FLEXIBLE ROBOT DYNAMICS AND CONTROLS

International Federation for Systems Research
International Series on Systems Science and Engineering

IFSR was established "to stimulate all activities associated with the scientific study of systems and to coordinate such activities at international level." The aim of this series is to stimulate publication of high-quality monographs and textbooks on various topics of systems science and engineering. This series complements the Federation's other publications.

Volumes 1–6 were published by Pergamon Press.

FLEXIBLE ROBOT DYNAMICS AND CONTROLS

Rush D. Robinett, III
Clark R. Dohrmann
G. Richard Eisler
John T. Feddema

Sandia National Laboratories
New Mexico

Gordon G. Parker

Michigan Technological University
Houghton, Michigan

David G. Wilson

WAYA Research, Inc.
Albuquerque, New Mexico

Dennis Stokes

S. Enterprises
Tacoma, Washington

Springer Science+Business Media, LLC

ISBN 978-1-4613-5122-1 ISBN 978-1-4615-0539-6 (eBook)
DOI 10.1007/978-1-4615-0539-6

Originally published by Kluwer Academic/Plenum Publishers in 2002
Softcover reprint of the hardcover 1st edition 2002
http://www.wkap.nl/

10 9 8 7 6 5 4 3 2 1

A C.I.P. record for this book is available from the Library of Congress

Preface

This book is the result of over ten (10) years of research and development in flexible robots and structures at Sandia National Laboratories. The authors decided to collect this wealth of knowledge into a set of viewgraphs in order to teach a graduate class in Flexible Robot Dynamics and Controls within the Mechanical Engineering Department at the University of New Mexico (UNM). These viewgraphs, encouragement from several students, and many late nights have produced a book that should provide an upper-level undergraduate and graduate textbook and a reference for experienced professionals.

The content of this book spans several disciplines including structural dynamics, system identification, optimization, and linear, digital, and nonlinear control theory which are developed from several points of view including electrical, mechanical, and aerospace engineering as well as engineering mechanics. As a result, the authors believe that this book demonstrates the value of solid applied theory when developing hardware solutions to real world problems. The reader will find many real world applications in this book and will be shown the applicability of these techniques beyond flexible structures which, in turn, shows the value of multidisciplinary education and teaming.

The authors have placed a strong emphasis on real world applications and *hands-on* hardware solutions. As a result, this book steps through a systematic procedure for applying theory to practice while leaving the rigorous theorem development to a rich collection of references. Also, the authors realize that most educational institutions do not have ready access to hardware platforms, so the authors have chosen an analytical example problem to be utilized throughout the book. By analytical, one means the example problem model is solvable in closed form and it has been verified against experimental results. The analytical example problem is the overhead gantry crane robot that supports a spherical pendulum payload. Gantry robots are found throughout the robot user community and provide a straight forward, real world application.

The authors want to thank Dr. Ray Harrigan for his continued support and encouragement of this research work from the beginning in the mid 1980's to the present. This book would not have been possible without Ray's commitment. The authors appreciate the initial theory and hardware development work performed by Professor Gregory P. Starr of UNM and Ben Petterson as well as the software

development work performed by Jill Fahrenholtz and Charlene Lennox. The authors wish to thank Dr. Keith Miller for teaming with Ray to fund the first flexible robot LDRD project, and Dr. David Martinez for his continuing support of adaptive and smart structures. The authors wish to acknowledge the theoretical development of the quadratic modes by Dr. Dan Segalman.

A homework solution manual that contains solutions to the homework problems as well as additional details and explanations will be marketed by WAYA Research, Inc. (ahwaya@aol.com,) and S. Enterprises (stokenterprises@aol.com). Both of these companies can provide experimental hardware and software, and design and outfit a college, university, or scientific institution (a flexible robot or control system laboratory) based on the material presented in this book. Additionally, both of these companies are available for course development consultation.

This book was typeset with the LaTeX 2_ε and edited in Tacoma, Washington by Dennis Stokes of S. Enterprises, P.O. Box 42251, Tacoma, WA 98442, stokenterprises@aol.com.

Book Nomenclature

The nomenclature used for scalars, vectors, and matrices throughout the book are defined as

- Scalars are represented as non-bold letters, e.g., A, a, α, and Γ.
- Vectors are represented as bold lowercase letters, e.g., $\mathbf{x}, \mathbf{y}$, and $\boldsymbol{\beta}$. Necessary exceptions are explained. Vectors with subscripts are non-bold, e.g., x_i, and y_i.
- Matrices are 2-D arrays of scalars and are represented as boldface uppercase letters, e.g., $\mathbf{A}, \mathbf{W}$, and $\mathbf{\Gamma}$. Necessary exceptions are explained. Matrices with subscripts are non-bold, e.g., A_{ij}.

A capital superscript T means a transpose of a vector or matrix, e.g., $(\mathbf{A}^T)$, and a superscript negative one means the inverse of a matrix, e.g., $(\mathbf{A}^{-1})$. A dot or dots above a variable define differentiation with respect to time, e.g., $\dot{\omega}$, and $\ddot{\omega}$. A prime or primes on a variable represent differentiation with respect to a spatial variable, e.g., $y'(x) = dy(x)/dx$. Necessary exceptions are explained. A hat or tilde above a variable, e.g., $\widehat{x}, \tilde{y}$, define a variable estimation or approximation. Necessary exceptions are explained. For example, a hat above a variable also represents a unit vector, e.g., $\hat{\mathbf{i}}, \hat{\mathbf{j}}$. A bar above a variable represents a mean or average, e.g., $\bar{x}$. Necessary exceptions are explained. The mathematical symbols used include $\exists$ (there exist), $\equiv$ (is identically equal to, defined as), $\Re$ (real set), $\forall$ (for all), $\in$ (belongs to), Δ (finite difference, or increment), iff (if and only if), $\| \ \|$ (vector magnitude), $| \ |$ (absolute value), and the end of proof symbol as $\square$. All other special variables and symbols are defined and explained as needed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Sandia National Laboratories

Sandia National Laboratories (Sandia) began in 1945 on Sandia Base in Albuquerque, New Mexico, as Z Division, part of what's now Los Alamos National Lab (LANL). Both laboratories were born out of America's World War II atomic bomb development effort—the Manhattan Project. Sandia came into being as an ordinance design, testing, and assembly facility, and was located on Sandia Base to be close to an airfield and work closely with the military.

Sandia is a national security laboratory operated for the U.S. Department of Energy (DOE) by the Sandia Corporation, a Lockheed Martin Company. Sandia designs all non-nuclear components for the nation's nuclear weapons, perform a wide variety of energy research and development projects, and work on assignments that respond to national security threats—both military and economic. Sandia encourages and seeks partnerships with appropriate U.S. industry and government groups to collaborate on emerging technologies that support Sandia's mission.

The laboratories original mission—providing engineering design for all non-nuclear components of the nation's nuclear weapons—continues today, but Sandia now also performs a wide variety of national security R&D work.

Sandia's broadly stated mission today:

> As a Department of Energy national laboratory, Sandia works in partnership with universities and industry to enhance the security, prosperity, and well-being of the nation. We provide scientific and engineering solutions to meet national needs in nuclear weapons and related defense systems, energy security, and environmental integrity, and to address emerging national challenges for both government and industry.

On Oct. 1, 1993, the Department of Energy awarded the Sandia management contract to the Martin Marietta Corp., now Lockheed Martin. Today, Sandia has two primary facilities, a large laboratory and headquarters in Albuquerque (more

R. D. Robinett III et al., *Flexible Robot Dynamics and Controls*

than 6,600 employees) and a smaller laboratory in Livermore, California (about 850 employees). Sandia is a government-owned/contractor-operated (GOCO) facility.

Sandia's special mix of core competencies, talented staff, and unique facilities for DOE missions increasingly doubles as a technology resource for other national challenges. Sandia's integrated capabilities—advanced manufacturing technology, electronics technology, advanced information technology, and pulsed power technology—are strategically required for DOE's defense, energy, and environmental missions. In concert with DOE's evolving mission to provide technological support to other federal agencies, Sandia will continue to serve as a resource to those agencies needing objective technical analyses, rapid prototyping of new concepts, or access to Sandia's special capabilities.

Over the past several years, Sandia has become a valuable resource for U.S. industry. By partnering with industry, both one-on-one and in consortia, Sandia accelerates the advancement of technology from research through development to commercialization. In return, collaborative exchanges strengthen Sandia by exercising its core competencies and providing opportunities for direct interaction with the nation's industrial R&D base. A strategy of great importance to Sandia involves joining forces with the complementary skills of university research laboratories. Adoption of this strategy is driven by our desire to participate in the formation and mobilization of a fully integrated technological resource for the nation. Partnerships with industry and universities increase the technological leverage Sandia hopes to gain over problems facing the nation, thereby increasing Sandia's ability to render *exceptional service in the national interest*[1].

1.2. Flexible Robotics Research Historical Background

In the early 1980's Sandia became interested in robotics and automation activities to support several DOE missions. Over the first few years, several projects and customers were being supported and within the first decade the robotics group grew rapidly into a center. To accommodate the center's unique needs for robotics and intelligent systems research, Sandia and the DOE mutually developed a 73,000 square-foot Robotics Manufacturing Science and Engineering Laboratory (RMSEL) facility. Today the Intelligent Systems & Robotics Center (ISRC) employs over 150 scientists, engineers, and technicians. The ISRC's diversified departments range from

1. Intelligent Systems Sensors & Controls,
2. Intelligent Systems Principles,
3. Engineering & Manufacturing Software Development,
4. Mobile Robotics, to

5. Applied Systems.

The results of one of the initial robotics projects successfully demonstrated swing-free techniques using robot technology[2] while handling and transporting U.S. Army munitions. From this initial project, many other basic R&D breakthroughs that utilized input shaping and control techniques were achieved for flexible robotics systems and payloads. These techniques were documented through publications and several U.S. patents.

1.2.1. Milestones To Present Flexible Robotics Research

1985
- Starr, G. P., "Swing-free transport of suspended objects with a path-controlled robot manipulator," *ASME Journal of Dynamic Systems, Measurement, and Control*, March 1985.

1987
- Jones, J. F., Petterson, B. J., and Werner, J. C., "Swing damped movement of suspended objects," SAND87-2189.

1989
- Werner, J. C., Robinett, R. D., III, and Petterson, B. J., "Swing-free movement of simply suspended objects employing parameter estimation," SAND89-2511.
- Petterson, B. J., Robinett, R. D., III, and Werner, J.C., "Parameter-scheduled trajectory planning for suppression of coupled horizontal and vertical vibrations in a flexible rod," SAND89-2522C.
- Eisler, G. R., Segalman, D. J., and Robinett, R. D., III, "Approximate minimum-time trajectories for two-link flexible manipulators," SAND89-1997C.
- Filed. Jones, J. F., Petterson, B. J., Strip, D. R., *Methods of and system for swing damping movement of suspended objects*, US Patent No. 4,997,095, March 5, 1991. A payload suspended from a gantry is swing damped in accordance with a control algorithm based on the periodic motion of the suspended mass or by servoing on the forces induced by the suspended mass.

1990
- Segalman, D. J. and Dohrmann, C. R., "Dynamics of rotating flexible structures by a method of quadratic modes," SAND90-2737.
- Feddema, J. T., "Integration of model-based and sensor-based control for a two-link flexible robotic arm," SAND90-1292C.
- Schoenwald, D. A., Feddema, J. T., Eisler, G. R., Segalman, D. J., "Minimum-time trajectory control of a two-link flexible robotic manipulator," SAND90-2472C.

1991
- Eisler, G. R., Segalman, D. J., Robinett, R. D., III, Feddema, J. T., "Approximate optimal trajectories for flexible-link manipulator slewing," SAND91-2858J.
- Petterson, B. J., Robinett, R. D., III, Werner, J. C., "Lag-stabilized force feedback damping," SAND91-0194.

1992
- Feddema, J. T., "Digital filter control of remotely operated flexible robotic structures," SAND92-1630C.

1994
- Parker, G. G., Eisler, G. R., Phelan, J., and Robinett, R. D., III, "Input shaping for vibration-damped slewing of a flexible beam using a heavy-lift hydraulic robot," SAND94-1702C.
- Segalman, D. J., Parker, G. G., Robinett, R. D., III, and Inman, D. J., "Decentralized sliding mode control of nonlinear flexible robots," SAND94-0337C.

- Segalman, D. J., Parker, G. G., Robinett, R. D., III, and Inman, D. J., "Sliding mode control of nonlinear flexible structural systems," SAND94-0174J.
- Dohrmann, C. R. and Robinett, R. D., III, "Robot trajectory planning via dynamic programming," SAND94-0623C.

1995

- Parker, G. G. and Robinett, R. D., III, "Output feedback sliding mode control with application to flexible multibody systems," SAND95-2573J.
- Feddema, J. T., Robinett, R. D., III, Eisler, G. R., Parker, G. G., "Optimal trajectories for flexible-link manipulator slewing using recursive quadratic programming: experimental verification," SAND95-2473J.
- Dohrmann, C. R. and Robinett, R. D., III, "Efficient sequential quadratic programming implementations for equality-constrained discrete-time optimal control," SAND95-2574J.
- Parker, G. G., Petterson, B. J., Dohrmann, C. R., Robinett, R. D., III, "Vibration suppression of fixed-jib crane maneuvers," SAND95-0139C.
- Parker, G. G., Petterson, B. J., Dohrmann, C. R., Robinett, R. D., III, "Command shaping for residual vibration free crane maneuvers," SAND95-1195C.

1996

- Parker, G. G., Robinett, R. D., III, Driessen, B. J., and Dohrmann, C. R., "Operator in-the-loop control of rotary cranes," SAND96-0373C.
- Wilson, D. G., Stokes, D., Starr, G. P., Robinett, R. D., III, "Optimized input shaping for a single flexible robot link," SAND96-0699C.
- Feddema, J. T., Baty, R. S., Dykhuizen, R. C., Dohrmann, C. R., Parker, G. G., Robinett, R. D., III, Romero, V. J., and Schmitt, D. J., "Modeling, system identification, and control for slosh-free motion of an open container of fluid," SAND96-0995.
- Filed. Feddema, J. T., Petterson, B. J., and Robinett, R. D., III, *Operator control systems and methods for swing-free gantry-style cranes*, US Patent No. 5,785,191, July 28, 1998. A system and method for eliminating swing motions in gantry-style cranes while subject to operator control is presented. The present invention includes an infinite impulse response (IRR) filter and a proportional-integral (PI) feedback controller to eliminate oscillations by adjusting acceleration and deceleration input profiles. In addition, this invention also compensates for variable-length cable motions from multiple cables attached to a suspended payload.

1997

- Wilson, D. G., Parker, G. G., Starr, G. P., Robinett, R. D., III, "Modeling and robust control of a flexible manipulator," SAND97-0057C.
- Dohrmann, C. R. and Robinett, R. D., III, "A dynamic programming method for constrained discrete-time optimal control," SAND97-1066J.
- Filed. Robinett, R. D., III, Parker, G. G., Feddema, J. T., Dohrmann, C. R., and Petterson, B. J., *Sway control method and system for rotary cranes*, US Patent No. 5,908,122, June 1, 1999. Methods and apparatuses for reducing the oscillatory motion of rotary crane payloads during operator-commanded or computer-controlled manuevers. An input-shaping filter receives input signals from multiple operator input devices and converts them into output signals readable by the crane controller to dampen the payload tangential and radial sway associated with rotation of the jib.

One of the R&D highlights was the real world hardware input shaping technology validation[3,4]. Most industrial cranes do not automatically compensate for payload

sway at the end of the motion. The operator relies on past experience to bring the payload to a swing-free stop. Of those cranes that have automatic compensation, they typically only work for pre-planned motions where the desired end position of the payload, in crane coordinates, is well specified. Unfortunately, cranes are most often used in unstructured environments where the end position, in crane coordinates, is not well specified, such as ship yards and factory floors. Consider the challenge of defining the desired position of a container on a ship as that ship oscillates with the waves. Because most cranes are guided to their final destination through an operator's joystick, a clever and unique solution was proposed. The Swing-free Operator Control (SOC) system uses the joystick input and modifies it in such a fashion that the resulting motion of the payload is *swing-free.*

The SOC features input shaping technology based on an Infinite Impulse Response Filter (IIR) design (see Chapter 5). The IIR filter transforms the operator's joystick input into a signal that prevents end-point swinging. Utilizing measurements from a hoist cable length encoder the IIR filter parameters are updated real-time. The IIR filter algorithm is programmed into a standalone embedded computer. The SOC is integrated between the operator's joystick and existing crane controller (see Figure 1.1).

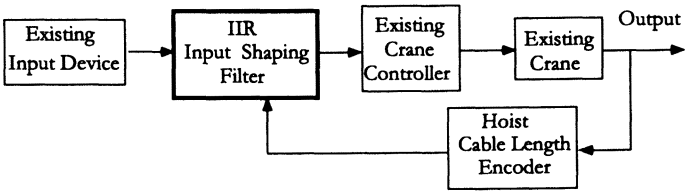

Figure 1.1. SOC IIR filter integration with crane control system

The SOC system successfully demonstrated input shaping technology on an overhead shipyard crane in Gulfport, Mississippi (see Figure 1.2). Numerous large containers were simultaneously moved from side-to-side, raised and lowered while the SOC system minimized container sway and reduced operation times.

In the next section, the content of each chapter is previewed.

1.3. Outline of the Book

This book is a compilation of over 10 years worth of research and development at Sandia in flexible robot dynamics and controls. The R&D began approximately 16 years ago with a munitions transportation project for the U.S. Army where Sandia enlisted Professor Greg Starr of The University of New Mexico (UNM) to solve the input shaping problem[2]. Since that initial project, the concept of

Figure 1.2. Overhead shipyard crane loader/unloader in Gulfport, Mississippi

input shaping has been expanded to include trajectory optimization, optimal path planning, optimal control, nonlinear programming, and dynamic programming as well as being integrated into linear and nonlinear feedback control techniques.

The main goal of this book is to demonstrate the value of solid applied mathematical theory when applied to *real world* hardware problems. This concept is continually emphasized by investigating and solving the overhead gantry robot problem within each subsequent chapter. As a result, this book provides a basic development of the required mathematics with many examples of real world hardware problems as well as many useful references for the interested reader. The basic development assumes that the reader is an upper level undergraduate or a graduate engineering student, or a practicing engineer.

Chapter 2 provides the basic mathematical techniques used throughout the rest of the book for modeling, simulation, control design, input shaping, and optimization. The experienced reader can skip this chapter and begin with Chapter 3. Chapter 3 is unique because it develops the method of quadratic modes for modeling flexible link robots. The quadratic modes approach accounts for the classic problem of link foreshortening that is often incorrectly modeled in many multibody simulation codes.

Chapter 4 and Chapter 5 develop the techniques of system identification and input shaping for path planning. System identification is a fundamental step in model verification for creating input shaping strategies. As you will see, the critical parameter for input shaping on the overhead gantry robot is the pendulum frequency of the load line. The input shaping techniques that are presented were mainly

developed at Sandia, but the authors believed that the work performed at MIT within Professor Warren Seering's research group deserved special attention.

Chapter 6 is a discussion of linear feedback control with a couple of special topics. The first topic is a technique that relies on delay or latency in the feedback loop for stability. Lag-stabilized feedback control goes against the conventional wisdom that feedback delay is a destabilizing effect. The second topic is non-collocation of sensors and actuators, commonly referred to as non-minimum phase. This situation is often encountered when designing a control system for a flexible link robot.

Chapter 7 and Chapter 8 present robot nonlinear control techniques. Chapter 7 provides the mathematical background while Chapter 8 extends the discussion to nonlinear adaptive control. This adaptive control discussion is based upon some of Sandia's most recent flexible robot research.

To further motivate the reader, the following problems, which were used as midterm and final projects during a class taught at UNM, were solved with the techniques discussed in this book.

Midterm Project

The midterm project will consist of designing an input shaped, open-loop control for a jib crane (see Figure 1.3).

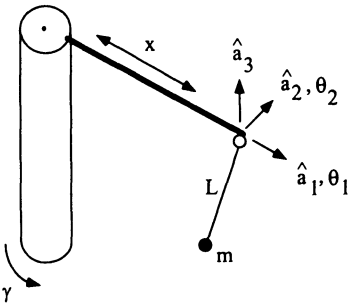

Figure 1.3. Jib crane schematic diagram

Task 1 Derive the equations of motion for a fixed length jib crane (the results of one of your homework problems).

Task 2 Develop a numerical simulation of the linearized jib crane.

Task 3 Design an input shaped command for the jib crane to produce a rest-to-rest, residual vibration-free, 90° maneuver (see Figure 1.4). One must formulate a trajectory optimization problem and solve it with the Recursive Quadratic Programming (RQP) algorithm in the MATLAB® Optimization Toolbox. The torque limit is approximately 3 rad/sec/sec and is based on the acceleration profile.

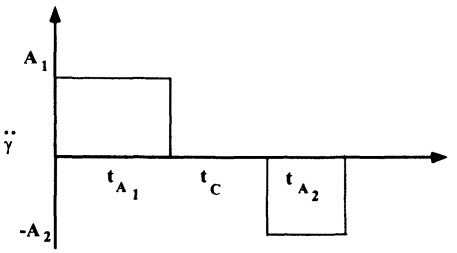

Figure 1.4. Analytical acceleration profile

Task 4 The results of the prior tasks must be formally documented in a midterm report including the derivations, descriptive figures, plots, and software code listings.

Task 5 A ten minute oral presentation will be performed by a designated team member.

Task 6 The input shaped command will be tested on the Sandia hardware and the results will be included in the final report.

Final Project

The final project will consist of designing an input shaped, open-loop control for a single flexible robot arm (see Figure 1.5).

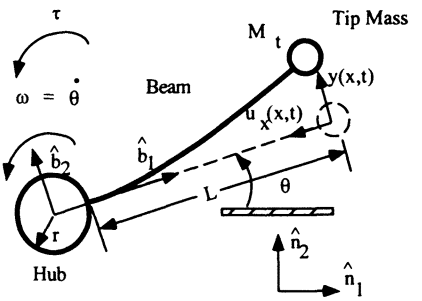

Figure 1.5. Flexible robot arm schematic diagram

Task 1 Derive the equations of motion for a single, flexible robot arm including the first two modes and the actuator dynamics.

Task 2 Develop a numerical simulation of the flexible robot arm.

Task 3 Perform system identification on the hardware to refine your model. Input-output data will be provided.

Task 4 Design an input shaped command for the flexible robot arm to produce a rest-to-rest, residual vibration-free 90° maneuver that is insensitive to a variation in tip mass. You must determine the torque limits and the motion limits of the hardware. In addition, you should minimize the control effort and perform the maneuver in less than 2 seconds.

Task 5 The results of the prior tasks must be formally documented in a final report including the derivations, descriptive figures, plots, and software code listings.

Task 6 A ten minute oral presentation will be performed by a designated team member.

Task 7 The input shaped command will be tested on the UNM hardware, and the results will be included in the final report.

Extra Credit Tasks

Task A Design a linear, closed-loop, tracking controller for the flexible arm.

Task B Design an input shaping filter for the flexible arm.

Task C Design a nonlinear, closed-loop, tracking controller for the flexible arm.

Further Reading

Reference books that closely address some of the topics in more detail are suggested for further reading. Several industry standard texts on robot dynamics and controls have been written by Craig,[5] Paul,[6] and Spong and Vidyasagar[7]. In the area of flexible multibody dynamics both Shabana,[8] and Amirouche[9] provide detailed model developments. Several books written by Junkins and Kim,[10] Kelar and Joshi,[11] and Canudas de Wit, Siciliano, and Bastin[12] most closely relate to the flexible robot material covered in this book. For more specific topics, such as optimal control, optimization, dynamic programming, input-shaping, etc., the reader is directed to each chapter's reference list. At the end of each chapter is a detailed list of references. These references serve as a foundation for the progressive development of each chapter. For the interested reader, they provide a ready resource of further details and developments. For the latest research and development in flexible robot dynamics and controls, some archival journals are a rich environment. The professional engineering societies such as IEEE,[13–15] AIAA,[16] ASME,[17] and several journal publishers[18–20] provide special emphasis in many aspects of dynamics and controls.

1.4. Chapter 1 Summary

This chapter introduced the history of flexible robot dynamics and controls at Sandia National Laboratories. Included was a list of flexible robotics research milestones. Many of the SAND reports resulted in published conference proceedings, journal articles, and some became U.S. patents. One particular application reports the successful implementation of an IIR input shaping filter to reduce residual oscillation for an overhead shipyard crane control system. Two projects (that were successfully used in the classroom environment) are presented with prescriptive tasks and objectives. Each project applied the analytical concepts (introduced in this book) and concluded with experimental validation. In the next chapter, mathematical fundamentals are introduced that apply throughout the remainder of the book. Chapter 3 presents the unique development of dynamic models for flexible robot dynamics. In Chapter 4, system identification techniques are presented that lead to calibrated mathematical models for implementation of open-loop control (see Chapter 5), linear closed-loop control (see Chapter 6) and advanced nonlinear control (see Chapters 7 and 8) design techniques.

The main emphasis of this book is applied research with an instruction basis, and real world hardware implementation.

1.5. Chapter 1 References

1. —,—, "About Sandia" [online]. Available from: http://www.sandia.gov/About.htm [Accessed March 26, 2001].
2. G. P. Starr, "Swing-free transport of suspended objects with a path-controlled robot manipulator," *ASME Journal of Dynamic Systems, Measurement, and Control*, March, 1985.
3. J. T. Feddema, "Digital filter control of remotely operated flexible structures," *Proceedings of the American Control Conference*, pp. 2710-2715, San Francisco, CA, June, 1993.
4. J. T. Feddema, B. J. Petterson, and R. D. Robinett, *Operator Control Systems and Methods for Swing-Free Gantry Style Cranes*, U.S. Patent 5,785,191, July 28, 1998.
5. J. J. Craig, *Introduction to Robotics Mechanics and Control*, Addison-Wesley Publishing Co., Inc., 1989.
6. R. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press, Cambridge, MA, 1982.
7. M. W. Spong, and M. Vidyasagar, *Robot Dynamics and Control*, John Wiley & Sons, New York, NY, 1989.
8. A. A. Shabana, *Dynamics of Multibody Systems*, John Wiley & Sons, New York, NY, 1989.
9. F. M. L. Amirouche, *Computational Methods for Multibody Dynamics*, Prentice-Hall, 1992.
10. J. L. Junkins and Y. Kim, *Introduction to Dynamics and Control of Flexible Struc-*

tures, AIAA, Inc., Washington, D.C., 1993.
11. A. Kelar, and S. Joshi, *Control of Nonlinear Multibody Flexible Structures*, Springer-Verlag, New York, NY, 1996.
12. C. Canudas de Wit, B. Siciliano, and G. Bastin, *Theory of Robot Control*, Springer-Verlag, Inc., New York, N.Y., 1997.
13. *IEEE Transactions on Robotics and Automation.*
14. *IEEE Transactions on System, Man, and Cybernetics.*
15. *IEEE Transactions on Automatic Control.*
16. *AIAA Journal of Guidance, Control, and Dynamics.*
17. *ASME Journal of Dynamic Systems, Measurement and Control.*
18. *Journal of Robotic Systems.*
19. *International Journal of Robotics & Automation*, IASTED.
20. *Mechanism and Machine Theory.*

1.6. Chapter 1 Problems

Homework 1.1. A research project.
Research the current literature and locate major applications associated with flexible robots, gantry robots with flexible payloads, and other controlled systems with structural flexibility. Are there any reoccurring problems from year to year? What are the new controls and dynamic modeling contributions and how did they relate to the particular problem being solved. Are there any problems that are considered unique? For the unique problems, construct formal problem definitions and propose feasible research approaches that may lead to potential solutions.

Chapter 2

Mathematical Preliminaries

2.1. Introduction

An understanding of a broad range of mathematical topics is required to analyze, simulate, and control flexible robotics systems. By no means does this chapter provide a complete coverage of all of the mathematical tools that can be used by an engineer, but this chapter does provide a brief review of the most pertinent concepts. For those readers who are familiar with the topics discussed in this chapter, this review is meant to *jog* one's memory and one can always choose to skip this chapter and go directly to Chapter 3 and/or (without any loss of continuity) review the references given at the end of the chapter . On the other hand, this chapter provides an unconventional review of variational calculus and methods for those who are interested in a different point of view. For those readers who are unfamiliar with any of these topics, one should refer to the appropriate references for a more detailed treatment of the material before proceeding to subsequent chapters.

2.2. Linear Algebra

Linear algebra is a fundamental tool that is used in flexible body modeling and system identification. The eigenvalues and eigenvectors of flexible body models are the natural frequencies and mode shapes of a flexible robot arm and will be used repeatedly in Chapters 3, 4, 5, and 6 to analyze experimental systems. Matrices and vectors enable one to perform system identification in a systematic way and will be applied to several hardware platforms.

It is assumed that the reader has at least an introductory understanding of linear algebra. If not, the reader should refer to one of the references for a more detailed treatment as one proceeds through this section[1-5]. This review shall highlight concepts and definitions needed for subsequent chapters as the building blocks for more advanced flexible robotics concepts.

R. D. Robinett III et al., *Flexible Robot Dynamics and Controls*

2.2.1. Linear Independence of Constant Vectors

Definition 2.1. *A set of vectors* $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\}$ *is linearly independent iff* $\exists$ *scalars* $\{\alpha_1, \alpha_2, \ldots \alpha_n\}$ *that are all zero such that*

$$\alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 + \ldots \alpha_n \mathbf{x}_n = \mathbf{0}.$$

2.2.2. Rank and Null Space of a Matrix

Definition 2.2. *The rank of a matrix* $\mathbf{A}$ *is the maximum number of linearly independent rows or columns of* $\mathbf{A}$ *and is denoted by* $\rho(\mathbf{A})$. *The rank of a matrix can be computed in two steps. First, apply a series of elementary row operations to convert the matrix to row echelon form. Second, count the number of nonzero rows to obtain the rank.*

Definition 2.3. *The null space of* $\mathbf{A}$, $\mathcal{N}(\mathbf{A})$ *is spanned by the basis*

$$\mathbf{A}\mathbf{x} = \mathbf{0}. \tag{2.1}$$

The dimension of $\mathcal{N}(\mathbf{A})$ *is called the nullity of* $\mathbf{A}$ *and is denoted by*

$$\nu(\mathbf{A}).$$

Theorem 2.1. *Let* $\mathbf{A} \in \Re^{n \times n}$, *then*

$$\rho(\mathbf{A}) + \nu(\mathbf{A}) = n.$$

Theorem 2.1 provides a powerful tool for determining if $\exists$ a solution to Eq. (2.1). Specifically, if $\mathbf{A}$ is square ($\mathbf{A} \in \Re^{n \times n}$), then there are two possibilities for its rank.

Case 2.1. $\rho(\mathbf{A}) = n$.

This implies that $\nu(\mathbf{A}) = 0$. The only solution to $\mathbf{A}\mathbf{x} = \mathbf{0}$ is the trivial solution $\mathbf{x} = \mathbf{0}$.

Case 2.2. $\rho(\mathbf{A}) < n$.

This implies that $\nu(\mathbf{A}) > 0$. Therefore, $\exists$ a nontrivial solution to $\mathbf{A}\mathbf{x} = \mathbf{0}$. This test is useful when determining eigenvalues as will be necessary in subsequent sections and chapters.

An interesting property of square matrices is that their rank is invariant with respect to pre or post multiplication by nonsingular square matrices. That is, if $\mathbf{A} \in \Re^{n \times n}$, then $\rho(\mathbf{A}) = \rho(\mathbf{A}\mathbf{B}) = \rho(\mathbf{B}\mathbf{A})$. Although the rank of $\mathbf{B}$ is n (since it is nonsingular), the rank of $\mathbf{A}$ can be less than $\mathbf{B}$.

Nonsquare matrices also have some useful properties. Consider an $n \times m$ matrix that has rank equal to one of its dimensions. A new matrix can then be formed that is square and nonsingular. Specifically, if $\mathbf{A} \in \mathbf{C}^{m \times n}$ and $\rho(\mathbf{A}) = n$, then the $n \times n$ matrix $\mathbf{B} = \mathbf{A}^*\mathbf{A}$ has rank n. Alternatively, if $\rho(\mathbf{A}) = m$, then the $m \times m$ matrix $\mathbf{D} = \mathbf{A}\mathbf{A}^*$ has rank m.

Note 2.1. The inverses of $\mathbf{B}$ and $\mathbf{D}$ are examples of pseudoinverses that will be used in system identification (see Chapter 4).

The reverse is also true, that is, if the matrix $\mathbf{B} = \mathbf{A}^*\mathbf{A}$ has rank n, then $\rho(\mathbf{A}) = n$.

2.2.3. Eigenvalues and Eigenvectors

Associated with any square, complex valued $n \times n$ matrix $\mathbf{A}$ are n scalar, vector pairs satisfying the equations

$$\mathbf{A}\mathbf{x}^{(r)} = \lambda_r \mathbf{x}^{(r)} \quad (r = 1, \ldots, n). \tag{2.2}$$

The scalar denoted as λ_r, in general, contains n distinct roots referred to as characteristic values or eigenvalues. The vectors $\mathbf{x}^{(r)}$ are the characteristic vectors or eigenvectors associated with each eigenvalue.

It should be noted that the eigenvalues may be real or complex even if the matrix has all real values. Eigenvalues can also be zero. The number of zero eigenvalues of $\mathbf{A}$ is equal to the nullity of $\mathbf{A}$. Alternatively, the rank of $\mathbf{A}$ tells us the number of nonzero eigenvalues.

Note 2.2. Zero eigenvalues are equivalent to rigid body modes and nonzero eigenvalues are equivalent to flexible body modes (see Chapter 3).

Theorem 2.1 provides a convenient method for computing eigenvalues and their corresponding eigenvectors. After rewriting Eq. (2.2) in the form

$$(\mathbf{A} - \mathbf{I}\lambda_r)\mathbf{x}^{(r)} = \mathbf{0},$$

observe (from Theorem 2.1) that a nontrivial solution exists if and only if

$$det(\mathbf{A} - \mathbf{I}\lambda_r) = \mathbf{0}.$$

If the characteristic polynomial of $\mathbf{A}$ is defined as

$$\Delta(\lambda) \equiv det(\mathbf{A} - \mathbf{I}\lambda),$$

then the roots of the characteristic polynomial are the eigenvalues of $\mathbf{A}$. The eigenvectors corresponding to each λ_r are found by solving Eq. (2.2) for $\mathbf{x}^{(r)}$.

Example 2.1. Find the eigenvalues and eigenvectors for the matrix $\mathbf{A}$

$$\mathbf{A} = \begin{bmatrix} 3 & -5 & -3 \\ 18 & -23 & -18 \\ -16 & 20 & 16 \end{bmatrix}.$$

Solution

The characteristic polynomial is

$$\Delta(\lambda) = det \begin{bmatrix} \lambda - 3 & 5 & 3 \\ -18 & \lambda + 23 & 18 \\ 16 & -20 & \lambda - 16 \end{bmatrix}$$

$$= \lambda(\lambda^2 + 4\lambda + 13).$$

The eigenvalues of $\mathbf{A}$ are the roots of $\Delta(\lambda)$, that is

$$\lambda_1 = 0$$
$$\lambda_2 = -2 + 3j$$
$$\lambda_3 = -2 - 3j.$$

To illustrate eigenvector computation, the first step is to substitute the λ_r's into Eq. (2.2) and solve for their corresponding eigenvectors $\mathbf{x}^{(r)}$. Starting with $\lambda_1 = 0$ gives

$$\begin{bmatrix} 3 & -5 & -3 \\ 18 & -23 & -18 \\ -16 & 20 & 16 \end{bmatrix} \begin{Bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix},$$

or written explicitly

$$3x_1^{(1)} - 5x_2^{(1)} - 3x_3^{(1)} = 0$$
$$18x_1^{(1)} - 23x_2^{(1)} - 18x_3^{(1)} = 0$$
$$-16x_1^{(1)} + 20x_2^{(1)} + 16x_3^{(1)} = 0.$$

Solving for the solution gives

$$\mathbf{x}^{(1)} = \begin{Bmatrix} 1 \\ 0 \\ 1 \end{Bmatrix}.$$

Of course, any multiple of this vector is also an eigenvector $\mathbf{x}^{(1)}$. Sometimes a normalization is introduced to ensure the uniqueness of the eigenvectors. For example, eigenvectors are sometimes normalized to have a magnitude of 1. For this example, the $\mathbf{x}^{(1)}$ eigenvector would be

$$\mathbf{x}_1 = \begin{Bmatrix} \frac{\sqrt{2}}{2} \\ 0 \\ \frac{\sqrt{2}}{2} \end{Bmatrix}.$$

The complete solution set of eigenvalue, eigenvector pairs are

$$\begin{Bmatrix} \lambda_1 \\ \lambda_1 \\ \lambda_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ -2+3j \\ -2-3j, \end{Bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} 1 & \frac{-3}{8} - \frac{-3}{8}j & \frac{3}{8} + \frac{3}{8}j \\ 0 & \frac{-6}{5} - \frac{-3}{20}j & \frac{6}{5} + \frac{3}{20}j \\ 1 & 1 & 1 \end{bmatrix}$$

where the columns of $\mathbf{Q}$ are composed of the eigenvectors $\mathbf{Q} = \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \mathbf{x}^{(3)} \end{bmatrix}$, respectively.

Given a matrix $\mathbf{A}$ its eigenvalues will either all be unique, or some will be repeated. If the eigenvalues are unique, one is guaranteed that a matrix $\mathbf{Q}$ exists that can transform $\mathbf{A}$ into a diagonal matrix where the eigenvalues are on the diagonal.

Note 2.3. By diagonalizing the $\mathbf{A}$ matrix, one can decouple the degrees-of-freedom and simplify the input shaping and feedback control designs (see Chapters 5, 6, 7, and 8).

If the eigenvalues are repeated, it is possible to obtain $\mathbf{Q}$, but it is not guaranteed.

Distinct Eigenvalues

If $\mathbf{A}$ has n distinct eigenvalues, then the eigenvectors $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}$ form a linearly independent set. If the nonsingular matrix $\mathbf{Q} \equiv \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \ldots & \mathbf{x}^{(n)} \end{bmatrix}$, then a diagonal matrix $\widehat{\mathbf{A}}$ is obtained by the similarity transformation $\widehat{\mathbf{A}} = \mathbf{Q}^{-1}\mathbf{A}\mathbf{Q}$ where the diagonal elements are the eigenvalues of $\mathbf{A}$.

Repeated Eigenvalues

If $\mathbf{A}$ has repeated eigenvalues, there may exist a set of n linearly independent eigenvectors. If so, matrix $\mathbf{Q}$ can be formed as above. For each distinct λ_r, there is an integer m_r associated with it called its multiplicity that represents the number of occurrences of λ_r. An eigenvalue with $m_r > 1$ is, by definition, repeated. Each repeated eigenvalue must have m_r independent eigenvectors in order to form $\mathbf{Q}_r$. In general, the eigenvectors of a repeated eigenvalue are not independent.

There is a test to determine if there exists m_r independent eigenvectors for any repeated eigenvalue. Specifically, if $\rho(\mathbf{A} - \lambda_r\mathbf{I}) = n - m_r$, then the eigenvalue λ_r has m_r independent eigenvectors.

2.3. Linear Control Systems

Linear control theory is the basic building block where one begins the development of flexible robot control. It enables the control designer to discuss controller

bandwidth, sampling rates, actuator performance, and sensor sensitivity. In addition, linear control theory is used to assess the linear stability of a system near its equilibrium points and provides a starting point for a nonlinear stability assessment (see Chapter 7).

This section covers a wide range of concepts concerning linear control systems. Initially, the systems considered are time-varying. Later, some features of the specific case of linear time-invariant systems will be addressed[2,3,6,7]. Once again, this section only provides a brief review of concepts and definitions that will be used as building blocks for more advanced concepts. So, the unfamiliar reader is encouraged to consult one or more of the references[2].

2.3.1. Discontinuous Functions

The unit step, pulse, unit impulse, and doublet functions make up a set of discontinous functions useful for describing a wide range of inputs. These functions will be used in Chapter 3 to model applied forces and torques.

Definition 2.4. *The unit step function is defined as*

$$u(t-t_1) \equiv \begin{cases} 0, & t < t_1 \\ 1, & t \geq t_1 \end{cases} .$$

Definition 2.5. *The pulse function is defined as*

$$\delta_\Delta(t-t_1) \equiv \begin{cases} 0, & t < t_1 \\ \frac{1}{\Delta}, & t_1 \leq t < t_1 + \Delta \\ 0, & t \geq t_1 + \Delta \end{cases} .$$

This is shown graphically in Figure 2.1.

Definition 2.6. *The impulse function (Dirac delta function)is defined as the limit of the pulse function as Δ goes to zero, or*

$$\delta(t-t_1) = \lim_{\Delta \to 0} \delta_\Delta(t-t_1).$$

Two important integral properties of the Dirac delta function are

$$\int_{-\infty}^{\infty} \delta(t-t_1)dt = \int_{t_1-\alpha}^{t_1+\alpha} \delta(t-t_1)dt = 1,$$

and

$$\int_{-\infty}^{\infty} f(t)\delta(t-t_1)dt = f(t_1).$$

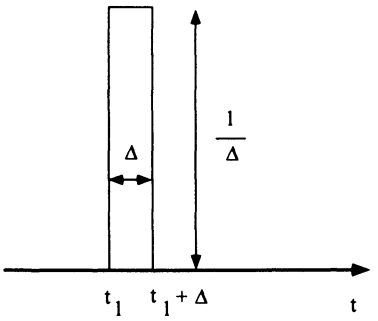

Figure 2.1. Pulse function description

Definition 2.7. *The doublet function is defined as*

$$\int_{-\infty}^{t} f(\tau)\eta(\tau - t_1)d\tau = f(t)\delta(t - t_1) - \dot{f}(t_1)u(t - t_1)$$

where $\eta(t - t_1)$ has been defined within an integration. An alternative definition of the doublet function (for the special case $f(t) = 1$) is

$$\eta(t - t_1) = \frac{d}{dt}\delta(t - t_1).$$

Note, at $t = t_1$ the Dirac delta function is undefined, hence so is the doublet function.

2.3.2. Impulse Response Function

With the impulse response function of a linear system given, one can determine the response of the system to any input by solving an integral equation. Consider a general linear operator $H[\mathbf{u}(t)]$ on the piecewise continuous input vector $\mathbf{u}(t)$. Assume that an output relation exists such that

$$\mathbf{y}(t) = H[\mathbf{u}(t)].$$

The input vector $\mathbf{u}(t)$ can be written as

$$\mathbf{u}(t) = \sum_i \mathbf{u}(t_i)\delta_\Delta(t - t_i)\Delta.$$

Use the linearity of H to obtain

$$\mathbf{y}(t) = \sum_i H[\delta_\Delta(t - t_i)]\mathbf{u}(t_i)\Delta.$$

Taking the limit as Δ go to 0 yields

$$\mathbf{y}(t) = \int_{-\infty}^{\infty} H[\delta(t-\tau)]\mathbf{u}(\tau)d\tau.$$

Definition 2.8. *The impulse response function of the linear function H is defined as*

$$g(t,\tau) = H[\delta(t-\tau)].$$

It is the output of the relaxed system at time t due to an impulse input at time τ. The output $\mathbf{y}(t)$ can be written in terms of the impulse response as

$$\mathbf{y}(t) = \int_{-\infty}^{\infty} g(t,\tau)\mathbf{u}(\tau)d\tau. \tag{2.3}$$

Summary
The impulse response function $g(t,\tau)$ is calculated by applying an impulsive input at time τ and measuring the response over time t. Once this function is known for a system, the output $\mathbf{y}(t)$ can be calculated for the specific input $\mathbf{u}(\tau)$ by solving the integral in Eq. (2.3). The Laplace transform of $g(t,\tau)$ will be used in Chapter 5 to design IIR input shaping filters.

2.3.3. Laplace Transform

The Laplace transform is one of many linear transformations used in linear system analysis that allows for algebraic manipulation of differential equations. This is especially helpful in linear control system design as well as input shaping design.

Definition 2.9. *An integral transform is an operation that transforms a function $f(t)$ to the function $F(s)$ by means of an integral. The general form is*

$$F(s) = \int_{\alpha}^{\beta} K(s,t)f(t)dt.$$

Definition 2.10. *The Laplace transform is an integral transform defined by*

$$L[f(t)] \equiv F(s) \equiv \int_0^{\infty} e^{-st}f(t)dt = \lim_{\beta\to\infty}\int_0^{\beta} e^{-st}f(t)dt.$$

One application of the Laplace transform is to convert ordinary differential equations into algebraic equations. After algebraic manipulations of the Laplace domain equation, it may be transformed back to the physical domain to obtain its solution.

The inverse Laplace transform is formally found by performing a complex contour integral. In practice, use a table[2].

Laplace Transform Properties

1. It is a linear transformation between the t and s domains, so

$$L[a_1 f_1(t) + a_2 f_2(t)] = a_1 F_1(s) + a_2 F_2(s).$$

2. The Laplace transform of the derivative of a function is

$$L\left[\frac{d^n}{dt^n} f(t)\right] = s^n F(s) - s^{n-1} f(0) - s^{n-2} \dot{f}(0) - \ldots - \overset{(n-1)}{f}(0)$$

where $\overset{(n-1)}{f} = d^{n-1} f(t)/dt^{n-1}$.

3. The Laplace transform of the integral of a function is

$$L\left[\int_0^{t_1} \int_0^{t_2} \ldots \int_0^{t_n} f(\tau) d\tau dt_1 \ldots dt_{n-1}\right] = \frac{F(s)}{s^n}.$$

4. The Laplace transform of a function delayed in time by T seconds is

$$L[f(t-T)u(t-T)] = e^{-Ts} F(s)$$

where $u(t)$ is the unit step function.

5. The initial value of $f(t)$ is

$$\lim_{t \to 0} f(t) = \lim_{s \to \infty} sF(s).$$

This can be applied as long as the limit exists.

6. The final value of $f(t)$ is

$$\lim_{t \to \infty} f(t) = \lim_{s \to 0} sF(s).$$

This can be applied as long as the roots of the denominator of $sF(s)$ all have negative real parts.

7. The Laplace transform of a time-scaled function is

$$L\left[f\left(\frac{t}{a}\right)\right] = aF(as)$$

where $F(s) = L[f(t)]$.

8. The inverse Laplace transform of a frequency scaled function is

$$L^{-1}\left[F\left(\frac{s}{a}\right)\right] = af(at)$$

where $L^{-1}[F(s)] = f(t)$.

9. The Laplace transform of a function of the form $e^{-at}f(t)$ is

$$L[e^{-at}f(t)] = F(s+a)$$

where $F(s) = L[f(t)]$.

10. The Laplace transform of the convolution of two time functions is the product of the Laplace transforms of the time functions, so

$$F_1(s)F_2(s) = L\left[\int_0^t f_1(\tau)f_2(t-\tau)d\tau\right].$$

Example 2.2. Find the solution to the differential equation $\ddot{x} + 6\dot{x} + 5x = u(t)$ where $u(t)$ is the unit step function, and the initial conditions are $\dot{x}(0) = x(0) = 0$.

Solution

Taking the Laplace transform of both sides of the differential equation gives

$$s^2X(s) - sx(0) - \dot{x}(0) + 6sX(s) - 6x(0) + 5X(s) = U(s).$$

Applying the given initial conditions permits the simplification,

$$(s^2 + 6s + 5)X(s) = U(s).$$

Replacing the input with the Laplace transform for a unit step results in

$$X(s) = \frac{1}{s(s^2 + 6s + 5)}.$$

Rewriting the left side as a partial fraction expansion gives

$$X(s) = \frac{1/5}{s} - \frac{1/4}{s+1} + \frac{1/20}{s+5}.$$

From a table of inverse Laplace transforms, the solution is

$$x(t) = \left[\frac{1}{5} - \frac{1}{4}e^{-t} + \frac{1}{20}e^{-5t}\right]u(t).$$

The transfer function is the Laplace domain analog of the impulse response function presented earlier. For Example 2.2, one can solve the original system for an impulse input. The Laplace transform of an impulse is unity, making the job easy, thus

$$\frac{X(s)}{U(s)} = \frac{X(s)}{1} = H(s) = \frac{1}{s^2 + 6s + 5} = \frac{1/4}{s+1} - \frac{1/4}{s+5}.$$

Using a table to look up the inverse Laplace transforms gives the impulse response as

$$x(t) = g(t) = \left(\frac{1}{4}e^{-t} - \frac{1}{4}e^{-5t}\right) u(t).$$

From the discussion on the impulse response, it is now known that the solution to any input can now be found by using Eq. (2.3). So, for the unit step response of the example

$$\begin{aligned} x(t) &= \int_0^t g(\tau)d\tau \\ &= \left(\frac{1}{5} - \frac{1}{4}e^{-t} + \frac{1}{20}e^{-5t}\right) u(t). \end{aligned}$$

2.3.4. State-space Realization

In this section, the notation for state-space descriptions of linear systems is presented. Multiple Input Multiple Output (MIMO) systems are the motivation for state-space techniques. Some basic concepts are presented including the fundamental matrix, the state transition matrix, the solution to the state equation, controllability, and observability.

The impulse response matrix and the transfer function matrix are MIMO analogs of the impulse response function and transfer function discussed previously.

Notation

The representation of the equations of motion for linear time-varying systems is

$$\begin{array}{rll} \dot{\mathbf{x}}(t) = & \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) & \mathbf{A} \in \Re^{n\times n} \quad \mathbf{x} \in \Re^{n\times 1} \\ \mathbf{y}(t) = & \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t) & \mathbf{B} \in \Re^{n\times p} \quad \mathbf{u} \in \Re^{p\times 1} \\ & & \mathbf{C} \in \Re^{q\times n} \quad \mathbf{y} \in \Re^{q\times 1} \\ & & \mathbf{D} \in \Re^{q\times p}. \end{array}$$

The Fundamental and State Transition Matrices

First, consider the zero-input state equation $\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t)$.

Definition 2.11. *The $n \times n$ matrix function $\Psi(t)$ is the fundamental matrix of $\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t)$ iff the n columns of $\Psi(t)$ are linearly independent solutions of the zero-input state equation.*

Note 2.4. The linearly independent solutions of the zero-input state equation may be found by taking linear combinations of its solutions.

Note 2.5. The fundamental matrix is not unique.

Example 2.3. Find a fundamental matrix for the system given by

$$\dot{\mathbf{x}} = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 2 \\ -1 \end{bmatrix} \mathbf{u}$$

$$\mathbf{y} = \begin{bmatrix} 5 & 3 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 2 \end{bmatrix} \mathbf{u}.$$

Solution

1. Solve the zero-input equations

$$\begin{aligned} \dot{x}_1 &= -x_1 \\ \dot{x}_2 &= -2x_2 \\ \dot{x}_1 + x_1 &= 0 \\ \dot{x}_2 + 2x_2 &= 0 \\ x_1 &= c_1 e^{-t} \\ x_2 &= c_2 e^{-2t}. \end{aligned}$$

2. Form the columns of a fundamental matrix $\Psi(t)$ as linear combinations of $x_1(t)$, and $x_2(t)$. The constants c_1, and c_2 can be selected arbitrarily as long as they result in linearly independent sets of solutions. Therefore,

$$\Psi(t) = \begin{bmatrix} e^{-t} & 0 \\ 5e^{-2t} & e^{-2t} \end{bmatrix}.$$

Definition 2.12. *If $\Psi(t)$ is a fundamental matrix of the zero-input system $\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x}(t)$, then*

$$\Phi(t, t_0) \equiv \Psi(t)\Psi^{-1}(t_0) \qquad \forall\ t, t_0\ \in (-\infty, \infty)$$

is the state transition matrix of the zero-input system. Note,

- $\Phi(t, t_0)$ *is unique.*
- $\Phi(t, t) = \mathbf{I}$.
- $\Phi^{-1}(t, t_0) = \Phi(t_0, t)$.
- $\Phi(t_2, t_0) = \Phi(t_2, t_1)\Phi(t_1, t_0)$.

The state transition matrix maps any initial state $\mathbf{x}(t_0)$ to the state $\mathbf{x}(t)$ when the input $\mathbf{u}(t)$ is zero. Note, $\Phi(t, t_0)$ satisfies the differential equation

$$\frac{\partial}{\partial t}\Phi(t, t_0) = \mathbf{A}(t)\Phi(t, t_0) \quad \textit{with} \quad \Phi(t_0, t_0) = \mathbf{I}.$$

Example 2.4. Find the state transition matrix of the system of Example 2.3.

