This is about twice as much as for computing R by the Cholesky algorithm, as in the dense case. In the last decade algorithms for general sparse QR factorization have been developed; see the survey of algorithms and available software by Matstoms [70; 71].

Generalized cross-validation as a method for choosing a good ridge parameter, by Golub, Heath, and Wahba [31]

Generalized cross-validation (GCV), due to Wahba [102], is a rule for choosing the parameter λ in the ridge estimate

$$\beta(\lambda) = (X^T X + n\lambda I)^{-1} X^T y, \tag{10.2}$$

for the standard regression model

$$y = X\beta + \epsilon, \quad X \in \mathbf{R}^{n \times p},$$

where ϵ is random with $E\epsilon = 0$, $E\epsilon\epsilon^T = \sigma^2 I$. The estimate is taken as the minimizer of

$$V(\lambda) = \frac{\frac{1}{n}\left\|(I - A(\lambda))y\right\|_2^2}{\frac{1}{n}\Big[\operatorname{trace}(I - A(\lambda))\Big]^2}, \quad A(\lambda) = X(X^T X + \lambda I)^{-1} X^T.$$

and does not require an estimate σ^2. This makes GCV convenient to use when the number of degrees of freedom for estimating σ^2 is small, or even in some cases when the real model involves more than n parameters.

In this paper the GCV estimate is derived as a rotation-invariant version of ordinary cross-validation and it is argued why it should be generally superior. Statistical properties of the GCV estimate are examined using the singular value decomposition $X = U\Sigma V^T$. The use of GCV in subset selection, truncated singular value methods, and more general model building is also discussed.

An attractive feature of GCV is that the expected value of the GCV value can be shown to coincide with the optimal regularization parameter when the number of data points goes to infinity; for details see [67; 68]. The paper gives numerical results comparing GCV with ordinary cross-validation, the range risk estimate, and maximum likelihood estimate.

For a fixed value of λ the estimate $\beta(\lambda)$ in (10.2) is the minimizer of

$$\frac{1}{n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2,$$

which is a regularized solution. The calculation of $V(\lambda)$ and $\beta(\lambda)$ can be done in $O(p)$ operations from the SVD of X; see [30]. Eldén [21] shows how the GCV estimate can be efficiently computed from the cheaper bidiagonal decomposition of X and notes that cancellation can be avoided by basing the computations on the identity

$$I - A(\lambda) = I - X(X^TX + \lambda I)^{-1}X^T = \lambda(XX^T + \lambda I)^{-1}.$$

It has been observed that the graph of the GCV functional $V(\lambda)$ often is very flat with many tiny wiggles near the optimal value. This makes it sometimes difficult to determine the minimum. However, this is a property shared by alternative rules for choosing λ from the data. An up-to-date discussion is given in Hansen [51].

In [47] are mentioned applications of GCV in remote sensing problems, multivariate smoothing spline regression, generalized likelihood estimation, penalized log-density and log-hazard estimations.

According to the survey by Hanke and Hansen [50] from 1993, GCV is the most widely used rule for choosing the regularization parameter in ill-posed problems. The rigorous analysis of such problems started in the early 1960s with the introduction of regularization methods by Tikhonov [89] in 1963. Interest in this field has been rapidly growing in science and industry. Application areas include global scale weather prediction, astronomy and chemistry. Recent applications include image processing [73; 74] and wavelet thresholding for noise reduction [57].

GCV is used together with Golub–Kahan bidiagonalization in [7] for regularization of large scale problems. An iterative method for approximating the GCV function for large scale problems is developed by Golub and von Matt [35]. Their approach is based on the Lanczos process and Gauss quadrature with a stochastic trace estimator by Hutchinson [56].

The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate, by Golub and Pereyra [37]

In many nonlinear least squares problems it is advantageous to separate the parameters into two sets. Suppose that data (y_i, t_i), $i = 1 : m$, are given and we want to determine parameters α, a that minimize the nonlinear functional $\|r(\alpha, a)\|_2^2$, where the ith component of the residual is

$$r_i(\alpha, a) = y_i - \sum_{i=1}^{n} a_i \phi_j(\alpha; t_i).$$

Typical examples of nonlinear functions $\phi_j(\alpha; t_i)$ are exponential or rational functions $\phi_j = e^{\alpha_j t}$, $\phi_j = 1/(t - \alpha_j)$. Since a is a vector of linear parameters, the nonlinear functional to be minimized can be written in the form

$$\|r(\alpha, a)\|_2^2 = \|y - \Phi(\alpha)a\|_2^2, \tag{10.3}$$

where $\Phi(\alpha)$ is a matrix whose elements in the jth column equal $\phi_j(\alpha; t_i)$. For any fixed values of the nonlinear parameters α the optimal value of a can be expressed as $a = \Phi(\alpha)^+ y$, where $\Phi(\alpha)^+$ denotes the pseudo-inverse of $\Phi(\alpha)$. This allows the linear parameters to be eliminated in (10.3). The original minimization problem can then be cast in the form

$$\min_{\alpha} \|(I - \Phi(\alpha)\Phi(\alpha)^+)y\|_2^2, \tag{10.4}$$

where $P_{\Phi(\alpha)} = \Phi(\alpha)\Phi(\alpha)^+$ is the orthogonal projector onto the column space of $\Phi(\alpha)$. This is a pure nonlinear problem of reduced dimension. The modified functional to be minimized in (10.4) is known as a variable projection functional. The variable projection method consists of solving (10.4), for example by a Gauss–Newton–Marquardt method, obtaining the optimal vector α. The linear parameters are then computed from $a = \Phi(\alpha)^+ y$.

This paper generalizes earlier work by Scolnik [86] and Guttman *et al.* [49]. Scolnik's original work dealt with the case in which the nonlinear functions depended on one variable each and were of exponential type.

In the variable projection algorithm a formula for the derivative of an orthogonal projection matrix is needed. Let the matrix $A(\alpha)$ be a function of the scalar variable α and have local constant rank. Let A^- denote any symmetric generalized inverse of $A = A(\alpha)$. Then $P_{\mathcal{R}(A)} = AA^-$ is the orthogonal projector onto the column space $\mathcal{R}(A)$ of A. The following fundamental formula is derived

$$\frac{d}{d\alpha} P_{\mathcal{R}(A)} = P^{\perp}_{\mathcal{R}(A)} \frac{dA}{d\alpha} A^- + (A^-)^T \frac{dA^T}{d\alpha} P^{\perp}_{\mathcal{R}(A)}. \tag{10.5}$$

In a digression from the main theme of the paper a related formula for the derivative of the pseudo-inverse of a matrix $A(\alpha)$ is given in the paper:

$$\frac{dA^+}{d\alpha} = -A^+ \frac{dA}{d\alpha} A^+ + (A^T A)^+ \frac{dA^T}{d\alpha} P^{\perp}_{\mathcal{R}(A^T)} + P^{\perp}_{\mathcal{R}(A)} \frac{dA^T}{d\alpha} (AA^T)^+.$$

(This formula actually had appeared, albeit a bit hidden, in Wedin [104, p. 21].) Selected results from the paper were presented at the Advanced Seminar on Generalized Inverses and Applications, October 8–10, 1973, University of Wisconsin, Madison; see [38].

The paper contains a detailed discussion of the implementation of a Gauss–Newton–Marquardt algorithm for solving (10.4) using the techniques for solving linear least squares developed by Golub in [27]. In the Stanford report [36] preceding the published paper there is included a carefully written Fortran implementation called VARPRO. This was instrumental in the quick adaption of the variable projection method and partly explains the paper's big impact.

Golub and LeVeque [34] extended the VARPRO algorithm to the case when several data sets are to be fitted to the model with the same nonlinear parameter vector; see also Kaufman and Sylvester [62].

The variable projection approach not only reduces the dimension of the parameter space but also leads to a better conditioned problem. Krogh [64] gives a slightly different implementation of the the variable projection algorithm. He notes that it solved several problems at JPL, which could not be solved using the old nonlinear least squares approach.

An important improvement of the algorithm was introduced by Kaufman in [60]. The jth column of the Jacobian of the reduced problem can be written

$$J = -\left[P^{\perp}_{\mathcal{R}(\Phi)} \frac{d\Phi}{d\alpha_j} \Phi^{-} + (\Phi^{-})^T \frac{d\Phi^T}{d\alpha_j} P^{\perp}_{\mathcal{R}(\Phi)} \right] y.$$

Kaufman's simplification consists of using an approximate Jacobian obtained by dropping the second term in this formula. The effect is to reduce the work per iteration at the cost of marginally increasing the number of iterations. Kaufman's simplifications were generalized to separable problems with constraints in [61], with applications, e.g., in fitting data by splines with free knots. More examples from system identification and nonlinear data fitting are discussed in [63].

Ruhe and Wedin [84] consider separation of variables in more general nonlinear least squares problems. They point out that separation may be of advantage not only when some variables occur linearly, but also when the Jacobian of the full nonlinear problem is structured. An example is the block angular structure that occurs in orthogonal regression. They carefully analyze the following algorithms: Algorithm G: the unseparated Gauss–Newton method; Algorithm I: the original variable projection algorithm; Algorithm II; Kaufman's variant. They find that the asymptotic convergence rate of Algorithm G is always the slowest. Comparing Algorithms I and II, either may be faster depending on the data, but the difference is small. The difference in work per step is rather small, except in the case that the derivative $d\Phi/d\alpha$ has a complicated structure, when Algorithm I can be considerably slower. These theoretical results are supported by numerical tests.

The program VARPRO was later modified by John Bolstad, who improved the documentation, included the modification of Kaufman and added the calculation of the covariance matrix. LeVeque later wrote a version called VARP2 which handles multiple right-hand sides. Both VARPRO and VARP2 are available in the public domain from Netlib (www.netlib.org/opt).

In a topical review Golub and Pereyra [39] survey 30 years of developments and applications of the variable projection algorithm for solving nonlinear least squares problems. They give a historical account of the basic theory of variable projection methods and survey various computer implementations. A list of 144 references is included, of which a large number are from the last decade. Applications in electrical engineering, medical and biological engineering, chemistry, robotics and environmental sciences are among those listed. A recent application to modelling and control of bilinear systems is described in [19]. This survey gives a striking account of the big impact of this paper on scientific computing.

Singular value decomposition and least squares solution, by Golub and Reinsch [40]

The singular value decomposition (SVD) of a matrix $A \in \mathbf{C}^{m\times n}$ is

$$A = U\Sigma V^H, \quad \Sigma = \mathrm{diag}(\sigma_k), \tag{10.6}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$, are the singular values of A. The corresponding left and right singular vectors are the columns of the square unitary matrices U and V. The SVD was independently developed by E. Beltrami 1873 and Camille Jordan 1874; see [87] for an interesting account of the early history of the SVD.

For a long time no stable algorithm for computing the SVD was known. First attempts used the eigenvalue decomposition of the matrices

$$A^H A = V^H \mathrm{diag}(\sigma_k^2) V \text{ and } AA^H = U^H \mathrm{diag}(\sigma_k^2) U;$$

see, e.g, Householder [55, p. 3]. However, with this approach one cannot get accurate values for the smaller singular values, which usually are the most important ones. This situation changed completely with the publication of the Algol program in this paper. The SVD is now regarded as one of the most fundamental tools in computational linear algebra and has found many uses in scientific and engineering application.

In the early 1960s Francis had developed the implicit-shift QR algorithm for computing the eigenvalue decomposition of a matrix. For the Hermitian case Wilkinson [106] proved the global convergence of the QR algorithm with a special choice of shifts. The Handbook contribution of Bowdler, Martin, Reinsch and Wilkinson [11] is based on these results.

An important step towards a stable algorithm for the SVD came in 1965 with the paper by Golub and Kahan [32] of this selected works. Here it was shown how to reduce any $m \times n$ complex matrix A to real bidiagonal form J using (unitary) Householder transformations from left and right. By an observation of Lanczos, the eigenvalues of the real symmetric matrix

$$K = \begin{pmatrix} 0 & J \\ J^T & 0 \end{pmatrix}$$

then are $\pm\sigma_k$. A permutation of the rows and columns of K brings this matrix into symmetric tridiagonal form with a zero diagonal and the QR algorithm can now be applied to get the SVD of K. The left and right singular vectors are retrieved from the eigenvectors of K and the initial transformation.

There still remained the problem of how to avoid the duplication of work caused by the each singular value and singular vector appearing twice in the eigendecomposition of K. A solution is outlined by Golub [28] and further developed in this paper. An implicitly shifted QR algorithm on J^TJ is used but all transformation are applied just to J using a "chasing" technique. By using the "Wilkinson" shift, global convergence is ensured.

Two separate Algol procedures are given in the paper. Procedure *svd* computes the SVD of a real matrix A. The computation of the singular vectors in U and V is optional. Procedure *minfit* computes matrices Σ, V, and C such that for given real $m \times n$ matrix A and $m \times p$ matrix B

$$U^TAV = \Sigma, \qquad U^TB = C.$$

Then $X = V\Sigma^+C$ is the solution of minimum Frobenius norm to the problem of minimizing $\|AX - B\|_F$. Businger and Golub [13] give a similar algorithm for the SVD of a complex matrix A.

Applications of the SVD discussed in this paper are the solution of homogeneous equations, computing the least squares solutions of minimum length, and total least squares problems. The SVD plays a key role in solving a number of least squares problems and is the most versatile decomposition for treating rank-deficient and ill-conditioned least squares problem.

Formally the rank of a matrix A is equal to the number k of positive singular values of A: $\sigma_k > 0$ and $\sigma_{k+1} = 0$. This definition cannot be used in numerical computations, because the rank is a discontinuous function of A; infinitely small changes in its entries can make the rank of A trivial, viz. $\min(m, n)$. A more meaningful question is: what is the distance of A to the set of $\mathcal{M}_k$ of all $m \times n$ matrices of rank $\leq k$? In the metric defined by the spectral norm $\|A\|_2 = \sigma_1$ or the Euclidean or Frobenius norm $\|A\|_F = \sqrt{\sum |a_{ij}|^2} = \sqrt{\sum \sigma_k^2}$ the answer is

$$\begin{aligned}
\operatorname{dist}(A, \mathcal{M}_k) &:= \min\{\|A - X\|_2 : X \in \mathcal{M}_k\} = \sigma_{k+1}, \\
\operatorname{dist}(A, \mathcal{M}_F) &:= \min\{\|A - X\|_2 : X \in \mathcal{M}_k\} = \Big(\sum_{j \geq k+1} \sigma_j^2\Big)^{1/2},
\end{aligned}$$

for $k < \min(m, n)$. Due to the triangle inequality, these distances are continuous functions of A and the SVD gives accurate and reliable numerical values. The minimizer is

$$\hat{X} = U_1\Sigma_1V_1^T, \quad \Sigma_1 = \operatorname{diag}(\sigma_1, \ldots, \sigma_k),$$

where $U_1 = (u_1, \ldots, u_k)$, $V_1 = (v_1, \ldots, v_k)$. The matrix $\hat{X}$ approximates A with a sum of k rank one matrices. In statistical applications (factor analysis), where A is a table of scores, these are called the factors and $A - \hat{X}$ is assumed to be random noise. Several related matrix approximation problems, whose solution can be expressed in terms of the SVD, are discussed in Golub [28].

A good example of how the SVD can be used to extract the linearly independent information in A to arrive at a more well-conditioned problem is given by its application to ill-posed linear systems $Ax = b$. Here an alternative to Tikhonov regularization is to use a truncated SVD (TSVD) solution, $x = V_1 \Sigma_1^{-1} U_1^T b$, for a suitably chosen value of k. This solution is the least squares solution restricted to the subspace $\mathcal{R}(V_1)$.

Some improvements in accuracy and efficiency have been achieved by modifications to the original QR-SVD algorithm. A survey of direct methods for computing the SVD is given in [2, Section 6.2]. Demmel and Kahan [18] showed that the singular values of a bidiagonal matrix are defined to full *relative precision independent of their magnitude.* They proceed to introduce a zero-shift QR algorithm that can give full relative accuracy also in small singular values.

Gu and Eisenstat [48] give a divide-and-conquer algorithm for finding the SVD of a bidiagonal matrix. This can speed up the bidiagonal part of the SVD algorithm by a large factor. As implemented in the LAPACK subroutine *xGESDD* it is faster than the QR algorithm already for bidiagonal matrices larger than about 25 × 25. The speedup increases with the dimension and is about 9 for $n = 400$.

The differential *qd* algorithm given by Fernando and Parlett [23] and von Matt [72] is potentially even faster than the divide-and-conquer algorithm and gives highly accurate singular values.

When only the singular values in a given interval and the corresponding singular vectors are wanted then bisection and inverse iteration may be preferable to use. Bisection algorithms rely on an accurate algorithm for counting singular values of a bidiagonal matrix analyzed in [22].

Thirty years ago computing the SVD was considered an expensive option. The rapid hardware development since 1970 has changed this—or rather, the definition of "a large problem" has changed. Subroutines for computing the SVD for dense rectangular matrices are available in most mathematical software libraries; see [51, Table 2.1] for a list. Typically, on a modern PC, the MATLAB command `[U,S,V] = svd(A)` computes the SVD of a matrix of dimension 1000 × 500 in less than ten seconds. However, since neither the bidiagonal factorization nor the SVD can be cheaply updated, an option like the ULV decomposition might be preferable when a sequence of modified problems is to be solved.

An analysis of the total least squares problem, by Golub and Van Loan [43]

In Errors-in-Variable (EIV) modeling a linear model is sought that explains a given set of multidimensional data points. The idea is to modify all data points (vectors) in such a way that some norm of the modification is minimized, subject to the constraint that the modified vectors satisfy a linear relation. The origin can be traced back to the beginning of the previous century. Then it was used mainly in statistical and psychometric research, where it is also known as latent root regression. Its relation to principal component analysis and factor analysis also was investigated long ago. The special case of orthogonal regression was studied already in 1878 by Adcock [1]; see also Pearson [78]. More on the history of EIV modeling can be found in [97].

The EIV technique did not become popular in computational mathematics and engineering until after the publication of this paper, with its efficient and numerically robust SVD-based algorithm. The apt name Total Least Squares (TLS), which was coined in this paper, may have contributed to its popularity! In the standard linear least squares problem one assumes that the data vector $b \in \mathbf{R}^m$ is related to the unknown parameter vector $x \in \mathbf{R}^n$ by a linear relation $Ax = b + \epsilon$, where the matrix A is error free and ϵ is a vector of uncorrelated random errors with zero means and the same variance. These assumptions are frequently unrealistic since sampling or modeling errors often affect the matrix A. This motivates the TLS model

$$\min \|(E,\ r)\|_F, \qquad (A+E)x = b + r, \tag{10.7}$$

where $\|\cdot\|_F$ denotes the Frobenius matrix norm. In this model random errors are allowed also in the data matrix A. The rows of the error matrix $(E,\ r)$ are assumed to be independently and identically distributed with zero mean and the same variance. In typical applications, gains of 10–15% in accuracy can be obtained by using a TLS model instead of standard least squares methods.

Golub and Van Loan analyze the TLS problem in terms of the singular value decomposition. A slightly more general weighted TLS problem

$$\min \|D(E,\ r)T\|_F, \qquad (A+E)x = b + r,$$

is considered, where D and T are diagonal weighting matrices. It is shown that if the smallest singular value of the matrix $D(A,\ b)T$ is unique, then a unique solution exists and can be expressed in terms of the corresponding right singular vector. This follows by noting that the solution $(E,\ r)$ equals the perturbation of minimum norm $\|D(A,\ b)T\|_F$ that lowers the rank of the matrix $(A,\ b)$. The sensitivity of the TLS problem and its relationship to the standard least squares problem is investigated in the paper. Algorithmic details are discussed, and a generalization to the mixed LS-TLS problem when some columns are error free is announced; see [76].

An algorithm for the TLS problem $AX \approx B$ having multiple right-hand side vectors is given in [44, Sec. 12.3]. A complication of the TLS problem is that a solution may not exist. The treatment of such nongeneric problems is discussed in [98]. Many other theoretical and computational aspects of the TLS problems are treated in detail by Van Huffel in [91; 92] and in a series of papers coauthored by Vandewalle [95; 99; 96; 97]. The state of the art in TLS modeling in the early 1990s is summarized in the excellent and comprehensive monograph by Van Huffel and Vandewalle [98]. Regularization algorithms based on TLS are introduced in [52] and [24].

A TLS algorithm based on rank-revealing two-sided orthogonal factorization is developed in [101]. The solution of a sequence of TLS problems, where rows are added and deleted, can be done more effectively with such factorizations; see Barlow and Yoon [3]. Algorithms based on inverse iteration and preconditioned conjugate gradient methods for large scale TLS problems are studied in [9].

Paige and Strakos [75] have studied a scaled linear least squares model which unifies the treatment of least squares, total least squares, and data least squares. (In the last model only perturbations in A are allowed.) It is shown how a generic problem of minimal dimension can always be extracted from the upper bidiagonal decomposition of the augmented matrix $(b,\ A)$.

Several other extensions to the original TLS problem have been suggested. The more difficult restricted TLS problem has been taken up in [100]. Structured TLS problems have attracted a lot of attention in the last decade; see [66; 69]. In particular Toeplitz and Hankel structured problems arise in many signal processing and system identification applications. Properties of EIV models in the presence of colored and non-Gaussian noise have been investigated.

TLS modeling remains a broad and active field. Four International Workshops on Total Least Squares and Errors-in-Variables modeling have been held in August 1991, 1996, 2001 and 2006 in Leuven, Belgium; see the proceedings from the 1996 and 2001 workshop [93; 94]. The 4th workshop featured nonlinear and structured TLS modeling, statistical estimators, and TLS modeling with bounded uncertainties.

Summary

The impact of the five papers reprinted in this section is hard to overstate. They are still worth careful reading both for their lucid style and for the abundance of ideas they contain, all of which may not yet have been followed up. It is hard to imagine how we managed before backward stable algorithms for the linear least squares problem and the SVD were available. These algorithms are now heavily used not only by numerical analysts but have spread to statistics, control theory, signal processing and numerous other application areas.

REFERENCES

1. R. J. Adcock. A problem in least squares. *The Analyst*, **5**, 53–54 (1878).
2. Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors. *Templates for the Solution of Algebraic Eigenvalue Problems. A Practical Guide.* SIAM, Philadelphia, PA (2000).
3. J. L. Barlow and P. A. Yoon. Solving recursive TLS problems using the rank-revealing ULV decomposition. In *Recent Advances in Total Least Squares Techniques and Errors-in-Variable Modeling.* Sabine Van Huffel (ed.), SIAM, Philadelphia, PA, pp. 117–126 (1997).
4. Commandant Benoit. Sur la méthode de résolution des équations normales, etc. (Procédés du commandant Cholesky). *Bull. Géodésique*, **2**, 67–77 (1924).
5. Å. Björck. Iterative refinement of linear least squares solutions I. *BIT*, **7**, 257–278 (1967).
6. Å. Björck. Solving linear least squares problems by Gram–Schmidt orthogonalization. *BIT*, **7**, 1–21 (1967).
7. Å. Björck. A bidiagonalization algorithm for solving ill-posed systems of linear equations. *BIT*, **28**, 659–670 (1988).
8. Å. Björck and G. H. Golub. Iterative refinement of linear least squares solution by Householder transformation. *BIT*, **7**, 322–337 (1967).
9. Å. Björck, P. Heggernes, and P. Matstoms. Methods for large scale total least squares problems. *SIAM J. Matrix Anal. Appl.*, **22**, 413–429 (2000).
10. Å. Björck and C. C. Paige. Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm. *SIAM J. Matrix Anal. Appl.*, **13**, 176–190 (1992).
11. H. Bowdler, R. S. Martin, C. Reinsch, and J. H. Wilkinson. The *QR* and *QL* algorithms for symmetric matrices. *Numer. Math.*, **11**, 293–306 (1968).
12. P. Businger and G. H. Golub. Linear least squares solutions by Householder transformations. *Numer. Math.*, **7**, 269–276 (1965).
13. P. Businger and G. H. Golub. Algorithm 358: Singular value decomposition of a complex matrix. *Comm. ACM*, **12**, 564–565 (1969).
14. T. F. Chan. Rank revealing QR-factorizations. *Linear Algebra Appl.*, **88/89**, 67–82 (1987).
15. T. F. Chan and P. C. Hansen. Some applications of the rank revealing QR factorization. *SIAM J. Sci. Statist. Comput.*, **13**, 727–741 (1992).
16. S. Chandrasekaran and I. C. F. Ipsen. On rank-revealing factorizations. *SIAM J. Matrix Anal. Appl.*, **15**, 592–622 (1994).
17. A. J. Cox and N. J. Higham. Stability of Householder QR factorization for weighted least squares problems. In *Numerical Analysis 1997: Proceedings of the 17th Dundee Biennal Conference,*. D. F. Griffiths, D. J. Higham,

and G. A. Watson (eds.), Pitman Research Notes Math. Ser. 380, Addison Wesley Longman, Harlow, UK, pp. 57–73 (1998).

18. J. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Statist. Comput.*, **11**, 873–912 (1990).
19. Mats Ekman. Modelling and control of bilinear systems. Application to the activated sludge process. *Uppsala dissertations, Faculty of Science and Technology*, Box 337, SE 751 05 Uppsala, Sweden (2005).
20. L. Eldén. Stepwise regression analysis with orthogonal transformations. Tech. Report LiTH-MAT-R-72-2, Department of Mathematics, Linköping University, Sweden (1972).
21. L. Eldén. A note on the computation of the generalized cross-validation function for ill-conditioned least squares problems. *BIT*, **24**, 467–472 (1984).
22. K. V. Fernando. Accurately counting singular values of bidiagonal matrices and eigenvalues of skew-symmetric tridiagonal matrices. *SIAM J. Matrix Anal. Appl.*, **20**(2), 373–399 (1998).
23. K. V. Fernando and B. N. Parlett. Accurate singular values and differential QD algorithms. *Numer. Math.*, **67**, 191–229 (1994).
24. R. D. Fierro, G. H. Golub, P. C. Hansen, and D. P. O'Leary. Regularization by truncated total least squares. *SIAM J. Sci. Comput.*, **18**(4), 1223–1241 (1997).

25*. P. E. Gill, G. H. Golub, W. Murray, and M.A. Saunders. Methods for modifying matrix factorizations. *Math. Comp*, **28**, 505–535 (1974).

26. W. Givens. Computation of plane unitary rotations transforming a general matrix to triangular form. *SIAM J. Appl. Math.*, **6**, 26–50 (1958).

27*. G. H. Golub. Numerical methods for solving least squares problems. *Numer. Math.*, **7**, 206–216 (1965).

28. G. H. Golub. Least squares, singular values and matrix approximations. *Aplikace Matematiky*, **13**, 44–51 (1968).
29. G. H. Golub. Matrix decompositions and statistical computation. In *Statistical Computation.* R.C. Milton and J. A. Nelder (ed.), Academic Press, New York, pp. 365–397 (1969).

30*. G. H. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, **15**, 318–344 (1973).

31*. G. H. Golub, M. T. Heath, and G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, **21**, 215–223 (1979).

32*. G. H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal. Ser. B*, **2**, 205–224 (1965).

33. G. H. Golub, V. Klema, and G. W. Stewart. Rank degeneracy and least squares problems. Tech. Report STAN-CS-76-559, August 1976, Computer Science Department, Stanford University, CA (1976).
34. G. H. Golub and R. J. LeVeque. Extensions and uses of the variable projection algorithm for solving nonlinear least squares problems. In *Proceed-*

ings of the 1979 Army Numerical Analysis and Computers Conference, White Sands Missile Range, White Sands, NM, ARO Report 79-3, pp. 1–12 (1979).

35. G. H. Golub and U. von Matt. Tikhonov regularization for large scale problems. In *Workshop on Scientific Computing.* G. H. Golub, S. H. Lui, F. Luk, and R. Plemmons (eds.), Springer-Verlag, New York (1997).

36. G. H. Golub and V. Pereyra. On pseudo-inverses of perturbed matrices. Tech. Report STAN-CS-72-261, Computer Science Department, Stanford, CA (1972).

37*. G. H. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM J. Numer. Anal.*, **10**, 413–432 (1973).

38. G. H. Golub and V. Pereyra. Differentiation of pseudoinverses, separable nonlinear least squares problems and other tales. In *Generalized Inverses and Applications. Proceedings from Advanced Seminar on Generalized Inverses and Applications, Madison, WI, 1973.* M. Z. Nashed (ed.), Academic Press, New York, pp. 303–324 (1976).

39. G. H. Golub and V. Pereyra. Separable nonlinear least squares: the variable projection method and its application. *Inverse Problems*, **19**, R1–R26 (2003).

40*. G. H. Golub and C. Reinsch. Singular value decomposition and least squares solution. *Numer. Math.*, **14**, 403–420 (1970).

41. G. H. Golub and M. A. Saunders. Linear least squares and quadratic programming. In *Integer and Nonlinear Programming.* J. Abadie (ed.), North-Holland, Amsterdam, pp. 229–256 (1970).

42. G. H. Golub and G. P. Styan. Numerical computations for univariate linear models. *J. Statist. Comput. Simul.*, **2**, 253–274 (1973).

43*. G. H. Golub and C. F. Van Loan. An analysis of the total least squares problem. *SIAM J. Numer. Anal.*, **17**, 883–893 (1980).

44. G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore (1983).

45. G. H. Golub and J. H. Wilkinson. Iterative refinement of least squares solutions. In *Proceedings of the IFIP Congress 65, New York, 1965*, W. A. Kalenich (ed.), Spartan Books, Washington, pp. 606–607 (1965).

46. G. H. Golub and J. H. Wilkinson. Note on the iterative refinement of least squares solution. *Numer. Math.*, **9**, 139–148 (1966).

47. C. Gu, D. M. Bates, Z. Chen, and G. Wahba. The computation of generalized cross-validation functions through Householder tridiagonalization with application to the fitting of interaction spline models. *SIAM J. Matrix. Anal. Appl.*, **10**, 457–480 (1989).

48. M. Gu and S. C. Eisenstat. A divide-and-conquer algorithms for the bidiagonal SVD. *SIAM J. Matrix. Anal. Appl.*, **16**(1), 79–92 (1995).

49. I. Guttman, V. Pereyra, and H. D. Scolnik. Least squares estimation for a class of nonlinear models. *Technometrics*, **15**, 209–218 (1973).

50. M. Hanke and P. C. Hansen. Regularization methods for large-scale problems. *Surveys Math. Indust.*, **3**, 253–315 (1993).
51. P. C. Hansen. *Rank-Deficient and Ill-Posed Problems; Numerical Aspects of Linear Inversion.* SIAM, Philadelphia (1998).
52. P. C. Hansen and D. O'Leary. Regularization algorithms based on total least squares. In *Recent Advances in Total Least Squares Techniques and Errors-in-Variable Modeling*, Sabine Van Huffel (ed.), SIAM, Philadelphia, PA, pp. 127–137 (1997).
53. R. J. Hanson and C. L. Lawson. Extensions and applications of the Householder algorithm for solving linear least squares problems. *Math. Comp.*, **23**, 787–812 (1969).
54. A. S. Householder. Unitary triangularization of a nonsymmetric matrix. *J. Assoc. Comput. Mach.*, **5**, 339–342 (1958).
55. A. S. Householder. *The Theory of Matrices in Numerical Analysis.* Blaisdell, New York (1964).
56. M. F. Hutchinson. A fast procedure for calculating minimum cross-validation cubic smoothing splines. *ACM Trans Math. Software*, **12**, 150–153 (1986).
57. M. Jansen, Maurits Malfait, and Adhemar Bulhteel. Generalized cross validation for wavelet thresholding. *Signal Processing.*, **56**, 33–44 (1997).
58. T. L. Jordan. Experiments on error growth associated with some linear least-squares procedures. *Math. Comp.*, **22**, 579–588 (1968).
59. W. Kahan. Numerical linear algebra. *Canad. Math. Bull.*, **9**, 757–801 (1966).
60. L. Kaufman. Variable projection methods for solving separable nonlinear least squares problems. *BIT*, **15**, 49–57 (1975).
61. L. Kaufman and V. Pereyra. A method for separable nonlinear least squares problems with separable nonlinear equality constraints. *SIAM J. Numer. Anal.*, **15**, 12–20 (1978).
62. L. Kaufman and G. Sylvester. Separable nonlinear least squares problems with multiple right hand sides. *SIAM J. Matrix Anal. Appl.*, **13**, 68–89 (1992).
63. L. Kaufman, G. Sylvester, and M. H. Wright. Structured linear least squares problems in system identification and separable nonlinear data fitting. *SIAM J. Optim.*, **4**, 847–871 (1994).
64. F. T. Krogh. Efficient implementation of a variable projection algorithm for nonlinear least squares. *Comm. ACM*, **17**, 167–169 (1974).
65. C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems.* Prentice Hall, Englewood Cliffs, NJ (1974). (Republished 1995 in revised form by SIAM, Classics in Applied Mathematics.)
66. Philippe Lemmerling and Sabine Van Huffel. Fast structured total least squares: Analysis, algorithms and applications. In *Total Least Squares and Errors-in-Variables Modeling*, Sabine Van Huffel and Philippe Lemmerling (ed.), Kluwer Academic Publishers, Dordrecht, pp. 79–92 (2002).

67. Ker-Chau Li. Asymptotic optimality of C_L and generalized cross-validation in ridge regression with application to spline smoothing. *The Annals of Statistics*, **14**, 1101–1112 (1986).
68. M. A. Lukas. Asymptotic optimality of generalized cross-validation for choosing the regularization parameter. *Numer. Math.*, **66**, 41–66 (1993).
69. Nicola Mastronardi, Philippe Lemmerling, and Sabine Van Huffel. Structured total least squares: Analysis, algorithms via exploitation of the displacement structure. In *Total Least Squares and Errors-in-Variables Modeling*, Sabine Van Huffel and Philippe Lemmerling (eds.), Kluwer Academic Publishers, Dordrecht, pp. 93–106 (2002).
70. Pontus Matstoms. Sparse QR factorization in MATLAB. *ACM Trans. Math. Software*, **20**, 136–159 (1994).
71. Pontus Matstoms. Sparse linear least squares problems in optimization. *Comput. Optim. Appl.*, **7**, 89–110 (1997).
72. Urs von Matt. The orthogonal QD-algorithm. *SIAM J. Sci. Comput.*, **18**, 1163–1186 (1997).
73. Nhat Nguyen, Peyman Milanfar, and G. Golub. A computationally efficient superresolution image reconstruction algorithm. *IEEE Trans. Image Proc.*, **10**(4), 573–583 (2001).
74. Nhat Nguyen, Peyman Milanfar, and G. Golub. Efficient generalized cross validation with applications to parametric image restoration and resolution enhancement. *IEEE Trans. Image Proc.*, **10**(9), 1299–1308 (2001).
75. C. C. Paige and Z. Strakoš. Core problems in linear algebraic systems. *SIAM J. Matrix. Anal. Appl.* (2005).
76. C. C. Paige and M. Wei. Analysis of the generalized total least squares problem $AX \approx B$ when some columns are free of errors. *Numer. Math.*, **65**, 177–202 (1993).
77. C. T. Pan and P. T. P. Tang. Bounds on singular values revealed by QR factorizations. *BIT*, **39**, 740–756 (1999).
78. K. Pearson. On lines and planes of closest fit to points in space. *Phil. Mag.*, **2**, 559–572 (1901).
79. G. Peters and J. H. Wilkinson. The least squares problem and pseudoinverses. *Comput. J.*, **13**, 309–316 (1970).
80. M. J. D. Powell and J. K. Reid. On applying Householder's method to linear least squares problems. In *Proceedings of the IFIP Congress 68*, A. J. M. Morell (ed.) North-Holland, Amsterdam, pp. 122–126 (1969).
81. J. K. Reid. A note on the least squares solution of a band system of linear equations by Householder reductions. *Comput J.*, **10**, 188–189 (1967).
82. J. R. Rice. Experiments on Gram–Schmidt orthogonalization. *Math. Comp.*, **20**, 325–328 (1966).
83. J. D. Riley. Solving systems of linear equations with a positive definite symmetric but possibly ill-conditioned matrix. *Math. Tables Aids. Comput.*, **9**, 96–101 (1956).
84. A. Ruhe and P.-Å. Wedin. Algorithms for separable nonlinear least squares problems. *SIAM Review*, **22**, 318–337 (1980).

85. H. Rutishauser. Once again: The least squares problem. *Linear Algebra Appl.*, **1**, 479–488 (1968).
86. H. D. Scolnik. On the solution of non-linear least squares problems. In *Proc. IFIP 71, Amsterdam*, M. Z. Nashed (ed.), North Holland, pp. 1258–1265 (1972).
87. G. W. Stewart. On the early history of the singular value decomposition. *SIAM Review*, **35**, 551–566 (1993).
88. G. W. Stewart. Updating a rank-revealing ULV decomposition. *SIAM J. Matrix Anal. Appl.*, **14**, 494–499 (1993).
89. A. N. Tikhonov. Regularization of incorrectly posed problems. *Soviet Math. Dokl.*, **4**, 1624–1627 (1963).
90. L. N. Trefethen and D. Bau, III. *Numerical Linear Algebra.* SIAM, Philadelphia (1997).
91. S. Van Huffel. Analysis of the Total Least Squares Problem and Its Use in Parameter Estimation. PhD thesis, Katholieke Universiteit Leuven, Belgium (1987).
92. S. Van Huffel. Iterative algorithms for computing the singular subspace of a matrix associated with its smallest singular values. *Linear Algebra Appl.*, **154/156**, 675–709 (1991).
93. Sabine Van Huffel, editor. *Recent Advances in Total Least Squares Techniques and Errors-in-Variables Modeling.* SIAM, Philadelphia, PA (1997).
94. Sabine Van Huffel and Philippe Lemmerling, editors. *Total Least Squares and Errors-in-Variables Modeling.* Kluwer Academic Publishers, Dordrecht (2002).
95. S. Van Huffel and J. Vandewalle. Algebraic relationships between classical regression and total least-squares estimation. *Linear Algebra Appl.*, **93**, 149–162 (1987).
96. S. Van Huffel and J. Vandewalle. Analysis and solution of the nongeneric total least squares problem. *SIAM J. Matrix Anal. Appl.*, **9**, 360–372 (1988).
97. S. Van Huffel and J. Vandewalle. Analysis and properties of the generalized total least squares problem $AX \approx B$ when some or all columns in A are subject to error. *SIAM J. Matrix. Anal. Appl.*, **10**, 294–315 (1989).
98. S. Van Huffel and J. Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis*, volume 9 of *Frontiers in Applied Mathematics.* SIAM, Philadelphia (1991).
99. S. Van Huffel, J. Vandewalle, and A. Haegemans. An efficient and reliable algorithm for computing the singular subspace of a matrix, associated with its smallest singular values. *J. Comp. Appl. Math.*, **19**, 313–330 (1987).
100. S. Van Huffel and H. Zha. The restricted total least squares problem: Formulation, algorithm, and properties. *SIAM J. Matrix Anal. Appl.*, **12**, 292–309 (1991).

101. S. Van Huffel and H. Zha. An efficient total least squares algorithm based on a rank revealing two-sided orthogonal decomposition. *Numer. Algorithms*, **4**, 101–133 (1993).

102. G. Wahba. Practical approximate solutions to linear operator equations when the data are noisy. *SIAM J. Numer. Anal.*, **14**, 651–667 (1977).

103. R. H. Wampler. A report on the accuracy of some widely used least squares computer programs. *J. Amer. Statist. Assoc.*, **65**, 549–565 (1970).

104. Per-Åke Wedin. On pseudoinverses of perturbed matrices. Tech. Report, Department of Computer Science, Lund University, Sweden (1969).

105. J. H. Wilkinson. Error analysis of transformations based on the use of matrices of the form $I - 2xx^{\mathrm{H}}$. In *Error in Digital Computation*, L. B. Rall (ed.), John Wiley, New York, pp. 77–101 (1965).

106. J. H. Wilkinson. Global convergence of tridiagonal QR algorithm with origin shifts. *Linear Algebra Appl.*, **1**, 409–420 (1968).

107. J. H. Wilkinson and C. Reinsch. *Handbook for Automatic Computation, Volume II: Linear Algebra.* Springer-Verlag, New York (1971).

An asterisk denotes a paper reprinted in this volume.

11

NUMERICAL METHODS FOR SOLVING LINEAR LEAST SQUARES PROBLEMS

Numer. Math. (7), 206–216 (1965).

Numerische Mathematik 7, 206—216 (1965)

Numerical Methods for Solving Linear Least Squares Problems*

By
G. Golub

Abstract. A common problem in a Computer Laboratory is that of finding linear least squares solutions. These problems arise in a variety of areas and in a variety of contexts. Linear least squares problems are particularly difficult to solve because they frequently involve large quantities of data, and they are ill-conditioned by their very nature. In this paper, we shall consider stable numerical methods for handling these problems. Our basic tool is a matrix decomposition based on orthogonal Householder transformations.

1. Introduction

Let A be a given $m\times n$ real matrix of rank r, and $\boldsymbol{b}$ a given vector. We wish to determine a vector $\hat{\boldsymbol{x}}$ such that

$$\|\boldsymbol{b}-A\hat{\boldsymbol{x}}\|=\min. \tag{1.1}$$

where $\|\ldots\|$ indicates the euclidean norm. If $m\geqq n$ and $r<n$ then there is no unique solution. Under these conditions, we require simultaneously to (1.1) that

$$\|\hat{\boldsymbol{x}}\|=\min. \tag{1.2}$$

Condition (1.2) is a very natural one for many statistical and numerical problems.

If $m\geqq n$ and $r=n$, then it is well known (cf. [4]) that $\hat{\boldsymbol{x}}$ satisfies the equation

$$A^T A\boldsymbol{x}=A^T\boldsymbol{b}. \tag{1.3}$$

Unfortunately, the matrix A^TA is frequently ill-conditioned [6] and influenced greatly by roundoff errors. The following example of Läuchli [3] illustrates this well. Suppose

$$A=\begin{bmatrix}1&1&1&1&1\\ \varepsilon&0&0&0&0\\ 0&\varepsilon&0&0&0\\ 0&0&\varepsilon&0&0\\ 0&0&0&\varepsilon&0\\ 0&0&0&0&\varepsilon\end{bmatrix},$$

* Reproduction in Whole or in Part is permitted for any Purpose of the United States government. This report was supported in part by Office of Naval Research Contract Nonr-225(37) (NR 044-11) at Stanford University.

then

$$A^T A = \begin{bmatrix} 1+\varepsilon^2 & 1 & 1 & 1 & 1 \\ 1 & 1+\varepsilon^2 & 1 & 1 & 1 \\ 1 & 1 & 1+\varepsilon^2 & 1 & 1 \\ 1 & 1 & 1 & 1+\varepsilon^2 & 1 \\ 1 & 1 & 1 & 1 & 1+\varepsilon^2 \end{bmatrix}. \tag{1.4}$$

Clearly for $\varepsilon \neq 0$, the rank of $A^T A$ is five since the eigenvalues of $A^T A$ are $5+\varepsilon^2$, ε^2, ε^2, ε^2, ε^2.

Let us assume that the elements of $A^T A$ are computed using double precision arithmetic, and then rounded to single precision accuracy. Now let η be the largest number on the computer such that $fl(1.0+\eta) \equiv 1.0$ where $fl(\ldots)$ indicates the floating point computation. Then if $\varepsilon < \frac{\sqrt{\eta}}{2}$, the rank of the computed representation of (1.4) will be one. Consequently, no matter how accurate the linear equation solver, it is impossible to solve the normal equations (1.3).

In [*2*], Householder stressed the use of orthogonal transformations for solving linear least squares problems. In this paper, we shall exploit these transformations and show their use in a variety of least squares problems.

2. A Matrix Decomposition

Throughout this section, we shall assume $m \geqq n = r$.

Since the euclidean norm of a vector is unitarily invariant,

$$\|\boldsymbol{b} - A\boldsymbol{x}\| = \|\boldsymbol{c} - QA\boldsymbol{x}\|$$

where $\boldsymbol{c} = Q\boldsymbol{b}$ and Q is an orthogonal matrix. We choose Q so that

$$QA = R = \begin{pmatrix} \tilde{R} \\ \ldots \\ O \end{pmatrix} \}\, (m-n) \times n \tag{2.1}$$

where $\tilde{R}$ is an upper triangular matrix. Clearly,

$$\hat{\boldsymbol{x}} = \tilde{R}^{-1} \boldsymbol{c}$$

where $\tilde{\boldsymbol{c}}$ is the first n components of $\boldsymbol{c}$ and consequently,

$$\|\boldsymbol{b} - A\hat{\boldsymbol{x}}\| = \left(\sum_{j=m+1}^{n} c_j^2 \right)^{\frac{1}{2}}.$$

Since $\tilde{R}$ is an upper triangular matrix and $\tilde{R}^T \tilde{R} = A^T A$, $\tilde{R}^T \tilde{R}$ is simply the Choleski decomposition of $A^T A$.

There are a number of ways to achieve the decomposition (2.1); e.g., one could apply a sequence of plane rotations to annihilate the elements below the diagonal of A. A very effective method to realize the decomposition (2.1) is via Householder transformations [*2*]. Let $A = A^{(1)}$, and let $A^{(2)}, A^{(3)}, \ldots, A^{(n+1)}$ be defined as follows:

$$A^{(k+1)} = P^{(k)} A^{(k)} \qquad (k = 1, 2, \ldots, n).$$

$P^{(k)}$ is a symmetric, orthogonal matrix of the form

$$P^{(k)} = I - 2\boldsymbol{w}^{(k)}\boldsymbol{w}^{(k)T}$$

for suitable $\boldsymbol{w}^{(k)}$ such that $\boldsymbol{w}^{(k)T}\boldsymbol{w}^{(k)} = 1$. A derivation of $P^{(k)}$ is given in [*9*].

In order to simplify the calculations, we redefine $P^{(k)}$ as follows:

$$P^{(k)} = I - \beta_k \boldsymbol{u}^{(k)}\boldsymbol{u}^{(k)T}$$

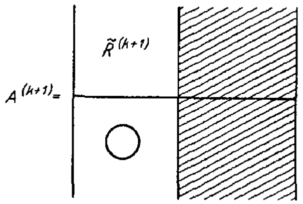
where

$$\begin{aligned}
\sigma_k &= \left(\sum_{i=k}^{m} (a_{i,k}^{(k)})^2\right)^{\frac{1}{2}}, \\
\beta_k &= [\sigma_k(\sigma_k + |a_{k,k}^{(k)}|)]^{-1}, \\
u_i^{(k)} &= 0 && \text{for} \quad i < k, \\
u_k^{(k)} &= \operatorname{sgn}(a_{k,k}^{(k)})\,(\sigma_k + |a_{k,k}^{(k)}|), \\
u_i^{(k)} &= a_{i,k}^{(k)} && \text{for} \quad i > k.
\end{aligned}$$

Thus

$$A^{(k+1)} = A^{(k)} - \boldsymbol{u}^{(k)}(\beta_k \boldsymbol{u}^{(k)T} A^{(k)}).$$

After $P^{(k)}$ has been applied to $A^{(k)}$, $A^{(k+1)}$ appears as follows:



where $\tilde{R}^{(k+1)}$ is a $k \times k$ upper triangular matrix which is unchanged by subsequent transformations. Now $a_{k,k}^{(k+1)} = -(\operatorname{sgn} a_{k,k}^{(k)})\sigma_k$ so that the rank of A is less than n if $\sigma_k = 0$. Clearly,

$$R = A^{(n+1)}$$

and

$$Q = P^{(n)} P^{(n-1)} \dots P^{(1)}$$

although one need not compute Q explicitly.

3. The Practical Procedure

WILKINSON [*10*] has shown that the Choleski decomposition is stable for a positive definite matrix even if no interchanges of rows and columns are performed. Since we are in effect performing a Choleski decomposition of $A^T A$, no interchanges of the columns of A are needed in most situations. However, numerical experiments have indicated that the accuracy is slightly improved by the interchange strategies outlined below, and consequently, in order to ensure the utmost accuracy one should choose the columns of A by some strategy. In what follows, we shall refer to the matrix $A^{(k)}$ even if some of the columns have been interchanged.

One possibility is to choose at the k^{th} stage the columns of $A^{(k)}$ which will maximize $|a_{k,k}^{(k+1)}|$. This is equivalent to searching for the maximum diagonal element in the Choleski decomposition of $A^T A$. Let

$$s_j^{(k)} = \sum_{i=k}^{m} (a_{i,j}^{(k)})^2 \quad \text{for} \quad j=k, k+1, \dots, n.$$

Then since $|a_{k,k}^{(k+1)}| = \sigma_k$, one should choose that column for which $s_j^{(k)}$ is maximized. After $A^{(k+1)}$ has been computed, one can compute $s_j^{(k+1)}$ as follows:

$$s_j^{(k+1)} = s_j^{(k)} - (a_{k,j}^{(k+1)})^2 \qquad (j=k+1, \dots, m)$$

since the orthogonal transformations leave the column lengths invariant. Naturally, the $s_j^{(k)}$'s must be interchanged if the columns of $A^{(k)}$ are interchanged. Although it is possible to compute σ_k directly from the $s_j^{(k)}$'s, it is best to compute σ_k at each stage using double precision inner products to ensure maximal accuracy.

The strategy described above is most appropriate when one has a sequence of vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_p$ for which one desires a least squares estimate. In many problems, there is one vector $\boldsymbol{b}$ and one wishes to express it in as few columns of A as possible. This is the stagewise multiple regression problem. We cannot solve this problem, but we shall show how one can choose that column of $A^{(k)}$ for which the sum of squares of residuals is maximally reduced at the k^{th} stage.

Let $\boldsymbol{c}^{(1)} = \boldsymbol{b}$ and $\boldsymbol{c}^{(k+1)} = P^{(k)} \boldsymbol{c}^{(k)}$. Now $\tilde{R}^{(k)} \hat{\boldsymbol{x}}^{(k-1)} = \tilde{\boldsymbol{c}}^{(k)}$ where $\hat{\boldsymbol{x}}^{(k-1)}$ is the least squares estimate based on $(k-1)$ columns of A, and $\tilde{\boldsymbol{c}}^{(k)}$ is the first $(k-1)$ elements of $\boldsymbol{c}^{(k)}$, and consequently

$$\|\boldsymbol{c}^{(k)} - \tilde{R}^{(k)} \hat{\boldsymbol{x}}^{(k-1)}\| = \left(\sum_{j=k}^{m} (c_j^{(k)})^2 \right)^{\frac{1}{2}}.$$

Since length is preserved under an orthogonal transformation, we wish to find that column of $A^{(k)}$ which will maximize $|c_k^{(k+1)}|$. Let

$$t_j^{(k)} = \sum_{i=k}^{m} a_{i,j}^{(k)} c_i^{(k)} \quad \text{for} \quad j=k, k+1, \dots, m.$$

Then since $|c_k^{(k+1)}| = \left| \sum_{i=k}^{m} a_{i,k}^{(k)} c_i^{(k)} / \sigma_k \right|$ one should choose that column of $A^{(k)}$ for which $(t_j^{(k)})^2 / s_j^{(k)}$ is maximized. After $P^{(k)}$ is applied to $A^{(k)}$, one can adjust $t_j^{(k)}$ as follows:

$$t_j^{(k+1)} = t_j^{(k)} - a_{k,j}^{(k+1)} c_k^{(k+1)}.$$

In many statistical applications, if $(t_j^{(k)})^2 / s^{(k)}$ is sufficiently small then no further transformations are performed.

Once the solution to the equations has been obtained then it is possible to obtain an improved solution by a simple iterative technique. This technique, however, requires that the orthogonal transformations be saved during their application. The best method for storing the transformation is to store the elements of $\boldsymbol{u}^{(k)}$ below the diagonal of the k^{th} column of $A^{(k+1)}$.

Let $\bar{x}$ be the initial solution obtained, and let $\hat{x}=\bar{x}+e$. Then

$$\|b-A\hat{x}\|=\|r-Ae\|$$

where

$$r=b-A\bar{x}, \quad \text{the residual vector.}$$

Thus the correction vector e is itself the solution to a linear least squares problem. Once A has been decomposed then it is a fairly simple matter to compute r and solve for e. Since e critically depends upon the residual vector, the components of r should be computed using double precision inner products and then rounded to single precision accuracy. Naturally, one should continue to iterate as long as improved estimates of $\hat{x}$ are obtained.

The above iteration technique will converge only if the initial approximation to $\hat{x}$ is sufficiently accurate. Let

$$x^{(q+1)}=x^{(q)}+e^{(q)} \qquad (q=0, 1, \ldots,)$$

with $x^{(0)}=0$. Then one should iterate only if $\|e^{(1)}\|/\|x^{(1)}\|\leq c$ where $c<\frac{1}{2}$, i.e. "at least one bit of the initial solution is correct"; otherwise there is little likelihood that the iterative method will converge. Since convergence tends to be linear, one should terminate the procedure as soon as

$$\|e^{(k+1)}\|>c\|e^{(k)}\| \quad \text{or} \quad \|e^{(k)}\|<\eta\|x^{(1)}\|$$

where η is the maximum positive number such that $fl(1+\eta)\equiv 1$.

4. A Numerical Example

In Table 1, we give the results of an extensive calculation. The matrix consists of the first 5 columns of the inverse of the 6×6 Hilbert matrix. The calculations were performed in single precision arithmetic. The columns were chosen so that the diagonal elements were maximized at each stage. The iteration procedure was terminated as soon as $\|e^{(k+1)}\|>0.25\|e^{(k)}\|$. Three iterations were performed but since $\|e^{(2)}\|>0.25\|e^{(1)}\|$, $x^{(2)}$ was taken to be the correct solution.

In Table 2, we show the results of using double precision inner products on the same problem. Note that the first iterate in Table 1 is approximately as accurate as the first iterate in Table 2. The double precision inner product routine converged to a solution for which all figures were accurate. The normal equations were formed using double precision inner products but even with a very accurate linear equation solver described by McKeeman [5] no solution could be obtained.

5. An Iterative Scheme

For many problems, even with the use of orthogonal transformations it may be impossible to obtain an accurate solution. Or, the rank of A may truly be less than n. In this section, we give an algorithm for finding the least squares solution even if A^TA is singular.

In [7], Riley suggested the following algorithm for solving linear least squares problems for $r=n$. Let $x^{(0)}$ be an arbitrary vector, then solve

$$(A^TA+\alpha I)x^{(q+1)}=A^Tb+\alpha x^{(q)}. \tag{5.1}$$

Table 1

			A			B
	$3.6000000000_{10}+01$	$-6.3000000000_{10}+02$	$3.3600000000_{10}+03$	$-7.5600000000_{10}+03$	$7.5600000000_{10}+03$	$4.6300000000_{10}+02$
	$-6.3000000000_{10}+02$	$1.4700000000_{10}+04$	$-8.8200000000_{10}+04$	$2.1168000000_{10}+05$	$-2.2050000000_{10}+05$	$-1.3860000000_{10}+04$
	$3.3600000000_{10}+03$	$-8.8200000000_{10}+04$	$5.6448000000_{10}+05$	$-1.4112000000_{10}+06$	$1.5120000000_{10}+06$	$9.7020000000_{10}+04$
	$-7.5600000000_{10}+03$	$2.1168000000_{10}+05$	$-1.4112000000_{10}+06$	$3.6288000000_{10}+06$	$-3.9690000000_{10}+06$	$-2.5872000000_{10}+05$
	$7.5600000000_{10}+03$	$-2.2050000000_{10}+05$	$1.5120000000_{10}+06$	$-3.9690000000_{10}+06$	$4.4100000000_{10}+06$	$2.9106000000_{10}+05$
	$-2.7720000000_{10}+03$	$8.3160000000_{10}+04$	$-5.8212000000_{10}+05$	$1.5523200000_{10}+06$	$-1.7463600000_{10}+06$	$-1.1642400000_{10}+05$
			Pivot			
	5	4	3	2	1	
			R			
	$-6.3706872199_{10}+06$	$5.7760446368_{10}+06$	$-2.2224495814_{10}+06$	$3.2875491420_{10}+05$	$-1.1522407952_{10}+04$	
		$-6.7135626821_{10}+04$	$6.0308471363_{10}+04$	$-1.6102552917_{10}+04$	$9.4910780925_{10}+02$	
			$-1.4425503500_{10}+03$	$9.4707042643_{10}+02$	$-1.0982644636_{10}+02$	
				$4.6175131643_{10}+01$	$-1.4949038075_{10}+01$	
					$2.0095559062_{10}+00$	
			Iteration			
1						
X	$9.9999912362_{10}-01$	$4.9999972493_{10}-01$	$3.3333322021_{10}-01$	$2.4999995198_{10}-01$	$1.9999998332_{10}-01$	
R	$1.4528632164_{10}-06$	$-4.7683715820_{10}-07$	$0.0000000000_{10}+00$	$0.0000000000_{10}+00$	$0.0000000000_{10}+00$	$0.00000000_{10}00+00$
E	$5.8284221383_{10}-07$	$1.7292551586_{10}-07$	$6.9381181279_{10}-08$	$2.9087566395_{10}-08$	$1.0038052093_{10}-08$	
2						
X	$9.9999970646_{10}-01$	$4.9999989786_{10}-01$	$3.3333328959_{10}-01$	$2.4999998107_{10}-01$	$1.9999999336_{10}-01$	
R	$3.2037496567_{10}-07$	$4.7683715820_{10}-07$	$3.8146972656_{10}-06$	$-3.8146972656_{10}-06$	$3.8146972656_{10}-06$	$0.0000000000_{10}+00$
E	$-3.7819874906_{10}-07$	$-1.1487774268_{10}-07$	$-4.6569725113_{10}-08$	$-1.9660767363_{10}-08$	$-6.8227426601_{10}-09$	

Table 2

Pivot

		5	4	3	2	1	
				R			
		$-6.3706872199_{10}+06$	$5.7760446368_{10}+06$	$-2.2224495814_{10}+06$	$3.2875491420_{10}+05$	$-1.1522407952_{10}+04$	
			$-6.7135626821_{10}+04$	$6.0308471363_{10}+04$	$-1.6102552917_{10}+04$	$9.4910780925_{10}+02$	
				$-1.4425503500_{10}+03$	$9.4707042643_{10}+02$	$-1.0982644637_{10}+02$	
					$4.6175131642_{10}+01$	$-1.4949038077_{10}+01$	
						$2.0095559061_{10}+00$	
				Iteration			
1							
	X	$9.9999912347_{10}-01$	$4.9999972435_{10}-01$	$3.3333321985_{10}-01$	$2.4999995180_{10}-01$	$1.9999998325_{10}-01$	
	R	$1.4480865502_{10}-06$	$-8.8621163741_{10}-08$	$9.3205017038_{10}-08$	$-2.6189081836_{10}-06$	$2.1314917831_{10}-07$	$3.2270327211_{10}-07$
	E	$8.7653116638_{10}-07$	$2.7564789022_{10}-07$	$1.1348434536_{10}-07$	$4.8201409291_{10}-08$	$1.6752167940_{10}-08$	
2							
	X	$1.0000000000_{10}+00$	$5.0000000000_{10}-01$	$3.3333333333_{10}-01$	$2.5000000000_{10}-01$	$2.0000000000_{10}-01$	
	R	$-7.5378920883_{10}-09$	$2.1391315386_{10}-07$	$-1.4423858374_{10}-06$	$3.7434801925_{10}-06$	$-4.1254679672_{10}-06$	$1.6236008378_{10}-06$
	E	$9.4773720382_{10}-18$	$2.6923536909_{10}-18$	$-6.0632875784_{10}-13$	$4.2556290006_{10}-19$	$-7.2759561775_{10}-13$	
3							
	X	$1.0000000000_{10}+00$	$5.0000000000_{10}-01$	$3.3333333333_{10}-01$	$2.5000000000_{10}-01$	$2.0000000000_{10}-01$	
	R	$-7.5378920883_{10}-09$	$2.1391315386_{10}-07$	$-1.4423858374_{10}-06$	$3.7434801925_{10}-06$	$-4.1254679672_{10}-06$	$1.6236008378_{10}-06$
	E	$9.4773720382_{10}-18$	$2.6923536909_{10}-18$	$-6.0632875784_{10}-13$	$4.2556290006_{10}-19$	$-7.2759561775_{10}-13$	

The sequence $\boldsymbol{x}^{(q)}$ converges to $\hat{\boldsymbol{x}}$ if $\alpha > 0$ since the spectral radius of $\alpha(A^T A + \alpha I)^{-1}$ is less than 1. Again we may implement this algorithm more effectively by the use of orthogonal transformations.

First, let us note that (5.1) is equivalent to the following:

$$\boldsymbol{r}^{(q)} = \boldsymbol{b} - A\boldsymbol{x}^{(q)}, \tag{5.2a}$$

$$(A^T A + \alpha I)\boldsymbol{e}^{(q)} = A^T \boldsymbol{r}^{(q)}, \tag{5.2b}$$

$$\boldsymbol{x}^{(q+1)} = \boldsymbol{x}^{(q)} + \boldsymbol{e}^{(q)}. \tag{5.2c}$$

The vector $\boldsymbol{e}^{(q)}$ is itself the solution of a linear least squares problem since $\boldsymbol{e}^{(q)}$ minimize $\|\boldsymbol{d}^{(q)} - C\boldsymbol{e}^{(q)}\|$ where

$$C = \begin{pmatrix} A \\ \cdots\cdot \\ \sqrt{\alpha}\, I \end{pmatrix}, \qquad \boldsymbol{d}^{(q)} = \begin{pmatrix} \boldsymbol{r}^{(q)} \\ \cdots \\ \boldsymbol{0} \end{pmatrix}.$$

Thus the numerical procedure should go as follows. Decompose C by the methods described in Section 2 so that

$$PC = S = \begin{pmatrix} \tilde{S} \\ \cdots \\ O \end{pmatrix}$$

where $P^T P = I$ and $\tilde{S}$ is an upper triangular matrix. Then let $\boldsymbol{x}^{(0)} = \boldsymbol{0}$,

$$\tilde{S}\boldsymbol{e}^{(q)} = \tilde{\boldsymbol{f}}^{(q)},$$

$$\boldsymbol{x}^{(q+1)} = \boldsymbol{x}^{(q)} + \boldsymbol{e}^{(q)}$$

and $\tilde{\boldsymbol{f}}^{(q)}$ is the vector whose components are the first n components of $P\boldsymbol{d}^{(q)}$. We choose $\boldsymbol{x}^{(0)} = \boldsymbol{0}$ since otherwise there is no assurance that $\boldsymbol{x}^{(q)}$ will converge to $\hat{\boldsymbol{x}}$.

Now going back to the original process (5.1),

$$\boldsymbol{x}^{(q+1)} = G\boldsymbol{x}^{(q)} + \boldsymbol{h} \tag{5.3}$$

where

$$G = \alpha(A^T A + \alpha I)^{-1} \quad \text{and} \quad \boldsymbol{h} = (A^T A + \alpha I)^{-1} A^T \boldsymbol{b}.$$

Thus

$$\boldsymbol{x}^{(q+1)} = (G^q + G^{q-1} + \cdots + I)\boldsymbol{h}. \tag{5.4}$$

It is well known (cf. [6]) that A may be written as

$$A = U\Sigma V^T$$

where Σ is an $m \times n$ matrix with the singular values σ_j on the diagonal and zeros elsewhere, and U and V are the matrices of eigenvectors of AA^T and $A^T A$, respectively. Then

$$A^T \boldsymbol{b} = V\Sigma^T U^T \boldsymbol{b} = \beta_1 \sigma_1 \boldsymbol{v}_1 + \beta_2 \sigma_2 \boldsymbol{v}_2 + \cdots + \beta_r \sigma_r \boldsymbol{v}_r$$

where $\boldsymbol{\beta} = U^T \boldsymbol{b}$, and r is the rank of A. Then from (5.4) we see that

$$\boldsymbol{x}^{(q)} = \gamma_1^{(q)} \boldsymbol{v}_1 + \cdots + \gamma_r^{(q)} \boldsymbol{v}_r$$

where

$$\gamma_j^{(q)} = \left[1 - \left(\frac{\alpha}{\alpha+\sigma_j^2}\right)^q\right] \frac{\beta_j}{\sigma_j} \qquad (j=1,2,\dots,r).$$

Thus as $q \to \infty$

$$\boldsymbol{x}^{(q)} \to \frac{\beta_1}{\sigma_1}\boldsymbol{v}_1 + \cdots + \frac{\beta_r}{\sigma_r}\boldsymbol{v}_r = \hat{\boldsymbol{x}}.$$

The choice of α will greatly affect the rate of convergence of the iterative method, and thus one must choose α with great care. If α is too small then the equations will remain ill-conditioned. If δ is a lower bound of the smallest non-zero singular value, then α should be chosen so that

$$\frac{\alpha}{\alpha+\delta^2} < 0.1, \quad \text{say}.$$

This means at each stage, there will be at least one more place of accuracy in the solution. There are a number of methods for accelerating the convergence of (5.1) (cf. [*1*]).

It is easy to see that

$$\boldsymbol{e}^{(q+1)} = G\boldsymbol{e}^{(q)} = \alpha(A^T A + \alpha I)^{-1}\boldsymbol{e}^{(q)}.$$

Since $\boldsymbol{e}^{(q)}$ lies in the space spanned by $\boldsymbol{v}_1, \dots, \boldsymbol{v}_r$, it follows immediately that

$$\|\boldsymbol{e}^{(q+1)}\| \leqq \frac{\alpha}{\alpha+\sigma_r^2} \|\boldsymbol{e}^{(q)}\| < \|\boldsymbol{e}^{(q)}\|.$$

Thus a good termination procedure is to stop iterating as soon as $\|\boldsymbol{e}^{(q)}\|$ increases or does not change.

6. Statistical Calculations

In many statistical calculations, it is necessary to compute certain auxiliary information associated with $A^T A$. These can readily be obtained from the orthogonal decomposition. Thus

$$\det(A^T A) = (r_{11} \times r_{22} \times \cdots \times r_{nn})^2.$$

Since

$$A^T A = \tilde{R}^T \tilde{R}, \qquad (A^T A)^{-1} = \tilde{R}^{-1} \tilde{R}^{-T}.$$

The inverse of $\tilde{R}$ can be readily obtained since $\tilde{R}$ is an upper triangular matrix. WAUGH and DWYER [*8*] have noted that it is possible to calculate $(A^T A)^{-1}$ directly from $\tilde{R}$. Let

$$(A^T A)^{-1} = X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n).$$

Then from the relationship

$$\tilde{R} X = \tilde{R}^{-T}$$

and by noting that $\{\tilde{R}^{-T}\}_{ii} = 1/r_{ii}$, it is possible to compute $\boldsymbol{x}_n, \boldsymbol{x}_{n-1}, \dots, \boldsymbol{x}_1$. The number of operations are roughly the same as in the first method but more accurate bounds may be established for this method provided all inner products are accumulated to double precision.

In some statistical applications, the original set of observations are augmented by an additional set of observations. In this case, it is not necessary to begin the calculation from the beginning again if the method of orthogonalization is used. Let $\tilde{R}_1, \tilde{\boldsymbol{c}}_1$ correspond to the original data after it has been reduced by orthogonal transformations and let $A_2, \boldsymbol{b}_2$ correspond to the additional observations. Then the up-dated least squares solution can be obtained directly from

$$A = \begin{pmatrix} \tilde{R}_1 \\ \cdots \\ A_2 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} \tilde{\boldsymbol{c}}_1 \\ \cdots \\ \boldsymbol{b}_2 \end{pmatrix}.$$

The above observation has another implication. One of the arguments frequently advanced for using normal equations is that only $n(n+1)/2$ memory locations are required. By partitioning the matrix A by rows, however, then similarly only $n(n+1)/2$ locations are needed when the method of orthogonalization is used.

7. Least Squares Problems with Constraints

Frequently, one wishes to determine $\hat{\boldsymbol{x}}$ so that $\|\boldsymbol{b} - A\hat{\boldsymbol{x}}\|$ is minimized subject to the condition that $H\hat{\boldsymbol{x}} = \boldsymbol{g}$ where H is a $p \times n$ matrix of rank p. One can, of course, eliminate p of the columns of A by Gaussian elimination after a $p \times p$ submatrix of H has been determined and then solve the resulting normal equations. This, unfortunately, would not be a numerically stable scheme since no row interchanges between A and H would be permitted.

If one uses Lagrange multipliers, then one must solve the $(n+p) \times (n+p)$ system of equations.

$$\left(\begin{array}{c|c} A^T A & H^T \\ \hline H & 0 \end{array}\right) \begin{pmatrix} \hat{\boldsymbol{x}} \\ \cdots \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} A^T \boldsymbol{b} \\ \cdots\cdot \\ \boldsymbol{g} \end{pmatrix}$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers. Since $\hat{\boldsymbol{x}} = (A^T A)^{-1} A^T \boldsymbol{b} - (A^T A)^{-1} H^T \boldsymbol{\lambda}$,

$$H(A^T A)^{-1} H^T \boldsymbol{\lambda} = H\boldsymbol{z} - \boldsymbol{g}$$

where

$$\boldsymbol{z} = (A^T A)^{-1} A^T \boldsymbol{b}.$$

Note $\boldsymbol{z}$ is the least squares solution of the original problem without constraints and one would frequently wish to compare this vector with the final solution $\hat{\boldsymbol{x}}$. The vector $\boldsymbol{z}$, of course, should be computed by the orthogonalization procedures discussed earlier.

Since $A^T A = \tilde{R}^T \tilde{R}$, $H(A^T A)^{-1} H^T = W^T W$ where $W = \tilde{R}^{-T} H^T$. After W is computed, it should be reduced to a $p \times p$ upper triangular matrix K by orthogonalization which is the Choleski decomposition of $W^T W$. The matrix equation

$$K^T K \boldsymbol{\lambda} = H\boldsymbol{z} - \boldsymbol{g}$$

should be solved by the obvious method. Finally, one finds

$$\hat{\boldsymbol{x}} = \boldsymbol{z} - (A^T A)^{-1} H \boldsymbol{\lambda}$$

where $(A^T A)^{-1} H \boldsymbol{\lambda}$ can be easily computed by using $\tilde{R}^{-1}$.

Acknowledgements. The author is very pleased to acknowledge the programming efforts of Mr. PETER BUSINGER and Mr. ALAN MERTEN. He also wishes to thank Professor THOMAS ROBERTSON for his critical remarks.

References

[1] GOLUB, G. H., and R. S. VARGA: Chebyshev Semi-Iterative Methods, Successive Over-Relaxation Iterative Methods, and Second Order Richardson Iterative Method. Numer. Math. **3**, 147—168 (1961).

[2] HOUSEHOLDER, A. S.: Unitary Triangularization of a Nonsymmetric Matrix. J. Assoc. Comput. Mach. **5**, 339—342 (1958).

[3] LÄUCHLI, P.: Jordan-Elimination und Ausgleichung nach kleinsten Quadraten. Numer. Math. **3**, 226—240 (1961).

[4] LINNIK, Y.: Method of Least Squares and Principles of the Theory of Observations. Translated from Russian by R. C. ELANDT. New York: Pergamon Press 1961.

[5] MCKEEMAN, W. M.: Crout with Equilibration and Iteration. Algorithm 135. Comm. Assoc. Comput. Mach. **5**, 553—555 (1962).

[6] OSBORNE, E. E.: On Least Squares Solutions of Linear Equations. J. Assoc. Comput. Mach. **8**, 628—636 (1961).

[7] RILEY, J. D.: Solving Systems of Linear Equations with a Positive Definite, Symmetric, but Possibly Ill-Conditioned Matrix. Math. Tables Aids Comput. **9**, 96—101 (1956).

[8] WAUGH, F. V., and P. S. DWYER: Compact Computation of the Inverse of a Matrix. Ann. Math. Stat. **16**, 259—271 (1945).

[9] WILKINSON, J. H.: HOUSEHOLDERS Method for the Solution of the Algebraic Eigenproblem. Comput. J. **3**, 23—27 (1960).

[10] — Error Analysis of Direct Methods of Matrix Inversion. J. Assoc. Comput. Mach. **8**, 281—330 (1961).

Stanford University
Computation Center
Stanford, California 94305 (USA)

(Received September 24, 1964)

12

SINGULAR VALUE DECOMPOSITION AND LEAST SQUARES SOLUTIONS (WITH C. REINSCH)

NUMER. MATH. (14), 403–420 (1970).

SVD is the triple factorization of a given $m \times n$ matrix $A \in \mathbb{C}^{m,n}$:

$$A =: U\,\Sigma\,V^H. \tag{12.1}$$

Here $U \in \mathbb{C}^{m,m}$ and $V \in \mathbb{C}^{n,n}$ are unitary: $U^H U = I_m, \quad V^H V = I_n, \quad \Sigma = \operatorname{diag}(\sigma_k)$. $\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$ are the so-called *singular values of the matrix* A in decreasing order.

(Equation 12.1) means that any linear map $\mathbb{C}^n \longrightarrow \mathbb{C}^m$ splits into separated *one-dimensional* linear maps

$$y_k = \sigma_k\, x_k, \qquad k = 1, \ldots, m, \tag{12.2}$$

provided we use the appropriate orthogonal coordinate systems in the spaces $\mathbb{C}^n$ and $\mathbb{C}^m$. (If $m > n$ then the last $m - n$ equations in (12.2) are trivial: $y_k = 0$ for $k > n$.)

Of course, m one-dimensional maps are much simpler than a map from $\mathbb{C}^n$ into $\mathbb{C}^m$, so that the SVD is the great simplifier and therefore has applications in many fields.

The SVD is one of the most fundamental tools in linear algebra and has found therefore proper attention from the beginning of numerical analysis.

First attempts to compute the factors in (12.1) started with the diagonalization of the matrices $A^H A = V^H \operatorname{diag}(\sigma_k^2)\, V$ and $A\,A^H = U \operatorname{diag}(\sigma_k^2)\, U^H$, using the Jacobi method, then state of the art. We see that the singular values are the square roots of the eigenvalues of the positive (semi)definite matrices $A^H A$ or $A\,A^H$. This means numerical difficulties with the accurate computation of the smaller singular values, which usually are the more interesting ones and should be computed as precisely as possible. An equally severe problem is the correct association of the eigenvectors of $A^H A$ (columns of V) with the eigenvectors of $A\,A^H$ (columns of U) which are not unique in the event of multiple eigenvalues. Thus, this method was soon abandoned.

In 1965 Golub and Kahan published an algorithm for the computation of the SVD based on the $(m+n) \times (m+n)$ matrix

$$C = \begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix}.$$

With the standard reduction of the complex A to real bidiagonal form B by applying either Givens rotations or Householder reflections from left and right to A, and after an obvious permutation of the rows and columns of the reduced C, we arrive at a real, tridiagonal, and symmetric $(m+n) \times (m+n)$ matrix T with spectrum $\sigma_1 \geq \ldots \geq \sigma_r > 0 = \ldots = 0 > -\sigma_r \geq \ldots \geq -\sigma_1$ and zeros along the diagonal. To this matrix Golub and Kahan applied the symmetric version of the QR-transformation with a special shift-strategy: a conventionally chosen shift s for *one* QR-step and the shift with $-s$ for the next QR-step, which restores the zero-diagonal. At that time the insight into the convergence properties of the QR-iteration with shifts was not yet fully developed, so that it was overlooked how disastrous this shift technique is for a special class of tridiagonal matrices frequently encountered in physical applications. It was easy to find sample matrices with unsatisfactory convergence rates and in some cases the algorithm would not converge at all.

At this time, I had pushed the editors (F. L. Bauer, H. Rutishauser, J. Wilkinson *et al.*) to include the symmetric QR algorithm in their Handbook series of established computer routines in linear algebra, where originally only a Jacobi routine and Givens bisection method were included for solving the symmetric eigenvalue problem. This QR-iteration would use as shift one of the two eigenvalues of the lower 2×2 diagonal block as Wilkinson had used in Francis' non-symmetric QR-iteration with double steps. Instinct guided me, however, to use a different rule for selecting which of the two eigenvalues should become the next shift. With this serendipitous choice Jim Wilkinson could then prove the *global* convergence of the QR-iteration for real, symmetric, tridiagonal matrices in 1969.

For me, it was clear that a wonderful method for computing the SVD would result if one could transform the bidiagonal matrix B mentioned above into a bidiagonal matrix B', such that the transition from B^TB to B'^TB' were precisely the QR-step of the last paragraph with its global convergence property and its unique convergence rate. The problem was that a shift s had to be applied to B^TB which would make the matrix indefinite: there is no Cholesky decomposition of $B^TB - sI$. Thus a straightforward path to follow was blocked. But J. G. F. Francis (1961/62) had invented his ingenious way of applying the shift implicitly by the "chasing" technique. It turned out that this technique could be used also for the transition $B \mapsto B'$ so that the (never explicitly formed) transition $B^TB \mapsto B'^TB'$ would be as desired. The resulting SVD algorithm was designed in 1967 and soon became very popular.

The editors of the mentioned Handbook series *Linear Algebra* now had the choice between two proposals for an SVD routine: Gene Golub had submitted in 1967 a program based on the Golub–Kahan algorithm (1965) and there was my routine with its proven convergence properties. Their wise decision let them choose my algorithm under joint authorship. Incidentally, in *Matrix Computations* by Golub and Van Loan, one of the most popular textbooks in this field, that algorithm with the implicit shift and the chasing technique is described under the heading *Golub-Kahan SVD step*, and it is mentioned that the material is from Golub and Kahan (1965).

The evaluation process of the Handbook series with approval from several international editors was a very slow process. The Internet did not exist at that time, and snail mail had to be used, which back and forth across the Atlantic would take more than two weeks in the most favorable cases. Thus, the Handbook article, although finished in 1967, did not appear in print before 1970. The complex version (the reduction $A \in \mathbb{C}^{m,n} \mapsto B \in \mathbb{R}^{m,n}$) was programmed in FORTRAN by Businger and Golub (1968) and needed communication paths just within the Standford University Campus to reach the *ACM Communications* editor. Therefore it overtook the Handbook article and appeared earlier in print.

Singular value decomposition and least squares solutions (with C. Reinsch)

Our joint paper *Singular Value Decomposition and Least Squares Solutions (1970)* has been included in Part III, *Least Squares Problems*, rather than in Part IV, *Matrix Factorizations* in this volume. The reason is perhaps that I included in the section *Applicability* of that paper a reference to what I then called "A Generalization of the Least Squares Problem". Nowadays the problem is known as the *total* least squares problem, see *Matrix Computations* by Golub and Van Loan where the theory is outlined. In particular, it is shown that the problem need not have a solution in all cases. Thus, it must be ill-posed. Indeed, it is, and I regret now my attempt to demonstrate the usage of the SVD.

Christian Reinsch
Munich, Germany

Numer. Math. 14, 403–420 (1970)

Handbook Series Linear Algebra

Singular Value Decomposition and Least Squares Solutions*

Contributed by

G. H. Golub** and C. Reinsch

1. Theoretical Background

1.1. Introduction

Let A be a real $m\times n$ matrix with $m\geqq n$. It is well known (cf. [4]) that

$$A=U\Sigma V^T \tag{1}$$

where

$$U^TU=V^TV=VV^T=I_n \quad\text{and}\quad \Sigma=\operatorname{diag}(\sigma_1,\ldots,\sigma_n).$$

The matrix U consists of n orthonormalized eigenvectors associated with the n largest eigenvalues of AA^T, and the matrix V consists of the orthonormalized eigenvectors of A^TA. The diagonal elements of Σ are the non-negative square roots of the eigenvalues of A^TA; they are called *singular values*. We shall assume that

$$\sigma_1\geqq\sigma_2\geqq\cdots\geqq\sigma_n\geqq 0.$$

Thus if $rank(A)=r$, $\sigma_{r+1}=\sigma_{r+2}=\cdots=\sigma_n=0$. The decomposition (1) is called the *singular value decomposition* (SVD).

There are alternative representations to that given by (1). We may write

$$A=U_c\left(\frac{\Sigma}{0}\right)V^T \quad\text{with}\quad U_c^TU_c=I_m$$

or

$$A=U_r\Sigma_rV_r^T \quad\text{with}\quad U_r^TU_r=V_r^TV_r=I_r \quad\text{and}\quad \Sigma_r=\operatorname{diag}(\sigma_1,\ldots,\sigma_r).$$

We use the form (1), however, since it is most useful for computational purposes.

If the matrix U is not needed, it would appear that one could apply the usual diagonalization algorithms to the symmetric matrix A^TA which has to be formed explicitly. However, as in the case of linear least squares problems, the com-

* *Editor's note.* In this fascicle, prepublication of algorithms from the Linear Algebra series of the Handbook for Automatic Computation is continued. Algorithms are published in ALGOL 60 reference language as approved by the IFIP. Contributions in this series should be styled after the most recently published ones.

** The work of this author was in part supported by the National Science Foundation and Office of Naval Research.

putation of $A^T A$ involves unnecessary numerical inaccuracy. For example, let

$$A = \begin{bmatrix} 1 & 1 \\ \beta & 0 \\ 0 & \beta \end{bmatrix},$$

then

$$A^T A = \begin{bmatrix} 1+\beta^2 & 1 \\ 1 & 1+\beta^2 \end{bmatrix}$$

so that

$$\sigma_1(A) = (2+\beta^2)^{\frac{1}{2}}, \quad \sigma_2(A) = |\beta|.$$

If $\beta^2 < \varepsilon_0$, the machine precision, the computed $A^T A$ has the form $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$, and the best one may obtain from diagonalization is $\tilde{\sigma}_1 = \sqrt{2}$, $\tilde{\sigma}_2 = 0$.

To compute the singular value decomposition of a given matrix A, Forsythe and Henrici [2], Hestenes [8], and Kogbetliantz [9] proposed methods based on plane rotations. Kublanovskaya [10] suggested a QR-type method. The program described below first uses Householder transformations to reduce A to bidiagonal form, and then the QR algorithm to find the singular values of the bidiagonal matrix. The two phases properly combined produce the singular value decomposition of A.

1.2. Reduction to Bidiagonal Form

It was shown in [6] how to construct two finite sequences of Householder transformations

$$P^{(k)} = I - 2x^{(k)} x^{(k)T} \quad (k = 1, 2, \ldots, n)$$

and

$$Q^{(k)} = I - 2y^{(k)} y^{(k)T} \quad (k = 1, 2, \ldots, n-2)$$

(where $x^{(k)T} x^{(k)} = y^{(k)T} y^{(k)} = 1$) such that

$$P^{(n)} \ldots P^{(1)} A Q^{(1)} \ldots Q^{(n-2)} = \left[\begin{array}{cccccc} q_1 & e_2 & 0 & \cdots & \cdots & 0 \\ & q_2 & e_3 & & 0 & \vdots \\ & & \ddots & \ddots & & \vdots \\ & & & \ddots & \ddots & 0 \\ & 0 & & & \ddots & e_n \\ & & & & & q_n \\ \hline & & & 0 & & \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \\ \\ \big\} (m-n) \times n \end{array} \equiv J^{(0)},$$

an upper bidiagonal matrix. If we let $A^{(1)} = A$ and define

$$A^{(k+\frac{1}{2})} = P^{(k)} A^{(k)} \qquad (k = 1, 2, \ldots, n)$$

$$A^{(k+1)} = A^{(k+\frac{1}{2})} Q^{(k)} \qquad (k = 1, 2, \ldots, n-2)$$

then $P^{(k)}$ is determined such that

$$a_{ik}^{(k+\frac{1}{2})} = 0 \quad (i = k+1, \ldots, m)$$

and $Q^{(k)}$ such that

$$a_{kj}^{(k+1)} = 0 \quad (j = k+2, \ldots, n).$$

The singular values of $J^{(0)}$ are the same as those of A. Thus, if the singular value decomposition of

$$J^{(0)} = G\Sigma H^T$$

then

$$A = PG\Sigma H^T Q^T$$

so that $U = PG$, $V = QH$ with $P \equiv P^{(1)} \dots P^{(n)}$, $Q \equiv Q^{(1)} \dots Q^{(n-2)}$.

1.3. *Singular Value Decomposition of the Bidiagonal Matrix*

By a variant of the QR algorithm, the matrix $J^{(0)}$ is iteratively diagonalized so that

$$J^{(0)} \to J^{(1)} \to \cdots \to \Sigma$$

where

$$J^{(i+1)} = S^{(i)T} J^{(i)} T^{(i)},$$

and $S^{(i)}$, $T^{(i)}$ are orthogonal. The matrices $T^{(i)}$ are chosen so that the sequence $M^{(i)} = J^{(i)T} J^{(i)}$ converges to a diagonal matrix while the matrices $S^{(i)}$ are chosen so that all $J^{(i)}$ are of the bidiagonal form. In [7], another technique for deriving $\{S^{(i)}\}$ and $\{T^{(i)}\}$ is given but this is equivalent to the method described below.

For notational convenience, we drop the suffix and use the notation

$$J \equiv J^{(i)},\quad \bar{J} \equiv J^{(i+1)},\quad S \equiv S^{(i)},\quad T \equiv T^{(i)},\quad M \equiv J^T J,\quad \bar{M} \equiv \bar{J}^T \bar{J}.$$

The transition $J \to \bar{J}$ is achieved by application of Givens rotations to J alternately from the right and the left. Thus

$$\bar{J} = \underbrace{S_n^T S_{(n-1)}^T \cdots S_2^T}_{S^T} J \underbrace{T_2 T_3 \dots T_n}_{T} \tag{2}$$

where

$$S_k = \begin{array}{c} \\ \left[\begin{array}{cccccccc} 1 & 0 & & & & & & \\ 0 & \ddots & & & & 0 & & \\ & & 1 & & & & & \\ & & & \cos\theta_k & -\sin\theta_k & & & \\ & & & \sin\theta_k & \cos\theta_k & & & \\ & & & & & 1 & & \\ & 0 & & & & & \ddots & 0 \\ & & & & & & 0 & 1 \end{array}\right] \end{array} \begin{array}{c} \\ \\ \\ (k-1) \\ (k) \\ \\ \\ \\ \end{array}$$

with the columns of $\cos\theta_k$ and $-\sin\theta_k$ labelled $(k-1)$ and (k) above,

and T_k is defined analogously to S_k with φ_k instead of θ_k.

Let the first angle, φ_2, be arbitrary while all the other angles are chosen so that $\bar{J}$ has the same form as J. Thus,

$$\begin{array}{ll} T_2 \text{ annihilates nothing,} & \text{generates an entry } \{J\}_{21}, \\ S_2^T \text{ annihilates } \{J\}_{21}, & \text{generates an entry } \{J\}_{13}, \\ T_3 \text{ annihilates } \{J\}_{13}, & \text{generates an entry } \{J\}_{32}, \\ \vdots & \end{array} \tag{3}$$

and finally

$$S_n^T \text{ annihilates } \{J\}_{n,n-1}, \text{ and generates nothing.}$$

(See Fig. 1.)

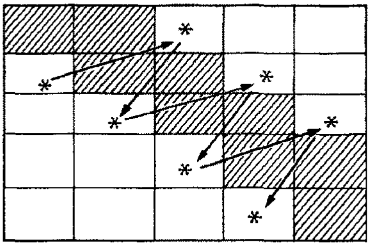


Fig. 1

This process is frequently described as "chasing". Since $\bar{J}=S^T J T$,

$$\bar{M}=\bar{J}^T\bar{J}=T^T M T$$

and $\bar{M}$ is a tri-diagonal matrix just as M is. We show that the first angle, φ_2, which is still undetermined, can be chosen so that the transition $M\to\bar{M}$ is a QR transformation with a given shift s.

The usual QR algorithm with shifts is described as follows:

$$\begin{aligned} M-sI&=T_s R_s\\ R_s T_s+sI&=\bar{M}_s \end{aligned} \tag{4}$$

where $T_s^T T_s=I$ and R_s is an upper triangular matrix. Thus $\bar{M}_s=T_s^T M T_s$. It has been shown by Francis [5] that it is not necessary to compute (4) explicitly but it is possible to perform the shift implicitly. Let T be for the moment an arbitrary matrix such that

$$\{T_s\}_{k,1}=\{T\}_{k,1}\quad (k=1,2,\dots,n),$$

(i.e., the elements of the first column of T_s are equal to the first column of T) and

$$T^T T=I.$$

Then we have the following theorem (Francis): If

i) $\bar{M}=T^T M T$,

ii) $\bar{M}$ is a tri-diagonal matrix,

iii) the sub-diagonal elements of M are non-zero,

it follows that $\bar{M}=D\bar{M}_s D$ where D is a diagonal matrix whose diagonal elements are ± 1.

Thus choosing T_2 in (3) such that its first column is proportional to that of $M-sI$, the same is true for the first column of the product $T=T_2 T_3 \dots T_n$ which therefore is identical to that of T_s. Hence, if the sub-diagonal of M does not contain any non-zero entry the conditions of the Francis theorem are fulfilled and T is therefore identical to T_s (up to a scaling of column ± 1). Thus the transition (2) is equivalent to the QR transformation of $J^T J$ with a given shift s.

The shift parameter s is determined by an eigenvalue of the lower 2×2 minor of M. Wilkinson [13] has shown that for this choice of s, the method converges globally and almost always cubically.

1.4. Test for Convergence

If $|e_n| \leqq \delta$, a prescribed tolerance, then $|q_n|$ is accepted as a singular value, and the order of the matrix is dropped by one. If, however, $|e_k| \leqq \delta$ for $k \neq n$, the matrix breaks into two, and the singular values of each block may be computed independently.

If $q_k = 0$, then at least one singular value must be equal to zero. In the absence of roundoff error, the matrix will break if a shift of zero is performed. Now, suppose at some stage

$$|q_k| \leqq \delta .$$

At this stage an extra sequence of Givens rotations is applied from the left to J involving rows $(k, k+1)$, $(k, k+2)$, ..., (k, n) so that

$$\begin{aligned} e_{k+1} \equiv \{J\}_{k,k+1} &\text{ is annihilated, but } \{J\}_{k,k+2},\ \{J\}_{k+1,k} \text{ are generated,} \\ \{J\}_{k,k+2} &\text{ is annihilated, but } \{J\}_{k,k+3},\ \{J\}_{k+2,k} \text{ are generated,} \\ &\vdots \end{aligned}$$

and finally

$$\{J\}_{k,n} \quad \text{is annihilated, and } \{J\}_{n,k} \text{ is generated.}$$

The matrix obtained thusly has the form

$$\begin{array}{cc} \qquad\qquad\qquad (k) & \\ \bar J = \begin{bmatrix} q_1 & e_2 & & & & & \\ & \ddots & \ddots & & & 0 & \\ & & \ddots & e_k & & & \\ & & & \bar q_k & 0 & & \\ & & & \delta_{k+1} & \bar q_{k+1} & \bar e_{k+2} & \\ & & & \vdots & & \ddots & \ddots \quad \bar e_n \\ 0 & & & \delta_n & & & \bar q_n \end{bmatrix} & (k). \end{array}$$

Note that by orthogonality

$$\bar q_k^2 + \delta_{k+1}^2 + \cdots + \delta_n^2 = q_k^2 \leqq \delta^2 .$$

Thus choosing $\delta = \|J^{(0)}\|_\infty \varepsilon_0$ (ε_0, the machine precision) ensures that all δ_k are less in magnitude than $\varepsilon_0 \|J^{(0)}\|_\infty$. Elements of $\bar J$ not greater than this are neglected. Hence $\bar J$ breaks up into two parts which may be treated independently.

2. Applicability

There are a large number of applications of the singular value decomposition; an extensive list is given in [7]. Some of these are as follows:

2.1. Pseudoinverse (Procedure SVD)

Let A be a real $m \times n$ matrix. An $n \times m$ matrix X is said to be the pseudoinverse of A if X satisfies the following four properties:

i) $AXA = A$,
ii) $XAX = X$,
iii) $(AX)^T = AX$,
iv) $(XA)^T = XA$.

The unique solution is denoted by A^+. It is easy to verify that if $A = U\Sigma V^T$, then $A^+ = V\Sigma^+ U^T$ where $\Sigma^+ = \operatorname{diag}(\sigma_i^+)$ and

$$\sigma_i^+ = \begin{cases} 1/\sigma_i & \text{for} \quad \sigma_i > 0 \\ 0 & \text{for} \quad \sigma_i = 0. \end{cases}$$

Thus the pseudoinverse may easily be computed from the output provided by the procedure *SVD*.

2.2. Solution of Homogeneous Equations (Procedure SVD or Procedure Minfit)

Let A be a matrix of rank r, and suppose we wish to solve

$$A x_i = \theta \quad \text{for} \quad i = r+1, \ldots, n$$

where θ denotes the null vector.

Let

$$U = [u_1, u_2, \ldots, u_n] \quad \text{and} \quad V = [v_1, v_2, \ldots, v_n].$$

Then since $A v_i = \sigma_i u_i$ $(i = 1, 2, \ldots n)$,

$$A v_i = \theta \quad \text{for} \quad i = r+1, \ldots, n$$

and $x_i = v_i$.

Here the procedure *SVD* or the procedure *Minfit* with $p = 0$ may be used for determining the solution. If the rank of A is known, then a modification of the algorithm of Businger and Golub [1] may be used.

2.3. Solutions of Minimal Length (Procedure Minfit)

Let b_1 be a given vector. Suppose we wish to determine a vector x so that

$$\|b_1 - A x\|_2 = \min \tag{5}$$

If the rank of A is less than n then there is no unique solution. Thus we require amongst all x which satisfy (5) that

$$\|\hat{x}\|_2 = \min$$

and this solution is unique. It is easy to verify that

$$\hat{x} = A^+ b_1 = V\Sigma^+ U^T b_1 \equiv V\Sigma^+ c_1.$$

The procedure *Minfit* with $p > 0$ will yield $V, \Sigma, c_1, \ldots, c_p$. Thus the user is able to determine which singular values are to be declared as zero.

2.4. A Generalization of the Least Squares Problem (Procedure SVD)

Let A be a real $m \times n$ matrix of rank n and let b be a given vector. We wish to construct a vector x such that

$$(A + \Delta A) x = b + \Delta b$$

and

$$\operatorname{trace}(\Delta A^T \Delta A) + K^2 \Delta b^T \Delta b = \min.$$

Here $K>0$ is a given weight and the standard problem is obtained for $K\to 0$. Introducing the augmented matrices $\bar{A}=(A, Kb)$ and $\Delta\bar{A}=(\Delta A, K\Delta b)$ and the vector

$$\bar{x}=\begin{pmatrix} x \\ -1/K \end{pmatrix},$$

we have to minimize $\operatorname{trace}(\Delta\bar{A}^T\Delta\bar{A})$ under the constraint $(\bar{A}+\Delta\bar{A})\,\bar{x}=0$. For fixed $\bar{x}$ the minimum is attained for $\Delta\bar{A}=-\bar{A}\,\bar{x}\,\bar{x}^T/\bar{x}^T\bar{x}$ and it has the value $\bar{x}^T\bar{A}^T\bar{A}\,\bar{x}/\bar{x}^T\bar{x}$. Minimizing with respect to $\bar{x}$ amounts to the computation of the smallest singular value of the matrix $\bar{A}$ and $\bar{x}$ is the corresponding column of the matrix $\bar{V}$ in the decomposition (1) with proper normalization [3].

3. Formal Parameter List

3.1. Input to Procedure SVD

m — number of rows of A, $m\geqq n$.

n — number of columns of A.

withu — **true** if U is desired, **false** otherwise.

withv — **true** if V is desired, **false** otherwise.

eps — a constant used in the test for convergence (see Section 5, (iii)); should not be smaller than the machine precision ε_0, i.e., the smallest number for which $1+\varepsilon_0>1$ in computer arithmetic.

tol — a machine dependent constant which should be set equal to β/ε_0 where β is the smallest positive number representable in the computer, see [11].

$a[1:m, 1:n]$ — represents the matrix A to be decomposed.

Output of procedure *SVD*.

$q[1:n]$ — a vector holding the singular values of A; they are non-negative but not necessarily ordered in decreasing sequence.

$u[1:m, 1:n]$ — represents the matrix U with orthonormalized columns (if *withu* is **true**, otherwise u is used as a working storage).

$v[1:n, 1:n]$ — represents the orthogonal matrix V (if *withv* is **true**, otherwise v is not used).

3.2. Input to Procedure Minfit

m — number of rows of A.

n — number of columns of A.

p — number of columns of B, $p\geqq 0$.

eps — same as for procedure *SVD*.

tol — same as for procedure *SVD*.

$ab[1:max(m,n), 1:n+p]$ — $ab[i,j]$ represents $a_{i,j}$, $1\leqq i\leqq m$, $1\leqq j\leqq n$,
$ab[i,n+j]$ represents $b_{i,j}$, $1\leqq i\leqq m$, $1\leqq j\leqq p$.

Output of procedure *Minfit*.

$ab[1:max(m,n), 1:n+p]$	$ab[i,j]$ represents $v_{i,j}$, $1 \leqq i \leqq n$, $1 \leqq j \leqq n$, $ab[i, n+j]$ represents $c_{i,j}$, $1 \leqq i \leqq max(m,n)$, $1 \leqq j \leqq p$ viz. $C = U_c^T B$.
$q[1:n]$	same as for procedure *SVD*.

4. ALGOL Programs

```
procedure SVD (m, n, withu, withv, eps, tol) data: (a) result: (q, u, v);
   value m, n, withu, withv, eps, tol;
   integer m, n;
   Boolean withu, withv;
   real eps, tol;
   array a, q, u, v;
comment Computation of the singular values and complete orthogonal decom-
         position of a real rectangular matrix A,
                     A = U diag(q) V^T,    U^T U = V^T V = I,
         where the arrays a[1:m, 1:n], u[1:m, 1:n], v[1:n, 1:n], q[1:n] re-
         present A, U, V, q respectively. The actual parameters corresponding
         to a, u, v may all be identical unless withu = withv = true. In this
         case, the actual parameters corresponding to u and v must differ.
         m ≧ n is assumed;
begin
   integer i, j, k, l, l1;
   real c, f, g, h, s, x, y, z;
   array e[1:n];
   for i := 1 step 1 until m do
   for j := 1 step 1 until n do u[i, j] := a[i, j];
comment Householder's reduction to bidiagonal form;
   g := x := 0;
   for i := 1 step 1 until n do
   begin
      e[i] := g; s := 0; l := i + 1;
      for j := i step 1 until m do s := s + u[j, i]↑2;
      if s < tol then g := 0 else
      begin
         f := u[i, i]; g := if f < 0 then sqrt(s) else -sqrt(s);
         h := f×g - s; u[i, i] := f - g;
         for j := l step 1 until n do
         begin
            s := 0;
            for k := i step 1 until m do s := s + u[k, i]×u[k, j];
            f := s/h;
            for k := i step 1 until m do u[k, j] := u[k, j] + f×u[k, i]
         end j
      end s;
```

```
        q[i] := g; s := 0;
        for j := l step 1 until n do s := s + u[i, j]↑2;
        if s < tol then g := 0 else
        begin
           f := u[i, i+1]; g := if f < 0 then sqrt(s) else -sqrt(s);
           h := f×g - s; u[i, i+1] := f - g;
           for j := l step 1 until n do e[j] := u[i, j]/h;
           for j := l step 1 until m do
           begin
              s := 0;
              for k := l step 1 until n do s := s + u[j, k]×u[i, k];
              for k := l step 1 until n do u[j, k] := u[j, k] + s×e[k]
           end j
        end s;
        y := abs(q[i]) + abs(e[i]); if y > x then x := y
     end i;

comment accumulation of right-hand transformations;
     if withv then for i := n step -1 until 1 do
     begin
        if g ≠ 0 then
        begin
           h := u[i, i+1] × g;
           for j := l step 1 until n do v[j, i] := u[i, j]/h;
           for j := l step 1 until n do
           begin
              s := 0;
              for k := l step 1 until n do s := s + u[i, k]×v[k, j];
              for k := l step 1 until n do v[k, j] := v[k, j] + s×v[k, i]
           end j
        end g;
        for j := l step 1 until n do v[i, j] := v[j, i] := 0;
        v[i, i] := 1; g := e[i]; l := i
     end i;

comment accumulation of left-hand transformations;
     if withu then for i := n step -1 until 1 do
     begin
        l := i + 1; g := q[i];
        for j := l step 1 until n do u[i, j] := 0;
        if g ≠ 0 then
        begin
           h := u[i, i]×g;
           for j := l step 1 until n do
```

```
            begin
                s := 0;
                for k := l step 1 until m do s := s + u[k, i] × u[k, j];
                f := s/h;
                for k := i step 1 until m do u[k, j] := u[k, j] + f × u[k, i]
            end j;
            for j := i step 1 until m do u[j, i] := u[j, i]/g
        end g
        else for j := i step 1 until m do u[j, i] := 0;
        u[i, i] := u[i, i] + 1
    end i;

comment diagonalization of the bidiagonal form;
    eps := eps × x;
    for k := n step −1 until 1 do
    begin
        test f splitting:
            for l := k step −1 until 1 do
            begin
                if abs(e[l]) ≦ eps then goto test f convergence;
                if abs(q[l−1]) ≦ eps then goto cancellation
            end l;

comment cancellation of e[l] if l > 1;
    cancellation:
        c := 0; s := 1; l1 := l−1;
        for i := l step 1 until k do
        begin
            f := s × e[i]; e[i] := c × e[i];
            if abs(f) ≦ eps then goto test f convergence;
            g := q[i]; h := q[i] := sqrt(f × f + g × g); c := g/h; s := −f/h;
            if withu then for j := 1 step 1 until m do
            begin
                y := u[j, l1]; z := u[j, i];
                u[j, l1] := y × c + z × s; u[j, i] := −y × s + z × c
            end j
        end i;
    test f convergence:
        z := q[k]; if l = k then goto convergence;

comment shift from bottom 2×2 minor;
        x := q[l]; y := q[k−1]; g := e[k−1]; h := e[k];
        f := ((y−z) × (y+z) + (g−h) × (g+h))/(2 × h × y); g := sqrt(f × f + 1);
        f := ((x−z) × (x+z) + h × (y/(if f < 0 then f−g else f+g) − h))/x;

comment next QR transformation;
        c := s := 1;
        for i := l+1 step 1 until k do
        begin
           g := e[i]; y := q[i]; h := s×g; g := c×g;
           e[i−1] := z := sqrt(f×f+h×h); c := f/z; s := h/z;
           f := x×c+g×s; g := −x×s+g×c; h := y×s; y := y×c;
           if withv then for j := 1 step 1 until n do
           begin
              x := v[j, i−1]; z := v[j, i];
              v[j, i−1] := x×c+z×s; v[j, i] := −x×s+z×c
           end j;
           q[i−1] := z := sqrt(f×f+h×h); c := f/z; s := h/z;
           f := c×g+s×y; x := −s×g+c×y;
           if withu then for j := 1 step 1 until m do
           begin
              y := u[j, i−1]; z := u[j, i];
              u[j, i−1] := y×c+z×s; u[j, i] := −y×s+z×c
           end j
        end i;
        e[l] := 0; e[k] := f; q[k] := x; goto test f splitting;
     convergence:
        if z < 0 then
        begin comment q[k] is made non-negative;
           q[k] := −z;
           if withv then for j := 1 step 1 until n do v[j, k] := −v[j, k]
        end z
     end k
end SVD;

procedure Minfit (m, n, p, eps, tol) trans: (ab) result: (q);
     value m, n, p, eps, tol;
     integer m, n, p;
     real eps, tol;
     array ab, q;
comment Computation of the matrices diag(q), V, and C such that for given
           real m×n matrix A and m×p matrix B

              U_c^T A V = diag(q) and U_c^T B = C with orthogonal matrices U_c and V.

           The singular values and the matrices V and C may be used to de-
           termine X̄ minimizing (1) ‖A X − B‖_F and (2) ‖X‖_F with the solution

                          X̄ = V × Pseudo-inverse of diag(q) × C.

           The procedure can also be used to determine the complete solution
           of an underdetermined linear system, i.e., rank(A) = m < n.
              The array q[1:n] represents the matrix diag(q). A and B together
           are to be given as the first m rows of the array ab[1:max(m,n), 1:n+p].
           V is returned in the first n rows and columns of ab while C is returned
           in the last p columns of ab (if p > 0);
```

```
begin
   integer i, j, k, l, l1, n1, np;
   real c, f, g, h, s, x, y, z;
   array e[1:n];

comment Householder's reduction to bidiagonal form;
   g:= x:= 0; np:= n+p;
   for i:= 1 step 1 until n do
   begin
      e[i]:= g; s:= 0; l:= i+1;
      for j:= i step 1 until m do s:= s+ab[j, i]↑2;
      if s< tol then g:= 0 else
      begin
         f:= ab[i, i]; g:= if f< 0 then sqrt(s) else -sqrt(s);
         h:= f×g-s; ab[i, i]:= f-g;
         for j:= l step 1 until np do
         begin
            s:= 0;
            for k:= i step 1 until m do s:= s+ab[k, i]×ab[k, j];
            f:= s/h;
            for k:= i step 1 until m do ab[k, j]:= ab[k, j]+f×ab[k, i]
         end j
      end s;
      q[i]:= g; s:= 0;
      if i≦m then for j:= l step 1 until n do s:= s+ab[i, j]↑2;
      if s< tol then g:= 0 else
      begin
         f:= ab[i, i+1]; g:= if f< 0 then sqrt(s) else -sqrt(s);
         h:= f×g-s; ab[i, i+1]:= f-g;
         for j:= l step 1 until n do e[j]:= ab[i, j]/h;
         for j:= l step 1 until m do
         begin
            s:= 0;
            for k:= l step 1 until n do s:= s+ab[j, k]×ab[i, k];
            for k:= l step 1 until n do ab[j, k]:= ab[j, k]+s×e[k]
         end j
      end s;
      y:= abs(q[i])+abs(e[i]); if y> x then x:= y
   end i;

comment accumulation of right-hand transformations;
   for i:= n step -1 until 1 do
   begin
      if g≠0 then
      begin
         h:= ab[i, i+1]×g;
         for j:= l step 1 until n do ab[j, i]:= ab[i, j]/h;
         for j:= l step 1 until n do
```

```
            begin
               s := 0;
               for k := l step 1 until n do s := s + ab[i, k] × ab[k, j];
               for k := l step 1 until n do ab[k, j] := ab[k, j] + s × ab[k, i]
            end j
         end g;
         for j := l step 1 until n do ab[i, j] := ab[j, i] := 0;
         ab[i, i] := 1; g := e[i]; l := i
      end i;
      eps := eps × x; n1 := n + 1;
      for i := m + 1 step 1 until n do
      for j := n1 step 1 until np do ab[i, j] := 0;

comment diagonalization of the bidiagonal form;
      for k := n step −1 until 1 do
      begin
         test f splitting:
            for l := k step −1 until 1 do
            begin
               if abs(e[l]) ≦ eps then goto test f convergence;
               if abs(q[l − 1]) ≦ eps then goto cancellation
            end l;

comment cancellation of e[l] if l > 1;
         cancellation:
            c := 0; s := 1; l1 := l − 1;
            for i := l step 1 until k do
            begin
               f := s × e[i]; e[i] := c × e[i];
               if abs(f) ≦ eps then goto test f convergence;
               g := q[i]; q[i] := h := sqrt(f × f + g × g); c := g/h; s := −f/h;
               for j := n1 step 1 until np do
               begin
                  y := ab[l1, j]; z := ab[i, j];
                  ab[l1, j] := c × y + s × z; ab[i, j] := −s × y + c × z
               end j
            end i;
         test f convergence:
            z := q[k]; if l = k then goto convergence;

comment shift from bottom 2 × 2 minor;
            x := q[l]; y := q[k − 1]; g := e[k − 1]; h := e[k];
            f := ((y − z) × (y + z) + (g − h) × (g + h))/(2 × h × y); g := sqrt(f × f + 1);
            f := ((x − z) × (x + z) + h × (y/(if f < 0 then f − g else f + g) − h))/x;

comment next QR transformation;
            c := s := 1;
            for i := l + 1 step 1 until k do
```

```
        begin
           g := e[i]; y := q[i]; h := s×g; g := c×g;
           e[i-1] := z := sqrt(f×f+h×h); c := f/z; s := h/z;
           f := x×c+g×s; g := -x×s+g×c; h := y×s; y := y×c;
           for j := 1 step 1 until n do
           begin
              x := ab[j, i-1]; z := ab[j, i];
              ab[j, i-1] := x×c+z×s; ab[j, i] := -x×s+z×c
           end j;
           q[i-1] := z := sqrt(f×f+h×h); c := f/z; s := h/z;
           f := c×g+s×y; x := -s×g+c×y;
           for j := n1 step 1 until np do
           begin
              y := ab[i-1, j]; z := ab[i, j];
              ab[i-1, j] := c×y+s×z; ab[i, j] := -s×y+c×z
           end j
        end i;
        e[l] := 0; e[k] := f; q[k] := x; goto test f splitting;
     convergence:
        if z < 0 then
        begin comment q[k] is made non-negative;
           q[k] := -z;
           for j := 1 step 1 until n do ab[j, k] := -ab[j, k]
        end z
  end k
end Minfit;
```

5. Organizational and Notational Details

(i) The matrix U consists of the first n columns of an orthogonal matrix U_c. The following modification of the procedure *SVD* would produce U_c instead of U: After

```
comment accumulation of left-hand transformations;
```

insert a statement

```
if withu then for i := n+1 step 1 until m do
begin
   for j := n+1 step 1 until m do u[i, j] := 0;
   u[i, i] := 1
end i;
```

Moreover, replace n by m in the fourth and eighth line after that, i.e., write twice **for** $j := l$ **step** 1 **until** m **do**.

(ii) $m \geqq n$ is assumed for procedure *SVD*. This is no restriction; if $m < n$, store A^T, i.e., use an array $at[1:n, 1:m]$ where $at[i, j]$ represents $a_{j,i}$ and call $SVD(n, m, withv, withu, eps, tol, at, q, v, u)$ producing the $m \times m$ matrix U and the $n \times m$ matrix V. There is no restriction on the values of m and n for the procedure *Minfit*.

(iii) In the iterative part of the procedures an element of $J^{(i)}$ is considered to be negligible and is consequently replaced by zero if it is not larger in magnitude than εx where ε is the given tolerance and

$$x = \max_{1 \leqq i \leqq n} (|q_i| + |e_i|).$$

The largest singular value σ_1 is bounded by $x/\sqrt{2} \leqq \sigma_1 \leqq x\sqrt{2}$.

(iv) A program organization was chosen which allows us to save storage locations. To this end the actual parameters corresponding to a and u may be identical. In this event the original information stored in a is overwritten by information on the reduction. This, in turn, is overwritten by u if the latter is desired. Likewise, the actual parameters corresponding to a and v may agree. Then v is stored in the upper part of a if it is desired, otherwise a is not changed. Finally, all three parameters a, u, and v may be identical unless *withu* = *withv* = **true**.

This special feature, however, increases the number of multiplications needed to form U roughly by a factor m/n.

(v) Shifts are evaluated in a way as to reduce the danger of overflow or underflow of exponents.

(vi) The singular values as delivered in the array q are not necessarily ordered. Any sorting of them should be accompanied by the corresponding sorting of the columns of U and V, and of the rows of C.

(vii) The formal parameter list may be completed by the addition of a limit for the number of iterations to be performed, and by the addition of a failure exit to be taken if no convergence is reached after the specified number of iterations (e.g., 30 per singular value).

6. Numerical Properties

The stability of the Householder transformations has been demonstrated by Wilkinson [12]. In addition, he has shown that in the absence of roundoff the QR algorithm has global convergence and asymptotically is almost always cubically convergent.

The numerical experiments indicate that the average number of complete QR iterations on the bidiagonal matrix is usually less than two per singular value. Extra consideration must be given to the implicit shift technique which fails for a split matrix. The difficulties arise when there are small q_k's or e_k's. Using the techniques of Section 1.4, there cannot be numerical instability since stable orthogonal transformations are used but under special circumstances there may be a slowdown in the rate of convergence.

7. Test Results

Tests were carried out on the UNIVAC 1108 Computer of the Andrew R. Jennings Computing Center of Case Western Reserve University. Floating point numbers are represented by a normalized 27 bit mantissa and a 7 bit exponent to the radix 2, whence $eps = 1.5_{10}-8$, $tol = {}_{10}-31$. In the following, computed values are marked by a tilde and $m(A)$ denotes $\max|a_{i,j}|$.

First example:

$$A=\begin{bmatrix} 22 & 10 & 2 & 3 & 7\\ 14 & 7 & 10 & 0 & 8\\ -1 & 13 & -1 & -11 & 3\\ -3 & -2 & 13 & -2 & 4\\ 9 & 8 & 1 & -2 & 4\\ 9 & 1 & -7 & 5 & -1\\ 2 & -6 & 6 & 5 & 1\\ 4 & 5 & 0 & -2 & 2 \end{bmatrix},\qquad B=\begin{bmatrix} -1 & 1 & 0\\ 2 & -1 & 1\\ 1 & 10 & 11\\ 4 & 0 & 4\\ 0 & -6 & -6\\ -3 & 6 & 3\\ 1 & 11 & 12\\ 0 & -5 & -5 \end{bmatrix},$$

$$\sigma_1=\sqrt{1248},\qquad \sigma_2=20,\qquad \sigma_3=\sqrt{384},\qquad \sigma_4=\sigma_5=0.$$

The homogeneous system $A x=0$ has two linearly independent solutions. Six QR transformations were necessary to drop all off-diagonal elements below the internal tolerance $46.4_{10}-8$. Table 1 gives the singular values in the sequence as computed by procedures *SVD* and *Minfit*. The accuracy of the achieved decomposition is characterized by

$$m(A-\tilde U\tilde\Sigma\tilde V^T)=238_{10}-8,\quad m(\tilde U^T\tilde U-I)=8.1_{10}-8,\quad m(\tilde V^T\tilde V-I)=3.3_{10}-8.$$

Because two singular values are equal to zero, the procedures *SVD* and *Minfit* may lead to other orderings of the singular values for this matrix when other tolerances are used.

Table 1

$\tilde\sigma_k$	$\sigma_k-\tilde\sigma_k$	
$0.96_{10}-7$	-9.6	$\times 10^{-8}$
19.595916	191	
19.999999	143	
$1.97_{10}-7$	-19.7	
35.327038	518	

The computed solutions of the homogeneous system are given by the first and fourth columns of the matrix $\tilde V$ (Table 2).

Table 2

$\tilde v_1$	$\tilde v_4$	$v_1-\tilde v_1$	$v_4-\tilde v_4$	
−0.4190 9545	0	−1.5	0 (Def.)	$\times 10^{-8}$
0.4405 0912	0.4185 4806	1.7	0.6	
−0.0520 0457	0.3487 9006	1.2	−1.3	
0.6760 5915	0.2441 5305	1.0	0.3	
0.4129 7730	−0.8022 1713	1.3	−0.8	

Procedure *Minfit* was used to compute the solutions of the minimization problem of Section 2.3 corresponding to the three right-hand sides as given by the columns of the matrix B. Table 3 lists the exact solutions and the results obtained when the first and fourth values in Table 1 are replaced by zero.

Table 3

x_1	x_2	x_3	$\tilde{x}_1$	$\tilde{x}_2$	$\tilde{x}_3$
$-1/12$	0	$-1/12$	$-0.0833\,3333$	$0.17_{10}-8$	$-0.0833\,3333$
0	0	0	$-0.58_{10}-8$	$-1.09_{10}-8$	$-1.11_{10}-8$
$1/4$	0	$1/4$	$0.2500\,0002$	$1.55_{10}-8$	$0.2500\,0003$
$-1/12$	0	$-1/12$	$-0.0833\,3332$	$0.74_{10}-8$	$-0.0833\,3332$
$1/12$	0	$1/12$	$0.0833\,3334$	$0.33_{10}-8$	$0.0833\,3334$
	Residual				
0	$8\sqrt{5}$	$8\sqrt{5}$			

A second example is the 20×21 matrix with entries

$$a_{i,j}=\begin{cases}0 & \text{if} \quad i>j\\ 21-i & \text{if} \quad i=j\\ -1 & \text{if} \quad i<j\end{cases} \qquad \begin{matrix}1\leqq i\leqq 20\\ 1\leqq j\leqq 21\end{matrix}$$

which has orthogonal rows and singular values $\sigma_{21-k}=\sqrt{k(k+1)}$, $k=0,\ldots,20$. Theoretically, the Householder reduction should produce a matrix $J^{(0)}$ with diagonal $-20, 0, \ldots, 0$ and super-diagonal $-\sqrt{20}, \sigma_2, \ldots, \sigma_{20}$. Under the influence of rounding errors a totally different matrix results. However, within working accuracy its singular values agree with those of the original matrix. Convergence is reached after 32 QR transformations and the $\tilde{\sigma}_k$, $k=1,\ldots,20$ are correct within several units in the last digit, $\tilde{\sigma}_{21}=1.61_{10}-11$.

A third example is obtained if the diagonal of the foregoing example is changed to

$$a_{i,i}=1, \qquad 1\leqq i\leqq 20.$$

This matrix has a cluster of singular values, σ_{10} to σ_{19} lying between 1.5 and 1.6, $\sigma_{20}=\sqrt{2}$, $\sigma_{21}=0$. Clusters, in general, have a tendency to reduce the number of required iterations; in this example, 26 iterations were necessary for convergence. $\tilde{\sigma}_{21}=1.49_{10}-8$ is found in eighteenth position and the corresponding column of $\tilde{V}$ differs from the unique solution of the homogeneous system by less than $3.4_{10}-8$ in any component.

A second test was made by Dr. Peter Businger on the CDC 6600.

A third test was performed on the IBM 360/67 at Stanford University. The example used was the 30×30 matrix with entries

$$a_{ij}=\begin{cases}0 & \text{if} \quad i>j\\ 1 & \text{if} \quad i=j\\ -1 & \text{if} \quad i<j.\end{cases}$$

The computed singular values are given in Table 4.

Table 4. *Singular values*

18.2029 0555 7529 2200	6.2231 9652 2604 2340	3.9134 8020 3335 6160	2.9767 9450 2557 7960
2.4904 5062 9660 3570	2.2032 0757 4479 9280	2.0191 8365 4054 5860	1.8943 4154 7685 6890
1.8059 1912 6612 3070	1.7411 3576 7747 9500	1.6923 5654 4395 2610	1.6547 9302 7369 3370
1.6253 2089 2877 9290	1.6018 3335 6666 2670	1.5828 6958 8713 6990	1.5673 9214 4480 0070
1.5546 4889 0109 3720	1.5440 8471 4076 0510	1.5352 8356 5544 9020	1.5279 2951 2160 3040
1.5217 8003 9063 4950	1.5166 4741 2836 7840	1.5123 8547 3899 6950	1.5088 8015 6801 8850
1.5060 4262 0723 9700	1.5038 0424 3812 6520	1.5021 1297 6754 0060	1.5009 3071 1977 0610
1.5002 3143 4775 4370	0.0000 0000 2793 9677		

Note that $\sigma_{30}/\sigma_1 \approx 1.53 \times 10^{-10}$ so that this matrix is very close to being a matrix of rank 29 even though the determinant equals 1.

Acknowledgement. The authors wish to thank Dr. Peter Businger of Bell Telephone Laboratories for his stimulating comments.

References

1. Businger, P., Golub, G.: Linear least squares solutions by Householder transformations. Numer. Math. 7, 269—276 (1965).
2. Forsythe, G.E., Henrici, P.: The cyclic Jacobi method for computing the principal values of a complex matrix. Proc. Amer. Math. Soc. **94**, 1—23 (1960).
3. — Golub, G.: On the stationary values of a second-degree polynomial on the unit sphere. J. Soc. Indust. Appl. Math. **13**, 1050—1068 (1965).
4. — Moler, C.B.: Computer solution of linear algebraic systems. Englewood Cliffs, New Jersey: Prentice-Hall 1967.
5. Francis, J.: The QR transformation. A unitary analogue to the LR transformation. Comput. J. **4**, 265—271 (1961, 1962).
6. Golub, G., Kahan, W.: Calculating the singular values and pseudo-inverse of a matrix. J. SIAM. Numer. Anal., Ser. B **2**, 205—224 (1965).
7. — Least squares, singular values, and matrix approximations. Aplikace Matematiky **13**, 44—51 (1968).
8. Hestenes, M.R.: Inversion of matrices by biorthogonalization and related results. J. Soc. Indust. Appl. Math. **6**, 51—90 (1958).
9. Kogbetliantz, E.G.: Solution of linear equations by diagonalization of coefficients matrix. Quart. Appl. Math. **13**, 123—132 (1955).
10. Kublanovskaja, V.N.: Some algorithms for the solution of the complete problem of eigenvalues. V. Vyčisl. Mat. i. Mat. Fiz. **1**, 555—570 (1961).
11. Martin, R.S., Reinsch, C., Wilkinson, J.H.: Householder's tridiagonalization of a symmetric matrix. Numer. Math. **11**, 181—195 (1968).
12. Wilkinson, J.: Error analysis of transformations based on the use of matrices of the form $I-2w\,w^H$. Error in digital computation, vol. II, L.B. Rall, ed., p. 77—101. New York: John Wiley & Sons, Inc. 1965.
13. — Global convergence of QR algorithm. Proceedings of IFIP Congress, 1968.

Prof. G.H. Golub
Computer Science Dept.
Stanford University
Stanford, California 94305
USA

Dr. C. Reinsch
Math. Institut
der Techn. Hochschule
8000 München 2
Arcisstr. 21

13

THE DIFFERENTIATION OF PSEUDO-INVERSES AND NON-LINEAR LEAST SQUARES PROBLEMS WHOSE VARIABLES SEPARATE (WITH V. PEREYRA)

SIAM J. Numer. Anal. (10), 413–432 (1973).

The work on this paper originated during a 1972 visit to the Northern Patagonia city of Bariloche, Argentina, by invitation of Hugo Scolnik. Although a skiing resort, Bariloche was best visited at the time during the summer due to its lack of roads, rather than the winter, when we spent three weeks there. The work continued by snail mail (I was in Caracas, Venezuela at the time) and during my periodical visits to Serra House at Stanford.

Interestingly enough, due to Gene's long connections with Zürich, the idea of separable nonlinear least squares came from the Doctoral Thesis of Scolnik at the University of Zürich, which was signed by Rutishauser.

This work has had a long list of applications through the years and in 2002 we decided that a review was due, in order to put together many of these applications to disparate fields such as medicine, robotics, communications, mechanics, and mathematics. This review appeared in *Inverse Problems*, 2003. By the way, this paper had a record number of downloads from the Inverse Problems site in 2003–2004.

Gene has been an icon and a great supporter of foreign students and young scholars all his career. After 42 years of friendship I have been a witness many times to his kindness and to his dedication to scholarship, education, and plain humanity.

V. Pereyra
Cupertino, CA, USA

SIAM J. NUMER. ANAL.
Vol. 10, No. 2, April 1973.

THE DIFFERENTIATION OF PSEUDO-INVERSES AND NONLINEAR LEAST SQUARES PROBLEMS WHOSE VARIABLES SEPARATE*

G. H. GOLUB† AND V. PEREYRA‡

Abstract. For given data (t_i, y_i), $i = 1, \cdots, m$, we consider the least squares fit of nonlinear models of the form

$$\eta(\mathbf{a}, \boldsymbol{\alpha}; t) = \sum_{j=1}^{n} a_j \varphi_j(\boldsymbol{\alpha}; t), \qquad \mathbf{a} \in \mathscr{R}^n, \quad \boldsymbol{\alpha} \in \mathscr{R}^k.$$

For this purpose we study the minimization of the nonlinear functional

$$r(\mathbf{a}, \boldsymbol{\alpha}) = \sum_{i=1}^{m} (y_i - \eta(\mathbf{a}, \boldsymbol{\alpha}, t_i))^2.$$

It is shown that by defining the matrix $\{\Phi(\boldsymbol{\alpha})\}_{i,j} = \varphi_j(\boldsymbol{\alpha}; t_i)$, and the modified functional $r_2(\boldsymbol{\alpha}) = \|\mathbf{y} - \Phi(\boldsymbol{\alpha})\Phi^+(\boldsymbol{\alpha})\mathbf{y}\|_2^2$, it is possible to optimize first with respect to the parameters $\boldsymbol{\alpha}$, and then to obtain, a posteriori, the optimal parameters $\hat{\mathbf{a}}$. The matrix $\Phi^+(\boldsymbol{\alpha})$ is the Moore–Penrose generalized inverse of $\Phi(\boldsymbol{\alpha})$. We develop formulas for the Fréchet derivative of orthogonal projectors associated with $\Phi(\boldsymbol{\alpha})$ and also for $\Phi^+(\boldsymbol{\alpha})$, under the hypothesis that $\Phi(\boldsymbol{\alpha})$ is of constant (though not necessarily full) rank. Detailed algorithms are presented which make extensive use of well-known reliable linear least squares techniques, and numerical results and comparisons are given. These results are generalizations of those of H. D. Scolnik [20] and Guttman, Pereyra and Scolnik [9].

1. Introduction. The least squares fit of experimental data is a common tool in many applied sciences and in engineering problems. Linear problems have been well studied, and stable and efficient methods are available (see, for instance, Björck and Golub [3], Golub [8]).

Methods for the nonlinear problems fall mainly in two categories: (a) general minimization techniques; (b) methods of Gauss–Newton type. The latter type of method takes into consideration the fact that the functional to be minimized is a sum of squares of functions (cf. Daniel [5], Osborne [14], Pereyra [15]). The well-known reliable linear techniques have been used mainly in connection with the successive linearization of the nonlinear models. Very recently it has been noticed that by restricting the class of models to be treated, a much more significant use of linear techniques can be made (cf. [2], [9], [12], [13], [17], [20], [23]–[26], [36]).

In this paper we consider the following problem. Given data (t_i, y_i), $i = 1, \cdots, m$, find optimal parameters $\hat{\mathbf{a}} = (\hat{a}_1, \cdots, \hat{a}_n)^{\mathrm{T}}$, $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \cdots, \hat{\alpha}_k)^{\mathrm{T}}$ that minimize the nonlinear functional

$$r(\mathbf{a}, \boldsymbol{\alpha}) = \sum_{i=1}^{m} \left[y_i - \sum_{j=1}^{n} a_j \varphi_j(\boldsymbol{\alpha}; t_i) \right]^2. \tag{1.1}$$

Throughout this paper a lower-case letter in boldface will indicate a column vector, while the same letter with a subscript will indicate a component of the

* Received by the editors April 13, 1972, and in revised form September 26, 1972.

† Computer Science Department, Stanford University, Stanford, California 94305. The work of this author was supported in part by the Atomic Energy Commission.

Gene H. Golub is Professor of Computer Science at Stanford University where, since 1962, he was a close colleague of Professor Forsythe.

‡ Departamento de Computacion, Universidad Central de Venezuela, Caracas, Venezuela.

vector. Matrices which are not vectors are denoted by capital letters, and the (i, j) element of (say) a matrix A will be indicated by either a_{ij} or $\{A\}_{i,j}$. The transpose of a vector $\mathbf{u}$ is indicated by $\mathbf{u}^{\mathrm{T}}$. Given a function $f(t)$, we shall denote by $\mathbf{f}$ the vector whose components are $(f(t_1), f(t_2), \cdots, f(t_m))^{\mathrm{T}}$. The scalar product of two vectors $\mathbf{u}$ and $\mathbf{v}$ is indicated by

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{v}^{\mathrm{T}}\mathbf{u}.$$

The only norm which will be used is the Euclidean norm, viz. $\|\mathbf{v}\|^2 = \langle \mathbf{v}, \mathbf{v} \rangle$. Given a matrix A and a vector $\mathbf{b}$, then we say

$$A\hat{\mathbf{x}} \cong \mathbf{b}$$

if $\|\mathbf{b} - A\hat{\mathbf{x}}\| \leqq \|\mathbf{b} - A\mathbf{x}\|$ for all $\mathbf{x} \in R^n$.

We shall use the symbol $\mathbf{D}$ for the Fréchet derivative of a mapping and ∇ for the gradient of a functional. We assume that the reader has some familiarity with pseudo-inverses and Fréchet derivatives and their properties. A useful reference for the pseudo-inverse is [19]; for details on the formalism and manipulation of Fréchet derivatives, we suggest [6, Chap. 8].

Let

$$\{\Phi\}_{i,j} = \varphi_j(\boldsymbol{\alpha}; t_i), \quad i = 1, \cdots, m; \quad j = 1, 2, \cdots, n,$$

and

$$\mathbf{a} = (a_1, \cdots, a_n)^{\mathrm{T}}.$$

With the given notation, we can rewrite (1.1) as

$$r(\mathbf{a}, \boldsymbol{\alpha}) = \|\mathbf{y} - \Phi(\boldsymbol{\alpha})\mathbf{a}\|^2. \tag{1.2}$$

Our approach to finding a critical point or a minimum of the functional (1.2) requires an important hypothesis.

H1. *The matrix* $\Phi(\boldsymbol{\alpha})$ *has constant rank* $r \leqq \min(m, n)$ *for* $\boldsymbol{\alpha} \in \Omega \subset \mathscr{R}^k$, Ω *being an open set containing the desired solution.*

We say that a matrix function satisfying H1 on an open neighborhood of a point $\boldsymbol{\alpha}^0$ has *local constant rank* at $\boldsymbol{\alpha}^0$.

Our aim is firstly to minimize a modified functional which depends only on the nonlinear parameters $\boldsymbol{\alpha}$, and then to proceed to obtain the linear parameters $\mathbf{a}$. In [9], [20] simpler models were treated, i.e., $\varphi_j(\boldsymbol{\alpha}; t) = t^{\alpha_j}$, and $\varphi_j(\boldsymbol{\alpha}; t) = \varphi_j(\alpha_j; t)$. A similar point of view was used but different analytic tools were employed. The reader should also note the independent results obtained by Pérez and Scolnik [17], who in addition deal with nonlinear constraints.

In order to obtain the separation of variables we consider, as in [9], [17], [20], the modified functional

$$r_2(\boldsymbol{\alpha}) = \|\mathbf{y} - \Phi(\boldsymbol{\alpha})\Phi^{+}(\boldsymbol{\alpha})\mathbf{y}\|^2, \tag{1.3}$$

which will be called the *variable projection functional.* Once optimal parameters $\hat{\boldsymbol{\alpha}}$ have been obtained by minimizing (1.3), then the parameters $\hat{\mathbf{a}}$ are obtained as a solution of $\Phi(\hat{\boldsymbol{\alpha}})\mathbf{a} \cong \mathbf{y}$.

We shall show in Theorem 2.1 the relationship between critical or minimal points of the original functional $r(\mathbf{a}, \boldsymbol{\alpha})$ and those obtained from the functional $r_2(\boldsymbol{\alpha})$.

Both for our proof and for the numerical algorithms of § 5 we need to develop formulas for the gradient of the functional (1.3). In § 4 we develop these formulas and obtain the derivatives of projectors, the Jacobian of the residual vector, and the pseudo-inverse of a matrix function. The only hypothesis we make on the rank of the matrix is that it must be locally constant at the point where the derivative is calculated. This is necessary, since otherwise the pseudo-inverse is not a continuous function, and therefore it could hardly be differentiable. Our proof and final results are coordinate-free. For the full rank case similar formulas have been obtained for the pseudo-inverse by Fletcher and Lill [7] (without proof), by Hanson and Lawson [10], and by Pérez and Scolnik [17]. In [7] and [17] this is used to deal with constraints via penalty functions, while in [17] the authors also obtain a formula for the rank deficient case in terms of a full rank factorization of the given matrix. Our formula in turn is given exclusively in terms of the original matrix, its derivative, and its pseudo-inverse, and it seems to be new.[1]

In § 5 we give a detailed explanation of how to implement the method in an efficient way and in § 6 we present some numerical examples and comparisons. Extensive use is made of linear least squares techniques. A FORTRAN program based on the ideas of this paper is given in [25].

2. A class of nonlinear least squares problems whose parameters separate. We are going to consider in this paper models of the form

$$\eta(\mathbf{a}, \boldsymbol{\alpha}; t) = \sum_{j=1}^{n} a_j \varphi_j(\boldsymbol{\alpha}; t), \tag{2.1}$$

where $\mathbf{a} \in \mathscr{R}^n$, $\boldsymbol{\alpha} \in \mathscr{R}^k$, and the functions φ_j are continuously differentiable with respect to $\boldsymbol{\alpha}$. We remark that the parameters $\mathbf{a}$ and $\boldsymbol{\alpha}$ form two completely disjoint sets.

The independent variable t could be a vector itself as in [9], [17]. This requires only small notational changes and we shall not pursue it here.

Given the data (t_i, y_i), $i = 1, \cdots, m$, $m \geqq n + k$, our task is to find the values of the parameters $\mathbf{a}, \boldsymbol{\alpha}$, that minimize the nonlinear functional

$$r(\mathbf{a}, \boldsymbol{\alpha}) = \|\mathbf{y} - \boldsymbol{\eta}(\mathbf{a}, \boldsymbol{\alpha})\|^2 = \sum_{i=1}^{m} (y_i - \eta(\mathbf{a}, \boldsymbol{\alpha}; t_i))^2. \tag{2.2}$$

The approach to the solution of this problem is, as in [9], [17], [20], to modify the functional $r(\mathbf{a}, \boldsymbol{\alpha})$, in such a way that consideration of the linear parameters $\mathbf{a}$ is deferred.

In what follows we call $\Phi(\boldsymbol{\alpha})$ the matrix function

$$\Phi(\boldsymbol{\alpha}) = [\boldsymbol{\varphi}_1(\boldsymbol{\alpha}), \cdots, \boldsymbol{\varphi}_n(\boldsymbol{\alpha})]. \tag{2.3}$$

For each $\boldsymbol{\alpha}$, the linear operator

$$P_{\Phi(\boldsymbol{\alpha})} = \Phi(\boldsymbol{\alpha})\Phi^{+}(\boldsymbol{\alpha}) \tag{2.4}$$

is the orthogonal projector on the linear space spanned by the columns of the matrix $\Phi(\boldsymbol{\alpha})$. We denote the linear operator $(I - P_{\Phi(\boldsymbol{\alpha})})$ by $P^{\perp}_{\Phi(\boldsymbol{\alpha})}$. The operator

[1] As this paper was being sent to the printers, the authors learned that Decell and Fries [34] had also obtained this result recently.

$P^{\perp}_{\Phi(\boldsymbol{\alpha})}$ is the projector on the orthogonal complement of the column space of $\Phi(\boldsymbol{\alpha})$. Similarly,

$$(2.4') \qquad {}_{\Phi}P = \Phi^{+}\Phi$$

is the orthogonal projector on the row space of Φ, and ${}_{\Phi}P^{\perp} = I - {}_{\Phi}P$. When there is no possibility of confusion we shall omit either the matrix subindex or the arguments in projections and functions, or both.

For any given $\boldsymbol{\alpha}$ we have the minimal least squares solution

$$(2.5) \qquad \hat{\mathbf{a}}(\boldsymbol{\alpha}) \equiv \Phi^{+}(\boldsymbol{\alpha})\mathbf{y}.$$

Thus,

$$(2.6) \qquad \min_{\mathbf{a}} r(\mathbf{a},\boldsymbol{\alpha}) = r(\hat{\mathbf{a}},\boldsymbol{\alpha}) = \|\mathbf{y} - \Phi(\boldsymbol{\alpha})\Phi^{+}(\boldsymbol{\alpha})\mathbf{y}\|^2 = \|P^{\perp}_{\Phi(\boldsymbol{\alpha})}\mathbf{y}\|^2.$$

The modified functional is then the variable projection functional that we mentioned earlier and can be rewritten as

$$(2.7) \qquad r_2(\boldsymbol{\alpha}) = \|P^{\perp}_{\Phi(\boldsymbol{\alpha})}\mathbf{y}\|^2.$$

Once a critical point (or a minimizer) $\hat{\boldsymbol{\alpha}}$ is found for this functional, then $\hat{\mathbf{a}}$ is obtained by replacing $\boldsymbol{\alpha}$ by $\hat{\boldsymbol{\alpha}}$ in (2.5).

The justification for employing this procedure is given by the following theorem.

THEOREM 2.1. *Let $r(\mathbf{a},\boldsymbol{\alpha})$ and $r_2(\boldsymbol{\alpha})$ be defined as above. We assume that in the open set $\Omega \subset \mathscr{R}^k$, $\Phi(\boldsymbol{\alpha})$ has constant rank $r \leqq \min(m, n)$.*

(a) *If $\hat{\mathbf{a}}$ is a critical point (or a global minimizer in Ω) of $r_2(\boldsymbol{\alpha})$, and*

$$(2.8) \qquad \hat{\mathbf{a}} = \Phi^{+}(\hat{\boldsymbol{\alpha}})\mathbf{y},$$

then $(\hat{\mathbf{a}},\hat{\boldsymbol{\alpha}})$ is a critical point of $r(\mathbf{a},\boldsymbol{\alpha})$ (or a global minimizer for $\boldsymbol{\alpha} \in \Omega$) and $r(\hat{\mathbf{a}},\hat{\boldsymbol{\alpha}}) = r_2(\hat{\boldsymbol{\alpha}})$.

(b) *If $(\hat{\mathbf{a}},\hat{\boldsymbol{\alpha}})$ is a global minimizer of $r(\mathbf{a},\boldsymbol{\alpha})$ for $\boldsymbol{\alpha} \in \Omega$, then $\hat{\boldsymbol{\alpha}}$ is a global minimizer of $r_2(\boldsymbol{\alpha})$ in Ω and $r_2(\hat{\boldsymbol{\alpha}}) = r(\hat{\mathbf{a}},\hat{\boldsymbol{\alpha}})$. Furthermore, if there is an unique $\hat{\mathbf{a}}$ among the minimizing pairs of $r(\mathbf{a},\boldsymbol{\alpha})$, then $\hat{\mathbf{a}}$ must satisfy* (2.8).

We shall postpone the proof of this theorem until the end of § 4, where we obtain a convenient expression for the gradient of the functional $r_2(\boldsymbol{\alpha})$.

Although separation of variables has been used elsewhere as indicated earlier, the correspondence between the critical points of $r(\mathbf{a},\boldsymbol{\alpha})$ and $r_2(\boldsymbol{\alpha})$ has only been studied before in [9] for a simpler case. See also [36].

3. Algorithmia I. Residual calculation. One of our main points in the algorithmic parts of this paper is to emphasize, when possible and appropriate, the use of stable and efficient linear least squares techniques. Thus it is convenient to review some of the tools and to introduce the necessary notation.

If Q is an orthogonal matrix, then for every vector $\mathbf{z}$, $\|Q\mathbf{z}\| = \|\mathbf{z}\|$. It is well known (cf. [8], [10], [18], [22]) that every $m \times n$ matrix Φ ($m \geqq n$) of rank $r \leqq n$

can be orthogonally transformed into "trapezoidal" form. That is, there exists an orthogonal matrix Q and a permutation matrix S such that

$$Q\Phi S = \left[\begin{array}{c|c} T_{11} & T_{12} \\ \hline 0 & 0 \end{array}\right] \equiv T_0, \tag{3.1}$$

where T_{11} is an $r \times r$ nonsingular upper triangular matrix. Naturally, $\Phi = Q^{\mathrm{T}} T_0 S^{\mathrm{T}}$.

We indicate by Φ^- any $n \times m$ matrix which satisfies the two properties

$$\Phi\Phi^-\Phi = \Phi, \qquad (\Phi\Phi^-)^{\mathrm{T}} = (\Phi\Phi^-). \tag{3.2}$$

We observe (cf. [19], [22]) that

$$P_\Phi = \Phi\Phi^-,$$

and hence Φ^+ is not necessary for computing P_Φ.

From the decomposition (3.1), we can obtain a Φ^-. Let

$$\Phi^B = S\begin{bmatrix} T_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix} Q. \tag{3.3}$$

It is easy to verify that Φ^B satisfies (3.2) and also $\Phi^B\Phi\Phi^B = \Phi^B$. Hence from (3.1) and (3.3), it follows that

$$\begin{aligned} P_\Phi &= \Phi\Phi^B = Q^{\mathrm{T}}\begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} Q, \\ P_\Phi^\perp &= I - P_\Phi = Q^{\mathrm{T}}\begin{bmatrix} 0 & 0 \\ 0 & I_{m-r} \end{bmatrix} Q. \end{aligned} \tag{3.4}$$

Due to the isometric properties of the orthogonal transformation Q, the least squares problem $\Phi\mathbf{a} \cong \mathbf{y}$ is equivalent to $Q\Phi\mathbf{a} \cong Q\mathbf{y}$. We define

$$\bar{\mathbf{y}} \equiv Q\mathbf{y} \equiv \begin{bmatrix} \bar{\mathbf{y}}_1 \\ \cdots \\ \bar{\mathbf{y}}_2 \end{bmatrix} \begin{matrix} \}r \\ \}(m-r) \end{matrix}.$$

A simple computation shows that

$$\|\mathbf{y} - \Phi\hat{\mathbf{a}}\|^2 = \|P_\Phi^\perp \mathbf{y}\|^2 = \|\bar{\mathbf{y}}_2\|^2. \tag{3.5}$$

Therefore, one can evaluate the nonlinear functional $r_2(\boldsymbol{\alpha})$ of (2.8) for any value of $\boldsymbol{\alpha}$ in the following way: First the orthogonal matrix $Q(\boldsymbol{\alpha})$ that is used in the reduction of $\Phi(\boldsymbol{\alpha})$ is determined; simultaneously, $\bar{\mathbf{y}} = Q\mathbf{y}$ is computed, and finally

$$r_2(\boldsymbol{\alpha}) = \|\bar{\mathbf{y}}_2\|^2 \tag{3.6}$$

is evaluated.

For minimization techniques not requiring derivatives this is all that is needed. For iterative techniques using the gradient of the functional or the Jacobian of the residual vector function $P_{\Phi(\boldsymbol{\alpha})}^\perp \mathbf{y}$ we shall provide in the next section formulas which will also be useful in the proof of Theorem 2.1.

4. Fréchet derivatives of projectors, residual vectors, and pseudo-inverses, and their applications. In this section we develop formulas for the Fréchet derivative of orthogonal projectors associated with differentiable matrix functions. This leads to expressions for the derivatives of the vector function

$$\mathbf{r}_2(\boldsymbol{\alpha}) = P^{\perp}_{\Phi(\boldsymbol{\alpha})}\mathbf{y}, \tag{4.1}$$

and that of the pseudo-inverse of $\Phi(\boldsymbol{\alpha})$. These expressions were developed in [25]. The arguments given here, however, are considerably simpler due to an observation of G. W. Stewart [27].

An $m \times n$ matrix function $A(\boldsymbol{\alpha})$ is a nonlinear mapping between the linear space of parameters, $\mathscr{R}^k$, and the space of linear transformations $\mathscr{L}(\mathscr{R}^n, \mathscr{R}^m)$. Consequently, $\mathbf{D}A(\boldsymbol{\alpha})$ will be, for each $\boldsymbol{\alpha}$, an element of $\mathscr{L}(\mathscr{R}^k, \mathscr{L}(\mathscr{R}^n, \mathscr{R}^m))$. Thus, $\mathbf{D}A(\boldsymbol{\alpha})$ could be interpreted as a tridimensional tensor, formed with k $(m \times n)$ matrices (slabs), each one containing the partial derivatives of the elements of A with respect to one of the variables α_i. Still in another way, each column in the k-direction is the gradient of the corresponding matrix element.

Since all dimensions involved are different, it will be always clear in the algebraic manipulations how the different vectors, matrices, and tensors interact. We expect that our efforts in not burdening the reader with a more formal but considerably heavier notation will be appreciated.

First we compute the Fréchet derivative of the orthogonal projector $P_{A(\boldsymbol{\alpha})}$ associated with a differentiable $m \times n$ matrix function $A(\boldsymbol{\alpha})$ of local constant rank r.