Solution

1. Find $\Psi^{-1}(t_0)$ (see Eq. 2.8)

$$\Psi^{-1}(t_0) = \begin{bmatrix} e^{t_0} & 0 \\ -5e^{t_0} & e^{2t_0} \end{bmatrix}.$$

2. Apply Definition 2.12,

$$\Phi(t, t_0) = \begin{bmatrix} e^{-t} & 0 \\ 5e^{-2t} & e^{-2t} \end{bmatrix} \begin{bmatrix} e^{t_0} & 0 \\ -5e^{t_0} & e^{2t_0} \end{bmatrix} = \begin{bmatrix} e^{(t_0-t)} & 0 \\ 0 & e^{2(t_0-t)} \end{bmatrix}.$$

When the dynamical system is time varying, a closed form of $\Phi(t, t_0)$ may not exist. However, some special cases can be found.

1. $\mathbf{A}(t)$ is triangular.

   The uncoupled equation (formed by the last row of $\mathbf{A}(t)$) can be solved. Its solution is then back-substituted to solve the rest.

2. If $\mathbf{A}(t)$ satisfies

$$\mathbf{A}(t)\left(\int_{t_0}^{t} \mathbf{A}(\tau)d\tau\right) = \left(\int_{t_0}^{t} \mathbf{A}(\tau)d\tau\right)\mathbf{A}(t) \quad \forall(t, t_0),$$

   then the state transition matrix is

$$\Phi(t, t_0) = e^{\left[\int_{t_0}^{t} \mathbf{A}(\tau)d\tau\right]}.$$

3. If $\dot{\mathbf{A}}(t) = \mathbf{A}_1\mathbf{A}(t) - \mathbf{A}(t)\mathbf{A}_1$, then $\Psi(t) = e^{\mathbf{A}_1 t}e^{\mathbf{A}_2 t}$ with $\mathbf{A}_2 \equiv \mathbf{A}(0) - \mathbf{A}_1$.

Example 2.5. Find the state transition matrix for the zero-input system

$$\dot{\mathbf{x}} = \begin{bmatrix} -1 & e^{2t} \\ 0 & -1 \end{bmatrix}\mathbf{x} \tag{2.4}$$

Solution

1. Solve the decoupled equation resulting from the last row of Eq. (2.4)

$$\dot{x}_2(t) + x_2(t) = 0.$$

   By inspection the solution is

$$x_2(t) = c_2 e^{-t}. \tag{2.5}$$

2. Solve the coupled equation using the back-substituted solution for $x_2(t)$. The differential equation from the first row of Eq. (2.4) is

$$\dot{x}_1(t) + x_1(t) = e^{2t}x_2(t). \tag{2.6}$$

Initially, solve for the homogeneous solution by setting the right-hand side of Eq. (2.6) to zero and obtain

$$x_{1_h}(t) = c_1 e^{-t}.$$

Next, solve for the particular solution by replacing $x_2(t)$ in Eq. (2.6) with the solution from Eq. (2.5)

$$\dot{x}_1(t) + x_1(t) = e^{2t}\left(c_2 e^{-t}\right) = c_2 e^{t}. \tag{2.7}$$

The particular solution has the form

$$x_{1_p}(t) = c_3 e^{t}.$$

Taking the first derivative and substituting both into Eq. (2.7) yields

$$c_3 e^{t} + c_3 e^{t} = c_2 e^{t},$$

or

$$c_3 = \frac{1}{2}c_2.$$

Therefore,

$$x_{1_p}(t) = \frac{1}{2}c_2 e^{t}.$$

Combining both the homogeneous and particular solutions gives the general solution as

$$x_1(t) = c_1 e^{-t} + \frac{1}{2}c_2 e^{t}.$$

3. Using linear combinations of the solutions for $x_1(t)$ and $x_2(t)$ form a fundamental matrix $\Psi(t)$. The first column comes from setting $c_1 = c_2 = 1$, and the second column comes from setting $c_1 = 1$ and $c_2 = 0$.

$$\Psi(t) = \begin{bmatrix} e^{-t} + \frac{1}{2}e^{t} & e^{-t} \\ e^{-t} & 0 \end{bmatrix} \quad \Psi^{-1}(t_0) = \begin{bmatrix} 0 & e^{t_0} \\ e^{t_0} & -e^{t_0} - \frac{1}{2}e^{3t_0} \end{bmatrix}.$$

4. Apply Definition 2.12 to form the state transition matrix $\mathbf{\Phi}(t, t_0)$ to obtain

$$\mathbf{\Phi}(t, t_0) = \begin{bmatrix} e^{-(t-t_0)} & \frac{1}{2}\left(e^{(t+t_0)} - e^{-(t-3t_0)}\right) \\ 0 & e^{-(t-t_0)} \end{bmatrix}.$$

Solution of the State Equation

Theorem 2.2. *The solution to the state equation*

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u}(t); \qquad \mathbf{x}(t_0) = \mathbf{x}_0$$

is

$$\begin{aligned} \mathbf{x}(t) &= \mathbf{\Phi}(t, t_0)\mathbf{x}_0 + \int_{t_0}^{t} \mathbf{\Phi}(t, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau \\ &= \mathbf{\Phi}(t, t_0)\mathbf{x}_0 + \int_{t_0}^{t} \mathbf{\Phi}(t, t_0)\mathbf{\Phi}(t_0, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau \\ &= \mathbf{\Phi}(t, t_0)\left[\mathbf{x}_0 + \int_{t_0}^{t} \mathbf{\Phi}(t_0, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau\right]. \end{aligned}$$

Proof 2.1. *Consists of two parts.*

1. *The solution satisfies the state equation.*
2. *The solution satisfies the initial condition.*

Part 1

Given

$$\frac{d}{dt}\mathbf{x}(t) = \frac{\partial}{\partial t}\mathbf{\Phi}(t, t_0)\mathbf{x}_0 + \frac{\partial}{\partial t}\int_{t_0}^{t} \mathbf{\Phi}(t, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau.$$

Recall, Liebnitz Rule and apply

$$\frac{d}{dt}\int_a^t f(t, t^*)dt^* = \int_a^t \frac{\partial}{\partial t} f(t, t^*)dt^* + f(t, t).$$

Direct application to the partial derivative of the integral gives

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{A}(t)\mathbf{\Phi}(t, t_0)\mathbf{x}_0 + \int_{t_0}^{t} \mathbf{A}(t)\mathbf{\Phi}(t, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau + \mathbf{\Phi}(t, t)\mathbf{B}(t)\mathbf{u}(t)$$

Note, $\mathbf{\Phi}(t, t) = \mathbf{I}$ *therefore the expression above simplifies to*

$$\begin{aligned} \frac{d}{dt}\mathbf{x}(t) &= \mathbf{A}(t)\mathbf{\Phi}(t, t_0)\mathbf{x}_0 + \mathbf{A}(t)\int_{t_0}^{t} \mathbf{\Phi}(t, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau + \mathbf{B}(t)\mathbf{u}(t) \\ &= \mathbf{A}(t)\left[\mathbf{\Phi}(t, t_0)\mathbf{x}_0 + \int_{t_0}^{t} \mathbf{\Phi}(t, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau\right] + \mathbf{B}(t)\mathbf{u}(t) \\ &= \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t). \end{aligned}$$

Part 2
Given

$$\mathbf{x}(t_0) = \mathbf{x}_0 + \int_{t_0}^{t} \mathbf{\Phi}(t_0, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau = \mathbf{x}_0.$$

The solution to the state equation is commonly divided into two parts,
Zero Input Response ($\mathbf{u} = \mathbf{0}$)

$$\mathbf{\Phi}(t; (t_0, \mathbf{x}_0, \mathbf{0})) = \mathbf{\Phi}(t, t_0)\mathbf{x}_0,$$

and Zero State Response ($\mathbf{x}_0 = \mathbf{0}$)

$$\mathbf{\Phi}(t; (t_0, \mathbf{0}, \mathbf{u})) = \int_{t_0}^{t} \mathbf{\Phi}(t, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau.$$

By substituting the solution of the state equation into its definition, the output solution may be written as

$$\mathbf{y}(t) = \int_{t_0}^{t} \left[\mathbf{C}(t)\mathbf{\Phi}(t, \tau)\mathbf{B}(\tau) + \mathbf{D}(t)\delta(t - \tau)\right] \mathbf{u}(\tau)d\tau.$$

The impulse response matrix is easily seen to be

$$\mathbf{G}(t, \tau) = \mathbf{C}(t)\mathbf{\Phi}(t, \tau)\mathbf{B}(\tau) + \mathbf{D}(t)\delta(t - \tau).$$

Using this notation, reveals the output response as

$$\mathbf{y}(t) = \int_{t_0}^{t} \mathbf{G}(t, \tau)\mathbf{u}(\tau)d\tau. \quad \square$$

2.3.5. Time Invariant Linear Systems

All of the previous discussions for time-varying systems applies to the more specific case of time-invariant systems. However, time-invariant systems afford many simplifications, some of which are addressed in this section.

The linear time-invariant dynamical equations are

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}.$$

Before proceeding, consider the Taylor series expansion of the function

$$e^{\lambda t} = 1 + \lambda t + \frac{1}{2}\lambda^2 t^2 \ldots .$$

This can be generalized to the matrix exponential as

$$e^{\mathbf{A}t} = \mathbf{I} + \mathbf{A}t + \frac{1}{2}\mathbf{A}^2 t^2 + \ldots = \sum_{k=0}^{\infty} \frac{1}{k!} t^k \mathbf{A}^k.$$

The time derivative of $e^{\mathbf{A}t}$ is

$$\frac{d}{dt}\, e^{\mathbf{A}t} = \sum_{k=1}^{\infty} \frac{1}{(k-1)!}\, t^{k-1}\mathbf{A}^k = \mathbf{A}\left[\sum_{k=0}^{\infty} \frac{1}{k!}\, t^k \mathbf{A}^k\right] = \mathbf{A}e^{\mathbf{A}t} = e^{\mathbf{A}t}\mathbf{A}.$$

Since $e^{\mathbf{A}t}$ satisfies

$$\frac{d}{dt} e^{\mathbf{A}t} = \mathbf{A}e^{\mathbf{A}t},$$

$e^{\mathbf{A}t}$ is a fundamental matrix of the dynamical equation. This gives a method for finding the state transition matrix. Namely,

$$\mathbf{\Phi}(t, t_0) = e^{\mathbf{A}t}\left(e^{\mathbf{A}t_0}\right)^{-1} = e^{\mathbf{A}(t-t_0)}. \tag{2.8}$$

This can be used in writing the solution to the dynamical equation where the initial time t_0 has been assigned to zero (this is a mathematical convenience exploited for time-invariant systems). The dynamical equations are

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}_0 + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau \tag{2.9}$$

$$\mathbf{y}(t) = \mathbf{C}e^{\mathbf{A}t}\mathbf{x}_0 + \mathbf{C}e^{\mathbf{A}t}\int_0^t e^{-\mathbf{A}\tau}\mathbf{B}\mathbf{u}(\tau)d\tau + \mathbf{D}\mathbf{u}(t). \tag{2.10}$$

The impulse response matrix is easily seen to be

$$\mathbf{G}(t, \tau) = \mathbf{C}e^{\mathbf{A}(t-\tau)}\mathbf{B} + \mathbf{D}\delta(t-\tau),$$

or

$$\mathbf{G}(t) = \mathbf{C}e^{\mathbf{A}t}\mathbf{B} + \mathbf{D}\delta(t).$$

The dynamical equations (see Eq. 2.9 and Eq. 2.10) can also be analyzed in the Laplace domain. Taking the Laplace transform of the solution to the dynamical equations results in

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}_0 + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s)$$
$$\mathbf{Y}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}_0 + \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s) + \mathbf{D}\mathbf{U}(s).$$

The direct analogy of the time-domain impulse response matrix to the Laplace domain is called the transfer function matrix. From the solution to the dynamical equation, this is easily seen to be

$$\mathbf{G}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$$

where the Laplace transform of $e^{\mathbf{A}t}$ has been used

$$L\left[e^{\mathbf{A}t}\right] = (s\mathbf{I} - \mathbf{A})^{-1}. \tag{2.11}$$

2.3.6. Controllability

It is important to know whether a system is completely controllable or if one needs to add more actuators. This concept will be discussed in detail in Chapter 7 with respect to augmented and output feedback sliding mode controllers.

Definition 2.13. *The system* $\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{x}$ *is controllable at time* t_0 *if* $\exists$ *at time* t_1 *such that* $\forall$ *initial state* $\mathbf{x}_0$, *and* $\forall$ *final states* $\mathbf{x}_1$ $\exists$ *an input* $\mathbf{u}$ *that will transfer the state from the initial state to the final state at time* t_1.

Note 2.6. For the time-invariant case, the controllability of the system is not dependent on time.

Theorem 2.3. *The n-dimensional system above is controllable iff any of the following conditions are satisfied:*

1. *All rows of* $e^{-\mathbf{A}t}\mathbf{B}$ *(or* $e^{\mathbf{A}t}\mathbf{B}$*) are linearly independent over the field of complex numbers* $\mathbb{C}$.
2. *All rows of* $(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$ *are linearly independent over* $\mathbb{C}$.
3. *The* $n \times np$ *controllability matrix* $\left[\mathbf{B}\ \mathbf{AB}\ \mathbf{A}^2\mathbf{B}\ \ldots\ \mathbf{A}^{n-1}\mathbf{B}\right]$ *has rank* n.

If one knows that a linear system is controllable, then it is known that an input exists that can move the states from any initial position to any final position. Other methods do exist to obtain the same information[2].

2.3.7. Observability

It is important to know whether a system is completely observable or if one needs to add more sensors and/or a Kalman filter. This concept will be discussed in detail in Chapter 7.

Definition 2.14. *The system*

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u},$$

is observable at t_0 *if* $\exists$, *at time* t_1 *such that knowledge of the input* $\mathbf{u}$ *and the output* $\mathbf{y}$ *over the time interval* $[t_0, t_1]$ *is sufficient to determine the initial state* $\mathbf{x}_0$.

Theorem 2.4. *The n-dimensional system above is observable iff any of the following conditions are satisfied:*

1. *All columns of* $\mathbf{C}e^{\mathbf{A}t}$ *are linearly independent over* $\mathbb{C}$.
2. *All columns of* $\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}$ *are linearly independent over* $\mathbb{C}$.
3. *The* $nq \times n$ *observability matrix* $\left[\mathbf{C}\ \mathbf{CA}\ \mathbf{CA}^2\ \ldots\ \mathbf{CA}^{n-1}\right]^T$ *has rank* n.

If one knows that a linear system is observable, then it is known that the state history can be calculated based on knowledge of the past inputs and outputs.

2.4. Digital Systems

Any control system that one designs for a flexible robot will be implemented on an embedded computer. As a result, the control system must be directly written in digital form and/or transformed from continuous time to discrete time.

In this section, a few topics regarding sampled data systems are covered. Specifically, digital integration methods for simulating linear and nonlinear systems. To this end, the Z-transform and the unit impulse response function (the discrete analog of the impulse response function) are introduced. Finally, the discrete form of the state equations is discussed[2,6,7,8].

2.4.1. Introduction to Integration Algorithms

Special notation is needed to describe sampled data systems. Namely,

n	=	An integer sample number
h	=	The time interval between samples
$\mathbf{x}_n$	=	The value of $\mathbf{x}$ at time $t = nh$
$\mathbf{x}_{n+1}$	=	The value of $\mathbf{x}$ at time $t = h(n+1)$.

Consider a simple dynamical system represented as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t). \tag{2.12}$$

Integrating both sides of this equation from $t = nh$ to $t = h(n+1)$ gives

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \int_{nh}^{h(n+1)} \mathbf{f}(t)dt, \tag{2.13}$$

and is represented graphically in Figure 2.2. However, the area under the $\mathbf{f}(t)$ curve must be approximated because $\mathbf{f}(t)$ is only known at the sample points $t = nh$ and $t = h(n+1)$. One approximation known as Euler integration is shown graphically in Figure 2.3. The integral of Eq. (2.13) is approximated as

$$\int_{nh}^{h(n+1)} \mathbf{f}(t)dt \approx h\mathbf{f}_n \tag{2.14}$$

and when Eq. (2.14) is substituted into Eq. (2.13) one obtains

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{f}_n.$$

If the area under the $\mathbf{f}(t)$ curve is approximated using a trapezoid (instead of a rectangle), then the approximation for the state $\mathbf{x}_{n+1}$ becomes

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{2}(\mathbf{f}_n + \mathbf{f}_{n+1}).$$

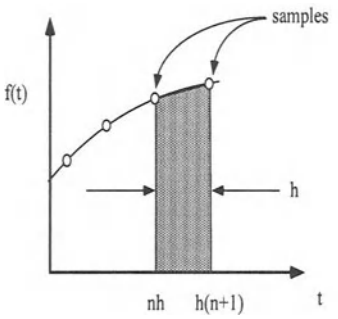

Figure 2.2: Graphical representation of numerical integration

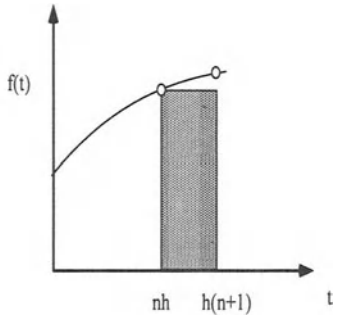

Figure 2.3: Graphical representation of Euler integration

This requires knowledge of the function evaluated at $t = h(n+1)$. If $\mathbf{f}$ is also a function of $\mathbf{x}$, then the evaluation of $\mathbf{f}_{n+1}$ requires knowledge of $\mathbf{x}_{n+1}$. If $\mathbf{f}$ is a linear function of $\mathbf{x}$, then the approximation of $\mathbf{x}_{n+1}$ can be solved. However, if $\mathbf{f}$ is a nonlinear function of $\mathbf{x}$ and $\mathbf{x}_{n+1}$ cannot be solved for, then this trapezoidal method becomes implicit and not appropriate for real-time simulation.

These concepts can be generalized to dynamical systems represented by

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$$

where $\mathbf{u}$ is the system input and t appears when there is an explicit time dependency. First, define $\mathbf{f}_n$ as the evaluation of $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$ at $t = nh$, that is

$$\mathbf{f}_n = \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n, nh).$$

Again, the approximations of states at time $h(n+1)$ for Euler and Trapezoidal integration are

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{f}_n$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{2}(\mathbf{f}_n + \mathbf{f}_{n+1}).$$

Several of the common integration algorithms, along with error information, used for solving dynamical system equations are given on the following pages. Several measures of error are used when evaluating an integration method. The one presented in Eq. (2.15) indicates how well the time constant of the simulated first-order linear system $\tau\dot{x} + x = u$ match those of the true linear system.

$$e_\tau = \frac{\tau^* - \tau}{\tau} \tag{2.15}$$

where τ^* is the approximate time constant and τ is the exact value.

2.4.2. Explicit (Single Pass)

These methods allow the next value $\mathbf{x}$ to be computed using previous values of $\mathbf{f}$ and $\mathbf{x}$. Because of this feature, they are called *explicit* and are suitable for real-time, operator-in-the-loop simulations. These algorithms are listed without derivations for the ease of reference.

Euler

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + h\mathbf{f}_n \\ e_\tau &= \frac{h}{2}. \end{aligned} \tag{2.16}$$

AB-2 (Adams-Bashforth)

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{h}{2}(3\mathbf{f}_n - \mathbf{f}_{n-1}) \\ e_\tau &= \frac{5}{12}h^2. \end{aligned}$$

AB-3

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{h}{12}(23\mathbf{f}_n - 16\mathbf{f}_{n-1} + 5\mathbf{f}_{n-2}) \\ e_\tau &= \frac{3}{8}h^3. \end{aligned}$$

AB-4

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{h}{24}(55\mathbf{f}_n - 59\mathbf{f}_{n-1} + 37\mathbf{f}_{n-2} - 9\mathbf{f}_{n-3}) \\ e_\tau &= \frac{251}{720}h^4. \end{aligned}$$

AB-5

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{h}{720}(1901\mathbf{f}_n - 2774\mathbf{f}_{n-1} + 2616\mathbf{f}_{n-2} - 1274\mathbf{f}_{n-3} + 251\mathbf{f}_{n-4}) \\ e_\tau &\approx h^5. \end{aligned}$$

2.4.3. Implicit (Single Pass)

These methods compute the next value of $\mathbf{x}$ using, in general, $\mathbf{x}_{n+1}$. If $\mathbf{f}$ depends on $\mathbf{x}$ nonlinearly, then these methods are not suitable for real-time applications. These algorithms are listed without derivations for the ease of reference.

Trapezoidal

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{2}(\mathbf{f}_n + \mathbf{f}_{n+1})$$
$$e_\tau = -\frac{1}{12}h^2.$$

3rd-Order

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{12}(5\mathbf{f}_{n+1} + 8\mathbf{f}_n - \mathbf{f}_{n-1})$$
$$e_\tau = -\frac{1}{24}h^3.$$

4th-Order

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{24}(9\mathbf{f}_n + 19\mathbf{f}_{n-1} - 5\mathbf{f}_{n-2} + \mathbf{f}_{n-3})$$
$$e_\tau = -\frac{19}{720}h^4.$$

Next, several Runge-Kutta methods will be presented with a brief discussion regarding real-time compatibility. In short, an integration method is real-time compatible if the algorithm does not require $\mathbf{u}_{n+1}$ to calculate $\mathbf{x}_{n+1}$. This is important for simulating real-time systems where the input is the result of a closed-loop control law. Typically, the control law is a function of the current state and physically is not available until the new state information has been calculated. Note, the number of passes determines the order of the routing.

2.4.4. Second-order Runge-Kutta

First Pass

$$\mathbf{f}_n = \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n)$$
$$\mathbf{x}^*_{n+1} = \mathbf{x}_n + h\mathbf{f}_n. \qquad \textit{Euler}$$

Second Pass

$$\mathbf{f}^*_{n+1} = \mathbf{f}\left(\mathbf{x}^*_{n+1}, \mathbf{u}_{n+1}\right)$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{2}\left(\mathbf{f}_n + \mathbf{f}^*_{n+1}\right) \qquad \textit{Trapezoidal}$$
$$e_\tau = \frac{1}{6}h^2.$$

Note, this is not r al-time compatible.

2.4.5. Real-Time, Second-Order Runge-Kutta

First Pass

$$\mathbf{f}_n = \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n)$$
$$\mathbf{x}^*_{n+\frac{1}{2}} = \mathbf{x}_n + \frac{h}{2}\mathbf{f}_n. \qquad \textit{Euler}$$

Second Pass

$$\mathbf{f}^*_{n+\frac{1}{2}} = \mathbf{f}\left(\mathbf{x}^*_{n+\frac{1}{2}}, \mathbf{u}_{n+\frac{1}{2}}\right)$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{f}^*_{n+\frac{1}{2}} \qquad \textit{Euler}$$
$$e_r = \frac{1}{6}h^2.$$

Note, this is real-time compatible.

2.4.6. Third-order Runge-Kutta

First Pass

$$\mathbf{f}_n = \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n)$$
$$\mathbf{x}^*_{n+\frac{1}{3}} = \mathbf{x}_n + \frac{h}{3}\mathbf{f}_n. \qquad \textit{Euler}$$

Second Pass

$$\mathbf{f}^*_{n+\frac{1}{3}} = \mathbf{f}\left(\mathbf{x}^*_{n+\frac{1}{3}}, \mathbf{u}_{n+\frac{1}{3}}\right)$$
$$\mathbf{x}^*_{n+\frac{2}{3}} = \mathbf{x}_n + \frac{2}{3}h\mathbf{f}^*_{n+\frac{1}{3}}. \qquad \textit{Euler}$$

Third Pass

$$\mathbf{f}^*_{n+\frac{2}{3}} = \mathbf{f}\left(\mathbf{x}^*_{n+\frac{2}{3}}, \mathbf{u}_{n+\frac{2}{3}}\right)$$
$$\mathbf{f}^{**}_{n+\frac{1}{3}} = \frac{1}{2}\left(\mathbf{f}_n + \mathbf{f}^*_{n+\frac{2}{3}}\right) \qquad 1st\ Average$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{2}\left(\mathbf{f}^{**}_{n+\frac{1}{3}} + \mathbf{f}^*_{n+\frac{2}{3}}\right) \qquad 2nd\ Average$$
$$e_r = \frac{1}{24}h^3.$$

Note, this method is real-time compatible.

2.4.7. Fourth-order Runge-Kutta (RK-4)

First Pass

$$\mathbf{f}_n = \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n)$$
$$\mathbf{x}^*_{n+\frac{1}{2}} = \mathbf{x}_n + \frac{1}{2}h\mathbf{f}_n. \qquad \textit{Euler}$$

Second Pass

$$\mathbf{f}^*_{n+\frac{1}{2}} = \mathbf{f}\left(\mathbf{x}^*_{n+\frac{1}{2}}, \mathbf{u}_{n+\frac{1}{2}}\right)$$
$$\mathbf{x}^{**}_{n+\frac{1}{2}} = \mathbf{x}_n + \frac{1}{2}h\mathbf{f}^*_{n+\frac{1}{2}}. \qquad \textit{Euler}$$

Third Pass

$$\mathbf{f}^{**}_{n+\frac{1}{2}} = \mathbf{f}\left(\mathbf{x}^{**}_{n+\frac{1}{2}}, \mathbf{u}_{n+\frac{1}{2}}\right)$$
$$\mathbf{x}^*_{n+1} = \mathbf{x}_n + h\mathbf{f}^{**}_{n+\frac{1}{2}}. \qquad \textit{Euler}$$

Fourth Pass

$$\mathbf{f}^*_{n+1} = \mathbf{f}\left(\mathbf{x}^*_{n+1}, \mathbf{u}_{n+1}\right)$$
$$\begin{aligned}\mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{h}{3}\left[\frac{\mathbf{f}_n + \mathbf{f}^*_{n+\frac{1}{2}}}{2} + \frac{\mathbf{f}^*_{n+\frac{1}{2}} + \mathbf{f}^{**}_{n+\frac{1}{2}}}{2} + \frac{\mathbf{f}^{**}_{n+\frac{1}{2}} + \mathbf{f}^*_{n+1}}{2}\right] \\ &= \mathbf{x}_n + \frac{h}{6}\left(\mathbf{f}_n + 2\mathbf{f}^*_{n+\frac{1}{2}} + 2\mathbf{f}^{**}_{n+\frac{1}{2}} + \mathbf{f}^*_{n+1}\right)\end{aligned}$$
$$e_\tau = \frac{1}{120}h^4.$$

Note, this method is not real-time compatible.

2.4.8. Stability

Stability is conveniently discussed in relation to simulating a stable first-order system with eigenvalue λ. All of the methods presented will be stable if the quantity λh is *small enough*. By stable, it is meant that the solution to the dynamical system will be stable. Errors in the solution may still exist. However, *small enough* is a relative term and is different for each method. If the eigenvalues are known, one can calculate the bound on stability exactly. In general, certain statements can be made regarding the algorithms presented.

- Trapezoidal integration is always stable.
- For single pass methods, the stability boundary shrinks with increasing accuracy.
- For Runge-Kutta methods, the stability boundary expands with increasing accuracy.
- Runge-Kutta methods have larger stability boundaries than single pass methods (approximately 2 to 5 times larger).
- All methods may stabilize an unstable system, that is a system with $\lambda < 0$ (especially AB-3, RK-4).

2.4.9. Z-transform

Any control law implemented on a digital computer must be formed as a difference equation. One approach is to design the controller in the discrete time domain using, perhaps, discrete time equivalents of the root-locus or Bode plots. Alternatively, the continuous domain (or Laplace) version of a controller can be converted to a difference equation. In either case, the Z-transform is introduced and its use in converting transfer functions to difference equations is presented. It will be assumed that the system is being sampled at a fixed interval.

This section is presented with the following considerations:

- Restricted to linear sampled-data systems.
- Requires equally spaced time steps between samples.

Definition 2.15. *The Z-transform of the data sequence* $\{f_n\}$ *is*

$$Z\{f_n\} = F^*(z) = \sum_{n=0}^{\infty} f_n z^{-n}$$

where z *is a complex variable and* $\{f_n\} = \{f_0, f_1, f_2, \dots\}$.

For data sequences of exponential functions, its Z-transform can be expressed in closed form. Consider the exponential function $f(t) = e^{\sigma t}$. The corresponding data sequence is

$$\{f_n\} = \{e^{\sigma nh}\} = \{(e^{\sigma h})^n\} = \{A^n\}$$

where $A = e^{\sigma h}$. Using the definition of the Z -transform results in the series

$$F^*(z) = \sum_{n=0}^{\infty} A^n z^{-n} = (1 - Az^{-1})^{-1} = \frac{z}{z - A}. \tag{2.17}$$

This can be shown by applying the binomial theorem to the right side of Eq. (2.17)

$$(1 - Az^{-1})^{-1} = z^0 + Az^{-1} + A^2z^{-2} + \dots .$$

Now, whenever a Z-transform of a data sequence has the term of the form

$$F^* = Z\{f_n\} = \frac{z}{z - A},$$

the corresponding time sequence is exponential of the form $\{f_n\} = \{e^{\sigma nh}\}$ where $\sigma = (1/h) \cdot ln(A)$.

Example 2.6. Consider the system $\dot{x} = \lambda x + u(t)$.

Solution
Simulate this system using Euler integration to get

$$x_{n+1} = x_n + hf_n$$

where $f_n = \lambda x_n + u_n$.
The resulting difference equation is

$$x_{n+1} = x_n + h(\lambda x_n + u_n) = (1 + \lambda h)x_n + hu_n.$$

Next, take the Z-transform of this system. Considering the left side first gives

$$\begin{aligned} Z\{x_{n+1}\} &= x_1 + z^{-1}x_2 + z^{-2}x_3 + \dots \\ &= z(x_0 + z^{-1}x_1 + z^{-2}x_2 + \dots) - zx_0 \\ &= zX^*(z) - zx_0. \end{aligned}$$

The Z-transformed system with initial conditions becomes

$$zX^*(z) - zx_0 = (1 + \lambda h)X^*(z) + hU^*(z).$$

Solving for $X^*(z)$ gives

$$X^*(z) = \frac{zx_0}{z - 1 - \lambda h} + \frac{h}{z - 1 - \lambda h}U^*(z). \tag{2.18}$$

Note 2.7. The first term is dependent on x_0 and is the discrete analog of the zero-input solution for continuous time systems. The second term is the discrete analog of the zero-state solution for continuous time systems.

Now, focus on the zero-input portion. Recall, the zero-state portion written as a discrete-time transfer function

$$X^*(z) = \frac{zx_0}{z - 1 - \lambda h}.$$

The term on the right has the familiar form of $z/(z - A)$ where $A = 1 + \lambda h$. Therefore, the resulting time sequence is an exponential sequence of

$$x_0\{(e^{\sigma h})^n\}$$

where

$$\sigma = \frac{1}{h} ln(1 + \lambda h).$$

Expanding $ln(1 + \lambda h)$ in a Taylor series about $\lambda h = 0$ gives

$$\begin{aligned} \sigma &= \frac{1}{h}\left(\lambda h - \frac{1}{2}\lambda^2 h^2 + \ldots\right) \\ &\approx \lambda - \frac{1}{2}\lambda^2 h. \end{aligned}$$

The error in the approximate eigenvalue σ is

$$e_\lambda = \frac{\sigma - \lambda}{\lambda} \approx -\frac{1}{2}\lambda h,$$

which was stated previously for Euler integration. (see the error expression for Eq. 2.16).

Thus, the transient solution of the Euler integration based simulation, of the original linear system, contains a transient portion (the zero-input response) that behaves as a decaying exponential. However, the time constant of that exponential (the eigenvalue) is different from the true system by e_λ. For stability reasons, this is the case for $\lambda h \ll 1$. The comforting feature of this analysis is that as the time step is made smaller the simulated eigenvalue approaches the true eigenvalue.

2.4.10. Unit data-point integrator

It is time to focus on the zero-state portion of Eq. (2.18)

$$\frac{X^*(z)}{U^*(z)} = \frac{h}{z - 1 - \lambda h}.$$

First, the unit data point input is defined as

$$u_0 = 1, u_1 = 0, \quad u_2 = 0, \ldots .$$

Its Z-transform is simply

$$U^*(z) = 1.$$

The zero-state, unit data-point response of a system will be denoted as $H^*(z)$ and it is the discrete analog of the impulse response for continuous time systems. For this system,

$$H^*(z) = \frac{h}{z - 1 - \lambda h}.$$

Now, the unit data-point integrator response $H_I^*(z)$ can be discussed in detail. Each integration method discussed earlier has a unique $H_I^*(z)$ determined by considering the simple linear system

$$\dot{x} = u(t).$$

For Euler integration, the difference equation for this system is

$$x_{n+1} = x_n + hu_n.$$

Taking the Z-transform gives

$$\frac{X^*(z)}{U^*(z)} = H_I^*(z) = \frac{h}{z - 1}.$$

For another example, consider Trapezoidal integration

$$x_{n+1} = x_n + \frac{1}{2}h(u_n + u_{n+1}).$$

Taking the Z-transform gives

$$zX^*(z) = X^*(z) + \frac{1}{2}h\left(U^*(z) + zU^*(z)\right),$$

or

$$\frac{X^*(z)}{U^*(z)} = H_I^*(z) = \frac{z + h}{2(z - 1)}.$$

This can be used to implement digital filters and control laws using digital computers. Consider the closed loop system in Figure 2.4 where the plant is denoted by $G(s)$ and some linear compensator is denoted by $H(s)$. If the plant is real (i.e., this is an experiment, not a simulation), one needs to implement the compensator digitally based on current inputs $e(t)$, and outputs $u(t)$. If the plant is not real (i.e., for the simulation case), typically, one would simulate the plant using a sample rate sufficiently fast to capture the relevant dynamics. However, the compensator would use a different sample rate (typically slower) consistent with the hardware that may eventually be used to test the system. In either case, a discrete time representation of the compensator is created.

For zero-state analysis, the Laplace operator can be treated as a differential operator. Thus, the operator s is replaced with the inverse of the unit data-point

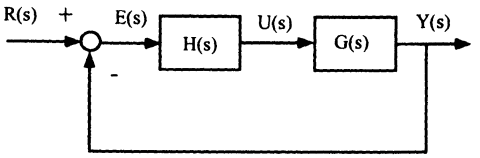

Figure 2.4. Closed loop system

response for the chosen integration method. The result is a digital implementation of the compensator.

For example, consider the compensator

$$\frac{U(s)}{E(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}.$$

For the sake of computation speed, Euler integration is chosen. Its unit data-point response is

$$H_I^*(z) = \frac{h}{z-1}.$$

To obtain the discrete implementation of the compensator, replace all occurrences of s with the inverse of the unit data-point response, or

$$s \text{ becomes } \frac{z-1}{h}.$$

The result is

$$\frac{U^*(z)}{E^*(z)} = \frac{\omega_n^2 h^2}{z^2 + (2\zeta\omega_n h - 2)z + (\omega_n^2 h^2 - 2\zeta\omega_n h + 1)}.$$

Taking the inverse Z-transform gives

$$u_{n+2} + (2\zeta\omega_n h - 2)u_{n+1} + (\omega_n^2 h^2 - 2\zeta\omega_n h + 1)u_n = \omega_n^2 h^2 e_n,$$

which is easily implemented digitally.

2.4.11. State-space methods

Consider the system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x}.$$

The solution to this system has been previously developed in Section 2.3.5. as

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}_0 + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t).$$

Now, consider the input $\mathbf{u}(t)$ sampled by zero-order hold such that

$$\mathbf{u}(t) = \mathbf{u}_n \quad nh \leq t < h(n+1).$$

The solution to this system is

$$\begin{aligned}\mathbf{x}_{n+1} &= e^{\mathbf{A}(n+1)h}\mathbf{x}_0 + \int_0^{h(n+1)} e^{\mathbf{A}(nh+h-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau \\ &= e^{\mathbf{A}h}\left[e^{\mathbf{A}nh}\mathbf{x}_0 + \int_0^{nh} e^{\mathbf{A}(nh-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau\right] + \int_{nh}^{h(n+1)} e^{\mathbf{A}(nh+h-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau.\end{aligned} \tag{2.19}$$

The term in brackets (see Eq. 2.19) is $\mathbf{x}_n$ and the input $\mathbf{u}(\tau)$ is constant. This results in

$$\mathbf{x}_{n+1} = e^{\mathbf{A}h}\mathbf{x}_n + \left[\int_0^h e^{\mathbf{A}\alpha}d\tau\right]\mathbf{B}\mathbf{u}_n.$$

This can be written in the same form as the original dynamical equation

$$\mathbf{x}_{n+1} = \widehat{\mathbf{A}}\mathbf{x}_n + \widehat{\mathbf{B}}\mathbf{u}_n$$
$$\mathbf{y}_n = \widehat{\mathbf{C}}\mathbf{x}_n$$

where

$$\widehat{\mathbf{A}} = e^{\mathbf{A}h}$$
$$\widehat{\mathbf{B}} = \left[\int_0^h e^{\mathbf{A}\alpha}d\tau\right]\mathbf{B}$$
$$\widehat{\mathbf{C}} = \mathbf{C}.$$

The resulting data sequence is exact for a zero-order hold input. This technique is powerful for both implementing compensators, and the simulation of the plant (if the plant is linear).

2.5. Calculus of Variations

Many fields of applied mathematics and engineering rely on calculus of variations. These fields include optimization and approximation theories in which system

identification (see Chapter 4) is used to verify system models relative to experimental data. These system models include finite element models that are derived from a so-called weak variational form[10]. Presented in this chapter are the basics of calculus of variations and the development of equations of motion for mechanical and dynamical system models utilizing integral/energy methods. Since calculus of variations is the generalization of calculus of variables, the discussion will begin with calculus of variables.

2.5.1. Calculus of Several Variables

Because most realistic calculus of variations problems are transformed into calculus of several variables problems to be solved numerically, the calculus of several variables is useful and practical. This procedure is called parameter optimization and is referred to as a suboptimal solution because the space of possible solutions is constrained by the choice of approximation functions (for example finite elements). The goal of parameter optimization is to pick a scalar performance function to minimize and thus, determine the optimal parameter values. First, a scalar function of two independent variables is presented. Consider,

$$J = F(x_1, x_2)$$

where F has continuous first and second derivatives. The extrema of J are determined by taking the first partial derivatives and equating them to zero, or

$$\frac{\partial J}{\partial x_1} = F_{x_1} = 0$$
$$\frac{\partial J}{\partial x_2} = F_{x_2} = 0.$$

The result is a set of simultaneous equations that must be solved for the parameters (x_1, x_2). If these equations are nonlinear, then a Newton root solver is needed[11]. Therefore, for the benefit of the reader, a short description of Newton's method is presented.

Newton's Method

Solve the equation (or system of n-equations) $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ for $\mathbf{x}$ (n-unknowns). Expand $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ in a Taylor's series to first-order and fashion an interative scheme

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}_0) + \mathbf{f}_{\mathbf{x}_0}(\mathbf{x}_0)[\mathbf{x} - \mathbf{x}_0] + H.O.T. = \mathbf{0}.$$

If one neglects the higher-order terms (H.O.T.), one makes the assumption that the function(s) one is (are) trying to solve is (are) locally linear and that one can solve for a new $\mathbf{x}$ that is closer to the actual solution of $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. Specifically, the mathematical formulation of this iterative scheme is

$$\mathbf{f}(\mathbf{x}_{i+1}) = \mathbf{f}(\mathbf{x}_i) + \mathbf{f}_{\mathbf{x}_i}(\mathbf{x}_i)\,[\mathbf{x}_{i+1} - \mathbf{x}_i] \approx \mathbf{0}$$
$$\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{f}_{\mathbf{x}_i}^{-1}(\mathbf{x}_i)\mathbf{f}(\mathbf{x}_i),$$

which is presented graphically in Figure 2.5 for several iterations. Newton's methods

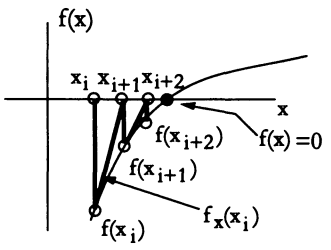

Figure 2.5. Newton's method iterations

works best on monotonically increasing or decreasing functions. Additionally, the iteration will converge to the correct $\mathbf{x}$ if the function does not experience changes in curvature.

Now, a set of linear equations are solved directly. Consider

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

or

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}.$$

Once these equations are solved, the type of extrema can be determined. One is typically only interested if the extrema is a minimum. A minimum exist if

$$F_{x_1x_1} > 0 \tag{2.20}$$

$$F_{x_1x_1}F_{x_2x_2} - F_{x_1x_2}^2 > 0. \tag{2.21}$$

The generalization of this procedure to many variables is via matrix and vector analysis. The scalar performance function is written as

$$J = F(\mathbf{x}); \qquad \mathbf{x} \text{ is a } (n \times 1) \text{ vector}$$

and the extrema vector function is

$$\frac{\partial J}{\partial \mathbf{x}} = \mathbf{F}_\mathbf{x} = \mathbf{0}; \qquad \mathbf{F}_\mathbf{x} \text{ is a } (n \times 1) \text{ vector.}$$

The generalized minimum check is

$$\frac{\partial^2 J}{\partial \mathbf{x}^2} = \mathbf{H} > 0; \qquad \mathbf{H} \text{ is a } (n \times n) \text{ positive definite matrix}$$

where $\mathbf{H}$ is referred to as the Hessian matrix. The check for a positive definite matrix is performed by determining if

1. All eigenvalues are positive.
2. All leading minors are positive (referred to as Sylvester's Criterion as given in Eq. (2.20) and Eq. (2.21) for two variables).

Example 2.7. Linear Least Squares (LLS)

- Function

$$J = (\mathbf{z} - \mathbf{z}_m)^T \mathbf{W} (\mathbf{z} - \mathbf{z}_m)$$
$$\mathbf{z} = \mathbf{A}\mathbf{x}$$

where

$$\begin{aligned} \mathbf{z}_m &= \text{measurements, } (M \times 1) \text{ vector, } M > n. \\ \mathbf{W} &= \text{positive definite, symmetric weighting matrix.} \\ \mathbf{A} &= (M \times n)\text{matrix.} \\ \mathbf{x} &= \text{states, } (n \times 1)\text{vector.} \end{aligned}$$

- Extrema (Referred to as normal equations with a psuedoinverse)

$$\frac{\partial J}{\partial \mathbf{x}} = 2\mathbf{A}^T\mathbf{W}\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{W}\mathbf{z}_m = \mathbf{0}.$$

Therefore,

$$\mathbf{x} = \left(\mathbf{A}^T\mathbf{W}\mathbf{A}\right)^{-1} \mathbf{A}^T\mathbf{W}\mathbf{z}_m.$$

- Check for a minimum

$$\frac{\partial^2 J}{\partial \mathbf{x}^2} = \mathbf{A}^T\mathbf{W}\mathbf{A} > 0.$$

Example 2.8. Check for positive definiteness

1. Positive Eigenvalues

$$\mathbf{H} = \begin{bmatrix} 3 & 3 \\ 3 & 4 \end{bmatrix}$$
$$(\lambda\mathbf{I} - \mathbf{H}) = \begin{bmatrix} (\lambda - 3) & -3 \\ -3 & (\lambda - 4) \end{bmatrix}$$
$$det(\lambda\mathbf{I} - \mathbf{H}) = \lambda^2 - 7\lambda + 3$$
$$\lambda_{1,2} > 0.$$

2. Positive Leading Minors

$$\mathbf{H} = \begin{bmatrix} 3 & 3 \\ 3 & 4 \end{bmatrix}$$
$$M_{11} = |3| > 0$$
$$M_{22} = \begin{vmatrix} 3 & 3 \\ 3 & 4 \end{vmatrix}$$
$$= 3 > 0.$$

As a side note, a local minimum check can be performed on each extremum by evaluating $J = F(\mathbf{x})$ in a region near the extremum, or

$$J(\mathbf{x}_E) < J(\mathbf{x}_E + \Delta\mathbf{x})$$
$$J(\mathbf{x}_E) < J(\mathbf{x}_E - \Delta\mathbf{x}).$$

This check is useful for complicated nonlinear problems where the Hessian matrix is not readily available. This is the case for most real world applications.

2.5.2. Lagrange Multipliers

When not all of the variables are independent, Lagrange multipliers provide a straightforward method to solve for extrema. For example, it may be difficult or impossible to eliminate the dependent variables, or the system is easier to model with the dependent variables. The idea is to include additional unknown variables and additional independent equations that are the constraint equations to the minimization problem. Swokowski[9] presents a good basic explanation of Lagrange multipliers. The calculus of many variables with several constraint equations is a straight forward problem. Begin with a function

$$J = F(\mathbf{x}); \text{ where } \mathbf{x} \text{ is a } (n \times 1) \text{ vector,}$$

with contraints

$$\mathbf{g}(\mathbf{x}) = \mathbf{0}; \text{ where } \mathbf{g} \text{ is a } (m \times 1) \text{ vector and } m < n.$$

Define the Lagrange multipliers as

$$\boldsymbol{\lambda} \text{ is a } (m \times 1) \text{ vector.}$$

Create an augmented function

$$J' = F(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}),$$

and the Extrema become

$$\frac{\partial J'}{\partial \mathbf{x}} = \mathbf{0} = \mathbf{F_x} + \mathbf{g_x}^T \boldsymbol{\lambda}$$
$$\frac{\partial J'}{\partial \boldsymbol{\lambda}} = \mathbf{0} = \mathbf{g}(\mathbf{x}).$$

The minimum check is

$$\frac{\partial^2 J'}{\partial \mathbf{x}^2} > 0.$$

An illustrative example is borrowed from Swokowski[9].

Example 2.9. If $f(x, y, z) = 4x^2 + y^2 + 5z^2$, find the point on the plane $2x + 3y + 4z = 12$ at which $f(x, y, z)$ has its least value.

Solution

Find the minimum value of $f(x, y, z)$ subject to the constraint $g(x, y, z) = 2x + 3y + 4z - 12 = 0$. From

$$w = f(x, y, z) + \lambda g(x, y, z),$$

let

$$w = 4x^2 + y^2 + 5z^2 + \lambda(2x + 3y + 4z - 12)$$

then

$$w_x = 0, \quad w_y = 0, \quad w_z = 0, \quad w_\lambda = g(x, y, z) = 0.$$

Thus,

$$\begin{aligned} w_x &= 8x + 2\lambda &&= 0 \\ w_y &= 2y + 3\lambda &&= 0 \\ w_z &= 10z + 4\lambda &&= 0 \\ w_\lambda &= 2x + 3y + 4z - 12 = 0. \end{aligned} \tag{2.22}$$

The first three equations give

$$\lambda = -4x = -\frac{2}{3}y = -\frac{5}{2}z.$$

Consequently, for a local extremum

$$y = 6x \quad \text{and} \quad z = \frac{8}{5}x.$$

Substitute into Eq. (2.22) (w_λ) to obtain

$$2x + 18x + \frac{32}{5}x - 12 = 0,$$

or

$$x = \frac{5}{11}.$$

Hence,

$$y = 6\left(\frac{5}{11}\right) = \frac{30}{11},$$

and

$$z = \left(\frac{8}{5}\right)\left(\frac{5}{11}\right) = \frac{8}{11}.$$

It follows that the minimum value occurs at the point

$$\left(\frac{5}{11}, \frac{30}{11}, \frac{8}{11}\right).$$

2.5.3. Relationship Between Calculus of Variables and Calculus of Variations

In the previous sections, were developed a general minimization procedure for parameter optimization that was used to produce the LLS solution. Next, the goal is to generalize this procedure, of minimization for several variables, to the minimization of functionals. A functional is a scalar performance index or cost value that relates a real value to a function of time by integrating the function over time. The question arises, How does the first derivative relate and generalize to the first variation?

First, a function $J(x)$ assigns a real number to each point x in the one-dimensional real space, whereas a functional $J(y)$ assigns a real number to each function y belonging to some class of functions, such as the polynomials. Second, the first derivative of the function $J(x)$ produces an algebraic equation (possibly nonlinear) in x. Whereas, the first variation of the functional $J(y)$ produces a differential equation for $y(t)$. Third, one simple way to relate these two concepts is via the virtual displacement.

A virtual displacement is an imagined infinitesimal change in either a point x, or a function $y(t)$ that is arbitrary other than it must be consistent with the constraints

of the system. The variation of $J(x)$ with respect to a virtual displacement of x defined as δx is

$$\delta J(x) = \frac{\partial J}{\partial x}\delta x$$

that can be extended to Lagrange multipliers as

$$\begin{aligned} J'(\mathbf{x}) &= J(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}) \\ \delta J'(\mathbf{x}) &= \delta J(\mathbf{x}) + \boldsymbol{\lambda}^T \delta \mathbf{g}(\mathbf{x}) \\ &= \left[\frac{\partial J}{\partial \mathbf{x}} + \frac{\partial \mathbf{g}^T}{\partial \mathbf{x}}\boldsymbol{\lambda}\right]\delta \mathbf{x} \end{aligned}$$

where $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ is a holonomic constraint[13].

The first variation of the functional $J(y)$ can be defined with respect to the virtual displacement presented in Figure 2.6 and several conditions. First, the

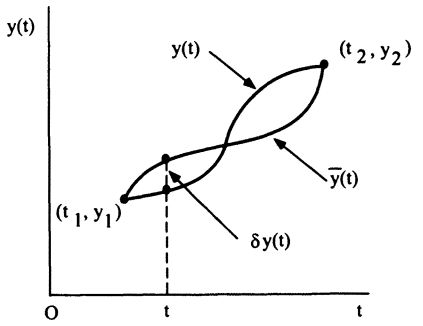

Figure 2.6. Virtual displacement

functional of interest is

$$J(y) = \int_a^b F[t, y(t), \dot{y}(t)]dt$$

where $y(t)$ is a continuously differentiable function, and $F[t, y(t), \dot{y}(t)]$ is typically continuously differentiable up to second-order[10]. Second, this functional is an integral, therefore the first variation gives rise to a weak extremum by way of a differential equation for $y(t)$. A weak extremum means that only small virtual displacements are considered and multiple extrema may exist. For example, the shortest distance between the north and south poles (referred to as conjugate points) is the Great Circle Arc, which has an infinite number of equivalent paths. Third, the first variation is often derived by a Taylor's series expansion up to the first-order,

but this approach will be derived directly with the variational derivative that is based upon the virtual displacement.

One starts the derivation with the virtual displacement

$$\delta y(t) = \bar{y}(t) - y(t); \text{ where } \bar{y}(t) \text{ is the actual path,}$$

and the functional

$$J(y) = \int_a^b F(t, y, \dot{y}) dt,$$

with the appropriate boundary conditions

$$\begin{aligned} y(a) &= A \\ y(b) &= B. \end{aligned}$$

Note, $\bar{y}(t)$ is the solution to this problem that is assumed to be known before one solves the problem. Although this seems backwards, it provides a way to *take the first derivative* of $J(y)$ with respect to $y(t)$ by perturbing $\bar{y}(t)$ or *taking a differential* of $y(t)$. Define the variation as

$$\delta J = \int_a^b \left[\frac{\partial F}{\partial y} \delta y + \frac{\partial F}{\partial \dot{y}} \delta \dot{y} \right] dt = \int_a^b \left[F_y \delta y + F_{\dot{y}} \delta \dot{y} \right] dt = 0.$$

Integrate by parts ($\delta \dot{y}$ depends upon δy)

$$\int_a^b F_{\dot{y}} \delta \dot{y} dt = F_{\dot{y}} \delta y(t) \Big|_a^b - \int_a^b \frac{d}{dt} \left(F_{\dot{y}} \right) \delta y dt.$$

Impose boundary conditions

$$\delta y(a) = \delta y(b) = 0.$$

Therefore,

$$\begin{aligned} F_{\dot{y}} \delta y(t) \Big|_a^b &= 0 \\ \delta J &= \int_a^b \left[F_y - \frac{d}{dt} \left(F_{\dot{y}} \right) \right] \delta y dt = 0. \end{aligned}$$

Since $\delta y(t)$ over the interval $[a, b]$ is arbitrary (*Fundamental Lemma* of the calculus of variations)

$$F_y - \frac{d}{dt}(F_{\dot{y}}) = 0. \tag{2.23}$$

Equation (2.23) is the *Euler-Lagrange Differential Equation.* This is a differential equation for $y(t)$ and the extremal is a function of time. To demonstrate the first variation of a functional and the usefulness of the Euler-Lagrange differential equation, several examples problems are solved.

Example 2.10.

$$J(y) = \int_0^1 (\frac{1}{2}y^2 - y)dt \quad \text{with } y(0) = 1 \text{ and } y(1) = 1.$$

Solution

Apply Euler-Lagrange differential equation.

$$\begin{aligned} F_y &= y - 1 \\ F_{\dot{y}} &= 0 \\ \frac{d}{dt}(F_{\dot{y}}) &= 0 \\ F_y - \frac{d}{dt}(F_{\dot{y}}) &= y - 1 = 0, \end{aligned}$$

thus

$$y = 1.$$

Example 2.11.

$$J(y) = \int_0^1 \dot{y}^2 dt \quad \text{with } y(0) = 1 \text{ and } y(1) = 1.$$

Solution

Apply the Euler-Lagrange differential equation.

$$\begin{aligned} F_y &= 0 \\ F_{\dot{y}} &= 2\dot{y} \\ \frac{d}{dt}(F_{\dot{y}}) &= 2\ddot{y} \\ F_y - \frac{d}{dt}(F_{\dot{y}}) &= 2\ddot{y} = 0 \\ 2\ddot{y} &= 0 \\ 2\dot{y} &= C_1 \\ 2y &= C_1 t + C_2 \end{aligned}$$

Evaluate C_1 and C_2

$$\begin{aligned} t &= 0 \\ y(0) &= 0 \\ C_2 &= 0 \\ t &= 1 \\ y(1) &= 1 \\ C_1 &= 2 \\ y &= t. \end{aligned}$$

Example 2.12.

$$J(y) = \int_0^1 [\dot{y} - t]^2 \, dt \quad \text{with} \quad y(0) = 0 \quad \text{and} \quad y(1) = 1.$$

Solution

Apply the Euler-Lagrange differential equation.

$$\begin{aligned} F_y &= 0 \\ F_{\dot{y}} &= 2(\dot{y} - t) \\ \frac{d}{dt}(F_{\dot{y}}) &= 2(\ddot{y} - 1) \\ F_y - \frac{d}{dt}(F_{\dot{y}}) &= 2(\ddot{y} - 1) = 0 \\ 2(\ddot{y} - 1) &= 0 \\ 2(\dot{y} - t) &= C_1 \\ 2y &= t^2 + C_1 t + C_2 \end{aligned}$$

Evaluate C_1 and C_2

$$\begin{aligned} t &= 0 \\ y(0) &= 0 \\ C_2 &= 0 \\ t &= 1 \\ y(1) &= 1 \\ C_1 &= 1 \\ y &= \frac{1}{2}(t^2 + t). \end{aligned}$$

2.6. Hamilton's Principle & Lagrange's Equations

Typically, the derivation of equations of motion for a system proceeds down one of two paths. The first path uses Newton's equations of motion, which is referred

to as a mechanics-based approach. The second path utilizes calculus of variations by defining a functional, which is based on the system energy and is termed an *integral/energy method.* The second path is presented along with the advantages of integral/energy methods.

The advantages of integral/energy methods include the ability to handle complex systems, rigid and flexible bodies simultaneously, and nonmechanical systems (i.e., fields) in the form of classical mechanics. One example of these complex system solvers is finite elements, which arises as a weak extremum from a properly formed functional. Additional advantages include the fact that the *functional/integral* is invariant to the choice of generalized coordinates. This enables the analyst to choose the best degrees-of-freedom for interpretation (i.e., control design). In fact, this invariance leads to the development of *general recipes* for mathematical system model development, such as Kane's equations[14]. Our discussion will begin with Hamilton's principle, continue with the so-called extended Hamilton's principle, and finish with Lagrange's equations and applicable examples.

2.6.1. Hamilton's Principle

Hamilton's principle is the application of calculus of variations to dynamical systems to derive equations of motion. Hamilton's principle is often referred to as the *Principle of Least/Stationary Action*[10] because the functional is related to system energy. The initial goal was to find the minimum energy state (i.e. least action), but later it was determined that only a stationary point was required to determine the equations of motion. The requirements for a weak and strong extremum (i.e., minimum energy state) are beyond the scope of this discussion, but the interested reader is referred to Gelfand and Fomin[10] for an understandable discussion.

Hamilton's principle is typically stated as[12]

$$J = \int_{t_1}^{t_2} L dt$$

where $L = T - V$ has a stationary value for the correct path of the motion; and T is the kinetic energy, V is the potential energy, J is the action or *action integral*, and L is the Lagrangian.

The variation is

$$\delta J = \delta \int_{t_1}^{t_2} L dt = \int_{t_1}^{t_2} \delta L dt$$

where

$$\delta L = \delta T - \delta V.$$

The choice of L, as the integrand, will be explained in the next section and is only applicable to conservative systems (i.e., all applied forces can be modeled as potential fields). The extended Hamilton's principle will eliminate this limitation and include damping forces. At this point, it is instructive to develop Langrange's equations for conservative systems from the application of the Euler-Lagrange differential equations.

The first variation of the action integral is equivalent to applying the Euler-Lagrange differential equation to each independent variable to produce Lagrange's equations

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0 \quad i = 1, 2, \ldots, n$$

where

$$L = T - V$$
$$q_i = i^{th} \textit{generalized coordinate.}$$

2.6.2. Extended Hamilton's Principle

As eluded to earlier, the extended Hamilton's principle is applicable to both conservative and nonconservative systems. The part of the integrand that is limited to conservative systems is the potential energy, which after the first variation is taken, can be replaced by the equivalent virtual work. To be more specific[13],

$$\delta J = \int_{t_1}^{t_2} [\delta T + \delta W] dt$$

where

$$\delta W = \delta W_C + \delta W_{NC}$$
$$\delta W_C = \text{conservative virtual work}$$
$$\delta W_{NC} = \text{nonconservative virtual work.}$$

Hamilton's principle is a subset when

$$\delta W_C = -\delta V$$
$$\delta W_{NC} = 0.$$

Since the integrand is the sum of kinetic energy and virtual work of the external forces, this formulation shows why these techniques are called integral/energy methods.

It is useful to discuss several specific cases of virtual work for damping and nonconservative external forces, as well as dependent generalized coordinates via

Lagrange multipliers. One way to account for viscous proportional damping is by adding a so-called Rayleigh's dissipation function

$$R = \frac{1}{2}\sum_{i=1}^{n} c_i \dot{q}_i^2$$

that modifies Lagrange's equations to include viscous damping, or

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} + \frac{\partial R}{\partial \dot{q}_i} = 0 \quad i = 1, 2, \ldots, n.$$

This approach can be generalized by writing the nonconservative work in terms of the generalized forces Q_i as

$$\delta W_{NC} = \sum_{i=1}^{n} Q_i \delta q_i,$$

which leads to the definition of the Rayleigh's dissipation function generalized force

$$Q_i = -\frac{\partial R}{\partial \dot{q}_i},$$

and the more general form of Lagrange's equations that includes nonconservative forces

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i \quad i = 1, 2, \ldots, n.$$

An example of a nonconservative external force is the joint torque applied at the hub of a single-link robot. The virtual work is

$$\delta W_{NC} = Q\delta\phi = \tau\delta\phi$$

where τ is the hub torque and ϕ is the rigid body angle between the reference orientation and the robot link (see Figure 2.7).

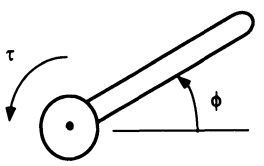

Figure 2.7. Single-link robot

If it is easier to model a system with several dependent generalized coordinates, then Lagrange's equations can be modified through the generalized forces by applying Lagrange multipliers. Given m holonomic constraints among n generalized coordinates

$$f_j(q_1, q_2, \ldots q_n) = 0 \quad j = 1, 2 \ldots, m$$

which can be rewritten as

$$\sum_{i=1}^{n} \frac{\partial f_j}{\partial q_i} \delta q_i = 0.$$

The Lagrange multipliers can be applied in order to augment the Lagrange's equations

$$\sum_{i=1}^{n} \lambda_j \frac{\partial f_j}{\partial q_i} \delta q_i = 0,$$

which can be summed over all j's to obtain

$$Q_i \delta q_i = \sum_{j=1}^{m} \lambda_j \frac{\partial f_j}{\partial q_i} \delta q_i,$$

and

$$\delta W_{NC} = \sum_{i=1}^{n} Q_i \delta q_i = \sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_j \frac{\partial f_j}{\partial q_i} \delta q_i.$$

These generalized forces can be used to account for the additional dependent generalized coordinates in Lagrange's equations. Now, Newton's equations of motion and the overhead gantry robot model can be derived with Lagrange's equations.

2.6.3. Newton's Equations of Motion

One way to show the general utility of the action integral and calculus of variations is to derive Newton's equations of motion. In addition, this exercise will demonstrate that the integral/energy techniques are equivalent to the *force/momentum* techniques. Begin with a point mass that is moving within a potential field. The kinetic energy of the particle is defined as

$$T = \frac{1}{2} m \left(\dot{x}_1^2 + \dot{x}_2^2 + \dot{x}_3^2 \right)$$

where $\mathbf{x} = (x_1, x_2, x_3)^T$ is the position vector. The potential energy of the particle is

$$V = V(x_1, x_2, x_3)$$

where the applied force on the particle due to the potential field is

$$X_i = -\frac{\partial V}{\partial x_i} = F_i = \text{Force}.$$

The final step is to apply Lagrange's equations ($L = T - V$),

$$\frac{\partial L}{\partial \dot{x}_i} = m\dot{x}_i$$

so,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}_i}\right) = m\ddot{x}_i,$$

accordingly

$$\frac{\partial L}{\partial x_i} = -\frac{\partial V}{\partial x_i} = X_i,$$

which results in Newton's equations of motion

$$X_i = m\ddot{x}_i \text{ or } F_i = ma_i.$$

2.6.4. Gantry Robot Model

It is time to finish this chapter section on integral/energy methods with the derivation of the overhead gantry model, which will be used throughout the book. For presentation simplicity, this model is restricted to a two-dimensional plane. Note, a three-dimensional spherical model can be reduced to two orthogonal planar models for swing angles less than 10°. The derivation begins with the rigid body diagram presented in Figure 2.8 where θ is the swing angle, l is the cable length, and m is the payload mass. The kinetic energy is defined as

$$T = \frac{1}{2}m\dot{\mathbf{r}} \cdot \dot{\mathbf{r}} = \frac{1}{2}m\left[\dot{x}^2 + 2\dot{x}\dot{\theta}l\cos\theta + \left(\dot{\theta}l\right)^2\right],$$

and the potential energy is

$$V = mgl(1 - \cos\theta).$$

Now, apply Lagrange's equation ($L = T - V$) to obtain

$$\ddot{\theta} + \frac{g}{l}\sin\theta = -\frac{1}{l}\ddot{x}\cos\theta.$$

This is a nonlinear equation that can be linearized in several ways, such as the perturbation technique used by Feddema[15], et al.. One way is to simply linearize the equation about the equilibrium point $\theta = 0$ (Hanging straight down).

$$\ddot{\theta} + \frac{g}{l}\theta = -\frac{1}{l}\ddot{x}.$$

This is a robust choice (i.e., can tolerate θ up to 10°) and this linearized model will be used in later chapters to design swing-free input shaping maneuvers (see Chapter 5) and adaptive estimators (see Chapter 4).

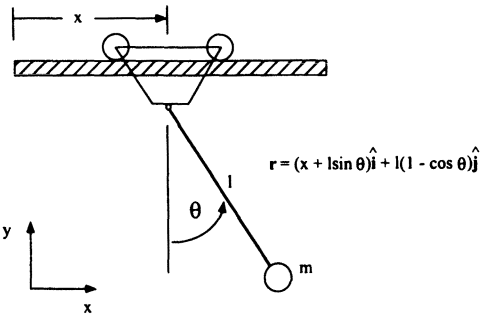

Figure 2.8. Gantry robot kinematic definition

2.7. Analytical Optimization

This section follows the structure of Section 2.5., but it is specifically applied to optimization. The use of the term optimization connotes either minimization or maximization of some useful metric. For robot motion, it can be the judicious use of joint controls to minimize time or energy for a given maneuver, or configuring the robot to maximize its workspace. In order to drive a robot to follow a desired trajectory within hardware constraints (i.e., maximum slew rates, maximum joint angles, etc.), optimization techniques will be used to compute open-loop controls. A dynamical model of the robot derived in some fashion (e.g., Lagrange, or Newton formulations) will define a given output for a given input. Optimization in conjunction with the model will be used to reverse the process—for a given desired output, provide the inputs (open-loop controls) that will accomplish it.

It would be nice to be able to solve a system model to get closed-form solutions for the joint controls as functions of time and known system states. If the mathematics are not too involved, this can be shown to be possible. However, useful flexible robot configurations may result in dynamical systems that may not lend

themselves to closed-form answers. As such, one must resort to numerical procedures that manifest themselves as techniques that iteratively converge to a solution. Essentially, a set of numerical values of the joint controls (such as torque values as a function of time) is initially guessed and an iteration scheme is devised to update these values to *approximately* follow a prescribed trajectory and satisfy any system constraints.

This discussion of optimization techniques will begin with a review of some mathematical background, progress to methods *for* and problems amenable *to* analytical solutions, and finally arrive at the general approach to complex dynamical systems—numerical optimization[16–19].

2.7.1. Preliminaries

Because the systems may require numerous states to describe the dynamical variables (i.e., accelerations, and velocities) at different locations on the robot, vector and matrix counterparts to familiar scalar concepts are required.

Differentiation of a scalar f with respect to vector $\mathbf{x}$ is

$$df = \mathbf{f}_{\mathbf{x}} d\mathbf{x}$$

where

$$\mathbf{f}_{\mathbf{x}} = [f_{x_1} \rightarrow f_{x_n}]$$
$$d\mathbf{x} = [dx_1 \rightarrow dx_n]^T .$$

Differentiation of a column vector $\mathbf{f}$ with respect to a vector $\mathbf{x}$ is

$$d\mathbf{f} = \mathbf{F}_{\mathbf{x}} d\mathbf{x}$$

where

$$\mathbf{F}_{\mathbf{x}} = \begin{bmatrix} f_{1_{x_1}} & \cdots & f_{1_{x_n}} \\ \vdots & & \vdots \\ f_{n_{x_1}} & \cdots & f_{n_{x_n}} \end{bmatrix}$$
$$d\mathbf{x} = [dx_1 \rightarrow dx_n]^T .$$

Differentiation of a row vector $\mathbf{f}$ with respect to a vector $\mathbf{x}$ is

$$d\mathbf{f} = d\mathbf{x}^T \mathbf{F}_{\mathbf{x}}^T .$$

2.7.2. Positive Definite Matrices

A positive definite matrix, as described in Section 2.5.1., is a recurring construct in optimization because of its linkage to quadratic (concave up) surfaces that have

minimums. Its counterpart, the negative definite matrix, is used to represent a concave down surface (containing a maximum).

Positive definite matrix $\mathbf{A}$ is positive definite if for any vector $\mathbf{x} \neq \mathbf{0}$, $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$. The criterion for positive definiteness are as follows:

1. Nonsingular
2. All eigenvalues are positive
3. All principle minors have positive determinants
4. $\mathbf{A}^{-1}$ is positive definite

Positive semi-definite matrix $\mathbf{A}$ is positive semi-definite if for any vector $\mathbf{x} \neq \mathbf{0}$, $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$.

2.7.3. Simple Parameter Minimization

For a scalar function ϕ, which is a function of a vector $\mathbf{x}$ of n-parameters, the conditions for a minimum as described in Section 2.5.1. are

$$\phi_{\mathbf{x}} = \left.\frac{d\phi}{d\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_{min}} = \mathbf{0}$$

$$\phi_{\mathbf{xx}} = \frac{d^2\phi}{d\mathbf{x}^2} > 0.$$

Figure 2.9 presents a graphical illustration of a concave up scalar function. These same conditions can also be used to produce a maximum, whereby one would be *minimizing* $-\phi_{\mathbf{x}}$.

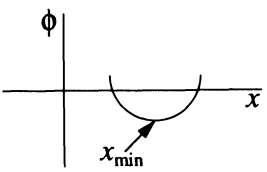

Figure 2.9. Graphic illustration of concave up minimum of $\phi(x)$

For minimization of a scalar function $\phi(\mathbf{x})$ subject to a vector of m-constraints $\Psi(\mathbf{x}) = \mathbf{0}$, an augmented scalar function $G(\mathbf{x}) = \phi(\mathbf{x}) + \nu^T\Psi(\mathbf{x})$ is formed in an analogous way to Section 2.5.2. where ν is a vector of constant Lagrange multipliers.

Conditions for a minimum are

$$G_{\mathbf{x}}|_{\mathbf{x}=\mathbf{x}_{\min}} = \phi_{\mathbf{x}} + \boldsymbol{\nu}^T \boldsymbol{\Psi}_x = 0 \tag{2.24}$$

$$\boldsymbol{\Psi}(\mathbf{x}) = \mathbf{0} \tag{2.25}$$

$$\mathbf{G}_{\mathbf{xx}} > 0. \tag{2.26}$$

Equations (2.24) and (2.25) generate $n + m$ necessary conditions to solve for the vector components of $\mathbf{x}$ and $\boldsymbol{\nu}$. The final condition Eq. (2.26) stipulates the sufficient condition of positive definiteness at the minimum. Figure 2.10 presents a graphical interpretation of the constrained minimum of $\phi(\mathbf{x})$.

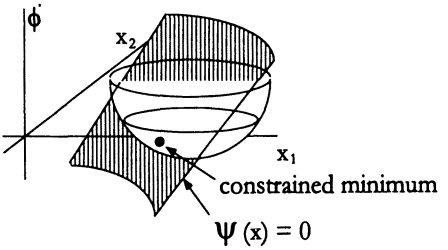

Figure 2.10. Constrained minimum of $\phi(\mathbf{x})$

2.7.4. Variational Conditions for Unconstrained Scalar Minimization

Now, it is time to start the process of extending parameter optimization to functional optimization in a manner that is analogous to the virtual displacement. Assume that one wishes to test that a scalar $\widehat{x}$ minimizes a scalar unconstrained function $\phi(\widehat{x})$ (see Figure 2.11). One can expand ϕ in a Taylor series about this test minimum to describe an admissible comparison point $\widehat{x} + \delta x$.

$$\phi(\widehat{x} + \delta x) = \phi(\widehat{x}) + \left.\frac{\partial \phi}{\partial x}\right|_{\widehat{x}} [\widehat{x} + \delta x - \widehat{x}] + \left.\frac{\partial^2 \phi}{\partial x^2}\right|_{\widehat{x}} \frac{[\widehat{x} + \delta x - \widehat{x}]^2}{2!} + \left.\frac{\partial^3 \phi}{\partial x^3}\right|_{\widehat{x}} \frac{[\widehat{x} + \delta x - \widehat{x}]^3}{3!} + \ H.O.T.$$

If $\widehat{x}$ is indeed the minimum, then the difference $\Delta\phi = \phi(\widehat{x} + \delta x) - \phi(\widehat{x})$ must be positive for an arbitrary δx. Examining the first term in

$$\Delta\phi = \phi_x|_{\widehat{x}} \delta x + \phi_{xx}|_{\widehat{x}} \frac{[\delta x]^2}{2!} + \phi_{xxx}|_{\widehat{x}} \frac{[\delta x]^3}{3!} + \ H.O.T. \tag{2.27}$$

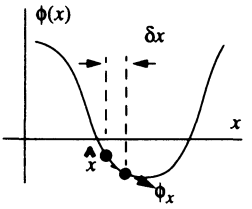

Figure 2.11. Minimization test for an unconstrained scalar function

reveals $\phi_x|_{\widehat{x}}\delta x$ can switch sign depending on which direction one proceeds. Thus, for an arbitrary $\delta x, \phi_x|_{\widehat{x}}$ must be zero at a minimum to ensure that $\phi(\widehat{x})$ cannot be decreased any further. This is the *first variation* condition for a minimum (extremal).

Once the first variation $\phi_x|_{\widehat{x}} = 0$ has been stipulated, what's left? Rewrite Eq. (2.27) as

$$\Delta\phi = \phi_{xx}|_{\widehat{x}}\frac{[\delta x]^2}{2!} + \phi_{xxx}|_{\widehat{x}}\frac{[\delta x]^3}{3!} + \ H.O.T.,$$

and upon further examination the first remaining term $\phi_{xx}|_{\widehat{x}}\,[\delta x]^2\,/2!$ will be positive (or at least not allow a decrease in $\Delta\phi$) if $\phi_{xx}|_{\widehat{x}} \geq 0$. This is the *second variation* condition. One can go on to develop other variations (think of the functions $\phi(x) = x^3, x^4$), but usually only the first two are important.

2.7.5. A Multivariable View of the Variational Conditions for Unconstrained Scalar Minimization

Now, it is time to minimize a scalar function ϕ as a function of a vector of independent variables $\mathbf{x}$. The change in ϕ about a test minimum $\widehat{\mathbf{x}}$ is

$$\phi(\widehat{\mathbf{x}}+\delta\mathbf{x}) - \phi(\widehat{\mathbf{x}}) = [\phi_{x_1}\ \ \phi_{x_2}\ldots\phi_{x_N}]\Big|_{\widehat{\mathbf{x}}}\begin{bmatrix}\delta x_1\\ \delta x_2\\ \vdots\\ \delta x_N\end{bmatrix}$$
$$+\frac{1}{2!}[\delta x_1\ldots\delta x_N]\begin{bmatrix}\phi_{x_1x_1} & \cdots & \phi_{x_1x_N}\\ \vdots & \ddots & \vdots\\ \phi_{x_Nx_1} & \cdots & \phi_{x_Nx_N}\end{bmatrix}\Bigg|_{\widehat{\mathbf{x}}}\begin{bmatrix}\delta x_1\\ \vdots\\ \delta x_N\end{bmatrix} + \ H.O.T.. \quad (2.28)$$

Examining the first term on the right hand side of Eq. (2.28) for an arbitrary vector of variations $\delta\mathbf{x}$ reveals the respective ϕ_{x_i} must be zero *at a minimum* to ensure that no additional decrease in $\Delta\phi$ results.

The first variation necessary conditions are $\phi_{x_i}|_{\widehat{\mathbf{x}}} = \mathbf{0}$ for $i = 1, \ldots, N$. What's left? Rewriting Eq. (2.28) as

$$\begin{aligned}\Delta\phi &= \phi(\widehat{\mathbf{x}} + \delta\mathbf{x}) - \phi(\widehat{\mathbf{x}}) \\ &= \frac{1}{2!}[\delta x_1 \ldots \delta x_N] \begin{bmatrix} \phi_{x_1x_1} & \cdots & \phi_{x_1x_N} \\ \vdots & \ddots & \vdots \\ \phi_{x_Nx_1} & \cdots & \phi_{x_Nx_N} \end{bmatrix}\Bigg|_{\widehat{\mathbf{x}}} \begin{bmatrix} \delta x_1 \\ \vdots \\ \delta x_N \end{bmatrix} + \ H.O.T.\end{aligned}$$

provides the second variation sufficient condition $[\phi_{x_ix_i}]|_{\widehat{\mathbf{x}}} \geq 0$ (positive semi-definite).

Example 2.13. Minimize the following function and show necessary and sufficient conditions:

$$\phi = x^2 + y^2 + (1 - x + 2y)^2$$

Solution

$$\begin{aligned}\phi_x &= 2x - 2(1 - x + 2y) \\ \phi_y &= 2y + 4(1 - x + 2y) \\ \phi_{xx} &= 4 \\ \phi_{yy} &= 10 \\ \phi_{xy} = \phi_{yx} &= -4\end{aligned}$$

Solving for $\phi_x = \phi_y = 0$ gives

$$\begin{aligned}x - y &= 0.5 \\ y - 0.4x &= -0.4\end{aligned}$$

so,

$$\begin{aligned}x &= \frac{1}{6} \\ y &= -\frac{1}{3}.\end{aligned}$$

Thus, the eigenvalues are

$$[\phi_{xx}] = \frac{14 \pm \sqrt{14^2 - 4 \times 24}}{2} = 12, 2$$

Figure 2.12 graphically shows the analysis is correct.

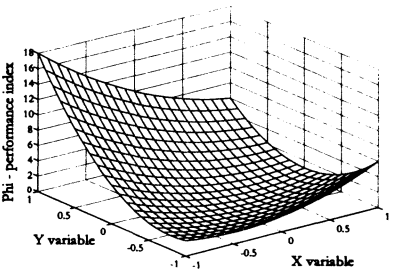

Figure 2.12. Example 2.13 results

2.7.6. Variational Approach to Constrained Parameter Minimization

For an augmented performance index $G(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1, \mathbf{x}_2) + \boldsymbol{\nu}^T \boldsymbol{\Psi}(\mathbf{x}_1, \mathbf{x}_2)$ where the n length vector $\mathbf{x}$ is composed of the $n - m$ independent variables $\mathbf{x}_1$ and m dependent variables $\mathbf{x}_2$ subject to the m constraints $\boldsymbol{\Psi}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{0}$. The variation of this is

$$\delta G(\mathbf{x}) = \delta G_1 + \delta G_2 + \quad H.O.T.$$

where

$$\delta G_1 = \begin{bmatrix} \phi_{\mathbf{x}_1} + \boldsymbol{\nu}^T \boldsymbol{\Psi}_{\mathbf{x}_1} & \phi_{\mathbf{x}_2} + \boldsymbol{\nu}^T \boldsymbol{\Psi}_{\mathbf{x}_2} \end{bmatrix} \begin{bmatrix} \delta \mathbf{x}_1 \\ \delta \mathbf{x}_2 \end{bmatrix},$$

and $\boldsymbol{\nu}$ = vector of constant constraint multipliers, and

$$\delta G_2 = \frac{1}{2} \begin{bmatrix} \delta \mathbf{x}_1 & \delta \mathbf{x}_2 \end{bmatrix} \begin{bmatrix} \phi_{\mathbf{x}_1 \mathbf{x}_1} + \boldsymbol{\nu}^T \boldsymbol{\Psi}_{\mathbf{x}_1 \mathbf{x}_1} & \phi_{\mathbf{x}_1 \mathbf{x}_2} + \boldsymbol{\nu}^T \boldsymbol{\Psi}_{\mathbf{x}_1 \mathbf{x}_2} \\ \phi_{\mathbf{x}_1 \mathbf{x}_2} + \boldsymbol{\nu}^T \boldsymbol{\Psi}_{\mathbf{x}_1 \mathbf{x}_2} & \phi_{\mathbf{x}_2 \mathbf{x}_2} + \boldsymbol{\nu}^T \boldsymbol{\Psi}_{\mathbf{x}_2 \mathbf{x}_2} \end{bmatrix} \begin{bmatrix} \delta \mathbf{x}_1 \\ \delta \mathbf{x}_2 \end{bmatrix}. \tag{2.29}$$

For an extremal, the first variation δG_1 must vanish. Because of the constraints, the variations $\delta \mathbf{x}_2$ are *not free to vary*. Therefore, one can choose the m Lagrange multipliers to make the variation $\phi_{\mathbf{x}_2} + \boldsymbol{\nu}^T \boldsymbol{\Psi}_{\mathbf{x}_2} = \mathbf{0}$.

Since $\delta \mathbf{x}_1$ is free to vary, the only way to make the first variation vanish is to set $\phi_{\mathbf{x}_1} + \boldsymbol{\nu}^T \boldsymbol{\Psi}_{\mathbf{x}_1} = \mathbf{0}$. To make sure the extremal is indeed a minimum, the remaining variation $\delta G(\mathbf{x}) = \delta G_2 \geq 0$. To do this, solve for $\delta \mathbf{x}_2$ in terms of $\delta \mathbf{x}_1$ and substitute into δG_2.

Since $\boldsymbol{\Psi}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{0}$, $\delta \boldsymbol{\Psi} = \boldsymbol{\Psi}_{\mathbf{x}_1} \delta \mathbf{x}_1 + \boldsymbol{\Psi}_{\mathbf{x}_2} \delta \mathbf{x}_2 = \mathbf{0}$, thus $\delta \mathbf{x}_2 = -\boldsymbol{\Psi}_{\mathbf{x}_2}^{-1} \boldsymbol{\Psi}_{\mathbf{x}_1} \delta \mathbf{x}_1$.

Substituting this into Eq. (2.29) gives the quadratic expression for δG_2 as

$$\left(\phi_{\mathbf{x}_1\mathbf{x}_1} + \boldsymbol{\nu}^T\boldsymbol{\Psi}_{\mathbf{x}_1\mathbf{x}_2} - 2\left(\phi_{\mathbf{x}_1\mathbf{x}_2} + \boldsymbol{\nu}^T\boldsymbol{\Psi}_{\mathbf{x}_1\mathbf{x}_2}\right)\left(\boldsymbol{\Psi}_{\mathbf{x}_2}^{-1}\boldsymbol{\Psi}_{\mathbf{x}_1}\right) + \left(\phi_{\mathbf{x}_2\mathbf{x}_2} + \boldsymbol{\nu}^T\boldsymbol{\Psi}_{\mathbf{x}_2\mathbf{x}_2}\right)\left(\boldsymbol{\Psi}_{\mathbf{x}_2}^{-1}\boldsymbol{\Psi}_{\mathbf{x}_1}\right)^2\right)\delta\mathbf{x}_1^2 \geq 0,$$

or

$$\left(\phi_{\mathbf{x}_1\mathbf{x}_1} + \boldsymbol{\nu}^T\boldsymbol{\Psi}_{\mathbf{x}_1\mathbf{x}_2} - 2\left(\phi_{\mathbf{x}_1\mathbf{x}_2} + \boldsymbol{\nu}^T\boldsymbol{\Psi}_{\mathbf{x}_1\mathbf{x}_2}\right)\left(\boldsymbol{\Psi}_{\mathbf{x}_2}^{-1}\boldsymbol{\Psi}_{\mathbf{x}_1}\right) + \left(\phi_{\mathbf{x}_2\mathbf{x}_2} + \boldsymbol{\nu}^T\boldsymbol{\Psi}_{\mathbf{x}_2\mathbf{x}_2}\right)\left(\boldsymbol{\Psi}_{\mathbf{x}_2}^{-1}\boldsymbol{\Psi}_{\mathbf{x}_1}\right)^2\right) \geq 0. \quad (2.30)$$

It is assumed that in Eq. (2.30) the number of constraints assures dimensional compatibility of all terms and the nonsingularity of $\boldsymbol{\Psi}_{\mathbf{x}_2}$.

Example 2.14. Minimize $\phi = x_1 + x_2$ subject to $x_1^2 + x_1x_2 + x_2^2 = 1$ $(n = 2, m = 1)$ by using $\partial/\partial\mathbf{x}\left(x_1 + x_2 + \nu\left(x_1^2 + x_1x_2 + x_2^2 - 1\right)\right) = \phi_{\mathbf{x}} + \boldsymbol{\nu}^T\boldsymbol{\Psi}_{\mathbf{x}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ (see Figure 2.13).

Solution

$$0 = 1 + \nu(2x_1 + x_2)$$
$$0 = 1 + \nu(x_1 + 2x_2).$$

So,

$$x_2 = -2x_1 - \frac{1}{\nu}$$
$$x_1 = -\frac{1}{3\nu} = x_2$$
$$\nu = \pm\frac{1}{\sqrt{3}}.$$

Because decisions are needed, one must test both solutions to find a minimum.

$$\text{for} \quad \nu = \frac{1}{\sqrt{3}} \qquad x_1 = x_2 = -\frac{1}{\sqrt{3}}, \quad \phi = -\frac{2}{\sqrt{3}}$$
$$\text{for} \quad \nu = -\frac{1}{\sqrt{3}} \qquad x_1 = x_2 = \frac{1}{\sqrt{3}}, \quad \phi = \frac{2}{\sqrt{3}}.$$

The first solution is the best. Therefore, one checks the second-order condition $\delta G_2 \geq 0$. After eliminating the dependent variation,

$$\phi_{\mathbf{x}_1\mathbf{x}_1} + \boldsymbol{\nu}^T\boldsymbol{\Psi}_{\mathbf{x}_1\mathbf{x}_2} - 2\left(\phi_{\mathbf{x}_1\mathbf{x}_2} + \boldsymbol{\nu}^T\boldsymbol{\Psi}_{\mathbf{x}_1\mathbf{x}_2}\right)\left(\boldsymbol{\Psi}_{\mathbf{x}_2}^{-1}\boldsymbol{\Psi}_{\mathbf{x}_1}\right) + \left(\phi_{\mathbf{x}_2\mathbf{x}_2} + \boldsymbol{\nu}^T\boldsymbol{\Psi}_{\mathbf{x}_2\mathbf{x}_2}\right)\left(\boldsymbol{\Psi}_{\mathbf{x}_2}^{-1}\boldsymbol{\Psi}_{\mathbf{x}_1}\right)^2 \geq 0.$$

After substitution, this can be reduced to

$$\frac{6\nu\left(x_1^2 + x_1x_2 + x_2^2\right)}{\left(x_1 + 2x_2\right)^2} \geq 0.$$

If the argument $x_1^2 + x_2^2 = r^2$, then $x_1^2 + x_1x_2 + x_2^2 = r^2 \ (1 + \cos\theta \sin\theta) \geq 0$ is applied for any arbitrary r, θ, then $\delta G_2 \geq 0$, if $\nu \geq 0$. Since, $\nu = 1/\sqrt{3}$, this solution is indeed a minimum.

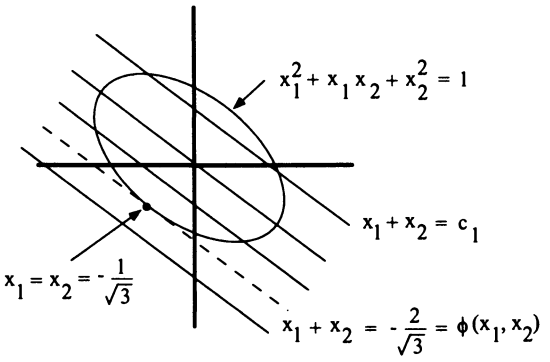

Figure 2.13. Graphical representation of Example 2.14

Some Observations on Example 2.14.

This example demonstrates constrained parameter optimization on a simple problem of low-order (two parameters). The functions were explicit in the parameters and differentiation is straightforward. Note, both parameters have the same order-of-magnitude therefore no scaling in the performance index is required. The multi-valued Lagrange multiplier ν is indicative of the decision nature of optimization problems. Among the constraint lines shown in Figure 2.13, the active constraint (ν) line is tangent to the performance index (ϕ) contour value ellipse. ν has units that make the constraint equation value consistent with those of ϕ.

2.7.7. Optimal Control Problem

The rest of this section presents the application of calculus of variations to optimal control. The goal of optimal control is to derive open-loop and closed-loop, closed-form control laws $u(x,t)$ that do something useful. The painful reality is that few problems will yield nice closed-form solutions. A good place to start is with the fixed final time problem.

Minimize

$$J = \phi(\mathbf{x}_f, t_f) + \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}, t)dt.$$

Subject to

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$$
$$\Psi(\mathbf{x}_f, t_f) = \mathbf{0}; \quad t_0, t_f, \mathbf{x}(t_0) \text{ given}$$

where $\mathbf{x}$ is the n vector of dynamic states, $\mathbf{u}$ is the r vector of controls ($\mathbf{u}(\mathbf{x}, t)$), t is time, and $\mathbf{f}$ is a n vector of first-order O.D.E.'s that describe the dynamical systems.

In this case, the optimal control $\mathbf{u}$ will be of class C^1 (continuous in its first derivatives) and the state $\mathbf{x}$ will be C^2.

Example 2.15. Single-link slewing in the horizontal plane is a typical dynamic model of interest and an optimal control problem (see Figure 2.14).

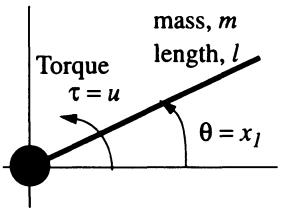

Figure 2.14. Kinematic definition for a single-link slewing in the horizontal plane

With

$$\tau = \frac{ml^2}{3}\ddot{\theta},$$

choose

$$x_1 = \theta$$
$$x_2 = \dot{\theta}$$
$$u = \tau,$$

then the state-space equations are

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = \frac{3u}{ml^2}.$$

A typical performance index is minimum control effort, or

$$J = \frac{1}{2}\int_{t_0}^{t'} u^2 dt.$$

One last detail—the concept of admissible comparison paths—must be discussed before one derives the variational equations. Basically, the concept of admissible comparison paths is equivalent to the virtual displacements discussed in Section 2.5.3..

Optimal paths $\mathbf{x}(t)$ (or *extremals* if one has not applied the second variation conditions) are defined with respect to *admissible comparison paths.* An admissible comparison path is any function $\mathbf{x}_*(t)$ that lies in the neighborhood of the extremal path, satisfies $\mathbf{f}(\mathbf{x}, \mathbf{u}, t) - \dot{\mathbf{x}} = \mathbf{0}$ and the prescribed boundary conditions. A dynamical system is said to be *controllable* if the entire region surrounding the extremal can be filled with admissible comparison paths. Admissible comparison paths (see Figure 2.15) can either be generated by small (weak variation) changes in the state $\delta\mathbf{x}$ and the control $\delta\mathbf{u}$, or by small changes in the state and large changes in the control (strong variations). In the next derivations, only weak variations are

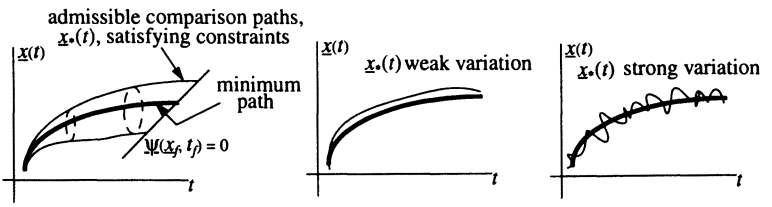

Figure 2.15. Various shapes for the variation of $\mathbf{x}_*(t)$

considered.

2.7.8. Derivation of the First Variation Conditions

The first step is to form the augmented *scalar* performance index J' by attaching the dynamic constraints $\mathbf{f}(\mathbf{x}, \mathbf{u}, t) - \dot{\mathbf{x}}(t) = \mathbf{0}$ and the end point constraints $\boldsymbol{\Psi}(\mathbf{x}_f, t_f) = \mathbf{0}$, with separate vectors of Lagrange multipliers $\boldsymbol{\lambda}$ that are time varying

and $\boldsymbol{\nu}$ that are *constants*, or

$$J' = \phi(\mathbf{x}_f, t_f) + \boldsymbol{\nu}^T \boldsymbol{\Psi}(\mathbf{x}_f, t_f) + \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T(t)(\mathbf{f}(\mathbf{x}, \mathbf{u}, t) - \dot{\mathbf{x}}(t))dt$$

$$= G\left(\mathbf{x}_f, t_f, \boldsymbol{\nu}^T\right) + \int_{t_0}^{t_f} \left(H(\mathbf{x}, \mathbf{u}, t, \boldsymbol{\lambda}) - \boldsymbol{\lambda}^T(t)\dot{\mathbf{x}}(t)\right) dt$$

where

$$G\left(\mathbf{x}_f, t_f, \boldsymbol{\nu}^T\right) = \phi(\mathbf{x}_f, t_f) + \boldsymbol{\nu}^T \boldsymbol{\Psi}(\mathbf{x}_\mathbf{f}, t_f) \tag{2.31}$$

$$H(\mathbf{x}, \mathbf{u}, t, \boldsymbol{\lambda}) = L(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T(t)\mathbf{f}(\mathbf{x}, \mathbf{u}, t). \tag{2.32}$$

$H(\mathbf{x}, \mathbf{u}, t, \boldsymbol{\lambda})$ is called the Hamiltonian and plays a key role in optimal control theory.

Next, take the first variation of J' and set it equal to zero, or

$$\delta J' = \mathbf{G}_{\mathbf{x_f}} \delta \mathbf{x_f} + \int_{t_0}^{t_f} \left[H_\mathbf{x} \delta \mathbf{x} + H_\mathbf{u} \delta \mathbf{u} + H_{\boldsymbol{\lambda}} \delta \boldsymbol{\lambda} - \boldsymbol{\lambda}^T \delta \dot{\mathbf{x}} - \delta \boldsymbol{\lambda}^T \dot{\mathbf{x}}\right] dt = 0$$

$$\delta \dot{\mathbf{x}} = \dot{\mathbf{x}}_* - \dot{\mathbf{x}},$$

$$\delta \mathbf{x} = \mathbf{x}_* - \mathbf{x},$$

$$\delta \dot{\mathbf{x}} = \frac{d}{dt}\delta \mathbf{x}, \text{ and}$$

$$d(\delta \mathbf{x}) = \delta \dot{\mathbf{x}} dt$$

where $\mathbf{x}$ denotes the extremal path and $\mathbf{x}_*$ denotes an admissible comparison path. Integrating the $\delta \dot{\mathbf{x}}$ term by parts yields

$$\int_{t_0}^{t_f} -\boldsymbol{\lambda}^T \delta \dot{\mathbf{x}} dt = \int_{t_0}^{t_f} -\boldsymbol{\lambda}^T d(\delta \mathbf{x}) = -\boldsymbol{\lambda}^T \delta \mathbf{x}\Big|_{t_0}^{t_f} + \int_{t_0}^{t_f} d\boldsymbol{\lambda}^T \delta \mathbf{x}$$

$$\int_{t_0}^{t_f} d\boldsymbol{\lambda}^T \delta \mathbf{x} = \int_{t_0}^{t_f} d\boldsymbol{\lambda}^T \frac{dt}{dt} \delta \mathbf{x} = \int_{t_0}^{t_f} \dot{\boldsymbol{\lambda}}^T \delta \mathbf{x} dt.$$

Substituting into $\delta J'$ and using $H_{\boldsymbol{\lambda}} = \mathbf{f}^T(\mathbf{x}, \mathbf{u}, t)$, $\delta \boldsymbol{\lambda}^T \dot{\mathbf{x}} = \dot{\mathbf{x}}^T \delta \boldsymbol{\lambda}$ (scalar), and $\delta \mathbf{x}(t_0) = \mathbf{0}$ gives

$$\delta J' = \mathbf{G}_{\mathbf{x_f}} \delta \mathbf{x_f} - \boldsymbol{\lambda}^T(t_f) \delta \mathbf{x_f}$$

$$+ \int_{t_0}^{t_f} \left(H_\mathbf{x} \delta \mathbf{x} + H_\mathbf{u} \delta \mathbf{u} + \mathbf{f}^T(\mathbf{x}, \mathbf{u}, t) \delta \boldsymbol{\lambda} + \dot{\boldsymbol{\lambda}}^T \delta \mathbf{x} - \dot{\mathbf{x}}^T \delta \boldsymbol{\lambda}\right) dt = 0.$$

Grouping like terms gives

$$\delta J' = \left(\mathbf{G}_{\mathbf{x_f}} - \boldsymbol{\lambda}^T(t_f)\right) \delta \mathbf{x_f}$$

$$+ \int_{t_0}^{t_f} \left(\left(H_\mathbf{x} + \dot{\boldsymbol{\lambda}}^T\right) \delta \mathbf{x} + H_\mathbf{u} \delta \mathbf{u} + \left[\mathbf{f}^T(\mathbf{x}, \mathbf{u}, t) - \dot{\mathbf{x}}^T\right] \delta \boldsymbol{\lambda}\right) dt = 0.$$

Choose the $\boldsymbol{\lambda}$'s to drive the $\delta\mathbf{x}$ and $\delta\mathbf{x_f}$ variations to zero. Assume that the control variables ($\delta\mathbf{u}'s$) are independent, then the only way for $\delta J'$ to vanish is for $H_{\mathbf{u}} = \mathbf{0}$. The results are known as the *Euler-Lagrange* differential equations (equivalent to Eq. 2.23) that consist of

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, t) && n - \text{state equations} \\ \dot{\boldsymbol{\lambda}} &= -H_{\mathbf{x}}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) && n - \text{costate (multiplier) equations} \\ \mathbf{0} &= H_{\mathbf{u}}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) && r - \text{control equations} \end{aligned} \tag{2.33}$$

The constants of integration are determined from the boundary conditions (B.C.'s)

at t_0	given n-prescribed B.C.'s,	t_0 given
at t_f	$\mathbf{G}_{x_f} = \frac{\partial}{\partial \mathbf{x}} \left(\phi(\mathbf{x}_f, t_f) + \boldsymbol{\nu}^T \boldsymbol{\Psi}(\mathbf{x_f}, t_f)\right)\Big\|_{\mathbf{x}_f, t_f} = \boldsymbol{\lambda}^T(t_f)$	n-natural B.C.'s
	$\boldsymbol{\Psi}(\mathbf{x_f}, t_f) = \mathbf{0}$ p-prescribed B.C.'s,	t_f given.

This structure is referred to as the *two-point-boundary-value-problem* (TPBVP), in that boundary conditions are specified at both t_0 and t_f.

2.7.9. First Integral and a Second Variation Condition

A *first* integral of the TPBVP exists if $L = L(\mathbf{x}, \mathbf{u})$ and $\mathbf{f} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ (i.e., are not explicit functions of time). Differentiating this Hamiltonian form $H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda})$ with respect to t gives

$$\dot{H} = H_{\mathbf{x}}\dot{\mathbf{x}} + H_{\mathbf{u}}\dot{\mathbf{u}} + \dot{\boldsymbol{\lambda}}^T H\boldsymbol{\lambda} = H_{\mathbf{u}}\dot{\mathbf{u}} + \left(H_{\mathbf{x}} + \dot{\boldsymbol{\lambda}}^T\right)\mathbf{f} = 0,$$

on the extremal. This implies that $H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda})$ is a constant on the extremal.

As was the case in parameter optimization, second variations must be checked to ascertain the type of extremal derived from solving the TPBVP. Derivation of the second variation is lengthy and a single condition is stated here[16].

Note 2.8. If the extremal controls $\mathbf{u}$ are assumed to be independent, the Legendre condition for a weak minimum states that the matrix $\mathbf{H_{uu}} > 0$ (i.e., is positive definite).

Example 2.16. Minimize the effort to slew a single-link horizontally (see Figure 2.14) to a given angle in a rest-to-rest maneuver, or minimize

$$J = \frac{1}{2}\int_{t_0}^{t_f} u^2 dt$$

subject to

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{3u}{ml^2} \end{aligned}$$

where

$$x_1, x_2, u = \theta, \dot{\theta}, \tau, \text{ respectively}$$

and with B.C.'s

$$\begin{aligned} t_0, t_f &\text{ given} \\ x(t_0) &= x_2(t_0) = 0 \\ x_1(t_f) &= x_{1_{desired}},\ x_2(t_f) = 0. \end{aligned}$$

Form the Hamiltonian $H(\mathbf{x}, \mathbf{u}, t, \boldsymbol{\lambda})$ and the end condition function $G\left(\mathbf{x_f}, t_f, \boldsymbol{\nu}^T\right)$.

$$\begin{aligned} H &= L + \boldsymbol{\lambda}^T \mathbf{f} = u^2 + \lambda_1 x_2 + \lambda_2 \frac{3u}{ml^2} \\ G &= \boldsymbol{\nu}^T \boldsymbol{\Psi} = \nu_1 (x_1(t_f) - x_{1_{desired}}) + \nu_2 x_2(t_f). \end{aligned}$$

Derive the Euler-Lagrange equations

$$\begin{array}{lll} \dot{\boldsymbol{\lambda}} = -H_{\mathbf{x}}^T & \dot{\lambda}_1 = -(0) & \dot{\lambda}_2 = -(\lambda_1) \\ 0 = 2u + c_1 \lambda_2 & \text{where } c_1 = \dfrac{3}{ml^2} & \text{note } H_{\mathbf{uu}} = 2 > 0 \\ G_{\mathbf{x_f}} = \boldsymbol{\lambda}^T(t_f) & \nu_1 = \lambda_1(t_f) & \nu_2 = \lambda_2(t_f). \end{array}$$

From the costate equations

$$\begin{aligned} \lambda_1 &= \text{const}, \\ \lambda_2 &= -\lambda_1 (t - t_0) + \lambda_2(t_0). \end{aligned}$$

From the extremal condition

$$\begin{aligned} u &= -\frac{c_1}{2} \lambda_2 \\ &= -\frac{c_1}{2} (\lambda_2(t_0) - \lambda_1 (t - t_0)). \end{aligned}$$

From the end conditions

$$\begin{aligned} \lambda_1(t_f) &= \nu_1, \\ \lambda_2(t_f) &= \nu_2 = \lambda_2(t_0) - \lambda_1 T \end{aligned}$$

where

$$T = t_f - t_0.$$

Solve for $\lambda_2(t_0)$

$$\begin{aligned}\lambda_2(t_0) &= \nu_2 + \nu_1 T \\ \lambda_2 &= (\nu_2 + \nu_1 T) - \nu_1(t - t_0).\end{aligned}$$

Integrate $\dot{x}_1, \dot{x}_2$ and apply B.C.'s to get algebraic equations in terms of ν_1, and ν_2

$$\begin{aligned}x_{1_{desired}} &= -\frac{c_1^2}{2}\left(\nu_1\frac{T^3}{3} + \nu_2\frac{T^2}{2}\right) \\ x_{2_f} = 0 &= -\frac{c_1^2}{2}\left(\nu_1\frac{T^2}{2} + \nu_2 T\right),\end{aligned}$$

which yields

$$\begin{aligned}\nu_1 &= -\frac{24x_{1_{desired}}}{c_1^2 T^3} \\ \nu_2 &= \frac{12x_{1_{desired}}}{c_1^2 T^2} \\ u(t) &= -\frac{c_1}{2}\lambda_2 \\ &= -\frac{c_1}{2}\left((\nu_2 + \nu_1 T) - \nu_1(t - t_0)\right) \\ &= \frac{6x_{1_{desired}}}{c_1 T^3}\left((t_f - t) - (t - t_0)\right) \\ &= \tau(t).\end{aligned}$$

Use the open-loop control law (and a lot of algebra) to solve for the state trajectories

$$\dot{x}_2 = c_1 u(t), \text{ therefore, } \quad x_2(t) = \frac{6x_{1_{desired}}}{T^3}(t_f - t)(t - t_0) \qquad = \dot{\theta}(t)$$

$$\dot{x}_1 = x_2, \text{ therefore, } \quad x_1(t) = \frac{6x_{1_{desired}}}{T^3}(t - t_0)^2\left(\frac{t_f - t}{2} + \frac{t - t_0}{6}\right) \quad = \theta(t).$$

Some observations on Example 2.16

- The dynamical system was small—only two states!
- The dynamical system was linear—a major advantage.
- Derivatives could be analytically computed.
- Linearity allowed the system to be completely *integrable.*
- The integral solutions were polynomial functions of t.
- The solution forms allowed easy comparisons.

Reality tends to be vastly different from this example.

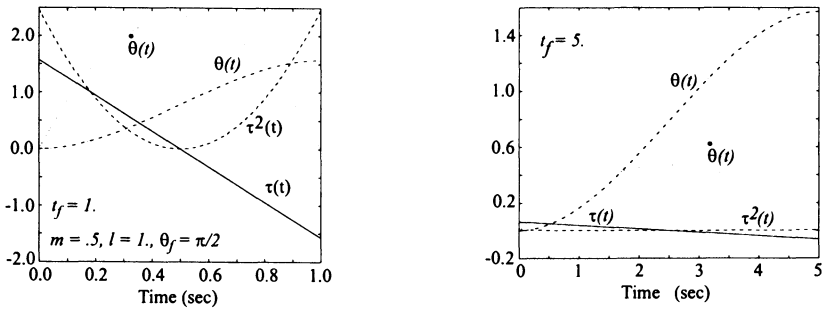

Figure 2.16. Optimal torque and state variable time histories

2.7.10. Free Final Time Problems

The preceding formulation can be extended to the free final time t_f (or independent variable) problem. By taking differentials of J using Liebnitz's rule for differentiation of an integral and using the relationship

$$dx_f + \delta x_f = \dot{x} dt_f$$

to relate total differentials dx_f to variations, a *transversality* condition to be satisfied for arbitrary changes in final time dt_f arises. This condition is given by

$$\frac{\delta G}{\delta t_f} + H(t_f) = 0$$

where G and H are defined in Eqs.(2.31) and (2.32). Bryson & Ho[16] present a rigorous derivation of this condition.

Example 2.17. Determine the free final time to slew a single-link horizontally to a given angle in a rest-to-rest maneuver.

For an example of free final time, one can use the previous problem structure and change the performance index from one of minimum effort for fixed final time to one of minimum time $J = \int_{t_0}^{t_f} dt$ for a bounded control effort $|u| = u_b$, whose ramifications will be discussed shortly. Using the same initial and final state boundary conditions, allows the Hamiltonian and end function to become

$$\begin{aligned} H &= L + \boldsymbol{\lambda}^T \mathbf{f} \\ &= 1 + \lambda_1 x_2 + \lambda_2 \frac{3u}{ml^2}, \end{aligned} \tag{2.34}$$

and

$$\begin{aligned} G &= \nu^T \Psi \\ &= \nu_1 (x_1(t_f) - x_{1_f}) + \nu_2 x_2(t_f). \end{aligned}$$

Since the emphasis here is on final time, the costate equations are rewritten as

$$\begin{aligned} \lambda_1(t) &= \lambda_1 = \text{constant} \\ \lambda_2(t) &= \lambda_1(t_f - t) + \lambda_2. \end{aligned}$$

(since the differential equations remained the same).