LEMMA 4.1. *Let $A^-(\boldsymbol{\alpha})$ be an $n \times m$ matrix function such that $AA^-A = A$ and $(AA^-)^{\mathrm{T}} = AA^-$. Then*

$$\mathbf{D}P_A = P_A^{\perp}\mathbf{D}AA^- + (P_A^{\perp}\mathbf{D}AA^-)^{\mathrm{T}}. \tag{4.2}$$

Proof. Since $P_AA = A$,

$$\mathbf{D}(P_AA) = \mathbf{D}P_AA + P_A\mathbf{D}A = \mathbf{D}A,$$

and hence,

$$\mathbf{D}P_AA = \mathbf{D}A - P_A\mathbf{D}A = P_A^{\perp}\mathbf{D}A.$$

Thus, since $P_A = AA^-$,

$$\mathbf{D}P_AP_A = \mathbf{D}P_AAA^- = P_A^{\perp}\mathbf{D}AA^-. \tag{4.3}$$

Since

$$(\mathbf{D}P_AP_A)^{\mathrm{T}} = P_A\mathbf{D}P_A \tag{4.4}$$

(the transposition being done within each symmetric slab $(\partial/\partial\alpha_i)P_A$ of the tridimensional tensor $\mathbf{D}P_A$) we finally obtain from (4.3) and (4.4)

$$\mathbf{D}P_A = \mathbf{D}(P_A^2) = \mathbf{D}P_AP_A + P_A\mathbf{D}P_A = P_A^{\perp}\mathbf{D}AA^- + (P_A^{\perp}\mathbf{D}AA^-)^{\mathrm{T}},$$

which completes the proof.

Lemma 4.1 is, of course, valid when A^- is replaced by A^+. Also

$$\mathbf{D}P_A^{\perp} = -\mathbf{D}P_A. \tag{4.5}$$

We now use the results of Lemma 4.1 to obtain the gradient of the variable projection functional $r_2(\boldsymbol{\alpha})$. Since

$$r_2(\boldsymbol{\alpha}) = \|P^{\perp}_{\Phi(\boldsymbol{\alpha})}\mathbf{y}\|^2 = \langle P^{\perp}\mathbf{y}, P^{\perp}\mathbf{y}\rangle, \tag{4.6}$$

we have immediately by (4.2) and (4.5),

$$\tfrac{1}{2}\nabla r_2(\boldsymbol{\alpha}) = -\mathbf{y}^{\mathrm{T}}P^{\perp}[P^{\perp}\mathbf{D}\Phi\Phi^{-} + (\Phi^{-})^{\mathrm{T}}\mathbf{D}\Phi^{\mathrm{T}}P^{\perp}]\mathbf{y}^{\mathrm{T}}.$$

Now we assume that Φ^{-} satisfies the additional hypothesis

$$\Phi^{-}\Phi\Phi^{-} = \Phi^{-}.$$

Then

$$\begin{aligned} P^{\perp}(\Phi^{-})^{\mathrm{T}} &= (\Phi^{-})^{\mathrm{T}} - \Phi\Phi^{-}(\Phi^{-})^{\mathrm{T}} \\ &= (\Phi^{-})^{\mathrm{T}} - (\Phi\Phi^{-})^{\mathrm{T}}(\Phi^{-})^{\mathrm{T}} \\ &= 0. \end{aligned}$$

Hence,

$$\tfrac{1}{2}\nabla r_2(\boldsymbol{\alpha}) = -\mathbf{y}^{\mathrm{T}}P^{\perp}_{\Phi(\boldsymbol{\alpha})}\mathbf{D}\Phi(\boldsymbol{\alpha})\Phi^{-}(\boldsymbol{\alpha})\mathbf{y}. \tag{4.7}$$

D. Jupp has independently developed (4.7) in the full rank case. This formula is particularly useful in association with variable metric minimization procedures (cf. [23], [24]).

Now we have the elements for proving Theorem 2.1.

Proof of Theorem 2.1. From (2.2) we have that $r(\mathbf{a}, \boldsymbol{\alpha}) = \|\mathbf{y} - \Phi(\boldsymbol{\alpha})\mathbf{a}\|^2$. Therefore,

$$\tfrac{1}{2}\nabla r(\mathbf{a}, \boldsymbol{\alpha}) = -(\mathbf{y} - \Phi\mathbf{a})^{\mathrm{T}}(\mathbf{D}\Phi\mathbf{a} \oplus \Phi), \tag{4.8}$$

where $\oplus$ stands for direct sum.

Assume that $\hat{\boldsymbol{\alpha}}$ is a critical point of $r_2(\boldsymbol{\alpha})$, and that $\mathbf{a}$ is defined by

$$\hat{\mathbf{a}} \equiv \Phi^{+}(\hat{\boldsymbol{\alpha}})\mathbf{y}. \tag{4.9}$$

Then,

$$\begin{aligned} \tfrac{1}{2}\nabla r(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}}) &= -(P^{\perp}_{\Phi}\mathbf{y})^{\mathrm{T}}(\mathbf{D}\Phi\Phi^{+}\mathbf{y} \oplus \Phi) \\ &= \tfrac{1}{2}\nabla r_2(\hat{\boldsymbol{\alpha}}) \oplus \mathbf{0} \end{aligned} \tag{4.10}$$

since $\mathbf{y}^{\mathrm{T}}P^{\perp}_{\Phi}\Phi = \mathbf{0}$. Thus $(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}})$ is a critical point of $r(\mathbf{a}, \boldsymbol{\alpha})$.

Assume now that $\hat{\boldsymbol{\alpha}}$ is a global minimizer of $r_2(\boldsymbol{\alpha})$ in Ω, and $\hat{\mathbf{a}}$ satisfies (4.9). Then clearly, $r(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}}) = r_2(\hat{\boldsymbol{\alpha}})$. Assume that there exists $(\mathbf{a}^*, \boldsymbol{\alpha}^*)$, $\boldsymbol{\alpha}^* \in \Omega$, such that $r(\mathbf{a}^*, \boldsymbol{\alpha}^*) < r(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}})$. Since for any $\boldsymbol{\alpha}$ we have $r_2(\boldsymbol{\alpha}) \leqq r(\mathbf{a}, \boldsymbol{\alpha})$, then it follows that $r_2(\boldsymbol{\alpha}^*) \leqq r(\mathbf{a}^*, \boldsymbol{\alpha}^*) < r(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}}) = r_2(\hat{\boldsymbol{\alpha}})$, which is a contradiction to the fact that $\hat{\boldsymbol{\alpha}}$ was a global minimizer of $r_2(\boldsymbol{\alpha})$ in Ω. Therefore $(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}})$ is a global minimizer of $r(\mathbf{a}, \boldsymbol{\alpha})$ in Ω, and part (a) of the Theorem is proved.

Conversely, suppose that $(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}})$ is a global minimizer of $r(\mathbf{a}, \boldsymbol{\alpha})$ in Ω. Then as above $r_2(\hat{\boldsymbol{\alpha}}) \leqq r(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}})$. Now let $\mathbf{a}^* = \Phi^{+}(\hat{\boldsymbol{\alpha}})\mathbf{y}$. Then we have $r_2(\hat{\boldsymbol{\alpha}}) = r(\mathbf{a}^*, \hat{\boldsymbol{\alpha}}) \leqq r(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}})$; but since $(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}})$ was a global minimizer we must have equality. If there was an unique $\mathbf{a}$ among the minimizers of $r(\mathbf{a}, \boldsymbol{\alpha})$, then $\mathbf{a}^* \equiv \hat{\mathbf{a}}$. We still have to show that $\hat{\boldsymbol{\alpha}}$ is a global minimizer of $r_2(\boldsymbol{\alpha})$. Assume that it is not. Thus, there will be $\bar{\boldsymbol{\alpha}} \in \Omega$, such that $r_2(\bar{\boldsymbol{\alpha}}) < r_2(\hat{\boldsymbol{\alpha}})$. Let $\bar{\mathbf{a}}$ be equal to $\Phi^+(\bar{\boldsymbol{\alpha}})\mathbf{y}$. Then $r_2(\bar{\boldsymbol{\alpha}}) = r(\bar{\mathbf{a}}, \bar{\boldsymbol{\alpha}}) < r_2(\hat{\boldsymbol{\alpha}}) = r(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}})$, which is a contradiction to the fact that $(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}})$ was a global minimizer of $r(\mathbf{a}, \boldsymbol{\alpha})$. This completes the proof.

With Lemma 4.1 we have the machinery for obtaining the derivative of $\Phi^+(\boldsymbol{\alpha})$. Although this is a digression from the main theme of this paper, we develop the formula because of its novelty and its importance in related applications, some of which we shall mention briefly.

In order to prove Theorem 4.3 we need the following corollary to Lemma 4.1.

COROLLARY 4.2. *Let ${}_AP = A^+A$. Then*

$$\mathbf{D}_A P = A^+ \mathbf{D} A_A P^\perp + (A^+ \mathbf{D} A_A P^\perp)^\mathsf{T}. \tag{4.11}$$

Proof. Since $(P_{A^\mathsf{T}})^\mathsf{T} = (A^\mathsf{T}(A^\mathsf{T})^+)^\mathsf{T} = A^+A = {}_AP$, then (4.11) follows readily from (4.2) with A replaced by A^T.

THEOREM 4.3. *Let $\Omega \subset \mathscr{R}^k$ be an open set and for $\boldsymbol{\alpha} \in \Omega$ let $A(\boldsymbol{\alpha})$ be an $m \times n$ Fréchet differentiable matrix function with local constant rank $r \leqq \min(m, n)$ in Ω. Then for any $\boldsymbol{\alpha} \in \Omega$,*

$$\mathbf{D}A^+(\boldsymbol{\alpha}) = -A^+\mathbf{D}AA^+ + A^+A^{+\mathsf{T}}\mathbf{D}A^\mathsf{T}P_A^\perp + {}_AP^\perp \mathbf{D}A^\mathsf{T}A^{+\mathsf{T}}A^+. \tag{4.12}$$

Proof. Since $(P_{A^\perp})^\mathsf{T} = (A^\mathsf{T}(A^\mathsf{T})^+)^\mathsf{T} = A^+A = {}_AP$, then (4.11) follows readily

$$\mathbf{D}A^+ = \mathbf{D}({}_APA^+) = \mathbf{D}_APA^+ + {}_AP\mathbf{D}A^+.$$

Combining this with (4.11), and observing that ${}_AP^\perp A^+ = 0$, we obtain

$${}_AP^\perp \mathbf{D}A^+ = {}_AP^\perp \mathbf{D}_APA^+ = {}_AP^\perp \mathbf{D}A^\mathsf{T}A^{+\mathsf{T}}A^+. \tag{4.13}$$

Now

$$\mathbf{D}A^+ = \mathbf{D}(A^+AA^+) = \mathbf{D}A^+P_A + A^+\mathbf{D}AA^+ + {}_AP\mathbf{D}A^+,$$

and thus,

$$\mathbf{D}A^+P_A = -A^+\dot{\mathbf{D}}AA^+ + {}_AP^\perp \mathbf{D}A^+.$$

Combining this last expression with (4.13), we have

$$\mathbf{D}A^+P_A = -A^+\mathbf{D}AA^+ + {}_AP^\perp \mathbf{D}A^\mathsf{T}A^{+\mathsf{T}}A^+. \tag{4.14}$$

But,

$$\mathbf{D}A^+ = \mathbf{D}(A^+P_A) = \mathbf{D}A^+P_A + A^+\mathbf{D}P_A$$

and therefore,

$$\mathbf{D}A^+P_A^\perp = A^+\mathbf{D}P_A = A^+A^{+\mathsf{T}}\mathbf{D}A^\mathsf{T}P_A^\perp, \tag{4.15}$$

since

$$A^+P_A^\perp = 0.$$

The theorem follows from the relationship

$$\mathbf{D}A^+ = \mathbf{D}A^+P_A + \mathbf{D}A^+P_A^\perp,$$

and from (4.14), (4.15).

Formula (4.12) is new. Let $B = A + dA$ where dA is an arbitrary incremental matrix. Wedin [29], [30] has shown that

$$B^+ - A^+ = -B^+\, dAA^+ + {}_BP^\perp\, dA^{\mathrm{T}}A^{+\mathrm{T}}A^+ + B^+B^{+\mathrm{T}}\, dA^{\mathrm{T}}P_A^\perp. \tag{4.16}$$

Formula (4.16) can be used for deriving (4.12) by letting $dA \to 0$ as was done in the full rank case in [10]. This technique was pointed out to the authors by W. Kahan [28].

There are many potential applications for the formulas developed in this section, besides the one we explicitly give. We shall mention a few of them.

(a) *Optimization with nonlinear equality constraints.* Consider the problem

$$\min_{\mathbf{x}} f(\mathbf{x}), \qquad \mathbf{x} \in \mathscr{R}^n,$$

subject to

$$\mathbf{c}(\mathbf{x}) = \mathbf{0}, \qquad \mathbf{c}: \mathscr{R}^n \to \mathscr{R}^k, \qquad k \leqq n,$$

where $f(\mathbf{x})$ is a functional.

In Fletcher and Lill [7] methods are proposed which require the derivative of $(\mathbf{Dc})^+$. Our formulas would permit $(\mathbf{Dc})^+$ to be rank deficient, though the theory of this problem is not well understood at the present time. See also [17].

(b) *Generalized Newton's method for* $f(\mathbf{x}) = \mathbf{0}$. In [31] Ben-Israel considers the following iterative procedure:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - (\mathbf{Df}(\mathbf{x}^k))^+\mathbf{f}(\mathbf{x}^k). \tag{4.17}$$

Formula (4.12) could allow a direct study of the convergence properties of (4.17).

(c) *Stability of the solution of perturbed linear least squares problems.* Let $A(\varepsilon) = A + \varepsilon B$. We assume

(i) rank $(A(\varepsilon))$ = rank (A) for all $\varepsilon > 0$ sufficiently small,

(ii) $\|A\| = \|B\| = 1$.

We wish to consider the behavior of $\hat{\mathbf{x}}(\varepsilon)$ satisfying

$$\hat{\mathbf{x}}(\varepsilon) = A^+(\varepsilon)\mathbf{b}$$

as $\varepsilon \to 0$. Using Taylor's formula, we obtain

$$A^+(\varepsilon) - A^+(0) = \varepsilon(\mathbf{D}A^+)(0)B + O(\varepsilon^2). \tag{4.18}$$

Now by (4.12) we have

$$(\mathbf{D}A^+)(0) = -A^+BA^+ + A^+A^{+\mathrm{T}}B^{\mathrm{T}}P_A^\perp + {}_AP^\perp B^{\mathrm{T}}A^{+\mathrm{T}}A^+ \tag{4.19}$$

by using the fact that $(\mathbf{D}A)(0) = B$. Hence, using (4.18) and (4.19), we obtain by the usual norm argument,

$$\|\mathbf{x}(\varepsilon) - \mathbf{x}\| = \|A^+(\varepsilon)\mathbf{b} - A^+\mathbf{b}\| \leqq 2\varepsilon\|A^+\|\,\|\hat{\mathbf{x}}\| + \varepsilon\|A^+\|^2\|\hat{\mathbf{r}}\| + O(\varepsilon^2), \tag{4.20}$$

where $\hat{\mathbf{r}} = \mathbf{b} - A\hat{\mathbf{x}}$. Equation (4.20) is the rank deficient generalization of results given in [32]. A similar treatment can be used to obtain more detailed estimates of the type given in [16], [29], [30], [33].

5. Algorithmia II. Detailed implementation of the Gauss–Newton–Marquardt algorithm. Computation of the Jacobian of the variable projection vector. We shall now explain in detail how to apply the results of § 4 to the Marquardt modification of the Gauss–Newton iterative procedure; we make extensive use of linear least squares techniques. We describe an economical implementation of the Marquardt algorithm devised earlier by Golub (see also [11], [14]).

We define the vector

$$\mathbf{r}_2(\boldsymbol{\alpha}) = P^{\perp}_{\Phi(\boldsymbol{\alpha})}\mathbf{y}.$$

The generalized Gauss–Newton (G.N.) iteration with step control for the nonlinear least squares problem

$$\min_{\boldsymbol{\alpha}} r_2(\boldsymbol{\alpha}) = \min_{\boldsymbol{\alpha}} \|\mathbf{r}_2(\boldsymbol{\alpha})\|^2 = \min_{\boldsymbol{\alpha}} \|P^{\perp}_{\Phi(\boldsymbol{\alpha})}\mathbf{y}\|^2 \tag{5.1}$$

is given as follows:

$$\boldsymbol{\alpha}^{l+1} = \boldsymbol{\alpha}^l - t_l[\mathbf{D}\mathbf{r}_2(\boldsymbol{\alpha}^l)]^+\mathbf{r}_2(\boldsymbol{\alpha}^l), \qquad l = 0, \cdots. \tag{5.2}$$

The parameters $t_l > 0$, which control the size of the step, are used to prevent divergence. Usually $t_l = 1$, unless $r_2(\boldsymbol{\alpha}^{l+1}) > r_2(\boldsymbol{\alpha}^l)$, in which case t_l is reduced. Another use of the parameters t_l is to minimize $r_2(\boldsymbol{\alpha}^{l+1})$ along the direction $-[\mathbf{D}\mathbf{r}_2(\boldsymbol{\alpha}^l)]^+\mathbf{r}_2(\boldsymbol{\alpha}^l)$.

Marquardt's modification calls for the introduction of a sequence of nonnegative auxiliary parameters $v_l \geqq 0$.

(G.N.M.) Define

$$K_l \equiv \left[\frac{\mathbf{D}\mathbf{r}_2(\boldsymbol{\alpha}^l)}{v_l F_l}\right], \qquad \mathbf{r}_l \equiv \left[\frac{\mathbf{r}_2(\boldsymbol{\alpha}^l)}{0}\right]\}k,$$

where for each l, F_l is the upper triangular Cholesky factor of a $k \times k$ symmetric positive definite matrix M_l. Then the Gauss–Newton–Marquardt iteration is given by

$$\boldsymbol{\alpha}^{l+1} = \boldsymbol{\alpha}^l - K_l^+\mathbf{r}_l, \qquad l \geqq 0. \tag{5.3}$$

Reasons for this modification are well known. For more details and an interesting study of the convergence of this method we refer to [14]. We wish to make explicit now the "two-stage orthogonal factorization" given in [11] and [14], in order to show how to take advantage of the special structure of the problem.

Calling

$$\mathbf{h} = \boldsymbol{\alpha}^{l+1} - \boldsymbol{\alpha}^l, \qquad DP = \mathbf{D}\mathbf{r}_2(\boldsymbol{\alpha}^l) = \mathbf{D}P^{\perp}_{\Phi}\mathbf{y}$$

and dropping the superscript l from here on in, one step of the Marquardt algorithm is equivalent to solving the linear least squares problem

$$K\mathbf{h} \cong \left[\frac{-\mathbf{r}_2(\boldsymbol{\alpha})}{\mathbf{0}}\right].$$

In the first stage of the orthogonal factorization of K an $m \times m$ orthogonal matrix Q is chosen so that

$$QDP = R_1 = \left[\begin{array}{c} \diagdown\!\!\!\!\; 0 \\ \hline 0 \end{array}\right]\!\}n \equiv \left[\begin{array}{c} R'_1 \\ \hline 0 \end{array}\right].$$

Thus,

$$\left[\begin{array}{c|c} Q & 0 \\ \hline 0 & I_k \end{array}\right] \cdot \left[\begin{array}{c} DP \\ \hline vF \end{array}\right] \equiv Q_1 \left[\begin{array}{c} DP \\ \hline vF \end{array}\right] = \left[\begin{array}{c} R_1 \\ \hline vF \end{array}\right],$$

$$Q_1 \left[\begin{array}{c} -\mathbf{r}_2(\boldsymbol{\alpha}) \\ \hline \mathbf{0} \end{array}\right] \equiv \left[\begin{array}{c} \bar{\mathbf{r}} \\ \hline \mathbf{0} \end{array}\right].$$

R'_1 and $\bar{\mathbf{r}}$ are saved for future use.

In the second stage we choose an $(m + k) \times (m + k)$ orthogonal matrix Q_2 to reduce $A \equiv \left[\begin{array}{c} R'_1 \\ \hline vF \end{array}\right]$ to "triangular" form. For this purpose we shall use successive Householder transformations as in [3], from where we adopt the notation.

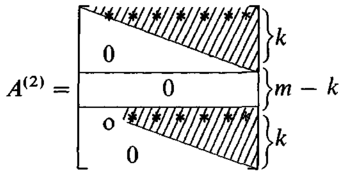
On reducing the first column of A, which is of the form

$$\mathbf{a}_1^{(1)} = \left[\begin{array}{c} a_{11}^{(1)} \\ 0 \\ \vdots \\ 0 \\ \hline \mu \\ 0 \\ \vdots \\ 0 \end{array}\right] \begin{array}{l} \left.\vphantom{\begin{array}{c} a \\ 0 \\ \vdots \\ 0 \end{array}}\right\} m, \qquad \mu = vf_{11}, \\ \vphantom{\begin{array}{c} \mu \\ 0 \\ \vdots \\ 0 \end{array}} \end{array}$$

we use $Q^{(1)} = I - \beta_1 \mathbf{u}^{(1)} \mathbf{u}^{(1)\mathrm{T}}$, where

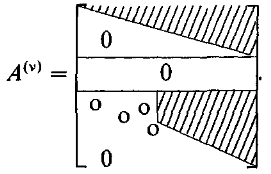
$$\begin{aligned} u_1^{(1)} &= \operatorname{sign}(a_{11}^{(1)})(\sigma_1 + |a_{11}^{(1)}|), \\ \sigma_1 &= (a_{11}^2 + \mu^2)^{1/2}, \\ u_{m+1}^{(1)} &= \mu, \\ u_i^{(1)} &= 0, \quad \text{otherwise}, \\ \beta_1 &= (\sigma_1 |u_1^{(1)}|)^{-1}. \end{aligned}$$

Now we observe that when $Q^{(1)}$ is applied to a vector, any component corresponding to a zero component of $\mathbf{u}^{(1)}$ is left unchanged. In particular, the band of zeros in A is preserved. Thus, in this first step we only need to transform the elements of rows number 1 and $m + 1$. Consequently, $A^{(2)} = Q^{(1)}A$ will have the schematic form



where the asterisks indicate the modified elements.

It is now clear that at step ν, $A^{(\nu)}$ will have the form



The matrix $A^{(\nu+1)}$, $\nu = 1, \cdots, k$, is obtained as follows:

(i) $$\sigma_\nu = ((a_{\nu\nu}^{(\nu)})^2 + \sum_{i=1}^{\nu} (a_{m+i,\nu}^{(\nu)})^2)^{1/2},$$

(ii) $$\beta_\nu = (\sigma_\nu(\sigma_\nu + |a_{\nu\nu}^{(\nu)}|))^{-1},$$

(iii) $u_i^{(\nu)} = 0$ for $i < \nu$, $\nu + 1 \leqq i \leqq m$, $m + \nu < i$;

$u_\nu^{(\nu)} = \operatorname{sign}(a_{\nu\nu}^{(\nu)})(\sigma_\nu + |a_{\nu\nu}^{(\nu)}|)$;

$u_i^{(\nu)} = a_i^{(\nu)}$, $m + 1 \leqq i \leqq m + \nu$.

(iv) $$\mathbf{y}^{\mathrm{T}} = \beta_\nu \mathbf{u}^{(\nu)\mathrm{T}} A^{(\nu)},$$

$$y_j = \beta_{(\nu)}\left[u_\nu^{(\nu)} a_{\nu j}^{(\nu)} + \sum_{i=1}^{\nu} a_{m+i,\nu}^{(\nu)} a_{m+i,j}^{(\nu)}\right], \qquad j = \nu + 1, \cdots, k.$$

Finally,

(v) $a_{ij}^{(\nu+1)} = a_{ij}^{(\nu)} - u_i^{(\nu)} y_j$, $i = \nu; m + 1, \cdots, m + \nu$; $j = \nu + 1, \cdots, k$;

$a_{\nu\nu}^{(\nu+1)} = -\operatorname{sign}(a_{\nu\nu}^{(\nu)})\sigma_\nu$.

These formulas are similar to those given in [3], but are modified to take advantage of the structure of the matrix A. A Fortran implementation can be found in [25].

We shall evaluate $\mathbf{D}\mathbf{r}_2(\boldsymbol{\alpha}) = \mathbf{D}P^{\perp}_{\Phi(\boldsymbol{\alpha})}\mathbf{y}$ for a given $\boldsymbol{\alpha}$, according to

(5.4) $$\mathbf{D}P^{\perp}_{\Phi(\boldsymbol{\alpha})}\mathbf{y} = -(P^{\perp}_{\Phi}\mathbf{D}\Phi)\Phi^{B}\mathbf{y} - (\Phi^{B})^{\mathrm{T}}(P^{\perp}_{\Phi}\mathbf{D}\Phi)^{\mathrm{T}}\mathbf{y},$$

which is readily obtained from (4.2) and (4.5). The matrix Φ^B is constructed as in (3.3).

In many applications, each component function φ_j depends only upon a few of the parameters $\{\alpha_t\}_{t=1}^{k}$, and therefore its derivatives with respect to the other parameters will vanish. Those vanishing derivatives will produce m-columns of zeros in the tensor $\mathbf{D}\Phi$. In order to avoid a waste of storage and useless computation with zeros it is convenient to introduce from the outset the $k \times n$ incidence matrix $E = (e_{jt})$. This matrix will be defined as follows:

$$e_{jt} = \begin{cases} 1 \text{ if and only if parameter } \alpha_t \text{ appears in function } \varphi_j; \\ 0, \text{ otherwise.} \end{cases}$$

We shall also call p the number of nonzero derivatives in $\mathbf{D}\Phi : p = \sum_{t,j} e_{jt}$. The nonzero derivative vectors can then be stored sequentially in a bidimensional array $B(m \times p)$. In our implementation we chose to store the nonzero m-columns varying first the index corresponding to the different differentiations, and then that corresponding to the different functions. This information can then be decoded for use in algebraic manipulations by means of the incidence matrix E.

We now introduce some notation in order to describe the compressed storage of the nonzero columns of the tensor $\mathbf{D}\Phi$ in a more explicit fashion. We define, for $t = 1, \cdots, k$,

$$S_t = \{\text{set of ordered indices for which } e_{jt} \neq 0, j = 1, \cdots, n\};$$

$$\psi_{tj}(\boldsymbol{\alpha}) = \partial\varphi_j(\boldsymbol{\alpha})/\partial\alpha_t, \qquad j = 1, \cdots, n, \quad t = 1, \cdots, k.$$

We write the matrix B in partitioned form

$$B = [B_1, B_2, \cdots, B_k],$$

where

$$B_t = [\boldsymbol{\psi}_{tj_1}, \boldsymbol{\psi}_{tj_2}, \cdots, \boldsymbol{\psi}_{tj_{l_t}}]_{j_i \in S_t}.$$

A step-by-step description of the computation of $\mathbf{D}P_\Phi^\perp \mathbf{y}$ follows. We assume that the rank of $\Phi(\boldsymbol{\alpha})$ is computationally determined and equal to $r \leqq \min(m, n)$.

(a) Compute $\Phi(\boldsymbol{\alpha})$, $\mathbf{D}\Phi(\boldsymbol{\alpha})$.

(b) Form the $m \times (n + p + 1)$ array

$$G \equiv [\Phi(\boldsymbol{\alpha}); \mathbf{y}; \mathbf{D}\Phi(\boldsymbol{\alpha})] = [A; \mathbf{y}; B].$$

(c) Obtain the orthogonal factorization of A (cf. § 3):

$$QAS = T_0 = \left[\begin{array}{c|c} T_{11} & T_{12} \\ \hline 0 & 0 \end{array}\right]; \qquad T_{11} = \begin{bmatrix} \diagdown & \\ 0 & \end{bmatrix}_{r \times r}$$

Also $\mathbf{v} = Q\mathbf{y}$; $C = QB$ (T_{11}, T_{12}, $\mathbf{v}$, and C will be stored in the array G). Note again that (see § 3)

$$P_{\Phi(\boldsymbol{\alpha})}^\perp = Q^{\mathrm{T}} \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & I_{m-r} \end{array}\right] Q$$

(d) Get the intermediary values:

$$\bar{D} = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & I_{m-r} \end{array}\right] \cdot C$$

(i.e., remember that the nonzero information of $\bar{D}$ is stored in the last p columns and last $m - r$ rows of G);

$$\mathbf{x} = A^B\mathbf{y} = S\left[\frac{T_{11}^{-1}\mathbf{v}_1}{\mathbf{0}}\right] \qquad \left(\mathbf{v} = \begin{bmatrix}\mathbf{v}_1\\ \hline \mathbf{v}_2\end{bmatrix}\begin{matrix}\}r\\ \}m-r\end{matrix}\right).$$

(e) $$\underset{n\times k}{U} = (P^\perp\mathbf{D\Phi})^{\mathrm{T}}\mathbf{y} = \mathbf{D\Phi}^{\mathrm{T}}Q^{\mathrm{T}}\left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & I_{m-r}\end{array}\right]Q\mathbf{y} = \bar{D}^{\mathrm{T}}\mathbf{v} = C^{\mathrm{T}}\left[\frac{\mathbf{0}}{\mathbf{v}_2}\right]$$

(transposition in the tensor $\mathbf{D\Phi}$ refers to transposition within the "slabs" corresponding to the different derivatives, and must be interpreted adequately when decoding the information from the compressed storage array G; the appropriate ALGOL-60 code for computing U with our storage convention would be (assuming that $C = QB$ is stored in the same place B is):

```
$n1 \leftarrow n + 1$;
$L \leftarrow n1$;
for $t \leftarrow 1$ step 1 until $k$ do
for $j \leftarrow 1$ step 1 until $n$ do
if $E[j, t] = 0$ then $U[j, t] \leftarrow 0$ else
begin $L \leftarrow L + 1$; acum $\leftarrow 0$;
  for $i \leftarrow n1$ step 1 until $m$ do
    acum $\leftarrow$ acum $+ G[i, L] \times G[i, n1]$;
  $U[j, t] \leftarrow$ acum
end;).
```

(f) Compute $H_{n\times k} = S^{\mathrm{T}}U$. Solve the k, $r \times r$ lower triangular systems:

$$T_{11}^{\mathrm{T}}W = \tilde{H},$$

where $\tilde{H}_{r\times k}$ contains the first r rows of H. Store W in the first r rows of the $m \times k$ array B. Compute $\bar{D}\mathbf{x}$ and store the nonzero information in the last $m - r$ rows of B.

(g) Finally, the $m \times k$ matrix B is obtained as

$$B \leftarrow \mathbf{D}P^\perp_{\Phi(\alpha)}\mathbf{y} = -\underset{m\times m}{Q^{\mathrm{T}}}\left\{\left[\frac{W}{0}\right] + \bar{D}\mathbf{x}\right\} = -Q^{\mathrm{T}}B.$$

We emphasize the systematic use made of the triangular orthogonal decomposition of the matrix $\Phi(\boldsymbol{\alpha})$. We also warn the reader about the correct interpretation of the algebraic operations in which any tridimensional tensor intervenes, as we exemplified in (e).

Computation of the gradient of the variable projection functional for variable metric minimization procedures. We recall (4.7):

$$\nabla r_2(\boldsymbol{\alpha}) = -2\mathbf{y}^{\mathrm{T}}P^\perp_{\Phi(\boldsymbol{\alpha})}\mathbf{D\Phi}(\boldsymbol{\alpha})\Phi^-(\alpha)\mathbf{y}.$$

In order to implement efficiently this formula, we proceed as in the case of $\mathbf{D}P_{\Phi}^{\perp}\mathbf{y}$, which we have just described:

(a) and (b) as before;

(c) as before, except that $C = QB$ is *not* necessary;

(d) $\mathbf{x} = T_{11}^{-1}\mathbf{v}_1$;

(e) $\mathbf{z} = P_{\Phi}^{\perp}\mathbf{y} = Q^{\mathrm{T}}\begin{bmatrix}\mathbf{0}\\ \cdots \\ \mathbf{v}_2\end{bmatrix}$;

(f) $\Psi = -2\mathbf{z}^{\mathrm{T}}\mathbf{D}\Phi$;

(g) $\nabla r_2(\boldsymbol{\alpha}) = \Psi\mathbf{x}$.

In order to use a variable metric minimization procedure such as the Davidon–Fletcher–Powell method [5], it is simply necessary to compute the residual as in § 3 and its gradient as given above.

6. Numerical experiments. We have implemented four different algorithms based on the developments of the previous sections. For each example rank $(\Phi(\boldsymbol{\alpha})) = n$. The methods minimize the variable projection functional $r_2(\boldsymbol{\alpha}) = \|P_{\Phi(\boldsymbol{\alpha})}^{\perp}\mathbf{y}\|^2$ first, in order to obtain the optimal parameters $\hat{\boldsymbol{\alpha}}$, and then complete the optimization according to our explanation in § 2. The algorithms differ in the procedure used for the minimization of $r_2(\boldsymbol{\alpha})$. We also minimize the original functional $r(\mathbf{a}, \boldsymbol{\alpha})$ and compare the results.

A1. *Minimization without derivatives.* We use PRAXIS, a FORTRAN version of a program developed by R. Brent [4], who very kindly made it available to us. All that PRAXIS essentially requires from the user is the value of the functional for any $\boldsymbol{\alpha}$. This is computed using the results of § 3. In fact, the user has only to give code for filling the matrix Φ for any $\boldsymbol{\alpha}$, and our program will effect the triangular reduction and so on. It turns out that many times (see the examples) the models have some terms which are exclusively linear, i.e., functions φ_j which are independent of $\boldsymbol{\alpha}$. Those functions produce columns in $\Phi(\boldsymbol{\alpha})$ which are constant throughout the process. If they are considered first, then it is possible to reduce them once and for all, saving the repetition of computation. This is done in our program.

A2. *Minimization by Gauss–Newton with control of step* (see (5.2)). The user is required to provide the incidence matrix E and the array of functions φ_j and nonvanishing partial derivatives G. See § 5 for a more detailed description.

A3. *Minimization by Marquardt's modification.* It is as explained in § 5 with $F_l \equiv I$. User supplied information is the same as in A2.

A4. *Variable metric procedure.* We have used a FORTRAN program of M. Osborne. The user supplied information is the same as for $A2$ and $A3$, but here only the gradient of $r_2(\boldsymbol{\alpha})$ is required and this is computed according to § 5.

Test problems. Problems 1 and 2 are taken from Osborne [14], where the necessary data can be found.

P1. *Exponential fitting.* The model is of the form

$$\eta_1(\mathbf{a}, \boldsymbol{\alpha}; t) = a_1 + a_2\, e^{-\alpha_1 t} + a_3\, e^{-\alpha_2 t}.$$

The functions φ_i are obviously $\varphi_1(\boldsymbol{\alpha}; t) \equiv 1$, $\varphi_{j+1}(\boldsymbol{\alpha}; t) = e^{-\alpha_j t}$, $j = 1, 2$. So the different constants, in the notation of § 2, are $n = 3$, $k = 2$. For the problem considered, $m = 33$. The *number of constant functions* (NCF) equals 1. The number of nonvanishing partial derivatives p equals 2.

In Table 1 we compare our results for methods A1, A2, A3, A4, and those obtained by minimizing the full functional $r(\mathbf{a}, \boldsymbol{\alpha})$.

TABLE 1
Exponential fit

Method	Functional	Number of Function Evaluations	Number of Derivative Evaluations	Time (seconds)
A1	FF	1832	—	191.00
	VP	100	—	9.00
A2	FF	11	11	5.05
	VP	4	4	3.20
A3	FF	32	26	12.55
	VP	4	4	3.12
A4	VP	27	18*	8.94

$r(\hat{\mathbf{a}},\hat{\boldsymbol{\alpha}}), r_2(\hat{\boldsymbol{\alpha}}) \leqq 0.5465 \times 10^{-4}$

* This figure corresponds to gradient evaluations.

P2. *Fitting Gaussians with an exponential background.*

$$\eta_2(\mathbf{a},\boldsymbol{\alpha};t) = a_1 e^{-\alpha_1 t} + a_2 e^{-\alpha_2(t-\alpha_5)^2} + a_3 e^{-\alpha_3(t-\alpha_6)^2} + a_4 e^{-\alpha_4(t-\alpha_7)^2}.$$

The functions φ_j are

$$\varphi_1(\boldsymbol{\alpha};t) = e^{-\alpha_1 t}; \quad \varphi_j(\boldsymbol{\alpha};t) = e^{-\alpha_j(t-\alpha_{j+3})^2}, \quad j = 2,3,4.$$

Thus, $n = 4$, $k = 7$, $m = 65$, $p = 7$. Results for this problem appear in Table 2.

TABLE 2
Gaussian fit

Method	Functional	Number of Function Evaluations	Number of Derivative Evaluations	Time (seconds)
A3	FF	11	9	23.35
	VP	10	8	26.82
A4	VP	72	65	84.34

$r(\hat{\mathbf{a}},\hat{\boldsymbol{\alpha}}), r_2(\hat{\boldsymbol{\alpha}}) \leqq 0.048$

Methods $A1$ and $A2$ were either slowly convergent or nonconvergent.

P3. *Iron Mössbauer spectrum with two sites of different electric field gradient and one single line* [21]. The model here is the following:

$$\begin{aligned}\eta_3(\mathbf{a}, \boldsymbol{\alpha}; t) = {} & a_1 + a_2 t + a_3 t^2 \\ & - a_4\left[\frac{1}{1 + ((\alpha_1 + 0.5\alpha_2 - t)/\alpha_3)^2} + \frac{1}{1 + ((\alpha_1 - 0.5\alpha_2 - t)/\alpha_3)^2}\right] \\ & - a_5\left[\frac{1}{1 + ((\alpha_4 + 0.5\alpha_5 - t)/\alpha_6)^2} + \frac{1}{1 + ((\alpha_4 - 0.5\alpha_5 - t)/\alpha_6)^2}\right] \\ & - a_6\left[\frac{1}{1 + ((\alpha_7 - t)/\alpha_8)^2}\right].\end{aligned}$$

Clearly, $\varphi_j(\boldsymbol{\alpha}; t) = t^{j-1}$, $j = 1, 2, 3$; and φ_4, φ_5, φ_6 are the functions inside the square brackets.

Here $n = 6$, $k = 8$, NCF $= 3$, $p = 8$, $m = 188$.

For this example we wish to thank Dr. J. C. Travis of NBS who kindly supplied the problem and results from his own computer program.

Comparisons are offered in Table 3.

TABLE 3
Mössbauer Iron Spectrum

Method	Functional	Initial Values	Number of Function Evaluations	Number of Derivative Evaluations	Time (seconds)
A1	FF	$\boldsymbol{\beta}^0$			*
	VP	$\boldsymbol{\beta}^0$	65	0	70.00
A2	FF	$\boldsymbol{\beta}^0$	4	4	34.34
	VP	$\boldsymbol{\beta}^0$	4	4	41.64
	FF	$\tilde{\boldsymbol{\beta}}^0$	7	7	52.27
	VP	$\tilde{\boldsymbol{\beta}}^0$	6	6	59.60
A3	FF	$\boldsymbol{\beta}^0$	16	16	118.22
	VP	$\boldsymbol{\beta}^0$	3	3	35.35
	FF	$\tilde{\boldsymbol{\beta}}^0$	18	18	130.50
	VP	$\tilde{\boldsymbol{\beta}}^0$	6	6	61.92

$r(\hat{\mathbf{a}}, \hat{\boldsymbol{\alpha}}), r_2(\hat{\boldsymbol{\alpha}}) \leq 3.0444 \times 10^8$
$(\tilde{\boldsymbol{\beta}}^0 = (80, 49, 5, 81, 24, 9.5, 100, 4)^T)$

* Did not converge in finite amount of time.

The qualitative behavior of the four different minimization procedures used in our computation follows the pattern that has been expounded in recent comparisons (Bard[1]). Gauss–Newton is fastest whenever it converges from a good initial estimate. As is shown in the fitting of Gaussians (Table 2), if the problem is troublesome, then a more elaborate strategy is called for. Brent's program has the advantage of not needing derivatives, which in this case leads to a big simplification. On the other hand, it is a very conservative program which really tries to obtain rigorous results. This, of course, can lead to a long search in cases where it is not entirely justified. The variable metric procedure did not seem to be competitive despite the simplification in the calculation of derivatives.

As a consequence of our Theorem 2.1, and of our numerical experience, we strongly recommend, even in the case when our procedure is not used, to obtain initial values for the linear parameters $a_j, \mathbf{a}^0 = \Phi^+(\boldsymbol{\alpha}^0)\mathbf{y}$. This is done in our program for the full functional and also, in the program of Travis with excellent results.

The computer times shown in Table 1 and Table 2 correspond to the CPU times (execution of the object code) on an IBM 360/50. All calculations were performed in long precision; viz. 14 hexadecimal digits in the mantissa of each number. We compare the results of minimizing the reduced functional when the variable projection (VP) technique is used with that of minimizing the full functional (FF) for various minimization algorithms. In order to eliminate the coding aspect, we have used essentially the same code for minimizing the two functionals. The only difference was in the subroutine DPA which computes in both cases the Jacobian of the residual vector.

In the FF approach, the subroutine DPA computed the $m \times (n + k)$ matrix B as follows: the first n columns consisted of the vectors $\boldsymbol{\varphi}_j(\boldsymbol{\alpha})$ while the remaining columns were the partial derivatives

$$\frac{\partial}{\partial \alpha_l}(\mathbf{y} - \Phi(\boldsymbol{\alpha})\mathbf{a}) = -\sum_{j=1}^{n} a_j \frac{\partial \boldsymbol{\varphi}_j(\boldsymbol{\alpha})}{\partial \alpha_l}, \qquad l = 1, 2, \cdots, k.$$

These derivatives were constructed using the same information provided by the user subroutine ADA. We also obtained from DPA in the FF case, the automatic initialization of the linear parameters, viz. $\mathbf{a}^0 = \Phi^+(\boldsymbol{\alpha}^0)\mathbf{y}$.

For the numerical examples given here, the cost per iteration was somewhat higher for the VP functional. However, we see that in some cases there has been a dramatic decrease in the number of iterations (this has been observed previously (cf. [12], [20])), thus in these cases the total computing time is much more favorable for the VP approach. This was especially true for all three methods of minimization when the exponential fit was made and when Marquardt's method was used in the Mössbauer spectrum problem.

For the Mössbauer spectrum problem, we used two sets of initial values. We used those given by Travis [21], (say) $\boldsymbol{\beta}^0$, and also $\tilde{\boldsymbol{\beta}}^0 \approx \boldsymbol{\beta}^0 \pm 0.05\,\boldsymbol{\beta}^0$. For $\boldsymbol{\beta}^0$, the value of the functional is 3.04467×10^8 while for $\tilde{\boldsymbol{\beta}}^0$, the value of the functional is 6.405×10^8; the final estimates of the parameters yielded a residual sum of squares less than 3.0444×10^8. When Brent's method was used on the full functional, the method did not seem to converge, but for the reduced functional, Brent's method converged reasonably well. In fact, after twenty minutes Brent's algorithm applied to the full functional with $\boldsymbol{\beta}^0$ did *not* achieve the desired reduction in the functional.

The results we have obtained in minimizing the full functional for the first two problems using the Marquardt method, and those of P3 with Newton's method and $\boldsymbol{\beta}^0$, are consistent with the results reported by Osborne and Travis.

From a rough count of the number of arithmetic operations (function and derivative evaluation per step are the same for both procedures, so that the work they do can be disregarded), it seems that for almost no combination of the parameters (m, n, k, p) will the VP procedure require fewer operations per iteration than the FF procedure. It is an open problem then to determine *a priori* under what conditions the VP procedure will converge more quickly than the FF procedure when the same minimization algorithm using derivatives is used.

Another important problem is that of stability. The numerical stability of the process *and* of the attained solution must be studied. By insisting on the use of stable linear techniques, we have tried to achieve an overall numerically stable procedure for this nonlinear situation. Since the standards of stability for nonlinear problems are ill-defined at this time, it is hard to say whether we have succeeded in obtaining our goal.

Acknowledgments. The authors wish to thank Professor Olof Widlund of the Courant Institute for his careful reading of this manuscript, and Miss Godela Scherer of the Institute Venezolano de Investigaciones Cientificas for programming assistance. We are also pleased to acknowledge the kind hospitality and stimulating conversations with Dr. Hugo D. Scolnik of the Bariloche Foundation, Argentina, where this work was initiated in July 1971. Several helpful suggestions were made by Ms. Linda Kaufman and Mr. Michael Saunders. The authors give special "gracias" to Professor G. W. Stewart, III, who whipped out a proof of a special case of (4.2) during a lecture of one of the authors at the University of Texas.

REFERENCES

[1] YONATHAN BARD, *Comparison of gradient methods for the solution of nonlinear parameter estimation problems*, this Journal, 7 (1970), pp. 157–186.

[2] I. BARRODALE, F. D. K. ROBERTS AND C. R. HUNT, *Computing best l_p approximations by functions nonlinear in one parameter*, Comput. J., 13 (1970), pp. 382–386.

[3] Å. BJÖRCK AND G. H. GOLUB, *Iterative refinement of linear least squares solutions by Householder transformations*, BIT, 7 (1967), pp. 322–337.

[4] RICHARD P. BRENT, *Algorithms for Finding Zeros and Extrema of Functions Without Calculating Derivatives*, Prentice-Hall, Englewood Cliffs, N.J., 1973.

[5] J. W. DANIEL, *The Approximate Minimization of Functionals*, Prentice-Hall, Englewood Cliffs, N.J., 1971.

[6] J. DIEUDONNÉ, *Foundations of Modern Analysis*, Academic Press, New York, 1960.

[7] R. FLETCHER AND SHIRLEY A. LILL, *A class of methods for non-linear programming. II: Computational experience*, Nonlinear Programming, J. B. Rosen, O. L. Mangasarian, and K. Ritter, eds., Academic Press, New York, 1970, pp. 67–92.

[8] GENE H. GOLUB, *Matrix decompositions and statistical calculations*, Statistical Computation, Roy C. Milton and John A. Nelder, eds., Academic Press, New York, 1969, pp. 365–397.

[9] I. GUTTMAN, V. PEREYRA AND H. D. SCOLNIK, *Least squares estimation for a class of nonlinear models*, Centre de Recherche Mathématique, Univ. de Montréal, 1971, to appear in Technometrics.

[10] RICHARD J. HANSON AND CHARLES L. LAWSON, *Extensions and applications of the Householder algorithm for solving linear least squares problems*, Math. Comp., 23 (1969), pp. 787–812.

[11] L. S. JENNINGS AND M. R. OSBORNE, *Applications of orthogonal matrix transformations to the solution of systems of linear and non-linear equations*, Tech. Rep. 37, Computer Center Australian National Univ., 1970.

[12] WILLIAM H. LAWTON AND E. A. SYLVESTRE, *Elimination of linear parameters in nonlinear regression*, Technometrics, 13 (1971), pp. 461–467.

[13] M. R. OSBORNE, *A class of nonlinear regression problems*, Data Representation, R. S. Anderssen and M. R. Osborne, eds., 1970, pp. 94–101.

[14] ———, *Some aspects of nonlinear least squares calculations*, unpublished manuscript, 1971.

[15] V. PEREYRA, *Iterative methods for solving nonlinear least squares problems*, this Journal, 4 (1967), pp. 27–36.

[16] ———, *Stability of general systems of linear equations*, Aequationes Math., 2(1969), pp. 194–206.

[17] A. PÉREZ AND H. D. SCOLNIK, *Derivatives of pseudoinverses and constrained non-linear regression problems*, Numer. Math., to appear.

[18] G. PETERS AND J. H. WILKINSON, *The least squares problem and pseudo-inverses*, Comput. J., 13 (1970), pp. 309–316.

[19] RADHAKRISHNA C. RAO AND SUJIT KUMAR MITRA, *Generalized Inverse of Matrices and its Applications*, John Wiley, New York, 1971.

[20] H. D. SCOLNIK, *On the solution of nonlinear least squares problems*, Proc. IFIP-71, Numerical Mathematics, North-Holland, Amsterdam, 1971, pp. 18–23; also Doctoral thesis, Univ. of Zurich, 1970.

[21] J. C. TRAVIS, Tech. Note 501, Radiochemical Analysis Section, National Bureau of Standards, Washington, D.C., 1970, pp. 19–33.

[22] G. GOLUB AND G. STYAN, *Numerical computations for univariate linear models*, J. Statist. Comp. and Simulation, to appear.

[23] D. JUPP, *Non-linear least square spline approximation*, Tech. Rep., The Flinders University of South Australia, Australia, 1971, 22 pp.

[24] ———, *Curve fitting by splines as an example of unconstrained non-linear optimization*, Manuscript, 1972.

[25] G. GOLUB AND V. PEREYRA, *The differentiation of pseudoinverses and nonlinear least squares problems whose variables separate*, Tech. Rep. STAN-CS-72-261, Stanford Univ., Stanford, Calif.,' 1972.

[26] M. KLEIN, Personal communication, Los Alamos Scientific Laboratory, N.M., 1970.

[27] G. W. STEWART, Personal communication, Univ. of Texas, Austin, 1972.

[28] W. KAHAN, Personal communication, Univ. of California, Berkeley, 1971.

[29] PER-ÅKE WEDIN, *Perturbation bounds in connection with singular value decomposition*, BIT, 12 (1972), pp. 99–111.

[30] ———, *On pseudoinverses of perturbed matrices*, Tech. Rep., Department of Computer Science, Lund Univ., Sweden, 1969, 56 pp.

[31] A. BEN-ISRAEL, *A Newton–Raphson method for the solution of systems of equations*, J. Math. Anal. Appl., 15 (1966), pp. 243–252.

[32] G. H. GOLUB AND J. WILKINSON, *Iterative refinement of least squares solutions*, Numer. Math., 9 (1966), pp. 139–148.

[33] G. W. STEWART, *On the continuity of the generalized inverse*, SIAM J. Appl. Math., 17 (1969), pp. 33–45.

[34] H. P. DECELL AND J. FRIES, *On the derivative of the generalized inverse of a matrix*, J. Linear Alg., submitted for publication (1972).

[35] R. H. BARTELS, *Nonlinear least squares without derivatives: an application of the QR matrix decomposition*, CNA-44, Center for Numerical Analysis, Univ. of Texas, Austin, 1972, 47 pp.

[36] M. R. OSBORNE, *Some special nonlinear least squares problems*, Manuscript, 1972.

14

GENERALIZED CROSS-VALIDATION AS A METHOD FOR CHOOSING A GOOD RIDGE PARAMETER (WITH M. HEATH AND G. WAHBA)

TECHNOMETRICS (21:2), 215–223 (1979).

The paper I co-authored with Gene and Grace Wahba was written while I was a graduate student at Stanford. The division of labor was roughly that Grace came up with the background theory and motivating applications, Gene suggested the algorithmic approach, and I did the implementation. Among the graduate students at Stanford at the time, I suppose I was a natural choice for this project because I had previously worked on ill-posed problems at Oak Ridge National Laboratory, so I already knew about regularization, etc. I thought we made a very complementary team. Grace was very much at home with reproducing kernel Hilbert spaces, which provided the theoretical framework for this research, whereas my knowledge of them is still rather fuzzy to this day. But Grace wouldn't have been able to push her ideas through to concrete fruition and prove their merit without the computational know-how that Gene and I brought to the project.

I did not follow up with any subsequent work in that area, so in a sense that project had relatively little impact on my subsequent career, but on the other hand, the paper we wrote has been one of my most cited publications, so in that sense its impact has been substantial.

Michael Heath
Urbana, Illinois, USA

Gene has been a valued mentor and very dear friend from my graduate student days in the 1960s up to the present time, and I always looked forward then, and still do, to the opportunity to talk to him whenever it presents itself. Sometime around 1975 Gene was visiting Zürich and invited me to visit him and to give a talk at ETH. Peter Huber was there at the time and invited the two of us to join him and his wife at their ski lodge in Klosters. Gene and I took the train from Zürich to Klosters, and there on the train, the ideas in the paper were hatched. Gene then took them back to Stanford where Michael Heath ran the experiments reported in the paper. It was a memorable trip due to Gene's and Peter's warm hospitality and the exquisite beauty of the Swiss mountains in the winter. I still remember watching the skiers from the Hubers' lodge come down out of the mountains in the twilight and the excitement of the hatching of our project. The results in this paper have been tremendously influential in my later work. The idea is a method for solving the so-called bias-variance tradeoff, and it or various variants of it appear in almost everything I have done since.

Grace Wahba
Madison, Wisconsin, USA

TECHNOMETRICS ©, VOL. 21, NO. 2, MAY 1979

Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter

Gene H. Golub
Department of Computer Science
Stanford University
Stanford, CA 94303

Michael Heath
Computer Sciences Division
Oak Ridge National Laboratory
Oak Ridge, TN 37830

Grace Wahba
Department of Statistics
University of Wisconsin
Madison, WI 53705

Consider the ridge estimate $\hat{\beta}(\lambda)$ for β in the model $y = X\beta + \epsilon$, $\epsilon \sim \mathfrak{N}(0, \sigma^2 I)$, σ^2 unknown, $\hat{\beta}(\lambda) = (X^TX + n\lambda I)^{-1} X^Ty$. We study the method of generalized cross-validation (GCV) for choosing a good value $\hat{\lambda}$ for λ, from the data. The estimate $\hat{\lambda}$ is the minimizer of $V(\lambda)$ given by

$$V(\lambda) = \frac{1}{n} \|(I - A(\lambda))y\|^2 \Big/ \left[\frac{1}{n} \text{Trace}\,(I - A(\lambda))\right]^2,$$

where $A(\lambda) = X(X^TX + n\lambda I)^{-1}X^T$. This estimate is a rotation-invariant version of Allen's PRESS, or ordinary cross-validation. This estimate behaves like a risk improvement estimator, but does not require an estimate of σ^2, so can be used when $n - p$ is small, or even if $p \geq n$ in certain cases. The GCV method can also be used in subset selection and singular value truncation methods for regression, and even to choose from among mixtures of these methods.

KEY WORDS

Ridge regression
Cross-validation
Ridge parameter

1. INTRODUCTION

Consider the standard regression model

$$y = X\beta + \epsilon \qquad (1.1)$$

where y and ϵ are column n-vectors, β is a p-vector and X is an $n \times p$ matrix; ϵ is random with $E\epsilon = 0$, $E\epsilon\epsilon^T = \sigma^2 I$, where I is the $n \times n$ identity.

For $p \geq 3$, it is known that there exist estimates of β with smaller mean square error than the minimum variance unbiased, or Gauss-Markov, estimate $\hat{\beta}(0) = (X^TX)^{-1} X^Ty$. (See Berger [8], Thisted [39], for recent results and references to the earlier literature.) Allowing a bias may reduce the variance tremendously.

In this paper we primarily consider the (one parameter) family of ridge estimates $\hat{\beta}(\lambda)$ given by

$$\hat{\beta}(\lambda) = (X^TX + n\lambda I)^{-1}X^Ty. \qquad (1.2)$$

The estimate $\hat{\beta}(\lambda)$ is the posterior mean of β if β has the prior $\beta \sim \mathfrak{N}(0, aI)$, and $\lambda = \sigma^2/na$. $\hat{\beta}(\lambda)$ is also the solution to the problem:

Find β which satisfies the constraint

$$\|\beta\| = \gamma$$

and for which

$$\frac{1}{n} \|y - X\beta\| = \min.$$

Here $\|\cdot\|$ indicates the Euclidean norm and we use this norm throughout the paper. Introducing the

Received June 1977; revised April 1978

Lagrangian we find that the above problem is equivalent to finding the minimum over β of

$$\frac{1}{n}\|y - X\beta\|^2 + \lambda\|\beta\|^2 \qquad (1.3)$$

where λ is a Lagrange multiplier. Methods for computing λ given γ are given in [17]. See [29] for discussion of (1.3). The method of minimizing equation (1.3), or its Hilbert space generalizations, is called the *method of regularization* in the approximation theory literature (see [21, 44] for further references).

It is known that for any problem there is a $\lambda > 0$ for which the expected mean square error $E\|\beta - \hat{\beta}(\lambda)\|^2$ is less than the Gauss-Markov estimate; however the λ which minimizes, say $E\|\beta - \hat{\beta}(\lambda)\|^2$, or any other given nontrivial quadratic loss function depends on σ^2 and the unknown β.

There has been a substantial amount of interest in estimating a good value of λ from the data. See [10, 11, 12, 15, 20, 22, 23, 25, 26, 27, 30, 31, 32, 35, 38, 39]. A conservative guess might put the number of published estimates for λ at several dozen.

In this paper we examine the properties of the method of generalized cross-validation (GCV) for obtaining a good estimate of λ from the data. The GCV estimate of λ in the ridge estimate (1.2) is the minimizer of $V(\lambda)$ given by

$$V(\lambda) = \frac{1}{n}\|(I - A(\lambda))y\|^2 \Big/ \left[\frac{1}{n}\mathrm{Trace}\,(I - A(\lambda))\right]^2, \qquad (1.4)$$

where

$$A(\lambda) = X(X^TX + n\lambda I)^{-1}X^T. \qquad (1.5)$$

A discussion of the source of $V(\lambda)$ will be given in Section 2. This estimate is a rotation-invariant version of Allen's PRESS or ordinary cross-validation, as described in Hocking's discussion to Stone's paper [36] (see also Allen [3], and Geisser [13]).

Let $T(\lambda)$ be the mean square error in estimating $X\beta$, that is,

$$T(\lambda) = \frac{1}{n}\|X\beta - X\hat{\beta}(\lambda)\|^2. \qquad (1.6)$$

It is straightforward to show that

$$ET(\lambda) = \frac{1}{n}\|(I - A(\lambda))g\|^2 + \frac{\sigma^2}{n}\mathrm{Tr}\,A^2(\lambda) \qquad (1.7)$$

where

$$g = X\beta.$$

An unbiased estimator $\hat{T}(\lambda)$ of $ET(\lambda)$, for $n > p$, is given by

$$\hat{T}(\lambda) = \frac{1}{n}\|(I - A(\lambda))y\|^2 - \frac{2\hat{\sigma}^2}{n}\mathrm{Tr}(I - A(\lambda)) + \hat{\sigma}^2, \qquad (1.8)$$

where

$$\hat{\sigma}^2 = \frac{1}{n-p}\|(I - X(X^TX)^{-1}X^T)y\|^2.$$

Mallows [28, p. 672] has suggested choosing λ to minimize Mallows' C_L, which is equivalent to minimizing $n\,\hat{T}(\lambda)/\hat{\sigma}^2$. (This follows from [28] upon noting that $\|(I - A(\lambda))y\|^2$ is the "residual sum of squares.") The minimizer of $\hat{T}$ was also suggested by Hudson [25]. We shall call an estimate formed by minimizing $\hat{T}$ an RR ("range risk") estimate.

We shall show that the GCV estimate is, for large n, an estimate for the λ which approximately minimizes $ET(\lambda)$ of (1.7), *without the necessity of estimating σ^2*. As a consequence of not needing an estimate of σ^2, GCV can be used on problems where $n - p$ is small, or (in certain circumstances), where the "real" model may be

$$y_i = \sum_{j=1}^{\infty} x_{ij}\beta_j + \epsilon_i, \qquad i = 1, 2, \cdots, n. \qquad (1.9)$$

It is also natural for solving regression-like problems that come from an attempt to solve ill-posed linear operator equations numerically. In these problems there is typically no way of estimating σ^2 from the data. See Hanson [19], Hilgers [21], Varah [40] for descriptions of these problems. See Wahba [44] for the use of GCV in estimating λ in the context of ridge-type approximate solutions for ill-posed linear operator equations, and for further references to the numerical analysis literature. See Wahba, Wahba and Wold, and Craven and Wahba [9, 42, 43, 45, 46] for the use of GCV for curve smoothing, numerical dif ferentiation, and the optimal smoothing of density and spectral density estimates. At the time of this writing, the only other methods we know of for estimating λ from the data without either knowledge of or an estimate of σ^2, are PRESS and maximum likelihood, to be described. We shall indicate why GCV can be expected to be generally better than either. (PRESS and GCV will coincide if XX^T is a circulant matrix.)

A fundamental tool in our analysis and in our computations is the *singular value decomposition.* Given any $n \times p$ matrix X, we may write

$$X = UDV^T$$

where U is an $n \times n$ orthogonal matrix, V is a $p \times p$ orthogonal matrix, and D is an $n \times p$ diagonal matrix whose entries are the square roots of the eigenvalues of X^TX. The number of non-zero entries in D is equal to the rank of X. The singular value decomposition arises in a number of statistical applications [18]. Good numerical procedures are given in [16].

In Section 2 we derive the GCV estimate as a rotation-invariant version of Allen's PRESS and discuss why it should be generally superior to PRESS. In Section 3 we give some theorems concerning its properties. In Section 4 we show how GCV can be used in other regression procedures, namely, subset selection, and eigenvalue truncation, or principal components. Indeed GCV can be used to compare between the best of the three different methods, or mixtures, of them, if you will. In Section 5 we present the results of a Monte Carlo example.

2. THE GENERALIZED CROSS-VALIDATION ESTIMATE OF λ AS AN INVARIANT VERSION OF ALLEN'S PRESS

The Allen's PRESS, or ordinary cross-validation estimate of λ, goes as follows. Let $\beta^{(k)}(\lambda)$ be the ridge estimate (1.2) of β with the kth data point y_k, omitted. The argument is that if λ is a good choice, then the kth component $[X\beta^{(k)}(\lambda)]_k$ of $X\beta^{(k)}(\lambda)$ should be a good predictor of y_k. Therefore, the Allen's PRESS estimate of λ is the minimizer of

$$P(\lambda) = \frac{1}{n}\sum_{k=1}^{n} ([X\beta^{(k)}(\lambda)]_k - y_k)^2. \qquad (2.1)$$

It has been observed by one of the referees that $P(\lambda)$ may be viewed as a direct sample estimate of $\frac{1}{n}E_{y^*}\|y^* - X\hat{\beta}(\lambda)\|^2 \equiv T(\lambda) + \sigma^2$, where here $\hat{\beta}(\lambda)$ is supposed fixed, y^* is a future hypothetical observation vector, and E_{y^*} denotes expectation over the distribution of y^*.

It can be shown, by use of the Sherman-Morrison-Woodbury formula (see [24]), that

$$P(\lambda) = \frac{1}{n}\|B(\lambda)(I - A))y\|^2, \qquad (2.2)$$

where $B(\lambda)$ is the diagonal matrix with jjth entry $1/(1 - a_{jj}(\lambda))$, $a_{jj}(\lambda)$ being the jjth entry of $A(\lambda) = X(X^TX + n\lambda I)^{-1}X^T$.

Although the idea of PRESS is intuitively appealing, it can be seen that in the extreme case where the entries of X are 0 except for x_{ii}, $i = 1, 2, \cdots, p$, then $[X\beta^{(k)}(\lambda)]_k$ cannot be expected to be a good predictor of y_k. In fact, in this case $A(\lambda)$ is diagonal.

$$P(\lambda) = \frac{1}{n}\sum_{k=1}^{n} y_k^2,$$

and so $P(\lambda)$ does not have a unique minimizer. It is reasonable to conclude that PRESS would not do very well in the near diagonal case. If β and ϵ both have spherical normal priors, then various arguments can be brought to bear that any good estimate of λ should be invariant under rotations of the (measurement) coordinate system. The GCV estimate is a rotation-invariant form of ordinary cross-validation. It may be derived as follows: Let the singular value decomposition [16] of X be

$$X = UDV^T.$$

Let W be the unitary matrix which diagonalizes the circulants. (See Bellman [7], Wahba [41].) In complex form the jkth entry $[W]_{jk}$ of W is

$$[W]_{jk} = \frac{1}{\sqrt{n}}e^{2\pi ijk/n}, \qquad j, k = 1, 2, \cdots, n.$$

The GCV estimate for λ can be defined as the result of using Allen's PRESS on the transformed model

$$\begin{aligned}\tilde{y} = WU^Ty &= WDV^T\beta + WU^T\epsilon\\ &\equiv \tilde{X}\beta + WU^T\epsilon.\end{aligned}$$

The new "data vector" is $\tilde{y} = (\tilde{y}_1, \cdots, \tilde{y}_n)^T$, and the new "design matrix" is $\tilde{X} = WDV^T$. $\tilde{X}\tilde{X}^*$ ("*" means complex conjugate transpose) is a circulant matrix (see [6,41]). Thus intuitively, $[\tilde{X}\beta^{(k)}(\lambda)]_k$ should contain a "maximal" amount of information about $\tilde{y}_k$, on the average. By substituting X and y into (2.2), and observing that $\tilde{A}(\lambda) \equiv \tilde{X}(\tilde{X}^*\tilde{X} + n\lambda I)^{-1}\tilde{X}^*$ is a circulant matrix and hence constant down the diagonals, and $\tilde{A}(\lambda)$ and $A(\lambda)$ have the same eigenvalues, it is seen that $P(\lambda)$ becomes $V(\lambda)$ (see (1.4)) given by

$$\begin{aligned}V(\lambda) &= \frac{1}{n}\|(I - \tilde{A}(\lambda))\tilde{y}\|^2 \Big/ \left[\frac{1}{n}\mathrm{Tr}(I - \tilde{A}(\lambda))\right]^2\\ &\equiv \frac{1}{n}\sum_{\nu=1}^{n}\left(\frac{n\lambda}{\lambda_{\nu n} + n\lambda}\right)^2 z_\nu^2 \Big/ \left[\frac{1}{n}\sum_{\nu=1}^{p}\frac{n\lambda}{\lambda_{\nu n} + n\lambda} + n - p\right]^2 \qquad (2.3)\end{aligned}$$

where $z = (z_1, \cdots, z_n)^T = U^Ty$ and $\lambda_{\nu n}$, $\nu = 1, 2, \cdots, n$, are the eigenvalues of XX^T, $\lambda_{\nu n} = 0$, $\nu > p$.

It can also be shown that $V(\lambda)$ is a weighted version of $P(\lambda)$, namely

$$V(\lambda) \equiv \frac{1}{n}\sum_{k=1}^{n} ([X\beta^{(k)}(\lambda)]_k - y_k)^2 w_k^{(\lambda)}$$

where

$$w_k(\lambda) = \frac{1 - a_{kk}(\lambda)}{1 - \frac{1}{n}\mathrm{Tr}\,A(\lambda)}.$$

We define the GCV estimate of λ as the minimizer of (1.4), equivalently (2.3), and proceed to an investigation of its properties.

3. PROPERTIES OF THE GCV ESTIMATE OF λ

Theorem 1 (The GCV Theorem).

Let $\mu_1 = \frac{1}{n} \operatorname{Tr} A(\lambda)$, $\mu_2 = \frac{1}{n} \operatorname{Tr} A^2(\lambda)$, $b^2 =$

$$\frac{1}{n} \|(I - A(\lambda))g\|^2.$$

Then

$$\frac{ET(\lambda) - EV(\lambda) + \sigma^2}{ET(\lambda)} = \frac{-\mu_1(2 - \mu_1)}{(1 - \mu_1)^2} + \frac{\sigma^2}{b^2 + \sigma^2\mu_2} \frac{\mu_1^2}{(1 - \mu_1)^2} \qquad (3.1)$$

and so

$$\frac{|ET(\lambda) - EV(\lambda) + \sigma^2|}{ET(\lambda)} < \left(2\mu_1 + \frac{\mu_1^2}{\mu_2}\right) \frac{1}{(1 - \mu_1)^2}$$

whenever $0 < \mu_1 < 1$.

Proof: Since $ET = b^2 + \sigma^2\mu_2$ $EV = [b^2 + \sigma^2(1 - 2\mu_1 + \mu_2)]/(1 - \mu_1)^2$, the result follows from

$$ET - EV = (b^2 + \sigma^2\mu_2)\left(1 - \frac{1}{(1 - \mu_1)^2}\right) - \sigma^2 \frac{(1 - 2\mu_1)}{(1 - \mu_1)^2}$$

$$ET + \sigma^2 - EV = ET\left(1 - \frac{1}{(1 - \mu_1)^2}\right) + \sigma^2 \frac{\mu_1^2}{(1 - \mu_1)^2}$$

Remark: This theorem implies that if

$$\frac{1}{n} \operatorname{Tr} A(\lambda) = \mu_1 \to 0 \qquad \text{as } n \to \infty$$

and

$$\left(\frac{1}{n} \operatorname{Tr} A(\lambda)\right)^2 \bigg/ \left(\frac{1}{n} \operatorname{Tr} A^2(\lambda)\right) = \frac{\mu_1^2}{\mu_2} \to 0. \qquad \text{as } n \to \infty$$

then the difference between $ET(\lambda) + \sigma^2$ and $EV(\lambda)$ is small compared to $ET(\lambda)$. This result and the fact that in the extreme diagonal case $P(\lambda)$ does not have a unique minimum suggests that the minimizer of $V(\lambda)$ is preferable to the minimizer of $P(\lambda)$ if one wants to choose λ to minimize

$$\frac{1}{n} E_{y^*} \|y^* - X\beta(\lambda)\|^2.$$

Corollary: Let

$$h = \left(2\mu_1 + \frac{\mu_1^2}{\mu_2}\right) \frac{1}{(1 - \mu_1)^2}$$

Let λ^0 be the minimizer of $ET(\lambda)$. Then $EV(\lambda)$ always has a (possibly local) minimum $\tilde{\lambda}$ so that the "expectation inefficiency" I^0 defined by

$$I^0 = \frac{ET(\tilde{\lambda})}{ET(\lambda^0)}$$

satisfies

$$I^0 \leq \frac{1 + h(\lambda^0)}{1 - h(\tilde{\lambda})}.$$

Remark: This corollary says that if $h(\lambda^0)$ and $h(\tilde{\lambda})$ are small then the mean square error at the minimizer of $EV(\lambda)$ is not much bigger than the minimum possible mean square error $\min_\lambda ET(\lambda)$.

Proof: Let $\Lambda = \{\lambda\colon 0 \leq \lambda \leq \infty,\ \mathrm{EV}(\lambda) - \sigma^2 \leq T(\lambda^0)(1 + h(\lambda^0))\}$.

Since

$$ET(\lambda)(1 - h(\lambda)) < EV(\lambda) - \sigma^2 < ET(\lambda)(1 + h(\lambda)),$$
$$0 \leq \lambda < \infty,$$

and ET, EV and h are continuous functions of λ, then Λ is a non-empty closed set. If 0 is not a boundary point of Λ, then $EV(\lambda) - \sigma^2$ has at least one minimum in the interior of Λ, call it $\tilde{\lambda}$. (See Figure 1.) Now by the theorem

$$ET(\tilde{\lambda})(1 - h(\tilde{\lambda})) < EV(\tilde{\lambda}) - \sigma^2 < ET(\lambda^0)(1 + h(\lambda^0))$$

and so

$$I^0 = \frac{T(\tilde{\lambda})}{T(\lambda^0)} \leq \frac{1 + h(\lambda^0)}{1 - h(\tilde{\lambda})}.$$

If Λ includes 0, then $\tilde{\lambda}$ may be on the boundary of Λ, i.e., $\tilde{\lambda} = 0$, but the above bound on I^0 still holds.

Example 1. Note that

$$\mu_1 = \frac{1}{n} \operatorname{Tr} A = \frac{1}{n} \sum_{\nu=1}^{p} \frac{\lambda_{\nu n}}{\lambda_{\nu n} + n\lambda} \leq \frac{p}{n}$$

$$\frac{\mu_1^2}{\mu_2} = \frac{\left(\frac{1}{n} \operatorname{Tr} A\right)^2}{\frac{1}{n} \operatorname{Tr} A^2} = \frac{1}{n} \frac{\left(\sum_{\nu=1}^{p} \frac{\lambda_{\nu n}}{\lambda_{\nu n} + \lambda}\right)^2}{\sum_{\nu=1}^{p} \left(\frac{\lambda_{\nu n}}{\lambda_{\nu n} + \lambda}\right)^2} \leq \frac{p}{n}.$$

Then

$$h \leq 3 \frac{p}{n} \frac{1}{\left(1 - \frac{p}{n}\right)^2}.$$

Hence for p fixed and $n \to \infty$, it follows that

$$I^0 \leq 1 + 6 \frac{p}{n} + 0\left(\frac{p}{n}\right).$$

Example 2. p > n.

It is not necessary that $p \ll n$ for I^0 to tend to 1, as this example suggests. What is required is that XX^T become ill conditioned for n large.

Let

$$y_i = \sum_{j=1}^{p} x_{ij}\beta_j + \epsilon_i, \qquad i = 1, 2, \cdots,$$
$$p > n \tag{3.2}$$

with

$$\sum_{j=1}^{\infty} x_{ij}^2 \le k_1 < \infty, \quad \text{all} \quad i, \qquad \sum_{j=1}^{\infty} \beta_j^2 \le k_2 < \infty.$$

Suppose

$$\lim_{n\to\infty} \frac{1}{n} \operatorname{Tr} XX^T = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{\infty} x_{ij}^2 = k_3 < \infty$$

and suppose the eigenvalues $\{\lambda_{\nu n}, \nu = 1, 2, \cdots, n\}$ of XX^T satisfy

$$\lambda_{\nu n} \simeq n\nu^{-m},$$

say, for some $m > 1$; $(k_3 = \sum_{\nu=1}^{\infty} \nu^{-m})$.

Then

$$\mu_1 = \frac{1}{n} \sum_{\nu=1}^{n} \frac{\lambda_{\nu n}}{\lambda_{\nu n} + n\lambda} \simeq \frac{1}{n} \sum_{\nu=1}^{n} \frac{1}{1 + \lambda\nu^m}$$
$$\simeq \frac{1}{n} \int_0^\infty \frac{dx}{(1 + \lambda x^m)} = \frac{1}{n\lambda^{1/m}} \int_0^\infty \frac{dx}{(1 + x^m)}$$
$$\mu_2 = \frac{1}{n} \sum_{\nu=1}^{n} \left(\frac{\lambda_{\nu n}}{\lambda_{\nu n} + n\lambda}\right)^2 \simeq \frac{1}{n} \sum_{\nu=1}^{n} \frac{1}{(1 + \lambda\nu^m)^2}$$
$$\simeq \frac{1}{n} \int_0^\infty \frac{dx}{(1 + \lambda x^m)^2} = \frac{1}{n\lambda^{1/m}} \int_0^\infty \frac{dx}{(1 + x^m)^2}.$$

and $\mu_1 \to 0$, $\mu_1^2/\mu_2 \to 0$ if $n\lambda^{1/m} \to \infty$.
Now

$$b^2(\lambda) = \lambda\, \beta^T(X^TX + n\lambda I)^{-1}(n\lambda)X^TX(X^TX + n\lambda I)^{-1}\beta$$
$$\le \frac{\lambda}{2} \|\beta\|^2 \le \frac{\lambda}{2} k_2,$$

since the largest eigenvalue of

$$(X^TX + n\lambda I)^{-1}(n\lambda)X^TX(X^TX + n\lambda I)^{-1}$$
$$= \max_\nu \frac{(\lambda_{\nu n})(n\lambda)}{(\lambda_{\nu n})^2 + (n\lambda)^2} \le \frac{1}{2}.$$

As $n \to \infty$, the minimizing sequence $\lambda^0 = \lambda^0(n)$ of $ET(\lambda) = b^2(\lambda) + \sigma^2\mu_2(\lambda)$ clearly must satisfy $\lambda^0 \to 0$, $n(\lambda^0)^{1/m} \to \infty$, so that the GCV Theorem may be applied. It is proved in [9, 44] in a different context that $\tilde{\lambda}$ as well as λ^0 satisfies $(n\lambda^{1/m}) \to \infty$ so that $h(\tilde{\lambda}) \to 0$, $h(\lambda^0) \to 0$ and $I^0 \downarrow 1$ as $n \to \infty$.

Instead of viewing β as fixed but unknown, suppose that β has the prior $\beta \sim \mathfrak{N}(0, aI)$. Let E_β be expectation with respect to the prior. (We reserve E for expectation with respect to ϵ.) Then

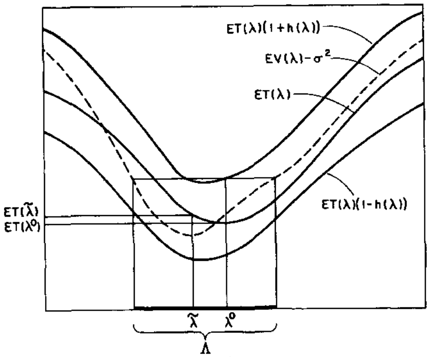


FIGURE 1. Graphical suggestion of the proof of the corollary to the GCV theorem.

Theorem 2.

The minimizer of $E_\beta EV(\lambda)$ is the same as the minimizer of $E_\beta ET(\lambda)$ and is $\lambda = \sigma^2/na$.
Proof: Since $Eg\, g^T = E\, X\beta\beta^TX^T = a\, XX^T$,

$$E_\beta ET(\lambda) = \frac{a}{n} \operatorname{Tr} (I - A)^2XX^T + \frac{\sigma^2}{n} \operatorname{Tr} A^2$$
$$E_\beta EV(\lambda) = \left[\frac{a}{n} \operatorname{Tr} (I - A)^2XX^T + \frac{\sigma^2}{n} \operatorname{Tr} (I - A)^2\right] \Big/ \left[\frac{1}{n} \operatorname{Tr} (I - A)\right]^2. \tag{3.3}$$

The proof proceeds by differentiating (3.3) with respect to λ and setting the remainder equal to 0. This calculation has appeared elsewhere [43 p. 8], and will be omitted.

4. GCV IN SUBSET SELECTION AND GENERAL LINEAR MODEL BUILDING

Let $y = g + \epsilon$, where g is a fixed (unknown) n-vector and $\epsilon \sim \mathfrak{N}(0, \sigma^2I)$, σ^2 unknown. Let $A(\nu)$, ν in some index set, be a family of symmetric nonnegative definite $n \times n$ matrices and let

$$\mu_1(\nu) = \frac{1}{n} \operatorname{Tr} A(\nu)$$
$$\mu_2(\nu) = \frac{1}{n} \operatorname{Tr} A^2(\nu).$$

Letting

$$T(\nu) = \frac{1}{n} \| g - A(\nu)y \|^2$$

and $V(\nu)$ as before with $A(\lambda)$ replaced by $A(\nu)$, then (3.1) clearly holds irrespective of the nature of A.

A different way of dealing with ill conditioning in the design matrix is to reduce the number of predictor variables by choosing a subset $\beta_{j_1}, \beta_{j_2}, \cdots, \beta_{j_k}$ of the β_i's. Let ν be an index on the 2^p possible subsets of $\beta_1, \cdots, \beta_p$, let $X^{(\nu)}$ be the $n \times k(\nu)$ design matrix corresponding to the ν^{th} subset, and let

$$\hat{\beta}(\nu) = (X^{(\nu)T}X^{(\nu)})^{-1}X^{(\nu)}y$$

$$A(\nu) = X^{(\nu)}(X^{(\nu)T}X^{(\nu)})X^{(\nu)}y.$$

Then

$$\mu_1 = k/n, \qquad \mu_1{}^2/\mu_2 = k/n.$$

Mallows [28] suggestion to choose the subset minimizing C_p becomes, in our notation, the equivalent of minimizing $\hat{T}(\cdot)$ of (1.8) with $A(\lambda)$ replaced by $A(\nu)$, see also Allen [2]. This assumes that an estimate of σ^2 is available. Parzen [33] has observed that, if one prefers to choose a subset without estimating σ^2, (because one believed in the model (3.2), say), GCV can be used. The subset of size $\leq k_{\max}$ with smallest V can be chosen, knowing that

$$\left| \frac{ET(\nu) - EV(\nu) - \sigma^2}{ET(\nu)} \right| \leq \frac{k_{\max}}{n},$$

even if the model (3.2) is nontrivially true.

In the subset selection case, GCV asymptotically coincides with the use of Akaike's information criterion AIC [1] since

$$\begin{aligned} \text{AIC} &= (-2) \log \text{maximum likelihood} + 2k \\ &= n \log \frac{1}{n} \|(I - A)y\|^2 + 2k \end{aligned}$$

and so

$$e^{AIC/n} = \frac{\frac{1}{n}\|(I - A)y\|^2}{\left(e - \frac{k}{n}\right)^2} \approx \frac{\frac{1}{n}\|(I - A)y\|^2}{\left(1 - \frac{k}{n}\right)^2} = V$$

as

$$\frac{k}{n} \to 0.$$

We thank E. Parzen for pointing this out. M. Stone, [37] has investigated the relations between AIC and (ordinary) cross-validation.

Another approach, the principal components approach, is also popular in solving ill-posed linear operator equations, see Baker et al. [6], Hanson [19], Varah [40]). The method is to replace X by $X(\nu)$ defined by $X(\nu) = U\, D(\nu)\, V^T$, where $D(\nu)$ is the diagonal matrix of singular values of V with all but the ν^{th} subset of singular values set equal to 0. Then

$$\begin{aligned} A(\nu) &= U\, D(\nu)\, (D(\nu)D(\nu)^T)^+\, D(\nu)\, U^T \\ &= U \begin{pmatrix} 1 & & & & & 0 \\ & \ddots & & & & \\ & & 1 & & & \\ & & & 0 & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{pmatrix} U^T \end{aligned}$$

where the ones are located at positions of the ν^{th} subset of singular values, and, again $\mu_1 \leq p/n$, $\mu_1{}^2/\mu_2 \leq p/n$, where p can be replaced by the number of singular values in the largest subset considered.

In fact, it is reasonable to select from among any family $\{A(\nu)\}$ of matrices for which the corresponding μ_1 and $\mu_1{}^2/\mu_2$ are uniformly small, by choosing that member for which $V(\nu)$ is smallest. Mixtures of the above methods, e.g. a ridge method on a subset, can be handled this way. Note that the conditions μ_1 small, $\mu_1{}^2/\mu_2$ small are just those conditions which make it plausible that the "signal" g can be separated from the noise. These conditions say that the A matrix essentially maps the data vector (roughly) into some much smaller subspace than the whole space. Parzen [34] has also indicated how GCV can be used to choose the order of an autoregressive model to fit a stationary time series.

5. A NUMERICAL EXAMPLE

We choose a discretization of the Laplace transform as given in Varah, [40, p. 262] as an example in which X^TX is very ill conditioned.

We emphasize that the following is nothing more than a single example, with a single X and β. It does not indicate what may happen as X and β are varied. It is intended as an indication of the type of Monte Carlo evaluation study that an experimenter might perform with the particular X that he has at hand, and perhaps one or several β that represent the class of β's he believes he is likely to encounter. We suggest that an experimenter with particular design matrix at hand evaluate candidate methods (at least crudely), perhaps including subset selection and/or principal components, as well as ridge methods against his X and against a realistic set of β, before final selection of a method. The values for n and p in the experiment presented here were 21 and 10 and the condition number of X, namely the ratio of the largest to the smallest (non-zero) singular value, was 1.54×10^6. The value of $\|X\beta\|^2$ was 370.84.

Four values of σ^2, namely $\sigma^2 = 10^{-8}, 10^{-6}, 10^{-4}$ and 10^{-2} were tried and for each value of σ^2 the experiment was replicated four times, giving a total of 16 runs. The ϵ_i were generated as pseudo-random $\mathfrak{N}(0, \sigma^2)$ independent *r.v.*'s, $V(\lambda)$ was computed using the right-hand side of (2.3) and the Golub-Reinsch singular value decomposition [16]. The minimizer $\hat{\lambda}$ of $V(\lambda)$ was determined by a global search. $T(\lambda)$ was also computed and the relative inefficiencies I_D and I_R of $\hat{\lambda}$ defined by

$$\begin{aligned} I_D &= \|\beta - \hat{\beta}_{\hat{\lambda}}\|^2 / (\min_{\lambda} \|\beta - \hat{\beta}_{\lambda}\|^2) \\ I_R &= T(\hat{\lambda}) / \min_{\lambda} T(\lambda) \end{aligned} \tag{5.1}$$

were computed. (D = "domain", R = "range.")

TABLE 1—*Observed inefficiencies in sixteen Monte Carlo runs.*

	Replication 1		Replication 2		Replication 3		Replication 4	
	I_D	I_R	I_D	I_R	I_D	I_R	I_D	I_R
				$\sigma^2=10^{-8}$, S/N ≈ 4200				
GCV	4.43	1.06	1.65	1.03	16.71	1.10	1.02	1.01
RR	1.46	1.00	1.66	1.03	8.69	1.01	1.22	1.03
MLE	1.67E3	1.31	1.45E2	1.23	2.00E3	1.53	9.12E3	1.51
PRESS	2.31E3	4.8E4	6.31E2	8.6E4	3.84E3	2.1E5	2.87E3	1.2E5
Min Sol'n	1.00	1.02	1.00	1.54	1.00	2.27	1.00	1.00
Min Data	1.20	1.00	2.89	1.00	5.97	1.00	1.00	1.00
				$\sigma^2=10^{-6}$, S/N ≈ 420				
GCV	1.92	1.05	1.32	1.00	1.51E2	1.26	2.20	1.02
RR	1.83	1.06	1.90	1.01	7.03E1	1.10	1.18	1.00
MLE	1.99E2	1.19	1.70E2	1.45	1.76E2	1.29	1.49E2	1.32
PRESS	5.80	1.01	2.41E2	1.39E4	36.37	2.43E3	67.00	6.07E2
Min Sol'n	1.00	1.38	1.00	1.02	1.00	1.20	1.00	1.03
Min Data	3.56	1.00	1.28	1.00	7.85	1.00	41.29	1.00
				$\sigma^2=10^{-4}$, S/N ≈ 42				
GCV	1.27	1.07	1.50	2.58	1.00	1.11	1.00	1.03
RR	1.18	1.08	1.03	2.27	1.07	1.13	1.00	1.03
MLE	1.56	1.20	12.16	3.43	1.90	1.49	2.97	1.07
PRESS	3.53	1.57	2.03	3.43	8.66	2.63	2.90	24.34
Min Sol'n	1.00	1.21	1.00	2.05	1.00	1.11	1.00	1.03
Min Data	3.26	1.00	1.16	1.00	2.39	1.00	1.16	1.00
				$\sigma^2=10^{-2}$, S/N ≈ 4.2				
GCV	1.40	2.47	2.01	1.60	1.59	1.01	31.20	17.2
RR	1.38	2.39	2.41	1.70	1.41	1.02	10.8	10.6
MLE	2.13	3.56	3.81	1.87	2.00	1.00	28.8	16.8
PRESS	1.04	1.01	2.02	2.68	1.00	1.22	2.16	21.5
Min Sol'n	1.00	1.31	1.00	1.01	1.00	1.25	1.00	1.98
Min Data	1.02	1.00	1.00	1.00	2.66	1.00	1.21	1.00

The results of a comparison with three other methods are also presented. The methods are, respectively,

1. PRESS, the minimizer of $P(\lambda)$.
2. Range risk, (RR) the minimizer of $\hat{T}(\lambda)$.
3. Maximum likelihood (MLE).

The maximum likelihood estimate is obtained from the model

$$y = X\beta + \epsilon$$

with $\epsilon \sim \mathfrak{N}(0, \sigma^2 I)$ and β having the prior distribution $\beta \sim \mathfrak{N}(0, aI)$. Then the posterior distribution of y is

$$y \sim \mathrm{N}(0, a(XX^T + n\lambda I)) \qquad (5.2)$$

where $\lambda = \sigma^2/na$. The ML estimate for λ from the model (5.2) is then the minimizer of $M(\lambda)$ given by

$$M(\lambda) = \frac{1}{n}\frac{y^T(I-A(\lambda))y}{[\mathrm{Det}(I-A(\lambda))]^{1/n}}. \qquad (5.3)$$

This estimate is the general form of the maximum likelihood estimate suggested by Anderssen and Bloomfield in the context of numerical differentiation [4,5]. It can be shown that the minimizer of $E_\beta\, E\, M(\lambda)$ is σ^2/na. However, it can also be shown that if β behaves as though it did not come from the prior

$$\left(\text{e.g. as in the model (1.9)}, \quad \sum_{i=1}^{\infty} \beta_i^2 < \infty\right),$$

then the minimizer of $E\, M(\lambda)$ may not be a good estimate of the minimizer of $ER(\lambda)$.

I_D and I_R of (5.1) were determined for each of these three methods as well as GCV and the results are presented in Table 1. The entries next to "Min Sol'n" and "Min Data" are the inefficiencies (5.1) with $\hat{\lambda}$ replaced by the minimizers of $\|\beta-\hat{\beta}_\lambda\|^2$ and $T(\lambda)$ respectively. S/N, the "signal to noise ratio" is defined by S/N $= [1/n\|X\beta\|^2/\sigma^2]^{1/2}$ Figure 2 gives a

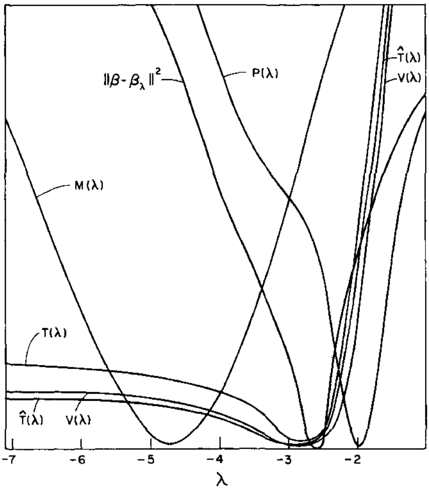


FIGURE 2. $V(\lambda)$, $T(\lambda)$, $\hat{T}(\lambda)$, $M(\lambda)$, $P(\lambda)$ and $\|\beta-\hat{\beta}_\lambda\|^2$.

plot of $V(\lambda)$, $\hat{T}(\lambda)$, $M(\lambda)$, $P(\lambda)$, $\|\beta-\hat{\beta}_\lambda\|^2$ and $T(\lambda)$ for Replicate 2 of the $\sigma^2 = 10^{-6}$ case. The $V(\lambda)$, $\hat{T}(\lambda)$ and $T(\lambda)$ curves tend to follow each other as predicted.

D. I. Gibbons [14] has recently completed a Monte Carlo comparison of 10 methods of choosing k. Three estimators, GCV, HKB (described in [23]), and RIDGM (described in [10,11]) were identified as the best performers in the examples studied. HKB and RIDGM use estimates of σ^2.

6. CONCLUSIONS

The generalized cross-validation method for estimating the ridge parameter in ridge regression has been given. This estimate does not require an estimate of σ^2, and thus may be used when the number of degrees of freedom for estimating σ^2 is small or even; in some cases, when the "real" model actually involves more than n parameters. The method may also be used to do subset selection or selection of principal components instead of ridge regression, or even to choose between various combinations of ridge, subset selection or principal components methods. A numerical example, briefly suggestive of the behavior of the method, has been carried out. It illustrates what an experimenter might wish to do to examine the properties of the method with respect to his/her design matrix.

7. ACKNOWLEDGMENTS

The work of Gene H. Golub was initiated while a guest of the Eidgenössische Technische Hochschule. He is very pleased to acknowledge the gracious hospitality and stimulating environment provided by Professors Peter Henrici and Peter Huber. His research was supported in part under Energy Research and Development Administration Grant E(04-3) PA # 30, and in part under U.S. Army Grant DAHC04-75-G-0185.

Michael Heath's research was supported in part under Energy Research and Development Administration Grant E(04-3) 326 PA # 30.

The work of Grace Wahba was initiated while she was a visitor at the Oxford University Mathematical Institute at the invitation of Professor J. F. C. Kingman. The hospitality of Professor Kingman, the Mathematical Institute, and St. Cross College, Oxford, is gratefully acknowledged. Her research was supported by the Science Research Council (GB), and by U. S. Air Force Grant AF-AFOSR-2363-C.

REFERENCES

[1] AKAIKE, H. (1974). A new look at the statistical model identification. *IEEE Transaction on Automatic Control, AC-19, 6,* 716–730.

[2] ALLEN, D. M. (1971). Mean square error of prediction as a criterion for selecting variables. *Technometrics, 13,* 469–475.

[3] ALLEN, D. M. (1974). The relationship between variable selection and data augmentation and a method for prediction. *Technometrics, 16,* 125–127.

[4] ANDERSSEN, B. and BLOOMFIELD, P. (1974). Numerical differentiation procedures for non-exact data. *Numer. Math., 22,* 157–182.

[5] ANDERSSEN, R. S. and BLOOMFIELD, P. (1974). A time series approach to numerical differentiation. *Technometrics, 16,* 69–75.

[6] BAKER, C. T. H., FOX, L., MAYERS, D. F., and WRIGHT, K. (1964). Numerical solution of Fredholm integral equations of the first kind. *Comp. J., 7,* 141–148.

[7] BELLMAN, R. (1960). *Introduction to Matrix Analysis.* New York: McGraw-Hill.

[8] BERGER, J. (1976). Minimax estimation of a multivariate normal mean under arbitrary quadratic loss. *J. Multivariate Analysis, 6,* 256–264.

[9] CRAVEN, P. and WAHBA, G. (1979). Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. *Numer. Math., 31,* 377–403.

[10] DEMPSTER, A. P. (1973). Alternatives to least squares in multiple regression, In *Multivariate Statistical Conference, Proceedings of the Research Seminar at Dalhousie University, Halifax, March 23-25, 1972,* ed. by D. G. Kabe and R. P. Gupta.

[11] DEMPSTER, A. P., SCHATZOFF, M., and WERMUTH, N. (1975). A simulation study of alternatives to ordinary least squares. *J. Amer. Statist. Assoc., 70,* 77–106.

[12] FAREBROTHER, R. W. (1975). The minimum mean square error linear estimator and ridge regression. *Technometrics, 17,* 127–128.

[13] GEISSER, S. (1975). The predictive sample reuse method with applications. *J. Amer. Statist. Assoc., 70,* 320–328.

[14] GIBBONS, D. I. (1978). A simulation study of some ridge estimators. General Motors Research Laboratories, Research Publication GMR-2659, Warren, Michigan.

[15] GOLDSTEIN, M., and SMITH, A. F. M. (1974). Ridge type estimators for regression analysis. *J. Roy. Statist. Soc., Ser. B, 36,* 284–291.

[16] GOLUB, G., and REINSCH, C. (1970). Singular value decomposition and least squares solutions. *Numer. Math.*, *14*, 403–420.

[17] GOLUB, G. H. (1973). Some modified matrix eigenvalue problems. *SIAM Review*, *15*, 318–334.

[18] GOLUB, G. H. and LUK, F. T. (1977). Singular value decomposition: applications and computations. *Transactions of the Twenty-Second Conference of Army Mathematicians*, 577–605.

[19] HANSON, R. J. (1971). A numerical method for solving Fredholm integral equations of the first kind using singular values. *SIAM J. Num. Anal.*, *8*, 616–622.

[20] HEMMERLE, W. J. (1975). An explicit solution for generalized ridge regression. *Technometrics*, *17*, 309–313.

[21] HILGERS, J. W. (1976). On the equivalence of regularization and certain reproducing kernel Hilbert space approaches for solving first kind problems. *SIAM J. Num. Anal.*, *13*, 172–184.

[22] HOERL, A. E., and KENNARD, R. W. (1976). Ridge regression: iterative estimation of the biasing parameter. *Comm. in Statist.*, *A5*, 77–88.

[23] HOERL, A. E., KENNARD, R. W., and BALDWIN, K. F. (1975). Ridge regression: some simulations. *Comm. in Statist.*, *4*, 105–123.

[24] HOUSEHOLDER, A. (1964). *The Theory of Matrices in Numerical Analysis*. New York: Blaisdell.

[25] HUDSON, H. M. (1974). Empirical Bayes estimation. Technical Report No. 58, Stanford University, Department of Statistics, Stanford, CA.

[26] LAWLESS, J. F. and WANG, P. (1976). A simulation study of ridge and other regression estimators. *Comm. in Statist.*, *A5*, 307–324.

[27] LINDLEY, D. V., and SMITH, A. F. M. (1972). Bayes estimate for the linear model (with discussion), part 1. *J. Roy. Statist. Soc.*, *B*, *34*, 1–41.

[28] MALLOWS, C. L. (1973). Some comments on C_p. *Technometrics*, *15*, 661–675.

[29] MARQUARDT, D. W. (1970). Generalized inverses, ridge regression, biased linear estimation and nonlinear estimation. *Technometrics*, *12*, 591–64.

[30] MARQUARDT, D. W., and SNEE, R. D. (1975). Ridge regression in practice. *The American Statistician*, *29*, 3–20.

[31] MCDONALD, G. and GALARNEAU, D. (1975). A Monte Carlo evaluation of some ridge-type estimators. *J. Amer. Statist. Assoc.*, *70*, 407–416.

[32] OBENCHAIN, R. L. (1975). Ridge Analysis following a preliminary test of the shrunken hypothesis. *Technometrics*, *17*, 431–446.

[33] PARZEN, E. (1976). Time series theoretic nonparametric statistical methods. Preliminary Report, Statistical Science Division, SUNY, Buffalo, New York.

[34] PARZEN, E. (1977). Forecasting and whitening filter estimation. Manuscript.

[35] ROLPH, J. E. (1976). Choosing shrinkage estimators for regression problems. *Comm. in Statist.*, *A5*, 789–802.

[36] STONE, M. (1974). Cross-validatory choice and assessment of statistical prediction. *J. Roy. Statist. Soc.*, *B*, *36*, 111–147.

[37] STONE, M. (1977). An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. *J. Roy. Statist. Soc.*, *B.*, *39*, 44–47.

[38] SWINDEL, B. F. (1976). Good ridge estimators based on prior information. *Comm. in Statist.*, *A5*, 985–997.

[39] THISTED, R. A. (1976). Ridge regression, minimax estimation, and empirical Bayes methods. Division of Biostatistics, Stanford University, Tech. Report No. 28.

[40] VARAH, J. M. (1973). On the numerical solution of ill-conditioned linear systems with applications to ill posed problems. *SIAM J. Num. Anal.*, *10*, 257–267.

[41] WAHBA, G. (1968). On the distribution of some statistics useful in the analysis of jointly stationary time series. *Ann. Math. Statist.*, *39*, 1849–1862.

[42] WAHBA, G. (1976). A survey of some smoothing problems and the method of generalized cross-validation for solving them. In *Proceedings* of the Conference on the Applications of Statistics, held at Dayton, Ohio, June 14–17, 1976, ed. by P. R. Krishnaiah.

[43] WAHBA, G. (1976). Optimal smoothing of density estimates. *Classification and Clustering*, pp. 423–458, ed. by J. Van Ryzin. New York: Academic Press.

[44] WAHBA, G. (1977). The approximate solution of linear operator equations when the data are noisy. *SIAM J. Num. Anal.*, *14*, 651–667.

[45] WAHBA, G., and WOLD, S. (1975). Periodic splines for spectral density estimation: the use of cross-validation for determining the correct degree of smoothing. *Comm. in Statist.*, *4*, 125–141.

[46] WAHBA, G., and WOLD, S. (1975). A completely automatic French curve: fitting spline functions by cross-validation. *Comm. in Statist.*, *4*, 1–17.

15

AN ANALYSIS OF THE TOTAL LEAST SQUARES PROBLEM (WITH C. VAN LOAN)

SIAM J. NUMER. ANAL. (17), 883–893 (1980).

Our total least squares work was prompted by my all-time favorite paper in numerical linear algebra:

G. H. Golub, "Some Modified Matrix Eigenvalue Problems", SIAM REVIEW 15, 318–334 (1973).

This paper is included in this volume and was a thrill of read back in 1973 when I had just completed graduate school. Privately, I have always called the paper "Look How Far You Can Run with QR, SVD, and Schur", a title that reminds me of Gene's uncanny ability to turn difficult problems into tractable ones with elegant solutions. Gene's SIREV paper is filled with ideas that kept the whole field busy for years: generalized eigenproblems, inverse eigenproblems, rank-1 modified eigenproblems, and of course, total least squares (TLS). TLS is there as an example of an eigenproblem that is buried inside a least squares problem and the paper we wrote was one of several that we did during the production of the first edition of our book.

Charlie Van Loan
Ithaca, New York, USA

SIAM J. NUMER ANAL.
Vol. 17, No. 6, December 1980

AN ANALYSIS OF THE TOTAL LEAST SQUARES PROBLEM*

GENE H. GOLUB† AND CHARLES F. VAN LOAN‡

Dedicated to Professor Garrett Birkhoff on the occasion of his seventieth birthday

Abstract. Total Least Squares (TLS) is a method of fitting that is appropriate when there are errors in both the observation vector b ($m \times 1$) and in the data matrix A ($m \times n$). The technique has been discussed by several authors, and amounts to fitting a "best" subspace to the points (a_i^T, b_i), $i = 1, \cdots, m$, where a_i^T is the ith row of A. In this paper a singular value decomposition analysis of the TLS problem is presented. The sensitivity of the TLS problem as well as its relationship to ordinary least squares regression is explored. An algorithm for solving the TLS problem is proposed that utilizes the singular value decomposition and which provides a measure of the underlying problem's sensitivity.

1. Introduction. In the least squares (LS) problem we are given an $m \times n$ "data matrix" A, a "vector of observations" b having m components, and a nonsingular diagonal matrix $D = \operatorname{diag}(d_1, \cdots, d_m)$, and are asked to find a vector x such that

$$\|D(b - Ax)\|_2 = \min. \tag{1.1}$$

Here $\|\cdot\|_2$ denotes Euclidean length. It is well known that any solution to the LS problem satisfies the following system of "normal equations:"

$$A^T D^2 A x = A^T D^2 b. \tag{1.2}$$

The solution is unique if rank $(A) = n$. However, regardless of the rank of A there is always a unique minimal 2-norm solution to the LS problem given by

$$x_{\mathrm{LS}} = (DA)^+ Db, \tag{1.3}$$

where $(DA)^+$ denotes the Moore-Penrose pseudo-inverse of DA.

In the (classical) LS problem there is an underlying assumption that all the errors are confined to the observation vector b. Unfortunately, this assumption is frequently unrealistic; sampling errors, human errors, modeling errors, and instrument errors may preclude the possibility of knowing the data matrix A exactly. Methods for estimating the effect of such errors on x_{LS} are given in Hodges and Moore [11] and Stewart [19]. The representation of data errors in a statistically meaningful way is a difficult task that can be appreciated by reading the survey article by Cochrane [2].

In this paper we analyze the method of total least squares (TLS), which is one of several fitting techniques that have been devised to compensate for data errors. A good way to motivate the method is to recast the ordinary LS problem as follows:

$$\text{minimize } \|Dr\|_2$$

$$\text{subject to } b + r \in \operatorname{Range}(A)$$

If $\|Dr\|_2 = \min$ and $b + r = Ax$, then x solves the LS problem (1.1). Thus the LS problem amounts to perturbing the observation b by a minimum amount r so the $b + r$ can be "predicted" by the columns of A.

* Received by the editors February 22, 1980.

† Department of Computer Science, Stanford University, Stanford California, 94305. The work of this author was supported in part by the U.S. Department of Energy under Contract DE-AC03-76SF00326 and the U.S. Army Research Office under Grant DAAG 29-78-G-0179.

‡ Department of Computer Science, Cornell University, Ithaca, New York, 14853. The work of this author was supported in part by the National Science Foundation under Contract MCS80-04106.

Now simply put, the idea behind total least squares is to consider perturbations of both b *and* A. More precisely, given the nonsingular weighting matrices

$$D=\operatorname{diag}(d_1,\cdots,d_m),\qquad d_i>0,\ i=1,\cdots,m,$$
$$T=\operatorname{diag}(t_1,\cdots,t_{n+1}),\qquad t_i>0,\ i=1,\cdots,n+1,$$

we seek to

$$\text{(1.4)}\qquad \begin{aligned}&\underset{E,r}{\text{minimize}}\ \|D[E\,|\,r]T\|_F\\ &\text{subject to } b+r\in \text{Range}\,(A+E).\end{aligned}$$

Here, $\|\cdot\|_F$ denotes the Frobenius norm, viz. $\|B\|_F^2=\sum_i\sum_j|b_{ij}|^2$. Once a minimizing $[\tilde{E}\,|\,\tilde{r}]$ is found, then any x satisfying

$$(A+\tilde{E})x=b+\tilde{r}$$

is said to solve the TLS problem (1.4). Thus, the TLS problem is equivalent to the problem of solving a nearest compatible LS problem $\min\|(A+\tilde{E})x-(b+\tilde{r})\|_2$ where "nearness" is measured by the weighted Frobenius norm above.

Total least squares is not a new method of fitting; the $n=1$ case has been scrutinized since the turn of the century. More recently, the method has been discussed in the context of the subset selection problem, see [9], [10], and [20]. In Deming [3] and Gerhold [4] the following more general problem is analyzed:

$$\text{(1.5)}\qquad \begin{aligned}&\underset{E,r}{\text{minimize}}\ \sum_{i=1}^{m}\left\{\Delta_i r_i^2+\sum_{j=1}^{n}\omega_{ij}e_{ij}^2\right\}\\ &\text{subject to } b+r\in\text{Range}\,(A+E),\end{aligned}$$

where $E=(e_{ij})$, $r^T=(r_1,\cdots,r_m)$, and the Δ_i and ω_{ij} are given positive weights.

The TLS approach to fitting has also attracted interest outside of statistics. For example, many algorithms for nonlinearly constrained minimization require estimates of the vector of Lagrange multipliers. This typically involves the solution of an LS problem where the matrix is the Jacobian of the "active constraints." Because of uncertainties in this matrix, Gill and Murray [5] have suggested using total least squares. Similar in spirit is the work of Barrera and Dennis [1], who have developed a "fuzzy Broyden" method for systems of nonlinear equations.

In the present paper we analyze the TLS problem by making heavy use of the singular value decomposition (SVD). As is pointed out in Golub and Reinsch [7] and more fully in Golub [6], this decomposition can be used to solve the TLS problem. We indicate how this can be accomplished in § 2. An interesting aspect of the TLS problem is that it may fail to have a solution. For example, if

$$A=\begin{bmatrix}1&0\\0&0\end{bmatrix},\quad b=\begin{bmatrix}1\\1\end{bmatrix},\quad D=T=I_2,$$

then for every $\varepsilon>0$, $b\in\text{Range}\,(A+E_\varepsilon)$ where $E_\varepsilon=\operatorname{diag}(0,\varepsilon)$. Thus, there is no "smallest" $\|[E\,|\,r]\|_F$ for which $b+r\in\text{Range}\,(A+E)$ since $b\notin\text{Range}\,(A)$. This kind of pathological situation raises several important questions. Under what set of circumstances does the TLS problem lack a solution? More generally, what constitutes an ill-conditioned TLS problem? Answers to these and other related theoretical questions of practical importance are offered in § 3 and § 4. In § 5 some algorithmic considerations are briefly mentioned.

2. The TLS problem and the singular value decomposition. If $b+r$ is in the range of $A+E$, then there is a vector $x \in R^n$ such that

$$(A+E)x = b+r;$$

i.e.,

$$\{D[A\,|\,b]T + D[E\,|\,r]T\}T^{-1}\begin{bmatrix} x \\ -1 \end{bmatrix} = 0. \tag{2.1}$$

This equation shows that the TLS problem involves finding a perturbation matrix $\Delta \in R^{mx(n+1)}$ having minimal norm such that $C+\Delta$ is rank deficient, where

$$C = D[A\,|\,b]T. \tag{2.2}$$

The singular value decomposition can be used for this purpose. Let

$$\begin{gathered} U^TCV = \operatorname{diag}(\sigma_1, \cdots, \sigma_{n+1}), \\ U = [u_1, \cdots, u_m], \qquad V = [v_1, \cdots, v_{n+1}], \qquad u_i \in R^m, v_i \in R^{n+1}, \\ \sigma_1 \geqq \cdots \geqq \sigma_k > \sigma_{k+1} = \cdots = \sigma_{n+1}, \end{gathered} \tag{2.3}$$

be the SVD of C with $U^TU = I_m$ and $V^TV = I_n$. A discussion of this decomposition and its elementary properties may be found in Stewart [17]. In particular, it can be shown that

$$\sigma_{n+1} = \min_{\operatorname{rank}(C+\Delta)<n+1} \|\Delta\|_F. \tag{2.4}$$

Moreover, the minimum is attained by setting $\Delta = -Cvv^T$, where v is any unit vector in the subspace S_C defined by

$$S_C = \operatorname{span}\{v_{k+1}, \cdots, v_{n+1}\}. \tag{2.5}$$

Suppose we can find a vector v in S_C having the following form:

$$v = \begin{bmatrix} y \\ \alpha \end{bmatrix}, \qquad y \in R^n, \quad \alpha \neq 0.$$

If

$$x = \frac{-1}{\alpha t_{n+1}} T_1 y, \qquad T_1 = \operatorname{diag}(t_1, \cdots, t_n), \tag{2.6}$$

and we define $\tilde{E}$ and $\tilde{r}$ by

$$D[\tilde{E}\,|\,\tilde{r}]T = -\mathring{C}vv^T,$$

then

$$\{D[A\,|\,b]T + D[\tilde{E}\,|\,\tilde{r}]T\}T^{-1}\begin{bmatrix} x \\ -1 \end{bmatrix} = C(I - vv^T)(-v/\alpha t_{n+1}) = 0.$$

In light of the remarks made after (2.1), it follows that x solves the TLS problem.

If $e_{n+1} = (0, \cdots, 0, 1)^T$ is orthogonal to S_C, then the TLS problem has no solution. On the other hand, if σ_{n+1} is a repeated singular value of C, then the TLS problem may lack a unique solution. However, whenever this is the case it is possible to single out a unique "minimum norm" TLS solution which we denote by x_{TLS}. In particular, let Q be an orthogonal matrix of order $n-k+1$ with the property that

$$[v_{k+1},\cdots,v_{n+1}]Q=\begin{array}{c}\left[\begin{array}{cc}W & y\\ 0 & \alpha\end{array}\right]\begin{array}{l}\}n-k\\ \}1\end{array}\\ \underbrace{}_{n-k}\;\underbrace{}_{1}\end{array}. \tag{2.7}$$

If we set $x_{\mathrm{TLS}}=-T_1y/(\alpha t_{n+1})$, and if we define the τ-norm by

$$\|w\|_\tau=\|T_1^{-1}w\|,\qquad w\in \boldsymbol{R}^n, \tag{2.8}$$

then it is easy to show that $\|x_{\mathrm{TLS}}\|_\tau<\|x\|_\tau$ for all other solutions x to the TLS problem (1.4).

3. A geometric interpretation of the TLS problem. If the SVD of $C=D[A|b]T$ is given by (2.3), then it is easy to verify that

$$\frac{\|D[A|b]Tv\|_2}{\|v\|_2}\geqq\sigma_{n+1},\qquad v\neq 0,$$

and that equality holds for nonzero v if and only if v is in the subspace S_C defined by (2.5). Combining this fact with (2.6), we see that the TLS problem amounts to finding an $x\in\boldsymbol{R}^n$ (if possible) such that

$$\frac{\left\|D[A|b]TT^{-1}\begin{bmatrix}x\\-1\end{bmatrix}\right\|_2}{\left\|T^{-1}\begin{bmatrix}x\\-1\end{bmatrix}\right\|_2}=\sigma_{n+1}.$$

The geometry of the TLS problem comes to light when we write

$$\frac{\left\|D[A|b]TT^{-1}\begin{bmatrix}x\\-1\end{bmatrix}\right\|_2^2}{\left\|T^{-1}\begin{bmatrix}x\\-1\end{bmatrix}\right\|_2^2}=\sum_{i=1}^m d_i^2\frac{|a_i^Tx-b_i|^2}{x^TT_1^{-2}x+t_{n+1}^{-2}}, \tag{3.1}$$

where $a_i^T=(a_{i1},\cdots,a_{in})$, the ith row of A. The quantity

$$\frac{|a_i^Tx-b_i|^2}{x^TT_1^{-2}x+t_{n+1}^{-2}}$$

is the square of the distance from $\left[\begin{smallmatrix}a_i\\b_i\end{smallmatrix}\right]\in\boldsymbol{R}^{n+1}$ to the nearest point in the subspace P_x defined by

$$P_x=\left\{\begin{bmatrix}a\\b\end{bmatrix}\middle|\, a\in\boldsymbol{R}^n,\ b\in\boldsymbol{R},\ b=x^Ta\right\}.$$

Here, the "distance" between two points u and v in $\boldsymbol{R}^{n+1}$ is given by $\|T(u-v)\|_2$.

Thus, the TLS problem is tantamount to finding a "closest" subspace P_x to the $(n+1)$-tuples $\left[\begin{smallmatrix}a_i\\b_i\end{smallmatrix}\right]$, $i=1,\cdots,m$. The simple case when $n=1$ and D and T are both identities is worth illustrating. In Fig. 1 the LS and the TLS measures of goodness-of-fit are depicted. In the LS problem it is the vertical distances that are important while in the TLS problem it is the perpendicular distances that are critical. (When $T\neq I$, these perpendiculars are "skewed".) To say that the TLS problem has no solution in the $n=1$ case is to say that the TLS fitting line is vertical. This would be the case, for example, if the three data points in Figure 1 are $(1, 8)$, $(2,-2)$, and $(4,-1)$, for then the line $a=\frac{7}{3}$ is closest to the data in the sense of minimizing the sum of the squared perpendicular distances.

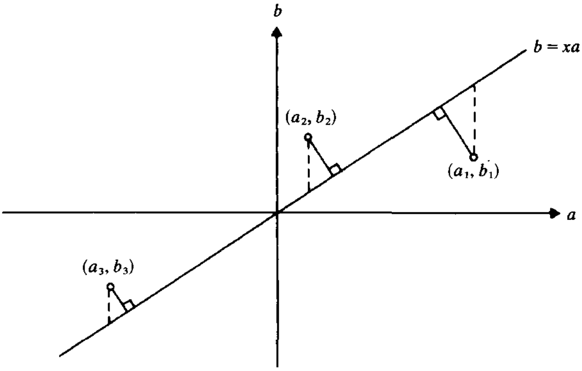


FIG. 1. *Least Squares versus Total Least Squares.*

The fitting of straight lines when both variables are subject to error has received a lot of attention in the statistics literature. We refer the interested reader to the papers by Pearson [15], Madansky [14], Riggs et al. [16], and York [22], as well as Chapter 13 of Linnik [13].

4. The sensitivity of the TLS problem. In this section we establish some inequalities that shed light on the sensitivity of the TLS problem as well as on the relationship between x_{LS} and x_{TLS}. The starting point in the analysis is to formulate the TLS problem as an eigenvalue problem. Recall the definitions of the matrix C and the subspace S_C in § 2. It is easy to show that the "singular vectors" v_i in (2.3) are eigenvectors of the C^TC, and that in particular, S_C is the invariant subspace associated with σ_{n+1}^2, the smallest eigenvalue of this matrix. Thus, if $x \in R^n$ is such that

$$C^TCT^{-1}\begin{bmatrix} x \\ -1 \end{bmatrix} = \sigma_{n+1}^2 T^{-1}\begin{bmatrix} x \\ -1 \end{bmatrix}, \tag{4.1}$$

then x solves the TLS problem. With the definitions

$$\hat{A} = DAT_1, \quad \hat{b} = Db, \quad \lambda = t_{n+1}, \tag{4.2}$$

(4.1) is readily seen to have the following block structure:

$$\begin{bmatrix} \hat{A}^T\hat{A} & \lambda\hat{A}^T\hat{b} \\ \lambda\hat{b}^T\hat{A} & \lambda^2\hat{b}^T\hat{b} \end{bmatrix}\begin{bmatrix} T_1^{-1}x \\ \lambda^{-1} \end{bmatrix} = \sigma_{n+1}^2\begin{bmatrix} T_1^{-1}x \\ -\lambda^{-1} \end{bmatrix}. \tag{4.3}$$

Moreover, if

$$\hat{U}^T\hat{A}\hat{V} = \hat{\Sigma} = \operatorname{diag}(\hat{\sigma}_1, \cdots, \hat{\sigma}_n), \qquad \hat{U}^T\hat{U} = I_m, \quad \hat{V}^T\hat{V} = I_n, \quad \hat{\sigma}_1 \geqq \hat{\sigma}_2 \geqq \cdots \geqq \hat{\sigma}_n \geqq 0, \tag{4.4}$$

is the SVD of $\hat{A}$, and if we define

$$K = \hat{\Sigma}^T\hat{\Sigma} = \operatorname{diag}(\hat{\sigma}_1^2, \cdots, \hat{\sigma}_n^2), \qquad g = \hat{\Sigma}^T\hat{U}^Tb, \quad h^2 = \hat{b}^T\hat{b}, \quad z = \hat{V}^TT_1^{-1}x, \tag{4.5}$$

then (4.3) transforms to

$$\begin{bmatrix} K & \lambda g \\ \lambda g^T & \lambda^2 h^2 \end{bmatrix}\begin{bmatrix} z \\ -\lambda^{-1} \end{bmatrix} = \sigma_{n+1}^2 \begin{bmatrix} z \\ -\lambda^{-1} \end{bmatrix}. \tag{4.6}$$

From this equation we see that

$$(K - \sigma_{n+1}^2 I)z = g \tag{4.7}$$

and

$$\frac{\sigma_{n+1}^2}{\lambda^2} + g^T z = h^2. \tag{4.8}$$

With these reductions, we now obtain some useful characterizations of both x_{TLS} and σ_{n+1}. In order for the subsequent analysis to be uncluttered, we freely make use of the notation established in (2.2)–(2.8) and (4.2)–(4.5).

THEOREM 4.1. *If $\hat{\sigma}_n > \sigma_{n+1}$, then x_{TLS} exists and is the only solution to the* TLS *problem. Moreover,*

$$x_{\text{TLS}} = T_1(\hat{A}^T\hat{A} - \sigma_{n+1}^2 I)^{-1}\hat{A}^T\hat{b}, \tag{4.9}$$

and

$$\sigma_{n+1}^2\left[\frac{1}{\lambda^2} + \sum_{i=1}^{n} \frac{c_i^2}{\hat{\sigma}_i^2 - \sigma_{n+1}^2}\right] = \rho_{\text{LS}}^2, \tag{4.10}$$

where

$$c = (c_1, \cdots, c_m)^T = \hat{U}^T\hat{b}, \tag{4.11}$$

$$\rho_{\text{LS}}^2 = \min \|D(b - Ax)\|_2^2 = \|D(b - Ax_{\text{LS}})\|_2^2. \tag{4.12}$$

Proof. The separation theorem [21, p. 103] for eigenvalues of symmetric matrices implies that

$$\sigma_1 \geqq \hat{\sigma}_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_n \geqq \hat{\sigma}_n \geqq \sigma_{n+1}. \tag{4.13}$$

The assumption $\hat{\sigma}_n > \sigma_{n+1}$ thus insures that σ_{n+1} is not a repeated singular value of C. If $C^TC\left[\begin{smallmatrix} y \\ 0 \end{smallmatrix}\right] = \sigma_{n+1}^2\left[\begin{smallmatrix} y \\ 0 \end{smallmatrix}\right]$ and $0 \neq y \in R^n$, then it clearly follows that $\hat{A}^T\hat{A}y = \sigma_{n+1}^2 y$, a contradiction since $\hat{\sigma}_n^2$ is the smallest eigenvalue of $\hat{A}^T\hat{A}$. Thus, S_c must contain a vector whose $(n+1)$st component is nonzero. This implies that TLS problem has a solution. Since S_C has dimension 1, this solution is unique. The formula (4.9) follows directly from the "top half" of (4.3).

To establish (4.10) we observe from (4.7) and (4.8) that

$$\frac{\sigma_{n+1}^2}{\lambda^2} + g^T(K - \sigma_{n+1}^2 I)^{-1} g = h^2.$$

By using the definitions (4.5) and (4.11) this can be rewritten as

$$\frac{\sigma_{n+1}^2}{\lambda^2} + \sum_{i=1}^{n} \frac{\hat{\sigma}_i^2 c_i^2}{\hat{\sigma}_i^2 - \sigma_{n+1}^2} = \sum_{i=1}^{m} c_i^2,$$

or

$$\sigma_{n+1}^2\left[\frac{1}{\lambda^2} + \sum_{i=1}^{n} \frac{c_i^2}{\hat{\sigma}_i^2 - \sigma_{n+1}^2}\right] = \sum_{i=n+1}^{m} c_i^2.$$

Inequality (4.10) now follows since

$$\min_x \|D(b-Ax)\|_2^2 = \min_y \|\hat{b}-\hat{A}y\|_2^2 = \min_w \|c-\hat{\Sigma}w\|_2^2 = \sum_{i=n+1}^{m} c_i^2. \qquad \square$$

We shall make use of (4.10) in the next section. The characterization (4.9) points out an interesting connection between total squares and ridge regression. Ridge regression is a way of "regularizing" the solution to an ill-conditioned LS problem. (See [12, pp. 190ff.].) Consider, for example, the minimization of

$$\phi(x;\mu) = \|D(b-Ax)\|_2^2 + \mu \, \|T_1^{-1}x\|_2^2,$$

where μ is a positive scalar. It is easy to show that

$$x_{\mathrm{LS}}(\mu) = T_1(\hat{A}^T\hat{A}+\mu I)^{-1}\hat{A}^T\hat{b}$$

solves this problem, and that $\|T_1^{-1}x_{\mathrm{LS}}(\mu)\|_2 = \|x_{\mathrm{LS}}(\mu)\|_\tau$ becomes small as μ becomes large. This is the key to ridge regression; by controlling μ we can control the τ-norm of $x_{\mathrm{LS}}(\mu)$.

What is particularly interesting, however, is that $x_{\mathrm{TLS}} = x_{\mathrm{LS}}(-\sigma_{n+1}^2)$. That is, total least squares is a *de*regularizing procedure, a kind of "reverse" ridge regression. As we shall see, this implies that the condition of the TLS problem is always worse than the condition of the corresponding LS problem. For this reason it is interesting to compare the LS and TLS fits with one another.

COROLLARY 4.2. *Let* $\rho_{\mathrm{LS}} = \|D(b-Ax_{\mathrm{LS}})\|_2$. *If* $\hat{\sigma}_n > \sigma_{n+1}$, *then*

$$\|x_{\mathrm{TLS}}-x_{\mathrm{LS}}\|_\tau \leqq \frac{\lambda\,\|\hat{b}\|_2\rho_{\mathrm{LS}}}{\hat{\sigma}_n^2-\sigma_{n+1}^2} \tag{4.14}$$

and

$$\|D(b-x_{\mathrm{TLS}})\|_2 \leqq \rho_{\mathrm{LS}}\left[1+\frac{\lambda\,\|\hat{b}\|_2}{\hat{\sigma}_n-\sigma_{n+1}}\right]. \tag{4.15}$$

Proof. From (1.2) it is clear that $x_{\mathrm{LS}} = T_1(\hat{A}^T\hat{A})^{-1}\hat{A}^T\hat{b}$ and so from (4.9) we have

$$\begin{aligned} x_{\mathrm{TLS}}-x_{\mathrm{LS}} &= T_1[(\hat{A}^T\hat{A}-\sigma_{n+1}^2 I)^{-1}-(\hat{A}^T\hat{A})^{-1}]\hat{A}^T\hat{b} \\ &= \sigma_{n+1}^2 T_1(\hat{A}^T\hat{A}-\sigma_{n+1}^2 I)^{-1}T_1^{-1}x_{\mathrm{LS}} \end{aligned} \tag{4.16}$$

Applying T_1^{-1} to both sides of this equation and taking norms gives

$$\|x_{\mathrm{TLS}}-x_{\mathrm{LS}}\|_\tau \leqq \frac{\sigma_{n+1}^2\,\|x_{\mathrm{LS}}\|_\tau}{\hat{\sigma}_n^2-\sigma_{n+1}^2}.$$

This result coupled with the inequalities

$$\rho_{\mathrm{LS}} = \|D(Ax_{\mathrm{LS}}-b)\|_2 = \left\| D[A\,|\,b]TT^{-1}\begin{bmatrix} x_{\mathrm{LS}} \\ -1 \end{bmatrix}\right\|_2 \geqq \sigma_{n+1}\|x_{\mathrm{LS}}\|_\tau, \tag{4.17}$$

$$\lambda\,\|\hat{b}\|_2 = \|D[A\,|\,b]Te_{n+1}\| \geqq \sigma_{n+1}, \qquad (e_{n+1}^T = (0,\cdots,0,1)), \tag{4.18}$$

establish (4.14).

To prove (4.15), note that

$$\|D(b-Ax_{\mathrm{TLS}})\|_2 \leqq \rho_{\mathrm{LS}} + \|DA(x_{\mathrm{TLS}}-x_{\mathrm{LS}})\|_2. \tag{4.19}$$

Now by (4.16),

$$DA(x_{\mathrm{TLS}}-x_{\mathrm{LS}}) = \sigma_{n+1}^2\hat{A}(\hat{A}^T\hat{A}-\sigma_{n+1}^2 I)^{-1}T_1^{-1}x_{\mathrm{LS}},$$

and so by invoking (4.17) and (4.18) we find

$$\begin{aligned}\|DA(x_{\text{TLS}}-x_{\text{LS}})\|_2 &\leqq \rho_{\text{LS}}\lambda\,\|\hat{b}\|_2\|\hat{A}(\hat{A}^T\hat{A}-\sigma_{n+1}^2)^{-1}\|_2\\ &=\rho_{\text{LS}}\lambda\,\|\hat{b}\|_2 \max_{1\leqq k\leqq n}\frac{\hat{\sigma}_k}{\hat{\sigma}_k+\sigma_{n+1}}\cdot\frac{1}{\hat{\sigma}_k-\sigma_{n+1}}\\ &\leqq \rho_{\text{LS}}\lambda\,\|\hat{b}\|_2/(\hat{\sigma}_n-\sigma_{n+1})\end{aligned}$$

Inequality (4.15) follows by substituting this result into (4.19). □

The corollary shows that $x_{\text{TLS}}\to x_{\text{LS}}$ as $\lambda\downarrow 0$. Thus, by reducing the "observation weight" $\lambda = t_{n+1}$, the TLS problem "converges" to the LS problem. Of course, if $\rho_{\text{LS}}=0$ and A has full rank, then $x_{\text{TLS}}=x_{\text{LS}}$ regardless of λ.

The bounds in (4.14) and (4.15) are large whenever σ_{n+1} is close to $\hat{\sigma}_n$. (This occurs, for example, whenever σ_{n+1} is a nearly repeated singular value.) Our next results indicate the extent to which $(\hat{\sigma}_n-\sigma_{n+1})^{-1}$ measures the sensitivity of the TLS problem.

LEMMA 4.3. *If $\hat{U}=[\hat{u}_1,\cdots,\hat{u}_m]$ is a column partitioning of the matrix $\hat{U}$ in the* SVD (4.4) *and if $\hat{\sigma}_n>\sigma_{n+1}$, then*

$$\frac{|\hat{u}_n^T\hat{b}|}{2(\hat{\sigma}_n-\sigma_{n+1})}\leqq\|x_{\text{TLS}}\|_\tau\leqq\frac{\|\hat{b}\|_2}{\hat{\sigma}_n-\sigma_{n+1}}.$$

Proof. Substituting the SVD (4.4) into (4.9) and taking the τ-norm of both sides gives

$$\|x_{\text{TLS}}\|_\tau^2=\sum_{i=1}^n\left[\frac{\hat{\sigma}_i}{(\hat{\sigma}_i+\sigma_{n+1})}\frac{\hat{u}_i^T\hat{b}}{(\hat{\sigma}_i-\sigma_{n+1})}\right]^2.$$

The lemma follows from the inequalities $\frac{1}{2}\leqq\hat{\sigma}_i/(\hat{\sigma}_i+\sigma_{n+1})\leqq 1$ □

THEOREM 4.4. *If $A'\in R^{m\times n}$ and $b'\in R^m$ are such that*

$$\eta=\|D[A'-A\,|\,b'-b]T\|_F\leqq\frac{\varepsilon}{6},$$

where

$$\varepsilon=\hat{\sigma}_n-\sigma_{n+1}>0,$$

then the perturbed TLS *problem*

$$\text{(4.20)}\qquad\begin{aligned}&\underset{E,r}{\text{minimize}}\ \|D[E|r]T\|_F\\ &\text{subject to } b'+r\in\text{Range}\,(A'+E)\end{aligned}$$

has a unique solution x'_{TLS}. Moreover, if $x_{\text{TLS}}\neq 0$, then

$$\text{(4.21)}\qquad\frac{\|x_{\text{TLS}}-x'_{\text{TLS}}\|_\tau}{\|x_{\text{TLS}}\|_\tau}\leqq\frac{9\eta\sigma_1}{\sigma_n-\sigma_{n+1}}\left\{1+\frac{\lambda\,\|\hat{b}\|}{\hat{\sigma}_n-\sigma_{n+1}}\right\}\frac{1}{\|\lambda\hat{b}\|_2-\sigma_{n+1}}.$$

Proof. Denote the singular values of the matrices $\hat{A}'=DA'T$ and $C'=D[A'\,|\,b']T$ by $\hat{\sigma}'_1\geqq\cdots\geqq\hat{\sigma}'_n$ and $\sigma'_1\geqq\cdots\geqq\sigma'_{n+1}$ respectively. Well-known perturbation results for singular values ensure that

$$\text{(4.22)}\qquad|\hat{\sigma}'_n-\sigma'_{n+1}|\geqq|\hat{\sigma}_n-\sigma_{n+1}|-|\hat{\sigma}'_n-\hat{\sigma}_n|-|\sigma'_{n+1}-\sigma_{n+1}|\geqq\varepsilon-\frac{\varepsilon}{6}-\frac{\varepsilon}{6}=\frac{2}{3}\varepsilon.$$

In view of Theorem 4.1, this implies that the perturbed TLS problem above has a unique solution x'_{TLS}.

Let $\left[\begin{smallmatrix} y \\ \alpha \end{smallmatrix}\right] (y \in \boldsymbol{R}^n, \alpha \in \boldsymbol{R})$ be a unit right singular vector of C associated with σ_{n+1}. Using the SVD perturbation theory of Stewart [18], it is possible to bound the difference between $\left[\begin{smallmatrix} y \\ \alpha \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} z \\ \beta \end{smallmatrix}\right]$, a corresponding singular vector of C′ associated with σ'_{n+1}. Not surprisingly, the bound involves the separation of σ_n and σ_{n+1}:

$$\left\| \begin{bmatrix} y \\ \alpha \end{bmatrix} - \begin{bmatrix} z \\ \beta \end{bmatrix} \right\|_2 \leqq \frac{3\eta}{\sigma_n - \sigma_{n+1}}.$$

Now from § 2 we have $x_{\text{TLS}} = -T_1 y/(\lambda\alpha)$ and $x'_{\text{TLS}} = -T_1 z/(\lambda\beta)$, where $\lambda = t_{n+1}$ and $T_1 = \text{diag}(t_1, \cdots, t_n)$. Thus,

$$\|x_{\text{TLS}} - x'_{\text{TLS}}\|_\tau = \frac{1}{\lambda}\|(y/\alpha) - (z/\beta)\|_2 \leqq \frac{1}{|\lambda\alpha|}\left\{\|y - z\|_2 + \frac{\|z\|_2}{|\beta|}|\alpha - \beta|\right\},$$

and so

$$\|x_{\text{TLS}} - x'_{\text{TLS}}\|_\tau \leqq \frac{3\eta}{\sigma_n - \sigma_{n+1}} \frac{\|x_{\text{TLS}}\|_\tau}{\|y\|_2}[1 + \lambda\|x'_{\text{TLS}}\|_\tau].$$

Set $\hat{b}' = Db'$. From Lemma 4.3, (4.22), and the fact that $\|\lambda(\hat{b} - \hat{b}')\|_2 \leqq \varepsilon/6$, we have

$$\|x'_{TLS}\|_\tau \leqq \frac{\|\hat{b}'\|_2}{\hat{\sigma}_n - \sigma_{n+1}} \leqq \frac{3}{2}\left\{\frac{\|\hat{b}\|_2}{\hat{\sigma}_n - \sigma_{n+1}} + \frac{1}{6\lambda}\right\},$$

and so

$$\frac{\|x_{\text{TLS}} - x'_{\text{TLS}}\|_\tau}{\|x_{\text{TLS}}\|_\tau} \leqq \frac{3\eta}{\sigma_n - \sigma_{n+1}}\left\{\frac{5}{4} + \frac{3}{2}\cdot\frac{\lambda\|\hat{b}\|}{\hat{\sigma}_n - \sigma_{n+1}}\right\}\frac{1}{\|y\|_2}. \tag{4.23}$$

In order to get a lower bound on $\|y\|_2$, observe that

$$\lambda|\alpha|\,\|\hat{b}\|_2 \leqq \left\|[\hat{A}\,|\,\lambda\hat{b}]\begin{bmatrix} y \\ \alpha \end{bmatrix}\right\|_2 + \|\hat{A}y\|_2 \leqq \sigma_{n+1} + \|\hat{A}\|_2\|y\|_2,$$

i.e.,

$$\begin{aligned} \lambda\|\hat{b}\|_2 - \sigma_{n+1} &\leqq (1 - |\alpha|)\lambda\|\hat{b}\|_2 + \|\hat{A}\|_2\|y\|_2 \\ &\leqq \|y\|_2[\lambda\|\hat{b}\|_2 + \|\hat{A}\|_2] \leqq 2\|y\|_2\|C\|_2. \end{aligned}$$

The assumption that $x_{\text{TLS}} \neq 0$ implies that $\lambda\|\hat{b}\|_2 > \sigma_{n+1}$ for otherwise $\left[\begin{smallmatrix} y \\ \alpha \end{smallmatrix}\right] = \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$ is a singular vector of C. The theorem now follows because

$$\frac{1}{\|y\|_2} \leqq \frac{2\sigma_1}{\|\lambda\hat{b}\|_2 - \sigma_{n+1}}. \qquad \square$$

Both the lemma and the theorem suggest that the TLS problem is unstable whenever $\hat{\sigma}_n$ is close to σ_{n+1}. This is borne out by some results established in [23] where it is shown that a change of order $|\alpha|$ in C can result in an insoluble TLS problem. Using Lemma 4.3, this translates into the assertion that $\hat{\sigma}_n - \sigma_{n+1}$ is a measure of how close (1.4) is to the class of insoluble TLS problems.

Finally, we remark that if the LS problem is ill-conditioned, i.e., $\hat{\sigma}_n$ is small, then the TLS problem is likewise sensitive.

5. Algorithmic considerations. Although a stable and efficient algorithm for computing the SVD exists [7], there are numerical difficulties associated with the determination of the dimension of S_C, i.e., the multiplicity of σ_{n+1}. One approach is to regard all computed singular values in the interval $[\sigma_{n+1}, \sigma_{n+1} + \varepsilon]$ as being identical, where $\varepsilon > 0$ is some small machine-dependent parameter. This leads to the following overall procedure for computing the solution to the TLS problem:

1. Compute the SVD $U^T(D[A|b]T)V = \operatorname{diag}(\sigma_1, \cdots, \sigma_{n+1})$. Accumulate V.
2. Define the index p by $\sigma_p > \sigma_{n+1} + \varepsilon \geqq \sigma_{p+1} \geqq \cdots \geqq \sigma_{n+1}$.
3. Let $V = [v_1, \cdots, v_n]$ be a column partition of V, and compute a Householder matrix Q such that

$$[v_{p+1}, \cdots, v_{n+1}]Q = \left[\begin{array}{c|c} \times & y \\ \hline 0 \cdots 0 & \alpha \end{array}\right].$$

4. If $\alpha = 0$, then the TLS problem has no solution. Otherwise, $x_{\mathrm{TLS}} = -T_1 y/(\alpha t_{n+1})$.

A shortcoming of this scheme is that it does not compute $\hat{\sigma}_n - \sigma_{n+1}$, which as we have seen, is a measure of TLS sensitivity. To rectify this it may be more desirable to compute the SVD $\hat{U}^T\hat{A}\hat{V} = \operatorname{diag}(\hat{\sigma}_1, \cdots, \hat{\sigma}_n) = \hat{\Sigma}$ and then make use of the TLS "secular equation":

$$\Psi(\sigma) = \sigma^2\left[\frac{1}{\lambda^2} + \sum_{i=1}^{n} \frac{c_i^2}{\hat{\sigma}_i^2 - \sigma^2}\right] = \rho_{\mathrm{LS}}^2.$$

In view of (4.9) and (4.11), if a σ can be found that satisfies this equation and is less than $\hat{\sigma}_n$, then

$$x_{\mathrm{TLS}} = T_1\hat{V}(\hat{\Sigma}^T\hat{\Sigma} - \sigma^2 I)^{-1}\hat{\Sigma}^T\hat{U}^T\hat{b}.$$

Standard root-finding techniques can be used for this purpose. (The function Ψ has monotonicity properties in the bracketing interval $[0, \hat{\sigma}_n]$.) Notice how easy it is to compute the TLS solution for different values of the weight $\lambda = t_{n+1}$. A detailed discussion of these and other algorithmic aspects of the TLS problem, such as the choosing of the weights, will appear elsewhere.

Acknowledgments. We are grateful to the following people for calling our attention to various aspects of the TLS problem: A. Bjork, R. Byers, P. Diaconis, C. Moler, C. Paige, C. Reinsch, B. Rust, P. Velleman, and J. H. Wilkinson.

Note added in proof. In collaboration with R. Byers, a graduate student at Cornell, the authors have recently shown how to solve the TLS problem for the case when some of the columns of A are known exactly. The technique involves (a) computing a QR factorization of the "known" columns and (b) solving a TLS problem of reduced dimension. This work will be described elsewhere.

REFERENCES

[1] P. BARRERA AND J. E. DENNIS, *Fuzzy Broyden Methods*, private communication, 1979.

[2]•W. G. COCHRANE, *Errors of measurement in statistics*, Technometrics, 10 (1968), pp. 637–666.

[3] W. E. DEMING, *Statistical Adjustment of Data*, John Wiley, New York, 1946.

[4] G. A. GERHOLD, *Least squares adjustment of weighted data to a general linear equation*, Amer. J. Phys., 37 (1969), pp. 156–161.

[5] P. E. GILL AND W. MURRAY, *Computation of Lagrange multiplier estimates for constrained minimization*, Math. Programming, 17 (1979), pp. 32–60.

[6] G. H. GOLUB, *Some modified eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–344.

[7] G. H. GOLUB AND C. REINSCH, *Singular value decomposition and least squares solutions*, Numer. Math., 14 (1970), pp. 403–420.

[8] G. H. GOLUB AND C. VAN LOAN, *Total least squares*, in Smoothing Techniques for Curve Estimation, T. Gasser and M. Rosenblatt, eds., Springer-Verlag, New York, 1979, pp. 69–76.

[9] R. F. GUNST, J. T. WEBSTER, AND R. L. MASON, *A comparison of least squares and latent root regression estimators*, Technometrics, 18 (1976), pp. 75–83.

[10] D. M. HAWKINS, *On the investigation of alternative regressions by principal component analysis*, Applied Statistics, 22 (1973), pp. 275–286.

[11] S. D. HODGES AND P. G. MOORE, *Data uncertainties and least squares regression*, Applied Statistics, 21 (1972), pp. 185–195.

[12] C. LAWSON AND R. HANSON, *Solving Least Squares Problems*, Prentice Hall, Englewood Cliffs, NJ, 1974.

[13] I. LINNIK, *Method of Least Squares and Principles of the Theory of Observations*, Pergamon Press, New York, 1961.

[14] A. MADANSKY, *The fitting of straight lines when both variables are subject to error*, J. Amer. Statist. Assoc., 54 (1959), pp. 173–205.

[15] K. PEARSON, *On lines and planes of closest fit to points in space*, Phil. Mag., 2 (1901), pp. 559–572.

[16] D. RIGGS, J. GUARNIERI, AND S. ADELMAN, *Fitting straight lines when both variables are subject to error*, Life Sciences, 22 (1978), pp. 1305–1360.

[17] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[18] ———, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Rev., 15 (1973), pp. 727–764.

[19] ———, *Sensitivity Coefficients for the Effects of Errors in the Independent Variables in a Linear Regression*, Technical Report TR-571, Department of Computer Science, University of Maryland, College Park MD, 1977.

[20] J. T. WEBSTER, R. F. GUNST, AND R. L. MASON, *Latent root regression analysis*, Technometrics, 16 (1974), pp. 513–522.

[21] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.

[22] D. YORK, *Least squares fitting of a straight line*, Canad. J. Physics, 44 (1966), 1079–1086.

[23] C. VAN LOAN, *On Stewart's Singular Value Decomposition for Partitioned Orthogonal Matrices*, Department of Computer Science Report STAN-CS-79-767, Stanford University, Stanford CA, 1979.

PART IV

MATRIX FACTORIZATIONS AND APPLICATIONS

This page intentionally left blank

16

COMMENTARY, BY NICHOLAS HIGHAM

One of the fundamental tenets of numerical linear algebra is to exploit matrix factorizations. Doing so has numerous benefits, ranging from allowing clearer analysis and deeper understanding to simplifying the efficient implementation of algorithms. Textbooks in numerical analysis and matrix analysis nowadays maximize the use of matrix factorizations, but this was not so in the first half of the 20th century. Golub has done as much as anyone to promulgate the benefits of matrix factorization, particularly the QR factorization and the singular value decomposition, and especially through his book *Matrix Computations* with Van Loan [28]. The five papers in this part illustrate several different facets of the matrix factorization paradigm.

On direct methods for solving Poisson's equations, by Buzbee, Golub, and Nielson [9]

Cyclic reduction is a recurring topic in numerical analysis. In the context of solving a tridiagonal linear system of order $2^n - 1$, the idea is to eliminate the odd-numbered unknowns, thus halving the size of the system, and to continue this procedure recursively until a single equation remains. One unknown can now be solved for and the rest are obtained by substitution. Cyclic odd–even reduction – to give it its full name – is thus a particular instance of the divide and conquer principle. The method was derived by Golub when he was a PhD student, but it became widely known only through Hockney's paper on solving the Poisson equation [32], in which he acknowledges the help of Golub.

This paper presents cyclic reduction for block tridiagonal systems arising in the discretization of Poisson's equation on a rectangle or L-shaped region with Dirichlet, Neumann, or periodic boundary conditions. The required properties of the coefficient matrix are essentially that it be symmetric block tridiagonal and block Toeplitz, with commuting blocks. Two variants of cyclic reduction are described. The first, more straightforward one (Section 3), is found to be "virtually useless" for practical computation, because of numerical instability, the nature of which is analyzed in Section 10. The second variant, suggested by Buneman, is shown to be mathematically equivalent to the first (Section 11), and to be numerically stable (Section 13).

At the time this work was done, iterative methods were prevalent for solving linear systems arising in the discretization of partial differential equations. The development of cyclic reduction and of methods exploiting the fast Fourier transform represented a shift back to direct methods and created the research topic of “fast Poisson solvers”. Indeed Hockney [32] was able to solve Poisson's equation on a 48×48 mesh in 0.9 seconds on an IBM 7090, claiming a factor of 10 improvement in speed over the best iterative solvers of the time.

A flurry of papers followed this one, generalizing the method to irregular regions and to different equations. See Swartztrauber [43] for a list of references. Swartztrauber also gives a detailed analysis of the FACR(ℓ) method. This method, originally suggested by Hockney, carries out ℓ steps of cyclic reduction and then uses the fast Fourier transform to diagonalize the blocks of the reduced matrix (exploiting the fact that the blocks have the Fourier matrix as their eigenvector matrix), thereby producing a number of independent tridiagonal systems. He shows that by a suitable choice of ℓ, an operation count of $O(mn \log\log n)$ can be achieved, where the original matrix is block $n \times n$ tridiagonal with $m \times m$ blocks. This operation count beats the $O(mn \log n)$ count for cyclic reduction itself and is close to linear in the dimension of the matrix.

Cyclic reduction for a tridiagonal system $Ax = b$ is equivalent to Gaussian elimination on $(PAP^T)Px = Pb$, where P is a certain permutation matrix [31]. This connection has proved useful in some more recent error analyses of the method [1], [45].

Interest in cyclic reduction was rekindled by the advent of parallel computers, since the method produces smaller, independent “eliminated” systems that can be solved in parallel [33]. More recently, cyclic reduction has been applied to infinite block tridiagonal linear systems arising in the numerical solution of Markov chains, in which context it is called logarithmic reduction; for details and references see Bini, Latouche, and Meini [6, Chapter 7].