However, the extremal condition $H_u = 0 = 3\lambda_2/ml^2$ is now devoid of the control since H was linear in u. For such problems, no solution will exist unless bounds are placed on the controls or states. If one were to consider minimizing time for an unbounded control scenario, one could envision an infinite amount of control used to achieve a designated slew position in no time at all as a possible solution! To handle the finite final time case, one must apply Pontryagin's Minimum principle[20] (whose derivation is outside the scope of this work). It states that the control must minimize the Hamiltonian. Since the third term in the Hamiltonian (see Eq. 2.34) contains u, this sets up the relations

$$u = -u_b \text{ if } \lambda_2 > 0$$

and

$$u = u_b \text{ if } \lambda_2 < 0$$

for the equi-sided bounded controls.

To accomplish the rest-to-rest maneuver torque must be exerted to start the device moving and reversed to slow it down. If positive torque is exerted to start the motion, then a negative torque is required to slow it and attain both the final position x_{1_f}, and zero final velocity. To satisfy the Minimum principle, the controls would be constrained to the bounds $\pm u_b$ (to provide the most minimum values for H) and would alternate between bounds according to the number of switches (or changes in the sign of λ_2) in what's known as *bang-bang* fashion. Since $\lambda_2(t)$ changes linearly there is at most one switch from positive to negative. Using the same state variables as before and ordering the torques, yields two sets of equations to solve:

For $t_0 \leq t \leq t_s$ where t_s is the lone switch time

$$\begin{aligned} u &= u_b \\ \lambda_2 &< 0 \\ \dot{x}_1 &= x_2 \\ \dot{x}_2 &= c u_b \end{aligned}$$

where $c = 3/ml^2$, thus

$$x_2 = cu_b(t - t_0),$$
$$x_1 = \frac{cu_b}{2}(t - t_0)^2$$

For $t_s \leq t \leq t_f$

$$u = -u_b$$
$$\lambda_2 > 0$$
$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -cu_b,$$

thus

$$x_2 = -cu_b((t - t_s) - (t_s - t_0)),$$
$$x_1 = -\frac{cu_b}{2}(t - t_s)\left[\frac{t - t_s}{2} - (t_s - t_0)\right] + x_{1s}$$

Equating positions and velocities at t_s yields $t_s = (t_f + t_0)/2$ for the equi-sided bounds. Using this result (and assuming $t_0 = 0$), reveals that the switch location $x_{1_s} = cu_b t_f^2/8$ and final time can be solved for as $t_f = 2\sqrt{x_{1_f}/cu_b}$. By using the following numerical values: $m = 0.5, l = 1.0$, and $\theta_f = \pi/2$, with $u_b = 1$, a result of $t_f = 1.023$ seconds is achieved. Using the transversality condition $(\lambda_1 x_2 + \lambda_2 cu)_f = -1$, solve for the multipliers as

$$\lambda_{2_f} = \frac{-sign(u_f)}{cu_b}$$
$$\lambda_1 = \frac{sign(u_f)}{t_s cu_b}.$$

Time histories are shown in Figure 2.17.

2.8. Numerical Optimization

Since analytical optimization is out of the question, one must resort to numerical means. In general, numerical optimization methods are typically divided into two categories, direct and indirect. The direct methods[21,22], also referred to as nonlinear programming methods, fall into this category because cost functional gradient information is employed to find the optimum solution. The indirect methods[23–25] solve the Euler-Lagrange equations to obtain an optimum solution and can be recognized as Pontryagin methods. As inferred by the list of references, many numerical schemes exist for the solution of optimal control problems. These schemes solve either the Euler-Lagrange equations, or the suboptimal control formulation (i.e.,

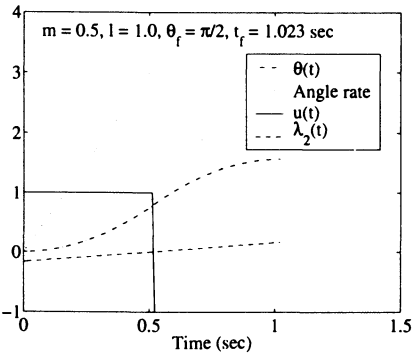

Figure 2.17: Minimum time torque and state variable time histories from Example 2.17

direct method). The Euler-Lagrange methods include quasilinearization[23], method of particular solutions[24], other shooting methods[26], and perturbation methods[25] that solve the *true* optimal control problem. *True* implies that the optimal control functions are not parameterized nor restricted to a particular class of functions (other than being piecewise continuous). Even though this method seems intuitively appealing, it is often misleading because many of these methods are sensitive to the initial guess (costates at the initial time[26]) or provide only an approximate solution due to implementation or convergence issues[25]. As a result, the control functions are either directly or indirectly restricted to a particular class of functions so that a solution can be found that is consistent with the physical system under consideration.

The suboptimal control formulation is described as ***suboptimal*** because it restricts the class of control functions directly by parameterizing the control history[26,27]. The parameterization of the history enables the analyst to convert the functional optimization problem into a parameter optimization problem. Of particular importance, the parameters themselves have physical significance so absurd solutions can be immediately identified and discarded. This formulation is quite desirable because the parameter optimization algorithms are typically less sensitive to the initial guess (control parameters)[21,26,28]. In this section, a nonlinear programming method, called Recursive Quadratic Programming (RQP)[21] is used to solve the optimization problems.

2.8.1. Parameter Optimization - Themes

As mentioned in Section 2.7., most problems are too complex to render closed-form solutions. Parameter optimization methods allow one to find approximate open-loop control solutions for highly nonlinear dynamical systems[16–19].

What are parameters? The actual time-varying control $u(t)$ is recast as a func-

tion of a vector of constants ξ or parameters. Although there may be differential equations involved, the parameters are invariant to any integration, or algebraic method that may compute performance measures $G(t)$ (such as final time). Only the optimization solution method or *wrapper* around the dynamical model has the ability to change the parameters. For a time-varying problem, the problem is changed from dependence on t to ξ and performance measures are recast likewise to $G(\xi)$ for $\xi = \xi_j,\ j = 1, \ldots, n$.

What is the advantage of doing this? The advantage of this approach is that one can handle virtually any numerical performance index or constraint statement that can be computed via a mathematical model (certainly a powerful incentive!).

What is the disadvantage of doing this? One is now relegated to finding approximate, suboptimal solutions that typically depend on the following:

1. Numerical concerns of the dynamical model (e.g., stability, scaling, etc.).
2. The selection of built-in tuning constants provided by the optimization method.
3. The skill of the analyst in managing items 1 and 2.

What do approximate solution forms look like? Typically, approximate forms are iterative schemes that update the parameter vectors based on gradients of the performance measure as

$$\xi_{i+1} = \xi_i + \alpha f\left(G_\xi\left(\xi_i\right), G_{\xi\xi}\left(\xi_i\right), \cdots\right)$$

where α is a user-specified constant, f is some function of the performance measure gradients, i is the current iteration, and $i+1$ is the update.

2.8.2. Development of a First-Order Method-Unconstrained Functions

The first step is to develop a gradient-based iterative scheme (i.e., a repetitive generation of new ξ_i's) that will attempt to minimize an unconstrained function $G(\xi)$.

Assume $G(\xi_i)$ is the result of some arbitrary nonlinear function (e.g., analytical or numerical integration of a set of O.D.E.'s). Using a Taylor series expansion gives

$$G(\xi_{i+1}) = G(\xi_i) + \mathbf{G}_{\xi_i}\left[\xi_{i+1} - \xi_i\right] + H.O.T.$$
$$G(\xi_{i+1}) - G(\xi_i) = \Delta G = \mathbf{G}_{\xi_i}\left[\xi_{i+1} - \xi_i\right]$$

In order for ΔG to be negative (i.e., G decreasing) for function minimization, choose $\left[\xi_{i+1} - \xi_i\right] = -\alpha \mathbf{G}_{\xi_i}^T$, therefore $\Delta G = -\alpha \mathbf{G}_{\xi_i}\mathbf{G}_{\xi_i}^T \leq 0$ for positive scalar constant α. The gradient rule update is $\xi_{i+1} = \xi_i - \alpha \mathbf{G}_{\xi_i}^T$.

There is the need to be able to compute $\mathbf{G}_{\xi_i}$, which is the *search direction* and have a way to choose α. For a computer code, you would need routines to evaluate

G, and $\mathbf{G}_{\boldsymbol{\xi}_i}$. α is usually chosen to be a small positive constant < 0.1, but may need to be several orders of magnitude less for numerical stability. One would like to update the $\boldsymbol{\xi}_i$'s with as large an α as possible.

2.8.3. Gradient Search - Stepping along the Search Direction

Given the gradient rule parameter update $\boldsymbol{\xi}_{i+1} = \boldsymbol{\xi}_i - \alpha \mathbf{G}_{\boldsymbol{\xi}_i}^T$. One needs to be able to move along the gradient vector $\mathbf{G}_{\boldsymbol{\xi}_i}$ an arbitrary distance depending on how close the iteration is to the minimum. This is the art of picking α, or executing the

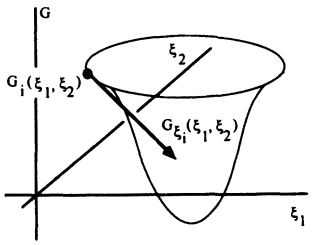

Figure 2.18. Minimizing surface for two parameters

one-dimensional search. A simple-minded approach is as follows:

1. Evaluate the arbitrary function G and the gradient $\mathbf{G}_{\boldsymbol{\xi}_i}$ analytically or by some other means.
2. Start with a small value of α (say 0.001).
3. Update the parameters via the gradient rule.
4. Evaluate a new G with the new parameters.
5. If it is less than the previous G, double α and take a *bigger* step in the $\mathbf{G}_{\boldsymbol{\xi}_i}$ direction.
6. Do this until G increases, then compute the true $\boldsymbol{\xi}_{i+1}$ and update with the last *good* value of α.

The Rosenbrock's problem is given as

$$G = 100\left(\xi_1^2 - \xi_2\right)^2 + \left(1 - \xi_1\right)^2.$$

This problem will strain a gradient scheme because of the elongated contours (more sensitive to changes in parameter ξ_1 versus ξ_2). The gradient scheme will tend to *chatter* on this one. In order to get to the minimum, one must be careful in selecting α (see Figure 2.19). In addition, the gradient method can easily become unstable for poor choices of α. As you might imagine, one could easily send it searching into *outer space*. This problem is explored in more detail in Homework 2.7, #1.

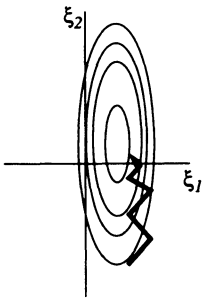

Figure 2.19. Gradient search solution progression for Rosenbrock's problem

2.8.4. A Second-Order Method

This method uses more derivative information to advance to the minimum faster by using curvature information (i.e., the Hessian matrix). Expand $G(\boldsymbol{\xi})$ in a Taylor series as

$$G(\boldsymbol{\xi}_{i+1}) = G(\boldsymbol{\xi}_i) + \mathbf{G}_{\boldsymbol{\xi}_i}\left[\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i\right] + \frac{1}{2!}\left[\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i\right]^T \mathbf{G}_{\boldsymbol{\xi}_i\boldsymbol{\xi}_i}\left[\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i\right] + H.O.T.$$

$$\Delta G = \mathbf{G}_{\boldsymbol{\xi}_i}\left[\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i\right] + \frac{1}{2!}\left[\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i\right]^T \mathbf{G}_{\boldsymbol{\xi}_i\boldsymbol{\xi}_i}\left[\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i\right] + H.O.T..$$

After differentiating both sides with respect to the vector $\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i$ and setting the result to $\mathbf{0}$ (corresponding to an extremal), one gets the *Newton-Rhapson* search direction, or

$$\mathbf{0} = \mathbf{G}_{\boldsymbol{\xi}_i} + \mathbf{G}_{\boldsymbol{\xi}_i\boldsymbol{\xi}_i}\left[\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i\right],$$

and therefore

$$\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i = -\mathbf{G}^{-1}_{\boldsymbol{\xi}_i\boldsymbol{\xi}_i}\mathbf{G}^T_{\boldsymbol{\xi}_i},$$

which can be fashioned into an update procedure as $\boldsymbol{\xi}_{i+1} = \boldsymbol{\xi}_i - \alpha\mathbf{G}^{-1}_{\boldsymbol{\xi}_i\boldsymbol{\xi}_i}\mathbf{G}^T_{\boldsymbol{\xi}_i}$ for a positive constant α. Note, $\mathbf{G}_{\boldsymbol{\xi}_i\boldsymbol{\xi}_i}$ must be nonsingular to use this method. Checking $\mathbf{G}_{\boldsymbol{\xi}_i\boldsymbol{\xi}_i}$ for positive definiteness gives one the assurance that each iteration is moving on a *minimum surface* (concave up).

2.8.5. Finite Differences

What if one does not have analytical derivatives? One can use approximations based on Taylor series. For example, a first-order derivative for a function f can be

obtained by the following:

$$f(\xi + \Delta\xi) = f(\xi) + f_\xi (\xi + \Delta\xi - \xi) + \frac{1}{2!}(\xi + \Delta\xi - \xi)^T f_{\xi\xi} (\xi + \Delta\xi - \xi) + H.O.T.,$$

or

$$\frac{f(\xi + \Delta\xi) - f(\xi)}{\Delta\xi} = f_\xi + \frac{1}{2!} f_{\xi\xi} (\Delta\xi) + \frac{H.O.T.}{\Delta\xi}$$

where Δ indicates a small difference and all derivatives are evaluated about the current parameter vector. This *forward-difference* approximation of f_ξ is accurate to *first-order* in $\Delta\xi$, which is satisfactory if $\Delta\xi$ is small. Only a single additional set of function evaluations $f(\xi + \Delta\xi)$ is necessary to approximate f_ξ.

2.8.6. Second-Order Finite Differences

For more accurate approximations, one can difference the following Taylor series and divide through by $2\Delta\xi$ for a second-order accurate f_ξ:

$$f(\xi + \Delta\xi) = f(\xi) + f_\xi [\xi + \Delta\xi - \xi] + \frac{1}{2!} f_{\xi\xi} [\xi + \Delta\xi - \xi]^2 + \frac{1}{3!} f_{\xi\xi\xi} [\xi + \Delta\xi - \xi]^3 + H.O.T.$$

$$f(\xi - \Delta\xi) = f(\xi) + f_\xi [\xi - \Delta\xi - \xi] + \frac{1}{2!} f_{\xi\xi} [\xi - \Delta\xi - \xi]^2 + \frac{1}{3!} f_{\xi\xi\xi} [\xi - \Delta\xi - \xi]^3 + H.O.T.$$

$$\frac{f(\xi + \Delta\xi) - f(\xi - \Delta\xi)}{2\Delta\xi} = f_\xi + \frac{1}{3!} f_{\xi\xi\xi} [\Delta\xi]^2 + \frac{H.O.T.}{2\Delta\xi}.$$

However, now two function evaluations $f(\xi + \Delta\xi)$ and $f(\xi - \Delta\xi)$ are necessary. Typical perturbations for second-order accurate derivatives are $O(10^{-4})$. If one instead adds the above Taylor series, subtracts $2f(\xi)$ from each side, and divides through by $\Delta\xi^2$, the result is

$$\frac{f(\xi + \Delta\xi) - 2f(\xi) + f(\xi - \Delta\xi)}{\Delta\xi^2} = f_{\xi\xi} + \frac{2}{4!} f_{\xi\xi\xi\xi} [\Delta\xi]^2 + \frac{H.O.T.}{\Delta\xi^2},$$

and a second-order accurate approximation to $f_{\xi\xi}$ results. Again, this accuracy requires two additional function evaluations (in addition to the *nominal* $f(\xi)$).

2.8.7. Penalty Functions

The goal is to reveal a way to attach constraints $\Psi(\xi) = \mathbf{0}$. In Section 2.7.8., Lagrange multipliers were used to attach equality constraints and enable analytic closed-form optimal solutions. An analog for parameter optimization is to attach the constraint vector $\Psi(\xi)$ to a scalar performance index $\phi(\xi)$ to produce the scalar form

$$G(\xi) = \phi(\xi) + \frac{1}{2}\Psi^T(\xi)\mathbf{W}\Psi(\xi)$$

where $\mathbf{W}$ is a positive-definite diagonal matrix of user-specified weights. The quadratic *penalty* term will be positive and therefore increase the value of $G(\xi)$ that does not satisfy $\Psi(\xi) = \mathbf{0}$ (i.e., lie on the constraints). Since one is operating in a minimization framework, the best that can be hoped for, from the penalty term, is to drive it to zero at which point $G(\xi)$ is satisfied. This is a *soft* constraint formulation in that the constraints are lumped in with the performance measure ϕ and their derivatives are not treated explicitly. Successful implementation of gradient update schemes to treat penalty functions is heavily dependent on the selection of the weight matrix. If the components of $\mathbf{W}$ are too small, the minimization favors $\phi(\xi)$ at the expense of satisfying the equality constraints. For large weights, the converse is true.

Solution Form

A *gradient-based* iterative scheme (the repetitive generation of new ξ_i's) attempts to minimize a function of the form $G(\xi_i) = \phi(\xi_i) + \Psi^T(\xi_i)\mathbf{W}\Psi(\xi_i)$ for a matrix of user-selected weights $\mathbf{W}$. The iterative scheme will have *converged* to an approximate answer as follows:

1. $\mathbf{G}_{\xi_i}$ goes to $\mathbf{0}$.
2. ξ_{i+1} differs little from ξ_i.
3. $\Psi(\xi_{i+1})$ is a vector of small residuals.
4. Some combination of 1—3.

2.8.8. Parameterized Controls - the use of tabular functions $\mathbf{u}_*(\xi)$

Often, the desired parameter solutions are the actual control values (i.e., torques) applied by the flexible robot joint motor(s) as a function(s) of time. In the analytical solution of Example 2.16, the single-link torque time history was solved as a continuous function of time $u(t)$ for $0 \leq t \leq t_{\text{final}}$. In the parameterized control scenario, the approximate solution for $u(t)$ takes the form $\mathbf{u}_*(\xi) = [\xi_1, \xi_2, \ldots, \xi_n]$, which occurs at fixed *nodal* times $\mathbf{t}_* = [t_1, t_2, \ldots, t_n]$ where subscripts 1 and n correspond to the initial and final times. Although it is often the case, these nodal

times do not need to be selected as constant increments. In the limit, the nodal times would become infinitesimally close together resulting in an infinite number of parameters to solve in order to span the continuous time of the problem. This is not practical, therefore a smaller number of parameters are selected.

When performing numerical integration of the dynamical model the approximate control $\mathbf{u}_*$ would reside in a lookup table of parameterized control values corresponding to specific discrete values of time $\mathbf{t}_*$. An interpolation routine is employed to determine other *off-node* values of u that correspond to a higher resolution of time generated by the integration scheme. Each time the update scheme varies a component ξ_j of the control vector by some amount $\Delta\xi$ the parameterized *history* will change. Interpolation gives rise to slightly different motion histories for the states (e.g., acceleration, velocity, or displacement). The parameter optimization process may produce a final $\boldsymbol{\xi}$ that is quite different from your initial guess.

To introduce some of the concepts that are required to set-up the parameterized controls problem, consider Example 2.16, Figure 2.20, and Table 2.1. To solve this same problem using numerical optimization, the control variable u is parameterized in time and compared conceptually with its analytical counterpart in Figure 2.20. Note, the parameterized controls are now discretized at various nodal times. The relevant continuous expressions along with their corresponding converted parameterized control expressions are given in Table 2.1. This example is solved completely in Chapter 5 (see Example 5.1).

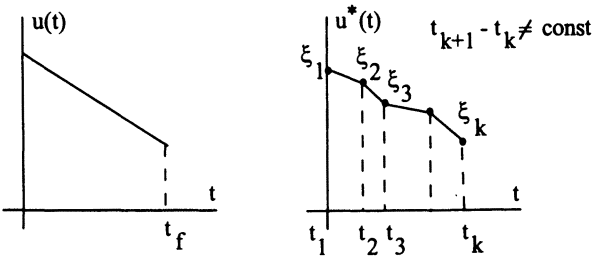

Figure 2.20. Discretizing control effort $u(t)$

Before leaving this section, it is instructive to discuss a general parameter optimization solution procedure. One way to numerically solve the general parameter optimization problem is to follow the methodology of Section 2.7. while replacing the analytical computations with numerical procedures. To be more specific, the original formulation of minimize

$$J = \int_{t_0}^{t_f} L(\mathbf{x}, \boldsymbol{\xi}, t)dt,$$

Table 2.1. Parameterized control problem

analytical	approximate
$\mathbf{x}(t), u(t)$	$\mathbf{x}(\boldsymbol{\xi}), \mathbf{u}_*(\boldsymbol{\xi})$
$\phi(t) = \int_{t_0}^{t_f} u^2 dt$	$\phi(\boldsymbol{\xi}) \approx \int_{t_0}^{t_f} u_*^2(\boldsymbol{\xi}) dt$
$\boldsymbol{\Psi}(\mathbf{x_f}, t_f) = \begin{bmatrix} x_{1_f} - x_{1_d} \\ x_{2_f} \end{bmatrix} = \mathbf{0}$	$\boldsymbol{\Psi}(\boldsymbol{\xi}) =$ small

subject to

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \boldsymbol{\xi}, t),$$

and

$$\boldsymbol{\Psi}(\boldsymbol{\xi}) \geq \mathbf{0}$$

where

$\mathbf{x}$ = state vector (n x 1)
$\boldsymbol{\xi}$ = parameter vector (a x 1 constants)
$\boldsymbol{\Psi}$ = constraint vector (b x 1)

is replaced by a four step conversion process.

First, the state-space equations are augmented by the integral equation as defined by

$$J' = x_{n+1}(\mathbf{x}(t_f), \boldsymbol{\xi}, t_f) \tag{2.35}$$

$$\dot{\mathbf{x}} = \mathbf{f}$$

$$\dot{x}_{n+1} = L. \tag{2.36}$$

Second, the state-space equations are numerically integrated with respect to the time to obtain

$$\mathbf{x}(t_f) = \mathbf{x}(\boldsymbol{\xi})$$

$$J' = x_{n+1}(\boldsymbol{\xi}) \tag{2.37}$$

$$\boldsymbol{\Psi} \geq \mathbf{0}, \tag{2.38}$$

and Eq. (2.37) and (2.38) are numerically differentiated with respect to the parameter vector to provide

$$\frac{\partial J'}{\partial \boldsymbol{\xi}} = \mathbf{f}_1(\boldsymbol{\xi}) \tag{2.39}$$

$$\frac{\partial \Psi}{\partial \boldsymbol{\xi}} = \mathbf{f}_2(\boldsymbol{\xi}). \tag{2.40}$$

Third, the values of Eq. (2.37)–(2.40) are substituted into the RQP program to find the optimum correction to the solution (feasible search direction) at the i^{th} iteration. Finally, the process is iterated at the second step until the RQP program converges to the overall optimum solution.

Ideally, the conversion process works wonderfully, but in practice pitfalls do exist in every application. These pitfalls include numerical instabilities, local minima, and reachability of feasible solutions. These issues are discussed further in Chapters 4 and 5.

Now, how does one convert the optimal control problem into a parameter optimization problem? The original formulation of the optimal control problem is structured to minimize

$$J = \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}, t)dt,$$

subject to

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$$

where $\mathbf{x}(t_f), t_0$, and t_f) are prescribed *a priori*. An alternative formulation is developed wherein the unknown function $\mathbf{u}(t)$ is eliminated in favor of a set of unknown parameters through the following six steps. First, the state-space equations are augmented by the integral equation and a *Mayer form* final time[29] problem is formed (see Eqs. 2.35, and 2.36). Second, the time is normalized to simplify the control history description and to accommodate free final time problems.

$$\tau = \frac{t}{t_f}$$

$$dt = t_f d\tau$$

$$\mathbf{x}' = t_f \mathbf{f}$$

$$x'_{n+1} = t_f L.$$

While this use of normalized time allows one to use the same stopping condition ($\tau = 1$), it does not free one from the necessity to determine or to prescribe the actual t_f.

Third, the control history is parameterized by one of two methods. One method subdivides the normalized time interval into an integer number of subintervals that

are bounded by equally spaced (in time) node points and piecewise linear interpolation is used to approximate the intermediate control history[26,28,30]. Note, the weights of the function are defined as the optimization parameters (see Figure 2.21).

$$\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_m)^T$$
$$\Delta\tau = \frac{1}{m-1}.$$

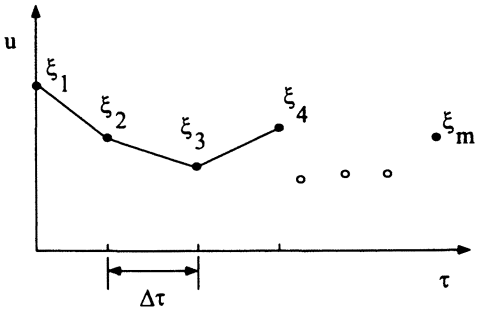

Figure 2.21. Control parameterization

The other method assumes a linear combination of functions (e.g., Fourier series, splines, orthogonal polynomials, etc.) to describe the control history[27]. The multiplicative constants become the optimization parameters

$$\mathbf{u} = \sum_{i=1}^{m} \xi_i f_i$$
$$\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_m)^T$$
$$f_i = i^{\text{th}} \text{ approximation function.}$$

Fourth, the state equations are numerically integrated with respect to the normalized time to obtain

$$\mathbf{x}(\tau = 1) = \mathbf{x}(\boldsymbol{\xi})$$
$$J' = x_{n+1}(\boldsymbol{\xi}), \tag{2.41}$$

and Eq. (2.41) in numerically differentiated with respect to the parameter vector to produce

$$\frac{\partial J'}{\partial \boldsymbol{\xi}} = \mathbf{f}_3(\boldsymbol{\xi}). \tag{2.42}$$

Fifth, the values of Eq. (2.41) and (2.42) are substituted into the RQP program to find the optimal solution at the i^{th} iteration. Finally, the process is iterated at the fourth step until the RQP program converges to a local optimum solution. If the iteration is initiated within the domain of attraction of the globally optimum solution, this locally optimum solution will of course be the globally optimum solution.

2.9. Chapter 2 Summary

This chapter reviewed many preliminary mathematical fundamentals that are found throughout the remainder of the book. The goal was to present enough applied material to help the reader independently follow subsequent developments (dynamic modeling, system identification techniques, and control system designs). Fundamental developments were reviewed for linear algebra, linear control systems, digital systems, and calculus of variations. Methods used in the development of dynamic equations of motion were also reviewed (Newton's equations of motion, energy methods based on Hamilton's principle, and Lagrange's equations). These energy based methods become fundamental concepts for the Chapter 3 developments. The concept of analytical optimization was introduced for finding open-loop optimal control profiles. In addition, for an applied emphasis, numerical optimization fundamentals were introduced. This methodology is useful for dynamic systems that do not have readily available analytical solutions. This methodology's usefulness will be demonstrated for both system identification (see Chapter 4) and parameterized controls (see Chapter 5) applications. In Chapter 3, the dynamical equations of motion for flexible manipulators are introduced and the control system fundamentals presented in this chapter will become the starting point for further development in Chapters 4, 6, 7, and 8.

2.10. Chapter 2 References

1. A. W. Roberts, *Elementary Linear Algebra*, The Benjamin Cummings Publishing Co. Inc., 1982.
2. B. C. Kuo, *Automatic Control Systems*, Prentic-Hall, Inc., 1982.
3. C. T. Chen, *Linear System Theory and Design*, Holt Rinehart, and Winston, 1970.
4. M. D. Greenberg, *Foundations of Applied Mathematics*, Wellesley-Cambridge Press, 1986.
5. G. Strang, *Introduction to Applied Mathematics*, Prentice-Hall, 1978.
6. W. E. Boyce, and R. C. DiPrima, *Elementary Differential Equations and Boundary Value Problems*, John Wiley & Sons, 1977.
7. B. C. Kuo, *Digital Control Systems*, Holt Rinehart and Winston, 1980.
8. R. L. Burden, and J. D. Faires, *Numerical Analysis*, Prindle, Wever & Schmidt, 1985.
9. E. W. Swokowski, *Calculus with Analytic Geometry*, PWS-KENT, 1975.
10. I. M. Gelfand, and S. V. Fomin, *Calculus of Variations*, Prentice-Hall, 1963.
11. D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, 1989.

12. H. Goldstein, *Classical Mechanics*, Second Edition, Addison-Wesley Publishing Company, 1981.
13. L. Meirovitch, *Methods of Analytical Dynamics*, McGraw-Hill, 1970.
14. T. R. Kane, and D. A. Levinson, *Dynamics: Theory and Applications*, McGraw-Hill Book Company, 1985.
15. J. Feddema, R. Baty, R. Dykhuizen, C. Dohrmann, G. Parker, R. Robinett, V. Romero, and D. Schmitt, "System identification, and control for slosh-free motion of an open container of liquid," SAND Report96-0995, Sandia National Laboratories, April, 1996.
16. A. E. Bryson, and Y. C. Ho, *Applied Optimal Control*, Halsted Press, 1975.
17. P. E. Gill, et al., *Practical Optimization*, Academic Press, 1981.
18. J. E. Dennis, and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, 1983.
19. G. Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, 1986.
20. L. S. Pontryagin, et al., *The Mathematical Theory of Optimal Precesses*, Interscience, 1962.
21. M. J. D. Powell, "A fast algorithm for nonlinearly constrained optimization calculators," Proceedings of the Biennial Conference on Numerical Analysis, 28 June-1 July 1977, Waston, G.A., Editor, Springer-Verlag, 1978, pp. 144-157.
22. P. E. Gill, W. Murray, M. A. Saunders, and Wright, "User's guide for SQP/ NPSOL: A Fortran package for nonlinear programming," Report SOL 83-12, Department of Operations Research, Stanford University, Stanford, California, 1983.
23. B. P. Yao, and K. B. Sng, "Numerical solution of the constrained re-entry vehicle trajectory problem via quasilinearization," Journal of Guidance and Control, Vol. 3, No. 5, Sept.-Oct. 1980, pp. 392-397.
24. A. Miele, and R. R. Iyer, "General technique for solving nonlinear two-point boundary-value problems via the method of particular solutions," Journal of Optimization Theory and Applications, Vol. 5, No. 5, 1970, pp, 382-399.
25. J. L. Junkins, and R. C. Thompson, "An asymptotic perturbation method for nonlinear optimal control problems," AAS/AIAA Astrodynamics Specialist Conference, Vail CO, AAS 85-364, August 1985.
26. D. G. Hull, ASE 3800 - Optimal control II notes, Department of Aerospace Engineering, University of Texas At Austin, 1984.
27. R. F. Stengel, *Stochastic Optimal Control*, John Wiley and Sons, 1986, pp. 192-201.
28. D. E. Outka, "Parameter optimization capability in the trajectory code PMAST," SAND86-2917, 1987.
29. G. Leitmann, *Optimization Techniques - With Applications to Aerospace Systems*, Academic Press, 1962, Chapters 4 and 6.
30. D. E. Outka, "Launch Vehicle Trajectory Optimization," Master's Thesis, Univer sity of Texas at Austin, 1984.

2.11. Chapter 2 Problems

Homework 2.1. From Sections 2.2.-2.4.11.

1. Find the eigenvalues of the following matrices:

$$\mathbf{A}_1 = \begin{bmatrix} 1 & 3 & -5 \\ 2 & 0 & 1 \\ 3 & 3 & 3 \end{bmatrix} \quad \mathbf{A}_2 = \begin{bmatrix} 1 & 4 & 5 \\ 2 & 0 & 2 \\ 4 & 1 & 2 \end{bmatrix}.$$

2. Consider the system given by

$$\dddot{x} + 6.5\ddot{x} + 12.5\dot{x} + 7x = u.$$

   a. Find the transfer function for this system.

   b. Find the impulse response function for this system.

   c. Find the response of this system by using the impulse response for the inputs given below where all initial conditions are zero.

   i. $u = unit\ step$

   ii. $u = sin2\,t$

3. Consider the system given by

$$\ddot{x}_1 + 3\dot{x}_1 + 2x_1 - 0.1x_2 = 3u$$
$$\dot{x}_2 + x_2 + 3x_1 = u,$$

   with initial conditions

$$x_1(0) = 1.0 \quad \dot{x}_1(0) = 0.1 \quad x_2(0) = -1.0.$$

   Assuming the input u is zero, find the state at time $t = 3.0$ seconds. *Hint*: Use the state transition matrix.

4. Find the state transition matrix of

$$\mathbf{A} = \begin{bmatrix} -1 & \sin t \\ 0 & -1 \end{bmatrix}$$

   by using Definition 2.12 where the initial time t_0 is zero. Examine the series solution numerically by comparing the exact solution to a one term, two term, and three term Taylor series expansion at time $t = 1.0$ seconds.

5. For the system of Problem 3, write the solution to a step input using the solution to the state equation.

6. For the system of Problem 3, find the impulse response matrix, and the transfer function matrix.

7. Determine the controllability of the system of Problem 3.

8. Use Euler and AB-3 with a step size of 0.05 seconds to integrate the system of Problem 3, from 0.0 to 10.0 seconds where the input is

$$u(t) = sin\ t.$$

Show the output in tabular form (using 1 second increments) (see Table 2.2). Discuss the differences between the methods used.

Table 2.2. Output table for Homework 2.1, #8

Time	Exact	Euler	AB-3
0.0			
1.0			
2.0			
3.0			

9. Find the difference equations for the compensator

$$\frac{Y}{X} = \frac{s+3}{(s+1)(s+2)}$$

by using Euler and the state-matrix methods. Use an update of 0.1 seconds. Tabulate the results for 2 seconds of execution. Assume the initial conditions of the compensator are zero. Discuss the relative differences of these two methods for implementing this compensator.

Homework 2.2. From Section 2.5.

1. Find a local minimum of
   a. $J = \frac{1}{2}(x_1^2 + x_2^2) + (x_1 + x_2)$.
   b. $J = \frac{1}{2}(x_1^2 + x_2^2) + (x_1 + x_2), x_2 = -x_1$. (use Lagrange multipliers)

2. Find a local minimum of $J = x_1^2 - 4x_1x_2 + x_2^3 + 4x_2$.

3. Prove this matrix is positive definite $H = \begin{bmatrix} 4 & 3 & 1 \\ 3 & 5 & 0 \\ 1 & 0 & 6 \end{bmatrix}$.

4. Determine the modified normal equations for

$$J = (\mathbf{z} - \mathbf{z}_m)^T \mathbf{W} (\mathbf{z} - \mathbf{z}_m)$$
$$\mathbf{z} = \mathbf{A}\mathbf{x}$$
$$\mathbf{g}(\mathbf{x}) = \mathbf{0}.$$

5. Find a weak extremal for

$$J(y) = \int_1^2 \frac{\sqrt{1+\dot{y}}}{t}\, dt \quad \text{for } y(1) = 0 \text{ and } y(2) = 1.$$

6. Find a weak extremal for

$$J(y) = \int_0^1 (t-y)^2 dt \quad \text{for } y(0) = 0 \text{ and } y(1) = 1.$$

7. Find a weak extremal for

$$J(y) = \int_1^2 \frac{\sqrt{1+\dot{y}}}{\sqrt{2gy}}\, dt \quad \text{for } y(1) = 1 \text{ and } y(2) = 2.$$

Hint: This is the functional that describes the brachistochrone problem.

Homework 2.3. From Section 2.6.

1. Derive the equations of motion for a simple pendulum as follows:
   a) Use the angle as in Figure 2.8.
   b) Use x, y coordinates with Lagrange multipliers.

2. Derive the equations of motion for a two-link, rigid planar robot with springs, dampers, and torques at the two joints.

3. Derive the equations of motion for a jib crane (see Figure 1.3) and linearize the nonlinear equations of motion.

Homework 2.4. From Section 2.7.6.

1. Minimize the distance to parabola $G = \frac{1}{2}\left(x^2 + \left(x^2 + a\right)\right)$. Specify the minimum for $a = 1, 0, -1$.

2. A tin can manufacturer wants to find the dimensions of a cylindrical can (closed top and bottom) such that for a given amount of tin, the volume of the can is a *maximum*. If the thickness of the tin stock is constant, a given amount of tin implies a given surface area of the can. Use the height and the radius as variables.

Homework 2.5. From Section 2.7.8.

1. Find the extremal control for the performance index

$$J = cx_f^2 + \int_{t_0}^{t_f} \left(\frac{u^2}{2}\right) dt \quad \text{where c is a positive constant}$$

dynamical system $\dot{x} = u$ boundary conditions t_0, t_f, x_0 given.

2. Prove that the Hamiltonian $H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda})$ in Example 2.16 (for the horizontal slewing link) is a constant.

3. Find the minimum energy control to slew a single-link in a rest-to-rest maneuver to a given angle in the *vertical* plane. The system equation is

$$\tau = \frac{ml^2}{3}\ddot{\theta} + \frac{mgl}{2}\sin\theta$$

where g is the gravitational acceleration.

Work this problem analytically as far as you can and state your observations.

Homework 2.6. From Section 2.5.1.

1. Solve $e^x = tan(x)$. Try starting with $x_i = 1, 2$. Plot $f(x)$ for $x = [-4, 2]$.

2. Solve $x_1^3 - x_2 = 0 \;\; x_1^2 + x_2^2 = 1$. Try starting with $x_i = -1, 1$ for both x_1, x_2.

Homework 2.7. From Section 2.8.8. (Software will be provided for 1 and 3—see Appendix A, or B)

1. Solve Rosenbrock's problem by using both a gradient method and a second-order scheme. Compare the number of iterations for convergence.

2. Derive a finite difference approximation to the mixed derivative f_{xy} where x, y are scalar variables. *Hint*: Use the Taylor series expansion form in two variables for the perturbations: $\Delta x, -\Delta x, \Delta y$, and $-\Delta y$.

3. Solve the single-link vertical slew problem given earlier (see Homework 2.5, #3) by using the gradient scheme, finite differences, and the following:

   (a) Choose constraint weights, $W_1, W_2 = 10$.

   (b) Choose constraint weights = 100.

   (c) Remove $\int_{t_0}^{t_f} u_*^2(\xi_i)dt$ from the G function and solve.

Chapter 3

Flexible Robot Dynamic Modeling

3.1. Introduction

Several of the control strategies for flexible link robots described in the remainder of the book rely on an accurate dynamic model of the system. Creating a dynamic model that accounts for link flexibility adds additional challenges beyond the standard rigid link robot dynamics. The most apparent complexity arises due to the additional degrees-of-freedom associated with link deformations. Although in theory this adds an infinite number of degrees-of-freedom, in practice only a finite number are used to generate a model that is sufficiently accurate for predictive simulation and control design. Another complexity (and perhaps a less obvious one) is the appearance of first-order (not negligible) dynamic effects due to second-order kinematic and force effects that at first glance appear to be negligible. For simple robot configurations, these effects can be handled in several intuitive ways. However, for complicated geometries, a systematic approach is needed to ensure that coupling effects are not inadvertently lost. Much of this chapter is devoted to describing such an approach, called the method of quadratic modes.

This chapter focuses on the presentation of two methods for generating dynamic models for flexible link robots. The first approach is the method of quadratic modes and it is applicable to general robot configurations. The second approach is a finite-element-like method that is applicable to planar robots. Before proceeding into the details of these methods, some preliminary mathematical concepts and their relationship to beam analysis are reviewed. A separate section on robot actuator dynamics is also included at the end of this chapter.

3.2. Flexible Link Modeling Preliminaries

In this section, several techniques for obtaining mode shapes are reviewed. This review is demonstrated by utilizing an analysis of Euler-Bernoulli structures. Beams and beam-like structures are the fundamental building block for investigating flexi-

R. D. Robinett III et al., *Flexible Robot Dynamics and Controls*

ble link dynamic effects. Later in the chapter link deformations will be decomposed into a series of basis functions, called mode shape functions. How one chooses these mode shapes dramatically impacts the accuracy of the dynamic model and some critical aspects of its form. For example, if the mode shapes are orthogonal, then the dynamic equations are partially decoupled. This form can be advantageous during control system design.

First, several preliminary concepts are developed. Orthogonality is defined and a classical method for generating orthogonal functions is reviewed. Therefore, the concept of linear independence of functions is first introduced. Next, the analysis of beam dynamics is introduced. Periodically, several methods are discussed that lead to the selection of mode shapes used in flexible link analysis. Finally, the rotational kinematics are defined.

3.2.1. Linear Independence of Functions

The set of functions $\phi_1(\chi), \phi_2(\chi), \ldots, \phi_n(\chi)$ are linearly independent over the range $\chi_1 \geq \chi \geq \chi_2$ if they satisfy

$$\alpha_i \phi_i(\chi) = 0 \qquad \chi_1 \geq \chi \geq \chi_2,$$

and the constants $\alpha_1, \alpha_2, \ldots, \alpha_n$ must all be zero. Although this definition gives insight into the meaning of linear independence, it does not directly help one in determining if a given set of functions are linearly independent. To assess linear independence of vector functions $\Phi = [\phi_1(\chi), \phi_2(\chi), \ldots, \phi_n(\chi)]^T$, the scalar equation

$$det\left[\int_{\chi_1}^{\chi_2} \Phi \Phi^T d\chi\right] \neq 0 \tag{3.1}$$

must be satisfied.

3.2.2. Orthogonality of Functions

Similar to linear independence, the concept of orthogonal functions is dependent on the range of interest. That is, when discussing the orthogonality of a set of functions one must also specify the range over which the functions are orthogonal.

Two functions $\psi_1(\chi)$ and $\psi_2(\chi)$ are orthogonal over the interval $[\chi_1, \chi_2]$ if they satisfy

$$\int_{\chi_1}^{\chi_2} \psi_1(\chi)\psi_2(\chi) d\chi = 0. \tag{3.2}$$

A set of functions $\psi_1(\chi), \psi_2(\chi), \ldots, \psi_n(\chi)$ are orthogonal if Eq. (3.2) is satisfied for all $\psi_i(\chi)\psi_j(\chi)$ pairs with $i \neq j$.

A given set of linear independent functions $\phi_1(\chi), \phi_2(\chi), \ldots, \phi_n(\chi)$ can be made

orthogonal by using the Gram-Schmidt orthogonalization process, or

$$\begin{aligned}
\psi_1 &= \phi_1 \\
\psi_2 &= \phi_2 - \frac{\int_{\chi_1}^{\chi_2} \psi_1 \phi_2 d\chi}{\int_{\chi_1}^{\chi_2} \psi_1 \psi_1 d\chi} \psi_1 \\
\psi_3 &= \phi_3 - \frac{\int_{\chi_1}^{\chi_2} \psi_1 \phi_3 d\chi}{\int_{\chi_1}^{\chi_2} \psi_1 \psi_1 d\chi} \psi_1 - \frac{\int_{\chi_1}^{\chi_2} \psi_2 \phi_3 d\chi}{\int_{\chi_1}^{\chi_2} \psi_2 \psi_2 d\chi} \psi_1 \\
&\vdots = \vdots \\
\psi_n &= \phi_n - \sum_{i=1}^{n-1} \frac{\int_{\chi_1}^{\chi_2} \psi_i \phi_n d\chi}{\int_{\chi_1}^{\chi_2} \psi_i \psi_i d\chi} \psi_i.
\end{aligned} \tag{3.3}$$

Example 3.1. Show that the two functions

$$\begin{aligned}
\phi_1(\chi) &= \chi \\
\phi_2(\chi) &= sin(\chi)
\end{aligned}$$

are linear independent over the interval $[0, \pi]$, and use them to create two orthogonal functions $\psi_1(\chi)$ and $\psi_2(\chi)$.

Solution

According to Eq. (3.1) to check the linear independence of $\phi_1(\chi)$ and $\phi_2(\chi)$ the determinant of

$$\mathbf{P} = \int_0^{\pi} \begin{bmatrix} \chi^2 & \chi \sin \chi \\ \chi \sin \chi & \sin^2 \chi \end{bmatrix} d\chi$$

must be nonzero. Taking the determinant of $\mathbf{P}$ gives

$$det\ \mathbf{P} = \frac{1}{9}\pi^6 - \frac{1}{2}\pi^2$$

which is nonzero, thus indicating that $\phi_1(\chi)$, and $\phi_2(\chi)$ are linearly independent over $[0, \pi]$.

Using the Gram-Schmidt orthogonalization process and starting with $\phi_1(\chi)$ gives

$$\begin{aligned}
\psi_1(\chi) &= \chi \\
\psi_2(\chi) &= \sin \chi - \frac{\int_0^{\pi} \chi \sin \chi}{\int_0^{\pi} \chi^2} \chi \\
&= \sin \chi - \frac{3}{\pi^2} \chi.
\end{aligned}$$

Note, the orthogonalization process is unique up to a multiplicative constant. That is, if $\psi_1(\chi)$, and $\psi_2(\chi)$ are orthogonal, then so are $\alpha_1 \psi_1(\chi)$, and $\alpha_2 \psi_2(\chi)$

where α_1, and α_2 are constants. This additional freedom allows one to impose other constraints that result in unique orthogonal functions. One such constraint, called normalization, requires

$$\int_{\chi_1}^{\chi_2} \psi_i \psi_i d\chi = 1$$

and produces orthonormal functions ψ_1 through ψ_n. More formally,

$$\psi_i(\chi) = \frac{\psi_i}{\sqrt{\int_{\chi_1}^{\chi_2} \psi_i \psi_i d\chi}}.$$

Example 3.2. By using the orthogonal functions derived in Example 3.1 create an orthonormal set and plot them over the region $[0, \pi]$.

Solution

$$\psi_1 = \frac{\sqrt{3}}{\pi^{3/2}}\chi$$

$$\psi_2 = \frac{1}{\sqrt{\frac{\pi^2-6}{2\pi}}}\left(\sin\chi - \frac{3}{\pi^2}\chi\right)$$

The orthonormal set is plotted in Figure 3.1.

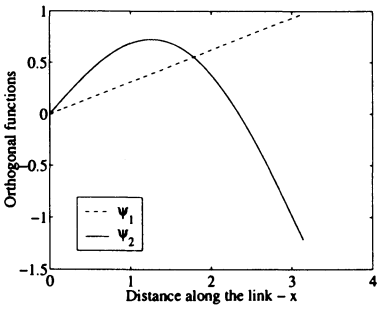

Figure 3.1. Orthonormal functions of Example 3.2

3.2.3. Beam Dynamic Analysis - Analytical Solution

The equations of motion for transverse vibrations of Euler-Bernoulli beams is well known[1,2]. The boundary-value problem for a beam in flexure is defined by a

fourth-order partial differential equation with two boundary conditions at each end. A brief review of dynamic beam analysis is provided to introduce key concepts, such as boundary conditions, natural frequencies, and mode shapes.

The two-point boundary value problem is

$$\frac{\partial^2}{\partial x^2}\left[EI(x)\frac{\partial^2 y(x,t)}{\partial x^2}\right] + \rho(x)\frac{\partial^2 y(x,t)}{\partial t^2} = 0 \tag{3.4}$$

where boundary conditions are specified at $x = 0$, and $x = L$. The cantilever beam is a popular component to represent flexible links. The definitions for a beam with cantilever boundary conditions is illustrated in Figure 3.2. At $x = 0$, both the displacement and slope are zero. At the free end $x = L$, both the moment and shear are zero. By applying the boundary conditions to Eq. (3.4) the natural frequency

Figure 3.2. Cantilevered beam free vibration

relationship[2] is given as

$$\omega_i = \frac{\lambda_i^2}{L^2}\left(\frac{EI}{\rho}\right)^{\frac{1}{2}},$$

and the mode shapes as

$$Y_i(x) = A_i\big[(\sin\beta_i L - \sinh\beta_i)(\sin\beta_i x - \sinh\beta_i x) - (\cos\beta_i L + \cosh\beta_i L)(\cos\beta_i x - \cosh\beta_i x)\big] \tag{3.5}$$

where A_i is the normalization coefficient. The frequencies come from solving a transendental equation, called the characteristic equation. Numerical values[2] for the first three cantilever mode shapes are as follows: $\lambda_i = \beta_i L = 1.8751, 4.694$, and 7.855.

3.2.4. Mode Shapes from Static Loading Conditions

Mode shapes can be approximated by investigating different loading conditions[3]. These loading conditions may consist of point forces, moments, distributed loads, or various combinations. From Euler-Bernoulli beam theory the bending moment is related to the curvature as

$$M = EI(\frac{\partial^2 y}{\partial x^2}). \tag{3.6}$$

For a given moment (and starting with Eq. (3.6)), several integrations result in a deflection curve. By performing a static beam analysis the transverse displacement or linear mode shape can be approximated (see Example 3.3).

Example 3.3. Consider a concentrated transverse load $M = P(L - x)$ applied at the tip of a cantilevered beam (see Figure 3.3). Find an expression for the linear mode.

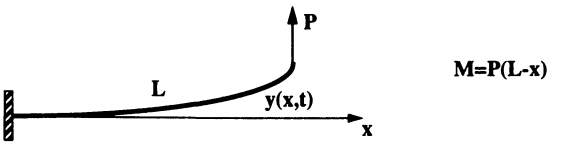

Figure 3.3. Cantilever beam with transverse tip force

Solution

Substituting the concentrated transverse load into Eq. (3.6) and integrating yields

$$\frac{d^2y}{dx^2} = \frac{P[L-x]}{EI}$$
$$\frac{dy}{dx} = \frac{P}{2EI}\left[2Lx - x^2\right]$$
$$y(x) = \frac{P}{6EI}\left[3Lx^2 - x^3\right].$$

The expression for $y(x)$ represents the transverse displacement linear response to the concentrated load. Thus, for the linear mode, one may choose

$$\phi(x) = \left[3Lx^2 - x^3\right], \tag{3.7}$$

or equivalently

$$y(x,t) = \left[3Lx^2 - x^3\right] q(t). \tag{3.8}$$

3.2.5. Assumed Modes Method

Exact (closed-form) solutions to most vibration problems cannot be found. Complicating factors include boundary conditions, irregular geometries, nonuniform mass or stiffness distribution, and material or geometric nonlinearities. Instead of deriving and solving for the general solution of $w(\chi, t)$ where both χ and t are free to vary during the application of Hamilton's principle, an approximate approach,

called the *assumed modes* method[4] can be used. Let the solution take the form

$$w(\chi, t) = \sum_{i=1}^{n} \phi_i(\chi) q_i(t) \tag{3.9}$$

where the basis functions $\phi_i(\chi)$ are specified in advance. Therefore, only the generalized coordinates $q_i(t)$ are time varying. The basis functions must satisfy the kinematic boundary conditions of the system, form a linearly independent set, and resemble the actual mode shapes. In addition, if the $\phi_i(\chi)$ are orthogonal or satisfy the natural boundary conditions, additional benefits may arise.

The approximation of this solution arises from two features of the approach. First, the mode shapes or basis functions are (in general) not the true analytical mode shapes that one would obtain from solving the partial differential equation associated with the system. Second, only a finite number of mode shapes are considered, thereby truncating the solution. Many approximate methods for solving boundary value problems of engineering interest use this approach (i.e., finite element analysis). A simple example that demonstrates the steps used to generate an assumed modes solution is given next.

Example 3.4. Consider the free vibrations of a cantilevered beam (see Figure 3.2).

Solve the free vibration problem by using the assumed modes approach in conjunction with Lagrange's equations. Compare your results with the exact solution.

Solution

1. Select admissible function(s) for the cantilevered beam. For the sake of simplicity, one can use a single assumed-mode ($n = 1$). Furthermore, one can use the mode shape previous developed (see Eqs. 3.7 and 3.8) and associated with a transverse force at the beam tip, or

$$\phi(x) = x^2(3L - x) \tag{3.10}$$

$$y(x, t) = x^2(3L - x)q(t). \tag{3.11}$$

2. Form expressions for the kinetic and strain energies of the beam, or

$$\begin{aligned} T &= \frac{1}{2}\int_0^L \rho(x)\left(\frac{\partial y(x,t)}{\partial t}\right)^2 dx \\ &= \frac{1}{2}\left(\frac{33\rho L^7 \dot{q}^2}{35}\right) \\ U &= \frac{1}{2}\int_0^L EI(x)\left(\frac{\partial^2(x,t)}{\partial x^2}\right)^2 dx \\ &= \frac{1}{2}(12EIL^3 q^2). \end{aligned}$$

3. Form equations of motion by using Lagrange's equations, or

$$\ddot{q} + \left[\frac{140EI}{11\rho L^4}\right] q = 0$$

$$\omega = \left[\frac{140EI}{11\rho L^4}\right]^{\frac{1}{2}} = \frac{(1.8888)^2}{L^2}\left[\frac{EI}{\rho}\right]^{\frac{1}{2}}.$$

The exact solution for a uniform beam is

$$\omega_e = \frac{(1.8751)^2}{L^2}\left[\frac{EI}{\rho}\right]^{\frac{1}{2}}.$$

The assumed modes method underlines the framework behind the method of quadratic modes architecture.

3.2.6. The Finite Element Method

Discretization methods, such as the *finite element method* are popular in commercial disciplines (e.g., automotive, aerospace, factory automation, etc.). Many times analytical solutions do not exist for real problems that are normally complex structural systems. Approximate solutions are often used to solve these problems. The finite element method[1] is a powerful numerical solution technique that is used to solve vibration problems with complicating factors. Many of these factors are the result of complex geometries, nonuniform mass or stiffness distributions, and material or geometric nonlinearities. The finite element mathematical model is composed of a finite number of elements n. Upon global assembly, the linear second-order system becomes

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{F}$$

where $\mathbf{M}$ is the global mass matrix, $\mathbf{K}$ is the global stiffness matrix, and $\mathbf{F}$ is the external force vector. By performing an eigenvalue analysis, the equations of motion become decoupled. The resulting modal equations of motion are

$$\ddot{\mathbf{q}} + diag\left[\omega^2\right]\mathbf{q} = \mathbf{\Phi}^T\mathbf{f}$$

where ω contains distinctive eigenvalues or frequencies of the system. The matrix $\mathbf{\Phi}$ contains the system mode shapes.

Existing mode shapes for complex structures (from finite element modeling) can be recast to support quadratic modes development. For example, cubic spline functions can be employed to map the linear mode shapes into quadratic modes.

Example 3.5. Consider the cantilevered beam shown in Figure 3.2. For the following numerical values: $L = 10, EI = 1.4e4$, and $\rho = 1.2$ a 4-element model was

generated to approximate the cantilever beam. Cubic splines composed of piecewise cubic polynomials were defined as

$$\phi_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \quad 1 \leq i \leq n$$

over the interval $x_i \leq x \leq x_{i+1}$. A cubic polynomial was mapped between mode shape nodes (depending on the number and type of elements, a coarser mapping may be adopted). For comparison purposes, the analytical mode shapes were also generated from Eq. (3.5). The results for the first cantilever mode are shown in Figure 3.4. These include both the translational deflection $\phi(x)$ and rotational deflection $\phi_x(x)$, respectively that are generated by mapping the finite element mode shapes with polynomial functions (such as cubic splines). In later developments, quadratic modes can be created with Eq. (3.30). Other curve fitting methods (e.g., higher-order polynomials or numerical integration techniques) can also be utilized. Although this mapping is shown for a simple beam, the same procedure would be used for more complex structures that do not have analytical solutions.

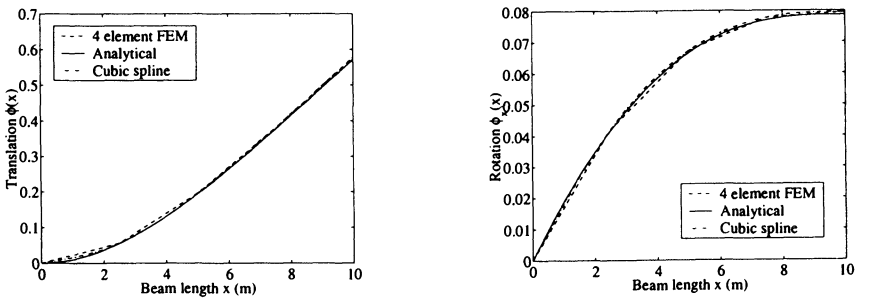

Figure 3.4. First cantilever beam mode shape data fitting comparisons

3.2.7. Mode Shape Discussion

As stated earlier, there is no unique method for selecting the basis functions ϕ_i. Any set that satisfies the kinematic boundary conditions and are linearly independent are acceptable. However, the quality of the approximate solution varies greatly with the choice of basis functions. Furthermore, if the mode shapes are orthogonal, then by definition the mass and stiffness matrices will be diagonal and the system equations will be decoupled. This feature may be beneficial for control system design.

One of the most important features of a mode shape (beyond the required conditions) is how well it approximates the true mode shapes of the analytical solution. For academic examples (e.g., free vibration), this can be demonstrated by showing

the degradation of the solution by assumed modes that deviate from the analytical solutions. For more complicated systems, it may be difficult to assess whether the chosen mode shape *resembles* the true ones since the true mode shapes are not known analytically. At this point, engineering judgment is required in the analysis. One approach is to use the true mode shapes associated with the closest analytically solvable system. Alternatively, the static deflection shapes for a variety of force basis functions could be used. Finally, if additional terms can be added to the mode shapes such that they satisfy not only the kinematic, but also the natural boundary conditions, then often the solution will be of higher fidelity since it meets more of the system constraints. One feature of the assumed modes approximation is that the approximate natural frequencies will always be greater than the true natural frequencies. As more basis functions are added, all the natural frequency approximations get closer to the true values. This can be exploited to assess the quality of the mode shapes used by iterating on the solution with additional mode shapes.

3.2.8. Rotational Fundamentals

Lagrange's equations are an attractive method for obtaining the equations of motion for a rigid body with three translational degrees-of-freedom and three rotational degrees-of-freedom. Of course, this requires one to obtain an expression for the energy of the system. Several methods exist to express the rotational degrees-of-freedom. One popular method, called Euler angles is often used to express the rotational portion of the kinetic energy expression.

Why not use the Cartesian components of the angular velocity to write the expression of rotational kinetic energy, or

$$T_{rot} = \frac{1}{2} \sum_i \sum_j I_{ij} \omega_i \omega_j.$$

This is a fine way to write the rotational kinetic energy. However, there are no generalized coordinates that have ω_i as their derivatives. Therefore, the Cartesian components of angular velocity are not well suited for use in Lagrange's equations.

Euler angles are suitable for use in Lagrange's equations because they are generalized coordinates for describing the orientation of a rigid body and their time derivatives can be used in representing the rotational kinetic energy of the system. Euler angles can be visualized as the orientation of the body-fixed xyz system relative to the fixed XYZ system. One assumes that the two coordinate systems are initially coincident. A series of rotations about the body axes, in a specified sequence, will permit the xyz system to reach any final orientation. This is shown graphically in Figure 3.5. Euler angles are used in the following sections to represent various combinations of rigid-body rotational degrees-of-freedom.

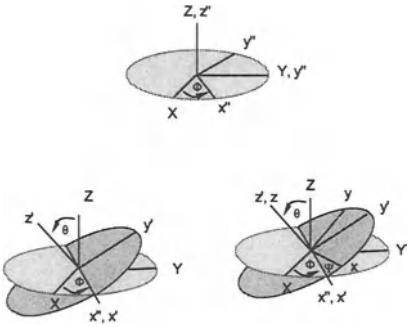

Figure 3.5: Rotate ϕ about the Z-axis; Rotate θ about the $x^{''}$-axis; Rotate Ψ about the $z^{'}$-axis

3.3. The Method of Quadratic Modes

Until now, the discussion on modeling distributed parameter systems has been focused on systems that have no rigid body motion (e.g., a cantilever beam). When a flexible body undergoes general rigid body motion, interesting coupling can occur that may be difficult to capture by using traditional assumed modes methods. For example, higher-order deformation terms not captured by the assumed modes expansion can couple with rigid body motion to form first-order, non-negligible, dynamic effects. The method of quadratic modes uses a systematic procedure to capture these effects in both the kinetic energy terms, and in the externally applied loads and gravity terms. The heart of the approach is an expansion of the deformation as

$$w(\chi,t) = \sum_{i=1}^{n} \phi_i(\chi) q_i(t) + \sum_{i=1}^{n} \sum_{j=1}^{n} g_{ij}(\chi) q_i(t) q_j(t) \tag{3.12}$$

where $\phi_i(\chi)q_i(t)$ are the same terms encountered from the assumed modes approach (see Eq. 3.9). The second term $g_{ij}(\chi)q_i(t)q_j(t)$ is called the quadratic mode expansion where the quadratic mode shapes $g_{ij}(\chi)$ are derived from the linear mode shapes $\phi_i(\chi)$. n is the number of mode shapes included in the expansion.

3.3.1. An Introductory Example

Before going into the details of the quadratic modes procedure, a one degree-of-freedom example of a rigid system is presented that highlights the main features of retaining higher-order components during the kinetic energy formulation.

Example 3.6. Free vibrations of a simple one degree-of-freedom rotating flexible system. A massless rod of length L is attached by a torsional spring with a spring constant k to a disk of radius r. Attached at the other end of the rod is a point mass m. The flexible system kinematic configuration is shown in Figure 3.6. The body and inertial coordinate frames are defined as $\hat{\mathbf{b}}_i$ and $\hat{\mathbf{n}}_i$, respectively for $i = 1, 2, 3$. The disk undergoes a prescribed rotation θ about the $\hat{\mathbf{n}}_3$ axis. By using the method of quadratic modes derive the differential equations of motion for the mass. Assume the deflections of the torsional spring are small $q \ll 1$. The hub is allowed to move through large rigid body rotations defined by the angle θ. Contrast your results with the *traditional* assumed modes method.

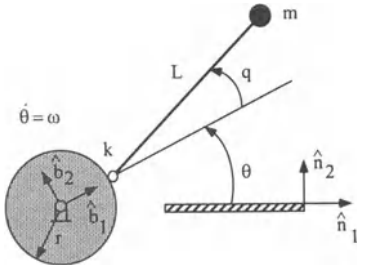

Figure 3.6. Simple 1-degree-of-freedom rotating flexible system

Solution

To illustrate the method of quadratic modes, higher-order kinematics are included in the formulation in order to capture the centrifugal stiffening effect. The motion of m is composed of an undeformed rigid body motion $\mathbf{p}(0)$ and a flexible body motion $\mathbf{u}(q)$. The flexible body motion will be linearized up to second-order in the generalized coordinates q. By using this representation, form the Lagrangian and develop the dynamic equations for this system.

The absolute position vector is

$$\begin{aligned}\mathbf{p}(q) &= \mathbf{p}(0) + \mathbf{u}(q) \\ &= (r + L\cos q)\hat{\mathbf{b}}_1 + L\sin q\hat{\mathbf{b}}_2. \end{aligned} \tag{3.13}$$

Solving for the flexible body motion yields

$$\begin{aligned}\mathbf{u}(q) &= \mathbf{p}(q) - \mathbf{p}(0) \\ &= L(\cos q - 1)\hat{\mathbf{b}}_1 + L\sin q\hat{\mathbf{b}}_2. \end{aligned} \tag{3.14}$$

Next, define Taylor series expansions for the trigonometric terms as

$$\cos(q) = 1 - \frac{1}{2!}q^2 + \frac{1}{4!}q^4 - \cdots$$
$$\sin(q) = q - \frac{1}{3!}q^3 + \frac{1}{5!}q^5 - \cdots,$$

and retain terms up to second-order and substitute into Eq. (3.14) to obtain

$$\begin{aligned}\mathbf{u}(q) &\approx \left(L\hat{\mathbf{b}}_2\right) q + \left(\frac{-L}{2}\hat{\mathbf{b}}_1\right) q^2 \\ &\approx \boldsymbol{\phi}_1 q + \mathbf{g}_{11} q^2. \end{aligned} \tag{3.15}$$

These equations (see Eq. 3.15) reflect the main idea of quadratic modes—that of forming the flexible body motion $\mathbf{u}(q)$ in terms of Eq. (3.12). It is observed that the linear and quadratic modes are

$$\boldsymbol{\phi}_1 = L\hat{\mathbf{b}}_2$$
$$\mathbf{g}_{11} = \frac{-L}{2}\hat{\mathbf{b}}_1.$$

With ϕ_1 as the magnitude of $\boldsymbol{\phi}_1$ and g_{11} as the magnitude of $\mathbf{g}_{11}$, or

$$\phi_1 = L \tag{3.16}$$
$$g_{11} = \frac{-L}{2}, \tag{3.17}$$

then the absolute position vector (see Eq. 3.13) becomes

$$\mathbf{p}(q) = (r + L + g_{11}q^2)\hat{\mathbf{b}}_1 + \phi_1 q\hat{\mathbf{b}}_2. \tag{3.18}$$

To evaluate the absolute velocity of the tip mass in the fixed-frame, the required body-frame coordinate kinematics are

$$\hat{\mathbf{b}}_1 = \cos\theta\hat{\mathbf{n}}_1 + \sin\theta\hat{\mathbf{n}}_2$$
$$\hat{\mathbf{b}}_2 = -\sin\theta\hat{\mathbf{n}}_1 + \cos\theta\hat{\mathbf{n}}_2.$$

The time derivatives become $\dot{\mathbf{b}}_1 = \omega\hat{\mathbf{b}}_2$ and $\dot{\mathbf{b}}_2 = -\omega\hat{\mathbf{b}}_1$ where $\omega = \dot{\theta}$. Taking the derivative of Eq. (3.18) gives

$$\dot{\mathbf{p}}(q) = \left(2g_{11}q\dot{q} - \dot{\theta}\phi_1 q\right)\hat{\mathbf{b}}_1 + \left(\phi_1\dot{q} + \dot{\theta}(r + L + g_{11}q^2)\right)\hat{\mathbf{b}}_2. \tag{3.19}$$

The kinetic energy T is expressed as

$$\begin{aligned} T &= \frac{1}{2}m\dot{\mathbf{p}}\cdot\dot{\mathbf{p}} \\ &= \frac{1}{2}m\left[(2g_{11}q\dot{q} - \dot{\theta}\phi_1 q)^2 + \left(\phi_1\dot{q} + \dot{\theta}(r + L + g_{11}q^2)\right)^2\right]. \end{aligned} \tag{3.20}$$

The strain energy U is expressed as

$$U = \frac{1}{2}kq^2. \tag{3.21}$$

Substitution of Eq. (3.20) and Eq. (3.21) into the Lagrangian $L = T - U$ gives

$$L = \frac{1}{2}m\left(2g_{11}q\dot{q} - \dot{\theta}\phi_1 q\right)^2 + \frac{1}{2}m\left(\phi_1\dot{q} + \dot{\theta}(r + L + g_{11}q^2)\right)^2 - \frac{1}{2}kq^2. \tag{3.22}$$

Applying Lagrange's equation and linearizing with respect to q gives

$$m\phi_1^2\ddot{q} + \left[k - \left(m\phi_1^2 + 2mg_{11}(r + L)\omega^2\right)\right]q = -m\phi_1(r + L)\dot{\omega}. \tag{3.23}$$

Next, substitute Eq. (3.16) into Eq. (3.23) and normalize, or

$$\ddot{q} + \left[\omega_n^2 - \omega^2 - 2\omega^2\left(\frac{r}{L} + 1\right)\left(\frac{g_{11}}{L}\right)\right]q = -\dot{\omega}\left(\frac{r}{L} + 1\right) \tag{3.24}$$

where

$$\omega_n^2 = \frac{k}{mL^2}.$$

The substitution of Eq. (3.17) into Eq. (3.24) produces

$$\ddot{q} + \left[\omega_n^2 + \omega^2\frac{r}{L}\right]q = -\dot{\omega}\left(\frac{r}{L} + 1\right). \tag{3.25}$$

If only linear kinematics are considered, then g_{11} is zero in Eq. (3.24). The result is

$$\ddot{q} + \left[\omega_n^2 - \omega^2\right]q = -\dot{\omega}\left(\frac{r}{L} + 1\right). \tag{3.26}$$

Discussion

It is evident from the comparison of Eq. (3.25) and Eq. (3.26) that the two approaches (quadratic modes and traditional assumed modes) result in different linearized equations of motion. The centrifugal stiffening of the system is predicted correctly when the quadratic kinematics are included in the derivation. This situation is in contrast to the equations of motion based on linear kinematics, which indicate softening or buckling of the system. The differences between the two approaches becomes significant when the absolute value of the spin rate is of the same order-of-magnitude as ω_n. Specifically, if the quadratic modes are ignored ($\mathbf{g}_{11} \approx 0$), then the dynamic equations predict that the beam softens as the rotational speed is increased. Of course, this is not correct.

Example 3.6 illustrates that a traditional assumed modes approach based on linear kinematics may lead to spurious equations of motion unless some corrective measure is taken. The corrective measure taken in the references[5,6] was to include geometric stiffness matrices in the analysis. The present approach does not require the use of such matrices, as geometric stiffening is accounted for implicitly by the quadratic modes.

3.3.2. Computing Quadratic Mode Shapes

The method of quadratic modes is an assumed modes approach where the approximated kinetic and potential energy functions are formed by using kinematic relationships that are accurate up to second-order. For a beam in bending, this means the displacement of any point along the beam must capture both the axial and transverse components. In the traditional assumed modes approach, the energy functions would only capture the first-order transverse motion. Note, the quadratic mode shapes $\mathbf{g}_{ij}$ are not independent of the linear mode shapes ϕ. Once the linear mode shapes have been agreed upon, the quadratic mode shapes must be computed as a function of all the 2-tuple combinations of ϕ. Determining the functional relationship between $\mathbf{g}_{ij}$ and ϕ depends on the structure being considered and may not have an easily obtained analytical solution. For illustration purposes, the analytical equations for the quadratic modes of various beam configurations will be described as a series of examples.

A practical approach developed by Segalman and Dohrmann[7,8] works for general structures. First, the structure is modeled by using a finite element code with nonlinear deformation capability. Next, a set of independent loading configurations is used to generate a corresponding set of static deflection functions. The first-order deflections of these deformations are used as the linear mode shapes. The second-order deformations become the $\mathbf{g}_{ij}$ mode shapes. The $\mathbf{g}_{ij}$ $(i \neq j)$ are found by forming all possible combinations of force basis functions and extracting the second-order deformations. This method has been applied to general structures[8].

To illustrate how to compute quadratic modes, two separate methods are presented. Both methods exploit the following assumptions used in the derivations:

1. The beam is constrained to be inextensible in the axial direction.
2. Motion of a material point due to deformation in the lateral direction is much greater than in the longitudinal direction.

Method 1

Consider an undeformed beam segment of length dx whose endpoints have Cartesian coordinates $p_1 = (x, 0)$ and $p_2 = (x + dx, 0)$. Next, compare the location of the endpoints for the same segment when the beam is in a slightly deformed configuration (points p_1' and p_2'). The geometry definitions are shown in Figure 3.7. The second-order longitudinal displacement is defined as u_x. The first-order lateral displacement is defined as y. Both of these displacements vary over the length of the beam. By assumption 1, the segment lengths in both the deformed and undeformed configuration must be the same, or

$$\|p_2 - p_1\|^2 = \|p_2' - p_1'\|^2 .$$

The results is

$$dx^2 = \left[dx + \frac{du_x}{dx}dx\right]^2 + \left[\frac{dy}{dx}dx\right]^2 .$$

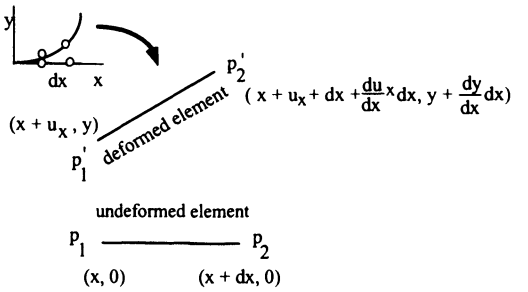

Figure 3.7. Undeformed and deformed small segment of a beam

Dividing by dx^2 yields

$$1 = 1 + 2\frac{du_x}{dx} + \left[\frac{du_x}{dx}\right]^2 + \left[\frac{dy}{dx}\right]^2.$$

By assumption 2, this reduces to

$$1 = 1 + 2\frac{du_x}{dx} + \left[\frac{dy}{dx}\right]^2.$$

Solving for u_x gives the quadratic modes definition

$$u_x = -\frac{1}{2}\int_0^x \left[\frac{dy}{dx}\right]^2 dx. \tag{3.27}$$

Method 2

Since the beam is inextensible (assumption 1), the axial strain must be zero, or

$$\varepsilon_{xx} = \left\|\frac{\partial \mathbf{w}}{\partial \chi}\right\| - 1 = 0 \tag{3.28}$$

where $\mathbf{w}$ is the deformation expanded as in Eq. (3.12). First, define the motion of a particle originally located at χ as

$$\mathbf{w}(\chi) = (\chi + g_{ij}q_iq_j)\hat{\mathbf{i}} + \phi_i q_i \hat{\mathbf{j}}.$$

Take the partial derivative for $\mathbf{w}$

$$\frac{\partial \mathbf{w}}{\partial \chi} = (1 + g'_{ij}q_iq_j)\hat{\mathbf{i}} + \phi'_i q_i \hat{\mathbf{j}},$$

This leads to a magnitude, or

$$\left\| \frac{\partial \mathbf{w}}{\partial \chi} \right\| = \sqrt{1 + q_i q_j (2g'_{ij} + \phi'_i \phi'_j) + q_i q_j q_k q_p g'_{ij} g'_{kp}}.$$

The quantity on the right has the form $\sqrt{1+x}$ and can be expanded in a binomial series as

$$\sqrt{1+x} \approx 1 + \frac{1}{2}x.$$

Therefore,

$$\left\| \frac{\partial \mathbf{w}}{\partial \chi} \right\| \approx 1 + \frac{1}{2} q_i q_j \left(2g'_{ij} + \phi'_i \phi'_j\right). \tag{3.29}$$

Substitute Eq. (3.29) into the axial strain constraint equation (see Eq. 3.28) and solve for g_{ij} to obtain

$$g_{ij} = -\frac{1}{2} \int_0^{\chi} \phi'_i \phi'_j d\xi, \tag{3.30}$$

which is analogous to the result obtained by Method 1. In the following example, the method of quadratic modes is applied to a rotating flexible beam.

Example 3.7. Consider the free vibrations of a beam cantilevered to a rotating hub. The thin extensible beam of length L, mass per unit length m, modulus of elasticity E, and area moment of inertia I is free to bend in only the y-direction. The beam is cantilevered to a hub, that is allowed to rotate under a prescribed rotation θ. The geometry is defined in Figure 3.8. The body and inertial coordinate frames are defined as $\hat{\mathbf{b}}_i$ and $\hat{\mathbf{n}}_i$, respectively for $i = 1, 2, 3$. Generate the dynamic equations of motion for the system by using the method of quadratic modes.

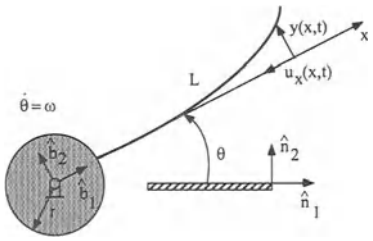

Figure 3.8. Flexible beam attached to a rotating hub

Solution

Use Lagrange's equations where the kinetic and potential energy contain both linear and second-order deformation kinematics. The position vector from the origin of the inertial frame to any point along the beam is

$$\mathbf{p}(\chi,t) = [r+\chi]\hat{\mathbf{b}}_1 + \mathbf{u}(\chi,t)$$

where the flexible body motion is

$$\mathbf{u}(\chi,t) = u_\chi(\chi,t)\hat{\mathbf{b}}_1 + y(\chi,t)\hat{\mathbf{b}}_2.$$

The linear mode shape is approximated as

$$\phi(\chi) = \left[3L\chi^2 - \chi^3\right]\hat{\mathbf{b}}_2, \tag{3.31}$$

or equivalently

$$y(\chi,t) = \left[3L\chi^2 - \chi^3\right]q(t) \tag{3.32}$$

from a static loading condition derived in Section 3.2.4.. The quadratic component u_χ is determined by using Eq. (3.27), or

$$u_\chi(\chi,t) = -\frac{1}{2}\left[\frac{9}{5}\chi^5 - 9L\chi^4 + 12L^2\chi^3\right]q^2.$$

In summary,

$$\mathbf{u}(\chi,t) = \phi(\chi)q\hat{\mathbf{b}}_2 + g_{11}(\chi)q^2\hat{\mathbf{b}}_1 \tag{3.33}$$

where Eq. (3.33) takes a form analogous to Eq. (3.12). The linear and quadratic modes are

$$\phi(\chi) = 3L\chi^2 - \chi^3 \tag{3.34}$$

$$g_{11}(\chi) = -\frac{1}{2}\left[\frac{9}{5}\chi^5 - 9L\chi^4 + 12L^2\chi^3\right]. \tag{3.35}$$

The velocity of each material point along the beam's length is given by

$$\begin{aligned}
\dot{\mathbf{p}}(\chi,t) &= {}^n\!\frac{d}{dt}\left[[r+\chi]\hat{\mathbf{b}}_1 + \mathbf{u}(\chi,t)\right] \\
&= {}^n\!\frac{d}{dt}\left[[r+\chi+g_{11}(\chi)q^2]\hat{\mathbf{b}}_1 + \phi(\chi)q\hat{\mathbf{b}}_2\right] \\
&= \left[2g_{11}(\chi)q\dot{q} - \omega\phi(\chi)q\right]\hat{\mathbf{b}}_1 + \left[\phi(\chi)\dot{q} + \omega(r+\chi+g_{11}(\chi)q^2)\right]\hat{\mathbf{b}}_2.
\end{aligned}$$

The kinetic energy is

$$T = \frac{1}{2}\int_0^L \rho(\chi)\dot{\mathbf{p}}(\chi,t)\cdot\dot{\mathbf{p}}(\chi,t)d\chi. \tag{3.36}$$

The strain energy U is

$$U = \frac{1}{2}\int_0^L EI(\chi)\left(\frac{\partial^2 y(\chi,t)}{\partial \chi^2}\right)^2 d\chi$$
$$= \frac{1}{2}\int_0^L EI\left[\phi''(\chi)\right]^2 d\chi q^2. \tag{3.37}$$

Substituting Eq. (3.36) and Eq. (3.37), respectively into Lagrange's equation and linearizing the result leads to (after a lot of algebra!)

$$\left(\int_0^L \rho[\phi(\chi)]^2 d\chi\right)\ddot{q} + \left(EI\int_0^L [\phi''(\chi)]^2 d\chi - \left(\int_0^L \rho\left[[\phi(\chi)]^2 + 2[r+\chi]g_{11}(\chi)\right]d\chi\right)\omega^2\right)q =$$
$$-\left(\int_0^L \rho[r+\chi]\phi(\chi)d\chi\right)\dot{\omega}. \tag{3.38}$$

Substituting Eq. (3.34) and Eq. (3.35) (for $\phi(\chi)$ and $g_{11}(\chi)$, respectively) gives

$$\ddot{q} + \left(\frac{140EI}{11\rho L^4} + \frac{1}{22}\left[35\frac{r}{L} + 5\right]\omega^2\right)q = -\frac{\dot{\omega}}{L^2}\left(\frac{35r}{44L} + \frac{7}{12}\right). \tag{3.39}$$

In contrast, the equation obtained without the quadratic mode is

$$\ddot{q} + \left(\frac{140EI}{11\rho L^4} - \omega^2\right)q = -\frac{\dot{\omega}}{L^2}\left(\frac{35r}{44L} + \frac{7}{12}\right). \tag{3.40}$$

Example 3.8. This example demonstrates the centrifugal stiffening effects both with (Eq. 3.39) and without (Eq. 3.40) the quadratic mode. Use the following numerical values: $L = 10$, $r = 0$, $EI = 1.4e4$, $\rho = 1.2$, $\Omega = 6$, and $T = 15$. Simulate the free vibrations of a beam cantilevered to a rotating hub. Use the dynamic equations (see Eq. 3.39 and Eq. 3.40) developed in Example 3.7. The prescribed rigid body motion is defined as

$$\omega(t) = \frac{\Omega}{T}\left[t - \left(\frac{T}{2\pi}\right)\sin\left(\frac{2\pi t}{T}\right)\right]$$
$$\dot{\omega}(t) = \frac{\Omega}{T}\left[1 - \cos\left(\frac{2\pi t}{T}\right)\right]$$

for $0 \leq t \leq T$ and for $t > T$, $\dot{\omega}(t) = 0$ and $\omega(t) = \Omega$.

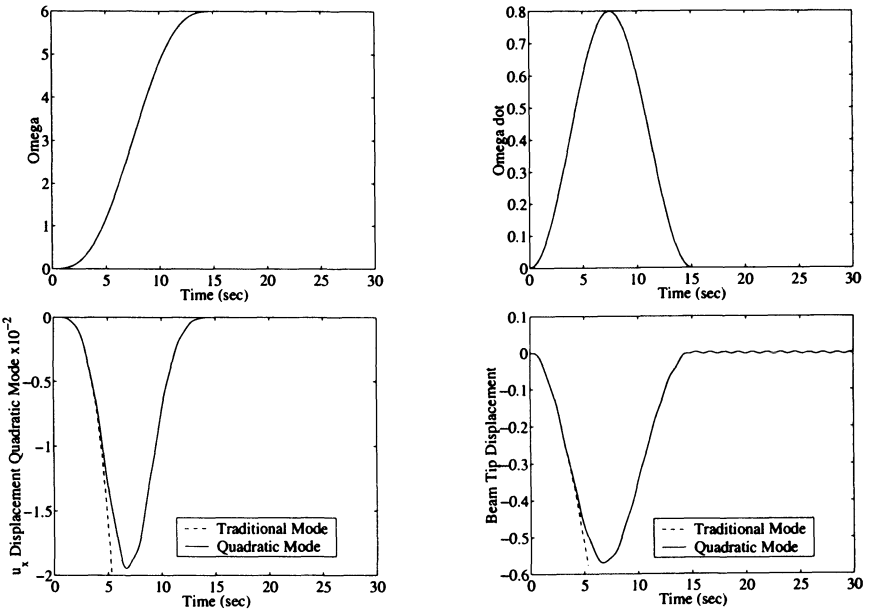

Figure 3.9. Rotating cantilever beam numerical results for Example 3.8

Solution

The numerical results are shown in Figure 3.9. The prescribed motion for the rotating hub include angular rotation rate and acceleration profiles (see upper plots in Figure 3.9). The bottom two plots in Figure 3.9 show the axial and tip displacement responses. These responses include both traditional assumed modes Eq. (3.40) and quadratic modes Eq. (3.39). The traditional assumed modes approach predicts premature softening/buckling of the beam.

3.3.3. Formal Quadratic Modes Equations

For the purposes of developing Lagrange's equations, the quadratic mode development has proceeded strictly from the idea of expanding the kinematics up to second-order in the deformation. In the original published development of the method of quadratic modes[7,8], more general equations were developed from a general evaluation of Lagrange's equations. The advantage of this formulation is that the complexity of forming the energy quantities, which are sometimes quite involved, is not required. In the remainder of this section, two versions of these equations will be presented followed by two examples.

Inertial Reference Frame Equations

The quadratic mode equations are given below where the inertial frame is the reference frame used to describe the motion of the body.

Translational

$$\int_{vol}\int\int \delta\hat{\mathbf{p}}\cdot\left(-\rho\ddot{\hat{\mathbf{x}}}+\hat{\mathbf{F}}_{ext}\right)dV=0. \tag{3.41}$$

Rotational

$$\int_{vol}\int\int \left(\hat{\boldsymbol{\lambda}}\times\{\mathbf{R}\cdot[\boldsymbol{\chi}+q_i\boldsymbol{\phi}_i]\}\right)\cdot\left(-\rho\ddot{\hat{\mathbf{x}}}+\hat{\mathbf{F}}_{ext}\right)dV=0. \tag{3.42}$$

Deformational

$$\delta q_i \int_{vol}\int\int (\mathbf{R}\cdot[\boldsymbol{\phi}_i+2q_j\mathbf{g}_{ij}])\cdot\left(-\rho\ddot{\hat{\mathbf{x}}}+\hat{\mathbf{F}}_{ext}-\hat{\mathbf{F}}_s\right)dV=0. \tag{3.43}$$

$\ddot{\hat{\mathbf{x}}}(\boldsymbol{\chi},t)$	=	acceleration of a particle in the inertial frame
$\hat{\mathbf{p}}(t)$	=	translation of the reference configuration in inertial frame
$\hat{\mathbf{F}}_{ext}$	=	external force in inertial frame
$\mathbf{R}$	=	rotation matrix of the reference configuration in inertial frame
$\hat{\boldsymbol{\lambda}}$	=	vector of virtual rotations
$\boldsymbol{\chi}$	=	reference location of some particle
$q_i(t)$	=	i^{th} generalized coordinate for the i^{th} assumed mode shape
$\boldsymbol{\phi}_i(\boldsymbol{\chi})$	=	i^{th} deformation mode shape
$\mathbf{g}_{ij}(\boldsymbol{\chi})$	=	i^{th}, j^{th} quadratic mode shape
$\hat{\mathbf{F}}_s$	=	static reaction force for a beam

Body Reference Frame Equations

The quadratic mode equations are given below where a body fixed frame is the reference frame used to describe the motion of the body.

Translational

$$\int_{vol}\int\int (-\rho\ddot{\mathbf{x}}+\mathbf{F}_{ext})\,dV=0. \tag{3.44}$$

Rotational

$$\int_{vol}\int\int [\boldsymbol{\chi}+q_i\boldsymbol{\phi}_i]^T\,\boldsymbol{\Lambda}\,(-\rho\ddot{\mathbf{x}}+\mathbf{F}_{ext})\,dV=0. \tag{3.45}$$

Deformational

$$\delta q_i \int_{vol} \int \int \left([\phi_i + 2q_j \mathbf{g}_{ij}]^T \right) (-\rho \ddot{\mathbf{x}} + \mathbf{F}_{ext} - \mathbf{F}_s)\, dV = 0. \tag{3.46}$$

$\mathbf{\Lambda}$ is determined from the relationship $\delta \mathbf{R} = \mathbf{\Lambda}^T \mathbf{R}$.

Example 3.9. Consider the dynamics of the unidirectional rotating beam in gravity. Shown in Figure 3.10 is the rotating flexible link. A single revolute joint θ rotates in a vertical plane. Gravity acts in the negative $\hat{Y}$-direction. Develop both the rotational and deformational equations of motion by using the method of quadratic modes.

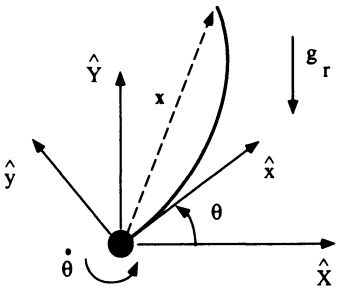

Figure 3.10. Kinematics for a unidirectional rotating beam in gravity

Solution
The position vector to any point along the beam is

$$\mathbf{x} = (\chi + g_{ij} q_i q_j)\hat{\mathbf{i}} + \phi_i q_i \hat{\mathbf{j}}.$$

Define the relevant quadratic modes terms as

$$\ddot{\mathbf{x}} = \left(-\chi \dot{\theta}^2 - q_i \phi_i \ddot{\theta} - 2\dot{q}_i \phi_i \dot{\theta} \right) \hat{\mathbf{i}} + \left(\chi \ddot{\theta} - \phi_i q_i \dot{\theta}^2 + \phi_i \ddot{q}_i \right) \hat{\mathbf{j}} \tag{3.47}$$

$$\phi_i = \phi_i \hat{\mathbf{j}}$$

$$\mathbf{F}_s = EI q_i \phi_i^{iv} \hat{\mathbf{j}}$$

$$\mathbf{g}_{ij} = g_{ij} \hat{\mathbf{i}}$$

$$\mathbf{F}_{ext} = -\rho g_r \cos\theta \hat{\mathbf{j}} - \rho g_r \sin\theta \hat{\mathbf{i}} + \tau\eta(\chi)\hat{\mathbf{j}}, \tag{3.48}$$

and inserting into the Eq. (3.46) (Deformational) gives

$$(\delta q_i)\int_0^L \left(\begin{bmatrix} 2q_i g_{ij} \\ \phi_i \end{bmatrix} \cdot \begin{bmatrix} \rho\left(\chi\dot{\theta}^2 + q_i\phi_i\ddot{\theta} + 2\dot{q}_i\phi_i\dot{\theta}\right) - \rho g_r \sin\theta \\ \rho\left(\phi_i q_i\dot{\theta}^2 - \chi\ddot{\theta} - \phi_i\ddot{q}_i\right) - \rho g_r\cos\theta - EIq_i\phi_i^{iv} + \tau\eta(\chi) \end{bmatrix} \right) d\chi = 0.$$

After performing the indicated computations, the results are

$$\begin{aligned} \rho\ddot{q}_i\int_0^L \phi_i\phi_j d\chi + \rho\ddot{\theta}\int_0^L \chi\phi_i d\chi + EIq_i\int_0^L \phi_i''\phi_j'' d\chi \\ - q_i\dot{\theta}^2\left[2\rho\int_0^L \chi g_{ij}d\chi + \rho\int_0^L \phi_i\phi_j d\chi\right] \\ + 2\rho g_r\cos\theta\int_0^L \phi_i d\chi + \rho g_r q_i\sin\theta\int_0^L g_{ij}d\chi = \tau\frac{d}{d\chi}(\phi_i)\Big|_{\chi=0}. \end{aligned} \tag{3.49}$$

Equation (3.45) (Rotational) becomes

$$\int_0^L [\boldsymbol{\chi} + q_i\boldsymbol{\phi}_i]^T\boldsymbol{\Lambda}(-\rho\ddot{\mathbf{x}} + \mathbf{F}_{ext})d\chi = 0. \tag{3.50}$$

The matrix of virtual rotations $\boldsymbol{\Lambda}$ is defined by

$$\delta\mathbf{R} = \Lambda^T\mathbf{R}.$$

For this example, the rotation matrix $\mathbf{R}$ is

$$\mathbf{R} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Treating the virtual operator δ the same as the partial differential operator gives

$$\delta\mathbf{R} = \begin{bmatrix} -\delta\theta\sin\theta & \delta\theta\cos\theta & 0 \\ -\delta\theta\cos\theta & -\delta\theta\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

From this, $\boldsymbol{\Lambda}$ is

$$\boldsymbol{\Lambda} = \begin{bmatrix} 0 & -\delta\theta & 0 \\ \delta\theta & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The external *force* (actually a torque) due to the hub motor is

$$\mathbf{F}_{motor} = \tau\eta(\chi)\hat{\mathbf{j}}.$$

Here, the doublet function is used to represent two equal in magnitude, but opposite direction impulses occurring at the hub in the rotating $\hat{\mathbf{j}}$-direction (see Figure 3.11). Equation (3.50) becomes

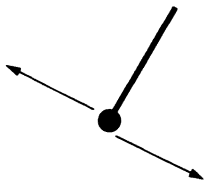

Figure 3.11. Two opposing direction impulses to represent hub torque

$$\int_0^L \begin{bmatrix} \chi \\ q_i\phi_i \\ 0 \end{bmatrix}^T \cdot \begin{bmatrix} 0 & -\delta\theta & 0 \\ \delta\theta & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} (-\rho\ddot{\mathbf{x}} + \mathbf{F}_{ext}) d\chi = 0. \tag{3.51}$$

Substituting Eq. (3.47) and Eq. (3.48) into Eq. (3.51) gives

$$\delta\theta \int_0^L \begin{bmatrix} q_i\phi_i \\ -\chi \\ 0 \end{bmatrix} \cdot \begin{bmatrix} \rho\left(\chi\dot{\theta}^2 + q_i\phi_i\ddot{\theta} + 2\dot{q}_i\phi_i\dot{\theta}\right) - \rho g_r \sin\theta \\ \rho\left(q_i\phi_i\dot{\theta}^2 - \chi\ddot{\theta} - \ddot{q}_i\phi_i - g_r\cos\theta\right) + \tau\eta(\chi) \\ 0 \end{bmatrix} d\chi = 0.$$

Performing the indicated inner product and integration gives

$$\frac{1}{3}\rho L^3\ddot{\theta} + \rho\ddot{q}_i \int_0^L \chi\phi_i d\chi - \rho g_r q_i \sin\theta \int_0^L \phi_i d\chi + \frac{1}{2}\rho g_r L^2 \cos\theta = \tau. \tag{3.52}$$

Equations Eq. (3.49) and Eq. (3.52) describe the dynamics for a rotating beam in a vertical plane subject to a gravitational field. For a beam rotating in a horizontal plane, gravity is zero. In addition, the effect of a tip mass can be included. This modified model is used in Chapter 8, Example 8.2 (gravity neglected), and (with a slight modification to the kinematics) also used in Chapter 5, Section 5.4.11..

Example 3.10. Consider the dynamics of a bidirectional, spherically rotating beam in gravity. The rod is free to vibrate in two transverse directions. It is connected to a motor by using a U-joint. Motors at the U-joint allow torques to be applied to the connection point causing rigid body rotation. Compute the form of the dynamic equations of motion. Assume that the motion occurs in a gravitational field that effects deformation.

Solution

Two Euler angles are used to describe the orientation of the undeformed beam due to the U-joint rotations at the hub. It is assumed that two independent torques can be applied in the direction of the two Euler angles. The Euler angle descriptions are shown in Figure 3.12. The rotation sequence is as follows:

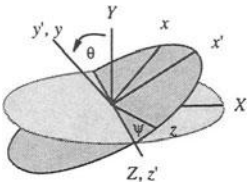

Figure 3.12. Euler angles used to describe spherical rotation

1. Rotate the XYZ system through θ about the positive Z-axis to obtain the $x'y'z'$ system.

2. Rotate the $x'y'z'$ system through ψ about the positive y'-axis to obtain the xyz system.

The total rotation matrix $\mathbf{R}$ is the transformation matrix from the XYZ system to the xyz system, or

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}.$$

The total rotation matrix $\mathbf{R}$ can be expressed as the product of two simpler rotational matrices corresponding to the two Euler angles, i.e., $\mathbf{R} = \mathbf{R}_2\mathbf{R}_1$, or

$$\mathbf{R} = \begin{bmatrix} \cos\psi & 0 & -\sin\psi \\ 0 & 1 & 0 \\ \sin\psi & 0 & \cos\psi \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The quantities necessary to evaluate the rotational and deformational equations of

motion are

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & -\dot{\theta} & \dot{\psi} \\ \dot{\theta} & 0 & 0 \\ -\dot{\psi} & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{\phi}_i = \phi_i^{\alpha}\hat{\mathbf{j}} + \phi_i^{\beta}\hat{\mathbf{k}}$$

$$\boldsymbol{\chi} = \chi\hat{\mathbf{i}}$$

$$\ddot{\mathbf{x}} = (\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega}^2)(\boldsymbol{\chi} + q_i\boldsymbol{\phi}_i) + 2\boldsymbol{\Omega}\dot{q}_i\boldsymbol{\phi}_i + \ddot{q}_i\boldsymbol{\phi}_i$$

where only terms up to first-order in q_i have been retained. Making the appropriate substitutions gives

$$\ddot{\mathbf{x}} = \begin{bmatrix} -\chi(\dot{\theta}^2 + \dot{\psi}^2) - \ddot{\theta}q_i^{\alpha}\phi_i^{\alpha} + \ddot{\psi}q_i^{\beta}\phi_i^{\beta} - 2\dot{\theta}\dot{q}_i^{\alpha}\phi_i^{\alpha} + 2\dot{\psi}\dot{q}_i^{\beta}\phi_i^{\beta} \\ \chi\ddot{\theta} - \dot{\theta}^2 q_i^{\alpha}\phi_i^{\alpha} + \dot{\theta}\dot{\psi}q_i^{\beta}\phi_i^{\beta} + \ddot{q}_i^{\alpha}\phi_i^{\alpha} \\ -\chi\ddot{\psi} + \dot{\theta}\dot{\psi}q_i^{\alpha}\phi_i^{\alpha} - \dot{\psi}^2 q_i^{\beta}\phi_i^{\beta} + \ddot{q}_i^{\beta}\phi_i^{\beta} \end{bmatrix}$$

$$\mathbf{F}_s = (EI)^{\alpha}q_i^{\alpha}(\phi_i^{\alpha})^{iv}\hat{\mathbf{j}} + (EI)^{\beta}q_i^{\beta}(\phi_i^{\beta})^{iv}\hat{\mathbf{k}}$$

$$\mathbf{F}_{ext} = \mathbf{F}_{grav} + \tau_1\eta(\chi)\hat{\mathbf{j}} + \tau_2\eta(\chi)\hat{\mathbf{k}}$$

$$\mathbf{F}_{grav} = \mathbf{R}\begin{bmatrix} 0 \\ -\rho g_r \\ 0 \end{bmatrix} = -\rho g_r \begin{bmatrix} \cos\psi\sin\theta \\ \cos\theta \\ \sin\psi\sin\theta \end{bmatrix}.$$

There will be two sets of deformation equations—one for the vertical α and one for the horizontal β directions of deformation, respectively. Inserting into Eq. (3.46) (Deformational) for α gives

$$(\delta q_i^{\alpha})\int_0^L \begin{bmatrix} 2q_j^{\alpha}g_{ij}^{\alpha} \\ \phi_i^{\alpha} \\ 0 \end{bmatrix} \cdot \begin{bmatrix} -\rho\ddot{x}_x + F_{ext_x} \\ -\rho\ddot{x}_y + F_{ext_y} - F_{s_y} \\ -\rho\ddot{x}_z + F_{ext_z} - F_{s_z} \end{bmatrix} d\chi = 0. \tag{3.53}$$

After performing the indicated computations, the results are

$$\rho\ddot{q}_i^{\alpha}\int_0^L \phi_i^{\alpha}\phi_j^{\alpha}d\chi + \rho\ddot{\theta}\int_0^L \chi\phi_i^{\alpha}d\chi + (EI)^{\alpha}q_i^{\alpha}\int_0^L (\phi_i^{\alpha})''(\phi_j^{\alpha})''d\chi$$
$$- 2\rho q_i^{\alpha}\left[(\dot{\theta}^2 + \dot{\psi}^2)\int_0^L \chi g_{ij}^{\alpha}\,d\chi + \frac{1}{2}\dot{\theta}^2\int_0^L \phi_i^{\alpha}\phi_j^{\alpha}d\chi\right]$$
$$+ 2\rho g_r\cos\psi\sin\theta\int_0^L g_{ij}^{\alpha}\,d\chi + \rho g_r\cos\theta\int_0^L \phi_i^{\alpha}d\chi = \tau_1\frac{d}{d\chi}(\phi_i^{\alpha})\big|_{\chi=0}.$$

To determine the deformational equations of motion for β, computations similar to Eq. (3.53) can be performed. Note, the ϕ_i^β component would be inserted in the z-location. Finally, the rotational equations of motion can be derived by using Eq. (3.45). The production of these final equations are left as an exercise for the reader.

3.3.4. Multibody Quadratic Modes

This method uses Hamilton's principle for each link rather than for the complete multibody system. This requires resolving away the constraint forces. The advantage of this method is that the form of the equations for each link is identical and need only be performed once. The recursive substitution of constraint forces can be automated by using symbolic mathematics software packages.

The multibody system consists of n-links and n-revolute joints. The first joint (the hub) is in general not fixed in an inertial reference frame. The first link is of arbitrary geometry and is connected to the hub on its inboard side and to the second joint on its outboard side. The multibody system of joints and links is terminated by the n^{th}-link. Each joint has one rotational degree-of-freedom and a torque producing actuator. The solution procedure[9] is described by a sequence of steps. To manually obtain the equations of motion for even a two-link planar flexible robot is difficult. Symbolic equation solvers, such as Mathematica® or Maple® are recommended.

The coordinate systems, torques, and reaction forces for each flexible link/joint subsystem are defined in Figure 3.13. Note, the small rotation at joint $k+1$ due to the deformation of link k is denoted by $\tilde{\theta}_k$. The multibody quadratic mode procedure[9] is described next.

Step 1. Evaluate the equations of motion for translation, rotation, and deformation of the k^{th}-link. This is identical to the Examples 3.7 and 3.9, but with the external force F_{ext} consisting of gravity, torques, and reaction forces at both the inboard and outboard links. These reaction forces are denoted by F_{R_k} and $F_{R_{k+1}}$, respectively.

Step 2. Resolve out the unknown reaction forces F_{R_k} recursively. Start by solving the translational equation of motion for the n^{th} reaction force F_{R_n} (function of $\ddot{p}_n$). Inboard reaction forces can be evaluated successively inward up to the hub. If the hub is fixed, then F_{r_1} is not required since it is a workless constraint force. These reaction forces will be expressed in terms of the joint translational accelerations ($\ddot{p}_i$). These accelerations can be written as functions of the angular velocity vectors and angular acceleration vectors of each joint, and the generalized velocities and generalized accelerations of the deformation for each link. This step eliminates the translational equations of motion by eliminating the reaction force terms and leaving only the rotational and deformational equations.

The interested reader is encouraged to consult reference[9] for further details.

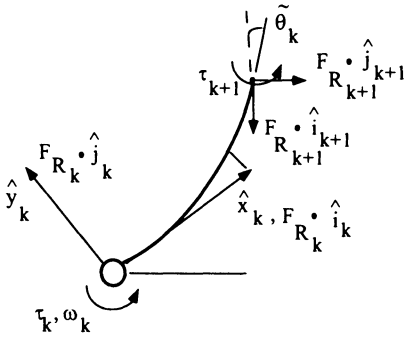

Figure 3.13. The k^{th} joint and link of a flexible multibody system

3.4. Planar Flexible Robot Dynamics

The second method for modeling flexible planar robots is based on material taken from reference[10]. The key points of the method are as follows:

- Links are modeled by using specially formulated beam elements.
- Generalized coordinates are nodal rotations of the beam elements.
- Equations of motion are obtained by using Lagrange's equations.

The assumed form of the kinematics is given by

$$x(s,t) = x_i(t) + \int_0^s \cos\theta(r,t)dr \tag{3.54}$$

$$y(s,t) = y_i(t) + \int_0^s \sin\theta(r,t)dr \tag{3.55}$$

where the meaning of x, y, and θ are shown in Figure 3.14.