This is one of Golub's most highly cited papers. In 1992 it was deemed a Citation Classic by the Institute for Scientific Information, who collate the Science Citation Index. Buzbee wrote an article explaining the background to the paper [10]. The collaboration began when Golub visited Los Alamos National Laboratory and the three authors tried to understand a program written by Buneman for solving the Poisson equation “at a speed and accuracy that far exceeded established techniques such as relaxation and alternating directions”. Understanding the numerical stability properties of the two variants of block cyclic reduction, and showing that they were mathematically equivalent, took some time. Buzbee notes that “Over a period of about 18 months, with no small amount of mathematical sleuthhounding, we completed this now-*Classic* paper. During that 18 months, we were tempted on several occasions to publish intermediate results. However, we continued to hold out for a full understanding, and, in the end, we were especially pleased that we waited until we had a comprehensive report.”

The simplex method of linear programming using LU decomposition, by Bartels and Golub [4]

In the simplex method for linear programming, the basis matrix changes one column at a time, and some linear systems involving each new basis matrix are solved to determine the next basis change. By 1957, implementations were using sparse LU factors of an initial basis B_0 [35] and updating the factors a number of times before factorizing the current basis directly. The updates were in product form: $B_0 = LU$, $B_k = LUE_1E_2 \dots E_k$, where each E_k differs from the identity in only one column. (Most authors thought they were updating the *inverse* of each basis matrix, but the updates they used involved the same numbers as in the product form given here.) Bartels and Golub recognized that the product-form update is potentially unstable. They focused the linear programming community on the importance of maintaining numerical stability and the possibility of achieving it by updating the LU factors directly. Their method replaces the old column by the new column, effectively creating a "column spike" in the U factor, then cyclically permutes the spike to make it the last column of U, which becomes upper Hessenberg. Row operations, with row interchanges for stability, are performed to restore the upper triangular form. The L factor is updated in product form, but U is maintained as an explicit triangle.

Various LU updates were proposed by subsequent authors, most of whom were concerned with sparse problems. Forrest and Tomlin [15] store U column-wise and update it by deleting a column and row of U and adding a new column at the end, while generating a triangular matrix E_j^T (differing from the identity in only one row) to update L in product-form. This was soon recognized to be equivalent to the Bartels–Golub update but without any row interchanges for stability. (See Nocedal and Wright [37, Section 13.4] for an illustrated comparison of the methods.) Reid [40] stores U row-wise and applies the cyclic permutation to both the rows and columns of U, creating a "row spike" that tends to be sparse. Row operations are applied with row interchanges for stability, thus achieving a sparse form of the Bartels–Golub update with reasonable efficiency. Fletcher and Matthews [14] show how to update explicit LU factors in a stable way (keeping both L and U triangular), but sparsity cannot be preserved. Sparse forms of the Bartels–Golub update are in use today in Reid's code LA15 [39] and in LUSOL [19] (and hence in MINOS [36] and SNOPT [20]), but commercial linear programming codes flirt with instability by using the Forrest–Tomlin update because of an overriding desire for speed. Fortunately the Bartels–Golub viewpoint suggests a way to avoid significant instability: test if any element of E_j above is large, and if so refactorize rather than update.

A curious feature of the Bartels and Golub paper is that it does not describe the LU factorization updating that is proposed. Details of that are given in an earlier paper of the same authors on Chebyshev solution of overdetermined systems (the first paper in their reference list) and by Bartels [2], who gives a detailed error analysis of it. The present paper accompanies CACM Algorithm 250, an Algol 60 code that appears a few pages later in the same issue of the journal [3]. The Bartels–Golub method has comparable numerical stability properties to Gaussian elimination with partial pivoting, and so can in rare cases suffer from exponential error growth. A detailed analysis of this phenomenon is given by Powell [38], who constructs pathological examples for both the Bartels–Golub and the Fletcher–Matthews approaches.

Calculating the singular values and pseudo-inverse of a matrix, by Golub and Kahan [22]

The singular value decomposition (SVD) is today so widely used and ubiquitous that it is hard to imagine the computational scene in the early 1960s, when the SVD was relatively little known, its utility for solving a wide range of problems was unrecognized, and no satisfactory way of computing it was available. This paper helped bring the SVD to prominence by explaining two of its major applications, pointing out pitfalls in the more obvious methods of computation, and developing a strategy for a numerically stable SVD algorithm.

The paper begins by recalling the role of the pseudo-inverse in providing the minimum-norm solution to a linear least squares problem. The necessity and the difficulty of determining the rank when the matrix is (nearly) rank deficient are then pointed out, by arguments and examples that are now standard, but were only just beginning to be recognized. The use of the SVD to determine the pseudo-inverse and thereby to solve the rank-deficient least squares problem is advocated, along with the now standard device, motivated by the Eckart–Young theorem, of setting to zero any computed singular values that are negligible.

The paper makes two main algorithmic contributions. The first is the algorithm for bidiagonalizing a matrix by Householder transformations, given in Section 2. This provides a numerically stable reduction of the SVD problem to that of computing the SVD of a bidiagonal matrix. The second contribution builds on an observation of Lanczos [34, Section 3.7] that the eigenvalues of $C = \left[\begin{smallmatrix} 0 & A \\ A^* & 0 \end{smallmatrix}\right]$ are plus and minus the singular values of A. With A an upper bidiagonal matrix, Golub and Kahan note that C can be symmetrically permuted into a tridiagonal matrix by the operation (in MATLAB notation) $C(p, p)$, where $p = [n+1,\ 1,\ n+2,\ 2,\ \ldots,\ 2n,\ n]$. The available machinery for solving the symmetric tridiagonal eigenvalue problem is then applicable, and the last part of Section 3, and Section 4, concentrate on describing and adapting appropriate techniques (Sturm sequences and deflation). Here is where there is a surprise: the paper makes no mention at all of the QR algorithm. Yet a number of papers and textbooks state that this paper derives the complete SVD algorithm! In fact, how to adapt the Francis QR algorithm [16], [17] to the bidiagonal SVD was shown later by Golub in a less easily accessed paper [21], and the fine details were published in a Fortran code [8] and subsequently in [25].

Along with the Householder reduction to bidiagonal form, the use of the Lanczos method is mentioned at the end of Section 2, anticipating later work such as that in [23], where the Lanczos method would be developed for large-scale SVD computations.

Numerical methods for computing angles between linear subspaces, by Björck and Golub [7]

Björck and Golub's paper treats the problem of computing the principal angles between subspaces, along with associated principal vectors. This work is particularly aimed at the case where the subspaces are the ranges of given rectangular matrices. The key observation (Theorem 1) is that the cosines of the angles and the principal vectors are the singular values and transformed singular vectors of the matrix product $Q_1^* Q_2$, where the columns of Q_i form an orthonormal basis for the ith subspace. The authors propose computing the required orthonormal bases by Householder QR factorization or modified Gram–Schmidt.

The cosines of the angles are called canonical correlations in the statistics literature, and prior to this work they were computed via a "normal equations" eigenproblem. This paper is illustrative of the theme in Golub's work that orthogonalization methods should generally preferred because of their excellent numerical stability properties.

Björck and Golub go on to analyze the sensitivity of the problem, obtaining a perturbation bound for the errors in the angles that is proportional to the sum of the condition numbers of the two rectangular matrices. Armed with this information, they show via rounding error analysis that the numerical method produces results with forward error consistent with the perturbation bound. Special attention is given to the computation of small angles, since obtaining them via their cosines is not numerically reliable. The case of rank deficient matrices is also analyzed.

This is a remarkably thorough paper giving a beautiful, and relatively early, example of the power of the QR factorization and the SVD. It has hardly aged in the more than 30 years since publication. Only a few subsequent papers have attempted to improve upon or extend it. Golub and Zha [29] give a more detailed perturbation analysis, while in [30] they discuss equivalent characterizations of the principal angles and algorithms for large sparse or structured matrices. Drmač [13] shows that the Björck–Golub algorithm is mixed forward–backward stable: the computed singular values approximate with small relative error the exact cosines of the principal angles between the ranges of slight perturbations of the original two matrices.

Methods for modifying matrix factorizations, by Gill, Golub, Murray, and Saunders [18]

This paper treats the updating of the Cholesky factorization and the complete orthogonal decomposition after a rank-1 change, as well as (in the last section) updating of a QR factorization after the addition or deletion of rows or columns. Such changes are common in many applications, including in data fitting and signal processing problems, in which the data may be generated in real time. This paper was the first to emphasize the use of orthogonal transformations – particularly Givens transformations, which it uses extensively – and to give careful attention to the numerical stability of updating formulae. As such it has been very influential, as indicated by its being in Golub's top 10 most cited papers.

For Cholesky updating the rank-1 perturbation is allowed to be positive or negative semidefinite, but the perturbed matrix is assumed still to be positive definite. The case of a negative semidefinite perturbation can be tricky numerically, and in addition to the five different methods given here other methods have subsequently been proposed and analyzed, including ones based on hyperbolic transformations, which originate in an observation of Golub [26]. For an overview, see Stewart [41, Section 4.3].

It is tempting for the reader to skip over Section 2 of the paper, thinking that this is now standard material. However, Lemma 1 is not well known and new applications of it have recently been discovered. The lemma essentially says that the orthogonal QR factor of a tridiagonal matrix T has semiseparable structure – a property that can be exploited in computing trace(T^{-1}), for example [5].

Summary

This section has reviewed just five of Golub's contributions on matrix factorizations. Others include [11], [12], [24], [27], [44]. Golub and Van Loan [28] remains a standard reference for all aspects of matrix factorizations, including for what Stewart [42] calls the "big six" matrix factorizations.

REFERENCES

1. Pierluigi Amodio and Francesca Mazzia. Backward error analysis of cyclic reduction for the solution of tridiagonal systems. *Math. Comp.*, **62**(206), 601–617 (1994).
2. Richard H. Bartels. A stabilization of the simplex method. *Numer. Math.*, **16**, 414–434 (1971).
3. Richard H. Bartels and Gene H. Golub. Algorithm 350. Simplex method procedure employing LU decomposition. *Comm. ACM*, **12**(5), 275–278 (1969).

4*. Richard H. Bartels and Gene H. Golub. The simplex method of linear programming using LU decomposition. *Comm. ACM*, **12**(5), 266–268 (1969).

5. Dario A. Bini, Luca Gemignani, and Françoise Tisseur. The Ehrlich-Aberth method for the nonsymmetric tridiagonal eigenvalue problem. *SIAM J. Matrix Anal. Appl.*, **27**(1), 153–175 (2005).
6. Dario A. Bini, Guy Latouche, and Beatrice Meini. *Numerical Methods for Structured Markov Chains*. Oxford University Press (2005).

7*. Åke Björck and Gene H. Golub. Numerical methods for computing angles between linear subspaces. *Math. Comp.*, **27**(123), 579–594 (1973).

8. Peter A. Businger and Gene H. Golub. Algorithm 358: Singular value decomposition of a complex matrix. *Comm. ACM*, **12**(10), 564–565 (1969).

9*. B. L. Buzbee, G. H. Golub, and C. W. Nielson. On direct methods for solving Poisson's equations. *SIAM J. Numer. Anal.*, **7**(4), 627–656 (1970).

10. Bill Buzbee. Poisson's equation revisited. *Current Contents*, **36**, 8 (1992). Available from http://garfield.library.upenn.edu/classics1992/classics1992.html
11. Moody T. Chu, Robert E. Funderlic, and Gene H. Golub. A rank-one reduction formula and its applications to matrix factorizations. *SIAM Rev.*, **37**(4), 512–530 (1995).
12. Bart L. R. De Moor and Gene H. Golub. The restricted singular value decomposition: Properties and applications. *SIAM J. Matrix Anal. Appl.*, **12**(3), 401–425 (1991).
13. Zlatko Drmač. On principal angles between subspaces of Euclidean space. *SIAM J. Matrix Anal. Appl.*, **22**(1), 173–194 (2000).
14. R. Fletcher and S. P. J. Matthews. Stable modification of explicit LU factors for simplex updates. *Math. Programming*, **30**, 267–284 (1984).
15. J. J. H. Forrest and J. A. Tomlin. Updated triangular factors of the basis to maintain sparsity in the product form Simplex method. *Math. Programming*, **2**, 263–278 (1972).
16. J. G. F. Francis. The QR transformation: A unitary analogue to the LR transformation – part 1. *Comput. J.*, **4**, 265–271 (1961).

17. J. G. F. Francis. The QR transformation – part 2. *Comput. J.*, **4**, 332–345 (1962).

18*. P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Math. Comp.*, **28**(126), 505–535 (1974).

19. P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Maintaining *LU* factors of a general sparse matrix. *Linear Algebra Appl.*, **88/89**, 239–270 (1987).

20. Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Rev.*, **47**(1), 99–131 (2005).

21. G. H. Golub. Least squares, singular values and matrix approximations. *Aplikace Matematiky*, **13**, 44–51 (1968).

22*. G. H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal.*, **2**(2), 205–224 (1965).

23. G. H. Golub, F. T. Luk, and M. L. Overton. A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Trans. Math. Software*, **7**, 149–169 (1981).

24. G. H. Golub, S. Nash, and C. F. Van Loan. A Hessenberg–Schur method for the problem $AX + XB = C$. *IEEE Trans. Automat. Control*, **AC-24**(6), 909–913 (1979).

25*. G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numer. Math.*, **14**, 403–420 (1970).

26. Gene H. Golub. Matrix decompositions and statistical calculations. In Roy C. Milton and John A. Nelder, editors, *Statistical Computation*, Academic Press, New York, pp. 365–397 (1969).

27. Gene H. Golub and Charles F. Van Loan. Unsymmetric positive definite linear systems. *Linear Algebra Appl.*, **28**, 85–97 (1979).

28. Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, USA, 3rd edn. (1996).

29. Gene H. Golub and Hongyuan Zha. Perturbation analysis of the canonical correlations of matrix pairs. *Linear Algebra Appl.*, **210**, 3–28 (1994).

30. Gene H. Golub and Hongyuan Zha. The canonical correlations of matrix pairs and their numerical computation. In Adam W. Bojanczyk and George Cybenko, editors, *Linear Algebra for Signal Processing*, volume 69 of *IMA Volumes in Mathematics and Its Applications*, Springer-Verlag, New York, pp. 27–49 (1995).

31. Don Heller. Some aspects of the cyclic reduction algorithm for block tridiagonal linear systems. *SIAM J. Numer. Anal.*, **13**(4), 484–496 (1976).

32. R. W. Hockney. A fast direct solution of Poisson's equation using Fourier analysis. *J. Assoc. Comput. Mach.*, **12**(1), 95–113 (1965).

33. R. W. Hockney and C. R. Jesshope. *Parallel Computers 2: Architecture, Programming and Algorithms.* Adam Hilger, Bristol (1988).

34. Cornelius Lanczos. *Linear Differential Operators.* Van Nostrand, New York, 1961. Reprinted by Dover, New York (1997).

35. Harry M. Markowitz. The elimination form of the inverse and its application to linear programming. *Management Science*, **3**(3), 255–269 (1957).
36. B. A. Murtagh and M. A. Saunders. MINOS 5.5 user's guide. Technical Report SOL 83-20R (revised), Department of Operations Research, Stanford University, Stanford, CA (1998).
37. Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer-Verlag, New York (1999).
38. M. J. D. Powell. On error growth in the Bartels–Golub and Fletcher–Matthews algorithms for updating matrix factorizations. *Linear Algebra Appl.*, **88/89**, 597–622 (1987).
39. J. K. Reid. LA15. Sparse mathematical programming bases: Factorize and update. HSL 2004 Catalogue, http://www.aspentech.com/hsl/hslnav/hsl2004.htm
40. J. K. Reid. A sparsity-exploiting variant of the Bartels–Golub decomposition for linear programming bases. *Math. Programming*, **24**, 55–69 (1982).
41. G. W. Stewart. *Matrix Algorithms. Volume I: Basic Decompositions.* Society for Industrial and Applied Mathematics, Philadelphia, PA (1998).
42. G. W. Stewart. The decompositional approach to matrix computation. *Computing in Science and Engineering*, **2**(1), 50–59 (2000).
43. Paul N. Swarztrauber. The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle. *SIAM Rev.*, **19**(3), 490–501 (1977).
44. W. P. Tang and G. H. Golub. The block decomposition of a Vandermonde matrix and its applications. *BIT*, **21**, 505–517 (1981).
45. Plamen Yalamov and Velisar Pavlov. Stability of the block cyclic reduction. *Linear Algebra Appl.*, **249**, 341–358 (1996).

An asterisk denotes a paper reprinted in this volume.

17

CALCULATING THE SINGULAR VALUES AND PSEUDO-INVERSE OF A MATRIX (WITH W. KAHAN)

SIAM J. Numer. Anal. (2), Ser. B., 205–224 (1965).

J. SIAM Numer. Anal.
Ser. B, Vol. 2, No. 2
Printed in U.S.A.. 1965

CALCULATING THE SINGULAR VALUES AND PSEUDO-INVERSE OF A MATRIX*

G. GOLUB† and W. KAHAN‡

Abstract. A numerically stable and fairly fast scheme is described to compute the unitary matrices U and V which transform a given matrix A into a diagonal form $\Sigma = U^*AV$, thus exhibiting A's singular values on Σ's diagonal. The scheme first transforms A to a bidiagonal matrix J, then diagonalizes J. The scheme described here is complicated but does not suffer from the computational difficulties which occasionally afflict some previously known methods. Some applications are mentioned, in particular the use of the pseudo-inverse $A^I = V\Sigma^I U^*$ to solve least squares problems in a way which dampens spurious oscillation and cancellation.

1. Introduction. This paper is concerned with a numerically stable and fairly fast method for obtaining the following decomposition of a given rectangular matrix A:

$$A = U\Sigma V^*, \tag{1.1}$$

where U and V are unitary matrices and Σ is a rectangular diagonal matrix of the same size as A with nonnegative real diagonal entries. These diagonal elements are called the *singular values* or *principal values* of A; they are the nonnegative square roots of the eigenvalues of A^*A or AA^*.

Some applications of the decomposition (1.1) will be mentioned in this paper. In particular, the pseudo-inverse A^I of A will be represented in the form

$$A^I = V\Sigma^I U^*, \tag{1.2}$$

where Σ^I is obtained from Σ by replacing each positive diagonal entry by its reciprocal. The properties and applications of A^I are described in papers by Greville [15], Penrose [25], [26], and Ben-Israel and Charnes [3]. The pseudo-inverse's main value, both conceptually and practically, is that it provides a solution for the following least-squares problem.

Of all the vectors $\mathbf{x}$ *which minimize the sum of squares* $\| \mathbf{b} - A\mathbf{x} \|^2$, *which is the shortest* (*has the smallest* $\| \mathbf{x} \|^2 = \mathbf{x}^*\mathbf{x}$)?

The solution is $\mathbf{x} = A^I\mathbf{b}$. If there were only one vector $\mathbf{x}$ which minimized

* Received by the editors July 14, 1964. This paper was first presented at the Symposium on Matrix Computations at Gatlinburg, Tennessee, in April, 1964.

† Computation Center, Stanford University, Stanford, California. The work of the first author was in part supported by Contract Nonr 225(37) at Stanford University and by The Boeing Scientific Research Laboratories, Seattle, Washington.

‡ University of Toronto, Toronto, Ontario. The second author wishes to thank the National Research Council (of Canada) for their support of the Institute of Computer Sciences at the University of Toronto.

$\| \mathbf{b} - A\mathbf{x} \|$ we would save a bit of work by using

$$A^I = (A^*A)^{-1}A^*$$

instead of (1.2), and this is what we often try to do. But if A^*A is (nearly) singular then there will be infinitely many vectors $\mathbf{x}$ which (nearly) minimize $\| \mathbf{b} - A\mathbf{x} \|$ and the last formula will have to be modified in a way which takes A's rank into account (cf. [4], [6], [7]). The methods considered in this paper simplify the problem of assigning a rank to A.

In the past the conventional way to determine the rank of A was to convert A to a row-echelon form, e.g.,

$$\begin{pmatrix} x & x & x & x & x & \cdot & \cdot & \cdot \\ 0 & x & x & x & x & \cdot & \cdot & \cdot \\ 0 & 0 & x & x & x & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot \end{pmatrix}, \qquad (\text{rank} = 3),$$

in which x's represent nonzero elements and 0's represent zeros. The transformation was accomplished by premultiplying A by a succession either of elementary matrices (cf. [5]) or of unitary matrices (cf. [17]) designed to liquidate the subdiagonal elements of each column in turn. In order to obtain a simple picture like the one above it would have been necessary to perform column-interchanges to ensure that the largest possible numbers were being left on the diagonal (cf. "complete pivoting" as described by Wilkinson [33]). It is certainly possible to arrange that in the row-echelon form of A each row will have its largest element on the diagonal. Consequently the rank of A is just the number r of consecutive nonzero terms on the diagonal of its row-echelon form; all rows after the rth are zero. And Σ, correspondingly, should have just r nonzero singular values on its diagonal.

But in floating-point calculations it may not be so easy to decide whether some number is effectively zero or not. Rather, one will try to determine the rank r by observing whether all rows after the rth are negligible in comparison to the first r, with the expectation that the same will be true of the singular values. Even this criterion is hard to apply, as the following example shows:

$$\begin{pmatrix} 1 & -1 & -1 & -1 & -1 & -1 & \cdot & \cdot & \cdot \\ & 1 & -1 & -1 & -1 & -1 & \cdot & \cdot & \cdot \\ & & 1 & -1 & -1 & -1 & \cdot & \cdot & \cdot \\ & & & 1 & -1 & -1 & \cdot & \cdot & \cdot \\ & & \text{\Huge 0} & & 1 & -1 & \cdot & \cdot & \cdot \\ & & & & & 1 & \cdot & \cdot & \cdot \\ & & & & & & \cdot & \cdot & \cdot \end{pmatrix}.$$

If this matrix, already in row-echelon form, has a sufficiently large number of rows and columns, then, although it may not appear to the naked eye to be deficient in rank, it is violently ill-conditioned (it has a very tiny singular value), as can be seen by applying the matrix to the column vector whose elements are, in turn,

$$1, 2^{-1}, 2^{-2}, 2^{-3}, \cdots, 2^{-n}, \cdots .$$

On the other hand, when all the -1's in the matrix are replaced by $+1$'s then the resulting matrix is quite docile. Therefore, it would be very hard to tell, by looking at only the diagonal elements of the row-echelon form, whether or not the original matrix A had a singular value sufficiently small to be deleted during the calculation of A^I. In other words, without looking explicitly at the singular values there seems to be no satisfactory way to assign a rank to A.

The singular values of a matrix A are the nonnegative square roots of the eigenvalues of A^*A or AA^*, whichever has fewer rows and columns (see [1]). But the calculation of A^*A using ordinary floating point arithmetic does serious violence to the smaller singular values as well as to the corresponding eigenvectors which appear in U and V in (1.1). A discussion of these points can be found in a paper by Osborne [24], which also contains a nice proof of the existence of the decomposition (1.1). Since the columns of U are the eigenvectors of AA^* and the columns of V are the eigenvectors of A^*A, there is some possibility that a simple calculation of the decomposition (1.1) could be accomplished by using double-precision arithmetic to deal with A^*A and AA^* directly in some way. Such a scheme would be convenient with a machine like the IBM 7094 which has double-precision hardware. But for most other machines, and especially when a programming language deficient in double-precision facilities is used, the complicated scheme described in this paper seems to be the best we have.

Kogbetliantz [18], Hestenes [16], and Forsythe and Henrici [9] have proposed rotational or Jacobi-type methods for obtaining the decomposition (1.1). Kublanovskaja [19] has suggested a QR-type method. These methods are accurate but are slow in terms of total number of operations.

Our scheme is based upon an idea exploited by Lanczos [20]; the matrix

$$\tilde{A} = \begin{pmatrix} 0 & A \\ A^* & 0 \end{pmatrix}$$

has for its eigenvalues the singular values of A, each appearing with both a positive and a negative sign. The representation $\tilde{A}$ could not be treated directly by a standard eigenvalue-vector program without dealing with the problems which we shall discuss in detail in what follows.

2. A matrix decomposition. In order to facilitate the computation of the singular values and the pseudo-inverse of the complex $m \times n$ matrix A, we describe a convenient matrix decomposition. We assume throughout our discussion that $m \geqq n$ without any loss of generality.

THEOREM 1. *Let A be any $m \times n$ matrix with complex elements. Then A can be decomposed as*

$$A = PJQ^*,$$

where P and Q are unitary matrices and J is an $m \times n$ bidiagonal matrix of the form

$$J = \left.\begin{bmatrix} \alpha_1 & \beta_1 & 0 & \cdot & \cdot & \cdot & 0 \\ & \alpha_2 & \beta_2 & 0 & \cdot & \cdot & \cdot \\ & \mathbf{0} & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & \\ & & & & & \cdot & \beta_{n-1} \\ & & & & & & \alpha_n \\ & & & \mathbf{0} & & & \end{bmatrix}\right. . \quad \Big\} (m-n) \times n$$

Proof. The proof will be a constructive one in which Householder transformations (see [17], [21], [32]) are used. Let $A = A^{(1)}$ and let $A^{(3/2)}$, $A^{(2)}, \cdots, A^{(n)}, A^{(n+1/2)}$ be defined as follows:

$$A^{(k+1/2)} = P^{(k)}A^{(k)}, \qquad k = 1, 2, \cdots, n,$$

$$A^{(k+1)} = A^{(k+1/2)}Q^{(k)}, \qquad k = 1, 2, \cdots, n-1.$$

$P^{(k)}$ and $Q^{(k)}$ are hermitian, unitary matrices of the form

$$P^{(k)} = I - 2\mathbf{x}^{(k)}\mathbf{x}^{(k)*}, \qquad \mathbf{x}^{(k)*}\mathbf{x}^{(k)} = 1,$$

$$Q^{(k)} = I - 2\mathbf{y}^{(k)}\mathbf{y}^{(k)*}, \qquad \mathbf{y}^{(k)*}\mathbf{y}^{(k)} = 1.$$

The unitary transformation $P^{(k)}$ is determined so that

$$a_{i,k}^{(k+1/2)} = 0, \qquad i = k+1, \cdots, m,$$

and $Q^{(k)}$ is determined so that

$$a_{k,j}^{(k+1)} = 0, \qquad j = k+2, \cdots, n,$$

and $A^{(k+1)}$ has the form

$$A^{(k+1)} = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & \cdot & \cdot & & & & & \\ 0 & \alpha_2 & \beta_2 & 0 & \cdot & & & & & \\ \cdot & 0 & \cdot & \cdot & \cdot & & & \mathbf{0} & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & & & & \\ & & & & \alpha_k & \beta_k & & & & \\ & & & & & x & x & \cdot & \cdot & \cdot \\ & & \mathbf{0} & & & x & x & \cdot & \cdot & \cdot \\ & & & & & \cdot & \cdot & \cdot & \cdot & \cdot \\ & & & & & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}.$$

We illustrate the derivation of the formula for $P^{(k)}$. In order not to disturb those elements which have already been annihilated we set

$$x_i^{(k)} = 0, \qquad i = 1, 2, \cdots, k - 1.$$

Since $P^{(k)}$ is a unitary transformation, length is preserved and consequently

$$|\alpha_k|^2 = \sum_{i=k}^{m} |a_{i,k}^{(k)}|^2. \tag{2.1}$$

Also, since $P^{(k)}$ is hermitian,

$$P^{(k)}A^{(k+1/2)} = A^{(k)},$$

so that

$$(1 - 2|x_k^{(k)}|^2)\alpha_k = a_{k,k}^{(k)},$$
$$-2x_i^{(k)}\bar{x}_k^{(k)}\alpha_k = a_{i,k}^{(k)}, \qquad i = k + 1, \cdots, m,$$

and hence

$$|x_k^{(k)}|^2 = \frac{1}{2}\left(1 - \frac{a_{k,k}^{(k)}}{\alpha_k}\right), \tag{2.2}$$

$$x_i^{(k)} = \frac{-a_{i,k}^{(k)}}{2\alpha_k\bar{x}_k^{(k)}}. \tag{2.3}$$

Equations (2.1), (2.2), and (2.3) define two possible vectors $\mathbf{x}^{(k)}$ to within scalar factors of modulus one. In the interest of numerical stability, let us choose sgn α_k so that $x_k^{(k)}$ is as large as possible. Thus

$$\alpha_k = -\frac{a_{k,k}^{(k)}}{|a_{k,k}^{(k)}|}\left(\sum_{i=k}^{m} |a_{i,k}^{(k)}|^2\right)^{1/2}.$$

Summarizing, we have

$$A^{(k+1/2)} = A^{(k)} - \mathbf{x}^{(k)}\cdot 2(\mathbf{x}^{(k)*}A^{(k)}),$$

with

$$s_k = \left(\sum_{i=k}^{m} |a_{i,k}^{(k)}|^2\right)^{1/2},$$

$$\alpha_k = -s_k\left(\frac{a_{k,k}^{(k)}}{|a_{k,k}^{(k)}|}\right),$$

$$x_i^{(k)} = 0 \quad \text{for} \quad i < k,$$

$$x_k^{(k)} = \left[\frac{1}{2}\left(1 + \frac{|a_{k,k}^{(k)}|}{s_k}\right)\right]^{1/2}, \quad \text{(say)},$$

$$c_k = \left(2s_k\frac{a_{k,k}^{(k)}}{|a_{k,k}^{(k)}|}x_k^{(k)}\right)^{-1},$$

and

$$x_i^{(k)} = c_k a_{i,k}^{(k)} \quad \text{for} \quad i > k.$$

(If $s_k = 0$, just set $\alpha_k = 0$ and $\mathbf{x}^{(k)} = \mathbf{0}$.) Similarly,

$$A^{(k+1)} = A^{(k+1/2)} - 2(A^{(k+1/2)}\mathbf{y}^{(k)})\cdot\mathbf{y}^{(k)*},$$

with

$$\begin{aligned}
t_k &= \left(\sum_{j=k+1}^{n} | a_{k,j}^{(k+1/2)} |^2\right)^{1/2}, \\
\beta_k &= -t_k \cdot \frac{a_{k,k+1}^{(k+1/2)}}{| a_{k,k+1}^{(k+1/2)} |}, \\
y_j^{(k)} &= 0 \quad \text{for} \quad j \leqq k, \\
y_{k+1}^{(k)} &= \left[\frac{1}{2}\left(1 + \frac{| a_{k,k+1}^{(k+1/2)} |}{t_k}\right)\right]^{1/2}, \quad \text{(say)}, \\
d_k &= \left(2t_k \frac{a_{k,k+1}^{(k+1/2)}}{| a_{k,k+1}^{(k+1/2)} |} y_{k+1}^{(k)}\right)^{-1},
\end{aligned}$$

and

$$y_j^{(k)} = d_k \bar{a}_{k,j} \quad \text{for} \quad j > k + 1.$$

An alternative approach to bidiagonalizing A is to generate the columns of P and Q sequentially as is done by the Lanczos algorithm for tridiagonalizing a symmetric matrix. The equations

$$AQ = PJ \quad \text{and} \quad P^*A = JQ^*$$

can be expanded in terms of the columns $\mathbf{p}_i$ of P and $\mathbf{q}_i$ of Q to yield

$$\begin{aligned}
A\mathbf{q}_1 &= \alpha_1 \mathbf{p}_1, \\
A\mathbf{q}_i &= \beta_{i-1}\mathbf{p}_{i-1} + \alpha_i \mathbf{p}_i, \\
\mathbf{p}_{i-1}^* A &= \alpha_{i-1}\mathbf{q}_{i-1}^* + \beta_{i-1}\mathbf{q}_i^*,
\end{aligned} \Bigg\} \quad i = 2, 3, \cdots, n,$$

$$\mathbf{p}_n^* A = \alpha_n \mathbf{q}_n^*.$$

These lead to the following algorithm.

(2.4) *Choose* $\mathbf{q}_1$ *arbitrarily with* $\| \mathbf{q}_1 \| = 1$; *then set* $\mathbf{w}_1 = A\mathbf{q}_1$; $\alpha_1 = \| \mathbf{w}_1 \|$, $\mathbf{p}_1 = (\alpha_1)^{-1}\mathbf{w}_1$. *Set* $\mathbf{z}_i^* = \mathbf{p}_i^* A - \alpha_i \mathbf{q}_i^*$, $\beta_i = \| \mathbf{z}_i \|$, $\mathbf{q}_{i+1}^* = (\beta_i)^{-1}\mathbf{z}_i$ *for* $i = 1, 2, \cdots, n - 1$; set $\mathbf{w}_i = A\mathbf{q}_i - \beta_{i-1}\mathbf{p}_{i-1}$, $\alpha_i = \| \mathbf{w}_i \|$, $\mathbf{p}_i = (\alpha_i)^{-1}\mathbf{w}_i$ *for* $i = 2, \cdots, n$.

Of course if α_k (β_k) equals zero, one must choose a new vector $\mathbf{p}_k$ ($\mathbf{q}_k$) which is orthogonal to the previously computed $\mathbf{p}_i$'s ($\mathbf{q}_i$'s). It is easy to show then by an inductive proof that the $\mathbf{p}_i$'s and $\mathbf{q}_i$'s generated by (2.4) are the first n columns of the desired unitary matrices P and Q.

Unless an α_k or β_k vanishes, the vector $\mathbf{q}_1$ will completely determine the rest of the vectors $\mathbf{p}_i$ and $\mathbf{q}_i$. Consequently $\mathbf{q}_1$ could be so chosen that the Lanczos-type algorithm would be mathematically identical to the Householder-type algorithm except for a diagonal unitary similarity transformation. But the Lanczos-type algorithm is unstable in the presence of rounding error unless reorthogonalization along the lines suggested by Wilkinson [30] is used. That is, one must restore the orthogonality of the generated vectors by using the Gram-Schmidt method to reorthogonalize each newly generated vector $\mathbf{p}_i$ or $\mathbf{q}_i$ to the previously generated vectors $\mathbf{p}_i$ or $\mathbf{q}_i$, respectively. With the extra work involved in this reorthogonalization, the Lanczos-type algorithm is noticeably slower than the previously described Householder algorithm except possibly if A is a sparse matrix.

3. Computation of the singular values. The singular values of A and of J are the same; they are the positive square roots of J^*J. Let them be called, in order,

$$\sigma_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_n, \quad \text{where} \quad \sigma_n \geqq 0.$$

These are the numbers which appear on the diagonal of the matrix Σ which was introduced in (1.1), i.e.,

$$\Sigma = \left[\begin{array}{ccccc} \sigma_1 & & & & \\ & \sigma_2 & & 0 & \\ & & \sigma_3 & & \\ & 0 & & \ddots & \\ & & & & \sigma_n \\ \hline & & 0 & & \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \\ \left.\right\} (m-n) \times n \end{array}$$

Analogous to (1.1) is the decomposition

$$J = X\Sigma Y^* \tag{3.1}$$

in which X and Y are unitary matrices which, when they have been calculated, will lead via Theorem 1, $A = PJQ^*$, to the desired decomposition (1.1), namely,

$$A = U\Sigma V^*,$$

with $U = PX$, $V = QY$.

Evidently the last $m - n$ rows of zeros in J do not contribute to the singular values, nor do they have more than a trivial effect upon the determination of X and Y. Therefore it is convenient to delete J's last $m - n$ rows and write

$$J = \begin{pmatrix} \alpha_1 & \beta_1 & & & & \\ & \alpha_2 & \beta_2 & & & \\ & & \alpha_3 & \beta_3 & & \\ & & & \cdot & \cdot & \cdot \\ & \mathbf{0} & & & \cdot\ \cdot\ \cdot & \beta_{n-1} \\ & & & & \cdot & \alpha_n \end{pmatrix}$$

without introducing any new notation to distinguish this $n \times n$ matrix J from the $m \times n$ matrix J. This can be done because the previous equations remain valid after the following process of "abbreviation":

(i) delete the last $m - n$ rows of zeros in J and Σ;

(ii) delete the last $m - n$ columns of P and U;

(iii) delete the last $m - n$ rows and columns of X; these coincide with the last rows and columns of an $m \times m$ unit matrix. In this section and the next we deal only with the abbreviated matrices.

The singular values σ_i of J are known (cf. [20]) to be related to the eigenvalues of the $2n \times 2n$ matrix

$$\tilde{J} = \begin{pmatrix} 0 & J \\ J^* & 0 \end{pmatrix},$$

whose eigenvalues are just $+\sigma_i$ and $-\sigma_i$ for $i = 1, 2, \cdots, n$. The calculation of the eigenvalues of $\tilde{J}$ is simplified conceptually by a transformation to tridiagonal form via a permutation similarity which will be exhibited now.

Consider the matrix equation

$$(3.2) \qquad \begin{pmatrix} 0 & J \\ J^* & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x} \\ \pm\mathbf{y} \end{pmatrix} = \pm\,\sigma \begin{pmatrix} \mathbf{x} \\ \pm\mathbf{y} \end{pmatrix},$$

which, when expanded, takes the form

$$J\mathbf{y} = \sigma\mathbf{x}, \qquad J^*\mathbf{x} = \sigma\mathbf{y},$$

that is, $\alpha_i y_i + \beta_i y_{i+1} = \sigma x_i$, $\bar{\alpha}_1 x_1 = \sigma y_1$, $\alpha_n y_n = \sigma x_n$, $\bar{\beta}_{i-1} x_{i-1} + \bar{\alpha}_i x_i = \sigma y_i$. Now the substitution

$$z_{2i} = x_i, \qquad z_{2i-1} = \pm y_i,$$

leads to the equation

$$Tz = \pm\sigma z$$

in which T is a $2n \times 2n$ tridiagonal matrix

$$T = \begin{pmatrix} 0 & \bar{\alpha}_1 & & & & \multicolumn{2}{c}{\text{\Large 0}} \\ \alpha_1 & 0 & \beta_1 & & & & \\ & \bar{\beta}_1 & 0 & \bar{\alpha}_2 & & & \\ & & \alpha_2 & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & \text{\Large 0} & & & \cdot & \cdot & \bar{\alpha}_n \\ & & & & & \alpha_n & 0 \end{pmatrix}.$$

Clearly there exists a unitary diagonal matrix D such that the similarity transformation

$$(3.3) \qquad DTD^* = S = \begin{pmatrix} 0 & s_1 & & & & \text{\Large 0} & \\ s_1 & 0 & t_1 & & & & \\ & t_1 & 0 & s_2 & & & \\ & & s_2 & \cdot & \cdot & & \\ & & \text{\Large 0} & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & s_n \\ & & & & & s_n & 0 \end{pmatrix}$$

yields a tridiagonal matrix S whose elements

$$s_i = |\alpha_i| \quad \text{and} \quad t_i = |\beta_i|$$

are all real and nonnegative.

There are a number of methods for obtaining the eigenvalues of a tridiagonal symmetric matrix. One of the most accurate and effective methods is to use Sturm sequences; an ALGOL program is given by Wilkinson [35]. One can simplify the algorithm, of course, by taking advantage of the fact that the diagonal elements of T are zero.

Another method of computing the singular values of J is to compute the eigenvalues of

$$J^*J = \begin{pmatrix} |\alpha_1|^2 & \bar{\alpha}_1\beta_1 & & & & \text{\Large 0} \\ \alpha_1\bar{\beta}_1 & |\alpha_2|^2 + |\beta_1|^2 & \bar{\alpha}_2\beta_2 & & & \\ & \alpha_2\bar{\beta}_2 & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & \text{\Large 0} & & \cdot & \cdot & \bar{\alpha}_{n-1}\beta_{n-1} \\ & & & & \alpha_{n-1}\bar{\beta}_{n-1} & |\alpha_n|^2 + |\beta_{n-1}|^2 \end{pmatrix}.$$

Note again that since J^*J is a tridiagonal hermitian matrix there exists a diagonal unitary matrix Δ such that

$$\Delta(J^*J)\Delta^* = K = \begin{bmatrix} s_1^2 & s_1 t_1 & & & & \\ s_1 t_1 & s_2^2 + t_1^2 & s_2 t_2 & & 0 & \\ & s_2 t_2 & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & 0 & & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & s_{n-1} t_{n-1} \\ & & & & & s_{n-1} t_{n-1} & s_n^2 + t_{n-1}^2 \end{bmatrix},$$

where $s_i = | \alpha_i |$ and $t_i = | \beta_i |$. Hence K is a real, symmetric, positive semidefinite, tridiagonal matrix and its eigenvalues can be computed by the Sturm sequence algorithm.

Although the smaller eigenvalues of A^*A are usually poorly determined, a simple error analysis shows that all the eigenvalues of K are as well-determined as those of T. The reason for this is that the computation of the Sturm sequences is algebraically the same for both T and K. Thus to use K is preferable since the total number of operations in calculating its eigenvalues is certainly less than in computing the eigenvalues of T.

4. Orthogonal vectors properly paired. We consider now the calculation of the unitary matrices X and Y which were introduced in (3.1):

$$J = X\Sigma Y^*.$$

As pointed out in §3, J can be transformed into a real matrix by means of unitary diagonal transformations, and we shall assume henceforth that this has been done (cf. (3.3)).

To each singular value σ_i corresponds a column $\mathbf{x}_i$ of X and $\mathbf{y}_i$ of Y satisfying

$$(4.1) \qquad J\mathbf{y}_i = \sigma_i\mathbf{x}_i, \qquad J^t\mathbf{x}_i = \sigma_i\mathbf{y}_i.$$

Since $J^tJ\mathbf{y}_i = \sigma_i^2\mathbf{y}_i$ one could, in principle, calculate $\mathbf{y}_i$ as the normalized eigenvector of J^tJ corresponding to the eigenvalue σ_i^2, and $\mathbf{x}_i$ could be obtained fom the vector $J\mathbf{y}_i$ either by dividing it by σ_i or by normalizing it. However, if σ_i is small but not quite negligible, then $J\mathbf{y}_i$ will be so much contaminated by the roundoff errors left over after cancellation that the calculated $\mathbf{x}_i$ may well be neither normalized nor orthogonal to the previously calculated $\mathbf{x}$'s.

Another way to calculate X and Y might be to obtain the eigenvectors $\mathbf{x}_i$ and $\mathbf{y}_i$ of JJ^t and J^tJ independently, and then to order the vectors according to the ordering of the corresponding singular values σ_i. But if some of the singular values are too close together then the equations (4.1) are unlikely to be satisfied.

A third way, which seems at first to be free from the objections in the two preceding paragraphs, is to obtain the eigenvectors $\mathbf{z}_i$ of the $2n \times 2n$ tridiagonal matrix S of (3.3). Then the odd-numbered components of $\mathbf{z}_i$ would constitute a vector $\mathbf{y}_i$ and the even-numbered components a vector $\mathbf{x}_i$ which would satisfy (4.1). But in practice trouble shows up here in two ways. First, the facts that (4.1) is very nearly satisfied and that $\mathbf{z}_i$ has been normalized so that $\mathbf{z}_i{}^t\mathbf{z}_i = 2$ do not, in practice, though they should in theory, ensure that $\mathbf{x}_i{}^t\mathbf{x}_i = \mathbf{y}_i{}^t\mathbf{y}_i = 1$. Fortunately, unless σ_i is nearly negligible, one can normalize $\mathbf{x}_i$ and $\mathbf{y}_i$ separately without causing serious extra error. And if σ_i is negligible one can find $\mathbf{x}_i$ and $\mathbf{y}_i$ separately and ensure that they are normalized. The claims in the last two sentences can be proved, but there is no point in doing so because the second source of trouble is more drastic; if the $\mathbf{z}_i$'s are not orthogonal then neither will the $\mathbf{x}_i$'s be orthogonal, nor will the $\mathbf{y}_i$'s. The problem of ensuring that the $\mathbf{z}_i$'s are orthogonal is, in the present state of the art of computation, a serious one.

One way to ensure the orthogonality of calculated eigenvectors of a symmetric matrix is to use Jacobi's method [13], but this is slow. Another way is to reorthogonalize the calculated eigenvectors obtained, say, by inverse iteration with a tridiagonal matrix (cf. [30]); but the extra work done here is no guarantee that the vectors after orthogonalization will still be acceptable as eigenvectors. A third method, and one which seems very promising, involves the use of deflation to "remove" each eigenvector as it is obtained and thereby ensure orthogonality. We shall digress to discuss deflation methods suitable for use with symmetric tridiagonal matrices, and then adapt them to our bidiagonal matrix.

In this digression let K be some real symmetric tridiagonal matrix,

$$K = \begin{pmatrix} a_1 & b_1 & & & & \\ b_1 & a_1 & b_2 & & & \mathbf{0} \\ & b_2 & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & \mathbf{0} & & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot & b_{n-1} \\ & & & & b_{n-1} & a_n \end{pmatrix},$$

of which we already know an eigenvalue λ and its eigenvector $\mathbf{v}$. Rutishauser [27] shows how, in principle, to construct an orthogonal Hessenberg matrix H from the vector $\mathbf{v}$ so that $K_1 = H^tKH$ will have zero in place of b_{n-1}. After deleting the last row and column of the tridiagonal matrix K_1, another eigenvalue, eigenvector and deflation would be calculated, and so on. The eigenvectors of K would be the columns of an orthogonal matrix obtained by multiplying together all the H's. The orthogonality of the eigenvectors would be guaranteed (to within the limits of acceptable rounding error) irrespective of the closeness of some eigenvalues of K. Rutishauser's method needs some modification because, as Wilkinson [34, p. 189] has shown, the effect of rounding errors in the transformation $K_1 = H^t K H$ could destroy K's tridiagonal form if **v**'s first few components were too small.

In Rutishauser's deflation the matrix H can be interpreted as a product of 2×2 Jacobi-like rotations applied in succession to the first and second, second and third, third and fourth, $\cdots$, $(n-1)$th and nth rows and columns of K. After the first rotation, each rotation is chosen to annihilate a spurious term which was introduced by the previous rotation. For example, an asterisk in the following figure marks the spurious term which the third rotation must annihilate:

$$\begin{bmatrix} x & x & & & & & & \\ x & x & x & * & & & \text{\Large 0} & \\ & x & x & x & & & & \\ & * & x & x & x & & & \\ & & & x & x & x & & \\ & & & & \cdot & \cdot & \cdot & \\ & & & & & \cdot & \cdot & \cdot \\ & \text{\Large 0} & & & & & \cdot & \cdot \ \ \cdot \end{bmatrix}.$$

The first rotation, which fixes all the subsequent ones, can be determined from the first two elements of K's eigenvector **v** as suggested by Rutishauser [28, p. 226] or else from the first two elements of $K - \lambda I$. In effect, the deflation of the tridiagonal matrix K is equivalent to a QR-transformation applied to $K - \lambda I$ in the manner suggested by Ortega and Kaiser [22]. Unfortunately, this method also can be shown to be numerically unsatisfactory whenever **v**'s last few components are abnormally small, because then the element in K_1 which replaces b_{n-1} in K remains too large, in general, to be ignored. Wilkinson [34, p. 187] hints at another method analogous to the one he described in [31, pp. 351–353]; we shall outline this method briefly because we believe it to be an effective compromise between Rutishauser's two schemes.

Having found an eigenvalue λ of K we calculate the corresponding eigenvector **v** and normalize it so that its largest component lies between $\frac{1}{2}$ and 2, say. The calculation of **v** can be accomplished using the inverse iteration described by Wilkinson [30]; but since there is no way to prove that, in general, one of his inverse iterations will suffice to provide an adequately accurate **v**, we describe the following method whose properties can be established rigorously. We require that λ be chosen to be the algebraically greatest or else the least eigenvalue of K; this is no inconvenience since in the course of K's successive deflations each of its eigenvalues will be at some time the greatest or the smallest of the current matrix on hand. Next we apply

Gaussian elimination to $K - \lambda I$ *without* pivotal interchanges; there will be no trouble here (cf. [33, pp. 285–286]) provided floating point calculation is used and provided λ, if not exactly right, is larger than K's largest or smaller than K's smallest eigenvalue by perhaps a unit or so in λ's last place. The point here is that each nonzero pivot u_i in the elimination process must be of the same sign as $(K - \lambda I)$'s diagonal elements. The result of the elimination process is to express $K - \lambda I = LU$, where

$$L = \begin{pmatrix} 1 & & & & & \\ l_1 & 1 & & & 0 & \\ & l_2 & 1 & & & \\ & & l_3 & \cdot & & \\ & & & \cdot & \cdot & \\ & 0 & & & \cdot & \cdot \\ & & & & l_{n-1} & 1 \end{pmatrix}$$

and

$$U = \begin{pmatrix} u_1 & b_1 & & & & \\ & u_2 & b_2 & & 0 & \\ & & u_3 & \cdot & & \\ & & & \cdot & \cdot & \\ & 0 & & & \cdot & b_{n-1} \\ & & & & & u_n \end{pmatrix}.$$

Here $u_1 = a_1 - \lambda$ and $l_i = b_i/u_i$, $u_{i+1} = a_{i+1} - \lambda - l_ib_i$ for $i = 1, 2, \cdots, n - 1$. Next we attempt the solution of $(K - \lambda I)\mathbf{v} = \mathbf{r}$ using for $\mathbf{r}$ a vector whose elements all have the same magnitude but signs chosen to maximize the elements of $\mathbf{v}$. The choice of sign is accomplished by first solving $L\,\mathbf{s} = \mathbf{r}$ as follows:

$$s_1 = +1,$$

$$s_{i+1} = (-l_is_i) + \operatorname{sgn}(-l_is_i), \quad i = 1, 2, \cdots, n - 1.$$

The solution of $U\mathbf{v} = \mathbf{s}$ for $\mathbf{v}$ and the subsequent normalization of $\mathbf{v}$ complete the calculation. Provided no two pivots u_i have opposite signs one can show that the elements of $\mathbf{v}$ each have the same signs as the corresponding elements of the desired eigenvector despite the rounding errors committed during $\mathbf{v}$'s calculation. Furthermore, the elements of $\mathbf{r}$ exhibit the same signs as those of $+\mathbf{v}$ or $-\mathbf{v}$, depending upon the sign of the u_i's. Consequently the cosine of the angle between $\mathbf{r}$ and the correct eigenvector is at least $N^{-1/2}$ in magnitude, and finally we conclude that $K\mathbf{v}$ must differ from $\lambda\mathbf{v}$ by no more than a few units in the last place (cf. the argument in [30]). Now even if $\mathbf{v}$ is contaminated by components of the eigenvectors corresponding to other eigenvalues pathologically close to λ, it will look enough like a true eigenvector to permit the deflation process to proceed. This process for calculating $\mathbf{v}$ is simpler and a little faster than Wilkinson's.

Now that we have $\mathbf{v}$ we proceed to the deflation along the lines outlined by Wilkinson. Each 2 × 2 rotation is embedded in an $n \times n$ matrix

$$P_j = \begin{pmatrix} 1 & & & & & & & & \\ & 1 & & & & & \mathbf{0} & & \\ & & \ddots & & & & & & \\ & & & 1 & & & & & \\ & & & & c_j & s_j & & & \\ & & & & -s_j & c_j & & & \\ & & & & & & 1 & & \\ & & & & & & & 1 & \\ & & \mathbf{0} & & & & & \ddots & \\ & & & & & & & & 1 \end{pmatrix}, \quad j = 1, 2, \cdots, n-1,$$

with c_j for its jth and $(j+1)$th diagonal elements, where $c_j^2 + s_j^2 = 1$. Suppose the products

$$P_jP_{j-1} \cdots P_1(K - \lambda I)P_1^t \cdots P_{j-1}^tP_j^t \quad \text{and} \quad P_jP_{j-1} \cdots P_1\mathbf{v}$$

have the forms

$$P_j\,P_{j-1} \cdots P_1(K - \lambda I)P_1^t \cdots P_{j-1}^t\,P_j^t = \begin{pmatrix} x & x & & & & & \\ x & x & x & & & & \\ & x & x & h_j & w_j & & \\ & & h_j & x & x & & \\ & & w_j & x & x & x & \\ & & & & x & x & x \\ & & & & & x & x \end{pmatrix}$$

and

$$P_j\,P_{j-1} \cdots P_1\,\mathbf{v} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \phi_{j+1} \\ v_{j+2} \\ x \\ x \end{pmatrix}.$$

At the start we can take

$$h_0 = a_1 - \lambda, \qquad w_0 = b_1\,, \qquad \phi_1 = v_1\,.$$

To continue the deflation we must so determine P_{j+1} that its application will simultaneously annihilate the spurious element w_j in the jth row and column of the matrix as well as the vector's $(j + 1)$th element ϕ_{j+1}. But in practice the accumulation of rounding errors will prevent the exact annihilation of both elements; instead we shall have to be satisfied with a P_{j+1} which leaves negligible residuals in place of w_j and ϕ_{j+1}. Wilkinson, having scaled $K - \lambda I$ so that its largest element lies between $\frac{1}{2}$ and 2, would use whichever of the equations

$$w_j c_{j+1} = h_j s_{j+1}, \qquad \phi_{j+1} c_{j+1} = -v_{j+2} s_{j+1},$$

contained the largest coefficient $|w_j|$, $|h_j|$, $|\phi_{j+1}|$, or $|v_{j+2}|$ to determine, in conjunction with $c_{j+1}^2 + s_{j+1}^2 = 1$, the values c_{j+1} and s_{j+1}. This method seems to be effective and we believe that it should always work, but since we cannot prove the method's infallibility, our work is incomplete.

Now we can show how to construct a deflation process for the bidiagonal matrix J. The first step is to obtain J's largest singular value σ; σ^2 is the largest eigenvalue of the tridiagonal matrix J^tJ (see §3). The next step requires the corresponding vectors $\mathbf{x}$ and $\mathbf{y}$ which can be obtained either by solving $J^tJ\mathbf{y} = \sigma^2\mathbf{y}$ for $\mathbf{y}$ and setting $\mathbf{x} = \sigma^{-1}J\mathbf{y}$, or by calculating σ's eigenvector $\mathbf{z}$ of S in (3.3) and hence obtaining $\mathbf{x}$ and $\mathbf{y}$ from $\mathbf{z}$'s even and odd components respectively. Both methods for getting $\mathbf{x}$ and $\mathbf{y}$ are numerically stable when performed in floating point. The deflation of J is accomplished by a sequence of 2×2 rotations applied in succession to its first and second columns, its first and second rows, its second and third columns, its second and third rows, its third and fourth columns, $\cdots$, its $(n - 1)$th and nth rows. The ith rotation applied to rows i and $i + 1$ of J must simultaneously annihilate a spurious subdiagonal element, introduced into row $i + 1$ by the previous column rotation, and the ith element in the current $\mathbf{x}$-vector. The ith column rotation, except for the first, must annihilate a spurious term introduced by the previous row rotation into the $(i + 1)$th column just above the first superdiagonal, and simultaneously the transpose of the ith column rotation must liquidate the ith element of the current $\mathbf{y}$-vector. The first column rotation would when applied to $J^tJ - \sigma^2 I$ annihilate the element in its first row and second column. At the end of the deflation process J's element b_{n-1} should have been replaced by zero. Of course, rounding errors will prevent the rotations from performing their roles exactly upon both the matrix J and the vectors $\mathbf{x}$ and $\mathbf{y}$, but just as in the deflation of a tridiagonal matrix we are able so to determine the rotations that negligible residuals are left behind in place of the elements we wished to liquidate.

After deflating J we delete its last row and column and repeat the process until J is deflated to a 1×1 matrix or the deflated J becomes negligibly small. At the end we multiply the rotations in reverse order to construct the matrices X and Y which put J into the form (3.1):

$$J = X\Sigma Y.$$

(If J was complex, a unitary diagonal transformation should be incorporated here.) Finally the matrices P and Q of Theorem 1 are multiplied thus:

$$U = PX, \qquad V = QY,$$

to exhibit the decomposition (1.1):

$$A = U\Sigma V.$$

The two matrix multiplications PX and QY take most of the work.

5. Applications. The basic decomposition given by (1.1) has many applications in data analysis and applied mathematics. Suppose the matrix A arises from statistical observation, and we wish to replace A by another matrix $\hat{A}$ (say) which has lower rank p and is the best approximation to A in some sense. If we use the Frobenius norm (i.e., $\| A \|^2 = \text{trace } A^*A$) then the problem has been solved [8] as follows.

THEOREM 2. *Let A be an $m \times n$ matrix of rank r which has complex elements. Let S_p be the set of all $m \times n$ matrices of rank $p < r$. Then for all $B \in S_p$,*

$$\| A - \hat{A} \| \leqq \| A - B \|,$$

where

$$\hat{A} = U\hat{\Sigma}V^*$$

and $\hat{\Sigma}$ is obtained from the Σ of (1.1) *by setting to zero all but its p largest singular values σ_i.*

Proof. Since $A = U\Sigma V^*$ and the Frobenius norm is unitarily invariant,

$$\| A - B \| = \| \Sigma - U^*BV \|.$$

Let $U^*BV = C$. Then

$$\| \Sigma - C \|^2 = \sum_{i=1}^{n} | \sigma_i - c_{ii} |^2 + \sum_{i \neq j} | c_{ij} |^2 \geqq \sum_{i=1}^{n} | \sigma_i - c_{ii} |^2.$$

Now it is convenient to order the singular values in such a way that $\sigma_i \geqq \sigma_{i+1}$. Thus, $\| A - B \|^2$ is minimized if $c_{ii} = \sigma_i$ for $i = 1, 2, \cdots, p$, and $c_{ij} = 0$ otherwise, i.e., for $C = \hat{\Sigma}$. Obviously,

$$\| A - \hat{A} \| = (\sigma_{p+1}^2 + \cdots + \sigma_r^2)^{1/2}.$$

Finding the vector $\mathbf{x}$ of shortest length which minimizes $\| \mathbf{b} - A\mathbf{x} \|$ is equivalent to finding the vector $\mathbf{y}$ of shortest length which minimizes

$\| \mathbf{c} - J\mathbf{y} \|$, where $\mathbf{c} = P^*\mathbf{b}$ and $\mathbf{y} = Q^*\mathbf{x}$. Here a natural question arises: is there any method which bypasses the complicated scheme in §3 and §4 for exhibiting J's singular values explicitly, and instead takes advantage of J's simple bidiagonal form to solve the least squares problem or to calculate J^I? Such a method, if it exists, must retain provision for intentional perturbations designed to delete, in effect, negligible singular values without inducing too large a discrepancy in J or A. Unfortunately, J's simple form is deceptive; even J's rank is hard to estimate without further calculation. For example, if J's rank r is less than n, then at least $n - r$ of the α_i's, and possibly more, should vanish; but in practice none of the α_i's may be negligible even though several may be very small compared with adjacent β_i's and, in consequence, a few of J's singular values may be negligible.

Perhaps the recurrence described by Greville [15] can be modified by the introduction of pivoting and then applied to J to calculate J^I. Until this scheme is worked out, the best method we can suggest for solving the least squares problem together with controllable perturbations is the following. Compute explicitly the representation

$$A = U\Sigma V^*,$$

decide which of the singular values are small enough to ignore, replace the remaining singular values by their reciprocals to obtain Σ^I, and finally use

$$A^I = V\Sigma^I U^*$$

to obtain the least squares solution $\mathbf{x} = A^I\mathbf{b}$. Once again, to ignore some singular values $\sigma_{r+1}, \sigma_{r+2}, \cdots, \sigma_n$ is equivalent to perturbing A by a matrix whose norm is $(\sum_{i=r+1}^n \sigma_i^2)^{1/2}$.

In some scientific calculations it is preferable that a given square matrix A be perturbed as little as possible (just rounding errors), but instead a perturbation $\delta\mathbf{b}$ in the right-hand side $\mathbf{b}$ of the equation $A\mathbf{x} = \mathbf{b}$ is permissible provided $\| \delta\mathbf{b} \|$ does not exceed a given tolerance ϵ. The substitution

$$\mathbf{y} = V^*\mathbf{x}, \qquad \mathbf{c} = U^*\mathbf{b}, \qquad \delta\mathbf{c} = U^*\delta\mathbf{b},$$

transforms the perturbed equation $A\mathbf{x} = \mathbf{b} + \delta\mathbf{b}$ into an equivalent diagonal system

$$\Sigma\mathbf{y} = \mathbf{c} + \delta\mathbf{c},$$

in which the permissible perturbation $\delta\mathbf{c}$ still satisfies

$$(5.1) \qquad \| \delta\mathbf{c} \| < \epsilon.$$

Subject to this constraint, $\delta\mathbf{c}$ may be chosen to optimize some other criterion. For example, suppose we wish to minimize $\| \mathbf{x} \| = \| \mathbf{y} \|$. Then ideally $\delta\mathbf{c}$

should satisfy $\Sigma^2 \delta\mathbf{c} + \lambda(\mathbf{c} + \delta\mathbf{c}) = 0$ with some suitable positive value of the Lagrange multiplier λ sufficiently small so that (5.1) is satisfied too. But for most practical purposes it is sufficient to use trial and error to determine λ to within a factor of two so that $\delta\mathbf{c} = -(I + \lambda^{-1}\Sigma^2)^{-1}\mathbf{c}$ will satisfy $\delta\mathbf{c}^*\delta\mathbf{c} < \epsilon^2$. The use of such a technique in least squares problems tends to suppress violent oscillation and cancellation which might otherwise detract from the usefulness of the solution $\mathbf{x}$.

A similar technique is valuable for the solution of the sets of linear equations which approximate integral equations of the form

$$\int A(i, j)x(j)\, dj = b(i).$$

Here the numerical treatment of the integral equation, in using singular values, is similar to the theoretical treatment found in [29]. Once again, the use of the decomposition $A = U\Sigma V^*$ aids the suppression of spurious oscillations in the function x.

We close with a warning; diagonal transformations can change A's singular values and A^I in a nontrivial way. Therefore some sort of equilibration may be necessary to allow each row and column of A to communicate its proper significance to the calculation. Two useful forms of equilibration are:

(i) scale each row and column of A in such a way that all the rows have roughly the same norm and so have all the columns;

(ii) scale each row and column of A in such a way that the absolute uncertainty in each element of A does not vary much from element to element. On least squares problems such equilibration is accomplished by weighting each residual in the sum of squares (see [2], [10], [11], [23] on equilibration algorithms, and [14]).

REFERENCES

[1] A. R. Amir-Moéz and A. L. Fass, *Elements of Linear Spaces*, Pergamon, New York, 1962, Chap. 12.

[2] F. L. Bauer, *Optimally scaled matrices*, Numer. Math., 5 (1963), pp. 73–87.

[3] A. Ben-Israel and A. Charnes, *Contributions to the theory of generalized inverses*, J. Soc. Indust. Appl. Math., 11 (1963), pp. 667–699.

[4] A. Ben-Israel and S. J. Wersan, *An elimination method for computing the generalized inverse of an arbitrary complex matrix*, J. Assoc. Comput. Mach., 10 (1963), pp. 532–537.

[5] G. Birkhoff and S. MacLane, *A Survey of Modern Algebra*, Macmillan, New York, 1953, Chap. VII, §6.

[6] J. W. Blattner, *Bordered matrices*, J. Soc. Indust. Appl. Math., 10 (1962), pp. 528–536.

[7] J. C. G. Boot, *The computation of the generalized inverse of singular or rectangular matrices*, Amer. Math. Monthly, 70 (1963), pp. 302–303.

[8] C. Eckart and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, 1 (1936), pp. 211–218.

[9] G. E. Forsythe and P. Henrici, *The cyclic Jacobi method for computing the principal values of a complex matrix*, Trans. Amer. Math. Soc., 94 (1960), pp. 1–23.

[10] G. E. Forsythe and E. G. Straus, *On best conditioned matrices*, Proc. Amer. Math. Soc., 6 (1955), pp. 340–345.

[11] D. R. Fulkerson and P. Wolfe, *An algorithm for scaling matrices*, SIAM Rev., 4 (1962), pp. 142–146.

[12] W. Givens, *Numerical computation of the characteristic values of a real symmetric matrix*. Oak Ridge National Laboratory, Report No. 1574, 1954.

[13] H. H. Goldstine, F. J. Murray, and J. von Neumann, *The Jacobi method for real symmetric matrices*, J. Assoc. Comput. Mach., 6 (1959), pp. 59–96.

[14] G. H. Golub, *Comparison of the variance of minimum variance and weighted least squares regression coefficients*, Ann. Math. Statist., 34 (1963), pp. 984–991.

[15] T. N. E. Greville, *Some applications of the pseudo-inverse of a matrix*, SIAM Rev., 2 (1960), pp. 15–22.

[16] M. R. Hestenes, *Inversion of matrices by biorthogonalization and related results*, J. Soc. Indust. Appl. Math., 6 (1958), pp. 51–90.

[17] A. S. Householder, *Unitary triangularization of a nonsymmetric matrix*, J. Assoc. Comput. Mach., 5 (1958), pp. 339–342.

[18] E. G. Kogbetliantz, *Solution of linear equations by diagonalization of coefficients matrix*, Quart. Appl. Math., 13 (1955), pp. 123–132.

[19] V. N. Kublanovskaja, *Some algorithms for the solution of the complete problem of eigenvalues*, V. Vyčisl. Mat.i. Mat. Fiz., 1 (1961), pp. 555–570.

[20] C. Lanczos, *Linear Differential Operators*, Van Nostrand, London, 1961, Chap. 3.

[21] D. D. Morrison, *Remarks on the unitary triangularization of a nonsymmetric matrix*, J. Assoc. Comput. Mach., 7 (1960), pp. 185–186.

[22] J. M. Ortega and H. F. Kaiser, *The LL^t and QR methods for symmetric tridiagonal matrices*, Comput. J., 6 (1963), pp. 99–101.

[23] E. E. Osborne, *On pre-conditioning of matrices*, J. Assoc. Comput. Mach., 7 (1960), pp. 338–345.

[24] ———, *On least squares solutions of linear equations*, Ibid., 8 (1961), pp. 628–636.

[25] R. Penrose, *A generalized inverse for matrices*, Proc. Cambridge Philos. Soc., 51 (1955), pp. 406–413.

[26] ———, *On best approximate solutions of linear matrix equations*, Ibid., 52 (1956), pp. 17–19.

[27] H. Rutishauser, *Deflation bei Bandmatrizen*, Z. Angew. Math. Phys., 10 (1959), pp. 314–319.

[28] ———, *On Jacobi rotation patterns*, Proc. Symp. Appl. Math. XV, Experimental Arithmetic, High Speed Computing, and Mathematics, Amer. Math. Soc., 1963, pp. 219–240.

[29] F. Smithies, *Integral Equations*, Cambridge University Press, Cambridge, 1958, Chap. VIII.

[30] J. H. Wilkinson, *The calculation of the eigenvectors of codiagonal matrices*, Comput. J., 1 (1958), pp. 148–152.

[31] ———, *Stability of the reduction of a matrix to almost triangular and triangular forms by elementary similarity transformations*, J. Assoc. Comput. Mach., 6 (1959), pp. 336–359.

[32] ———, *Householder's method for the solution of the algebraic eigenproblem*, Comput. J., 3 (1960), pp. 23–27.

[33] ———, *Error analysis of direct methods of matrix inversion*, J. Assoc. Comput. Mach., 8 (1961), pp. 281–330.

[34] ———, *Error analysis of eigenvalue techniques based on orthogonal transformations*, J. Soc. Indust. Appl. Math., 10 (1962), pp. 162–195.

[35] ———, *Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection*, Numer. Math., 4 (1962), pp. 362–367.

18

THE SIMPLEX METHOD OF LINEAR PROGRAMMING USING LU DECOMPOSITION (WITH R. H. BARTELS)

Comm. ACM (12:5), 266–268 (1969).

This paper is a model for the way that Gene likes to work: find someone interesting in another field to talk with, find out what the problems, approaches, and computational practices are in that field, and then come up with some possible way in which good numerical linear algebra might help. Then tell someone else how a thing should really be done and get them to do the deed. The interesting person in question that led to this work was George Dantzig and the field was linear programming. My first contact with this interaction between Gene and George Dantzig must have been after Gene had the tableau method of the simplex method presented to him. I recall an incredulous statement from Gene roughly like, "It's crazy; they use Gauss–Jordan elimination without pivot selection!" Gene proposed that some method of updating the LU decomposition of the basis matrix be found rather than the Gauss–Jordan update of the basis matrix inverse, as was used in the tableau. I needed a PhD thesis topic, so I became the "someone else" who was to do the deed. The updating process was Gene's, and I was asked to see what I could say about its numerical stability using Wilkinson round-off analysis.

The paper cited here worked on the explicit LU decomposition and its update. As far as the linear programming community was concerned, this was of no value, since sparsity must be honored due to the extreme size of linear programming problems, and sparsity was quickly lost in updating explicit LU decompositions. It wasn't until product-form updates and sparsity orderings were studied by Michael Saunders that the work gained street credibility.

The basic insight that many matrix decompositions could, with stability, be updated (and, later, downdated), either in explicit or in product form, spawned about a decade of activity in its own right, involving a number of researchers and generating several important papers. The results had fundamental impact on the way in which linear and nonlinear optimization algorithms and certain sequential statistical procedures are designed.

Oh, and I did get a PhD from the work. Thanks, Gene.

Richard Bartels
Waterloo, Ontario, Canada

Numerical Analysis

J. F. TRAUB, Editor

The Simplex Method of Linear Programming Using LU Decomposition

Richard H. Bartels and Gene H. Golub
Stanford University, Stanford, California*

Standard computer implementations of Dantzig's simplex method for linear programming are based upon forming the inverse of the basic matrix and updating the inverse after every step of the method. These implementations have bad round-off error properties. This paper gives the theoretical background for an implementation which is based upon the LU decomposition, computed with row interchanges, of the basic matrix. The implementation is slow, but has good round-off error behavior. The implementation appears as CACM Algorithm 350.

KEY WORDS AND PHRASES: simplex method, linear programming, LU decomposition, round-off errors, computational stability
CR CATEGORIES: 5.41

1. LU Decomposition

The linear programming problem:

$$\text{maximize } d^T x$$

$$\text{subject to } \begin{cases} Gx = b \\ x \geq 0 \end{cases} \qquad (1)$$

where $d^T = [d_0, \cdots, d_{n-1}]$, $b^T = [b_0, \cdots, b_{m-1}]$, and

$$G = \begin{bmatrix} g_{0,0} & \cdots & g_{0,n-1} \\ \vdots & & \vdots \\ g_{m-1,0} & \cdots & g_{m-1,n-1} \end{bmatrix}$$

are given, with $b \geq 0$, is commonly solved by a two-phase process built upon the simplex method of G. B. Dantzig [2]. Each step of this method requires the solution of three systems of linear equations involving a common matrix of coefficients, the basis matrix P. It is the usual practice to solve these systems by forming P^{-1}, either directly or in factored form as a product of transformations, and then applying it to the right-hand sides of the systems.

The basis matrix of any simplex step differs from that of the preceding step in only one column, so it is easier to update P^{-1} than to invert the new P. While this generally produces satisfactory results, it is vulnerable to computational problems in two respects. First, if P^{-1} is continually updated, computational inaccuracies imposed by limited-precision arithmetic (round-off errors) are allowed to propagate from step to step. Second, the updating, in whatever way it is carried out, is equivalent to premultiplying P^{-1} by a matrix of the form:

$$\begin{array}{c|c|c|c|}
\cline{2-4}
 & \begin{matrix}1 & & 0\\ & \ddots & \\ 0 & & 1\end{matrix} & \begin{matrix}-y_0/y_k\\ \vdots\\ -y_{k-1}/y_k\end{matrix} & O \\ \cline{2-4}
k & 0 & 1/y_k & 0 \\ \cline{2-4}
 & O & \begin{matrix}-y_{k+1}/y_k\\ \vdots \\ -y_{m-1}/y_k\end{matrix} & \begin{matrix}1 & & 0\\ & \ddots & \\ 0 & & 1\end{matrix} \\ \cline{2-4}
\multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{k} & \multicolumn{1}{c}{}
\end{array}$$

In the presence of round-off errors this computation can be quite inaccurate if y_k is small relative to the other y_i.

Both of these problems can be eliminated if, instead of P^{-1}, the LU decomposition of P is computed using row interchanges to select pivots. The problem of solving a linear system based upon P is then reduced to that of backsolving two triangular linear systems. The numerical stability of this scheme is generally quite good (see [4]). Advantage can be taken of the similarity between any two successive matrices so as to economize on the computation of the LU decompositions. For details see [1, Secs. 5–6].

Additional accuracy is obtained in the program (see Algorithm 350) by iteratively refining the solution to problem (1), after it has been found, according to the scheme given in [4, p. 121].

The algorithm requires $n^3/6 + O(n^2)$ multiplications per exchange on the average. Thus the algorithm is most effectively used for very ill-conditioned problems or in conjunction with other implementations of the simplex algorithm which produces an initial basis more cheaply. A fast and accurate version of the simplex algorithm is described in [5] but this requires additional storage.

This paper gives the theoretical background of Algorithm 350, "Simplex-Method Procedure Employing LU Decomposition," by the same authors, which appears on pages 275–278.

* Computer Science Department. This project was supported in part by NSF under project GP948 and ONR under project NR 044 211.

2. The Two-Phase Solution Process

The possibility of degeneracy is ignored, as is customary.

Phase 1 (Find a basic feasible solution.)

$$\text{maximize } -e^T a$$

$$\text{subject to}\begin{cases} Gx + a = b \\ x \geq 0, \quad a \geq 0, \end{cases} \tag{2}$$

where $a^T = [a_0, \cdots, a_{m-1}]$ is a vector of "artificial variables" and $e^T = [1, \cdots, 1]$ (m components). The solution is obtained by starting with $a = b$ as a basic feasible solution and applying the simplex method. As each a_i becomes nonbasic, it may be dropped from the problem altogether. Since $-e^T a \leq 0$ for all a, problem (2) has a solution. If $max(-e^T a) \neq 0$, then problem (1) has no basic feasible solution. Otherwise, the second phase is entered.

Phase 2 (Find an optimal basic feasible solution.)

If the solution to problem (2) contains only components of x as basic variables, these become the initial basic variables for solving problem (1) by the simplex method. If artificial variables, say $a_{j_0}, \cdots, a_{j_l}$, remain basic but at zero level in the solution of (2), an equivalent problem to (1) is solved:

$$\text{maximize } d^T x$$

$$\text{subject to}\begin{cases} \left[\begin{array}{c|c|c|c|c} G & e_{j_0} & \cdots & e_{j_l} & 0 \\ \hline 0 & 1 & \cdots & 1 & 1 \end{array}\right] \left[\begin{array}{c} x \\ \hline a_{j_0} \\ \hline \vdots \\ \hline a_{j_l} \\ \hline s \end{array}\right] = \left[\begin{array}{c} b \\ \hline 0 \end{array}\right] \\ x \geq 0, s \geq 0, a_{j_0} \geq 0, \cdots, a_{j_l} \geq 0 \end{cases} \tag{3}$$

where e_k represents the kth column of the identity matrix (column numbering begun at zero). The additional variable s, together with the additional constraint

$$s + \sum_{i=0}^{l} a_{j_i} = 0,$$

holds the artificial variables at zero level throughout further computation. The simplex method is applied to problem (3) with s and the basic variables from problem (2) as the initial set of basic variables. (For an example of this two-phase process in operation see [3, Sec. 7.7].)

3. Some Computational Details

Phase 1 is skipped if a basic feasible set of variables for problem (1) is known a priori. If a partial set of basic variables is known for problem (1), the computation of phase 1 can be restructured to take this into account. To be precise, let κ $(1 \leq \kappa \leq m - 1)$ columns of G (without loss of generality these can be made $g_{n-\kappa}, \cdots, g_{n-1}$, the last columns of G) be used to construct the following matrix:

$$P = \left[\begin{array}{c|c|c|c|c|c} e_0 & \cdots & e_{m-\kappa-1} & g_{n-\kappa} & \cdots & g_{n-1} \end{array}\right].$$

Suppose that $P^{-1}b \geq 0$. (This situation arises if the variables are numbered so that $x_{n-\kappa}, \cdots, x_{n-1}$ are the slack variables and the constraints are ordered so that $g_{n-\kappa} = e_{m-\kappa}, \cdots, g_{n-1} = e_{m-1}$, whereby P becomes the identity matrix.) Then problem (2) can be replaced by the problem:

$$\text{maximize } -\tilde{e}^T \tilde{a}$$

$$\text{subject to}\begin{cases} \left[\begin{array}{c|c|c} G^{(1)} & \begin{array}{c} \tilde{I} \\ \hline 0 \end{array} & G^{(2)} \end{array}\right] \left[\begin{array}{c} x^{(1)} \\ \hline \tilde{a} \\ \hline x^{(2)} \end{array}\right] = b \\ x^{(1)} \geq 0, \quad x^{(2)} \geq 0, \quad \tilde{a} \geq 0 \end{cases} \tag{4}$$

where $x^{(1)} = [x_0, \cdots, x_{n-\kappa-1}]$, $x^{(2)} = [x_{n-\kappa}, \cdots, x_{n-1}]$, $G^{(1)}$ and $G^{(2)}$ are appropriate submatrices of G, $\tilde{a} = [a_0, \cdots, a_{m-\kappa-1}]$, $\tilde{e} = [1, \cdots, 1]$ ($m - \kappa$ components), and $\tilde{I}$ is the $(m - \kappa)$-order identity matrix. The simplex method is begun with $a_0, \cdots, a_{m-\kappa-1}, x_{n-\kappa}, \cdots, x_{n-1}$ as basic variables. Since there are fewer artificial variables to be driven to zero in problem (4) than in problem (2), we expect problem (4) to require on the average less work to solve.

A reduction of computation beyond that described in [1] can be arranged for the LU decompositions calculated in phase 1. Each time an artificial variable becomes nonbasic, a row and column interchange can be applied to the basis matrix so that it always has the partitioned form

$$P = \left[\begin{array}{c|c} I & R \\ \hline O & S \end{array}\right]$$

where I is the identity matrix of appropriate order, and R and S are composed of components of G. The LU decomposition of P is known when the decomposition $S = \mathcal{L}\mathcal{U}$ has been computed; viz.

$$P = \left[\begin{array}{c|c} I & O \\ \hline O & \mathcal{L} \end{array}\right] \left[\begin{array}{c|c} I & R \\ \hline O & \mathcal{U} \end{array}\right]$$

The triangular systems of equations induced by this format are solved in a particularly efficient manner by Algorithm 350.

4. Example Application

Let V be a random variable restricted to a finite set of values $v_0, \cdots, v_{n-1}$. Let p_i be the probability that V has value v_i. If certain moments

$$\mu_k = \sum_{j=0}^{n-1} v_j^k p_j, \qquad k \in \{k_0, \cdots, k_{m-1}\}$$

are known together with the values v_j, estimates of the

p_i and of unknown moments μ_l can be made via linear programming; viz.

$$\alpha_i \le p_i \le \beta_i \quad \text{and} \quad \gamma_l \le \mu_l \le \delta_l \,,$$

where

$$\alpha_i = -\max(-x_i), \qquad \beta_i = \max(x_i)$$

$$\gamma_l = -\max(-\sum_{j=0}^{n-1} v_j{}^l x_j), \qquad \delta_l = \max(\sum_{j=0}^{n-1} v_j{}^l x_j),$$

all subject to the constraints

$$x_0 \ge 0, \cdots, x_{n-1} \ge 0, \qquad \sum_{j=0}^{n-1} x_j = 1,$$

$$\sum_{j=0}^{n-1} v_j{}^k x_j = \mu_k \,, \qquad k \in \{k_0 \,, \cdots, k_{m-1}\}, \quad k \ne 0.$$

The basis matrices encountered in solving these problems are submatrices of a Vandermonde matrix, making them somewhat poorly conditioned; so it is important that the simplex method be carried out accurately. Only the objective function differs from one estimation problem to the next; so the solution of any one of the problems would provide an initial basis for the other problems. Phase 1 would need to be carried out only once.

Thus, for example, to compute upper bounds for $p_0 \,, \cdots, p_{n-1}$ from given values $v_0 \,, \cdots, v_{n-1}$ and given moments $\mu_0 \,, \cdots, \mu_{m-1}$, one would set up arrays $G[i, j] \equiv v_j{}^i$ and $b[i] \equiv \mu_i$ and execute

```
kappa := 0;  for $i$ := 0 step 1 until $n - 1$ do $d[i]$ := 0;
for $i$ := 0 step 1 until $n - 1$ do
begin $d[i]$ := 1;
  if $i \ge 1$ then $d[i - 1]$ := 0;
  linprog(m, n, kappa, G, b, d, x, z, ind, infeas, unbdd, sing);
  upper bound[i] := z;
end;
```

As an illustration, the preceding was run on Stanford University's B5500 computer with the data $\mu_0 = 1$, $\mu_1 = 3$, $\mu_2 = 10.5$, $\mu_3 = 40.5$ and $v_i = i$ for $i = 0, \cdots, 6$. The results are listed below.

Computed upper bound	*Actual probability*
.083333333325	.015625
.19999999998	.09375
.50000000001	.234375
.83333333335	.3125
.49999999999	.234375
.20000000001	.09375
.083333333334	.015625

Lower bounds for the p_i computed from the above data in a similar fashion were all zero. Computed bounds for μ_4 through μ_{10} were as follows.

Computed lower bound	*Actual moment*	*Computed upper bound*
165.5	168.0	175.5
700.5	738.0	850.5
3035.5	3396.75	4495.5
13390.5	16251.75	25150.5
59965.5	80335.5	145435.5
272200.5	408280.5	856210.5
1251135.5	2124764.25	5088055.5

Received September, 1967; revised November, 1968

REFERENCES

1. Bartels, R. H., and Golub, G. H. Stable numerical methods for obtaining the Chebyshev solution to an overdetermined system of equations. *Comm. ACM 11*, 6 (June 1968), 401–406.
2. Dantzig, G. B. *Linear Programming and Extensions.* Princeton U. Press, Princeton, N.J., 1963.
3. Hadley, G. *Linear Programming.* Addison-Wesley, Reading, Mass., 1962.
4. Wilkinson, J. H. *Rounding Errors in Algebraic Processes.* Prentice-Hall Series in Automatic Computation, Englewood Cliffs, N.J., 1963.
5. Bartels, Richard H. A numerical investigation of the simplex method. Tech. Rep. No. CS 104, Comput. Sci. Dep., Stanford U., Stanford, Calif., July 1968.

Rough and Ready Error Estimates in Gaussian Integration of Analytic Functions

Philip Rabinowitz
The Weizmann Institute of Science, Rehovoth, Israel*

KEY WORDS AND PHRASES: numerical integration, analytic functions, error estimates, Gaussian integration, tabulated error coefficients, computable error coefficients, Cauchy integral formula, Chebyshev polynomials
CR CATEGORIES: 5.11, 5.16

Two expressions are derived for use in estimating the error in the numerical integration of analytic functions in terms of the maximum absolute value of the function in an appropriate region of regularity. These expressions are then specialized to the case of Gaussian integration rules, and the resulting error estimates are compared with those obtained by the use of tables of error coefficients.

Davis and Rabinowitz [1] presented a method for estimating the error committed in integrating an analytic function by an arbitrary integration rule

$$I(f) = \sum_{i=1}^{n} w_i f(x_i).$$

If the error is denoted by $E(f) = \int_{-1}^{1} f(x)\,dx - I(f)$, we

* Department of Applied Mathematics

19

ON DIRECT METHODS FOR SOLVING POISSON'S EQUATION (WITH B. L. BUZBEE AND C. W. NIELSON)

SIAM J. Numer. Anal. (7), 627–656 (1970).

Deciphering, Analyzing and Extending Buneman's Stabilization of Cyclic Odd-Even Reductions – An Exercise in Good Fortune, Even Serendipity!

Clair Nielson was a scientist in the fusion energy division at Los Alamos. About 1968 or 1969, Clair acquired a software subroutine that involved a very fast, direct technique for solving Poisson's equation on a rectangle. The routine had been written by Oscar Buneman at the Stanford Linear Accelerator and Buneman didn't document the mathematical details in it. Clair and others at Los Alamos tested this routine and found that it was accurate and *much faster* than widely used iterative techniques at that time, e.g., successive overrelaxation. Since there was no documentation, Clair stopped by my office and quizzed me about it, i.e. what algorithm did it use?

The subroutine (written in Fortran) was ingeniously and elegantly compact. *Fortunately*, about a month later Gene Golub made his first visit to Los Alamos as a consultant. I arranged for Clair to stop by and the three of us discussed this mysterious piece of software/algorithm. It was clearly solving a hierarchy of tridiagonal systems, i.e., it started with a bunch of tridiagonals and reduced them down to a single tridiagonal system then solved it and expanded back out. Gene surmised that this was some sort of cyclic odd–even reduction, with which he had some familiarity.

So over the next couple of months, we worked out what we thought were the details of the algorithm and wrote it up, i.e., our algorithm was similar to, but different from what was in Buneman's subroutine. *Again – fortunately* – we decided that, just to be sure, we should program the odd–even reduction algorithm we had developed and verify that it would do what we intended. It turned out to be catastrophically *Unstable*, so that left us scratching our heads! By that point in time, given what we had done with odd–even reduction, we were able then to decipher the Buneman code, formulate the equations, and prove that the "Buneman stabilization of odd–even reduction" was – in fact – very stable and reliable. Again, to be sure, we programmed and computationally verified these results! Then we incorporated these results into the manuscript. In hindsight, we felt very *fortunate* that we didn't rush to publication with cyclic odd–even reduction and, instead, took our time to test it and eventually unraveled, extended and documented Buneman's scheme.

Gene, thanks again for helping unravel the mystery of the Buneman stablization! And Happy Birthday!

Bill Buzbee
Denver, Colorado, USA

SIAM J. Numer. Anal.
Vol. 7, No. 4, December 1970

ON DIRECT METHODS FOR SOLVING POISSON'S EQUATIONS*

B. L. BUZBEE,† G. H. GOLUB‡ AND C. W. NIELSON†

Abstract. Some efficient and accurate direct methods are developed for solving certain elliptic partial difference equations over a rectangle with Dirichlet, Neumann or periodic boundary conditions. Generalizations to higher dimensions and to L-shaped regions are included.

1. Introduction. In many physical applications, one must solve an $N \times N$ system of linear algebraic equations,

$$Mx = y, \tag{1.1}$$

where M arises from a finite difference approximation to an elliptic partial differential equation. For this reason, the matrix M is sparse and the nonzero elements occur very regularly. As an example, let

$$M = \begin{bmatrix} I & F \\ F^T & I \end{bmatrix}, \tag{1.2}$$

and partition **x** and **y** to conform with M. If one can interchange both the rows and the columns of a matrix so that it has the form (M – I), the matrix is said to be two-cyclic [12]. Expanding (1.1), we have

$$\mathbf{x}_1 + F\mathbf{x}_2 = \mathbf{y}_1,$$
$$F^T\mathbf{x}_1 + \mathbf{x}_2 = \mathbf{y}_2.$$

Multiplying the first equation by $-F^T$ and adding, we have

$$(I - F^TF)\mathbf{x}_2 = \mathbf{y}_2 - F^T\mathbf{y}_1. \tag{1.3}$$

By this simple device, we have reduced the number of equations. If $(I - F^TF)$ is also two-cyclic, we can again eliminate a number of the variables and continue until the resulting matrix is no longer two-cyclic. In fact, F has block structure in many applications, and one has the freedom to specify the number of blocks in it. In these cases, a proper choice of the number of blocks may enable the reduction process to continue until the final $(I - F^TF)$ has block dimension one; the reduction methods to be developed impose such a condition.

Direct methods for solving (1.1) are attractive since in theory they yield the exact solution to the difference equation, whereas commonly used methods seek to approximate the solution by iterative procedures [12]. Based on a suggestion of one of the authors, Hockney [8] has devised an efficient direct method which uses the reduction process. Also Buneman [2] recently developed an efficient direct method for solving the reduced system of equations. Since these methods offer considerable economy over older techniques [5], the purpose of

* Received by the editors June 2, 1969, and in revised form May 20, 1970. This work was supported by the U.S. Atomic Energy Commission.

† University of California, Los Alamos Scientific Laboratory, Los Alamos, New Mexico 87544.

‡ Department of Computer Science, Stanford University, Stanford, California 94305.

this paper is to present a unified mathematical development and generalization ol them. Additional generalizations are given by George [6].

In § 2 we develop the method of matrix decomposition or discrete separation of variables. In § 3 we develop the block-cyclic reduction process and techniques for solving the reduced systems. In §§ 4, 5 and 6 we apply the results of §§ 2 and 3 to Poisson's equation on a rectangle with Dirichlet, Neumann and periodic boundary conditions, respectively. Section 7 extends the results of §§ 4, 5 and 6 to higher dimensions; § 8 extends §§ 2 and 3 to other applications; § 9 extends §§ 2 and 3 to "L-shaped" regions. In § 10, we show that straightforward applications of the results of § 3 can result in severe roundoff error in many applications of interest. In § 11 we develop the Buneman algorithms, which are mathematically equivalent to the reduction process of § 3, but are not subject to severe roundoff. In § 11 we apply the Buneman algorithm to Poisson's equation with Dirichlet, Neumann and periodic boundaries. Finally, in § 12 we show the stability of the Buneman algorithms.

2. Matrix decomposition. Consider the system of equations

$$Mx = y, \tag{2.1}$$

where M is an $N \times N$ real symmetric matrix of block tridiagonal form,

$$M = \begin{bmatrix} A & T & & & \\ T & A & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & T \\ & & & T & A \end{bmatrix}. \tag{2.2}$$

The matrices A and T are $p \times p$ symmetric matrices, and we assume that

$$AT = TA.$$

Such a situation arises for those problems that can be handled by the classical separation-of-variables technique. Indeed, the methods we discuss amount to an efficient computer implementation of the idea of separation of variables carried out on a discretized model of the elliptic differential equation. Since A and T commute and are symmetric, it is well known [1] that there exists an orthogonal matrix Q such that

$$Q^T A Q = \Lambda, \qquad Q^T T Q = \Omega, \tag{2.3}$$

and Λ and Ω are real diagonal matrices. The matrix Q is the set of eigenvectors of A and T, and Λ and Ω are the diagonal matrices of eigenvalues of A and T, respectively.

To conform with the matrix M, we write the vectors $\mathbf{x}$ and $\mathbf{y}$ in partitioned form,

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \hline \mathbf{x}_2 \\ \hline \vdots \\ \hline \mathbf{x}_q \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \hline \mathbf{y}_2 \\ \hline \vdots \\ \hline \mathbf{y}_q \end{bmatrix}.$$

Furthermore, it is quite natural to write

$$\mathbf{x}_j = \begin{bmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{pj} \end{bmatrix}, \quad \mathbf{y}_j = \begin{bmatrix} y_{1j} \\ y_{2j} \\ \vdots \\ y_{pj} \end{bmatrix}. \tag{2.4}$$

System (2.2) may be written

$$A\mathbf{x}_1 + T\mathbf{x}_2 = \mathbf{y}_1, \tag{2.5a}$$

$$T\mathbf{x}_{j-1} + A\mathbf{x}_j + T\mathbf{x}_{j+1} = \mathbf{y}_j, \qquad j = 2, 3, \cdots, q-1, \tag{2.5b}$$

$$T\mathbf{x}_{q-1} + A\mathbf{x}_q = \mathbf{y}_q. \tag{2.5c}$$

Using (2.3), this becomes

$$\begin{aligned} &\Lambda\bar{\mathbf{x}}_1 + \Omega\bar{\mathbf{x}}_2 = \bar{\mathbf{y}}_1, \\ &\Omega\bar{\mathbf{x}}_{j-1} + \Lambda\bar{\mathbf{x}}_j + \Omega\bar{\mathbf{x}}_{j+1} = \bar{\mathbf{y}}_j, \qquad j = 2, 3, \cdots, q-1, \\ &\Omega\bar{\mathbf{x}}_{q-1} + \Lambda\bar{\mathbf{x}}_q = \bar{\mathbf{y}}_q, \end{aligned} \tag{2.6}$$

where

$$\bar{\mathbf{x}}_j = Q^T\mathbf{x}_j, \qquad \bar{\mathbf{y}}_j = Q^T\mathbf{y}_j, \qquad j = 1, 2, \cdots, q.$$

The components of $\bar{\mathbf{x}}_j$ and $\bar{\mathbf{y}}_j$ are labeled as in (2.4). Then (2.6) may be rewritten for $i = 1, 2, \cdots, p$,

$$\begin{aligned} &\lambda_i\bar{x}_{i1} + \omega_i\bar{x}_{i2} = \bar{y}_{i1}, \\ &\omega_i\bar{x}_{ij-1} + \lambda_i\bar{x}_{ij} + \omega_i\bar{x}_{ij+1} = \bar{y}_{ij}, \qquad j = 2, \cdots, q-1, \\ &\omega_i\bar{x}_{iq-1} + \lambda_i\bar{x}_{iq} = \bar{y}_{iq}. \end{aligned} \tag{2.7}$$

Now let us write

$$\Gamma_i = \begin{bmatrix} \lambda_i & \omega_i & & & \\ \omega_i & \lambda_i & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \omega_i \\ & & & \omega_i & \lambda_i \end{bmatrix}_{q \times q},$$

$$\hat{\mathbf{x}}_i = \begin{bmatrix} \bar{x}_{i1} \\ \bar{x}_{i2} \\ \vdots \\ \bar{x}_{iq} \end{bmatrix}, \quad \hat{\mathbf{y}}_i = \begin{bmatrix} \bar{y}_{i1} \\ \bar{y}_{i2} \\ \vdots \\ \bar{y}_{iq} \end{bmatrix},$$

so that (2.7) is equivalent to the system of equations

$$\Gamma_i \hat{\mathbf{x}}_i = \hat{\mathbf{y}}_i. \tag{2.8}$$

Thus, the vector $\hat{\mathbf{x}}_i$ satisfies a symmetric tridiagonal system of equations that has a constant diagonal element and a constant super- and subdiagonal element. After (2.8) has been solved, it is possible to solve for $\mathbf{x}_j = Q\hat{\mathbf{x}}_j$. Thus the algorithm proceeds as follows:

(i) Compute or determine the eigenvectors of A and the eigenvalues of A and T.
(ii) Compute $\bar{\mathbf{y}}_j = Q^T\mathbf{y}_j$, $j = 1, 2, \cdots, q$.
(iii) Solve $\Gamma_i\hat{\mathbf{x}}_i = \hat{\mathbf{y}}_i$, $i = 1, 2, \cdots, p$.
(iv) Compute $\mathbf{x}_j = Q\bar{\mathbf{x}}_j$, $j = 1, 2, \cdots, q$.

Hockney [8] has carefully analyzed this algorithm for solving Poisson's equation in a square. He has taken advantage of the fact that, in this case, the matrix Q is known and that one can use the fast Fourier transform [4] to perform steps (ii) and (iv). Shintani [11] has given methods for solving for the eigenvalues and eigenvectors in a number of special cases.

A and T need not commute. Assume that T is positive definite and symmetric. It is well known [1] that there is a matrix P such that

$$T = PP^T, \qquad A = P\Delta P^T, \tag{2.9}$$

where Δ is the diagonal matrix of eigenvalues of $T^{-1}A$ and P^{-T} is the matrix of eigenvectors of $T^{-1}A$. Thus, using (2.9), we modify the algorithm as follows:

(i) Compute or determine the eigenvalues and eigenvectors of $T^{-1}A$.
(ii) Compute $\bar{\mathbf{y}}_j = P^{-1}\mathbf{y}_j$.
(iii) Solve $\Gamma_i\hat{\mathbf{x}}_i = \hat{\mathbf{y}}_i$, where

$$\Gamma_i = \begin{bmatrix} \delta_i & 1 & & & \\ 1 & \delta_i & \cdot & & \\ & & \cdot & \cdot & \\ & & \cdot & \cdot & 1 \\ & & & 1 & \delta_i \end{bmatrix}.$$

(iv) Compute $\mathbf{x}_j = P^{-T}\bar{\mathbf{x}}_j$.

Of course, one should avoid computing $T^{-1}A$, because this would destroy the sparseness of the matrices. Golub et al [7] has proposed an algorithm for solving $A\mathbf{u} = \delta T\mathbf{u}$ when A and T are sparse.

3. Block-cyclic reduction. In § 2 we gave a method for which one had to know the eigenvalues and eigenvectors of some matrix. We now give a more direct method for solving the system of (2.1).

We assume again that A and T are symmetric and that A and T commute. Furthermore, we assume that $q = m - 1$ and $m = 2^{k+1}$, where k is some positive integer. Let us rewrite (2.5b) as follows:

$$\begin{aligned} T\mathbf{x}_{j-2} + A\mathbf{x}_{j-1} + T\mathbf{x}_j \qquad\qquad\qquad &= \mathbf{y}_{j-1}, \\ T\mathbf{x}_{j-1} + A\mathbf{x}_j + T\mathbf{x}_{j+1} \qquad\quad &= \mathbf{y}_j, \\ T\mathbf{x}_j + A\mathbf{x}_{j+1} + T\mathbf{x}_{j+2} &= \mathbf{y}_{j+1}. \end{aligned}$$

Multiplying the first and third equation by T, the second equation by $-A$ and adding, we have

$$T^2\mathbf{x}_{j-2} + (2T^2 - A^2)\mathbf{x}_j + T^2\mathbf{x}_{j+2} = T\mathbf{y}_{j-1} - A\mathbf{y}_j + T\mathbf{y}_{j+1}.$$

Thus, if j is even, the new system of equations involves $\mathbf{x}_j$'s with even indices. Similar equations hold for $\mathbf{x}_2$ and $\mathbf{x}_{m-2}$. The process of reducing the equations in this fashion is known as *cyclic reduction*. Then (2.1) may be written as the following equivalent system:

$$\begin{bmatrix} (2T^2 - A^2) & T^2 & & & 0 \\ T^2 & (2T^2 - A^2) & T^2 & & \\ & & \cdot & \cdot & T^2 \\ 0 & & \cdot & \cdot & \\ & & & T^2 & (2T^2 - A^2) \end{bmatrix} \begin{bmatrix} \mathbf{x}_2 \\ \mathbf{x}_4 \\ \vdots \\ \\ \mathbf{x}_{m-2} \end{bmatrix}$$

$$(3.1) \qquad = \begin{bmatrix} T\mathbf{y}_1 + T\mathbf{y}_3 & - A\mathbf{y}_2 \\ T\mathbf{y}_3 + T\mathbf{y}_5 & - A\mathbf{y}_4 \\ \vdots & \\ T\mathbf{y}_{m-1} + T\mathbf{y}_{m-3} & - A\mathbf{y}_{m-2} \end{bmatrix}$$

and

$$(3.2) \qquad \begin{bmatrix} A & 0 & & 0 \\ 0 & A & \cdot & \\ & \cdot & \cdot & \cdot \\ & \cdot & \cdot & 0 \\ 0 & & 0 & A \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_3 \\ \\ \cdot \\ \mathbf{x}_{m-1} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 - T\mathbf{x}_2 \\ \mathbf{y}_3 - T\mathbf{x}_2 - T\mathbf{x}_4 \\ \vdots \\ \mathbf{y}_m - T\mathbf{x}_{m-2} \end{bmatrix}.$$

Since $m = 2^{k+1}$ and the new system of (3.1) involves $\mathbf{x}_j$'s with even indices, the block dimension of the new system of equations is $2^k - 1$. Note that once (3.1) is solved, it is easy to solve for the $\mathbf{x}_j$'s with odd indices, as evidenced by (3.2). We shall refer to the system of (3.2) as the *eliminated equations*.

Also, note that the algorithm of § 2 may be applied to system (3.1). Since A and T commute, the matrix $(2T^2 - A^2)$ has the same set of eigenvectors as A and T. Also, if

$$\lambda(A) = \lambda_i, \; \lambda(T) = \omega_i,$$

$$\lambda(2T^2 - A^2) = 2\omega_i^2 - \lambda_i^2 \quad \text{for } i = 1, 2, \cdots, m-1.$$

Hockney [8] has advocated this procedure.

Since system (3.1) is block tridiagonal and of the form of (2.2), we can apply the reduction repeatedly until we have one block. However, as noted above, we can stop the process after any step and use the methods of § 2 to solve the resulting equations.

To define the procedure recursively, let

$$A^{(0)} = A, \quad T^{(0)} = T; \quad \mathbf{y}_j^{(0)} = \mathbf{y}_j, \qquad j = 1, 2, \cdots, m - .$$

Then for $r = 0, 1, \cdots, k$,

$$\begin{aligned} &A^{(r+1)} = 2(T^{(r)})^2 - (A^{(r)})^2, \\ (3.3) \qquad &T^{(r+1)} = (T^{(r)})^2, \\ &\mathbf{y}_j^{(r-1)} = T^{(r)}(\mathbf{y}_{j-2^r}^{(r)} + \mathbf{y}_{j+2^r}^{(r)}) - A^{(r)}\mathbf{y}_j^{(r)}. \end{aligned}$$

At each stage, we have a new system of equations

$$M^{(r)}\mathbf{z}^{(r)} = \mathbf{f}^{(r)}$$

to solve, where

$$M^{(r)} = \begin{bmatrix} A^{(r)} & T^{(r)} & & & 0 \\ T^{(r)} & A^{(r)} & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & T^{(r)} \\ & 0 & & T^{(r)} & A^{(r)} \end{bmatrix},$$

(3.4)

$$\mathbf{z}^{(r)} = \begin{bmatrix} \mathbf{x}_{2^r} \\ \mathbf{x}_{2^{r+1}} \\ \vdots \\ \mathbf{x}_{j2^r} \\ \vdots \end{bmatrix}, \qquad \mathbf{f}^{(r)} = \begin{bmatrix} \mathbf{y}_{2^r}^{(r)} \\ \mathbf{y}_{2^r}^{(r)} \\ \vdots \\ \mathbf{y}_{j2^r}^{(r)} \\ \vdots \end{bmatrix}.$$

The eliminated equations are the solution of the block diagonal system

$$(3.5) \qquad N^{(r)}\mathbf{w}^{(r)} = \mathbf{g}^{(r)},$$

where

$$N^{(r)} = \begin{bmatrix} A^{(r-1)} & 0 & & & 0 \\ 0 & A^{(r-1)} & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & 0 \\ & 0 & & 0 & A^{(r-1)} \end{bmatrix},$$

$$\mathbf{w}^{(r)} = \begin{bmatrix} \mathbf{x}_{2^r - 2^{r-1}} \\ \mathbf{x}_{2^{r+1} - 2^{r-1}} \\ \vdots \\ \mathbf{x}_{j2^r - 2^{r-1}} \\ \vdots \end{bmatrix},$$

$$\mathbf{g}^{(r)} = \begin{bmatrix} \mathbf{y}^{(r-1)}_{2^r - 2^{r-1}} & -T\mathbf{x}^{(r-1)}_{2^r} \\ \mathbf{y}^{(r-1)}_{2^{r+1} - 2^{r-1}} & -T(\mathbf{x}^{(r-1)}_{2^{r+1}} + \mathbf{x}^{(r-1)}_{2^r}) \\ & \vdots \\ \mathbf{y}^{(r-1)}_{j2^r - 2^{r-1}} & -T(\mathbf{x}^{(r-1)}_{j2^r} + \mathbf{x}_{(j-1)2^r}) \\ & \vdots \end{bmatrix}.$$

We can use the methods of § 2 to solve the system $M^{(r)}\mathbf{z}^{(r)} = \mathbf{f}^{(r)}$, or we can proceed to compute $M^{(r+1)}$ and eliminate half of the unknowns. After k steps, we must solve the system of equations

$$A^{(k)}\mathbf{x}_{2^k} = \mathbf{y}^{(k)}_{2^k}. \tag{3.6}$$

In either case, we must solve (3.5) to find the eliminated unknowns, just as in (3.2). This can be done by

(i) direct solution,
(ii) eigenvalue–eigenvector factorization, or
(iii) polynomial factorization.

The direct solution is attractive when k is small. One can form the powers of A and T quite easily and solve the resulting equations by Gaussian elimination. Thus, if $k = 1$, and A and T are tridiagonal matrices, $A^{(1)}$ is a five diagonal matrix, and for such band matrices it is easy to solve the resulting system of equations.

It is possible to compute the eigenvalue–eigenvector decomposition of $A^{(r)}$ and $T^{(r)}$. Since $A^{(0)} = Q\Lambda Q^T$ and $T^{(0)} = Q\Omega Q^T$, we may write

$$A^{(r)} = Q\Lambda^{(r)}Q^T \quad \text{and} \quad T^{(r)} = Q\Omega^{(r)}Q^T.$$

From (3.3), it follows that

$$\Lambda^{(r+1)} = 2(\Omega^{(r)})^2 - (\Lambda^{(r)})^2,$$
$$\Omega^{(r+1)} = (\Omega^{(r)})^2.$$

Thus the eigenvalues of $A^{(r)}$ and $T^{(r)}$ can be generated by the simple rule

$$\lambda_i^{(r+1)} = 2(\omega_i^{(r)})^2 - (\lambda_i^{(r)})^2, \qquad \lambda_i^{(0)} = \lambda_i,$$
$$\omega_i^{(r+1)} = 2(\omega_i^{(r)})^2, \qquad \omega_i^{(0)} = \omega_i, \qquad i = 1, 2, \cdots, m-1.$$

Hence, the methods of § 2 can easily be applied to solving the system $M^{(r)}\mathbf{z}^{(r)} = \mathbf{f}^{(r)}$ and $N^{(r)}\mathbf{w}^{(r)} = \mathbf{g}^{(r)}$. Hockney [9] has described this algorithm as the FACR(l) algorithm where l refers to the number of cyclic reductions performed.

From (3.1), we note that $A^{(1)}$ is a polynomial of degree 2 in A and T. By induction, it is easy to show that $A^{(r)}$ is a polynomial of degree 2^r in the matrices A and T, so that

$$A^{(r)} = \sum_{j=0}^{2^{r-1}} c_{2j}^{(r)} A^{2j} T^{2^r - 2j} \equiv P_{2^r}(A, T).$$

We shall proceed to determine the linear factors of $P_{2^r}(A, T)$.

Let

$$p_{2^r}(a, t) = \sum_{j=0}^{2^{r-1}} c_{2j}^{(r)} a^{2j} t^{2^r - 2j}, \qquad c_{2^r}^{(r)} = -1.$$

For $t \neq 0$, we make the substitution

$$a/t = -2\cos\theta. \tag{3.7}$$

From (3.3), we note that

$$p_{2^{r+1}}(a, t) = 2t^{2^{r+1}} - (p_{2^r}(a, t))^2. \tag{3.8}$$

It is then easy to verify, using (3.7) and (3.8), that

$$p_{2^r}(a, t) = -2t^{2^r} \cos 2^r\theta,$$

and, consequently,

$$p_{2^r}(a, t) = 0 \quad \text{when } a/(2t) = -\cos\left(\frac{2j-1}{2^{r+1}}\right)\pi \quad \text{for } j = 1, 2, \cdots, 2^r.$$

Thus, we may write

$$p_{2^r}(a, t) = -\prod_{j=1}^{2^r}\left[a + 2t\cos\left(\frac{2j-1}{2^{r+1}}\right)\pi\right]$$

and hence

$$A^{(r)} = -\prod_{j=1}^{2^r}(A + 2\cos\theta_j^{(r)} T),$$

where $\theta_j^{(r)} = (2j-1)\pi/2^{r+1}$.

Let us write

$$G_j^{(k)} = A + 2\cos\theta_j^{(k)} T.$$

Then, to solve (3.6), we set $\mathbf{z}_1 = -\mathbf{y}_{2^k}^{(k)}$ and repeatedly solve

$$G_j^{(k)}\mathbf{z}_{j+1} = \mathbf{z}_j \quad \text{for } j = 1, 2, \cdots, 2^k. \tag{3.9}$$

Thus,

$$\mathbf{z}_{2^k+1} = \mathbf{x}_{2^k}.$$

If A and T are of band structure, it is simple to solve (3.9). To determine the solution to the eliminated (3.5), a similar algorithm may be used with

$$A^{(r)} = - \prod_{j=1}^{2^r} G_j^{(r)}. \tag{3.10}$$

The factorization for $A^{(r)}$ may also be used to compute $\mathbf{y}_j^{(r+1)}$ in (3.3). It is possible, however, to take advantage of the recursive nature of the polynomials $p_{2^r}(a, t)$. Let

$$p_s(a,t) = -2t^s \cos s\theta,$$

where, again, for $t \neq 0$, $a/t = -2\cos\theta$. Then a short manipulation shows that

$$p_s(a,t) = -ap_{s-1}(a,t) - t^2 p_{s-2}(a,t), \qquad s \geqq 2,$$
$$p_0(a,t) = -2, \qquad p_1(a,t) = a.$$

Therefore, to compute $A^{(r)}\mathbf{y}_j^{(r)}$ as in (3.3), we compute the following sequence:

$$\boldsymbol{\eta}_0 = 2\mathbf{y}_j^{(r)}, \qquad \boldsymbol{\eta}_1 = A\mathbf{y}_j^{(r)},$$
$$\boldsymbol{\eta}_s = -A\boldsymbol{\eta}_{s-1} - T^2\boldsymbol{\eta}_{s-2} \qquad \text{for } s = 2, 3, \cdots, 2^r.$$

Thus,

$$\boldsymbol{\eta}_{2^r} = P_{2^r}(A, T)\mathbf{y}_j^{(r)} \equiv A^{(r)}\mathbf{y}_j^{(r)}.$$

We call this method the *cyclic odd-even reduction and factorization* (CORF) *algorithm*. In § 10 we will show that the numerical calculation of $\mathbf{y}_j^{(r)}$ in (3.3) is subject to severe rounding errors in many cases of interest. Consequently, numerical application of the results of this section must be accompanied by close attention to the results of § 10. In fact, from a computational viewpoint, the CORF algorithm, as developed here, is virtually useless; however, the theoretical results of this section are necessary for the development of the stable, Buneman variants of CORF.

4. Poisson's equation with Dirichlet boundary conditions. It is instructive to apply the results of § 3 to the solution of the finite difference approximation to Poisson's equation on a rectangle, R, with specified boundary values. Consider the equation

$$\begin{aligned} &u_{xx} + u_{yy} = f(x,y) && \text{for } (x,y) \in R, \\ &u(x,y) = g(x,y) && \text{for } (x,y) \in \partial R. \end{aligned} \tag{4.1}$$

(Here ∂R indicates the boundary of R.) We assume that the reader is familiar with the general technique of imposing a mesh of discrete points onto R and approximating (4.1). The equation $u_{xx} + u_{yy} = f(x,y)$ is approximated at (x_i, y_j) by

$$\frac{v_{i-1,j} - 2v_{i,j} + v_{i+1,j}}{(\Delta x)^2} + \frac{v_{i,j-1} - 2v_{i,j} + v_{i,j+1}}{(\Delta y)^2} = f_{i,j},$$
$$1 \leqq i \leqq n-1, \quad 1 \leqq j \leqq m-1,$$

with

$$v_{0,j} = g_{0,j}, \qquad v_{n,j} = g_{n,j}, \qquad 1 \leqq j \leqq m-1,$$

and

$$v_{i,0} = g_{i,0}, \qquad v_{i,m} = g_{i,m}, \qquad 1 \leqq i \leqq n-1.$$

Then v_{ij} is an approximation to $u(x_i, y_j)$, and $f_{i,j} = f(x_i, y_j)$, $g_{i,j} = g(x_i, y_j)$. Hereafter, we assume that $m = 2^{k+1}$.

When $u(x, y)$ is specified on the boundary, we have the Dirichlet boundary condition. For simplicity, we shall assume hereafter that $\Delta x = \Delta y$. This leads to the system of equations

$$M_D \mathbf{v} = \mathbf{y},$$

where M_D is of the form of (2.1) with

$$A = \begin{bmatrix} -4 & 1 & & & 0 \\ 1 & -4 & 1 & & \\ & \cdot & \cdot & \cdot & \cdot \\ & 0 & \cdot & \cdot & 1 \\ & & & 1 & -4 \end{bmatrix}_{(n-1)\times(n-1)}$$

and

$$T = I_{n-1}.$$

The matrix I_{n-1} indicates the identity matrix of order $(n-1)$. A and T are symmetric, and commute, and thus, the results of §§ 2 and 3 are applicable. In addition, since A is tridiagonal, the use of the factorization (3.10) is greatly simplified.

5. Neumann boundary conditions. When the normal derivative, $\partial u/\partial n$, is specified on the boundary, we have the Neumann boundary condition. Assume that

$$\frac{\partial u}{\partial n} = g(x, y) \quad \text{when} \quad (x, y) \in R.$$

We make the approximation

$$\frac{\partial u}{\partial x} \doteq \frac{u(x + \Delta x, y) - u(x - \Delta x, y)}{2\Delta x}, \qquad \frac{\partial u}{\partial y} \doteq \frac{u(x, y + \Delta y) - u(x, y - \Delta y)}{2\Delta y}.$$

This approximation leads to the matrix equation

$$M_N \mathbf{v} = \mathbf{y},$$

where M_N is of the form

$$\begin{bmatrix} A & 2T & & & 0 \\ T & A & T & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & 0 & T & \cdot & T \\ & & & 2T & A \end{bmatrix}. \tag{5.1}$$

Here,

$$A = \begin{bmatrix} -4 & 2 & & & & \\ 1 & -4 & 1 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & & 1 & -4 & 1 \\ & 0 & & & 2 & -4 \end{bmatrix}_{(n+1)\times(n+1)}, \qquad T = I_{n+1}.$$

Again A and T commute, but M_N no longer has the structure given by (2.2). Therefore it is necessary to modify the algorithm of § 3.

From (5.1), we see that

$$\begin{aligned} A\mathbf{v}_0 + 2T\mathbf{v}_1 &= \mathbf{y}_0, \\ T\mathbf{v}_{j-1} + A\mathbf{v}_j + T\mathbf{v}_{j+1} &= \mathbf{y}_j, \qquad j = 1, 2, \cdots, m-1, \\ 2T\mathbf{v}_{m-1} + A\mathbf{v}_m &= \mathbf{y}_m. \end{aligned}$$

Performing the cyclic reduction as in § 3, we have

$$\begin{aligned} (2T^2 - A^2)\mathbf{v}_0 + 2T\mathbf{v}_2 &= -A\mathbf{y}_0 + 2T\mathbf{y}_1, \\ T^2\mathbf{v}_{j-2} + (2T^2 - A^2)\mathbf{v}_j + T^2\mathbf{v}_{j+2} &= T(\mathbf{y}_{j-1} + \mathbf{y}_{j+1}) - A\mathbf{y}_j, \\ & \qquad j = 2, 4, \cdots, m-2, \\ 2T\mathbf{v}_{m-2} + (2T^2 - A^2)\mathbf{v}_m &= 2T\mathbf{y}_{m-1} - A\mathbf{y}_m. \end{aligned} \tag{5.2}$$

The similarity of (5.2) to (3.1) should now be evident. Since (5.2) is of block dimension $2^k + 1$, after k steps we have the system

$$\begin{bmatrix} A^{(k)} & 2T^{(k)} & 0 \\ T^{(k)} & A^{(k)} & T^{(k)} \\ 0 & 2T^{(k)} & A^{(k)} \end{bmatrix} \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_{2^k} \\ \mathbf{v}_{2^{k+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_0^{(k)} \\ \mathbf{y}_{2^k}^{(k)} \\ \mathbf{y}_{2^{k+1}}^{(k)} \end{bmatrix}, \tag{5.3}$$

and a final reduction yields

$$[4(T^{(k)})^2 - (A^{(k)})^2]\mathbf{v}_{2^k} = T(\mathbf{y}_0^{(k)} + \mathbf{y}_{2^{k+1}}^{(k)}) - A\mathbf{y}_{2^k}^{(k)}. \tag{5.4}$$

Equation (5.4) is equivalent to writing

$$P_{2^{k+1}}^{(N)}(A, T)\mathbf{v}_{2^k} = \mathbf{y}_{2^k}^{(k+1)}, \tag{5.5}$$

where $P_{2^{k+1}}^{(N)}(A, T)$ is again a polynomial of degree 2^{k+1} in A and T. Note that from (3.8),

$$p_{2^{r+1}}^{(N)}(a, t) = 2t^{2^{r+1}} - [p_{2^r}^{(N)}(a, t)]^2, \qquad r = 0, 1, \cdots, k-1,$$

and from (5.4),

$$p_{2^{k+1}}^{(N)}(a, t) = 4t^{2^{k+1}} - [p_{2^k}^{(N)}(a, t)]^2.$$

Therefore, since $p_{2^k}(a, t) = -2t^{2^k} \cos 2^k\theta$,

$$p_{2^{k+1}}^{(N)}(a, t) = (2t^{2^k} \sin 2^k\theta)^2,$$

and, thus,

$$p^{(N)}_{2^{k+1}}(a, t) = 0 \quad \text{when } a/2t = -\cos\frac{j\pi}{2^k} \quad \text{for } j = 1, 2, \cdots, 2^{k+1}.$$

Consequently, we may rewrite (5.5) as

$$\prod_{j=1}^{2^{k+1}} [A + 2\cos\theta_j^{(k+1)}T]\mathbf{v}_{2^k} = -\mathbf{y}_{2^k}^{(k+1)},$$

where $\theta_j^{(k+1)} = j\pi/2^k$. Again, $\mathbf{v}_{2^k}$ is determined by solving 2^{k+1} tridiagonal systems. The other components of $\mathbf{v}$ are solved in the manner indicated in § 3.

Note that the application of matrix decomposition to this problem only requires the modification of the tridiagonal matrices Γ_i in (2.8); i.e., the first superdiagonal and the last subdiagonal elements must be $2\omega_i$.

It is well known that the solution to Poisson's equation is not unique in this case. Therefore, we would expect the finite difference approximation to be singular. This is easy to verify by noting that

$$M_N\mathbf{e} = \mathbf{0},$$

where $\mathbf{e}^T = (1, 1, \cdots, 1)$. In addition, one of the systems of the tridiagonal matrices in (5.6) is also singular. On a uniform mesh, the eigenvalues of $(A + 2\cos\theta_j T)$ satisfy the equation

$$\lambda_l(A + 2\cos\theta_j T) = 4 - 2\cos\left(\frac{l\pi}{n}\right) + 2\cos\left(\frac{j\pi}{2^k}\right),$$

$$l = 0, 1, 2, \cdots, n, \quad j = 1, 2, \cdots, 2^{k+1}.$$

Then $\lambda_0 = 0$ when $j = 2^k$. Similarly, one can show that Γ_0 of matrix decomposition is singular. Normally, the physics of the problem determines the coefficient of the homogeneous solution for the singular case.

6. Periodic boundary conditions. In this section, we shall consider the problem of solving the finite difference approximation to Poisson's equation over a rectangle when

$$\begin{aligned} u(x_0, y) &= u(x_n, y), \\ u(x, y_0) &= u(x, y_n). \end{aligned} \tag{6.1}$$

The periodic boundary conditions (6.1) lead to the matrix equation

$$M_P\mathbf{v} = \mathbf{y}, \tag{6.2}$$

where

$$M_P = \begin{bmatrix} A & T & 0 & \cdot & \cdot & 0 & T \\ T & A & T & \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & \cdot & \cdot & & & \cdot \\ & & & & & 0 & \\ \cdot & & \cdot & \cdot & \cdot & & \cdot \\ & 0 & & & & & \\ \cdot & & & \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & & & T & A & T \\ T & 0 & \cdot & \cdot & 0 & T & A \end{bmatrix}$$

and

$$A = \begin{bmatrix} -4 & 1 & 0 & \cdot & \cdot & 0 & 1 \\ 1 & -4 & 1 & 0 & \cdot & \cdot & 0 \\ 0 & \cdot & \cdot & \cdot & & & \cdot \\ & & & & & 0 & \\ \cdot & & \cdot & \cdot & \cdot & & \cdot \\ & 0 & & & & & \\ \cdot & & & \cdot & \cdot & \cdot & 0 \\ 0 & & & & 1 & -4 & 1 \\ 1 & 0 & \cdot & \cdot & 0 & 1 & -4 \end{bmatrix}_{n \times n}, \qquad T = I_n.$$

Note that M_P is "almost" an m block tridiagonal system and, similarly, A is "almost" an $n \times n$ tridiagonal matrix. The cyclic reduction can again be performed on (6.2), and this leads to the reduced system

$$\begin{aligned} &(2T^2 - A^2)\mathbf{v}_2 + T^2\mathbf{v}_4 + T^2\mathbf{v}_m = T(\mathbf{y}_1 + \mathbf{y}_3) - A\mathbf{y}_2, \\ (6.3) \qquad &T^2\mathbf{v}_{j-2} + (2T^2 - A^2)\mathbf{v}_j + T^2\mathbf{v}_{j+2} = T(\mathbf{y}_{j-1} + \mathbf{y}_{j+1}) - A\mathbf{y}_j, \\ &\qquad\qquad\qquad\qquad\qquad\qquad j = 2, 4, \cdots, m-2, \\ &T^2\mathbf{v}_2 + T^2\mathbf{v}_{m-2} + (2T^2 - A^2)\mathbf{v}_m = T(\mathbf{y}_1 + \mathbf{y}_{m-1}) - A\mathbf{y}_m. \end{aligned}$$

The similarity with the previous cases is again evident. Equation (6.3) has block dimension 2^k. After $(k-1)$ reductions, we have

$$M_P^{(k-1)} = \begin{bmatrix} A^{(k-1)} & T^{(k-1)} & 0 & T^{(k-1)} \\ T^{(k-1)} & A^{(k-1)} & T^{(k-1)} & 0 \\ 0 & T^{(k-1)} & A^{(k-1)} & T^{(k-1)} \\ T^{(k-1)} & 0 & T^{(k-1)} & A^{(k-1)} \end{bmatrix}$$

and finally, after k reductions,

$$(6.4) \qquad \begin{bmatrix} A^{(k)} & 2T^{(k)} \\ 2T^{(k)} & A^{(k)} \end{bmatrix} \begin{bmatrix} \mathbf{v}_{2^k} \\ \mathbf{v}_{2^{k+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_{2^k}^{(k)} \\ \mathbf{y}_{2^{k+1}}^{(k)} \end{bmatrix}.$$

From (6.4), the final equation becomes

$$[4T^{(k)2} - A^{(k)2}]\mathbf{v}_{2^k} = \mathbf{y}_{2^k}^{(k+1)},$$

which is equivalent to

$$P_{2^{k+1}}^{(N)}(A, T)\mathbf{v}_{2^k} = \mathbf{y}_{2^k}^{(k+1)}.$$

The analysis of the factorization of $P_{2^{k+1}}^{(N)}(A, T)$ is identical to that of the Neumann case, including the fact that one of the factors of the polynomial must be singular.

Again, the application of matrix decomposition to (6.2) is straightforward; however, the resulting Γ_i matrices are no longer tridiagonal, since ω_i appears in the $(1, n)$ and $(n, 1)$ elements. Standard algorithms for solving tridiagonal systems can be modified to solve these systems such that storage of the full $n \times n$ matrix is avoided. As above, one of the Γ_i will be singular.

7. Higher-dimensional problems. It is not difficult to extend the applications given in §§ 4, 5 and 6 to higher-dimensional problems. We show this by a simple example. Consider Poisson's equation in three dimensions over the rectangle R:

$$u_{xx} + u_{yy} + u_{zz} = f(x, y, z), \qquad (x, y, z) \in R,$$
$$u(x, y, z) = g(x, y, z), \qquad (x, y, z) \in \partial R.$$

Again, we assume that the mesh is uniform in each direction, so that

$$\begin{aligned} x_{i+1} &= x_i + \Delta x, & i &= 0, 1, \cdots, n, \\ y_{j+1} &= y_j + \Delta y, & j &= 0, 1, \cdots, m, \\ z_{l+1} &= z_l + \Delta z, & l &= 0, 1, \cdots, p. \end{aligned}$$

At the point (x_i, y_j, z_l), we approximate $u(x_i, y_j, z_l)$ by $v_{i,j,l}$. Let

$$\mathbf{w}_l = \begin{bmatrix} \mathbf{v}_{1,l} \\ \mathbf{v}_{2,l} \\ \vdots \\ \mathbf{v}_{m-1,l} \end{bmatrix}, \quad \text{where} \quad \mathbf{v}_{j,l} = \begin{bmatrix} v_{1,j,l} \\ v_{2,j,l} \\ \vdots \\ v_{n-1,j,l} \end{bmatrix}.$$

Assume that the usual finite difference approximation is made to u_{zz} for fixed (x, y, z), namely,

$$u_{zz}(x, y, z) \doteq \frac{u(x, y, z - \Delta z) - 2u(x, y, z) + u(x, y, z + \Delta z)}{(\Delta z)^2}.$$

It is then easy to verify that for $l = 1, 2, \cdots, p - 1$,

$$\mathbf{w}_{l-1} + H\mathbf{w}_l + \mathbf{w}_{l+1} = \mathbf{f}_l,$$

where $\mathbf{w}_0$ and $\mathbf{w}_p$ are prescribed by the initial conditions and $\mathbf{f}_l$ is a function of the given data. Thus, again, we have a block tridiagonal matrix, and can use the previously developed methods. Note, also, that H is a block tridiagonal matrix so that one can solve any of the eliminated systems of equations by applying the CORF algorithm repeatedly. Other boundary conditions can be handled in the manner prescribed in §§ 5 and 6.

8. Further applications. Consider the elliptic equation in self-adjoint form

$$(8.1)\qquad \begin{aligned} (\alpha(x)u_x)_x + (\beta(y)u_y)_y + u(x,y) &= q(x,y), \qquad (x,y) \in R,\\ u(x,y) &= g(x,y), \qquad (x,y) \in \partial R. \end{aligned}$$

Many equations can be transformed to this form. The usual five-point difference equation, when $\Delta x = \Delta y$, leads to the following equation:

$$(8.2)\qquad \begin{aligned} &-\alpha_{i+1/2}v_{i+1,j} - \alpha_{i-1/2}v_{i-1,j} - \beta_{j+1/2}v_{i,j+1} - \beta_{j-1/2}v_{i,j-1}\\ &\quad + [\alpha_{i+1/2} + \alpha_{i-1/2} + \beta_{j+1/2} + \beta_{j-1/2} - (\Delta x)^2]v_{i,j} = -(\Delta x)^2 q_{i,j}, \end{aligned}$$

where

$$\alpha_{i\pm 1/2} = \alpha(x_i \pm \Delta x/2), \qquad \beta_{j\pm 1/2} = \beta(y_j \pm \Delta y/2),$$
$$q_{i,j} = q(x_i, y_j).$$

If the equations are ordered with

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_{m-1} \end{bmatrix}, \qquad \mathbf{v}_j = \begin{bmatrix} v_{1,j} \\ v_{2,j} \\ \vdots \\ v_{n-1,j} \end{bmatrix},$$

the linear system of equations $M\mathbf{v} = \mathbf{d}$ will have the block form

$$\begin{bmatrix} A_1 & T_1 & & 0 \\ T_1 & A_2 & T_2 & \\ & \cdot & \cdot & \cdot \\ & \cdot & \cdot & T_{m-2} \\ 0 & & T_{m-2} & A_{m-1} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_{m-1} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \\ \vdots \\ \mathbf{d}_{m-1} \end{bmatrix}.$$

Here

$$A_j = [\beta_{j+1/2} + \beta_{j-1/2} - (\Delta x)^2]I$$
$$+ \begin{bmatrix} \alpha_{1/2} + \alpha_{3/2} & -\alpha_{3/2} & & & 0 \\ -\alpha_{3/2} & \alpha_{3/2} + \alpha_{5/2} & -\alpha_{5/2} & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & -\alpha_{n-3/2} \\ 0 & & & -\alpha_{n-3/2} & \alpha_{n-3/2} + \alpha_{n-1/2} \end{bmatrix}$$
$$\equiv \gamma_j I + C, \qquad T_j = \beta_{j+1/2} I.$$

Now suppose we have the decomposition

$$Q^T C Q = \Phi,$$

where $Q^TQ = I$ and $\operatorname{diag}(\Phi) = (\varphi_1, \varphi_2, \cdots, \varphi_{n-1})$. Thus,

$$\lambda_i(A_j) = \gamma_j + \varphi_i \quad (i = 1, 2, \cdots, n-1)$$
$$\equiv \lambda_{i,j}.$$

As in § 2, we define

$$\bar{\mathbf{v}}_j = Q^T\mathbf{v}_j, \qquad \bar{\mathbf{d}}_j = Q^T\mathbf{d}_j,$$

and after a suitable permutation we are led to the equations

$$\Gamma_i\hat{\mathbf{v}}_i = \hat{\mathbf{d}}_i, \qquad i = 1, 2, \cdots, n-1,$$

where

$$\Gamma_i = \begin{bmatrix} \lambda_{i,1} & \beta_{3/2} & & & 0 \\ \beta_{3/2} & \lambda_{i,2} & \beta_{5/2} & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & 0 & & \cdot \quad \cdot & \beta_{m-3/2} \\ & & & \beta_{m-3/2} & \lambda_{i,m-1} \end{bmatrix},$$

$$\hat{\mathbf{v}}_i = \begin{bmatrix} \bar{v}_{i1} \\ \bar{v}_{i2} \\ \vdots \\ \bar{v}_{i,m-1} \end{bmatrix}, \qquad \hat{\mathbf{d}}_i = \begin{bmatrix} \bar{d}_{i1} \\ \bar{d}_{i2} \\ \vdots \\ \bar{d}_{i,m-1} \end{bmatrix}.$$

Thus, the vector $\hat{\mathbf{v}}_i$ satisfies a symmetric tridiagonal system of equations. Again, once $\hat{\mathbf{v}}_i$ is computed for all i, it is possible to compute $\hat{\mathbf{v}}$.

Lynch et al. [10] have considered similar methods, but their algorithm requires more operations. Unfortunately, it does not seem possible to use the methods of § 3 on (8.2) in this situation.

Now we may write the equivalent to Poisson's equation in two dimensions in cylindrical coordinates as follows:

$$(ru_r)_r + r^{-1}u_{\theta\theta} = s(r, \theta),$$

and

$$(ru_r)_r + ru_{zz} = t(r, z).$$

The matrix A will still be tridiagonal, and T will be a diagonal matrix with positive diagonal elements. We may make the transformation $\hat{\mathbf{v}}_j = T^{1/2}\hat{\mathbf{v}}_j$, and are thus led to the equations

$$\hat{\mathbf{v}}_{j-1} + T^{-1/2}AT^{-1/2}\hat{\mathbf{v}}_j + \hat{\mathbf{v}}_{j+1} = T^{-1/2}\mathbf{d}_j.$$

Thus, by ordering the equations correctly and by making a simple transformation, one can apply the cyclic reduction and the CORF algorithm to solve the finite difference approximation to Poisson's equation in cylindrical coordinates.

Another situation in which the methods of §§ 2 and 3 are applicable is in using the nine-point formula to solve the finite difference approximation to Poisson's equation in the rectangle. In this case, when $\Delta x = \Delta y$,

$$A = \begin{bmatrix} -20 & 4 & & & 0 \\ 4 & -20 & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & 4 \\ 0 & & & 4 & -20 \end{bmatrix}_{(n-1)\times(n-1)}$$

$$T = \begin{bmatrix} 4 & 1 & & 0 \\ 1 & 4 & \cdot & \\ & \cdot & \cdot & \cdot \\ & & \cdot & 1 \\ 0 & & 1 & 4 \end{bmatrix}_{(n-1)\times(n-1)}$$

It is easy to verify that $AT = TA$ and that the eigenvalues of A and T are

$$\lambda_i(A) = -20 + 8\cos\frac{i\pi}{n}, \qquad i = 1, 2, \cdots, n-1,$$

$$\lambda_i(T) = 4 + 2\cos\frac{i\pi}{n}, \qquad i = 1, 2, \cdots, n-1.$$

Because of the structure of A and T, the fast Fourier transform may be employed when using the methods of § 2.

9. Nonrectangular regions. In many situations, one wishes to solve an elliptic equation over the region in Fig. 1.

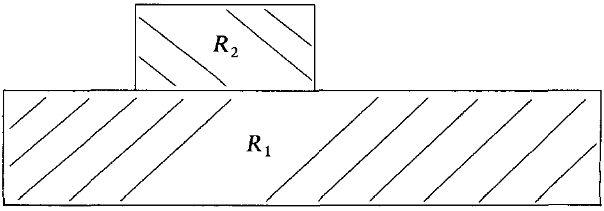


FIG. 1

We shall assume that Dirichlet boundary conditions are given. When Δx is the same throughout the region, one has a matrix equation of the form

$$\text{(9.1)} \qquad \left[\begin{array}{c|c} G & {}_P 0 \\ \hline 0^{P^T} & H \end{array}\right] \left[\begin{array}{c} \mathbf{x}^{(1)} \\ \hline \mathbf{x}^{(2)} \end{array}\right] = \left[\begin{array}{c} \mathbf{y}^{(1)} \\ \hline \mathbf{y}^{(2)} \end{array}\right],$$

where

$$G = \begin{bmatrix} A & T & & & 0 \\ T & A & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & T \\ 0 & & & T & A \end{bmatrix}, \qquad H = \begin{bmatrix} B & S & & & 0 \\ S & B & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & S \\ 0 & & & S & B \end{bmatrix}. \tag{9.2}$$

Also, we write

$$\mathbf{x}^{(1)} = \begin{bmatrix} \mathbf{x}_1^{(1)} \\ \mathbf{x}_2^{(1)} \\ \vdots \\ \mathbf{x}_r^{(1)} \end{bmatrix}, \qquad \mathbf{x}^{(2)} = \begin{bmatrix} \mathbf{x}_1^{(2)} \\ \mathbf{x}_2^{(2)} \\ \vdots \\ \mathbf{x}_s^{(2)} \end{bmatrix}. \tag{9.3}$$

We assume again that $AT = TA$ and $BS = SB$.

From (9.1), we see that

$$\mathbf{x}^{(1)} = G^{-1}\mathbf{y}^{(1)} - G^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ P \end{bmatrix} \mathbf{x}_1^{(2)}, \tag{9.4}$$

and

$$\mathbf{x}^{(2)} = H^{-1}\mathbf{y}^{(2)} - H^{-1} \begin{bmatrix} P^T \\ 0 \\ \vdots \\ 0 \end{bmatrix} \mathbf{x}_r^{(1)}. \tag{9.5}$$

Now let us write

$$G\mathbf{z}^{(1)} = \mathbf{y}^{(1)}, \qquad H\mathbf{z}^{(2)} = \mathbf{y}^{(2)}, \tag{9.6}$$

$$GW^{(1)} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ P \end{bmatrix}, \qquad HW^{(2)} = \begin{bmatrix} P^T \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{9.7}$$

Then as we partition the vectors $\mathbf{z}^{(1)}$, $\mathbf{z}^{(2)}$ and the matrices $W^{(1)}$ and $W^{(2)}$ as in (9.3), (9.4) and (9.5) become

$$\begin{aligned} \mathbf{x}_j^{(1)} &= \mathbf{z}_j^{(1)} - W_j^{(1)}\mathbf{x}_1^{(2)}, & j &= 1, 2, \cdots, r, \\ \mathbf{x}_j^{(2)} &= \mathbf{z}_j^{(2)} - W_j^{(2)}\mathbf{x}_r^{(1)}, & j &= 1, 2, \cdots, s. \end{aligned} \tag{9.8}$$

For (9.8), we have

$$\begin{bmatrix} I & W_r^{(1)} \\ W_1^{(2)} & I \end{bmatrix} \begin{bmatrix} \mathbf{x}_r^{(1)} \\ \mathbf{x}_1^{(2)} \end{bmatrix} = \begin{bmatrix} \mathbf{z}_r^{(1)} \\ \mathbf{z}_1^{(2)} \end{bmatrix}.$$

This system of equations is two-cyclic, so we may reduce the system to

$$(I - W_r^{(1)}W_1^{(2)})\mathbf{x}_r^{(1)} = \mathbf{z}_r^{(1)} - W_r^{(1)}\mathbf{z}_1^{(2)}. \tag{9.9}$$

This system of equations can most easily be solved using Gaussian elimination. Once the two-cyclic system of (9.9) has been solved, all other components may be computed using (9.8) or by solving the system

$$G\mathbf{x}^{(1)} = \mathbf{y}^{(1)} - \begin{bmatrix} 0 \\ 0 \\ \vdots \\ P \end{bmatrix} \mathbf{x}_1^{(2)},$$

$$H\mathbf{x}^{(2)} = \mathbf{y}^{(2)} - \begin{bmatrix} P^T \\ 0 \\ \vdots \\ 0 \end{bmatrix} \mathbf{x}_r^{(1)}.$$

If system (9.1) is to be solved for a number of different right-hand sides, it is best to save the LU decomposition of

$$(I - W_r^{(1)}W_1^{(2)}). \tag{9.10}$$

Thus, the algorithm proceeds as follows:

(i) Solve for $\mathbf{z}_r^{(1)}$ and $\mathbf{z}_1^{(2)}$ using the methods of § 2 or 3.

(ii) Solve for $W_r^{(1)}$ and $W_1^{(2)}$ using the methods of § 2 or 3.

(iii) Solve (9.9) using Gaussian elimination. Save the LU decomposition of (9.10).

(iv) Solve for the unknown components of $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$.

10. Accuracy of the CORF algorithm. As will be shown in § 11, the CORF algorithm and the Buneman algorithms are mathematically identical. The difference between the methods lies in the way the right-hand side is calculated at each stage of the reduction. To the authors' knowledge, this is the only direct method for solving linear equations in which the right-hand side of the equations plays an important role in the numerical solution of the equations. In this section, we show the difficulties encountered in using the CORF algorithm. In § 13, we will prove the stability of the Buneman algorithms.

Recall from § 3 that it is possible to compute $A^{(r)}\mathbf{y}_j^{(r)}$ by the following algorithm:

$$\boldsymbol{\eta}_0 = -2\mathbf{y}_j^{(r)}, \qquad \boldsymbol{\eta}_1 = A\mathbf{y}_j^{(r)},$$

(10.1)

$$\boldsymbol{\eta}_s = -A\boldsymbol{\eta}_{s-1} - T^2\boldsymbol{\eta}_{s-2}$$

for $s = 2, 3, \cdots, 2^r$, so that

$$\boldsymbol{\eta}_{2^r} = A^{(r)}\mathbf{y}_j^{(r)}.$$

Because of roundoff error, one actually computes the sequence

(10.2)

$$\tilde{\boldsymbol{\eta}}_0 = -2\mathbf{y}_j^{(r)}, \qquad \tilde{\boldsymbol{\eta}}_1 = A\mathbf{y}_j^{(r)} + \boldsymbol{\delta}_0,$$

$$\tilde{\boldsymbol{\eta}}_s = -A\tilde{\boldsymbol{\eta}}_{s-1} - T^2\tilde{\boldsymbol{\eta}}_{s-2} + \boldsymbol{\delta}_{s-1}, \qquad s = 2, \cdots, 2^r,$$

where $\boldsymbol{\delta}_s$ is the perturbation induced by the roundoff error. Again, as in § 2, we write

$$A = Q\Lambda Q^T, \qquad T = Q\Omega Q^T, \tag{10.3}$$

where Q is the set of orthonormalized eigenvectors of A and T, and Λ and Ω are the diagonal matrices of eigenvalues of A and T, respectively. Thus, substituting (10.3) into (10.2), we have

$$\boldsymbol{\xi}_0 = -2\bar{\mathbf{y}}, \qquad \boldsymbol{\xi}_1 = -\tfrac{1}{2}\Lambda\boldsymbol{\xi}_0 + \boldsymbol{\tau}_0, \tag{10.4a}$$

$$\boldsymbol{\xi}_s = -\Lambda\boldsymbol{\xi}_{s-1} - \Omega^2\boldsymbol{\xi}_{s-2} + \boldsymbol{\tau}_{s-1}, \tag{10.4b}$$

where

$$\bar{\mathbf{y}} = Q^T\mathbf{y}_j^{(r)}, \quad \boldsymbol{\xi}_s = Q^T\tilde{\boldsymbol{\eta}}_s, \quad \boldsymbol{\tau}_s = Q^T\boldsymbol{\delta}_s.$$

Because Λ and Ω are diagonal, we may write an equation for each component of ξ_s; namely,

$$\xi_{j,s+1} + \lambda_j \xi_{j,s} + \omega_j^2 \xi_{j,s-1} = \tau_{j,s}, \qquad j = 1, 2, \cdots, p. \tag{10.5}$$

The solution of (10.5) can be given explicitly. Consider the characteristic equation

$$\varphi_j(\alpha) \equiv \alpha^2 + \lambda_j \alpha + \omega_j^2 = 0,$$

which has roots β_j and γ_j; then

$$\begin{aligned} \xi_{j,s} = {} & \frac{\beta_j^s - \gamma_j^s}{\beta_j - \gamma_j}\xi_{j,1} - \beta_j\gamma_j \frac{\beta_j^{s-1} - \gamma_j^{s-1}}{\beta_j - \gamma_j}\xi_{j,0} \\ & + \sum_{k=1}^{s-1} \frac{\beta_j^{s-k} - \gamma_j^{s-k}}{\beta_j - \gamma_j}\tau_{j,k} \quad \text{when } \beta_j \neq \gamma_j, \end{aligned} \tag{10.6a}$$

$$= s\beta_j^{s-1}\xi_{j,1} - (s-1)\beta_j^s \xi_{j,0} + \sum_{k=1}^{s-1} (s-k)\beta_j^{s-k-1}\tau_{j,k} \quad \text{when } \beta_j = \gamma_j. \tag{10.6b}$$

Let

$$\begin{aligned} -\lambda_j/(2\omega_j) &= \cos\theta_j \quad \text{when } |\lambda_j/(2\omega_j)| \leqq 1, \\ &= \cosh z_j \quad \text{when } |\lambda_j/(2\omega_j)| \geqq 1. \end{aligned}$$

Then using the initial conditions of (10.4a), we may write (10.6a) as

$$\xi_{j,s} = -2\omega_j^s \cos(s\theta_j)\bar{y}_j + \sum_{k=0}^{s-1} \omega_j^{s-k-1} \frac{\sin(s-k)\theta_j}{\sin\theta_j}\tau_{j,k} \quad \text{when } |\lambda_j/(2\omega_j)| \leqq 1, \tag{10.7a}$$

$$= -2\omega_j^s \cosh(sz_j)\bar{y}_j + \sum_{k=0}^{s-1} \omega_j^{s-k-1} \frac{\sinh(s-k)z_j}{\sinh z_j}\tau_{j,k} \quad \text{when } |\lambda_j/(2\omega_j)| \geqq 1. \tag{10.7b}$$

Note that

$$-2\omega_j^s \times \begin{cases} \cos s\theta_j \\ \cosh sz_j \end{cases} \equiv p_s(\lambda_j, \omega_j), \tag{10.8}$$

given in § 3. Thus

$$\tilde{\boldsymbol{\eta}}_s = P_s(A, T)\mathbf{y}_j^{(r)} + \sum_{k=0}^{s-1} QS_{s-k}Q^T\, \boldsymbol{\delta}_k, \tag{10.9}$$

where

$$\{S_m\}_{ij} = \omega_j^{m-1} \times \begin{cases} \dfrac{\sin m\theta_j}{\sin \theta_j} & \text{when } |\lambda_j/(2\omega_j)| \leqq 1 \quad \text{and} \quad i = j, \\[2ex] \dfrac{\sinh m z_j}{\sinh z_j} & \text{when } |\lambda_j/(2\omega_j)| \geqq 1 \quad \text{and} \quad i = j, \end{cases}$$

$$= 0 \qquad\qquad \text{for } i \neq j.$$

Therefore, if $|\lambda_j/(2\omega_j)| > 1$, each $\xi_{j,s}$ has good relative accuracy. However, if $z_1 > z_p$, then $\xi_{1,s}$ may become very large relative to $\xi_{p,s}$; and since $\boldsymbol{\eta}_s = Q\boldsymbol{\xi}_s$, $\xi_{p,s}$ will be lost in rounding.