The integrals in Eq. (3.54) and Eq. (3.55) are evaluated by assuming each integrand is constant and equal to the average of its values at the endpoints. Thus,

$$x(s,t) = x_i(t) + \frac{1}{2}\int_0^s (\cos\theta_j(t) + \cos\theta_{j+1}(t))dr \tag{3.56}$$

$$y(s,t) = x_i(t) + \frac{1}{2}\int_0^s (\sin\theta_j(t) + \sin\theta_{j+1}(t))dr. \tag{3.57}$$

The approximation used in Eq. (3.56) and Eq. (3.57) is justified as long as $(\theta_{j+1} - \theta_j) \ll 1$. Carrying out the integrations in Eq. (3.56) and Eq. (3.57) yields

$$x(s,t) = x_i + s(\cos\theta_j + \cos\theta_{j+1})/2 \tag{3.58}$$

$$y(s,t) = y_i + s(\sin\theta_j + \sin\theta_{j+1})/2. \tag{3.59}$$

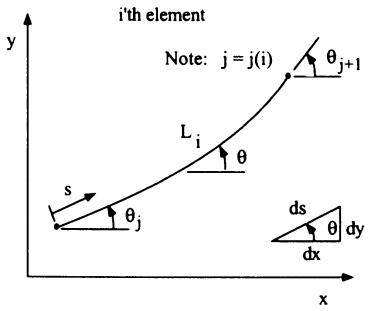

Figure 3.14. Kinematic definitions for the i^{th} element

Differentiating Eq. (3.58) and Eq. (3.59) with respect time gives

$$\dot{x}(s,t) = \dot{x}_i - s(\dot{\theta}_j \sin\theta_j + \dot{\theta}_{j+1}\sin\theta_{j+1})/2 \quad (3.60)$$

$$\dot{y}(s,t) = \dot{y}_i + s(\dot{\theta}_j \cos\theta_j + \dot{\theta}_{j+1}\cos\theta_{j+1})/2. \quad (3.61)$$

Setting $s = L_i$ in Eq. (3.60) and Eq. (3.61) yields

$$\dot{x}_{i+1} = \dot{x}_i - L_i(\dot{\theta}_j \sin\theta_j + \dot{\theta}_{j+1}\sin\theta_{j+1})/2 \quad (3.62)$$

$$\dot{y}_{i+1} = \dot{y}_i + L_i(\dot{\theta}_j \cos\theta_j + \dot{\theta}_{j+1}\cos\theta_{j+1})/2. \quad (3.63)$$

Combining Eqs. (3.60)–(3.63) leads to

$$\dot{x}(s,t) = (1 - s/L_i)\dot{x}_i + (s/L_i)\dot{x}_{i+1} \quad (3.64)$$

$$\dot{y}(s,t) = (1 - s/L_i)\dot{y}_i + (s/L_i)\dot{y}_{i+1}. \quad (3.65)$$

The kinetic energy of the i'th element is given by

$$T_i = \frac{1}{2}\int_0^{L_i} \rho_i(s)(\dot{x}^2 + \dot{y}^2)ds \quad (3.66)$$

where $\rho_i(s)$ is the mass density per unit length of the element. Substituting Eq. (3.64) and Eq. (3.65) into Eq. (3.66) and performing the integrations yields

$$T_i = \frac{1}{2}([\dot{x}_i\ \dot{x}_{i+1}]\mathbf{H}_i[\dot{x}_i\ \dot{x}_{i+1}]^T + [\dot{y}_i\ \dot{y}_{i+1}]\mathbf{H}_i[\dot{y}_i\ \dot{y}_{i+1}]^T) \quad (3.67)$$

where

$$\mathbf{H}_i = \begin{bmatrix} \int_0^{L_i} \rho_i(s)(1 - s/L_i)^2 ds & \int_0^{L_i} \rho_i(s)(s/L_i)(1 - s/L_i) ds \\ \int_0^{L_i} \rho_i(s)(s/L_i)(1 - s/L_i) ds & \int_0^{L_i} \rho_i(s)(s/L_i)^2 ds \end{bmatrix}. \quad (3.68)$$

It proves useful to define

$$m_{rs}^{i} = \frac{\partial^2 T_i}{\partial \dot{\theta}_r \partial \dot{\theta}_s}. \tag{3.69}$$

Substituting Eq. (3.67) into Eq. (3.69) yields

$$\begin{aligned} m_{rs}^{i} &= \frac{1}{2}\frac{\partial^2}{\partial \dot{\theta}_r \partial \dot{\theta}_s}\left([\dot{x}_i\ \dot{x}_{i+1}]\mathbf{H}_i[\dot{x}_i\ \dot{x}_{i+1}]^T + [\dot{y}_i\ \dot{y}_{i+1}]\mathbf{H}_i[\dot{y}_i\ \dot{y}_{i+1}]^T\right) \\ &= \frac{\partial}{\partial \dot{\theta}_r}\left(\begin{bmatrix}\frac{\partial \dot{x}_i}{\partial \dot{\theta}_s} & \frac{\partial \dot{x}_{i+1}}{\partial \dot{\theta}_s}\end{bmatrix}\mathbf{H}_i\,[\dot{x}_i\ \dot{x}_{i+1}]^T + \begin{bmatrix}\frac{\partial \dot{y}_i}{\partial \dot{\theta}_s} & \frac{\partial \dot{y}_{i+1}}{\partial \dot{\theta}_s}\end{bmatrix}\mathbf{H}_i\,[\dot{y}_i\ \dot{y}_{i+1}]^T\right) \\ &= \begin{bmatrix}\frac{\partial \dot{x}_i}{\partial \dot{\theta}_s} & \frac{\partial \dot{x}_{i+1}}{\partial \dot{\theta}_s}\end{bmatrix}\mathbf{H}_i\begin{bmatrix}\frac{\partial \dot{x}_i}{\partial \dot{\theta}_r} & \frac{\partial \dot{x}_{i+1}}{\partial \dot{\theta}_r}\end{bmatrix}^T + \begin{bmatrix}\frac{\partial \dot{y}_i}{\partial \dot{\theta}_s} & \frac{\partial \dot{y}_{i+1}}{\partial \dot{\theta}_s}\end{bmatrix}\mathbf{H}_i\begin{bmatrix}\frac{\partial \dot{y}_i}{\partial \dot{\theta}_r} & \frac{\partial \dot{y}_{i+1}}{\partial \dot{\theta}_r}\end{bmatrix}^T. \end{aligned} \tag{3.70}$$

The kinetic energy of the entire system is the sum of the kinetic energies of each element and any rotational inertias. Thus,

$$T = \sum_{i=1}^{N_e} T_i + \frac{1}{2}\sum_{j=1}^{N_a} J_j \dot{\theta}_j^2 \tag{3.71}$$

where J_j is the rotational inertia associated with θ_j. It is straightforward to show that

$$T = \frac{1}{2}\sum_{r=1}^{N_a}\sum_{s=1}^{N_a} m_{rs}\dot{\theta}_r\dot{\theta}_s. \tag{3.72}$$

Taking the partial derivative of both sides of Eq. (3.72) with respect to $\dot{\theta}_r$ and $\dot{\theta}_s$ gives

$$\begin{aligned} m_{rs} &= \frac{\partial^2 T}{\partial \dot{\theta}_r \partial \dot{\theta}_s} \\ &\overset{\text{Eq. (3.71)}}{=} \sum_{i=1}^{N_e}\frac{\partial^2 T_i}{\partial \dot{\theta}_r \partial \dot{\theta}_s} + \sum_{j=1}^{N_a} J_j \delta_{js}\delta_{jr} \\ &\overset{\text{Eq. (3.69)}}{=} \sum_{i=1}^{N_e} m_{rs}^{i} + \sum_{j=1}^{N_a} J_j \delta_{js}\delta_{jr} \end{aligned} \tag{3.73}$$

where

$$\delta_{js} = \begin{cases} 1 & \text{if } j = s \\ 0 & \text{if } j \neq s \end{cases}. \tag{3.74}$$

The general strain energy of the i^{th}-element is given by

$$U^i = \int_0^{L_i} (EI)_i \left(\frac{\partial \theta}{\partial s}\cdot\frac{\partial \theta}{\partial s}\right) ds \tag{3.75}$$

where $(EI)_i$ is the flexural rigidity of the i^{th}-element. The strain energy is simplified by a finite difference equation approximation, or

$$\frac{\partial \theta}{\partial s} = \frac{\theta_{j+1} - \theta_j}{L_i}. \tag{3.76}$$

Substitute Eq. (3.76) into Eq. (3.75) to obtain

$$U^i = \frac{1}{2}(EI)_i(\theta_{j+1} - \theta_j)^2/L_i. \tag{3.77}$$

The strain energy of the entire system is the sum of the strain energies of each element, thus

$$U = \frac{1}{2}\sum_{i=1}^{N_e}(EI)_i(\theta_{j+1} - \theta_j)^2/L_i. \tag{3.78}$$

Lagrange's equations are expressed as

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta}_k}\right) - \frac{\partial T}{\partial \theta_k} + \frac{\partial U}{\partial \theta_k} + \frac{\partial R}{\partial \dot{\theta}_k} = Q_k \tag{3.79}$$

where R is the Rayleigh dissipation function and Q_k is the k^{th} generalized force.

Starting with Eq. (3.72) obtain

$$\begin{aligned}
\frac{\partial T}{\partial \dot{\theta}_k} &= \sum_{r=1}^{N_a} m_{kr}\dot{\theta}_r \\
\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta}_k}\right) &= \sum_{r=1}^{N_a}\left(\dot{m}_{kr}\dot{\theta}_r + m_{kr}\ddot{\theta}_r\right) \\
&= \sum_{r=1}^{N_a}\sum_{s=1}^{N_a}\left(\frac{\partial m_{kr}}{\partial \theta_s}\dot{\theta}_s\right)\dot{\theta}_r + \sum_{r=1}^{N_a} m_{kr}\ddot{\theta}_r,
\end{aligned} \tag{3.80}$$

and

$$\frac{\partial T}{\partial \theta_k} = \frac{1}{2}\sum_{r=1}^{N_a}\sum_{s=1}^{N_a}\frac{\partial m_{rs}}{\partial \theta_k}\dot{\theta}_r\dot{\theta}_s. \tag{3.81}$$

Substituting Eq. (3.80) and Eq. (3.81) into Eq. (3.79) reveals

$$\sum_{r=1}^{N_a} m_{kr}\ddot{\theta}_r + \sum_{r=1}^{N_a}\sum_{s=1}^{N_a}\left(\frac{\partial m_{kr}}{\partial \theta_s} - \frac{1}{2}\frac{\partial m_{rs}}{\partial \theta_k}\right)\dot{\theta}_r\dot{\theta}_s + \frac{\partial U}{\partial \theta_k} + \frac{\partial R}{\partial \dot{\theta}_k} = Q_k. \tag{3.82}$$

One can show from Eq. (3.62), Eq. (3.63), Eq. (3.70), and Eq. (3.73) that the m_{rs} terms can be expressed as

$$m_{rs} = \bar{m}_{rs}\cos(\theta_r - \theta_s) \tag{3.83}$$

where each $\bar{m}_{rs}$ term is a constant (this point will become evident in the example problem). Substitution of Eq. (3.83) into Eq. (3.82) leads to the simplification

$$\sum_{r=1}^{N_a} \left(\bar{m}_{kr} \cos(\theta_k - \theta_r) \ddot{\theta}_r + \bar{m}_{kr} \dot{\theta}_r^2 \sin(\theta_k - \theta_r) \right) + \frac{\partial U}{\partial \theta_k} + \frac{\partial R}{\partial \dot{\theta}_k} = Q_k, \tag{3.84}$$

or equivalently in matrix notation

$$[\mathbf{M}(\boldsymbol{\theta})]\, \ddot{\boldsymbol{\theta}} + [\mathbf{C}(\boldsymbol{\theta})]\, \dot{\boldsymbol{\theta}}^2 = \mathbf{F}$$

where

$$\begin{aligned} M_{k,r}(\theta) &= \bar{m}_{kr} \cos(\theta_k - \theta_r) \\ C_{k,r}(\theta) &= \bar{m}_{kr} \sin(\theta_k - \theta_r) \\ F_k &= Q_k - \frac{\partial U}{\partial \theta_k} - \frac{\partial R}{\partial \dot{\theta}_k}. \end{aligned}$$

In this derivation, the strain energy is approximated by using Eq. (3.76). However, by retaining the general form of the strain energy expressed in Eq. (3.75), the following model is defined[10]:

The dynamic equations for a planar two-link flexible robot can be expressed as

$$[\mathbf{M}(\boldsymbol{\theta})]\, \ddot{\boldsymbol{\theta}} + [\mathbf{C}(\boldsymbol{\theta})]\, \dot{\boldsymbol{\theta}}^2 + [\mathbf{K}(\boldsymbol{\theta})]\, \mathbf{I}_1 = \boldsymbol{\tau}$$

where

$$\begin{aligned} M_{m,n}(\theta) &= M_{m,n} \cos(\theta_m - \theta_n) \\ C_{m,n}(\theta) &= M_{m,n} \sin(\theta_m - \theta_n) \\ M_{m,n} &= \int_0^L \rho(s) q_m(s) q_n(s) ds + \sum_{k=1}^{masses} M_k q_m(s_k) q_n(s_k) + \sum_{l=1}^{inertias} I_l p_m(s_l) p_n(s_l) \\ K_{m,n}(\theta) &= -K_{m,n} \sin(\theta_m - \theta_n) \\ K_{m,n} &= \int_0^L EI(s) p'_m(s) p'_n(s) ds \\ q_m(s) &= \int_0^s p_m(\hat{s}) d\hat{s} \\ \tau &= (-\tau_1, \tau_1, 0, 0, 0, 0, -\tau_2, \tau_2, 0, 0, 0, 0) \\ \mathbf{I}_1 &= \text{column of ones.} \end{aligned}$$

This model includes both point masses M_k and interias I_l. The shape functions $p_m(s)$ are nonzero over intervals that are small relative to the anticipated radii of

curvature. The shape functions are the traditional tent-shaped basis functions and assure compliance with the condition of nonextension[10]. An example of $p_m(s)$ is shown as part of the integrand of components of $\mathbf{H}_i$ in Eq. (3.68).

This is the dynamic model used in Chapter 5 for optimization studies and in Chapter 6 for control system studies.

3.4.1. Summary

The equations of motion for the system can be derived as follows:

1. Obtain expressions for $(\dot{x}_2, \ldots, \dot{x}_{N_e+1})$ and $(\dot{y}_2, \ldots, \dot{y}_{N_e+1})$ by recursively using Eq. (3.62) and Eq. (3.63) starting with $\dot{x}_1 = 0$ and $\dot{y}_1 = 0$.

2. Obtain expressions for the partial derivatives $\partial \dot{x}_i / \partial \dot{\theta}_r$ $(i = 2, \ldots, N_e + 1; r = 1, \ldots, N_a)$ by differentiating the results of Step 1.

3. Calculate the integrals appearing in Eq. (3.68) for all the elements.

4. Obtain expressions for the m^i_{rs} terms for all the elements by substituting the results of Steps 2 and 3 into Eq. (3.70).

5. Obtain expressions for the elements of the mass matrix m_{rs} by substituting the results of Step 4 into Eq. (3.73). Identify the $\bar{m}_{rs}$ terms by inspection.

6. Obtain an expression for the strain energy U by using Eq. (3.78).

7. Determine the generalized forces Q_k and specify the Rayleigh dissipation function R.

8. Assemble the equations of motion per Eq. (3.84).

Comments

The eight steps presented above account for the effects of centrifugal stiffening. Correctly modeling these effects is critical whenever joint rotation rates are of the same order-of-magnitude as the first natural frequency of any of the robot's links.

The equations of motion given by Eq. (3.84) are nonlinear as such they must be integrated numerically for simulation purposes. If an explicit numerical integration scheme is used (e.g., Runge-Kutta), care must be taken when modeling stiff members (e.g., the mounting brackets in the Sandia two-link manipulator). Stiff members are better modeled as rigid elements rather than as flexible elements with high stiffnesses. Otherwise the stable time step of the numerical integration scheme may become prohibitively small. Modeling a part as a rigid element is accomplished by appending to Eq. (3.84) a constraint equation for the rotations at the two ends of the element.

Example 3.11. Determine the mass components of the dynamic equations for a two-link flexible robot. Perform Steps 1-4 of this procedure. Model each link by using a single element for each link. Both links have torsional rigidities EI and uniform mass densities of ρ. Other parameters defining the problem are shown in Figure 3.15.

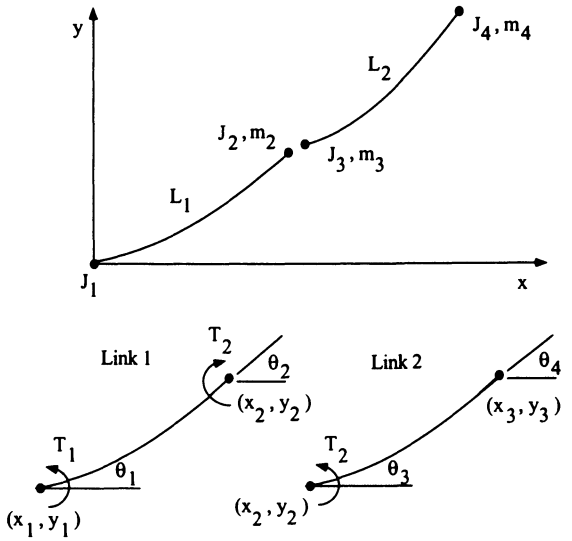

Figure 3.15. Two-link planar flexible robot element definitions

Solution

Step 1.

$$\dot{x}_2 = \frac{-L_1(\dot{\theta}_1 \sin\theta_1 + \dot{\theta}_2 \sin\theta_2)}{2}$$

$$\dot{y}_2 = \frac{L_1(\dot{\theta}_1 \cos\theta_1 + \dot{\theta}_2 \cos\theta_2)}{2}$$

$$\dot{x}_3 = \frac{-L_1(\dot{\theta}_1 \sin\theta_1 + \dot{\theta}_2 \sin\theta_2)}{2} - \frac{L_2(\dot{\theta}_3 \sin\theta_3 + \dot{\theta}_4 \sin\theta_4)}{2}$$

$$\dot{y}_3 = \frac{L_1(\dot{\theta}_1 \cos\theta_1 + \dot{\theta}_2 \cos\theta_2)}{2} + \frac{L_2(\dot{\theta}_3 \cos\theta_3 + \dot{\theta}_4 \cos\theta_4)}{2}.$$

Step 2.

$$\begin{array}{llll}
\dfrac{\partial \dot{x}_2}{\partial \dot{\theta}_1} = \dfrac{-L_1 \sin\theta_1}{2} & \dfrac{\partial \dot{x}_2}{\partial \dot{\theta}_2} = \dfrac{-L_1 \sin\theta_2}{2} & \dfrac{\partial \dot{x}_2}{\partial \dot{\theta}_3} = 0 & \dfrac{\partial \dot{x}_2}{\partial \dot{\theta}_4} = 0 \\
\dfrac{\partial \dot{y}_2}{\partial \dot{\theta}_1} = \dfrac{L_1 \cos\theta_1}{2} & \dfrac{\partial \dot{y}_2}{\partial \dot{\theta}_2} = \dfrac{L_1 \cos\theta_2}{2} & \dfrac{\partial \dot{y}_2}{\partial \dot{\theta}_3} = 0 & \dfrac{\partial \dot{y}_2}{\partial \dot{\theta}_4} = 0 \\
\dfrac{\partial \dot{x}_3}{\partial \dot{\theta}_1} = \dfrac{-L_1 \sin\theta_1}{2} & \dfrac{\partial \dot{x}_3}{\partial \dot{\theta}_2} = \dfrac{-L_1 \sin\theta_2}{2} & \dfrac{\partial \dot{x}_3}{\partial \dot{\theta}_3} = \dfrac{-L_2 \sin\theta_3}{2} & \dfrac{\partial \dot{x}_3}{\partial \dot{\theta}_4} = \dfrac{-L_2 \sin\theta_4}{2} \\
\dfrac{\partial \dot{y}_3}{\partial \dot{\theta}_1} = \dfrac{L_1 \cos\theta_1}{2} & \dfrac{\partial \dot{y}_3}{\partial \dot{\theta}_2} = \dfrac{L_1 \cos\theta_2}{2} & \dfrac{\partial \dot{y}_3}{\partial \dot{\theta}_3} = \dfrac{L_2 \cos\theta_3}{2} & \dfrac{\partial \dot{y}_3}{\partial \dot{\theta}_4} = \dfrac{L_2 \cos\theta_4}{2}.
\end{array}$$

Step 3. As an example, look at $(h_1)_{22}$

$$\begin{aligned}
(h_1)_{22} &\overset{\text{Eq. (3.68)}}{=} \int_0^{L_1} \rho(s/L_1)^2 ds \\
&= \rho s^3/(3L_1^2)|_0^{L_1} + m_2 \\
&= \rho L_1/3 + m_2.
\end{aligned}$$

Following a similar procedure for the other elements of $\mathbf{H}_1$ and $\mathbf{H}_2$ gives

$$\begin{aligned}
\mathbf{H}_1 &= \begin{bmatrix} \rho L_1/3 & \rho L_1/6 \\ \rho L_1/6 & \rho L_1/3 + m_2 \end{bmatrix} \\
\mathbf{H}_2 &= \begin{bmatrix} \rho L_2/3 + m_3 & \rho L_2/6 \\ \rho L_2/6 & \rho L_2/3 + m_4 \end{bmatrix}.
\end{aligned}$$

Step 4. As an example, look at m_{12}^2

$$\begin{aligned}
m_{12}^2 &\overset{\text{Eq. (3.70)}}{=} \begin{bmatrix} \dfrac{\partial \dot{x}_2}{\partial \dot{\theta}_2} & \dfrac{\partial \dot{x}_3}{\partial \dot{\theta}_2} \end{bmatrix} \mathbf{H}_2 \begin{bmatrix} \dfrac{\partial \dot{x}_2}{\partial \dot{\theta}_1} & \dfrac{\partial \dot{x}_3}{\partial \dot{\theta}_1} \end{bmatrix}^T + \begin{bmatrix} \dfrac{\partial \dot{y}_2}{\partial \dot{\theta}_2} & \dfrac{\partial \dot{y}_3}{\partial \dot{\theta}_2} \end{bmatrix} \mathbf{H}_i \begin{bmatrix} \dfrac{\partial \dot{y}_2}{\partial \dot{\theta}_1} & \dfrac{\partial \dot{y}_3}{\partial \dot{\theta}_1} \end{bmatrix}^T \\
&= L_1^2/4 \begin{bmatrix} \sin\theta_2 & \sin\theta_2 \end{bmatrix} \begin{bmatrix} \rho L_2/3 + m_3 & \rho L_2/6 \\ \rho L_2/6 & \rho L_2/3 + m_4 \end{bmatrix} \begin{bmatrix} \sin\theta_1 \\ \sin\theta_1 \end{bmatrix} + \\
&\quad L_1^2/4 \begin{bmatrix} \cos\theta_2 & \cos\theta_2 \end{bmatrix} \begin{bmatrix} \rho L_2/3 + m_3 & \rho L_2/6 \\ \rho L_2/6 & \rho L_2/3 + m_4 \end{bmatrix} \begin{bmatrix} \cos\theta_1 \\ \cos\theta_1 \end{bmatrix} \\
&= [L_1^2(\rho L_2 + m_3 + m_4)/4] \cos(\theta_1 - \theta_2).
\end{aligned}$$

All the values of m_{rs}^i can be calculated in a similar manner for $s \geq r$. Note, by

definition $m^i_{rs} = m^i_{sr}$. Only nonzero values are reported for

$$
\begin{aligned}
m^1_{11} &= \left[L_1^2(\frac{\rho L_1}{12} + \frac{m_2}{4})\right] \\
m^1_{12} &= \left[L_1^2(\frac{\rho L_1}{12} + \frac{m_2}{4})\right] \cos(\theta_1 - \theta_2) \\
m^1_{22} &= \left[L_1^2(\frac{\rho L_1}{12} + \frac{m_2}{4})\right] \\
m^2_{11} &= \left[L_1^2(\frac{\rho L_2 + m_3 + m_4)}{4}\right] \\
m^2_{12} &= \left[L_1^2(\frac{\rho L_2 + m_3 + m_4)}{4}\right] \cos(\theta_1 - \theta_2) \\
m^2_{13} &= \left[L_1 L_2(\frac{\rho L_2}{8} + \frac{m_4}{4})\right] \cos(\theta_1 - \theta_3) \\
m^2_{14} &= \left[L_1 L_2(\frac{\rho L_2}{8} + \frac{m_4}{4})\right] \cos(\theta_1 - \theta_4) \\
m^2_{22} &= \left[L_1^2(\frac{\rho L_2 + m_3 + m_4)}{4}\right] \\
m^2_{23} &= \left[L_1 L_2(\frac{\rho L_2}{8} + \frac{m_4}{4})\right] \cos(\theta_2 - \theta_3) \\
m^2_{24} &= \left[L_1 L_2(\frac{\rho L_2}{8} + \frac{m_4}{4})\right] \cos(\theta_2 - \theta_4) \\
m^2_{33} &= \left[L_2^2(\frac{\rho L_2}{12} + \frac{m_4}{4})\right] \\
m^2_{34} &= \left[L_2^2(\frac{\rho L_2}{12} + \frac{m_4}{4})\right] \cos(\theta_3 - \theta_4) \\
m^2_{44} &= \left[L_2^2(\frac{\rho L_2}{12} + \frac{m_4}{4})\right].
\end{aligned}
$$

The elements of the mass matrix are obtained by substituting the results of the previous step into Eq. (3.73). It is clear from the results of Step 4 that one can express the elements of the mass matrix in the simple form $m_{rs} = \bar{m}_{rs} \cos(\theta_r - \theta_s)$

where $\bar{m}_{rs}$ are constants (see Eq. 3.83). The values of $\bar{m}_{rs}$ are given below for $s \geq r$.

$$\bar{m}_{11} = J_1 + L_1^2\left(\frac{\rho L_1}{12} + \frac{\rho L_2}{4} + \frac{(m_2+m_3+m_4)}{4}\right)$$

$$\bar{m}_{12} = L_1^2\left(\frac{\rho L_1}{12} + \frac{\rho L_2}{4} + \frac{(m_2+m_3+m_4)}{4}\right)$$

$$\bar{m}_{13} = L_1 L_2\left(\frac{\rho L_2}{8} + \frac{m_4}{4}\right)$$

$$\bar{m}_{14} = L_1 L_2\left(\frac{\rho L_2}{8} + \frac{m_4}{4}\right)$$

$$\bar{m}_{22} = J_2 + L_1^2\left(\frac{\rho L_1}{12} + \frac{\rho L_2}{4} + \frac{(m_2+m_3+m_4)}{4}\right)$$

$$\bar{m}_{23} = L_1 L_2\left(\frac{\rho L_2}{8} + \frac{m_4}{4}\right)$$

$$\bar{m}_{24} = L_1 L_2\left(\frac{\rho L_2}{8} + \frac{m_4}{4}\right)$$

$$\bar{m}_{33} = J_3 + L_2^2\left(\frac{\rho L_2}{12} + \frac{m_4}{4}\right)$$

$$\bar{m}_{34} = L_2^2\left(\frac{\rho L_2}{12} + \frac{m_4}{4}\right)$$

$$\bar{m}_{44} = J_4 + L_2^2\left(\frac{\rho L_2}{12} + \frac{m_4}{4}\right).$$

Note, $\bar{m}_{rs} = \bar{m}_{sr}$. The remaining Steps 5-8 are completed in Homework 3.7.

3.5. Actuator Dynamics

The details of actuator dynamics were not included in the previous equation of motion developments. The inputs were torques. Unfortunately, there is no ideal method for generating a desired torque. That is, a particular actuator must be chosen (e.g., D.C. motor, hydraulics, pneumatics, etc.) for developing the torques. No matter which actuator is chosen it will certainly have some dynamics to be considered.

One can consider creating the torques by using a D.C. motor. Assume the motor has armature inertia J_a, armature damping B_a, armature resistance R_a, a torque constant K_t, and a gear-head with gear ratio N_g. The relationship between the motor's armature speed θ_a, the armature voltage V_a, and the output torque τ is

$$\tau = \frac{N_g K_t}{R_a} V_a - \left(\frac{{K_t}^2}{R_a} + B_a\right) {N_g}^2 \dot{\theta}_a - J_a {N_g}^2 \ddot{\theta}_a. \tag{3.85}$$

Replacing the τ in the rotational equations of motion with Eq. (3.85) changes the

effective mass of the system and adds damping. The consequences of these changes are further discussed in the rest of the book with respect to system identification, input shaping, and control system design.

3.6. Chapter 3 Summary

In this chapter the foundation was developed for subsequent flexible robot control system design and analysis. Fundamental concepts for modeling flexible links were presented that included both linear independence and orthogonality of functions. Several methods were reviewed for obtaining approximate mode shapes for beams and beam-like structures. The method of quadratic modes was developed. Several examples reinforced the need to include higher-order kinematics in the kinetic energy term in order to correctly and systematically capture the effects of centrifugal stiffening. A short explanation was given for the extension of quadratic modes to multibody applications. A second method presented a unique finite element based modeling technique for planar flexible robot dynamics. This method also captures the higher-order kinematic terms. Both methods develop dynamic models that were incorporated into simulation and experimental validation of input shaping techniques (see Chapter 5) and control system designs (Chapters 6, 7, and 8). This chapter concluded with a brief introduction of actuator dynamic effects.

3.7. Chapter 3 References

1. L. Meirovitch, *Elements of Vibration Analysis*, McGraw-Hill, Inc., New York, 1975.
2. R. D. Blevins, *Formulas for Natural Frequency and Mode Shape*, Krieger Pub. Co., Florida, 1993.
3. I. H. Shames, and F. A. Cozzarelli, *Elastic and Inelastic Stress Analysis*, Prentice Hall, New Jersey, 1992
4. L. Meirovitch, *Analytical Methods in Vibrations*, The Macmillan Company, London, 1967.
5. T. Zeiler and C. Buttrill, "Dynamic analysis of an unrestrained rotating structure through nonlinear simulation," in Proceedings of the AIAA/ASME/ASCE/AHS 27th Structures, Structural Dynamics and Materials Conference, (Washington D.C.), pp. 167-177, April 1988.
6. K. Banerjee and J. Dickens, "Dynamics of an arbitrary flexible body in large rotation and translation," Journal of Guidance, Control, and Dynamics, Vol. 13, no. 2, pp. 221-227, 1990.
7. D .J. Segalman and C. R. Dohrmann, "A method for calculating the dynamics of rotating flexible structures, Part 1: Derivation," Journal of Vibration and Acoustics, Vol. 118, 1996, pp. 313-317.
8. D. J. Segalman, C. R. Dohrmann and A. M. Slavin, " A method for calculating the dynamics of rotating flexible structures, Part 2: Example calculations," Journal of Vibration and Acoustics, Vol. 118, 1996, pp. 318-322.

9. G. G. Parker, *Control Techniques for Multibody Flexible Structures Modelled by a Method of Quadratic Modes*, Ph.D. Dissertation: State University of New York, Buffalo, April, 1994
10. G. R. Eisler, R. D. Robinett, D. J. Segalman, and J. T. Feddema, "Approximate Optimal Trajectories for flexible-link manipulator slewing using recursive quadratic programming," Journal of Dynamic Systems, Measurement, and Control, Vol. 115, 1993, pp. 405-410.
11. G. G. Parker, D. J. Segalman, R. D Robinett, and D. J. Inman, "Decentralized sliding mode control for flexible link robots," Journal of Intelligent and Robotic Systems, 17: pp. 61-79,1996.

3.8. Chapter 3 Problems

Homework 3.1. From Section 3.2.
Determine orthogonal modes $\psi_1(\chi)$ and $\psi_2(\chi)$ by using the Gram-Schmidt orthogonalization process. The first mode shape determined by applying a distributed load to a cantilevered beam of 1 N/m and computing the static deflection is

$$\phi_1(\chi) = \frac{1}{48}\chi^2 - \frac{1}{720}\chi^3 + \frac{1}{28800}\chi^4$$

The second mode shape is developed by approximating the exact cantilever beam mode shape by a seventhth-order polynomial, or

$$\phi_2(\chi) = \left(92465\chi^2 + 4.8\chi^3 - 56.5\chi^4 + 8.181\chi^5 - 0.45\chi^6 + 0.01\chi^7\right)/10000.$$

Homework 3.2. From Section 3.3.
Derive the equation of motion for a beam cantilevered to a rotating hub with a point mass m at its tip. *Hint*: This problem won't be too difficult if you start with Eq. (3.38). Perform a simulation for the maneuver considered in the example and plot the beam tip deflection versus time. Use the same parameter values given in Example 3.8 and set $m = 2$.

Homework 3.3. From Section 3.3.
Given the following quadratic definitions:

$$\begin{aligned}
\mathbf{x} &= \left(\chi + q_i^\alpha q_j^\alpha g_{ij}^\alpha + q_i^\beta q_j^\beta g_{ij}^\beta\right)\hat{\mathbf{i}} + q_i^\alpha \phi_i^\alpha \hat{\mathbf{j}} + q_i^\beta \phi_i^\beta \hat{\mathbf{k}} \\
\ddot{\mathbf{x}} &= 2\left(\ddot{q}_i^\alpha q_j^\alpha g_{ij}^\alpha + \dot{q}_i^\alpha \dot{q}_j^\alpha g_{ij}^\alpha + \ddot{q}_i^\beta q_j^\beta g_{ij}^\beta + \dot{q}_i^\beta \dot{q}_j^\beta g_{ij}^\beta\right)\hat{\mathbf{i}} + \ddot{q}_i^\alpha \phi_i^\alpha \hat{\mathbf{j}} + \ddot{q}_i^\beta \phi_i^\beta \hat{\mathbf{k}} \\
\boldsymbol{\phi}_i &= \phi_i^\alpha \hat{\mathbf{j}} + \phi_i^\beta \hat{\mathbf{k}} \\
\mathbf{F}_s &= (EI)^\alpha q_i^\alpha (\phi_i^\alpha)^{iv} \hat{\mathbf{j}} + (EI)^\beta q_i^\beta (\phi_i^\beta)^{iv} \hat{\mathbf{k}} \\
\mathbf{g}_{ij} &= (g_{ij}^\alpha + g_{ij}^\beta)\hat{\mathbf{i}},
\end{aligned}$$

determine the dynamic equations for both

a. an unidirectional, and

b. a bidirectional nonrotating beam.

For the unidirectional case, set all β terms to zero. *Hint*: Do not include terms higher than second-order in your final solutions.

Homework 3.4. From Section 3.3.

1. Given the location of a particle as $\mathbf{x}(\chi, t) = \mathbf{R}^T(t)[\boldsymbol{\chi} + q_i(t)\boldsymbol{\phi}_i(\boldsymbol{\chi})]$, find the expression for $\ddot{\mathbf{x}}(\boldsymbol{\chi}, t)$.
2. Previously, the time derivative of the rotation matrix is written as $\dot{\mathbf{R}}(t) = \boldsymbol{\Omega}^T\mathbf{R}$. Prove that if the total rotation matrix is the product of two matrices $\mathbf{R} = \mathbf{R}_1\mathbf{R}_2$, then the total *spin* matrix is the SUM of the two spin matrices, i.e., $\boldsymbol{\Omega} = \boldsymbol{\Omega}_1 + \boldsymbol{\Omega}_2$.

Homework 3.5. From Section 3.3.
Consider the thin extensible beam of length 10 meters, mass per unit length m, modulus of elasticity E, and area moment of inertia I that is free to bend in only the y-direction. It is attached to a joint undergoing a prescribed rotation θ about the z axis (see Figure 3.8). Generate the dynamic equations of motion for the system by using the method of quadratic modes. *Hint*: Use the mode shapes determined from first homework.

Homework 3.6. From Section 3.3.
By using the method of quadratic modes, derive the equations of motion for a planar flexible two-link robotic arm[11]. Consider the hubs as point masses.

Homework 3.7. From Section 3.4.
Given a Rayleigh dissipation function of the form

$$R = (c_1\dot{\theta}_1^2 + c_2(\dot{\theta}_3 - \dot{\theta}_2)^2)/2,$$

and joint torques

$$\begin{aligned} T_1 &= k_1(\theta_{1d} - \theta_1) - k_2\dot{\theta}_1 \\ T_2 &= k_3(\theta_{3d} - \theta_3) - k_4(\dot{\theta}_3 - \dot{\theta}_2), \end{aligned}$$

derive the equations of motion for the robot model by carrying out Steps 5 through 8 of the procedure given in Section 3.4.1.. You may express your results in terms of the $\bar{m}_{rs}$ terms for simplicity. Explain how you would go about simulating the motion of the robot given specific values for the parameters defining the problem.

Chapter 4

System Identification

4.1. Introduction

System Identification (System ID) plays a key role in control system design and input shaping[1,2]. The first thing that a controls engineer learns in the real world is that the transfer function is not written on the outside of the hardware container. So, how does one obtain the transfer function? System ID is used to obtain the transfer function and the critical parameters of simplified systems models that are required for input shaping designs. System models are usually an approximation and need to be refined by comparing to experimental data. On the other hand, empirical models can be developed directly from experimental data when no reasonable theoretical models exist for a system. In any case, System ID provides a systematic way to develop and/or refine the system model. This chapter describes the basic concepts of System ID including linear and nonlinear least squares, as well as, the more advanced concept of homotopy to increase the robustness of the System ID tools. The last section demonstrates the backward propagation technique for multiple-link robots and actuator System ID.

4.2. Linear Least Squares (LSS)

The first step in System ID is to develop a system model. The system model can be developed from first principles and/or empirically from experimental data. The goal of this chapter is to refine the models developed in Chapters 2 and 3 by determining the critical parameters that need to be identified in order to develop input shaping algorithms and feedback controllers.

Before complicating the situation with dynamical models, it is illustrative to begin the discussion of Linear Least Squares (LLS) with some simple general approximations of a given function. Some common approximation functions include polynomials, orthogonal functions, and Fourier series that are combined as a linear

R. D. Robinett III et al., *Flexible Robot Dynamics and Controls*

sum. The goal is to approximate a given nonlinear function evaluated at M times

$$\mathbf{y} = \mathbf{y}(t) = (y(t_1), y(t_2), \ldots, y(t_M))^T,$$

with

$$\mathbf{y}(t) = \sum_{i=1}^{N} p_i t^i = p_0 + p_1 t + p_2 t^2 + \cdots + p_N t^N,$$

or

$$\mathbf{y} = \mathbf{A}\mathbf{p} = \begin{bmatrix} 1 & t_1 & \ldots & t_1^N \\ 1 & t_2 & \ldots & t_2^N \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_M & \ldots & t_M^N \end{bmatrix} \begin{Bmatrix} p_0 \\ p_1 \\ \vdots \\ p_N \end{Bmatrix}$$

where $\mathbf{p}$ is a set of linear gains and $\mathbf{A}$ is an $M \times N$ matrix. If $M < N$, then a minimum norm problem results, whereas if $M = N$, then a unique solution exists if the matrix is full rank[3].

A least squares solution occurs when $M > N$ and can be derived from the following optimization problem:

$$\begin{aligned} J(\mathbf{p}) &= [\mathbf{y} - \mathbf{A}\mathbf{p}]^T \mathbf{W} [\mathbf{y} - \mathbf{A}\mathbf{p}] \\ \frac{\partial J}{\partial \mathbf{p}} &= -2\mathbf{A}^T\mathbf{W}\mathbf{y} + 2\mathbf{A}^T\mathbf{W}\mathbf{A}\mathbf{p} = \mathbf{0} \\ \mathbf{p} &= (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\mathbf{y}. \end{aligned} \tag{4.1}$$

Equation (4.1) is referred to as the normal equations[4]. Note, $\mathbf{W}$ is a positive definite weight matrix that must be picked by the analyst. It is customary to start with the identity matrix, but the measurement covariance matrix is typically used with experimental data[4]. Also, $\mathbf{A}^T\mathbf{W}\mathbf{A}$ will be diagonal if $\mathbf{W}$ is diagonal and one uses orthogonal functions. An example problem would be helpful at this point.

Example 4.1. Approximate a unit step with a quadratic function.

$$\begin{aligned} y(t) &= \text{unit step} = 0 \quad \text{for} \quad t < 0 \quad \text{and} \\ &= 1 \qquad\qquad\quad \text{for} \quad t \geq 0 \end{aligned}$$

and approximated for $-1 \leq t \leq 1$ with a second-order polynomial

$$\mathbf{y}(t) = p_0 + p_1 t + p_2 t^2.$$

Solution

With

$$\begin{aligned} y(-1) &= 0 \\ y(-\frac{1}{2}) &= 0 \\ y(0) &= 1 \\ y(1) &= 1. \end{aligned}$$

Then,

$$\begin{aligned} \mathbf{y} &= (0,0,1,1)^T \\ \mathbf{W} &= \mathbf{I} \\ \mathbf{A} &= \begin{bmatrix} 1 & -1 & 1 \\ 1 & -\frac{1}{2} & \frac{1}{4} \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}. \end{aligned}$$

Therefore,

$$\mathbf{A}^T\mathbf{W}\mathbf{A} = \begin{bmatrix} 4 & -\frac{1}{2} & 2\frac{1}{4} \\ -\frac{1}{2} & 2\frac{1}{4} & -\frac{1}{8} \\ 2\frac{1}{4} & -\frac{1}{8} & 2\frac{1}{16} \end{bmatrix},$$

and

$$(\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1} = \frac{1}{6.875}\begin{bmatrix} 4.625 & 0.75 & -5 \\ 0.75 & 3.1875 & -0.625 \\ -5 & -0.625 & 8.75 \end{bmatrix}.$$

Therefore,

$$\begin{aligned} \mathbf{p} &= (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\mathbf{y} \\ &= \begin{Bmatrix} 0.727 \\ 0.591 \\ -0.273 \end{Bmatrix}, \end{aligned}$$

and finally,

$$\begin{aligned} y(-1) &= -0.137 \\ y(-\frac{1}{2}) &= 0.363 \\ y(0) &= 0.727 \\ y(1) &= 1.045. \end{aligned}$$

The approximation can be made better by picking more points to match and/or pick more appropriate approximation functions, such as a Fourier series.

It is common to have numerical difficulties when performing a LLS System ID. In particular, the pseudoinverse can be ill-conditioned[4], so one should use an LU decomposition or a singular value decomposition. (These algorithms are available in Numerical Recipes[5]).

4.3. Nonlinear Least Squares

The LLS algorithm development in the previous section forms the basis of a powerful System ID tool, called nonlinear least squares (NLS). The NLS algorithm extends the LLS by linearizing the nonlinear system to be identified about an appropriate operating point and applying the LLS algorithm iteratively. This procedure is a generalization of Newton's root solving method[6] (see Section 2.5.1.) that is typically used to determine the roots of a characteristic equation, such as the root locus of a linear controller[7]. The NLS algorithm is better known as *Gauss's least square differential correction* algorithm and is defined as follows: Assume the analyst has developed a model of the system of interest $\mathbf{y} = \mathbf{F}(\mathbf{p})$ and has an idea which system parameters $\mathbf{p}$ are critical.

Assume there are M measurements of the system

$$\mathbf{y} = (y_1, y_2, \ldots, y_M)^T.$$

Assume the measurements are approximated by a set of nonlinear functions of a parameter vector $\mathbf{p}$

$$\mathbf{y} = \mathbf{F}(\mathbf{p}).$$

Optimize

$$J(\mathbf{p}) = [\mathbf{y} - \mathbf{F}(\mathbf{p})]^T \mathbf{W} [\mathbf{y} - \mathbf{F}(\mathbf{p})]$$

subject to the linearized propagation

$$\mathbf{p} = \mathbf{p}_{Current} + \Delta\mathbf{p} = \mathbf{p}_C + \Delta\mathbf{p},$$

and the prediction

$$\mathbf{F}(\mathbf{p}) \cong \mathbf{F}(\mathbf{p}_C) + \mathbf{A}\Delta\mathbf{p}$$

where

$$\mathbf{A} = \left[\left. \frac{\partial \mathbf{F}}{\partial \mathbf{p}} \right|_C \right].$$

Therefore,

$$J(\mathbf{p}) = [\Delta\mathbf{y} - \mathbf{A}\Delta\mathbf{p}]^T\mathbf{W}[\Delta\mathbf{y} - \mathbf{A}\Delta\mathbf{p}]$$

where

$$\Delta\mathbf{y} = \mathbf{y} - \mathbf{F}(\mathbf{p}_C).$$

Therefore,

$$\Delta\mathbf{p} = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\Delta\mathbf{y}. \tag{4.2}$$

This algorithm can be converted to an eight step procedure that can be directly implemented on a computer.

4.3.1. Gauss's Least Square Differential Correction Algorithm

1. Input measurements and weights

$$\mathbf{y} = (y_1, y_2, \ldots, y_M)^T$$
$$\mathbf{W} = [w_{ij}].$$

2. Input parameter starting guesses

$$\mathbf{p}_C = (p_{1_C}, p_{2_C}, \ldots, p_{N_C})^T.$$

3. Compute current values of approximation functions

$$\mathbf{y}_C = \mathbf{F}(\mathbf{p}_C).$$

4. Compute matrix of first partial derivatives

$$\mathbf{A} = \left[\left.\frac{\partial\mathbf{F}}{\partial\mathbf{p}}\right|_C\right].$$

5. Form measured minus computed values

$$\Delta\mathbf{y} = \mathbf{y} - \mathbf{F}(\mathbf{p}_C)$$

and their current weighted sum square

$$J = [\Delta\mathbf{y}]^T\mathbf{W}[\Delta\mathbf{y}],$$

then go to Step (8) upon convergence.

6. Determine the correction vector $\Delta\mathbf{p}$ that minimizes the predicted residual sum squares

$$\Delta\mathbf{p} = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\Delta\mathbf{y}.$$

7. Apply the corrections for Step (6), by replacing previous current parameter estimates according to

$$\mathbf{p}_C = \mathbf{p}_C + \Delta\mathbf{p}$$

and return to Step (3).

8. Set $\mathbf{p} = \mathbf{p}_C$ on the final (if converged) iteration.

Note 4.1. Steps (3) through (7) are iterated until the process converges (i.e., J changes negligibly on successive iterations and/or the correction vector $\Delta\mathbf{p}$ has negligibly small elements).

Numerical convergence difficulties will often occur and usually can be traced to two problems. First, the initial guesses of the parameters are too far away from the actual parameter values. This problem can be solved by the homotopy techniques discussed later in this chapter, or by determining better initial guesses from reviewing the model. Second, the pseudoinverse matrix is ill-conditioned, but can be solved by using the singular value decomposition mentioned in Section 4.2..

4.3.2. Overhead Gantry Robot Example

Now, that the basic mathematical development is complete it is time to solve the overhead gantry robot example. As will be discussed in the next chapter, there are many ways to produce a swing-free move of a simply suspended object (see Figure 4.1). One of the most general methods is described by Jones and Petterson[8] where a double pulse train is employed. The basic ingredient necessary for every approach to developing swing-free trajectories is knowledge of the frequency of oscillation for the suspended object (i.e., a critical parameter). Consequently, the period of oscillation must be measured each time a different object is moved in order to produce a swing-free move. In this section, a batch nonlinear least square estimator enables the computer to automatically calculate (i.e., adaptively control) the period of oscillation of the suspended object from force sensor measurements (see Figure 4.2) and produce swing-free trajectories.

The nonlinear least square estimator is based on the Gauss's least square differential correction (GLSDC) algorithm described in Section 4.3.1.. The GLSDC algorithm is chosen because it is easy to implement and highly flexible[9,10]. The estimator formulation is applied to a force sensor located on the arm of the gantry robot (see Figure 4.2) that provides the measurement data to estimate the mass,

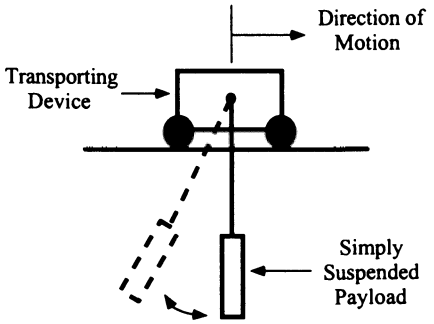

Figure 4.1. Diagram of overhead gantry robot

frequency, and two initial conditions of a planar pendulum. The actual measurements are M vertical force (F_x in Figure 4.2) measurements. The measurement model for these M data points is given by

$$T = mg\cos\theta + ml\dot{\theta}^2$$
$$F_x = T\cos\theta = mg\cos^2\theta + ml\dot{\theta}^2\cos\theta$$
$$\theta = \theta_0\cos\omega t + \frac{\dot{\theta}_0}{\omega}\sin\omega t$$

where

$$\begin{aligned} m &= \text{mass of pendulum} \\ l &= \text{length of pendulum} \\ \theta_0, \dot{\theta}_0 &= \text{initial conditions of pendulum} \\ g &= \text{gravity} \\ T &= \text{cable tension} \\ \omega &= \text{frequency} = \sqrt{\frac{g}{l}}. \end{aligned}$$

Thus, the nonlinear measurement model is

$$y_i = F_x(t_i) = mg\left[\cos^2\theta_i + \frac{\dot{\theta}_i^2}{\omega^2}\cos\theta_i\right] \tag{4.3}$$

where

$$\theta_i = \theta_0\cos\omega t_i + \frac{\dot{\theta}_0}{\omega}\sin\omega t_i,$$

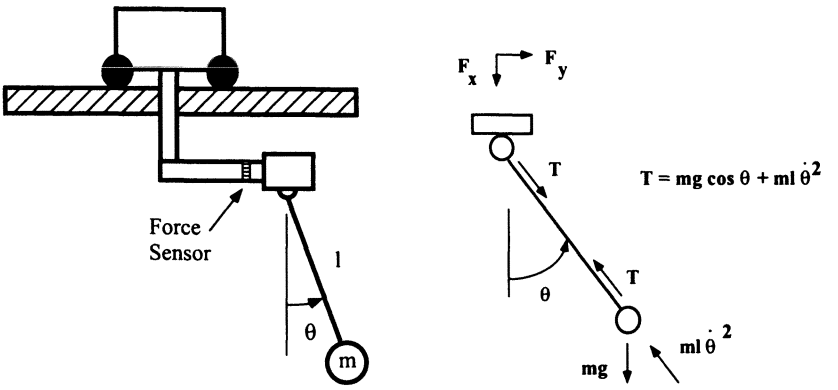

Figure 4.2. Gantry robot with force sensor

and

$$\mathbf{p} = (\theta_0, \dot{\theta}_0, m, \omega)^T.$$

Using only one channel of force data F_x simplifies the development of the estimator algorithm. Although forces F_y and F_z are also available from the force sensor, this algorithm may be applied on equipment where the force from only one direction is available.

The weight matrix $\mathbf{W}$ in Eq. (4.2) is a diagonal, positive definite, constant matrix and the Jacobian matrix is analytically developed from Eq. (4.3) by taking the first partial derivatives with respect to $\theta_0, \dot{\theta}_0, m$, and ω.

$$A = \left[\frac{\partial y_i}{\partial \theta_0} \frac{\partial y_i}{\partial \dot{\theta}_0} \frac{\partial y_i}{\partial m} \frac{\partial y_i}{\partial \omega}\right] \tag{4.4}$$

$$\begin{aligned}\frac{\partial y_i}{\partial \theta_0} &= \frac{\partial y_i}{\partial \theta_i}\frac{\partial \theta_i}{\partial \theta_0} + \frac{\partial y_i}{\partial \dot{\theta}_i}\frac{\partial \dot{\theta}_i}{\partial \theta_0} \\ &= m\Big[-g \sin 2\theta_i \cos \omega t_i + l \cos \theta_i \left(2\omega^2 \theta_0 \sin^2 \omega t_i - \omega \dot{\theta} \sin 2\omega t_i\right) \\ &\quad - l\dot{\theta}_i^{\,2} \cos \omega t_i \sin \theta_i\Big]\end{aligned} \tag{4.5}$$

(4.6)

$$\begin{aligned}\frac{\partial y_i}{\partial \theta_0} &= \frac{\partial y_i}{\partial \theta_i}\frac{\partial \theta_i}{\partial \theta_0} + \frac{\partial y_i}{\partial \dot{\theta}_i}\frac{\partial \dot{\theta}_i}{\partial \theta_0} \\ &= m\Big(-\frac{g}{\omega}\sin 2\theta_i \sin \omega t_i + l\cos\theta_i\Big[-\omega\theta_0 \sin 2\omega t_i + 2\dot{\theta}_0 \cos^2 \omega t_i\Big] \\ &\quad - \frac{l}{\omega}\dot{\theta}_i^2 \sin\theta_i \sin \omega t_i\Big)\end{aligned} \tag{4.7}$$

$$\frac{\partial y_i}{\partial m} = \frac{\partial y_i}{\partial \theta_i}\frac{\partial \theta_i}{\partial m} + \frac{\partial y_i}{\partial \dot{\theta}_i}\frac{\partial \dot{\theta}_i}{\partial m} = g\cos^2\theta_i + l\left(\dot{\theta}_i^2 \cos\theta_i\right) \tag{4.8}$$

Equations Eq. (4.3) and Eq. (4.4), the weight matrix, and the force sensor measurements are the required elements of the GLSDC algorithm needed to generate an estimate of the frequency (period of oscillation) for the swing-free trajectory planner.

The adaptive, swing-free trajectory planner was implemented at Sandia National Laboratories on a CIMCORP XR6100 robot. A simply suspended object is held by the robot gripper and a force-moment sensor (manufactured by JR3 Corporation) capable of transducing the forces in the three principal axes and the corresponding moments is used to monitor the oscillations of the payload. Four different suspended objects are used to test the adaptive, swing-free planner: a 22.7 kg (50 pound) weight suspended from a 1.93 m (76 inch) cable, a 45.4 kg (100 pound) weight suspended first from a 1.83 m (72 inch) cable and then from a 0.74 m (29 inch) cable, and a 1/5 scale (0.99 m (39 inches) tall) 11.4 kg (25 pound) model of a fuel rod assembly on a pivot joint (see Figure 4.3). The readings from the force sensor are stored on a Digital Equipment Corporation LSI11/73 executing the RT-11 real time operating system, which is also used to implement swing-free moves of the suspended objects. Finally, the batch estimation program was run on a SUN-3 computer that was networked with the LSI-11.

While the pendulum is swinging, the first step in executing the adaptive, swing-free trajectory planner is to gather force data. The suspended object is picked up, displaced, and allowed to swing freely. A set of 60 force readings (sufficient to assure a full period of oscillation for any of our payloads) is taken as the payload swings with each reading at an interval of 47 milliseconds, which is the update rate of the robot controller. Synchronizing readings with the controller updates ensures a sufficient length of time between readings. Once a set of force data has been recorded, a force servo damping program that uses a technique described by Jones, Petterson, and Werner[11] is used to damp-out the swinging motion of the suspended object.

The batch estimator runs while the robot damps the payload's oscillation. The constants necessary for the computations are a convergence criteria, the weighting matrix $\mathbf{W}$, gravity, a maximum number of iterations, the number of force readings (M), and the sampling period. The values needed to initialize the program variables are the initial guesses of the calculated parameters and a set of force readings $\tilde{\mathbf{y}}$. The calculated parameters $\mathbf{p}$ are the initial angle and initial angular velocity of

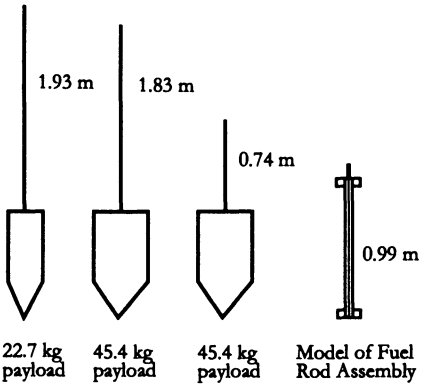

Figure 4.3. Suspended objects used in experiments

the suspended object, its mass, and its frequency. In each iteration, the batch estimator calculates the matrices $\mathbf{A}^T\mathbf{W}\mathbf{A}$ and $\mathbf{A}^T\mathbf{W}\Delta\mathbf{y}$ of Eq. (4.2) (using Eqs. 4.4) to calculate the Jacobian matrix $\mathbf{A}$, solves this linear system of equations by using LU decomposition[3] for the change in the desired parameters $\mathbf{p}$, and adjusts the values of the desired parameters by adding in this change. The iterations conclude when there is negligible change in the values of the desired parameters (i.e., $\Delta\mathbf{p} = \mathbf{0}$), or when the performance function gets smaller than the convergence criteria. The object's frequency is then converted into its period and passed into the swing-free trajectory program on the LSI-11.

The trajectory is computed by using the acceleration profile tailoring described by Jones and Petterson[8] and is sent to the robot controller as a series of displacement commands. The communication between the robot controller and the LSI-11 control computer is synchronized by the robot controller through the Realtime Path Modification (RPM) board. The RPM board sends a synchronizing pulse to the control computer every 47 milliseconds. When the swing-free trajectory program senses the synchronization pulse, it transmits a position update over a serial communication line to the RPM board that commands the robot to move. This process continues until the desired trajectory is complete.

Tests of the adaptive, swing-free trajectory planner were done by using the four different suspended objects. Each object's period was measured by timing its oscillations and also calculated by using the estimator. A comparison of these two periods and the resulting residual oscillations after a swing-free move can be seen in Table 4.1. As shown in that table, the calculated periods are within two percent of the measured values. Although the resulting residual oscillations for the calculated periods are about double the value of those with the measured periods, both are

over 95% smaller than the oscillation that occurs with an uncontrolled move (last entry in Table 4.1).

Table 4.1. Test results

Test Object	Length	Measured Period	Average Calculated Period	Residual Oscillation w/measured Period	Residual Oscillation w/calculated Period
	(m)	(sec)	(sec)	(degrees)	(sec)
22.7kg payload	1.93	2.79	2.80	0.05	0.09
45.4kg payload	1.83	2.71	2.77	0.05	0.10
45.4kg payload	0.74	1.75	1.76	0.15	0.31
11.4kg Fuel rod	0.99	1.64	1.66	0.14	0.27
45.4kg payload,no damp.	1.93	-	-	6.8	-

Using the batch estimator to determine the necessary parameters for oscillation damped moves was successful. However, a few precautions must be taken to ensure success with this method. The initial guess for the angle must be larger than the initial guess for the angular velocity. Nonnegative values were used for both. An error in the initial guesses of up to 15% of the mass and up to 20% of the length of the swinging object can be tolerated. The force sensor's coordinate frame must be aligned with gravity.

The batch estimator is based on a model that expects input data, which is symmetrical about the weight of the suspended object and it cannot calculate the desired parameters accurately if the input force data is skewed. Also, the force data is expected to be reasonably continuous. The algorithm checked to see if each force data point read was similar to the previous reading in order to eliminate a large noise spike. If the force reading was unacceptable, another reading was taken. If a number of sampling periods passed between acceptable readings, the estimator would not converge as closely as usual (i.e., the performance function value was higher at the end of the final iteration) and the value for the period was a few percent further away from the expected value. Another effect to note is that noise in the force readings will be more significant with a lighter object where the range of force readings is smaller, than for a heavier object. In these experiments, a force sensor with a range of 500 pounds and a resolution of 0.24 pounds was used. Thus, noise can have a larger effect with small ranges of force data. A noise error of just one least significant bit (0.24 lb) is a significant percentage (2%) of the range of the data for a typical test of the lightest test object. However, even with these potential errors the batch, NLS estimator determined the model parameters that were sufficient to formulate 95% oscillation damped trajectories.

For real hardware applications the following steps should be followed:

1. Develop a model.

2. Tune the estimator with simulated noisy data.

3. Retune with experimental data (pick initial estimates and weights).

4. Implement on hardware.

4.3.3. Frequency Domain NLS

An alternative to NLS in the time domain is NLS in the frequency domain. In some situations, it may be easier to identify a system model in the frequency domain (see Section 5.3.3.), such as the mode shapes and frequencies of a large flexible structure. The Fast Fourier Transform (FFT) is the cornerstone of this approach. The FFT is applied to the system inputs and outputs to determine transfer functions (i.e., Bode plots) in order to identify the critical parameters. Here is an example of adaptive input shaping with near real-time parameter identification using the FFT and NLS in the frequency domain (see Figure 4.4).

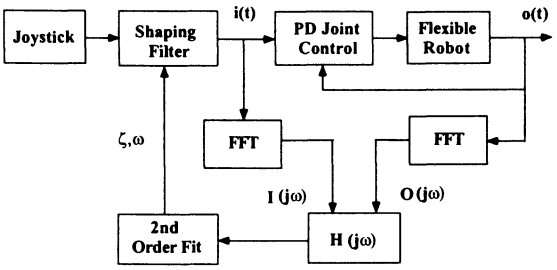

Figure 4.4. NLS in frequency domain

The steps are as follows:

1. Model (second-order transfer function) of flexible robot

$$H(s) = \frac{A}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

$$h(t) = \frac{A}{\omega_d} e^{-\zeta\omega_n t} \sin \omega_d t$$

$$\omega_d = \omega_n \sqrt{1 - \zeta^2}.$$

2. Frequency domain form

$$H(j\omega) = \frac{A}{(\omega_n^2 - \omega^2) + j2\zeta\omega_n\omega}$$

$$|H(j\omega)| = \frac{A}{[\omega^4 + 2\omega_n^2(2\zeta^2 - 1)\omega^2 + \omega_n^4]^{\frac{1}{2}}}.$$

3. Estimation/System ID problem

$$y_i = |H(j\omega)| = \frac{A}{[\omega_i^4 + 2\omega_n^2(2\zeta^2 - 1)\omega_i^2 + \omega_n^4]^{\frac{1}{2}}}$$
$$\mathbf{p} = (A, \zeta, \omega_n)^T.$$

4. Linearized matrix/NLS formulation

$$\mathbf{A} = \left[\frac{\partial y_i}{\partial A}, \frac{\partial y_i}{\partial \zeta}, \frac{\partial y_i}{\partial \omega_n}\right]$$
$$\frac{\partial y_i}{\partial A} = \frac{1}{[\omega_i^4 + 2\omega_n^2(2\zeta^2 - 1)\omega_i^2 + \omega_n^4]^{\frac{1}{2}}}$$
$$\frac{\partial y_i}{\partial \zeta} = \frac{-4A\zeta\omega_n^2\omega_i}{[\omega_i^4 + 2\omega_n^2(2\zeta^2 - 1)\omega_i^2 + \omega_n^4]^{\frac{3}{2}}}$$
$$\frac{\partial y_i}{\partial \omega_n} = \frac{-2A\omega_n[(2\zeta^2 - 1)\omega_i^2 + \omega_n^2]}{[\omega_i^4 + 2\omega_n^2(2\zeta^2 - 1)\omega_i^2 + \omega_n^4]^{\frac{1}{2}}}.$$

4.4. Homotopy Methods

The origin of modern homotopy, or continuation algorithms can be traced back to Davidenko[12] in 1953, when large scale digital computation was just a dream. Since then, the computer has become an integral part of scientific research and homotopy methods have been applied in a variety of studies[13,14]. The homotopy methods are actually based upon the one and two parameter families of functional variations of mathematical analysis. For example, Bliss[15] used these methods to prove early versions of the maximum principle. These methods are often found in existence and uniqueness proofs in solution of algebraic and differential equations.

Typically homotopy methods are used to ***unique your way*** to a solution for root solving, estimation, System ID, and optimization by increasing the convergence region. In other words, the nonlinear solver, such as the NLS algorithm will be insensitive to the initial guess of the parameters at a cost of increased run time.

4.4.1. Root Solving

A good example to demonstrate the concept is the imbedding process for root solving. Given a nonlinear vector function to solve

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}.$$

Break it up into linear and nonlinear vector equations

$$(\mathbf{A}\mathbf{x} + \mathbf{c}) + \mathbf{F}_N(\mathbf{x}) = \mathbf{0},$$

or add a linear part

$$(\mathbf{x} - \mathbf{c}) + \mathbf{F}(\mathbf{x}) = \mathbf{0}.$$

Next, rewrite the equations with an imbedded homotopy parameter β as

$$(\mathbf{A}\mathbf{x} + \mathbf{c}) + \beta \mathbf{F}_N(\mathbf{x}) = \mathbf{0},$$

or

$$(1 - \beta)(\mathbf{x} - \mathbf{c}) + \beta \mathbf{F}(\mathbf{x}) = \mathbf{0}.$$

The last step is to iteratively solve these modified equations with Newton's method from $\beta = 0$ to $\beta = 1$. This is equivalent to solving the linear problem to obtain an initial guess and then propagating it into the nonlinear solution. In order to illustrate the process, a simple numerical example is provided.

Example 4.2. Solve $x^2 + 2x + 1 = 0$

Solution

$$\beta x^2 + 2x + 1 = 0.$$

For $\beta = 0$,

$$x = \frac{-1}{2}.$$

For $\beta = \frac{1}{4}$,

$$x^2 + 8x + 4 = 0.$$

So,

$$x_{1,2} = \frac{-1.07}{2}, \frac{-14.93}{2}.$$

For $\beta = \frac{1}{2}$,

$$x^2 + 4x + 2 = 0.$$

So,

$$x_{1,2} = \frac{-1.17}{2}, \frac{-6.83}{2}.$$

For $\beta = \frac{3}{4}$,

$$3x^2 + 8x + 4 = 0.$$

So,

$$x_{1,2} = \frac{-2}{3}, 2$$

For $\beta = 1$,

$$x^2 + 2x + 1 = 0.$$

So,

$$x_{1,2} = -1.$$

The results are shown graphically in Figure 4.5.

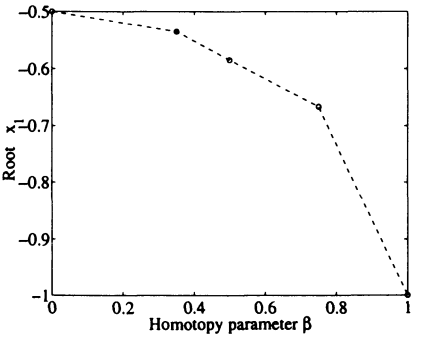

Figure 4.5. Homotopy map

A method to generate a continuous homotopy mapping is to employ Davidenko's method. Davidenko's method solves an initial value problem that is developed from the imbedding process.

Given the same nonlinear vector function to solve

$$\mathbf{F(x)} = \mathbf{0},$$

an imbedded equation is created and a homotopy map ($\mathbf{z} = \mathbf{z}(\beta)$) is defined to produce

$$\mathbf{H}(\mathbf{z}, \beta) = (1 - \beta)(\mathbf{z} - \mathbf{c}) + \beta \mathbf{F}(\mathbf{z}) = \mathbf{0}.$$

Davidenko's equation is generated by differentiating with respect to β

$$\frac{d}{d\beta}[\mathbf{H}(\mathbf{z}, \beta)] = \left[(1 - \beta)I + \beta \left[\frac{\partial \mathbf{F}}{\partial \mathbf{z}}\right]\right] \left(\frac{\partial \mathbf{z}}{\partial \beta}\right) - (\mathbf{z} - \mathbf{c}) + \mathbf{F}(\mathbf{z}), \tag{4.9}$$

with initial conditions of $\beta = 0$ and $\mathbf{z}(0) = \mathbf{c}$.

The homotopy problem is solved by integrating Eq. (4.9) from $\beta = 0(\mathbf{z}(0) = \mathbf{c})$ to $\beta = 1(\mathbf{z}(1) = \mathbf{x})$ to obtain the homotopy map ($\mathbf{z} = \mathbf{z}(\beta)$).

4.4.2. Increase the Convergence Region

Another application of interest for the homotopy method is to increase the convergence region of optimal control problems. In particular, a modified version of the differential equation imbedding and continuation method[14] is utilized to enhance the convergence of the Recursive Quadratic Programming (RQP) routine discussed in Chapter 5 (see Section 5.4.10.).

The differential equation imbedding and continuation method is most often employed when a set of differential equations can be separated into linear and nonlinear parts. The linear part is solved first and that solution is continuously deformed into the solution of the actual nonlinear problem. For example,

$$\begin{aligned} J &= \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}, t)dt \\ \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \beta(\text{nonlinear terms}) \\ &\mathbf{x}(t_0), \mathbf{x}(t_f), t_0, t_f \text{ prescribed} \\ &\mathbf{u}(t) \text{ unknown.} \end{aligned}$$

The continuous iteration is effected by imbedding a homotopy parameter (β) that is swept from zero (linear problem) to one (actual nonlinear problem), therefore continuously mapping the linear solution into the nonlinear solution.

Akin to this separation process is the augmentation of a similar, simpler set of differential equations to the set of interest. The similar set of differential equations is usually a simplification of the original problem, thereby creating a one parameter (β) family, for which, the solution at $\beta = 0$ is easy to solve, or

$$\begin{aligned} J &= \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}, t)dt \\ \dot{\mathbf{x}} &= (1 - \beta)\mathbf{f}_s(\mathbf{x}, \mathbf{u}, t) + \beta \mathbf{f}_A(\mathbf{x}, \mathbf{u}, t). \end{aligned}$$

The solution of the simplified problem is used to initialize the continuation process (at $\beta = 0$), and the subsequent solutions (for $\beta > 0$) can be initiated with an arbitrarily close neighboring solution[16]. This technique is used in the input shaping optimization process for flexible link robots in Chapter 5 and has been applied to a reentry vehicle trajectory optimization problem[16].

4.5. Robot and Actuator System ID

When developing simplified system models, the linearization of nonlinear models may not be valid for fast robotics maneuvers. In these situations, the full nonlinear equations of motion must be used for successful input shaping, trajectory planning, and control. In this section, a nonlinear system identification technique is described for determining the unknown parameters of a nonlinear robot model. This technique is called the Backward Propagation Technique (BPT). The BPT approach begins with the outermost joint and works inboard (or backwards) a joint at a time until the robot is completely identified. To illustrate this process, a three joint model of a robot (representing the last three joints of the Fanuc S-800) is identified with the BPT. Figure 4.6 shows the kinematic and dynamic structure of the three degree-of-freedom robot model. The equations of motion as derived with Lagrange's equations

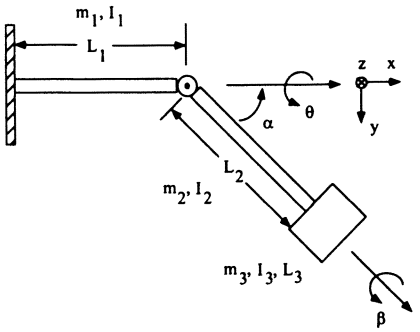

Figure 4.6. Three degree-of-freedom model

are

$$\begin{bmatrix} M_{11} & 0 & M_{13} \\ 0 & M_{22} & 0 \\ M_{31} & 0 & M_{33} \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix} + \begin{bmatrix} c_{11} & 0 & 0 \\ 0 & c_{22} & 0 \\ 0 & 0 & c_{33} \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} + \begin{bmatrix} k_{11} & 0 & 0 \\ 0 & k_{22} & 0 \\ 0 & 0 & k_{33} \end{bmatrix} \begin{bmatrix} \theta \\ \alpha \\ \beta \end{bmatrix}$$
$$+ \begin{bmatrix} -D_1(\sin\alpha)^2 \\ 0 \\ -I_{x3}\sin\alpha \end{bmatrix} \dot{\theta}\dot{\alpha} + \begin{bmatrix} 0 \\ I_{x3}\sin\alpha \\ 0 \end{bmatrix} \dot{\theta}\dot{\beta} + \begin{bmatrix} -I_{x3}\sin\alpha \\ 0 \\ 0 \end{bmatrix} \dot{\beta}\dot{\alpha}$$
$$+ \begin{bmatrix} 0 \\ 0.5D_1(\sin\alpha)^2 \\ 0 \end{bmatrix} \dot{\theta}^2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (4.10)$$

where

$$\begin{aligned} M_{33} &= I_{x3} \\ M_{31} &= M_{13} = I_{x3}\cos\alpha \\ M_{22} &= I_{y2} + I_{y3} + 0.25m_2L_2^2 + m_3(L_2 + 0.5L_3)^2 \\ M_{11} &= I_{x1} + (I_{x2} + M_{33})(\cos\alpha)^2 + M_{22}(\sin\alpha)^2 \\ D_1 &= I_{x2} + M_{33} - M_{22}. \end{aligned}$$

The first step in the BPT process is to analyze the equations of motion and identify the appropriate robot configurations that are necessary to isolate each joint and associated model parameters. The typical second-order system parameters are identified by performing Single-Input-Single-Output (SISO) maneuvers (typically, a step command and/or a sine sweep) with two different known payloads on each joint with the other joints held at zero degrees. The SISO maneuver on the third (outer most) joint provides the values of M_{33}, c_{33}, and k_{33}. The SISO maneuver on the second joint provides the values of M_{22}, c_{22}, and k_{22}. Next, the second joint is configured at $\alpha = 0$ and $\alpha = \pi/2$ in order to generate two sets of SISO maneuvers on the first joint to obtain I_{x1}, I_{x2}, c_{11}, and k_{11}. As a result M_{11}, M_{13}, M_{31}, and D_1 are calculated from the previously identified parameters and a fully coupled, nonlinear model is created from a series of SISO system identification procedures.

Whether performing linear system identification, or the nonlinear BPT of Eq. (4.10), various transfer functions must be fit to experimental data relating input joint angle commands to the joint angle response. In brief, the method employed for obtaining transfer functions may be stated by the following four step procedure:

1. Obtain experimental joint response data for a known input joint angle command profile.

2. Postulate a transfer function relating input joint angle commands to output joint angle response where the transfer function is parameterized by using several unknown quantities.

3. Define a cost function reflecting the amount of error between the experimental joint angle response data and the output of the transfer function for a known input.

4. Using a numerical optimization code find values for the initially unknown parameters of the transfer function that minimize the cost function.

Several types of input joint angle commands are examined in an effort to determine the best type of input/output data sets for generating accurate joint angle transfer functions. The results indicated that a simple step input yields the most reliable estimation of input/output transfer functions.

The postulated transfer function was based on the form of Eq. (4.11) which has a Proportional-Derivative (PD) controller connected in series. A block diagram representing the postulated system is shown in Figure 4.7 where θ_{in} is the input joint angle command, θ_{out} is the joint angle response, θ_e is joint angle error, K_p and K_d are the proportional and derivative gains, respectively of the PD controller, K and D_0 are parameters of the joint angle dynamics, and s is the Laplace transform variable.

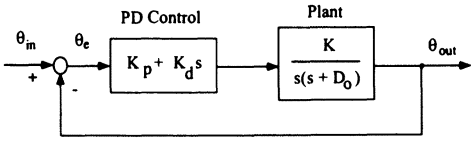

Figure 4.7: Block diagram of postulated input/output system for determining the input/output transfer function

The unknown parameters being estimated include K_p, K_d, K, and D_0. The net transfer function relating joint angle input commands to joint angle response is

$$\frac{\theta_{out}}{\theta_{in}} = \frac{K(K_d s + K_p)}{s^2 + (D_0 + KK_d)s + KK_p}. \tag{4.11}$$

The cost function used for the optimization procedure is the integrated weighted squared error between the measured joint angle response and the predicted joint angle response based on the transfer function of Eq. (4.11) for several sets of data. The cost function is given by

$$J = \sum_{i=1}^{n} W_i \int_{t_0}^{t_f} (\theta_m - \theta_a)_i^2 dt \tag{4.12}$$

where n is the number of data sets used, W_i are weighting constants, t_0 is the start time of the maneuver, t_f is the final time of the maneuver, θ_m is the experimentally measured joint angle response, and θ_a is the analytically determined joint angle

response. The primary feature of this cost function is that as the predicted angle response approaches the *measured* angle response, the cost function tends to zero.

The optimization code is taken directly from MATLAB®'s Optimization Toolbox via the *constr* function (see Appendix B). This code implements a quadratic programming technique for finding the values of K_p, K_d, K, and D_0 that minimize the cost function of Eq. (4.12).

This procedure was used to estimate the linear transfer function of joint 1 of the Fanuc S-800 robot. A similar analysis was performed on joints 2 through 6. The next step would have been to perform the nonlinear BPT procedure and use these results to modify the robot's motion profiles to account for robot dynamic effects at higher speeds. Because of time and funding constraints, this task was not done. Nevertheless, the following discussion gives a glimpse of how the identification was performed.

Three step response data sets are used to obtain a transfer function for joint 1 of the Fanuc S-800. The step responses are for input angle commands of $1°, 5°$, and $10°$. A significant nonlinear effect was observed in the angle response data for large angle rate commands. These effects are of sufficient magnitude for the maneuver that modeling of the nonlinearity is required. Several nonlinear elements were examined to predict the observed behavior of the experimental data. The best match was obtained when a saturation-like element was inserted between the PD controller and the plant for the analytical model. The block diagram of Figure 4.7 is modified to reflect this change as shown in Figure 4.8.

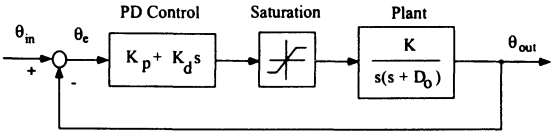

Figure 4.8: Block diagram of postulated input/output system for determining the input/output transfer function, with saturation

Since the exact form of the saturation function is unknown, like the system dynamics it too is parameterized with several unknown quantities. The optimization procedure is carried out to find not only the values of K_p, K_d, K, and D_0, but also the form of the saturation function. The resulting saturation function is shown in Figure 4.9 and the estimated model parameters are given in Table 4.2.

Table 4.2. Estimated parameters for joint 1

K_p	K_d	K	D_0
400	7.5	0.5	27.6

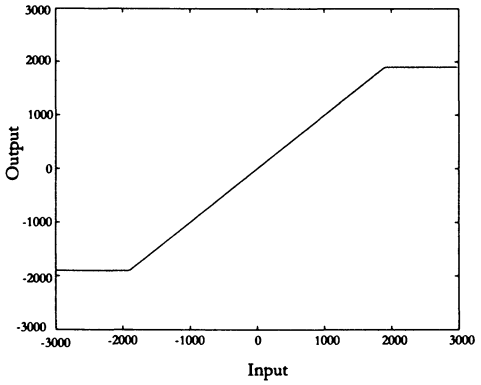

Figure 4.9. Saturation function profile

Plots of measured joint angle responses along with responses using the estimated analytical model are shown in Figure 4.10 for the three step response data sets (1°, 5°, and 10°) considered. The evident agreement between analytical and measured data is indicative of the accurate estimation achieved.

Example 4.3. System ID via numerical optimization (see Figure 8.7).

An analytical time-domain model is calibrated with respect to an actual single-axis servo system. The hub model was identified by creating a time-domain numerical simulation model within MATLAB®. Initially, three uncertain parameters the hub inertia $\widetilde{I}_{hub}$, viscous friction constant $\widetilde{B}_{vf}$, and the Coulomb friction constant $\widetilde{C}_{df}$ were singled-out for System ID. These parameters are associated with the slewing flexible link rigid hub and the servo system defined as

$$\tau = \widetilde{I}_{hub}\ddot{\theta} + \widetilde{B}_{vf}\dot{\theta} + \widetilde{C}_{df}sgn(\dot{\theta}). \tag{4.13}$$

A series of experiments were run without the flexible link attached to the rigid hub (see Figure 3.8). A simple PD control system was used to slew the hub

$$\tau = K_p\left(\theta_{REF} - \theta\right) + K_d\left(\dot{\theta}_{REF} - \dot{\theta}\right) \tag{4.14}$$

where the control gains are fixed constants and listed in Table 4.3.

The steps used to perform a System ID/optimization procedure are as follows:

1. Select arbitrary controller gains.

2. Use a general input trajectory, or simple step responses.

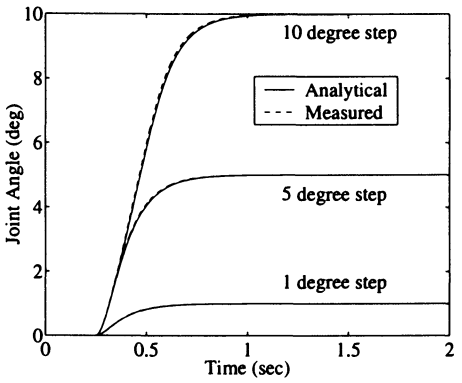

Figure 4.10: Comparison of measured and analytical joint angle responses to 1°, 5°, and 10° joint angle step commands, respectively

3. Run experiments that generate torque, angle, and angle rate responses.

4. Use same input trajectories to generate simulation responses.

5. Define a cost function that minimizes the error between experimental and simulation responses.

6. Set-up the optimizer such that the *parameters* are inequality constraints.

7. Allow the numerical optimization algorithm to minimize the error by adjusting the simulation model *parameters*.

8. Once the optimizer has converged to a minimum, the parameters are identified and the mathematical model is matched.

9. Validate the mathematical model response with a new data set.

10. If Step 9 is successful, then the mathematical model is calibrated.

The System ID problem was recast as a constrained optimization problem. It was implemented with the MATLAB® Optimization Toolbox and the *constr* function. The cost index per iteration used was

$$J = \int_{t_0}^{t_f} (\theta_{model} - \theta_{experimental}) \cdot (\theta_{model} - \theta_{experimental})\, dt$$

where t_0 and t_f are the initial and final times of interest. The cost index was subject

to the following constraints:

$$\left(\widetilde{I}_{hub}\right)_{min} \leq \widetilde{I}_{hub} \leq \left(\widetilde{I}_{hub}\right)_{max}$$
$$\left(\widetilde{B}_{vf}\right)_{min} \leq \widetilde{B}_{vf} \leq \left(\widetilde{B}_{vf}\right)_{max}$$
$$\left(\widetilde{C}_{df}\right)_{min} \leq \widetilde{C}_{df} \leq \left(\widetilde{C}_{df}\right)_{max}$$

The parameters were allowed to vary between their respective minimum and maximum values.

The servo hub assembly was slewed with two trajectories. A cubic spline trajectory generated θ_{REF} and $\dot{\theta}_{REF}$ servo inputs. Two cases (45° and 90°) were used to identify the parameters. In Case 1, the hub assembly was slewed from 0° to 90° in 0.15 seconds. The torque input angle and angle-rate responses were recorded and employed by the optimizer to identify the parameters in Eq. (4.13). In Case 2, the hub assembly was slewed from 0° to 45° in 0.15 seconds. By servoing as a *new* input to the model, Case 2 was used to verify the calibrated model. The results from the optimizer for the servo model parameters are given in Table 4.3. The hub angle responses for both cases along with their corresponding torque inputs are shown in Figure 4.11. The hub angle rate responses for both cases are shown respectively in the pair of plots in Figure 4.12. Reference angles and angle rate trajectories are also shown respectively in Figure 4.11 and Figure 4.12. The reference trajectories were intentionally set to aggressive levels in order to ensure sufficient excitation of the system so that the optimizer could consistently identify the dynamics of the system.

Table 4.3. System identification, physical and control system parameters

Parameter	Symbol	Value	Unit
Proportional Gain	K_p	2.0	$N-m/rad$
Derivative Gain	K_d	0.1	$N-m-sec/rad$
Hub Inertia	$\widetilde{I}_{hub}$	0.01328165	$kg-m^2$
Viscous Friction	$\widetilde{B}_{vf}$	0.00121639	$kg \cdot m^2/s$
Coulomb Friction	$\widetilde{C}_{df}$	0.2102728	$N \cdot m$

4.6. Chapter 4 Summary

In this chapter the need for acquiring the transfer function and/or the time domain model of dynamic systems was introduced. In order to design successful

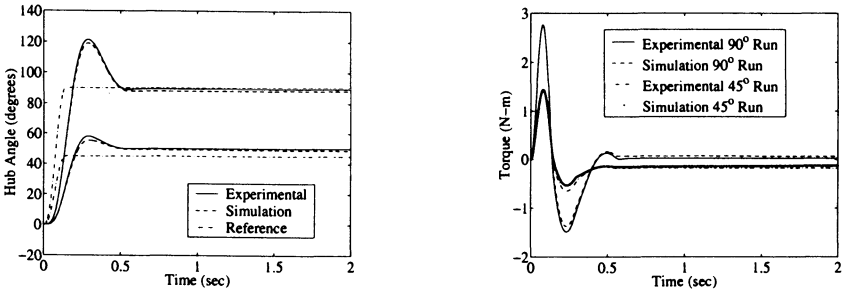

Figure 4.11. System identification hub angle responses and torque inputs

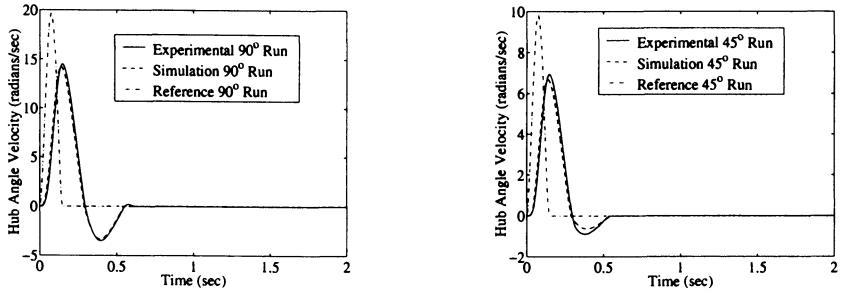

Figure 4.12. System identification hub angle rate responses

control systems, accurate models of the plant being controlled become a prerequisite. Several methods including linear least squares, nonlinear least squares, and the backward propagation technique were introduced. Several examples were reviewed that illustrate these methods for robot components, such as actuator and servo systems. Homotopy methods, which are nonlinear solvers, with applications for root solving solutions, estimation, and optimization were introduced. These methods help to increase the convergence region. An example used homotopy to help find a solution to a nonlinear function by starting with the linear system and evolving it to the nonlinear one. This method is useful in numerical optimization techniques as will be demonstrated in Chapters 5 and 6. In addition, the results of a System ID involving a real servo system is incorporated into a control system design in Chapter 8.

4.7. Chapter 4 References

1. P. A. Tzes, and S. Yurkovich, "A frequency domain identification scheme for flexible Structure Control," 27th IEEE Conference on Decision and Control, Austin, TX, December 1988.

2. W. L. Nelson, and D. Mitra, "Load estimation and load-adaptive optimal control for a flexible robot arm," IEEE Robotics and Automation Conference, San Francisco, CA, April 1986.
3. B. Noble, and J. Daniels, *Applied Linear Algebra*, Prentice-Hall, Englewood Cliffs, New Jersey, 1977.
4. J. L. Junkins, *An Introduction to Optimal Estimation of Dynamical Systems*, Sijthoff & Noordhoff, 1978.
5. W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes*, Cambridge University Press, Cambridge, 1986.
6. D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, 1989.
7. K. Ogata, *Modern Control Engineeing*,, Prentice-Hall, 1970.
8. J. F. Jones, and B. J. Petterson, "Oscillation damped movement of suspended objects," IEEE International Conference on Robotics and Automation, Philadelphia, PA, April, 1988.
9. G. R. Eisler, "FLAWLES: FLight Analysis by Weighted LEast Squares," SAND Report, Sandia National Laboratories, 1992.
10. J. L. McDowell, "TRAP: TRajectory Analysis Program," University of Texas at Austin, Center for Advanced Study in Orbital Mechanics, Austin, TX, TR77-33, 1977.
11. J. F. Jones, B. J. Petterson, and J. C. Werner, "Swing damped movement of suspended objects," SAND Report #87-2189, Sandia National Laboratories, 1987.
12. D. F. Davidenko, " On a new method if numerical solution of systems of non-linear equations," Dokl. Akad., Nauk. USSR, Vol. 88, 1953, pp. 601-602.
13. J. P. Dunyak, J. L. Junkins, and L. T. Watson, "Robust nonlinear least squares estimation using the Chow-York homotopy method," Journal of Guidance, Control, and Dynamics, Vol. 7, No. 6, Nov.-Dec. 1984, pp. 752-754.
14. J .L. Junkins, and J. P. Dunyak, " Continuation methods for enhancement of optimization algorithms," 19th Annual Meeting of the Society of Engineering Science, University of Missouri-Rolla, October 1982.
15. G. A. Bliss, Lectures on the Calculus of Variations, The University of Chicago Press, 1946, Chapter II.
16. R. D. Robinett, "A unified approach to vehicle design, control, and flight path optimization," Ph.D. Dissertation, Texas A&M University, December 1987.

4.8. Chapter 4 Problems

Homework 4.1. From Section 4.2.

1. Approximate $y(t) = \sin(2\pi t)$ for $0 \leq t \leq 1/4$ with a third-order polynomial and use five and nine equally spaced points

$$y(t) = p_0 + p_1 t + p_2 t^2 + p_3 t^3.$$

2. Approximate $y(t) = t$ for $0 \leq t \leq 1$ with

$$y(t) = p_0 + p_1 \sin(2\pi t) + p_2 \cos(2\pi t)$$

and use the same spacing as #1.

Homework 4.2. From Section 4.3.1.

1. Approximate

$$y(t) = \sin(\pi t)\cos(\pi t) + 4\sin(2\pi t)\cos(2\pi t)$$

for $0 \leq t \leq 2$ with

$$\mathbf{y}(t) = p_0 + p_1 \sin(p_2 t + p_3) + p_4 \cos(p_5 t + p_6)$$

and use 11 and 21 equally spaced points.

Homework 4.3. From Section 4.3.2.

1. Derive the first partial derivative for $\partial y_i / \partial \omega$ in the example section.
2. Estimate the period of the gantry robot by using the force sensor data in the y-direction. Try the estimation routine with and without added noise to the simulated force sensor data.

Homework 4.4. From Section 4.3.3.

Develop a nonlinear least squares frequency domain estimator for Homework problem 4.3, #2 described above. Perform simulations and initial tuning.

Homework 4.5. From Section 4.4.

1. Solve the following nonlinear equation:

$$x^3 + 2x^2 + x + 1 = 0.$$

   a) Use a homotopy imbedding process and Newton's method.
   b) Use Davidenko's method.

   Hint: Verify with the CRC Math Tables.

2. Solve the following nonlinear vector function:

$$\begin{aligned} x_1^3 + 2x_1^2 + x_2 + x_3 + 1 &= 0 \\ x_2^2 + 2x_1 + x_3 + 2 &= 0 \\ x_3^2 + 3x_1 + 2x_2 + x_3 + 1 &= 0. \end{aligned}$$

   a) Use a homotopy imbedding process and Newton's method.
   b) Use Davidenko's method.

Homework 4.6. From Section 4.5.

1. Perform System ID on the single joint model in Figure 4.7.

   a) Utilize the method developed in Homework problem 4.4.

   b) Convert Eq. (4.11) to state-space form and use the method discussed in Section 4.3..

   c) Try different sets of K_p, K_d, K, and D_0 that are perturbations of the values in Table 4.2.

Chapter 5

Input Shaping for Path Planning

5.1. Introduction

Input shaping is an effective way to optimize the performance of robots, flexible structures, spacecraft, telescopes, and other systems that have vibration, control authority, tracking, and/or pointing constraints. These constraints along with the dynamics and kinematics of the system under consideration can be included in a trajectory optimization/path planning procedure to ensure that the system meets the desired performance. Input shaping is particularly useful when the closed-loop controller cannot be modified or tuned. For example, many pedestal-based robots have closed architecture control systems that restrict access to the servo-loop controls. This chapter begins with the overhead gantry robot and a vibration constraint referred to as *swing-free* input shaping.

5.2. Analytic Solutions for Input Shaping

Analytical solutions are useful in developing intuition and rules-of-thumb as well as check cases for complicated simulations. One can derive analytical input shaping solutions from any set of differential equations that are closed form solvable. The gantry robot model that was derived in Section 2.6. is one such example.

The linearized equations of motion for that model are

$$\ddot{\theta} + w^2\theta = -\frac{1}{l}\ddot{x} \tag{5.1}$$

where

$$\omega^2 = \frac{g}{l}$$
$$\tau = \frac{2\pi}{\omega},$$

R. D. Robinett III et al., *Flexible Robot Dynamics and Controls*

with a typical set of initial conditions

$$\theta(0) = \dot{\theta}(0) = 0$$
$$x(0) = \dot{x}(0) = 0.$$

The input shaping problem of interest is referred to as *swing-free*, which implies that the gantry robot is moved from a starting position without the pendulum swinging to an arbitrary final position without the pendulum swinging. This swing-free constraint is defined mathematically as

$$\theta(0) = \dot{\theta}(0) = 0$$
$$x(0) = \dot{x}(0) = 0$$
$$\theta(t_f) = \dot{\theta}(t_f) = 0$$
$$\dot{x}(t_f) = 0$$
$$x(t_f) = X_f.$$

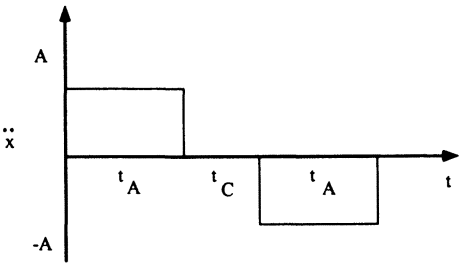

Figure 5.1. Input shaping acceleration profile (bang-coast-bang)

The input shaping variable is the gantry robot acceleration profile (i.e., $\ddot{x}(t)$ time history). A straight forward way to determine the desired acceleration profile is to assume the form presented in Figure 5.1 and recognize that the swing-free constraint can be rewritten as

$$\theta(t_A) = -\theta(t_A + t_C) \tag{5.2}$$
$$\dot{\theta}(t_A) = \dot{\theta}(t_A + t_C). \tag{5.3}$$

Equation (5.1) can be integrated over the assumed acceleration profile to determine the values of A, t_A, and t_C, which are referred to as constant acceleration, acceleration time, and coast time, respectively. The first integration is over the acceleration time $0 \leq t \leq t_A$, which results in

$$\ddot{\theta} + \omega^2\theta = -\frac{1}{l}A,$$

with

$$\omega^2 = \frac{g}{l}$$
$$\theta(0) = \dot{\theta}(0) = 0$$
$$x(0) = \dot{x}(0) = 0.$$

Therefore,

$$\theta(t) = \frac{A}{g}[\cos\omega t - 1] \tag{5.4}$$
$$\dot{\theta}(t) = \frac{-A\omega}{g}\sin\omega t. \tag{5.5}$$

The second integration is over the coast time $t_A \leq t \leq t_A + t_C$, which results in

$$\ddot{\theta} + \omega^2\theta = 0.$$

Therefore,

$$\theta(t) = \theta(t_A)\cos\omega(t - t_A) + \frac{\dot{\theta}(t_A)}{\omega}\sin\omega(t - t_A) \tag{5.6}$$
$$\dot{\theta}(t) = -\omega\theta(t_A)\sin\omega(t - t_A) + \dot{\theta}(t_A)\cos\omega(t - t_A). \tag{5.7}$$

At this point, a simplifying assumption is made and verified at the end of the derivation. Namely, $t_A + t_C = \tau = 2\pi/\omega$. Equation (5.6) and Eq. (5.7) are compared to Eq. (5.4) and Eq. (5.5), which meet the swing-free constraint of Eq. (5.2) and Eq. (5.3), or

$$\begin{aligned}
\theta(t_A + t_C) &= \theta(t_A)\cos\omega(\tau - t_A) + \frac{\dot{\theta}(t_A)}{\omega}\sin\omega(\tau - t_A) \\
&= \theta(t_A)\cos\omega t_A - \frac{\dot{\theta}(t_A)}{\omega}\sin\omega t_A \\
&= \frac{A}{g}[\cos\omega t_A - 1]\cos\omega t_A + \frac{1}{\omega}\left(\frac{A\omega}{g}\right)\sin\omega t_A \sin\omega t_A \\
&= \frac{A}{g}[1 - \cos\omega t_A] \\
&= -\theta(t_A).
\end{aligned}$$

$$\begin{aligned}
\dot{\theta}(t_A + t_C) &= -\omega\theta(t_A)\sin\omega(\tau - t_A) + \dot{\theta}(t_A)\cos\omega(\tau - t_A) \\
&= \omega\theta(t_A)\sin\omega t_A + \dot{\theta}(t_A)\cos\omega t_A \\
&= \frac{\omega A}{g}[\cos\omega t_A - 1]\sin\omega t_A - \frac{A\omega}{g}\sin\omega t_A\cos\omega t_A \\
&= -\frac{A\omega}{g}\sin\omega t_A \\
&= \dot{\theta}(t_A),
\end{aligned}$$

or rewritten as

$$\begin{aligned}
\theta(t_A) &= -\theta(\tau) \\
\dot{\theta}(t_A) &= \dot{\theta}(\tau),
\end{aligned}$$

for

$$\tau = t_A + t_C.$$

This result is general, in that, it produces swing-free maneuvers as well as meeting other constraints, such as bang-bang and swing-free coast (i.e., $\tau = t_A$ and t_C is arbitrary with no swinging). Note, the constraint of the final position $x(t_f) = X_f$ can be met by selecting the value of the constant acceleration A. Homework problem 5.1 is a further generalization where the impulses are replaced with a sinusoid. The sinusoid is band-limited, thus more robust to uncertainty in the vibrational frequency. Also note, a sine wave is the first term in a more general input shaping form of the Fourier series approximation to the bang-bang and bang-coast-bang profiles.

5.3. Input Shaping Filters

The control of flexible structures becomes important as the ratio of the structure's length to it's cross sectional area increases. This will certainly become an issue for the remediation of the Department of Energy's underground waste storage tanks where a robotic arm must enter a 12 inch diameter access hole and span tanks that are 80 feet in diameter and 35 feet deep. Initial plans suggested that an operator would remotely guide the robot through the tank during the characterization and remediation phases. If the arm is controlled by using only PD control schemes, this process will be painfully slow and expensive. Controller gains will have to be significantly reduced so that large overshoots do not occur in the robot end-effector's position.

Another application for the control of flexible structures is the precise placement of heavy payloads attached to the end of a gantry crane. Substantial time is needed for an experienced operator to move the payload through often cluttered

environments. While most operators are experienced in these matters, an appropriately designed controller would allow even an inexperienced operator to perform swing-free motions while using a button box or joystick.

In recent years, many researchers have worked on the control of flexible structures, especially robotic devices. To date a general solution to the controls problem has yet to be found. One important reason is that computationally efficient mathematical methods (i.e., real-time) do not exist for solving the extremely complex sets of partial differential equations and incorporating the associated boundary conditions that most accurately model flexible structures. While general solutions do not exist, some interesting solutions do exist for simplified problems[1-6].

This section addresses the control of flexible structures by simplifing the dynamics to a set of ordinary differential equations. Both finite impulse response (FIR) and infinite impulse response (IIR) filtering techniques for reducing structural vibration in remotely operated robotic systems are discussed. These techniques utilize a discrete filter between the operator's joystick and the robot controller to alter the inputs of the system so that residual vibration and swing are reduced (see Figure 5.2).

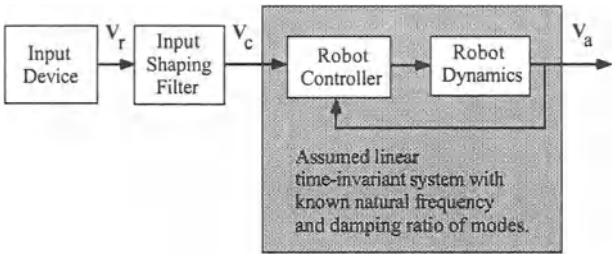

Figure 5.2. Input shaping for robotics systems

5.3.1. Finite Impulse Response Filter

Singer[1] was the first to apply a simple input shaping filter to linear time invariant (LTI) systems that modifies the reference input so that the residual vibrations are eliminated. This method involves convolving the input with a train of impulses that are calculated based on perfect knowledge of the plant's flexible mode parameters (see Figure 5.3). When these impulses are convolved with an arbitrary input, the plant follows the input without vibration and with a time delay approximately equal to the length of the impulse train (typically equal to the period of vibration). This simplification provides reasonable response when applied to a three degree-of-freedom flexible robot arm[2]. Murphy[4] later extended Singer's work by applying digital theory to the design of the shaping filter.

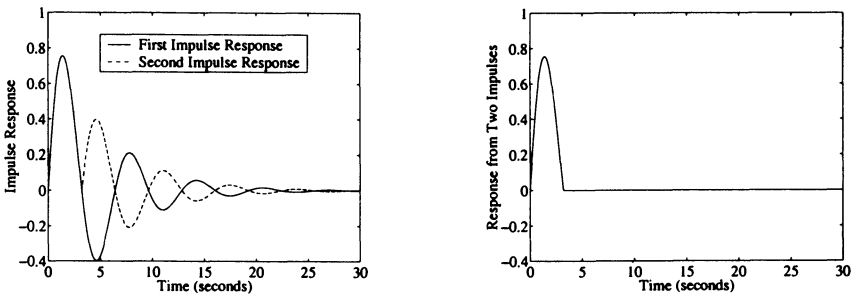

Figure 5.3: (a) Response to a LTI system to two separate impulses (b) Combined system response if the two responses in (a) are added together

Singer used a time domain approach to determine a filter that eliminates oscillations in linear systems. Consider the impulse response of a second-order underdamped system

$$y(t) = \frac{A\omega}{\sqrt{1-\zeta^2}} e^{-\zeta\omega(t-t_0)} \sin\left(\omega\sqrt{1-\zeta^2}(t-t_0)\right)$$

where ω is the natural frequency, $0 < \zeta < 1$ is the damping factor, t_0 is the time of the initial impulse, and A is the amplitude.

By superposition, the system's response to the multiple impulses is

$$y(t) = \sum_{i=1}^{N} B_i e^{-\zeta\omega t} \sin(\alpha t + \phi_i)$$

where

$$B_i = \frac{A_i\omega}{\sqrt{1-\zeta^2}} e^{\zeta\omega t_i}$$
$$\alpha = \omega\sqrt{1-\zeta^2}$$
$$\phi_i = -\omega\sqrt{1-\zeta^2}t_i.$$

For two impulses, the system is

$$y(t) = B_1 e^{-\zeta\omega t} \sin(\alpha t + \phi_1) + B_2 e^{-\zeta\omega t} \sin(\alpha t + \phi_2).$$

By combining terms, the system response can be written as

$$y(t) = Be^{-\zeta\omega t} \sin(\alpha t + \phi)$$

where

$$B = \sqrt{(B_1 \cos\phi_1 + B_2 \cos\phi_2)^2 + (B_1 \sin\phi_1 + B_2 \sin\phi_2)^2}$$
$$\phi = \tan^{-1}\left(\frac{(B_1 \sin\phi_1 + B_2 \sin\phi_2)}{(B_1 \cos\phi_1 + B_2 \cos\phi_2)}\right).$$

To eliminate vibration at the time of the second impulse, it is desirable for

$$B_1 \cos\phi_1 + B_2 \cos\phi_2 = 0$$
$$B_1 \sin\phi_1 + B_2 \sin\phi_2 = 0.$$

If the amplitude of the first pulse is one ($A_1 = 1$) and the time is zero ($t_1 = 0$), then

$$\frac{\omega}{\sqrt{1-\zeta^2}} + \frac{A_2\omega}{\sqrt{1-\zeta^2}} e^{\zeta\omega t_2} \cos(\omega\sqrt{1-\zeta^2}t_2) = 0$$
$$\frac{A_2\omega}{\sqrt{1-\zeta^2}} e^{\zeta\omega t_2} \sin(\omega\sqrt{1-\zeta^2}t_2) = 0.$$

Solving for the amplitude of the second pulse A_2 and time t_2 gives

$$t_2 = \frac{\pi}{\omega\sqrt{1-\zeta^2}}$$
$$A_2 = e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}}.$$

Finally, it is desirable to normalize the two pulses so that the overall gain of the filter is unity. The resulting FIR filter is shown in Figure 5.4

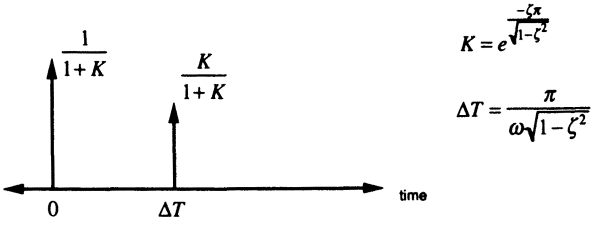

Figure 5.4. Two-impulse input shaping filter

When this filter is applied to an arbitrary reference velocity trajectory, as shown in Figure 5.2, it will alter the commanded trajectory sent to the LTI system.

For example, consider a filter for an undampened second-order system (i.e. $\zeta = 0$). If the input to the filter is a trapezoidal profile, the output of the filter is shown in Figure 5.5.

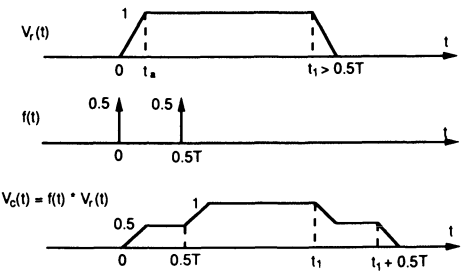

Figure 5.5: Resulting velocity trajectory after being convolved with a two-impulse input shaping filter

In general, for N impulses one must satisfy the constraints

$$V_1 = \sum_{i=1}^{N} B_i \cos \phi_i = 0$$

$$V_2 = \sum_{i=1}^{N} B_i \sin \phi_i = 0.$$

Satisfying these equations eliminates residual oscillation if ω and ζ are known exactly!

To add robustness to the system for errors in the estimate of ω and ζ, the partial derivatives of the above equations are set to zero

$$\frac{\partial}{\partial \omega} V_1 = 0$$

$$\frac{\partial}{\partial \omega} V_2 = 0$$

$$\frac{\partial}{\partial \zeta} V_1 = 0$$

$$\frac{\partial}{\partial \zeta} V_2 = 0.$$

This simplifies to the following additional constraints:

$$\sum_{i=1}^{N} B_i t_i \cos \phi_i = 0 \tag{5.8}$$

$$\sum_{i=1}^{N} B_i t_i \sin \phi_i = 0. \tag{5.9}$$

To increase robustness, filters with more pulses can be formed by using these additional constraints (see Eq. 5.8 and 5.9). The more pulses there are the more

robust the filter is to modeling errors. For example, a three-impulse shaping filter is shown in Figure 5.6. The derivation of this filter is a homework problem (see Homework problem 5.2, #1).

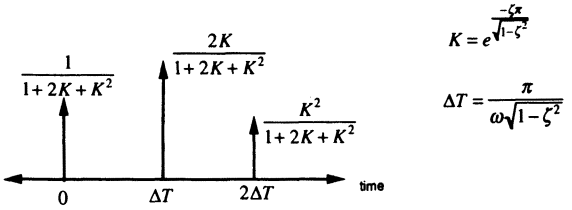

Figure 5.6. Three-impulse input shaping filter

Unfortunately, the more pulses there are the longer the delay in actuation. Figure 5.7 shows the step response of a three-impulse shaping filter. Notice, the filter delay for $\omega = 2\pi$ rad/s and $\zeta = 0.2$ is 1.02 seconds.

5.3.2. Infinite Impulse Response Filter Formulation

Singer/Seering[1] used a time domain approach to formulate the impulse shaping filters. In this section, a transfer function approach is used to formulate an Infinite Impulse Response (IIR) filter, which also eliminates oscillation. Both the design by emulation and direct z-domain methods for deriving IIR filters are investigated. The goal of this work is to reduce the time delay effect of the input shaping filters. For example, when applying Singer's three-impulse input shaping filter to a gantry crane with a suspended payload, the delay of the system was 2.75 seconds (equal to the period of oscillation). This length of delay is unacceptable when trying to position a payload. For this reason, the problem was first analyzed by using continuous time control methods and then with discrete time control methods.

Consider the block diagram in Figure 5.8. The desired transfer function is $G_d(z) = F(z)G(z)$. Next, the continuous time version of the IIR input shaping method is described.