We now apply the above results to Poisson's equation with Dirichlet boundary conditions. For the five-point difference operator with mesh width Δx in the x-direction and Δy in the y-direction, we have

$$\lambda_j = -2\left[1 + \rho^2\left(1 - \cos\frac{j\pi}{p+1}\right)\right], \qquad \omega_j = 1$$

and

$$\rho = \Delta x/\Delta y \quad \text{or} \quad \Delta y/\Delta x,$$

depending on how one orders the equations. By inspection

$$|\lambda_j/(2\omega_j)| > 1$$

for all j, and hence, as s increases, (10.1) leads to a numerically unstable algorithm. A similar result holds for the nine-point difference approximation to Poisson's equation.

Although the above results were obtained under the assumption of (10.1), similar results will be obtained regardless of how $A^{(r)}\mathbf{y}_j^{(r)}$ is computed. The problem is that $A^{(r)}$ becomes very ill-conditioned as r increases. For the five-point approximation, the ratio of the largest eigenvalue of $A^{(r)}$ to its smallest eigenvalue is

$$\frac{\omega_n^{2^r}\cosh(2^r z_n)}{\omega_1^{2^r}\cosh(2^r z_1)} \cong e^{2^r(z_n - z_1)}.$$

Give t digits of base β arithmetic, it will generally be impossible to represent $\boldsymbol{\eta}_{2^r}$ in t digits whenever

$$2^r(z_n - z_1) \geqq t \ln \beta.$$

As noted in § 3, Hockney [2], [9] has combined one or more steps of odd–even reduction with the fast Fourier transform to produce a Poisson solver. The above analysis allows one to determine the number of reductions that can be safely performed, and careful attention must be given to it.

11. The Buneman algorithm and variants. In this section, we shall describe in detail the Buneman algorithm [2] and a variation of it. The difference between the Buneman algorithm and the CORF algorithm lies in the way the right-hand side is calculated at each stage of the reduction. Henceforth, we shall assume that, in the system of (2.5), $T = I_p$, the identity matrix of order p.

Again consider the system of equations as given by (2.5) with $q = 2^{k+1} - 1$. After one stage of cyclic reduction, we have

$$\mathbf{x}_{j-2} + (2I_p - A^2)\mathbf{x}_j + \mathbf{x}_{j+2} = \mathbf{y}_{j-1} + \mathbf{y}_{j+1} - A\mathbf{y}_j \tag{11.1}$$

for $j = 2, 4, \cdots, q - 1$, with $\mathbf{x}_0 = \mathbf{x}_{q+1} = \boldsymbol{\theta}$, the null vector. Note that the right-hand side of (11.1) may be written as

$$\mathbf{y}_j^{(1)} = \mathbf{y}_{j-1} + \mathbf{y}_{j+1} - A\mathbf{y}_j = A^{(1)}A^{-1}\mathbf{y}_j + \mathbf{y}_{j-1} + \mathbf{y}_{j+1} - 2A^{-1}\mathbf{y}_j, \tag{11.2}$$

where $A^{(1)} = (2I_p - A^2)$.

Let us define

$$\mathbf{p}_j^{(1)} = A\ {}^{1}\mathbf{y}_j, \qquad \mathbf{q}_j^{(1)} = \mathbf{y}_{j-1} + \mathbf{y}_{j+1} - 2\mathbf{p}_j^{(1)}.$$

Then

$$\mathbf{y}_j^{(1)} = A^{(1)}\mathbf{p}_j^{(1)} + \mathbf{q}_j^{(1)}. \tag{11.3}$$

After r reductions, we have, by (3.3),

$$\mathbf{y}_j^{(r+1)} = (\mathbf{y}_{j-2^r}^{(r)} + \mathbf{y}_{j+2^r}^{(r)}) - A^{(r)}\mathbf{y}_j^{(r)}. \tag{11.4}$$

Let us write in a fashion similar to (11.3),

$$\mathbf{y}_j^{(r)} = A^{(r)}\mathbf{p}_j^{(r)} + \mathbf{q}_j^{(r)}. \tag{11.5}$$

Substituting (11.5) into (11.4) and making use of the identity $(A^{(r)})^2 = 2I_p - A^{(r+1)}$ from (3.3), we have the following relationships:

$$\mathbf{p}_j^{(r+1)} = \mathbf{p}_j^{(r)} - (A^{(r)})^{-1}(\mathbf{p}_{j-2^r}^{(r)} + \mathbf{p}_{j+2^r}^{(r)} - \mathbf{q}_j^{(r)}), \tag{11.6a}$$

$$\mathbf{q}_j^{(r+1)} = \mathbf{q}_{j-2^r}^{(r)} + \mathbf{q}_{j+2^r}^{(r)} - 2\mathbf{p}_j^{(r+1)} \tag{11.6b}$$

for $j = i2^{r+1}$, $i = 1, 2, \cdots, 2^{k-r} - 1$, with

$$\mathbf{p}_0^{(r)} = \mathbf{p}_{2^{k+1}}^{(r)} = \mathbf{q}_0^{(r)} = \mathbf{q}_{2^{k+1}}^{(r)} = \boldsymbol{\theta}.$$

To compute $(A^{(r)})^{-1}(\mathbf{p}_{j-2^r}^{(r)} + \mathbf{p}_{j+2^r}^{(r)} - \mathbf{q}_j^{(r)})$ in (11.6a), we solve the system of equations

$$A^{(r)}(\mathbf{p}_j^{(r)} - \mathbf{p}_j^{(r+1)}) = \mathbf{p}_{j-2^r}^{(r)} + \mathbf{p}_{j+2^r}^{(r)} - \mathbf{q}_j^{(r)},$$

where $A^{(r)}$ is given by the factorization (3.10), namely,

$$A^{(r)} = -\prod_{j=1}^{2^r} (A + 2\cos\theta_j^{(r)} I_p),$$

$$\theta_j^{(r)} = (2j - 1)\pi/2^{r+1}.$$

After k reductions, one has the equation

$$A^{(k)}\mathbf{x}_{2^k} = A^{(k)}\mathbf{p}_{2^k}^{(k)} + \mathbf{q}_{2^k}^{(k)},$$

and hence

$$\mathbf{x}_{2^k} = \mathbf{p}_{2^k}^{(k)} + (A^{(k)})^{-1}\mathbf{q}_{2^k}^{(k)}.$$

Again one uses the factorization of $A^{(k)}$ for computing $(A^{(k)})^{-1}\mathbf{q}_{2^k}^{(k)}$. To back-solve, we use the relationship

$$\mathbf{x}_{j-2^r} + A^{(r)}\mathbf{x}_j + \mathbf{x}_{j+2^r} = A^{(r)}\mathbf{p}_j^{(r)} + \mathbf{q}_j^{(r)}$$

for $j = i2^r$, $i = 1, 2, \cdots, 2^{k+1-r} - 1$, with $\mathbf{x}_0 = \mathbf{x}_{2^{k+1}} = \boldsymbol{\theta}$.

For $j = 2^r, 3{\cdot}2^r, \cdots, 2^{k+1} - 2^r$, we solve the system of equations

$$A^{(r)}(\mathbf{x}_j - \mathbf{p}_j^{(r)}) = \mathbf{q}_j^{(r)} - (\mathbf{x}_{j-2^r} + \mathbf{x}_{j+2^r}) \tag{11.7}$$

using the factorization of $A^{(r)}$; hence

$$\mathbf{x}_j = \mathbf{p}_j^{(r)} + (\mathbf{x}_j - \mathbf{p}_j^{(r)}). \tag{11.8}$$

Thus the Buneman algorithm (variant 1) proceeds as follows:

(i) Compute the sequence $\{\mathbf{p}_j^{(r)}, \mathbf{q}_j^{(r)}\}$ by (11.6) for $r = 1, \cdots, k$, with $\mathbf{p}_j^{(0)} = \boldsymbol{\theta}$ for $j = 0, \cdots, 2^{k+1}$, and $\mathbf{q}_j^{(0)} = \mathbf{y}_j$ for $j = 1, 2, \cdots, 2^{k+1} - 1$.

(ii) Back-solve for $\mathbf{x}_j$ using (11.7) and (11.8).

It is possible to eliminate the sequence $\{\mathbf{p}_j^{(r)}\}$. From (11.6b) we note that

$$\mathbf{p}_j^{(r+1)} = \tfrac{1}{2}(\mathbf{q}_{j-2h}^{(r)} + \mathbf{q}_{j+2h}^{(r)} - \mathbf{q}_j^{(r+1)}), \tag{11.9}$$

where

$$h = 2^{r-1}.$$

Using (11.9) in (11.6a) and modifying the subscripts and superscripts appropriately, we have

$$\begin{aligned} \mathbf{q}_j^{(r+1)} = {} & \mathbf{q}_{j-2h}^{(r)} - \mathbf{q}_{j-h}^{(r-1)} + \mathbf{q}_j^{(r)} - \mathbf{q}_{j+h}^{(r-1)} + \mathbf{q}_{j+2h}^{(r)} \\ & + (A^{(r)})^{-1}[\mathbf{q}_{j-3h}^{(r-1)} - \mathbf{q}_{j-2h}^{(r)} + \mathbf{q}_{j-h}^{(r-1)} - 2\mathbf{q}_j^{(r)} \\ & + \mathbf{q}_{j+h}^{(r-1)} - \mathbf{q}_{j+2h}^{(r)} + \mathbf{q}_{j+3h}^{(r-1)}] \end{aligned} \tag{11.10}$$

for $j = 2^r, 2^{r+1}, \cdots, 2^{k+1} - 2^r$, with

$$\mathbf{q}_0^{(r)} = \mathbf{q}_{2^{k+1}}^{(r)} = \boldsymbol{\theta} \qquad \text{for all } r,$$

$$\mathbf{q}_j^{(0)} = \mathbf{y}_j \qquad \text{for } j = 1, 2, \cdots, 2^{k+1} - 1,$$

$$\mathbf{q}_j^{(1)} = \mathbf{y}_{j-1} + \mathbf{y}_{j+1} - 2A^{-1}\mathbf{y}_j \qquad \text{for } j = 2, 4, \cdots, 2^{k+1} - 2.$$

To solve for $\mathbf{x}_j$, we use the relationships (11.7) and (11.9), so that

$$\begin{aligned} \mathbf{x}_j = {} & \tfrac{1}{2}(\mathbf{q}_{j-h}^{(r-1)} + \mathbf{q}_{j+h}^{(r-1)} - \mathbf{q}_j^{(r)}) \\ & - (A^{(r)})^{-1}(\mathbf{x}_{j-2h} + \mathbf{x}_{j+2h} - \mathbf{q}_j^{(r)}). \end{aligned} \tag{11.11}$$

Thus the Buneman algorithm (variant 2) proceeds as follows:

(i) Compute the sequence $\{\mathbf{q}_j^{(r)}\}$ by (11.10) for $r = 1, 2, \cdots, k$.

(ii) Back-solve for $\mathbf{x}_j$ using (11.11).

Note that the Buneman algorithm (variant 2) requires half the storage that the Buneman algorithm (variant 1) requires. However, the variant 2 algorithm requires approximately twice as many additions.

The $\mathbf{p}_j$'s and $\mathbf{q}_j$'s can be written in terms of the $\mathbf{x}_j$'s. In § 13, we shall show how this affects the stability of the methods. Note that

$$\mathbf{p}_j^{(1)} = A^{-1}\mathbf{y}_j = \mathbf{x}_j + A^{-1}(\mathbf{x}_{j-1} + \mathbf{x}_{j+1})$$

and

$$\begin{aligned}\mathbf{q}_j^{(1)} &= \mathbf{y}_{j-1} + \mathbf{y}_{j+1} - 2\mathbf{p}_j^{(1)}\\ &= \mathbf{x}_{j-2} - (A)^{-1}A^{(1)}(\mathbf{x}_{j-1} + \mathbf{x}_{j+1}) + \mathbf{x}_{j+2}.\end{aligned}$$

By an inductive argument, it is possible to show that

$$\mathbf{p}_j^{(r)} = \mathbf{x}_j + (-1)^{r+1}S^{(r)}\left\{\sum_{k=1}^{2^{r-1}} (\mathbf{x}_{j-(2k-1)} + \mathbf{x}_{j+(2k-1)})\right\} \tag{11.12}$$

and

$$\mathbf{q}_j^{(r)} = \mathbf{x}_{j-2^r} + (-1)^r S^{(r)}A^{(r)}\left\{\sum_{k=1}^{2^{r-1}} (\mathbf{x}_{j-(2k-1)} + \mathbf{x}_{j+(2k-1)})\right\} + \mathbf{x}_{j+2^r},$$

where

$$S^{(r)} = (A^{(r-1)}A^{(r-2)} \cdots A^{(0)})^{-1}.$$

12. Applications of the Buneman algorithm to Poisson's equation. As was pointed out in § 4, matrices of the form of (2.5) arise in solving the five-point finite difference approximation to Poisson's equation over a rectangular region with Dirichlet boundary conditions; hence, it is possible to use the methods of § 11. For the five-point approximation to Poisson's equation over a rectangular region with Neumann or periodic conditions it is necessary to modify the Buneman algorithms.

For the Neumann boundary conditions, we have the system of equations

$$\begin{aligned}A\mathbf{x}_0 + 2\mathbf{x}_1 \qquad &= \mathbf{y}_0,\\ \mathbf{x}_{j-1} + A\mathbf{x}_j + \mathbf{x}_{j+1} &= \mathbf{y}_j, \qquad j = 1, 2, \cdots, m-1,\\ 2\mathbf{x}_{m-1} + A\mathbf{x}_m &= \mathbf{y}_m, \qquad \text{with } m = 2^{k+1}.\end{aligned}$$

We define

$$\mathbf{y}_j^{(1)} = A^{(1)}\mathbf{p}_j^{(1)} + \mathbf{q}_j^{(1)} \qquad \text{for } j = 0, 2, 4, \cdots, 2^{k+1},$$

where

$$\begin{aligned}\mathbf{p}_0^{(1)} &= A^{-1}\mathbf{y}_0, & \mathbf{q}_0^{(1)} &= 2(\mathbf{y}_1 - \mathbf{p}_0^{(1)}),\\ \mathbf{p}_j^{(1)} &= A^{-1}\mathbf{y}_j, & \mathbf{q}_j^{(1)} &= \mathbf{y}_{j-1} + \mathbf{y}_{j+1} - 2\mathbf{p}_j^{(1)},\\ & & & \qquad j = 2, 4, \cdots, m-2,\\ \mathbf{p}_m^{(1)} &= A^{-1}\mathbf{y}_m, & \mathbf{q}_m^{(1)} &= 2(\mathbf{y}_{m-1} - \mathbf{p}_m^{(1)}).\end{aligned}$$

In general then, as in § 11, we have for $r = 1, 2, \cdots, k-1$,

$$\mathbf{y}_j^{(r+1)} = A^{(r+1)}\mathbf{p}_j^{(r+1)} + \mathbf{q}_j^{(r+1)},$$

where

$$\mathbf{p}_0^{(r+1)} = \mathbf{p}_0^{(r)} - (A^{(r)})^{-1}(2\mathbf{p}_{2^r}^{(r)} - \mathbf{q}_0^{(r)}), \qquad \mathbf{q}_0^{(r+1)} = 2(\mathbf{q}_{2^r}^{(r)} - \mathbf{p}_0^{(r+1)}),$$
$$\mathbf{p}_j^{(r+1)} = \mathbf{p}_j^{(r)} - (A^{(r)})^{-1}(\mathbf{p}_{j-2^r}^{(r)} + \mathbf{p}_{j-2^r}^{(r)} - \mathbf{q}_j^{(r)}),$$
$$\mathbf{p}_j^{(r+1)} = \mathbf{q}_{j-2^r}^{(r)} + \mathbf{q}_{j+2^r}^{(r)} - 2\mathbf{p}_j^{(r+1)},$$
$$\text{for } j = i2^{r+1}, \quad i = 1, 2, \cdots, 2^{k-r} - 1,$$
$$\mathbf{p}_m^{(r+1)} = \mathbf{p}_m^{(r)} - (A^{(r)})^{-1}(2\mathbf{p}_{m-2^r}^{(r)} - \mathbf{q}_m^{(r)}), \qquad \mathbf{q}_m^{(r+1)} = 2\mathbf{q}_{m-2^r}^{(r)} - 2\mathbf{p}_m^{(r+1)}.$$

Finally

$$\mathbf{y}_{2^k}^{(k+1)} = B^{(k+1)}\mathbf{p}_{2^k}^{(k+1)} + \mathbf{q}_{2^k}^{(k+1)}, \tag{12.1}$$

where

$$B^{(k+1)} = 4I - (A^{(k)})^2,$$
$$\mathbf{p}_{2^k}^{(k+1)} = \mathbf{p}_{2^k}^{(k)} - (A^k)^{-1}(\mathbf{p}_0^{(k)} + \mathbf{p}_{2^{k+1}}^{(k)} - \mathbf{q}_{2^k}^{(k)}), \tag{12.2}$$
$$\mathbf{q}_{2^k}^{(k+1)} = \mathbf{q}_0^{(k)} + \mathbf{q}_{2^{k+1}}^{(k)} - 4\mathbf{p}_{2^k}^{(k+1)}.$$

From (5.4), we see that

$$B^{(k+1)}\mathbf{x}_{2^k} = B^{(k+1)}\mathbf{p}_{2^k}^{(k+1)} + \mathbf{q}_{2^k}^{(k+1)},$$

so that

$$\mathbf{x}_{2^k} = \mathbf{p}_{2^k}^{(k+1)} + (B^{(k+1)})^{-}\mathbf{q}_{2^k}^{(k+1)}.$$

$(B^{(k+1)})^{-}\mathbf{q}_{2^k}^{(k+1)}$ indicates a solution to the singular system $B^{(k+1)}(\mathbf{x}_{2^k} - \mathbf{p}_{2^k}^{(k+1)}) = \mathbf{q}_{2^k}^{(k+1)}$. The factorization of $B^{(k+1)}$ is given by (5.6). The back-substitution process proceeds as in § 11. It is also possible to eliminate the $\mathbf{p}_j^{(r)}$ sequence as was done in the previous section.

For periodic boundary conditions, we have the system of equations

$$\begin{aligned} A\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_m &= \mathbf{y}_1, \\ \mathbf{x}_{j-1} + A\mathbf{x}_j + \mathbf{x}_{j+1} &= \mathbf{y}_j \qquad \text{for } j = 2, 3, \cdots, m-1, \\ \mathbf{x}_1 + \mathbf{x}_{m-1} + A\mathbf{x}_m &= \mathbf{y}_m. \end{aligned}$$

We define

$$\mathbf{y}_j^{(1)} = A^{(1)}\mathbf{p}_j^{(1)} + \mathbf{q}_j^{(1)} \qquad \text{for } j = 2, 4, \cdots, 2^{k+1},$$

where

$$\begin{aligned} \mathbf{p}_2^{(1)} &= A^{-1}\mathbf{y}_2, & \mathbf{q}_2^{(1)} &= \mathbf{y}_1 + \mathbf{y}_3 - 2\mathbf{p}_2^{(1)}, \\ \mathbf{p}_j^{(1)} &= A^{-1}\mathbf{y}_j, & \mathbf{q}_j^{(1)} &= \mathbf{y}_{j-1} + \mathbf{y}_{j+1} - 2\mathbf{p}_j^{(1)}, \quad j = 4, 6, \cdots, m-2, \\ \mathbf{p}_j^{(1)} &= A^{-1}\mathbf{y}_m, & \mathbf{q}_m^{(1)} &= \mathbf{y}_1 + \mathbf{y}_{m-1} - 2\mathbf{p}_m^{(1)}. \end{aligned}$$

In general, for $r = 1, 2, \cdots, k-1$,

$$\mathbf{y}_j^{(r+1)} = A^{(r+1)}\mathbf{p}_j^{(r+1)} + \mathbf{q}_j^{(r+1)} \quad \text{for } j = i2^{r+1}, \quad i = 1, 2, \cdots, 2^{k-r},$$

where

$$\begin{aligned}
\mathbf{p}_{2^{r+1}}^{(r+1)} &= \mathbf{p}_{2^{r+1}}^{(r)} - (A^{(r)})^{-1}(\mathbf{p}_{2^r}^{(r)} + \mathbf{p}_{3\times 2^r}^{(r)} - \mathbf{q}_{2^{r+1}}^{(r)}),\\
\mathbf{q}_{2^{r+1}}^{(r)} &= \mathbf{q}_{2^r}^{(r)} + \mathbf{q}_{3\times 2^r}^{(r)} - 2\mathbf{p}_{2^{r+1}}^{(r+1)},\\
\mathbf{p}_j^{(r+1)} &= \mathbf{p}_j^{(r)} - (A^{(r)})^{-1}(\mathbf{p}_{j-2^r}^{(r)} + \mathbf{p}_{j+2^r}^{(r)} - \mathbf{q}_j^{(r)}),\\
\mathbf{q}_j^{(r+1)} &= \mathbf{q}_{j-2^r}^{(r)} + \mathbf{q}_{j+2^r}^{(r)} - 2\mathbf{p}_j^{(r+1)},\\
\mathbf{p}_m^{(r+1)} &= \mathbf{p}_m^{(r)} - (A^{(r)})^{-1}(\mathbf{p}_{2^r}^{(r)} + \mathbf{p}_{m-2^r}^{(r)} - \mathbf{q}_m^{(r)}),\\
\mathbf{q}_m^{(r+1)} &= \mathbf{q}_{2^r}^{(r)} + \mathbf{q}_{m-2^r}^{(r)} - 2\mathbf{p}_m^{(r+1)}.
\end{aligned}$$

Finally, as (12.1),

$$\mathbf{y}_{2^k}^{(k+1)} = B^{(k+1)}\mathbf{p}_{2^k}^{(k+1)} + \mathbf{q}_{2^k}^{(k+1)},$$

where

$$\mathbf{p}_{2^k}^{(k+1)} = \mathbf{p}_{2^k}^{(k)} - (A^{(k)})^{-1}(2\mathbf{p}_{2^{k+1}}^{(k)} - \mathbf{q}_{2^k}^{(k)}), \qquad \mathbf{q}_{2^k}^{(k+1)} = 2\mathbf{q}_{2^{k+1}}^{(k)} - 4\mathbf{p}_{2^k}^{(k+1)}$$

and $B^{(k+1)}$ is defined by (12.2). Then

$$B^{(k+1)}\mathbf{x}_{2^k} = B^{(k+1)}\mathbf{p}_{2^k}^{(k+1)} + \mathbf{q}_{2^k}^{(k+1)},$$

so that

$$\mathbf{x}_{2^k} = \mathbf{p}_{2^k}^{(k+1)} + (B^{(k+1)})^{-}\mathbf{q}_{2^k}^{(k+1)}$$

The back-substitution process proceeds as in § 11.

It is possible to express $\mathbf{p}_j^{(r)}$ and $\mathbf{q}_j^{(r)}$ in terms of $\mathbf{x}_j$ as in (11.12) and (11.13).

13. Accuracy of the Buneman algorithms. As was shown in § 11, the Buneman algorithms consist of generating the sequence of vectors $\{\mathbf{p}_j^{(r)}, \mathbf{q}_j^{(r)}\}$. Let us write, using (11.12) and (11.13),

(13.1a) $$\mathbf{p}_j^{(r)} = \mathbf{x}_j^{(r)} + \mathbf{g}_j^{(r)},$$

(13.1b) $$\mathbf{q}_j^{(r)} = \mathbf{x}_{j-2^r} + \mathbf{x}_{j+2^r} - A^{(r)}\mathbf{g}_j^{(r)},$$

where

(13.2) $$\mathbf{g}_j^{(r)} = (-1)^{r+1}S^{(r)}\left\{\sum_{k=1}^{2^{r-1}} (\mathbf{x}_{j-(2k-1)} + \mathbf{x}_{j+(2k-1)})\right\}$$

and

(13.3) $$S^{(r)} = (A^{(r-1)} \cdots A^{(0)})^{-1}.$$

Then

(13.4) $$\|\mathbf{p}_j^{(r)} - \mathbf{x}_j^{(r)}\|_2 \leqq \|S^{(r)}\|_2\|\mathbf{x}\|',$$

(13.5) $$\|\mathbf{q}_j^{(r)} - (\mathbf{x}_{j-2^r} + \mathbf{x}_{j+2^r})\|_2 \leqq \|S^{(r)}A^{(r)}\|_2\|\mathbf{x}\|',$$

where

$\|\mathbf{v}\|_2$ indicates the Euclidean norm of a vector $\mathbf{v}$,
$\|C\|_2$ indicates the spectral norm of a matrix C, and
$\|\mathbf{x}\|' = \sum_{j=1}^{q} \|\mathbf{x}_j\|_2$.

When $T = I_p$, we may redefine the polynomials given in § 3 in the following way. Let

$$\psi = -a/2,$$

and define

$$\psi = \begin{cases} \cos\theta & \text{for } |\psi| \leqq 1, \\ \cosh\theta & \text{for } |\psi| \geqq 1. \end{cases}$$

Then in a similar fashion to (10.8),

$$p_{2^k}(a) = \begin{cases} -2\cos(2^k \cos^{-1}\psi) & \text{for } |\psi| \leqq 1, \\ -2\cosh(2^k \cosh^{-1}\psi) & \text{for } |\psi| \geqq 1. \end{cases}$$

Thus for $A = A^T$,

$$\begin{aligned} \|S^{(r)}\|_2 &\leqq \prod_{j=0}^{r-1} \|(A^{(j)})^{-1}\|_2 \\ &\leqq \prod_{j=0}^{r-1} \max_{\{\lambda_i\}} |[p_{2^j}(\lambda_i)]^{-1}|, \end{aligned}$$

where $\{\lambda_i\}$ are the eigenvalues of A. Therefore, for $|\lambda_i| \geq 2$,

$$\|S^{(r)}\|_2 \leqq 2^{-r} \prod_{j=0}^{r-1} \max_{\{\theta_i\}} [\cosh 2^j\theta_i]^{-1},$$

where

$$\theta_i = \cosh^{-1}(-\lambda_i/2).$$

Finally,

$$\|S^{(r)}A^{(r)}\|_2 \leqq 2^{-r+1} \cdot \max_{\{\theta_i\}} \left\{ \left(\prod_{j=0}^{r-1} [\cosh 2^j\theta_i]^{-1} \right) \cdot \cosh 2^r\theta_i \right\}$$

when $|\lambda_i| \geqq 2$.

For the five-point difference approximation to Poisson's equation over a rectangular region with Dirichlet boundary conditions,

$$\lambda_i = -2\left(1 + \rho^2\left(1 - \cos\frac{i\pi}{p+1}\right)\right),$$

where $\rho = \Delta x/\Delta y$ or $\Delta y/\Delta x$ depending on the ordering of the equations. Thus

$$\theta_i = \cosh^{-1}\left(1 + \rho^2\left(1 - \cos\frac{i\pi}{p+1}\right)\right),$$

which implies $\theta_i > 1$ for all i. Then

$$\max_{\{\theta_i\}} [\cosh 2^j\theta_i]^{-1} = \left[\cosh 2^j \left\{\cosh^{-1}\left(1 + \rho^2\left(1 - \cos\frac{\pi}{p+1}\right)\right)\right\}\right]^{-1}.$$

Thus, after some simplification,

$$(13.6) \qquad \|S^{(r)}\|_2 \leqq \frac{e^{-c_r\theta_1}}{\prod_{j=0}^{r-1}(1+e^{-2^{j+1}\theta_1})} < e^{-c_r\theta_1},$$

where $c_r = 2^r - 1, \cosh\theta_i = 1 + \rho^2\left(1 - \cos\frac{\pi}{p+1}\right)$.

A similar calculation shows that

$$(13.7) \qquad \|A^{(r)}S^{(r)}\|_2 < 2e^{\theta_p}.$$

Therefore, from (13.6), we see that for large r, $\mathbf{p}_j^{(r)}$ will be a good approximation to $\mathbf{x}_j$. And from (13.5) and (13.7), we see that

$$\|\mathbf{q}_j^{(r)} - (\mathbf{x}_{j-2^r} + \mathbf{x}_{j+2^r})\|_2 \leqq 2e^{\theta_p}\|\mathbf{x}\|',$$

so that the $\|\mathbf{q}_j^{(r)}\|_2$ remains bounded throughout the calculation. This explains why the Buneman algorithms lead to numerically stable results for solving the finite difference approximation to Poisson's equation.

14. Conclusion. Numerous applications require the repeated solution of a Poisson equation. The operation counts given by Dorr [5] indicate that the methods we have discussed should offer significant economies over older techniques; this has been verified in practice by many users. Computational experiments comparing the Buneman algorithm (variant one), the MD algorithm, the Peaceman–Rachford alternating direction algorithm and the point successive over-relaxation algorithm are given by Buzbee, et al. [3]. We conclude that the method of matrix decomposition, the Buneman algorithms, and Hockney's algorithm (when used with care) are valuable methods.

Acknowledgment. This paper has benefited greatly from the comments of Dr. F. Dorr, Mr. J. Alan George, Dr. R. Hockney and Professor O. Widlund.

REFERENCES

[1] RICHARD BELLMAN, *Introduction to Matrix Analysis*, McGraw-Hill, New York, 1960.

[2] OSCAR BUNEMAN, *A compact non-iterative Poisson solver*, Rep. 294, Stanford University Institute for Plasma Research, Stanford, Calif., 1969.

[3] B. L. BUZBEE, G. H. GOLUB AND C. W. NIELSON, *The method of odd even reduction and factorization with application to Poisson's equation. Part II*, Rep. LA-4288, Los Alamos Scientific Laboratory, University of California, Los Alamos, N. M.,

[4] J. W. COOLEY AND J. W. TUKEY, *An algorithm for machine calculation of complex Fourier series*, Math. Comp., 19 (1965), pp. 297–301.

[5] F. W. DORR, *The direct solution of the discrete Poisson equation on a rectangle*, SIAM Rev., 12 (1970), pp. 248–249.

[6] J. A. GEORGE, *The use of direct methods for the solution of the discrete Poisson equation on non-rectangular regions*, Computer Science Rep. 70-159, Stanford Univ., Stanford, Calif., 1970.

[7] G. H. GOLUB, R. UNDERWOOD AND J. WILKINSON, *Solution of $A\mathbf{x} = \lambda B\mathbf{x}$ when B is positive definite*, to appear.

[8] R. W. HOCKNEY, *A fast direct solution of Poisson's equation using Fourier analysis*, J. Assoc. Comput. Mach. (1965), pp. 95–113.

[9] ———, *The potential calculation and some applications*, Methods in Computational Physics, vol. 9, B. Adler, S. Fernbach and M. Rotenberg, eds., Academic Press, New York and London, 1969, pp. 136–211.

[10] R. E. LYNCH, J. R. RICE AND D. H. THOMAS, *Direct solution of partial difference equations by tensor product methods*, Numer. Math., 6 (1964), pp. 185–199.

[11] H. SHINTANI, *Direct solution of partial difference equations for a rectangle*, J. Sci. Hiroshima Univ. Ser. A-I, 32 (1968), pp. 17–53.

[12] R. S. VARGA, *Matrix Interative Analysis*, Prentice-Hall, New York, 1962.

20

NUMERICAL METHODS FOR COMPUTING ANGLES BETWEEN LINEAR SUBSPACES (WITH Å. BJÖRCK)

Math. Comp. (27), 579–594 (1973).

I spent the academic year 1968/69 as a postdoc at Stanford University and UC Berkeley. This gave me the possibility to see Gene regularly and George Forsythe occasionally. Another highlight I recall was to meet with the group of PhD students at the Computer Science Department, all of whom would later become famous.

Although I don't have a precise recollection of how the collaboration on this paper originated, I am pretty sure it was like this: Gene would sit down with me, take out a sheet of paper and in ten minutes he had sketched an elegant outline of the paper. The main starting point was the close connection between the minimax characterization of the singular values and that of the cosines of the principal angles. This works also in $\mathbf{C}^n$ and all the theory in the paper was done for the complex case. I recall that I was very excited about Gene's new algorithm for stably computing the SVD and this was a good application.

A main problem was to find a way to accurately compute small principal angles. A nice side-result of the solution was the explicit form (15) for the unitary matrix which performs a direct rotation of one of the two subspaces into the other – a precursor to the CS decomposition. In the numerical tests I took some pleasure in showing that the orthogonal bases computed by the modified Gram–Schmidt algorithm gave as accurate results as using Gene's Householder QR. This is now perhaps of minor interest. The paper first appeared as a technical report: STAN-CS-225-71, July 1971, two years before its publication.

Åke Björck
Linköping, Sweden

MATHEMATICS OF COMPUTATION, VOLUME 27, NUMBER 123, JULY, 1973

Numerical Methods for Computing Angles Between Linear Subspaces

By Åke Björck and Gene H. Golub*

Abstract. Assume that two subspaces F and G of a unitary space are defined as the ranges (or null spaces) of given rectangular matrices A and B. Accurate numerical methods are developed for computing the principal angles $\theta_k(F, G)$ and orthogonal sets of principal vectors $u_k \in F$ and $v_k \in G$, $k = 1, 2, \cdots, q = \dim(G) \leqq \dim(F)$. An important application in statistics is computing the canonical correlations $\sigma_k = \cos \theta_k$ between two sets of variates. A perturbation analysis shows that the condition number for θ_k essentially is $\max(\kappa(A), \kappa(B))$, where κ denotes the condition number of a matrix. The algorithms are based on a preliminary QR-factorization of A and B (or A^H and B^H), for which either the method of Householder transformations (HT) or the modified Gram-Schmidt method (MGS) is used. Then $\cos \theta_k$ and $\sin \theta_k$ are computed as the singular values of certain related matrices. Experimental results are given, which indicates that MGS gives θ_k with equal precision and fewer arithmetic operations than HT. However, HT gives principal vectors, which are orthogonal to working accuracy, which is not generally true for MGS. Finally, the case when A and/or B are rank deficient is discussed.

1. Introduction. Let F and G be given subspaces of a unitary space E^m, and assume that

$$p = \dim(F) \geqq \dim(G) = q \geqq 1. \tag{1}$$

The smallest angle $\theta_1(F, G) = \theta_1 \in [0, \pi/2]$ between F and G is defined by

$$\cos \theta_1 = \max_{u \in F} \max_{v \in G} u^H v, \qquad \|u\|_2 = 1, \ \|v\|_2 = 1.$$

Assume that the maximum is attained for $u = u_1$ and $v = v_1$. Then, $\theta_2(F, G)$ is defined as the smallest angle between the orthogonal complement of F with respect to u_1 and that of G with respect to v_1. Continuing in this way until one of the subspaces is empty, we are led to the following definition.

Definition. The principal angles $\theta_k \in [0, \pi/2]$ between F and G are recursively defined for $k = 1, 2, \cdots, q$ by

$$\cos \theta_k = \max_{v \in F} \max_{v \in G} u^H v = u_k^H v_k, \qquad \|u\|_2 = 1, \ \|v\|_2 = 1, \tag{2}$$

subject to the constraints

Received March 30, 1972.

AMS (MOS) subject classifications (1970). Primary 65F30, 65F20, 62H20, 62-04.

Key words and phrases. Numerical linear algebra, least squares, singular values, canonical correlations.

* The work of this author was in part supported by the Atomic Energy Commission and by the National Science Foundation.

$$u_j^H u = 0, \qquad v_j^H v = 0, \qquad j = 1, 2, \cdots, k-1.$$

The vectors $(u_1, \cdots, u_q)$ and $(v_1, \cdots, v_q)$ are called principal vectors of the pair of spaces.

We note that the principal vectors need not be uniquely defined, but the principal angles always are. The vectors $V = (v_1, \cdots, v_q)$ form a unitary basis for G and the vectors $U = (u_1, \cdots, u_q)$ can be complemented with $(p - q)$ unitary vectors so that $(u_1, \cdots, u_p)$ form a unitary basis for F. It can also be shown that

$$u_j^H v_k = 0, \qquad j \neq k, j = 1, \cdots, p, k = 1, \cdots, q.$$

For an introduction to these concepts, we refer to [1]. An up to date list of references can be found in [9].

Principal angles and vectors have many important applications in statistics and numerical analysis. In [7], the statistical models of canonical correlations, factor analysis and stochastic equations are described in these terms. The eigenvalue problem $Ax = \lambda Bx$ can have continuous eigenvalues if the nullspaces associated with A and B intersect [13]. By taking the vectors u_k corresponding to $\cos\theta_k = 1$, we get a unitary basis for the intersection, which can be used to simultaneously deflate A and B. Other applications are found in the theory of approximate least squares [8] and in the computation of invariant subspaces of a matrix [21].

The purpose of this paper is to develop new and more accurate methods for computing principal angles and vectors, when the subspaces are defined as the ranges (or nullspaces) of two given matrices A and B. In Section 2, we describe the standard method of computing canonical correlations and show why this method may give rise to a serious loss of accuracy. Assuming that unitary bases for F and G are known, we derive, in Section 3, formulas for computing principal angles and vectors from the singular values and vectors for certain matrices. To find out how accurately the angles are defined in the presence of uncertainties in A and B, first order perturbation results are given in Section 4. In Section 5, different numerical methods for computing the unitary bases, and the use of the formulas from Section 3, are discussed with respect to efficiency and accuracy. The special problems which arise when A and/or B are exactly or nearly rank deficient are discussed in Section 6. Finally, some numerical results are given in Section 7.

2. Canonical Correlations.

For a matrix A, we denote the range of A by $R(A)$ and the nullspace of A by $N(A)$:

$$R(A) = \{u \mid Ax = u\}, \qquad N(A) = \{x \mid Ax = 0\}. \tag{3}$$

In the problem of canonical correlations, we have $F = R(A)$, $G = R(B)$ where A and B are given rectangular matrices. Then, the canonical correlations are equal to $\cos\theta_k$, and it can be shown that

$$\cos\theta_k = \sigma_k, \qquad u_k = Ay_k, \qquad v_k = Bz_k, \qquad k = 1, 2, \cdots, q, \tag{4}$$

where $\sigma_k \geqq 0$ are eigenvalues and y_k, z_k properly normalized eigenvectors to the generalized eigenvalue problem

$$\begin{pmatrix} 0 & A^H B \\ B^H A & 0 \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = \sigma \begin{pmatrix} A^H A & 0 \\ 0 & B^H B \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix}. \tag{5}$$

Assume for convenience that A and B have full column rank. The standard method [6] of computing canonical correlations is to compute A^HA, B^HB, A^HB and perform the Choleski decompositions

$$A^HA = R_A^HR_A, \qquad B^HB = R_B^HR_B,$$

where R_A and R_B are upper triangular.

The eigenvalue problem (5) is then equivalent to the eigenvalue problems for a pair of Hermitian matrices

$$MM^H\hat{y}_i = \sigma_i^2\hat{y}_i, \qquad M^HM\hat{z}_i = \sigma_i^2\hat{z}_i$$

where

$$M = (R_A^H)^{-1}(A^HB)R_B^{-1}, \qquad \hat{y}_i = R_Ay_i, \qquad \hat{z}_i = R_Bz_i.$$

These can be solved by standard numerical methods.

When $q = 1$ and $B = b$, the principal angles and vectors are closely related to the least squares problem of minimizing $||b - Ax||_2$. In fact, with the notations above (but dropping subscripts), we have

$$y = x/||Ax||_2, \qquad z = 1/||b||_2, \qquad \sigma = ||Ax||_2/||b||_2,$$

and (5) is reduced to

$$A^Hbz = \sigma A^HAy, \qquad b^HAy = \sigma b^Hbz.$$

But the first equation here is the normal equations for $x = \sigma y/z$. Thus, the classical algorithm reduces for $q = 1$ to solution of the normal equations by Choleski's method.

Lately it has been stressed by several authors that forming the normal equations in single precision involves a loss of information which cannot be retrieved. For linear least squares problems, other methods without this disadvantage have been developed ([2], [16] and [17]). Our aim in this paper is to generalize these methods to the case when $q > 1$.

3. Solution Using Singular Values. In most applications, each subspace is defined as the range, or the complement of the range, of a given matrix. In this case, a unitary basis for the subspace may be computed in a numerically stable way by well-known methods for the QR-decomposition of a matrix. These methods will produce for an $m \times n$ matrix A, with $m \geqq n$, a decomposition

$$A = (Q' \mid Q'')\left(\begin{array}{c} S \\ \hline 0 \end{array}\right)\begin{array}{l} \}\,p \times n \\ \}\,(m - p) \times n \end{array}$$

where $\operatorname{rank}(S) = p$ and $Q = (Q' \mid Q'')$ is unitary. Then Q' gives a unitary basis for the range of A, $R(A)$, and Q'' a unitary basis for the complement $[R(A)]^{\perp}$. Notice that the case when a subspace is defined as the nullspace $N(A^H)$ of a matrix A^H is included, since $N(A^H) = [R(A)]^{\perp}$. The computation of unitary bases will be discussed in more detail in Sections 5 and 6, and we assume here that such bases have been obtained.

Recently, an efficient and numerically stable algorithm for computing the singular value decomposition [11] (SVD) of a matrix has been developed [17]. This algorithm will be our basic tool for computing principal angles and vectors. The relation between singular values and our problem is clear from the following theorem.

THEOREM 1. *Assume that the columns of Q_A and Q_B form unitary bases for two subspaces of a unitary space E^m. Put*

$$M = Q_A^H Q_B, \tag{6}$$

and let the SVD of this $p \times q$ matrix be

$$M = YCZ^H, \qquad C = \operatorname{diag}(\sigma_1, \cdots, \sigma_q), \tag{7}$$

where $Y^H Y = Z^H Z = ZZ^H = I_q$. If we assume that $\sigma_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_q$, then the principal angles and principal vectors associated with this pair of subspaces are given by

$$\cos \theta_k = \sigma_k(M), \qquad U = Q_A Y, \qquad V = Q_B Z. \tag{8}$$

Proof. It is known [18] that the singular values and singular vectors of a matrix M can be characterized by

$$\sigma_k = \max_{\|y\|_2 = \|z\|_2 = 1} (y^H M z) = y_k^H M z_k, \tag{9}$$

subject to

$$y^H y_j = z^H z_j = 0, \qquad j = 1, \cdots, k-1.$$

If we put

$$u = Q_A y \in R(Q_A), \qquad v = Q_B z \in R(Q_B),$$

then it follows that $\|u\|_2 = \|y\|_2$, $\|v\|_2 = \|z\|_2$ and

$$y^H y_j = u^H u_j, \qquad z^H z_j = v^H v_j.$$

Since $y^H M z = y^H Q_A^H Q_B z = u^H v$, (9) is equivalent to

$$\sigma_k = \max_{\|u\|_2 = \|v\|_2 = 1} (u^H v) = u_k^H v_k$$

subject to

$$u^H u_j = v^H v_j = 0, \qquad j = 1, \cdots, k-1.$$

Now (8) follows directly from the definition of principal angles and vectors (2), which concludes the proof.

For small angles, θ_k is not well determined from $\cos \theta_k$ and we now develop formulas for computing $\sin \theta_k$. Let Q_A and Q_B be defined as in Theorem 1. For convenience, we change the notations slightly and write (7) and (8) as

$$M = Y_A C Y_B^H, \qquad U_A = Q_A Y_A, \qquad U_B = Q_B Y_B. \tag{10}$$

We split Q_B according to

$$Q_B = P_A Q_B + (I - P_A) Q_B = P_A Q_B + P_A^{\perp} Q_B, \tag{11}$$

where $P_A = Q_A Q_A^H$ is the orthogonal projection onto $R(Q_A)$. Here

$$P_A Q_B = Q_A Q_A^H Q_B = Q_A M = Q_A Y_A C Y_B^H,$$

and hence the SVD of the matrix $P_A Q_B$ is given by

$$(12) \qquad P_AQ_B = U_ACY_B^H, \qquad C = \mathrm{diag}(\cos\theta_k).$$

Since $P_A(I - P_A) = 0$, we get, from squaring (11),

$$Q_B^H(I - P_A)^2Q_B = I - Q_B^HP_A^2Q_B = Y_B(I - C^2)Y_B^H.$$

It follows that the SVD of $(I - P_A)Q_B$ can be written

$$(13) \qquad (I - P_A)Q_B = W_ASY_B^H, \qquad S = \mathrm{diag}(\sin\theta_k).$$

Comparing (13) with (12), it is evident that W_A gives the principal vectors in the complement $[R(Q_A)]^{\perp}$ associated with the pair of subspaces $([R(Q_A)]^{\perp}, R(Q_B))$.

We will, for the rest of this section, assume that, in addition to (1), we have $p + q \leqq m$. (This is no real restriction, since, otherwise, we have $(m - p) + (m - q) \leqq m$, and we can work with the complements of $R(Q_A)$ and $R(Q_B)$ instead.) Then, $\dim([R(Q_A)]^{\perp}) = m - p \geqq q$, and we can choose the $m \times q$ matrix W_A in (13) so that $W_A^HU_A = 0$.

By analogy, we have formulas similar to (12) and (13) related to the splitting $Q_A = P_BQ_A + (I - P_B)Q_A$,

$$(14) \qquad P_BQ_A = U_BCY_A^H, \qquad (I - P_B)Q_A = W_BSY_A^H,$$

where again, since $m - q \geqq p \geqq q$, we can choose the $m \times q$ matrix W_B so that $W_B^HU_B^H = 0$. From (14), we get

$$U_A = Q_AY_A = (U_BC + W_BS)Y_A^HY_A = (U_B \; W_B)\begin{pmatrix} C \\ S \end{pmatrix}.$$

If we put

$$P_{B,A} = U_AU_B^H = (U_B \; W_B)\begin{pmatrix} C \\ S \end{pmatrix}U_B^H,$$

then, since $R(Q_B) = R(U_B)$, we have for any $x \in R(Q_B)$ that $P_{B,A}x = U_AU_B^H(U_By) = U_Ay$, and thus

$$P_{B,A}x \in R(Q_A), \qquad ||x||_2 = ||P_{B,A}x||_2.$$

We can now always find an $m \times (m - 2q)$ matrix Z_B such that $(U_BW_BZ_B)$ is a unitary basis in E^m. Then

$$(15) \qquad P_{B,A} = (U_BW_B \mid Z_B)\left[\begin{array}{cc|c} C & -S & 0 \\ S & C & \\ \hline 0 & & I \end{array}\right]\left[\begin{array}{c} U_B^H \\ W_B^H \\ \hline Z_B^H \end{array}\right]$$

is the matrix of a unitary transformation, mapping $R(Q_B)$ into $R(Q_A)$. Its restriction to $R(Q_B)$ is $P_{B,A}$, and it leaves all vectors in $R(Z_B)$ unchanged. This transformation is called a *direct rotation* [9] from $R(Q_B)$ into $R(Q_A)$. It is distinguished from other unitary transformations P taking $R(Q_B)$ into $R(Q_A)$ by the property that it minimizes each unitarily invariant norm of $(I - P)^H(I - P)$. If $R(Q_B) \cap [R(Q_A)]^{\perp}$ is empty, then all $\theta_k < \pi/2$ and the direct rotation is uniquely determined.

Similarly, we can construct a direct rotation taking $R(U_A)$ into $R(Q_B)$. It is obvious that the relations between the two subspaces are very completely characterized by the quantities C, S, U_A, W_A, U_B and W_B.

4. Perturbation of Principal Angles. We consider here how the principal angles between $R(A)$ and $R(B)$ change when the elements in A and B are subject to perturbations. We assume in this analysis that the matrices A and B are $m \times p$ and $m \times q$, respectively, and have linearly independent columns. Consider first a perturbation of A only,

$$A_\epsilon = A + \epsilon E = (A + \epsilon E_1) + \epsilon E_2,$$

where we have split the perturbation into components in and orthogonal to $R(A)$,

$$E_1 = P_A E, \qquad E_2 = (I - P_A)E.$$

Let the polar decomposition of $A + \epsilon E_1$ be

$$A + \epsilon E_1 = Q_A H_A, \qquad Q_A^H Q_A = I, \qquad H_A \text{ positive definite.}$$

Then, since $R(A) = R(A + \epsilon E_1)$, Q_A gives a unitary basis for $R(A)$.

To get a unitary basis for $R(A_\epsilon)$, we first note that for small absolute values of ϵ, the matrix

$$(A + \epsilon E)H_A^{-1} = Q_A + \epsilon F_2, \qquad F_2 = (I - P_A)F, \qquad F = EH_A^{-1},$$

is nearly orthogonal. Indeed, since $Q_A^H F_2 = Q_A^H P_A F_2 = 0$, we have

$$S = I - (Q_A + \epsilon F_2)^H (Q_A + \epsilon F_2) = -\epsilon^2 F_2^H F_2,$$

and

$$\sigma_1(S) = \epsilon^2 \cdot \sigma_1^2(F_2) \leqq \epsilon^2 \sigma_1^2(F). \tag{16}$$

Then, using a series expansion from [4] for the unitary factor Q_{A_ϵ} in the polar decomposition of $A_\epsilon H_A^{-1}$, it follows that

$$\begin{aligned}(Q_A + \epsilon F_2) &= Q_{A_\epsilon}(I - S)^{1/2} \\ &= Q_{A_\epsilon}\left(I - \tfrac{1}{2}S - \tfrac{1}{8}S^2 - \cdots - \binom{\frac{1}{2}}{p}S^p - \cdots\right),\end{aligned} \tag{17}$$

where the matrix series converges if $\rho(S) = \sigma_1(S) < 1$.

Also, asymptotically, when $\epsilon \to 0$, Q_{A_ϵ} is the unitary matrix with range $R(A_\epsilon)$ which is closest to Q_A.

From the well-known inequalities for singular values [15, p. 30],

$$\sigma_k(A + B) \leqq \sigma_k(A) + \sigma_k(B), \qquad \sigma_k(AB) \leqq \sigma_k(A)\sigma_k(B), \qquad k = 1, 2, \cdots,$$

we get

$$\sigma_1(F) \leqq \sigma_1(E)\sigma_1(H_A^{-1}) \leqq \sigma_1(E)/(\sigma_p(A) - \epsilon\sigma_1(E_1)). \tag{18}$$

Since certainly $\sigma_1(E) \geqq \sigma_1(E_1)$, a sufficient condition for convergence of (17) is that

$$\epsilon\sigma_1(E)/\sigma_p(A) \leqq \tfrac{1}{2}.$$

Premultiplying (17) by P_B, we get

$$P_B Q_{A_\epsilon} = P_B Q_A + \epsilon P_B(I - P_A)F + P_B Q_{A_\epsilon}\left(\tfrac{1}{2}S + \tfrac{1}{8}S^2 + \cdots + \binom{\frac{1}{2}}{p}S^p + \cdots\right)$$

from which we derive the inequality

$$|\sigma_k(P_B Q_{A_\epsilon}) - \sigma_k(P_B Q_A)| \leqq \epsilon \cdot \sigma_1(P_B(I - P_A)) + \tau,$$

where

$$\tau = \tfrac{1}{2}\sigma_1(S) + \tfrac{1}{8}\sigma_1^2(S) + \cdots \leqq \tfrac{1}{2}\sigma_1(S)/(1 - \sigma_1(S)).$$

Now, we have $\sigma_1(P_B(I - P_A)) = \sin \theta_{\max}$ and, estimating $\sigma_1(S)$ and $\sigma_1(F)$ by (16) and (18), it follows that

$$(19) \qquad |\Delta \cos \theta_k| \leqq \delta \cdot \sin \theta_{\max} + O(\delta^2), \qquad \delta = \epsilon\sigma_1(E)/\sigma_p(A).$$

If instead we premultiply (17) by $(I - P_B)$ and proceed in the same way, we arrive at

$$(20) \qquad |\Delta \sin \theta_k| \leqq \delta \cdot \cos \theta_{\min} + O(\delta^2), \qquad \delta = \epsilon\sigma_1(E)/\sigma_p(A).$$

Now, assume that both A and B are perturbed by δA and δB, respectively, where

$$||\delta A||_2/||A||_2 \leqq \epsilon_A, \qquad ||\delta B||_2/||B||_2 \leqq \epsilon_B.$$

Then

$$\cos \theta_k = [\sigma_k(P_{B_\epsilon} Q_{A_\epsilon}) - \sigma_k(P_{B_\epsilon} Q_A)] + [\sigma_k(P_{B_\epsilon} Q_A) - \sigma_k(P_B Q_A)]$$

and, from (19), we get

$$|\Delta \cos \theta_k| \leqq \epsilon_A \kappa(A) \sin \theta_{\max}(A, B_\epsilon) + \epsilon_B \kappa(B) \sin \theta_{\max}(A, B) + O(\delta^2),$$

where

$$(21) \qquad \kappa(A) = \sigma_1(A)/\sigma_p(A), \qquad \kappa(B) = \sigma_1(B)/\sigma_q(B), \qquad \delta = \epsilon_A \kappa(A) + \epsilon_B \kappa(B).$$

A corresponding estimate holds for $|\Delta \sin \theta_k|$. Obviously, we have $\theta(A, B_\epsilon) = \theta(A, B) + O(\delta)$, and, thus, these estimates can be simplified to

$$\left|\Delta \begin{Bmatrix} \cos \theta_k \\ \sin \theta_k \end{Bmatrix}\right| \leqq \delta \begin{Bmatrix} \sin \theta_{\max} \\ \cos \theta_{\min} \end{Bmatrix} + O(\delta^2).$$

Combining these two estimates yields

$$|2 \cdot \sin \tfrac{1}{2}\Delta\theta_k| \leqq \delta g(\hat{\theta}_k) + O(\delta^2)$$

where

$$g(\theta) = \min\left(\frac{\sin \theta_{\max}}{\sin \theta}, \frac{\cos \theta_{\min}}{\cos \theta}\right), \qquad \hat{\theta}_k = \theta_k + \tfrac{1}{2}\Delta\theta_k.$$

The maximum of $g(\theta)$ for $0 \leqq \theta \leqq \pi/2$ is attained for

$$\theta = \arctan(\sin \theta_{\max}/\cos \theta_{\min}).$$

Since $2 \cdot \sin \frac{1}{2}\Delta\theta = \Delta\theta + O(\Delta\theta^3)$, it follows that

$$(22) \qquad |\Delta\theta_k| \leqq g_{\max}(\epsilon_A \kappa(A) + \epsilon_B \kappa(B)) + O(\delta^2),$$

where

$$g_{\max} = (\sin^2 \theta_{\max} + \cos^2 \theta_{\min})^{1/2} \leqq 2^{1/2}.$$

We conclude that when both $\kappa(A)$ and $\kappa(B)$ are small, then the angles θ_k are well determined.

We note that, if the columns in A are scaled, then $\kappa(A)$ will change but not $R(A)$. Also, the numerical algorithms for the QR-decomposition have the property that, unless column pivoting is used, they give the same *numerical* results independent of such a scaling. Therefore, it is often more relevant to take in (21) as condition number for A the number

$$\kappa'(A) = \min_D \kappa(AD), \qquad D = \operatorname{diag}(d_1, \cdots, d_p).$$

It has been shown in [20] and [21] that $\kappa(AD)$ is not more than a factor of $p^{1/2}$ away from its minimum, if in AD all columns have equal L_2-norm. This suggests that A and B should be assumed to be preconditioned so that

$$||a_i||_2 = ||b_j||_2 = 1, \qquad i = 1, \cdots, p, j = 1, \cdots, q.$$

We remark that $\kappa'(A)$ is essentially the spanning precision of the basis in $R(A)$ provided by A as defined in [21].

5. Numerical Methods. We assume in this section that the columns in A and B are linearly independent. The singular and near singular case will be briefly discussed in Section 6. For convenience, we also assume that A and B are real matrices, although all algorithms given here can easily be generalized to the complex case. Computed quantities will be marked by a bar.

In order to get the orthogonal bases for F and G, we need the QR-decompositions of the matrices A and B. We now describe two efficient methods for computing these. In the method of *Householder triangularizations* (HT) [16], orthogonal transformations of the type $Q_k = I - 2w_kw_k^T$ are used, where

$$w_k = (0, \cdots, 0, w_{kk}, \cdots, w_{mk})^T, \qquad ||w_k||_2 = 1.$$

The $m \times p$ matrix A is reduced to triangular form using premultiplications

$$Q_p \cdots Q_2Q_1A = \begin{pmatrix} R_A \\ \hline 0 \end{pmatrix} \begin{matrix} \}p \\ \}m-p \end{matrix}$$

where w_k is chosen so that Q_k annihilates the appropriate elements in the kth column. Since $Q_k^{-1} = Q_k$, orthogonal bases Q_A for $R(A)$ can then be computed by premultiplying the first p columns in the unit matrix I_m by the same transformations in reversed order,

$$Q_A = Q_1Q_2 \cdots Q_p\begin{pmatrix} I_p \\ \hline 0 \end{pmatrix}.$$

For this method, a very satisfactory error analysis is given in [23].

Assume that floating point arithmetic with a mantissa of t binary digits is used, and that inner-products are accumulated in double precision wherever possible. Then, there exists an *exactly* orthogonal matrix Q such that the computed matrices satisfy

$$(23) \qquad Q^T(A + E_A) = \begin{pmatrix} \bar{R}_A \\ \hline 0 \end{pmatrix}, \qquad \bar{Q}_A = Q\begin{pmatrix} I_p \\ \hline 0 \end{pmatrix} + F_A = Q_{\bar{A}} + F_A,$$

$$||E_A||_F = 12.5p2^{-t}\,||A||_F, \qquad ||F_A||_F = 12.5p^{3/2}2^{-t},$$

where $Q_{\bar{A}}$ is an exactly orthogonal basis for $R(A + E_A)$. From this and a similar estimate for $\bar{Q}_B$, we get

$$|\sigma_k(\bar{M}) - \sigma_k(\tilde{M})| \leqq \sigma_1(\bar{M} - \tilde{M}) \leqq 13.0(p^{3/2} + q^{3/2})2^{-t}, \tag{24}$$

where $\tilde{M} = Q_{\bar{A}}^T Q_{\bar{B}}$ and the constant 13.0 accounts for the rounding errors in computing the product $\bar{Q}_A^T \bar{Q}_B$. We have $\sigma_k(\tilde{M}) = \cos \tilde{\theta}_k$, where $\tilde{\theta}_k$ are the exact angles between $(A + E_A)$ and $(B + E_B)$. Thus, the difference between $\tilde{\theta}_k$ and θ_k can be estimated from (22),

$$|\tilde{\theta}_k - \theta_k| \leqq 12.5 \cdot 2^{1/2} \cdot \delta + O(\delta^2), \qquad \delta = (p\kappa(A) + q\kappa(B))2^{-t}. \tag{25}$$

Finally, the errors $\bar{\sigma}_k(\bar{M}) - \sigma_k(\bar{M})$ in computing the singular values of $\bar{M}$, using the procedure in [17], will be of the same order of magnitude as those in (24).

The error estimate given above is satisfactory, except when $\theta_k \ll 1$. In this case, the errors in $\cos \theta_k$ from (24) will give rise to errors in θ_k which may be much larger than those in (25). We return later to the problem of accurately computing small angles.

An orthogonal basis Q_A' for $[R(A)]^\perp = N(A^T)$ can be obtained by applying the transformations Q_k, $k = p, \cdots, 1$, to the last $(m - p)$ columns in I_m,

$$Q_A' = Q_1 Q_2 \cdots Q_p \left(\frac{0}{I_{m-p}} \right).$$

Also, in this case, the estimate (23) for $\bar{Q}_A'$, (24) and (25) still hold if the factor $p^{3/2}$ is everywhere replaced by $p(m - p)^{1/2}$.

The QR-decomposition of a matrix A can also be computed using the modified Gram-Schmidt method (MGS) [2]. The matrix A is then transformed in p steps, $A = A_1, A_2, \cdots, A_{p+1} = Q_A$ where

$$A_k = (q_1, \cdots, q_{k-1}, a_k^{(k)}, \cdots, a_p^{(k)}).$$

The matrix A_{k+1}, $k = 1, 2, \cdots, p$, is computed by

$$q_k = a_k^{(k)} / ||a_k^{(k)}||_2, \qquad a_j^{(k+1)} = (I - q_k q_k^T) a_j^{(k)}, \qquad j > k,$$

and the elements in the kth row of R_A are

$$r_{kk} = ||a_k^{(k)}||_2, \qquad r_{kj} = q_k^T a_j^{(k)}, \qquad j > k.$$

It has been shown in [2, pp. 10, 15] that the computed matrices $\bar{R}_A$ and $\bar{Q}_A$ satisfy

$$\begin{aligned} A + E_A &= \bar{Q}_A \bar{R}_A, \qquad ||E_A||_F \leqq 1.5(p - 1)2^{-t}\, ||A||_F, \\ &||Q_{\bar{A}} - \bar{Q}_A||_2 \leqq p(p + 1)\kappa(A) \cdot 2^{-t}, \end{aligned} \tag{26}$$

where $Q_{\bar{A}}$ is an exactly orthogonal basis for $R(A + E_A)$ and quantities of order $\kappa^2(A)2^{-2t}$ have been neglected. With MGS, $\bar{Q}_A$ will in general not be orthogonal to working accuracy, and, therefore, we cannot hope to get principal vectors which are nearly orthogonal. Also, the condition numbers $\kappa(A)$ and $\kappa(B)$ will enter in the estimate corresponding to (24). However, since $\kappa(A)$ and $\kappa(B)$ already appear in (25), we can hope to get the principal angles as accurately as with HT. Experimental results reported in Section 7 indicate that this actually is the case.

An advantage with MGS is that the total number of multiplications required to compute $\bar{R}_A$ and $\bar{Q}_A$ is less than for HT, i.e.,

$$\text{MGS: } p^2 m, \qquad \text{HT: } 2p^2(m - p/3).$$

If only the principal angles are wanted, then the number of multiplications in the SVD-algorithm is approximately $2q^2(p - q/3)$. Thus, when $m \gg p$, the dominating work is in computing Q_A and Q_B and, in this case, MGS requires only half as much work as HT. If also the principal vectors are wanted, we must compute the full SVD of $M = Y_A C y_B^H$. Assuming two iterations per singular value, this requires approximately $7q^2(p + 10q/21)$ multiplications. To compute $U_A = Q_A Y_A$ and $U_B = Q_B Y_B$ a further $mq(p + q)$ multiplications are needed.

To get a basis for $[R(A)]^-$ using MGS, we have to apply the method to the bordered matrix $(A \mid I_m)$, and, after m steps, pick out $(m - p)$ appropriate columns. Especially when $(m - p) \ll m$, the number of multiplications compares unfavourably with HT,

$$\text{MGS: } m^2(m + 2p), \qquad \text{HT: } 2mp(m - p) + \tfrac{2}{3}p^3.$$

In some applications, e.g. canonical correlations, we want to express the principal vectors as linear combinations of the columns in A and B, respectively. We have $U_A = Q_A Y_A = A(R_A^{-1} Y_A)$, and hence

$$U_A = A X_A, \qquad U_B = B X_B,$$

where

$$X_A = R_A^{-1} Y_A, \qquad X_B = R_B^{-1} Y_B. \tag{27}$$

We remark that if we let $\bar{X}_A$ and $\bar{X}_B$ denote the computed matrices, then $A\bar{X}_A$ and $B\bar{X}_B$ will *not* in general be orthogonal to working accuracy even when HT is used.

We now turn to the problem of accurately determining small angles. One method is to compute $\sin \theta_k$ from the SVD (13) of the matrix

$$G = (I - P_A)Q_B = Q_B - Q_A M. \tag{28}$$

If we put $\tilde{G} = Q_{\bar{B}} - Q_{\bar{A}}\bar{M}$, then, using $\bar{Q}_A = Q_{\bar{A}} + F_A$, we get

$$\bar{Q}_B + \bar{Q}_A(\bar{Q}_A^T Q_B) = \tilde{G} + (I - Q_{\bar{A}} Q_{\bar{A}}^T) F_B + (Q_{\bar{A}} F_A^T + F_A Q_{\bar{A}}^T)(Q_{\bar{B}} + F_B).$$

Neglecting second order quantities,

$$||\bar{Q} - \tilde{G}||_2 \leqq ||F_B||_2 + 2\ ||F_A||_2 + 2q^{1/2}2^{-t},$$

where the last term accounts for the final rounding of the elements in $\bar{M}$ and $\bar{\bar{G}}$. Thus, if $\bar{Q}_A$ and $\bar{Q}_B$ are computed by HT, we have, from (23),

$$|\sigma_k(\bar{G}) - \sigma_k(\tilde{G})| \leqq 13.2(q^{3/2} + 2p^{3/2})2^{-t}.$$

It follows that the singular values of the computed matrix $\bar{G}$ will differ little from $\sin \tilde{\theta}_k$, and, thus, small angles will be as accurately determined as is allowed by (25). From (26), the corresponding error estimate for MGS is obtained. In the spirit of the modified Gram-Schmidt method, the matrix G should be computed as

$$G = (I - q_1 q_1^T) \cdots (I - q_p q_p^T) Q_B. \tag{29}$$

Computational experience indicates that this gives much improved accuracy in $\sin\theta$ when $\kappa(A) \gg \kappa(B)$.

Since the matrix G is $m \times q$, computing the singular values of G when $m \gg q$ will require about $2mq^2$ multiplications. If Y_A has been computed, then the SVD of G may be computed with only mq^2 multiplications from

$$GY_B = (I - P_A)Q_B Y_B = W_A S. \tag{30}$$

Moreover, if U_A and U_B are available, we can obtain $\sin\theta$ from

$$(U_B - U_A C)^T(U_B - U_A C) = S^2 \tag{31}$$

or, alternatively,

$$(U_B - U_A)^T(U_B - U_A) = 2(I - C). \tag{32}$$

From the last equation, we can compute $2\sin\frac{1}{2}\theta_k = (2(1 - \cos\theta_k))^{1/2}$, which, since $0 \leqq \frac{1}{2}\theta_k \leqq \pi/4$, accurately determines both $\sin\theta_k$ and $\cos\theta_k$.

We finally remark about an apparent imperfection of MGS. When $A = B$ (exactly), we will obviously get $\bar{Q}_A = \bar{Q}_B$. The exact angle equals zero, and HT will always give computed angles near zero. This is not true for MGS, however, since we only have the estimate

$$\|I - \bar{Q}_A^T\bar{Q}_A\|_2 \leqq 2p(p+1)\kappa(A)2^{-t}.$$

Therefore, the singular values of $\bar{M} = \bar{Q}_A^T\bar{Q}_A$ may not be near one when $\kappa(A)$ is large. If, however, only $A \simeq B$, then the rounding errors in computing Q_A and Q_B will not be correlated, and in an ill-conditioned case, we will probably not get all angles near zero either with HT or MGS.

When $A = B$, then $\bar{M} = \bar{Q}_A^T\bar{Q}_A$ will be symmetric and, thus, SVD will give $\bar{Y}_A \simeq \bar{Y}_B$ and, therefore, $\bar{U}_A \simeq \bar{U}_B$ also with MGS. It follows that, if (32) is used, MGS will always yield angles near zero in this case.

We have not tried to determine error estimates for the methods based on (30)–(32). On the test matrices described in Section 7, the method based on (28) gave significantly more accurate results, especially for the more well-conditioned angles.

6. **The Singular Case.** We now consider the case when A and/or B does not have full column rank. In this case, the problem of computing principal angles and vectors is not well posed, since arbitrarily small perturbations in A and B will change the rank of A and/or B. The main computational difficulty then lies in assigning the correct rank to A and B. The most satisfactory way of doing this generally is the following [10]. Let the $m \times p$ matrix A have the SVD

$$A = Q_A D_A V_A^T, \qquad D_A = \operatorname{diag}(\sigma_k(A)).$$

Let ϵ be a suitable tolerance and determine $p' \leqq p$ from

$$\sum_{i=p'+1}^{n} \sigma_i^2(A) \leqq \epsilon^2 < \sum_{i=p'}^{n} \sigma_i^2(A). \tag{33}$$

We then approximate A with an $m \times p$ matrix A' such that $\operatorname{rank}(A') = p'$,

$$A' = (Q_A' Q_A'')\begin{pmatrix} D_A' & 0 \\ 0 & 0 \end{pmatrix}(V_A' \, V_A'')^T, \qquad D_A' = \operatorname{diag}(\sigma_1, \cdots, \sigma_{p'}),$$

where $Q_A = (Q_A' Q_A'')$, $V_A = (V_A' V_A'')$ have been partitioned consistently with the diagonal matrix. The matrix B is approximated in the same way.

If, instead of (1), we assume that

$$p' = \operatorname{rank}(A') \geqq \operatorname{rank}(B') = q' \geqq 1,$$

then we can compute the principal angles and vectors associated with $R(A')$ and $R(B')$ by the previously derived algorithms, where now Q_A' and Q_B' should replace Q_A and Q_B.

In order to express the principal vectors of $R(A')$ as linear combinations of columns in A', we must solve the compatible system

$$A' X_A = U_A = Q_A' Y_A.$$

Since V_A'' is an orthogonal basis for $N(A)$, the general solution can be written

$$X_A = V_A' D_A'^{-1} Y_A + V_A'' C_A,$$

where C_A is an arbitrary matrix. It follows that, by taking $C_A = 0$, we get the unique solution which minimizes $\|X_A\|_F$; cf. [17]. Thus, we should take

$$X_A = V_A' D_A'^{-1} Y_A, \qquad X_B = V_B' D_B'^{-1} Y_B, \tag{34}$$

where X_A is $p \times p'$ and X_B is $q \times q'$.

The approach taken above also has the advantage that only one decomposition, the SVD, is used throughout. It can, of course, also be used in the nonsingular case. However, computing the SVD of A and B requires much more work than computing the corresponding QR-decompositions. In order to make the QR-methods work also in the singular case, column pivoting must be used. This is usually done in such a way ([2], [12] and [16]) that the triangular matrix $R = (r_{ij})$ satisfies

$$|r_{kk}|^2 \geqq \sum_{i=k}^{j} |r_{ij}|^2, \qquad k < j \leqq n.$$

Such a triangular matrix is called normalized, and, in particular, the sequence $|r_{11}|$, $|r_{22}|, \cdots, |r_{pp}|$ is nonincreasing. In practice, it is often satisfactory to take the numerical rank of A to be p' if for a suitable tolerance ϵ we have

$$|r_{p'p'}| > \epsilon \geqq |r_{p'+1,p'+1}|. \tag{35}$$

We then approximate $A = Q_A R_A$ by a matrix $A' = Q_A R_A'$ of rank p' by putting

$$r'_{ij} = r_{ij}, \qquad i \leqq p', \qquad r'_{ij} = 0, \qquad i > p'.$$

It has been shown in [14] how to obtain the solution (32) of minimum length from this decomposition.

If we use the criterion (33), there is a risk of choosing p' too large. Indeed, it seems difficult to improve on the inequalities [12]

$$3(4^k + 6k - 1)^{-1/2} |r_{kk}| \leqq \sigma_k(A) \leqq (n + k + 1)^{1/2} |r_{kk}| \tag{36}$$

from which it is seen that $\sigma_k(A)$ may be smaller than $|r_{kk}|$ by a factor of magnitude 2^{-k}.

However, this rarely occurs in practice. Often the inequality

$$\kappa(A) \geqq |r_{11}|/|r_{pp}|,$$

represents a pretty good approximation to the condition number for the nonsingular case.

7. Test Results. The algorithms in Section 5 have been tested on the UNIVAC 1108 of Lund University. Single precision floating-point numbers are represented by a normalized 27 bit mantissa, whence the machine precision is equal to $2^{-26} \approx 1.5 \cdot 10^{-8}$.

For the tests, we have taken $F = R(A)$, where A is the $m \times p$ matrix

$$A = \frac{1}{k^{1/2}} \begin{bmatrix} e & 0 & \cdots & 0 \\ 0 & e & \cdots & 0 \\ & & \cdots & \\ 0 & 0 & \cdots & e \end{bmatrix}, \qquad e = \left.\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}\right\} m/p$$

and m/p is an integer. Thus, A is already orthogonal, and we can take $Q_A = A$. Further, $G = R(B)$ where B is the $m \times p$ Vandermonde matrix

$$B = \begin{bmatrix} 1 & x_0 & \cdots & x_0^{p-1} \\ 1 & x_1 & \cdots & x_1^{p-1} \\ & & \cdots & \\ 1 & x_{m-1} & \cdots & x_{m-1}^{p-1} \end{bmatrix}, \qquad x_\iota = -1 + 2i/(m+1).$$

The condition number $\kappa(B)$ is known to grow exponentially with p, when the ratio m/p is kept constant. These matrices A and B are the ones appearing in [6]. There is exactly one vector, $u = (1, 1, \cdots, 1)^T$, which belongs to both F and G, so there will be one minimum angle $\theta = 0$.

For the tests, the matrix B was generated in double precision and then rounded to single precision. The procedures used for the QR-decompositions are apart from minor details identical with procedures published in [3] and [5]. The columns were implicitly scaled to have equal L_2-norm and column pivoting was performed. Inner products were *not* accumulated in double precision. For checking purposes, a three term recurrence relation [8] was used in double precision to compute an exact single precision orthogonal basis for $R(B)$.

For $m/p = 2$ and $p = 5(2)17$, Q_A was computed both by the method of Householder and the modified Gram-Schmidt method. Then $\cos \theta_k$, Y_A and Y_B were com-

TABLE 1

		Householder			Gram-Schmidt		
m	p	$F(\bar{U}_A)$	$F(\bar{U}_B)$	$m(\cos \bar{\theta}_k)$	$F(\bar{U}_A)$	$F(\bar{U}_B)$	$m(\cos \bar{\theta}_k)$
10	5	15	21	5	20	18	4
14	7	22	35	8	27	33	10
18	9	40	27	10	37	437	41
22	11	42	29	38	40	1130	49
26	13	58	48	1416	47	9013	621
30	15	62	51	2535	60	55322	1758
34	17	68	60	7582	76	788466	32650

TABLE 2

$m = 26$	$p = 13$	Householder	Gram-Schmidt
k	$\cos \theta_k$	$\Delta \cos \bar{\theta}_k \cdot 10^8$	$\Delta \cos \bar{\theta}_k \cdot 10^8$
1	0.99999991	2	2
2	0.99823275	0	51
3	0.99814397	− 32	−135
4	0.99032703	6	−137
5	0.98988846	5	351
6	0.97646081	0	− 58
7	0.96284604	38	21
8	0.94148906	− 5	− 10
9	0.91758607	− 31	− 40
10	0.87013717	25	−290
11	0.76365752	1416	620
12	0.06078817	106	− 18
13	0.01558526	− 52	− 55

TABLE 3

		Householder[1]			Gram-Schmidt[1]	
					from (29)	from (28)
m	p	$m(\sin \bar{\theta}_k)$	$m(\sin \tilde{\theta}_k)$	$m(\sin \bar{\theta}_k)$	$m(\sin \tilde{\theta}_k)$	$m(\sin \tilde{\theta}_k)$
10	5	3	2	3	3	6
14	7	7	8	3	4	8
18	9	32	31	55	9	87
22	11	141	142	46	39	612
26	13	1661	1662	366	517	5902
30	15	2919	2912	1290	1355	32537
34	17	7604	7608	37284	798	126731

[1] $\sin \bar{\theta}_k$ computed as $\sigma_k((I - P_A)Q_B)$, $\sin \tilde{\theta}_k$ as $\sigma_k((I - P_B)Q_A)$.

puted by the procedure in [17], and finally U_A and U_B from (11). The results are shown in Table 1, where

$$m(\sigma_k) = \max_k |\sigma_k - \bar{\sigma}_k| \cdot 10^8, \qquad F(U) = ||I - U^T U||_F \cdot 10^8.$$

Notice, that because of rounding errors in the computation of the SVD, the values σ_k are not exact to single precision.

For the Gram-Schmidt method, the predicted lack of orthogonality in U_B, when $\kappa(B)$ is large, is evident. However, there is no significant difference in the accuracy of $\cos \bar{\theta}_k$ between the two methods. In Table 2, we show for $m = 26$ and $p = 13$ the errors in $\cos \bar{\theta}_k$ for each k.

For the same values of m and p, $\sin \theta_k$ were computed from the singular values of both the matrix $(I - P_A)Q_B$ and the matrix $(I - P_B)Q_A$. The results in Table 3 again show no significant difference in accuracy between the two methods. For the Gram-Schmidt method, the values of $\sin \theta_k$ differ somewhat between the two matrices,

TABLE 4

$m = 26$	$p = 13$	Householder[1]		Gram-Schmidt[1]	
k	$\sin \theta_k$	$\Delta \sin \bar{\theta}_k \cdot 10^8$	$\Delta \sin \tilde{\theta}_k \cdot 10^8$	$\Delta \sin \bar{\theta}_k \cdot 10^8$	$\Delta \sin \tilde{\theta}_k \cdot 10^8$
1	0.00000000	2	5	2	4
2	0.05942260	0	0	0	− 2
3	0.06089681	485	484	69	76
4	0.13875174	56	55	2	22
5	0.14184706	− 30	− 29	− 23	− 71
6	0.21569431	32	31	− 35	− 20
7	0.27005038	− 127	− 127	− 20	− 26
8	0.33704301	26	25	− 2	3
9	0.39753669	91	90	− 25	− 5
10	0.49280931	− 44	− 42	260	318
11	0.64562106	−1661	−1662	−365	−517
12	0.99815036	5	11	−158	15
13	0.99987821	14	20	90	13

[1] $\sin \bar{\theta}_k$ computed as $\sigma_k((I - P_A)Q_B)$, $\sin \tilde{\theta}_k$ as $\sigma_k((I - P_B)Q_A)$.

whereas the corresponding values for the Householder method are almost identical· This is confirmed by Table 4, where, again for $m = 26, p = 13$, results for each k are shown. For the Gram-Schmidt method the matrix $(I - P_B)Q_A$ was computed both from (28) as $Q_A - Q_B(Q_B^T Q_A)$ and from (29) as $\prod (I - q_k q_k^T)Q_B$. The results in Table 3 clearly show that (29) should be used.

The authors are very pleased to acknowledge the help of Mr. Jan Svensson, who carried out most of the tests described in this section.

Department of Mathematics
Linköping University
S-581 83 Linköping, Sweden

Computer Science Department
Stanford University
Stanford, California 94305

1. S. N. Afriat, "Orthogonal and oblique projectors and the characteristics of pairs of vector spaces," *Proc. Cambridge Philos. Soc.*, v.53, 1957, pp. 800–816. MR **20** #1389.

2. Å. Björck, "Solving linear least squares problems by Gram-Schmidt orthogonalization," *Nordisk Tidskr. Informationsbehandling* (*BIT*), v. 7, 1967, pp. 1–21. MR **53** #5126.

3. Å. Björck, "Iterative refinement of linear least squares solutions. II," *Nordisk Tidskr. Informationsbehandling (BIT)*, v. 8, 1968, pp. 8–30. MR **39** #1094.

4. Å. Björck & C. Bowie, "An iterative algorithm for computing the best estimate of an orthogonal matrix," *SIAM J. Numer. Anal.*, v. 8, 1971, pp. 358–364.

5. P. Businger & G. H. Golub, "Handbook series linear algebra. Linear least squares solutions by Householder transformations," *Numer. Math.*, v. 7, 1965, pp. 269–276. MR **31** #862.

6. C. Cohen & A. Ben-Israel, "On the computation of canonical correlations," *Cahiers Centre Études Recherche Opér.*, v. 11, 1969, pp. 121–132. MR **40** #6683.

7. C. Cohen, *An Investigation of the Geometry of Subspaces for Some Multivariate Statistical Models*, Thesis, Dept. of Indust. Eng., University of Illinois, Urbana, Ill., 1969.

8. G. Dahlquist, B. Sjöberg & P. Svensson, "Comparison of the method of averages with the method of least squares," *Math. Comp.*, v. 22, 1968, pp. 833–845. MR **39** #1099.

9. C. Davies & W. Kahan, "The rotation of eigenvectors by a perturbation. III," *SIAM J. Numer. Anal.*, v. 7, 1970, pp. 1–46.

10. C. Eckart & G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, v. 1, 1936, pp. 211–218.

11. C. Eckart & G. Young, "A principal axis transformation for non-Hermitian matrices," *Bull. Amer. Math. Soc.*, v. 45, 1939, pp. 118–121.

12. D. K. Faddeev, V. N. Kublanovskaja & V. N. Faddeeva, *Sur les Systèmes Linéaires Algébriques de Matrices Rectangulaires et Mal-Conditionnées* (Actes Colloq. Internat. C.N.R.S. No. 165, Besançon, 1966) Éditions C.N.R.S., Paris, 1968, pp. 161–170. MR **37** #6017.

13. G. Fix & R. Heiberger, "An algorithm for the ill conditioned generalized eigenvalue problem," *Numer. Math.* (To appear.)

14. R. J. Hanson & C. L. Lawson, "Extensions and applications of the Householder algorithm for solving linear squares problems," *Math. Comp.*, v. 23, 1969, pp. 787–812. MR **41** #2905.

15. I. C. Gohberg & M. G. Kreĭn, *Introduction to the Theory of Linear Nonselfadjoint Operators*, "Nauka," Moscow, 1965; English transl. Transl. Math. Monographs, vol. 18, Amer. Math. Soc., Providence, R. I., 1969. MR **39** #7447.

16. G. H. Golub, "Numerical methods for solving linear least squares problems," *Numer. Math.*, v. 7, 1965, pp. 206–216. MR **31** #5323.

17. G. H. Golub & C. Reinsch, "Singular value decompositions and least squares solutions," *Numer. Math.*, v. 14, 1970, pp. 403–420.

18. I. J. Good, *The Estimation of Probabilities. An Essay on Modern Bayesian Methods*, Research Monograph No. 30, M.I.T. Press, Cambridge, Mass., 1965, pp. 87–89. MR **32** #3186.

19. H. Hotelling, "Relations between two sets of variates," *Biometrika*, v. 28, 1936, pp. 321–377.

20. A. van der Sluis, "Condition numbers and equilibration of matrices," *Numer. Math.*, v. 14, 1969/1970, pp. 14–23. MR **40** #6760.

21. J. M. Varah, "Computing invariant subspaces of a general matrix when the eigensystem is poorly conditioned," *Math. Comp.*, v. 24, 1970, pp. 137–149. MR **41** #9434.

22. P. Å Wedin, *On Pseudoinverses of Perturbed Matrices*, Dept. of Comput. Sci., Lund University, Lund, Sweden, 1969.

23. J. Wilkinson, "Error analysis of transformations based on the use of matrices of the form $I - 2ww^H$," *Error in Digital Computation*, vol. II (Proc. Sympos. Math. Res. Center, U.S. Army, Univ. of Wisconsin, Madison, Wis., 1965), Wiley, New York, 1965, pp. 77–101. MR **32** #6711.

21

METHODS FOR MODIFYING MATRIX FACTORIZATIONS (WITH P. E. GILL, W. MURRAY AND M. A. SAUNDERS)

Math. Comp. (28), 505–535 (1974).

The paper describes methods for updating the Cholesky factorization of a symmetric positive-definite matrix, and the complete orthogonal factorization of a rectangular matrix. These methods were proposed as an alternative to updating the inverse and pseudo-inverse. The bulk of the paper was written at Stanford over the spring and summer of 1972. The joint paper was proposed by Gene during a visit (with Michael Saunders) to the National Physical Laboratory the previous year. My visit to Stanford was funded by Gene, and I will always be grateful for his support and encouragement during that summer. This was my first visit to the United States and it opened up many opportunities, including the chance to work with my subsequent colleagues Michael Saunders and Margaret Wright.

Philip Gill
San Diego, California, USA

The genesis of the paper dated to 1970 when Gene and Michael visited the Mathematics Division of the National Physical Laboratory on their way to the VII International Symposium on Mathematical Programming in The Hague. Gene was a busy traveller even in those days and frequently visited the NPL (mainly to see Jim Wilkinson). This was the first time I had met Michael. He gave a talk in the office of Charlie Clenshaw and I recall thinking he talked rather slowly and incorrectly put it down to jet lag. Philip and I had been working on updating matrix factorizations that arose in algorithms for optimization. Typically at that time matrix inverses were recurred. Gene and Michael had been considering LP and QP problems. How best to update the factorizations impacted the efficiency, robustness and accuracy of the optimization methods. Since updating matrix factorizations arose in many settings we decided that a paper devoted just to this issue was worthwhile. The main work writing the paper was done at Stanford in the summer of 1972. Gene, Michael and I attended the Householder meeting in Los Alamos and afterwards I made a short visit to Stanford. Philip visited Stanford for the summer during which time the bulk of the paper was written. Michael was busy completing his thesis but still found time to help finish the paper.

Walter Murray
Stanford, California, USA

Gene was advocating stable modifications to matrix factors for many years before this review appeared. The review's existence illustrates two of Gene's outstanding qualities: his sense for when sufficient material exists for a review, and his outreach to young researchers by bringing them together (usually by inviting them to visit Stanford).

Gene opened a new world to me in Summer 1970 when he arranged a visit to the National Physical Laboratory in London, where Jim Wilkinson had long been analyzing numerical factorizations, and where Walter and Philip were applying them to optimization algorithms, my own area of interest. Quasi-Newton methods cried out for Cholesky updating and downdating (Pete Stewart's later terminology), and active-set methods seemed equally in need of QR factors to handle constraints. Gene had already pursued Cholesky and QR methods for modified least-squares problems.

Our paper began when Gene invited Walter and Philip and also Chris Paige to visit Serra House in 1972 just before I graduated. Philip and Chris helped me through those last desperate weeks of thesis-writing while pressing ahead with their own research. I can't help noting that this was a glimpse of their tremendous capacity for hard work, and their Gene-like breadth of vision in covering many approaches in a coordinated way. Their untiring efforts continue to this day.

The paper's title is more general than its contents. Cholesky, QR, and QRZ modifications are covered in depth, but there is no mention of LU factors or sparse matrices. Little did we know that John Reid would soon show how a sparse form of the Bartels–Golub LU update could be implemented efficiently, or that (much later) Davis and Hager would implement sparse versions of our Method C1 for LDL^T updates and downdates.

We have learned one more thing about updating triangular factors: a symmetric permutation works better than a one-sided one when a row or column is permuted to the end. In other words, we should *keep existing diagonal elements on the main diagonal*, thus avoiding Hessenberg matrices. This is a key feature of Reid's Bartels–Golub update (see Nick Higham's review in this volume), and it applies equally to LU and QR updates. Fewer elementary transformations will be needed because the elements to be eliminated are likely to be *sparse.*

On behalf of my coauthors, I send warmest thanks to Gene for fostering our interest in stable numerical methods and for bringing us all together.

Michael Saunders
Stanford, California, USA

MATHEMATICS OF COMPUTATION, VOLUME 28, NUMBER 126, APRIL 1974, PAGES 505–535

Methods for Modifying Matrix Factorizations

P. E. Gill, G. H. Golub, W. Murray and M. A. Saunders*

Abstract. In recent years, several algorithms have appeared for modifying the factors of a matrix following a rank-one change. These methods have always been given in the context of specific applications and this has probably inhibited their use over a wider field. In this report, several methods are described for modifying Cholesky factors. Some of these have been published previously while others appear for the first time. In addition, a new algorithm is presented for modifying the complete orthogonal factorization of a general matrix, from which the conventional QR factors are obtained as a special case. A uniform notation has been used and emphasis has been placed on illustrating the similarity between different methods.

1. Introduction. Consider the system of equations

$$Ax = b$$

where A is an $n \times n$ matrix and b is an n-vector. It is well known that x should be computed by means of some factorization of A, rather than by direct computation of A^{-1}. The same is true when A is an $m \times n$ matrix and the minimal least squares solution is required; in this case, it is usually neither advisable nor necessary to compute the pseudo-inverse of A explicitly (see Peters and Wilkinson [13]).

Once x has been computed, it is often necessary to solve a modified system

$$\bar{A}\bar{x} = \bar{b}.$$

Clearly, we should be able to modify the factorization of A to obtain factors for $\bar{A}$, from which $\bar{x}$ may be computed as before. In this paper, we consider one particular type of modification, in which $\bar{A}$ has the form

$$\bar{A} = A + \alpha yz^T$$

where α is a scalar and y and z are vectors of the appropriate dimensions. The matrix αyz^T is a matrix of rank one, and the problem is usually described as that of *updating* the factors of A following a *rank-one modification.*

There are at least three matters for consideration in computing modified factors:

(a) The modification should be performed in as few operations as possible. This is especially true for large systems when there is a need for continual updating.

(b) The numerical procedure should be stable. Many of the procedures for modifying matrix inverses or pseudo-inverses that have been recommended in the literature are numerically unstable.

(c) If the original matrix is sparse, it is desirable to preserve its sparsity as much as possible. The factors of a matrix are far more likely to be sparse than its inverse.

Received February 26, 1973.

AMS (MOS) subject classifications (1970). Primary 65F30; Secondary 15A06, 15A39, 90C05, 90C20, 90C30.

* The work of G. H. Golub and M. A. Saunders was in part supported by the AEC and the NSF.

Modification methods have been used extensively in numerical optimization, statistics and control theory. In this paper, we describe some methods that have appeared recently, and we also propose some new methods. We are concerned mainly with algebraic details and shall not consider sparsity hereafter. The reader is referred to the references marked with an asterisk for details about particular applications.

1.1. *Notation.* The elements of a matrix A and a vector x will be denoted by a_{ij} and x_j respectively. We will use A^T to denote the transpose of A, and $\|x\|_2$ to represent the 2-norm of x, i.e., $\|x\|_2 = (x^Tx)^{1/2}$. The symbols Q, R, L and D are reserved for matrices which are respectively orthogonal, upper triangular, unit lower triangular and diagonal. In particular, we will write $D = \text{diag}(d_1, d_2, \cdots, d_n)$. The jth column of the identity matrix I will be written as e_j and e will denote the vector $[1, 1, \cdots, 1]^T$.

2. Preliminary Results.

Most of the methods given in this paper are based in some way upon the properties of orthogonal matrices. In the following, we discuss some important properties of these matrices with the intention of using the material in later sections.

2.1. *Givens and Householder Matrices.* The most common application of orthogonal matrices in numerical analysis is the reduction of a given n-vector z to a multiple of a column of the identity matrix, e.g., find an $n \times n$ orthogonal matrix P such that

$$Pz = \pm\rho e_1 \tag{1}$$

or

$$Pz = \pm\rho e_n. \tag{2}$$

This can be done by using either a sequence of plane rotation (Givens) matrices or a single elementary hermitian (Householder) matrix. In order to simplify the notation we will define the former as

$$\begin{bmatrix} c & s \\ s & -c \end{bmatrix} \tag{3}$$

and call this a *Givens matrix* rather than a plane rotation since it corresponds to a rotation followed by a reflection about an axis.

This matrix has the same favorable numerical properties as the usual plane rotation matrix (see Wilkinson [16, pp. 131–152]), but it is symmetric. The choice of c and s to perform the reduction

$$\begin{bmatrix} c & s \\ s & -c \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} +\rho \\ 0 \end{bmatrix}$$

is given by

$$\begin{aligned} &\rho^2 = z_1^2 + z_2^2, \\ &\rho = \text{sign}\,(z_1)(\rho^2)^{1/2} \quad \text{and} \\ &c = z_1/\rho, \qquad s = z_2/\rho. \end{aligned} \tag{4}$$

Note that $0 \leqq c \leqq 1$. In order to perform the reduction (1) or (2), we must embed the matrix (3) in the n-dimensional identity matrix. We shall use $P_j{}^i$ to denote the matrix which, when applied to the vector $[z_1, z_2, \cdots, z_n]^T$, reduces z_j to zero by forming a linear combination of this element with z_i. If $i < j$, then

$$P_j^i z = \begin{array}{c} \begin{array}{ccccccccccc} & & & i & & & & j & & & \end{array} \\ \left[\begin{array}{ccccccccccc} 1 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ & & & c & & & & s & & & \\ & & & & 1 & & & & & & \\ & & & & & \ddots & & & & & \\ & & & & & & 1 & & & & \\ & & & s & & & & -c & & & \\ & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \end{array}\right] \end{array} \left[\begin{array}{c} z_1 \\ \vdots \\ z_i \\ \vdots \\ z_j \\ \vdots \\ z_n \end{array}\right] = \left[\begin{array}{c} z_1 \\ \vdots \\ \bar{z}_i \\ \vdots \\ 0 \\ \vdots \\ z_n \end{array}\right].$$

Alternatively, if $i > j$, the (i, i)th and (j, j)th elements of $P_j{}^i$ are $-c$ and $+c$, respectively. There are several sequences of Givens matrices which will perform the reduction (1) or (2); for example, if we want to reduce z to e_1, we can use

$$P_2^1 P_3^2 \cdots P_{n-1}^{n-2} P_n^{n-1} z \quad \text{or} \quad P_2^1 P_3^1 \cdots P_{n-1}^1 P_n^1 z. \tag{5}$$

To perform the same reduction in one step, using a single Householder matrix, we have

$$P = I + \tau^{-1} u u^T,$$

where

$$\begin{aligned} u &= z + \rho e_1, \\ \tau &= -\rho u_1 \quad \text{and} \\ \rho &= \operatorname{sign}(z_1) \| z \|_2. \end{aligned} \tag{6}$$

This time, P is such that $Pz = -\rho e_1$.

In the 2-dimensional case, we can show that the Householder matrix is of the form

$$P = \begin{bmatrix} -c & -s \\ -s & c \end{bmatrix} = -\begin{bmatrix} c & s \\ s & -c \end{bmatrix}$$

where c, s are the quantities defined earlier for the Givens matrix. Hence, when embedded in n dimensions, the 2×2 Householder and 2×2 Givens transformations are analytically the same, apart from a change of sign. (Although these matrices are $n \times n$, we shall often refer to them as "2×2" orthogonal matrices.)

There are several applications where 2-dimensional transformations are used. The amount of computation needed to multiply a $2 \times n$ matrix A by a 2×2 Householder matrix computed using Eqs. (6) is $4n + O(1)$ multiplications and $3n + O(1)$ additions. If this computation is arranged as suggested by Martin, Peters and Wilkinson [11] and the relevant matrix is written as

$$I + \begin{bmatrix} -u_1/\rho \\ -u_2/\rho \end{bmatrix} [1 \ \ u_2/u_1],$$

then the multiplication can be performed in $3n + O(1)$ multiplications and $3n + O(1)$ additions. Straightforward multiplication of A by a Givens matrix requires $4n + O(1)$ multiplications and $2n + O(1)$ additions. Again, the work can be reduced to $3n + O(1)$ multiplications and $3n + O(1)$ additions, as follows.

Let the Givens matrix be defined as in (4). Define the quantity

$$\mu = z_2/(z_1 + \rho), \qquad |\mu| \leqq 1.$$

Since $s = z_2/\rho$, we can write s as $s = \mu(c + 1)$. Similarly, we have $c = 1 - \mu s$. A typical product can be written in the form

$$(7) \qquad \begin{aligned} \begin{bmatrix} c & s \\ s & -c \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} &= \begin{bmatrix} c & s \\ \mu(c+1) & \mu s - 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \\ &= \begin{bmatrix} y_1 c + y_2 s \\ y_1 \mu(c + 1) + y_2(\mu s - 1) \end{bmatrix} \end{aligned}$$

which will be denoted by

$$= \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \end{bmatrix}.$$

Consequently, in order to perform the multiplication (7), we form

$$\bar{y}_1 = cy_1 + sy_2 \quad \text{and} \quad \bar{y}_2 = \mu(y_1 + \bar{y}_1) - y_2.$$

Note that this scheme is preferable only if the time taken to compute a multiplication is more than the time taken to compute an addition. Also, it may be advisable with both algorithms to modify the computation of ρ to avoid underflow difficulties.

In the following work, we will consider only 2×2 Givens matrices, although the results apply equally well to 2×2 Householder matrices since, as noted earlier, the two are essentially the same.

2.2. *Products of Givens Matrices.* The following results will help define some new notation and present properties of certain products of orthogonal matrices.

LEMMA I. *Let $P_{j+1}{}^{j}$ be a Givens matrix defined as in* (4). *Then the product*

$$P_n^{n-1}P_{n-1}^{n-2}\cdots P_2^1$$

is of the form

$$H_L(p,\beta,\gamma)=\begin{bmatrix} p_1\beta_1 & \gamma_1 & & & & \\ p_2\beta_1 & p_2\beta_2 & \gamma_2 & & & \\ p_3\beta_1 & p_3\beta_2 & p_3\beta_3 & \ddots & & \\ \vdots & \vdots & \vdots & & \gamma_{n-2} & \\ p_{n-1}\beta_1 & p_{n-1}\beta_2 & p_{n-1}\beta_3 & \cdots & p_{n-1}\beta_{n-1} & \gamma_{n-1} \\ p_n\beta_1 & p_n\beta_2 & p_n\beta_3 & \cdots & p_n\beta_{n-1} & p_n\beta_n \end{bmatrix}$$

where the quantities p_i, β_i and γ_i are defined by either of the following recurrence relations:

Forward Recurrence.

1. *Set $p_1 = c_1/\pi$, $\beta_1 = \pi$, $\eta_1 = s_1/\pi$, $\gamma_1 = s_1$, where π is an arbitrary nonzero scalar.*

2. *For $j = 2, 3, \cdots, n-1$, set $p_i = c_i\eta_{i-1}$, $\gamma_i = s_i$, $\beta_i = -c_{i-1}/\eta_{i-1}$, $\eta_i = s_i\eta_{i-1}$.*

3. *Set $p_n = \eta_{n-1}$, $\beta_n = -c_{n-1}/p_n$.*

Backward Recurrence.

1. *Set $p_n = \pi$, $\beta_n = -c_{n-1}/\pi$, $\eta_{n-1} = s_{n-1}/\pi$, $\gamma_{n-1} = s_{n-1}$, where π is an arbitrary nonzero scalar.*

2. *For $j = n-1, n-2, \cdots, 3, 2$, set $p_i = c_i/\eta_i$, $\gamma_{i-1} = s_{i-1}$, $\beta_i = -c_{i-1}\eta_i$, $\eta_{i-1} = s_{i-1}\eta_i$.*

3. *Set $p_1 = c_1/\beta_1$, $\beta_1 = \eta_1$.*

Proof. We will prove the lemma in the forward recurrence case; the remaining case can be proved in a similar way. Assume that the product $P_{k+1}{}^kP_k{}^{k-1}\cdots P_4{}^3P_3{}^2P_2{}^1$ $(k < n-1)$ is given by

$$(8)\qquad \begin{bmatrix} p_1\beta_1 & \gamma_1 & & & & & & \\ p_2\beta_1 & p_2\beta_2 & \ddots & & & & & \\ \vdots & \vdots & & & & & & \\ p_k\beta_1 & p_k\beta_2 & \cdots & p_k\beta_k & \gamma_k & & & \\ \eta_k\beta_1 & \eta_k\beta_2 & \cdots & \eta_k\beta_k & -c_k & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & \ddots \\ & & & & & & & & 1 \end{bmatrix}$$

This is true for $k = 1$ by definition. The next product $P_{k+2}{}^{k+1}P_{k+1}{}^kP_k{}^{k-1}\cdots P_3{}^2P_2{}^1$ is given by

$$\begin{pmatrix}
p_1\beta_1 & \gamma_1 & & & & & & & \\
p_2\beta_1 & p_2\beta_2 & \ddots & & & & & & \\
\vdots & \vdots & \ddots & & & & & & \\
p_k\beta_1 & p_k\beta_2 & \cdots & p_k\beta_k & \gamma_k & & & & \\
c_{k+1}\eta_k\beta_1 & c_{k+1}\eta_k\beta_2 & \cdots & c_{k+1}\eta_k\beta_k & -c_{k+1}c_k & s_{k+1} & & & \\
s_{k+1}\eta_k\beta_1 & s_{k+1}\eta_k\beta_2 & \cdots & s_{k+1}\eta_k\beta_k & -s_{k+1}c_k & -c_{k+1} & & & \\
 & & & & & & 1 & & \\
 & & & & & & & \ddots & \\
 & & & & & & & & 1
\end{pmatrix}.$$

If we define $p_{k+1} = c_{k+1}\eta_k$, $\gamma_{k+1} = s_{k+1}$, $\beta_{k+1} = -c_k/\eta_k$, $\eta_{k+1} = s_{k+1}\eta_k$, then the product $P_{k+2}{}^{k+1} \cdots P_2{}^1$ is of a similar form to (8). Continuing in this way, and finally setting $p_n = \eta_{n-1}$ and $\beta_n = -c_{n-1}/p_n$ gives the required result.

For later convenience, we shall use the notation

$$(H_L(p, \beta, \gamma))^T = H_U(\beta, p, \gamma).$$

The matrices $H_U(\beta, p, \gamma)$ and $H_L(p, \beta, \gamma)$ are defined as *special* upper- and lower-Hessenberg matrices respectively. In the same way, we define a *special* upper-triangular matrix $R(\beta, p, \gamma)$ as having the form

$$R(\beta, p, \gamma) = \begin{pmatrix}
\gamma_1 & \beta_1 p_2 & \beta_1 p_3 & \cdots\cdots & \beta_1 p_n \\
 & \gamma_2 & \beta_2 p_3 & \cdots\cdots & \beta_2 p_n \\
 & & \gamma_3 & \cdots\cdots & \beta_3 p_n \\
 & & & \ddots & \vdots \\
 & & & \gamma_{n-1} & \beta_{n-1} p_n \\
 & & & & \gamma_n
\end{pmatrix}.$$

The particular recurrence relation used to form $H_L(p, \beta, \gamma)$ will depend upon the order in which the Givens matrices are generated. For example, if $P_n{}^{n-1}$ is formed first, then the backward recurrence relation can be used.

We have only considered a particular sequence of Givens matrices. Similar formulae can be derived to compute the lower-Hessenberg matrix associated with the sequence

$$P_{n-1}^n P_{n-2}^{n-1} \cdots P_2^3 P_1^2.$$

We state the following lemma without proof.

LEMMA II. *Let* $D = \operatorname{diag}(d_1, d_2, \cdots, d_n)$, $\Gamma_1 = \operatorname{diag}(\gamma_1, \gamma_2, \cdots, \gamma_{n-1}, 1)$, $\Gamma_2 = \operatorname{diag}(1, \gamma_1, \gamma_2, \cdots, \gamma_{n-1})$ *and* $e = (1, 1, \cdots, 1, 1)^T$. *Then*

1. $DH_L(p, \beta, \gamma) = H_L(\bar{p}, \bar{\beta}, \bar{\gamma})D$ *where* $\bar{\beta}_i = \beta_i/d_i$, $\bar{p}_i = d_i p_i$, $i = 1, 2, \cdots, n$, $\bar{\gamma}_i = d_i\gamma_i/d_{i+1}$, $i = 1, 2, \cdots, n-1$.
2. $R(\beta, p, \gamma)D = DR(\bar{\beta}, \bar{p}, \gamma)$ *where* $\bar{\beta}_i = \beta_i/d_i$, $\bar{p}_i = d_i p_i$, $i = 1, 2, \cdots, n$.

3. *$R(\beta, p, \gamma) = DR(\bar{\beta}, p, e)$ where $\bar{\beta}_i = \beta_i/\gamma_i$, $i = 1, 2, \cdots, n-1$, $d_i = \gamma_i$, $i = 1, 2, \cdots, n$.*

4. *$H_L(p, \beta, \gamma) = \Gamma_1 H_L(\bar{p}, \beta, e) = H_L(p, \bar{\beta}, e)\Gamma_2$ where $\bar{p}_i = p_i/\gamma_i$ $(i < n)$, $\bar{p}_n = p_n$, and $\bar{\beta}_i = \beta_i/\gamma_i$ $(i > 1)$, $\bar{\beta}_1 = \beta_1$.*

5. *If $H_L(\bar{p}, \bar{\beta}, \gamma) = H_L(p, \beta, \gamma)$ then $\bar{\gamma}_i = \gamma_i$ and $\bar{p}_i = \alpha p_i$, $\bar{\beta}_i = \beta_i/\alpha$ for $i = 1, 2, \cdots, n$, where α is some constant.*

The next three lemmas show how the product of special matrices with various general matrices may be computed efficiently.

LEMMA III. *Let B be an $m \times n$ matrix and $H_L(p, \beta, \gamma)$ an $n \times n$ special lower-Hessenberg matrix. The product $\bar{B} = BH$ can be formed using either of the following recurrence relations:*

Forward Recurrence.

1. $w^{(1)} = Bp$, $\bar{b}_{i1} = \beta_1 w_i^{(1)}$, $i = 1, 2, \cdots, m$;

2.
$$\left.\begin{aligned} w_i^{(j)} &= w_i^{(j-1)} - p_{j-1} b_{i,j-1} \\ \bar{b}_{ij} &= \gamma_{j-1} b_{i,j-1} + \beta_j w_i^{(j)} \end{aligned}\right\}, \qquad \begin{aligned} i &= 1, 2, \cdots, m, \\ j &= 2, 3, \cdots, n. \end{aligned}$$

Backward Recurrence.

1. $w_i^{(n)} = p_n b_{in}$, $i = 1, 2, \cdots, m$;

2.
$$\left.\begin{aligned} \bar{b}_{ij} &= \gamma_{j-1} b_{i,j-1} + \beta_j w_i^{(j)} \\ w_i^{(j-1)} &= p_{j-1} b_{i,j-1} + w_i^{(j)} \end{aligned}\right\}, \qquad \begin{aligned} i &= 1, 2, \cdots, m, \\ j &= n, n-1, \cdots, 2; \end{aligned}$$

3.
$$\bar{b}_{i1} = \beta_1 w_i^{(1)}, \qquad i = 1, 2, \cdots, m.$$

Proof. We will give a proof for the forward recurrence case. The backward recurrence case can be shown in a similar way. The first column of $\bar{B}$ is given by

$$\bar{b}_{i1} = \beta_1 \sum_{j=1}^{n} b_{ij} p_j, \qquad i = 1, 2, \cdots, m.$$

If we define

$$w^{(1)} = Bp,$$

or

$$w_i^{(1)} = \sum_{j=1}^{n} b_{ij} p_j, \qquad i = 1, 2, \cdots, m, \tag{9}$$

then

$$\bar{b}_{i1} = \beta_1 w_i^{(1)}, \qquad i = 1, 2, \cdots, m.$$

Forming the second column, we have

$$\bar{b}_{i2} = \gamma_1 b_{i2} + \beta_1 \sum_{j=2}^{n} b_{ij} p_j, \qquad i = 1, 2, \cdots, m. \tag{10}$$

From Eq. (9), we have

$$w_i^{(1)} - b_{i1} p_1 = \sum_{j=2}^{n} b_{ij} p_j, \qquad i = 1, 2, \cdots, m,$$

and, if this vector is defined as $w^{(2)}$, then (10) becomes

$$\bar{b}_{i2} = \gamma_1 b_{i1} + \beta_2 w_i^{(2)}, \qquad i = 1, 2, \cdots, m.$$

The other columns of $\bar{B}$ are formed in exactly the same way.

The backward recurrence is more efficient, unless the product Bp is known *a priori*. It is also more convenient if $\bar{B}$ occupies the same storage as B.

The forward and backward recurrence relations require approximately 75% of the work necessary to form the same product by successively multiplying B by each of the individual Givens matrices. Since $H_L(p, \beta, \gamma)$ is an orthogonal matrix, there exists a vector v such that $H_L(p, \beta, \gamma)v = \alpha e_1$, and we can regard $H_L(p, \beta, \gamma)$ as the matrix which reduces v to αe_1. An equivalent reduction can be obtained by multiplying v by a single Householder matrix. If we have a product of the form

$$H_L(p^{(1)}, \beta^{(1)}, \gamma^{(1)}) \cdots H_L(p^{(r)}, \beta^{(r)}, \gamma^{(r)})B,$$

the computational effort involved in applying Lemma III is less than that using a similar product of the equivalent Householder matrices. This is because for D, a certain diagonal matrix, the product can be written as

$$DH_L(\bar{p}^{(1)}, \bar{\beta}^{(1)}, e) \cdots H_L(\bar{p}^{(r)}, \bar{\beta}^{(r)}, e)B$$

using Lemma II, parts 1 and 4.

LEMMA IV. *Let R be an upper-triangular matrix and $H_U(\beta, p, \gamma)$ a special upper-Hessenberg matrix. The product $\bar{H} = H_U(\beta, p, \gamma)R$ is an upper-Hessenberg matrix which can be determined using either of the following recurrence relations:*

Forward Recurrence.

1. *Set* $w^{(1)} = R^Tp$,

$$\bar{h}_{ij} = \beta_1 w_j^{(1)}, \qquad j = 1, 2, \cdots, n.$$

2. *For* $i = 2, 3, \cdots, n$, *set*

$$\bar{h}_{i,i-1} = \gamma_{i-1} r_{i-1,i-1},$$
$$\left.\begin{aligned} w_j^{(i)} &= w_j^{(i-1)} - p_{i-1} r_{i-1,j} \\ \bar{h}_{i,j} &= \gamma_{i-1} r_{i-1,j} + \beta_i w_j^{(i)} \end{aligned}\right\}, \qquad j = i, i+1, \cdots, n.$$

Backward Recurrence.

1. $w_n{}^{(n)} = p_n r_{nn}$.
2. *For* $i = n, n-1, \cdots, 3, 2$, *set*

$$\bar{h}_{i,i-1} = \gamma_{i-1} r_{i-1,i-1}, \qquad w_{i-1}^{(i-1)} = p_{i-1} r_{i-1,i-1},$$
$$\left.\begin{aligned} \bar{h}_{ij} &= \gamma_{i-1} r_{i-1,j} + \beta_i w_j^{(i)} \\ w_j^{(i-1)} &= p_{i-1} r_{i-1,j} + w_j^{(i)} \end{aligned}\right\}, \qquad j = i, i+1, \cdots, n.$$

3. $$\bar{h}_{1j} = \beta_1 w_j{}^{(1)}, \qquad j = 1, 2, \cdots, n.$$

Proof. This lemma is proved in a similar way to Lemma III.

LEMMA V. *Let R be upper-triangular and $R(\beta, p, \gamma)$ a special upper-triangular matrix. The product $\bar{R} = R(\beta, p, \gamma)R$ can be found using either of the following recur-*

rence relations:

Forward Recurrence.

1. *Set* $w^{(1)} = R^T p$.
2. *For* $i = 1, 2, \cdots, n$, *set*

$$\begin{aligned} \bar{r}_{ii} &= \gamma_i r_{ii}, \\ \left.\begin{aligned} w_j^{(i+1)} &= w_j^{(i)} - p_i r_{ij} \\ \bar{r}_{ij} &= \gamma_i r_{ij} + \beta_i w_j^{(i+1)} \end{aligned}\right\}, &\qquad j = i+1, i+2, \cdots, n. \end{aligned}$$

Backward Recurrence.

1. *For* $i = n, n-1, \cdots, 1$, *set*

$$\begin{aligned} w_i^{(i)} &= p_i r_{ii}, \qquad \bar{r}_{ii} = \gamma_i r_{ii}, \\ \left.\begin{aligned} \bar{r}_{ij} &= \gamma_i r_{ij} + \beta_i w_j^{(i+1)} \\ w_j^{(i)} &= w_j^{(i+1)} + p_i r_{ij} \end{aligned}\right\}, &\qquad j = i+1, i+2, \cdots, n. \end{aligned}$$

The forward recurrence relation can be formulated in the following alternative manner:

1. Set $w^{(1)} = R^T p$.
2. For $i = 1, 2, \cdots, n$, set

$$\begin{aligned} \bar{r}_{ii} &= \gamma_i r_{ii}, \\ \left.\begin{aligned} w_j^{(i+1)} &= w_j^{(i)} - p_i r_{ij} \\ \bar{r}_{ij} &= (\gamma_i - \beta_i p_i) r_{ij} + \beta_i w_j^{(i)} \end{aligned}\right\}, &\qquad j = i+1, \cdots, n. \end{aligned}$$

This formulation requires an additional $n^2/2$ multiplications. It has been shown by Gentleman [4] that the use of the more efficient relationship can lead to numerical instabilities in certain applications.

If the products of n 2×2 Givens matrices are accumulated into a single *special* matrix, it has been demonstrated in Lemmas I–V how certain savings can be made in subsequent computations. The nature of the forward and backward recurrence relations are such that, when a value of s_j is very small, underflow could occur in the subsequent computation of η_j. This will result in a division by zero during the computation of the next β_j. It will be shown in the following section how this difficulty can be avoided by judicious choice of the scalar π.

In certain applications, the vector v which is such that

$$H_U(\beta, p, \gamma) v = ||v||_2 \, e_1$$

is known. Since $H_U(\beta, p, \gamma)$ is orthogonal, we have, from its definition, that

$$H_U(\beta, p, \gamma)^T H_U(\beta, p, \gamma) v = ||v||_2 \, H_U(\beta, p, \gamma)^T e_1 = ||v||_2 \, H_L(p, \beta, \gamma) e_1$$

which gives $v = \beta_1 ||v||_2 \, p$, and the vector v is parallel to the vector p. If the value of π is chosen as $\pi = c_1/v_1$, then the vector p is *equal* to v. If ρ_j denotes the quantity defined at (4), this gives the modified algorithm:

Backward Recurrence.

1. Set $\beta_n = -c_{n-1}/v_n$, $\gamma_{n-1} = s_{n-1}$.
2. For $j = n-1, \cdots, 3, 2$, set $\beta_j = -c_{j-1}/\rho_j$, $\gamma_{j-1} = s_{j-1}$.
3. Set $\beta_1 = c_1/v_1 = 1/\rho_1$.

In the cases where v_i is not known *a priori*, π can be set at 2^{-t}, where the computation is carried out on a machine with a t-digit binary mantissa. Since the value of η_i is such that

$$\eta_i = s_i s_{i-1} \cdots s_1/\pi$$

during forward recurrence, and

$$\eta_i = s_i s_{i+1} \cdots s_{n-1}/\pi$$

during backward recurrence, this choice of π is such that η_i is unlikely to underflow.

If even this strategy is insufficient, the product of the Givens matrices can be split into subproducts. For example, if at the kth product, η_k is intolerably small, we can form the subproduct:

$$\left[\begin{array}{c|c} I & 0 \\ \hline 0 & H_L(p', \beta', \gamma') \end{array}\right] P_{k+1}^{k} \left[\begin{array}{c|c} H_L(p'', \beta'', \gamma'') & 0 \\ \hline 0 & I \end{array}\right]$$

where $H_L(p', \beta', \gamma')$ and $H_L(p'', \beta'', \gamma'')$ are smaller special matrices of dimension $(n - k) \times (n - k)$ and $k \times k$, respectively. Clearly, a product of separate Givens matrices can be viewed as being a product of special matrices in which a "split" has occurred at every step. Each element in the subproduct is an individual Givens matrix.

3. Modification of the Cholesky Factor. In this section, we consider the case where a symmetric positive definite matrix A is modified by a symmetric matrix of rank one, i.e., we have

$$\bar{A} = A + \alpha z z^T.$$

Assuming that the Cholesky factors of A are known, viz. $A = LDL^T$, we wish to determine the factors

$$\bar{A} = \bar{L}\bar{D}\bar{L}^T.$$

It is necessary to make the assumption that A and $\bar{A}$ are positive definite since otherwise the algorithms for determining the modified factors are numerically unstable, even if the factorization of $\bar{A}$ exists. Several alternative algorithms will be presented and comments made upon their relative merits. Any of these general methods can be applied when A is of the form $A = B^T B$ and rows or columns of the matrix B are being added or deleted. In this case, it may be better to use specialized methods which modify the orthogonal factorization of B:

$$QB = \left[\begin{array}{c} R \\ \hline 0 \end{array}\right].$$

The reader is referred to Section 5 for further details. The methods in this section are all based upon the fundamental equality

$$\bar{A} = A + \alpha z z^T = L(D + \alpha p p^T)L^T,$$

where $Lp = z$, and p is obtained from z by a forward substitution. If we form the factorization

$$D + \alpha pp^T = \tilde{L}\tilde{D}\tilde{L}^T, \tag{11}$$

the required modified Cholesky factors are of the form

$$\bar{A} = L\tilde{L}\tilde{D}\tilde{L}^TL^T$$

giving

$$\bar{L} = L\tilde{L} \quad \text{and} \quad \bar{D} = \tilde{D},$$

since the product of two lower-triangular matrices is a lower-triangular matrix. The manner in which the factorization (11) is performed will characterize a particular method.

Method C1. *Using Classical Cholesky Factorization.* The Cholesky factorization of $D + \alpha pp^T$ can be formed directly. We will use this method to prove inductively that $\tilde{L}$ is special.

Assume at the jth stage of the computation that

$$\tilde{l}_{rs} = p_r\beta_s, \qquad r = j, j + 1, \cdots, n; s = 1, 2, \cdots, j - 1, \tag{12}$$

and that all these elements have been determined. Explicitly forming the jth column of $\tilde{L}\tilde{D}\tilde{L}^T$ gives the following equations for $\tilde{d}_j$ and $\tilde{l}_{rj}$, $r = j + 1, \cdots, n$:

$$\sum_{i=1}^{j-1} \tilde{d}_i\tilde{l}_{ji}^2 + \tilde{d}_j = d_j + \alpha p_j^2 \tag{13}$$

and

$$\sum_{i=1}^{j-1} \tilde{d}_i\tilde{l}_{ji}\tilde{l}_{ri} + \tilde{d}_j\tilde{l}_{rj} = \alpha p_j p_r, \qquad r = j + 1, \cdots, n. \tag{14}$$

Using Eq. (12) with (13) and (14) gives

$$p_j^2 \sum_{i=1}^{j-1} \beta_i^2\tilde{d}_i + \tilde{d}_j = d_j + \alpha p_j^2$$

and

$$p_jp_r \sum_{i=1}^{j-1} \tilde{d}_i\beta_i^2 + \tilde{d}_j\tilde{l}_{rj} = \alpha p_j p_r, \qquad r = j + 1, \cdots, n.$$

From the last equation, we have

$$\tilde{l}_{rj} = \frac{p_j}{\tilde{d}_j}\left[\alpha - \sum_{i=1}^{j-1} \tilde{d}_i\beta_i^2\right]p_r, \qquad r = j + 1, \cdots, n,$$

and defining

$$\beta_j = \frac{p_j}{\tilde{d}_j}\left[\alpha - \sum_{i=1}^{j-1} \tilde{d}_i\beta_i^2\right]$$

gives $\tilde{l}_{rj} = p_r\beta_j$. Hence, the subdiagonal elements of the jth column of $\tilde{L}$ are multiples of the corresponding elements of the vector p.

Now forming the first column of $\tilde{L}\tilde{D}\tilde{L}^T$, we obtain the equations

$$\tilde{d}_1 = d_1 + \alpha p_1^2,$$
$$\tilde{d}_1 \tilde{l}_{r1} = \alpha p_1 p_r, \qquad r = 2, \cdots, n,$$

which shows that the subdiagonal elements of the first column of $\tilde{L}$ are multiples of the corresponding elements of p. Consequently, we have proved by induction that $\tilde{L}$ is special.

This result implies that we need only compute the values of $\tilde{d}_j$, β_j, $j = 1, \cdots, n$, in order to obtain the factorization of $D + \alpha pp^T$. In practice, we define the auxiliary quantity

$$\alpha_j = \alpha - \sum_{i=1}^{j-1} \tilde{d}_i \beta_i^2.$$

The recurrence relations for α_j, $\tilde{d}_j$ and β_j then become

$$\begin{aligned} \alpha_1 &= \alpha \\ \left.\begin{aligned} \tilde{d}_j &= d_j + \alpha_j p_j^2 \\ \beta_j &= \alpha_j p_j / \tilde{d}_j \\ \alpha_{j+1} &= \alpha_j d_j / \tilde{d}_j \end{aligned}\right\} &, \qquad j = 1, 2, \cdots, n. \end{aligned}$$

The product $\bar{L} = L\tilde{L}$ can be computed in terms of the β_j by forward recurrence using Lemma V. Note that L and $\tilde{L}$ are both *unit* lower-triangular matrices and that this results in some simplification of the algorithm. The vector $w^{(1)}$ needed to initialize the recurrence relations is known since $w^{(1)} = Lp = z$. Also, each of the vectors $w^{(j)}$ $(j = 1, 2, \cdots, n)$ can be obtained during the jth stage of the initial back substitution $Lp = z$, since

$$w_r^{(j)} = \sum_{i=j}^{n} l_{ri} p_i = z_r - \sum_{i=1}^{j-1} l_{ri} p_i, \qquad r = j, j+1, \cdots, n.$$

The final recurrence relations for modifying L and D are as follows:

Algorithm C1.

1. Define $\alpha_1 = \alpha$, $w^{(1)} = z$.
2. For $j = 1, 2, \cdots, n$, compute

$$\begin{aligned} p_j &= w_j^{(j)}, \\ \bar{d}_j &= d_j + \alpha_j p_j^2, \\ \beta_j &= p_j \alpha_j / \bar{d}_j, \\ \alpha_{j+1} &= d_j \alpha_j / \bar{d}_j, \\ \left.\begin{aligned} w_r^{(j+1)} &= w_r^{(j)} - p_j l_{rj} \\ \bar{l}_{rj} &= l_{rj} + \beta_j w_r^{(j+1)} \end{aligned}\right\} &, \qquad r = j+1, \cdots, n. \end{aligned}$$

Using the expression for $w_r^{(j+1)}$, we can rearrange the equation for $\bar{l}_{rj}$ in the form

$$\begin{aligned} \bar{l}_{rj} &= l_{rj} + \beta_j (w_r^{(j)} - p_j l_{rj}) \\ &= (1 - \beta_j p_j) l_{rj} + \beta_j w_r^{(j)} \\ &= (d_j / \bar{d}_j) l_{rj} + \beta_j w_r^{(j)}, \end{aligned}$$

which is the form of the algorithm given by Gill and Murray [5]. However, this increases the number of multiplications by 50%.

One of the earliest papers devoted to modifying matrix factorizations is that by Bennett [2], in which LDU factors are updated following a rank m modification:

$$\bar{L}\bar{D}\bar{U} = LDU + XCY^T,$$

where X, Y are $n \times m$ and C is $m \times m$. It should be noted that

(i) the algorithm given by Bennett is numerically stable only when $L = U^T$, $X = Y$ and both D and $\bar{D}$ are positive definite, and

(ii) Algorithm C1 is almost identical to the special case of Bennett's algorithm when $m = 1$, $C = \alpha$ and $X = Y = z$.

The number of operations necessary to compute the modified factorization using Algorithm C1 is $n^2 + O(n)$ multiplications and $n^2 + O(n)$ additions.

If the matrix A is sufficiently positive definite, that is, its smallest eigenvalue is sufficiently large relative to some norm of $\bar{A}$, then Algorithm C1 is numerically stable. However, if $\alpha < 0$ and $\bar{A}$ is near to singularity, it is possible that rounding error could cause the diagonal elements $\bar{d}_j$ to become zero or arbitrarily small. In such cases, it is also possible that the $\bar{d}_j$ could change sign, even when the modification may be known from theoretical analysis to give a positive definite factorization. It may then be advantageous to use one of the following methods, because with these the resulting matrix will be positive definite *regardless of any numerical errors made.*

Method C2. *Using Householder Matrices.* In this method, the factorization (11) is performed using Householder matrices. To do this, we must write

$$\bar{A} = LD^{1/2}(I + \alpha vv^T)D^{1/2}L^T,$$

where v is the solution of the equations $LD^{1/2}v = z$. The matrix $I + \alpha vv^T$ can be factorized into the form

$$(15) \qquad I + \alpha vv^T = (I + \sigma vv^T)(I + \sigma vv^T)$$

by choosing $\sigma = \alpha/(1 + (1 + \alpha v^Tv)^{1/2})$.

The expression under the root sign is a positive multiple of the determinant of $\bar{A}$. If $\bar{A}$ is positive definite σ will be real.

We now perform the Householder reduction of $I + \sigma vv^T$ to lower-triangular form

$$\hat{L} = (I + \sigma vv^T)P_1P_2 \cdots P_{n-1}.$$

We will only consider application of the first Householder matrix P_1. The effect of the remainder can easily be deduced.

Let $P_1 = I + uu^T/\tau$ and partition v in the form $v^T = [v_1 \vdots w^T]$. The (1, 1) element of $I + \sigma vv^T$ is then $\theta = 1 + \sigma v_1{}^2$ and P_1 must reduce the vector $[\theta \vdots \sigma v_1 w^T]$ to a multiple of $e_1{}^T$. Using the relations of Section 2, we define

$$\rho^2 = \theta^2 + \sigma^2 v_1^2 w^T w,$$

$$u_1 = \theta + \rho \quad \text{and}$$

$$\tau = -\rho u_1.$$

(Note that we have taken $\rho = +(\rho^2)^{1/2}$, because we know that $\theta > 0$.) Now u has the form

$$u^T = [u_1 \vdots \sigma v_1 w^T],$$

i.e., the vector of elements $u_2, \cdots, u_n$ is a multiple of the vector w.

The result of applying the first Householder transformation can therefore be written as

$$(I + \sigma v v^T)\left(I + \frac{1}{\tau} u u^T\right) = \left[\begin{array}{c|c} -\rho & 0 \\ \hline \delta w & I + \bar\sigma w w^T \end{array}\right]$$

for suitable values of the scalars δ and $\bar\sigma$ which will be determined as follows. The first column is given by

$$\left[\begin{array}{c} -\gamma \\ \hline \delta w \end{array}\right] = (I + \sigma v v^T)\left[e_1 + \frac{1}{\tau} u_1 u\right]$$

$$= \left[\begin{array}{c|c} 1 + \sigma v_1^2 & \sigma v_1 w^T \\ \hline \sigma v_1 w & I + \sigma w w^T \end{array}\right]\left[\begin{array}{c} 1 + \frac{1}{\tau} u_1^2 \\ \hline \frac{1}{\tau} u_1 \sigma v_1 w \end{array}\right]$$

which implies that

$$\delta w = \left(1 + \frac{1}{\tau} u_1^2\right)\sigma v_1 w + \frac{1}{\tau} u_1 \sigma v_1 (1 + \sigma w^T w) w$$

so

$$\delta = \left(1 + \frac{1}{\tau} u_1^2\right)\sigma v_1 + \frac{1}{\tau} u_1 \sigma v_1 (1 + \sigma w^T w).$$

A small amount of algebraic manipulation gives

$$\delta = -\sigma \frac{v_1}{\rho}(2 + \sigma v^T v).$$

Similarly, for the scalar $\bar\sigma$, we have

$$I + \bar\sigma w w^T = [\sigma v_1 w \mid I + \sigma w w^T]\left[\begin{array}{c} \frac{1}{\tau} u_1 \sigma v_1 w^T \\ \hline I + \frac{1}{\tau} \sigma^2 v_1^2 w w^T \end{array}\right]$$

giving

$$\bar\sigma = \frac{1}{\tau} u_1 \sigma^2 v_1^2 + \sigma + \frac{1}{\tau} \sigma^2 v_1^2 + \frac{1}{\tau} \sigma^3 v_1^2 w^T w$$

which can be shown to be equal to

$$\bar\sigma = -\frac{1}{\tau}\sigma(1 + \rho) = \frac{\sigma(1 + \rho)}{\rho(\theta + \rho)}.$$

The $(n - 1) \times (n - 1)$ submatrix $I + \bar{\sigma} w w^T$ has the same structure as $I + \sigma v v^T$ and a Householder matrix can be applied in exactly the same fashion. It can be shown that

$$1 + \bar{\sigma} w^T w = \frac{1}{\rho}(1 + \sigma v^T v)$$

and so the sign choice in the definition of each of the Householder matrices remains the same.

For notational convenience, we will write ρ_j, θ_j, δ_j, and σ_{j+1} for the quantities ρ, θ, δ, and $\bar{\sigma}$ at the *j*th step of the reduction, and use ρ, δ for the vectors (ρ_j), (δ_j).

The full reduction is now

$$(I + \sigma v v^T) P_1 P_2 \cdots P_{n-1} = R(\delta, v, -\rho)^T$$

which gives

$$\bar{A} = L D^{1/2} R(\delta, v, -\rho)^T R(\delta, v, -\rho) D^{1/2} L^T.$$

From Lemma II, we have

$$R(\delta, v, -\rho) D^{1/2} = R(\delta, D^{1/2} v, \rho) = D^{1/2} R(D^{-1/2}\delta, p, \rho) = D^{1/2} \Gamma R(\beta, p, e),$$

where

$$\Gamma = \operatorname{diag}(\rho_j),$$
$$\left.\begin{aligned} p_j &= d_j^{1/2} v_j \\ \beta_j &= -\delta_j/(d_j^{1/2}\rho_j) \end{aligned}\right\}, \qquad j = 1, \cdots, n.$$

(Note that p is the solution of $Lp = z$, as before.)

Following our convention for unit-triangular matrices, we define

$$L(p, \beta, e) = R(\beta, p, e)^T.$$

The net result is that

$$\bar{L} = L L(p, \beta, e) \quad \text{and} \quad \bar{D} = \Gamma D \Gamma,$$

which must be analytically equivalent to the factors obtained by Algorithm C1. What we have done is find alternative expressions for β_j and $\bar{d}_j$, the most important being $\bar{d}_j = \rho_j{}^2 d_j$. Since $\rho_j{}^2$ is computed as a sum of squares, this expression guarantees that the computed $\bar{d}_j$ can never become negative. In Algorithm C1, the corresponding relation is $\bar{d}_j = d_j + \alpha_j p_j{}^2$ where $\operatorname{sign}(\alpha_j) = \operatorname{sign}(\alpha)$. If $\alpha < 0$ and $\bar{L}\bar{D}\bar{L}^T$ is nearly singular, it is possible that rounding errors could give $\bar{d}_j \ll 0$. In such cases, Algorithm C2 is to be preferred.

The analytical equivalence of the two algorithms can be seen through the relation between α_j and σ_j. For example, Eq. (15) implies that

$$\alpha_1 = \sigma_1(2 + \sigma_1 v^T v)$$

and if this is substituted into $\bar{d}_1 = d_1 + \alpha_1 p_1{}^2$ we get

$$\bar{d}_1 = \rho_1^2 d_1,$$

which agrees with $\bar{D} = \Gamma D \Gamma$. In general, if we define

$$\alpha_j = \sigma_j\left(2 + \sigma_j \sum_{i=j}^{n} v_i^2\right),$$

the expression for δ_j simplifies, giving

$$\beta_j = -\frac{\delta_j}{d_j^{1/2}\rho_j} = \frac{\alpha_j v_j}{d_j^{1/2}\rho_j} = \frac{\alpha_j p_j}{d_j \rho_j{}^2} = \frac{\alpha_j p_j}{\bar{d}_j}$$

which is the expression obtained for β_j in Algorithm C1. In practice, we retain this form for Algorithm C2. The method for computing $\bar{L}$ from L and $L(p, \beta, e)$ is also the same as before. The iteration can be summarized as follows.

Algorithm C2.

1. Solve $Lp = z$.
2. Define

$$\left.\begin{aligned} w_j^{(1)} &= z_j \\ s_j &= \sum_{i=j}^{n} p_i^2/d_i = \sum_{i=j}^{n} q_i \end{aligned}\right\}, \qquad j = 1, 2, \cdots, n,$$

$$\alpha_1 = \alpha,$$

$$\sigma_1 = \alpha/[1 + (1 + \alpha s_1)^{1/2}].$$

3. For $j = 1, 2, \cdots, n$, compute

(a) $q_j = p_j^2/d_j$,

(b) $\theta_j = 1 + \sigma_j q_j$,

(c) $s_{j+1} = s_j - q_j$,

(d) $\rho_j^2 = \theta_j^2 + \sigma_j^2 q_j s_{j+1}$,

(e) $\bar{d}_j = \rho_j^2 d_j$,

(f) $\beta_j = \alpha_j p_j/\bar{d}_j$,

(g) $\alpha_{j+1} = \alpha_j/\rho_j^2$,

(h) $\sigma_{j+1} = \sigma_j(1 + \rho_j)/[\rho_j(\theta_j + \rho_j)]$,

(i) $$\left.\begin{aligned} w_r^{(j+1)} &= w_r^{(j)} - p_j l_{rj} \\ \bar{l}_{rj} &= l_{rj} + \beta_j w_r^{(j+1)} \end{aligned}\right\}, \qquad r = j+1, j+2, \cdots, n.$$

Note that the initial back substitution takes place separately from the computation of $L(p, \beta, e)$, because of the need to compute the vector p before computing s_1. This adds $n^2/2 + O(n)$ multiplications to the method but ensures that the algorithm will always yield a positive definite factorization even under extreme circumstances and allows $\bar{L}$ to be computed by either the forward or backward recurrence relations given in Lemma V. The method requires $3n^2/2 + O(n)$ multiplications and $n + 1$ square roots.

Method C3. *Using Givens Matrices* I. One of the most obvious methods of modifying the Cholesky factors of A in the particular case when $\alpha > 0$ is as follows.

Consider the reduction of the matrix $[\alpha^{1/2}z \;\vdots\; R^T]$ to lower-triangular form, i.e.,

$$[\alpha^{1/2}z \;\vdots\; R^T]P = [\bar{R}^T \;\vdots\; 0],$$

where P is a sequence of Givens matrices of the form $P = P_2{}^1 P_3{}^2 \cdots P_{n+1}{}^n$. We have

$$[\bar{R}^T \;\vdots\; 0]\left[\begin{array}{c}\bar{R}\\ \hdashline 0\end{array}\right] = \bar{R}^T\bar{R} = [\alpha^{1/2}z \;\vdots\; R^T]PP^T\left[\begin{array}{c}\alpha^{1/2}z^T\\ \hdashline R\end{array}\right] = R^TR + \alpha zz^T.$$

Consequently, $\bar{R}^T$ is the required factor.

This algorithm can be generalized when $\alpha < 0$. The rank-one modification will be written as

$$\bar{R}^T\bar{R} = R^TR - \alpha zz^T, \qquad \alpha > 0,$$

for convenience. The vector p is computed such that

$$R^Tp = z,$$

and we set

$$\delta_n^2 = (1 - \alpha p^Tp)/\alpha.$$

We now form the matrix

$$\left[\begin{array}{c:c}p & R\\ \hdashline \delta_n & 0\end{array}\right]$$

and premultiply by an orthogonal matrix P of the form $P = P_1{}^{n+1} \cdots P_{n-1}{}^{n+1}P_n{}^{n+1}$ such that the vector p is reduced to zero. This gives

$$P\left[\begin{array}{c:c}p & R\\ \hdashline \delta_n & 0\end{array}\right] = \left[\begin{array}{c:c}0 & \bar{R}\\ \hdashline \delta_0 & r^T\end{array}\right]$$

in which case the following relations must hold

$$p^Tp + \delta_n^2 = \delta_0^2, \tag{16}$$

$$R^Tp = \delta_0 r, \tag{17}$$

$$R^TR = \bar{R}^T\bar{R} + rr^T. \tag{18}$$

Equation (16) implies that $\delta_0{}^2 = 1/\alpha$, Eq. (17) implies that $r = z/\delta_0 = \alpha^{1/2}z$, and, finally, (18) gives $R^TR = \bar{R}^T\bar{R} + \alpha zz^T$, as required. This method requires $5n^2/2 + O(n)$ multiplications and $n + 1$ square roots.

Method C4. *Using Givens Matrices* II. For this method, we shall be modifying the factorization

$$\bar{R}^T\bar{R} = R^TR + \alpha zz^T.$$

From this equation we have

$$\bar{A} = R^T(I + \alpha pp^T)R, \tag{19}$$

where $R^Tp = z$. We can write $\bar{A}$ in the form

$$\bar{A} = R^TP^TP(I + \alpha pp^T)P^TPR, \tag{20}$$

where P is an orthogonal matrix. The matrix P is chosen as a product of Givens matrices such that

$$Pp = P_2^1 P_3^2 \cdots P_{n-1}^{n-2} P_n^{n-1} p = \rho e_1, \tag{21}$$

where $|\rho| = \|p\|_2$. Eq. (19) can be written as

$$\bar{A} = R^T P^T (I + \alpha \rho^2 e_1 e_1^T) P R.$$

As each Givens matrix $P_{j+1}{}^i$ is formed, it is multiplied into the upper-triangular matrix R. This has the effect of filling in the subdiagonal elements of R to give an upper-Hessenberg matrix H. We have

$$H = PR, \qquad \bar{A} = H^T J^T J H,$$

where J is an identity matrix except for the (1, 1) element which has the value $(1 + \alpha p^T p)^{1/2}$. If $\bar{A}$ is positive definite, the square root will be real. The formation of the product JH modifies the first row of H to give $\bar{H}$ which is still upper Hessenberg.

A second sequence of Givens matrices is now chosen to reduce $\bar{H}$ to upper-triangular form, i.e.,

$$\bar{P}\bar{H} = P_n^{n-1} P_{n-1}^{n-2} \cdots P_3^2 P_2^1 \bar{H} = \bar{R}.$$

Then

$$\bar{A} = \bar{H}^T \bar{H} = \bar{H}^T \bar{P}^T \bar{P} \bar{H} = \bar{R}^T \bar{R}$$

as required. This algorithm requires $9n^2/2 + O(n)$ multiplications and $2n - 1$ square roots.

Method C5. *Using Givens Matrices* III. If we write Eq. (19) as in Method C2, viz.

$$\bar{A} = R^T (I + \sigma p p^T)(I + \sigma p p^T) R,$$

where $\sigma = \alpha / (1 + (1 + \alpha p^T p)^{1/2})$. If P is the matrix defined in (21), we can write

$$\bar{A} = R^T (I + \sigma p p^T) P^T P (I + \sigma p p^T) R = R^T H^T H R, \tag{22}$$

where $H = P(I + \sigma p p^T) = P + \sigma \rho e_1 p^T$. According to Lemma I, P is a special upper-Hessenberg matrix of the form $P = H_U(\beta, \bar{p}, \gamma)$ for some vectors $\bar{p}$, β and γ. Now the first row of P is a multiple of $\bar{p}^T$ by definition, and, furthermore, $Pp = \rho e_1$ implies that $p = \rho P^T e_1$, so the first row of P is also a multiple of p. From Lemma II, it follows that by choosing $\bar{p}_n = p_n$ when forming P as a special matrix, we can ensure that $P = H_U(\beta, p, \gamma)$ for some β and γ.

Assuming this choice of $\bar{p}_n$ is made, we have

$$H = H_U(\beta, p, \gamma) + \sigma \rho e_1 p^T = H_U(\bar{\beta}, p, \gamma)$$

where $\bar{\beta}$ differs from β only in the first element, i.e. $\bar{\beta} = \beta + \sigma \rho e_1$. Now H can be reduced to upper-triangular form $\tilde{R}$ by a second sequence of Givens matrices $\bar{P}$:

$$\bar{P} H = P_n^{n-1} P_{n-1}^{n-2} \cdots P_3^2 P_2^1 H = \tilde{R}.$$

It can be readily shown that $\tilde{R}$ is of the form

$$\tilde{R} = R(\tilde{\beta}, p, \tilde{\gamma})$$

where the vectors $\tilde\beta$ and $\tilde\gamma$ are given by the following recurrence relations:

1. $\eta_1 = \bar\beta_1$;
2. $$\left.\begin{aligned} \tilde\beta_i &= c_i\eta_i + s_i\bar\beta_i \\ \tilde\gamma_i &= c_i\eta_i p_i + s_i\gamma_i \\ \eta_{i+1} &= s_i\eta_i - c_i\bar\beta_i \end{aligned}\right\}, \qquad i = 1, 2, \cdots, n-1;$$
3. $\tilde\gamma_n = \eta_n$.

The quantities c_i and s_i are the elements of the Givens matrices in $\bar P$. They reduce the subdiagonal elements γ_i of H to zero at each stage, and are defined in the usual way. The final product $\bar R = \tilde R R$ can be computed using Lemma V.

This algorithm requires $2n^2 + O(n)$ multiplications and $2n - 1$ square roots. The work has been reduced, relative to Method C4, by accumulating both sequences of Givens matrices into the special matrix $\tilde R$ and modifying R just once, rather than twice.

4. Modification of the Complete Orthogonal Factorization. If A is an $m \times n$ matrix of rank t, $m \geqq n$, $t \leqq n$, the complete orthogonal factorization of A is

$$QAZ = \left[\begin{array}{c|c} R & 0 \\ \hline 0 & 0 \end{array}\right] \tag{23}$$

where Q is an $m \times m$ orthogonal matrix, Z an $n \times n$ orthogonal matrix and R a $t \times t$ upper-triangular matrix (see Faddeev et al. [3], Hanson and Lawson [10]).

The pseudo-inverse of A is given by

$$A^+ = Z\left[\begin{array}{c|c} R^{-1} & 0 \\ \hline 0 & 0 \end{array}\right]Q.$$

In order to obtain the pseudo-inverse of $\bar A = A + yz^T$, where y and z are m and n vectors respectively, we consider modifying the complete orthogonal factorization of A. (With no loss of generality we have omitted the scalar α.)

From Eq. (23), we have

$$Q\bar AZ = \left[\begin{array}{c|c} R & 0 \\ \hline 0 & 0 \end{array}\right] + pq^T$$

where $p = Qy$ and $q = Z^Tz$. If the vectors p and q are partitioned as follows:

$$p = \left[\begin{array}{c} u \\ \hline \bar u \end{array}\right]\begin{matrix} \}t \\ \}m-t \end{matrix}, \qquad q = \left[\begin{array}{c} w \\ \hline \bar w \end{array}\right]\begin{matrix} \}t \\ \}n-t \end{matrix},$$

we can choose Q_{I} and Z_{I} to be either single Householder matrices or products of Givens matrices such that

$$Q_{\mathrm{I}}\bar u = \alpha e_1 \quad \text{and} \quad \bar w^T Z_{\mathrm{I}} = \beta e_1^T,$$

where α and β are scalars such that $|\alpha| = ||\bar u||_2$ and $|\beta| = ||\bar w||_2$. Note that application of these matrices leaves the matrix R unchanged. For convenience, we will now work with the $(t + 1) \times (t + 1)$ matrix S_{I} which is defined as

$$S_{\mathrm{I}} = \left[\begin{array}{c|c} R & 0 \\ \hline 0 & 0 \end{array}\right] + \left[\begin{array}{c} u \\ \hline \alpha \end{array}\right][w^T \ \vdots \ \beta].$$

We next perform two major steps which will be called *sweeps.*

First Sweep. Choose an orthogonal matrix Q_{II} such that

$$Q_{\mathrm{II}}\left[\begin{array}{c} u \\ \hline \alpha \end{array}\right] = P_2^1 P_3^2 \cdots P_t^{t-1} P_{t+1}^t \left[\begin{array}{c} u \\ \hline \alpha \end{array}\right] = \gamma_1 e_1$$

where $\gamma_1{}^2 = \|u\|_2{}^2 + \alpha^2$. If S_{I} is multiplied on the left by Q_{II} and the resulting product defined as S_{II}, we have

$$S_{\mathrm{II}} = Q_{\mathrm{II}} S_{\mathrm{I}} = \left[\begin{array}{c|c} r_{\mathrm{II}}^T & 0 \\ \hline R_{\mathrm{II}} & 0 \end{array}\right] + \gamma_1 e_1 [w^T \ \vdots \ \beta] = \left[\begin{array}{c|c} \bar{r}_{\mathrm{II}}^T & \gamma_1 \beta \\ \hline R_{\mathrm{II}} & 0 \end{array}\right],$$

where R_{II} is an upper-triangular matrix. The t diagonal elements of R_{II} are filled in one at a time by the application of each 2×2 orthogonal matrix. We have defined

$$\bar{r}_{\mathrm{II}}^T = r_{\mathrm{II}}^T + \gamma_1 w^T.$$

Second Sweep. We now construct an orthogonal matrix Q_{III} which, when applied to S_{II} from the left, reduces S_{II} to upper-triangular form. If this triangular matrix is defined as S_{III}, we have

$$S_{\mathrm{III}} = Q_{\mathrm{III}} \left[\begin{array}{c|c} \bar{r}_{\mathrm{II}}^T & \gamma_1 \beta \\ \hline R_{\mathrm{II}} & 0 \end{array}\right] = \left[\begin{array}{c|c} R_{\mathrm{III}} & s_{\mathrm{III}} \\ \hline 0 & \delta_{\mathrm{III}} \end{array}\right],$$

where Q_{III} is of the form

$$Q_{\mathrm{III}} = P_{t+1}^t \cdots P_3^2 P_2^1.$$

The matrix S_{III} may or may not be the upper-triangular matrix required, depending upon $\rho(\bar{A})$, the rank of $\bar{A}$. The different cases that can arise are summarized in the following table:

β α	$= 0$	$\neq 0$
$= 0$	$\rho(\bar{A}) = t$ or $t - 1$	$\rho(\bar{A}) = t$
$\neq 0$	$\rho(\bar{A}) = t$	$\rho(\bar{A}) = t + 1$

Case I. $\alpha \neq 0$, $\beta \neq 0$. In this case, S_{III} has full rank and

$$\left[\begin{array}{c|c} R_{\mathrm{III}} & s_{\mathrm{III}} \\ \hline 0 & \delta_{\mathrm{III}} \end{array}\right] = \bar{R}.$$

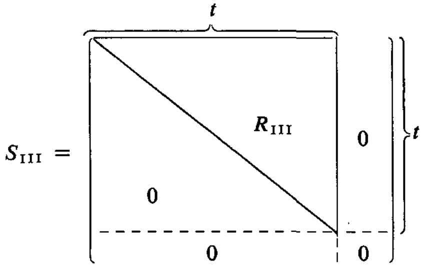
The final orthogonal matrix $\bar{Q}$ is given by

$$\bar{Q} = \left[\begin{array}{c|c} Q_{\mathrm{III}} & 0 \\ \hline \underbrace{0}_{t+1} & I \end{array}\right] \left[\begin{array}{c|c} Q_{\mathrm{II}} & 0 \\ \hline \underbrace{0}_{t+1} & I \end{array}\right] \left[\begin{array}{c|c} I & 0 \\ \hline \underbrace{0}_{t} & Q_{\mathrm{I}} \end{array}\right] Q \tag{24}$$

and

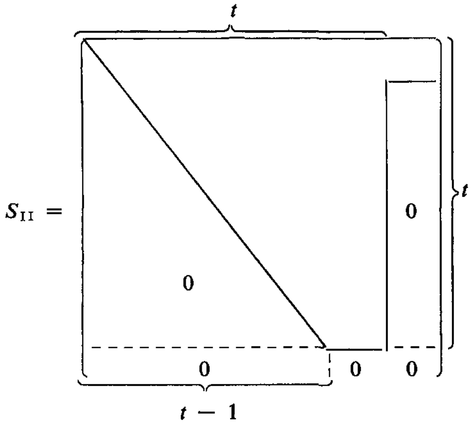
$$\bar{Z} = Z \left[\begin{array}{c|c} I & 0 \\ \hline \underbrace{0}_{t} & Z_{\mathrm{I}} \end{array}\right].$$

Case II. $\alpha \neq 0$, $\beta = 0$. If the first and second sweeps are followed carefully, it can be seen that S_{III} is of the form

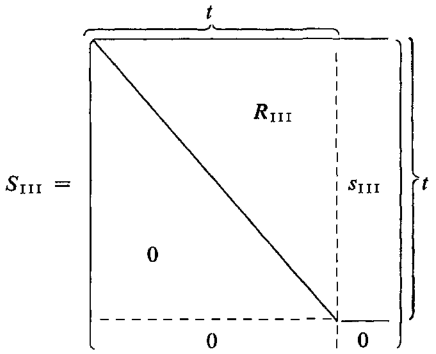


i.e., $s_{\mathrm{III}} = 0$ and $\delta_{\mathrm{III}} = 0$. As in Case I, S_{III} is in the required form and we define the modified factors accordingly.

Case III. $\alpha = 0$, $\beta \neq 0$. The first orthogonal transformation of the first sweep is an identity, and the matrix S_{II} has the form



Application of the second sweep (Q_{III}) gives the matrix S_{III} in the form



i.e., $\delta_{\mathrm{III}} = 0$.

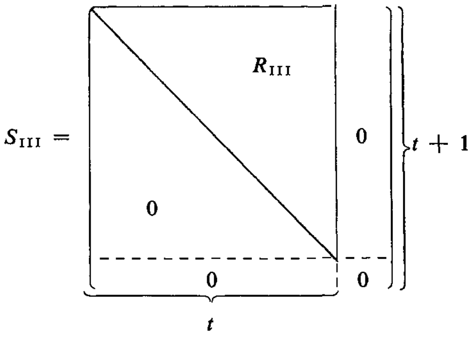
An orthogonal matrix Z_{II} is now applied on the right to reduce s_{III} to zero, thus

$$S_{\mathrm{III}}Z_{\mathrm{II}} = S_{\mathrm{III}}P_{t+1}^{t}P_{t+1}^{t-1}P_{t+1}^{t-2} \cdots P_{t+1}^{1} = \left[\begin{array}{c|c} \bar{R} & 0 \\ \hline 0 & 0 \end{array}\right].$$

The modified factors are $\bar{Q}$ as defined in (22), and

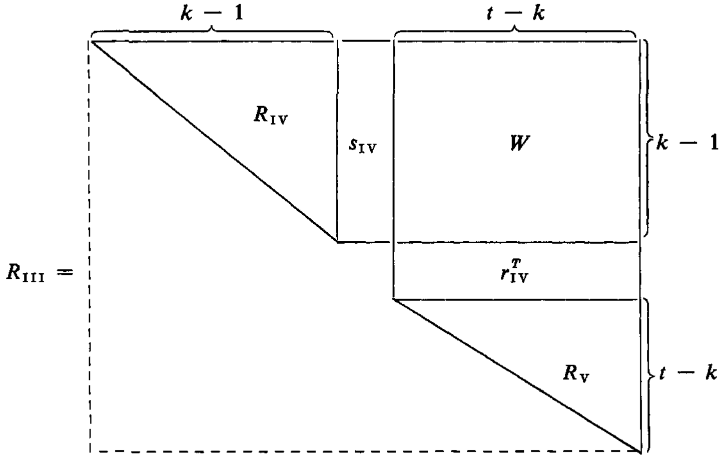
$$\bar{Z} = Z\left[\begin{array}{c|c} I & \\ \hline & Z_{\mathrm{I}} \end{array}\right]\left[\begin{array}{c|c} Z_{\mathrm{II}} & \\ \hline & I \end{array}\right].$$

Case IV. $\alpha = 0$, $\beta = 0$, $\rho(\bar{A}) = t$. The matrix S_{III} has the following form:



If the diagonal elements of R_{III} are all nonzero, then $\mathrm{rank}(\bar{A}) = \mathrm{rank}(R_{\mathrm{III}}) = t$ and the factors are completely determined. Otherwise, exactly one of the diagonal elements of R_{III} may be zero, since the rank of $\bar{A}$ can drop to $t - 1$. In this case, two more partial sweeps must be made to reduce R_{III} to strictly upper-triangular form, as follows.

Case V. $\alpha = 0$, $\beta = 0$, $\rho(\bar{A}) = t - 1$. Suppose that the kth diagonal of R_{III} is zero. The matrix can be partitioned in the form



where R_{IV}, R_V are upper-triangular with dimensions $(k-1) \times (k-1)$ and $(t-k) \times (t-k)$, respectively. An orthogonal transformation Q_{IV} is now applied on the left to reduce the submatrix

$$\begin{bmatrix} r_{IV}^T \\ \hdashline R_V \end{bmatrix}$$

to upper-triangular form in exactly the same way as the first sweep. Similarly, a transformation Z_{II} is applied (independently) from the right to reduce s_{IV} to zero in the submatrix $[R_{IV}\ s_{IV}]$. Thus

$$Q_{IV}R_{III}Z_{II} = \left(\begin{array}{c:c:c} \bar{R}_{IV} & 0 & W \\ \hdashline \multicolumn{2}{c:}{0} & \bar{R}_V \\ \cdashline{3-3} & & 0 \end{array}\right)$$

where $Q_{IV} = P_t{}^k P_{t-1}{}^k \cdots P_{k+2}{}^k P_{k+1}{}^k$ and $Z_{II} = P_k{}^{k-1} P_k{}^{k-2} \cdots P_k{}^2 P_k{}^1$.

Finally, a permutation matrix Z_{III} is applied to move the column of zeros to the right:

$$\left(\begin{array}{c:c:c} \bar{R}_{IV} & 0 & W \\ \hdashline \multicolumn{2}{c:}{0} & \bar{R}_V \\ \cdashline{3-3} & & 0 \end{array}\right) Z_{III} = \left(\begin{array}{cc:c} \bar{R}_{IV} & W & \\ & \bar{R}_V & 0 \\ \hdashline \multicolumn{2}{c:}{0} & 0 \end{array}\right) = \left(\begin{array}{c:c} \bar{R} & 0 \\ \hdashline 0 & 0 \end{array}\right).$$

The modified factors are

$$\bar{Q} = \left[\begin{array}{c|c} Q_{\mathrm{IV}} & \\ \hline & I \end{array}\right] \left[\begin{array}{c|c} Q_{\mathrm{III}} & \\ \hline & I \end{array}\right] \left[\begin{array}{c|c} Q_{\mathrm{II}} & \\ \hline & I \end{array}\right] \left[\begin{array}{c|c} I & \\ \hline & Q_{\mathrm{I}} \end{array}\right] Q$$

and

$$\bar{Z} = Z \left[\begin{array}{c|c} I & \\ \hline & Z_{\mathrm{I}} \end{array}\right] \left[\begin{array}{c|c} Z_{\mathrm{II}} & \\ \hline & I \end{array}\right] \left[\begin{array}{c|c} Z_{\mathrm{III}} & \\ \hline & I \end{array}\right].$$

The number of operations necessary to compute the modified factors are summarized in the following table:

Description	Order of multiplications
Compute p, q	$m^2 + n^2$
Determine α, β	$4m(m - t) + 4n(n - t)$
First sweep	$2t^2 + 4mt$
Second sweep	$2t^2 + 4mt$
Additional computation for case III	$2t^2 + 4nt$
*Additional computation for case V	$\frac{4}{3}t^2 + 2t(n + m)$

* It has been assumed that if $W(k)$ is the amount of work when the kth diagonal element of R_{III} is zero, then the expected work is $(1/t)\sum_{k=1}^{t} W(k)$.

The maximum amount of computation necessary, which is of the order of $6\frac{2}{3}t^2 + 5(m^2 + n^2) + 2t(3m - n)$ multiplications, will occur when Case V applies. In the special case, when $\tilde{A}$ and A are both of full column rank, then Z is the identity matrix and the amount of computation is of the order of $5m^2 + 4n^2 + 4mn$ multiplications. This reduces to $13n^2$ when $m = n$.

4.1. *Use of Special Matrices.* The number of operations can be decreased if some of the properties of special matrices outlined in Section 2 are utilized. Each Givens matrix must be multiplied into a Q matrix, Z matrix or upper-triangular matrix, depending upon the current stage of the algorithm. These multiplications can be performed by accumulating the product of each set of Givens matrices into the associated special matrix. Each Q_{I}, Z_{I}, Q_{II}, Z_{II}, $\cdots$, etc. will be either a special matrix or a permutation matrix. The orthogonal matrices Q_{I}, Z_{I}, $\cdots$, etc. will be formed, using Lemma I and Lemma II, as products of the form $\Delta_{\mathrm{I}}\tilde{Q}_{\mathrm{I}}$, $\nabla_{\mathrm{I}}\tilde{Z}_{\mathrm{I}}$, $\Delta_{\mathrm{II}}\tilde{Q}_{\mathrm{II}}$, $\nabla_{\mathrm{II}}\tilde{Z}_{\mathrm{II}}$, $\cdots$, etc. where Δ_{I}, ∇_{I}, Δ_{II}, ∇_{II}, $\cdots$, etc. are diagonal matrices and $\tilde{Q}_{\mathrm{I}}$, $\tilde{Z}_{\mathrm{I}}$, $\cdots$, etc. are special upper- (lower-) Hessenberg matrices with unit sub- (super-) diagonals. In addition, we assume that we modify the factorization

$$QAZ = \left[\begin{array}{c|c} DL^T & 0 \\ \hline 0 & 0 \end{array}\right].$$

At the initial stage, DL^T is unaffected by the pre- and post-multiplication with $\Delta_{\mathrm{I}}\tilde{Q}_{\mathrm{I}}$ and $\tilde{Z}_{\mathrm{I}}\nabla_{\mathrm{I}}$. The products

$$\left[\begin{array}{c|c} I & 0 \\ \hline 0 & \Delta_{\mathrm{I}}\tilde{Q}_{\mathrm{I}} \end{array}\right] Q, \qquad Z \left[\begin{array}{c|c} I & 0 \\ \hline 0 & \tilde{Z}_{\mathrm{I}}\nabla_{\mathrm{I}} \end{array}\right]$$

can be formed using Lemma III, the diagonal matrices being kept separate from the orthogonal products.

During the first sweep, we require the product

$$Q_{\mathrm{II}}\left[\begin{array}{c|c} R & 0 \\ \hline 0 & 0 \end{array}\right].$$

If this matrix is written in the form

$$\Delta_{\mathrm{II}}\tilde{Q}_{\mathrm{II}}\left[\begin{array}{c|c} DL^T & 0 \\ \hline 0 & 0 \end{array}\right],$$

it can be evaluated by bringing the diagonal matrix D to the left of $\tilde{Q}_{\mathrm{II}}$ by suitably altering the special matrix $\tilde{Q}_{\mathrm{II}}$ to $\tilde{Q}_{\mathrm{II}}'$ as in Lemma II. The remaining product involving $\tilde{Q}_{\mathrm{II}}'$ and L^T can be formed using Lemma III with backward recurrence. The multiplication of $\tilde{Q}_{\mathrm{II}}'$ by the current orthogonal matrix is performed similarly to that involving $\tilde{Q}_{\mathrm{I}}$ except that again the diagonal Δ_{I} must be brought through by altering $\tilde{Q}_{\mathrm{II}}$ to $\tilde{Q}_{\mathrm{II}}''$ (say).

If the remainder of the computation is carried out using the same techniques as those just described, the number of multiplications can be summarized as follows:

Description	Order of multiplications
Compute p, q	$m^2 + n^2$
Determine α, β	$2m(m - t) + 2n(n - t)$
First sweep	$t^2 + 2mt$
Second sweep	$2t^2 + 2mt$
Additional computation for case III	$2t^2 + 2nt$
Additional computation for case V	$\frac{4}{3}t^2 + t(n + m)$

The maximum amount of computation necessary is now of the order of $4\frac{1}{3}t^2 + 3(m^2 + n^2) + t(3m - n)$ multiplications, and this reduces to $3(m^2 + n^2) + 2mn$ multiplications in the full rank case. When $n = m = t$ the algorithm requires $8n^2 + O(n)$ operations.

5. Special Rank-One Modifications. We now consider some special cases of the complete orthogonal factorization which occur frequently, namely adding and deleting rows and columns from A. These cases deserve special attention because the modifications can be done in approximately half as many operations as in the general case. Since, in most applications, A is of full column rank, we will deal specifically with this case and modify the factorization

$$QA = \left[\begin{array}{c} R \\ \hline 0 \end{array}\right]$$

where A is $m \times n$, $m \geq n$.

5.1. *Adding and Deleting Rows of A.* We first consider adding a row a^T to A. Assuming, without loss of generality, that this row is added in the $(m + 1)$th position,

we have

$$\left[\begin{array}{c|c} Q & 0 \\ \hline 0 & 1 \end{array}\right]\left[\begin{array}{c} A \\ \hline a^T \end{array}\right] = \left[\begin{array}{c} R \\ \hline 0 \\ \hline a^T \end{array}\right] \equiv T.$$

Elementary orthogonal transformations are now applied from the left to reduce a^T to zero while maintaining the triangularity of R. This is done by defining the sequence

$$T^{(1)} = T, \quad T^{(j+1)} = P^j_{m+1}T^{(j)}, \qquad j = 1, 2, \cdots, n,$$

where $P_{m+1}{}^j$ reduces the $(m+1, j)$th element of $T^{(j)}$ to zero. Note in particular the effect on the column e_{m+1} which has been added to Q. The first n elements are filled in one by one, thereby forming the last column of $\bar{Q}$:

$$P^n_{m+1}P^{n-1}_{m+1}\cdots P^1_{m+1}\left[\begin{array}{c|c} Q & e_{m+1} \\ \hline 0 & \end{array}\right] = \bar{Q} = [\bar{Q}_m \; \vdots \; \bar{q}_{m+1}] \quad \text{say.}$$

Elements $n+1, n+2, \cdots, m$ of $\bar{q}_{m+1}$ remain zero.

To *remove* a row from A, we now simply reverse the above process. This time, we have

$$QA = \left[\begin{array}{c|c} Q_m & q_{m+1} \end{array}\right]\left[\begin{array}{c} \bar{A} \\ \hline a^T \end{array}\right] = \left[\begin{array}{c} R \\ \hline 0 \\ \hline 0 \end{array}\right]\begin{array}{l} \}n \\ \}m-n \\ \}1 \end{array}$$

giving $Q_m\bar{A} + q_{m+1}a^T = QA$. Transformations $P_m{}^{m+1}, P_{m-1}{}^{m+1}, \cdots, P_1{}^{m+1}$ are chosen such that

$$Pq_{m+1} \equiv P_1^{m+1}\cdots P_{m-1}^{m+1}P_m^{m+1}q_{m+1} = e_{m+1}.$$

The last n transformations each introduce a nonzero into the bottom row of

$$\left[\begin{array}{c} R \\ \hline 0 \\ \hline 0 \end{array}\right]$$

(from right to left), giving

$$PQA = \left[\begin{array}{c} \bar{R} \\ \hline 0 \\ \hline r^T \end{array}\right].$$

Looking at the effect on the various partitions of Q, we have

$$PQ = \left[\begin{array}{c|c} \bar{Q} & 0 \\ \hline u^T & 1 \end{array}\right]$$

and, since PQ is orthogonal, it follows immediately that $u = 0$. Thus

$$PQ\left[\begin{array}{c}\bar{A}\\ \hline a^T\end{array}\right] = \left[\begin{array}{c|c}\bar{Q} & 0\\ \hline 0 & 1\end{array}\right]\left[\begin{array}{c}\bar{A}\\ \hline a^T\end{array}\right]$$

$$= \left[\begin{array}{c}\bar{R}\\ \hline 0\\ \hline r^T\end{array}\right]$$

so that $r = a$, and also

$$\bar{Q}\bar{A} = \left[\begin{array}{c}\bar{R}\\ \hline 0\end{array}\right]$$

as required.

Often, it is necessary to modify R without the help of Q. In this case, we really want $\bar{R}$ such that

$$\bar{R}^T\bar{R} = R^TR \pm aa^T,$$

so, clearly, the methods of Section 3 would be applicable. Alternatively, we can continue to use elementary orthogonal transformations as just described. Adding a row to A is simple because Q was not required in any case. To delete a row, we first solve $R^Tp = a$ and compute $\delta^2 = 1 - ||p||^2$. The vector

$$\left[\begin{array}{c}p\\ \hline 0\\ \hline \delta\end{array}\right]\begin{array}{l}\}n\\ \}m-n\\ \}1\end{array} \tag{25}$$

now plays exactly the same role as q_{m+1} above. Dropping the unnecessary zeros in the center of this vector, we have

$$P_1^{n+1}\cdots P_{n-1}^{n+1}P_n^{n+1}\left[\begin{array}{c|c}p & R\\ \hline \delta & 0\end{array}\right] = \left[\begin{array}{c|c}0 & \bar{R}\\ \hline 1 & r^T\end{array}\right]$$

where as usual, the sequence $\{P_i{}^{n+1}\}$ has the effect of reducing p in (23) to zero and introducing the vector r^T beneath $\bar{R}$. Since the $P_i{}^{n+1}$ are orthogonal, it follows that

$$\left[\begin{array}{c|c}0 & 1\\ \hline \bar{R}^T & r\end{array}\right]\left[\begin{array}{c|c}0 & \bar{R}\\ \hline 1 & r^T\end{array}\right] = \left[\begin{array}{c|c}p^T & \delta\\ \hline R^T & 0\end{array}\right]\left[\begin{array}{c|c}p & R\\ \hline \delta & 0\end{array}\right]$$

or

$$\left[\begin{array}{c|c}1 & r^T\\ \hline r & \bar{R}^T\bar{R} + rr^T\end{array}\right] = \left[\begin{array}{c|c}||p||^2 + \delta^2 & p^TR\\ \hline R^Tp & R^TR\end{array}\right]$$

so that $r = R^Tp = a$, and $\bar{R}^T\bar{R} = R^TR - aa^T$ as required.

5.2. *Adding and Deleting Columns of A.* Suppose a column is added to the matrix A, giving

$$\bar{A} = [A \,\vdots\, a].$$

Since

$$QA = \left[\begin{array}{c} R \\ \hline 0 \end{array}\right],$$

we have

$$Q\bar{A} = \left[\begin{array}{c|c} R & u \\ \hline 0 & v \end{array}\right], \tag{26}$$

where $[u^T \vdots v^T] = a^T Q^T$ and u and v are n and $m - n$ vectors, respectively. If an orthogonal matrix P is constructed such that

$$P\left[\begin{array}{c} u \\ \hline v \end{array}\right] = \left[\begin{array}{c} u \\ \hline \gamma \\ \hline 0 \end{array}\right],$$

where $\gamma = \pm \|v\|_2$, then, premultiplying (24) by P leaves the upper-triangular matrix R unchanged and the new factors of $\bar{A}$ are

$$\bar{R} = \left[\begin{array}{c|c} R & u \\ \hline 0 & \gamma \end{array}\right] \quad \text{and} \quad \bar{Q} = PQ.$$

This method represents just a columnwise recursive definition of the QR factorization of A.

When Q is not stored or is unavailable, the vector u can be found by solving the system

$$R^T u = A^T a.$$

The scalar γ is then given by the relation

$$\gamma^2 = \|a\|_2^2 - \|u\|_2^2.$$

Rounding errors could cause this method to fail, however, if the new column a is nearly dependent on the columns of A. In fact, if R is built up by a sequence of these modifications, in which the columns of A are added one by one, the process is exactly that of computing the product $B = A^T A$ and finding the Cholesky factorization $B = R^T R$. It is well known that this is numerically less satisfactory than computing R using orthogonal matrices. In some applications, the sth column of $\bar{Q}$ is available even when $\bar{Q}$ is not and, consequently, γ can be computed more accurately from the relationship $\gamma = a^T \bar{q}_s$, where $\bar{q}_s$ is the sth column of $\bar{Q}$.

Some improvement in accuracy can also be obtained on machines which have the facility for performing the double-length accumulation of inner-products. In this case, the ith element of u is set to

$$u_i = \frac{1}{r_{ii}} \left\{ \sum_{j=1}^{n} a_{ij} a_j - \sum_{j=1}^{i-j} u_j r_{ij} \right\},$$

where the two inner-products are formed as a single sum. Despite these improvements, this is still numerically less satisfactory than the previous method where Q was available.

A further possibility of improving the method arises when one column is being deleted and another is being added. A new column replacing the deleted column is equivalent to a rank-two change in A^TA and can be performed by any one of the methods given in Section 3. Even this is still not ideal, since the computation of the rank-one vectors require the matrix vector product $A^T(a - \bar{a})$, where a is the column being added and $\bar{a}$ is the column being deleted.

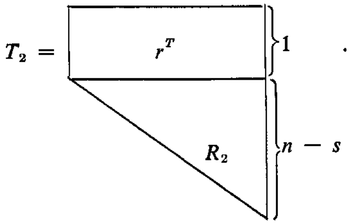
Finally, we describe how to modify the factors when a column is deleted from A. It will be assumed that $\bar{A}$ is obtained from A by deleting the sth column, which as usual will be denoted by a. Deleting the sth column of R gives

$$Q\bar{A} = \left[\begin{array}{c|c} R_1 & T_1 \\ \hline 0 & T_2 \\ \hline 0 & 0 \end{array}\right]\begin{array}{l} \}s-1 \\ \}n-s+1 \\ \}m-n \end{array}$$

where R_1 is an $(s - 1) \times (s - 1)$ upper-triangular matrix, T_1 is an $(s - 1) \times (n - s)$ rectangular matrix and T_2 is an $(n - s + 1) \times (n - s)$ upper-Hessenberg matrix. For example, with $n = 5$, $s = 3$ and $m = 7$, we have

$$\left[\begin{array}{c|c} R_1 & T_1 \\ \hline 0 & T_2 \\ \hline 0 & 0 \end{array}\right] = \left[\begin{array}{cc|cc} x & x & x & x \\ 0 & x & x & x \\ \hline 0 & 0 & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right].$$

Let partition T_2 be of the form



We now choose an orthogonal matrix P which reduces T_2 to upper-triangular form, using one of the methods described earlier. Thus

$$PT_2 = \left[\begin{array}{c} \bar{R}_2 \\ \hline 0 \end{array}\right]\begin{array}{l} \}n-s \\ \}1 \end{array}$$

where P is of the form $P = P_{n-s+1}{}^{n-s} \cdots P_3{}^2P_2{}^1$. The modified triangular factor for $\bar{A}$ is

$$\bar{R} = \begin{array}{c} \left[\begin{array}{c|c} R_1 & T_1 \\ \hline 0 & \bar{R}_2 \\ \hline 0 & 0 \end{array}\right] \end{array} \begin{array}{l} \}s - 1 \\ \}n - s \\ \}m - n + 1 \end{array}$$

If Q is to be updated also, the appropriate rows must be modified; thus

$$Q = \left[\begin{array}{c} Q_1 \\ \hline Q_2 \\ \hline Q_3 \end{array}\right] \begin{array}{l} \}s - 1 \\ \}n - s + 1, \\ \}m - n \end{array} \qquad \bar{Q} = \left[\begin{array}{c} Q_1 \\ \hline PQ_2 \\ \hline Q_3 \end{array}\right].$$

It is sometimes profitable to regard this computation from a different point of view. The partitions of T_2 satisfy the relation $\bar{R}_2{}^T\bar{R}_2 = R_2{}^TR_2 + rr^T$, and this is analogous to the equation $\bar{R}^T\bar{R} = R^TR + aa^T$ which holds when we add a row a^T to A. We conclude that *deleting a column* may be accomplished by essentially the same techniques as used for *adding a row*.

6. Conclusions. In this report, we have presented a comprehensive set of methods which can be used to modify nearly all the factorizations most frequently used in numerical linear algebra. It has not been our purpose to recommend a particular method where more than one exist. Although the amount of computation required for each is given, this will not be the only consideration since the relative efficiencies of the algorithms may alter when applied to particular problems. An example of this is when the Cholesky factors of a positive definite matrix are stored in product form. In this case, the choice of algorithm is restricted to those that form the special matrices explicitly. The relative efficiencies of Methods C1 and C2 are consequently altered.

Division of Numerical Analysis and Computing
National Physical Laboratory
Teddington, England

Computer Science Department
Stanford University
Stanford, California 94305

Division of Numerical Analysis and Computing
National Physical Laboratory
Teddington, England

Applied Mathematics Division
Division of Scientific and Industrial Research
Wellington, New Zealand

* 1. R. H. Bartels, G. H. Golub & M. A. Saunders, "Numerical techniques in mathematical programming," *Nonlinear Programming* (Proc. Sympos., Univ. of Wisconsin, Madison, Wis., 1970), Academic Press, New York, 1970, pp. 123–176. MR **42** #7277.

2. J. M. Bennett, "Triangular factors of modified matrices," *Numer. Math.*, v. 7, 1965, pp. 217–221. MR **31** #1766.

3. D. K. Faddeev, V. N. Kublanovskaya & V. N. Faddeeva, "Sur les systèmes linéaires algébriques de matrices rectangulaires et mal-conditionnées," *Programmation en Mathé-*

matiques Numériques (Actes Colloq. Internat. C.N.R.S. No. 165, Besançon, 1966), Éditions Centre Nat. Recherche Sci., Paris, 1968, pp. 161–170. MR **37** #6017.

* 4. W. M. Gentleman, *Least Squares Computations by Givens Transformations Without Square Roots*, Department of Applied Analysis and Computer Science Report CSRR-2062, University of Waterloo, Waterloo, Ontario, 1972.

* 5. P. E. Gill & W. Murray, *A Numerically Stable Form of the Simplex Algorithm*, National Physical Laboratory, DNAM Report Number 87, Teddington, England, 1970.

* 6. P. E. Gill & W. Murray, "Quasi-Newton methods for unconstrained optimization," *J. Inst. Math. Appl.*, v. 9, 1972, pp. 91–108. MR **45** #9456.

* 7. P. E. Gill & W. Murray, *Two Methods For the Solution of Linearly Constrained and Unconstrained Optimization Problems*, National Physical Laboratory, DNAC Report Number 25, Teddington, England, 1972.

* 8. G. H. Golub & M. A. Saunders, "Linear least squares and quadratic programming," *Integer and Nonlinear Programming*, North-Holland, Amsterdam, 1970.

* 9. G. H. Golub & P. H. Styan, *Numerical Computations for Univariate Linear Models*, Computer Science Department Report Number CS-236-71, Stanford University, Stanford, California, 1971.

10. R. J. Hanson & C. L. Lawson, "Extensions and applications of the Householder algorithm for solving linear least squares problems," *Math. Comp.*, v. 23, 1969, pp. 787–812. MR **41** #2905.

11. R. S. Martin, G. Peters & J. H. Wilkinson, "The *QR* algorithm for real Hessenberg matrices," *Handbook for Automatic Computation*, Vol. 2. Edited by J. H. Wilkinson and C. Reinsch, Springer-Verlag, Berlin and New York, 1971, pp. 359–371.

* 12. W. Murray, *An Algorithm for Indefinite Quadratic Programming*, National Physical Laboratory, DNAC Report Number 1, Teddington, England, 1971.

13. G. Peters & J. H. Wilkinson, "The least squares problem and pseudo-inverses," *Comput. J.*, v. 13, 1970, pp. 309–316.

* 14. M. A. Saunders, *Large-Scale Linear Programming Using the Cholesky Factorization*, Computer Science Department Report Number CS-72-252, Stanford University, Stanford, California, 1972.

* 15. M. A. Saunders, *Product Form of the Cholesky Factorization for Large-Scale Linear Programming*, Computer Science Department Report Number CS-72-301, Stanford University, Stanford, Calif., 1972.

16. J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965. MR **32** #1894.

This page intentionally left blank

PART V

ORTHOGONAL POLYNOMIALS AND QUADRATURE

This page intentionally left blank

22

COMMENTARY, BY WALTER GAUTSCHI

This group of five papers, especially the first and third, has a distinctly "interdisciplinary" character in the sense that classical analysis problems are recast in terms of, and successfully solved by, techniques of linear algebra and, vice versa, problems that have a linear algebra flavor are approached and solved using tools of classical analysis. A similar intriguing mix of analysis and algebra permeates the remaining three papers.

Calculation of Gauss quadrature rules, by Golub and Welsch [53]

The concern here is with the calculation of the n-point Gaussian quadrature rule

$$\int_a^b \omega(t)f(t)\mathrm{d}t = \sum_{\nu=1}^{n} w_\nu f(\tau_\nu) + R_n(f)$$

for the nonnegative weight function $\omega(t)$ on $[a,b]$, i.e., the calculation of the nodes τ_ν and weights w_ν. The connection of this problem with orthogonal polynomials is classical, thanks to work of Gauss [35], Jacobi [61], Christoffel [22], Stieltjes [86], and others: The Gaussian nodes τ_ν are the zeros of π_n, the nth-degree polynomial orthogonal with respect to the weight function ω, and the Gauss weights w_ν are also expressible, in different ways, in terms of these orthogonal polynomials.

An alternative characterization of the Gauss nodes τ_ν can be derived from the classical fact that the ortho*normal* polynomials $\{\tilde{\pi}_k\}$ satisfy a three-term recurrence relation

$$\sqrt{\beta_{k+1}}\tilde{\pi}_{k+1}(t) = (t-\alpha_k)\tilde{\pi}_k(t) - \sqrt{\beta_k}\tilde{\pi}_{k-1}(t), \quad k=0,1,2,\ldots,$$
$$\tilde{\pi}_{-1}=0, \quad \tilde{\pi}_0 = \mu_0^{-1/2},$$

with certain real, resp. positive coefficients α_k, β_k which depend on the weight function ω, and $\mu_0 = \int_a^b \omega(t)\mathrm{d}t$. If $\tilde{\boldsymbol{\pi}}(t) = [\tilde{\pi}_0(t), \tilde{\pi}_1(t), \ldots, \tilde{\pi}_{n-1}(t)]^{\mathrm{T}}$, then indeed,

$$t\tilde{\boldsymbol{\pi}}(t) = \boldsymbol{J}\tilde{\boldsymbol{\pi}}(t) + \sqrt{\beta_n}\tilde{\pi}_n(t)\boldsymbol{e}_n, \quad \boldsymbol{e}_n = [0,0,\ldots,1]^{\mathrm{T}},$$

where $\boldsymbol{J} = \boldsymbol{J}_n$ is the Jacobi matrix of order n for the weight function ω, i.e., the symmetric, tridiagonal matrix having the α_k, $k=0,1,\ldots,n-1$, on the diagonal, and the $\sqrt{\beta_k}$, $k = 1, \ldots, n-1$, on the two side diagonals. There follows, for $t = \tau_\nu$, since $\tilde{\pi}_n(\tau_\nu) = 0$,

$$\boldsymbol{J}\tilde{\boldsymbol{\pi}}(\tau_\nu) = \tau_\nu \tilde{\boldsymbol{\pi}}(\tau_\nu), \quad \nu = 1, 2, \ldots, n,$$

so that τ_ν are the eigenvalues of $\boldsymbol{J}$ and $\tilde{\boldsymbol{\pi}}(\tau_\nu)$ corresponding eigenvectors. This is the first important mathematical ingredient of the present paper. The other is an expression for the Gaussian weights,

$$w_\nu = \frac{1}{\tilde{\boldsymbol{\pi}}^{\mathrm{T}}(\tau_\nu)\tilde{\boldsymbol{\pi}}(\tau_\nu)}, \quad \nu = 1, 2, \ldots, n, \tag{22.1}$$

for which the authors refer to Wilf (apparently to [93, Section 2.9, eqn (69) or Ch. 2, Exercise 9]). The formula, however, is older; see Szegö [89, eqn (3.4.8)], where it is attributed to Shohat [85]. The authors re-express this formula in terms of the eigenvectors $\boldsymbol{q}_\nu$ normalized by $\boldsymbol{q}_\nu^{\mathrm{T}}\boldsymbol{q}_\nu = 1$, i.e., in terms of

$$\boldsymbol{q}_\nu = \frac{\tilde{\boldsymbol{\pi}}(\tau_\nu)}{[\tilde{\boldsymbol{\pi}}^{\mathrm{T}}(\tau_\nu)\tilde{\boldsymbol{\pi}}(\tau_\nu)]^{1/2}} = \tilde{\boldsymbol{\pi}}(\tau_\nu)w_\nu^{1/2},$$

by noting that the first component of $\tilde{\boldsymbol{\pi}}(\tau_\nu)$ is $\mu_0^{-1/2}$, hence

$$w_\nu = \mu_0 \boldsymbol{q}_{\nu,1}^2, \quad \nu = 1, 2, \ldots, n,$$

where $\boldsymbol{q}_{\nu,1}$ is the first component of $\boldsymbol{q}_\nu$.

There is a detailed discussion in the paper of how Francis's QR algorithm with appropriate shifts can be adapted to compute the eigenvalues of a symmetric, tridiagonal matrix (the matrix $\boldsymbol{J}$) and the first components of the normalized eigenvectors. Related software in Algol is provided in the microfiche supplement of the paper.

Interestingly, the same eigenvalue/vector characterization of Gauss rules, and even the same numerical method (QR algorithm), have been suggested a year earlier in the physics literature by Gordon [54, eqn (26) and p. 660]. This work has had considerable impact in the physical sciences and engineering, whereas the work of Golub and Welsch has had a wider impact in the areas of computational mathematics and information science. Both works have actually been submitted for publication less than a month apart, the former on October 20, the latter on November 13 of 1967. Rarely have two important and overlapping works, like these, popped up simultaneously in two entirely different venues!

Similar ideas have since been developed for other quadrature rules of Gaussian type. Indeed, Golub himself [45] was the first to derive eigenvalue/vector algorithms for Gauss–Radau and Gauss–Lobatto formulae. Laurie [65] did it for his anti-Gaussian formulae, and Calvetti and Reichel [19] for a symmetric modification thereof. Quadrature rules involving derivative terms of arbitrary orders on the boundary or outside the interval of integration require first the generation of the appropriate Jacobi matrix before the (simple) internal nodes can be calculated from its eigenvalues and the corresponding weights from the associated eigenvectors; see Golub and Kautsky [47, Section 6] and also Ezzirani and Guessab [32]. This has led to important work on the stable calculation of general interpolatory quadratures (Kautsky and Elhay [63], Elhay and Kautsky [31]). A rather substantial extension is the one to Gauss–Kronrod quadratures due to Laurie [66] (see also the commentary to the last paper). For other types of extended quadrature formulae, see Gout and Guessab [55]. Golub-Welsch type algorithms have been developed also for quadrature rules in the complex plane, for example Gauss-Szegö type formulae on the unit circle (Gragg [57, abstract], [56], Watkins [91, pp. 465–466], Jagels and Reichel [62]), Gauss quadrature on the semicircle (Gautschi and Milovanović [43]), Gauss formulae for the Jacobi weight function with complex parameters (Nuttal and Wherry [78]), or those used to approximate the Bromwich integral in the theory of Laplace transform inversion (Luvison [68], Piessens [79]), and complex Gauss formulae for weighted line integrals in the complex plane (Saylor and Smolarski [84, Section 6]).

There are instances in the area of orthogonal polynomials and quadrature where eigenvalues of more general matrices are of interest, for example banded lower Hessenberg matrices in the case of multiple orthogonal polynomials and related quadrature rules (Coussement and Van Assche [24], Borges [11]), or full-blown upper Hessenberg matrices for zeros of Sobolev orthogonal polynomials (Gautschi and Zhang [44, p. 161]) and also for the Gauss–Szegö quadrature rules mentioned above.

Any advances in improving the QR algorithm for computing eigenvalues and eigenvectors of a symmetric tridiagonal matrix give rise immediately to improved Golub-Welsch algorithms. Some possibilities in this regard are discussed by Laurie [67, Section 2]; for positive definite Jacobi matrices, see also Laurie [67, Section 5] and the references therein.

There still remains, of course, the problem of computing the recurrence coefficients α_k, β_k, if not known explicitly, given the weight function ω. This problem is addressed in Section 4 of the paper, where an algorithm of V.I. Mysovskih is described, which computes these coefficients by a Cholesky decomposition of the Hankel matrix in the moments $\mu_r = \int_a^b t^r \omega(t)\mathrm{d}t$ of the weight function. Any method based on moments, however, is notoriously unstable, owing to severe ill-conditioning (for large n) of the underlying moment map. This was first shown in 1968 by the writer [36]; see also [42, Sections 2.1.4, 2.1.6]. Shortly thereafter, Sack and Donovan, in a technical report [82], introduced the idea of "generalized moments" $m_r = \int_a^b p_r(t)\omega(t)\mathrm{d}t$, where p_r is a polynomial of exact degree r, which, at the suggestion of this writer, they renamed "modified moments" in their formal publication [83]. Under the assumption that the polynomials p_r also satisfy a three-term recurrence relation, but with known coefficients, Sack and Donovan developed an algorithm, later given a more definitive form by Wheeler [92], which computes the desired recurrence coefficients α_k, β_k directly from the modified moments. Wheeler suspected that Chebyshev might already have done something of this nature, which was confirmed by the writer and pinpointed to Chebyshev's 1859 memoir [21], where Wheeler's algorithm indeed appears at the end of Section 3 in the special case of ordinary moments ($p_r(t) = t^r$) and discrete orthogonal polynomials. The algorithm for ordinary, resp. modified moments was therefore named in [37] the Chebyshev, resp. modified Chebyshev algorithm. The latter is not only more efficient than Mysovskih's algorithm, having complexity $O(n^2)$ instead of $O(n^3)$, but is often also more stable. The condition of the underlying modified moment map has been studied in [37, Section 3.3] and [38]; see also [42, Sections 2.1.5, 2.1.6]. For alternative techniques of computing α_k, β_k, based on discretization, see [42, Section 2.2].

Updating and downdating of orthogonal polynomials with data fitting applications, by Elhay, Golub, and Kautsky [30]

The use, in data fitting applications, of (what today are called) discrete orthogonal polynomials can be traced back to a 1859 memoir of Chebyshev [21]. Forsythe [34], a hundred years later and independently, discussed the same procedure and developed it into a viable computer algorithm. The present paper introduces new ideas of updating and downdating in this context, although similar ideas have previously been applied in connection with the related problem of QR factorization of matrices. Mertens [69] reviews downdating algorithms in statistical applications and in the least squares context attributes the concept of downdating to Legendre and Gauss, the originators of least squares theory.

The problem of data fitting is here understood to be the following weighted least squares problem: Given a set $S_N = \{x_j, y_j, w_j^2\}_{j=1}^N$ of N data points $\{x_j, y_j\}$ and positive weights $\{w_j^2\}$, find the polynomial $\hat{q}_n \in \mathbb{P}_n$ of degree $\leq n$ ($< N$) such that

$$\sum_{j=1}^{N} w_j^2 [y_j - \hat{q}_n(x_j)]^2 \leq \sum_{j=1}^{N} w_j^2 [y_j - q(x_j)]^2 \quad \text{for all } q \in \mathbb{P}_n.$$

The inner product and norm naturally associated with this problem are

$$[u, v]_N = \sum_{j=1}^{N} w_j^2 u(x_j) v(x_j), \quad \|u\|_N = \sqrt{[u, u]_N},$$

in terms of which the least squares problem is simply $\|y - \hat{q}\|_N^2 \leq \|y - q\|_N^2$, all $q \in \mathbb{P}_n$. The solution is most conveniently expressed in terms of the polynomials $\{\pi_k\}_{k=0}^{N-1}$ orthonormal with respect to the inner product $[\,\cdot\,, \,\cdot\,]_N$ (the "discrete orthogonal polynomials"), namely as the nth-degree "Fourier polynomial" of y,

$$\hat{q}_n(x) = \sum_{j=0}^{n} c_j \pi_j(x), \quad c_j = [\pi_j, y]_N.$$

With regard to the least squares problem, updating means the following: Determine the solution $\hat{q}_n$ corresponding to the enlarged set $S_{N+1} = S_N \cup \{x_{N+1}, y_{N+1}, w_{N+1}^2\}$ in terms of the solution $\hat{q}_n$ corresponding to the original set S_N. Downdating, conversely, means the determination of $\hat{q}_n$ for S_N in terms of $\hat{q}_n$ for S_{N+1}.

There is a similar problem of up- and downdating for the orthogonal polynomials, more precisely for their Jacobi matrices $\boldsymbol{J}_n$ (cf. the commentary to the first paper): Knowing $\boldsymbol{J}_n$ for S_N, find $\boldsymbol{J}_n$ for S_{N+1}, and vice versa. An algorithm of Gragg and Harrod [58, Section 3] using a sequence of Givens similarity transformations, attributed essentially to Rutishauser [81], can be thought of as an updating procedure in this sense, since it introduces one data point and weight at a time.

As one would expect from the authors, both problems of up- and downdating are solved (in several different ways) by reformulating them in terms of matrices and then applying appropriate techniques of numerical linear algebra.

An application of the updating procedure for Jacobi matrices is made in [29] to generate Jacobi matrices for sums of weight functions.

Up- and downdating algorithms have subsequently been developed for least squares problems in the complex plane, for general complex nodes, for example, in [12, Section 4], and for nodes on the unit circle in [80, Section 3], [2]. For an updating procedure in connection with orthogonal rational functions, and function vectors, having prescribed poles, see [90, Section 3] and [27, Section 5].

Matrices, moments and quadrature, by Golub and Meurant [48]

One of the central themes here is the estimation of matrix functionals $\varphi(\boldsymbol{A}) = \boldsymbol{u}^{\mathrm{T}} f(\boldsymbol{A})\boldsymbol{v}$, where $\boldsymbol{A}$ is a symmetric (usually positive definite) matrix, f a smooth function for which $f(\boldsymbol{A})$ is meaningful, and $\boldsymbol{u}$, $\boldsymbol{v}$ are column vectors. A prototype example, and one given the most attention in this work, is estimating the (i,j)-entry of the inverse matrix $\boldsymbol{A}^{-1}$, in which case $f(t) = t^{-1}$ and $\boldsymbol{u} = \boldsymbol{e}_i$, $\boldsymbol{v} = \boldsymbol{e}_j$ are coordinate vectors. The problem has been treated previously by physicists in connection with the estimation of resolvents, where $\boldsymbol{A} = z\boldsymbol{I} - \boldsymbol{H}$, z is an energy, and $\boldsymbol{H}$ a Hamiltonian, thus $\boldsymbol{A}^{-1}$ is the resolvent of $\boldsymbol{H}$. Much related work can also be found in the quantum chemistry literature; see, e.g., [51, Introduction] and the examples and references given therein.

There are three basic steps in solving the problem: (i) The functional is written as an integral, $\varphi(\boldsymbol{A}) = \int_a^b f(\lambda)\mathrm{d}\alpha(\lambda)$, where $\mathrm{d}\alpha$ is a discrete measure supported on the spectrum $\sigma(\boldsymbol{A})$ of $\boldsymbol{A}$ and $[a,b]$ an interval containing $\sigma(\boldsymbol{A})$. This is done by a spectral resolution of $\boldsymbol{A}$, and in the important case $\boldsymbol{u} = \boldsymbol{v}$ yields a positive measure $\mathrm{d}\alpha$. (ii) The integral is estimated by quadrature rules, typically Gauss, Gauss–Radau, or Gauss–Lobatto rules. These, with an increasing number of nodes, are capable of providing increasingly sharper upper and lower bounds for the integral, provided the derivatives of f have constant sign on $[a,b]$ (as is the case, for example, when $f(t) = t^{-1}$, $a > 0$) and the measure $\mathrm{d}\alpha$ is positive. Otherwise, they may still yield estimates of increasing quality. (iii) Generating the quadrature rules requires the discrete orthogonal polynomials for $\mathrm{d}\alpha$, in particular the Jacobi matrix $\boldsymbol{J} = \boldsymbol{J}(\mathrm{d}\alpha)$ of the measure $\mathrm{d}\alpha$ (cf. the commentary to the first paper), which can be obtained by the Lanczos or the conjugate gradient algorithm. An interesting technical detail is the way the quadrature sums are expressed in terms of the $(1,1)$-element of $f(\boldsymbol{J}^0)$, where $\boldsymbol{J}^0$ is closely related (equal, in the case of Gauss formulae) to the Jacobi matrix $\boldsymbol{J}$ or a leading principal minor matrix thereof.

It is possible to generalize these ideas to the "block" case, where $\boldsymbol{u}$ and $\boldsymbol{v}$ are replaced by an $n \times m$ matrix $\boldsymbol{W}$ (typically with $m = 2$), in which case $\mathrm{d}\alpha$ becomes a matrix-valued measure and one has to deal with matrix-valued orthogonal polynomials and quadrature rules, as is done in Sections 3.3 and 4.3 of the present work.

When $f(t) = t^s$ is any power, not necessarily $s = -1$, and $\boldsymbol{u} = \boldsymbol{v}$, the procedure has previously been described by Golub in [46], and in the case $s = -2$, of interest in ℓ_2 error bounds for systems of linear equations, even before by Dahlquist *et al.* in [25] and also in [26, Section 3]. In the latter work, improved approximations are obtained by the conjugate gradient method and the respective errors estimated as described. In a sequel [49] to the present work, and already in [51, Section 4], the case $s = -1$ is further applied to obtain error bounds and stopping criteria in iterative methods for solving linear systems, notably the conjugate gradient method; see also [70], [33], and for the preconditioned conjugate gradient method, [71], [9]. Applications to constructing preconditioners can be found in [10]. Similar ideas have been pursued by M. Arioli and coworkers in a variety of application areas involving partial differential equations and their discretizations ([4], [8] , [6], [7], [3], [5]). A valuable exposition of error estimates in the conjugate gradient method is [88], where some of the recent results are traced back to the original work of Hestenes and Stiefel [59], and the influence of rounding errors is given serious attention. For the latter, see also [51, Section 5], [87, Section 4], and [94]. For a recent comprehensive review of these and related matters, see [72], especially Sections 3.3 and 5.3.

Altogether different applications of the work of Golub and Meurant are to highly ill-conditioned linear algebraic systems, specifically to the determination of the Tikhonov regularization parameter [14], [15], [20], or to the determination of upper and lower bounds for the Lagrange multipliers in constrained least squares and quadratic problems [52]. The blur identification problem in image processing [76, Section 6] contains yet another application.

The work of Golub and Meurant has inspired other researchers to develop variants of their techniques for estimating matrix functionals. We mention, for example, Calvetti *et al.* [17], [18], where next to Gauss and Gauss–Radau quadratures also anti-Gauss formulae are used (see the commentary to the first paper) and Calvetti *et al.* [16], where functionals $\boldsymbol{u}^{\mathrm{T}}[f(\boldsymbol{A})]^{\mathrm{T}}g(\boldsymbol{A})\boldsymbol{u}$ are estimated for matrices $\boldsymbol{A}$ that are no longer necessarily symmetric, and the quadrature and antiquadrature rules are therefore based on the Arnoldi rather than the Lanczos process.

A stable numerical method for inverting shape from moments, by Golub, Milanfar, and Varah [50]

The basic problem here is the determination of an n-sided polygon P in the complex plane, having vertices z_j, $j = 1, 2, \ldots, n$, given its first $2n-2$ "harmonic" moments $c_k = \int\int_P z^k \mathrm{d}x\mathrm{d}y$, $k = 0, 1, \ldots, 2n-3$. If the associated "complex" moments are defined by $\tau_0 = \tau_1 = 0$, $\tau_k = k(k-1)c_{k-2}$, $k = 2, 3, \ldots, 2n-1$, the vertex reconstruction amounts to solving the system of $2n$ nonlinear equations

$$\sum_{j=1}^{n} a_j z_j^k = \tau_k, \quad k = 0, 1, \ldots, 2n-1.$$

These are formally identical with the equations for a Gaussian quadrature formula (with nodes z_j, weights a_j, and moments τ_k of the underlying weight function), except that all these quantities are now complex and, moreover, the first two moments vanish. While the classical Prony's method is still applicable (it determines the coefficients of the monic polynomial of degree n having the z_j as its zeros), it is notoriously unstable. The object of this work is to develop a solution procedure which, though not necessarily perfectly stable, is more stable than Prony's method.

This is done essentially by reformulating the problem, implicit already in [89, eqn (2.2.9)], as a generalized eigenvalue problem involving two Hankel matrices in the moments, or better, in transformed moments obtained by appropriate scaling and shifting.

In practice, the number n of vertices is usually not known a priori and must be estimated from the given sequence of moments, which, to complicate matters, may be corrupted by noise.

There are a number of potential application areas for procedures as here described, one, discussed previously, to tomographic reconstruction, and another, described in the present work, to the problem of geophysical inversion from gravimetric measurements.

The theoretical results of sensitivity analysis are nicely corroborated by numerical examples. There remain, however, a number of issues for further study, for example, a sound statistical analysis of procedures for estimating the number of vertices, especially in the presence of noise, and the incorporation of a priori geometrical constraints. Some of these issues have been taken up in the more recent work [28].

Computation of Gauss–Kronrod quadrature rules, by Calvetti, Golub, Gragg, and Reichel [13]

In order to economically estimate the error $R_n(f)$ of the n-point Gauss quadrature rule (cf. the commentary to the first paper), Kronrod [64] in 1964 constructed (for the weight function $\omega = 1$ on $[-1, 1]$) an extended Gauss formula

$$\int_a^b \omega(t)f(t)\mathrm{d}t = \sum_{\nu=1}^{n} \lambda_\nu^K f(\tau_\nu) + \sum_{\mu=1}^{n+1} \lambda_\mu^{*K} f(\tau_\mu^K) + R_n^K(f),$$

now called the Gauss–Kronrod quadrature formula, by adjoining to the n Gauss nodes τ_ν additional $n+1$ nodes τ_μ^K – the Kronrod nodes – and selecting them, and all weights λ_ν^K, λ_μ^{*K}, such as to achieve maximum degree of exactness $3n+1$ (at least). The same idea, in a germinal form, can be traced back to the late 19th century (cf. [40]). It turns out that the Kronrod nodes must be the zeros of the polynomial π_{n+1}^K of degree $n+1$ orthogonal to all lower-degree polynomials with respect to the (sign-changing) weight function $\omega(t)\pi_n(t;\omega)$ on (a,b), where π_n is the orthogonal polynomial of degree n relative to the weight function ω. While the polynomial π_{n+1}^K (considered for $\omega = 1$ already by Stieltjes in 1894 without reference to quadrature) always exists uniquely, its zeros may or may not all be real and contained in $[a,b]$. An extensive literature thus evolved dealing precisely with this question of reality, and also with the question of positivity of all weights λ_ν^K, λ_μ^{*K}. (For surveys on this and other aspects of Gauss–Kronrod formulae, see Monegato [74], [75], Gautschi [39], and Notaris [77].) In comparison, the question of actually computing the Gauss–Kronrod formula, when it exists, i.e., computing its nodes and weights, has received less attention; see, however, the recent survey by Monegato [73].

Among the most remarkable computational advances in this area is the algorithm of Laurie [66] for computing positive Gauss–Kronrod formulae. Laurie recognizes the equivalence of this problem with an inverse eigenvalue problem for a symmetric tridiagonal matrix with prescribed entries on the side diagonal; see also [23, pp. 15–16]. His algorithm much resembles the Golub-Welsch algorithm (cf. the commentary to the first paper) for ordinary Gauss formulae. In the present work by Calvetti *et al.*, this algorithm is modified and simplified in the sense that the Gauss nodes τ_ν need not be recomputed (as they are in Laurie's algorithm) in cases where they are already known. Indeed, not even the full tridiagonal Jacobi–Kronrod matrix of order $2n+1$ needs to be generated. The resulting new algorithm is then used by the authors to compute also Kronrod extensions of Gauss–Radau and Gauss–Lobatto formulae.

Modifications required to deal with nonpositive Gauss–Kronrod rules are developed in [1].

The work is too recent to have had a major impact, but it can be expected to find many applications, most likely in the area of adaptive quadrature. One such application (to the motion of droplets) is already briefly mentioned in [60, p. 63].

Summary

Golub's work described here is characterized, on the one hand, by the imaginative use of linear algebra techniques in problems originating elsewhere, and on the other hand, by bringing tools outside of linear algebra to bear on problems involving matrices. Both these features of Golub's work are elaborated in greater detail in the recent essay [41].

Acknowledgment. The writer is grateful for comments by D.P. Laurie, L. Reichel, and Z. Strakoš on an earlier draft of these commentaries.

REFERENCES

1. G. S. Ammar, D. Calvetti, and L. Reichel, Computation of Gauss–Kronrod quadrature rules with non-positive weights. *Electron. Trans. Numer. Anal.*, **9**, 26–38 (1999).
2. G. S. Ammar, W. B. Gragg, and L. Reichel, Downdating of Szegö polynomials and data-fitting applications. *Linear Algebra Appl.*, **172**, 315–336 (1992).
3. M. Arioli, A stopping criterion for the conjugate gradient algorithm in a finite element method framework. *Numer. Math.*, **97**(1), 1–24 (2004).
4. Mario Arioli and Lucia Baldini, A backward error analysis of a null space algorithm in sparse quadratic programming. *SIAM J. Matrix Anal. Appl.*, **23**(2), 425–442 (2001).
5. M. Arioli, D. Loghin, and A. J. Wathen, Stopping criteria for iterations in finite element methods. *Numer. Math.*, **99**(3), 381–410 (2005).
6. Mario Arioli and Gianmarco Manzini, A null space algorithm for mixed finite-element approximations of Darcy's equation. *Comm. Numer. Meth. Engrg.*, **18**(9), 645–657 (2002).
7. Mario Arioli and Gianmarco Manzini, Null space algorithm and spanning trees in solving Darcy's equation. *BIT Numer. Math.*, **43**(Suppl.), 839–848 (2003).
8. M. Arioli, E. Nonlard, and A. Russo, Stopping criteria for iterative methods: applications to PDE's. *Calcolo*, **38**(2), 97–112 (2001).
9. Owe Axelsson and Igor Kaporin, Error norm estimation and stopping criteria in preconditioned conjugate gradient iterations. *Numer. Linear Algebra Appl.*, **8**(4), 265–286 (2001).
10. Michele Benzi and Gene H. Golub, Bounds for the entries of matrix functions with applications to preconditionings. *BIT*, **39**(3), 417–438 (1999).
11. Carlos F. Borges, On a class of Gauss-like quadrature rules. *Numer. Math.*, **67**(3), 271–288 (1994).
12. Adhemar Bultheel and Marc Van Barel, Vector orthogonal polynomials and least squares approximation. *SIAM J. Matrix Anal. Appl.*, **16**(3), 863–885 (1995).

13*. D. Calvetti, G. H. Golub, W. B. Gragg, and L. Reichel, Computation of Gauss–Kronrod quadrature rules. *Math. Comp.*, **69**(231), 1035–1052 (2000).

14. D. Calvetti, G. H. Golub, and L. Reichel, Estimation of the L-curve via Lanczos bidiagonalization. *BIT*, **39**(4), 603–619 (1999).
15. D. Calvetti, P. C. Hansen, and L. Reichel, L-curve curvature bounds via Lanczos bidiagonalization. *Electron. Trans. Numer. Anal.*, **14**, 20–35 (2002).
16. Daniela Calvetti, Sun-Mi Kim, and Lothar Reichel, Quadrature rules based on the Arnoldi process. *SIAM J. Matrix Anal. Appl.*, **26**(3), 765–781 (2005).
17. D. Calvetti, S. Morigi, L. Reichel, and F. Sgallari, Computable error bounds and estimates for the conjugate gradient method. *Numer. Algorithms*, **25**(1–4), 75–88 (2000).
18. D. Calvetti, S. Morigi, L. Reichel, and F. Sgallari, An iteration method with error estimators. *J. Comput. Appl. Math.*, **127**(1–2), 93–119 (2001).
19. Daniela Calvetti and Lothar Reichel, Symmetric Gauss–Lobatto and modified anti-Gauss rules. *BIT*, **43**(3), 541–554 (2003).

20. Daniela Calvetti and Lothar Reichel, Tikhonov regularization of large linear problems. *BIT*, **43**(2), 263–283 (2003).
21. P. L. Chebyshev, Sur l'interpolation par la méthode des moindres carrés. *Mém. Acad. Impér. Sci. St. Petersbourg*, (7) **1**(15), 1–24 (1859). [Also in *Œuvres* I, pp. 473–498.]
22. E. B. Christoffel, Sur une classe particulière de fonctions entières et de fractions continues. *Ann. Mat. Pura Appl.*, (2) **8**, 1–10 (1877). [Also in *Ges. Math. Abhandlungen* I, 65–87.]
23. Moody T. Chu and Gene H. Golub, Structured inverse eigenvalue problems, in *Acta Numerica 2002*, Vol. 11, pp. 1–71, Cambridge University Press, Cambridge (2002).
24. Jonathan Coussement and Walter Van Assche, Gaussian quadrature for multiple orthogonal polynomials. *J. Comput. Appl. Math.*, **178**(1–2), 131–145 (2005).
25. Germund Dahlquist, Stanley C. Eisenstat, and Gene H. Golub, Bounds for the error of linear systems of equations using the theory of moments. *J. Math. Anal. Appl.*, **37**(4), 151–166 (1972).
26. Germund Dahlquist, Gene H. Golub, and Stephen G. Nash, Bounds for the error in linear systems, *Lecture Notes in Control and Information Sci.* 15, pp. 154–172, Springer, Berlin (1979).
27. Steven Delvaux and Marc Van Barel, Orthonormal rational function vectors. *Numer. Math.*, **100**(3), 409–440 (2005).
28. Michael Elad, Peyman Milanfar, and Gene H. Golub, Shape from moments—an estimation theory perspective. *IEEE Trans. Signal Process.*, **52**(7), 1814–1829 (2004).
29. Sylvan Elhay, Gene H. Golub, and Jaroslav Kautsky, Jacobi matrices for sums of weight functions. *BIT*, **32**(1), 143–166 (1992).
30. Sylvan Elhay, Gene H. Golub, and Jaroslav Kautsky, Updating and downdating of orthogonal polynomials with data fitting applications. *SIAM J. Matrix Anal. Appl.*, **12**(2), 327–353 (1991).
31. Sylvan Elhay and Jaroslav Kautsky, Algorithm 655—IQPACK: FORTRAN subroutines for the weights of interpolatory quadratures. *ACM Trans. Math. Software*, **13**(4), 399–415 (1987).
32. Abdelkrim Ezzirani and Allal Guessab, A fast algorithm for Gaussian type quadrature formulae with mixed boundary conditions and some lumped mass spectral approximations. *Math. Comp.*, **68**(225), 217–248 (1999).
33. Bernd Fischer and Gene H. Golub, On the error computation for polynomial based iteration methods, in *Recent advances in iterative methods*, IMA Vol. Math. Appl. 60, Springer, New York, pp. 59–67 (1994).
34. G. E. Forsythe, Generation and use of orthogonal polynomials for data-fitting with a digital computer. *J. Soc. Indust. Appl. Math.*, **5**(2), 74–88 (1957).
35. C. F. Gauss, Methodus nova integralium valores per approximationem inveniendi. *Commentationes Societatis Regiae Scientarium Gottingensis Recentiores*, **3** (1814). [Also in *Werke* III, 163–196.]
36. Walter Gautschi, Construction of Gauss–Christoffel quadrature formulas. *Math. Comp.*, **22**(102), 251–270 (1968).
37. Walter Gautschi, On generating orthogonal polynomials. *SIAM J. Sci. Statist. Comput.*, **3**(3), 289–317 (1982).

38. Walter Gautschi, On the sensitivity of orthogonal polynomials to perturbations in the moments. *Numer. Math.*, **48**(4), 369–382 (1986).

39. Walter Gautschi, Gauss-Kronrod quadrature—a survey, In *Numerical Methods and Approximation Theory III*, G. V. Milovanović (ed.), University of Niš, Niš, pp. 39–66 (1988).

40. Walter Gautschi, A historical note on Gauss–Kronrod quadrature. *Numer. Math.*, **100**(3), 483–484 (2005).

41. Walter Gautschi, The interplay between classical analysis and (numerical) linear algebra — a tribute to Gene H. Golub. *Electron. Trans. Numer. Anal.*, **13**, 119–147 (2002).

42. Walter Gautschi, *Orthogonal polynomials: computation and approximation*, Numerical Mathematics and Scientific Computation, Oxford University Press, Oxford (2004).

43. Walter Gautschi and Gradimir V. Milovanović, Polynomials orthogonal on the semicircle. *J. Approx. Theory*, **46**(3), 230–250 (1986).

44. Walter Gautschi and Minda Zhang, Computing orthogonal polynomials in Sobolev spaces. *Numer. Math.*, **71**(2), 159–183 (1995).

45*. Gene H. Golub, Some modified matrix eigenvalue problems. *SIAM Rev.*, **15**(2), 318–334 (1973).

46. Gene H. Golub, Bounds for matrix moments. *Rocky Mountain J. Math.*, **4**(2), 207–211 (1974).

47. G. H. Golub, and J. Kautsky, Calculation of Gauss quadratures with multiple free and mixed knots. *Numer. Math.*, **41**(2), 147–163 (1983).

48*. Gene H. Golub and Gérard Meurant, Matrices, moments and quadrature, in *Numerical Analysis 1993 (Dundee, 1993)*, Pitman Res. Notes Math. Ser. 303, Longman Sci. Tech., Harlow, pp. 105–156 (1994).

49. G. H. Golub and G. Meurant, Matrices, moments and quadrature II; how to compute the norm of the error in iterative methods. *BIT*, **37**(3), 687–705 (1997).

50. Gene H. Golub, Peyman Milanfar, and James Varah, A stable numerical method for inverting shape from moments. *SIAM J. Sci. Comput.*, **21**(4), 1222–1243 (1999/00) (electronic).

51. Gene H. Golub and Zdeněk Strakoš, Estimates in quadratic formulas. *Numer. Algorithms*, **8**, 241–268 (1994).

52. Gene H. Golub and Urs von Matt, Quadratically constrained least squares and quadratic problems. *Numer. Math.*, **59**(6), 561–580 (1991).

53*. Gene H. Golub and John J. Welsch, Calculation of Gauss quadrature rules. *Math. Comp.*, **23**(106), 221–230 (1969). Microfiche suppl. A1–A10.

54. Roy G. Gordon, Error bounds in equilibrium statistical mechanics. *J. Math. Phys.*, **9**(5), 655–663 (1968).

55. C. Gout and A. Guessab, Extended Lagrange interpolation and nonclassical Gauss quadrature formulae. *Math. Comput. Modelling*, **38**(1–2), 209–228 (2003).

56. William B. Gragg, Positive definite Toeplitz matrices, the Arnoldi process for isometric operators, and Gaussian quadrature on the unit circle. *J. Comput. Appl. Math.*, **46**(1–2), 183–198 (1993).

57. William B. Gragg, The QR algorithm for unitary Hessenberg matrices. *J. Comput. Appl. Math.*, **16**(1), 1–8 (1986).

58. William B. Gragg and William J. Harrod, The numerically stable reconstruction of Jacobi matrices from spectral data. *Numer. Math.*, **44**(3), 317–335 (1984).
59. M. R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear syatems. *J. Research Nat. Bur. Standards*, **49**, 409–436 (1952).
60. W. Holländer and S. K. Zaripov, Hydrodynamically interacting droplets at small Reynolds numbers. *Internat. J. Multiphase Flow*, **31**(1), 53–68 (2005).
61. C. G. J. Jacobi, Ueber Gauß neue Methode, die Werthe der Integrale näherungsweise zu finden. *J. Reine Angew. Math.*, **1**, 301–308 (1826).
62. Carl Jagels and Lothar Reichel, Szegö-Lobatto quadrature rules. *Electr. Trans. Numer. Anal.*, (to appear).
63. J. Kautsky and S. Elhay, Calculation of the weights of interpolatory quadratures. *Numer. Math.*, **40**(3), 407–422 (1982).
64. A. S. Kronrod, *Nodes and weights of quadrature formulas. Sixteen-place tables* (Russian), Izdat. Nauka, Moscow (1964). [Authorized translation by Consultants Bureau, New York, 1965.]
65. Dirk P. Laurie, Anti-Gaussian quadrature formulas. *Math. Comp.*, **65**(214), 739–747 (1996).
66. Dirk P. Laurie, Calculation of Gauss–Kronrod quadrature rules. *Math. Comp.*, **66**(219), 1133–1145 (1997).
67. Dirk P. Laurie, Computation of Gauss-type quadrature formulas. *J. Comput. Appl. Math.*, **127**(1–2), 201–217 (2001).
68. Angelo Luvison, On the construction of Gaussian quadrature rules for inverting the Laplace transform. *Proc. IEEE*, **62**, 637–638 (1974).
69. B. J. A. Mertens, Downdating: interdisciplinary research between statistics and computing. *Statistica Neerlandica*, **55**(3), 358–366 (2001).
70. Gérard Meurant, The computation of bounds for the norm of the error in the conjugate gradient algorithm. *Numer. Algorithms*, **16**(1), 77–87 (1997).
71. Gérard Meurant, Numerical experiments in computing bounds for the norm of the error in the preconditioned conjugate gradient algorithm. *Numer. Algorithms*, **23**(3–4), 353–365 (1999).
72. Gérard Meurant and Zdeněk Strakoš, The Lanczos and conjugate gradient algorithms in finite precision arithmetic, in *Acta Numerica 2006*, Vol. 15, (pp. 471–542) Cambridge University Press, Cambridge (2006).
73. Giovanni Monegato, An overview of the computational aspects of Kronrod quadrature rules. *Numer. Algorithms*, **26**(2), 173–196 (2001).
74. Giovanni Monegato, An overview of results and questions related to Kronrod schemes. In *Numerische Integration*, G. Hämmerlin (ed.), Internat. Ser. Numer. Math. 45, Birkhäuser, Basel, pp. 231–240 (1979).
75. Giovanni Monegato, Stieltjes polynomials and related quadrature rules. *SIAM Rev.*, **24**(2), 137–158 (1982).
76. Nhat Nguyen, Peyman Milanfar, and Gene Golub, Efficient generalized cross-validation with applications to parametric image restoration and resolution enhancement. *IEEE Trans. Image Process.*, **10**(9), 1299–1308 (2001).
77. Sotirios E. Notaris, An overview of results on the existence or nonexistence and the error term of Gauss–Kronrod quadrature formulae, in *Approximation and Computation*, R.V.M. Zahar (ed.), Internat. Ser. Numer. Math. 119, Birkhäuser, Boston, pp. 485–496 (1994).

78. J. Nuttall and C. J. Wherry, Gaussian integration for complex weight functions. *J. Inst. Math. Appl.*, **21**(2), 165–170 (1978).
79. Robert Piessens, Comments on: “On the construction of Gaussian quadrature rules for inverting the Laplace transform” by A. Luvison. *Proc. IEEE*, **63**, 817–818 (1975).
80. L. Reichel, G. S. Ammar, and W. B. Gragg, Discrete least squares approximation by trigonometric polynomials. *Math. Comp.*, **57**(195), 273–289 (1991).
81. H. Rutishauser , On Jacobi rotation patterns. In *Experimental Arithmetic, High Speed Computing and Mathematics*, Proc. Sympos. Appl. Math. 15, Amer. Math. Soc., Providence, RI, pp. 219–239 (1963).
82. R. A. Sack and A. F. Donovan, An algorithm for Gaussian quadrature given generalized moments, Technical report, Dept. Math., University of Salford, Salford, UK (1969).
83. R. A. Sack and A. F. Donovan, An algorithm for Gaussian quadrature given modified moments. *Numer. Math.*, **18**(5), 465–478 (1972).
84. Paul E. Saylor and Dennis C. Smolarski, Why Gaussian quadrature in the complex plane?. *Numer. Algorithms*, **26**(3), 251–280 (2001).
85. J. A. Shohat, On a certain formula of mechanical quadratures with non-equidistant ordinates. *Trans. Amer. Math. Soc.*, **31**, 448–463 (1929).
86. T. J. Stieltjes, Quelques recherches sur la théorie des quadratures dites mécaniques. *Ann. Sci. Éc. Norm. Paris*, (3) **1**, 409–426 (1884). [Also in *Œuvres* I, 377–396.]
87. Z. Strakoš and J. Liesen, On numerical stability in large scale linear algebraic computations. *Z. Angew. Math. Mech.*, **85**(5), 307–325 (2005).
88. Zdeněk Strakoš and Petr Tichý, On error estimation in the conjugate gradient method and why it works in finite precision computations. *Electron. Trans. Numer. Anal.*, **13**, 56–80 (2002).
89. Gabor Szegö, *Orthogonal polynomials*, AMS Colloquium Publications 23, Amer. Math. Soc., New York (1939).
90. Marc Van Barel, Dario Fasino, Luca Gemignani, and Nicola Mastronardi, Orthogonal rational functions and structured matrices. *SIAM J. Matrix Anal. Appl.*, **26**(3), 810–829 (2005).
91. David S. Watkins, Some perspectives on the eigenvalue problem. *SIAM Rev.*, **35**(3), 430–471 (1993).
92. John C. Wheeler, Modified moments and Gaussian quadratures. *Rocky Mountain J. Math.*, **4**(2), 287–296 (1974).
93. Herbert S. Wilf, *Mathematics for the physical sciences*, Wiley, New York (1962).
94. Wolfgang Wülling, The stabilization of weights in the Lanczos and conjugate gradient method, *BIT Numer. Math.*, **45**(2), 395–414 (2005).

An asterisk denotes a paper reprinted in this volume.

23

CALCULATION OF GAUSS QUADRATURE RULES (WITH J. H. WELSCH)

Math. Comp. (23), 221–230, s1-s10 (1969).

Calculation of Gauss Quadrature Rules*

By Gene H. Golub and John H. Welsch**

Abstract. Several algorithms are given and compared for computing Gauss quadrature rules. It is shown that given the three term recurrence relation for the orthogonal polynomials generated by the weight function, the quadrature rule may be generated by computing the eigenvalues and first component of the orthornormalized eigenvectors of a symmetric tridiagonal matrix. An algorithm is also presented for computing the three term recurrence relation from the moments of the weight function. ■

Introduction. Most numerical integration techniques consist of approximating the integrand by a polynomial in a region or regions and then integrating the polynomial exactly. Often a complicated integrand can be factored into a nonnegative "weight" function and another function better approximated by a polynomial, thus

$$\int_a^b g(t)dt = \int_a^b \omega(t)f(t)dt \approx \sum_{j=1}^{N} w_j f(t_j) .$$

Hopefully, the quadrature rule $\{w_j, t_j\}_{j=1}^N$ corresponding to the weight function $\omega(t)$ is available in tabulated form, but more likely it is not. We present here two algorithms for generating the Gaussian quadrature rule defined by the weight function when:

(a) the three term recurrence relation is known for the orthogonal polynomials generated by $\omega(t)$, and

(b) the moments of the weight function are known or can be calculated.

In [6], Gautschi presents an algorithm for calculating Gauss quadrature rules when neither the recurrence relationship nor the moments are known.

1. Definitions and Preliminaries. Let $\omega(x) \geqq 0$ be a fixed *weight function* defined on $[a, b]$. For $\omega(x)$, it is possible to define a sequence of polynomials $p_0(x), p_1(x), \cdots$ which are orthonormal with respect to $\omega(x)$ and in which $p_n(x)$ is of exact degree n so that

$$(1.1) \qquad \begin{aligned} \int_a^b \omega(x)p_m(x)p_n(x)dx &= 1 \quad \text{when } m = n , \\ &= 0 \quad \text{when } m \neq n . \end{aligned}$$

The polynomial $p_n(x) = k_n \prod_{i=1}^n (x - t_i) , k_n > 0$, has n real roots $a < t_1 < t_2 <$

Received November 13, 1967, revised July 12, 1968.

* The work of the first author was in part supported by the Office of Naval Research and the National Science Foundation; the work of the second author was in part supported by the Atomic Energy Commission.

** Present address: Hewlett-Packard Company, Palo Alto, California 94304.

$\cdots < t_n < b$. The roots of the orthogonal polynomials play an important role in Gaussian quadrature.

THEOREM. *Let* $f(x) \in C^{2N}[a, b]$, *then*

$$\int_a^b \omega(x) f(x) dx = \sum_{j=1}^{N} w_j f(t_j) + \frac{f^{(2N)}(\xi)}{(2N)! k_N{}^2}, \qquad (a < \xi < b),$$

where

$$w_j = -\frac{k_{N+1}}{k_N} \frac{1}{p_{N+1}(t_j) p_N{}'(t_j)}, \quad \left(p_N{}'(t_j) = \left.\frac{dp_N(t)}{dt}\right|_{t=t_j} \right), \qquad j = 1, 2, \cdots, N.$$

Thus the Gauss quadrature rule is exact for all polynomials of degree $\leqq 2N - 1$.

Proofs of the above statements and Theorem can be found in Davis and Rabinowitz [4, Chapter 2].

Several algorithms have been proposed for calculating $\{w_j, t_j\}_{j=1}^N$; cf. [10], [11]. In this note, we shall give effective numerical algorithms which are based on determining the eigenvalues and the first component of the eigenvectors of a symmetric tridiagonal matrix.

2. Generating the Gauss Rule. Any set of orthogonal polynomials, $\{p_j(x)\}_{j=1}^N$, satisfies a three term recurrence relationship:

$$p_j(x) = (a_j x + b_j) p_{j-1}(x) - c_j p_{j-2}(x), \tag{2.1}$$

$$j = 1, 2, \cdots, N; \quad p_{-1}(x) \equiv 0, \quad p_0(x) \equiv 1,$$

with $a_j > 0$, $c_j > 0$. The coefficients $\{a_j, b_j, c_j\}$ have been tabulated for a number of weight functions $\omega(x)$, cf. [8]. In Section 4 we shall give a simple method for generating $\{a_j, b_j, c_j\}$ for any weight function.

Following Wilf [12], we may identify (2.1) with the matrix equation

$$x \begin{bmatrix} p_0(x) \\ p_1(x) \\ \cdot \\ \\ \cdot \\ \cdot \\ p_{N-1}(x) \end{bmatrix} = \begin{bmatrix} -b_1/a_1, & 1/a_1, & 0 & & & & \\ c_2/a_2, & -b_2/a_2, & 1/a_2 & & & \bigcirc & \\ 0 & \cdot & \cdot & \cdot & & & \\ & & & & & & \\ & & & \cdot & \cdot & \cdot & \\ \bigcirc & & & & \cdot & \cdot & 1/a_{N-1} \\ & & & & & c_N/a_N, & -b_N/a_N \end{bmatrix} \begin{bmatrix} p_0(x) \\ p_1(x) \\ \cdot \\ \\ \cdot \\ \cdot \\ p_{N-1}(x) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \\ p_N(x)/a_N \end{bmatrix}$$

or, equivalently in matrix notation

$$x\mathbf{p}(x) = T\mathbf{p}(x) + (1/a_N)p_N(x)\mathbf{e}_N$$

where T is the tridiagonal matrix and $\mathbf{e}_N = (0, 0, \cdots, 0, 1)^T$. Thus $p_N(t_j) = 0$ if and only if $t_j\mathbf{p}(t_j) = T\mathbf{p}(t_j)$ where t_j is an eigenvalue of the tridiagonal matrix T. In [12], it is shown that T is symmetric if the polynomials are orthonormal. If T is not symmetric, then we may perform a diagonal similarity transformation which will yield a symmetric tridiagonal matrix J. Thus

$$DTD^{-1} = J = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \bigcirc & \\ 0 & \cdot & \cdot & \cdot & & \\ & & & & & \\ & & \cdot & \cdot & \cdot & \\ \bigcirc & & & \cdot & \cdot & \beta_{N-1} \\ & & & & \beta_{N-1} & \alpha_N \end{bmatrix}$$

where

$$(2.2) \qquad \alpha_i = -\frac{b_i}{a_i}, \quad \beta_i = \left(\frac{c_{i+1}}{a_i a_{i+1}}\right)^{1/2}.$$

It is shown by Wilf [12] that as a consequence of the Christoffel-Darboux identity

$$(2.3) \qquad w_j[\mathbf{p}(t_j)]^T[\mathbf{p}(t_j)] = 1, \qquad j = 1, 2, \cdots, N$$

where $\mathbf{p}(t_j)$ corresponds to the eigenvector associated with the eigenvalue t_j. Suppose that the eigenvectors of T are calculated so that

$$(2.4) \qquad J\mathbf{q}_j = t_j\mathbf{q}_j, \qquad j = 1, 2, \cdots, N$$

with $\mathbf{q}_j{}^T\mathbf{q}_j = 1$. If

$$(2.5) \qquad \mathbf{q}_j^T = (q_{1,j}, q_{2,j}, \cdots, q_{N,j}),$$

then $q_{1,j}^2 = w_j(p_0(t_j))^2$ by (2.3). Thus from (1.1), we see

$$(2.6) \qquad w_j = \frac{q_{1,j}^2}{p_0^2(t_j)} = \frac{q_{1,j}^2}{k_0^2} = q_{1,j}^2 \times \int_a^b \omega(x)dx \equiv q_{1,j}^2 \times \mu_0.$$

Consequently, if one can compute the eigenvalues of T and the first component of the orthonormal eigenvectors, one is able to determine the Gauss quadrature rule.

3. The *Q-R* Algorithm. One of the most effective methods of computing the eigenvalues and eigenvectors of a symmetric matrix is the Q-R algorithm of Francis [5]. The Q-R algorithm proceeds as follows:

Begin with the given matrix $J = J^{(0)}$, compute the factorization $J^{(0)} = Q^{(0)}R^{(0)}$ where $Q^{(0)T}Q^{(0)} = I$ and $R^{(0)}$ is an upper triangular matrix, and then multiply the matrices in reverse order so that

$$J^{(1)} = R^{(0)}Q^{(0)} = Q^{(0)^T}J^{(0)}Q^{(0)} .$$

Now one treats the matrix $J^{(1)}$ in the same fashion as the matrix $J^{(0)}$, and a sequence of matrices is obtained by continuing *ad infinitum.* Thus

$$\begin{aligned} J^{(i)} &= Q^{(i)}R^{(i)} , \\ J^{(i+1)} &= R^{(i)}Q^{(i)} = Q^{(i+1)}R^{(i+1)} \end{aligned} \tag{3.1}$$

so that

$$J^{(i+1)} = Q^{(i)^T}J^{(i)}Q^{(i)} = Q^{(i)^T}Q^{(i-1)^T} \cdots Q^{(0)^T}JQ^{(0)}Q^{(1)} \cdots Q^{(i)} . \tag{3.2}$$

Since the eigenvalues of J are distinct and real for orthogonal polynomials, a real translation parameter λ may be chosen so that the eigenvalues of $J^{(i)} - \lambda I$ are distinct in modulus. Under these conditions, it is well known [5] that $J^{(i)} - \lambda I$ converges to the diagonal matrix of eigenvalues of $J - \lambda I$ as $i \to \infty$ and that $P^{(i)} = Q^{(0)} \times Q^{(1)} \times \cdots \times Q^{(i)}$ converges to the orthogonal matrix of eigenvectors of J. The method has the advantage that the matrix $J^{(i)} - \lambda I$ remains tridiagonal throughout the computation.

Francis has shown that it is not necessary to compute the decomposition (3.1) explicitly but it is possible to do the calculation (3.2) directly. Let

$$\{S^{(i)}\}_{k,1} = \{Q^{(i)}\}_{k,1} \qquad (k = 1, 2, \cdots, N) ,$$

(i.e., the elements of the first column of $S^{(i)}$ are equal to the elements of the first column of $Q^{(i)}$). Then if

(i) $K^{(i+1)'} = S^{(i)^T}J^{(i)}S^{(i)}$,

(ii) $K^{(i+1)}$ is a tridiagonal matrix,

(iii) $J^{(i)}$ is nonsingular,

(iv) the subdiagonal elements of $K^{(i+1)}$ are positive, it follows that $K^{(i+1)} = J^{(i+1)}$.

For the tridiagonal matrices, the calculation is quite simple. Dropping the iteration counter i, let

$$Z_p = \begin{array}{cc} \begin{array}{cccccccccc} & & & & (p) & (p+1) & & & & \end{array} & \\ \left[\begin{array}{cccccccccc} 1 & & & & \cdot & \cdot & & & & \\ & \cdot & & & \cdot & \cdot & & \bigcirc & & \\ & & \cdot & & \cdot & \cdot & & & & \\ & & & 1 & \cdot & \cdot & & & & \\ \cdot & \cdot & \cdot & \cdot & \cos\theta_p & \sin\theta_p & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \sin\theta_p & -\cos\theta_p & \cdot & \cdot & \cdot & \cdot \\ & & & & \cdot & \cdot & 1 & & & \\ & \bigcirc & & & \cdot & \cdot & & \cdot & & \\ & & & & \cdot & \cdot & & & \cdot & \\ & & & & \cdot & \cdot & & & & 1 \end{array}\right] & \begin{array}{c} \\ \\ \\ \\ (p) \\ (p+1) \\ \\ \\ \\ \\ \end{array} \end{array}$$

Then $\cos\theta_1$ is chosen so that

$$\{Z_1J\}_{k,1} = 0 , \qquad k = 2, 3, \cdots, N .$$

Let

$$J = \begin{bmatrix} a_1 & b_1 & 0 & & & \\ b_1 & a_2 & b_2 & \cdot & & \bigcirc \\ 0 & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & \bigcirc & & \cdot & \cdot & b_{N-1} \\ & & & & b_{N-1} & a_N \end{bmatrix}.$$

The matrix

$$Z_1 J Z_1 = \begin{bmatrix} a_1' & b_1' & d_1 & & & & \\ b_1' & a_2' & b_2' & 0 & & & \bigcirc \\ d_1 & b_2' & a_3 & b_3 & & & \\ & 0 & b_3 & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & \bigcirc & & & \cdot & \cdot & b_{N-1} \\ & & & & & b_{N-1} & a_N \end{bmatrix},$$

where the primes indicate altered elements of J; then

$$K = Z_{N-1} Z_{N-2} \cdots Z_1 J Z_1 \cdots Z_{N-1}$$

and $Z_2, \cdots, Z_{N-1}$ are constructed so that K is tridiagonal. The product of all the orthogonal rotations yields the matrix of orthogonal eigenvectors. To determine $\{w_j\}_{j=1}^N$, however, we need only the first component of the orthonormal eigenvector. Thus, using (2.3)

$$\mathbf{q}^T \equiv [q_{1,1}, q_{1,2}, \cdots, q_{1,N}] = [1, 0, 0, \cdots, 0] \times \prod_{i=0}^{\infty} (Z_1^{(i)} \times Z_2^{(i)} \times \cdots \times Z_{N-1}^{(i)})$$

and it is not necessary to compute the entire matrix of eigenvectors. More explicitly, for $j = 1, 2, \cdots, N-1$

$$\sin\theta_j^{(i)} = d_{j-1}^{(i)} / [(d_{j-1}^{(i)})^2 + (\bar{b}_{j-1}^{(i)})^2]^{1/2},$$

$$\cos\theta_j^{(i)} = \bar{b}_{j-1}^{(i)} / [(d_{j-1}^{(i)})^2 + (\bar{b}_{j-1}^{(i)})^2]^{1/2},$$

$$\begin{aligned} a_j^{(i+1)} &= \bar{a}_j^{(i)} \cos^2\theta_j^{(i)} + 2\tilde{b}_j^{(i)} \cos\theta_j^{(i)} \sin\theta_j^{(i)} + a_{j+1}^{(i)} \sin^2\theta_j^{(i)}, \\ \bar{a}_{j+1}^{(i)} &= \bar{a}_j^{(i)} \sin^2\theta_j^{(i)} - 2\tilde{b}_j^{(i)} \cos\theta_j^{(i)} \sin\theta_j^{(i)} + a_{j+1}^{(i)} \cos^2\theta_j^{(i)}, \\ b_{j-1}^{(i+1)} &= \bar{b}_{j-1}^{(i)} \cos\theta_j^{(i)} + d_{j-1}^{(i)} \sin\theta_j^{(i)} = [(\bar{b}_{j-1}^{(i)})^2 + (d_{j-1}^{(i)})^2]^{1/2}, \\ \bar{b}_j^{(i)} &= (\bar{a}_j^{(i)} - a_{j+1}^{(i)}) \sin\theta_j^{(i)} \cos\theta_j^{(i)} + \tilde{b}_j^{(i)} (\sin^2\theta_j^{(i)} - \cos^2\theta_j^{(i)}), \\ \tilde{b}_{j+1}^{(i)} &= -b_{j+1}^{(i)} \cos\theta_j^{(i)}, \\ d_j^{(i)} &= b_{j+1}^{(i)} \sin\theta_j^{(i)}, \; z_j^{(i+1)} = \bar{z}_j^{(i)} \cos\theta_j + z_{j+1}^{(i)} \sin\theta_j, \\ \bar{z}_{j+1}^{(i)} &= \bar{z}_j^{(i)} \sin\theta_j^{(i)} - z_{j+1}^{(i)} \cos\theta_j^{(i)}, \end{aligned} \tag{3.3}$$

with

$$d_0^{(i)} = b_1^{(i)}, \quad \bar{b}_0^{(i)} = (a_1^{(i)} - \lambda^{(i)}),$$

$$\bar{a}_1^{(i)} = a_1^{(i)}, \quad \tilde{b}_1^{(i)} = b_1^{(i)}, \quad \bar{z}_1^{(i)} = z_1^{(i)}.$$

Initially

$$z_1^{(0)} = 1, \quad z_j^{(0)} = 0 \quad \text{for } j = 2, \cdots, N$$

so that $\mathbf{z}^{(i)T} \to \mathbf{q}^T$ as $i \to \infty$. In the actual computation, no additional storage is required for $\{\bar{a}_j^{(i)}, \bar{b}_j^{(i)}, \bar{\bar{b}}_j^{(i)}, \bar{z}_j^{(i)}\}$ since they may overwrite $\{a_j^{(i)}, b_j^{(i)}, z_j^{(i)}\}$. We choose $\lambda^{(i)}$ as an approximation to an eigenvalue; usually, it is related to the eigenvalues of the matrix

$$\begin{bmatrix} a_{N-1}^{(i)} & b_{N-1}^{(i)} \\ b_{N-1}^{(i)} & a_N^{(i)} \end{bmatrix}.$$

When $b_{N-1}^{(i)}$ is sufficiently small, $a_N^{(i)}$ is taken as eigenvalue and N is replaced by $N - 1$.

4. Determining the Three Term Relationship from the Moments. For many weight functions, the three term relationship of the orthogonal polynomials have been determined. In some situations, however, the weight function is not known explicitly, but one has a set of $2N + 1$ moments, viz.

$$\mu_k = \int_a^b \omega(x)x^k dx \qquad k = 0, 1, \cdots, 2N\,.$$

Based on an elegant paper of Mysovskih [9], Gautschi*** has given a simple derivation of the three term relationship which we give below. This result also follows from certain determinantal relations (cf. [7]).

Let $\Omega \subset E^n$ be a domain in n-dimensional Euclidean space and $\omega(\mathbf{x}) \geqq 0$ be a weight function on Ω for which all "moments"

$$\mu_{\gamma_1,\gamma_2,\ldots,\gamma_n} = \int_\Omega \omega(\mathbf{x})x_1^{\gamma_1}x_2^{\gamma_2} \cdots x_n^{\gamma_n}d\mathbf{x}$$

exist, and $\mu_{0,0,\cdots,0} > 0$. Enumerate the monomials

$$x_1^{\gamma_1}x_2^{\gamma_2} \cdots x_n^{\gamma_n}, \quad \gamma_1 \geqq 0, \cdots, \gamma_n \geqq 0$$

as $\{\varphi_i(\mathbf{x})\}_{i=1}^L$, whereby $i < j$ if degree $\varphi_i <$ degree φ_j, the enumeration within the same degree being arbitrary. In particular, $\varphi_1(\mathbf{x}) = 1$. Let

$$M = [(\varphi_i, \varphi_j)]_{i,j=1}^L \equiv \{m_{ij}\}$$

denote the "Gram matrix" for the system $\{\varphi_i(\mathbf{x})\}_{i=1}^L$, where

$$(\varphi_i, \varphi_j) = \int_\Omega \omega(\mathbf{x})\varphi_i(\mathbf{x})\varphi_j(\mathbf{x})d\mathbf{x}\,.$$

Note M is positive definite. Let $M = R^TR$ be the Cholesky decomposition of M with

$$r_{ii} = \left(m_{ii} - \sum_{k=1}^{i-1} r_{ki}^2\right)^{1/2}$$

and

$$r_{ij} = \left(m_{ij} - \sum_{k=1}^{i-1} r_{ki}r_{kj}\right)\bigg/ r_{ii}\,, \qquad i < j \tag{4.1}$$

*** Personal communication.

for i and j between 1 and L. Let

$$R^{-1} \equiv \begin{bmatrix} s_{11} & s_{12} & \cdot & \cdot & \cdot & s_{1L} \\ & s_{22} & \cdot & \cdot & \cdot & s_{2L} \\ & & \cdot & & & \cdot \\ & & & & & \\ & & & \cdot & & \cdot \\ \bigcirc & & & & \cdot & \cdot \\ & & & & & s_{LL} \end{bmatrix}.$$

THEOREM (MYSOVSKIH). *The polynomials*

$$F_j(\mathbf{x}) = s_{1j}\varphi_1(\mathbf{x}) + s_{2j}\varphi_2(\mathbf{x}) + \cdots + s_{jj}\varphi_j(\mathbf{x}) \qquad (j = 1, 2, \cdots, L)$$

form an orthonormal system.

Now in the special case $n = 1$, one has $\varphi_j(x) = x^{j-1}$ with $L = N + 1$, and M is just the "Hankel" matrix in the moments. Moreover, we know in this case $F_j(x) = p_{j-1}(x)$, a polynomial of degree $j - 1$, and $\{p_j(x)\}_{j=0}^{N}$ satisfy

$$xp_{j-1}(x) = \beta_{j-1}p_{j-2}(x) + \alpha_j p_{j-1}(x) + \beta_j p_j(x)\,, \qquad j = 1, \cdots, N\,, \tag{4.2}$$

where $p_{-1}(x) \equiv 0$. Comparing the coefficients of x^j and x^{j-1} on either side of this identity, one gets

$$s_{jj} = \beta_j s_{j+1,j+1}$$

$$s_{j-1,j} = \alpha_j s_{jj} + \beta_j s_{j,j+1}$$

and so

$$\beta_j = \frac{s_{j,j}}{s_{j+1,j+1}}\,, \qquad \alpha_j = \frac{s_{j-1,j}}{s_{j,j}} - \frac{s_{j,j+1}}{s_{j+1,j+1}}\,.$$

Further, if

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdot & \cdot & \cdot & r_{1,N+1} \\ & r_{22} & \cdot & \cdot & \cdot & r_{2,N+1} \\ & & \cdot & & & \cdot \\ & & & & & \\ & & & \cdot & & \cdot \\ \bigcirc & & & & \cdot & \cdot \\ & & & & & r_{N+1,N+1} \end{bmatrix}$$

a straightforward computation shows that

$$s_{jj} = \frac{1}{r_{jj}}\,, \qquad s_{j,j+1} = \frac{-r_{j,j+1}}{r_{j,j}r_{j+1,j+1}}\,.$$

Thus

$$\begin{aligned} \alpha_j &= \frac{r_{j,j+1}}{r_{j,j}} - \frac{r_{j-1,j}}{r_{j-1,j-1}}\,, \qquad j = 1, 2, \cdots, N\,, \\ \beta_j &= \frac{r_{j+1,j+1}}{r_{j,j}}\,, \qquad j = 1, 2, \cdots, N - 1\,, \end{aligned} \tag{4.3}$$

with $r_{0,0} = 1$, $r_{0,1} = 0$.

5. Description of Computational Procedures. In the microfiche section of this issue there are three ALGOL 60 procedures for performing the algorithms presented above. We have tried to keep the identifiers as close to the notation of the equations as possible without sacrificing storage or efficiency. The weights and abscissas of the quadrature rule are the result of the procedure GAUSSQUADRULE which must be supplied with the recurrence relation by either procedure GENORTHOPOLY or CLASSICORTHOPOLY. The former requires the moments of the weight function and the latter the name of the particular orthogonal polynomial. A short description of each procedure follows.

CLASSICORTHOPOLY produces μ_0 and the normalized three term recurrence relationship (a_j, b_j) for six well-known kinds of orthogonal polynomials:

KIND = 1, Legendre polynomials $P_n(x)$ on $[-1.0, +1.0]$, $\omega(x) = 1.0$.

KIND = 2, Chebyshev polynomials of the first kind $T_n(x)$ on $[-1.0, +1.0]$, $\omega(x) = (1 - x^2)^{-1/2}$.

KIND = 3, Chebyshev polynomials of the second kind $U_n(x)$ on $[-1.0, +1.0]$, $\omega(x) = (1 - x^2)^{+1/2}$.

KIND = 4, Jacobi polynomials $P_n^{(\alpha,\beta)}(x)$ on $[-1.0, +1.0]$, $\omega(x) = (1 - x)^\alpha(1 + x)^\beta$ for $\alpha > -1$ and $\beta > -1$.

KIND = 5, Laguerre polynomials $L_n^{(\alpha)}(x)$ on $[0, +\infty)$, $\omega(x) = e^{-x}x^\alpha$ for $\alpha > -1$.

KIND = 6, Hermite polynomials $H_n(x)$ on $(-\infty, +\infty)$, $\omega(x) = e^{-x^2}$.

Notice that this procedure requires a real procedure to evaluate the gamma function $\Gamma(x)$.

GENORTHOPOLY uses the 2N + 1 moments of the weight function which are supplied in MU[0] through MU[2 $\otimes$ N] to compute the α_j's and β_j's of formula (4.2). First, the Cholesky decomposition (formula 4.1) of the moment matrix is placed in the upper right triangular part of the array R, then the formulas (4.3) are used to compute the α_j's and β_j's which are placed in the arrays A and B respectively.

GAUSSQUADRULE has two modes of operation controlled by the Boolean parameter SYMM which indicates whether the tridiagonal matrix is symmetric or not. When the recurrence relation is produced by GENORTHOPOLY or by CLASSICORTHOPOLY, SYMM is *true*. If SYMM is *false*, the matrix is symmetricized using the formulas (2.2). The diagonal elements α_i are stored in A[I] and the off diagonal elements β_i are stored in B[I].

Beginning at label SETUP, several calculations and initializations are done: the l_1 norm of the tridiagonal matrix and the relative zero tolerance are computed; the first component of each eigenvector W[I] and the Q-R iteration are initialized. LAMBDA is a variable subtracted off the diagonal elements to accelerate convergence of the Q-R iteration and control to some extent in what order the eigenvalues (abscissas) are found. It begins with a value (= NORM) outside and to the right of the interval containing the abscissas and moves to the left as the abscissas are found; thus the abscissas will be in ascending order in the array T (just to be sure an exchange sort is used at label SORT).

The maximum (EIGMAX) of the eigenvalues (LAMBD1 and LAMBD2) of the lower 2 × 2 submatrix is compared to the maximum (RHO) from the last iteration. If they are close, LAMBDA is replaced by EIGMAX. This scheme seems to stabilize LAMBDA and speed convergence immediately after deflation.

An eigenvalue has been found when the last off diagonal element falls below EPS (see Section 6). Its value is placed in T[I] and the corresponding weight W[I] is computed from formula (2.5). This convergence test and the test for matrix splitting are done following label INSPECT. Only the lower block (from K to M) needs to be transformed by the *Q-R* equation given in formulas (3.3). These equations have been rearranged to reduce the number of computer operations as suggested by W. Kahan in a report by Corneil [2].

TABLE

A Comparison of the Abscissas and Weights of the Gauss-Laguerre Quadrature Rule with $\alpha = -0.75$ *and* $N = 10$

	Analytic Recurrence Relationship + QR	Moment Matrix + QR	Concus _et_ _al_ [1].
		ABSCISSAS	
1	$2.7666558670\underline{80153}_{10}{-2}$	$2.76665586\underline{2878470}_{10}{-2}$	$2.76665586707972_{10}{-2}$
2	$4.547844226059\underline{642}_{10}{-1}$	$4.5478442\underline{19368714}_{10}{-1}$	$4.5478442260594 9_{10}{-1}$
3	$1.382425761158\underline{619}$	$1.3824257\underline{59256314}$	1.382425761158599
4	$2.833980012092\underline{734}$	$2.8339800\underline{08561162}$	2.833980012092697
5	$4.850971448764\underline{968}$	$4.85097144\underline{3442301}$	4.850971448764914
6	$7.500010942642\underline{896}$	$7.5000109\underline{35563904}$	7.500010942642825
7	$1.0888408023834\underline{46}_{10}{+1}$	$1.08884080\underline{1516104}_{10}{+1}$	$1.0888408023834404_{10}{+1}$
8	$1.51994780442376\underline{5}_{10}{+1}$	$1.51994780\underline{3419274}_{10}{+1}$	$1.5199478044237603_{10}{+1}$
9	$2.07892146210701\underline{8}_{10}{+1}$	$2.07892146\underline{0989977}_{10}{+1}$	$2.0789214621070107_{10}{+1}$
10	$2.8573060164922\underline{23}_{10}{+1}$	$2.85730601\underline{5294401}_{10}{+1}$	$2.8573060164922106_{10}{+1}$
		WEIGHTS	
1	$2.566765557790\underline{853}$	$2.56676555\underline{6932285}$	2.566765557790772
2	$7.733479703443\underline{168}_{10}{-1}$	$7.73347970\underline{6154000}_{10}{-1}$	$7.73347970344341_{10}{-1}$
3	$2.331328349732\underline{182}_{10}{-1}$	$2.3313283\underline{53678223}_{10}{-1}$	$2.33132834973219_{10}{-1}$
4	$4.643674708956\underline{677}_{10}{-2}$	$4.6436747\underline{24992909}_{10}{-2}$	$4.64367470895670_{10}{-2}$
5	$5.54912350203625\underline{6}_{10}\ 3$	$5.5491235\underline{31829512}_{10}{-3}$	$5.54912350203625_{10}{-3}$
6	$3.656466626776\underline{441}_{10}{-4}$	$3.6564666\underline{53186007}_{10}{-4}$	$3.65646662677638_{10}{-4}$
7	$1.186879857102\underline{525}_{10}{-5}$	$1.1868798\underline{67642139}_{10}{-5}$	$1.18687985710245_{10}{-5}$
8	$1.584410942056\underline{844}_{10}{-7}$	$1.5844109\underline{58350144}_{10}{-7}$	$1.58441094205678_{10}{-7}$
9	$6.193266726796\underline{867}_{10}{-10}$	$6.1932667\underline{97518338}_{10}{-10}$	$6.19326672679684_{10}{-10}$
10	$3.037759926517\underline{691}_{10}{-13}$	$3.0377599\underline{63698451}_{10}{-13}$	$3.03775992651750_{10}{-13}$

(Underlined figures are those which disagree with Concus _et_ _al_ [1].)

6. Test Program and Results. The procedures in the microfiche section have been extensively tested in Burroughs B5500 Algol and IBM OS/360 Algol. There are two machine dependent items which must be mentioned. First, the constant used to define the "relative zero tolerance" EPS in procedure GAUSSQUADRULE is dependent on the length of the fraction part of the floating-point number representation ($= 8^{-13}$ for the 13 octal digit fraction on the B5500, and $= 16^{-14}$ for a 14 hexadecimal digit long-precision fraction on the IBM 360). Second, the moment matrix M defined in Section 4 usually becomes increasingly ill conditioned with increasing N. Thus the round-off errors generated during Cholesky decomposition in GENORTHOPOLY cause an ill conditioned M to appear no longer positive definite and the procedure fails on taking the square root of a negative number.

The procedure GAUSSQUADRULE proves to be quite stable and when the recursion coefficients are known or supplied by the procedure CLASSICORTHOPOLY it loses only several digits off from full-word accuracy even for $N = 50$. Procedure GENORTHOPOLY usually failed to produce the recursion coefficients from the moments when N was about 20 for the IBM 360.

The driver program given in the microfiche section of this issue is designed to compare the two methods of generating the quadrature rules—from the moments or the recursion coefficients. N can be increased until GENORTHOPOLY fails. Numerical results may be checked against tables for Gauss-Legendre quadrature in [11] and Gauss-Laguerre quadrature in [1]. In the Table, we compare the abscissas and weights of the Gauss-Laguerre quadrature rule with $\alpha = -0.50$ and $N = 10$ computed by: (A) the analytic recurrence relationship and the Q-R algorithm; (B) the moment matrix and the Q-R algorithm; (C) Concus et al. [1]. The calculations for (A) and (B) were performed on the IBM 360.

Acknowledgment. The authors are very pleased to acknowledge Professor Walter Gautschi for his many helpful comments and for his derivation given in Section 4.

Computer Science Department
Stanford University
Stanford, California 94305

Stanford Linear Accelerator Center
Stanford University
Stanford, California 94305

1. P. Concus, D. Cassatt, G. Jaehnig & E. Melby, "Tables for the evaluation of $\int_0^\infty x^\beta e^{-x} f(x)\, dx$ by Gauss-Laguerre quadrature," *Math. Comp.*, v. 17, 1963, pp. 245–256. MR **28** #1757.

2. D. Corneil, *Eigenvalues and Orthogonal Eigenvectors of Real Symmetric Matrices*, Institute of Computer Science Report, Univ. of Toronto, 1965.

3. P. Davis & I. Polonsky, "Numerical interpolation, differentiation, and integration," in *Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables*, Nat. Bur. Standards Appl. Math. Ser., 55, Superintendent of Documents, U. S. Government Printing Office, Washington, D. C., 1964; 3rd printing with corrections, 1965. MR **29** #4914; MR **31** #1400.

4. P. J. Davis & P. Rabinowitz, *Numerical Integration*, Blaisdell, Waltham, Mass., 1967. MR **35** #2482.

5. J. G. F. Francis, "The Q-R transformation: A unitary analogue to the L-R transformation. I, II," *Comput. J.*, v. 4, 1961/62, pp. 265–271; 332–345. MR **23** #B3143; MR **25** #744.

6. W. Gautschi, "Construction of Gauss-Christoffel quadrature formulas," *Math. Comp.*, v. 22, 1968, pp. 251–270.

7. G. Golub & J. Welsch, *Calculation of Gauss Quadrature Rules*, Technical Report No. CS 81, Stanford University, 1967.

8. U. Hochstrasser, "Orthogonal polynomials," in *Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables*, Nat. Bur. Standards Appl. Math. Ser., 55, Superintendent of Documents, U. S. Government Printing Office, Washington, D. C., 1964; 3rd printing with corrections, 1965. MR **29** #4914; MR **31** #1400.

9. I. P. Mysovskih, "On the construction of cubature formulas with the smallest number of nodes," *Dokl. Akad. Nauk. SSSR*, v. 178, 1968, pp. 1252–1254 = *Soviet Math. Dokl.*, v. 9, 1968, pp. 277–280. MR **36** #7328.

10. H. Rutishauser, "On a modification of the Q-D algorithm with Graeffe-type convergence," *Z. Angew. Math. Phys.*, v. 13, 1963, pp. 493–496.

11. A. Stroud & D. Secrest, *Gaussian Quadrature Formulas*, Prentice-Hall, Englewood Cliffs, N. J., 1966. MR **34** #2185.

12. H. Wilf, *Mathematics for the Physical Sciences*, Wiley, New York, 1962.

ALGOL PROCEDURES FOR THE
CALCULATION OF GAUSS QUADRATURE RULES

The ALGOL procedures given here are described in this issue in detail in the paper 'Generation of Gauss quadrature rules" by Gene H. Golub and John H. Welsch. The procedure CLASSICORTHOPOLY generates the coefficients of the normalized three term recurrence relation for various classical orthogonal polynomials and it also yields the zeroth order, moment. From the first 2N+1 moments of a weight function, the procedure GENORTHOPOLY generates the coefficients of the three term recurrence relation for the normalized orthogonal polynomials. Given the coefficients of the three term recurrence relation, the procedure GAUSSQUADRULE computes the abscissas and the weights of the Gaussian type quadrature rule associated with the orthogonal polynomials by the QR method. Finally, a driver program is presented for testing out the procedures described above.

```
procedure CLASSICORTHOPOLY(KIND, ALFA, BETA, N, A, B, MUZERO);
     value KIND, N, ALFA, BETA;
     integer KIND, N;  real ALFA, BETA, MUZERO;
     real array  A, B;
begin comment  This procedure supplies the coefficients  (A, B)  of the
          normalized recurrence relation for various classical orthogonal
          polynomials and the moment  MUZERO  of its weight function. ;
     integer I;  real PI, ABI, A2B2;
     switch SWT := LEGENDRE, CHEBY1, CHEBY2, JACOBI,
        LAGUERRE, HERMITE;
     PI := 3.14159265358979324 ;
     go to SWT[KIND];
LEGENDRE:  MUZERO := 2.0;
     comment P(x) on [-1, +1],  ω(x) = 1.0 ;
     for I := 1 step 1 until N-1 do
        begin A[I] := 0;  B[I] := I/sqrt(4×I↑2-1) end;
     A[N} := 0;  go to RETURN;
CHEBY1:  MUZERO := PI;
     comment  T(x) on [-1, 1],  ω(x) = (1-x↑2)↑(-.5) ;
     for I := 1 step 1 until N-1 do
        begin A[I] := 0;  B[I] := 0.5 end;
     B[1] := sqrt(0.5);  A[N] := 0;
     go to RETURN;
CHEBY2:  MUZERO := PI/2.0;
     comment  U(x) on [-1, 1],  ω(x) = (1-x↑2)↑.5 ;
     for I := 1 step 1 until N-1 do
        begin A[I] := 0;  B[I] := 0.5 end;
     A[N] := 0;  go to RETURN;
```

```
JACOBI:  MUZERO := 2↑(ALFA+BETA+1)×GAMMA(ALFA+1)×GAMMA(BETA+1)
              /GAMMA(ALFA+BETA+2) ;
      comment  P(ALFA,BETA)(x) on [-1, +1],
           ω(x) = (1-x↑ALFA×(1+x)↑BETA,  ALFA AND BETA > -1 ;
      ABI := 2+ALFA+BETA;  A[1] := (BETA-ALFA)/ABI;
      B[1] := sqrt(4×(1+ALFA)×(1+BETA)/((ABI+1)×ABI↑2));
      A2B2 := BETA↑2-ALFA↑2;
      for I := 2 step 1 until N-1 do
         begin  ABI := 2×I+ALFA+BETA;
           A[I] := A2B2/((ABI-2)×ABI);
           B[I] := sqrt(4×I×(I+ALFA)×(I+BETA)×(I+ALFA+BETA)/
                           ((ABI↑2-1)×ABI↑2));
         end;
      ABI := 2×N+ALFA+BETA;
      A[N] := A2B2/((ABI-2)×ABI);
      go to RETURN;
LAGUERRE:  MUZERO := GAMMA(ALFA+1.0);
      comment L(ALFA)(x) on [0, INFINITY),
           ω(x) = EXP(-x)×x↑ALFA,  ALFA > -1 ;
      for I := 1 step 1 until N-1 do
         begin A[I] := 2×I-1+ALFA;
           B[I] := -sqrt(I×(I+ALFA));
         end;
      A[N] := 2×N-1+ALFA;  go to RETURN;
```

```
HERMITE:  MUZERO := sqrt(PI);
     comment  H(x) on (-INFINITY,+INFINITY), ω(x) = EXP(-x↑2) ;
     for I := 1 step 1 until N-1 do
        begin  A[I] := 0;  B[I] := sqrt(I/2) end;
     A[N] := 0;
RETURN:  end  CLASSICORTHOPOLY ;
```

```
procedure GENORTHOPOLY(N, MU, A, B);
     value N;  integer N;
     real array  MU, A, B;
begin comment  Given the  2N+1  moments  (MU)  of the weight function,
          generate the recursion coefficients  (A, B)  of the normalized
          orthogonal polynomials. ;
     real array  R[0:N+1,0:N+1];  real SUM ;
     integer I, J, K;
comment  Place the Cholesky decomposition of the moment matrix in  R[ ];
     for I := 1 step 1 until N+1 do
     for J := I step 1 until N+1 do
     begin  SUM := MU[I+J-2];
        for K := I-1 step -1 until 1 do
          SUM := SUM - R[K,I]×R[K,J] ;
        R[I,J] := (if I = J then sqrt(SUM) else SUM/R[I,I]);
     end;
comment  Compute the recursion coefficients from the decomposition  R[ ];
     R[0,0] := 1.0;  R[0,1] := 0;
     A[N] := R[N,N+1]/R[N,N]-R[N-1,N]/R[N-1,N-1] ;
     for J := N-1 step -1 until 1 do
     begin  B[J] := R[J+1,J+1]/R[J,J];
        A[J] := R[J,J+1]/R[J,J]-R[J-1,J]/R[J-1,J-1];
     end;
end  GENORTHOPOLY ;
```

```
procedure GAUSSQUADRULE(N, A, B, C, MUZERO, SYMM, T, W);
     value N, MUZERO, SYMM;
     integer N;  real MUZERO;  boolean SYMM;
     real array A, B, C, T, W ;
begin comment  Given the coefficients  (A, B, C)  of the three term
          recurrence relation:  P(K) = (A(K)X+B(K))P(K-1)-C(K)P(K-2),
          this procedure computes the abscissas  T  and the weights  W
          of the Gaussian type quadrature rule associated with the ortho-
          gonal polynomial by  QR  type iteration with origin shifting.;
     integer I, J, K, M, M1;
     real NORM, EPS, CT, ST, F, Q, AA, AI, AJ, A2, EIGMAX,
        LAMBDA, LAMBD1, LAMBD2, RHO, R, DET, BI, BJ, B2, WJ, CJ;
     boolean EX;
     real procedure MAX(X,Y);  value X, Y;  real X, Y;
          MAX := if X ≥ Y then X else Y;
     if SYMM then go to SETUP;
comment  Symmetrize the matrix, if required. ;
          for I := 1 step 1 until N-1 do
          begin  AI := A[I];  A[I] := B[I]/AI;
               B[I] := sqrt(C[I+1]/(AI×A[I+1])) ;
          end;
          A[N] := -B[N]/A[N];
comment  Find the maximum row sum norm and initialize  W[ ] ;
SETUP:  B[0] := 0;  NORM := 0;
     for I := 1 step 1 until N-1 do
     begin  NORM := MAX(NORM, abs(B[I-1]) +abs(A[I]) +abs(B[I])) ;
          W[I] := 0;
     end;
```

```
    NORM := MAX(NORM, abs(A[N])+abs(B[N-1]));
    EPS := NORM×16.0↑(-14); comment Relative zero tolerance ;
    W[I] := 1.0;  W[N] := 0;  M := N;
    LAMBDA := LAMBD1 := LAMBD2 := RHO := NORM;
comment Look for convergence of lower diagonal element ;
INSPECT:  if M = 0 then go to SORT else I := K := M1 := M-1;
    if abs(B[M1]) ≤ EPS then
        begin T[M] := A[M];  W[M] := MUZERO×W[M]↑2;
            RHO := (if LAMBD1 < LAMBD2 then LAMBD1 else LAMBD2);
            M := M1;  go to INSPECT ;
        end;
comment Small off diagonal element means matrix can be split ;
    for I := I-1 while abs(B[I]) > EPS do K := I;
comment Find eigenvalues of lower  2X2  and select accelerating shift ;
    B2 := B[M1]↑2;  DET := sqrt((A[M1]-A[M])↑2+4.0×B2) ;
    AA := A[M1]+A[M] ;
    LAMBD2 := 0.5×(if AA ≥ 0 then AA+DET else AA-DET);
    LAMBD1 := (A[M1]×A[M]-B2)/LAMBD2;
    EIGMAX := MAX(LAMBD1, LAMBD2);
    if abs(EIGMAX-RHO) ≤ 0.125×abs(EIGMAX) then
        LAMBDA := RHO := EIGMAX else RHO := EIGMAX;
comment Transform block from  K  to  M ;
    CJ := B[K];  B[K-1] := A[K]-LAMBDA;
    for J := K step 1 until M1 do
    begin  R := sqrt(CJ↑2+B[J-1]↑2) ;
        ST := CJ/R;  CT := B[J-1]/R;  AJ := A[J];
        B[J-1] := R;  CJ := B[J+1]×ST;
        B[J+1] := -B[J+1]×CT;
        F := AJ×CT + B[J]×ST;
        Q := B[J]×CT + A[J+1]×ST;
        A[J] := F×CT + Q×ST;  B[J] := F×ST - Q×CT;
        WJ := W[J];
        A[J+1] := AJ+A[J+1]-A[J];
        W[J] := WJ×CT+W[J+1]×ST;  W[J+1] := WJ×ST-W[J+1]×CT;
    end;
    B[K-1] := 0;  go to INSPECT;
comment  Arrange abscissas in ascending order ;
SORT:  for M := N step -1 until 2 do
        begin  EX := false;
            for I := 2 step 1 until M do
                if T[I-1] > T[I] then
                    begin
                        WJ := T[I-1];  T[I-1] := T[I];  T[I] := WJ;
                        WJ := W[I-1]:  W[I-1] := W[I];  W[I] := WJ;
                        EX := true
                    end;
            if ¬ EX then go to RETURN;
        end;
RETURN:  end GAUSSQUADRULE ;
```

```
begin
comment Driver program for GAUSSQUADRULE;
    real array A, C, T, W[1:10], MU[0:20], B[0:10];
    real MUZERO; integer I, N;
        N := 10;
comment Legendre polynomials. ;
        outstring (1, 'Legendre Quadrature.');
        CLASSICORTHOPOLY(1, 0, 0, N, A, B, MUZERO);
        GAUSSQUADRULE(N, A, B, C, MUZERO, true, T, W);
    outstring (1, 'abscissas:'); outarray (1, T);
    outstring (1, 'weights:'); outarray(1, W);
    for I := 0 step 1 until 2×N do MU[I] := 0;
    for I := 0 step 2 until 2×N do MU[I] := 2.0/(I+1);
        GENORTHOPOLY (N, MU, A, B);
        MUZERO := MU[0];
        GAUSSQUADRULE (N, A, B, C, MUZERO, true T, W);
    outstring (1, 'abscissas:'); outarray (1, T);
    outstring (1, 'weights:'); outarray (1, W);
comment Laguerre polynomials. ;
        outstring (1, 'Laguerre Quadrature.  alpha = -0.5');
        CLASSICORTHOPOLY (5, -0.5, 0, N, A, B, MUZERO);
        GAUSSQUADRULE (N, A, B, C, MUZERO, true T, W);
    outstring (1, 'abscissas:'); outarray (1, T);
    outstring (1, 'weights:'); outarray (1, W);
    MU[0] ;= MUZERO := 1.772453850905516; comment gamma (0.5);
    for I := 1 step 1 until 2×N 'do
```

```
        MU[I] := (I-0.5)×MU[I-1];
        GENORTHOPOLY (N, MU, A, B);
        GAUSSQUADRULE (N, A, B, C, MUZERO, true, T, W);
    outstring (1, 'abscissas:'); outarray (1, T);
    outstring (1, 'weights:'); outarray (1, W);
end;
```

24

MATRICES, MOMENTS, AND QUADRATURE (WITH GÉRARD MEURANT)

PROCEEDINGS OF THE 15TH DUNDEE CONFERENCE, JUNE-JULY 1993,
D. F. Griffiths and G. A. Watson (eds.), Longman Scientific & Technical (1994).

Vingt fois sur le métier remettez votre ouvrage: polissez-le sans cesse et le repolissez. Ajoutez quelquefois, et souvent effacez.
Nicolas Boileau, Art poétique, 1674

Gene has been interested for a long time in moments, quadrature and their relations to orthogonal polynomials, and as the French quote above says, he pursued his goal for many years always improving his results. He realized in the 1970s that bounds can be obtained for matrix moments using quadrature formulas. More generally, if one considers a quadratic form $u^T f(A)u$ where u is a given vector and f is a smooth function of the symmetric matrix A, it can be written as a Riemann–Stieltjes integral. Then using Gauss-type quadrature formulas provides bounds for the value $u^T f(A)u$. In his paper with Welsch, Gene has shown how to compute the nodes and weights of such a quadrature rule by using the eigenvalues and eigenvectors of the tridiagonal matrix given by the three-term recurrence of the orthogonal polynomials corresponding to the measure of the Riemann–Stieltjes integral. These polynomials are obtained by running the Lanczos algorithm for A with $u/\|u\|$ as an initial vector. This gives a very elegant framework associating several interesting areas of mathematics: linear algebra, integration and quadrature rules, orthogonal polynomials. Gene has a deep understanding of the interconnections of these mathematical areas and he likes to share his ideas with his collaborators.

In our paper we were interested in the more general problem of a bilinear form $u^T f(A)v$ where v is another given vector. Even though one can obtain results using the quadratic form case by writing $u^T f(A)v = 1/4[(u+v)^T f(A)(u+v) - (u-v)^T f(A)(u-v)]$ we suggested using other techniques. It turns out that estimates can also be obtained by using the nonsymmetric Lanczos algorithm (even with a symmetric matrix!) as well as the block Lanczos algorithm. Another aim in considering the block Lanczos algorithm was to try to give a precise mathematical content to what was done at that time by physicists working in solid state physics (Nex, *Computer Physics Communications*, **53**, pp 141–146 (1989)). Fortunately, we succeeded and we were able to prove some theorems about quadrature for integrals with matrix measures.

At that time I was amazed to see how well these techniques were working numerically when I did some computations with several functions f, like the inverse, the exponential or the square root. One can obtain very good bounds with only a few iterations of one of these Lanczos algorithms. Moreover, doing one or two Lanczos iterations "by hand" gives analytical bounds which in some cases are already interesting. This is what we did in our paper for the matrix inverse.

Unfortunately, we realized afterwards that what is obtained with the Gauss quadrature formula is equivalent to a result that was given in the Hestenes and Stiefel CG paper. It is a pity that this result was almost unnoticed for many years because it would have led to good lower bounds of the norm of the error in CG without almost any additional cost. Nevertheless, using Gene's quadrature approach is interesting because it also allows us to compute upper bounds of the error if we have an estimate of the smallest eigenvalue of A.

Gene and some of his collaborators have applied the MMQ techniques to other practical problems like computing estimates of the determinant of large sparse symmetric matrices or computing parameters in regularization methods for ill-posed problems. It is very satisfactory when a paper has such a nice and fruitful outcome.

As usual, it was very nice to be, again, a Golub number 1 in the writing of this paper. Gene is always full of ideas and good suggestions. Working on this paper I learnt a lot of things on moments and quadrature and this was at the root of my interest in estimation of errors, particularly in the Lanczos and CG algorithms, a topic on which I am still working.

In fact, these techniques have so many applications that the game is not over yet. In the middle of 2005, I was with Gene at CERFACS in Toulouse (France) as members of the committee for a PhD thesis defense. We were quietly chatting about these topics. Then Gene went to the men's room and he came back very excited and enthusiastic saying "I just had a bright idea: we should write a book about MMQ". This is now ongoing and when it is finished it is going to be the next episode of our collaboration and friendship.

Gérard Meurant
Bruyères-le-Châtel, France

Matrices, moments and quadrature *

Gene H. Golub[†]
Computer Science Department, Stanford University
Stanford CA 94305, USA
golub@sccm.stanford.edu

Gérard Meurant
CEA, Centre d'Etudes de Limeil-Valenton,
94195 Villeneuve St Georges cedex, France
meurant@etca.fr

September 29, 1994

[*] Presented as the first A.R. Mitchell lecture in Dundee, Scotland, July 1993. We dedicate this paper to Ron Mitchell who has given intellectual leadership and generous support to all.

[†] The work of this author was supported in part by the National Science Foundation under Grant NSF CCR-8821078

Abstract In this paper we study methods to obtain bounds or approximations of elements of a matrix $f(A)$ where A is a symmetric positive definite matrix and f is a smooth function. These methods are based on the use of quadrature rules and the Lanczos algorithm for diagonal elements and the block Lanczos or the non–symmetric Lanczos algorithms for the non diagonal elements. We give some theoretical results on the behavior of these methods based on results for orthogonal polynomials as well as analytical bounds and numerical experiments on a set of matrices for several functions f.

1 Definition of the problem

Let A be a real symmetric positive definite matrix of order n. We want to find upper and lower bounds (or approximations, if bounds are not available) for the entries of a function of a matrix. We shall examine analytical expressions as well as numerical iterative methods which produce good approximations in a few steps. This problem leads us to consider

$$u^T f(A)v, \tag{1.1}$$

where u and v are given vectors and f is some smooth (possibly C^∞) function on a given interval of the real line.

As an example, if $f(x) = \frac{1}{x}$ and $u^T = e_i^T = (0, \ldots, 0, 1, 0, \ldots, 0)$, the non zero element being in the i–th position and $v = e_j$, we will obtain bounds on the elements of the inverse A^{-1}.

We shall also consider

$$W^T f(A) W,$$

where W is an $n \times m$ matrix. For specificity, we shall most often consider $m = 2$.

Some of the techniques presented in this paper have been used (without any mathematical justification) to solve problems in solid state physics, particularly to compute elements of the resolvant of a Hamiltonian modeling the interaction of atoms in a solid, see [12], [14], [15]. In these studies the function f is the inverse of its argument.

Analytic bounds for elements of inverses of matrices using different techniques have been recently obtained in [17].

The outline of the paper is as follows. Section 2 considers the problem of characterizing the elements of a function of a matrix. The theory is developed in Section 3 and Section 4 deals with the construction of the orthogonal polynomials that are needed to obtain a numerical method for computing bounds. The Lanczos, non–symmetric Lanczos and block Lanczos methods used for the computation of the polynomials are presented there. Applications to the computation of elements of the inverse of a matrix are described in Section 5 where very simple iterative algorithms are given to compute bounds. Some numerical examples are given in Section 6, for different matrices and functions f.

2 Elements of a function of a matrix

Since $A = A^T$, we write A as

$$A = Q \Lambda Q^T,$$

where Q is the orthonormal matrix whose columns are the normalized eigenvectors of A and Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_i which we order as

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n.$$

By definition, we have

$$f(A) = Qf(\Lambda)Q^T.$$

Therefore,

$$\begin{aligned} u^T f(A)v &= u^T Q f(\Lambda)Q^T v \\ &= \alpha^T f(\Lambda)\beta, \\ &= \sum_{i=1}^{n} f(\lambda_i)\alpha_i\beta_i. \end{aligned}$$

This last sum can be considered as a Riemann–Stieltjes integral

$$I[f] = u^T f(A)v = \int_a^b f(\lambda)\ d\alpha(\lambda), \tag{2.1}$$

where the measure α is piecewise constant and defined by

$$\alpha(\lambda) = \begin{cases} 0 & \text{if } \lambda < a = \lambda_1 \\ \sum_{j=1}^{i} \alpha_j\beta_j & \text{if } \lambda_i \leq \lambda < \lambda_{i+1} \\ \sum_{j=1}^{n} \alpha_j\beta_j & \text{if } b = \lambda_n \leq \lambda \end{cases}$$

When $u = v$, we note that α is an increasing positive function.

The block generalization is obtained in the following way. Let W be an $n \times 2$ matrix, $W = (w_1\ w_2)$, then

$$W^T f(A)W = W^T Q f(\Lambda)Q^T W = \alpha f(\Lambda)\alpha^T,$$

where, of course, α is a $2 \times n$ matrix such that

$$\alpha = (\alpha_1 \ldots \alpha_n),$$

and α_i is a vector with two components. With these notations, we have

$$W^T f(A)W = \sum_{i=1}^{n} f(\lambda_i)\alpha_i\alpha_i^T.$$

This can be written as a matrix Riemann–Stieltjes integral

$$I_B[f] = W^T f(A)W = \int_a^b f(\lambda)\ d\alpha(\lambda).$$

$I_B[f]$ is a 2×2 matrix where the entries of the (matrix) measure α are piecewise constant and defined by

$$\alpha(\lambda) = \sum_{k=1}^{l} \alpha_k\alpha_k^T, \quad \lambda_l \leq \lambda < \lambda_{l+1}.$$

In this paper, we are looking for methods to obtain upper and lower bounds L and U for $I[f]$ and $I_B[f]$,

$$\begin{aligned} L &\leq I[f] \leq U \\ L &\leq I_B[f] \leq U. \end{aligned}$$

In the next section, we review and describe some basic results from Gauss quadrature theory as this plays a fundamental role in estimating the integrals and computing bounds.

3 Bounds on matrix functions as integrals

A way to obtain bounds for the Stieltjes integrals is to use Gauss, Gauss–Radau and Gauss–Lobatto quadrature formulas, see [3],[8],[9]. For 1.1, the general formula we will use is

$$\int_a^b f(\lambda)\, d\alpha(\lambda) = \sum_{j=1}^{N} w_j f(t_j) + \sum_{k=1}^{M} v_k f(z_k) + R[f], \tag{3.1}$$

where the weights $[w_j]_{j=1}^N$, $[v_k]_{k=1}^M$ and the nodes $[t_j]_{j=1}^N$ are unknowns and the nodes $[z_k]_{k=1}^M$ are prescribed, see [4],[5],[6],[7].

3.1 The case $u = v$

When $u = v$, the measure is a positive increasing function and it is known (see for instance [18]) that

$$R[f] = \frac{f^{(2N+M)}(\eta)}{(2N+M)!} \int_a^b \prod_{k=1}^{M} (\lambda - z_k) \left[\prod_{j=1}^{N} (\lambda - t_j) \right]^2 d\alpha(\lambda), \quad a < \eta < b. \tag{3.2}$$

If $M = 0$, this leads to the Gauss rule with no prescribed nodes. If $M = 1$ and $z_1 = a$ or $z_1 = b$ we have the Gauss–Radau formula. If $M = 2$ and $z_1 = a, z_2 = b$, this is the Gauss–Lobatto formula.

Let us recall briefly how the nodes and weights are obtained in the Gauss, Gauss–Radau and Gauss–Lobatto rules. For the measure α, it is possible to define a sequence of polynomials $p_0(\lambda), p_1(\lambda), \ldots$ that are orthonormal with respect to α:

$$\int_a^b p_i(\lambda) p_j(\lambda)\, d\alpha(\lambda) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

and p_k is of exact degree k. Moreover, the roots of p_k are distinct, real and lie in the interval $[a, b]$. We will see how to compute these polynomials in the next Section.

This set of orthonormal polynomials satisfies a three term recurrence relationship (see [20]):

$$\gamma_j p_j(\lambda) = (\lambda - \omega_j) p_{j-1}(\lambda) - \gamma_{j-1} p_{j-2}(\lambda), \quad j = 1, 2, \ldots, N \tag{3.3}$$
$$p_{-1}(\lambda) \equiv 0, \quad p_0(\lambda) \equiv 1,$$

if $\int d\alpha = 1$.

In matrix form, this can be written as

$$\lambda p(\lambda) = J_N p(\lambda) + \gamma_N p_N(\lambda) e_N,$$

where

$$p(\lambda)^T = [p_0(\lambda)\ p_1(\lambda) \cdots\ p_{N-1}(\lambda)],$$
$$e_N^T = (0\ 0\ \cdots\ 0\ 1),$$
$$J_N = \begin{pmatrix} \omega_1 & \gamma_1 & & & \\ \gamma_1 & \omega_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{N-2} & \omega_{N-1} & \gamma_{N-1} \\ & & & \gamma_{N-1} & \omega_N \end{pmatrix}. \tag{3.4}$$

The eigenvalues of J_N (which are the zeroes of p_N) are the nodes of the Gauss quadrature rule (i. e. $M = 0$). The weights are the squares of the first elements of the normalized eigenvectors of J_N, cf. [7]. We note that all the eigenvalues of J_N are real and simple.

For the Gauss quadrature rule (renaming the weights and nodes w_j^G and t_j^G) we have

$$\int_a^b f(\lambda)\ d\alpha(\lambda) = \sum_{j=1}^{N} w_j^G f(t_j^G) + R_G[f],$$

with

$$R_G[f] = \frac{f^{(2N)}(\eta)}{(2N)!} \int_a^b \left[\prod_{j=1}^{N}(\lambda - t_j^G)\right]^2 d\alpha(\lambda),$$

and the next theorem follows.

Theorem 3.1 *Suppose $u = v$ in 2.1 and f is such that $f^{(2n)}(\xi) > 0$, $\forall n$, $\forall \xi$, $a < \xi < b$, and let*

$$L_G[f] = \sum_{j=1}^{N} w_j^G f(t_j^G).$$

Then, $\forall N$, $\exists \eta \in [a, b]$ such that

$$L_G[f] \leq I[f],$$
$$I[f] - L_G[f] = \frac{f^{(2N)}(\eta)}{(2N)!}.$$

Proof: See [18]. The main idea of the proof is to use a Hermite interpolatory polynomial of degree $2N - 1$ on the N nodes which allows us to express the remainder as an integral of the difference between the function and its interpolatory polynomial and to apply the mean value theorem (as the measure is positive and increasing). As we know the sign of the remainder, we easily obtain bounds.

To obtain the Gauss–Radau rule ($M = 1$ in 3.1–3.2), we should extend the matrix J_N in 3.4 in such a way that it has one prescribed eigenvalue, see [8].

Assume $z_1 = a$, we wish to construct p_{N+1} such that $p_{N+1}(a) = 0$. From the recurrence relation 3.3, we have

$$0 = \gamma_{N+1}p_{N+1}(a) = (a - \omega_{N+1})p_N(a) - \gamma_N p_{N-1}(a).$$

This gives

$$\omega_{N+1} = a - \gamma_N \frac{p_{N-1}(a)}{p_N(a)}.$$

We have also

$$(J_N - aI)p(a) = -\gamma_N p_N(a)e_N.$$

Let us denote $\delta(a) = [\delta_1(a), \cdots, \delta_N(a)]^T$ with

$$\delta_l(a) = -\gamma_N \frac{p_{l-1}(a)}{p_N(a)} \quad l = 1, \ldots, N.$$

This gives $\omega_{N+1} = a + \delta_N(a)$ and

$$(J_N - aI)\delta(a) = \gamma_N^2 e_N. \tag{3.5}$$

From these relations we have the solution of the problem as: 1) we generate γ_N by the Lanczos process (see Section 4 for the definition), 2) we solve the tridiagonal system 3.5 for $\delta(a)$ and 3) we compute ω_{N+1}. Then the tridiagonal matrix $\hat{J}_{N+1}$ defined as

$$\hat{J}_{N+1} = \begin{pmatrix} J_N & \gamma_N e_N \\ \gamma_N e_N^T & \omega_{N+1} \end{pmatrix},$$

will have a as an eigenvalue and gives the weights and the nodes of the corresponding quadrature rule. Therefore, the recipe is to compute as for the Gauss quadrature rule and then to modify the last step to obtain the prescribed node.

For Gauss–Radau the remainder R_{GR} is

$$R_{GR}[f] = \frac{f^{(2N+1)}(\eta)}{(2N+1)!} \int_a^b (\lambda - z_1) \left[\prod_{j=1}^N (\lambda - t_j) \right]^2 d\alpha(\lambda).$$

Again, this is proved by constructing an interpolatory polynomial for the function and its derivative on the t_js and for the function on z_1.

Therefore, if we know the sign of the derivatives of f, we can bound the remainder. This is stated in the following theorem.

Theorem 3.2 *Suppose* $u = v$ *and* f *is such that* $f^{(2n+1)}(\xi) < 0$, $\forall n, \forall \xi, a < \xi < b$. *Let* U_{GR} *be defined as*

$$U_{GR}[f] = \sum_{j=1}^N w_j^a f(t_j^a) + v_1^a f(a),$$

w_j^a, v_1^a, t_j^a *being the weights and nodes computed with* $z_1 = a$ *and let* L_{GR} *be defined as*

$$L_{GR}[f] = \sum_{j=1}^N w_j^b f(t_j^b) + v_1^b f(b),$$

w_j^b, v_1^b, t_j^b *being the weights and nodes computed with* $z_1 = b$. *Then,* $\forall N$ *we have*

$$L_{GR}[f] \leq I[f] \leq U_{GR}[f],$$

and

$$I[f] - U_{GR}[f] = \frac{f^{(2N+1)}(\eta)}{(2N+1)!} \int_a^b (\lambda - a) \left[\prod_{j=1}^{N} (\lambda - t_j^a) \right]^2 d\alpha(\lambda),$$

$$I[f] - L_{GR}[f] = \frac{f^{(2N+1)}(\eta)}{(2N+1)!} \int_a^b (\lambda - b) \left[\prod_{j=1}^{N} (\lambda - t_j^b) \right]^2 d\alpha(\lambda).$$

Proof : With our hypothesis the sign of the remainder is easily obtained. It is negative if we choose $z_1 = a$, positive if we choose $z_1 = b$.

Remarks :

i) if the sign of the f derivatives is positive, the bounds are reversed.

ii) it is enough to suppose that there exists an n_0 such that $f^{(2n_0+1)}(\eta) < 0$ but, then $N = n_0$ is fixed.

Now, consider the Gauss–Lobatto rule ($M = 2$ in 3.1–3.2), with $z_1 = a$ and $z_2 = b$ as prescribed nodes. Again, we should modify the matrix of the Gauss quadrature rule, see [8]. Here, we would like to have

$$p_{N+1}(a) = p_{N+1}(b) = 0.$$

Using the recurrence relation 3.3 for the polynomials, this leads to a linear system of order 2 for the unknowns ω_{N+1} and γ_N:

$$\begin{pmatrix} p_N(a) & p_{N-1}(a) \\ p_N(b) & p_{N-1}(b) \end{pmatrix} \begin{pmatrix} \omega_{N+1} \\ \gamma_N \end{pmatrix} = \begin{pmatrix} a\, p_N(a) \\ b\, p_N(b) \end{pmatrix}. \tag{3.6}$$

Let δ and μ be defined as vectors with components

$$\delta_l = -\frac{p_{l-1}(a)}{\gamma_N p_N(a)}, \quad \mu_l = -\frac{p_{l-1}(b)}{\gamma_N p_N(b)},$$

then

$$(J_N - aI)\delta = e_N, \quad (J_N - bI)\mu = e_N,$$

and the linear system 3.6 can be written

$$\begin{pmatrix} 1 & -\delta_N \\ 1 & -\mu_N \end{pmatrix} \begin{pmatrix} \omega_{N+1} \\ \gamma_N^2 \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix},$$

giving the unknowns that we need. The tridiagonal matrix $\hat{J}_{N+1}$ is then defined as in the Gauss–Radau rule.

Having computed the nodes and weights, we have

$$\int_a^b f(\lambda) d\alpha(\lambda) = \sum_{j=1}^{N} w_j^{GL} f(t_j^{GL}) + v_1 f(a) + v_2 f(b) + R_{GL}[f],$$

where

$$R_{GL}[f] = \frac{f^{(2N+2)}(\eta)}{(2N+2)!} \int_a^b (\lambda - a)(\lambda - b) \left[\prod_{j=1}^{N} (\lambda - t_j) \right]^2 d\alpha(\lambda).$$

Then, we have the following obvious result.

Theorem 3.3 *Suppose $u = v$ and f is such that $f^{(2n)}(\eta) > 0$, $\forall n$, $\forall \eta, a < \eta < b$ and let*

$$U_{GL}[f] = \sum_{j=1}^{N} w_j^{GL} f(t_j^{GL}) + v_1 f(a) + v_2 f(b).$$

Then, $\forall N$

$$I[f] \leq U_{GL}[f],$$

$$I[f] - U_{GL}[f] = \frac{f^{(2N+2)}(\eta)}{(2N+2)!} \int_a^b (\lambda - a)(\lambda - b) \left[\prod_{j=1}^{N} (\lambda - t_j^{GL}) \right]^2 d\alpha(\lambda).$$

We remark that we need not always compute the eigenvalues and eigenvectors of the tridiagonal matrix. Let Y_N be the matrix of the eigenvectors of J_N (or $\hat{J}_N$) whose columns we denote by y_i and T_N be the diagonal matrix of the eigenvalues t_i which give the nodes of the Gauss quadrature rule. It is well known that the weights w_i are given by (cf. [21])

$$\frac{1}{w_i} = \sum_{l=0}^{N-1} p_l^2(t_i).$$

It can be easily shown that

$$w_i = \left(\frac{y_i^1}{p_0(t_i)} \right)^2,$$

where y_i^1 is the first component of y_i.

But, since $p_0(\lambda) \equiv 1$, we have,

$$w_i = (y_i^1)^2 = (e_1^T y_i)^2.$$

Theorem 3.4

$$\sum_{l=1}^{N} w_l f(t_l) = e_1^T f(J_N) e_1.$$

Proof:

$$\begin{aligned} \sum_{l=1}^{N} w_l f(t_l) &= \sum_{l=1}^{N} e_1^T y_l f(t_l) y_l^T e_1 \\ &= e_1^T \left(\sum_{l=1}^{N} y_l f(t_l) y_l^T \right) e_1 \\ &= e_1^T Y_N f(T_N) Y_N^T e_1 \\ &= e_1^T f(J_N) e_1. \end{aligned}$$

The same statement is true for the Gauss–Radau and Gauss–Lobatto rules. Therefore, in some cases where $f(J_N)$ (or the equivalent) is easily computable (for instance, if $f(\lambda) = 1/\lambda$, see Section 5), we do not need to compute the eigenvalues and eigenvectors of J_N.

3.2 The case $u \neq v$

We have seen that the measure in 2.1 is piecewise constant and defined by

$$\alpha(\lambda) = \sum_{k=1}^{l} \alpha_k \delta_k, \quad \lambda_l \leq \lambda < \lambda_{l+1}.$$

For variable signed weight functions, see [19]. We will see later that for our application, u and v can always be chosen such that $\alpha_k \delta_k \geq 0$. Therefore, in this case α will be a positive increasing function.

In the next Section, we will show that there exists two sequences of polynomials p and q such that

$$\begin{aligned} \gamma_j p_j(\lambda) &= (\lambda - \omega_j) p_{j-1}(\lambda) - \beta_{j-1} p_{j-2}(\lambda), \quad p_{-1}(\lambda) \equiv 0, \quad p_0(\lambda) \equiv 1, \\ \beta_j q_j(\lambda) &= (\lambda - \omega_j) q_{j-1}(\lambda) - \gamma_{j-1} q_{j-2}(\lambda), \quad q_{-1}(\lambda) \equiv 0, \quad q_0(\lambda) \equiv 1. \end{aligned}$$

Let

$$p(\lambda)^T = [p_0(\lambda)\ p_1(\lambda) \cdots\ p_{N-1}(\lambda)],$$
$$q(\lambda)^T = [q_0(\lambda)\ q_1(\lambda) \cdots\ q_{N-1}(\lambda)],$$

and

$$J_N = \begin{pmatrix} \omega_1 & \gamma_1 & & & \\ \beta_1 & \omega_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{N-2} & \omega_{N-1} & \gamma_{N-1} \\ & & & \beta_{N-1} & \omega_N \end{pmatrix}.$$

Then, we can write

$$\begin{aligned} \lambda p(\lambda) &= J_N p(\lambda) + \gamma_N p_N(\lambda) e_N, \\ \lambda q(\lambda) &= J_N^T q(\lambda) + \beta_N q_N(\lambda) e_N. \end{aligned}$$

Theorem 3.5

$$p_j(\lambda) = \frac{\beta_j \cdots \beta_1}{\gamma_j \cdots \gamma_1} q_j(\lambda).$$

Proof: The theorem is proved by induction. We have

$$\gamma_1 p_1(\lambda) = \lambda - \omega_1,$$

$$\beta_1 q_1(\lambda) = \lambda - \omega_1,$$

therefore

$$p_1(\lambda) = \frac{\beta_1}{\gamma_1} q_1(\lambda).$$

Now, suppose that

$$p_{j-1}(\lambda) = \frac{\beta_{j-1} \cdots \beta_1}{\gamma_{j-1} \cdots \gamma_1} q_{j-1}(\lambda).$$

We have

$$\begin{aligned}\gamma_j p_j(\lambda) &= (\lambda-\omega_j)p_{j-1}(\lambda)-\beta_{j-1}p_{j-2}(\lambda)\\ &= (\lambda-\omega_j)\frac{\beta_{j-1}\cdots\beta_1}{\gamma_{j-1}\cdots\gamma_1}q_{j-1}(\lambda)-\beta_{j-1}\frac{\beta_{j-2}\cdots\beta_1}{\gamma_{j-2}\cdots\gamma_1}q_{j-2}(\lambda).\end{aligned}$$

Multiplying by $\frac{\gamma_{j-1}\cdots\gamma_1}{\beta_{j-1}\cdots\beta_1}$ we obtain the result.

Hence q_N is a multiple of p_N and the polynomials have the same roots which are also the common eigenvalues of J_N and J_N^T.

We will see that it is possible to choose γ_j and β_j such that

$$\gamma_j = \pm\beta_j,$$

with, for instance, $\gamma_j \geq 0$. Then, we have

$$p_j(\lambda) = \pm q_j(\lambda).$$

We define the quadrature rule as

$$\int_a^b f(\lambda)\, d\alpha(\lambda) = \sum_{j=1}^N f(\lambda_j)s_jt_j + error, \tag{3.7}$$

where λ_j is an eigenvalue of J_N, s_j is the first component of the eigenvector u_j of J_N corresponding to λ_j and t_j is the first component of the eigenvector v_j of J_N^T corresponding to the same eigenvalue, normalized such that $v_j^T u_j = 1$.

We have the following results:

Proposition 3.1 *Suppose that $\gamma_j\beta_j \neq 0$, then the (non–symmetric) Gauss quadrature rule 3.7 is exact for polynomials of degree less than or equal to $N-1$.*

Proof:

The function f can be written as

$$f(\lambda) = \sum_{k=0}^{N-1} c_k p_k(\lambda),$$

and because of the orthonormality properties

$$\int_a^b f(\lambda)\, d\alpha(\lambda) = c_0.$$

For the quadrature rule, we have

$$\begin{aligned}\sum_{j=1}^N f(\lambda_j)s_jt_jq_l(\lambda_j) &= \sum_{j=1}^N\sum_{k=0}^{N-1} c_kp_k(\lambda_j)s_jt_jq_l(\lambda_j)\\ &= \sum_{k=0}^{N-1} c_k\sum_{j=1}^N p_k(\lambda_j)s_jt_jq_l(\lambda_j).\end{aligned}$$

But $p_k(\lambda_j)s_j$ and $q_l(\lambda_j)t_j$ are respectively the components of the eigenvectors of J_N and J_N^T corresponding to λ_j. Therefore they are orthonormal with the normalization that we chose. Hence,

$$\sum_{j=1}^{N} f(\lambda_j)s_jt_jq_l(\lambda_j) = c_l,$$

and consequently

$$\sum_{j=1}^{N} f(\lambda_j)s_jt_j = c_0,$$

which proves the result.

Now, as in [14], we extend the result to polynomials of higher degree.

Theorem 3.6 *Suppose that $\gamma_j\beta_j \neq 0$, then the (non–symmetric) Gauss quadrature rule 3.7 is exact for polynomials of degree less than or equal to $2N-1$.*

Proof:

Suppose f is a polynomial of degree $2N-1$. Then, f can be written as

$$f(\lambda) = p_N(\lambda)s(\lambda) + r(\lambda),$$

where s and r are polynomials of degree less or equal to $N-1$. Then,

$$\int_a^b f(\lambda)\ d\alpha(\lambda) = \int_a^b p_N(\lambda)s(\lambda)\ d\alpha(\lambda) + \int_a^b r(\lambda)\ d\alpha(\lambda) = \int_a^b r(\lambda)\ d\alpha(\lambda),$$

since p_N is orthogonal to any polynomial of degree less or equal to $N-1$ because of the orthogonality property of the p and q's.

For the quadrature rule, we have

$$\sum_{j=1}^{N} p_N(\lambda_j)s(\lambda_j)s_jt_j + \sum_{j=1}^{N} r(\lambda_j)s_jt_j.$$

But, as λ_j is an eigenvalue of J_N, it is a root of p_N and

$$\sum_{j=1}^{N} p_N(\lambda_j)s(\lambda_j)s_jt_j = 0.$$

As the quadrature rule has been proven to be exact for polynomials of degree less than $N-1$,

$$\int_a^b r(\lambda)\ d\alpha(\lambda) = \sum_{j=1}^{N} r(\lambda_j)s_jt_j,$$

which proves the Theorem.

We will see in the next Section how to obtain bounds on the integral 2.1.

Now, we extend the Gauss–Radau and Gauss–Lobatto rules to the non–symmetric case. This is almost identical (up to technical details) to the symmetric case.

For Gauss–Radau, assume that the prescribed node is a, then, we would like to have $p_{N+1}(a) = q_{N+1}(a) = 0$. This gives

$$(a - \omega_{N+1})p_N(a) - \beta_N p_{N-1}(a) = 0.$$

If we denote $\delta(a) = [\delta_1(a), \ldots, \delta_N(a)]^T$, with

$$\delta_l(a) = -\beta_N \frac{p_{l-1}(a)}{p_N(a)},$$

we have

$$\omega_{N+1} = a + \delta_N(a),$$

where

$$(J_N - aI)\delta(a) = \gamma_N \beta_N e_N.$$

Therefore, the algorithm is essentially the same as previously discussed.

For Gauss–Lobatto, the algorithm is also almost the same as for the symmetric case. We would like to compute p_{N+1} and q_{N+1} such that

$$p_{N+1}(a) = p_{N+1}(b) = 0, \quad q_{N+1}(a) = q_{N+1}(b) = 0.$$

This leads to solving the linear system

$$\begin{pmatrix} p_N(a) & p_{N-1}(a) \\ p_N(b) & p_{N-1}(b) \end{pmatrix} \begin{pmatrix} \omega_{N+1} \\ \beta_N \end{pmatrix} = \begin{pmatrix} ap_N(a) \\ bp_N(b) \end{pmatrix}.$$

The linear system for the q's whose solution is $(\omega_{N+1}, \gamma_N)^T$ can be shown to have the same solution for ω_{N+1} and $\gamma_N = \pm\beta_N$ depending on the signs relations between the p's and the q's.

Let $\delta(a)$ and $\mu(b)$ be the solutions of

$$(J_N - aI)\delta(a) = e_N, \quad (J_N - bI)\mu(b) = e_N.$$

Then, we have

$$\begin{pmatrix} 1 & -\delta(a)_N \\ 1 & -\mu(b)_N \end{pmatrix} \begin{pmatrix} \omega_{N+1} \\ \gamma_N \beta_N \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}.$$

When we have the solution of this system, we choose $\gamma_N = \pm\beta_N$ and $\gamma_N \geq 0$.

The question of establishing bounds on the integral will be studied in the next Section.

As for the case $u = v$, we do not always need compute the eigenvalues and eigenvectors of J_N but only the $(1,1)$ element of $f(J_N)$.

3.3 The block case

Now, we consider the block case. The problem is to find a quadrature rule. The integral $\int_a^b f(\lambda)d\alpha(\lambda)$ is a 2×2 symmetric matrix. The most general quadrature formula is of the form

$$\int_a^b f(\lambda)d\alpha(\lambda) = \sum_{j=1}^{N} W_j f(T_j) W_j + error,$$

where W_j and T_j are symmetric 2×2 matrices. In this sum, we have $6N$ unknowns. This quadrature rule can be simplified, since

$$T_j = Q_j \Lambda_j Q_j^T,$$

where Q_j is the orthonormal matrix of the eigenvectors, and Λ_j, the diagonal matrix of the eigenvalues of T_j. This gives

$$\sum_{j=1}^{N} W_j Q_j f(\Lambda_j) Q_j^T W_j.$$

But $W_j Q_j f(\Lambda_j) Q_j^T W_j$ can be written as

$$f(\lambda_1) z_1 z_1^T + f(\lambda_2) z_2 z_2^T,$$

where the vector z_i is 2×1. Therefore, the quadrature rule can be written as

$$\sum_{j=1}^{2N} f(t_j) w_j w_j^T,$$

where t_j is a scalar and w_j is a vector with 2 components. In this quadrature rule, there are also $6N$ unknowns.

In the next Section, we will show that there exists orthogonal matrix polynomials such that

$$\lambda p_{j-1}(\lambda) = p_j(\lambda)\Gamma_j + p_{j-1}(\lambda)\Omega_j + p_{j-2}(\lambda)\Gamma_{j-1}^T,$$
$$p_0(\lambda) \equiv I_2, \quad p_{-1}(\lambda) \equiv 0.$$

This can be written as

$$\lambda[p_0(\lambda), \dots, p_{N-1}(\lambda)] = [p_0(\lambda), \dots, p_{N-1}(\lambda)] J_N + [0, \dots, 0, p_N(\lambda)\Gamma_N],$$

where

$$J_N = \begin{pmatrix} \Omega_1 & \Gamma_1^T & & & \\ \Gamma_1 & \Omega_2 & \Gamma_2^T & & \\ & \ddots & \ddots & \ddots & \\ & & \Gamma_{N-2} & \Omega_{N-1} & \Gamma_{N-1}^T \\ & & & \Gamma_{N-1} & \Omega_N \end{pmatrix}, \tag{3.8}$$

is a block tridiagonal matrix of order $2N$ and a banded matrix whose half bandwidth is 2 (we have at most 5 non zero elements in a row).

If we denote $P(\lambda) = [p_0(\lambda), \dots, p_{N-1}(\lambda)]^T$, we have as J_N is symmetric

$$J_N P(\lambda) = \lambda P(\lambda) - [0, \dots, 0, p_N(\lambda)\Gamma_N]^T.$$

We note that if λ is an eigenvalue, say λ_r, of J_N and if we choose $u = u_r$ to be a two element vector whose components are the first two components of an eigenvector corresponding to λ_r, then $P(\lambda_r)u$ is this eigenvector (because of the relations that are satisfied) and if

Γ_N is non singular, $p_N^T(\lambda_r)u = 0$. The difference with the scalar case is that although the eigenvalues are real, it might be that they are of multiplicity greater than 1 (although this is unlikely except in the case of the Gauss-Radau and Gauss-Lobatto rule where this condition is enforced).

We define the quadrature rule as:

$$\int_a^b f(\lambda)\, d\alpha(\lambda) = \sum_{i=1}^{2N} f(\lambda_i) u_i u_i^T + error, \tag{3.9}$$

where $2N$ is the order of J_N, the eigenvalues λ_i are those of J_N and u_i is the vector consisting of the two first components of the corresponding eigenvector, normalized as before. In fact, if there are multiple eigenvalues, the quadrature rule should be written as follows. Let $\mu_i, i = 1, \ldots, l$ be the set of distinct eigenvalues and q_i their multiplicities. The quadrature rule is then

$$\sum_{i=1}^{l} \left(\sum_{j=1}^{q_i} (w_i^j)(w_i^j)^T \right) f(\mu_i). \tag{3.10}$$

We will show in the next Section that the Gauss quadrature rule is exact for polynomials of degree $2N-1$ and how to obtain estimates of the error.

We extend the process described for scalar polynomials to the matrix analog of the Gauss–Radau quadrature rule. Let a be an extreme eigenvalue of A. We would like a to be a double eigenvalue of J_{N+1}. We have

$$J_{N+1}P(a) = aP(a) - [0, \ldots, 0, p_{N+1}(a)\Gamma_{N+1}]^T.$$

Then, we need to require $p_{N+1}(a) \equiv 0$. From the recurrence relation this translates into

$$ap_N(a) - p_N(a)\Omega_{N+1} - p_{N-1}(a)\Gamma_N^T = 0.$$

Therefore, if $p_N(a)$ is non singular, we have

$$\Omega_{N+1} = aI_2 - p_N(a)^{-1}p_{N-1}(a)\Gamma_N^T.$$

We must compute the right hand side. This can be done by noting that

$$J_N \begin{pmatrix} p_0(a)^T \\ \vdots \\ p_{N-1}(a)^T \end{pmatrix} = a \begin{pmatrix} p_0(a)^T \\ \vdots \\ p_{N-1}(a)^T \end{pmatrix} - \begin{pmatrix} 0 \\ \vdots \\ \Gamma_N^T p_N(a)^T \end{pmatrix}.$$

Multiplying on the right by $p_N(a)^{-T}$, we get the matrix equation

$$(J_N - aI) \begin{pmatrix} -p_0(a)^T p_N(a)^{-T} \\ \vdots \\ -p_{N-1}(a)^T p_N(a)^{-T} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ \Gamma_N^T \end{pmatrix}.$$

Thus, we solve

$$(J_N - aI) \begin{pmatrix} \delta_0(a) \\ \vdots \\ \delta_{N-1}(a) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ \Gamma_N^T \end{pmatrix},$$

and hence

$$\Omega_{N+1} = aI_2 + \delta_{N-1}(a)^T\Gamma_N^T.$$

The generalization of Gauss–Lobatto to the block case is a little more tricky. We would like to have a and b as double eigenvalues of the matrix J_{N+1}. This leads to satisfying the following two matrix equations

$$ap_N(a) - p_N(a)\Omega_{N+1} - p_{N-1}(a)\Gamma_N^T = 0$$

$$bp_N(b) - p_N(b)\Omega_{N+1} - p_{N-1}(b)\Gamma_N^T = 0$$

This can be written as

$$\begin{pmatrix} I_2 & p_N^{-1}(a)p_{N-1}(a) \\ I_2 & p_N^{-1}(b)p_{N-1}(b) \end{pmatrix} \begin{pmatrix} \Omega_{N+1} \\ \Gamma_N^T \end{pmatrix} = \begin{pmatrix} aI_2 \\ bI_2 \end{pmatrix}.$$

We now consider the problem of computing (or avoid computing) $p_N^{-1}(\lambda)p_{N-1}(\lambda)$. Let $\delta(\lambda)$ be the solution of

$$(J_N - \lambda I)\delta(\lambda) = (0 \ldots 0 \; I_2)^T.$$

Then, as before

$$\delta_{N-1}(\lambda) = -p_{N-1}(\lambda)^T p_N(\lambda)^{-T}\Gamma_N^{-T}.$$

We can easily show that $\delta_{N-1}(\lambda)$ is symmetric. We consider solving a 2 × 2 block linear system

$$\begin{pmatrix} I & X \\ I & Y \end{pmatrix} \begin{pmatrix} U \\ V \end{pmatrix} = \begin{pmatrix} aI \\ bI \end{pmatrix}.$$

Consider the block factorization

$$\begin{pmatrix} I & X \\ I & Y \end{pmatrix} = \begin{pmatrix} I & 0 \\ I & W \end{pmatrix} \begin{pmatrix} I & X \\ 0 & Z \end{pmatrix},$$

thus $WZ = Y - X$.

The solution of the system

$$\begin{pmatrix} I & 0 \\ I & W \end{pmatrix} \begin{pmatrix} U_1 \\ V_1 \end{pmatrix} = \begin{pmatrix} aI \\ bI \end{pmatrix},$$

gives

$$WV_1 = (b - a)I.$$

The next step is

$$\begin{pmatrix} I & X \\ 0 & Z \end{pmatrix} \begin{pmatrix} U \\ V \end{pmatrix} = \begin{pmatrix} U_1 \\ V_1 \end{pmatrix},$$

and we get

$$ZV = V_1 = W^{-1}(b - a)I$$

or

$$(WZ)V = (b - a)I.$$

Therefore

$$V = (b - a)(Y - X)^{-1}.$$

Hence, we have

$$Y - X = p_N^{-1}(b)p_{N-1}(b) - p_N^{-1}(a)p_{N-1}(a) = \Gamma_N(\delta_{N-1}(a) - \delta_{N-1}(b)).$$

This means that

$$\Gamma_N^T = (b-a)(\delta_{N-1}(a) - \delta_{N-1}(b))^{-1}\Gamma_N^{-1},$$

or

$$\Gamma_N^T \Gamma_N = (b-a)(\delta_{N-1}(a) - \delta_{N-1}(b))^{-1}$$

Then, Γ_N is given as a Cholesky decomposition of the right hand side matrix. The right hand side is positive definite because $\delta_{N-1}(a)$ is a diagonal block of the inverse of $(J_N - aI)^{-1}$ which is positive definite because the eigenvalues of J_N are larger that a and $-\delta_{N-1}(b)$ is the negative of a diagonal block of $(J_N - bI)^{-1}$ which is positive definite because the eigenvalues of J_N are smaller that b.

From Γ_N, we can compute Ω_{N+1}:

$$\Omega_{N+1} = aI_2 + \Gamma_N \delta_{N-1}(a)\Gamma_N^T.$$

As for the scalar case, it is not always needed to compute the weights and the nodes for the quadrature rules.

Theorem 3.7 *We have*

$$\sum_{i=1}^{2N} f(\lambda_i)u_i u_i^T = e^T f(J_N)e,$$

where $e^T = (I_2 \; 0 \ldots 0)$.

Proof:

The quadrature rule is

$$\sum_{i=1}^{2N} u_i f(\lambda_i) u_i^T.$$

If y_i are the eigenvectors of J_N then $u_i = e^T y_i$ and

$$\begin{aligned}\sum_{i=1}^{2N} u_i f(\lambda_i) u_i^T &= \sum_{i=1}^{2N} e^T y_i f(\lambda_i) y_i^T e \\ &= e^T \left(\sum_{i=1}^{2N} y_i f(\lambda_i) y_i^T \right) e \\ &= e^T Y_N f(T_N) Y_N^T e \\ &= e^T f(J_N) e\end{aligned}$$

where Y_N is the matrix of the eigenvectors and T_N the diagonal matrix of the eigenvalues of J_N.

Note that bounds for non diagonal elements can also be obtained by considering $e_i^T f(A) e_i$, $e_j^T f(A) e_j$ and $\frac{1}{2}(e_i + e_j)^T f(A)(e_i + e_j)$.

4 Construction of the orthogonal polynomials

In this section we consider the problem of computing the orthonormal polynomials or equivalently the tridiagonal matrices that we need. A very natural and elegant way to do this is to use Lanczos algorithms.

4.1 The case $u = v$

When $u = v$, we use the classical Lanczos algorithm.

Let $x_{-1} = 0$ and x_0 be given such that $\|x_0\| = 1$. The Lanczos algorithm is defined by the following relations,

$$\begin{aligned} \gamma_j x_j = r_j &= (A - \omega_j I)x_{j-1} - \gamma_{j-1}x_{j-2}, \quad j = 1, \dots \\ \omega_j &= x_{j-1}^T A x_{j-1}, \\ \gamma_j &= \|r_j\|. \end{aligned}$$

The sequence $\{x_j\}_{j=0}^l$ is an orthonormal basis of the Krylov space

$$\mathrm{span}\{x_0, Ax_0, \dots, A^l x_0\}.$$

Proposition 4.1 *The vector x_j is given by*

$$x_j = p_j(A)x_0,$$

where p_j is a polynomial of degree j defined by the three term recurrence (identical to 3.3)

$$\gamma_j p_j(\lambda) = (\lambda - \omega_j)p_{j-1}j(\lambda) - \gamma_{j-1}p_{j-2}(\lambda), \quad p_{-1}(\lambda) \equiv 0, \quad p_0(\lambda) \equiv 1.$$

Proof:

$$\gamma_1 x_1 = (A - \omega_1 I)x_0,$$

is a first order polynomial in A. Therefore, the Proposition is easily obtained by induction.

Theorem 4.1 *If $x_0 = u$, we have*

$$x_k^T x_l = \int_a^b p_k(\lambda)p_l(\lambda)d\alpha(\lambda).$$

Proof: As the x_j's are orthonormal, we have

$$\begin{aligned} x_k^T x_l &= x_0^T P_k(A)^T P_l(A)x_0 \\ &= x_0^T Q P_k(\Lambda) Q^T Q P_l(\Lambda) Q^T x_0 \\ &= x_0^T Q P_k(\Lambda) P_l(\Lambda) Q^T x_0 \\ &= \sum_{j=1}^n p_k(\lambda_j)p_l(\lambda_j)\hat{x}_j^2, \end{aligned}$$

where $\hat{x} = Q^T x_0$.

Therefore, the p_j's are the orthonormal polynomials related to α that we were referring to in 3.3.

4.2 The case $u \neq v$

We apply the non–symmetric Lanczos algorithm to a symmetric matrix A.

Let $x_{-1} = \hat{x}_{-1} = 0$, and $x_0, \hat{x}_0$ be given with $x_0 \neq \hat{x}_0$ and $x_0^T \hat{x}_0 = 1$. Then we define the iterates for $j = 1, \ldots$ by

$$\gamma_j x_j = r_j = (A - \omega_j I)x_{j-1} - \beta_{j-1}x_{j-2}, \tag{4.1}$$

$$\beta_j \hat{x}_j = \hat{r}_j = (A - \omega_j I)\hat{x}_{j-1} - \gamma_{j-1}\hat{x}_{j-2}, \tag{4.2}$$

$$\omega_j = \hat{x}_{j-1}^T A x_{j-1},$$

$$\gamma_j \beta_j = \hat{r}_j^T r_j.$$

This algorithm generates two sequences of mutually orthogonal vectors as we have

$$x_l^T \hat{x}_k = \delta_{kl}.$$

We have basically the same properties as for the Lanczos algorithm.

Proposition 4.2

$$x_j = p_j(A)x_0, \quad \hat{x}_j = q_j(A)\hat{x}_0,$$

where p_j and q_j are polynomials of degree j defined by the three term recurrences

$$\gamma_j p_j(\lambda) = (\lambda - \omega_j)p_{j-1}(\lambda) - \beta_{j-1}p_{j-2}(\lambda), \quad p_{-1}(\lambda) \equiv 0, \quad p_0(\lambda) \equiv 1,$$

$$\beta_j q_j(\lambda) = (\lambda - \omega_j)q_{j-1}(\lambda) - \gamma_{j-1}q_{j-2}(\lambda), \quad q_{-1}(\lambda) \equiv 0, \quad q_0(\lambda) \equiv 1.$$

Proof: The Proposition is easily obtained by induction.

Theorem 4.2 *If $x_0 = u$ and $\hat{x}_0 = v$, then*

$$x_k^T \hat{x}_l = \int_a^b p_k(\lambda)q_l(\lambda)d\alpha(\lambda) = \delta_{kl}.$$

Proof: As the x_j's and $\hat{x}_j$'s are orthonormal the proof is identical to the proof of Theorem 4.1.

We have seen in the previous Section the relationship between the p and q's. The polynomials q are multiples of the polynomials p.

In this particular application of the non–symmetric Lanczos algorithm it is possible to choose γ_j and β_j such that

$$\gamma_j = \pm\beta_j = \pm\sqrt{|\hat{r}_j^T r_j|},$$

with, for instance, $\gamma_j \geq 0$ and $\beta_j = \mathrm{sgn}(\hat{r}_j^T r_j)\gamma_j$. Then, we have

$$p_j(\lambda) = \pm q_j(\lambda).$$

The main difference with the case of symmetric Lanczos is that this algorithm may break down, e.g. we can have $\gamma_j \beta_j = 0$ at some step.

We would like to use the non–symmetric Lanczos algorithm with $x_0 = e_i$ and $\hat{x}_0 = e_j$ to get estimates of $f(A)_{i,j}$. Unfortunately, this is not possible as this implies $x_0^T \hat{x}_0 = 0$. A way to get around this problem is to set $x_0 = e_i/\delta$ and $\hat{x}_0 = \delta e_i + e_j$. This will give an estimate of $f(A)_{i,j}/\delta + f(A)_{i,i}$ and we can use the bounds we get for the diagonal elements (using for instance symmetric Lanczos) to obtain bounds for the non diagonal entry. An added adantage is that we are able to choose δ so that $\gamma_j \beta_j > 0$ and therefore $p_j(\lambda) = q_j(\lambda)$. This can be done by starting with $\delta = 1$ and restarting the algorithm with a larger value of δ as soon as we find a value of j for which $\gamma_j \beta_j \leq 0$.

Regarding expressions for the remainder, we can do exactly the same as for symmetric Lanczos. We can write

$$R(f) = \frac{f^{(2N)}(\eta)}{(2N)!} \int_a^b p_N(\lambda)^2 \, d\alpha(\lambda).$$

However, we know that $p_N(\lambda) = \pm q_N(\lambda)$ and

$$\int_a^b p_N(\lambda) q_N(\lambda) \, d\alpha(\lambda) = 1.$$

This shows that the sign of the integral in the remainder can be computed using the algorithm and we have the following result.

Theorem 4.3 *Suppose f is such that $f^{(2n)}(\eta) > 0$, $\forall n$, $\forall \eta$, $a < \eta < b$. Then, the quadrature rule 3.7 gives a lower bound if*

$$\prod_{j=1}^{N} sgn(\hat{r}_j^T r_j) = 1,$$

and an upper bound otherwise. In both cases, we have

$$|R(f)| = \frac{f^{(2N)}(\eta)}{(2N)!}.$$

For Gauss–Radau and Gauss–Lobatto we cannot do the same thing. Bounds can be obtained if we choose the initial vectors (e.g. δ) such that the measure is positive and increasing. In this case we are in exactly the same framework as for the symmetric case and the same results are obtained. Note however that it is not easy to make this choice a priori. Some examples are given in Section 6. A way to proceed is to start with $\delta = 1$ and to restart the algorithm 4.2 with a larger value of δ whenever we have $\gamma_j \beta_j \leq 0$.

4.3 The block case

Now, we consider the block Lanczos algorithm, see [10],[16]. Let X_0 be an $n \times 2$ given matrix, such that $X_0^T X_0 = I_2$ (chosen as U defined before). Let $X_{-1} = 0$ be an $n \times 2$ matrix. Then

$$\Omega_j = X_{j-1}^T A X_{j-1},$$

$$R_j = AX_{j-1} - X_{j-1}\Omega_j - X_{j-2}\Gamma_{j-1}^T,$$
$$X_j\Gamma_j = R_j$$

The last step is the QR decomposition of R_j such that X_j is $n \times 2$ with $X_j^TX_j = I_2$ and Γ_j is 2×2. The matrix Ω_j is 2×2 and Γ_j is upper triangular.

It may happen that R_j is rank deficient and in that case Γ_j is singular. The solution of this problem is given in [10]. One of the columns of X_j can be chosen arbitrarily. To complete the algorithm, we choose this column to be orthogonal with the previous block vectors X_k. We can for instance choose another vector (randomly) and orthogonalize it against the previous ones.

This algorithm generates a sequence such that

$$X_j^TX_i = \delta_{ij}I_2,$$

where I_2 is the 2×2 identity matrix.

Proposition 4.3

$$X_i = \sum_{k=0}^{i} A^kX_0C_k^{(i)},$$

where $C_k^{(i)}$ *are* 2×2 *matrices.*

Proof: The proof is given by induction.

We define a matrix polynomial $p_i(\lambda)$, a 2×2 matrix, as

$$p_i(\lambda) = \sum_{k=0}^{i} \lambda^kC_k^{(i)}.$$

Thus, we have the following result.

Theorem 4.4

$$X_i^TX_j = \int_a^b p_i(\lambda)^T d\alpha(\lambda) p_j(\lambda) = \delta_{ij}I_2.$$

Proof:

Using the orthogonality of the X_is, we can write

$$\begin{aligned}
\delta_{ij}I_2 = X_i^TX_j &= \left(\sum_{k=0}^{i}(C_k^{(i)})^TX_0^TA^k\right)\left(\sum_{l=0}^{j}A^lX_0C_l^{(j)}\right)\\
&= \sum_{k,l}(C_k^{(i)})^TX_0^TQ\Lambda^{k+l}Q^TX_0C_l^{(j)}\\
&= \sum_{k,l}(C_k^{(i)})^T\alpha\Lambda^{k+l}\alpha^TC_l^{(j)}\\
&= \sum_{k,l}(C_k^{(i)})^T\left(\sum_{m=1}^{n}\lambda_m^{k+l}\alpha_m\alpha_m^T\right)C_l^{(j)}\\
&= \sum_{m=1}^{n}\left(\sum_k \lambda_m^k(C_k^{(i)})^T\right)\alpha_m\alpha_m^T\left(\sum_l \lambda_m^lC_l^{(j)}\right).
\end{aligned}$$

The p_js can be considered as matrix orthogonal polynomials for the (matrix) measure α. To compute the polynomials, we need to show that the following recurrence relation holds.

Theorem 4.5 *The matrix valued polynomials p_j satisfy*

$$p_j(\lambda)\Gamma_j = \lambda p_{j-1}(\lambda) - p_{j-1}(\lambda)\Omega_j - p_{j-2}(\lambda)\Gamma_{j-1}^T,$$

$$p_{-1}(\lambda) \equiv 0, \quad p_0(\lambda) \equiv I_2,$$

where λ is a scalar.

Proof: From the previous definition, it is easily shown by induction that p_j can be generated by the given (matrix) recursion.

As we have seen before this can be written as

$$\lambda[p_0(\lambda), \ldots, p_{N-1}(\lambda)] = [p_0(\lambda), \ldots, p_{N-1}(\lambda)]J_N + [0, \ldots, 0, p_N(\lambda)\Gamma_N],$$

and as $P(\lambda) = [p_0(\lambda), \ldots, p_{N-1}(\lambda)]^T$,

$$J_N P(\lambda) = \lambda P(\lambda) - [0, \ldots, 0, p_N(\lambda)\Gamma_N]^T,$$

with J_N defined by 3.8.

Most of the following results on the properties of the matrix polynomials are derived from [1].

Proposition 4.4 *The eigenvalues of J_N are the zeroes of $det[p_N(\lambda)]$.*

Proof: Let μ be a zero of $det[p_N(\lambda)]$. As the rows of $p_N(\mu)$ are linearly dependent, there exists a vector v with two components such that

$$v^T p_N(\mu) = 0.$$

This implies that

$$\mu[v^T p_0(\mu), \ldots, v^T p_{N-1}(\mu)] = [v^T p_0(\mu), \ldots, v^T p_{N-1}(\mu)]J_N.$$

Therefore μ is an eigenvalue of J_N. $det[p_N(\lambda)]$ is a polynomial of degree $2N$ in λ. Hence, there exists $2N$ zeroes of the determinant and therefore all eigenvalues are zeroes of $det[p_N(\lambda)]$.

Proposition 4.5 *For λ and μ real, we have the analog of the Christoffel–Darboux identity (see [21]) :*

$$p_{N-1}(\mu)\Gamma_N^T p_N^T(\lambda) - p_N(\mu)\Gamma_N p_{N-1}^T(\lambda) = (\lambda - \mu)\sum_{m=0}^{N-1} p_m(\mu)p_m^T(\lambda). \tag{4.3}$$

Proof: From the previous results, we have

$$\Gamma_{j+1}^T p_{j+1}^T(\lambda) = \lambda p_j^T(\lambda) - \Omega_{j+1} p_j^T(\lambda) - \Gamma_j p_{j-1}^T(\lambda),$$

$$p_{j+1}(\mu)\Gamma_{j+1} = \mu p_j(\mu) - p_j(\mu)\Omega_{j+1} - p_{j-1}(\mu)\Gamma_j^T.$$

Multiplying the first relation by $p_j(\mu)$ on the left and the second one by $p_j^T(\lambda)$ on the right gives

$$p_j(\mu)\Gamma_{j+1}^T p_{j+1}^T(\lambda) - p_{j+1}(\mu)\Gamma_{j+1} p_j^T(\lambda) =$$

$$= (\lambda - \mu) p_j(\mu) p_j^T(\lambda) - p_j(\mu)\Gamma_j p_{j-1}^T(\lambda) + p_{j-1}(\mu)\Gamma_j^T p_j^T(\lambda).$$

Summing these equalities, some terms cancel and we get the desired result.

In particular, if we choose $\lambda = \mu$ in 4.3, we have that $p_N(\lambda)\Gamma_N p_{N-1}^T(\lambda)$ is symmetric.

Proposition 4.6

$$\sum_{m=0}^{N-1} u_s^T p_m(\lambda_s) p_m^T(\lambda_r) u_r = \delta_{rs}.$$

Proof: If we set $\lambda = \lambda_s$ and $\mu = \lambda_r$ and multiply the Christoffel–Darboux identity 4.3 on the left by u_s^T and on the right by u_r, we have $p_N^T(\lambda_r)u_r = 0$ and $p_N^T(\lambda_s)u_s = 0$, and we get the result if $\lambda_s \neq \lambda_r$. Let

$$K_{N-1}(\mu, \lambda) = \sum_{m=0}^{N-1} p_m(\mu) p_m^T(\lambda).$$

As $p_0(\lambda) = I_2$, $K_{N-1}(\lambda, \lambda)$ is a symmetric positive definite matrix and therefore defines a scalar product. If λ_r is a multiple eigenvalue, there exist linearly independent eigenvectors that we could orthonormalize. If λ_r is an eigenvalue of multiplicity q_r, there exist q_r linearly independent vectors $v_r^1, \ldots, v_r^{q_r}$ with two components such that the vectors $P(\lambda_r)v_r^j, j = 1, \ldots, q_r$ are the eigenvectors associated with λ_r. We can certainly find a set of vectors $(w_r^1, \ldots, w_r^{q_r})$ spanning the same subspace as $(v_r^1, \ldots, v_r^{q_r})$ and such that

$$(w_r^k)^T K_{N-1}(\lambda_r, \lambda_r) w_r^l = \delta_{kl}.$$

This property is nothing else than the orthogonality relation of the eigenvectors.

Proposition 4.7

$$\sum_{r=1}^{2N} p_i^T(\lambda_r) u_r u_r^T p_l(\lambda_r) = \delta_{il} I_2, \quad i = 0, \ldots, N-1, \quad l = 0, \ldots, N-1.$$

Proof: Note that the eigenvectors of J_N are linearly independent. We take a set of N vectors with two elements : $\{y_0, \ldots, y_{N-1}\}$, and write

$$[y_0^T, \ldots, y_{N-1}^T]^T = \sum_{r=1}^{2N} \alpha_r [u_r^T p_0(\lambda_r), \ldots, u_r^T p_{N-1}(\lambda_r)]^T,$$

or

$$y_l = \sum_{r=1}^{2N} p_l^T(\lambda_r) u_r \alpha_r, \quad l = 0, \ldots, N-1.$$

Multiplying on the left by $u_s^T p_l(\lambda_s)$ and summing :

$$\sum_{l=0}^{N-1} u_s^T p_l(\lambda_s) y_l = \alpha_s, \quad s = 1, \ldots, 2N.$$

Therefore,

$$y_i = \sum_{r=1}^{2N} \sum_{l=0}^{N-1} p_i^T(\lambda_r) u_r u_r^T p_l(\lambda_r) y_l.$$

This gives the desired result.

To prove that the block quadrature rule is exact for polynomials of degree up to $2N-1$, we cannot use the same method as for the scalar case where the given polynomial is factored because of commutativity problems. Therefore, we take another approach that has been used in a different setting in [2]. The following results are taken from [2].

We will consider all the monomials $\lambda^k, k = 1, \ldots, 2N-1$. Let M_k the moment matrix, be defined as

$$M_k = \int_a^b \lambda^k \, d\alpha(\lambda).$$

We write the (matrix) orthonormal polynomials p_j as

$$p_j(\lambda) = \sum_{k=0}^{j} p_k^{(j)} \, \lambda^k,$$

$p_k^{(j)}$ being a matrix of order 2. Then, we have

$$\int_a^b p_j^T(\lambda) \, d\alpha(\lambda) = \sum_{k=0}^{j} (p_k^{(j)})^T \int_a^b \lambda^k \, d\alpha(\lambda) = \sum_{k=0}^{j} (p_k^{(j)})^T M_k,$$

and more generally

$$\int_a^b p_j^T(\lambda) \lambda^q \, d\alpha(\lambda) = \sum_{k=0}^{j} (p_k^{(j)})^T M_{k+q}.$$

We write these equations for $j = N-1$. Note that because of the orthogonality of the polynomials, we have

$$\int_a^b p_{N-1}^T(\lambda) \lambda^q \, d\alpha(\lambda) = 0, \quad q = 0, \ldots, N-2.$$

Let H_N be the block Hankel matrix of order $2N$, defined as

$$H_N = \begin{pmatrix} M_0 & \cdots & M_{N-1} \\ \vdots & & \vdots \\ M_{N-1} & \cdots & M_{2N-2} \end{pmatrix}.$$

Then

$$H_N \begin{pmatrix} p_0^{(N-1)} \\ \vdots \\ p_{N-2}^{(N-1)} \\ p_{N-1}^{(N-1)} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \int_a^b p_{N-1}^T(\lambda)\lambda^{N-1} \, d\alpha(\lambda) \end{pmatrix}.$$

We introduce some additional notation. Let L_N be a block upper triangular matrix of order $2N$,

$$L_N = \begin{pmatrix} p_0^{(0)} & p_0^{(1)} & \cdots & p_0^{(N-1)} \\ 0 & p_1^{(1)} & \cdots & p_1^{(N-1)} \\ & & \ddots & \vdots \\ & & & p_{N-1}^{(N-1)} \end{pmatrix}.$$

Let V_N be a $4N \times 2N$ matrix defined in block form as

$$V_N = \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_l \end{pmatrix},$$

where B_j is a $q_j \times 2N$ matrix,

$$B_j = \begin{pmatrix} I_2 & \mu_j I_2 & \cdots & \mu_j^{N-1} I_2 \\ \vdots & \vdots & \vdots & \vdots \\ I_2 & \mu_j I_2 & \cdots & \mu_j^{N-1} I_2 \end{pmatrix},$$

and the μ_j are the eigenvalues of A.

Let K_i^j be a $2q_i \times 2q_j$ matrix

$$K_i^j = \begin{pmatrix} K_{N-1}(\mu_i, \mu_j) & \cdots & K_{N-1}(\mu_i, \mu_j) \\ \vdots & \vdots & \vdots \\ K_{N-1}(\mu_i, \mu_j) & \cdots & K_{N-1}(\mu_i, \mu_j) \end{pmatrix},$$

and

$$K = \begin{pmatrix} K_1^1 & K_1^2 & \ldots & K_1^l \\ K_2^1 & \ldots & \ldots & K_2^l \\ \vdots & & & \vdots \\ K_l^1 & \ldots & \ldots & K_l^l \end{pmatrix}.$$

Proposition 4.8

$$V_N L_N = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_l \end{pmatrix},$$

where C_j is a $2q_j \times 2N$ matrix,

$$C_j = \begin{pmatrix} p_0(\mu_j) & p_1(\mu_j) & \cdots & p_{N-1}(\mu_j) \\ \vdots & \vdots & \vdots & \vdots \\ p_0(\mu_j) & p_1(\mu_j) & \cdots & p_{N-1}(\mu_j) \end{pmatrix}.$$

Proof: This is straightforward by the definition of the polynomials $p_j(\lambda)$.

Proposition 4.9

$$L_N^T H_N L_N = I.$$

Proof: the generic term of $H_N L_N$ is

$$(H_N L_N)_{ij} = \sum_{s=1}^{j} M_{s+i-2}\, p_{s-1}^{(j-1)}.$$

Therefore the generic term of $L_N^T H_N L_N$ is

$$(L_N^T H_N L_N)_{ij} = \sum_{r=1}^{i} \sum_{s=1}^{j} \int_a^b (p_{r-1}^{(i-1)})^T \lambda^{s+r-2}\; d\alpha(\lambda) p_{s-1}^{(j-1)}.$$

Splitting the power of λ we can easily see that this is

$$(L_N^T H_N L_N)_{ij} = \int_a^b p_{i-1}^T(\lambda)\; d\alpha(\lambda)\; p_{j-1}(\lambda).$$

Therefore because of the orthonormality properties, we have

$$(L_N^T H_N L_N)_{ij} = \begin{cases} I_2 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

This result implies that

$$H_N^{-1} = L_N L_N^T.$$

Proposition 4.10

$$V_N L_N (V_N L_N)^T = K.$$

Proof: this is just using the definition of K_i^j.

Now, we define a $2N \times 4N$ matrix W_N^T whose only non zero components in row i are in position $(i, 2i-1)$ and $(i, 2i)$ and are successively the two components of

$$(w_1^1)^T, \ldots, (w_1^{q_1})^T, (w_2^1)^T, \ldots, (w_l^{q_l})^T$$

Then because of the way the w_i^j are constructed we have

Proposition 4.11

$$W_N^T K W_N = I.$$

Proposition 4.12 *$W_N^T V_N$ is a non singular $2N \times 2N$ matrix.*

Proof:

$$W_N^T V_N H_N^{-1} V_N^T W_N = W_N^T V_N L_N L_N^T V_N^T W_N = W_N^T K W_N = I.$$

This shows that $W_N^T V_N$ is non singular.

Then, we have the main result

Theorem 4.6 *The quadrature rule 3.9 or 3.10 is exact for polynomials of order less than or equal to* $2N-1$.

Proof: From Proposition 4.12, we have

$$H_N^{-1} = (W_N^T V_N)^{-1} (V_N^T W_N)^{-1}.$$

Therefore,

$$H_N = (V_N^T W_N)(W_N^T V_N).$$

By identification of the entries of the two matrices we have,

$$M_k = \sum_{i=1}^{l} \left(\sum_{j=1}^{q_i} (w_i^j)(w_i^j)^T \right) \mu_i^k, \quad k = 0, \ldots, 2N-2.$$

It remains to prove that the quadrature rule is exact for $k = 2N-1$. As we have,

$$H_{N+1} \begin{pmatrix} p_0^{(N)} \\ \vdots \\ p_{N-1}^{(N)} \\ p_N^{(N)} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \int_a^b p_N^T(\lambda) \lambda^N \, d\alpha(\lambda) \end{pmatrix}.$$

Writing the $(N-1)$th block row of this equality, we get

$$M_{2N-1} \, p_N^{(N)} = - \sum_{r=0}^{N-1} M_{N+r-1} \, p_r^{(N)}.$$

We have proved before that

$$M_{N+r-1} = \sum_{i=1}^{l} \left(\sum_{j=1}^{q_i} w_i^j (w_i^j)^T \right) \mu_i^{N+r-1}.$$

By substitution, we get

$$M_{2N-1} \, p_N^{(N)} = - \sum_{r=0}^{N-1} \sum_{i=1}^{l} \sum_{j=1}^{q_i} w_i^j (w_i^j)^T \mu_i^{N+r-1} p_r^{(N)}.$$

We use the fact that

$$(w_i^j)^T \sum_{r=0}^{N-1} \mu_i^r p_r^{(N)} = (w_i^j)^T p_N(\mu_i) - (w_i^j)^T \mu_i^N p_N^{(N)},$$

and

$$(w_i^j)^T p_N(\mu_i) = 0.$$

This shows that

$$M_{2N-1} p_N^{(N)} = \sum_{i=1}^{l} \sum_{j=1}^{q_i} (w_i^j)(w_i^j)^T \mu_i^{2N-1} p_N^{(N)}.$$

As $p_N^{(N)}$ is non singular, we get the result.

To obtain expressions for the remainder, we would like to use a similar approach as for the scalar case. However there are some differences, as the quadrature rule is exact for polynomials of order $2N-1$ and we have $2N$ nodes, we cannot interpolate with an Hermite polynomial and we have to use a Lagrange polynomial. By Theorems 2.1.1.1 and 2.1.4.1 of [18], there exists a polynomial q of degree $2N-1$ such that

$$q(\lambda_j) = f(\lambda_j), \quad j = 1, \ldots, 2N$$

and

$$f(x) - q(x) = \frac{s(x) f^{(2N)}(\xi(x))}{(2N)!},$$

where

$$s(x) = (x - \lambda_1) \cdots (x - \lambda_{2N}).$$

If we can apply the mean value theorem, the remainder $R(f)$ which is a 2 × 2 matrix can be written as

$$R(f) = \frac{f^{(2N)}(\eta)}{(2N)!} \int_a^b s(\lambda)\, d\alpha(\lambda).$$

Unfortunately s does not have a constant sign over the interval $[a, b]$. Therefore this representation formula for the remainder is of little practical use for obtaining bounds with the knowledge of the sign of the entries of the remainder.

It is easy to understand why we cannot directly obtain bounds with this block approach. We must use $W = (e_i \; e_j)$ and the block Lanczos algorithm with $X_0 = W$. For the block Lanczos algorithm we multiply successively A with the Lanczos vectors. If A is sparse, most of the components of these products are 0 for the first few iterates of the algorithm. Therefore, it is likely that at the beginning, the estimates that we will get for the non diagonal entries will be 0. This explains why we cannot directly obtain upper or lower bounds with Gauss, Gauss–Radau or Gauss–Lobatto. A way to avoiding this difficulty is to use $W = (e_i + e_j \; e_j)$ but this cannot be done since $X_0^T X_0 \neq I_2$.

However, we will see in the numerical experiments that the estimates we get are often quite good.

5 Application to the inverse of a matrix

In this Section we consider obtaining analytical bounds for the entries of the inverse of a given matrix and simplifying the algorithms to compute numerical bounds and approximations.

We consider

$$f(\lambda) = \frac{1}{\lambda}, \quad 0 < a < b,$$

and hence

$$f^{(2n+1)}(\lambda) = -(2n+1)!\; \lambda^{-(2n+2)},$$

and

$$f^{(2n)}(\lambda) = (2n)!\; \lambda^{-(2n+1)}.$$

Therefore, the even derivatives are positive on $[a, b]$ and the odd derivatives are negative which implies that we can apply Theorems 3.1, 3.2 and 3.3.

Consider a dense non singular matrix $A = (a_{ij})_{i,j=1,\ldots,m}$. We choose $u = x_0 = e_i$ and we apply the Lanczos algorithm. From results of the first iteration we can obtain analytical results. The first step of the Lanczos algorithm gives us

$$\omega_1 = e_i^T A e_i = a_{ii},$$

$$\gamma_1 x_1 = r_1 = (A - \omega_1 I)e_i.$$

Let s_i be defined by

$$s_i^2 = \sum_{j \neq i} a_{ji}^2,$$

and let

$$d_i^T = (a_{1,i}, \ldots, a_{i-1,i}, 0, a_{i+1,i}, \ldots, a_{m,i}).$$

Then

$$\gamma_1 = s_i, \quad x_1 = \frac{1}{s_i} d_i.$$

From this, we have

$$\omega_2 = \frac{1}{s_i^2} \sum_{k \neq i} \sum_{l \neq i} a_{k,i} a_{k,l} a_{l,i}.$$

From this data, we compute the Gauss rule and get a lower bound on the diagonal element:

$$J_2 = \begin{pmatrix} \omega_1 & \gamma_1 \\ \gamma_1 & \omega_2 \end{pmatrix},$$

$$J_2^{-1} = \frac{1}{\omega_1\omega_2 - \gamma_1^2} \begin{pmatrix} \omega_2 & -\gamma_1 \\ -\gamma_1 & \omega_1 \end{pmatrix}.$$

The lower bound is given by

$$\frac{\omega_2}{\omega_1\omega_2 - \gamma_1^2} = \frac{\sum_{k \neq i} \sum_{l \neq i} a_{k,i} a_{k,l} a_{l,i}}{a_{i,i} \sum_{k \neq i} \sum_{l \neq i} a_{k,i} a_{k,l} a_{l,i} - \left(\sum_{k \neq i} a_{k,i}^2\right)^2}.$$

Note that this bound does not depend on the eigenvalues a and b.

Now, we consider the Gauss–Radau rule. Then,

$$\tilde{J}_2 = \begin{pmatrix} \omega_1 & \gamma_1 \\ \gamma_1 & x \end{pmatrix},$$

the eigenvalues λ are the roots of $(\omega_1 - \lambda)(x - \lambda) - \gamma_1^2 = 0$, which gives the relation

$$x = \lambda + \frac{\gamma_1^2}{\omega_1 - \lambda}.$$

To obtain an upper bound we set $\lambda = a$. The solution is

$$x = x_a = a + \frac{\gamma_1^2}{\omega_1 - a}.$$

For the Gauss–Lobatto rule, we have the same problem except that we want $\tilde{J}_2$ to have a and b as eigenvalues. This leads to solving the following linear system,

$$\begin{pmatrix} \omega_1 - a & -1 \\ \omega_1 - b & -1 \end{pmatrix} \begin{pmatrix} x \\ \gamma_1^2 \end{pmatrix} = \begin{pmatrix} a\omega_1 - a^2 \\ b\omega_1 - b^2 \end{pmatrix}.$$

Solving this system and computing the $(1,1)$ element of the inverse gives

$$\frac{a+b-\omega_1}{ab}.$$

Hence we have the following result.

Theorem 5.1 *We have the following bounds*

$$\frac{\sum_{k\neq i}\sum_{l\neq i} a_{k,i}a_{k,l}a_{l,i}}{a_{i,i}\sum_{k\neq i}\sum_{l\neq i} a_{k,i}a_{k,l}a_{l,i} - \left(\sum_{k\neq i} a_{k,i}^2\right)^2} \leq (A^{-1})_{i,i}$$

$$\frac{a_{i,i} - b + \frac{s_i^2}{b}}{a_{i,i}^2 - a_{i,i}b + s_i^2} \leq (A^{-1})_{i,i} \leq \frac{a_{i,i} - a + \frac{s_i^2}{a}}{a_{i,i}^2 - a_{i,i}a + s_i^2}$$

$$(A^{-1})_{i,i} \leq \frac{a+b-a_{ii}}{ab}.$$

It is not too easy to derive analytical bounds from the block Lanczos algorithm as we have to compute repeated inverses of 2 × 2 matrices.

It is much easier to use the non–symmetric Lanczos method with the Gauss–Radau rule. We are looking at the sum of the (i,i) and (i,j) elements of the inverse. Let

$$t_i = \gamma_1\beta_1 = \sum_{k\neq i} a_{k,i}(a_{k,i} + a_{k,j}) - a_{i,j}(a_{i,j} + a_{i,i}).$$

Then, the computations are essentially the same as for the diagonal case.

Theorem 5.2 *For $(A^{-1})_{i,j} + (A^{-1})_{i,i}$ we have the two following estimates*

$$\frac{a_{i,i} + a_{i,j} - a + \frac{t_i}{a}}{(a_{i,i} + a_{i,j})^2 - a(a_{i,i} + a_{i,j}) + t_i}, \quad \frac{a_{i,i} + a_{i,j} - b + \frac{t_i}{b}}{(a_{i,i} + a_{i,j})^2 - b(a_{i,i} + a_{i,j}) + t_i}.$$

If $t_i \geq 0$, the first expression with a gives an upper bound and the second one with b a lower bound. Then, we have to subtract the bounds for the diagonal term to get bounds on $(A^{-1})_{i,j}$.

The previous results can be compared with those obtained by other methods in [17]. Results can also be obtained for sparse matrices taking into account the sparsity structure.

In the computations using the Lanczos algorithm for the Gauss, Gauss–Radau and Gauss–Lobatto rules, we need to compute the $(1,1)$ element of the inverse of a tridiagonal matrix. This may be done in many different ways, see for instance [13]. Here, we will show that we can compute this element of the inverse incrementally as we go through the Lanczos algorithm and we obtain the estimates for very few additional operations. This is stated in the following theorem where b_j stands for the bounds for Lanczos iteration j and the ω_js and the γ_js are generated by Lanczos.

Theorem 5.3 *The following algorithm yields a lower bound b_j of A^{-1}_{ii} by the Gauss quadrature rule, a lower bound $\bar{b}_j$ and an upper bound $\hat{b}_j$ through the Gauss–Radau quadrature rule and an upper bound $\breve{b}_j$ through the Gauss–Lobatto rule.*

Let $x_{-1} = 0$ and $x_0 = e_i$, $\omega_1 = a_{ii}$, $\gamma_1 = \|(A - \omega_1 I)e_i\|$, $b_1 = \omega_1^{-1}$, $d_1 = \omega_1$, $c_1 = 1$, $\hat{d}_1 = \omega_1 - a$, $\bar{d}_1 = \omega_1 - b$, $x_1 = (A - \omega_1 I)e_i/\gamma_1$.

Then for $j = 2, \ldots$ we compute

$$r_j = (A - \omega_j I)x_{j-1} - \gamma_{j-1}x_{j-2},$$

$$\omega_j = x_{j-1}^T A x_{j-1},$$

$$\gamma_j = \|r_j\|,$$

$$x_j = \frac{r_j}{\gamma_j},$$

$$b_j = b_{j-1} + \frac{\gamma_{j-1}^2 c_{j-1}^2}{d_{j-1}(\omega_j d_{j-1} - \gamma_{j-1}^2)},$$

$$d_j = \omega_j - \frac{\gamma_{j-1}^2}{d_{j-1}},$$

$$c_j = c_{j-1}\frac{\gamma_{j-1}}{d_{j-1}},$$

$$\hat{d}_j = \omega_j - a - \frac{\gamma_{j-1}^2}{\hat{d}_{j-1}},$$

$$\bar{d}_j = \omega_j - b - \frac{\gamma_{j-1}^2}{\bar{d}_{j-1}},$$

$$\hat{\omega}_j = a + \frac{\gamma_j^2}{\hat{d}_j},$$

$$\bar{\omega}_j = b + \frac{\gamma_j^2}{\bar{d}_j},$$

$$\hat{b}_j = b_j + \frac{\gamma_j^2 c_j^2}{d_j(\hat{\omega}_j d_j - \gamma_j^2)},$$

$$\bar{b}_j = b_j + \frac{\gamma_j^2 c_j^2}{d_j(\bar{\omega}_j d_j - \gamma_j^2)},$$

$$\breve{\omega}_j = \frac{\hat{d}_j \bar{d}_j}{\bar{d}_j - \hat{d}_j}\left(\frac{b}{\hat{d}_j} - \frac{a}{\bar{d}_j}\right),$$

$$\breve{\gamma}_j^2 = \frac{\hat{d}_j \bar{d}_j}{\bar{d}_j - \hat{d}_j}(b - a),$$

$$\breve{b}_j = b_j + \frac{\breve{\gamma}_j^2 c_j^2}{d_j(\breve{\omega}_j d_j - \breve{\gamma}_j{}^2)}.$$

Proof: We have from 3.4

$$J_N = \begin{pmatrix} \omega_1 & \gamma_1 & & & \\ \gamma_1 & \omega_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{N-2} & \omega_{N-1} & \gamma_{N-1} \\ & & & \gamma_{N-1} & \omega_N \end{pmatrix}.$$

Let $x_N^T = (0 \ \ldots \ 0 \ \gamma_N)$, so that

$$J_{N+1} = \begin{pmatrix} J_N & x_N \\ x_N^T & \omega_{N+1} \end{pmatrix}.$$

Letting

$$\tilde{J} = J_N - \frac{x_N x_N^T}{\omega_{N+1}},$$

the upper left block of J_{N+1}^{-1} is $\tilde{J}^{-1}$. This can be obtained through the use of the Sherman–Morrison formula (see [11]),

$$\tilde{J}^{-1} = J_N^{-1} + \frac{(J_N^{-1} x_N)(x_N^T J_N^{-1})}{\omega_{N+1} - x_N^T J_N^{-1} x_N}.$$

Let $j_N = J_N^{-1} e_N$ be the last column of the inverse of J_N. With this notation, we have

$$\tilde{J}^{-1} = J_N^{-1} + \frac{\gamma_N^2 j_N j_N^T}{\omega_{N+1} - \gamma_N^2 (j_N)_N}.$$

Therefore, it is clear that we only need the first and last elements of the last column of the inverse of J_N. This can be obtained using the Cholesky decomposition of J_N. It is easy to check that if we define

$$d_1 = \omega_1, \quad d_i = \omega_i - \frac{\gamma_{i-1}^2}{d_{i-1}}, \quad i = 2, \ldots, N$$

then

$$(j_N)_1 = (-1)^{N-1} \frac{\gamma_1 \cdots \gamma_{N-1}}{d_1 \cdots d_N}, \quad (j_N)_N = \frac{1}{d_N}.$$

When we put all these results together we get the proof of the Theorem.

The algorithm is essentially the same for the non–symmetric case. The modified algorithm is the following, using $f_1 = 1$:

$$\begin{aligned} r_j &= (A - \omega_j I) x_{j-1} - \beta_{j-1} x_{j-2}, \\ \hat{r}_j &= (A - \omega_j I) \hat{x}_{j-1} - \gamma_{j-1} \hat{x}_{j-2}, \\ \omega_j &= \hat{x}_{j-1}^T A x_{j-1}, \\ \gamma_j \beta_j &= \hat{r}_j^T r_j., \end{aligned}$$

$$x_j = \frac{r_j}{\gamma_j},$$

$$\hat{x}_j = \frac{\hat{r}_j}{\beta_j},$$

$$b_j = b_{j-1} + \frac{\gamma_{j-1}\beta_{j-1}c_{j-1}f_{j-1}}{d_{j-1}(\omega_j d_{j-1} - \gamma_{j-1}\beta_{j-1})},$$

$$d_j = \omega_j - \frac{\gamma_{j-1}^2}{d_{j-1}},$$

$$c_j = c_{j-1}\frac{\gamma_{j-1}}{d_{j-1}},$$

$$f_j = f_{j-1}\frac{\beta_{j-1}}{d_{j-1}},$$

$$\hat{d}_j = \omega_j - a - \frac{\gamma_{j-1}\beta_{j-1}}{\hat{d}_{j-1}},$$

$$\bar{d}_j = \omega_j - b - \frac{\gamma_{j-1}\beta_{j-1}}{\bar{d}_{j-1}},$$

$$\hat{\omega}_j = a + \frac{\gamma_j\beta_j}{\hat{d}_j},$$

$$\bar{\omega}_j = b + \frac{\gamma_j\beta_j}{\bar{d}_j},$$

$$\hat{b}_j = b_j + \frac{\gamma_j\beta_j c_j f_j}{d_j(\hat{\omega}_j d_j - \gamma_j\beta_j)},$$

$$\bar{b}_j = b_j + \frac{\gamma_j\beta_j c_j f_j}{d_j(\bar{\omega}_j d_j - \gamma_j\beta_j)},$$

$$\breve{\omega}_j = \frac{\hat{d}_j\bar{d}_j}{\bar{d}_j - \hat{d}_j}\left(\frac{b}{\hat{d}_j} - \frac{a}{\bar{d}_j}\right),$$

$$\breve{\gamma}_j^2 = \frac{\hat{d}_j\bar{d}_j}{\bar{d}_j - \hat{d}_j}(b-a),$$

$$\breve{b}_j = b_j + \frac{\breve{\gamma}_j\beta_j c_j f_j}{d_j(\breve{\omega}_j d_j - \breve{\gamma}_j\beta_j)}.$$

We have the analog for the block case. For simplicity we only consider the Gauss rule. Then, in 3.8

$$J_N = \begin{pmatrix} \Omega_0 & \Gamma_0^T & & & \\ \Gamma_0 & \Omega_1 & \Gamma_1^T & & \\ & \ddots & \ddots & \ddots & \\ & & \Gamma_{N-3} & \Omega_{N-2} & \Gamma_{N-2}^T \\ & & & \Gamma_{N-2} & \Omega_{N-1} \end{pmatrix},$$

and

$$J_{N+1} = \begin{pmatrix} J_N & x_N \\ x_N^T & \omega_{N+1} \end{pmatrix},$$

with

$$x_N^T = (0 \ \dots \ 0 \ \Gamma_{N-1}),$$

and

$$\tilde{J} = J_n - x_N \Omega_N^{-1} x_N^T.$$

Here, we use the Sherman–Morrison–Woodbury formula (see [11]) which is a generalization of the formula we used before. Then,

$$\tilde{J}^{-1} = J_N^{-1} + J_N^{-1} x_N (\Omega_N - x_N^T J_N^{-1} x_N)^{-1} x_N^T J_N^{-1}.$$

In order to compute all the elements, we need a block Cholesky decomposition of J_N. We obtain the following algorithm which gives a 2×2 matrix B_i, the block element of the inverse that we need.

Theorem 5.4 *Let $B_0 = \Omega_0^{-1}$, $\quad D_0 = \Omega_0$, $\quad C_0 = I$, for $i = 1, \dots$ we compute*

$$B_i = B_{i-1} + C_{i-1} D_{i-1}^{-1} \Gamma_{i-1}^T (\Omega_i - \Gamma_{i-1} D_{i-1}^{-1} \Gamma_{i-1}^T)^{-1} \Gamma_{i-1} D_{i-1}^{-1} C_{i-1}^T,$$

$$D_i = \Omega_i - \Gamma_{i-1} D_{i-1}^{-1} \Gamma_{i-1}^T,$$

$$C_i = C_{i-1} \Gamma_{i-1}^T D_{i-1}^{-1}.$$

These recurrences for 2×2 matrices can be easily computed. Hence, the approximations can be computed as we apply the block Lanczos algorithm.

Given these algorithms to compute the estimates, we see that almost all of the operations are a result of the Lanczos algorithm. Computing the estimate has a complexity independent of the problem size.

To compute a diagonal entry, the Lanczos algorithm needs per iteration the following operations: 1 matrix–vector product, $4N$ multiplies and $4N$ adds. To compute two diagonal entries and a non diagonal one, the block Lanczos algorithm needs per iteration: 2 matrix–vector products, $9N$ multiplies, $7N$ adds plus the QR decomposition which is $8N$ flops (see [11]). The non–symmetric Lanczos algorithm requires per iteration: 1 matrix–vector product, $6N$ multiplies and $6N$ adds.

Therefore, if we only want to estimate diagonal elements it is best to use the Lanczos algorithm. If we want to estimate a non diagonal element, it is best to use the block Lanczos algorithm since we get three estimates in one run while for the non–symmetric Lanczos method we need also to have an estimate of a diagonal element. The number of flops is the same but for the block Lanczos we have three estimates instead of two with the non–symmetric Lanczos. On the other hand, the non–symmetric Lanczos gives bounds but the block Lanczos yields only estimates.

As we notice before, we can compute bounds for the non diagonal elements by considering $\frac{1}{2}(e_i + e_j)^T A^{-1} (e_i + e_j)$. For this, we need to run three Lanczos algorithms that is per iteration: 3 matrix–vector products, $12N$ multiplies and $12N$ adds to get 3 estimates. This no more operations than in the block Lanczos case but here, we can get bounds.

With the non–symmetric Lanczos, we have 2 bounds with 2 matrix–vector products, $10N$ multiplies and $10N$ adds.

One can ask why in the case of the inverse are we not solving the linear system

$$Au = e_i$$

to obtain the i^{th} column of the inverse at once. To our knowledge, it is not possible then to tell if the estimates are upper or lower bounds. Moreover this can be easily added to the algorithm of Theorem 5.3.

Let $Q_N = [x_0, \ldots, x_{N-1}]$ be the matrix of the Lanczos vectors. Then we have the approximate solution

$$u_N = Q_N y_N,$$

where y_N is the solution of

$$J_N y_N = Q_N^T e_i = e_1.$$

This tridiagonal linear system can be easily solved incrementally from the LDL^T decomposition, see [11]. This yields a variant of the conjugate gradient algorithm. We give numerical examples in Section 6 and show that our methods give better bounds.

6 Numerical examples

In this Section, we first describe the examples we use and then we give numerical results for some specific functions f.

6.1 Description of the examples

First we look at examples of small dimension for which the inverses are known. Then, we will turn to larger examples arising from the discretization of partial differential equations. Most of the numerical computations have been done with Matlab 3.5 on an Apple Macintosh Powerbook 170 and a few ones on a Sun workstation.

Example 1.

First, we consider

$$A = I + uu^T, \quad u^T = (1, 1, \ldots, 1)$$

This matrix has two distinct eigenvalues 1 and $n+1$. Therefore, the minimal polynomial is of degree 2 and the inverse can be written as

$$A^{-1} = \frac{2+n}{1+n} I - \frac{1}{1+n} A.$$

Example 2.

The entries of the Hilbert matrix are given by $\frac{1}{i+j-1}$. We consider a matrix of dimension 5 which is

$$A(\alpha) = \alpha I_5 + \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{pmatrix}.$$

The inverse of $A(0)$ is

$$A^{-1} = \begin{pmatrix} 25 & -300 & 1050 & -1400 & 630 \\ -300 & 4800 & -18900 & 26880 & -1260 \\ 1050 & -18900 & 79380 & -117600 & 56700 \\ -1400 & 26880 & -117600 & 179200 & -88200 \\ 630 & -1260 & 56700 & -88200 & 44100 \end{pmatrix},$$

and the eigenvalues of $A(0)$ are

$$(3.288\ 10^{-6},\ 3.059\ 10^{-4},\ 1.141\ 10^{-2},\ 0.209,\ 1.567)$$

Example 3.

We take an example of dimension 10,

$$A = \frac{1}{11}\begin{pmatrix} 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 9 & 18 & 16 & 14 & 12 & 10 & 8 & 6 & 4 & 2 \\ 8 & 16 & 24 & 21 & 18 & 15 & 12 & 9 & 6 & 3 \\ 7 & 14 & 21 & 28 & 24 & 20 & 16 & 12 & 8 & 4 \\ 6 & 12 & 18 & 24 & 30 & 25 & 20 & 15 & 10 & 5 \\ 5 & 10 & 15 & 20 & 25 & 30 & 24 & 18 & 12 & 6 \\ 4 & 8 & 12 & 16 & 20 & 24 & 28 & 21 & 14 & 7 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 & 16 & 8 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 & 18 & 9 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{pmatrix}.$$

It is easily seen (cf. [13]) that the inverse is a tridiagonal matrix

$$A^{-1} = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}.$$

The eigenvalues of A are therefore distinct and given by

$$(0.2552,\ 0.2716,\ 0.3021,\ 0.3533,\ 0.4377,\ 0.5830,\ 0.8553,\ 1.4487,\ 3.1497,\ 12.3435).$$

Example 4.

We have

$$A = \begin{pmatrix} 3 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix},$$

whose inverse is

$$A^{-1} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 3 & 3 & 3 \\ 1 & 3 & 5 & 5 & 5 \\ 1 & 3 & 5 & 7 & 7 \\ 1 & 3 & 5 & 7 & 9 \end{pmatrix},$$

whose eigenvalues are

$$(0.0979,\ 0.8244,\ 2.0000,\ 3.1756,\ 3.9021).$$

Example 5.

We use a matrix of dimension 10 constructed with the TOEPLITZ function of Matlab,

$$A = 21I_{10} + \text{TOEPLITZ}(1:10).$$

This matrix has distinct eigenvalues but most of them are very close together:

$$(0.5683,\ 14.7435,\ 18.5741,\ 19.5048,\ 20.0000,\ 20.2292,\ 20.3702,\ 20.4462,\ 20.4875,\ 65.0763).$$

Example 6.

This example is the matrix arising from the 5–point finite difference of the Poisson equation in a unit square. This gives a linear system

$$Ax = b$$

of order m^2, where

$$A = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix}$$

each block being of order m and

$$T = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}.$$

For $m = 6$, the minimum and maximum eigenvalues are 0.3961 and 7.6039.

Example 7.

This example arises from the 5–point finite difference approximation of the following equation in a unit square,

$$-\text{div}(a\nabla u)) = f,$$

with Dirichlet boundary conditions. $a(x, y)$ is a diagonal matrix with equal diagonal elements. This element is equal to 1000 in a square $]1/4, 3/4[\times]1/4, 3/4[$, 1 otherwise.

For $m = 6$, the minimum and maximum eigenvalues are 0.4354 and 6828.7.

6.2 Results for a polynomial function

To numerically check some of the previous theorems, f was chosen as a polynomial of degree q,

$$f(\lambda) = \prod_{i=1}^{q} (\lambda - i).$$

We chose Example 6 with $m = 6$, that is a matrix of order 36.

1) We compute the $(2, 2)$ element of $f(A)$ and we vary the order of the polynomial. In the next table, we give, as a function of the degree q of the polynomial, the value of N to have an "exact" result (4 digits in Matlab) for the Gauss rule.

q	2	3	4	5	6
N	2	2	3	3	4

From these results, we can conclude that the maximum degree for which the results are exact is $q = 2N - 1$, as predicted by the theory.

For the Gauss–Radau rule, we get

q	2	3	4	5	6	7	8	9
N	2	2	2	3	3	4	4	5

From this we deduce $q = 2N$ as predicted.

For the Gauss–Lobatto rule, we have

q	2	3	4	5	6	7	8	9
N	1	1	2	2	3	3	4	4

This shows that $q = 2N + 1$ which is what we expect.

2) If we consider the block case to compute the $(3, 1)$ element of the polynomial, we get the same results, therefore the block Gauss rule is exact for the degree $2N - 1$, the block Gauss–Radau rule is exact for degree $2N$ and the block Gauss–Lobatto is exact for degree $2N + 1$.

3) The same is also true for the non–symmetric Lanczos algorithm if we want to compute the sum of the $(3, 1)$ and $(3, 3)$ elements.

6.3 Bounds for the inverse

6.3.1 Diagonal elements

Now, we turn to some numerical experiments using Matlab on the examples described above. Usually the results will be given using the "short" format of Matlab. In the following results, Nit denotes the number of iterations of the Lanczos algorithm. This corresponds to N for the Gauss and Gauss–Radau rules and $N - 1$ for the Gauss–Lobatto rule.

Example 1.

Because of the properties of the matrix we should get the answer in two steps. We have $\omega_0 = 2$ and $\gamma_0 = n - 1$, therefore, the lower bound from Gauss–Radau is $\frac{n}{n+1}$ and the upper bound is $\frac{n}{n+1}$, the exact result. If we look at the lower bound from the Gauss rule, we find the same value. This is also true for the numerical experiments as well as for Gauss–Lobatto.

Example 2.

Let us consider $(A(0)^{-1})_{33}$ whose exact value is 79380. The Gauss rule, as a function of the degree of the quadrature, gives

Results from Gauss rule

Nit=1	2	3	4	5
5	26.6	1808.3	3666.8	79380

The Gauss–Radau rule gives upper and lower bounds. For a and b, we use the computed values from the EIG function of Matlab.

Results from the Gauss–Radau rule

	Nit=1	2	3	4	5
lw bnd	23.74	1801.77	3666.58	3559.92	79380
up bnd	257674	216812	202814	79380	79380

Results from Gauss–Lobatto rule

Nit=1	2	3	4	5
265330	216870	202860	79268	79380

The results are not as good as expected. The exact results should have been obtained for $Nit = 3$. The discrepancy comes from round off errors, particularly for the lower bound, because of the eigenvalue distribution of A and a poor convergence rate of the Lanczos algorithm in this case. To see how this is related to the conditioning of A, let us vary α. For simplicity we consider only the Gauss rule. The following tables give results for different values of α and the $(3,3)$ element of the inverse. The exact values are $(\alpha = 0.01,\ 70.3949)$, $(\alpha = 0.1,\ 7.7793)$, $(\alpha = 1,\ 0.9054)$.

Lower bound from Gauss rule for $\alpha = 0.01$

Nit=2	3	4	5
20.5123	69.7571	70.3914	70.3949

Lower bound from Gauss rule for $\alpha = 0.1$

Nit=2	3	4	5
6.7270	7.7787	7.7793	7.7793

Lower bound from Gauss rule for $\alpha = 1$

Nit=2	3	4	5
0.9040	0.9054	0.9054	0.9054

We see that when A is well conditioned, the numerical results follow the theory. The discrepancies probably arise from the poor convergence of the smallest eigenvalues of J_N towards those of A.

Example 3.
We are looking for bounds for $(A^{-1})_{55}$ whose exact value is, of course, 2.

Lower bounds for $(A^{-1})_{55}$ from the Gauss rule

Nit=1	2	3	4	5	6	7
0.3667	1.3896	1.7875	1.9404	1.9929	1.9993	2

Results for $(A^{-1})_{55}$ from the Gauss–Radau rule

	Nit=1	2	3	4	5	6	7
b_1	1.3430	1.7627	1.9376	1.9926	1.9993	2.0117	2
b_2	3.0330	2.2931	2.1264	2.0171	2.0020	2.0010	2

Upper bounds for $(A^{-1})_{55}$ from the Gauss–Lobatto rule

Nit=1	2	3	4	5	6	7
3.1341	2.3211	2.1356	2.0178	2.0021	2.0001	2

In this example 5 or 6 iterations should be sufficient, so we are a little off the theory.

Example 4.
We look at bounds for $(A^{-1})_{55}$ whose exact value is 4.5

Lower bounds for $(A^{-1})_{55}$ from the Gauss rule

Nit=1	2	3	4	5
1	2	3	4	4.5

Lower and upper bounds for $(A^{-1})_{55}$ from the Gauss–Radau rule

	Nit=1	2	3	4	5
lw bnd	1.3910	2.4425	3.4743	4.5	4.5
up bnd	5.8450	4.7936	4.5257	4.5	4.5

Upper bounds for $(A^{-1})_{55}$ from the Gauss–Lobatto rule

Nit=1	2	3	4	5
7.8541	5.2361	4.6180	4.5	4.5

Example 5.
We get for $(A^{-1})_{55}$, whose value is 0.0595,

Lower bounds for $(A^{-1})_{55}$ from the Gauss rule

Nit=1	2	3	4	5
0.0455	0.0511	0.0523	0.0585	0.0595

Lower and upper bounds for $(A^{-1})_{55}$ from the Gauss–Radau rule

	Nit=1	2	3	4	5
lw bnd	0.0508	0.0522	0.0582	0.0595	0.0595
up bnd	0.4465	0.0721	0.0595	0.0595	0.0595

Upper bounds for $(A^{-1})_{55}$ *from the Gauss–Lobatto rule*

Nit=1	2	3	4	5
1.1802	0.0762	0.0596	0.0595	0.0595

Because some eigenvalues are very close together, we get the exact answers a little sooner than it is predicted by theory.

Example 6.

Consider $m = 6$. Then we have a system of order 36 and we look for bounds on $(A^{-1})_{18,18}$ whose value is 0.3515. There are 19 distinct eigenvalues, therefore we should get the exact answer in about 10 iterations for Gauss and Gauss–Radau and 9 iterations for Gauss–Lobatto.

Lower bounds for $(A^{-1})_{18,18}$ *from the Gauss rule*

Nit=1	2	3	4	8	9
0.25	0.3077	0.3304	0.3411	0.3512	0.3515

Lower and upper bounds for $(A^{-1})_{18,18}$ *from the Gauss–Radau rule*

	Nit=1	2	3	4	8	9
lw bnd	0.2811	0.3203	0.3366	0.3443	0.3514	0.3515
up bnd	0.6418	0.4178	0.3703	0.3572	0.3515	0.3515

Upper bounds for $(A^{-1})_{18,18}$ *from the Gauss–Lobatto rule*

Nit=1	2	3	4	8
1.3280	0.4990	0.3874	0.3619	0.3515

Now, we consider $m = 16$ which gives a matrix of order 256. We want to compute bounds for the $(125, 125)$ element whose value is 0.5604. In this case there are 129 distinct eigenvalues, so we should find the exact answer in about 65 iterations at worst. These computations for a larger problem have been done on a Sun Sparcstation 1+. We find the following results.

Lower bounds for $(A^{-1})_{125,125}$ *from the Gauss rule*

Nit=2	3	4	5	6	7	8	9	10	20
0.3333	0.3929	0.4337	0.4675	0.4920	0.5084	0.5201	0.5301	0.5378	0.5600

Lower and upper bounds for $(A^{-1})_{125,125}$ *from the Gauss–Radau rule*

	Nit=2	3	4	5	6	7	8	10	20
lw bnd	0.3639	0.4140	0.4514	0.4804	0.5006	0.5146	0.5255	0.5414	0.5601
up bnd	1.5208	1.0221	0.8154	0.7130	0.6518	0.6139	0.5925	0.5730	0.5604

Upper bounds for $(A^{-1})_{125,125}$ from the Gauss–Lobatto rule

Nit=2	3	4	5	6	7	8	9	10	18
2.1011	1.2311	0.8983	0.7585	0.6803	0.6310	0.6012	0.5856	0.5760	0.5604

We have very good estimates much sooner than predicted. This is because there are distinct eigenvalues which are very close together.

We also ran two other examples with $m = 10$ and $m = 20$ that show that the number of iterations to reach a "correct" value with four exact digits grows like m. This can be expected from the Lanczos method.

Example 7.

We took $m = 6$ as in the previous example. So we have a matrix of dimension 36. The $(2,2)$ element of the inverse has an "exact" value of 0.3088 and there are 23 distinct eigenvalues so that the exact answer should be obtained after 12 iterations but the matrix is ill conditioned. We get the following results:

Lower bounds for $(A^{-1})_{2,2}$ from the Gauss rule

Nit=1	2	3	4	5	6	8	10	12	15
0.25	0.2503	0.2510	0.2525	0.2553	0.2609	0.2837	0.2889	0.3036	0.3088

Lower and upper bounds for $(A^{-1})_{2,2}$ from the Gauss–Radau rule

	Nit=2	3	4	5	6	7	8	10	12	15
lw bnd	0.2504	0.2516	0.2538	0.2583	0.2699	0.2821	0.2879	0.2968	0.3044	0.3088
up bnd	0.5375	0.5202	0.5121	0.5080	0.5060	0.5039	0.5013	0.3237	0.3098	0.3088

Upper bounds for $(A^{-1})_{2,2}$ from the Gauss–Lobatto rule

Nit=1	2	3	4	5	6	8	10	12	15
2.2955	0.5765	0.5289	0.5156	0.5093	0.5065	0.5020	0.3237	0.3098	0.3088

6.3.2 Non diagonal elements with non–symmetric Lanczos

Here, we use the non–symmetric Lanczos algorithm to get estimates on non diagonal elements.

Example 1.

The matrix is of dimension $n = 5$. All the non diagonal elements are $-1/6 = -0.1667$ and the diagonal elements are equal to 0.8333.

We compute the sum of the $(2,2)$ and $(2,1)$ elements (e.g. $\delta = 1$), that is 0.6667. With the Gauss rule, after 1 iteration we get 0.3333 and after 2 iterations 0.6667. With Gauss Radau, we obtain the result in one iteration as well as with Gauss Lobatto.

We note that for $\delta = 1$, the measure is not positive but for $\delta = 2$ the measure is positive and increasing.

Example 2.

Let us consider first the Hilbert matrix and $(A(0)^{-1})_{2,1}$ whose exact value is 79380. We compute the sum of the $(2,2)$ and $(2,1)$ elements, (i.e. $\delta = 1$) that is 4500. The

Gauss rule, as a function of the number of iterations, gives

Bounds from the non–symmetric Gauss rule

Nit=1	2	3	4	5
1.2	-21.9394	73.3549	667.1347	4500

Consider now the non–symmetric Gauss–Radau rule.

Bounds from the non–symmetric Gauss–Radau rule

	Nit=1	2	3	4	5
b1	-17.5899	73.1917	667.1277	667.0093	4500
b2	144710	155040	51854	4500	4500

Bounds from the non–symmetric Gauss–Lobatto rule

Nit=1	2	3	4	5
142410	155570	51863	3789.2	4500

Note that the measure is positive and increasing therefore, we obtain a lower bound with the Gauss rule, b_1 is a lower bound and b_2 an upper bound with Gauss–Radau and Gauss–Lobatto gives an upper bound.

Again the results are not so good. Consider now the results of the non–symmetric Gauss rule with a better conditioned problem by looking at $A(0.1)$. The sum of the elements we compute is 5.1389. In the last line we indicate if the product of the non diagonal coefficients is positive (p) or negative (n). If it is positive we should have a lower bound, an upper bound otherwise. Note that in this case, the measure is positive but not increasing.

Estimates from the non–symmetric Gauss rule for $\alpha = 0.1$

Nit=1	2	3	4	5
1.0714	6.1735	5.1341	5.1389	5.1389
p	n	p	p	p

We see that the algorithm is able to determine if it is computing a lower or an upper bound.

Estimates from the non–symmetric Gauss–Radau rule

	Nit=1	2	3	4
b1	6.5225	5.1338	5.1389	5.1389
b2	5.0679	5.2917	5.1390	5.1389

We remark that b_1 and b_2 are alternatively upper and lower bounds.

Estimates from the non–symmetric Gauss–Lobatto rule

Nit=1	2	3	4
5.0010	5.3002	5.1390	5.1389

Again, we do not have an upper bound with Gauss–Lobatto but the results oscillate around the exact value. In this case, this can be fixed by using a value $\delta = 3$ that gives a positive increasing measure.

Example 3.

We are looking for estimates for the sum of the $(2, 2)$ and $(2, 1)$ elements whose exact value is 1. First, we use $\delta = 1$ for which the measure is positive but not increasing.

Estimates from the non–symmetric Gauss rule

Nit=1	2	3	4	5	6	7
0.4074	0.6494	0.8341	0.9512	0.9998	1.0004	1
p	p	p	p	p	n	p

Estimates from the non–symmetric Gauss–Radau rule

	Nit=1	2	3	4	5	6	7
b1	0.6181	0.8268	0.9488	0.9998	1.0004	1.0001	1
b2	2.6483	1.4324	1.0488	1.0035	1.0012	0.9994	1

Estimates from the non–symmetric Gauss–Lobatto rule

Nit=1	2	3	4	5	6	7	8
3.2207	1.4932	1.0529	1.0036	1.0012	0.9993	0.9994	1

Here we have a small problem at the end near convergence, but the estimates are quite good. Note that for $\delta = 4$ the measure is positive and increasing.

Example 4.

This example illustrates some of the problems that can happen with the non–symmetric Lanczos algorithm. We would like to compute the sum of the $(2, 2)$ and $(2, 1)$ elements that is 2. After 2 iterations we have a breakdown of the Lanczos algorithm as $\gamma\beta = 0$. The same happens at the first iteration for the Gauss–Radau rule and at the second one for the Gauss–Lobatto rule. Choosing a value of δ different from 1 cures the breakdown problem. We can obtain bounds with a value $\delta = 10$ (with a positive and increasing measure). Then the value we are looking for is 1.55 and the results follow.

Bounds from the non–symmetric Gauss rule

Nit=1	2	3	4	5
0.5263	0.8585	1.0333	1.4533	1.55

Bounds from the non–symmetric Gauss–Radau rule

	Nit=2	3	4
b1	1.0011	1.2771	1.55
b2	1.9949	1.5539	1.55

Bounds from the non–symmetric Gauss–Lobatto rule

Nit=2	3	4
2.2432	1.5696	1.55

Example 5.

The sum of the (2, 2) and (2, 1) elements is 0.6158.

Bounds from the non–symmetric Gauss rule

Nit=1	2	3	4	5
0.0417	0.0974	0.4764	0.6155	0.6158
p	p	p	p	p

Bounds from the non–symmetric Gauss–Radau rule

	Nit=1	2	3	4
b1	0.0847	0.4462	0.6154	0.6158
b2	0.9370	0.6230	0.6158	0.6158

Bounds from the non–symmetric Gauss–Lobatto rule

Nit=1	2	3	4
1.1261	0.6254	0.6159	0.6158

Example 6.

We consider $m = 6$, then, we have a system of order 36 and we look for estimates of the sum of the (2, 2) and (2, 1) elements which is 0.4471. Remember there are 19 distinct eigenvalues.

Bounds from the non–symmetric Gauss rule

Nit=1	2	3	4	5	6	7	8	9
0.3333	0.4000	0.4262	0.4369	0.4419	0.4446	0.4461	0.4468	0.4471
p	p	p	p	p	p	p	p	p

Bounds from the non–symmetric Gauss–Radau rule

	Nit=1	2	3	4	5	6	7	8	9
b1	0.3675	0.4156	0.4320	0.4390	0.4436	0.4456	0.4466	0.4470	0.4471
b2	0.7800	0.5319	0.4690	0.4537	0.4490	0.4476	0.4472	0.4472	0.4471

Bounds from the non–symmetric Gauss–Lobatto rule

Nit=1	2	3	4	5	6	7	8	9	10
1.6660	0.6238	0.4923	0.4596	0.4505	0.4480	0.4473	0.4472	0.4472	0.4471

Example 7.

We took $m = 6$ as in the previous example. So we have a matrix of dimension 36. The sum of the (2, 2) and (2, 1) elements of the inverse is 0.3962 and there are 23 distinct eigenvalues. We get the following results:

Bounds from the non–symmetric Gauss rule

Nit=1	2	3	4	5	6	8	10	12	15
0.3333 P	0.3336 P	0.3340 P	0.3348 P	0.3363 P	0.3396 P	0.3607 P	0.3689 P	0.3899 P	0.3962 P

Bounds from the non–symmetric Gauss–Radau rule

	Nit=2	3	4	5	6	8	10	12	15
b1	0.3337	0.3343	0.3355	0.3380	0.3460	0.3672	0.3803	0.3912	0.3962
b2	0.6230	0.5930	0.5793	0.5725	0.5698	0.5660	0.4078	0.3970	0.3962

Bounds from the non–symmetric Gauss–Lobatto rule

Nit=1	2	3	4	5	6	8	10	12	15
2.2959	0.6898	0.6081	0.5850	0.5746	0.5703	0.5664	0.4078	0.3970	0.3962

Finally, one can ask why we do not store the Lanczos vectors x_j and compute an approximation to the solution of $Au = e_i$. This can be done doing the following. Let

$$Q_N = [x_0, \dots, x_{N-1}].$$

If we solve

$$J_N y_N = e_1,$$

then the approximate solution is given by $Q_N y_N$.

Unfortunately this does not give bounds and even the approximations are not as good as with our algorithms. Consider Example 5 and computing the fifth column of the inverse. For the element (1,5) whose "exact" value is 0.460, we find

Estimates from solving the linear system

Nit= 2	3	4	5	6
-0.0043	-0.0046	0.0382	0.0461	0.0460

By computing bounds for the sum of the (5,5) and (1,5) elements and subtracting the bounds for the (5,5) element, we obtain

Bounds from the Gauss–Radau quadrature rules

	Nit= 2	3	4
lw bnd	0.0048	0.0451	0.0460
up bnd	0.0551	0.0473	0.0460

We see that we get good bounds quite fast.

6.3.3 Non diagonal elements with block Lanczos

Here, we use the block Lanczos algorithm to get estimates on non diagonal elements. Unfortunately, most of the examples are too small to be of interest as for matrices of dimension 5 we cannot go further than 2 block iterations. Nevertheless, let us look at the results.

Example 1.

Consider a matrix of dimension $n = 5$. Then all the non diagonal elements are $-1/6 = -0.1667$.

We compute the $(2,1)$ element. With the block Gauss rule, after 2 iterations we get -0.1667. With block Gauss–Radau, we get the exact answer in 1 iteration as well as with Gauss–Lobatto.

Example 2.

The $(2,1)$ element of the inverse of the Hilbert matrix $A(0)$ of dimension 5 is -300.

With the block Gauss rule, after 2 iterations we find -90.968. Note that this is an upper bound. With block Gauss–Radau, 2 iterations give -300.2 as a lower bound and -300 as an upper bound. Block Gauss–Lobatto gives -5797 as a lower bound.

Now we consider $A(0.1)$ for which the $(2,1)$ element of the inverse is -1.9358. After 2 iterations, block Gauss gives -2.2059 a lower bound and block Gauss–Radau and Gauss–Lobatto give the exact answer.

Example 3.

The $(2,1)$ element is -1, the $(3,1)$ element is 0. After 2 iterations we get the exact answers with Gauss as well as with Gauss–Radau. Gauss–Lobatto gives -0.0609, a lower bound. Three iterations give the exact answer.

Example 4.

The $(2,1)$ element is $1/2$. After 2 iterations, we have 0.3182 which is a lower bound. Gauss–Radau gives 0.2525 and 0.6807. Gauss–Lobatto gives 0.7236 which is an upper bound.

Example 5.

The $(2,1)$ element is 0.2980. Two iterations give 0.2329, a lower bound and 3 iterations give the exact answer. Gauss–Radau and Gauss–Lobatto also give the exact answer in 3 iterations.

Example 6.

This example uses $n = 36$. The $(2,1)$ element is 0.1040. Remember that we should do about 10 iterations. We get the following figures.

Estimates from the block Gauss rule

Nit=2	3	4	5	6	7	8
0.0894	0.0974	0.1008	0.1024	0.1033	0.1037	0.1040

Estimates from the block Gauss–Radau rule

Nit=2	3	4	5	6	7	8
0.0931 0.1257	0.0931 0.1103	0.1017 0.1059	0.1029 0.1046	0.1035 0.1042	0.1038 0.1041	0.1040 0.1040

Estimates from the block Gauss–Lobatto rule

Nit=2	3	4	5	6	7	8
0.1600	0.1180	0.1079	0.1051	0.1041	0.1043	0.1041

Note that here everything works. Gauss gives a lower bound, Gauss–Radau a lower and an upper bound and Gauss–Lobatto an upper bound.

Example 7.

We would like to obtain estimates of the $(2,1)$ whose value is 0.0874. We get the following results.

Estimates from the block Gauss rule

Nit=2	4	6	8	10	12	14	15
0.0715	0.0716	0.0722	0.0761	0.0789	0.0857	0.0873	0.0874

Estimates from the block Gauss–Radau rule

Nit=2	4	6	8	10	12	14	15
0.0715	0.0717	0.0731	0.0782	0.0831	0.0861	0.0873	0.0874
0.1375	0.1216	0.1184	0.1170	0.0894	0.0876	0.0874	0.0874

Estimates from the block Gauss–Lobatto rule

Nit=2	4	6	8	10	12	14
0.1549	0.1237	0.1185	0.1176	0.0894	0.0876	0.0874

Note that in this example we obtain bounds. Now, to illustrate what we said before about the estimates being 0 for some iterations, we would like to estimate the $(36,1)$ element of the inverse which is 0.005.

Estimates from the block Gauss rule

Nit=2	4	6	8	10	11
0.	0.	0.0023	0.0037	0.0049	0.0050

Estimates from the block Gauss–Radau rule

Nit=2	4	6	8	10	11
0.	0.	0.0023	0.0037	0.0049	0.0050
0.	0.	0.0024	0.0050	0.0050	0.0050

Estimates from the block Gauss–Lobatto rule

Nit=2	4	6	8	10
0.	0.	0.0022	0.0043	0.0050

6.3.4 Dependence on the eigenvalue estimates

In this sub–Section, we numerically investigate how the bounds and estimates of the Gauss–Radau rules depend on the accuracy of the estimates of the eigenvalues of A. We take Example 6 with $m = 6$ and look at the results given by the Gauss–Radau rule as a function of a and b. Remember that in the previous experiments we took for a and b the values returned by the EIG function of Matlab.

It turns out that the estimates are only weakly dependent of the values of a and b (for this example). We look at the number of iterations needed to obtain an upper for the element $(18, 18)$ with four exact digits and with an "exact" value of b. The results are given in the following table.

a=10^{-4}	10^{-2}	0.1	0.3	0.4	1	6
15	13	11	11	8	8	9

We have the same properties when b is varied.

Therefore, we see that the estimation of the extreme eigenvalues does not seem to matter very much and can be obtained with a few iterations of Lanczos or with the Gerschgorin circles.

6.4 Bounds for the exponential

In this Section we are looking for bounds of diagonal elements of the exponential of the matrices of some of the examples.

6.4.1 diagonal elements

Example 1

We consider the $(2, 2)$ element whose value is 82.8604. With Gauss, Gauss–Radau and Gauss–Lobatto we obtain the exact value in 2 iterations.

Example 2

We would like to compute the $(3, 3)$ element whose value is 1.4344. Gauss gives the answer in 3 iterations, Gauss–Radau and Gauss–Lobatto in 2 iterations.

Example 3

The $(5, 5)$ entry is 4.0879 10^4. Gauss obtains the exact value in 4 iterations, Gauss–Radau and Gauss–Lobatto in 3 iterations.

Example 6

We consider the $(18, 18)$ element whose value is 197.8311. We obtain the following results.

Lower bounds from the Gauss rule

Nit=2	3	4	5	6	7
159.1305	193.4021	197.5633	197.8208	197.8308	197.8311

Lower and upper bounds from the Gauss–Radau rule

	Nit=2	3	4	5	6
lw bnd	182.2094	196.6343	197.7779	197.8296	197.8311
up bnd	217.4084	199.0836	197.8821	197.8325	197.8311

Upper bounds from the Gauss–Lobatto rule

Nit=2	3	4	5	6	7
273.8301	203.4148	198.0978	197.8392	197.8313	197.8311

We remark that to compute diagonal elements of the exponential the convergence rate is quite fast.

6.4.2 non diagonal elements

Here we consider only Example 6 and we would like to compute the element $(2, 1)$ whose value is -119.6646. First, we use the block Lanczos algorithm which give the following results.

Results from the block Gauss rule

Nit=2	3	4	5	6
-111.2179	-119.0085	-119.6333	-119.6336	-119.6646

Results from the block Gauss–Radau rule

	Nit=2	3	4	5	6
b1	-115.9316	-119.4565	-119.6571	-119.6644	-119.6646
b2	-122.2213	-119.7928	-119.6687	-119.6647	-119.6646

Results from the block Gauss–Lobatto rule

Nit=2	3	4	5	6
-137.7050	-120.6801	-119.7008	-119.6655	-119.6646

Now, we use the non–symmetric Lanczos algorithm. The sum of the $(2, 2)$ and $(2, 1)$ elements of the exponential is 73.9023.

Results from the non–symmetric Gauss rule

Nit=2	3	4	5	6	7
54.3971	71.6576	73.7637	73.8962	73.9021	73.9023

Results from the non–symmetric Gauss–Radau rule

	Nit=2	3	4	5	6
b1	65.1847	73.2896	73.8718	73.9014	73.9023
b2	84.0323	74.6772	73.9323	73.9014	73.9023

Results from the non–symmetric Gauss–Lobatto rule

Nit=2	3	4	5	6	7
113.5085	77.2717	74.0711	73.9070	73.9024	73.9023

6.5 Bounds for other functions

When one uses domain decomposition methods for matrices arising from the finite difference approximation of partial differential equations in a rectangle, it is known that the matrix

$$A = \sqrt{T + \frac{1}{4}T^2},$$

where T is the matrix of the one dimensional Laplacian, is a good preconditioner for the Schur complement matrix. It is interesting to see if we can estimate some elements of the matrix A to generate a Toeplitz tridiagonal approximation to A.

We have

$$T = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix},$$

and we choose an example of dimension 100. We estimate the $(50, 50)$ element whose exact value is 1.6367 with the Gauss–Radau rule. We obtain the following results.

Estimates from the Gauss–Radau rule

	Nit=2	3	4	5	10	15	20
lw	1.6014	1.6196	1.6269	1.6305	1.6355	1.6363	1.6365
up	1.6569	1.6471	1.6430	1.6409	1.6378	1.6371	1.6369

We estimate the non diagonal elements by using the block Gauss rule. We choose the $(49, 50)$ element.

Estimates from the block Gauss rule

Nit=2	3	4	5	10	15	20
-0.6165	-0.6261	-0.6302	-0.6323	-0.6354	-0.6361	-0.6363

Now, we construct a Toeplitz tridiagonal matrix C whose elements are chosen From the estimates given at the fifth iteration. We took the average of the Gauss–Radau values for the diagonal (1.6357) and -0.6323 for the non diagonal elements. We look at the spectrum of $C^{-1}A$. The condition number is 13.35 the minimum eigenvalue being 0.0837 and the maximum one being 1.1174, but there are 86 eigenvalues between 0.9 and the maximum eigenvalue. Therefore, the matrix C (requiring only 5 iterations of some Lanczos algorithms) seems to be a good preconditioner for A which is itself a good preconditioner for the Schur complement.

7 Conclusions

We have shown how to obtain bounds (or in certain cases estimates) of the entries of a function of a symmetric positive definite matrix. The proposed algorithms use the Lanczos algorithm to estimate diagonal entries and either the non–symmetric Lanczos or block Lanczos algorithms for the non diagonal entries.

The algorithms are particularly simple for the inverse of a matrix. Analytical bounds are derived by considering one or two iterations of these algorithms. We have seen in the numerical experiments that very good approximations are obtained in a few iterations.

References

[1] F.V. Atkinson, "Discrete and continuous boundary problems", (1964) Academic Press

[2] S. Basu, N.K. Bose, "Matrix Stieltjes series and network models", SIAM J. Math. Anal. v14 n2 (1983) pp 209–222

[3] G. Dahlquist, S.C. Eisenstat and G.H. Golub, "Bounds for the error of linear systems of equations using the theory of moments", J. Math. Anal. Appl. 37 (1972) pp 151–166

[4] P. Davis, P. Rabinowitz, "Methods of numerical integration", Second Edition (1984) Academic Press

[5] W. Gautschi, "Construction of Gauss–Christoffel quadrature formulas", Math. Comp. 22 (1968) pp 251–270

[6] W. Gautschi, "Orthogonal polynomials– constructive theory and applications", J. of Comp. and Appl. Math. 12 & 13 (1985) pp 61–76

[7] G.H. Golub, J.H. Welsch, "Calculation of Gauss quadrature rule" Math. Comp. 23 (1969) pp 221–230

[8] G.H. Golub, "Some modified matrix eigenvalue problems", SIAM Review v15 n2 (1973) pp 318–334

[9] G.H. Golub, "Bounds for matrix moments", Rocky Mnt. J. of Math., v4 n2 (1974) pp 207–211

[10] G.H. Golub, R. Underwood, "The block Lanczos method for computing eigenvalues", in Mathematical Software III, Ed. J. Rice (1977) pp 361–377

[11] G.H. Golub, C. van Loan, "Matrix Computations", Second Edition (1989) Johns Hopkins University Press

[12] R. Haydock, "Accuracy of the recursion method and basis non–orthogonality", Computer Physics Communications 53 (1989) pp 133–139

[13] G. Meurant, "A review of the inverse of tridiagonal and block tridiagonal matrices", SIAM J. Matrix Anal. Appl. v13 n3 (1992) pp 707–728

[14] C.M. Nex, "Estimation of integrals with respect to a density of states", J. Phys. A, v11 n4 (1978) pp 653–663

[15] C.M. Nex, "The block Lanczos algorithm and the calculation of matrix resolvents", Computer Physics Communications 53 (1989) pp 141–146

[16] D.P. O'Leary, "The block conjugate gradient algorithm and related methods", Linear Alg. and its Appl. v29 (1980) pp 293–322

[17] P.D. Robinson, A. Wathen, "Variational bounds on the entries of the inverse of a matrix", IMA J. of Numer. Anal. v12 (1992) pp 463–486

[18] J. Stoer, R. Bulirsch, "Introduction to numerical analysis", Second Edition (1983) Springer Verlag

[19] G.W. Struble, "Orthogonal polynomials: variable–signed weight functions", Numer. Math v5 (1963) pp 88–94

[20] G. Szegö, "Orthogonal polynomials", Third Edition (1974) American Mathematical Society

[21] H.S. Wilf, "Mathematics for the physical sciences", (1962) Wiley

25

COMPUTATION OF GAUSS-KRONROD QUADRATURE RULES (WITH D. CALVETTI, W.B. GRAGG AND L. REICHEL)

Math. Comp. (69): 231, 1035–1052 (2000).

If I think about the genesis of this paper, my semester at Stanford in 1995 comes to mind right away. For the mathematical orphan that I was, since my advisor, Peter Henrici died before I completed my PhD, the opportunity to spend a semester at Stanford was a truly unique opportunity. As Gene and I team-taught a seminar course on quadrature rules, linear algebra, and large-scale computations, I discovered more charming aspects of quadrature rules than I thought there could be. I must admit that when Gene first talked about the importance of Gauss–Kronrod rules in estimating errors in numerical integration, I thought that, for sure, I would *never* be caught working on such a dull topic. Well, soon enough I started seeing quadrature rules everywhere, and their elegance and secret connections with nearly everything began to get hold of me. Even Gauss–Kronrod quadrature rules, until then quickly dismissed even from my teaching, started to show their discrete charm. Well, never say never!

During my semester at Stanford, Lothar Reichel visited quite regularly and often Bill Gragg come up from Monterey. Appropriately interlaced with colorful stories from that life that normal people would never suspect mathematicians would have and the interruptions of my two trailing-along young children, we managed to talk about Gauss–Kronrod rules, linear algebra, and efficient and stable calculations of nodes and weights – all the ingredient which ended up being the paper. A few months ago, while looking for some information for a student, I saw that this paper is now used by *Mathematica* in the error control for numerical integration and all of a sudden I smelled the scent of rosemary hedges on the Stanford campus.

Daniela Calvetti
Cleveland, Ohio, USA

This paper illustrates Gene Golub's well-known ability to identify important problems that have an elegant solution. Gene has throughout his career provided beautiful solutions to a large number of significant computational problems and thereby made many important contributions to scientific computing. Several of Gene's papers use properties of orthogonal polynomials and Gauss quadrature in a linear algebra context. By exploiting the connections between linear algebra and orthogonal polynomials, Gene has been able to develop many elegant, significant, and powerful numerical methods.

The computation of Gauss-type quadrature rules has been one of Gene Golub's enduring interests. Let $d\omega$ be a nonnegative measure on the real axis, and let T_n be the symmetric tridiagonal matrix determined by the recursion coefficients for the first n orthonormal polynomials with respect to this measure. The eigenvalues of T_n are the nodes of the n-point Gauss quadrature rule G_n associated with $d\omega$, and the weights of G_n are the squares of the first components of normalized eigenvectors. In 1969 Gene published a variant of the QR algorithm for computing the nodes and weights of G_n from the recursion coefficients of the associated orthogonal polynomials in only $\mathcal{O}(n^2)$ arithmetic floating point operations. This algorithm, known as the Golub–Welsch algorithm, is discussed by Gautschi in this volume. In 1973 Gene described how to determine the symmetric tridiagonal matrices associated with Gauss–Radau and Gauss–Lobatto quadrature rules from the recursion coefficients of the associated orthogonal polynomials. This elegant work makes it possible to apply the Golub–Welsch algorithm to determine the nodes and weights of n-point Gauss–Radau and Gauss–Lobatto quadrature rules in only $\mathcal{O}(n^2)$ arithmetic floating point operations.

Gene Golub's experience with Gauss-type quadrature rules helped him to see the possibility of developing a new method for the computation of Gauss–Kronrod quadrature rules. In the fall of 1996 Gene sent me a postscript file for a new elegant paper by Laurie on the computation of Gauss–Kronrod quadrature rules. Let G_n be the n-point Gauss quadrature rule introduced above and let K_{2n+1} denote the associated $(2n+1)$-point Gauss–Kronrod rule. Typically, one is interested in Gauss–Kronrod rules with real nodes, n of which agree with the nodes of G_n. The remaining $n+1$ nodes of K_{2n+1} are interlaced by the nodes of G_n. There is a $(2n+1) \times (2n+1)$ symmetric tridiagonal matrix S_{2n+1} associated with K_{2n+1}, such that its eigenvalues are the nodes and the squares of the first components of normalized eigenvectors are the weights of K_{2n+1}. The matrix S_{2n+1} is not explicitly known. It is fairly easy to see that the leading principal $n \times n$ submatrix of S_{2n+1} is T_n, and Laurie showed that the trailing $n \times n$ principal submatrix of S_{2n+1} is similar to T_n. This observation is the basis of Laurie's algorithm for computing S_{2n+1}. The nodes and weights of the Gauss–Kronrod rule can then be determined from S_{2n+1} by the Golub–Welsch algorithm.

It is particularly nice to make progress on a paper in a nonstandard environment away from the office. Daniela and I did some of the work on this project with Gene during a sunny summer day at Geauga Lake, an amusement park between Cleveland and Kent, while water rides kept our kids, Alex and Rebecca, busy. Our new method for computing Gauss–Kronrod quadrature rules is based on formulas known from divide-and-conquer methods for the symmetric tridiagonal eigenvalue problem, and allows the computation of the nodes and weights of K_{2n+1} without forming S_{2n+1}. Bill Gragg got involved in this project because he had already written Matlab code for the particular divide-and-conquer method required in our method.

Laurie's algorithm, as well as our method, requires the Gauss–Kronrod rule to have $2n+1$ real distinct nodes. Ammar, Daniela Calvetti, and I later described a modification of Laurie's algorithm that allows complex conjugate nodes. There are some interesting open questions regarding Gauss–Kronrod rules with complex nodes. Maybe some day these questions will be investigated, e.g., in an amusement park.

Lothar Reichel
Kent, Ohio, USA

MATHEMATICS OF COMPUTATION
Volume 69, Number 231, Pages 1035–1052
S 0025-5718(00)01174-1
Article electronically published on February 17, 2000

COMPUTATION OF GAUSS-KRONROD QUADRATURE RULES

D. CALVETTI, G. H. GOLUB, W. B. GRAGG, AND L. REICHEL

ABSTRACT. Recently Laurie presented a new algorithm for the computation of $(2n+1)$-point Gauss-Kronrod quadrature rules with real nodes and positive weights. This algorithm first determines a symmetric tridiagonal matrix of order $2n+1$ from certain mixed moments, and then computes a partial spectral factorization. We describe a new algorithm that does not require the entries of the tridiagonal matrix to be determined, and thereby avoids computations that can be sensitive to perturbations. Our algorithm uses the consolidation phase of a divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. We also discuss how the algorithm can be applied to compute Kronrod extensions of Gauss-Radau and Gauss-Lobatto quadrature rules. Throughout the paper we emphasize how the structure of the algorithm makes efficient implementation on parallel computers possible. Numerical examples illustrate the performance of the algorithm.

1. INTRODUCTION

Let dw be a nonnegative measure on the real interval $[a, b]$ with an infinite number of points of increase, and such that the moments $\mu_k := \int_a^b x^k dw(x)$, $k = 0, 1, 2, \ldots$, exist and are bounded. For notational convenience, we assume that $\mu_0 = 1$. An n-point Gauss quadrature rule for the integral

$$\mathcal{I}f := \int_a^b f(x)dw(x) \tag{1.1}$$

is a formula of the form

$$\mathcal{G}_n f := \sum_{k=1}^{n} f(x_k)w_k \tag{1.2}$$

with the nodes $a < x_1 < x_2 < \cdots < x_n < b$ and positive weights w_k chosen so that

$$\mathcal{G}_n f = \mathcal{I}f \qquad \forall f \in \mathbb{P}_{2n-1}. \tag{1.3}$$

Received by the editor May 12, 1998.

1991 *Mathematics Subject Classification.* Primary 65D30, 65D32.

Key words and phrases. Jacobi matrix, inverse eigenvalue problem, divide-and-conquer algorithm, generalized Gauss-Kronrod rule.

Research of the first author was supported in part by NSF grants DMS-9404692 and DMS-9896073.

Research of the second author was supported in part by NSF grant CCR-9505393.

Research of the fourth author was supported in part by NSF grant DMS-9404706.

Here and throughout this paper $\mathbb{P}_j$ denotes the set of polynomials of degree at most j. The associated Gauss-Kronrod quadrature rule

$$\mathcal{K}_{2n+1}f := \sum_{k=1}^{2n+1} f(\tilde{x}_k)\tilde{w}_k \tag{1.4}$$

has the properties that

$$\{x_k\}_{k=1}^{n} \subset \{\tilde{x}_k\}_{k=1}^{2n+1} \tag{1.5}$$

and

$$\mathcal{K}_{2n+1}f = \mathcal{I}f \qquad \forall f \in \mathbb{P}_{3n+1}. \tag{1.6}$$

We present a new algorithm for the computation of Gauss-Kronrod quadrature rules with real nodes and positive weights when such rules exist. Our algorithm is based on recent results by Laurie [12] on properties of symmetric tridiagonal matrices associated with Gauss-Kronrod rules.

In typical applications of Gauss-Kronrod quadrature rules, both $\mathcal{G}_n f$ and $\mathcal{K}_{2n+1}f$ are evaluated, and this pair of approximations of $\mathcal{I}f$ is used to estimate the error in $\mathcal{G}_n f$. Applications in adaptive quadrature routines can be computationally demanding, and therefore it is important to develop accurate and fast algorithms that are well suited for implementation on a parallel computer for the computation of nodes and weights of Gauss-Kronrod rules; see [4, 8, 15] for recent discussions. Surveys of properties of Gauss-Kronrod quadrature rules are presented by Gautschi [7], Laurie [12] and Monegato [14]; see also Golub and Kautsky [9] for related discussions.

Let $\{p_j\}_{j=0}^{\infty}$ be a sequence of monic orthogonal polynomials with respect to the inner product

$$(f,g) := \int_a^b f(x)g(x)dw(x), \tag{1.7}$$

i.e.,

$$(p_j,p_k) = 0, \qquad j \neq k. \tag{1.8}$$

The p_j satisfy the recursion relations

$$\begin{aligned} p_{k+1}(x) &= (x-a_k)p_k(x) - b_k^2 p_{k-1}(x), \qquad k=1,2,\dots, \\ p_1(x) &:= x-a_0, \qquad p_0(x) := 1, \end{aligned} \tag{1.9}$$

with coefficients

$$a_k := \frac{(p_k,xp_k)}{(p_k,p_k)}, \qquad k=0,1,\dots, \tag{1.10}$$

$$b_k^2 := \frac{(p_k,p_k)}{(p_{k-1},p_{k-1})}, \qquad k=1,2,\dots. \tag{1.11}$$

Note that $(p_0,p_0) = \mu_0 = 1$. It follows from (1.11) that

$$(p_k,p_k) = b_k^2 b_{k-1}^2 \cdots b_1^2, \qquad k \geq 1. \tag{1.12}$$

Define the positive quantities $b_k := (b_k^2)^{1/2}$, $k \geq 1$. We refer to the a_k and b_k as recursion coefficients for the family of orthogonal polynomials (1.9). The $2n-1$

coefficients $\{a_k\}_{k=0}^{n-1}$ and $\{b_k\}_{k=1}^{n-1}$ determine the symmetric tridiagonal matrix

$$T_n := \begin{bmatrix} a_0 & b_1 & & & \\ b_1 & a_1 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-2} & a_{n-2} & b_{n-1} \\ & & & b_{n-1} & a_{n-1} \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{1.13}$$

with spectral factorization

(1.14)

$$T_n = W_n \Lambda_n W_n^T, \qquad \Lambda_n = \mathrm{diag}[\lambda_1, \lambda_2, \dots, \lambda_n], \qquad W_n W_n^T = I.$$

Due to the positivity of the off-diagonal entries b_k, the eigenvalues λ_j are distinct and all entries of the first row of W_n are nonvanishing. Moreover, it is well known that the nodes and weights of the Gauss rule (1.2) are given by

$$\begin{cases} x_j := \lambda_j, \\ w_j := (e_1^T W_n e_j)^2, \end{cases} \qquad 1 \le j \le n, \tag{1.15}$$

where e_j denotes the jth axis vector; see, e.g., [10]. We refer to the sets of eigenvalues and first or last components of normalized eigenvectors of a matrix as partial spectral resolution of the matrix. The sets $\{\lambda_j\}_{j=1}^n \cup \{e_1^T W_n e_j\}_{j=1}^n$ and $\{\lambda_j\}_{j=1}^n \cup \{e_n^T W_n e_j\}_{j=1}^n$ are partial spectral resolutions of T_n. We will assume that the recursion coefficients a_j and b_j are available. The nodes and weights (1.15) of the Gauss rule can then be computed in $\mathcal{O}(n^2)$ arithmetic operations by the Golub-Welsch algorithm [10].

Our algorithm for the computation of the nodes and weights of the Gauss-Kronrod rule (1.4) requires that the last entries of the normalized eigenvectors of T_n also be available. These can be computed.simultaneously with the Gauss weights by modifying the Golub-Welsch algorithm in a straightforward manner. The operation count for the modified algorithm is also $\mathcal{O}(n^2)$. The eigenvalues and first and last components of normalized eigenvectors can also conveniently be determined by one of the divide-and-conquer algorithms for the symmetric tridiagonal eigenproblem presented by Borges and Gragg [3] or Gu and Eisenstat [11]. These algorithms also require $\mathcal{O}(n^2)$ arithmetic operations, and with n processors the computations can be carried out in $\mathcal{O}(n)$ time steps.

Laurie [12] pointed out that if the Gauss-Kronrod rule (1.4) has distinct real nodes $\{\tilde{x}_k\}_{k=1}^{2n+1}$ and positive weights $\{\tilde{w}_k\}_{k=1}^{2n+1}$, then there is an associated symmetric tridiagonal matrix

(1.16)

$$\tilde{T}_{2n+1} := \begin{bmatrix} \tilde{a}_0 & \tilde{b}_1 & & & \\ \tilde{b}_1 & \tilde{a}_1 & \tilde{b}_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \tilde{b}_{2n-1} & \tilde{a}_{2n-1} & \tilde{b}_{2n} \\ & & & \tilde{b}_{2n} & \tilde{a}_{2n} \end{bmatrix} \in \mathbb{R}^{(2n+1)\times(2n+1)},$$

with spectral factorization

$$\tilde{T}_{2n+1} = \tilde{W}_{2n+1} \tilde{\Lambda}_{2n+1} \tilde{W}_{2n+1}^T, \ \ \tilde{\Lambda}_{2n+1} = \mathrm{diag}[\tilde{\lambda}_1, \tilde{\lambda}_2, \dots, \tilde{\lambda}_{2n+1}], \ \ \tilde{W}_{2n+1} \tilde{W}_{2n+1}^T = I,$$

such that

$$\left\{\begin{array}{l} \tilde{x}_j = \tilde{\lambda}_j, \\ \tilde{w}_j = (e_1^T \tilde{W}_{2n+1} e_j)^2, \end{array}\right. \qquad 1 \le j \le 2n+1, \tag{1.17}$$

analogously to (1.15). We refer to the matrix (1.16) as the Gauss-Kronrod matrix. Let the nodes be ordered according to $\tilde{x}_1 < \tilde{x}_2 < \cdots < \tilde{x}_{2n+1}$. Monegato [13] showed that the positivity of the weights $\tilde{w}_k$ associated with nodes $\tilde{x}_k \notin \{x_j\}_{j=1}^n$ is equivalent with the interlacing property

$$\tilde{x}_1 < \tilde{x}_2 = x_1 < \tilde{x}_3 < \tilde{x}_4 = x_2 < \tilde{x}_5 < \cdots < \tilde{x}_{2n} = x_n < \tilde{x}_{2n+1}.$$

Proposition 1.1 (Laurie [12]). *Let $\hat{T}_n$ and $\breve{T}_n$ denote the leading and trailing $n \times n$ principal submatrices of $\tilde{T}_{2n+1}$, respectively. Then $\hat{T}_n$ and $\breve{T}_n$ have the same eigenvalues. Moreover, for n odd,*

$$\tilde{a}_{j-1} = a_{j-1}, \qquad \tilde{b}_j = b_j, \qquad 1 \le j \le \frac{3n+1}{2}, \tag{1.18}$$

and, for n even,

$$\left\{\begin{array}{ll} \tilde{a}_j = a_j, & 0 \le j \le \frac{3n}{2}, \\ \tilde{b}_j = b_j, & 1 \le j \le \frac{3n}{2}. \end{array}\right. \tag{1.19}$$

Proof. Formulas (1.18) and (1.19) express that the first $3n+1$ coefficients of the matrices $\tilde{T}_{2n+1}$ and T_{2n+1} agree. This result follows immediately from (1.6). In particular, $\hat{T}_n = T_n$. This observation and the fact that $\{\lambda_j\}_{j=1}^n \subset \{\tilde{\lambda}_j\}_{j=1}^{2n+1}$ implies that $\hat{T}_n$ and $\breve{T}_n$ have the same spectrum, as can be seen by expanding $\det(\tilde{T}_{2n+1} - \lambda I)$ along the $(n+1)$st row; see [12] for details. □

It follows from Proposition 1.1 that the existence of a Gauss-Kronrod quadrature rule with real distinct nodes and positive weights is equivalent to the existence of a real solution to the following inverse eigenvalue problem.

Corollary 1.2. *Let the first $n-1$ entries of the $n \times n$ symmetric tridiagonal matrix*

$$\breve{T}_n := \begin{bmatrix} \tilde{a}_{n+1} & \tilde{b}_{n+2} & & & \\ \tilde{b}_{n+2} & \tilde{a}_{n+2} & \tilde{b}_{n+3} & & \\ & \ddots & \ddots & \ddots & \\ & & \tilde{b}_{2n-1} & \tilde{a}_{2n-1} & \tilde{b}_{2n} \\ & & & \tilde{b}_{2n} & \tilde{a}_{2n} \end{bmatrix} \tag{1.20}$$

be determined by (1.18) *when n is odd, and by* (1.19) *when n is even. Let the eigenvalues of $\breve{T}_n$ be the eigenvalues of the matrix* (1.13). *There is a real symmetric tridiagonal matrix $\breve{T}_n$ with these properties if and only if there is a* $(2n+1)$*-point Gauss-Kronrod quadrature rule* (1.4) *with real nodes and positive weights.*

Example 1.1. Let $n = 1$. The entries $\{\tilde{a}_j\}_{j=0}^1$ and $\{\tilde{b}_j\}_{j=1}^2$ of the Gauss-Kronrod matrix $\tilde{T}_3$ are recursion coefficients for the orthogonal polynomials associated with the measure dw. The entry marked by $*$ is not explicitly known,

$$\tilde{T}_3 := \begin{bmatrix} \tilde{a}_0 & \tilde{b}_1 & \\ \tilde{b}_1 & \tilde{a}_1 & \tilde{b}_2 \\ & \tilde{b}_2 & * \end{bmatrix}.$$

However, by Proposition 1.1, $\tilde{a}_2 = \tilde{a}_0$. In particular, any 3-point Gauss-Kronrod rule associated with a 1-point Gauss rule has real nodes and positive weights. □

Example 1.2. Let $n = 2$. The entries $\{\tilde{a}_j\}_{j=0}^{3}$ and $\{\tilde{b}_j\}_{j=1}^{3}$ of the Gauss-Kronrod matrix $\tilde{T}_5$ are recursion coefficients for orthogonal polynomials associated with the measure dw, but the entries marked by $*$ are not explicitly known,

$$(1.21) \qquad \tilde{T}_5 := \begin{bmatrix} \tilde{a}_0 & \tilde{b}_1 & & & \\ \tilde{b}_1 & \tilde{a}_1 & \tilde{b}_2 & & \\ & \tilde{b}_2 & \tilde{a}_2 & \tilde{b}_3 & \\ & & \tilde{b}_3 & \tilde{a}_3 & * \\ & & & * & * \end{bmatrix}.$$

By Proposition 1.1 the leading and trailing principal 2×2 submatrices of $\tilde{T}_5$ have the same trace. This yields the equation

$$(1.22) \qquad \tilde{a}_0 + \tilde{a}_1 = \tilde{a}_3 + \tilde{a}_4$$

for $\tilde{a}_4$. The determinants of the leading and trailing principal 2×2 submatrices are also the same, and this gives the equation

$$(1.23) \qquad \tilde{a}_0\tilde{a}_1 - \tilde{b}_1^2 = \tilde{a}_3\tilde{a}_4 - \tilde{b}_4^2$$

for $\tilde{b}_4$. When (1.23) is satisfied by a real positive value of $\tilde{b}_4$, a Gauss-Kronrod rule with real nodes and positive weights exists. □

Example 1.3. Let $[a, b] = [-1, 1]$ and $dw(x) := \frac{2}{\pi}(1 - x^2)^{1/2}dx$. Then the Gauss-Kronrod matrix (1.21) has the known entries

$$\tilde{T}_5 := \begin{bmatrix} 0 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ & 1/2 & 0 & 1/2 & \\ & & 1/2 & 0\text{-} & * \\ & & & * & * \end{bmatrix}.$$

Equations (1.22) and (1.23) yield $a_4 = 0$ and $b_4. = 1/2$. The eigenvalues and eigenveciors of this matrix are explicitly known, and we obtain the 5-point Gauss-Kronrod rule

$$\tilde{x}_k = \cos(\frac{\pi}{6}k), \qquad \tilde{w}_k = \frac{1}{3}\sin^2(\frac{\pi}{6}k), \qquad 1 \leq k \leq 5.$$

□

This paper describes a new algorithm for computing Gauss-Kronrod quadrature rules with real nodes and positive weights. The algorithm first determines the eigenvalues as well as the first and last components of normalized eigenvectors of the matrix T_n. This yields, in particular, the Gauss quadrature rule (1.15). The algorithm then proceeds by computing the first components of normalized eigenvectors of the matrix $\breve{T}_n$ defined in Proposition 1.1. This is described in Section 2. When n is even, we use a method proposed by Boley and Golub [2]. For n odd, we apply a closely related method. We remark that our algorithm does not explicitly determine the tridiagonal matrix (1.20). After these initial calculations, a consolidation step of the divide-and-conquer algorithm presented by Borges and Gragg [3] is used to determine the eigenvalues and first components of normalized eigenvectors of the matrix (1.16), and by (1.17) we obtain the Gauss-Kronrod rule. Relevant details of the divide-and-conquer method are discussed in Section 3. Our algorithm determines the $(2n + 1)$-point Gauss-Kronrod rule (1.17) from the recursion coefficients a_j and b_j in only $\mathcal{O}(n^2)$ arithmetic operations, and with n

processors only $\mathcal{O}(n)$ time steps are required. Section 4 describes how the algorithm of Section 3 can be applied to the computation of Gauss-Kronrod-Radau and Gauss-Kronrod-Lobatto rules. These rules are Kronrod extensions of Gauss-Radau and Gauss-Lobatto rules, respectively, and find application in adaptive composite quadrature rules. Section 5 contains numerical examples.

When only the measure dw but not the recursion coefficients a_j and b_j are available, the latter can be computed by (1.10) and (1.11). It may be attractive to evaluate necessary inner products by a Clenshaw-Curtis quadrature rule; see Gautschi [6] for a discussion.

Laurie [12] presented another algorithm for the computation of $(2n+1)$-point Gauss-Kronrod rules in $\mathcal{O}(n^2)$ arithmetic operations. This algorithm first determines certain mixed moments from which the symmetric tridiagonal matrix (1.16) is determined. The Gauss-Kronrod nodes and weights are then determined by applying the Golub-Welsch algorithm to the matrix (1.16).

Our algorithm avoids the explicit determination of the matrix (1.16). Experience from related problems indicates that the computation of the entries of $\tilde{T}_{2n+1}$ can be sensitive to round-off errors; see, e.g., [5].

2. Computation of eigenvector components of $\breve{T}_n$

We consider the determination of the first components of normalized eigenvectors of the real symmetric tridiagonal matrix (1.20), which is the trailing principal $n \times n$ submatrix of the Gauss-Kronrod matrix (1.16). The $n-1$ first entries of $\breve{T}_n$ are given by (1.18) or (1.19). The remaining diagonal and subdiagonal matrix entries are not known. The matrix (1.20) is required to have the same eigenvalues $\lambda_1 < \lambda_2 < \cdots < \lambda_n$ as T_n. We assume for the moment that such a real symmetric tridiagonal matrix $\breve{T}_n$ with positive subdiagonal elements exists.

We first outline a method due to Boley and Golub [2] that can be applied when n is even. A modification of this method, described below, can be used for n odd.

Recall that the matrix T_n is associated with a positive measure dw with support in a real interval $[a, b]$ and with the quadrature rule (1.15). Similarly, we may associate with the matrix $\breve{T}_n$ a nonnegative measure $d\breve{w}$ with support in a real interval $[\breve{a}, \breve{b}]$ and such that $\int_{\breve{a}}^{\breve{b}} d\breve{w}(x) = 1$. The eigenvalues λ_j and squares of the first components of normalized eigenvectors $\breve{w}_j$ define a quadrature rule $\{\lambda_j, \breve{w}_j\}_{j=1}^n$ associated with the matrix $\breve{T}_n$, such that

$$(2.1) \qquad \int_{\breve{a}}^{\breve{b}} x^k d\breve{w} = \sum_{j=1}^{n} \lambda_j^k \breve{w}_j, \qquad 0 \le k < n.$$

We remark that we may choose $d\breve{w}$ to be the discrete measure defined by the quadrature rule $\{\lambda_j, \breve{w}_j\}_{j=1}^n$.

Let n be even. Then the entries of the leading principal submatrix $\breve{T}_{n/2} \in \mathbb{R}^{n/2 \times n/2}$ are explicitly known. Let $\{x_j^*, w_j^*\}_{j=1}^{n/2}$ be the Gauss quadrature rule associated with the matrix $\breve{T}_{n/2}$, i.e., the x_j^* are eigenvalues and the w_j^* are the square of the first components of normalized eigenvectors of $T_{n/2}$; cf. (1.15). Both quadrature rules $\{x_j^*, w_j^*\}_{j=1}^{n/2}$ and $\{\lambda_j, \breve{w}_j\}_{j=1}^n$ can be regarded as discretizations of the measure $d\breve{w}$. Thus,

$$\int_{\breve{a}}^{\breve{b}} x^k d\breve{w}(x) = \sum_{j=1}^{n/2} (x_j^*)^k w_j^* = \sum_{j=1}^{n} \lambda_j^k \breve{w}_j, \qquad 0 \leq k < n. \tag{2.2}$$

The equations (2.2) can be expressed in terms of the Lagrange polynomials

$$\ell_k(x) := \prod_{\substack{j=1 \\ j\neq k}}^{n} \frac{x-\lambda_j}{\lambda_k - \lambda_j}, \qquad 1 \leq k \leq n, \tag{2.3}$$

and we obtain

$$\sum_{j=1}^{n/2} \ell_k(x_j^*) w_j^* = \sum_{j=1}^{n} \ell_k(\lambda_j)\breve{w}_j = \breve{w}_k, \qquad 1 \leq k \leq n. \tag{2.4}$$

We remark that the equations (2.2) can be formulated as a linear system of equations with a Vandermonde matrix for the weights $\breve{w}_j$. Numerical experiments reported in [5] indicate that the weights $\breve{w}_j$ are computed more accurately by formula (2.4).

We assumed above that a real symmetric tridiagonal matrix $\breve{T}_n$ with positive subdiagonal elements, with given spectrum $\{\lambda_j\}_{j=1}^n$ and with a given leading $n/2 \times n/2$ principal submatrix exists. However, this is not always the case. For instance, when $dw(x) = e^{-x}dx$ and $[a,b] = [0,\infty]$, the matrix $\breve{T}_n$ is for many values of n complex symmetric, with real diagonal entries and at least one purely imaginary subdiagonal element. The measure $d\breve{w}$ associated with such a matrix $\breve{T}_n$ is indefinite, and at least one weight $\breve{w}_j$ is negative. A numerical method for computing complex symmetric tridiagonal matrices of even order n with real diagonal entries and real or purely imaginary subdiagonal entries, given its real distinct eigenvalues and its real symmetric tridiagonal leading principal submatrix of order $n/2$, is described in [5].

The present paper is concerned with the computation of Gauss-Kronrod rules with distinct real nodes and positive weights, and by Corollary 1.2 we may restrict our attention to real symmetric tridiagonal matrices $\breve{T}_n$ with positive subdiagonal entries. In particular, we are only concerned with the case when the weights $\breve{w}_j$ are positive.

When n is odd, the algorithm described above has to be modified. The entries $\{\tilde{a}_j\}_{j=n+1}^{(3n-1)/2}$ and $\{\tilde{b}_j\}_{j=n+2}^{(3n+1)/2}$ of $\breve{T}_n$ are known. The largest leading $k \times k$ principal submatrix of $\breve{T}_n$ with all entries explicitly known is of order $k = (n-1)/2$, and the Gauss rule associated with this submatrix is not of high enough order to allow the matching of n moments, analogously to (2.2). Therefore a formula similar to (2.4) cannot be applied before some preliminary calculations have been carried out.

The computations are divided into two steps. First we compute the diagonal entry $\tilde{a}_{(3n+1)/2}$. Then the leading principal submatrix of order $(n+1)/2$ of $\breve{T}_n$ is known, and we can compute the weights $\breve{w}_j$ by a formula analogous to (2.4).

Let $\{\breve{p}_j\}_{j=0}^{(n-1)/2}$ be the first $(n+1)/2$ monic orthogonal polynomials associated with the inner product

$$\langle f, g\rangle := \int_{\breve{a}}^{\breve{b}} f(x)g(x)d\breve{w}(x).$$

These polynomials can be computed from the available recursion coefficients. The desired diagonal entry of $\breve{T}_n$ is given by

$$\tilde{a}_{(3n+1)/2} = \frac{\langle \breve{p}_{(n-1)/2}, x\breve{p}_{(n-1)/2} \rangle}{\langle \breve{p}_{(n-1)/2}, \breve{p}_{(n-1)/2} \rangle},$$

where

$$\langle \breve{p}_{(n-1)/2}, \breve{p}_{(n-1)/2} \rangle = \tilde{b}^2_{(3n+1)/2} \tilde{b}^2_{(3n-1)/2} \cdots \tilde{b}^2_{n+2}.$$

Note that

$$\breve{p}_n(x) = \prod_{j=1}^{n} (x - \lambda_j) \tag{2.5}$$

and that $x\breve{p}^2_{(n-1)/2}(x) - \breve{p}_n(x) \in \mathbb{P}_{n-1}$. The latter polynomial can be written as

$$x\breve{p}^2_{(n-1)/2}(x) - \breve{p}_n(x) = \breve{p}_{(n-1)/2}(x) \sum_{j=0}^{(n-1)/2} c_{(n-1)/2+j} \breve{p}_j(x) + \sum_{j=0}^{(n-3)/2} c_j \breve{p}_j(x) \tag{2.6}$$

for certain coefficients c_k. Integrating (2.6) and using the orthogonality of the polynomials $\breve{p}_j$ yields

$$\begin{aligned} \langle \breve{p}_{(n-1)/2}, x\breve{p}_{(n-1)/2} \rangle &= \langle \breve{p}_{(n-1)/2}, \sum_{j=0}^{(n-1)/2} c_{(n-1)/2+j} \breve{p}_j \rangle + \langle \sum_{j=0}^{(n-3)/2} c_j \breve{p}_j, \breve{p}_0 \rangle \\ &= c_{n-1} \langle \breve{p}_{(n-1)/2}, \breve{p}_{(n-1)/2} \rangle + c_0 \langle \breve{p}_0, \breve{p}_0 \rangle, \end{aligned}$$

and therefore

$$\tilde{a}_{(3n+1)/2} = c_{n-1} + \frac{c_0}{\tilde{b}^2_{(3n+1)/2} \tilde{b}^2_{(3n-1)/2} \cdots \tilde{b}^2_{n+2}}. \tag{2.7}$$

It remains to determine the coefficients c_{n-1} and c_0. Note that c_{n-1} is the leading coefficient of the polynomial $x\breve{p}^2_{(n-1)/2}(x) - \breve{p}_n(x)$ in power form. Straightforward expansion in terms of powers of x yields

$$x\breve{p}^2_{(n-1)/2}(x) = x^n - \left(2 \sum_{j=n+1}^{(3n-1)/2} \tilde{a}_j \right) x^{n-1} + \mathcal{O}(x^{n-2})$$

and

$$\breve{p}_n(x) = x^n - \left(\sum_{j=1}^{n} \lambda_j \right) x^{n-1} + \mathcal{O}(x^{n-2}).$$

Therefore

$$x\breve{p}^2_{(n-1)/2}(x) - \breve{p}_n(x) = \left(\sum_{j=1}^{n} \lambda_j - 2 \sum_{j=n+1}^{(3n-1)/2} \tilde{a}_j \right) x^{n-1} + \mathcal{O}(x^{n-2}).$$

Comparison with (2.6) shows that

$$c_{n-1} = \sum_{j=1}^{n} \lambda_j - 2 \sum_{j=n+1}^{(3n-1)/2} \tilde{a}_j. \tag{2.8}$$

We turn to the computation of the coefficient c_0. Determine the Gauss quadrature rule $\{x_j^*, w_j^*\}_{j=1}^{(n-1)/2}$ associated with the leading principal $\frac{n-1}{2} \times \frac{n-1}{2}$ submatrix of $\breve{T}_n$, all of whose entries are known. We then apply this quadrature rule to the right-hand side and left-hand side of (2.6) to obtain

$$c_0 = -\sum_{j=1}^{(n-1)/2} \breve{p}_n(x_j^*)w_j^*, \tag{2.9}$$

where $\breve{p}_n$ is given by (2.5). In the derivation of (2.9), we have used that the nodes x_j^* are the zeros of $\breve{p}_{(n-1)/2}$, and that by orthogonality

$$\sum_{j=1}^{(n-1)/2} \breve{p}_k(x_j^*)w_j^* = 0, \qquad 1 \le k \le \frac{n-3}{2}.$$

Thus, we can evaluate the coefficient $\tilde{a}_{(3n+1)/2}$ by using formulas (2.7)–(2.9).

The leading principal $\frac{n+1}{2} \times \frac{n+1}{2}$ submatrix of $\breve{T}_n$ is now explicitly known, and we can determine the weights $\breve{w}_j$ analogously as when n is even. Thus, compute the Gauss rule $\{x_j', w_j'\}_{j=1}^{(n+1)/2}$ associated with the leading principal submatrix of $\breve{T}_n$ of order $(n+1)/2$. Analogously to (2.2), we obtain

$$\int_{\breve{a}}^{\breve{b}} x^k d\breve{w}(x) = \sum_{j=1}^{(n+1)/2} (x_j')^k w_j' = \sum_{j=1}^{n} \lambda_j^k \breve{w}_j, \qquad 0 \le k < n,$$

which, similarly to (2.4), yields the formula for the weights

$$\breve{w}_k = \sum_{j=1}^{(n+1)/2} \ell_k(x_j')w_j', \qquad 1 \le k \le n,$$

where the Lagrange polynomials ℓ_k are given by (2.3).

The computations described in this section require $\mathcal{O}(n^2)$ arithmetic operations and can be carried out in $\mathcal{O}(n)$ time steps by n processors when a divide-and-conquer method is used for computing the required quadrature rules.

3. Computation of Gauss-Kronrod rules

We assume in this section that the eigenvalues and the last components of normalized eigenvectors of the matrix T_n, as well as the first components $\{\breve{w}_j^{1/2}\}_{j=1}^n$ of normalized eigenvectors of the matrix $\breve{T}_n$, are available. The computation of these quantities is discussed in the previous sections. Recall that the matrices T_n and $\breve{T}_n$ have the same eigenvalues. We are now in a position to apply the consolidation phase of the divide-and-conquer algorithm described in [3] to determine the eigenvalues and first components of normalized eigenvectors of $\tilde{T}_{2n+1}$. The associated Gauss-Kronrod rule is then obtained from (1.17).

The Gauss-Kronrod matrix (1.16) can be written as

$$\tilde{T}_{2n+1} := \begin{bmatrix} T_n & e_n b_n & \\ b_n e_n^T & a_n & b_{n+1}e_1^T \\ & e_1 b_{n+1} & \breve{T}_n \end{bmatrix} \tag{3.1}$$

The matrix $\breve{T}_n$ has the spectral factorization

$$\breve{T}_n = \breve{W}_n \Lambda_n \breve{W}_n^T, \qquad \breve{W}_n \breve{W}_n^T = I,$$

where Λ_n is defined by (1.14). Introduce

$$\tilde{U} := \begin{bmatrix} W_n & & \\ & 1 & \\ & & \breve{W}_n \end{bmatrix} \in \mathbb{R}^{(2n+1)\times(2n+1)}.$$

Then

$$\tilde{U}^T(\tilde{T}_{2n+1} - \lambda I)\tilde{U} = \begin{bmatrix} \Lambda_n - \lambda I & W_n^T \boldsymbol{e}_n b_n & \\ b_n \boldsymbol{e}_n^T W_n & a_n - \lambda & b_{n+1}\boldsymbol{e}_1^T \breve{W}_n \\ & \breve{W}_n^T \boldsymbol{e}_1 b_{n+1} & \Lambda_n - \lambda I \end{bmatrix}. \tag{3.2}$$

Note that the entries of the vectors $\boldsymbol{e}_n^T W_n$ and $\boldsymbol{e}_1^T \breve{W}_n$ are known. The matrix on the right-hand side is the sum of a diagonal matrix and a Swiss cross

$$\tilde{U}^T(\tilde{T}_{2n+1} - \lambda I)\tilde{U} = \begin{bmatrix} x & & & & \\ & x & & & \\ & & x & & \\ & & & x & \\ & & & & x \end{bmatrix} + \begin{bmatrix} & & x & & \\ & & x & & \\ x & x & x & x & x \\ & & x & & \\ & & x & & \end{bmatrix},$$

which we permute to an arrow matrix by a similarity transformation with the permutation matrix $P^{(n+1)} = [\boldsymbol{e}_1, \boldsymbol{e}_2, \dots, \boldsymbol{e}_n, \boldsymbol{e}_{n+2}, \dots, \boldsymbol{e}_{2n}, \boldsymbol{e}_{n+1}] \in \mathbb{R}^{(2n+1)\times(2n+1)}$. Thus,

$$\begin{aligned} (P^{(n+1)})^T \tilde{U}^T (\tilde{T}_{2n+1} \quad &- \quad \lambda I)\tilde{U}P^{(n+1)} \\ &= \begin{bmatrix} \Lambda_n & & W_n^T \boldsymbol{e}_n b_n \\ & \breve{\Lambda}_n & \breve{W}_n^T \boldsymbol{e}_1 b_{n+1} \\ b_n \boldsymbol{e}_n^T W_n & b_{n+1}\boldsymbol{e}_1^T \breve{W}_n & a_n \end{bmatrix} - \lambda I. \end{aligned} \tag{3.3}$$

We apply rotation similarity transformations to rows j and $j+n$, for $j = 1, 2, \dots, n$, in order to annihilate the first n entries of row and column $2n+1$. This process is sometimes referred to as combo-deflation and yields

$$G^T(P^{(n+1)})^T \tilde{U}^T (\tilde{T}_{2n+1} - \lambda I)\tilde{U}P^{(n+1)}G = \begin{bmatrix} \Lambda_n & & \\ & \Lambda_n & \boldsymbol{c} \\ & \boldsymbol{c}^T & a_n \end{bmatrix} - \lambda I, \tag{3.4}$$

where the matrix $G \in \mathbb{R}^{(2n+1)\times(2n+1)}$ is made up of the product of the n rotations applied to the matrix (3.3) from the right, and the vector $\boldsymbol{c} = [\gamma_1, \gamma_2, \dots, \gamma_n] \in \mathbb{R}^n$ consists of the entries in positions $n+1$ through $2n$ of the vector

$$(G^T[b_n \boldsymbol{e}_n^T W_n, b_{n+1}\boldsymbol{e}_1^T \breve{W}_n, a_n])^T.$$

The right-hand side of (3.4) shows that the matrix $\tilde{T}_{2n+1}$ has the diagonal entries of Λ_n as eigenvalues, and these are the nodes of the Gauss rule (1.2). Thus, the computed nodes of the Gauss and Gauss-Kronrod quadrature rules satisfy (1.5).

The remaining $n+1$ eigenvalues of $\tilde{T}_{2n+1}$ are eigenvalues of the trailing principal $(n+1) \times (n+1)$ submatrix of the matrix (3.4), which for $\lambda \notin \{x_j\}_{j=1}^n$ can be factored according to

$$\begin{bmatrix} \Lambda_n & \boldsymbol{c} \\ \boldsymbol{c}^T & a_n \end{bmatrix} - I\lambda = \begin{bmatrix} I & \boldsymbol{0} \\ \boldsymbol{c}^T(\Lambda_n - \lambda I)^{-1} & 1 \end{bmatrix} \begin{bmatrix} \Lambda_n - \lambda I & \boldsymbol{c} \\ \boldsymbol{0}^T & -f(\lambda) \end{bmatrix},$$

where f is the spectral function

$$f(\lambda) := \lambda - a_n + \sum_{j=1}^{n} \frac{\gamma_j^2}{x_j - \lambda}. \tag{3.5}$$

The $n+1$ distinct zeros of f are the Gauss-Kronrod nodes $\{\tilde{x}_{2j-1}\}_{j=1}^{n+1}$, and they interlace the Gauss nodes $\{x_j\}_{j=1}^{n}$. A zero finder that yields sequences of cubically and monotonically convergent approximations of the zeros of f is described in [3]. The computation of the eigenvalues requires $\mathcal{O}(n^2)$ arithmetic operations. Given the eigenvalues of T_{2n+1}, the first components of normalized eigenvectors also can be determined in $\mathcal{O}(n^2)$ arithmetic operations; see [3] for details. These components are computed by an approach suggested by Gu and Eisenstat; see [11] and references therein. Only $\mathcal{O}(n)$ time steps are required when n processors are available.

The computations required to compute an n-point Gauss quadrature rule and the associated $(2n+1)$-point Gauss-Kronrod rule are summarized in the following algorithm.

Algorithm 1. Computation of Gauss and Gauss-Kronrod rules.

- **Input**: n, first $3n+1$ recursion coefficients $a_0, b_1, a_1, b_2, \ldots$ for orthogonal polynomials associated with a positive measure dw scaled so that $\mu_0 = 1$.
- **Output**: n-point Gauss rule $\{x_j, w_j\}_{j=1}^{n}$ and associated $(2n+1)$-point Gauss-Kronrod rule $\{\tilde{x}_j, \tilde{w}_j\}_{j=1}^{2n}$.
- Compute eigenvalues as well as first and last components of normalized eigenvectors of the tridiagonal matrix T_n. The eigenvalues and first components of the eigenvectors yield the Gauss quadrature rule $\{x_j, w_j\}_{j=1}^{n}$ associated with the measure dw.
- Compute weights $\{\breve{w}_j\}_{j=1}^{n}$ of trailing $n \times n$ principal submatrix $\breve{T}_n$ as described in Section 2.
- The entries of row and column $n+1$ of the matrix $\tilde{T}_{2n}$, the eigenvalues and last component of normalized eigenvectors of T_n and the square-root of the weights $\{\breve{w}_j\}_{j=1}^{n}$ are used to compute the Gauss-Kronrod rule $\{\tilde{x}_j, \tilde{w}_j\}_{j=1}^{2n}$ by application of a consolidation step of a divide-and-conquer algorithm. □

4. Generalized Gauss-Kronrod quadrature rules

This section discusses the computation of Gauss-Kronrod rules with one or two preassigned nodes. We refer to these quadrature rules as Gauss-Kronrod-Radau and Gauss-Kronrod-Lobatto rules, respectively. Properties of these rules are discussed by Gautschi [7].

4.1. **Gauss-Kronrod-Radau rules.** Let dw be the nonnegative measure introduced in Section 1. An $(n+1)$-point Gauss-Radau quadrature rule for the integral (1.1) with a fixed node at $x = a$ is a formula of the form

$$\mathcal{G}_{n+1,a} f := \sum_{k=0}^{n} f(x_{k,a}) w_{k,a} \tag{4.1}$$

with nodes $a = x_{0,a} < x_{1,a} < \cdots < x_{n,a} < b$ and positive weights $w_{k,a}$ chosen so that

$$\mathcal{G}_{n+1,a} f = \mathcal{I} f \qquad \forall f \in \mathbb{P}_{2n}.$$

The associated Gauss-Kronrod-Radau quadrature rule

$$\mathcal{K}_{2n+2,a}f := \sum_{k=0}^{2n+1} f(\tilde{x}_{k,a})\tilde{w}_{k,a} \tag{4.2}$$

has the properties that

$$\{x_{k,a}\}_{k=0}^{n} \subset \{\tilde{x}_{k,a}\}_{k=0}^{2n+1}$$

and

$$\mathcal{K}_{2n+2,a}f = \mathcal{I}f \qquad \forall f \in \mathbb{P}_{3n+2}. \tag{4.3}$$

In addition, we would like the weights $\tilde{w}_{k,a}$ to be positive and the nodes $\tilde{x}_{k,a}$ to satisfy $a = \tilde{x}_{0,a} < \tilde{x}_{1,a} < \cdots < \tilde{x}_{2n+1,a}$. The "free" nodes $\{x_{k,a}\}_{k=1}^{n}$ of the Gauss-Radau rule (4.1) are zeros of the nth degree orthogonal polynomial associated with the measure

$$dw'(x) := (x-a)dw(x), \qquad a \le x \le b; \tag{4.4}$$

see, e.g., [9]. Analogously, Gautschi [7] showed that the nodes $\{\tilde{x}_{k,a}\}_{k=1}^{2n+1}$ of the Gauss-Kronrod-Radau rule (4.2) are nodes of a $(2n+1)$-point Gauss-Kronrod quadrature rule

$$\mathcal{K}'_{2n+1}f = \sum_{k=1}^{2n+1} f(\tilde{x}'_k)\tilde{w}'_k \tag{4.5}$$

associated with the measure (4.4). We apply Algorithm 1 to compute the nodes $\tilde{x}'_k$ and weights $\tilde{w}'_k$ of (4.5) and thereby the nodes $\tilde{x}_{k,a}$ of the Gauss-Kronrod-Radau rule (4.2). The following proposition shows how to compute the weights $\tilde{w}_{k,a}$ of (4.2) from the weights $\tilde{w}'_k$.

Proposition 4.1. *Let* $\{\tilde{w}'_k\}_{k=1}^{2n+1}$ *be the weights of the Gauss-Kronrod quadrature rule* (4.5) *associated with the measure* (4.4). *The weights* $\tilde{w}_{k,a}$ *of the* $(2n+2)$*-point Gauss-Kronrod-Radau rule* (4.2) *associated with the measure* dw *are given by*

$$\begin{aligned} \tilde{w}_{k,a} &= \frac{\tilde{w}'_k}{\tilde{x}_{k,a}-a}, \qquad 1 \le k \le 2n+1, \\ \tilde{w}_{0,a} &= \mu_0 - \sum_{k=1}^{2n+1} \tilde{w}_{k,a}. \end{aligned} \tag{4.6}$$

Proof. Introduce the Lagrange polynomials

$$\ell_k(x) := \prod_{\substack{j=1 \\ j\neq k}}^{2n+1} \frac{x-\tilde{x}_{j,a}}{\tilde{x}_{k,a}-\tilde{x}_{j,a}}$$

and

$$\ell_{k,a}(x) := \ell_k(x)\frac{x-a}{\tilde{x}_{k,a}-a},$$

for $1 \le k \le 2n+1$. It follows from (4.3)–(4.5) that

$$\begin{aligned} \tilde{w}_{k,a} &= \mathcal{K}_{2n+2,a}\ell_{k,a} = \int_a^b \ell_{k,a}(x)dw(x) \\ &= \frac{1}{\tilde{x}_{k,a}-a}\int_a^b \ell_k(x)dw'(x) = \frac{1}{\tilde{x}_{k,a}-a}\mathcal{K}'_{2n+1}\ell_k = \frac{\tilde{w}'_k}{\tilde{x}_{k,a}-a}, \end{aligned}$$

for $1 \le k \le 2n+1$. The formula for $\tilde{w}_{0,a}$ follows from $\sum_{k=0}^{2n+1}\tilde{w}_{k,a} = \int_a^b dw(x)$. □

Algorithm 1 requires the first $3n+1$ recursion coefficients $a'_0, b'_1, a'_1, \ldots$ for the orthogonal polynomials with respect to the measure (4.4). When dw is a measure of Jacobi-type

$$dw(x) := c_0(b-x)^{\alpha}(x-a)^{\beta}dx, \qquad a < x < b, \qquad \alpha, \beta > -1, \tag{4.7}$$

then so is dw', and explicit formulas for the recursion coefficients a'_j and b'_j are available; see, e.g., [16]. The scaling factor c_0, where

$$c_0^{-1} := (b-a)^{\alpha+\beta+1}B(\alpha+1, \beta+1) \tag{4.8}$$

and B denotes the beta function, secures that $\mu_0 = 1$.

When dw is not of Jacobi-type and recursion coefficients a_j and b_j for orthogonal polynomials associated with the measure dw are available, a scheme by Golub and Kautsky [9] can be used to compute recursion coefficients a'_j and b'_j for orthogonal polynomials associated with the measure dw'. Let the symmetric tridiagonal matrix $T_m \in \mathbb{R}^{m \times m}$ be defined by the first $2m-1$ recursion coefficients a_j and b_j given by (1.10)–(1.11); cf. (1.13). Compute the Choleski factorization

$$T_m - aI = L_m L_m^T. \tag{4.9}$$

Then the matrix

(4.10)

$$T'_m := \begin{bmatrix} a'_0 & b'_1 & & & \\ b'_1 & a'_1 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b'_{m-2} & a'_{m-2} & b'_{m-1} \\ & & & b'_{m-1} & a'_{m-1} \end{bmatrix} := L_m^T L_m + aI + \gamma_m e_m e_m^T,$$

where $\gamma_m := b_m^2/(e_m^T L_m e_m)^2$, contains the first $2m-1$ recursion coefficients for orthogonal polynomials associated with the measure dw'; see [9, Theorem 3]. The coefficients a'_j and b'_j are used as input for Algorithm 1.

4.2. **Gauss-Kronrod-Lobatto rules.** Let dw be the nonnegative measure introduced in Section 1. An $(n+2)$-point Gauss-Lobatto quadrature rule for the integral (1.1) with fixed nodes at $x = a$ and $x = b$ is a formula of the form

$$\mathcal{G}_{n+2,a,b}f := \sum_{k=0}^{n+1} f(x_{k,a,b})w_{k,a,b} \tag{4.11}$$

with nodes $a = x_{0,a,b} < x_{1,a,b} < \cdots < x_{n,a,b} < x_{n+1,a,b} = b$ and positive weights $w_{k,a,b}$ chosen so that

$$\mathcal{G}_{n+2,a,b}f = \mathcal{I}f \qquad \forall f \in \mathbb{P}_{2n+1}.$$

The associated Gauss-Kronrod-Lobatto quadrature rule

$$\mathcal{K}_{2n+3,a,b}f := \sum_{k=0}^{2n+2} f(\tilde{x}_{k,a,b})\tilde{w}_{k,a,b} \tag{4.12}$$

has the properties that

$$\{x_{k,a,b}\}_{k=0}^{n+1} \subset \{\tilde{x}_{k,a,b}\}_{k=0}^{2n+1}$$

and

$$\mathcal{K}_{2n+3,a,b}f = \mathcal{I}f \qquad \forall f \in \mathbb{P}_{3n+3}. \tag{4.13}$$

We would like the weights $\tilde{w}_{k,a,b}$ to be positive and the nodes $\tilde{x}_{k,a,b}$ to satisfy

$$a = \tilde{x}_{0,a,b} < \tilde{x}_{1,a,b} < \cdots < \tilde{x}_{2n+1,a,b} < \tilde{x}_{2n+2,a,b} = b.$$

The "free" nodes $\{x_{k,a,b}\}_{k=1}^{n}$ of the Gauss-Lobatto rule (4.11) are zeros of the nth degree orthogonal polynomial associated with the measure

$$dw''(x) := (b-x)(x-a)dw(x), \qquad a \leq x \leq b; \tag{4.14}$$

see [9]. Analogously, Gautschi [7] showed that the nodes $\{\tilde{x}_{k,a,b}\}_{k=1}^{2n+1}$ of the Gauss-Kronrod-Lobatto rule (4.12) are nodes of a $(2n+1)$-point Gauss-Kronrod quadrature rule

$$\mathcal{K}''_{2n+1}f = \sum_{k=1}^{2n+1} f(\tilde{x}_{k,a,b})\tilde{w}''_k \tag{4.15}$$

associated with the measure (4.14). We apply Algorithm 1 to compute the nodes $\tilde{x}_{k,a,b}$ and weights $\tilde{w}''_k$ of (4.15) and thereby the nodes of the Gauss-Kronrod-Lobatto rule (4.12). The following proposition shows how to compute the weights $\tilde{w}_{k,a,b}$ of (4.12) from the weights $\tilde{w}''_k$.

Proposition 4.2. *Let $\{\tilde{w}''_k\}_{k=1}^{2n+1}$ be the weights of the Gauss-Kronrod quadrature rule (4.15) associated with the measure (4.14). The weights $\tilde{w}_{k,a,b}$ of the $(2n+3)$-point Gauss-Kronrod-Lobatto rule (4.12) associated with the measure dw are given by*

$$\tilde{w}_{k,a,b} = \frac{\tilde{w}''_k}{(\tilde{x}_{k,a,b}-a)(b-\tilde{x}_{k,a,b})}, \qquad 1 \leq k \leq 2n+1, \tag{4.16}$$

$$\tilde{w}_{0,a,b} = \frac{1}{b-a}\left(b\mu_0 - \mu_1 + \sum_{k=1}^{2n+1} \tilde{w}_{k,a,b}(\tilde{x}_{k,a,b}-b)\right), \tag{4.17}$$

$$\tilde{w}_{2n+2,a,b} = \frac{1}{b-a}\left(\mu_1 - a\mu_0 - \sum_{k=1}^{2n+1} \tilde{w}_{k,a,b}(\tilde{x}_{k,a,b}-a)\right). \tag{4.18}$$

Proof. Formula (4.16) can be shown similarly to (4.6). Integration of $x-b$ by the rule (4.12) yields

$$\mu_1 - b\mu_0 = \int_a^b (x-b)dw(x) = \sum_{k=0}^{2n+1} \tilde{w}_{k,a,b}(\tilde{x}_{k,a,b}-b),$$

from which (4.17) follows. Similarly, (4.18) is obtained by integrating $x-a$. □

Algorithm 1 requires the $3n+1$ first recursion coefficients $a''_0, b''_1, a''_1, \ldots$ for the orthogonal polynomials with respect to the measure (4.14). When dw is a measure of Jacobi-type (4.7), explicit formulas for these recursion coefficients are available. Otherwise, we can can determine the coefficients a''_j and b''_j from the matrix (4.10) as follows. Compute the Choleski factorization

$$bI - T'_m = L'_m(L'_m)^T.$$

Then the matrix

$$T''_m := \begin{bmatrix} a''_0 & b''_1 & & & \\ b''_1 & a''_1 & b''_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b''_{m-2} & a''_{m-2} & b''_{m-1} \\ & & & b''_{m-1} & a''_{m-1} \end{bmatrix}$$

$$:= bI - (L'_m)^T L'_m + aI + \gamma'_m e_m e_m^T,$$

where $\gamma'_m := (b'_m)^2/(e_m^T L_m e_m)^2$, contains the first $2m-1$ recursion coefficients for orthogonal polynomials associated with the measure dw''; see Golub and Kautsky [9] for details.

5. Numerical examples

The computations were carried out on an HP 9000 workstation in double precision arithmetic, i.e., with almost 16 significant digits, and in quadruple precision arithmetic. A Matlab implementation of our divide-and-conquer based algorithm is referred to as "d+c based alg." in the tables. This implementation uses double precision arithmetic and is compared to a Fortran implementation using double precision arithmetic of the algorithm presented by Laurie [12]. Laurie's algorithm is referred to as "mixed moment alg." in the tables. We used a QR algorithm from LAPACK [1] to compute the Gauss-Kronrod rule from the matrix (1.16) determined by Laurie's algorithm. A Fortran implementation in quadruple precision arithmetic of Laurie's algorithm and the QR algorithm were used to compute highly accurate Gauss-Kronrod quadrature rules. The nodes and weights computed in quadruple precision were considered exact, and were used to determine the error in the quadrature rules computed by our and Laurie's algorithms in double precision arithmetic.

In our experiments we computed Gauss-Kronrod and Gauss-Kronrod-Radau quadrature rules associated with Jacobi measures

(5.1)
$$dw(x) := c_0(1-x)^\alpha(1+x)^\beta dx, \qquad -1 < x < 1, \qquad \alpha, \beta > -1,$$

where the scaling factor c_0, given by (4.8) with $a = -1$ and $b = 1$, is chosen to make $\mu_0 = 1$. Recursion coefficients for the associated orthogonal polynomials are explicitly known; see, e.g., [16].

TABLE 5.1. Errors in computed Gauss-Kronrod weights

n	α	β	d + c based alg.		mixed moment based alg.	
			max abs. error	max rel. error	max abs. error	max rel. error
10	-0.20	-0.99	8.68 E-16	4.42 E-15	1.29 E-14	2.36 E-13
10	-0.70	1.00	4.18 E-15	2.52 E-14	3.67 E-14	2.61 E-13
15	-0.97	-0.97	3.24 E-14	4.62 E-13	1.35 E-13	2.25 E-12
15	-0.99	-0.50	1.20 E-14	2.16 E-13	1.56 E-13	2.81 E-12
20	-0.60	-0.90	2.38 E-14	1.91 E-13	2.33 E-13	1.87 E-12
20	-0.99	-0.90	4.59 E-15	4.19 E-13	1.21 E-13	2.36 E-12

TABLE 5.2. Errors in computed Gauss-Kronrod nodes

n	α	β	d + c based alg. max abs. error	 max rel. error	mixed moment based alg. max abs. error	 max rel. error
10	-0.20	-0.99	5.86 E-16	4.65 E-15	2.50 E-15	2.50 E-15
10	-0.70	1.00	5.46 E-16	7.71 E-15	5.45 E-15	5.65 E-15
15	-0.97	-0.97	1.07 E-15	4.08 E-15	1.26 E-15	4.63 E-15
15	-0.99	-0.50	7.12 E-16	4.74 E-15	4.59 E-15	1.75 E-14
20	-0.60	-0.90	1.24 E-15	2.85 E-15	3.68 E-15	5.00 E-14
20	-0.99	-0.90	1.83 E-15	1.66 E-14	3.64 E-15	2.36 E-14

TABLE 5.3. Errors in computed Gauss-Kronrod-Radau weights, $\alpha = -0.99$, $\beta = -0.9$, fixed node at $x = -1$

n	d + c based alg. max abs. error	 max rel. error	mixed moment based alg. max abs. error	 max rel. error
9	3.46 E-15	6.35 E-14	8.12 E-14	1.46 E-12
15	1.62 E-14	2.67 E-13	8.52 E-14	1.58 E-12
21	1.42 E-14	6.22 E-13	2.86 E-13	5.40 E-12

TABLE 5.4. Errors in computed Gauss-Kronrod-Radau nodes, $\alpha = -0.99$, $\beta = -0.9$, fixed node at $x = -1$

n	d + c based alg. max abs. error	 max rel. error	mixed moment based alg. max abs. error	 max rel. error
9	4.62 E-16	3.31 E-15	2.74 E-15	5.64 E-15
15	9.89 E-16	1.30 E-14	2.95 E-15	8.63 E-15
21	2.27 E-15	1.08 E-14	4.76 E-15	4.76 E-15

TABLE 5.5. Errors in computed Gauss-Kronrod rules, $\alpha = -0.9999$, $\beta = -0.5$

n	d + c based alg. max abs. error in weights	 max abs. error in nodes	mixed moment based alg. max abs. error in weights	 max abs. error in nodes
16	7.87 E-16	9.84 E-16	2.11 E-15	2.64 E-15
32	3.52 E-15	1.07 E-15	1.28 E-14	1.15 E-15
64	1.64 E-15	1.77 E-15	5.50 E-14	4.10 E-15
128	3.80 E-14	2.18 E-15	2.12 E-13	2.43 E-15
256	8.28 E-14	1.52 E-15	2.36 E-12	5.53 E-15

Our computational results can be summarized as follows. For many choices of α and β in (5.1) both Laurie's and our methods yield high accuracy. However, when at least one of the exponents in (5.1) is fairly close to -1, our method generally gives smaller errors. The higher accuracy in the computed weights achieved by the method of the present paper is particularly noteworthy.

Tables 5.1 and 5.2 display the magnitudes of the largest absolute and relative errors in the computed Gauss-Kronrod nodes and weights. These errors are referred to as "max abs. error" and "max rel. error", respectively. We use the notation 5.11E-14 for $5.11 \cdot 10^{-14}$. The examples in the tables illustrate the performance of the methods for a variety of choices of α and β for a few fairly small values of n. When $\alpha = \beta$, the Gauss-Kronrod rule has a node at the origin by symmetry. In the example with $n = 15$ and $\alpha = \beta = -0.97$, we set the computed node closest to the origin to zero before computing absolute and relative errors of the nodes.

Tables 5.3 and 5.4 show the errors in a few computed Gauss-Kronrod-Radau rules associated with the measure (5.1) and a fixed node at $x = -1$. These rules were computed by applying Algorithm 1 to the measure (4.4). Due to the scaling ($\mu_0 = 1$) assumed by the algorithm, it follows from (4.8) that the weights determined by Algorithm 1 have to be scaled by the factor $s(\alpha,\beta) := 2B(1+\alpha, 2+\beta)/B(1+\alpha, 1+\beta)$ to yield the weights $\{\tilde{w}'_j\}_{j=1}^{2n+1}$ of the Gauss-Kronrod rule (4.5). These weights are required in (4.6) to determine the Gauss-Kronrod-Radau weights $\tilde{w}_{k,a}$. Table 5.3 shows the errors in the computed weights $\{\tilde{w}_{k,a}\}_{k=0}^{2n+1}$ for the Jacobi measure (5.1) with $\alpha = -0.99$ and $\beta = -0.9$. For these values of α and β, we have $s(\alpha,\beta) = 20/11$. Table 5.4 shows the error in the computed nodes $\{\tilde{x}_{k,a}\}_{k=0}^{2n+1}$. Finally, Table 5.5 illustrates the performance of the methods for some large values of n.

6. Conclusion

The paper describes a new algorithm for the computation of Gauss-Kronrod quadrature rules and compares it to an algorithm recently proposed by Laurie. Both algorithms yield high accuracy for many problems. However, when an exponent in the Jacobi weight function (5.1) is close to -1, the algorithm of the present paper typically yields smaller errors. We also show how our algorithm can be applied to compute Kronrod extensions of Gauss-Radau and Gauss-Lobatto quadrature rules. The structure of the new algorithm makes efficient implementation in a parallel computing environment possible. This may be important in certain applications; see, e.g., [4, 8, 15].

Acknowledgments

We would like to thank Dirk Laurie for a Matlab implementation of his algorithm and for comments on a preliminary version of the manuscript. We also wish to thank Walter Gautschi and Gradimir Milovanović for reference [7].

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, 1992.

[2] D. Boley and G. H. Golub, A survey of matrix inverse eigenvalue problems, *Inverse Problems*, 3 (1987), pp. 595–622. MR **89m**:65036

[3] C. F. Borges and W. B. Gragg, A parallel divide and conquer algorithm for the generalized symmetric definite tridiagonal eigenvalue problem, *Numerical Linear Algebra*, (L. Reichel, A. Ruttan and R. S. Varga, eds.), de Gruyter, Berlin, 1993, pp. 11–29. MR **94k:**65051

[4] J. M. Bull and T. L. Freeman, Parallel globally adaptive quadrature on the KSR-1, *Adv. Comput. Math.*, 2 (1994), pp. 357–373. CMP 95:01

[5] D. Calvetti and L. Reichel, On an inverse eigenproblem for Jacobi matrices, *Adv. Comput. Math.*, 11 (1999), pp. 11–20.

[6] W. Gautschi, On generating orthogonal polynomials, *SIAM J. Sci. Stat. Comput.*, 3 (1982), pp. 289–317. MR **84e:**65022

[7] W. Gautschi, Gauss-Kronrod quadrature—a survey, *Numerical Methods and Approximation Theory* III, (G. V. Milovanović, ed.), University of Niš, 1987, pp. 39–66. MR **89k:**41035

[8] I. Gladwell, Vectorization of one dimensional quadrature codes, *Numerical Integration*, (P. Keast and G. Fairweather, eds.), Reidel, Dordrecht, 1987, pp. 231–238. MR **88i:**65039

[9] G. H. Golub and J. Kautsky, Calculation of Gauss quadratures with multiple free and fixed knots, *Numer. Math.*, 41 (1983), pp. 147–163. MR **84i:**65030

[10] G. H. Golub and J. H. Welsch, Calculation of Gauss quadrature rules, *Math. Comp.*, 23 (1969), pp. 221–230. MR **39:**6513

[11] M. Gu and S. C. Eisenstat, A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem, *SIAM J. Matrix Anal. Appl.*, 16 (1995), pp. 172–191. MR **95j:**65035

[12] D. P. Laurie, Calculation of Gauss-Kronrod quadrature rules, *Math. Comp.*, 66 (1997), pp. 1133–1145. MR **98m:**65030

[13] G. Monegato, A note on extended Gaussian quadrature rules, *Math. Comp.*, 30 (1976), pp. 812–817. MR **55:**13746

[14] G. Monegato, Stieltjes polynomials and related quadrature rules, *SIAM Rev.*, 24 (1982), pp. 137–158. MR **83d:**65067

[15] M. A. Napierala and I. Gladwell, Reducing ranking effects in parallel adaptive quadrature, in Proc. Seventh SIAM Conference on Parallel Processing for Scientific Computing, D. H. Bailey, J. R. Gilbert, M. V. Masagni, R. S. Schreiber, H. D. Simon, V. J. Torzon and L. T. Watson, eds., SIAM, Philadelphia, 1995, pp. 261–266.

[16] G. Szegő, Orthogonal Polynomials, 4th ed., Amer. Math. Society, Providence, 1975. MR **51:**8724

Department of Mathematics, Case Western Reserve University, Cleveland, Ohio 44106
E-mail address: dxc57@po.cwru.edu

Department of Computer Science, Stanford University, Stanford, California 94305
E-mail address: golub@chebyshev.stanford.edu

Department of Mathematics, Naval Postgraduate School, Monterey, California 93943
E-mail address: gragg@nps.navy.mil

Department of Mathematics and Computer Science, Kent State University, Kent, Ohio 44242
E-mail address: reichel@mcs.kent.edu

This page intentionally left blank

PART VI

EIGENVALUE PROBLEMS

This page intentionally left blank

26

COMMENTARY, BY G. W. STEWART

The four papers treated here reflect Gene Golub's interest in eigenvalue problems. However, they do not exhibit a general theme. Rather the papers are organized around specific problems or topics and have little connection with one another. For this reason we will treat the papers separately and make no attempt to describe interconnections.

Some modified matrix eigenvalue problems, by Golub [10]

This paper is a potpouri of problems that can be resolved by considering the eigensystems of certain matrices. A listing of the sections heads gives the flavor:

1. Stationary values of a quadratic form subject to linear constraints.
2. Stationary values of a bilinear form subject to linear constraints.
3. Some inverse eigenvalue problems.
4. Intersection of spaces.
5. Eigenvalues of a matrix modified by a rank one matrix.
6. Least squares problems.
7. Gauss-type quadrature rules with preassigned nodes.

All of these are interesting problems that are treated with admirable dispatch. But three of them — the one in Section 5 and two from Section 6 — are particularly worth discussion because of their impact on research after the paper was published.

The problem posed in Section 5 can be motivated as follows. Let A be a real symmetric matrix with an eigendecomposition $A = VDV^{\mathrm{T}}$ where D is real diagonal and V is orthogonal. Suppose that A is modified by a rank one matrix to give $B = A+\sigma xx^{\mathrm{T}}$, where $\sigma = \pm 1$. Can we determine the eigendecomposition of B from that of A?

This problem may be solved as follows. We have $V^{\mathrm{T}}BV = D + uu^{\mathrm{T}} \equiv C$, where $u = V^{\mathrm{T}}x$. Hence, if we can determine the eigendecomposition $C = WMW^{\mathrm{T}}$ of C, then $A+\sigma xx^{\mathrm{T}} = (VW)M(VW)^{\mathrm{T}}$ is the eigendecomposition of B. Thus the problem is reduced to finding the eigenvalues and eigenvectors of C, which is the problem treated by Golub. He proposes to do it in $O(n^2)$ operations, where n is the order of A. This means that the bulk of the work in updating the decomposition of A will be in the formation of MV, which requires $O(n^3)$ operations.

Finding the eigenvalues is comparatively easy. One can evaluate the characteristic polynomial $\det(C - \lambda I)$ by evaluating the successive characteristic polynomials of the leading principal submatrices of $C - \lambda I$. The eigenvalues can then be computed by standard root-finding techniques. Golub also shows that the eigenvalues can be obtained by solving the secular equation[1]

$$1+\sigma\sum_{i=1}^{n}\frac{u_i^2}{d_i-\lambda}=0,$$

which has turned out to be the preferred approach.

To find the eigenvectors, Golub forms a generalized problem and proposes to use a method of Peters and Wilkinson [24]. Unfortunately, as Golub notes, the method can fail to produce orthonormal eigenvectors.

Although Golub did not fully solve the problem he posed, it is extremely difficult and was only laid to rest some 20 years later. The solution is also at the heart of Cuppen's divide-and-conquer algorithm for the symmetric tridiagonal eigenvalue problem. For details and references see [28, p. 201].

The two problems considered in Section 6 are actually least squares problems, whose connection with eigenproblems is only through the use of the singular value decomposition to solve them. Nonetheless, they are both harbingers of research to come.

The first problem can be motivated by considering ordinary least squares approximation. We are given an $m\times n$ matrix A and a vector b which we wish to approximate as a linear combination Ax of the columns of A. The least squares approximation is found by finding a vector d of minimal Euclidean norm such that $Ax=b+d$. Under certain statistical assumptions, the vector d tells us a lot about the error in b that caused it to deviate from the column space of A.

Now suppose that A also has errors. A generalization of the least squares procedure is to determine a vector d and a matrix E such that $(A+E)x=b+d$ and $\|(E\ d)\|$ is minimal. Golub shows that the x consists of the first n components of the right singular vector corresponding to the smallest singular value of $(A\ b)$ normalized so that its last component is one. (Actually, Golub also allows row and column scaling in the errors.)

This is the origin of the method of total least squares, which was announced to the numerical community in a 1980 paper by Golub and Van Loan [12]. It was later realized that the technique, in a different form, had been used by statisticians to investigate measurement error models [9]. The area has continued to develop to the advantage of both numerical analysis and statistics. For more on this topic see [30; 29].

The second least squares problem is to solve the constrained least squares problem

$$\begin{array}{ll}\text{minimize} & \|b-Ax\|\\ \text{subject to} & \|x\|\le\alpha\end{array}\tag{26.1}$$

(Golub actually writes $\|x\|=\alpha$, but he then restricts the problem so that the case $\|x\|<\alpha$ cannot occur.) Using Lagrange multipliers, Golub shows that the solution of the problem is the solution of the system

$$(A^{\mathrm{T}}A+\lambda I)x=A^{\mathrm{T}}b\tag{26.2}$$

[1] A quick web search will show that this term is used in a variety of ways, not all consistent. It comes from celestial mechanics, where it appears in the computation of secular (i.e., long-term) perturbations of orbiting bodies.

where λ is chosen so the $\|x\| = \alpha$. He goes on to show how the problem of determining λ can be solved using the singular value decomposition of A.

This problem is of importance in optimization. In 1944, Levenberg [20] proposed using (26.2) to damp Gauss–Newton steps, and the method was rediscovered by Marquardt in 1963 [22], although neither actually solved the problem (26.1). In the 1970s, the problem resurfaced in trust-region methods for optimization [5], where it is important to determine a value of λ that makes $\|x\|$ approximately equal to α. Unfortunately, the singular value decomposition is too expensive to use in this context, and the problem is usually solved by regarding $\|x\|$ as a function of λ and applying root-finding techniques to compute the solution. This approach requires repeated factorizations of $A^{\mathrm{T}}A + \lambda I$ for different values of λ, but the total work is less than the cost of a single SVD.

Ill-conditioned eigensystems and the computation of the Jordan canonical form, by Golub and Wilkinson [13]

This is an important paper that concerns the use of similarity transformations to simplify a matrix. Specifically, let A be a matrix of order n and let X be nonsingular. Then the matrix

$$B = X^{-1}AX \tag{26.3}$$

has the same eigenvalues as A, and in fact the two matrices have the same Jordan canonical form. If X has been chosen so that B has a simpler structure than A, then problems – both mathematical and computational – that are easy to solve for B can be solved for A by using X to transform A to B and again using X to transform a solution for B back to a solution for A.

Of the possible candidates for B, the Jordan canonical form, mentioned above, is the most widely known. Here B is a block diagonal matrix, whose blocks $J_m(\lambda)$ have the form illustrated here for $m = 4$:

$$J_4(\lambda) = \begin{pmatrix} \lambda & 1 & 0 & 0 \\ 0 & \lambda & 1 & 0 \\ 0 & 0 & \lambda & 1 \\ 0 & 0 & 0 & \lambda \end{pmatrix}. \tag{26.4}$$

The larger part of the present paper is devoted to algorithms for computing the Jordan form – or rather some near relatives.

The algorithms are based on the notion of a vector of grade p. Specifically, x is of grade p if it satisfies the equations

$$(A - \lambda I)^{p-1}x \neq 0 \quad \text{and} \quad (A - \lambda I)^{p}x = 0. \tag{26.5}$$

For example, the ith column of the $m\times m$ identity matrix is a vector of grade i of $J_m(\lambda)$. If a complete set of graded vectors of A can be computed, then they can be assembled into a matrix X such that B has the same block structure as the Jordan form, but with full upper triangular blocks. Further reduction of B can produce the Jordan form, if it is required.

The authors propose three algorithms. The first is based on the powers of $A - \lambda I$. It has the fatal defect that powering causes small eigenvalues to coalesce numerically, making them indistinguishable. The second algorithm, closely related to algorithms of Kublanovskaya [18] and Ruhe [25], uses orthogonal deflation to compute its graded vectors. It is very costly. The third algorithm, of the authors' devising, is comparatively inexpensive but may possibly produce nearly dependent vectors.

Before deriving the algorithms, the authors consider two ancillary problems: the condition of eigenvalues and eigenvectors and the recognition of null spaces. The first relates to the fact that in practice computed eigenvalues must appear in the relations (26.5) that lie at the base of the algorithms. The treatment of eigenvalues is standard, even classical. The treatment of eigenvectors, due to Wilkinson, is old fashioned and vitiated by the fact that perturbations are expanded in terms of the original eigensystem, which may itself be nearly degenerate.

The second problem is one of rank detection, and the authors conclude that only the singular value decomposition is fully reliable. They consider several alternatives, and show by example how they can fail. It is worth noting that they fail to consider the pivoted QR decomposition, which, though technically fallible, is generally reliable and far less costly than the singular value decomposition.

In an interesting coda to the derivation of the algorithms, the authors consider the ill-conditioning of the Jordan form itself. They use Frank matrices [6; 8] to show that a matrix can be near several matrices with different Jordan forms. Such considerations throw doubt on the desirability of attempting to compute the Jordan form.

The authors then turn to the related problem of computing invariant subspaces – or equivalently producing a block diagonal matrix B in (26.3) by a well conditioned X. If the blocks are small enough, this may suffice for some applications. The general idea is to reduce A to an upper triangular matrix T by an orthogonal transformation (T is called a Schur form of A). Then a basis for the invariant subspace corresponding to an initial set of eigenvalues T can be calculated by solving linear equations. By moving other eigenvalues into the initial position, other invariant subspaces may be computed. The chief problem here is how to select a group of eigenvalues to define an invariant subspace.

The paper concludes with two sections, one on the inverse power method and defective matrices and the other on the refinement of bases for invariant subspaces. Both are interesting but are peripheral to the main concerns of the paper.

This paper is important both as an early paper on the subject of block diagonalization and for the depth of its treatment of the subject. For more references see [28, p. 25].

The block Lanczos method for computing eigenvalues, by Golub and Underwood [11]

This paper concerns a block variant of the Lanczos algorithm for the symmetric eigenproblem [19]. To understand how block algorithms came about, we must first sketch the original Lanczos algorithm as it was understood at the time of this paper.

We begin by presenting a general algorithm due to Arnoldi [1], who generalized the formal aspects of Lanczos's algorithm to nonsymmetric matrices. Let A be of order n. Given a normalized vector u_1, the Arnoldi algorithm produces a sequence of vectors $u_1, \ldots, u_i$ as follows.

> Given $u_1, \ldots, u_{i-1}$, the vector u_i is the normalized linear combination of $u_1, \ldots, u_{i-1}$, and Au_{i-1} that is orthogonal to $u_1, \ldots, u_{i-1}$. (26.6)

This ongoing process is traditionally summarized by the Arnoldi relation

$$AU_i = U_{i+1}\hat{H}_i, \tag{26.7}$$

where $U_i = (u_1 \ \cdots \ u_i)$ and $\hat{H}_i$ is an $(i{+}1)\times i$ upper Hessenberg matrix, that is a matrix whose elements below its first subdiagonal are zero.

The subspace spanned by the matrix U_i is called the ith Krylov subspace of A and u_1. With increasing i, these subspaces often contain increasingly accurate approximations to eigenvectors of A corresponding to well-separated eigenvalues on the periphery of the spectrum of A [26; 27]. They can be approximated by the following Rayleigh–Ritz procedure to compute an approximation to an eigenpair (λ, x).

1. Form the Rayleigh quotient $H_i = U_i^{\mathrm{T}} A U_i$. [Note that H_i consists of the first i rows of $\hat{H}_i$ in (26.7)].
2. Compute an eigenpair (μ, w) of H_i, where μ approximates λ. (26.8)
3. Take $(\mu, z) = (\mu, U_i w)$ to be the approximation to (λ, x).

Convergence tests are inexpensive to perform, since the norm of the residual $Az - \mu z$ is $|h_{i+1,i} w_i|$, where w_i is the last component of w.

The Arnoldi process is expensive in both time and storage. It requires that Au_i be orthogonalized against all the vectors $u_1, \ldots, u_{i-1}$, a process that requires $O(ns)$ operations. Thus s steps of the process require $O(ns^2)$ work, not to mention $O(ns)$ storage for the vectors.

When A is symmetric, however, the resulting Lanczos algorithm simplifies remarkably. The key is that the matrix $H_i = U_i^{\mathrm{T}} A U_i$, which consists of the first i rows of $\hat{H}_i$ in (26.7), is symmetric. Since H_i is Hessenberg, it is also tridiagonal; i.e.,

$$H_i = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ & & \ddots & \ddots & \ddots \end{pmatrix}.$$

It follows that the vectors u_i satisfy a three-term recurrence of the general form

$$\beta_i u_{i+1} = Au_i - \alpha_i u_i - \beta_{i-1} u_{i-1}. \tag{26.9}$$

Thus the work for s steps of the process is $O(ns)$. The storage is $O(n)$, since only the vectors u_i and u_{i-1} have to be present to advance the process. In addition, the Rayleigh–Ritz process (26.8) simplifies, since the Rayleigh quotient is symmetric tridiagonal.[2] Finally, the convergence theory becomes both simpler and stronger.

Unfortunately, there is a fly in the ointment. Because u_i is not explicitly orthogonalized against $u_1, \ldots, u_{i-1}$ in (26.9), rounding error will cause the global orthogonality of the u_i to be lost as the iteration progresses. At the time of the present paper, there was no agreement about what to do about this problem, but it was eventually resolved without sacrificing too much of the efficiency of the Lanczos process. We cannot pause to tell the story here, but we will return to the reorthogonalization problem in connection with the present paper.[3]

[2] Here we do need all the u_i to compute approximate eigenvectors. However, they can be squirreled away on a backing store until they are needed at the very end of the computation.

[3] For more see [23; 28].

A second problem with the Lanczos procedure is that it cannot see more than one eigenvector of a multiple eigenvalue. Consider, for example, a nondefective eigenvalue λ of multiplicity two. The set of eigenvectors corresponding to λ, along with the zero vector, is a two-dimensional subspace, which is spanned by any two orthonormal eigenvectors corresponding to λ, say x and y. If we expand u_1 in an eigenvector expansion, we get

$$u_1 = \alpha x + \beta y + \cdots .$$

Then Au_1 will have the form

$$Au_1 = \lambda(\alpha x + \beta y) + \cdots .$$

Thus the passage from u_1 to u_2 introduces no new information about the subspace corresponding to λ. As far as the Lanczos process is concerned, A has a simple eigenvalue λ with eigenvector $\alpha x + \beta y$.

One way of solving this problem is to somehow start the Lanczos process with a block U_1 having two or more columns. Each column of the block will, in general, have a different linear combination of x and y in its expansion. For this reason, among others, people began looking at block variants of the Lanczos process in the mid-1970s. It is not easy to sort out priorities, since the literature, such as it is, consists of oral communications, PhD theses, technical reports, and conference proceedings that are not readily available. A list of players would at least include J. Cullum, W. E. Donath, G. H. Golub, W. Kahan, J. Lewis, B. N. Parlett and R. Underwood [3; 16; 21].

The present paper by Golub and Underwood is a summary of the thesis of the latter, which was directed by the former. It is a lucid exposition of the ideas that underlie any block Lanczos algorithm and is well worth reading today. But it may help the reader to mention a few highlights here. We follow the authors' notation.

First, the basic step in the Lanczos process starts with an $n \times p$ matrix X_1 with orthonormal columns and continues in analogy with (26.6) as follows.

1. Given $X_1, \ldots, X_{i-1}$, let Z_i be the linear combination of $X_i, \ldots, X_{i-1}$, and AX_{i-1} that is orthogonal to $X_1, \ldots, X_{i-1}$.

2. Orthonormalize the columns of Z_i to get $Z_i = X_i R_i$, where R_i is upper triangular.

It is possible for Z_i to be deficient in rank, in which case X_i must be supplemented by orthonormal columns that are not only orthogonal to the initial columns of X_i but to the columns of all the previous X_i.

Second, the X_i satisfy a three-term recurrence of the general form

$$X_{i+1}R_{i+1} = AX_i - X_i M_i - X_{i-1}R_i^{\mathrm{T}}.$$

As in the Lanczos process, however, global orthogonality can quickly disappear. At the time this paper was written, the only effective cure was to orthogonalize against all the previous X_i, which is what the authors do. Today we would call the algorithm a symmetric block Arnoldi method; but that is a matter of hindsight.

Third, the Rayleigh–Ritz technique remains essentially the same, but the Rayleigh quotient is now block-tridiagonal. This complicates the computation of the Ritz approximations.

Fourth, the authors give a convergence analysis for the Ritz values that generalizes a result of Kaniel [17]. The analysis shows faster convergence for the block method, but the authors do not address the issue of whether that compensates for the extra work involved in the block algorithm.

Finally, storage considerations eventually limit the number of steps. In this case the authors propose restarting with an X_1 consisting of unconverged eigenvectors. Everything must be orthogonalized against the converged eigenvectors to prevent their reappearance in subsequent iterations.

In summary, this is a well organized, well written paper that represents the state of the art at its time. For a view of things yet to come see the paper by Grimes, Lewis, and Simon [14].

The numerically stable reconstruction of a Jacobi matrix from spectral data, by De Boor and Golub [4]

This elegant paper is the first of a long run of papers reflecting Golub's interest in inverse matrix eigenvalue problems (which has culminated with a survey of the subject by Chu and Golub [2]). Loosely speaking, an inverse eigenvalue problem is one of determining facts about a matrix from the distribution of its spectrum. Of the three problems considered in the present paper, the central one is the following. Given real numbers $\lambda_1, \ldots, \lambda_n$ and $\mu_1, \ldots, \mu_{n-1}$ with

$$\lambda_i < \mu_i < \lambda_{i+1}, \qquad i = 1, \ldots, n-1, \tag{26.10}$$

determine a symmetric tridiagonal matrix

$$J = \begin{pmatrix} a_1 & b_1 & & & & \\ b_1 & a_2 & b_2 & & & \\ & b_2 & a_3 & b_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & b_{n-2} & a_{n-1} & b_{n-1} \\ & & & & b_{n-1} & a_n \end{pmatrix}$$

with $b_i > 0$ such that

1. the eigenvalues of J are the λ_i,
2. the eigenvalues of the leading principal submatrix of order $n-1$ of J are the μ_i.

The degrees of freedom for this problem match up with the requirements: the $2n-1$ quantities a_i and b_i are to be determined from the $2n-1$ quantities λ_i and μ_i. Note that the inequalities (26.10) are necessary for the problem to have a solution, since they follow from standard interleaving theorems for eigenvalues of symmetric matrices.

At the time of this paper, the problem had been solved in the sense that it had been shown that there is a unique solution that depends continuously on the λ_i and μ_i. Moreover, Hald [15] had produced an algorithm for computing the a_i and b_i based on the connection of the problem with orthogonal polynomials and an algorithm of Wendroff [31]. However, the algorithm is numerically unstable, and the main contribution of this paper is to replace it with a more stable one.

To see the connection with orthogonal polynomials, let J_i $(i = 1, \ldots, n)$ be the leading principal submatrix of J of order i and let

$$p_i(t) = \det(tI - J_i)$$

with $p_0(t) = 1$ and $p_{-1}(t) = 0$. Then it is easily seen that the the p_i satisfy a three-term recurrence

$$p_i(t) = (t - a_i)p_{i-1}(t) - b_{i-1}^2 p_{i-2}. \tag{26.11}$$

Hald's algorithm amounts to constructing the polynomials p_n and p_{n-1} from the λ_i and μ_i and then running the recursion (26.11) backward to determine the a_i and b_i. However, the resulting algorithm is "badly unstable."

De Boor and Golub address this problem by observing that the polynomials generated by (26.11) are orthogonal with respect to an inner product. They show that the weights for this inner product can be computed from p_n and p_{n-1}. Knowing the inner product, the a_i and b_i can be generated by procedure due to Forsythe [7]. It is worth noting that the procedure is an exact analog of the Lanczos tridiagonalization of a symmetric matrix.

The authors go on to consider the problem in which A is persymmetric (invariant under reflection in the secondary diagonal). Since $a_i = a_{n-i+1}$ and $b_i = b_{n-i}$, this problem has only n degrees of freedom, and the natural conjecture that the elements are determined by the λ_i alone is true. In addition, when the original problem is considered in the light of the persymmetric problem there results a new algorithm with a superior formula for the weights.

In some numerical experiments, the authors demonstrate the effectiveness of their algorithm on well-conditioned cases. They also include an ill-conditioned problem in which μ_1 is very near λ_1, even though the λ_i are well separated. Such a problem will occur whenever the last two components of the eigenvector corresponding to λ_1 are very small, a situation that often occurs in practice (for more see [32, pp. 308–321]).

REFERENCES

1. W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, **9**, 17–29 (1951).

2. M. T. Chu and G. H. Golub. *Inverse Eigenvalue Problems: Theory, Algorithms, and Applications.* Oxford University Press, New York (2005).

3. J. Cullum and W. W. Donath. A block generalization of the symmetric s-step Lanczos algorithm. Report RC 4845 (21570), IBM Thomas J. Watson Research Center (1974).

4*. C. de Boor and G. H. Golub. The numerically stable reconstruction of a Jacobi matrix from spectral data. *Linear Algebra Appl.*, **21**(3), 245–260 (1978).

5. J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice–Hall, Englewood Cliffs, NJ (1983).

6. P. J. Eberlein. A note on the matrices denoted B_n. *SIAM Journal on Applied Mathematics*, **20**, 87–92 (1977).

7. G. E. Forsythe. Generation and use of orthogonal polynomials for data-fitting with a digital computer. *SIAM Journal*, **5**, 74–88 (1957).

8. W. L. Frank. Computing eigenvalues of complex matrices by determinant evaluation and by methods of Danilewski and Wielandt. *Journal of the Society for Industrial and Applied Mathematics*, **6**, 378–392 (1958).

9. W. A. Fuller. *Measurement Error Models.* John Wiley, New York (1987).

10*. G. H. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, **15**, 318–344 (1973).

11*. G. H. Golub and R. Underwood. The block Lanczos method for computing eigenvalues. In J. Rice, editor, *Mathematical Software III*, pages 364–377. Academic Press, New York (1977).

12*. G. H. Golub and C. F. Van Loan. An analysis of the total least squares problem. *SIAM Journal on Numerical Analysis*, **17**, 883–893 (1980).

13*. G. H. Golub and J. H. Wilkinson. Ill-conditioned eigensystems and the computation of the Jordan canonical form. *SIAM Review*, **18**, 578–619 (1976).

14. R. G. Grimes, J. G. Lewis, and H. D. Simon. A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, **15**, 228–272 (1994).

15. O. H. Hald. Inverse eigenvalue problems for Jacobi matrices. *Linear Algebra and Its Applications*, **14**, 63–85 (1976).

16. W. Kahan and B. N. Parlett. How far should you go with the Lanczos process? J. Bunch and D. Rose, eds., *Sparse Matrix Computations*, pages 131–144. Academic Press, New York (1976).

17. S. Kaniel. Estimates for some computational techniques in linear algebra. *Mathematics of Computation*, **20**, 369–378 (1966).

18. V. N. Kublanovskaya. On a method of solving the complete eigenvalue problem for a degenerate matrix. *Z. Vyčisl. Math. mat. Fiz.*, **6**, 611–620 (1966). AMS Translations, Vol. 18 (1969).

19. C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, **45**, 255–282 (1950).
20. K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quart. Appl. Math.*, **2**, 164–168 (1944).
21. J. Lewis. *Algorithms for Sparse Matrix Eigenvalue Problems.* PhD thesis, Stanford University (1977).
22. D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, **11**, 431–441 (1963).
23. B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice–Hall, Englewood Cliffs, NJ, 1980. Reissued with revisions by SIAM, Philadelphia, PA (1998).
24. G. Peters and J. H. Wilkinson. Eigenvalues of $Ax = \lambda Bx$ with band symmetric A and B. *The Computer Journal*, **12**, 398–404 (1969).
25. A. Ruhe. An algorithm for numerical determination of the structure of a general matrix. *BIT*, **10**, 196–216 (1969).
26. Y. Saad. On the rates of convergence of the Lanczos and the block Lanczos methods. *SIAM Journal on Numerical Analysis*, **17**, 687–706 (1980).
27. Y. Saad. Variations of Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Linear Algebra and Its Applications*, **34**, 269–295 (1980).
28. G. W. Stewart. *Matrix Algorithms II: Eigensystems.* SIAM, Philadelphia, PA (2001).
29. S. Van Huffel, editor. *Recent Advances in Total Least Squares Techniques and Errors-in-Variables Modeling.* SIAM, Philadelphia, PA (1996). Proceeding of the Second International Workshop on Total Least Squares and Errors-in-Variables Modeling.
30. S. Van Huffel and J. Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis.* SIAM, Philadelphia (1991).
31. B. Wendroff. On orthogonal polynomials. *Proceedings of the American Mathematical Society*, **12**, 554–555 (1961).
32. J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford (1965).

An asterisk denotes a paper reprinted in this volume.

27

SOME MODIFIED MATRIX EIGENVALUE PROBLEMS

SIAM Review (15), 318–335 (1973).

SIAM Review
Vol. 15, No. 2, April 1973

SOME MODIFIED MATRIX EIGENVALUE PROBLEMS*

GENE H. GOLUB†

Dedicated to the memory of Professor H. Rutishauser

Abstract. We consider the numerical calculation of several matrix eigenvalue problems which require some manipulation before the standard algorithms may be used. This includes finding the stationary values of a quadratic form subject to linear constraints and determining the eigenvalues of a matrix which is modified by a matrix of rank one. We also consider several inverse eigenvalue problems. This includes the problem of determining the coefficients for the Gauss–Radau and Gauss–Lobatto quadrature rules. In addition, we study several eigenvalue problems which arise in least squares.

Introduction and notation. In the last several years, there has been a great development in devising and analyzing algorithms for computing eigensystems of matrix equations. In particular, the works of H. Rutishauser and J. H. Wilkinson have had great influence on the development of this subject. It often happens in applied situations that one wishes to compute the eigensystem of a slightly modified system or one wishes to specify some of the eigenvalues and then compute an associated matrix. In this paper we shall consider some of these problems and also some statistical problems which lead to interesting eigenvalue problems. In general, we show how to reduce the modified problems to standard eigenvalue problems so that the standard algorithms may be used. We assume that the reader has some familiarity with some of the standard techniques for computing eigensystems.

We ordinarily indicate matrices by capital letters such as A, B, Σ; vectors by boldface lower-case letters such as $\mathbf{x}$, $\mathbf{y}$, $\boldsymbol{\alpha}$, and scalars by lower-case letters. We indicate the eigenvalues of a matrix as $\lambda(X)$ where X may be an expression, e.g., $\lambda(A^2 + I)$ indicates the eigenvalues of $A^2 + I$, and in a similar fashion we indicate the singular values of a matrix by $\sigma(X)$. We assume that the reader has some familiarity with singular values (cf. [9]). Usually we order the singular values $\sigma(A) = [\lambda(A^TA)]^{1/2}$ of a matrix A so that $\sigma_1(A) \leqq \sigma_2(A) \leqq \cdots \leqq \sigma_n(A)$ and if A is symmetric, the eigenvalues so that $\lambda_1(A) \leqq \lambda_2(A) \leqq \cdots \leqq \lambda_n(A)$.

1. Stationary values of a quadratic form subject to linear constraints. Let A be a real symmetric matrix of order n, and $\mathbf{c}$ a given vector with $\mathbf{c}^T\mathbf{c} = 1$.

In many applications (cf. [10]) it is desirable to find the stationary values of

$$\mathbf{x}^TA\mathbf{x} \tag{1.1}$$

subject to the constraints

$$\mathbf{x}^T\mathbf{x} = 1, \tag{1.2}$$

$$\mathbf{c}^T\mathbf{x} = 0. \tag{1.3}$$

* Received by the editors September 8, 1971.

† Computer Science Department, Stanford University, Stanford, California 94305. This work was supported in part by grants from the National Science Foundation and the Atomic Energy Commission.

Let

$$\varphi(\mathbf{x}, \lambda, \mu) = \mathbf{x}^T A\mathbf{x} - \lambda(\mathbf{x}^T\mathbf{x} - 1) + 2\mu\mathbf{x}^T\mathbf{c}, \tag{1.4}$$

where λ, μ are Lagrange multipliers. Differentiating (1.4), we are led to the equation

$$A\mathbf{x} - \lambda\mathbf{x} + \mu\mathbf{c} = \mathbf{0}. \tag{1.5}$$

Multiplying (1.5) on the left by $\mathbf{c}^T$ and using the condition that $\|\mathbf{c}\|_2 = 1$, we have

$$\mu = -\mathbf{c}^T A\mathbf{x}. \tag{1.6}$$

Then substituting (1.6) into (1.5), we obtain

$$PA\mathbf{x} = \lambda\mathbf{x}, \tag{1.7}$$

where $P = I - \mathbf{c}\mathbf{c}^T$. Although P and A are symmetric, PA is not necessarily so. Note that $P^2 = P$, so that P is a projection matrix.

It is well known (cf. [14, p. 54]) that for two arbitrary square matrices G and H, the eigenvalues of GH equal the eigenvalues of HG. Thus,

$$\lambda(PA) = \lambda(P^2A) = \lambda(PAP).$$

The matrix PAP is symmetric and hence one can use one of the standard algorithms for finding its eigenvalues. Then if

$$K = PAP$$

and if

$$K\mathbf{z}_i = \lambda_i\mathbf{z}_i,$$

it follows that

$$\mathbf{x}_i = P\mathbf{z}_i, \qquad i = 1, 2, \cdots, n,$$

where $\mathbf{x}_i$ is the eigenvector which satisfies (1.7). At least one eigenvalue of K will be equal to zero, and $\mathbf{c}$ will be an eigenvector associated with a zero eigenvalue.

Now suppose we replace the constraint (1.3) by the set of constraints

$$C^T\mathbf{x} = \mathbf{0}, \tag{1.8}$$

where C is an $n \times p$ matrix of rank r. It can be verified that if

$$P = I - CC^-, \tag{1.9}$$

where C^- is a generalized inverse which satisfies

$$CC^-C = C, \qquad CC^- = (CC^-)^T, \tag{1.10}$$

then the stationary values of $\mathbf{x}^T A\mathbf{x}$ subject to (1.2) and (1.8) are eigenvalues of $K = PAP$. At least r of the eigenvalues of K will be equal to zero, and hence it would be desirable to deflate the matrix K so that these eigenvalues are eliminated.

By permuting the columns of C, we may compute the orthogonal decomposition

$$C = Q^T\begin{bmatrix} R & S \\ 0 & 0 \end{bmatrix}\Pi, \tag{1.11}$$

where R is an upper triangular matrix of order r, S is $r \times (p - r)$, $Q^T Q = I_n$, and Π is a permutation matrix. The matrix Q may be constructed as the product of r Householder transformations (cf. [8]). A simple calculation shows

$$P = Q^T \begin{bmatrix} 0 & 0 \\ 0 & I_{n-r} \end{bmatrix} Q \equiv Q^T J Q, \tag{1.12}$$

and thus

$$\lambda(PAP) = \lambda(Q^T J Q A Q^T J Q) = \lambda(J Q A Q^T J).$$

Then if

$$G = QAQ^T = \begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{bmatrix}, \tag{1.13}$$

where G_{11} is an $r \times r$ matrix and G_{22} is an $(n - r) \times (n - r)$ matrix,

$$JQAQ^T J = \begin{bmatrix} 0 & 0 \\ 0 & G_{22} \end{bmatrix}.$$

Hence the stationary values of $\mathbf{x}^T A \mathbf{x}$ subject to (1.2) and (1.8) are simply the eigenvalues of the $(n - r) \times (n - r)$ matrix G_{22}. Finally, if

$$G_{22} \mathbf{z}_i = \lambda_i \mathbf{z}_i, \qquad i = 1, 2, \cdots, n - r,$$

then

$$\mathbf{x}_i = Q^T \begin{bmatrix} 0 \\ I_{n-r} \end{bmatrix} \mathbf{z}_i.$$

The details of the algorithm are given in [10].

From (1.13) we see that $\lambda(G) = \lambda(A)$. Then by the Courant–Fischer theorem (cf. [14, p. 101]),

$$\lambda_j(A) \leqq \lambda_j(G_{22}) \leqq \lambda_{r+j}(A), \qquad j = 1, 2, \cdots, n - r, \tag{1.14}$$

when

$$\lambda_j(A) \leqq \lambda_{j+1}(A) \quad \text{and} \quad \lambda_j(G_{22}) \leqq \lambda_{j+1}(G_{22}).$$

Furthermore, if the columns of the matrix C span the same space as the r eigenvectors associated with the r smallest eigenvalues of A,

$$\lambda_j(G_{22}) = \lambda_{r+j}(A). \tag{1.15}$$

Thus, we see that there is a strong relationship between the eigenvalues of A and the stationary values of the function

$$\varphi(\mathbf{x}, \lambda, \boldsymbol{\mu}) = \mathbf{x}^T A \mathbf{x} - \lambda(\mathbf{x}^T \mathbf{x} - 1) + 2\boldsymbol{\mu}^T C^T \mathbf{x}, \tag{1.16}$$

where $\boldsymbol{\mu}$ is a vector of Lagrange multipliers.

2. Stationary values of a bilinear form subject to linear constraints. Now let us consider the problem of determining the nonnegative stationary values of

$$(\mathbf{x}^T A \mathbf{y})/(\|\mathbf{x}\|_2 \|\mathbf{y}\|_2), \tag{2.1}$$

where A is an $m \times n$ matrix, subject to the constraints

$$C^T\mathbf{x} = \mathbf{0}, \qquad D^T\mathbf{y} = \mathbf{0}. \tag{2.2}$$

The nonnegative stationary values of (2.1) are the singular values of A (i.e., $\sigma(A) = [\lambda(A^TA)]^{1/2}$). It is easy to verify that the nonnegative stationary values of (2.1) subject to (2.2) are the singular values of

$$P_CAP_D\,, \tag{2.3}$$

where

$$P_C = I - CC^-, \qquad P_D = I - DD^-.$$

The singular values of P_CAP_D can be computed using the algorithm given in [9].

Again it is not necessary to compute the matrices P_C and P_D explicitly. If, as in (1.11),

$$C = Q_C^T\begin{bmatrix} R_C & S_C \\ 0 & 0 \end{bmatrix}\Pi_C, \qquad D = Q_D^T\begin{bmatrix} R_D & S_D \\ 0 & 0 \end{bmatrix}\Pi_D,$$

then

$$P_C = Q_C^T\begin{bmatrix} 0 & 0 \\ 0 & I_{n-r} \end{bmatrix}Q_C \equiv Q_C^TJ_CQ_C,$$

$$P_D = Q_D^T\begin{bmatrix} 0 & 0 \\ 0 & I_{n-s} \end{bmatrix}Q_D \equiv Q_D^TJ_DQ_D,$$

where r is the rank of C and s is the rank of D. Then

$$\begin{aligned}\sigma(P_CAP_D) &= \sigma(Q_C^TJ_CQ_CAQ_D^TJ_DQ_D) \\ &= \sigma(J_CQ_CAQ_D^TJ_D).\end{aligned}$$

Hence if

$$G = Q_CAQ_D^T = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix},$$

where G_{11} is $r \times s$ and G_{22} is $(m - r) \times (n - s)$, then

$$J_CQ_CAQ_D^TJ_D = \begin{bmatrix} 0 & 0 \\ 0 & G_{22} \end{bmatrix}.$$

Thus the desired stationary values are the singular values of G_{22}.

3. Some inverse eigenvalue problems. Suppose we are given a symmetric matrix A with eigenvalues $\{\lambda_i\}_{i=1}^n$ ($\lambda_i < \lambda_{i+1}$) and we are given a set of values $\{\bar{\lambda}_i\}_{i=1}^{n-1}$ ($\bar{\lambda}_i < \bar{\lambda}_{i+1}$) with

$$\lambda_i < \bar{\lambda}_i < \lambda_{i+1}. \tag{3.1}$$

We wish to determine the linear constraint $\mathbf{c}^T\mathbf{x} = 0$ so that the stationary values of $\mathbf{x}^TA\mathbf{x}$ subject to $\mathbf{x}^T\mathbf{x} = 1$ and $\mathbf{c}^T\mathbf{x} = 0$ ($\mathbf{c}^T\mathbf{c} = 1$) are the set of $\{\bar{\lambda}_i\}_{i=1}^{n-1}$. From (1.5) we have

$$\mathbf{x} = -\mu(A - \lambda I)^{-1}\mathbf{c},$$

and hence,

$$\mathbf{c}^T\mathbf{x} = -\mu\mathbf{c}^T(A - \lambda I)^{-1}\mathbf{c} = 0. \tag{3.2}$$

Assuming $\mu \neq 0$, and given $A = Q\Lambda Q^T$ where Λ is the diagonal matrix of eigenvalues of A and Q is the matrix of orthonormalized eigenvectors, substitution into (3.2) gives

$$\sum_{i=1}^{n} \frac{d_i^2}{(\lambda_i - \lambda)} = 0 \tag{3.3a}$$

with

$$\sum_{i=1}^{n} d_i^2 = 1 \tag{3.3b}$$

where $Q\mathbf{d} = \mathbf{c}$. Setting $\lambda = \bar{\lambda}_j$, $j = 1, 2, \cdots, n - 1$, then leads to a system of linear equations defining the d_i^2. We shall, however, give an explicit solution to this system.

Let the characteristic polynomial be

$$\varphi(\lambda) = \prod_{j=1}^{n-1} (\bar{\lambda}_j - \lambda). \tag{3.4}$$

We convert the rational form (3.3a) to a polynomial,

$$\begin{aligned} \psi(\lambda) &\equiv \prod_{j=1}^{n} (\lambda_j - \lambda)\left[\sum_{i=1}^{n} \frac{d_i^2}{(\lambda_i - \lambda)}\right] \\ &= \sum_{i=1}^{n} d_i^2 \prod_{\substack{j=1 \\ j \neq i}}^{n} (\lambda_j - \lambda). \end{aligned} \tag{3.5}$$

We wish to compute $\mathbf{d}$ ($\mathbf{d}^T\mathbf{d} = 1$) so that $\psi(\lambda) \equiv \varphi(\lambda)$. Then let us equate the two polynomials at n points. Now

$$\varphi(\lambda_k) = \prod_{j=1}^{n-1} (\bar{\lambda}_j - \lambda_k),$$

$$\psi(\lambda_k) = d_k^2 \prod_{\substack{j=1 \\ j \neq k}}^{n} (\lambda_j - \lambda_k).$$

Hence $\varphi(\lambda_k) = \psi(\lambda_k)$ for $k = 1, 2, \cdots, n$, if

$$d_k^2 = \frac{\prod_{j=1}^{n-1} (\bar{\lambda}_j - \lambda_k)}{\prod_{j=1, j \neq k}^{n} (\lambda_j - \lambda_k)}. \tag{3.6}$$

The condition (3.1) guarantees that the right-hand side of (3.6) will be positive. Note that we may assign d_k a positive or negative value so that there are 2^n different solutions. Once the vector $\mathbf{d}$ has been computed, it is an easy matter to compute $\mathbf{c}$.

We have seen in § 1 that the stationary values of (1.16) interlace the eigenvalues of A. In certain statistical applications [4] the following problem arises. Given a matrix A and an $n \times p$ matrix C, we wish to find an orthogonal matrix H so that the stationary values of

$$\varphi(\mathbf{x}, \lambda, \boldsymbol{\mu}) = \mathbf{x}^T A\mathbf{x} - \lambda(\mathbf{x}^T\mathbf{x} - 1) + \boldsymbol{\mu}^T(HC)^T\mathbf{x} \tag{3.7}$$

are equal to the $n - r$ largest eigenvalues of A.

As was pointed out in the last paragraph of § 1, the stationary values of (3.7) will be equal to the $n - r$ largest eigenvalues of A providing the columns of HC span the space associated with the r smallest eigenvalues of A. For simplicity, we assume that rank $(C) = p$. From (1.11), we see that we may write

$$C = Q^T\begin{bmatrix} R \\ 0 \end{bmatrix}.$$

Let us assume that the columns of some $n \times p$ matrix V span the same space as eigenvectors associated with the p smallest eigenvalues. We can construct the decomposition

$$V = W^T\begin{bmatrix} S \\ 0 \end{bmatrix},$$

where $W^TW = I_n$ and S is upper triangular. Then the constraints

$$(HC)^T\mathbf{x} = \mathbf{0}$$

are equivalent to

$$[R^T \vdots 0]QH^T\mathbf{x} = \mathbf{0},$$

and thus if H is chosen to be

$$H = W^TQ,$$

the stationary values of (3.7) will be equal to the $n - p$ largest eigenvalues of A.

4. Intersection of spaces. Suppose we are given two symmetric $n \times n$ matrices A and B with B positive definite and we wish to compute the eigensystem for

$$A\mathbf{x} = \lambda B\mathbf{x}. \tag{4.1}$$

One ordinarily avoids computing $C = B^{-1}A$ since the matrix C is usually not symmetric. Since B is positive definite, it is possible to compute a matrix F such that

$$F^TBF = I$$

and we can verify from the determinantal equation that

$$\lambda(F^TAF) = \lambda(B^{-1}A).$$

The matrix F^TAF is symmetric and hence one of the standard algorithms may be used for computing its eigenvalues.

Now let us consider the following example. Suppose

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

where ε is a small positive value. Note B is no longer positive definite. When $\mathbf{x}^T = [1, 0, 0]$, then $A\mathbf{x} = B\mathbf{x}$ and hence $\lambda = 1$. When $\mathbf{x}^T = [0, 1, 0]$, then $A\mathbf{x} = \varepsilon^{-1}B\mathbf{x}$. Here $\lambda = \varepsilon^{-1}$ and hence as ε gets arbitrarily small, $\lambda(\varepsilon)$ becomes arbitrarily large. This eigenvalue is unstable; such problems have been carefully studied by Fix and Heiberger [5]. Finally for $\mathbf{x}^T = [0, 0, 1]$, $A\mathbf{x} = \lambda B\mathbf{x}$ for all values of λ. Thus we have the situation of continuous eigenvalues. We shall now examine ways of eliminating the problem of continuous eigenvalues.

The eigenvalue problem $A\mathbf{x} = \lambda B\mathbf{x}$ can have continuous eigenvalues if the null space associated with A and the null space associated with B intersect. Therefore we wish to determine a basis for the intersection of these two null spaces. Let us assume we have determined X and Y so that

$$AX = 0, \qquad BY = 0$$

with

(4.2) $$X^TX = I_p \quad \text{and} \quad Y^TY = I_q.$$

Let

(4.3) $$Z = [X \vdots Y].$$

Suppose H is an $n \times v$ basis for the null space of Z with

$$H = \begin{bmatrix} E \\ \cdots \\ F \end{bmatrix},$$

where E is $p \times v$ and F is $q \times v$. Then

$$ZH = XE + YF = 0.$$

Hence the nullity of Z determines the rank of the basis for the intersection of the two spaces.

Consider the matrix

$$L = Z^TZ.$$

Note nullity(L) = nullity(Z). From (4.3), we see that

(4.4) $$L = \begin{bmatrix} I_p & X^TY \\ Y^TX & I_q \end{bmatrix} \equiv I_{p+q} + \begin{bmatrix} 0_p & T \\ T^T & 0_q \end{bmatrix} \equiv I_{p+q} + W.$$

Since $\lambda(L) = \lambda(I + W) = 1 + \lambda(W)$,

(4.5) $$\lambda(L) = 1 \pm \sigma(T).$$

Therefore, if $\sigma_j(T) = 1$ for $j = 1, 2, \cdots, t$, from (4.5) we see that the nullity $(L) = t$.

Consider the singular value decomposition of the matrix

$$T = X^TY = U\Sigma V^T,$$

where

$$U = [\mathbf{u}_1, \cdots, \mathbf{u}_p], \qquad V = [\mathbf{v}_1, \cdots, \mathbf{v}_q].$$

The matrices $\tilde{X} = XU$ and $\tilde{Y} = YV$ yield orthonormal bases for the null space of the matrices A and B, respectively. Since $\sigma(T) = 1$ for $j = 1, \cdots, t$,

$$\tilde{\mathbf{x}}_j = X\mathbf{u}_j = Y\mathbf{v}_j = \tilde{\mathbf{y}}_j \quad \text{for } j = 1, \cdots, t,$$

and thus the vectors $\{\tilde{\mathbf{x}}_j\}_{j=1}^t$ yield a basis for the intersection of the two spaces.

The singular values of X^TY can be thought of as the cosines between the spaces generated by X and Y. An analysis of the numerical methods for computing angles between linear subspaces is given in [2]. There are other techniques for computing a basis for the intersection of the subspaces, but the advantage of this method is that it also gives a way of finding vectors which are *almost* in the intersection of the subspaces.

5. Eigenvalues of a matrix modified by a rank one matrix. It is sometimes desirable to determine some eigenvalues of a diagonal matrix which is modified by a matrix of rank one. In this section, we give an algorithm for determining in $O(n^2)$ numerical operations some or all of the eigenvalues and eigenvectors of $D + \sigma\mathbf{u}\mathbf{u}^T$, where $D = \operatorname{diag}(d_i)$ is a diagonal matrix of order n.

Let $C = D + \sigma\mathbf{u}\mathbf{u}^T$; we denote the eigenvalues of C by $\lambda_1, \lambda_2, \cdots, \lambda_n$ and we assume $\lambda_i \leqq \lambda_{i+1}$ and $d_i \leqq d_{i+1}$. It can be shown (cf. [14]) that

(i) if $\sigma \geqq 0$,

$$d_i \leqq \lambda_i \leqq d_{i+1}, \qquad i = 1, 2, \cdots, n-1,$$
$$d_n \leqq \lambda_n \leqq d_n + \sigma\mathbf{u}^T\mathbf{u};$$

(ii) if $\sigma \leqq 0$,

$$d_{i-1} \leqq \lambda_i \leqq d_i, \qquad i = 2, \cdots, n,$$
$$d_1 + \sigma\mathbf{u}^T\mathbf{u} \leqq \lambda_1 \leqq d_1.$$

Thus, we have precise bounds on each of the eigenvalues of C.

The eigenvalues of the matrix C satisfy the equation

$$\det(D + \sigma\mathbf{u}\mathbf{u}^T - \lambda I) = 0,$$

which after some manipulation can be shown to be equivalent to the characteristic equation

$$\varphi_n(\lambda) = \prod_{i=1}^{n} (d_i - \lambda) + \sigma \sum_{i=1}^{n} u_i^2 \prod_{\substack{j=1 \\ j \neq i}}^{n} (d_j - \lambda) = 0. \tag{5.1}$$

Now if we write

$$\varphi_k(\lambda) = \prod_{i=1}^{k} (d_i - \lambda) + \sigma \sum_{i=1}^{k} u_i^2 \prod_{\substack{j=1 \\ j \neq i}}^{k} (d_j - \lambda),$$

then it is easy to verify that

$$\begin{aligned} &\varphi_{k+1}(\lambda) = (d_{k+1} - \lambda)\varphi_k(\lambda) + \sigma u_{k+1}^2 \psi_k(\lambda), && k = 0, 1, \cdots, n-1, \\ &\psi_k(\lambda) = (d_k - \lambda)\psi_{k-1}(\lambda), && k = 1, 2, \cdots, n-1, \end{aligned} \tag{5.2}$$

with

$$\psi_0(\lambda) = \varphi_0(\lambda) = 1.$$

Thus it is a simple matter to evaluate the characteristic equation for any value of λ. Several well-known methods may be used for computing the eigenvalues of C. For instance, it is a simple matter to differentiate the expressions (5.2) with respect to λ and hence determine $\varphi_n'(\lambda)$ for any value of λ. Thus Newton's method can be used in an effective manner for computing the eigenvalues.

An alternative method has been given in [1] and we shall describe that technique. Let K be a bidiagonal matrix of the form

$$K = \begin{bmatrix} 1 & r_1 & & & & \\ & 1 & r_2 & & 0 & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & \cdot & \cdot \\ & 0 & & & \cdot & r_{n-1} \\ & & & & & 1 \end{bmatrix},$$

and let $M = \operatorname{diag}(\mu_i)$. Then KMK^T is a symmetric tridiagonal matrix with elements $\{\mu_k r_{k-1}, (\mu_k + \mu_{k+1} r_k^2), \mu_{k+1} r_k\}_{k=1}^n$, $r_0 = r_n = \mu_{n+1} = 0$.

Consider the matrix equation

$$(D + \sigma \mathbf{u}\mathbf{u}^T)\mathbf{x} = \lambda \mathbf{x}. \tag{5.3}$$

Multiplying (5.3) on the left by K and letting $\mathbf{x} = K^T\mathbf{y}$, we have

$$K(D + \sigma \mathbf{u}\mathbf{u}^T)K^T\mathbf{y} = \lambda K K^T \mathbf{y}$$

or

$$(KDK^T + \sigma K\mathbf{u}\mathbf{u}^T K^T)\mathbf{y} = \lambda K K^T \mathbf{y}.$$

Let us assume that we have reordered the elements of $\mathbf{u}$ (and hence of D, also) so that

$$u_1 = u_2 = \cdots = u_{p-1} = 0 \quad \text{and} \quad 0 < |u_p| \leqq |u_{p+1}| \leqq \cdots \leqq |u_n|.$$

Now it is possible to determine the elements of K so that

$$K\mathbf{u} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ u_n \end{bmatrix}. \tag{5.4}$$

Specifically,

$$r_i = \begin{cases} 0 & \text{for } i < p, \\ -u_i/u_{i+1} & \text{for } i \geqq p, \end{cases}$$

and we note $|r_i| \leqq 1$. Therefore, if $K\mathbf{u}$ satisfies (5.4), we see that $KDK^T + \sigma K\mathbf{u}\mathbf{u}^T K^T$ is a symmetric tridiagonal matrix and so is KK^T. Thus we have a problem of the form $A\mathrm{y} = \lambda B\mathrm{y}$, where A and B are symmetric, tridiagonal matrices and B is positive definite.

Peters and Wilkinson [13] have shown how linear interpolation may be used effectively for computing the eigenvalues of such matrices when the eigenvalues are isolated. The algorithm makes use of $\det(A - \lambda B)$ which is quite simple to compute when A and B are tridiagonal. Once the eigenvalues have been computed it is easy to compute the eigenvectors by inverse iteration. Even if several of the eigenvalues are equal, it is often possible to compute accurate eigenvectors. This can be accomplished by choosing the initial vector in the inverse iteration process to be orthogonal to all the previously computed eigenvectors and by forcing the computed vector after the inverse iteration to be orthogonal to the previously computed eigenvectors. In some unusual situations, however, this procedure may fail.

Another technique which is useful for finding the eigenvalues of (5.3) is to note that if $u_i \neq 0$ for $i = 1, 2, \cdots, n$, then

$$\begin{aligned}\det(D + \sigma\mathbf{u}\mathbf{u}^T - \lambda I) &= \det(D - \lambda I)\det(I + \sigma(D - \lambda I)^{-1}\mathbf{u}\mathbf{u}^T) \\ &= \prod_{i=1}^{n}(d_i - \lambda)\left(1 + \sigma\sum_{i=1}^{n}\frac{u_i^2}{(d_i - \lambda)}\right).\end{aligned}$$

Thus, the eigenvalues of (5.3) can be computed by finding the zeros of the secular equation

$$\omega(\lambda) \equiv 1 + \sigma\sum_{i=1}^{n}\frac{u_i^2}{(d_i - \lambda)}.$$

6. Least squares problems. In this section we shall show how eigenvalue problems arise in linear least squares problems. The first problem we shall consider is that of performing a fit when there is error in the observations and in the data. The approach we take here is a generalization of the one in [9]. Let A be a given $m \times n$ matrix and let $\mathbf{b}$ be a given vector with m components. We wish to construct a vector $\hat{\mathbf{x}}$ which satisfies the constraints

$$(A + E)\mathbf{x} = \mathbf{b} + \boldsymbol{\delta} \tag{6.1}$$

and for which

$$\|P[E\,\vdots\,\boldsymbol{\delta}]Q\| = \text{minimum}, \tag{6.2}$$

where P is a given diagonal matrix with $p_i > 0$, Q is a given diagonal matrix with $q_j > 0$, and $\|\cdot\|$ indicates the Euclidean norm of the matrix. We rewrite (6.1) as

$$[A\,\vdots\,\mathbf{b}]\begin{bmatrix}\mathbf{x} \\ -1\end{bmatrix} + [E\,\vdots\,\boldsymbol{\delta}]\begin{bmatrix}\mathbf{x} \\ -1\end{bmatrix} = \mathbf{0},$$

or equivalently as

$$By + Fy = \mathbf{0}, \tag{6.3}$$

where

$$B = [A \vdots \mathbf{b}]Q, \quad F = [E \vdots \boldsymbol{\delta}]Q, \quad \mathbf{y} = Q^{-1}\begin{bmatrix} \mathbf{x} \\ -1 \end{bmatrix}. \tag{6.4}$$

Our problem now is to determine $\hat{\mathbf{y}}$ so that (6.3) is satisfied, and

$$\|PF\| = \text{minimum}.$$

Again we use Lagrange multipliers as a device for minimizing $\|PF\|$ subject to (6.3).

Consider the function

$$\varphi(F, \mathbf{y}, \boldsymbol{\lambda}) = \sum_{i=1}^{m} \sum_{j=1}^{n+1} p_i^2 f_{ij}^2 - 2 \sum_{i=1}^{m} \lambda_i \sum_{j=1}^{n+1} (b_{ij} + f_{ij}) y_j. \tag{6.5}$$

Then

$$\frac{\partial \varphi}{\partial f_{rs}} = 2p_r^2 f_{rs} - 2\lambda_r y_s$$

so that we have a stationary point of (6.5) when

$$P^2 F = \boldsymbol{\lambda}\mathbf{y}^T. \tag{6.6}$$

Note that the matrix F must be of rank one. Substituting (6.6) into (6.3) we have

$$\boldsymbol{\lambda} = -P^2 B\mathbf{y}/(\mathbf{y}^T\mathbf{y}),$$

and hence,

$$PF = -PB\mathbf{y}\mathbf{y}^T/(\mathbf{y}^T\mathbf{y}).$$

Thus,

$$\|PF\|^2 = \mathbf{y}^T B^T P^2 B\mathbf{y}/(\mathbf{y}^T\mathbf{y}),$$

and hence $\|PF\| = $ minimum when $\hat{\mathbf{y}}$ is the eigenvector associated with the smallest eigenvalue of $B^T P^2 B$. Of course a more accurate procedure is to compute the smallest singular value of PB.

Then, in order to compute $\hat{\mathbf{x}}$, we perform the following calculations:

(a) Form the singular value decomposition of PB, viz.,

$$PB = U\Sigma V^T.$$

(It is generally not necessary to compute U.)

(b) Let $\mathbf{v}$ be the column vector of V associated with $\sigma_{\min}(PB)$ so that $\mathbf{v} = \hat{\mathbf{y}}$. Compute

$$\mathbf{z} = Q\mathbf{v}.$$

(c) From (6.4),

$$\begin{bmatrix} \hat{\mathbf{x}} \\ -1 \end{bmatrix} = -\frac{1}{z_{n+1}}\mathbf{z}.$$

Note that $\min\|PF\| = \sigma_{\min}(PB)$, and that

$$[E \vdots \boldsymbol{\delta}] = -[A \vdots \mathbf{b}]\mathbf{v}\mathbf{v}^T Q^{-1}.$$

The solution will not be unique if the smallest singular value is multiple. Furthermore, it will not be possible to compute the solution if $z_{n+1} = 0$. This will occur, for example, if $P = I_m$, $Q = I_{n+1}$, $A^T\mathbf{b} = \mathbf{0}$ and $\sigma_{\min}(A) < \|\mathbf{b}\|_2$.

Another problem which arises frequently is that of finding a least squares solution with a quadratic constraint; we have considered this problem previously in [1]. We seek a vector $\hat{\mathbf{x}}$ such that

$$\|\mathbf{b} - A\mathbf{x}\|_2 = \text{minimum} \tag{6.7}$$

with the constraint that

$$\|\mathbf{x}\|_2 = \alpha. \tag{6.8}$$

The condition (6.8) is frequently imposed when the matrix A is ill-conditioned. Now let

$$\varphi(\mathbf{x}, \alpha) = (\mathbf{b} - A\mathbf{x})^T(\mathbf{b} - A\mathbf{x}) + \lambda(\mathbf{x}^T\mathbf{x} - \alpha^2), \tag{6.9}$$

where λ is a Lagrange multiplier. Differentiating (6.9), we are led to the equation

$$A^TA\mathbf{x} - A^T\mathbf{b} + \lambda\mathbf{x} = \mathbf{0} \tag{6.10}$$

or

$$(A^TA + \lambda I)\mathbf{x} = A^T\mathbf{b}. \tag{6.11}$$

Note that (6.10) represents the usual normal equations that arise in the linear least squares problem, with the diagonal elements of A^TA shifted by λ. The parameter λ will be positive when

$$\alpha < \|A^+\mathbf{b}\|_2,$$

and we assume that this condition is satisfied.

Since $\mathbf{x} = (A^TA + \lambda I)^{-1}A^T\mathbf{b}$, we have from (6.8) that

$$\mathbf{b}^TA(A^TA + \lambda I)^{-2}A^T\mathbf{b} - \alpha^2 = 0. \tag{6.12}$$

By repeated use of the identity

$$\det\begin{bmatrix} X & Y \\ Z & W \end{bmatrix} = \det(X)\det(W - ZX^{-1}Y) \quad \text{if } \det(X) \neq 0,$$

we can show that (6.12) is equivalent to the equation

$$\det((A^TA + \lambda I)^2 - \alpha^{-2}A^T\mathbf{b}\mathbf{b}^TA) = 0. \tag{6.13}$$

Finally if $A = U\Sigma V^T$, the singular value decomposition of A, then

$$A^TA = VDV^T, \qquad V^TV = I, \tag{6.14}$$

where $D = \Sigma^T\Sigma$ and (6.13) becomes

$$\det((D + \lambda I)^2 - \mathbf{u}\mathbf{u}^T) = 0, \tag{6.15}$$

where $\mathbf{u} = \alpha^{-1}\Sigma^T U^T \mathbf{b}$. Equation (6.15) has $2n$ roots; it can be shown (cf. [6]) that we need the largest real root of (6.15) which we denote by λ^*. By a simple argument, it can be shown that λ^* is the unique root in the interval $[0, \mathbf{u}^T\mathbf{u}]$. Thus we have the problem of determining an eigenvalue of a diagonal matrix which is modified by a matrix of rank one.

As in § 5, we can determine a matrix K so that $K\mathbf{u}$ satisfies (5.4), and hence (6.15) is equivalent to

$$\det(K(D + \lambda I)^2 K^T - K\mathbf{u}\mathbf{u}^T K^T) = 0. \tag{6.16}$$

The matrix $G(\lambda) = K(D + \lambda I)^2 K^T - K\mathbf{u}\mathbf{u}^T K^T$ is tridiagonal so that it is easy to evaluate $G(\lambda)$ and $\det G(\lambda)$. Since we have an upper and lower bound on λ^*, it is possible to use linear interpolation to find λ^*, even though $G(\lambda)$ is quadratic in λ. Numerical experiments have indicated it is best to compute $G(\lambda) = K(D + \lambda I)^2 K^T - K\mathbf{u}\mathbf{u}^T K^T$ for each approximate value of λ^* rather than computing

$$G(\lambda) = (KD^2K^T - K\mathbf{u}\mathbf{u}^T K^T) + 2\lambda KDK^T + \lambda^2 KK^T.$$

Another approach for determining λ^* is the following: we substitute the decomposition (6.14) into (6.12) and are led to the equation

$$\varphi_n(\lambda) \equiv \sum_{i=1}^{n} \frac{u_i^2}{(d_i + \lambda)^2} - 1 = 0, \tag{6.17}$$

with $\mathbf{u} = \alpha^{-1}\Sigma^T U^T \mathbf{b}$. It is easy to verify that if

$$\begin{aligned} \psi_k(\lambda) &= \prod_{j=1}^{k} (d_i + \lambda)^2 \left[\sum_{i=1}^{k} \frac{u_i^2}{(d_i + \lambda)^2} - 1 \right], \\ \psi_{k+1}(\lambda) &= (d_{k+1} + \lambda)^2 \psi_k(\lambda) - u_{k+1}^2 \xi_k(\lambda), && k = 0, 1, \cdots, n-1, \\ \xi_k(\lambda) &= (d_k + \lambda)^2 \xi_{k-1}(\lambda), && k = 1, 2, \cdots, n-1, \end{aligned} \tag{6.18}$$

with

$$\psi_0(\lambda) = \xi_0(\lambda) = 1.$$

Then using (6.18) we can easily evaluate $\psi_n(\lambda)$ and $\psi_n'(\lambda)$, and hence use one of the standard root finding techniques for determining λ^*. It is easy to verify that $\hat{\mathbf{x}} = V(D + \lambda^* I)^{-1}\Sigma U^T \mathbf{b}$.

A similar problem arises when it is required to make

$$\|\mathbf{x}\|_2 = \text{minimum}$$

when

$$\|\mathbf{b} - A\mathbf{x}\|_2 = \beta,$$

where

$$\beta > \min_{\mathbf{x}} \|\mathbf{b} - A\mathbf{x}\|.$$

Again the Lagrange multiplier λ satisfies a quadratic equation which is similar to the equation given by (6.14).

7. Gauss-type quadrature rules with preassigned nodes. In many applications it is desirable to generate Gauss-type quadrature rules with preassigned nodes. This is particularly true for numerical methods which depend on the theory of moments for determining bounds (cf. [3]), and for solving boundary value problems [12]. We shall show that it is possible to generate these quadrature rules as a modified eigenvalue problem.

Let $\omega(x) \geqq 0$ be a fixed *weight function* defined on the interval $[a, b]$. For $\omega(x)$ it is possible to define a sequence of polynomials $p_0(x)$, $p_1(z)$, $\cdots$ which are orthonormal with respect to $\omega(x)$ and in which $p_n(x)$ is of exact degree n so that

$$\int_a^b p_n(x)p_m(x)\omega(x) = \begin{cases} 1 & \text{when } m = n, \\ 0 & \text{when } m \neq n. \end{cases}$$

The polynomial $p_n(x) = k_n \prod_{i=1}^n (x - t_i^{(n)})$, $k_n > 0$, has n distinct real roots $a < t_1^{(n)} < t_2^{(n)} < \cdots < t_n^{(n)} < b$. The roots of the orthogonal polynomials play an important role in Gauss-type quadrature.

THEOREM. *Let $f(x) \in C^{2N}[a, b]$; then it is possible to determine positive w_j so that*

$$\int_a^b f(x)\omega(x)\,dx = \sum_{j=1}^N w_j f(t_j^{(N)}) = R[f],$$

where

$$R[f] = \frac{f^{(2N)}(\eta)}{(2N)!} \int_a^b \left[\prod_{i=1}^N (x - t_i^{(N)}) \right]^2 \omega(x)\,dx, \qquad a < \eta < b.$$

Thus, the Gauss-type quadrature rule is exact for all polynomials of degree less than or equal to $2N - 1$.

Any set of orthonormal polynomials satisfies a three-term recurrence relationship:

$$\begin{aligned} &\beta_j p_j(x) = (x - \alpha_j)p_{j-1}(x) - \beta_{j-1}p_{j-2}(x) \quad \text{for } j = 1, 2, \cdots, N; \\ &p_{-1}(x) \equiv 0, \qquad p_0(x) \equiv 1. \end{aligned} \tag{7.1}$$

We may identify (7.1) with the matrix equation

$$x\mathbf{p}(x) = J_N\mathbf{p}(x) + \beta_N p_N(x)\mathbf{e}_N, \tag{7.2}$$

where

$$[\mathbf{p}(x)]^T = [p_0(x), p_1(x), \cdots, p_{N-1}(x)],$$
$$\mathbf{e}_N^T = [0, 0, \cdots, 1],$$

and

$$J_N = \begin{bmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & 0 & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \beta_{N-1} \\ & 0 & & & \cdot & \cdot \\ & & & & \beta_{N-1} & \alpha_N \end{bmatrix}.$$

Suppose that the eigenvalues of J_N are computed so that

$$J_N\mathbf{q}_j = \lambda_j\mathbf{q}_j, \qquad j = 1, 2, \cdots, N,$$

with

$$\mathbf{q}_j^T\mathbf{q}_j = 1$$

and

$$\mathbf{q}_j^T = [q_{1j}, q_{2j}, \cdots, q_{Nj}].$$

Then it is shown in [11] that

$$(7.3) \qquad t_j^{(N)} = \lambda_j, \qquad w_j = (q_{1j})^2.$$

(From here on in, we drop the superscripts on the $t_j^{(N)}$'s.)

A very effective way to compute the eigenvalues of J_N and the first component of the orthonormalized eigenvectors is to use the QR method of Francis (cf. [14]).

Now let us consider the problem of determining the quadrature rule so that

$$\int_a^b f(x)\omega(x)\,dx \approx \sum_{j=1}^N w_j f(t_j) + \sum_{k=1}^M v_k f(z_k),$$

where the nodes $\{z_k\}_{k=1}^M$ are prescribed. It is possible to determine $\{w_j, t_j\}_{j=1}^N$, $\{v_k\}_{k=1}^M$ so that we have for the remainder

$$R[f] = \frac{f^{(2N+M)}(\eta)}{(2N+M)!}\int_a^b \prod_{k=1}^M (x - v_k)\left[\prod_{j=1}^N (x - t_j)\right]^2 \omega(x)\,dx, \qquad a < \eta < b.$$

For $M = 1$ and $z_1 = a$ or $z_1 = b$, we have the Gauss–Radau-type formula, and for $M = 2$ with $z_1 = a$ and $z_2 = b$, we have the Gauss–Lobatto-type formula.

First we shall show how the Gauss–Radau-type rule may be computed. For convenience, we assume that $z_1 = a$. Now we wish to determine the polynomial $p_{N+1}(x)$ so that

$$p_{N+1}(a) = 0.$$

From (7.1) we see that this implies that

$$0 = p_{N+1}(a) = (a - \alpha_{N+1})p_N(a) - \beta_N p_{N-1}(a)$$

or

$$(7.4) \qquad \alpha_{N+1} = a - \beta_N \frac{p_{N-1}(a)}{p_N(a)}.$$

From equation (7.2) we have

$$(J_N - aI)\mathbf{p}(a) = -\beta_N p_N(a)\mathbf{e}_N,$$

or equivalently,

$$(7.5) \qquad (J_N - aI)\boldsymbol{\delta}(a) = \beta_N^2\mathbf{e}_N,$$

where

$$\delta_j(a) = -(\beta_N p_{j-1}(a))/p_N(a), \qquad j = 1, 2, \cdots, N.$$

Thus,

$$\alpha_{N+1} = a + \delta_N(a). \tag{7.6}$$

Hence, in order to compute the Gauss–Radau-type rule, we do the following:

(a) Generate the matrix J_N and the element β_N.
(b) Solve the system of equations (7.5) for $\delta_N(a)$.
(c) Compute α_{N+1} by (7.6) and use it to replace the $(N + 1, N + 1)$ element of J_{N+1}.
(d) Use the QR algorithm to compute the eigenvalues and first element of the eigenvector of the tridiagonal matrix

$$\tilde{J}_{N+1} = \left[\begin{array}{c|c} J_N & \beta_N \mathbf{e}_N \\ \hline \beta_N \mathbf{e}_N^T & \alpha_{N+1} \end{array}\right].$$

Of course, one of the eigenvalues of the matrix J_{N+1} must be equal to a.

Since $a < \lambda_{\min}(J_N)$, the matrix $J_N - aI$ will be positive definite and hence Gaussian elimination without pivoting may be used to solve (7.5). It is not even necessary to solve the complete system since it is only necessary to compute the element $\delta_N(a)$. However, one may wish to use iterative refinement to compute $\delta_N(a)$ very precisely since for N large, $\lambda_{\min}(J)$ may be close to a, and hence the system of equations (7.5) may be quite ill-conditioned. When $z_1 = b$, the calculation of $\tilde{J}_{N+1}$ is identical except with b replacing a in equations (7.5) and (7.6). The matrix $J_N - bI$ will be negative definite since $b > \lambda_{\max}(J)$.

To compute the Gauss–Lobatto quadrature rule, we need to compute a matrix $\tilde{J}_{N+1}$ such that

$$\lambda_{\min}(\tilde{J}_{N+1}) = a \quad \text{and} \quad \lambda_{\max}(\tilde{J}_{N+1}) = b.$$

Thus, we wish to determine $p_{N+1}(x)$ so that

$$p_{N+1}(a) = p_{N+1}(b) = 0. \tag{7.7}$$

Now from (7.1) we have

$$\beta_{N+1} p_{N+1}(x) = (x - \alpha_{N+1}) p_N(x) - \beta_N p_{N-1}(x),$$

so that (7.7) implies that

$$\begin{aligned} \alpha_{N+1} p_N(a) + \beta_N p_{N-1}(a) &= a p_N(a), \\ \alpha_{N+1} p_N(b) + \beta_N p_{N-1}(b) &= b p_N(b). \end{aligned} \tag{7.8}$$

Using the relationship (7.2), if

$$(J_N - aI)\,\gamma = \mathbf{e}_N \quad \text{and} \quad (J_N - bI)\boldsymbol{\mu} = \mathbf{e}_N, \tag{7.9}$$

then

$$\gamma_j = -\frac{1}{\beta_N}\frac{p_{j-1}(a)}{p_N(a)}, \qquad \mu_j = -\frac{1}{\beta_N}\frac{p_{j-1}(b)}{p_N(b)}, \qquad j = 1, 2, \cdots, N. \tag{7.10}$$

Thus, (7.8) is equivalent to the system of equations

$$\alpha_{N+1} - \gamma_N \beta_N^2 = a, \qquad \alpha_{N+1} - \mu_N \beta_N^2 = b. \tag{7.11}$$

Hence, in order to compute the Gauss–Lobatto-type rule, we perform the following calculations:

(a) Generate the matrix J_N.

(b) Solve the systems of equations (7.9) for γ_N and μ_N.

(c) Solve (7.11) for α_{N+1} and β_N^2.

(d) Use the QR algorithm to compute the eigenvalues and first element of the eigenvectors of the tridiagonal matrix

$$\tilde{J}_{N+1} = \left[\begin{array}{c|c} J_N & \beta_N \mathbf{e}_N \\ \hline \beta_N \mathbf{e}_N^T & \alpha_{N+1} \end{array}\right].$$

Galant [7] has given an algorithm for computing the Gaussian-type quadrature rules with preassigned nodes which is based on a theorem of Christoffel. His method constructs the orthogonal polynomials with respect to a modified weight function.

Acknowledgments. The author wishes to thank Mr. Michael Saunders for making several helpful suggestions for improving the presentation, Mrs. Erin Brent for performing so carefully some of the calculations indicated in §§ 5 and 6, Miss Linda Kaufman for expertly performing numerical experiments associated with § 7. Special thanks to Dr. C. Paige for his careful reading of this manuscript and his many excellent suggestions. The referees also made several beneficial comments.

REFERENCES

[1] R. H. Bartels, G. H. Golub and M. A. Saunders, *Numerical techniques in mathematical programming*, Nonlinear Programming, J. B. Rosen, O. L. Mangasarian and K. Ritter, eds., Academic Press, New York, 1970, pp. 123–176.

[2] A. Björck and G. H. Golub, *Numerical methods for computing angles between linear subspaces*, Math. Comp., to appear.

[3] G. Dahlquist, S. Eisenstat and G. H. Golub, *Bounds for the error of linear systems of equations using the theory of moments*, J. Math. Anal. Appl., 37 (1972), pp. 151–166.

[4] J. Durbin, *An alternative to the bounds test for testing for serial correlation in least squares regression*, Econometrica, 38 (1970), pp. 422–429.

[5] G. Fix and R. Heiberger, *An algorithm for the ill-conditioned generalized eigenvalue problem*, Numer. Math., to appear.

[6] G. E. Forsythe and G. H. Golub, *On the stationary values of a second degree polynomial on the unit sphere*, SIAM J. Appl. Math., 13 (1965), pp. 1050–1068.

[7] D. Galant, *An implementation of Christoffel's theorem in the theory of orthogonal polynomials*, Math. Comp., 24 (1971), pp. 111–113.

[8] G. H. Golub, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206–216.

[9] G. H. Golub and C. Reinsch, *Singular value decomposition and least squares solutions*, Ibid., 14 (1970), pp. 403–420.

[10] G. H. Golub and R. Underwood, *Stationary values of the ratio of quadratic forms subject to linear constraints*, Z. Angew. Math. Phys., 21 (1970), pp. 318–326.

[11] G. H. Golub and J. H. Welsch, *Calculation of Gauss quadrature rules*, Math. Comp., 23 (1969), pp. 221–230.

[12] V. I. Krylov, *Approximate Calculation of Integrals*, Macmillan, New York, 1962.

[13] G. Peters and J. H. Wilkinson, *Eigenvalues of* $A\mathbf{x} = \lambda B\mathbf{x}$ *with band symmetric A and B*, Comput. J., 12 (1969), pp. 398–404.

[14] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

28

ILL-CONDITIONED EIGENSYSTEMS AND THE COMPUTATION OF THE JORDAN CANONICAL FORM (WITH J. H. WILKINSON)

SIAM Review (18), 578–619 (1976).

James Hardy Wilkinson

Born 27 September 1919, in Strood, Kent, England;
Died 5 October 1986, in Teddington, Middlesex, England.

SIAM Review
Vol. 18, No. 4, October 1976

ILL-CONDITIONED EIGENSYSTEMS AND THE COMPUTATION OF THE JORDAN CANONICAL FORM*

G. H. GOLUB† AND J. H. WILKINSON‡

Abstract. The solution of the complete eigenvalue problem for a nonnormal matrix A presents severe practical difficulties when A is defective or close to a defective matrix. Moreover, in the presence of rounding errors, one cannot even determine whether or not a matrix is defective. Several of the more stable methods for computing the Jordan canonical form are discussed, together with the alternative approach of computing well-defined bases (usually orthogonal) of the relevant invariant subspaces.

1. Introduction. From the standpoint of classical algebra, the algebraic eigenvalue problem has been completely solved. The problem is the subject of classical *similarity* theory, and the fundamental result is embodied in the Jordan canonical form (J.c.f.). Most mathematicians encounter similarity theory in an abstract setting, but since we are concerned here with practical algorithms, we first review the basic result purely in matrix terms.

The J.c.f. is described with reference to matrices known as *elementary Jordan* blocks. A Jordan block of order r associated with an eigenvalue λ_i will be denoted by $J_r(\lambda_i)$, and its general form is adequately illustrated by the definition

$$J_4(\lambda_i) = \begin{bmatrix} \lambda_i & 1 & 0 & 0 \\ 0 & \lambda_i & 1 & 0 \\ 0 & 0 & \lambda_i & 1 \\ 0 & 0 & 0 & \lambda_i \end{bmatrix}. \tag{1.1}$$

The basic theorem is that given any $n \times n$ matrix with complex elements, there exists a nonsingular matrix X such that

$$X^{-1}AX = J, \qquad AX = XJ, \tag{1.2}$$

where J, the J.c.f. of A, is block diagonal, each diagonal matrix being an elementary Jordan block. Apart from the ordering of the blocks along the diagonal of J (which can be arbitrary), the J.c.f. is unique, although X is far from unique. It will be convenient to order the blocks in some standard way. Unless reference is made to the contrary, we assume that the $|\lambda_i|$ are in order of nonincreasing magnitude and that the blocks associated with a specific λ_i are ordered to be of nondecreasing size. Thus if the matrix A of order 12 has only 2 distinct eigenvalues λ_1 and λ_2 with $|\lambda_1| \geqq |\lambda_2|$, and λ_1 is associated with 2 blocks of order 2 and one of order 3 while

* Received by the editors January 17, 1975, and in revised form July 8, 1975. This invited paper was prepared in part and published under Contract DA HC 19-69-C-0022 with the U.S. Army Reserve Office.

† Computer Science Department, Stanford University, Stanford, California 94305. This work was supported in part by the National Science Foundation under Grant GJ35135X and by the Atomic Energy Commission under Grant AT(04-3)-326PA #30.

‡ Division of Numerical Analysis and Computing, Department of Industry, National Physical Laboratory, Teddington, Middlesex, England.

λ_2 is associated with one block of order 2 and one of order 3, its J.c.f. will be presented in the form

$$(1.3)\qquad \begin{bmatrix} J_2(\lambda_1) & & & & \\ & J_2(\lambda_1) & & & \\ & & J_3(\lambda_1) & & \\ & & & J_2(\lambda_2) & \\ & & & & J_3(\lambda_2) \end{bmatrix}.$$

Here λ_1 is an eigenvalue of multiplicity $2 + 2 + 3 = 7$ and λ_2 of multiplicity $2 + 3 = 5$. The example illustrates that there may be more than one block of a given dimension associated with a specific λ_i.

Let us consider the significance of the existence of a block $J_r(\lambda_i)$ in J, where $J_r(\lambda_i)$ starts in rows and columns s and ends in rows and columns t, and

$$(1.4)\qquad r = t - s + 1.$$

Equating columns s to t on both sides of equation (1.2), we have

$$(1.5)\qquad \begin{array}{ll} Ax_s \;\;\;= \lambda_i x_s, & (A - \lambda_i I)x_s \;\;\;= 0, \\ Ax_{s+1} = \lambda_i x_{s+1} + x_s, & (A - \lambda_i I)x_{s+1} = x_s, \\ Ax_{s+2} = \lambda_i x_{s+2} + x_{s+1}, & (A - \lambda_i I)x_{s+2} = x_{s+1}, \\ \cdots\cdots\cdots\cdots\cdots\cdots & \cdots\cdots\cdots\cdots\cdots\cdots \\ Ax_t \;\;\;= \lambda_i x_t + x_{t-1}, & (A - \lambda_i I)x_t \;\;\;= x_{t-1}, \end{array}$$

where, here and later, we shall denote the ith column of a matrix X (say) by x_i. The first of these relations implies that x_s is an eigenvector corresponding to λ_i. The remaining equations imply that

$$(1.6)\qquad \begin{array}{l} (A - \lambda_i I)^2 x_{s+1} = 0, \; (A - \lambda_i I)^3 x_{s+2} = 0, \\ \qquad\qquad \cdots, (A - \lambda_i I)^{t-s+1} x_t \equiv (A - \lambda_i I)^r x_t = 0. \end{array}$$

Notice that in general the x_{s+i} satisfy the relations

$$(1.7)\qquad (A - \lambda_i I)^{p-1} x_{s+p-1} = x_s \neq 0 \quad \text{and} \quad (A - \lambda_i I)^p x_{s+p-1} = 0.$$

We shall refer to *any vector* x such that $(A - \lambda I)^{p-1}x \neq 0$, $(A - \lambda I)^p x = 0$, as a *vector of grade p*, and for uniformity, an eigenvector becomes a vector of grade 1. It is evident, for example, that

$$(1.8)\qquad \begin{array}{l} (A - \lambda I)^2(\alpha_2 x_{s+2} + \alpha_1 x_{s+1} + \alpha_0 x_s) = \alpha_2 x_s, \\ (A - \lambda I)^3(\alpha_2 x_{s+2} + \alpha_1 x_{s+1} + \alpha_0 x_s) = 0, \end{array}$$

so that $\alpha_2 x_{s+2} + \alpha_1 x_{s+1} + \alpha_0 x_s$ is a vector of grade 3 for all α_i provided $\alpha_2 \neq 0$. The vectors x_{s+i} arising in the Jordan canonical reduction are special in that they satisfy the *chain* relations (1.5). We shall refer to the vectors of grades $1, 2, 3, \cdots$ associated with a Jordan block as *principal vectors of grades* $1, 2, 3, \cdots$.

Clearly $\det(\lambda I - J_r(\lambda_i)) = (\lambda - \lambda_i)^r$, and we may associate such a polynomial with each of the blocks in the J.c.f. These polynomials are called the *elementary* *divisors of A*. An enumeration of the elementary divisors gives a unique specification of the J.c.f. Corresponding to a Jordan block of dimension unity the elementary divisor is $(\lambda - \lambda_i)$, i.e., it is linear. If all the Jordan blocks in the J.c.f. are of dimension unity, then the J.c.f. is *strictly* diagonal, the matrix has n independent eigenvectors given by the columns of X and all the elementary divisors are linear. These four properties are fully equivalent to each other. Notice that if there are n distinct λ_i, then all the blocks are necessarily of dimension unity. Departure from strict diagonal form can occur only if there is at least one multiple eigenvalue, though even in this case the J.c.f. can be diagonal.

A matrix is said to be *defective* if the J.c.f. is not strictly diagonal. In this case, at least one elementary divisor is nonlinear and the number of independent eigenvectors is less than n; the remaining columns of X are principal vectors of the appropriate grades.

A matrix is said to be *derogatory* if there is at least one λ_i which is associated with more than one diagonal block in the J.c.f. If such a λ_i is associated with k different blocks, then there are precisely k independent eigenvectors associated with λ_i.

It should be emphasized that a matrix may be defective without being derogatory and vice versa, or it can be both defective and derogatory. If the λ_i are distinct, it cannot be either. If A is normal (including Hermitian, skew Hermitian or unitary), then *its J.c.f. is always strictly diagonal, and the X producing the* J.c.f. may be chosen to be unitary. A normal matrix with a multiple eigenvalue is therefore derogatory but not defective.

We do not report on numerical experiments in this paper, although many of the algorithms described have been implemented with success. It is the aim of this paper to emphasize the problems associated with computing invariant subspaces and to stimulate research in this area. We have not attempted to be encyclopedic (despite the length of the paper) but state those principles which we feel are of importance in this area.

2. Linear differential equations and the J.c.f. The practical significance of the J.c.f. of a matrix A is that it provides the general solution of the associated system of linear differential equations with constant coefficients defined by

$$\frac{du}{dt} = Au, \tag{2.1}$$

where u is a vector of order n. Under the linear transformation $u = Xv$, the equation becomes

$$X\frac{dv}{dt} = AXv \quad \text{or} \quad \frac{dv}{dt} = X^{-1}AXv = Jv. \tag{2.2}$$

Hence the J.c.f. gives a simplified version of the original system. If J is strictly diagonal (i.e., A is not defective), the transformed system is

$$\frac{dv_i}{dt} = \lambda_i v_i, \tag{2.3}$$

and in terms of variables v_i, the equations are completely decoupled. The general solution is

$$v_i = v_i^{(0)} e^{\lambda_i t}, \qquad u = \sum v_i^{(0)} x_i e^{\lambda_i t}, \tag{2.4}$$

and is therefore directly expressible in terms of the n independent eigenvectors x_i and n independent constants $v_i^{(0)}$, the initial values of the v_i. Notice that the analysis is not affected by any multiplicities in the λ_i provided J is strictly diagonal. An eigenvalue λ_i of multiplicity r is then associated with r independent eigenvectors and r arbitrary $v_j^{(0)}$. When A is defective, the linear transformation does not give a complete decoupling of the equations, but there is a decoupling of those equations involving the v_i associated with each specific block from those associated with all other v_j. The general solution is most readily exposed in terms of the concept of the "exponential" of a matrix. We define $\exp(B)$ by the relation

$$\exp(B) = I + \frac{1}{1!}B + \frac{1}{2!}B^2 + \cdots + \frac{1}{r!}B^r + \cdots, \tag{2.5}$$

the matrix series being convergent for all B. The solution of (2.1) such that $u = u^{(0)}$ when $t = 0$ is given by

$$u = \exp(At)u^{(0)}. \tag{2.6}$$

From the series expansion it will readily be verified that

$$\exp(XBX^{-1}t) = X\exp(Bt)X^{-1}, \tag{2.7}$$

and hence the solution of (2.1) is

$$u = X\exp(Jt)X^{-1}u^{(0)}$$

or

$$v = \exp(Jt)v^{(0)}, \quad \text{where } v = X^{-1}u. \tag{2.8}$$

If $J_r(\lambda_i)$ is a typical block in J, then $\exp(Jt)$ has the same block structure, with $\exp(J_r(\lambda_i)t)$ in place of each $J_r(\lambda_i)$, and the form of $\exp(J_r(\lambda_i)t)$ is fully illustrated by the relation

$$\exp(J_4(\lambda_i)t) = \exp(\lambda_i t)\begin{bmatrix} 1 & t/1! & t^2/2! & t^3/3! \\ & 1 & t/1! & t^2/2! \\ & & 1 & t/1! \\ & & & 1 \end{bmatrix}. \tag{2.9}$$

Hence on transforming back from the v-coordinates to the u-coordinates, the solution corresponding to the initial problem is again given in terms of the vectors x_i but corresponding to a Jordan block $J_r(\lambda_i)$, terms involving $\exp(\lambda_i t)t^s/s!$ $(s = 0, \cdots, r-1)$ arise.

This discussion gives the impression that the theoretical significance of the J.c.f. is fully matched by its practical importance since it is precisely because of its relationship to the solution of systems of linear differential equations that the algebraic eigenvalue problem occupies such a prominent position in practical applied mathematics. The principal objective of the remainder of this paper is to show the basic limitations of the J.c.f. from the point of view of practical computation and, indeed, to cast doubt on the advisability of trying to determine it.

Before proceeding, it is useful to consider the degree of arbitrariness in the matrix X involved in the reduction to J.c.f. If the λ_i are distinct, J is diagonal and the x_i are the unique eigenvectors. The only degree of arbitrariness is in the scaling of the x_i. We have

$$D^{-1}X^{-1}AXD = D^{-1}JD = J, \tag{2.10}$$

where D is a nonsingular diagonal matrix.

Turning now to the case where J has a single block of dimension r, we see that there is already a wide freedom of choice in X. Suppose, for illustration, that there is a block of order 4 associated with λ_i; then from (1.5) we see, writing $B \equiv A - \lambda_i I$, that

$$\begin{aligned} B(ax_{s+3} + bx_{s+2} + cx_{s+1} + dx_s) &= ax_{s+2} + bx_{s+1} + cx_s, \\ B(ax_{s+2} + bx_{s+1} + cx_s) &= ax_{s+1} + bx_s, \\ B(ax_{s+1} + bx_s) &= ax_s, \\ B(ax_s) &= 0, \end{aligned} \tag{2.11}$$

where the a, b, c, d are arbitrary, but $a \neq 0$. Hence the chain of vectors x_{s+3}, x_{s+2}, x_{s+1}, x_s may be replaced by the chain of vectors given in (2.11) and on this account X may be replaced by XP, where

$$P = \left[\begin{array}{c|cccc|c} I & & & & & \\ \hline & a & b & c & d & \\ & & a & b & c & \\ & & & a & b & \\ & & & & a & \\ \hline & & & & & I \end{array}\right] \tag{2.12}$$

The derogatory case, i.e., the case when there is more than one block associated with a given λ_i, may be illustrated by the case when there are blocks of orders 2 and 3 starting in positions s and t, respectively. From the two chains

$$\begin{aligned} Bx_s &= 0, \\ Bx_{s+1} &= x_s, \\ Bx_{s+2} &= x_{s+1}, \end{aligned} \qquad Bx_t = 0, \tag{2.13}$$

the two generalized chains defined by

$$\begin{aligned} &B(ax_{s+2} + bx_{s+1} + cx_s + dx_{t+1} + ex_t) = ax_{s+1} + bx_s + dx_t, \\ (2.14\text{a})\qquad &B(ax_{s+1} + bx_s + dx_t) \qquad\qquad\qquad = ax_s, \\ &B(ax_s) \qquad\qquad\qquad\qquad\qquad\qquad\qquad = 0, \end{aligned}$$

and

$$\begin{aligned} &B(fx_{s+1} + gx_s + hx_{t+1} + ix_t) \qquad = fx_s + hx_t, \\ (2.14\text{b})\qquad &B(fx_s + hx_t) \qquad\qquad\qquad\qquad = 0, \end{aligned}$$

may be derived, where the $a, b, \cdots, i$ are arbitrary, except that $a \neq 0$, $h \neq 0$, and X may be varied correspondingly.

3. Sensitivity of the eigenvalues of a defective matrix. Blocks of dimension greater than unity in the J.c.f. can emerge, if at all, only as the result of the presence of multiple eigenvalues. In the classical theory there is a clear-cut distinction between equal and unequal eigenvalues. In practice, the situation is very different since a matrix may not be representable exactly in the computer and, in any case, rounding errors are, in general, involved in computing transformations. Let us consider the effect of small perturbations on the eigenvalues of an elementary Jordan block $J_r(\lambda_i)$. If the zero element in position $(r, 1)$ is replaced by ε, the characteristic equation

$$(3.1)\qquad (\lambda - \lambda_i)^r = \varepsilon$$

and the multiple eigenvalue λ_i is replaced by r distinct eigenvalues $\lambda_i + \varepsilon^{1/r}(\cos(2s\pi/r) + i\sin(2s\pi/r))$ $(s = 0, \cdots, r - 1)$. Suppose λ_i is of order unity, $r = 10$ and $\varepsilon = 10^{-10}$. Then the separation of the perturbed roots is of order 10^{-1} and they cannot in any reasonable sense be regarded as "close".

In practice, we have to diagnose multiplicities and the degree of defectiveness from computed eigenvalues. When these are determined by a very stable algorithm, we cannot rely on any of them being recognizably "close", even when the given A really does have some multiple eigenvalues. When A has an elementary divisor of high degree, this danger appears to be particularly severe.

However, even this remark somewhat oversimplifies the situation. One tends to be seduced by the simplicity of the J.c.f. and as a result to attach too much significance to every detail of it. When attempting to construct "difficult" matrices for practical experiments, it is common to take a nondiagonal J.c.f., subject it to some exact similarity transformation and then to regard the resulting matrix as wholly typical of a defective matrix.

But this is to attach too much significance to the unity elements in the Jordan blocks. If $D = \operatorname{diag}(d_i)$ is any nonsingular diagonal matrix, then from (1.2) we have

$$(3.2)\qquad D^{-1}X^{-1}AXD = D^{-1}JD.$$

Hence if J has a unity element in position $(p, p + 1)$, the matrix $D^{-1}JD$ has d_p^{-1}, d_{p+1} in this position; by a suitable choice of the d_i the unity elements may be given arbitrary values. The choice of the unity elements in the J.c.f. is purely for notational convenience. However, in classical mathematics we can make a sharp distinction between zero and nonzero elements, a luxury we are denied in practical computation. We refer to a matrix as being in *quasi-J.c.f.* if the only difference from strict J.c.f. is that some of the super-diagonals have values other than unity.

It is *possible* for a matrix A to be highly defective without its eigenvalues being unduly sensitive. Suppose, for example, that A is such that there is an orthogonal matrix X for which

$$X^{-1}AX = \tilde{J}, \tag{3.3}$$

where $\tilde{J}$ is of quasi-J.c.f. in which nonzero super-diagonal elements are all 10^{-10}. Perturbations of order 10^{-10} in J (which correspond to perturbations of order 10^{-10} in A since X is orthogonal) produce perturbations of order 10^{-10} at most in the eigenvalues. If $\|A_2\|$ is of the order of unity, then *from the point of view of 10-digit decimal computation*, the eigenvalues of A are not at all sensitive. One cannot even *rely* on defectiveness being characterized by sensitivity of the corresponding eigenvalues. Nevertheless it is true that $\partial\lambda_i/\partial\varepsilon = O(\varepsilon^{1/r-1})$ for some perturbations when J has a block of order r, and hence, $\partial\lambda_i/\partial\varepsilon \to \infty$ as $\varepsilon \to 0$. This means that if we are prepared to extend the precision of computation indefinitely, we shall ultimately gain only one figure of accuracy for r extra figures of precision.

At this stage, one might ask what is the "natural" quasi-J.c.f. for computational purposes. A reasonable definition is that it is the $\tilde{J}$ for which the corresponding $\|X\|_2\|X^{-1}\|_2 = \varkappa(X)$ is a minimum. If this $\tilde{J}$ has super-diagonal elements which are all small relative to $\|\tilde{J}\|_2$, the matrix A will not have sensitive eigenvalues.

As a final result relating small eigenvalues and small singular values, we note the following theorem (for the definition of singular values, see § 7).

THEOREM. *Let A be an $n \times n$ matrix with $\lambda_n = \varepsilon$ and $|\lambda_n| \leqq |\lambda_j|$ and such that there are p Jordan blocks of dimensions $k_1, k_2, \cdots, k_p$, with $k_1 \leqq k_2 \leqq \cdots \leqq k_p$, associated with λ_n. Then if $A = XJX^{-1}$,*

$$\sigma_{n-j+1}(A) \leqq \|X\|_2\|X^{-1}\|_2|\varepsilon|^{k_{p-j+1}} + O(|\varepsilon|^{k_{p-j+1}+2}), \qquad j = 1, 2, \cdots, p. \tag{3.4}$$

Proof.

$$\begin{aligned}\sigma_{n-j+1}(A) = \sigma_{n-j+1}(XJX^{-1}) &\leqq \sigma_1(X)\sigma_{n-j+1}(JX^{-1})\\ &\leqq \sigma_1(X)\sigma_1(X^{-1})\sigma_{n-j+1}(J).\end{aligned} \tag{3.5}$$

Since the singular values of J are given by $[\lambda_i(JJ^T)]^{1/2}$, it is obvious that they are singular values of the elementary Jordan blocks. Consider the $k \times k$ block

$$K = \begin{bmatrix} \varepsilon & 1 & & & \\ & \varepsilon & 1 & & \\ & & \cdot & & \\ & & & \cdot & 1 \\ & & & & \varepsilon \end{bmatrix}. \tag{3.6}$$

From the form of KK^T, $k - 1$ of the singular values are close to unity and since their product is ε^k, the remaining singular value is $O(\varepsilon^k)$. In fact,

$$\sigma_k(K) = \min_{x \neq 0} \frac{\|Kx\|_2}{\|x\|_2} \tag{3.7}$$

and taking $\tilde{x}^T = (1, -\varepsilon, \varepsilon^2, \cdots, (-1)^{k-1}\varepsilon^{k-1})$, we have

$$\sigma_k(K) = |\varepsilon|^k + O(|\varepsilon|^{k+2}). \tag{3.8}$$

The result is thus established. Note that although we have shown that the singular values are small, we have not shown and cannot show that the elements of the corresponding singular vectors are correspondingly small.

4. Ill-conditioned eigenvalues. Since in practice it will usually be impossible to determine whether a matrix has exactly equal eigenvalues, it is necessary to consider the problem of the sensitivity of a *simple* eigenvalue with respect to perturbations in A. If J is the J.c.f., we have

$$AX = XJ, \quad ZA = JZ, \quad Z = X^{-1}. \tag{4.1}$$

When λ_1 is a simple eigenvalue, x_1 is the corresponding right-hand eigenvector and

$$Ax_1 = \lambda_1 x_1. \tag{4.2}$$

If z_1^T is the first row of Z, then

$$z_1^T A = z_1^T \lambda_1. \tag{4.3}$$

It is customary to define the left-hand eigenvector y_1 of A corresponding to λ_1 as the vector satisfying

$$y_1^H A = y_1^H \lambda_1, \tag{4.4}$$

and hence if we write $Y = Z^H$, the first column of Y gives this eigenvector and

$$Y^H X = I. \tag{4.5}$$

Consider now the corresponding eigenvalue $\lambda_1(\varepsilon)$ and right-hand eigenvector $x_1(\varepsilon)$ of $A + \varepsilon B$, where $\|B\|_2 = 1$. For sufficiently small ε, it is easy to show that $\lambda_1(\varepsilon)$ and $x_1(\varepsilon)$ may be expanded as convergent power series

$$\lambda_1(\varepsilon) = \lambda_1 + p_1\varepsilon + p_2\varepsilon^2 + \cdots, \qquad x_1(\varepsilon) = x_1 + v_1\varepsilon + v_2\varepsilon^2 + \cdots, \tag{4.6}$$

where the v_i lie in the space spanned by $x_2, \cdots, x_n$. (Note that in general these x_i will include principal vectors which are not eigenvectors.) Equating coefficients of ε in the relation

$$(A + \varepsilon B)(x_1 + v_1\varepsilon + \cdots) = (\lambda_1 + p_1\varepsilon + \cdots)(x_1 + v_1\varepsilon + \cdots) \tag{4.7}$$

gives

$$Bx_1 + Av_1 = \lambda_1 v_1 + p_1 x_1. \tag{4.8}$$

Now both v_1 and Av_1 lie in the space spanned by $x_2, \cdots, x_n$, and from (4.5),

$y_1^H x_i = 0$ $(i = 2, \cdots, n)$. Hence premultiplying (4.8) by y_1^H, we obtain

$$p_1 = y_1^H B x_1 / y_1^H x_1. \tag{4.9}$$

As derived above, $y_1^H x_1 = 1$, but clearly in (4.9), x_1 and y_1 may be arbitrarily scaled and it is convenient computationally to have $\|x_1\|_2 = \|y_1\|_2 = 1$. In this case, $y_1^H x_1 = s_1$ (in the notation of [25]), where s_1 is the cosine of the angle between x_1 and y_1. From (4.9),

$$\left|\frac{\partial \lambda_1}{\partial \varepsilon}\right|_{\varepsilon=0} = |p_1| \leq \frac{\|y_1\|_2 \|B\|_2 \|x_1\|_2}{s_1} = \frac{1}{|s_1|}. \tag{4.10}$$

The derivative is finite for any "direction" of B. This is in contrast to the case where λ_i is associated with a defective matrix when $|\partial\lambda_i/\partial\varepsilon|_{\varepsilon=0} = \infty$. This latter result is in agreement with (4.10) since the left-hand and right-hand eigenvectors are orthogonal corresponding to a "defective" λ_i. The bound in (4.10) is attained when $B = y_1 x_1^H$, since then

$$y_1^H B x_1 = y_1^H y_1 x_1^H x_1 = 1. \tag{4.11}$$

Further, taking $B = e^{i\theta} y_1 x_1^H$, we can make $(\partial\lambda_1/\partial\varepsilon)_{\varepsilon=0}$ have any required phase. There is one very unsatisfactory feature of the above analysis. The quantity s_i is not invariant with respect to diagonal similarity transformation. Consider the matrix

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \tag{4.12}$$

with

$$\lambda_1 = 3, \quad \lambda_2 = 1, \quad y_1^H = \frac{[1, 1]}{2^{1/2}}, \quad x_1^H = \frac{[1, 1]}{2^{1/2}}, \quad s_1 = 1. \tag{4.13}$$

The eigenvalue λ_1 is therefore very well-conditioned, as indeed are all eigenvalues of all normal matrices. However, we have

$$D^{-1}AD = \begin{bmatrix} 2 & \alpha \\ \alpha^{-1} & 2 \end{bmatrix}, \quad \text{where } D = \begin{bmatrix} 1 & \\ & \alpha \end{bmatrix}, \tag{4.14}$$

and now

$$y_1^H = \frac{[1, \alpha]}{(1 + \alpha^2)^{1/2}}, \quad x_1^H = \frac{[\alpha, 1]}{(1 + \alpha^2)^{1/2}}, \quad s_1 = \frac{2\alpha}{(1 + \alpha^2)}. \tag{4.15}$$

Hence we may make s_1 arbitrarily small by taking α sufficiently large or sufficiently small. It is clear that a small s_i induced in this way is a very artificial phenomenon. In this example, when s_1 is small, $\|D^{-1}AD\|_2 \gg \|A\|_2$. In practice, the relevant values of s_i are those for $D^{-1}AD$, where D has been chosen so that $\|D^{-1}AD\|_2$ is a minimum. Reducing this norm to a true minimum is not vital, and in practice, the process of *balancing* described by Parlett and Reinsch in [12] is usually adequate.

High sensitivity of an eigenvalue λ_i has now been encountered in two different contexts, first when λ_i is associated with defectiveness and secondly when a value of s_i is small. We now show that when an s_i is small, A is necessarily relatively close to a matrix with a multiple eigenvalue. Let

$$(4.16)\qquad Ax_1 = \lambda_1 x_1, \quad y_1^H A = y_1^H \lambda_1, \quad s_1 = y_1^H x, \quad \text{with } \|x_1\|_2 = \|y_1\|_2 = 1,$$

and suppose P is a unitary matrix such that $Px_1 = e_1$, where $e_1^T = (1, 0, \cdots, 0)$. Then

$$(4.17)\qquad PAP^H Px_1 = \lambda_1 Px_1, \qquad (PAP^H)e_1 = \lambda_1 e_1,$$

and $B = PAP^H$ must be of the form

$$(4.18)\qquad B = \left[\begin{array}{c|c} \lambda_1 & b_1^H \\ \hline 0 & B_1 \end{array}\right].$$

Further,

$$(4.19)\qquad s_1 = y_1^H x_1 = (y_1^H P^H)(Px_1) = (Py_1)^H e_1,$$

and writing $Py_1 = p_1$, we have

$$(4.20)\qquad p_1^H B = p_1^H PAP^H = y_1^H AP^H = \lambda_1(y_1^H P^H) = \lambda_1 p_1^H,$$

while

$$(4.21)\qquad s_1 = p_1^H e_1 = \bar{p}_{11}.$$

Hence if we write $p_1^H = (\bar{p}_{11} | v^H)$, where v is of order $n - 1$,

$$(4.22)\qquad \bar{p}_{11} b_1^H + v^H B_1 = \lambda_1 v^H, \qquad v^H\left(B_1 - \lambda_1 I + \bar{p}_{11}\frac{vb_1^H}{v^H v}\right) = 0,$$

i.e., the matrix $B_1 + \bar{p}_{11}(vb_1^H/v^H v)$ has λ_1 as an eigenvalue and v as a left-hand eigenvector. Now

$$(4.23)\qquad \left\|\bar{p}_{11}\left(\frac{vb_1^H}{v^H v}\right)\right\| \leqq |\bar{p}_{11}|\frac{\|v\|\,\|b_1\|}{\|v\|^2} = \frac{|s_1|\,\|b_1\|}{(1 - s_1^2)^{1/2}} \leqq \frac{s_1\|B\|_2}{(1 - s_1^2)^{1/2}} = \frac{s_1\|A\|_2}{(1 - s_1^2)^{1/2}},$$

and when s_1 is small, a small relative perturbation in B converts λ_1 into an eigenvalue of multiplicity at least two. Since the l_2-norm is invariant with respect to unitary transformations, the same remark is true of A. By a similar argument, Kahan in an unpublished paper has shown that the denominator $(1 - s^2)^{1/2}$ may be replaced by 1 in the final bound. However, the above argument shows that the relevant bound is $|s_1|\,\|b_1\|_2/(1 - s_1^2)^{1/2}$ and in replacing $\|b_1\|_2$ by $\|B\|_2$ and hence by $\|A\|_2$, the result is weakened. When A is normal, B is also normal and $b_1 = 0$. Hence if $|s_1| < 1$ for a normal matrix, λ_1 must already be a multiple eigenvalue. This is otherwise obvious, since if λ_1 is a simple eigenvalue of a normal matrix, $y_1 = x_1$ and $s_1 = 1$. The bound we have given is, in general, a considerable improvement on the bound given by Ruhe [16].

5. Almost linearly dependent eigenvectors. The perturbation analysis described above can be used to give the first order perturbation of x_1 resolved in the directions

$x_2, \cdots, x_n$. In the case when A is nondefective, this leads to

$$x_1(\varepsilon) = x_1 + \varepsilon\left\{\sum_{i=2}^{n}\left(\frac{y_i^H B x_1}{s_i(\lambda_i - \lambda_1)}\right)x_i\right\} + O(\varepsilon^2), \tag{5.1}$$

and the coefficient of x_i is bounded by $1/|s_i(\lambda_i - \lambda_1)|$. Hence we obtain a large perturbation *in the direction of* x_i if s_i or $\lambda_i - \lambda_1$ is small. However, this analysis is rather unsatisfactory. When A has an ill-conditioned eigenvalue problem, the set of x_i will be almost linearly dependent, as we show below. The fact that some of the x_i have large coefficients need not necessarily mean that the perturbation as a whole is large.

The left-hand eigenvector y_1 is orthogonal to $x_2, \cdots, x_n$, and hence x_1 may be expanded in terms of $y_1, x_2, \cdots, x_n$. In fact,

$$x_1 = s_1 y_1 + \sum_{i=2}^{n} \alpha_i x_i \tag{5.2}$$

since $y_1^H x_1 = s_1$ and $y_1^H x_i = 0$ $(i = 2, \cdots, n)$. Equation (5.2) may be expressed in the form

$$\sum_{i=1}^{n} \beta_i x_i = s_1 y_1/(1 + \sum \alpha_i^2)^{1/2}, \tag{5.3}$$

where

$$\beta_1 = 1/(1 + \sum \alpha_i^2)^{1/2}, \quad \beta_i = -\alpha_i/(1 + \sum \alpha_i^2)^{1/2}, \quad \|\beta\|_2 = 1. \tag{5.4}$$

Hence we have a unit vector β so that

$$\|X\beta\|_2 = |s_1|/(1 + \sum \alpha_i^2)^{1/2} < |s_1|, \tag{5.5}$$

and when s_1 is small, the vectors x_i are "almost linearly dependent". (Note that in general, the x_i $(i = 2, \cdots, n)$ will include principal vectors which are not eigenvectors.) Anticipating § 7, we note that (5.5) implies that $\sigma_n(X) < |s_1|$. Conversely, if a set of the x_i are almost linearly dependent, then at least one of the associated s_i is small and A has an ill-conditioned eigenvalue. Suppose, for example,

$$\sum_{i=1}^{p} \alpha_i x_i = u, \quad \text{where } \|u\|_2 = \varepsilon, \quad \sum_{i=1}^{p} \alpha_i^2 = 1. \tag{5.6}$$

Then if the vectors y_i are the normalized columns of $(X^{-1})^H$, we have

$$\alpha_i y_i^H x_i = y_i^H u, \qquad s_i = y_i^H u/\alpha_i, \quad |s_i| \leqq \varepsilon/|\alpha_i|. \tag{5.7}$$

Since at least one α_i is such that $|\alpha_i| > p^{-1/2}$, this means that at least one s_i is small. In fact, it is obvious that at least two of the s_i must be small, since otherwise just one of the eigenvalues would be sensitive and the remainder insensitive; as the trace is obviously not sensitive, this is impossible.

This result emphasizes one very unsatisfactory feature of ill-conditioned eigensystems. Suppose we have managed (in spite of the practical difficulties) to obtain *correctly rounded* versions of a set of ill-conditioned eigenvectors $x_1, \cdots, x_p$. We may now wish to determine an accurate orthogonal basis for this subspace of dimension p. However, since the vectors $x_1, \cdots, x_p$ are almost linearly dependent, when we perform the Schmidt orthogonalization process on these x_i, the orthogonal basis is bound to be poorly-determined. In fact, information about the last of the orthogonal vectors will be completely vitiated by the rounding errors which will usually be inherent in the representation of the x_i in the computer.

This casts doubt on the advisability of attempting to determine the x_i themselves and suggests that it might be better to determine directly an orthogonal basis for the subspace corresponding to such vectors.

6. Orthogonal bases for invariant subspaces. The eigenvectors of A corresponding to λ are the solutions of the equation $(A - \lambda I)x = 0$. If $A - \lambda I$ is of nullity n_1 (rank $= n - n_1$), then there will be n_1 independent eigenvectors. These vectors span a subspace P_1, the nullspace of $A - \lambda I$. Let $x_1^{(1)}, x_2^{(1)}, \cdots, x_{n_1}^{(1)}$ be an orthogonal basis of this subspace P_1.

Turning now to the solutions of the equation $(A - \lambda I)^2x$, we can clearly see that they include any vector in P_1, since if $(A - \lambda I)x = 0$, then certainly $(A - \lambda I)^2x = 0$. The nullity of $(A - \lambda I)^2$ may therefore be denoted by $n_1 + n_2$, where $n_2 \geqq 0$. If the nullspace is denoted by P_2, then $P_2 \supset P_1$ and the basis $x_1^{(1)}, x_2^{(1)}, \cdots, x_{n_1}^{(1)}$ may be extended to an orthogonal basis of P_2 by the addition of further orthogonal vectors $x_1^{(2)}, x_2^{(2)}, \cdots, x_{n_2}^{(2)}$. These additional vectors satisfy the relations

$$(6.1) \qquad u_i = (A - \lambda I)x_i^{(2)} \neq 0, \quad (A - \lambda I)^2x_i^{(2)} = 0, \quad i = 1, \cdots, n_2,$$

and hence they are vectors of grade 2.

We now show that $n_2 \leqq n_1$, for the vectors u_i are nonnull and satisfy the relation $(A - \lambda I)u_i = 0$. Hence they lie in P_1, and if $n_2 > n_1$,

$$(6.2) \qquad \sum \alpha_i u_i = 0, \quad \text{i.e., } (A - \lambda I)\sum \alpha_i x_i^{(2)} = 0,$$

which means that $\sum \alpha_i x_i^{(2)} \subset P_1$. But $\sum \alpha_i x_i^{(2)}$ is orthogonal to the $x_i^{(1)}$ by the construction, and hence we have a contradiction.

Continuing in this way by considering the nullities of $(A - \lambda I)^3, (A - \lambda I)^4, \cdots$, we obtain numbers $n_3, n_4, \cdots$ such that $n_{i+1} \leqq n_i$ and orthogonal bases of subspaces P_i such that $P_{i+1} \supset P_i$. The subspace P_i is of dimension $m_i = n_1 + \cdots + n_i$. In general, the orthogonal vectors $x_i^{(s)}$ are such that $(A - \lambda I)^{s-1}x_i^{(s)} \neq 0$ but $(A - \lambda I)^s x_i^{(s)} = 0$.

The sequence comes to an end with $(A - \lambda I)^k$, where $(A - \lambda I)^{k+1}$ is of the same nullity as $(A - \lambda I)^k$.

Comparing these spaces with those spanned by the chains of vectors associated with λ in the J.c.f., we see that P_1 is the space spanned by the principal vectors of grade 1, P_2 that spanned by principal vectors of grades 1 and 2, etc. Notice, though, that the space spanned by $x_1^{(2)}, \cdots, x_{n_2}^{(2)}$ is *not* in general the same as that spanned by the principal vectors of grade 2 in the Jordan chains.

n_1 is equal to the number of blocks associated with λ in the J.c.f., and, in general, n_s is the number of those blocks which are of dimension not less than s.

The derivation of these orthogonal bases is in some ways more satisfactory than that of the Jordan chains themselves, and though the chains may be derived from the orthogonal bases, there will in general be a loss of digital information in this process.

7. The singular values. In the previous section it was shown that in the solution of the complete eigenvalue problem, we are concerned with the determination of the nullities or ranks of sequences of matrices. Rank determination is a notoriously dangerouş numerical problem, and in practice the only reliable way of doing it is via the singular value decomposition (S.V.D). Accordingly we now give a brief review of the S.V.D. and the properties of singular values.

For our purposes, the singular values of a complex $m \times n$ matrix A may be defined to be the nonnegative square roots of the eigenvalues of the matrix A^HA. Clearly A^HA is an $n \times n$ nonnegative definite Hermitian matrix, and its eigenvalues may be denoted by σ_i^2 $(i = 1, \cdots, n)$; the σ_i are the singular values of A. Although apparently a more sophisticated *concept* than the eigenvalues, the *determination* of the singular values is more satisfactory from the computational point of view. The σ_i are defined in terms of the eigenvalues of a Hermitian matrix, and these are always insensitive to small perturbations in elements of that matrix. We shall assume that the σ_i are ordered so that $\sigma_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_n$. The σ_i^2 may be defined via the min-max properties of $(x^HA^HAx)/x^Hx$, i.e., of

$$\frac{\|Ax\|_2^2}{\|x\|_2^2}.$$

$$\sigma_1(X) = \max \frac{\|Ax\|_2}{\|x\|_2} = \|A\|_2, \qquad \sigma_n(X) = \min \frac{\|Ax\|_2}{\|x\|_2} \tag{7.1}$$

and

$$\begin{aligned}\sigma_r(A) - \|B\|_2 &= \sigma_r(A) - \sigma_1(B) \\ &\leqq \sigma_r(A + B) \leqq \sigma_r(A) + \sigma_1(B) \leqq \sigma_r(A) + \|B\|_2.\end{aligned} \tag{7.2}$$

From the last of these relations, the well-conditioned nature of the σ_r is well exposed.

Although we have defined the σ_i via A^HA, they should not be determined in this way. In practice, they are computed via the S.V.D., which is defined as follows.

Any $m \times n$ complex matrix A may be factorized in the form

$$A = U\Sigma V^H, \tag{7.3}$$

where U and V are $m \times m$ and $n \times n$ unitary matrices, respectively, and Σ is an $m \times n$ matrix with $\Sigma_{ii} = \sigma_i$ and $\Sigma_{ij} = 0$ otherwise. Golub and Reinsch [4] have described an extremely efficient and stable method for determining the S.V.D. and hence the σ_i. The computed U and V^H are almost orthogonal to the working accuracy, and the computed σ_i correspond to those of some $(A + E)$, where $\|E\|_2/\|A\|_2$ is a modest multiple of the machine precision. Since the σ_r are insensitive to E, this is very satisfactory.

Clearly, from (7.3),

$$A^HA = V\Sigma^H\Sigma V^H, \qquad A^HAV = V\Sigma^H\Sigma, \tag{7.4}$$

so that the columns of V are orthogonal eigenvectors of A^HA. Similarly

$$AA^H = U\Sigma\Sigma^HU^H, \qquad AA^HU = U\Sigma\Sigma^H, \tag{7.5}$$

and the columns of U are orthogonal eigenvectors of AA^H.

Turning now to the case when A is $n \times n$, we have, from the definitions of the eigenvalues and of the singular values,

$$\prod \lambda_i = \det(A), \qquad \prod (\sigma_i^2) = \det(A^H A) = |\det(A)|^2, \tag{7.6}$$

and hence

$$\prod |\lambda_i| = \prod \sigma_i. \tag{7.7}$$

We have the fundamental result that $\lambda_n = 0$ iff $\sigma_n = 0$ and both imply that A is singular. The three properties are fully equivalent.

From this it is intuitively obvious that if A is "nearly" singular, λ_n and σ_n are "small" with appropriate determination of the terms "nearly" singular and "small". As a measure of the proximity of A to singularity we shall take $\|E\|_2/\|A\|_2 = \varepsilon$, where E is the matrix of minimum norm such that $A + E$ is singular. Since $A + E$ is singular, there exists a y such that

$$(A + E)y = 0. \tag{7.8}$$

Hence

$$\sigma_n = \min \frac{\|Ax\|}{\|x\|} \leqq \frac{\|Ay\|}{\|y\|} = \frac{\|-Ey\|}{\|y\|} \leqq \|E\|_2 = \varepsilon\|A\|_2. \tag{7.9}$$

On the other hand, since $\min(\|Ax\|/\|x\|)$ is attained for some unit vector, y (say),

$$\sigma_n = \|Ay\|, \qquad Ay = \sigma_n z \quad \text{with } \|z\|_2 = 1. \tag{7.10}$$

Hence $(A - \sigma_n z y^H)y = 0$, and $A - \sigma_n z y^H$ must be singular. But $\|\sigma_n z y^H\| = \sigma_n$ and $\varepsilon = \min(\|E\|_2/\|A\|_2) \leqq \sigma_n/\|A\|_2$; hence $\sigma_n/\|A\|_2 = \varepsilon$.

Turning now to λ_n, we have

$$Ay = \lambda_n y \quad \text{for some } \|y\|_2 = 1$$

and

$$\sigma_n = \min \frac{\|Ax\|}{\|x\|} \leqq \frac{\|Ay\|}{\|y\|} = |\lambda_n|. \tag{7.11}$$

On the other hand, from (7.7),

$$|\lambda_n|^n \leqq \sigma_n \sigma_1^{n-1}, \tag{7.12}$$

$$|\lambda_n/\sigma_1|^n \leqq \sigma_n/\sigma_1 = \sigma_n/\|A\|_2 = \varepsilon, \tag{7.13}$$

giving

$$|\lambda_n| \leqq \sigma_1 \varepsilon^{1/n}. \tag{7.14}$$

This last relation is disappointing, but unfortunately it is a best possible result, as is illustrated by the matrices K_n typified by

$$K_4 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \varepsilon & 0 & 0 & 0 \end{bmatrix}. \tag{7.15}$$

In general, $|\lambda_i| = \varepsilon^{1/n}$ $(i = 1, \cdots, n)$, but $\sigma_1 = \cdots = \sigma_{n-1} = 1$ and $\sigma_n = \varepsilon$. All extreme examples are of this kind, since we have equality in (7.12) only if $|\lambda_i| = |\lambda_n|$ (all n) and $\sigma_1 = \sigma_2 = \cdots = \sigma_{n-1}$. In practice, then, we may well have a matrix which is singular to working accuracy and therefore has a negligible singular value but which has no eigenvalues which can be regarded as in any sense small.

The practical consequences of this theorem are very serious. The most stable algorithms for computing eigenvalues can guarantee only that each computed eigenvalue λ_i' is exact for some $A + E_i$, where $\|E_i\|_2/\|A\|_2$ is a modest multiple of the machine precision, and it is difficult to conceive how such algorithms can be improved upon, except, of course, by working to higher accuracy at least in some significant part of the computation. This means that $(A + E_i - \lambda_i' I)$ is exactly singular and hence that $A - \lambda_i' I$ is within $\|E_i\|_2$ of a singular matrix. Hence $A - \lambda_i' I$ has a *singular value* bounded by $\|E_i\|_2$, but the bound for the smallest eigenvalue of $A - \lambda_i' I$ involves $\|E_i\|_2^{1/n}$. All that we can guarantee a priori is that each computed λ_i' will have an error which involves the factor $\|E_i\|_2^{1/n}$, and this may be far from small.

For a normal matrix, $|\lambda_r| = \sigma_r$, and hence this weakness disappears. If λ_i is an eigenvalue of A, then $A + E_i$ has an eigenvalue λ_i' such that

$$|\lambda_i - \lambda_i'| \leqq \|E_i\|. \tag{7.16}$$

Unfortunately, the realization that this result is true has tended to lead to an overconfidence when dealing with real symmetric and Hermitian matrices, which are the commonest examples of normal matrices.

8. Factorizations of almost-singular matrices. If B is an exactly singular matrix and $B = XY$ is a factorization of B, then either X or Y (or both) is exactly singular. Most of the common factorizations used in practice ensure that one of the factors is certainly not singular, and hence with exactly singular B and exact factorization, the other factor must be singular.

A factorization which is frequently used is $B = QR$, where Q is unitary and R is upper triangular. Clearly Q is nonsingular, and hence if B is singular, R must also be singular and therefore have a zero eigenvalue and a zero singular value. But the eigenvalues of R are its diagonal elements and hence at least one r_{ii} must be zero, indeed r_{nn} unless B is "special".

Now consider the case when B is *almost* singular and let us assume for simplicity that B is factorized exactly. We have $\sigma_i(R) = \sigma_i(B)$ since the σ_i are invariant with respect to unitary transformations. Hence R must still have a negligible singular value. However, we can no longer guarantee that any r_{ii} is pathologically small since the r_{ii} are merely the eigenvalues, the bound for which involves $(\sigma_n(B))^{1/n}$.

This result is important in practice because many algorithms for solving the complete eigenproblem of a matrix first compute the eigenvalues and then attempt to determine the eigenvectors from them. If λ is an eigenvalue given by a stable algorithm, $(A + E - \lambda I)$ will be exactly singular with $\|E\|/\|A\|$ small, and hence $B = A - \lambda I$ will be almost singular. The situation appears particularly favorable when A is normal since the computed λ will then have an error which is small relative to $\|A\|_2$, i.e., to $|\lambda_1|$. Unfortunately, although B is normal, the same is not true of R, and hence we still cannot guarantee that R will have any pathologically small r_{ii}. Now the weak bound for λ_n is attained only when B is extremely pathological, and hence one might expect that failure of R to have a small diagonal element would be rare. Unfortunately, this is far from true. Attempts were made to construct an algorithm based on this factorization in the case where A is a symmetric tridiagonal matrix. For such matrices, a particularly satisfactory algorithm is known for the determination of the λ's. Nevertheless, it was found in practice that when the QR factorization of $A - \lambda I$ was performed for each of the n computed λ in turn, almost invariably some of the R were such that they had no small r_{ii}, and all algorithms based on a search for a negligible r_{ii} failed disastrously.

The LL^T factorization of a positive definite matrix A is known to be extremely stable, and it might be thought that when such an A was near to singularity, this would be bound to reveal itself in the corresponding L. That this is not true is illustrated by the matrices $A = L_nL_n^T$, where L_n is of the form illustrated by

$$(8.1) \qquad L_4 = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ -1 & -1 & 1 & \\ -1 & -1 & -1 & 1 \end{bmatrix}.$$

It is easy to show that $\sigma_n(A_n) = \lambda_n(A_n) = O(4^{-n})$, and hence for quite modest values of n, the matrix A_n is almost singular. Yet there is no obvious indication of this in the factor L_n since all of its diagonal elements are unity.

Finally, we consider the factorization given by Gaussian elimination with complete pivoting. This, too, would appear to be quite favorable, and yet it can fail quite catastrophically. Indeed, if A_n is of the form illustrated by

$$(8.2) \qquad A_4 = \begin{bmatrix} 1 & -1 & -1 & -1 \\ & 1 & -1 & -1 \\ & & 1 & -1 \\ & & & 1 \end{bmatrix},$$

then it can be shown that $\sigma_n(A_n) = O(2^{-n})$, and hence A_n is almost singular for quite modest n. Yet the factorization given by Gaussian elimination with complete pivoting is

$$(8.3) \qquad A_n = I \times A_n,$$

i.e., A_n is itself the upper triangular factor, and its diagonal elements are all unity.

These examples illustrate the fact that the determination of singularity, much less than rank, by means of simple factorizations is not a practical proposition. On the other hand, the S.V.D. is extremely reliable, and since the computed σ_i correspond to $A + E$ where $\|E\|_2/\|A\|_2$ is of the order of the machine precision, it provides an excellent means of determining the numerical rank.

9. Vectors by matrix powering. In the next three sections, we discuss some of the algorithms which have been designed to find bases for the successive nullspaces of powers of $(A - \lambda I)$ corresponding to an eigenvalue λ.

For simplicity of notation, we shall work throughout with $B = A - \lambda I$. We shall not for the moment discuss numerical stability, but knowing that most simple factorizations are numerically unreliable for finding the rank of a matrix, we shall use only the S.V.D. for this purpose. Let the S.V.D. of B be denoted by

$$B_1 \equiv B = U_1\Sigma_1 V_1^H, \tag{9.1}$$

where U_1 and V_1 are $n \times n$ unitary matrices. Since λ is an eigenvalue, B is a singular matrix. If it is of nullity n_1, then B_1 will have n_1 zero singular values, and we may write

$$BV_1 = U_1\Sigma_1 = [\underbrace{A_2}_{n-n_1} \mid \underbrace{0}_{n_1}]. \tag{9.2}$$

For consistency with later stages, we write $W_1 \equiv V_1$, and the last n_1 columns of W_1 clearly give an orthogonal basis for the principal vectors of grade 1, while the matrix A_2 has orthogonal columns.

Proceeding to the nullspace of B^2, we have

$$B^2V_1 \equiv B^2W_1 = [BA_2 \mid \underbrace{0}_{n_1}] = [B_2 \mid \underbrace{0}_{n_1}], \tag{9.3}$$

the zero columns obviously persisting. We now compute the S.V.D. of B_2:

$$B_2 = U_2\Sigma_2 V_2^H, \tag{9.4}$$

where U_2 is an $n \times n$ unitary matrix and V_2 an $(n - n_1) \times (n - n_1)$ unitary matrix. Writing

$$\tilde{V}_2 = \left[\begin{array}{cc} V_2 & \\ & I \end{array}\right]_{\substack{\underbrace{\quad}_{n-n_1}\ \underbrace{\quad}_{n_1}}}, \qquad W_2 = W_1\tilde{V}_2, \tag{9.5}$$

we have

$$B^2W_2 = [\underbrace{U_2\Sigma_2}_{n-n_1} \mid \underbrace{0}_{n_1}]. \tag{9.6}$$

Since the nullity of B^2 is $n_1 + n_2$, B_2 will have n_2 zero singular values, and we have

$$B^2W_2 = [A_3 \mid \underbrace{0}_{n_2} \mid \underbrace{0}_{n_1}]. \tag{9.7}$$

Writing $\sum_{i=1}^{s} n_i = m_s$, the matrix A_3 has $n - m_2$ orthogonal columns. The last m_2 columns of W_2 give an orthogonal basis for vectors of grade 2 and grade 1. The last n_1 of these columns are those of W_1 having been unaltered by this second step.

The general step is then as follows:

$$B^s W_s = [\underbrace{A_{s+1}}_{n-m_s} | \underbrace{0}_{n_s} | \cdots | \underbrace{0}_{n_1}], \tag{9.8}$$

$$B^{s+1} W_s = [B_{s+1} | 0 | \cdots | 0], \quad \text{where } B_{s+1} = BA_{s+1}, \tag{9.9}$$

$$B_{s+1} = U_{s+1}\Sigma_{s+1}V^H_{s+1}, \tag{9.10}$$

where U_{s+1} is an $n \times n$ unitary matrix and V_{s+1} an $(n - m_s) \times (n - m_s)$ unitary matrix. B_{s+1} has n_{s+1} zero singular values and writing

$$\tilde{V}_{s+1} = \begin{bmatrix} V_{s+1} & \\ & I \end{bmatrix}, \qquad W_{s+1} = W_s\tilde{V}_{s+1}, \tag{9.11}$$

$$B_{s+1}W_{s+1} = [U_{s+1}\Sigma_{s+1} | 0 | \cdots | 0] \tag{9.12}$$

$$= [A_{s+2} | \underbrace{0}_{n_{s+1}} | \cdots | \underbrace{0}_{n_1}]. \tag{9.13}$$

The process terminates when A_{s+1} is of full rank.

The main weakness of this algorithm is the difficulty of recognizing which of the elements of σ_i may be treated as zero. This is well illustrated when A and therefore B is normal. If such a matrix were inserted into this algorithm, then at the first step, the singular values would be $|\lambda_1|, |\lambda_2| \cdots |\lambda_n|$, of which n_1 would be treated as zero. For a normal matrix, the process should terminate here since all vectors are of grade 1. However, if one continues, the singular values in the second step would be $|\lambda_1|^2, |\lambda_2|^2, \cdots, |\lambda_n|^2$, and some of these might well be regarded as negligible. The algorithm can be modified to limit this shortening, but even then it compares unfavorably in most respects with the algorithm of the next section.

10. Vectors by orthogonal deflation. Again it is convenient to work with B, and we assume that it has an eigenvalue of multiplicity k. We write $B^{(1)} = B$ and denote the S.V.D. of $B^{(1)}$ by

$$B^{(1)} = U^{(1)}\Sigma^{(1)}(V^{(1)})^H, \tag{10.1}$$

where there will be n_1 zero singular values. Hence

$$B^{(2)} = (V^{(1)})^H B^{(1)} V^{(1)} = (V^{(1)})^H U^{(1)}\Sigma^{(1)} = W^{(1)}\Sigma^{(1)}, \tag{10.2}$$

and we may write

$$B^{(2)} = \left[\begin{array}{c|c} B^{(2)}_{11} & 0 \\ \hline B^{(2)}_{21} & 0 \end{array}\right] \begin{array}{l} \}n - n_1 \\ \}n_1 \end{array}. \tag{10.3}$$

From the orthogonality of $W^{(1)}$, the first $n - n_1$ columns of $B^{(2)}$ are orthogonal and therefore independent. Relation (10.1) shows that the last n_1 columns of $V^{(1)}$ give n_1 orthogonal eigenvectors (i.e., vectors of grade 1) of $B^{(1)}$ corresponding to $\lambda = 0$.

If $n_1 = k$, then we have dealt with all the eigenvalues. Otherwise $B^{(2)}_{11}$ will have $k - n_1$ zero eigenvalues and we can proceed to the consideration of vectors of grade 2. Let z be an arbitrary nonnull vector partitioned conformally with $B^{(2)}$

so that $z^T = [x^T \mid y^T]$. Then

$$B^{(2)}z = \left[\begin{array}{c} B_{11}^{(2)} \\ \hline B_{21}^{(2)} \end{array}\right] x, \tag{10.4}$$

and when $x = 0$ and $y \neq 0$, z is a vector of grade 1. If $x \neq 0$, then it follows from the independence of the first $n - n_1$ columns of $B^{(2)}$ that $B^{(2)}z \neq 0$. However, we have

$$(B^{(2)})^2 z = \left[\begin{array}{c} B_{11}^{(2)} \\ \hline B_{21}^{(2)} \end{array}\right] B_{11}^{(2)} x, \tag{10.5}$$

and from the same linear independence, z is a vector of grade 2 iff $B_{11}^{(2)}x = 0$.

Hence we may proceed as follows. Let the S.V.D. of $B_{11}^{(2)}$ be given by

$$B_{11}^{(2)} = U^{(2)}\Sigma^{(2)}(V^{(2)})^H, \tag{10.6}$$

where $\Sigma^{(2)}$ has n_2 zero diagonal elements if $B_{11}^{(2)}$ is of nullity n_2. Hence

$$(V^{(2)})^H B_{11}^{(2)} V^{(2)} = (V^{(2)})^H U^{(2)}\Sigma^{(2)} = W^{(2)}\Sigma^{(2)}, \tag{10.7}$$

and we may write

$$(V^{(2)})^H B_{11}^{(2)} V^{(2)} = \left[\begin{array}{c|c} B_{13}^{(3)} & 0 \\ \hline B_{21}^{(3)} & 0 \end{array}\right] \begin{array}{l} \}n - m_2 \\ \}n_2 \end{array}. \tag{10.8}$$

Again the first $n - m_2$ columns of $(V^{(2)})^H B_{11}^{(2)} V^{(2)}$ are orthogonal and hence independent. Introducing the unitary matrix

$$\tilde{V}^{(2)} = \left[\begin{array}{c|c} V^{(2)} & 0 \\ \hline 0 & I \end{array}\right], \tag{10.9}$$

$$B^{(3)} = (\tilde{V}^{(2)})^H B^{(2)} \tilde{V}^{(2)} = \left[\begin{array}{ccc} B_{11}^{(3)} & 0 & 0 \\ B_{21}^{(3)} & 0 & 0 \\ \underbrace{B_{31}^{(3)}}_{n-m_2} & \underbrace{B_{32}^{(3)}}_{n_2} & \underbrace{0}_{n_1} \end{array}\right]. \tag{10.10}$$

It is obvious that $n_2 < n_1$; otherwise $B^{(3)}$ and hence $B^{(1)}$ would have been of nullity greater than n_1.

Again if $m_2 = k$, the process is complete. Otherwise $B_{11}^{(3)}$ has some zero eigenvalues, and we proceed via its S.V.D., this next stage being typical. If

$$B^{(3)} = U^{(3)}\Sigma^{(3)}(V^{(3)})^H, \tag{10.11}$$

then

$$(V^{(3)})^H B_{11}^{(3)} V^{(3)} = (V^{(3)})^H U^{(3)}\Sigma^{(3)} = W^{(3)}\Sigma^{(3)} \tag{10.12}$$

and again introducing

$$\tilde{V}^{(3)} = \left[\begin{array}{c|c} V^{(3)} & 0 \\ \hline 0 & I \end{array}\right], \tag{10.13}$$

we are led to

$$(10.14)\qquad B^{(4)} = (\tilde{V}^{(3)})^H B^{(3)} \tilde{V}^{(3)} = \begin{bmatrix} B_{11}^{(4)} & 0 & 0 & 0 \\ B_{21}^{(4)} & 0 & 0 & 0 \\ B_{31}^{(4)} & B_{32}^{(4)} & 0 & 0 \\ \underbrace{B_{41}^{(4)}}_{n-m_3} & \underbrace{B_{42}^{(4)}}_{n_3} & \underbrace{B_{43}^{(4)}}_{n_2} & \underbrace{0}_{n_1} \end{bmatrix},$$

where n_3 is the nullity of $B_{11}^{(3)}$. By an argument similar to that used above, the nonnull columns of $B^{(4)}$ and of leading principal submatrices of orders $n - m_1$, $n - m_2$ are linearly independent. The process clearly terminates when $m_s = k$, at which stage $B_{11}^{(s+1)}$ is no longer singular. Since

$$(10.15)\qquad B^{(s+1)} = V^H B^{(1)} V \equiv V^H B V,$$

where $V = V^{(1)}\tilde{V}^{(2)}\tilde{V}^{(s)} \cdots \tilde{V}^{(s)}$, the principal vectors of $B^{(1)}$ may be found via those of $B^{(s+1)}$. For simplicity of notation, we expose the case when $s = 3$ which is wholly typical. We may write

$$(10.16)\qquad B^{(4)} = \left[\begin{array}{c|c} B_{11}^{(4)} & 0 \\ \hline P & C \end{array}\right]\begin{matrix} \}n - m_3 \\ \}m_3 \end{matrix}$$

and it is evident that

$$(10.17)\qquad [B^{(4)}]^t = \begin{bmatrix} (B_{11}^{(4)})^t & 0 \\ P_t & C^t \end{bmatrix}.$$

Hence

$$(10.18)\qquad [B^{(4)}]^t \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} (B_{11}^{(4)})^t x \\ P_t x + C^t y \end{bmatrix},$$

and since $B_{11}^{(4)}$ is nonsingular, $(B_{11}^{(4)})^t x$ is not null unless $x = 0$. All vectors in the relevant invariant subspace have their first $n - m_3$ components equal to zero, and since

$$(10.19)\qquad [B^{(4)}]^t \begin{bmatrix} 0 \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ C^t y \end{bmatrix},$$

it is evident that we may concentrate on the matrix C given explicitly by

$$(10.20)\qquad C = \begin{bmatrix} 0 & 0 & 0 \\ B_{32}^{(4)} & 0 & 0 \\ B_{42}^{(4)} & B_{43}^{(4)} & 0 \end{bmatrix}.$$

A discussion of vectors of grade 3 will be fully illustrative. Let us take any vector x

of order m_3 and partition conformally into $x^T = [x_1^T \mid x_2^T \mid x_3^T]$. If $x_1 \neq 0$, we have

$$(10.21) \qquad Cx = \begin{bmatrix} 0 \\ B_{32}^{(4)} \\ B_{42}^{(4)} \end{bmatrix} x_1 + \begin{bmatrix} 0 \\ 0 \\ B_{43}^{(4)} \end{bmatrix} x_2,$$

$$(10.22) \qquad C^2x = \begin{bmatrix} 0 \\ 0 \\ B_{43}^{(4)} \end{bmatrix} B_{32}^{(4)} x_1 = \begin{bmatrix} 0 \\ 0 \\ B_{43}^{(4)} \end{bmatrix} z \quad \text{(say)}.$$

But since we know the columns of $B_{32}^{(4)}$ are independent, $z \neq 0$, and since also the columns of $B_{43}^{(4)}$ are independent, $C^2x \neq 0$. On the other hand, $C^3x = 0$ for any x. The last n_1 columns of the identity matrix therefore give n_1 orthogonal vectors of grade 1, the next n_2 columns of it give vectors of grade 2 and the next n_3 columns give vectors of grade 3.

Interpreting this result in terms of B for the general case, we see that the last n_1 columns of V give orthogonal vectors of grade 1, the next n_2 give orthogonal vectors of grade 2, etc.

When the process terminates, $B_{11}^{(s+1)}$ is nonsingular and its eigenvalues are the remaining eigenvalues of B, i.e. $B_{11}^{(s+1)} + \lambda I$ gives the remaining eigenvalues of A. We can now turn to the next eigenvalue of A and repeat this process starting from $B^{(s+1)} + \lambda I$. In this way, a canonical form is ultimately attained, which may be illustrated in the case when A has only three distinct eigenvalues $\lambda_1, \lambda_2, \lambda_3$ by

$$(10.23) \qquad V^HAV = \left[\begin{array}{cc|ccc|cc} \lambda_1 I & & & & & & \\ Y_{21} & \lambda_1 I & & & & & \\ \hline X_{31} & X_{32} & \lambda_2 I & & & & \\ X_{41} & X_{42} & Y_{43} & \lambda_2 I & & & \\ X_{51} & X_{52} & Y_{53} & Y_{54} & \lambda_2 I & & \\ \hline X_{61} & X_{62} & X_{63} & X_{64} & X_{65} & \lambda_3 I & \\ X_{71} & X_{72} & X_{73} & X_{74} & X_{75} & Y_{76} & \lambda_3 I \end{array}\right] \begin{array}{l} \}n_2^{(1)} \\ \}n_1^{(1)} \\ \}n_3^{(2)} \\ \}n_2^{(2)}. \\ \}n_1^{(2)} \\ \}n_2^{(3)} \\ \}n_1^{(3)} \end{array}$$

In the example given here, there were two stages with λ_3, three stages with λ_2 and two stages with λ_1 and the integers $n_j^{(i)}$ are the nullities exposed in the successive stages of the process. The matrix V being the product of unitary matrices is itself unitary. Note that we have denoted the submatrices in the diagonal blocks by Y_{ij} and outside these blocks by X_{ij}. From the definition of the algorithm, we have $n_j^{(i)} \geqq n_{j+1}^{(i)}$, and the columns of $Y_{i+1,i}$ are linearly independent. We already know that $n_1^{(3)}, n_2^{(3)}$ give the number of vectors of grades 1 and 2, respectively, associated with λ_3, and the corresponding columns of V provide the vectors themselves. The remaining columns of V cannot, of course, give vectors corresponding to

λ_2 and λ_1 since, in general, the latter will not be orthogonal to those of λ_3. We have not yet established that $n_1^{(2)}$, $n_2^{(2)}$, $n_3^{(2)}$ gives the *number* of vectors of grades 1, 2, 3 associated with λ_2, and that $n_1^{(1)}$ and $n_2^{(1)}$ the vectors of grades 1, 2 associated with λ_1, and this we now do.

We cannot proceed further with the reduction without departing from *unitary* similarities. However, if we now admit general similarities, the submatrices denoted by the X_{ij} may be annihilated. To annihilate X_{42}, for example, we premultiply by Z_{42}^{-1} and postmultiply by Z_{42}, where Z_{42} is equal to the identity matrix with a block $X_{42}/(\lambda_1 - \lambda_2)$ in the same position as is occupied by X_{42}. The X_{ij} are eliminated in this way in the order $X_{32}, X_{31}, X_{42}, X_{41}, X_{52}, X_{51}, X_{65}, X_{64}, \cdots$. It is obvious that the Y_{ij} are unaffected by this. The final result is that we have

$$(10.24) \qquad C = M^{-1}AM = \left[\begin{array}{cc|ccc|cc} \lambda_1 I & & & & & & \\ Y_{21} & \lambda_1 I & & 0 & & 0 & \\ \hline & & \lambda_2 I & & & & \\ & 0 & Y_{43} & \lambda_2 I & & 0 & \\ & & Y_{53} & Y_{54} & \lambda_2 I & & \\ \hline & 0 & & 0 & & \lambda_3 I & \\ & & & & & Y_{76} & \lambda_3 I \end{array}\right],$$

where M is no longer unitary (though its last $n_1^{(3)} + n_2^{(3)}$ columns are still orthogonal). From the properties of the $Y_{i+1,i}$ described above, it is now evident that the $n_j^{(i)}$ have the significance described above and indeed that all the columns of M give independent (but not orthogonal) vectors of the relevant grades corresponding to $\lambda_1, \lambda_2, \lambda_3$. Notice that we have now proved that the canonical form (10.23) which is achieved purely by unitary transformations gives a full specification of the J.c.f. There is no need actually to proceed to the form (10.24) in order to find the J.c.f. However, from the form C and the rank properties of the $Y_{i+1,i}$, we may proceed to a demonstration of the J.c.f. itself. It is easy to show that by further similarities (10.24) may be reduced to

$$(10.25) \qquad \left[\begin{array}{cc|ccc|cc} \lambda_1 I & & & & & & \\ K_{21} & \lambda_1 I & & & & & \\ \hline & & \lambda_2 I & & & & \\ & & K_{43} & \lambda_2 I & & & \\ & & & K_{54} & \lambda_2 I & & \\ \hline & & & & & \lambda_3 I & \\ & & & & & K_{76} & \lambda_3 I \end{array}\right],$$

where $K_{i+1,i}$ is a matrix of the form

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{10.26}$$

having the same dimension as $Y_{i+1,i}$. Apart from the ordering of the rows and columns, this is the J.c.f.

It should be emphasized that we are not recommending proceeding beyond the form (10.23), and, indeed, if one requires an orthogonal basis associated with each of the λ_i, one should return to the *original matrix* with each eigenvalue in turn.

The outstanding weakness of the algorithms of this section and the previous one is that the volume of work may be excessive. To find the vectors for a matrix of order n corresponding to an eigenvalue λ_1 of multiplicity k having just one block $J_r(\lambda_1)$ in the J.c.f., one must perform an S.V.D. on matrices of orders n, $n - 1, \cdots, n - r$ in succession (the last one merely to reveal that there are no more eigenvalues equal to λ_1!).

Both algorithms were suggested by Kublanovskaya [10], but not in terms of the S.V.D., and have also been described by Ruhe [14], though in different terms from those used here.

11. Economical algorithm for determination of vectors. An alternative algorithm suggested by Golub and Wilkinson is considerably more economical in general (though not necessarily superior in other respects). Again corresponding to an eigenvalue λ, one works with $B = A - \lambda I$. We first give the basic motivation. Suppose we have already determined independent vectors u_1, u_2, u_3 of grade 1, vectors v_1, v_2 of grade 2 and vectors w_1, w_2 of grade 3 (not necessarily orthogonal).

If x is any vector of grade 4, then Bx is of grade 3 and hence lies in the subspace spanned by the u_i, v_i, w_i. In fact, x must satisfy a relation

$$Bx = [u_1 \mid u_2 \mid u_3 \mid v_1 \mid v_2 \mid w_1 \mid w_2]\alpha, \tag{11.1}$$

where α is a vector of order 7. However, the totality of independent solutions of (11.1) includes v_1, v_2, w_1, w_2, which will have been obtained by previously solving

$$Bx = [u_1 \mid u_2 \mid u_3 \mid v_1 \mid v_2]\beta \quad \text{and} \quad Bx = [u_1 \mid u_2 \mid u_3]\gamma. \tag{11.2}$$

We need a procedure which will reject these previous solutions. Indeed, the solutions needed at the current stage are solutions of (11.1) which are independent of v_1, v_2, w_1, w_2. To this end, we observe that instead of solving (11.1), we may equally well solve

$$Bx = ([u_1 \mid u_2 \mid u_3 \mid v_1 \mid v_2 \mid w_1 \mid w_2]Z)\alpha, \tag{11.3}$$

where Z is any nonsingular 7×7 matrix, preferably unitary if one does not wish to sacrifice numerical information. Now B is a singular matrix, and a convenient method for solving (11.1) is via the S.V.D. of B:

$$B = U\Sigma V^H, \tag{11.4}$$

where Σ has n_1 zero elements, assumed to be in the last n_1 diagonal positions. Hence we wish to solve

$$\Sigma(V^H x) \equiv \Sigma y = U^H([u_1 \mid u_2 \mid u_3 \mid v_1 \mid v_2 \mid w_1 \mid w_2]Z)\alpha \tag{11.5}$$

$$= ([U^H u_1 \mid U^H u_2 \mid U^H u_3 \mid U^H v_1 \mid U^H v_2 \mid U^H w_1 \mid U^H w_2]Z)\alpha. \tag{11.6}$$

Solutions are obtained via these values of α for which the right-hand side p has zero components in the last n_1 positions. In our example, $n_1 = 3$, and the equations become

$$\begin{bmatrix} \sigma_1 y_1 = p_1 \\ \sigma_2 y_2 = p_2 \\ \sigma_3 y_3 = p_3 \\ \sigma_4 y_4 = p_4 \\ 0y_5 = 0 \\ 0y_6 = 0 \\ 0y_7 = 0 \end{bmatrix}. \tag{11.7}$$

Components y_5, y_6, y_7 are therefore arbitrary and in our algorithm are taken to be zero, since they merely result in including multiples of u_1, u_2, u_3 in the vector α derived from y.

We have still to discuss how we avoid duplicating the vectors we have previously produced. Suppose at the stage when we are determining the vectors v_1 and v_2 we have computed a Z (which might be called Z_1) such that

$$[U^H u_1 \mid U^H u_2 \mid U^H u_3]Z_1 = \begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \\ X & X & X \\ X & 0 & 0 \\ X & 0 & 0 \\ X & 0 & 0 \end{bmatrix} = [p^{(1)} p^{(2)} p^{(3)}]. \tag{11.8}$$

Then v_1 and v_2 are obtained by solving $\Sigma y = p$, where p takes in turn each of the vectors $p^{(2)}$ and $p^{(3)}$, giving independent solutions. Now when we have to solve the set

$$\Sigma y = ([U^H u_1 \mid U^H u_2 \mid U^H u_3 \mid U^H v_1 \mid U^H v_2]Z)\alpha, \tag{11.9}$$

if we take $Z = \begin{bmatrix} Z_1 & \\ & I \end{bmatrix}[W_2]$, then (11.9) becomes

$$(11.10) \qquad \Sigma y = ([p^{(1)} \mid p^{(2)} \mid p^{(3)} \mid U^H v_1 \mid U^H v_2][W_2])\alpha .$$

Columns two and three have already been dealt with and gave us v_1 and v_2. The new solutions are obtained by solving

$$(11.11) \qquad \Sigma y = ([p^{(1)} \mid U^H v_1 \mid U^H v_2]Z_2)\alpha .$$

Notice that in this stage we are left with only three columns (i.e., n_1) just as in the previous stage. Again we determine a Z_2 which will make the trailing columns of the matrix in parentheses on the right of (11.10) of the requisite form, i.e., with zero in positions 5, 6, 7 (the last n_1 positions). The number of vectors of this form will decide how many vectors of grade 3 we obtain. The algorithm is now complete, and we observe that at each stage we are dealing with a system of the form

$$(11.12) \qquad \Sigma y = (RZ)\alpha ,$$

where R is always an $n \times n_1$ matrix, and we wish to determine Z so that RZ is of the requisite form, i.e., its trailing columns have zeros in the last n_1 positions. This is conveniently done via an S.V.D. composition. We write

$$(11.13) \qquad R = \underbrace{\begin{array}{|c|} \hline R_1 \\ \hline R_2 \\ \hline \end{array}}_{n_1} \begin{array}{l} \\ \}n_1 \end{array},$$

where R_2 is an $n_1 \times n_1$ matrix. If $R_2 = U_2 \Sigma_2 V_2^H$, where Σ_2 has at the sth stage n_s zero diagonals, then taking $Z = V_2$,

$$(11.14) \qquad RZ = \begin{bmatrix} R_1 V_2 \\ U_2 \Sigma_2 \end{bmatrix},$$

and the last n_s columns are of the required form with regard to the n_s zero elements in Σ_2.

The general algorithm may now be described by its typical stage at which we determine vectors of grade $s + 1$. We assume that by this time we have

$$(11.15) \qquad \begin{array}{l} n_1 \text{ vectors of grade } 1: u_1^{(1)}, u_2^{(1)}, \cdots, u_{n_1}^{(1)}, \\ n_2 \text{ vectors of grade } 2: u_1^{(2)}, u_2^{(2)}, \cdots, u_{n_2}^{(2)}, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ n_s \text{ vectors of grade } s: u_1^{(s)}, u_2^{(s)}, \cdots, u_{n_s}^{(s)}. \end{array}$$

We then assemble an $n \times n_1$ matrix $R^{(s+1)}$, the first $n_1 - n_s$ columns of which will be denoted by $P^{(s)}$, the origin of which will become obvious during the description of this next stage. The remaining n_s columns are $U^H u_1^{(s)}, \cdots, U^H u_{n_s}^{(s)}$. This matrix $R^{(s+1)}$ is partitioned in the form

$$(11.16) \qquad R^{(s+1)} = \underbrace{\begin{bmatrix} R_1^{(s+1)} \\ R_2^{(s+1)} \end{bmatrix}}_{n_1} \begin{array}{l} \\ \}n_1 \end{array}.$$

If the S.V.D. of $R_2^{(s+1)}$ is $U^{(s+1)}\Sigma^{(s+1)}(V^{(s+1)})^H$, where the number of zero elements in $\Sigma^{(s+1)}$ is denoted by n_{s+1}, then

$$R^{(s+1)}V^{(s+1)} = \left[\begin{array}{c} R_1^{(s+1)}V^{(s+1)} \\ \hline U^{(s+1)}\Sigma^{(s+1)} \end{array}\right] = \left[\begin{array}{c|c} P_{11}^{(s+1)} & P_{12}^{(s+1)} \\ \hline P_{21}^{(s+1)} & 0 \end{array}\right] \tag{11.17}$$

$$\hphantom{R^{(s+1)}V^{(s+1)} = \left[\begin{array}{c} R_1^{(s+1)}V^{(s+1)} \end{array}\right] =} \underbrace{\hphantom{P_{21}^{(s+1)}}}_{n_1 - n_{s+1}} \quad \underbrace{\hphantom{P_{12}^{(s+1)}}}_{n_{s+1}}$$

and the vectors $y_1^{(s+1)}, \cdots, y_{n_{s+1}}^{(s+1)}$ are obtained by solving $\Sigma y = p$, where p takes each of the last n_{s+1} columns of the matrix on the right in turn. The $u_i^{(s+1)}$ are then obtained by multiplying these vectors by V. The $n \times (n_1 - n_{s+1})$ matrix in the first $n - n_{s+1}$ columns of the matrix on the right of (11.17) is the matrix $P^{(s+1)}$ required in the next stage. The process terminates when $n_{s+1} = 0$.

In the first stage we solve

$$\Sigma y = 0 \tag{11.18}$$

and obtain the solutions $y = e_{n-n_1+1}, e_{n-n_1+2}, \cdots, e_n$, the last n_1 columns of the identity matrix and hence $u_1^{(1)}, \cdots, u_{n_1}^{(1)}$ are the last n_1 columns of V. The vectors of grade 1 are therefore orthogonal, but this is not true of any of the subsequent sets of vectors, though by taking $y_{n-n_1+1}, \cdots, y_n$ to be zero in each of the subsequent solutions of $\Sigma y = p$, one ensures that all vectors of grades higher than one are orthogonal to those of grade 1.

Observe that the successive S.V.D.'s are all performed on a matrix of order $n_1 \times n_1$. In the case when $n_1 = 1$ and there is only one block in the J.c.f. associated with the current eigenvalue, this will be a 1×1 matrix at each stage and the process comes to an end when the last element of $U^H u_1^{(s)}$ is nonzero!

12. Comments on algorithms for principal vectors. So far we have concentrated mainly on the formal aspects of the algorithms, though in using the S.V.D. we are tacitly recognizing numerical difficulties. The first problem is how to select our λ when forming $B = A - \lambda I$. In practice, the eigenvalues of A should have been found using some stable algorithm such as the QR algorithm. Although the computed λ_i may be arbitrarily bad, each should be exact for some matrix $A + E_i$, where E_i is such that $\|E_i\|_2/\|A\|_2$ is merely a modest multiple of the computer precision. Hence $B = A - \lambda_i I$ should have at least one negligible singular value relative to $\|A\|_2$, however "poor" λ_i may be in an absolute sense. However, if A really is defective, the computed λ_i are probably not the best values to use. If, for example, A has a well-defined J.c.f. (i.e., in the optimum quasi-J.c.f., the superdiagonal elements are of the order of magnitude of $\|A\|_2$) and there is just one block $J_r(\lambda_1)$ associated with λ_1, one will expect the computed λ_i to include a set of r values $\bar{\lambda}_1, \cdots, \bar{\lambda}_r$ which, though not particularly close to λ_1, will be such that their sum is very close to $r\lambda_1$. If one could *recognize* such a block, one should use the mean of those values $\bar{\lambda}$ and then work with $B = A - \bar{\lambda}I$. However, in practice, the situation will be much more obscure than this, and it is a difficult problem to decide which values of λ to use.

Whichever of the algorithms we use, we shall need at each stage when an S.V.D. is performed a satisfactory criterion for deciding which singular values may be regarded as "zero". The situation is most satisfactory in connection with the deflation technique. At each stage, the matrix on which the S.V.D. is performed has been determined by a unitary similarity on $(A - \lambda I)$, and it is reasonable to use some tolerance $\varepsilon\|A\|_2$ throughout, when ε is "small" but appreciably larger than the machine precision.

In the powering algorithm, the rth matrix is of degree r in the elements of A, and the decision is much less satisfactory. A modification of the procedure has been developed which ameliorates this difficulty, but matrix powering would seem to have nothing to recommend it in comparison with the deflation algorithm.

The Golub–Wilkinson algorithm is far superior from the point of view of economy of computation; while the first S.V.D. is done on $A - \lambda I$, the others are all performed on a submatrix of a set of n_1 vectors. If the vectors $u_j^{(i)}$ are normalized at each stage, a negligible singular value would be one which is small compared with unity. If in the matrix Σ obtained from $B = A - \lambda I$ itself the smallest singular value to be regarded as nonzero is quite close to the tolerance, then in determining all subsequent solutions of equations of the form $\Sigma y = p$, the element y_{n-n_i} is obtained by dividing by this *almost negligible* σ_{n-n_i}. The vectors obtained with this process are not orthogonal, as they are with the other two and there does appear to be a danger that they may be almost linearly dependent with a consequent loss of digital information.

None of the three processes gives principal vectors satisfying the chain reaction typical of the columns of the X producing the J.c.f. Modified vectors satisfying the chain reaction can be determined from the computed vectors, but the volume of work is substantial and care is needed to avoid losing digital information. Some such loss is inevitably involved in going from the *orthogonal* sets given by the powering and deflation algorithms, since the vectors in the chains may be arbitrarily near to linear dependence. Indeed, one might well ask whether one *should* move from the orthogonal sets to sets satisfying the chain relations. The answer must depend on what the vectors are needed for, and here numerical analysts would welcome discussion with applied mathematicians, since this is clearly a subjective matter. Further experimentation is necessary before the algorithm can be fully assessed.

13. Poorly-defined J.c.f. As mentioned previously, there is a natural tendency to construct "difficult" examples for testing purposes by taking a J.c.f. and subjecting it to some simple similarity transformation. Such examples severely underestimate the difficulties associated with ill-conditioned matrices. The point is well illustrated by considering the Frank matrices F_n defined typically by

$$F_5 = \begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 4 & 3 & 2 & 1 \\ & 3 & 3 & 2 & 1 \\ & & 2 & 2 & 1 \\ & & & 1 & 1 \end{bmatrix}. \tag{13.1}$$

Even for quite modest values of n, some of the eigenvalues and eigenvectors are very ill-conditioned, and yet one has a simple method of determining them by observing that, for example,

$$\begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix} (F_5 - \lambda I) \tag{13.2}$$

$$= \begin{bmatrix} 1-\lambda & \lambda & & & \\ 4 & 1-\lambda & \lambda & & \\ & 3 & 1-\lambda & \lambda & \\ & & 2 & 1-\lambda & \lambda \\ & & & 1 & 1-\lambda \end{bmatrix} = G_5.$$

This result is quite general and enables us to determine the eigenvalues of F_5, for example, from those of the quasi-symmetric tridiagonal matrix

$$T_5 = \begin{bmatrix} 0 & 1 & & & \\ 4 & 0 & 1 & & \\ & 3 & 0 & 1 & \\ & & 2 & 0 & 1 \\ & & & 1 & 0 \end{bmatrix}. \tag{13.3}$$

The determination of these latter eigenvalues is a well-conditioned problem for all values of n. We are able to remove the ill-condition in this way because the transformation can be performed exactly, i.e., without rounding error. The eigenvalues of F_n are very sensitive to perturbation in elements in the top right-hand corner, and by transforming to T_n and then working *explicitly with a tridiagonal matrix* one ensures that no rounding errors are effectively made in these elements! From this transformation it is easy to show that eigenvalues of F_n are such that $\lambda_r = 1/\lambda_{n-r+1}$. It is the smaller eigenvalues which are ill-conditioned.

To illustrate the nature of the ill-conditioning, we concentrate for the moment on F_{12} and discuss the problem from the point of view of computation on KDF9 which has a 39-digit binary mantissa, i.e., rather less than 12 decimal digits of accuracy.

By row transformations, we see that $\det(F_n) = 1$, and if $\tilde{F}_n$ is the matrix resulting from a perturbation ε in position $(1, n)$, we have $\det(\tilde{F}_n) = 1 \pm (n-1)!\varepsilon$. Since the determinant is the product of the eigenvalues, it is evident that changes of $\pm 1/(n-1)!$ in this element alter the product of the eigenvalues from the true value, 1, to 0 and 2, respectively. When $n = 100$, for example, this represents a change of approximately 10^{-158}. To obtain the eigenvalues correct to 10 decimals even with an extremely stable *general purpose* algorithm would require computation with a mantissa of about 170 decimal digits. Yet the eigenvalues may be determined via T_{100} working to ten decimals only.

For $n = 12$, the situation is not yet too serious with 12-digit decimal computation, since $11! = 4 \times 10^7$. One can expect to obtain some correct digits even in the most ill-conditioned eigenvalues. The quantities s_i for the four smallest eigenvalues are

$$s_{12} = 5 \times 10^{-8}, \quad s_{11} = 3 \times 10^{-8}, \quad s_{10} = 4 \times 10^{-8}, \quad s_9 = 15 \times 10^{-8}, \tag{13.4}$$

the corresponding eigenvalues being

$$\begin{aligned} \lambda_{12} &= 0.0310 \cdots, & \lambda_{11} &= 0.0495 \cdots, \\ \lambda_{10} &= 0.0812 \cdots, & \lambda_9 &= 0.1436 \cdots, \end{aligned} \tag{13.5}$$

where we have given only the order of magnitude of the s_i. In fact the errors in the eigenvalues as computed on KDF9 using the very stable QR algorithm were 4×10^{-6}, 7×10^{-6}, 5×10^{-6} and 10^{-7}, respectively, and from the sensitivity considerations discussed in § 4, these results are seen to be extremely creditable.

From the discussion in that section, we also know that there is certainly a matrix having a double eigenvalue λ_{11} at a distance within $\|F_{12}\|s_{11}$, but in fact, F_{12} is much nearer to a defective matrix than this. Indeed, it is near to quite a number of different defective matrices. Let us consider first the possibilty of inducing defectiveness by a perturbation ε in the (1, 12)-element only. The modified characteristic equation is

$$\prod (\lambda_i - \lambda) - 11!\varepsilon = 0. \tag{13.6}$$

If we draw the graph $y = \prod (\lambda_i - \lambda)$, then the modified eigenvalues are at the values of λ for which $\prod (\lambda_i - \lambda) = 11!\varepsilon$. The situation is illustrated in Fig. 1.

Taking ε to be negative, we obtain a double root when the line $y = 11!\varepsilon$ is tangential to the curve, which first occurs at a point between λ_{11} and λ_{12}. If we take ε positive, a double root is obtained when the line is tangential to the curve at a point between λ_{10} and λ_{11}. It is surprisingly easy to compute these points quite accurately, provided $\prod (\lambda_i - \lambda)$ is computed from G_{12}, not from F_{12}! The value of ε is quite a lot smaller than $\|F_{12}\|s_{11}$, and on reflection, this is not surprising. In establishing that result, we attempted to induce a double eigenvalue at the value λ_i itself for which the s_i is small. It is to be expected that a smaller perturbation is needed to produce a double eigenvalue at some point "between" that λ_i and some other λ_j. As we have seen, there are always perturbations εB for which $\partial\lambda_i/\partial\varepsilon = \pm 1/s_i$. At least one of the other λ_j must be changing fast "to keep the trace correct", and we would expect to be able to make $\lambda_i(\varepsilon)$ and some $\lambda_j(\varepsilon)$ move towards each other. As they get nearer, we would expect s_i to get even smaller, and one feels intuitively that a perturbation nearer the order of magnitude $|\frac{1}{4}(\lambda_i - \lambda_j)s_i|$ is likely to give a double eigenvalue at a value of roughly $\frac{1}{2}(\lambda_i + \lambda_j)$. The quantity $|\frac{1}{4}(\lambda_i - \lambda_j)s_i|$ is likely to be much smaller than $\|A\|s_i$ since the relevant λ_j is likely to be at least "fairly close" to λ_i. This certainly proves to be true for F_{12}. In fact, a value $\varepsilon = -10^{-10}(3.95 \cdots)$ gives a double root between λ_{11} and λ_{12} at $\lambda = 0.038 \cdots$.

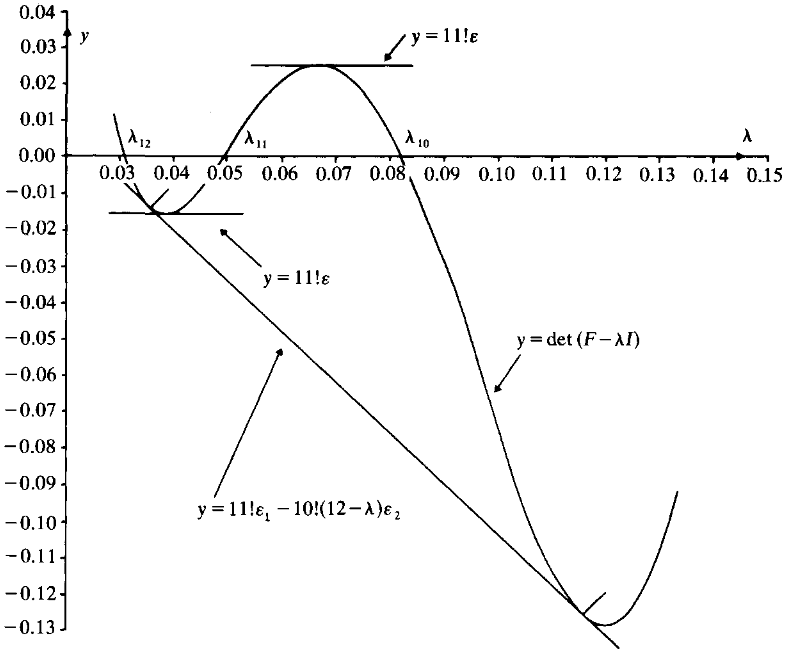


FIG. 1

If perturbations ε_1 and ε_2 are made in $F_{12}(1, 12)$ and $F_{12}(2, 12)$, respectively, then the characteristic equation becomes

$$(13.7) \qquad \prod(\lambda_i - \lambda) - 11!\varepsilon_1 + 10!\varepsilon_2(12 - \lambda) = 0,$$

and the eigenvalues are at the intersection of the straight line $y = 11!\varepsilon_1 - 10!\varepsilon_2 \times (12 - \lambda)$ with the curve $y = \prod(\lambda_i - \lambda)$. By appropriate choices of ε_1 and ε_2, this line can be made tangential to the curve at two points, one between λ_{12} and λ_{11} and one between λ_{10} and λ_9. The values are in fact $\varepsilon_1 = -10^{-7}(6.24 \cdots)$ and $\varepsilon_2 = -10^{-7}(3.9 \cdots)$ and it gives coincident eigenvalues at $0.036 \cdots$ and 0.116. Notice that if one attempts to solve these perturbed matrices by the QR algorithm on KDF9, the separation of the "paired" eigenvalues may, at first sight, seem disappointing. Two points should be emphasized. First, since the KDF9 has a mantissa with less than 12 decimal digits, the perturbations ε_i cannot be inserted with any great precision since they occur via entries $1 + \varepsilon_i$. Hence even if the ε_i are determined accurately, they cannot be included in $1 + \varepsilon_i$ without incurring an error of between 10^{-11} and 10^{-12}. Further, in solving the perturbed matrix $A + E$ on KDF9, the effect of rounding errors will imply that a computed λ_i is an eigenvalue of $A + E + E_i$ when $\|E_i\|_2/\|A\|_2$ is likely to be a modest multiple of 2^{-39} (i.e., $10^{-11.7}$). Since we are now extremely close to a defective matrix, s_i will be quite a lot smaller than the corresponding value for A itself. In fact, with $\varepsilon_1 = -10^{-10}(3.9 \cdots)$, the two close computed values of λ were $0.03758 \cdots$ and $0.03968 \cdots$, the mean of these being $0.03863 \cdots$; this is very close to the minimum point of $\prod(\lambda - \lambda_i)$ between λ_{12} and λ_{11}. Again working with the perturbed version of G_{12}, it is possible not only to insert the perturbations accurately (since they now arise as $\varepsilon_1 - \varepsilon_2$ and ε_2 and not as $1 + \varepsilon_1$ and $1 + \varepsilon_2$), but also to compute the eigenvalues of the perturbed matrix accurately. Altogether, the Frank matrices provide good material for investigating ill-conditioned eigenvalues and eigenvectors. It is clear that by the time $n = 20$, F_n is very near to a large number of defective matrices having different sets of multiple eigenvalues and even elementary divisors of different degrees. It is natural to ask what information one should really extract and why.

Continuing with F_{12} and KDF9 (and we make no excuse for being so specific—the "difficulty" involved in dealing with a matrix is intimately associated with the precision of computation one is prepared to use; on a 40-decimal digit computer F_{12} could reasonably be regarded as well-conditioned!) the dilemma is particularly acute. The computed $\lambda_9, \lambda_{10}, \lambda_{11}, \lambda_{12}$ all have *some* accuracy, and it is debatable whether there is anything to be gained by pretending that they are equal or equal in pairs, etc. On the other hand, if one treats them as distinct and computes the corresponding eigenvectors, not only will these eigenvectors inevitably be inaccurately determined, but they will also be almost linearly dependent. Indeed, if we use the QR algorithm, they will be exact for some $A + E$ with $\|E\|_2/\|A\|_2$ of the order of 2^{-39}. The s_i for this matrix will be quite close to those of A itself, and the smallest of these is roughly 3×10^{-8}. How much information do we have at best about the space of dimension four spanned by the corresponding eigenvectors? From § 5 we see that these vectors are linearly dependent to within less than 3×10^{-8}. Certainly the fourth orthogonal direction is extremely poorly-determined. Indeed, all four vectors are "fairly" parallel, and in performing the Schmidt orthogonalization process, there will be a loss of figures at each stage.

Would it not be better to group these four eigenvalues together and to attempt to determine directly a set of four orthogonal vectors spanning the corresponding invariant subspace? One can certainly determine the subspace in this way much more accurately. Whether it is better or not depends on what one really wants. If accuracy is an overriding consideration, the "best" thing to do is to group all 12 eigenvalues together, and then any 12 orthogonal vectors specify the subspace exactly, $e_1, e_2, \cdots, e_{12}$ being an obvious choice! Here we have the ultimate absurdity of perfect accuracy in a set of vectors but no information.

A sensible compromise would seem to be the following. On a t-digit computer, we might aim to determine the smallest groupings of the eigenvalues for which one can claim that all the computed orthogonal bases "have t' correct digits". Obviously one must have $t' < t$, and if one insists on t' being too close to t, one runs the risk of being forced into large groups with a consequent loss of information. There is no need to get into abstruse discussions about the meaning to be attached to the angle between a computed set of s orthogonal vectors and an exact set of s vectors defining the subspace. Since we are unlikely to require less than 3 decimal digits (say), we would merely be arguing about the relative merits of θ, $\sin\theta$, $\tan\theta$, $2\sin(\theta/2)$, etc., when $\theta < 10^{-3}$, and clearly such matters are of no importance. The following is a perfectly adequate measure of the angle between the orthonormal set $u_1, u_2, \cdots, u_s$ and the orthonormal set $v_1, v_2, \cdots, v_s$. We may write

$$u_i = \alpha_{i1}v_1 + \cdots + \alpha_{is}v_s + r_i, \qquad i = 1, \cdots, s, \tag{13.8}$$

and the r_i might reasonably be called the residual vectors. If the two bases spanned the same subspace, then $r_i = 0$. Therefore max $\|r_i\|$ may be regarded as a measure of the errors in the u_i relative to the v_i. In fact, $\|r_i\|$ is the sine of the angle between u_i and the space spanned by the v_i.

14. Calculation of orthogonal bases of invariant subspaces. In classical similarity theory, unitary similarities play quite an important role, since when $X^H = X^{-1}$,

$$B = X^{-1}AX = X^HAX, \tag{14.1}$$

and hence matrices which are unitarily similar are also *conjunctive*. The fundamental result with respect to unitary similarities is that for any complex matrix A, there exists a unitary matrix X such that

$$X^HAX = T, \tag{14.2}$$

where T is upper triangular with the eigenvalues of A on its diagonal. This is known as the *Schur canonical form*. The ordering of the λ_i on the diagonal may be chosen arbitrarily.

Unitary transformations are of great significance for numerical analysts because a wide range of algorithms based on them are numerically stable. When A is real, it may in general have some complex eigenvalues, though these of course occur in conjugated pairs. It is convenient to remain in the real field whenever possible, and there is a single modification of Schur's result which states that when A is real, there is an *orthogonal* X (i.e., a real unitary X) such that

$$X^TAX = T, \tag{14.3}$$

where T is now almost triangular, except that corresponding to each complex conjugate pair of eigenvalues T has a 2×2 block on the diagonal having as its two eigenvalues this complex pair. This is usually known as the Wintner–Murnaghan canonical form [29].

It is precisely this form which is produced by the double Francis QR algorithm, perhaps the most widely used general-purpose algorithm for finding the eigensystem of a nonnormal real matrix. This algorithm works directly with a real-upper Hessenberg matrix but a general real matrix may be reduced to this form by a (real) orthogonal similarity. (For detailed discussions, see [28].) The combined reduction from general form to real almost triangular T is extremely stable, and it has been proved [25] that the computed matrix is such that

$$T = X^T(A + E)X, \tag{14.4}$$

where X is exactly orthogonal and $\|E\|/\|A\|$ is a modest multiple of the machine precision. Further, the computed $\bar{X}$ is very close to the exactly orthogonal X for which (14.4) is true and hence, in particular, has columns which are orthogonal almost to working accuracy. Since the computed T is exactly orthogonally similar to $A + E$ and the s_i are invariant with respect to orthogonal transformations, the s_i of the matrix T give information that really is relevant. The left-hand and right-hand eigenvectors of T may be readily computed; the right-hand eigenvectors are required in any case, and the additional work needed to compute the left-hand eigenvectors of T is a negligible percentage of that for the complete reduction. Ignoring the 2×2 blocks for the moment, we determine the left-hand and right-hand vectors for the eigenvalue in position r on the diagonal by a triangular back substitution with matrices of order $n - r$ and r, respectively. The vectors are of the forms

$$(0, \cdots, 0, y_r, y_{r+1}, \cdots, y_n) \quad \text{and} \quad (x_1, x_2, \cdots, x_r, 0, \cdots, 0), \tag{14.5}$$

and if these are normalized vectors, the corresponding $s = x_r y_r$. The complication caused by the 2×2 blocks is not substantial and is discussed in detail in [28, pp. 372, 374]. The computed s_i are invaluable in any case, since they give the sensitivities of the eigenvalues of T, i.e., of $(A + E)$.

Now let us consider the eigenvalues in the first s positions along the diagonal of T. We may write

$$X^{-1}(A + E)X = \begin{array}{c} \left[\begin{array}{cc} T_{11} & T_{12} \\ 0 & T_{22} \end{array}\right] \begin{array}{l} \}s \\ \}n-s \end{array} \\ \begin{array}{cc} \underbrace{\quad}_{s} & \underbrace{\quad}_{n-s} \end{array} \end{array}, \tag{14.6}$$

and hence

$$(A + E)X_s = X_s T_{11}, \tag{14.7}$$

where X_s consists of the first s columns of the orthogonal matrix X. Notice that this is true even if there are 2×2 blocks included in T_{11}, provided the first of a pair of conjugate eigenvalues is not in position s. These s orthogonal vectors therefore provide an orthogonal basis for the invariant subspace of $A + E$ corresponding to this group of s eigenvalues, and, as we have remarked, even the *computed* columns of X are accurately orthogonal. They do, of course, provide information only about the subspaces of $A + E$ rather than of A itself, but any loss of accuracy due to this perturbation is inherent in the problem and cannot be avoided without working to a higher precision (or exactly!) at least in some significant part of the computation. Although the individual eigenvectors corresponding to those s eigenvalues may be almost linearly dependent, the columns of X_s, being orthogonal, cannot have this shortcoming.

There is no *correspondingly simple* method for computing a set of orthogonal vectors giving the invariant subspace corresponding to a set of λ_i which are not in the leading position. However, given any collection of λ_i, it is possible to transform T into an upper triangular $\tilde{T}$ having these λ_i in the leading positions by means of an orthogonal similarity. Hence we have an orthogonal Y such that

$$Y^T(T + F)Y = \tilde{T}, \tag{14.8}$$

where F is the result of rounding errors, and since the process is stable, $\|F\|/\|T\|$ and hence $\|F\|/\|A\|$ is of the order of the machine precision. Hence, finally,

$$Y^T X^T(A + E + G)YX = \tilde{T}, \tag{14.9}$$

where $G = XFX^T$ and $\|G\|_2 = \|F\|_2$, and the first s columns of YX give an orthogonal basis of the subspace corresponding to the selected s eigenvalues.

The transformation from T to $\tilde{T}$ was first described by Ruhe [14]. It is achieved by a sequence of orthogonal similarities, each of which is a plane rotation and is based on the following observation. If

$$T = \begin{bmatrix} p & q \\ 0 & r \end{bmatrix}, \tag{14.10}$$

then there is a plane rotation R such that

$$R^T T R = \begin{bmatrix} r & q \\ 0 & p \end{bmatrix}.$$

If this is true, then clearly

$$R^T \begin{bmatrix} 0 & q \\ 0 & r - p \end{bmatrix} R = \begin{bmatrix} r - p & q \\ 0 & 0 \end{bmatrix} \tag{14.11}$$

or

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 0 & q \\ 0 & r - p \end{bmatrix} \quad \begin{bmatrix} r - p & q \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{14.12}$$

For this to be true, $(r - p)\cos\theta - q\sin\theta = 0$, giving

$$\cos\theta = q/\alpha, \quad \sin\theta = (r - p)/\alpha, \quad \alpha = \sqrt{(r - p)^2 + q^2}, \tag{14.13}$$

and a simple verification shows that with this choice of R, the relation *is* true. Ruhe gave the analogous result in the complex case; in this, q becomes $\bar{q}$ in the transformed matrix. Using this algorithm, any eigenvalue may be brought into any required position along the diagonal by a sequence of plane rotations. When T is real but has 2 × 2 blocks corresponding to complex eigenvalues, an analogous result is true in which a complex pair is always kept together in the form of a real 2 × 2 block. One needs only two additional algorithms which serve to interchange the position of a single real diagonal element and a real 2 × 2 block and to interchange the positions of two 2 × 2 blocks. (N.B., the 2 × 2 blocks need not remain invariant; only their eigenvalues.) The relevant algorithms have been coded on KDF9 and are numerically stable.

There remains the problem of the grouping, and there does not yet appear to be a perfectly satisfactory method of deciding on this. It cannot be decided purely on the basis of the separation, since even multiple eigenvalues corresponding to elementary divisors of moderate degree will not in general lead to "close" eigenvalues in the computed set. Further, even when the exact λ_i and λ_j are by no means pathologically close, they may be so sensitive that small perturbations in A may make them so. A good working test is that a perturbation E *may* make them coincident if

$$\frac{\|E\|_2}{|s_i|} + \frac{\|E\|_2}{|s_j|} \geqq |\lambda_i - \lambda_j|, \tag{14.14}$$

though since $\|E\|/s_i$ is merely a first order perturbation, a smaller $\|E\|$ than this may well be adequate.

However, we are not merely concerned with whether the computed λ_i and λ_j could belong to a multiple root. If this is our criterion, then the groups will be much smaller than is advisable. A reasonably satisfactory rule is that if our aim is to have t' decimal digits correct in the subspace on a t-digit decimal computer, then λ_j should be coupled with λ_i when

$$(14.15) \qquad |\lambda_i - \lambda_j| \max(|s_i|, |s_j|) \leqq 10^{(t'-t)}\|A\|_F,$$

where $\|A\|_F$ rather than $\|A\|_2$ is used since a practical criterion is required. This criterion has been applied on KDF9 and found to be quite sound. We may illustrate this in action by means of a simple example. Consider the matrix

$$(14.16) \qquad \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ \varepsilon & 0 & 1 \end{bmatrix}.$$

The eigenvalues are $1 + \varepsilon^{1/3}$, $1 + \omega\varepsilon^{1/3}$, $1 + \omega^2\varepsilon^{1/3}$, where ω is a complex cube root of unity. The separation is $\varepsilon^{1/3}\sqrt{3}$, and hence when ε is of the order of machine precision, the eigenvalues will not appear unduly close. But the left-hand eigenvector corresponding to $\varepsilon^{1/3}$ is $[\varepsilon^{2/3}, \varepsilon^{1/3}, 1]$, and the right-hand eigenvector is $[1, \varepsilon^{1/3}, \varepsilon^{2/3}]$, and hence the corresponding $s_1 = 3\varepsilon^{2/3}/(1 + \varepsilon^{2/3} + \varepsilon^{4/3})$ with similar results for the other eigenvalues. Hence $(\lambda_1 - \lambda_2)s_1 \doteqdot 3\sqrt{3}\varepsilon$, and this product fully exposes the danger. These two eigenvalues would be grouped together even if one were making the tolerance very lax.

One difficulty encountered in experimentation with algorithms for finding invariant subspaces is that of obtaining a correct orthogonal basis against which to test computed subspaces except in the case of rather artificially constructed matrices. In practice, we have found it useful to work with A itself and with $\tilde{A}$ such that $\tilde{a}_{ij} = a_{n+1-i,n+1-j}$, i.e., $\tilde{A}$ is the reflection of A in its center point. Eigenvectors, etc., of $\tilde{A}$ are merely those of A with components in the reverse order. If A and $\tilde{A}$ are solved by the same algorithm, then one can compare orthogonal bases obtained with the two matrices. At least one computed subspace has an error which is of the order of magnitude of the angle between the two computed subspaces. Where it has been possible to determine a correct basis by independent means, the error in each of the computed subspaces has proved to be of the same order of magnitude as the angle between them. One might expect this to be true generally unless there is some special reason for errors to be correlated in some way.

For matrices with well-defined J.c.f.'s, the orthogonal bases determined by an algorithm based on the above have been correct almost to working accuracy. Even if one takes t' almost equal to t, only the eigenvalues associated with multiple roots have been grouped together.

The results obtained with the Frank matrices are interesting. For $n = 16$, the 9 smallest computed eigenvalues and their true values are given in Table 1. Six of the computed values are complex and with imaginary parts which are quite comparable with the real parts. Only with λ_{10} do we begin to have any significant accuracy, and λ_9 has four correct figures. The largest eigenvalues were given very accurately.

TABLE 1

Computed eigenvalues	True eigenvalues
$\lambda_{16} = -0.02710 + i\,(0.04506)$	$\lambda_{16} = 0.02176$
$\lambda_{15} = -0.02710 - i\,(0.04506)$	$\lambda_{15} = 0.03133$
$\lambda_{14} = 0.06121 + i\,(0.09907)$	$\lambda_{14} = 0.04517$
$\lambda_{13} = 0.06121 - i\,(0.09907)$	$\lambda_{13} = 0.06712$
$\lambda_{12} = 0.1882 + i\,(0.06248)$	$\lambda_{12} = 0.1051$
$\lambda_{11} = 0.1882 - i\,(0.06248)$	$\lambda_{11} = 0.1775$
$\lambda_{10} = 0.3342$	$\lambda_{10} = 0.3307$
$\lambda_9 = 0.6809$	$\lambda_9 = 0.6809$
$\lambda_8 = 1.469$	$\lambda_8 = 1.469$

Orthogonal bases were computed for subspaces of dimensions 2, 4, 6, 7, 8, 9 obtained by grouping the corresponding number of smallest eigenvalues together. (Notice we did not compute spaces of dimension 3, 5 since conjugate pairs were always kept together in order to be able to work in the real field.) The angles between the computed bases and the true subspace are given in Table 2. The subspaces of order 2 and 4 are scarcely of any significant accuracy, but that of order 6 is correct to about 3 decimals and that of order 7 to almost six decimals. Notice that this accuracy in the subspace is attained, although some of the λ_i are very poor. (It should be emphasized, though, that every computed λ_i is an eigenvalue of some $A + E$, with $\|E\|/\|A\|$ of the order of 2^{-39}.)

TABLE 2

Dimension	Angle between computed and true subspace
2	3.05×10^{-2}
4	1.73×10^{-2}
6	6.23×10^{-4}
7	1.74×10^{-6}
8	1.73×10^{-8}
9	2.67×10^{-10}

We may look at these results from an alternative point of view. If the matrix F_{16} is regarded as having relative errors of order 10^{-12} in its elements, then the invariant subspace corresponding to its two smallest elements is scarcely determined at all, while that corresponding to its smallest 7 eigenvalues for example is determined to about six decimals.

15. Inverse iteration and ill-conditioned eigensystems. Inverse iteration is one of the main tools used in practice for the calculation of eigenvectors from computed eigenvalues. The motivation for inverse iteration, due to Wielandt [23], springs from the observation that if A is a matrix with a complete set of eigenvectors x_i, then an arbitrary vector may be expressed in the form

$$y = \sum_{i=1}^{n} \alpha_i x_i \tag{15.1}$$

and hence

$$(A - kI)^{-1}y = \sum_{i=1}^{n} \alpha_i x_i/(\lambda_i - k). \tag{15.2}$$

If $|\lambda_j - k| \ll |\lambda_i - k|$ $(i \neq j)$, the components of x_j will be very much larger than the coefficients of the remaining x_i, unless the vector y happens to be very deficient in x_j. If, in particular, k is a very accurate approximation to λ_j, the right-hand side of (15.2) may be written in the form

$$\left(\frac{1}{\lambda_j - k}\right)\left[\alpha_j x_j + \sum_{i \neq j} \alpha_i(\lambda_j - k)x_i/(\alpha_i - k)\right] \tag{15.3}$$

and the normalized form of this vector will be x_j to very high accuracy. However, for nonnormal matrices, a computed λ may not be particularly near to *any* eigenvalue, and it appears that one can no longer expect such a spectacular performance in one iteration.

Varah was the first to point out that this is not so. The simplest way to see this is to forget about the expansion of y and concentrate directly on the solution of $(A - \lambda I)z = y$, where $\|y\|_2 = 1$. We may write

$$(A - \lambda I)z/\|z\|_2 = y/\|z\|_2, \quad w = z/\|z\|_2, \quad y/\|z\|_2 = r \tag{15.4}$$

giving

$$(A - \lambda I)w = r, \quad \|w\|_2 = 1, \quad \|r\| = 1/\|z\|_2 = \varepsilon \quad \text{(say)}. \tag{15.5}$$

The first of equations (15.5) may be expressed in the form

$$(A - rw^H)w = \lambda w, \tag{15.6}$$

and hence λ and w are an exact eigenvalue of eigenvector of the matrix $A - rw^H$. Since $\|rw^H\|_2 = \|r\|_2 = \varepsilon$, it is evident that if $\|z\|_2$ is "large", λ and w are satisfactory since they are exact for a neighboring matrix.

Now if we start with a value of λ which is an exact eigenvalue of $A + E$, then however poor λ may otherwise be,

$$(A + E - \lambda I)q = 0 \quad \text{for some } \|q\|_2 = 1. \tag{15.7}$$

Hence $(A - \lambda I)q = -Eq$ and if one takes $y = -Eq/\|Eq\|_2$, the solution of $(A - \lambda I)z = y$ is $z = q/\|Eq\|_2$ and $\|z\| \geqq 1/\|E\|_2$. With this choice of y, then, we obtain a very large z in one iteration, and the corresponding $w = z/\|z\|_2$ is a satisfactory eigenvector corresponding to λ. Obviously, if we take as initial y an arbitrary unit vector, the probability of it being *very* deficient in the vector $-Eq/\|Eq\|_2$ is very small and hence inverse iteration will "work" in one iteration with almost any starting vector.

However, Varah also produced an argument which suggested that when λ is related to an ill-conditioned eigenvalue, there are severe disadvantages in performing more than one step of inverse iteration, and a satisfactory analysis of the phenomenon was subsequently given by Wilkinson [27]. It is instructive to analyze this phenomenon in terms of the S.V.D. decomposition. We showed in § 5 that if λ_i is an ill-conditioned eigenvalue, the associated s_i is small and the matrix X of eigenvectors has a small singular value $\sigma_n < |s_i|$. If the S.V.D. of X is

$$X = U\Sigma V^H, \qquad XV = U\Sigma, \tag{15.8}$$

then

$$u_n = \frac{Xv_n}{\sigma_n} = \frac{1}{\sigma_n}[\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n], \quad \text{where } \alpha_i = v_{in}, \tag{15.9}$$

and hence the unit vector u_n expanded in terms of the x_i has very large coefficients. If we take an arbitrary vector y, it can be expressed in the form

$$y = \beta_1 u_1 + \cdots + \beta_n u_n, \tag{15.10}$$

where the β_i are distributed in a natural way. When it is transformed to its expansion in terms of the x_i, we have

$$y = \left[\frac{\alpha_1\beta_n}{\sigma_n} + \cdots\right]x_1 + \left[\frac{\alpha_2\beta_n}{\sigma_n} + \cdots\right]x_2 + \cdots + \left[\frac{\alpha_n\beta_n}{\sigma_n} + \cdots\right]x_n, \tag{15.11}$$

and in general all the coefficients of the x_i will be very large but will be in ratios which are independent of the β_i, provided β_n is not small. From (15.11),

$$\begin{aligned} z = (A - \lambda I)^{-1}y = &\left[\frac{\alpha_1\beta_n}{\sigma_n} + \cdots\right]\frac{x_1}{\lambda_1 - \lambda} + \left[\frac{\alpha_2\beta_n}{\sigma_n} + \cdots\right]\frac{x_2}{\lambda_2 - \lambda} + \cdots \\ &+ \left[\frac{\alpha_n\beta_n}{\sigma_n} + \cdots\right]\frac{x_n}{\lambda_n - \lambda}, \end{aligned} \tag{15.12}$$

and z will in general be a large vector for two reasons: first, because σ_n is small and second, because usually one of the $(\lambda_i - \lambda)$ will be moderately small (though not usually pathologically so). Now when z is normalized prior to doing the second iteration, the coefficients of the x_i in this normalized z will no longer be special in the way that they were in the first "arbitrary" y.

In fact, the normalized vector will be essentially

$$\begin{aligned} &\frac{\lambda_i - \lambda}{\lambda_1 - \lambda}[\alpha_1 + \cdots]x_1 + \frac{\lambda_i - \lambda}{\lambda_2 - \lambda}[\alpha_2 + \cdots]x_2 + \cdots + [\alpha_i + \cdots]x_i \\ &\qquad + \cdots + \frac{\lambda_i - \lambda}{\lambda_n - \lambda}[\alpha_n + \cdots]x_n, \end{aligned} \tag{15.13}$$

and the coefficients of the x_i will be of order unity. In the first vector these coefficients were all large but canceled out to give a vector of normal size. Consequently, in the second step of inverse iteration, the growth in size will come only from the comparative smallness of a $\lambda_i - \lambda$ and will not be reinforced by the smallness of σ_n. This will be true of all subsequent steps unless at the rth step all the quantities $((\lambda_i - \lambda)/(\lambda_j - \lambda))^r$ are almost equal, when the normalized value will have large components of each of the x_i in the same ratios as in the first vector. In this case, every rth iteration will give a large growth and consequently a satisfactory vector. This situation will usually occur when A has an elementary divisor of degree r. Varah has effectively used this behavior of the iterates to give information on the structure of the J.c.f. of A [21].

The analysis may be carried out in an alternative way which is also instructive. We observe first that if λ_i is an exact eigenvalue of A, then $A - \lambda_i I$ is singular, and if

$$(A - \lambda_i I) = U\Sigma V^H, \tag{15.14}$$

then $\sigma_n = 0$. Consequently,

$$(A - \lambda_i I)v_n = 0, \qquad u_n^H(A - \lambda_i I) = 0, \tag{15.15}$$

and v_n and u_n are normalized right-hand and left-hand eigenvectors of A, with $u_n^H v_n = s_i$.

Now suppose λ_i is an exact eigenvalue of $A + E$; then $\sigma_n(A - \lambda_i I) \leqq \|E\|_2$. If we now write

$$A - \lambda_i I = U\Sigma V^H \quad \text{and} \quad (A - \lambda_i I)v_s = \sigma_s u_s, \tag{15.16}$$

then $\sigma_n \leqq \|E\|$. An arbitrary unit vector y may now be expanded in the form

$$y = \sum \alpha_j u_j, \quad \text{with } \|\alpha\|_2 = 1 \tag{15.17}$$

and

$$z = (A - \lambda I)^{-1}y = \sum \frac{\alpha_j v_j}{\sigma_j}. \tag{15.18}$$

The coefficient of v_n is α_n/σ_n, and

$$|\alpha_n/\sigma_n| \geqq \alpha_n/\|E\|. \tag{15.19}$$

Unless y is accidentally deficient in u_n, the full growth takes place in the first iteration. The normalized z is essentially of the form

$$v_n + \sum_1^{n-1} \gamma_i v_i, \tag{15.20}$$

where the γ_i are small. To see the effect of the second iteration, one requires an expansion in terms of the u_i rather than the v_i, and we now show that in this expansion the coefficient of u_n is small. Indeed, since $u_n^H v_n$ is roughly s_i from the previous argument, and all the γ_i are small, this is immediately obvious. The normalized z is therefore an unfortunate vehicle for inverse iteration since it is deficient in u_n.

16. Improvement of an invariant subspace. Suppose A has been reduced to upper triangular form T by a unitary similarity X with a group of associated λ_t in the s leading diagonal positions of T. We then have, for the computed X and T,

$$AX - XT = E. \tag{16.1}$$

The error analysis guarantees that E will be almost negligible to working accuracy.

Each element of the matrix E may be determined in practice by accumulating the whole of the inner product involved in double-precision before rounding. If $F = X^{-1}E$, then

$$X^{-1}AX = T + X^{-1}E = T + F, \tag{16.2}$$

and since the computed X is almost exactly orthogonal, one can compute F via X^TE. From an invariant subspace of $T + F$ one can improve the corresponding subspace of A itself. We partition $T + F$ in the form

$$\begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} + \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}, \tag{16.3}$$

where T_{11} contains the grouped eigenvalues. The relevant invariant subspace of T is spanned by the first s columns of I, and hence if we write

$$\left(\begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} + \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}\right)\begin{bmatrix} I \\ \hline Y \end{bmatrix} = \begin{bmatrix} I \\ \hline Y \end{bmatrix}[T_{11} + G_{11}], \tag{16.4}$$

$[I \quad Y^T]$ gives the improved subspace. From (16.4), neglecting second order quantities,

$$T_{11} + F_{11} + T_{12}Y = T_{11} + G_{11}, \quad T_{22}Y + F_{21} = YT_{11}, \tag{16.5}$$

and Y is the solution of

$$[T_{22}Y - YT_{11}] = -F_{21}. \tag{16.6}$$

The matrix Y may be determined column by column via the relations

$$T_{22}y_1 - t_{11}y_1 = -f_1, \qquad (T_{22} - t_{11}I)y_1 = -f_1, \tag{16.7}$$

$$T_{22}y_2 - t_{12}y_1 - t_{22}y_2 = f_2, \qquad (T_{22} - t_{22}I)y_2 = -f_2 + t_{12}y_1. \tag{16.8}$$

In general, the rth column of Y is the solution of a triangular system of equations with matrix $(T_{22} - t_{rr}I)$.

From Y one can determine G_{11} via (16.5).

If one includes the second order terms, then (16.6) becomes

$$[T_{22}Y - YT_{11}] = -F_{21} + [-F_{22}Y + Y(T_{12}Y + F_{11} + F_{12}Y)], \tag{16.9}$$

and after solving (16.6), an improved right-hand side is that in (16.9) in which the computed Y is used. In this way, Y may be repeatedly improved by iteration.

However, there is little point in this. The matrix F is not known exactly. There are errors made in computing E in the first place and further errors in computing $X^{-1}E$, and here no purpose is served in computing Y accurately. In (16.3) we have purposely refrained from writing

$$\begin{bmatrix} \bar{T}_{11} & \bar{T}_{12} \\ F_{21} & \bar{T}_{22} \end{bmatrix}, \tag{16.10}$$

$$\text{where } \bar{T}_1 = T_{11} + F_{11}, \quad \bar{T}_{12} = T_{12} + F_{12}, \quad \bar{T}_{22} = T_{22} + F_{22},$$

although this would have simplified the expressions. This is because it is necessary to keep the F matrix separate from the T matrix on the computer. The information in F would promptly be lost if the addition were carried out. However, there are obviously slight advantages in replacing (16.6) by

$$(16.11)\qquad (\bar{T}_{22}Y - Y_{11}\bar{T}_{11}) = -F_{21}.$$

The improved subspace X_1 is now $\tilde{X}_1 = X_1 + X_2 Y$. It is no longer quite orthogonal, but this is of no importance. If one wishes to continue with the refinement of the subspace, one should return to the computation of the residual via the relation

$$(16.12)\qquad A[\tilde{X}_1 \mid X_2] - [\tilde{X}_1 \mid X_2][\tilde{T}] = \tilde{E},$$

where

$$(16.13)\qquad \tilde{T} = \begin{bmatrix} T_{11} + G_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}.$$

The new $\tilde{E}$ will not be smaller than E in general, but the next correction to the subspace will be. If $\tilde{X} = [\tilde{X}_1 \mid X_2]$, then

$$(16.14)\qquad \tilde{X}^{-1}A\tilde{X} - \tilde{T} = (\tilde{X})^{-1}\tilde{E} = \tilde{F},$$

and one can still use the approximation $(\tilde{X})^{-1} = X^T$. When computing the new correction Y, the equations corresponding to (16.6) will be

$$(16.15)\qquad [T_{22}\tilde{Y} - \tilde{Y}\tilde{T}_{11}] = -\tilde{F}_{21},$$

and $\tilde{T}_{11} = T_{11} + S_{11}$ is no longer upper triangular. However, we may use T_{11} in place of $\tilde{T}_{11}$ since YS_{11} will be of second order.

The process of iterative refinement is wholly analogous to that used with linear equations (see, e.g., [25, Chap. 4]). In general, we can continue until we have a basis which is correct to working accuracy. Indeed, at the time when the process terminates, the new X_1 will be obtained in terms of the old X_1 and the original X_2 by the relation

$$(16.16)\qquad X_1\,(\text{new}) = X_1\,(\text{old}) + X_2 Y,$$

where $X_2 Y$ is small. The true sum on the right-hand side will be accurate to more than the working precision.

The final X_1 will not have orthogonal columns, but they will be almost orthogonal. If true orthogonality is required, no appreciable loss of accuracy will occur when the Schmidt orthogonalization process is performed.

Acknowledgments. The authors wish to thank Mrs. G. Peters of the National Physical Laboratory and Dr. Peter Businger of the Bell Telephone Laboratories for the computational work on invariant subspaces, and Dr. B. Heap and Mr. Clive Hall also of the National Physical Laboratory for the production of Fig. 1 on the computer KDF9.

REFERENCES

[1] C. Davis and W. M. Kahan, *The rotation of eigenvectors by a perturbation. III*, SIAM J. Numer. Anal., 7 (1970), pp. 1–46.

[2] D. K. Faddeev and V. N. Faddeeva, *Computational Methods of Linear Algebra*, W. H. Freeman, San Francisco and London, 1963.

[3] G. H. GOLUB AND W. M. KAHAN, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM J. Numer. Anal., 2 (1965), pp. 205–224.

[4] G. H. GOLUB AND C. REINSCH, *Singular value decomposition and least squares solutions*, Numer. Math., 14 (1970), pp. 403–420.

[5] R. T. GREGORY, *Defective and derogatory matrices*, this Review, 2 (1960), pp. 134–139.

[6] A. S. HOUSEHOLDER, *The Theory of Matrices in Numerical Analysis*, Ginn (Blaisdell), Boston, 1964.

[7] W. M. KAHAN, *Conserving confluence curbs ill-condition*, Computer Sci. Tech. Rep. 6, Univ. of Calif., 1972.

[8] T. KATO, *Perturbation Theory for Linear Operators*, Springer-Verlag, Berlin, 1966.

[9] G. KREISSELMEIER, *On the determination of the Jordan form of a matrix*, IEEE Trans. Automatic Control, AC-18 (1973), pp. 686–687.

[10] V. N. KUBLANOVSKAYA, *On a method of solving the complete eigenvalue problem for a degenerate matrix*, Ž. Vyčisl. Mat. i Mat. Fiz., 6 (1966), pp. 611–620 = USSR Computational Math and Math. Phys., 6 (1968), no. 4, pp. 1–14.

[11] P. LANCASTER, *Theory of Matrices*, Academic Press, New York, 1969.

[12] B. N. PARLETT AND C. REINSCH, *Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numer. Math., 13 (1969), pp. 293–304.

[13] G. PETERS AND J. H. WILKINSON, *Studies in Numerical Analysis*, B. K. P. Scaife, ed., Academic Press, New York and London, 1974.

[14] A. RUHE, *An algorithm for numerical determination of the structure of a general matrix*, BIT, 10 (1970), pp. 196–216.

[15] ———, *Perturbation bounds for means of eigenvalues and invariant subspaces*, Ibid., 10 (1970), pp. 343–354.

[16] ———, *Properties of a matrix with a very ill-conditioned eigenproblem*, Numer. Math., 15 (1970), pp. 57–60.

[17] G. W. STEWART, *On the sensitivity of the eigenvalue problem*, SIAM J. Numer. Anal., 9 (1972), pp. 669–686.

[18] ———, *Error bounds for invariant subspaces of closed operators*, Ibid., 8 (1971), pp. 796–808.

[19] ———, *Introduction to Matrix Computations*, Academic Press, New York and London, 1973.

[20] J. M. VARAH, *Rigorous machine bounds for the eigensystem of a general complex matrix*, Math. Comp., 22 (1968), pp. 793–801.

[21] ———, *Computing invariant subspaces of a general matrix when the eigensystem is poorly conditioned*, Ibid., 24 (1970), pp. 137–149.

[22] ———, *Invariant subspace perturbations for a non-normal matrix*, IFIP Conf., 1971.

[23] H. WIELANDT, *Bestimmung höherer eigenwerte durch gebrochene iteration*, Ber, B44/J/37 der Aerodynamischen Versuchanstalt Göttingen, 1944.

[24] J. H. WILKINSON, *Rounding Errors in Algebraic Processes*, Her Majesty's Stationery Office, Prentice-Hall, Englewood Cliffs, N.J., 1963.

[25] ———, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1968.

[26] ———, *Note on matrices with a very ill-conditioned eigenproblem*, Numer Math., 19 (1972), pp. 176–178.

[27] ———, *Inverse iteration in theory and in practice*, Istituto Nazionale di Alta Matematica Symposia Mathematica, 10 (1972).

[28] J. H. WILKINSON AND C. REINSCH, *Handbook for Automatic Computation, Vol. II: Linear Algebra*, Springer-Verlag, Berlin and New York, 1971.

[29] A. WINTNER AND F. D. MURNAGHAN, *A canonical form for real matrices under orthogonal transformations*, Proc. Nat. Acad. Sci. U.S.A., 17 (1931), pp. 417–420.

29

THE BLOCK LANCZOS METHOD FOR COMPUTING EIGENVALUES (WITH R. UNDERWOOD)

PROCEEDINGS OF THE SYMPOSIUM ON MATHEMATICAL SOFTWARE,
University of Wisconsin, Madison, March 1977.

Richard Ray Underwood

Born 29 January 1945, in Long Beach, California, USA;
Died 3 January 1985, in St. Louis, Missouri, USA.

The Block Lanczos Method for Computing Eigenvalues

G. H. Golub
R. Underwood

ABSTRACT

In this paper, we describe a Block Lanczos method for computing a few of the least or greatest eigenvalues of a sparse symmetric matrix. A basic result of Kaniel and Paige describing the rate of convergence of Lanczos' method will be extended to the Block Lanczos method. The results of experiments conducted with this method will be presented and discussed.

1. INTRODUCTION.

Often it is necessary to compute the algebraically greatest or least eigenvalues of a large, sparse symmetric matrix A where r is much less than n , the order of A. In this paper, we describe an algorithm for solving this problem and present the results of experiments in which a program implementing this algorithm was applied to a variety of problems. A theorem describing the rates of convergence of the computed eigenvalues to the true eigenvalues will also be presented along with its proof.

The algorithm we will describe is an extension of the method of minimized iterations due to Lanczos [1]. In this procedure, the elements of a symmetric tridiagonal matrix similar to A are generated by an iterative procedure. Lanczos proposed using these elements to compute the coefficients of a polynomial whose roots are eigenvalues of A and described how vectors generated in the course of the iteration could be combined to form the eigenvectors corresponding to these eigenvalues. However, due to certain practical difficulties, Lanczos' method was supplanted by Given's and Householder's methods for solving the general symmetric eigenproblem.

Paige [9] experimented with Lanczos' method and found that a few of the least and greatest eigenvalues of the tridiagonal matrix would often converge rapidly to those of A long before the process was completed. A similar approach was also suggested by Karush [10].

The method we will describe, the Block Lanczos Method, is an extension of Lanczos' method in which we iterate with a block of vectors rather than a single vector. In place of the tridiagonal matrix generated in Lanczos' method, we generate a <u>block</u> tridiagonal matrix. The Block Lanczos method can furthermore be used in the manner proposed by Paige. That is, one can compute a sequence of estimates to the eigenvalues of A from the block tridiagonal matrix. Our extension is similar in spirit to the work of Hestenes and Karush [2] on the method of steepest descent and generalizes Lanczos' method in the same way that simultaneous iteration generalizes the power method. It not only enables us to compute several eigenvalues and eigenvectors simultaneously, but affords us improved rates of convergence.

Several researchers have been instrumental in developing the Block Lanczos method. In particular, Kahan and Parlett [3], Cullum and Donath [4], Lewis [5], and Underwood [6] following a suggestion of Golub. The results and work reported here are taken in large part from [6].

In the next section we will describe and develop the Block Lanczos method. In Section 3, we will discuss its numerical properties and problems associated with its application and in Section 4, we will describe the results of certain experiments carried out with a program implementing this method. Finally, in Section 5, we will mention recent developments and work done with the Block Lanczos method.

2. <u>A BLOCK LANCZOS ALGORITHM</u>.

Our development of this algorithm follows the path taken by Lanczos in the development of his method and is described in greater detail in [6].

Starting from an initial n-by-p orthonormal matrix X, our goal is to compute a sequence of mutually orthonormal n-by-p matrices $X_1, X_2, \dots, X_s$ such that the space of vectors spanned by the columns of these matrices contains the columns of the matrices $X, AX, A^2X, \dots, A^{s-1}X$. To simplify our explanation, we assume that $n \gg 1$, the value of p is greater than zero and less than or equal to $n/2$, and s is greater than 1 and less than or equal to n/p. Note that it will usually be the case that $p \cdot s \ll n$. To start with, we let $X_1 = X$, compute AX_1 and Z_2 where

$$Z_2 = AX_1 - X_1M_1 \tag{2.1}$$

and M_1 is a p-by-p matrix chosen so that the Euclidean norm of Z_2 is minimized with respect to all possible choices of M_1. It can be shown that $\|Z_2\|$ is minimized when $M_1 = X_1^tAX_1$. With this choice for M_1, we have that

$$Z_2 = (I-X_1X_1^t)AX_1 . \tag{2.2}$$

That is, Z_2 is the projection of AX_1 onto the space orthogonal to that spanned by the columns of X_1. The matrix X_2 is then obtained by orthonormalizing Z_2:

$$Z_2 = X_2R_2 \tag{2.3}$$

where R_2 is a p-by-p matrix which is usually but not necessarily taken to be upper triangular. (X_2 and R_2 are by no means unique.) Note that R_2 will be singular if Z_2 is rank deficient. In the case that R_2 is upper triangular rank deficiency means that some of the columns of X_2 can be chosen arbitrarily. To make the algorithm go through, it is necessary only to choose these columns so that they are orthonormal to X_1. In any event, we have that X_2 is orthogonal to X_1. From (2.2) and (2.3), it is easily seen that the space spanned by X_1 and X_2 contains the columns of X and AX.

The remaining matrices in the sequence $X_1, X_2, \dots, X_s$ are computed as follows: Given $X_1, X_2, \dots, X_i$ for

$i \geq 2$, we compute AX_i and Z_{i+1} where

$$(2.4) \qquad Z_{i+1} = AX_i - X_1B^t_{i,1} - X_2B^t_{i,2} - \dots - X_{i-1}B^t_{i,i-1} - X_iM_i$$

and $B_{i,1}$, $B_{i,2}$, ... , $B_{i,i-1}$, and M_i are chosen so that the Euclidean norm of Z_{i+1} is minimized with respect to all possible choices of these matrices. It can be shown that $\| Z_{i+1} \|$ is minimized when

$$B_{i,j} = X^t_iAX_j$$

and

$$M_i = X^t_iAx_i$$

and hence,

$$(2.5) \qquad Z_{i+1} = (I - X_1X^t_1 - X_2X^t_2 - \dots - X_iX^t_i)AX_i \ .$$

Thus, Z_{i+1} is the projection of AX_i onto the space orthogonal to that spanned by the columns of X_1, X_2, ... , X_i . We now compute X_{i+1} by orthonormalizing Z_{i+1} :

$$(2.6) \qquad Z_{i+1} = X_{i+i}R_{i+1}$$

where R_{i+1} is a p-by-p matrix and X_{i+1} is orthonormal. Since Z_{i+1} is orthogonal to X_1, X_2, ... , X_i , we have that X_{i+1} is orthogonal to all previous X_j . As before, if Z_{i+1} is rank deficient, we choose the additional columns of X_{i+1} so that they are orthonormal to the columns of X_1, X_2, ... , X_i .

By equations (2.4) and (2.6), we have that

$$AX_i = X_1B^t_{i,1} + X_2B^t_{i,2} + \dots + X_{i-1}B^t_{i,i-1} + X_iM_i + X_{i+1}R_{i+1} \ .$$

From this last equation, we conclude from the orthogonality of the X_j that

$$B_{i,j} = \theta$$

for $j = 1, 2, \dots , i-2$, since AX_j is a combination of X_1, X_2, ... , X_{j+1} . Thus, we have

(2.7) $$AX_i = X_{i-1}B^t_{i,i-1} + X_iM_i + X_{i+1}R_{i+1} \ .$$

It can be easily shown that

$$B_{i,i-1} = R_i \ .$$

Equations (2.1), (2.3), and (2.7) can therefore be written as follows:

(2.8) $$AX_1 = X_1M_1 + X_2R_2$$

(2.9) $$AX_i = X_{i-1}R^t_i + X_iM_i + X_{i+1}R_{i+1}$$

for $i = 2, 3, \dots, s-1$, and

(2.10) $$AX_s = X_{s-1}R^t_s + X_sM_s + Z_{s+1} \ ,$$

or

$$A\bar{X}_s = \bar{X}_s\bar{M}_s + \bar{Z}_{s+1}$$

where

$$\bar{M}_s = \begin{pmatrix} M_1 & R_2^t & \theta & & . & . & . & \theta \\ R_2 & M_2 & R_3^t & . & & & & . \\ \theta & R_3 & M_3 & & . & & & . \\ . & . & & . & & . & & . \\ . & & . & & . & & . & . \\ . & & & . & & . & & \theta \\ & & & & . & & M_{s-1} & R_s^t \\ \theta & . & . & . & & \theta & R_s & M_s \end{pmatrix}$$

$$\bar{X}_s = (X_1, X_2, \dots, X_s)$$

and

$$\bar{Z}_{s+1} = (\theta, \theta, \dots, \theta, Z_{s+1}) \ .$$

Since Z_{s+1} is orthogonal to $X_1, X_2, \dots, X_s$ we have that

$$\bar{X}^t_s\bar{Z}_{s+1} = \theta$$

and

$$\bar{X}^t_sA\bar{X}_s = \bar{M}_s \ .$$

From (2.5) and (2.6), it can be shown [6] that the space spanned by $X_1, X_2, \ldots, X_i$ contains the columns of the matrices $X, AX, A^2X, \ldots, A^{i-1}X$ for all $i = 1, 2, \ldots, s$.

Using equations (2.8), (2.9), and (2.10), we define the following <u>Block Lanczos Algorithm</u>: Given p and s where $1 \leq p \leq n/2$ and $1 \leq s \leq n/p$, and X, an n-by-p orthonormal matrix, compute sequences of matrices $X_1, X_2, \ldots, X_s$, $M_1, M_2, \ldots, M_s$, and $R_2, R_3, \ldots, R_s$ satisfying equations (2.8), (2.9), and (2.10) as follows:

Step 1. Let $X_1 = X$. Compute AX_1 and $M_1 = X_1^t AX_1$.

Step 2. For $i=1, 2, \ldots, s-1$, do the following three steps:

Step 2a. Compute

$$Z_{i+1} = \begin{cases} AX_1 - X_1M_1 & \text{if } i=1 \text{, or} \\ AX_i - X_iM_i - X_{i-1}R_i^t & \text{if } i>1 . \end{cases}$$

Step 2b. Compute X_{i+1} and R_{i+1} such that X_{i+1} is orthonormal, R_{i+1} is upper triangular and

$$Z_{i+1} = X_{i+1}R_{i+1} .$$

If Z_{i+1} is rank deficient, choose the columns of X_{i+1} so that they are orthogonal to all previous X_j.

Step 2c. Compute AX_{i+1} and

$$M_{i+1} = X_{i+1}^t AX_{i+1} .$$

Because of the numerical properties of this algorithm (cf. Section 3), it will be necessary to modify step 2b to ensure that X_{i+1} is orthogonal to all previous X_j even when Z_{i+1} is <u>not</u> rank deficient. This point will be discussed further in Section 3.

Note that $\bar{M}_s$ is a symmetric block tridiagonal matrix and since the R_i are chosen to be upper triangular, it is also a band matrix with half-band width $p+1$. If it were the case that $p \cdot s = n$, then $\bar{M}_s$ would be similar to A and the eigenvalues of $\bar{M}_s$ would also be eigenvalues of A. In general, because of the numerical properties of the Block Lanczos Algorithm mentioned above, it is not practical to carry the method through to completion. The value of the algorithm lies in the fact that some of the least (and greatest) eigenvalues of $\bar{M}_s$ will closely approximate the corresponding eigenvalues of A for values of s such that $p \cdot s << n$, as the following theorem indicates.

<u>Theorem 1</u> [6]. Let $\lambda_1 \le \lambda_2 \le \dots \le \lambda_n$ be the eigenvalues of A with orthonormalized eigenvectors $q_1, q_2, \dots, q_n$. Assume that $\lambda_p < \lambda_{p+1}$. Let $\mu_1 \le \mu_2 \le \mu_3 \dots \le \mu_{ps}$ be the eigenvalues of $\bar{M}_s$. Let

$$W \equiv \begin{bmatrix} W_1 \\ W_2 \end{bmatrix} \equiv Q^t X$$

where

$$Q = (q_1, q_2, q_3, \dots, q_n)$$

and W_1 is a p-by-p matrix composed of the first p rows of W. Suppose that W_1 is *nonsingular* so that $\sigma_{\min}$, the smallest singular value of W_1 , is greater than zero.

Then for $i = 1, 2, \dots, p$,

$$\lambda_k \le \mu_k \le \lambda_k + \varepsilon_k^2$$

where

$$\varepsilon_k^2 = \frac{(\lambda_n - \lambda_k)\tan^2\theta}{T_{s-1}^2\left(\frac{1+\gamma_k}{1-\gamma_k}\right)}$$

$$\theta = \text{arc cos } \sigma_{\min}$$

$$\gamma_k = (\lambda_k - \lambda_{p+1})/(\lambda_k - \lambda_n) \text{ , and}$$

T_{s-1} is the (s-1)st Chebyshev polynomial of the first kind.

<u>Proof</u>. We will only outline the proof here. For complete details, see [6].

By the Courant-Fischer theorem [7], it can be shown that

$$\lambda_k \le \mu_k \le \lambda_k' \tag{2.11}$$

where

$$\lambda_k' = \max_{\substack{y \in E_k \\ y \ne \theta}} \frac{y^t A y}{y^t y} \tag{2.12}$$

where E_k is <u>any</u> k-dimensional subspace spanned by k independent vectors in L(s,X,A), the space spanned by $X_1, X_2, \dots, X_s$. We will show that there are p vectors $g_1, g_2, \dots, g_p$ in L(s,X,A) such that if E_k is the space spanned by the first k of these vectors, then

$$\lambda_k' \le \lambda_k + \epsilon_k^2 . \tag{2.13}$$

By combining (2.11) and (2.13), we have Theorem 1.

The vectors g_i are chosen as follows: Let P be the polynomial such that

$$P(\lambda) = T_{s-1}(z)$$

where

$$z = 1 - 2(\lambda_{p+1} - \lambda)/(\lambda_{p+1} - \lambda_n) .$$

By the properties of Chebyshev polynomials,

$$P(\lambda_1) \ge P(\lambda_2) \ge \dots \ge P(\lambda_p) \ge 1$$

and

$$|P(\lambda_i)| \le 1$$

for $i = p+1, p+2, \dots , n$. Let Λ be the diagonal matrix of eigenvalues whose ith diagonal element is λ_i . Similarly, let Λ_1 and Λ_2 be diagonal matrices of orders p and $n-p$ respectively, whose ith diagonal elements are λ_i and λ_{p+i} respectively. It follows that

$$AQ = Q\Lambda$$

and

$$P(A)Q = QP(\Lambda)$$

where $P(A)$ and $P(\Lambda)$ are matrices computed by evaluating P at A and Λ. Now, let

$$G = (g_1, g_2, \dots , g_p) \equiv P(A)XW_1^{-1}P(\Lambda_1)^{-1} .$$

It can be easily seen that each column of G is a combination of the columns of $X, AX, A^2X, \dots , A^{s-1}X$. Since $L(s,X,A)$ contains the columns of the matrices $X, AX, \dots , A^{s-1}X$, each g_i is in $L(s,X,A)$. The remainder of the proof involves showing that with E_k chosen to be the spac spanned by $g_1, g_2, \dots , g_k$, the value of λ_k' defined by equation (2.12) satisfies equation (2.13).

This theorem is an extension of a result due to Kaniel [8,9], and, indeed, reduces to Kaniel's theorem when $p=1$. A similar theorem can be given for the largest eigenvalues.

Example. Suppose A is of order 1000 and $\lambda_1 = 0.0$, $\lambda_2 = 0.1$, $\lambda_3 = 0.5$, and $\lambda_{1000} = 1.0$. Suppose further that

X is such that $\sigma_{min} = 0.04$. If we then apply the Block Lanczos method to A and X with p=2 and s=10, we will find that the two least eigenvalues μ_1 and μ_2 of $\bar{M}_{10}$ will satisfy

$$\lambda_1 \le \mu_1 \le \lambda_1 + 2.6_{10}{}^{-8},$$

and

$$\lambda_2 \le \mu_2 \le \lambda_2 + 3.6_{10}{}^{-7}.$$

Thus for the relatively small cost of computing $\bar{M}_{10}$ (of order 20) and its two least eigenvalues, we are able to obtain very accurate approximations to the two least eigenvalues of A.

As is the case with the simultaneous iteration method, there is no need in the Block Lanczos method to explicitly modify the matrix A. Rather, all that is needed is a procedure for multiplying a vector x by A. This is what makes it especially valuable for sparse matrices. In contrast to the simultaneous iteration procedure, however, one can get either the largest <u>or</u> smallest eigenvalues of A automatically with the Block Lanczos method regardless of whether they are of largest modulus or not.

Note that, as we have defined it, the Block Lanczos method is not a method for computing eigenvalues and eigenvectors per se. Rather, it is a procedure for computing a block tridiagonal matrix which is similar to A. To produce a complete algorithm for computing eigenvalues and eigenvectors, we need to combine it with a technique for computing the eigenvalues μ_i and eigenvectors y_i of $\bar{M}_s$. In [6], the QR method is used.

Given an eigenvector y_i of $\bar{M}_s$ corresponding to μ_i, the vector $\bar{q}_i$ will be a good approximation to q_i where

$$\bar{q}_i = \bar{X}_s y_i.$$

It can be shown that $\bar{q}_i$ and μ_i satisfy

$$A\bar{q}_i - \mu_i\bar{q}_i = r_i = \bar{Z}_{s+1}y_i = Z_{s+1}w_i \tag{2.14}$$

where w_i denotes the vector of order p composed of the last p components of y_i. We see from equation (2.14) that the eigenvalues of $\bar{M}_s$ will be exact eigenvalues of A if we have $Z_{s+1}=0$. While we might normally expect that the eigenvalues of $\bar{M}_s$

might not bear any relationship to those of A unless all the elements of Z_{s+1} were very small or zero, it will turn out to be the case that the elements of the vector $Z_{s+1}w_i$ for the least and greatest eigenvalues of eigenvectors of $\overline{M}_s$ will be very small.

In practice, it usually is the case that the number of steps s of the Block Lanczos method that can be carried out must satisfy $p \cdot s \leq q$ where q is a value dictated by the available memory on the computing system and in most instances, q will be much less than n. After carrying out these maximum number of steps, we may find that the eigenvalues μ_i have not converged to the desired accuracy. The result of Theorem 1 suggests that more accurate approximations may be computed by re-applying the Block Lanczos method to A and the matrix X composed of the eigenvector approximations $\overline{q}_i$. In fact, one may apply the procedure repeatedly starting each iteration after the first with a matrix X composed of the eigenvector approximations computed during the previous step. This idea leads us to the following Iterative Block Lanczos Algorithm: Let X be an arbitrary orthonormal n-by-p matrix where $1 \leq p \leq n/2$. Let s be an integer value such that $1 \leq s \leq n/p$. Compute approximations to the p least eigenvalues and eigenvectors of A as follows:

Step 1. Apply the Block Lanczos Algorithm to X computing $\overline{M}_s$ and $\overline{X}_s$.

Step 2. Compute the p least eigenvalues μ_i and eigenvectors y_i of $\overline{M}_s$.

Step 3. Compute $\overline{q}_i = \overline{X}_s y_i$ for $i = 1, 2, \ldots, p$.

Step 4. If $\mu_1, \mu_2, \ldots, \mu_p$ are sufficiently accurate, then stop. Otherwise, return to step 1 with X replaced by the matrix $(\overline{q}_1, \overline{q}_2, \ldots, \overline{q}_p)$.

It may be the case, however, that the number r of eigenvalues and eigenvectors that we need is not equal to p, the block size. Furthermore, once a few eigenvalues and eigenvectors have converged, one need not iterate with them any longer. However, it is necessary to maintain the orthogonality of the matrices $X_1, X_2, \ldots, X_s$ computed during each iteration with respect to eigenvectors already computed. This may be conveniently carried out in the context of step 2b of the Block

Lanczos Algorithm. For a complete description of an Iterative Block Lanczos Algorithm incorporating this concept and allowing the computation of an arbitrary number r of eigenvalues and eigenvectors subject only to memory limitations, see [6].

3. IMPLEMENTATION.

In this section, we will consider certain properties of the Block Lanczos Algorithm and problems associated with its implementation and application.

By the way in which the X_i were defined in Equations (2.1), (2.2), (2.5), and (2.6), it would seem that they should be mutually orthogonal. In practice, however, because of the loss of figures when Z_{i+1} is computed, they rapidly lose orthogonality so that after a few steps of the Block Lanczos process, the current X_i will no longer be orthogonal to the initial blocks in the sequence $X_1, X_2, \ldots, X_i$. It is this phenomenon that makes the Lanczos algorithm unsuitable for the general symmetric eigenproblem. To maintain the stability of the process requires costly reorthogonalization of each X_i with respect to all previous X_j, making the method relatively more costly to apply in relation to, say, the Givens or Householder methods for tridiagonalizing a matrix.

When used as described in the previous section, however, loss of orthogonality is a mixed blessing. As Paige [9] pointed out in his thesis, loss of orthogonality goes hand-in-hand with convergence of some of the eigenvalues of $\bar{M}_s$ to eigenvalues of A. That is, at the very point that the process is about to go awry, we have achieved our desired goal of computing accurate approximations to a few of the eigenvalues of A. The difficulty in using the Lanczos method in this way lies in reliably and efficiently determining at what point orthogonality is being lost. If the process is continued beyond this point, then it in essence restarts and we may recompute eigenvalues that have already been computed. That is, a simple eigenvalue of A may appear more than once among the eigenvalues of $\bar{M}_s$ even though its multiplicity is one. One is then faced with the task of determining whether an eigenvalue of $\bar{M}_s$ which appears more than once is truly a multiple root of A or whether it is a simple root which has been computed more than once. To this date, there is no completely reliable solution to this problem. Kahan and Parlett [3] examined this problem and concluded that not enough is known at the present time to design a universal Lanczos program although the process can often be tailored to a particular problem with considerable success.

In our work we have chosen to include a reorthogonalization step. In particular, we replace step 2b in the description of the Block Lanczos Algorithm with the following modified step:

Step 2b'. Reorthogonalize Z_{i+1} with respect to all previous X_j and compute X_{i+1} and R_{i+1} such that $Z_{i+1} = X_{i+1}R_{i+1}$.

It should be pointed out that this modification preserves the stability of the algorithm at a considerable cost. The reorthogonalization process not only requires a large number of arithmetic operations but requires that each of the X_j be in memory during each step of the method as opposed to only the previous two blocks in the basic process. However, our experiments indicated that even with a reorthogonalization step it was competitive with the simultaneous iteration algorithm, one of the best techniques for solving large, sparse symmetric eigenproblems.

The Iterative Block Lanczos method described in the last section provides a convenient method for estimating the accuracy of the computed eigenvalues and eigenvectors. It can be shown [6] that the ith column of the matrix Z_2 (cf. section 2) computed in each iteration after the first will be the residual vector for the ith eigenvalue μ_i and eigenvector $\bar{q}_i$ computed in the previous iteration. One may then determine at the start of an iteration the accuracy of the eigenvalues and eigenvectors computed at the end of the previous iteration and discontinue iterating with those that have converged.

We end this section with a few suggestions on how to choose the block size p . If possible, it is usually best to choose p equal to the number r of eigenvalues and eigenvectors we are attempting to compute. It seldom pays to iterate with more than r vectors since this will mean that we will be able to carry through fewer steps due to memory limitations. If one is afforded some a priori information on the spectrum of A, Theorem 1 suggests that a good choice for p is one for which the gap between λ_p and λ_{p+1} is fairly large. Also, it is best to choose the block size at least as large as the largest multiplicity possessed by any eigenvalue of A. Note that if A has an eigenvalue of multiplicity m and $m > p$, then we can compute at most p of these eigenvalues at a time.

4. EXAMPLES.

In this section we report on experiments conducted with a Fortran program implementing the iterative Block Lanczos Algorithm described in [6]. The results reported here were computed on an IBM 370/168 computer and are also taken from [6].

Example 1 [6, ch. 4, ex. 1]. The matrix A is a diagonal matrix of order $n=454$ with eigenvalues $\lambda_1 = -10.00$, $\lambda_2 = -9.99$, $\lambda_3 = -9.98$, $\lambda_4 = -9.00$, ... , $\lambda_{454} = 0.0$. We used the program to compute the three least eigenvalues and eigenvectors ($r = 3$) with an initial block size of three ($p = 3$). A total of fifteen vectors were allowed for storing the X_i ($q = 15$ and $p \cdot s \leq 15$) and an accuracy of eps $= 1.0_{10}-8$ was requested. In 11 iterations, the program computed $\mu_1 = -9.9999\ 99999\ 99996$, $\mu_2 = -9.9899\ 99999\ 99994$, and $\mu_3 = -9.9799\ 99999\ 99991$ with relative errors less than $1.0_{10}-8$ in all three cases. Note that the error is the error in the eigenvectors and that the eigenvalues, since they are Rayleigh quotients, are approximately twice as accurate as the eigenvectors.

Example 2 [6, ch. 4, ex. 2]. The matrix in this example is the same as in example 1 except that the spread in the eigenvalues has been reduced by a factor of 10. That is, $\lambda_1 = -10.00$, $\lambda_2 = -9.999$, $\lambda_3 = -9.998$, $\lambda_4 = -9.900$, ... , $\lambda_{454} = -9.000$. As in example 1, the three least eigenvalues were computed to an approximate accuracy of $1.0_{10}-8$ using fifteen working vectors. In this case, the program took 10 iterations to compute the desired eigenvalues to about the same accuracy as in example 1. This example together with example 1 serves to illustrate the point that the rates of convergence of the Block Lanczos Algorithm depend on the relative spread of the eigenvalues as Theorem 1 suggests.

Example 3 [6, ch. 4, ex. 4]. In this example, A is of order 180 with eigenvalues $\lambda_1 = \lambda_2 = 0.0$, $\lambda_3 = \lambda_4 = 0.1$, $\lambda_5 = .25$, ... , and $\lambda_{180} = 2.00$. We computed the four least eigenvalues and eigenvectors to a relative precision of $1.0_{10}-4$ using ten working vectors $(q = 10)$. In this case we tried block sizes of $p = 1, 2, 3$, and 4. The program took the fewest iterations with $p = 2$ and the greatest number with $p = 4$. Even though the rates of convergence were better with the larger block size, the program could carry through more steps of the Block Lanczos Algorithm with $p = 2$. This example demonstrates that the program often has no difficulty computing multiple eigenvalues and that the least eigenvalues need not be the eigenvalues of greatest modulus as is the case with the simultaneous iteration method.

Example 4 [6, ch. 4, ex. 5]. The matrix A is of order 300 with $\lambda_1 = 0.0$, $\lambda_2 = \lambda_3 = \lambda_4 = 0.1$, $\lambda_5 = 0.25$, ... , $\lambda_{300} = 1.0$. Block sizes of $p = 1, 2$, and 3 were tried with $p = 3$ requiring the fewest iterations. In particular, with $p = 3$, the program took four iterations to compute the three least eigenvalues to an approximate precision of $1.0_{10}-3$ using 12 working vectors. Note that with $p = 3$ there is no gap between λ_3 and λ_4 and in spite of this, the algorithm worked very well. Theorem 1 therefore clearly does not tell the whole story regarding the convergence of this method. We suspect that in this example λ_5 plays a role but we have discovered no way of proving this.

Example 5 [6, ch. 4, ex. 7]. In this example, we compute the twelve smallest eigenvalues of the discrete biharmonic operator H [11] of order 1024. Rather than compute the eigenvalues of H directly, we compute those of $A = -H^{-1}$ where the matrix vector product $Ax = y$ required by the Block Lanczos procedure is obtained by solving

$$Hz = x$$

for z using the method of Buzbee, Dorr and Golub [16], and then setting

$$y = -z \ .$$

This serves to illustrate an important point mentioned previously regarding the Block Lanczos method. Namely, it is not necessary to explicitly modify the matrix A. Rather, all that is required is a procedure for computing the matrix-vector product Ax given x.

Using 16 working vectors ($q = 16$) and an initial block size of $p = 3$, the program took 19 iterations to compute the required number of eigenvalues and eigenvectors to precisions which varied from $1.0_{10}{-3}$ to $1.0_{10}{-6}$.

Using the eigenvalues of A, we can compute those of B which are in turn related to the frequencies of vibration of a square, clamped elastic plate. The frequencies computed using our program compare favorably with those reported by Bauer and Reiss [11] and the rough plots of the eigenvectors indicate that they accurately describe the fundamental modes of vibration of clamped plates.

We note, finally, that this program has been successfully applied to the eigenproblem of the Laplace operator over arbitrary bounded plane regions by Proskurowski [12].

A listing of the program used in the above examples is contained in [6] and copies of the program are available from the authors upon request.

5. EXTENSIONS.

Although the emphasis of the paper has been on computing the least eigenvalues, we note that the technique applies equally well to the problem of computing the greatest eigenvalues of A. One could easily construct a program for computing the greatest eigenvalues and eigenvectors directly or apply the ideas in this paper to the matrix $-A$.

Golub, Luk and Overton [13] have developed a program for computing a few of the largest singular values of a matrix using a variant of the Block Lanczos method. The basic idea is to use several vectors to generate a block bidiagonal matrix which has band structure.

The Block Lanczos method can also be used for generating a band matrix with a given eigenvalue structure. In particular, suppose one is given the eigenvalues of the first p+1 leading principal minors of a symmetric matrix. This information is sufficient to compute the first p elements of the eigenvectors of the matrix. From these vectors, it is possible to generate a symmetric matrix which has a half band width p+1 . Details of this algorithm are given in [14].

Finally, it is well known that there is a close theoretical connection between the Conjugate Gradient method and Lanczos' method. Underwood [15] has developed a Block Conjugate Gradient method which extends the Conjugate Gradient method in the same way that the Block Lanczos method extends Lanczos' method. That is, using a program based on this method, he has been able to solve several systems of equations with the same matrix simultaneously with improved rates of convergence over those afforded by the standard Conjugate Gradient method.

REFERENCES

1. C. Lanczos (1950), An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, Journal of Research of the National Bureau of Standards, 45, pp. 255-282.

2. M. R. Hestenes and W. Karush (1951), A method of gradients for the calculation of the characteristic roots and vectors of a real symmetric matrix, Journal of Research of the National Bureau of Standards, 47, pp. 45-61.

3. W. Kahan and B. N. Parlett (1976), How far should you go with the Lanczos process, Sparse Matrix Computations, J. Bunch and D. Rose, eds., Academic Press, New York, pp. 131-144.

4. J. Cullum and W. E. Donath (1974), A block generalization of the symmetric s-step Lanczos algorithm, Report RC 4845 (21570), IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

5. J. Lewis (1977), Algorithms for sparse matrix eigenvalue problems, Ph.D. dissertation, Stanford University, Stanford, California.

6. R. R. Underwood (1975), An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems, Ph.D. dissertation, Stanford University, Stanford, California.

7. J. H. Wilkinson (1965), The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, England.

8. S. Kaniel (1966), Estimates for some computational techniques in linear algebra, Mathematics of Computation, 20, pp. 369-378.

9. C. C. Paige (1971), The computation of eigenvalues and eigenvectors of very large sparse matrices, Ph.D. dissertation, The University of London, London, England.

10. W. Karush (1951), An iterative method for finding characteristic vectors of a symmetric matrix, Pacific Journal of Mathematics, 1, pp. 233-248.

11. L. Bauer and E. L. Reiss (1972), Block five diagonal matrices and the fast numerical solution of the biharmonic equation, Mathematics of Computation, 26, pp. 311-326.

12. W. Proskurowski (1976), On the numerical solution of the eigenvalue problem of the Laplace operator by a capacitance matrix method, Internal report, Lawrence Berkeley Laboratory, University of California, Berkeley, California.

13. G. H. Golub, F. Luk, and M. Overton (1977), A block Lanczos algorithm for computing the largest singular values and associated vectors of a matrix, Computer Science Department Technical Report, to appear, Stanford University, Stanford, California.

14. D. Boley and G. H. Golub (1977), Solving the inverse eigenvalue problem for band matrices, to appear in the Proceedings of the Dundee Conference on Numerical Analysis.

15. R. R. Underwood (1977), A block generalization of the conjugate gradient method, to appear.

16. B. L. Buzbee and F. W. Dorr (1973), The direct solution of the biharmonic equation on rectangular regions and the Poisson equation on irregular regions, Report LA-UR-73-636, Los Alamos Scientific Laboratory, Los Alamos, New Mexico.

The authors were supported in part by the National Science Foundation under Grants GJ35135X, GJ29988X, and MCS75-13497-A01, by the Atomic Energy Commission under Grant AT-(04-3)-326-PA30, and by the Energy Research and Development Administration under Grant E(04-3)-326-PA30.

Computer Science Department
Stanford University
Stanford, California 94305

Nuclear Energy Division
General Electric Company
San Jose, California 95127

30

THE NUMERICALLY STABLE RECONSTRUCTION OF A JACOBI MATRIX FROM SPECTRAL DATA (WITH C. DE BOOR)

Linear Algebra and its Appl. (21), 245–260 (1978).

This was a fun paper to work on. Gene had come to the Mathematics Research Center (MRC) in Madison, WI, in May of 1976 (giving Gene the pleasure to write in the paper's acknowledgements of having been a "guest worker" there) to work with Paul Concus, had recently talked with Victor Barcilon about the inverse eigenvalue problem for 5-diagonal matrices, and must have given a talk about the general problem, with stress on tridiagonal matrices. I became intrigued since Gene mentioned some unsolved problems and I had recently gone through, jointly with Allan Pinkus, Gantmakher and Krein's "Oszillationsmatrizen, Oszillationskerne und kleine Schwingungen mechanischer Systeme", hence felt at home with Jacobi matrices and the power of Sylvester's determinant identity (which, so Allan had taught me, is the only determinant identity one really needs, other than Cauchy–Binet).

My folder concerning this paper is quite thick with reprints and preprints of the relevant literature, most of it provided by Gene, and I recall the pleasure of gaining an understanding of all that material.

Our actual work on the paper was quite short. As soon as Gene raised a question, we usually settled on an answer within the day. My folder contains a message, delivered to Gene at the Howard Johnson Motor Lodge in Madison, which says: "Golub Gene / The persymetric case is trivial/ Carl Deboor" with the word 'very' added in red in Gene's handwriting before 'trivial' – testifying to the lively back-and-forth of those discussions.

The writing of the paper took place a little later. For example, a letter from Gene to me, dated August 9, 1976, announces the imminent arrival of a copy of the "1974 Summer Seminar on Inverse Problems" and the fact that he won't be home until September 24, and ends with the sentence: "Hopefully, I'll find a draft of the manuscript when I return!" I do not now recall whether Gene's hopes were fulfilled. We did receive a referee's report in April of 1977 whose major suggestion we failed to act on since we didn't understand it but the paper was published anyway.

Thank you, Gene!

Carl de Boor
Madison, Wisconsin, USA

The Numerically Stable Reconstruction of a Jacobi Matrix from Spectral Data

C. de Boor*

Mathematics Research Center
University of Wisconsin—Madison
Madison, Wisconsin 53706

and

G. H. Golub†

Computer Science Department
Stanford University
Stanford, California 94305

Submitted by Hans Schneider

ABSTRACT

We show how to construct, from certain spectral data, a discrete inner product for which the associated sequence of monic orthogonal polynomials coincides with the sequence of appropriately normalized characteristic polynomials of the left principal submatrices of the Jacobi matrix. The generation of these orthogonal polynomials via their three-term recurrence relation, as popularized by Forsythe, then provides a stable means of computing the entries of the Jacobi matrix. Our construction provides, incidentally, very simple proofs of known results concerning the existence and uniqueness of a Jacobi matrix satisfying given spectral data and its continuous dependence on those data.

1. INTRODUCTION

Gantmacher and Krein [3] take the term "Jacobi matrix" to mean nothing more than "tridiagonal matrix." But it seems to have become accepted in papers on the problem of concern here to mean by "Jacobi matrix" *a real, symmetric, tridiagonal matrix whose next-to-diagonal elements are positive.* We follow this usage here, and write such a Jacobi matrix

*Sponsored by the United States Army under Contract No. DAAG29-75-C-0024.

†Supported, in part, by MRC under Army Contract No. DAAG29-75-C-0024, and in part by ERDA grant AT(04-3)-326PA #30.

J of order n as

$$J=\begin{bmatrix} a_1 & b_1 & & & & \\ b_1 & a_2 & b_2 & & & \\ & b_2 & a_3 & b_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & b_{n-2} & a_{n-1} & b_{n-1} \\ & & & & b_{n-1} & a_n \end{bmatrix}, \qquad b_i>0, \quad \text{all } i. \tag{1}$$

Further, we denote its left principal submatrix of order r by J_r.

We consider the following inverse problem.

PROBLEM A. *Given the sequences* $\lambda:=(\lambda_i)_1^n$ *and* $\mu:=(\mu_i)_1^{n-1}$ *with*

$$\lambda_i<\mu_i<\lambda_{i+1}, \qquad i=1,\dots,n-1, \tag{S}$$

construct an n th-order Jacobi matrix J which has $\lambda_1,\dots,\lambda_n$ *as its eigenvalues and* $\mu_1,\dots,\mu_{n-1}$ *as the eigenvalues of its left principal submatrix* J_{n-1} *of order* $n-1$.

It is well known that the eigenvalues of J_{n-1} strictly separate those of $J_n=J$, so that the condition (S) is necessary for the existence of a solution. Hochstadt [7] proved that the problem has at most one solution and proved in [8] that this solution (if it exists) depends continuously on λ and μ. L. J. Gray and D. G. Wilson [5] showed it to have at least one solution, as did O. H. Hald [6]. The latter also demonstrated in more explicit detail the continuous dependence of J on λ and μ and described an algorithm for the construction of J which, however, fails to be stable. He also announced an iterative, linearly convergent procedure for the determination of J. A different iterative procedure was developed by Barcilon [1].

By contrast, the algorithm described below in Sec. 4 is direct, i.e., not iterative, and is stable. Its derivation provides simple proofs of the results concerning Problem A just mentioned.

We also consider the following related problems.

PROBLEM B. *Given two strictly increasing sequences* $\lambda:=(\lambda_i)_1^n$ *and* $\lambda^*:=(\lambda_i^*)_1^n$ *with* $\lambda_i<\lambda_i^*<\lambda_{i+1}$, *all i, determine an n th-order Jacobi matrix J which has* $\lambda_1,\dots,\lambda_n$ *as its eigenvalues and for which the matrix* J^*, *obtained from J by changing* a_n *to* a_n^*, *has* $\lambda_1^*,\dots,\lambda_n^*$ *as its eigenvalues.*

PROBLEM C. *Given the strictly increasing sequence* $\lambda:=(\lambda_i)_1^n$, *construct an n th-order persymmetric Jacobi matrix J having* $\lambda_1,\dots,\lambda_n$ *as its eigenvalues.*

Here, a matrix $A=(a_{ij})$ is called *persymmetric* if it is symmetric with respect to its second diagonal, i.e., if $a_{ij}=a_{n+1-j,n+1-i}$, all i and j. The Jacobi matrix (1) is persymmetric iff $a_i=a_{n+1-i}$ and $b_i=b_{n-i}$, all i.

Hochstadt [7] showed Problem C to have at most one solution. Hald [6] showed it to have at least one solution and showed the solution to depend continuously on λ.

In the analysis of these problems, the intimate connection between Jacobi matrices and orthogonal polynomials plays an essential role. We recall the salient facts of this connection in the next section.

2. JACOBI MATRICES AND ORTHOGONAL POLYNOMIALS

We continue to use the notation J_i for the left principal submatrix of order i of the Jacobi matrix (1). Let

$$p_i(t):=\det(t-J_i), \qquad i=1,\dots,n.$$

Then p_i is a monic polynomial of degree i, all i, and one verifies easily that the sequence (p_i) satisfies the three-term recurrence

$$\begin{aligned} p_i(t)&=(t-a_i)\,p_{i-1}(t)-b_{i-1}^2\,p_{i-2}(t), \qquad i=1,\dots,n \\ p_{-1}(t)&:=0, \qquad p_0(t):=1. \end{aligned} \tag{2}$$

Conversely, if we start with a sequence (p_i) of monic polynomials with $\deg p_i=i$, all i, which also satisfies the recurrence (2), then the Jacobi matrix (1) belongs to it in the sense that then $p_i(t)=\det(t-J_i)$ for $i=1,\dots,n$. Since the zeros of p_i are the eigenvalues of J_i, all i, we can therefore phrase Problem A equivalently as follows.

Problem A′. *Given the sequences* $\lambda:=(\lambda_i)_1^n$ *and* $\mu:=(\mu_i)_1^{n-1}$ *with* $\lambda_i<\mu_i<\lambda_{i+1}$, *all* i, *construct sequences* $a:=(a_i)_1^n$ *and* $b:=(b_i)_1^{n-1}$ *so that the sequence* (p_i) *of polynomials given by* (2) *satisfies*

$$p_{n-1}(t)=\prod_{j=1}^{n-1}(t-\mu_j) \quad \textit{and} \quad p_n(t)=\prod_{j=1}^{n}(t-\lambda_j).$$

It is clear that this problem has at most one solution, since we can always run the recurrence (2) *backwards*: If we already know the monic polynomials p_i and p_{i-1} (of degree i and $i-1$, respectively), then a_i is uniquely determined by the requirement that

$$q(t) := p_i(t) - (t - a_i)\, p_{i-1}(t)$$

be a polynomial of degree $i-2$. Further, the number $-b_{i-1}^2$ is then found as the leading coefficient of q, and p_{i-2} is then constructed by dividing q by its leading coefficient.

This construction of (p_i) satisfying (2) from p_{n-1} and p_n goes back to Wendroff [9] and has been used by Hald to solve Problem A or A′ numerically. We, too, tried it in some examples and found it to be badly unstable. But, in trying to understand Hochstadt's procedure for the reconstruction of J from λ and μ [8], it occurred to us that it should be possible to construct a discrete inner product whose corresponding orthogonal polynomials satisfy (2), thus allowing us to generate a and b in the manner advocated by Forsythe [2].

We recall the details. Denote by $\mathbf{P}_k$ the linear space of polynomials of *order* k, i.e., of degree $<k$, with real coefficients, and let $\langle\ ,\ \rangle$ be a symmetric bilinear form which is an inner product on $\mathbf{P}_n$. Then there exists exactly one sequence $(q_i)_0^n$ of monic polynomials, with q_i of degree i, all i, which is orthogonal with respect to the inner product $\langle\ ,\ \rangle$, i.e., for which

$$\langle q_i, q_j \rangle = 0 \qquad \text{for} \quad i \neq j.$$

One may determine q_i as the error in the best approximation from $\mathbf{P}_i$ to the function $f(t) := t^i$, with respect to the norm

$$\|f\| := \langle f, f \rangle^{1/2}$$

in $\mathbf{P}_n$. Alternatively, one may construct $(q_i)_0^n$ by its three-term recurrence, an idea popularized specifically for the case of a discrete inner product by Forsythe [2]: One computes

$$\begin{aligned} q_{-1}(t) &= 0, \qquad q_0(t) = 1, \\ q_i(t) &= (t - \alpha_i) q_{i-1}(t) - \beta_{i-1}^2 q_{i-2}(t), \qquad i = 1, \ldots, n, \end{aligned} \tag{3}$$

with the numbers α_i and β_i computed concurrently by

$$\alpha_i := \langle t q_{i-1}, q_{i-1} \rangle / \langle q_{i-1}, q_{i-1} \rangle, \qquad i = 1, \ldots, n, \tag{4a}$$

$$\beta_i := \|q_i\| / \|q_{i-1}\|, \qquad i = 1, \ldots, n-1. \tag{4b}$$

Here, it is assumed that $\langle tf(t), g(t)\rangle = \langle f(t), tg(t)\rangle$.

The computational process (3)–(4) for the vectors α and β is very stable. We will, therefore, have solved Problem A in a satisfactory manner provided we can construct a suitable inner product for which $q_i = p_i$ for $i = n-1$ and $i = n$. This we now do.

From a computational point of view, the simplest bilinear form $\langle\ ,\ \rangle$ which is an inner product in $\mathbf{P}_n$ is of the form

$$\langle f,g\rangle := \sum_{i=1}^{n} f(\xi_i)\, g(\xi_i)\, w_i, \qquad \text{all } f,g, \tag{5}$$

with $\xi_1 < \cdots < \xi_n$, and $w_i > 0$, $i = 1,\dots,n$.

LEMMA 1. *Let* $(q_i)_0^n$ *be the sequence of monic orthogonal polynomials for the inner product* (5). *Then*

$$\prod_{j=1}^{n} (t-\xi_j) = q_n(t) \tag{6}$$

and

$$w_i = \frac{\gamma}{q_{n-1}(\xi_i)\, q_n'(\xi_i)}, \qquad i = 1,\dots,n, \qquad \textit{with} \quad \gamma := \frac{\|q_{n-1}\|^2}{\displaystyle\sum_{j=1}^{n} \frac{q_{n-1}(\xi_j)}{q_n'(\xi_j)}}. \tag{7}$$

Consequently, we can recover (5) *from* q_{n-1} *and* q_n.

Proof. The polynomial $q(t) := \prod_{j=1}^{n}(t-\xi_j)$ is a monic polynomial of degree n which is orthogonal with respect to the inner product (5) to all functions, and hence must agree with q_n. This proves (6). As to (7), we know that q_{n-1} is orthogonal to $\mathbf{P}_{n-1}$. This means that the linear functional L given by the rule

$$Lf := \langle f, q_{n-1}\rangle = \sum_{i=1}^{n} f(\xi_i)\, q_{n-1}(\xi_i)\, w_i, \qquad \text{all } f,$$

vanishes on $\mathbf{P}_{n-1}$. Since any $n-1$ distinct point evaluations are linearly independent on $\mathbf{P}_{n-1}$, this implies that

$$q_{n-1}(\xi_i) \neq 0, \qquad \text{all } i.$$

For the same reason, there is, up to multiplication by a scalar, exactly one linear functional M of the form $Mf = \Sigma_1^n f(\xi_i) m_i$, all f, which vanishes on $\mathbf{P}_{n-1}$. Since both L and the $(n-1)$st divided difference $[\xi_1, \ldots, \xi_n]$ on the points $\xi_1, \ldots, \xi_n$ are such linear functionals, it follows that, for an appropriate scalar γ,

$$Lf = \gamma[\xi_1, \ldots, \xi_n] f = \gamma \sum_{i=1}^{n} \frac{f(\xi_i)}{\prod\limits_{\substack{j=1 \\ j \neq i}}^{n} (\xi_i - \xi_j)}, \qquad \text{all } f.$$

But this states, with (6), that

$$q_{n-1}(\xi_i) w_i = \gamma / q_n'(\xi_i), \qquad i = 1, \ldots, n,$$

and, in particular,

$$\|q_{n-1}\|^2 = \gamma \sum_{i=1}^{n} \frac{q_{n-1}(\xi_i)}{q_n'(\xi_i)},$$

thus proving (7). ■

One may view Lemma 1 as giving a way to construct the computationally simplest inner product with respect to which a given sequence $(p_i)_0^n$ of monic polynomials satisfying a three-term recurrence (2) is orthogonal.

3. A SOLUTION OF PROBLEMS A, B, C

Lemma 1 shows how to reconstruct the discrete inner product (5) from its last two orthogonal polynomials. It also shows the well-known facts that q_n has n real zeros, all simple, and that the $n-1$ zeros of q_{n-1} strictly separate those of q_n. Indeed, the positivity of the w_i's demands by (7) that $q_{n-1}(\xi_i) q_n'(\xi_i) \operatorname{signum} \gamma > 0$, all i, while, clearly, $(-)^{n-i} q_n'(\xi_i) \geqslant 0$, all i, and therefore

$$q_{n-1}(\xi_i)q_{n-1}(\xi_{i+1})<0, \qquad i=1,\dots,n-1,$$

showing q_{n-1} to have a simple zero between any two zeros of q_n.

Conversely, suppose we compute w by

$$w_i := \frac{1}{p_{n-1}(\xi_i)\,p_n'(\xi_i)}, \qquad i=1,\dots,n, \tag{8a}$$

where

$$\xi_i := \lambda_i, \qquad i=1,\dots,n, \tag{8b}$$

$$p_{n-1}(t) := \prod_{j=1}^{n-1}(t-\mu_j), \tag{8c}$$

$$p_n(t) := \prod_{j=1}^{n}(t-\lambda_j), \tag{8d}$$

with $\lambda_i<\mu_i<\lambda_{i+1}$, all i. Then $w_i>0$, all i; hence (5) is then an inner product on $\mathbf{P}_n$, and necessarily $p_n=q_n$, by (6), and $p_{n-1}=q_{n-1}$ since $p_{n-1}(\lambda_i)=q_{n-1}(\lambda_i)$, $i=1,\dots,n$, by (7), and both polynomials are of degree $<n$.

This proves that *Problem A has exactly one solution for given* λ *and* μ *satisfying* (S). Further, since $a=\alpha$ and $b=\beta$ as determined by (3)–(4) depend continuously on ξ and w, while the latter, as determined by (8), depend continuously on λ and μ, it follows that *J depends continuously on* λ *and* μ.

Problem B is closely related to Problem A. In terms of the monic polynomials

$$p_i(t)=\det(t-J_i), \qquad i=1,\dots,n,$$

we are given the information that

$$\prod_{j=1}^{n}(t-\lambda_j)=p_n(t)=(t-a_n)\,p_{n-1}(t)-b_{n-1}^2\,p_{n-2}(t)$$

and that

$$\prod_{j=1}^{n}(t-\lambda_j^*)=p_n^*(t)=(t-a_n^*)p_{n-1}(t)-b_{n-1}^2p_{n-2}(t).$$

We conclude that

$$\prod_{j=1}^{n}(t-\lambda_j^*)-\prod_{j=1}^{n}(t-\lambda_j)=(a_n-a_n^*)p_{n-1}(t),$$

and therefore, comparing coefficients (or else comparing the trace of J with that of J^*),

$$\sum_{j=1}^{n}(\lambda_j-\lambda_j^*)=a_n-a_n^*.$$

This allows calculation of a_n^* once we know a_n. Further, since we only need to know the weights w for the inner product (5) up to a scalar multiple in order to reconstruct a and b via (3)–(4), it follows that we get J (and uniquely so) by choosing

$$\xi_i:=\lambda_i,\qquad i=1,\ldots,n,$$

$$w_i:=\frac{-1}{p_n'(\lambda_i)\prod_{j=1}^{n}(\lambda_i-\lambda_j^*)}. \tag{9}$$

Note how the assumption $\lambda_i<\lambda_i^*<\lambda_{i+1}$, all i, insures that $w_i>0$, all i.

The solution of Problem C leads to an intriguing fact which is also of help in the final algorithm for the solution of these problems. We came upon this fact accidentally. We had somehow gained the impression in reading Hochstadt's paper [8] that the correct weights in Lemma 1 would probably be

$$w_i=q_{n-1}(\xi_i)/q_n'(\xi_i),\qquad i=1,\ldots,n, \tag{10}$$

and a quick numerical experiment confirmed this guess. Yet, when it came to proving it, we could only prove that $w_i=1/[q_{n-1}(\xi_i)q_n'(\xi_i)]$, all i. This seeming contradiction is resolved by consideration of the characteristic polynomials of the *right* principal submatrices of J.

Let S be the permutation matrix carrying $(1,2,\ldots,n)$ into $(n,n-1,\ldots,1)$, i.e.,

$$S:=\left(\delta_{i+j,n+1}\right)_{i,j=1}^{n}=S^{-1},$$

and denote by $\bar{J}$ the reflection of J across its second diagonal,

$$\bar{J}:=S^{-1}JS=:\begin{bmatrix} \bar{a}_1 & \bar{b}_1 & & & \\ \bar{b}_1 & \bar{a}_2 & \bar{b}_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \bar{b}_{n-2} & \bar{a}_{n-1} & \bar{b}_{n-1} \\ & & & \bar{b}_{n-1} & \bar{a}_n \end{bmatrix}$$

with $\bar{a}_i=a_{n+1-i}$, $\bar{b}_i=b_{n-i}$, all i. Correspondingly, let

$$\bar{p}_{-1}(t):=0,\quad \bar{p}_0(t):=1,\quad \bar{p}_i(t):=\det\left(t-\bar{J}_i\right),\qquad i=1,\ldots,n.$$

Lemma 2. *For $i=1,\ldots,n$, $p_{n-1}(\lambda_i)\,\bar{p}_{n-1}(\lambda_i)=(b_1\cdot\cdots\cdot b_{n-1})^2$.*

Proof. For each i, $p_{n-1}(\lambda_i)\,\bar{p}_{n-1}(\lambda_i)$ is the product of the $(n-1)$st-order left principal minor with the $(n-1)$st-order right principal minor of the singular matrix

$$A:=\lambda_i-J,$$

i.e.,

$$p_{n-1}(\lambda_i)=\det A\begin{pmatrix}1,\ldots,n-1\\1,\ldots,n-1\end{pmatrix}\quad\text{and}\quad \bar{p}_{n-1}(\lambda_i)=\det A\begin{pmatrix}2,\ldots,n\\2,\ldots,n\end{pmatrix},$$

and $\det A=0$. Apply Sylvester's identity (see, e.g., [3, p. 15]), using

$A\begin{pmatrix}2,\ldots,n-1\\2,\ldots,n-1\end{pmatrix}$ as pivotal block, to get

$$0=\det A\begin{pmatrix}2,\ldots,n-1\\2,\ldots,n-1\end{pmatrix}\det A=\det\begin{bmatrix}\det A\begin{pmatrix}1,\ldots,n-1\\1,\ldots,n-1\end{pmatrix} & \det A\begin{pmatrix}1,\ldots,n-1\\2,\ldots,\ n\end{pmatrix}\\ \det A\begin{pmatrix}2,\ldots,\ n\\1,\ldots,n-1\end{pmatrix} & \det A\begin{pmatrix}2,\ldots,n\\2,\ldots,n\end{pmatrix}\end{bmatrix}$$

$$=p_{n-1}(\lambda_i)\,\bar{p}_{n-1}(\lambda_i)-\left[(-)^{n-1}b_1\cdot\cdots\cdot b_{n-1}\right]^2. \qquad \blacksquare$$

Here, we have used the abbreviation

$$A\begin{pmatrix}i_1,\ldots,i_r\\j_1,\ldots,j_s\end{pmatrix}:=(a_{i_\rho j_\sigma})_{\rho=1,\ \sigma=1}^{r\qquad s}.$$

If now J is persymmetric, then $J=\bar{J}$ and so $p_i=\bar{p}_i$, all i. The lemma then implies that $[p_{n-1}(\lambda_i)]^2=(b_1\cdot\cdots\cdot b_{n-1})^2$, all i. Since we only need to know the weight vector w up to a scalar multiple, it follows that we only need to know p_n in order to reconstruct a persymmetric J, thus solving Problem C.

We conclude further that the computations (3)–(4) always generate the diagonals α and β of a persymmetric Jacobi matrix if we use the discrete inner product

$$\langle f,g\rangle:=\sum_{i=1}^{n}\frac{f(\xi_i)\,g(\xi_i)}{\prod\limits_{\substack{j=1\\j\neq i}}^{n}|\xi_i-\xi_j|}.$$

We were interested in Lemma 2 because of its importance for the algorithm in the next section and have therefore not followed the more customary treatment of Problem C. This treatment goes back to Gantmacher and Krein and consists in using the persymmetry of J to construct an equivalent problem of the form B and of half the size, thus reducing it to a problem with a known solution.

4. AN ALGORITHM

Lemma 2 shows that we could also determine the correct weights w for the generation of a and b via (4) by

$$w_i=\bar{p}_{n-1}(\lambda_i)/p_n'(\lambda_i),\qquad i=1,\dots,n.$$

To say it differently, *if the inner product* (5) *is given by*

$$\begin{aligned}\xi_i&:=\lambda_i, & i&=1,\dots,n,\\ w_i&:=p_{n-1}(\lambda_i)/p_n'(\lambda_i), & i&=1,\dots,n,\end{aligned}\tag{11}$$

then the quantities generated by (3)–(4) *are* $q_i=\bar{p}_i$, $\alpha_i=\bar{a}_i$, $\beta_i=\bar{b}_i$, *all* i. This says that use of the weights (11) rather than the weights (8a) in the computations (3)–(4) also generates the nonzero entries of J, but *in reverse order*. This explains the success in our numerical experiments using the weights (10) referred to earlier: all examples happened to have been persymmetric.

Use of the weights (11) in preference to (8a) has some computational advantages. Because of the interlacing conditions (S), we get the bounds

$$\frac{\lambda_i-\mu_{i-1}}{\lambda_i-\lambda_1}\,\frac{\mu_i-\lambda_i}{\lambda_n-\lambda_i}<\frac{p_{n-1}(\lambda_i)}{p_n'(\lambda_i)}<1,\tag{12}$$

where the first (last) factor in the lower bound is to be omitted in case $i=1$ ($i=n$). This shows that overflow or underflow is highly unlikely to occur in the calculation of the weights (11). By contrast, the computation of the numbers $1/[p_{n-1}(\lambda_i)\,p_n'(\lambda_i)]$ has to be carefully monitored, in general, for the occurrence of overflow or underflow, else one has to compute the logarithms of these numbers, a somewhat more expensive procedure.

We offer the following algorithm for the solution of Problem A, and recall that Problems B and C can also be solved by it, if the definition of $p_{n-1}(\lambda_i):=\prod_1^{n-1}(\lambda_i-\mu_j)$ used here is modified appropriately.

ALGORITHM. *Given the n eigenvalues* $\lambda_1<\cdots<\lambda_n$ *of the Jacobi matrix* (1) *and the* $n-1$ *eigenvalues* $\mu_1<\cdots<\mu_{n-1}$ *of its left principal minor of order* $n-1$. *Note that, necessarily,* $\lambda_i<\mu_i<\lambda_{i+1}$, *all* i.

```
1. Compute the weights $w$ from $\lambda$ and $\mu$:
   1.1 $\text{temp}(i-1):=\lambda_i,\ i=2,\ldots,n$
   1.2 for $i=1,\ldots,n$, do:
      1.21 $w_i:=\Pi_{j=1}^{n-1}(\lambda_i-\mu_j)/[\lambda_i\text{-temp}(j)]$
      1.22 $\text{temp}(i):=\lambda_i$
2. Generate the values at $\lambda$ of the first two orthogonal polynomials:
   2.1 $s:=\Sigma_{j=1}^{n}w_j=\langle\bar{p}_0,\bar{p}_0\rangle$
   2.2 $\bar{a}_1:=(\Sigma_{j=1}^{n}w_j\lambda_j)/s=\langle\bar{p}_0,\bar{p}_1\rangle/s$
   2.3 for $i=1,\ldots,n$, do:
      2.31 $\text{pkml}(i):=1=\bar{p}_0(\lambda_i)$
      2.32 $\text{pk}(i):=\lambda_i-\bar{a}_1=\bar{p}_1(\lambda_i)$
3. Compute $\|\bar{p}_k\|^2$ and $\bar{a}_k$, $\bar{b}_{k-1}^2$, then use them to generate the values at $\lambda$
   of $\bar{p}_{k+1}$ from those of $\bar{p}_k$ and $\bar{p}_{k-1}$ by the three-term recurrence:
   3.1 for $k=2,\ldots,n$, do:
      3.11 $s':=s=\|\bar{p}_{k-1}\|^2$
      3.12 $s:=t:=0$
      3.13 for $i=1,\ldots,n$, do:
         3.131 $p:=w_i\times\text{pk}(i)^2$
         3.132 $s:=s+p$
         3.133 $t:=t+\lambda_i p$
      3.14 $\bar{b}_{k-1}^2:=s/s'$
      3.15 $\bar{a}_k:=t/s$
      3.16 for $i=1,\ldots,n$, do:
         3.161 $p:=\text{pk}(i)$
         3.162 $\text{pk}(i):=(\lambda_i-\bar{a}_k)p-\bar{b}_{k-1}^2\text{pkml}(i)$
         3.163 $\text{pkml}(i):=p$
4. Compute $\bar{b}_k$ from $\bar{b}_k^2$. Also, if $a$ and $b$, rather than the vectors $\bar{a}$ and
   $\bar{b}$, are wanted, this is the place to reorder them.
   4.1 $\bar{b}_k:=\text{sqrt}(\bar{b}_k^2),\ k=1,\ldots,n-1$
```

Output consists of the vectors $\bar{a}$ and $\bar{b}$, with $\bar{a}_i=a_{n+1-i}$, $\bar{b}_i=b_{n-i}$, all i, and a and b the diagonals of (1).

We have carried out various numerical experiments with this algorithm and describe here only three.

For the nth-order Jacobi matrix J_n with general row

$$1,\quad -2,\quad 1, \tag{i}$$

the eigenvalues are given explicitly (and in order) by the formula

$$\lambda_j = 2\left(\cos\frac{j\pi}{n+1} - 1\right), \qquad j = 1,\dots,n.$$

Starting with these values and the corresponding sequence $(\mu_j)_1^{n-1}$ for J_{n-1}, the algorithm produced approximations to the diagonal and the off-diagonal entries of J_n whose maximum and average error are recorded for $n = 25$, 50, 100, and 200 in the first columns of the Table 1. All calculations were carried out on a UNIVAC 1110 in single precision (27-bit floating-point mantissa).

For variety, we also consider the n-th order Jacobi matrix J_n with general row

$$1-\frac{i}{n}, \quad \frac{i}{n}-2, \quad 1-\frac{i+1}{n}, \qquad i = 1,\dots,n. \tag{ii}$$

We know of no simple formula for its eigenvalues, and therefore used the algorithm tql 1 on pp. 232–233 of Wilkinson and Reinsch's handbook [10] to compute them and those of J_{n-1}. The tolerance (relative error requirement) for tql 1 we chose as $1.-7$. With this spectral information, we entered the above algorithm and so reconstructed J_n approximately. Errors of this reconstruction are also given in Table 1, in the last four columns. There is a significant deterioration as n increases.

As can be expected from the formula (11) for the weights (w_i), the condition of the problem of determining J_n from (λ_i) and (μ_i) deteriorates as one or more μ_i approach the corresponding λ_i, since then one or more of the weights approach zero. This is shown even more strikingly when the matrix of the last example is reflected across its second diagonal, i.e., when the

TABLE 1

MAXIMUM AND AVERAGE ERROR IN THE DIAGONAL AND OFF-DIAGONAL ENTRIES OF TWO SPECIFIC JACOBI MATRICES[a]

	(i)				(ii)			
	Diagonal		Off-Diagonal		Diagonal		Off-Diagonal	
n	Max.	Ave.	Max.	Ave.	Max.	Ave.	Max.	Ave.
25	4.−7	2.−7	2.−7	6.−8	2.−6	5.−7	1.−6	4.−7
50	9.−7	1.−7	4.−7	2.−7	6.−4	1.−5	5.−5	2.−6
100	2.−6	7.−7	8.−7	2.−7	7.−1	3.−2	3.−1	1.−2
200	3.−6	9.−7	1.−6	3.−7	8.−1	2.−2	5.−1	1.−2

[a] As reconstructed with the algorithm of this section from (approximate) spectral data.

Jacobi matrix with the general row

$$1-(n+1-i)/n, \quad (n+1-i)/n-2, \quad 1-(n-i)/n, \qquad i=1,\dots,n,$$

is considered. Now the reconstruction breaks down in single precision already for $n=30$, since $\mu_1-\lambda_1$ becomes too small. Even for $n=20$, we have $\mu_1-\lambda_1 \sim 2.-7$. In fact, in computations using tql 1 to obtain the spectral information, some weights become negative for $n=30$, while for $n=10$ and 20 we obtain approximations with errors of the order of $1.-4$ and $2.-2$, respectively.

5. THE CONNECTION WITH GAUSS QUADRATURE

For the given Jacobi matrix J in (1), let $(P_i)_0^n$ be the polynomial sequence generated by the recursion

$$\begin{gathered} b_{j+1}P_{j+1}(t)=(t-a_{j+1})P_j(t)-b_jP_{j-1}(t), \qquad j=0,\dots,n-1 \\ P_{-1}(t):=0, \qquad P_0(t):=1, \end{gathered} \tag{13}$$

with b_0 arbitrary, $b_n \neq 0$. The sequence (P_i) is related to the sequence (p_i) with $p_i(t):=\det(t-J_i)$, all i, of monic polynomials by

$$p_i=(b_1\cdot\dots\cdot b_i)P_i, \tag{14}$$

as one verifies easily, e.g., by comparing (13) and (2).

Let now ω be a monotone function on some interval $[A,B]$ so that $(P_i)_0^n$ is *orthonormal* with respect to the inner product

$$\langle f,g\rangle_\omega := \int_A^B f(t)\,g(t)\omega(dt).$$

(Lemma 1 provides a simple proof of the existence of such ω.) Then the zeros $\lambda_1<\cdots<\lambda_n$ of P_n must lie in $[A,B]$, and there exist positive weights $w_1,\dots,w_n$ such that for every $f\in C^{2n}[A,B]$,

$$\int_A^B f(t)\omega(dt)=\sum_{j=1}^n w_jf(\lambda_j)+\frac{b_1\cdots b_nf^{(2n)}(\xi)}{(2n)!} \qquad \text{for some } \xi\in\,]A,B[. \tag{15}$$

If we take this fact for granted, then it follows that

$$\delta_{i-j}=\langle P_i,P_j\rangle_\omega=\sum_{r=1}^{n} w_r P_i(\lambda_r)P_j(\lambda_r) \qquad \text{for} \quad i+j<2n,$$

showing that

$$\langle f,g\rangle:=\sum_{j=1}^{n} f(\lambda_i)\,g(\lambda_i)\,w_i \tag{5}$$

is an inner product on $\mathbf{P}_n$ with respect to which $(P_i)_0^{n-1}$ is orthonormal, and hence for which $(p_i)_0^n$ is an orthogonal sequence.

This shows that the construction of the weights (w_i) for (5), which was crucial for our numerical solution of the various inverse eigenvalue problems, can be started from any convenient formula for the weights in a Gaussian quadrature formula.

For instance, one could start with the following consequence of the Christoffel-Darboux formula:

$$w_j\mathbf{P}^T(\lambda_j)\mathbf{P}(\lambda_j)=1 \qquad \text{for} \quad j=1,\dots,n. \tag{16}$$

Here, $\mathbf{P}(\lambda)$ denotes the n-vector $\left(P_0(\lambda),\dots,P_{n-1}(\lambda)\right)$. By (13), $\mathbf{P}(\lambda_j)$ is an eigenvector for J belonging to the eigenvalue λ_j. Therefore, with $\mathbf{u}_j:=(u_{1j},\dots,u_{nj})$ a unit eigenvector of J for λ_j, (16) implies that

$$w_j=u_{ij}^2/P_{i-1}^2(\lambda_j), \qquad i=1,\dots,n. \tag{17}$$

Since $P_0(\lambda)=1$, we obtain in this way the formula

$$w_j=u_{1j}^2, \qquad j=1,\dots,n, \tag{18}$$

used by Golub and Welsch [4] to compute the weights. Problems A, B, and C can now be solved by deriving from the given data information about the eigenvectors of J.

A more direct approach might be to start with the well-known formula

$$w_j=-\frac{k_{n+1}}{k_n}\frac{1}{P_{n+1}(\lambda_j)P_n'(\lambda_j)}, \qquad j=1,\dots,n, \tag{19}$$

with k_j the leading coefficient of P_j, i.e., $k_j=1/(b_1\cdot\cdots\cdot b_j)$. This formula involves the "next" orthogonal polynomial P_{n+1}. But, since $P_n(\lambda_j)=0$ for all j,

we have

$$P_{n+1}(\lambda_j)/k_{n+1}=p_{n+1}(\lambda_j)=-b_n^2p_{n-1}(\lambda_j)=-b_n^2P_{n-1}(\lambda_j)/k_{n-1}$$

by the three-term recurrence; therefore we also have

$$w_j=\frac{1}{b_nP_{n-1}(\lambda_j)P_n'(\lambda_j)}, \qquad j=1,\dots,n, \tag{19'}$$

which shows how (7) could have been derived from standard results in Gauss quadrature.

The authors are very pleased to acknowledge stimulating discussions with Professor Victor Barcilon of the University of Chicago.

The authors wish to thank Mr. Daniel Boley of Stanford University for corroborating some of their calculations.

The work of G. H. G. was begun when he was a guest worker at the Mathematics Research Center, whose hospitality he greatly enjoyed.

REFERENCES

1 V. Barcilon, Iterative solutions of inverse eigenvalue problems for Jacobi matrices, unpublished manuscript.

2 George E. Forsythe, Generation and use of orthogonal polynomials for data-fitting with a digital computer, *J. SIAM* **5** (1957) 74–88.

3 F. R. Gantmacher and M. G. Krein, *Oszillationsmatrizen, Oszillationskerne und kleine Schwingungen mechanischer Systeme,* Akademie-Verlag, Berlin, 1960.

4 G. H. Golub and J. H. Welsch, Calculation of Gauss quadrature rules, *Math. Comp.* **23** (1969) 221–230.

5 L. J. Gray and D. G. Wilson, Construction of a Jacobi matrix from spectral data, *Linear Algebra Appl.* **14** (1976) 131–134.

6 Ole H. Hald, Inverse eigenvalue problems for Jacobi matrices, *Linear Algebra Appl.* **14** (1976) 63–85.

7 Harry Hochstadt, On some inverse problems in matrix theory, *Arch. Math.* **18** (1967) 201–207.

8 Harry Hochstadt, On the construction of a Jacobi matrix from spectral data, *Linear Algebra Appl.* **8** (1974) 435–446.

9 Burton Wendroff, On orthogonal polynomials, *Proc. Amer. Math. Soc.* **12** (1961) 554–555.

10 J. H. Wilkinson and C. Reinsch, *Linear Algebra, Handbook for Automatic Computation II,* Springer, Berlin, 1971.

Received 25 February 1977; revised 12 April 1977

INDEX