Design by Emulation

Assume the measured transfer function of the system is the second-order underdamped system

$$G(s) = \frac{K\omega^2}{s^2 + 2\zeta\omega s + \omega^2} = \frac{K(\alpha^2 + \beta^2)}{(s+\alpha)^2 + \beta^2}$$

where $\alpha = \zeta\omega$, $\beta = \omega\sqrt{1-\zeta^2}$, K is the overall system gain, ω is the natural frequency, and ζ is the damping factor. If it is desired that the response of the

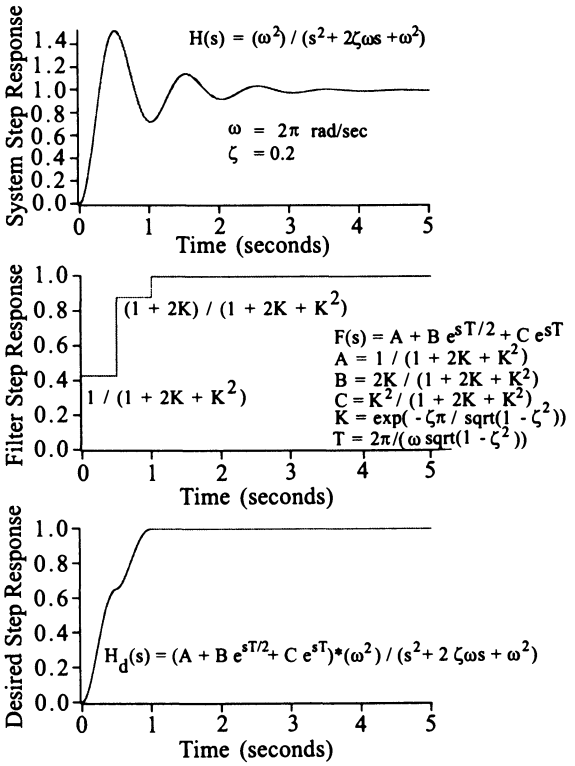

Figure 5.7: (a) LTI plant responses before input shaping (b) Step response of three-pulse input shaping filter (c) Desired response of LTI plant after input shaping

system be a critically damped second-order system, then

$$G_d(s) = \frac{K\sigma^2}{(s+\sigma)^2} \tag{5.10}$$

where σ is the desired second-order time constant. To cancel the poles of the original system and insert the poles of the desired system, the transfer function of the filter is

$$\begin{aligned} F(s) &= \frac{G_d(s)}{G(s)} \\ &= \left(\frac{\sigma^2}{\omega^2}\right) \frac{s^2 + 2\zeta\omega s + \omega^2}{(s+\sigma)^2}. \end{aligned}$$

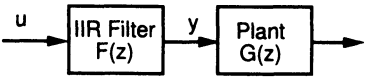

Figure 5.8. Block diagram of filter and plant

The continuous time domain filter is

$$f(t) = \left(\frac{\sigma}{\omega}\right)^2 \left[\delta(t) + 2(\zeta\omega - \sigma)e^{-\sigma t} + (\omega^2 - 2\zeta\omega\sigma - \sigma^2)te^{-\sigma t}\right].$$

The zero-order hold discrete equivalent is

$$F(z) = (1 - z^{-1})Z\left(\frac{F(s)}{s}\right).$$

Using partial fraction expansion yields

$$\frac{F(s)}{s} = \left(\frac{\sigma}{\omega}\right)^2 \left[\frac{A}{s} + \frac{B}{s+\sigma} + \frac{C}{(s+\sigma)^2}\right]$$

where

$$A = \left(\frac{\omega}{\sigma}\right)^2$$
$$B = \frac{\sigma^2 - \omega^2}{\sigma^2}$$
$$C = \frac{2\zeta\omega\sigma - \omega^2 - \sigma^2}{\sigma}.$$

Using Z-transform Tables gives

$$F(z) = \left(\frac{\sigma}{\omega}\right)^2 \left(\frac{z-1}{z}\right) \left[\frac{Az}{z-1} + \frac{B(z-1)}{(z - e^{-\sigma T})} + \frac{CTe^{-\sigma T}(z-1)}{(z - e^{-\sigma T})^2}\right]$$

where T is the sample period. Combining terms and dividing by z^2 yields

$$F(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

where

$$a_1 = -2e^{-\sigma T}$$
$$a_2 = e^{-2\sigma T}$$
$$b_0 = \left(\frac{\sigma}{\omega}\right)^2$$
$$b_1 = \left(\frac{\sigma}{\omega}\right)^2 \left[-B + (-2A - B + CT)e^{-\sigma T}\right]$$
$$b_2 = \left(\frac{\sigma}{\omega}\right)^2 \left[(B - CT)e^{-\sigma T} + Ae^{-2\sigma T}\right].$$

Discrete transfer functions of this form can be written as the following difference equation:

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) + b_0 u(k) + b_1 u(k-1) + b_2 u(k-2) \qquad (5.11)$$

where $y(k)$ and $u(k)$ are the filter's output and input, respectively. If the natural frequency ω and the dampening ratio ζ of the system are known, the coefficients a_i and b_i can be pre-computed by using Eq. (5.11). Because of the recursive terms (i.e., the poles), this equation is called an Infinite Impulse Response (IIR) filter. Whereas a filter with no recursive terms is an FIR filter. Specifically, the recursive terms create feedback and are used to weight the importance of past output data versus the input data. This equation is easy to implement on an embedded computer.

Figure 5.9 shows the step response to the above input shaping filter. By a judicious choice of σ, the designer can specify the settling time of the system. The disadvantage of this method is that the designer must be careful not to choose σ such that the torque limits of the motors are exceeded. A third pole in the denominator of $G_d(s)$ may be used to reduce the height of the initial spike in the filter. Experimentally, it was found that performance degrades considerably when the σ is chosen such that the settling time is less than 33% of the period of oscillation. Since the sensitivity of the control to parameter identification errors increases as the desired settling time reduces, this is expected.

Direct Z Domain Design

When designing the discrete version of this filter, the design by emulation method[8] worked well when the sampling rate was relatively high (2 ms for the flexible two-link arm). A direct Z-domain design method was used when the sampling rate was low (48 ms for the gantry robot). In this latter case, the plant model is first discretized and then the filter design is performed in the z-plane. The zero-order hold discrete time equivalent of the plant model is given by

$$G(z) = (1 - z^{-1})\mathcal{Z}\left(\frac{G(s)}{s}\right)$$
$$= \frac{K(Az + B)}{(z^2 - 2e^{-\alpha T}\cos(\beta T)z + e^{-2\alpha T})}$$

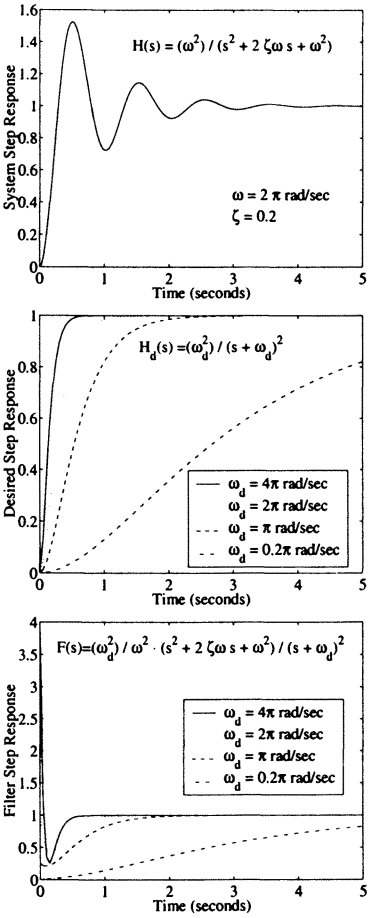

Figure 5.9: (a) LTI plant response before IIR input shaping (b) Desired response of LTI plant after IIR input shaping (c) Step response of IIR input shaping filter

where

$$A = 1 - e^{-\alpha T}\cos(\beta T) - \frac{\alpha}{\beta}e^{-\alpha T}sin(\beta T)$$

$$B = e^{-2\alpha T} + \frac{\alpha}{\beta}e^{-\alpha T}sin(\beta T) - e^{-\alpha T}\cos(\beta T)\,.$$

Suppose one would like the desired response of the system to be a critically damped third-order system. Third-order is necessary to eliminate the impulse seen

in the previous filter (see Figure 5.9). Consider

$$G_d(s) = \frac{K\sigma^3}{(s+\sigma)^3}.$$

The zero-order hold, discrete time equivalent of the desired plant is

$$G_d(z) = (1 - z^{-1})Z\left(\frac{G_d(s)}{s}\right) = \frac{z^2C + zD + E}{(z - e^{-\sigma T})^3}$$

where

$$\begin{aligned} C &= 1 - e^{-\sigma T}\left(1 + \sigma T + 0.5\sigma^2 T^2\right) \\ D &= e^{-\sigma T}\left(-2 + \sigma T + 0.5\sigma^2 T^2\right) + e^{-2\sigma T}\left(2 + \sigma T - 0.5\sigma^2 T^2\right) \\ E &= e^{-2\sigma T}\left(1 - \sigma T + 0.5\sigma^2 T^2\right) - e^{-3\sigma T}. \end{aligned}$$

The transfer function of the filter is

$$F(z) = \frac{G_d(z)}{G(z)} = \frac{\left(z^2C + zD + E\right)\left(z^2 - 2e^{-\alpha T}\cos(\beta T)\, z + e^{-2\alpha T}\right)}{(Az + B)\left(z - e^{-\sigma T}\right)^3}.$$

The pole placement resulting from this filter is shown in Figure 5.10.

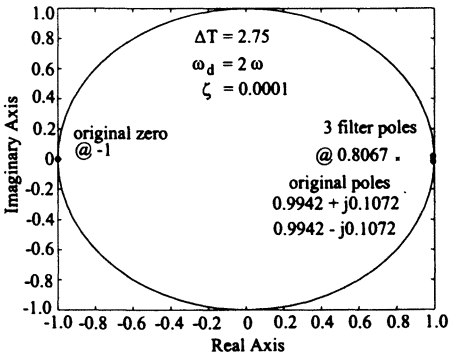

Figure 5.10. Pole placement in direct Z-domain design

Notice the pole at B/A on the unit circle at -1 if $A = B$. This means that one of the filter's output modes changes sign every sample. The residual of this mode happens to be small if the sampling period is small. However, as the sampling period increases this effect becomes noticeable (see Figure 5.11 and Figure 5.12, respectively).

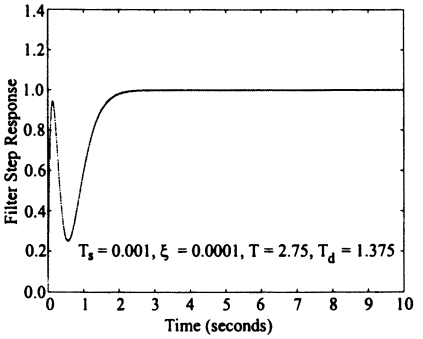

Figure 5.11: Input shaping filters step response when sampling period = 0.001 seconds

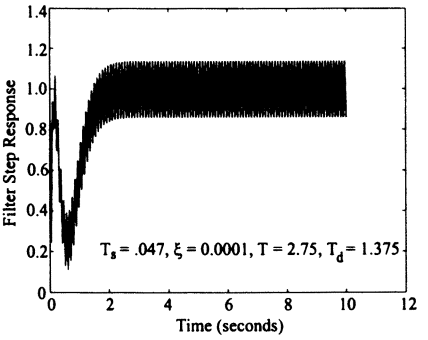

Figure 5.12: Input shaping filters step response when sampling period = 0.047 seconds

To eliminate this oscillation in the control input, this pole was moved to zero and the gain of the filter was constrained to provide unity steady state gain, i.e., $-a_1 - a_2 - a_3 + b_0 + b_1 + b_2 + b_3 + b_4 = 1$.

The resulting transfer function can be written as

$$F(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}},$$

which can then be converted to the difference equation

$$\begin{aligned} y(k) = &-a_1 y(k-1) - a_2 y(k-2) - a_3 y(k-3) + b_0 u(k) \\ &+ b_1 u(k-1) + b_2 u(k-2) + b_3 u(k-3) + b_4 u(k-4) \end{aligned}$$

where

$$
\begin{aligned}
a_1 &= -3e^{-\sigma T} \\
a_2 &= 3e^{-2\sigma T} \\
a_3 &= -e^{-3\sigma T} \\
M &= \frac{B}{A} \\
b_0 &= \frac{C}{A(1+M)} \\
b_1 &= \frac{1}{(1+M)}\left(\frac{D}{A} - 2\frac{C}{A}e^{-\alpha T}\cos(\beta T)\right) \\
b_2 &= \frac{1}{(1+M)}\left(\frac{E}{A} - 2\frac{D}{A}e^{-\alpha T}\cos(\beta T) + \frac{C}{A}e^{-2\alpha T}\right) \\
b_3 &= \frac{1}{(1+M)}\left(-2\frac{E}{A}e^{-\alpha T}\cos(\beta T) + \frac{D}{A}e^{-2\alpha T}\right) \\
b_4 &= \frac{1}{(1+M)}\left(\frac{E}{A}e^{-2\alpha T}\right)
\end{aligned}
$$

where $y(k)$ and $u(k)$ are the filter's output and input, respectively.

Figure 5.13 shows the step response to this filter. Note, the initial impulse as shown in Figure 5.9 has been removed by using a third-order desired response. Also note, for $\sigma < 2\omega$, the filter output is greater than one for a short time. In order to shorten the filters time delay, the filter must drive the LTI system harder initially. Again, the sensitivity of the control to parameter identification errors increases as the desired settling time decreases.

5.3.3. Flexible Two-Link Manipulator Example

The IIR filter technique (see Section 5.3.2.) was successfully applied to the Sandia flexible two-link manipulator (see Figure 5.24). Figure 5.14 shows the change in strain gauge measurements on the second link of the manipulator before and after input shaping. Notice how quickly the residual vibrations are dampened out. Similar results have been demonstrated with a suspended payload on the gantry crane.

As shown above, one can completely remove residual vibrations if the natural frequency ω and the dampening ratio ζ are known *a priori*. For unknown system dynamics and changing payloads, this information will not be readily available. To illustrate this condition, some simple on-line system identification experiments were performed on the flexible manipulator. One strain gauge on each link of the manipulator was used to measure the link's curvature as a function of time. These two measurements are used to determine the dominant mode of vibration on each link. The desired velocity of the joints, as commanded by the joystick,

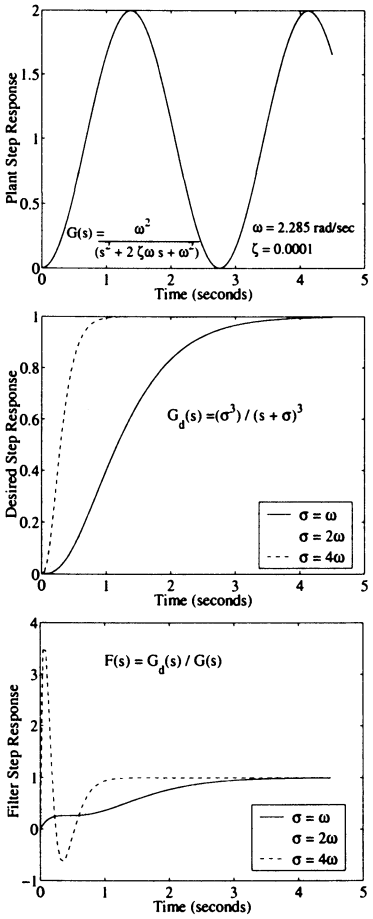

Figure 5.13: (a) LTI plant response before IIR input shaping (b) Desired response of LTI plant after IIR input shaping (c) Step response IIR input shaping filter

are used as the input to the system. By taking the FFT (Fast Fourier Transform) of both the output (strain gauge measurements) and the input (commanded joint velocities) and dividing, the system's transfer function can be determined (refer to Section 4.3.3.). The FFT of each transfer function is fit to an underdamped second-order model by using the Levenberg-Marquardt method[9]. The natural frequency and damping ratio determined from this nonlinear fit are then used to adjust the system's control. Experimentally, the above procedure was performed in a near real-time mode that was transparent to the user (see Figure 5.14). The procedure

follows. During periods of excitation, input and output data are collected, the FFTs are computed, the nonlinear fit is made, and the parameters of the IIR filter are adjusted. The input and output data are transferred via a VME-to-VME bus adapter from a 68030 board performing the control to an i860 board performing the FFTs and nonlinear fit. The i860 board is capable of performing a 1024 point FFT in one millisecond. After the nonlinear fit is performed, the natural frequency and the damping ratio are transferred back to the control CPU. Tests showed that one should be able to continually perform the system identification and update the filter at better than a 10 Hz rate. This would allow the control parameters to be adjusted during periods of excitation caused by payload variations, kinematic changes, and external disturbances.

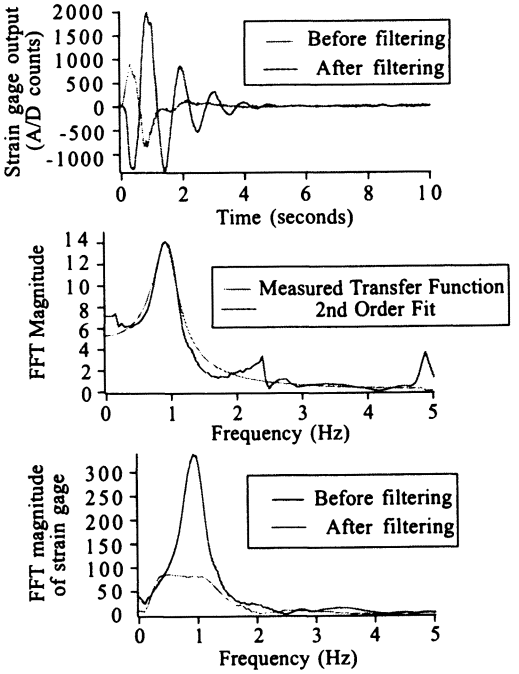

Figure 5.14: Results of system identification experiments with a flexible two-link manipulator

5.3.4. Gantry Robot Example

The IIR filter technique (see Section 5.3.2.) was successfully applied to a gantry robot with a suspended payload (see Figure 5.15). The application of Lagrange's

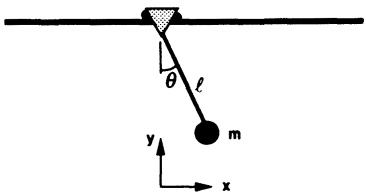

Figure 5.15. Gantry robot with suspended payload model

equation, the linearization of the nonlinear terms, and taking the Laplace transform produce the following linearized dynamic model:

$$\frac{\theta(s)}{s^2 X(s)} = \frac{\frac{-1}{l}}{s^2 + 2\zeta\omega s + \omega^2}$$

where $\omega = \sqrt{g/l}$ is the natural frequency and ζ is the damping factor. Therefore, the resulting impulse response to the acceleration in x is

$$\theta(t) = \frac{-1}{l\omega\sqrt{1-\zeta^2}} e^{-\zeta\omega t} \sin\left(\omega\sqrt{1-\zeta^2}\right).$$

Note, the sinusoidal response to acceleration input. The impulse response with respect to velocity is

$$\theta(t) = \frac{-1}{l\omega\sqrt{1-\zeta^2}} e^{-\zeta\omega t} \sin\left(\omega\sqrt{1-\zeta^2} + \phi\right)$$

where

$$\phi = \tan^{-1}\left(\frac{\sqrt{1-\zeta^2}}{-\zeta}\right).$$

Also note, the sinusoidal response to the velocity input.

Now apply the IIR filter

$$\frac{sX(s)}{V(s)} = \left(\frac{\sigma^3}{\omega^2}\right) \frac{s^2 + 2\zeta\omega s + \omega^2}{(s+\sigma)^3}.$$

The resulting transfer function and impulse response is

$$\frac{\theta(s)}{V(s)} = \left(\frac{\sigma^3}{\omega^2}\right) \frac{(\frac{-1}{l})s}{(s+\sigma)^3}.$$

The impulse response to a shaped velocity input is

$$\theta(t) = \frac{-\sigma^3}{l\omega^2}(t - 0.5\sigma t^2)e^{-\sigma t}.$$

Figure 5.16 shows the resulting damped response. Note, this is not critically damped. However, a step input in velocity would result in a critically damped third-order response.

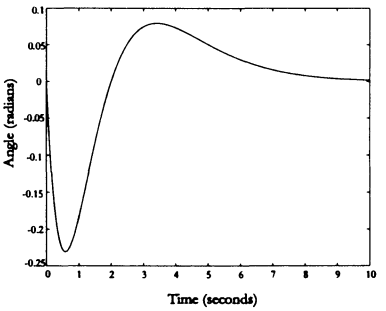

Figure 5.16. Angle response of suspended payload with IIR input filtering

5.4. Constrained Optimization with RQP

Optimization goals for aerospace problems at Sandia have ranged from maximizing re-entry vehicle impact velocity at a ground target to multi-stage rocket performance for atmospheric *sounding*, and payload insertion to support ballistic missile development efforts. Since the initial use of numerical optimization for aerospace problems at Sandia, other areas have benefited from these methods. As discussed earlier, work in the area of flexible robotics stemmed from nuclear waste site cleanup requirements. Long boom links are envisioned to evacuate contents of holding tanks built in the late 50's whose structures are steadily deteriorating. Since these links have high aspect ratios and restrict the use of typical *beefy* design structures found in traditional applications, lighter and inherently more flexible structures have to be considered. The control system design for these problems is an example of control time history parameterization (see Section 2.8.8.). These robot

models consist of finite element beam approximations and require irregular time spacings between open-loop control values to capture the motor torque variation for vibration-damped motion. This section will explore numerical (i.e., iterative) methods to generate open-loop parameterized control histories (or input shaping profiles) for flexible-link robot motion by using optimization techniques. These optimization techniques are used when the robot models are nonlinear and/or the control designer needs to develop some intuition and a feasible solution.

5.4.1. Quadratic Surfaces

The basis of flexible robot optimization is to formulate a meaningful cost function or performance index that demonstrates extremal behavior as a function of suitable parameters. A quadratic surface is a desirable multivariable performance index in $\mathbf{x} = [x_1, \cdots, x_n]^T$ that contains an extremal (maximum or minimum) and is given by

$$G(\mathbf{x}) = c + \mathbf{b}(\mathbf{x} - \mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0)^T \mathbf{A}(\mathbf{x} - \mathbf{x}_0).$$

For example, a minimum quadratic surface for $n = 2$ variables with the following coefficients:

$$c = 1 \quad \mathbf{b} = \begin{bmatrix} 0 & 0 \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

is shown in Figure 5.17 where a grid of coordinate pairs spanned by the individual ranges $-2 \leq x_i \leq 2$ was used to generate the scalar $G(\mathbf{x})$ values.

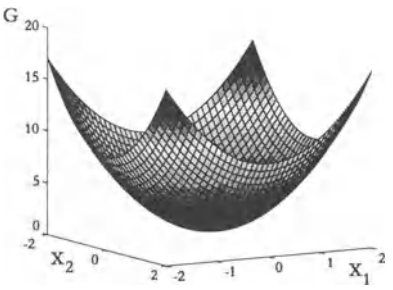

Figure 5.17. Example of a quadratic surface

5.4.2. Quadratic Approximation

The first step in a numerical optimization procedure is to develop a quadratic approximation. The general scalar, multivariable function $G(\boldsymbol{\xi})$ is expanded about a known point $\boldsymbol{\xi}_0 = [\xi_{1_0}, \ldots, \xi_{n_0}]$ by the following Taylor series (TS) expansion:

$$G(\boldsymbol{\xi}) = G(\boldsymbol{\xi}_0) + \mathbf{G}_{\boldsymbol{\xi}} \mid_{\boldsymbol{\xi}_0} (\boldsymbol{\xi} - \boldsymbol{\xi}_0) + \frac{[\boldsymbol{\xi} - \boldsymbol{\xi}_0]^T \mathbf{G}_{\boldsymbol{\xi}\boldsymbol{\xi}} \mid_{\boldsymbol{\xi}_0} (\boldsymbol{\xi} - \boldsymbol{\xi}_0)}{2!} + H.O.T..$$

Keeping terms up to second-order produces a quadratic approximation to $G(\boldsymbol{\xi})$ where the coefficients of the TS have the obvious relation to those in the quadratic surface expression (see Figure 5.18), or

$$G(\boldsymbol{\xi}) \approx c + \mathbf{b}\,(\boldsymbol{\xi} - \boldsymbol{\xi}_0) + (\boldsymbol{\xi} - \boldsymbol{\xi}_0)^T \mathbf{A}\,(\boldsymbol{\xi} - \boldsymbol{\xi}_0).$$

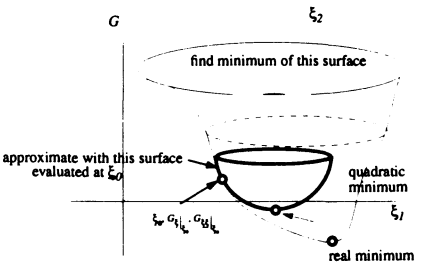

Figure 5.18. Quadratic approximation to $G(\boldsymbol{\xi})$

5.4.3. A Second-Order Iterative Method for Unconstrained Minimization

Now, it is time to develop a second-order iterative method to numerically solve unconstrained minimization problems.

Reiterating from Section 2.8.4., to provide a second-order *Newton-type* method to advance to the optimum, expand $G(\boldsymbol{\xi})$ about a current known point $\boldsymbol{\xi}_i$ in a TS as follows:

$$G(\boldsymbol{\xi}_{i+1}) = G(\boldsymbol{\xi}_i) + \mathbf{G}_{\boldsymbol{\xi}_i} [\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i] + \frac{1}{2!} [\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i]^T \mathbf{G}_{\boldsymbol{\xi}_i \boldsymbol{\xi}_i} [\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i] + H.O.T.$$

$$\Delta G = \mathbf{G}_{\boldsymbol{\xi}_i} [\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i] + \frac{1}{2!} [\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i]^T \mathbf{G}_{\boldsymbol{\xi}_i \boldsymbol{\xi}_i} [\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i] + H.O.T..$$

If one differentiates both sides with respect to the vector $\boldsymbol{\xi}_{i+1} - \boldsymbol{\xi}_i$ and sets the result to $\mathbf{0}$ (corresponding to an extermal) one gets the *Newton-Rhapson* search

direction, or

$$\mathbf{0} = \mathbf{G}_{\xi_i} + \mathbf{G}_{\xi_i \xi_i} \left[\xi_{i+1} - \xi_i \right].$$

Therefore,

$$\xi_{i+1} - \xi_i = -\mathbf{G}_{\xi_i \xi_i}^{-1} \mathbf{G}_{\xi_i}^T,$$

which can be fashioned into an update procedure as

$$\xi_{i+1} = \xi_i - \alpha \mathbf{G}_{\xi_i \xi_i}^{-1} \mathbf{G}_{\xi_i}^T$$

for a positive constant α that is determined by computing the step length[10]. Note, $\mathbf{G}_{\xi_i \xi_i}$ must be nonsingular to use this method. Checking $\mathbf{G}_{\xi_i \xi_i}$ for positive definiteness gives you assurance that you're moving on a *minimum surface* (concave up).

5.4.4. Recursive Quadratic Programming - a method to handle constraints explicitly

In Section 2.8.7., penalty functions were added to a scalar performance measure to handle equality constraints. However, neither the performance measure nor the equality constraints were handled explicitly, so this section provides an alternative method.

The goal is to minimize a scalar function $\phi(\xi)$ subject to a vector of equality constraints $\Psi(\xi) = [\Psi_1(\xi), \ldots, \Psi_m(\xi)] = \mathbf{0}$, but the numerical method minimizes the *quadratic* approximation to the scalar function (about known or updated points) in the hope of deriving an update scheme to march to the solution of the true problem. The quadratic approximation to the minimum is constrained via a *linear* approximation to the actual constraint $\Psi(\xi) = \mathbf{0}$ (about known points ξ_i) as follows:

$$\Delta\phi_{i+1} = \phi(\xi_{i+1}) - \phi(\xi_i) = \phi_\xi |_{\xi_i} \left[\xi_{i+1} - \xi_i \right] + \frac{\left[\xi_{i+1} - \xi_i \right]^T \mathbf{B} \left[\xi_{i+1} - \xi_i \right]}{2!}$$

subject to

$$\Psi(\xi_{i+1}) = \Psi(\xi_i) + \Psi_\xi |_{\xi_i} \left[\xi_{i+1} - \xi_i \right] = \mathbf{0}.$$

The matrix $\mathbf{B}$ is an approximation of $\phi_{\xi\xi}|_{\xi_i}$ and is computed and updated by the variable metric method[10], which requires only first derivatives. This method is used because, for the problems of interest in this book, the analytical expressions for $\phi_{\xi\xi}|_{\xi_i}$ are most likely not obtainable in analytic form, and second-order accurate finite difference approximations would require $2n$ function evaluations of $\phi(\xi)$ for n variables. This is possibly an unacceptable time expenditure if the function evaluation is lengthy.

Substituting for the search direction $\xi_{i+1} - \xi_i$ with $\mathbf{e_i}$ and augmenting the quadratic function approximation with the linearized constraint via constant Lagrange multipliers $\nu = [\nu_1, \ldots, \nu_m]$ yields

$$G(\mathbf{e}_i, \nu) = \phi_\xi \mid_{\xi_i} \mathbf{e}_i + \frac{\mathbf{e}_i^T \mathbf{B} \mathbf{e}_i}{2!} + \nu^T \left[\Psi(\xi_i) + \Psi_\xi \mid_{\xi_i} \mathbf{e}_i\right]$$

where ν is a vector of constant Lagrange multipliers.

The next step is to solve for $\mathbf{e}_i$, ν by using first variation conditions.

Note 5.1. The performance index uses derivatives of both ϕ and Ψ directly while the penalty function method adds *soft* constraints.

5.4.5. RQP Equality Constraints - solving for the unknowns

Taking the first variation gives

$$\begin{aligned}
\delta G(\mathbf{e}_i, \nu) &= \frac{\partial G}{\partial \mathbf{e}_i} \delta \mathbf{e}_i + \frac{\partial G}{\partial \nu} \delta \nu = 0 \\
\frac{\partial G}{\partial \mathbf{e}_i} &= \phi_\xi \mid_{\xi_i} + \mathbf{e}_i^T \mathbf{B} + \nu^T \Psi_\xi \mid_{\xi_i} = 0 \\
\frac{\partial G}{\partial \nu} &= \Psi(\xi_i) + \Psi_\xi \mid_{\xi_i} \mathbf{e}_i = 0
\end{aligned}$$

and performing matrix algebra with the knowledge that $\Psi_\xi \mid_{\xi_i}$ is not necessarily *square* yields the following solution:

$$\begin{aligned}
\nu(\xi_i) &= \left[\Psi_\xi \mathbf{B}^{-1} \Psi_\xi^T\right]^{-1} \left[\Psi(\xi_i) - \Psi_\xi \mathbf{B}^{-1} \phi_\xi^T\right] \\
\mathbf{e}_i(\xi_i) &= -\mathbf{B}^{-1} \left[\Psi_\xi^T \left[\Psi_\xi \mathbf{B}^{-1} \Psi_\xi^T\right]^{-1} \left[\Psi(\xi_i) - \Psi_\xi \mathbf{B}^{-1} \phi_\xi^T\right] + \phi_\xi^T\right].
\end{aligned}$$

Since $\mathbf{e}_i(\xi_i) = \xi_{i+1} - \xi_i$, one can fashion an update scheme $\xi_{i+1} = \xi_i + \alpha \mathbf{e}_i(\xi_i)$ for scalar $0 < \alpha < 1$. The largest value of α for which the scheme remains stable would be used.

Note 5.2. If the number of parameters ξ is less than the number of constraints Ψ the matrix $\left[\Psi_\xi \mathbf{B}^{-1} \Psi_\xi^T\right]$ will be singular. This leads to the general rule-of-thumb *number of parameters* $\geq$ *number of constraints +1.*

5.4.6. Formalized solution methods for implementing RQP

The previous section derives both the Lagrange multipliers and the parameter update vector for a RQP problem with equality constraints. To enable this solution, one still has to provide an approximation to the Hessian matrix $\mathbf{G}_{xx}$ denoted as $\mathbf{B}$ in the RQP update equations. There are library and vendor codes that provide complete solution environments for RQP. Two that will be discussed here are

the *VFO2AD* (*VF13AD* is currently available) FORTRAN routine from the Harwell library[11] and the *fmincon* routine from Mathwork's MATLAB® Optimization Toolbox[12]. Both of these methods draw heavily from Powell[13].

These general nonlinear optimization solvers address a problem of the following form to find the minimum of a constrained nonlinear multivariable function:

$$\min\ \phi(\boldsymbol{\xi}) \quad \text{subject to,} \quad \begin{aligned} \boldsymbol{\Psi}^{(1)}(\boldsymbol{\xi}) &= \mathbf{0} \quad \text{(equality contraints)} \\ \boldsymbol{\Psi}^{(2)}(\boldsymbol{\xi}) &\leq \mathbf{0}. \quad \text{(inequality contraints)} \end{aligned}$$

fmincon also allows linear constraint specifications as well as bounds specifications on $\boldsymbol{\xi}$ as follows:

$$\begin{aligned} A^{*}\boldsymbol{\xi} &\leq \mathbf{b} \\ \boldsymbol{\xi}_{\text{lower}} \leq \boldsymbol{\xi} &\leq \boldsymbol{\xi}_{\text{upper}}. \end{aligned}$$

As the computation progresses, the constraint equations will continue to generate *residuals*, either $\boldsymbol{\Psi}^{(1)}(\boldsymbol{\xi}) \neq \mathbf{0}$ for non-satisfaction of the equality constraints, or $\boldsymbol{\Psi}^{(2)}(\boldsymbol{\xi}) > \mathbf{0}$ in the case of the inequality constraints. The inequality constraints will need to be fashioned such that the code will continue to operate on them until they meet the desired forms above. For example, one may be integrating a set of state differential dynamic equations of motion $\dot{\mathbf{x}}(\boldsymbol{\xi}, t)$ to arrive at some final value that one wishes to constrain, say, $x_1(\boldsymbol{\xi}, t_f) \geq 10$. The inequality constraint will need to be expressed as $\Psi_1^{(2)} = 10 - x_1(\boldsymbol{\xi}, t_f)$. If $\Psi_1^{(2)}(\boldsymbol{\xi}) > 0$, the RQP algorithm will continue to operate on it until either it conforms to the problem statement above or computation ceases due to a user-specified *convergence* criteria for the approximate solution.

Note 5.3. Depending on what version of the algorithms one uses, the inequality signs on the problem statement constraints above may be reversed, in which case you would have to reverse the order of terms in the sample constraint statements.

The solution methods act in conjunction with the following computer code to form the computation environment:

1. A driver routine to do initialization, handle I/O, and call the optimization solver. Initialization provides a guess for $\boldsymbol{\xi}_0$ and specifies to what accuracy you wish to compute and to what tolerances you want the solver to achieve, as indicative of an approximate solution (i.e., $\phi(\boldsymbol{\xi}), \boldsymbol{\Psi}(\boldsymbol{\xi})$ changing by tiny amounts). I/O encompasses user-specified reads and writes.

2. Function routines to compute the scalar metric $\phi(\boldsymbol{\xi})$, the constraint residuals $\boldsymbol{\Psi}^{1}(\boldsymbol{\xi}) \neq \mathbf{0}$, and $\boldsymbol{\Psi}^{2}(\boldsymbol{\xi}) > \mathbf{0}$. This may entail sub-levels of functions to provide the numerical information to compute the residual violations, such as numerical integration routines (see Section 2.4.1.) to compute final state values.

3. Function routines to provide derivatives $\phi_{\xi}(\xi)$ (vector), and $[\Psi_{\xi}(\xi)]$(matrix). As mentioned in Section 2.8., this can be done in simple cases via analytical expressions, but for the bulk of the cases it will come from finite difference computations. An alternative is the ADIFOR and ADIC codes that provide analytical derivatives based on chain-rule differentiation[14]. Note, only first-order derivatives are needed for Hessian update methods.

In general, the code developed using *VFO2AD* will be more extensive, since one will need to develop all of the aforementioned routines. However, it will run faster than the MATLAB® equivalents since it will execute as a compiled entity. The routines developed for item 2 will be embedded in the finite difference derivative approximation routines for this method. On the other hand, the MATLAB® environment (by using *fmincon*, for example) provides a higher level language, an interpreted (versus compiled) environment, derivative approximations, and a standardized setup to integrate with a variety of optimization routines. These niceties are paid for with longer run times. Further details are included for both *VF02AD* and MATLAB® optimization in Appendices A and B, respectively.

5.4.7. Parameterized Controls - how to handle a changing final time

In Homework problem 5.4, #3 (single-link vertical slew), a parameterized control is developed for a fixed final time problem. Specific fixed times are chosen over the fixed final time period and corresponding parameter variations are assigned to these fixed times only. However, in many problems the final time varies. In the Homework problem 5.4, #1, the final time is a combination of two acceleration periods separated by a coast period. The acceleration and coast times are two of the three parameters computed via the optimization procedure.

In problems where final time is to be determined, the final time becomes an additional parameter and is passed to integration routines and interpolation routines (see Section 2.4.1.). If the controls are parameterized with respect to discrete times, then those times must expand or contract as does the current final time solution. If you do not allow for this, you would either have a huge gap between a newly computed final time from the optimization method and your next smallest time, or you would be overrunning the new time span with your *hard-fixed* discrete times. Typically, one can provide the adjustments to the discrete times by assigning them as fractions of the current final time. For example, $t_i = 0.0t_f^*,\ 0.1t_f^*,\ 0.2t_f^*,\ \ldots,\ 1.0t_f^*$ where parameter t_f^* is the latest value of the final time or some other variation that would allow appropriate expansion and contraction over a changing time span. t_f^* is passed to both the integration and control interpolation routines.

Note 5.4. A nonuniform spacing could be employed if knowledge of the problem dictates that *time granularity* of the control parameterization needs to vary at different points in the timeline.

5.4.8. Parameterized Controls - treating fixed bounds on the controls

Physical joint motors have bounded outputs, both positive and negative ($\tau_{lower} \leq \tau \leq \tau_{upper}$). Thus far, the problems discussed have not considered this. The open loop torque controls $u(t)$ were allowed to vary (via the computed parameters) to whatever is needed to minimize $\phi(\boldsymbol{\xi})$ and make the constraint violation residuals $\boldsymbol{\Psi}(\boldsymbol{\xi})$ small. How can the fact that there are physical bounds get communicated to the problem formulation? (As mentioned earlier, the MATLAB® Optimization Toolbox routines[12] do this for you automatically as a feature).

By using the typical constraint inequality formulation for the $\boldsymbol{\Psi}(\boldsymbol{\xi}) \leq \mathbf{0}$ convention, one can formulate the two-sided bounds as single-sided ones. For example, if the goal is to bound a single parameter ξ_i to vary its magnitude no greater than 10 (i.e., equi-sided bounds), the following constraint forms can be devised:

$$\Psi_1(\boldsymbol{\xi}) = \xi_i - 10$$
$$\Psi_2(\boldsymbol{\xi}) = -10 - \xi_i,$$

which would be needed for every parameter ξ_i that was bounded. Since it was stated earlier that the number of parameters must at least equal the number of constraints plus 1, this approach is not acceptable (if at the very least, from the burdensomeness of carrying all those constraints).

An alternative is to convert the two-sided inequality constraint to an equality constraint by minimizing an integral over the time span of the form

$$\Psi_i(\boldsymbol{\xi}) = \int_{t_0}^{t_f} \max(|u(t)| - 10, 0)dt$$

where $u(t)$ would be found by interpolation within the *table* of control parameters and their associated discrete times and *max* is a simple maximum comparison at every time that the integrand is evaluated. As this constraint is based on calculating

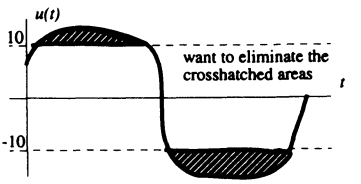

Figure 5.19. Integrated violation of the control constraint

an integral, an additional state (to those needed for the n-physical dynamics states) can be created via a differential equation $\dot{x}_{n+1} = \max(|u(t)| - 10, 0)$. $x_{n+1}(t_f)$ will

then be the constraint violation $\Psi_i(\boldsymbol{\xi}) > 0$. The optimization process will attempt to make the overshoot areas small ($x_{n+1}(t_f) \approx 0$) resulting in some *ripple* of the controls about the boundary $|u(t)| = 10$.

The previous approach creates another constraint to handle the two-sided bound. For most cases, this isn't an issue, but in the event that you're *short* on parameters with which to work, a second method embeds the bounds in the definition of the controls. Define a new control form to be $u(t) = u_{bound}\sin(\alpha(t))$. From this form, it is obvious that $u(t)$ cannot lie outside of the equi-sided bounds $\pm u_{bound}$. Instead of parameterizing $u(t)$ or torque directly one would parameterize $\alpha(t_i)$ at discrete times. Parameter units then switch from torque to angles. A variation on this theme allows unequal or (same sign) two-sided bounds with the form

$$u(t) = \frac{(u_{upper}+u_{lower})+(u_{upper}-u_{lower})\sin(\alpha(t))}{2}.$$

Drawbacks to this formulation are the parameters $\xi_i = \alpha(t_i)$ can take on large values since the angle values are unconstrained, and there may be some convergence issues because of the non-uniqueness of $\sin(\alpha(t))$ about $\alpha(t) = [2n + 1]\pi/2$ for $n = 0, 1, 2, \ldots$.

5.4.9. Examples with Parameterized Controls

In this section several examples that demonstrate how to solve parameterized controls optimization problems are presented. The first example covers the horizontal link introduced in Chapter 2 and shown in Figure 2.14. The second example considers a planar two-link rigid manipulator. The goal is to minimize a cost function subject to several constraints.

Example 5.1. Find the minimum effort control for a single-link, fixed-time, rest-to-rest maneuver for a given angle in the *horizontal* plane.

Solution

This problem is cast as a constrained nonlinear optimization problem. One wishes to find the control vector function $u(t)$ for $t_0 \leq t \leq t_f$ to minimize the following cost function:

$$J = \phi(\boldsymbol{\xi}) = \int_{t_0}^{t_f} u^2(\boldsymbol{\xi})dt$$

subject to the constraints

$$\Psi_1(\boldsymbol{\xi}) = \theta(t_f) - \theta_{desired} = 0$$
$$\Psi_2(\boldsymbol{\xi}) = \dot{\theta}(t_f) = 0,$$

with the initial conditions

$$\theta(t_0) = \dot{\theta}(t_0) = 0,$$

and the following plant dynamics:

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{Bmatrix} 0 \\ \frac{3}{ml^2} \end{Bmatrix} u$$

where $x_1 = \theta$, $x_2 = \dot{\theta}$, and $u = \tau$. In addition, the link length is l and it's mass is m. For this example, let $\theta_{desired} = \pi/2$, $m = 0.5$ kg, and $l = 1.0$ m. The fixed final time is 1.0 second. The problem is solved by using the MATLAB® Optimization Toolbox function *fmincon*. The control values are parameterized along the time axis $u(t_j) = \xi_j$, $j = 1, \ldots, n_{max}$. The optimizer is configured to minimize the control effort subject to satisfying the constraints. Further details on how to setup the problem and the corresponding MATLAB® code can be found in Appendix B.

The results are shown in the left plot of Figure 5.20. Note, the similarities to the analytical solution given in Chapter 2 (see Figure 2.16). This solution shows a linear torque profile with both the initial and final velocity being zero. The desired position is achieved.

Suppose that in addition to minimizing the control effort, bounds are placed on the control profile

$$u_{lower} \leq u \leq u_{upper}.$$

For this example, the bounds were set to ±1.2 N-m. With *fmincon* this problem is taken care of automatically by using built-in upper and lower bounds on each ξ_j (see Appendix B). The results are shown in the right plot of Figure 5.20. Now, the control torque is limited at the beginning and end of the trajectory. The velocity has increased during the middle of the maneuver such that the link can meet the desired end position while meeting all the original constraints.

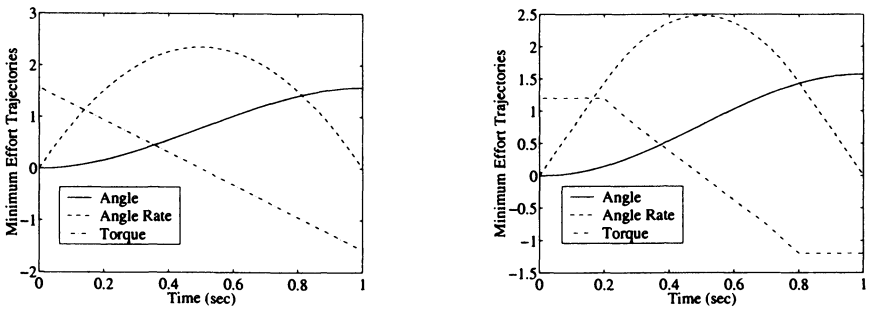

Figure 5.20. Minimum control effort for the horizontal slewing link of Example 5.1

In the final variation of the horizontal slewing link, utilize the final time is the cost function to minimize along with bounded controls. This is stated formally as follows: find the minimum time for a single-link, rest-to-rest maneuver to a given angle in the *horizontal* plane. In the optimization formulation, the final time is allowed to vary along with the bounded, discretized control parameters $u(t_j) = \xi_j,\ j = 1, \ldots, n,\ t_{final} = \xi_{n+1}$.

The results are shown in Figure 5.21. The minimum time parameter was computed as $t_f = 0.9368$ seconds. The numerical optimization results are similar to the analytical minimum time solution (see Figure 2.17). The exceptions are the control torque has to confine itself to linear interpolation among the discretely chosen times and the associated computed torque values while the analytical solution would be strictly bang-bang with a discountinous jump between upper and lower bounds at mid-trajectory. In addition, the analytic velocity comes to a sharp peak at mid-trajectory. The MATLAB® code is listed in Appendix B.

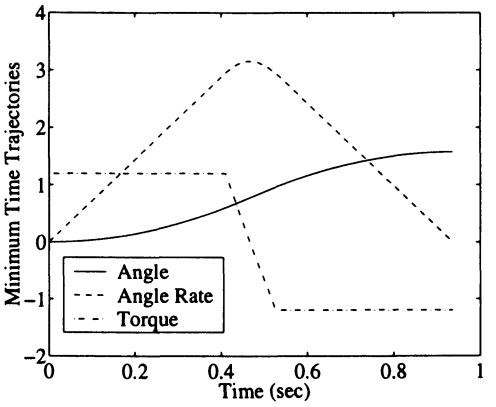

Figure 5.21: Minimum time optimization solution for the horizontal slewing link of Example 5.1

In the second example, a slewing two degree-of-freedom planar manipulator is examined.

Example 5.2. The goal is to find the minimum effort controls for a planar two-link arrangement executing a fixed-time, rest-to-rest maneuver between given angles in the *horizontal* plane.

Solution

This problem is cast as a constrained nonlinear optimization problem. One wishes to find the parameterized control vector $u_{1,2}(t)$, for $t_0 \leq t \leq t_f$ to minimize

the following cost function:

$$J = \phi(\boldsymbol{\xi}) = \int_{t_0}^{t_f} u_1^2(\boldsymbol{\xi})dt + \int_{t_0}^{t_f} u_2^2(\boldsymbol{\xi})dt$$

subject to the constraints

$$\begin{aligned}
\Psi_1(\boldsymbol{\xi}) &= \theta_1(t_f) - \theta_{1_{desired}} = 0 \\
\Psi_2(\boldsymbol{\xi}) &= \theta_2(t_f) - \theta_{2_{desired}} = 0 \\
\Psi_3(\boldsymbol{\xi}) &= \dot{\theta}_1(t_f) = 0 \\
\Psi_4(\boldsymbol{\xi}) &= \dot{\theta}_2(t_f) = 0,
\end{aligned}$$

with the initial conditions

$$\begin{aligned}
\theta_1(t_0) &= \theta_{10} \\
\theta_2(t_0) &= \theta_{20} \\
\dot{\theta}_1(t_0) &= \dot{\theta}_2(t_0) = 0,
\end{aligned}$$

and the following plant dynamics:

$$\begin{Bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \end{Bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\mathbf{H}^{-1}[\mathbf{C}_{damp} + \mathbf{C}_{coriolis}] \end{bmatrix} \begin{Bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{Bmatrix} + \begin{Bmatrix} \mathbf{0} \\ \mathbf{H}^{-1} \end{Bmatrix} \mathbf{u}$$

where $\mathbf{x}_1 = [\theta_1 \ \theta_2]^T$, $\mathbf{x}_2 = [\dot{\theta}_1 \ \dot{\theta}_2]^T$, and $\mathbf{u} = [\tau_1 \ \tau_2]^T$. The planar manipulator dynamic model matrices are given as

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} \\ h_{12} & h_{22} \end{bmatrix}$$

$$\mathbf{C}_{damp} = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix},$$

and

$$\mathbf{C}_{coriolis} = c_{12}^{(1)} \begin{bmatrix} 0 & -(\dot{\theta}_2 + 2\dot{\theta}_1) \\ \dot{\theta}_1 & 0 \end{bmatrix}$$

where the details of the coefficients h_{ij} and $c_{12}^{(1)}$ are given as the result of Homework problem 2.3, #2. In addition, the link lengths, masses, inertias, and other physical parameters used are those found in Homework problem 5.5, #4 (with the spring coefficients set to zero). The problem is solved by using the MATLAB® Optimization Toolbox function *fmincon*. The control values are parameterized for each degree-of-freedom motor input along the time axis $\tau_1(t_j) = \xi_{1j}$, $j = 1, \ldots, n_{max}$

and $\tau_2(t_j) = \xi_{2j}$, $j = 1, \ldots, m_{max}$, respectively. The optimizer is configured to minimize the control effort subject to meeting the constraints.

For the minimum effort unbounded controls problem, the initial link positions were set to $[\theta_{10}\ \theta_{20}] = [-20°\ 10°]$ and commanded to slew to the final link positions $[\theta_{1_{desired}}\ \theta_{2_{desired}}] = [-30°\ 20°]$ with a fixed final time of $t_f = 1.0$ second. The results from the optimization process are shown in Figure 5.22. The angles and rates are shown in the left plot while the required motor torques are shown in the right plot. The final positions are achieved while the angle rates meet the given constraint criteria (rest-to-rest). The bounds on the motor torques are set to $\tau_1 = \pm 5$ N-m and $\tau_2 = \pm 1.5$ N-m, respectively. For this problem, the motor torques stayed within the given bounds.

In the next variation, the problem is changed to a minimum effort with bounded controls problem. Again, the *fmincon* function automatically takes into account the upper and lower control bounds. By increasing the final positions for each link to $[\theta_{1_{desired}}\ \theta_{2_{desired}}] = [-40°\ 62.5°]$, more control effort is required, which saturates the motor torques (see right plot in Figure 5.23). The optimizer simulation results are shown in Figure 5.23. Note, even though bounds on the controls are in effect, the constraint conditions are still met (i.e., zero initial and final link velocities). The final variation of this example would be to solve a minimum time bounded control problem. This is left to the reader as a homework exercise (see Homework problem 5.5, #4).

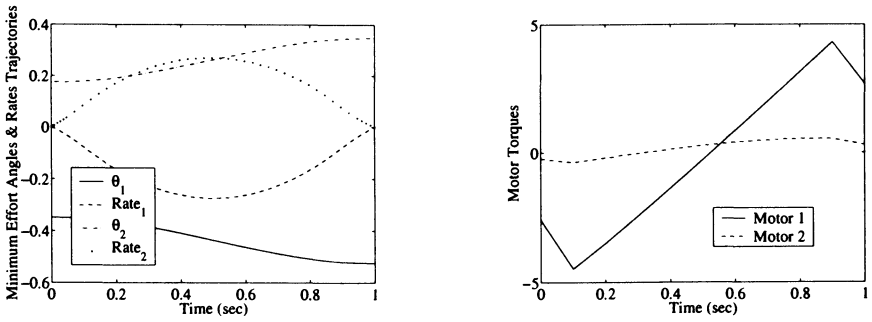

Figure 5.22. Minimum control effort for the planar manipulator of Example 5.2

Note, in both of these examples, which involve minimum effort control scenarios, the final torques are non-zero (see Figure 5.20, left plot and Figure 5.22, right plot, respectively). The non-zero torques occur because the final acceleration was not constrained. The interested reader is encouraged to add this constraint and rerun the optimization to generate a new solution. For the planar manipulator, the constraint would be stated as $\ddot{\theta}_{1,2}(t_f) = 0$.

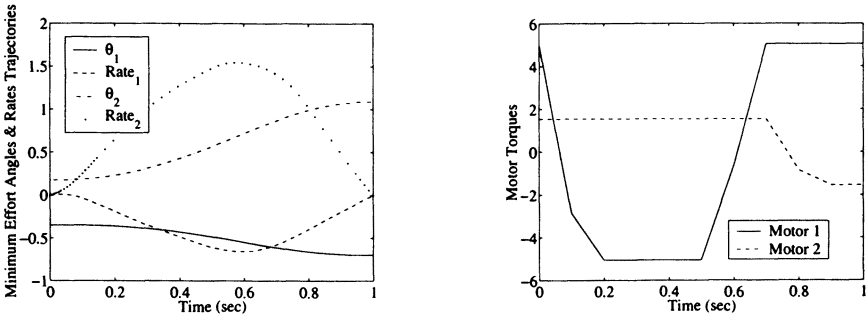

Figure 5.23: Minimum control effort with bounds for the planar manipulator of Example 5.2

5.4.10. Optimal Trajectories for Flexible Link Manipulators

This section is based on the work developed by Eisler, Robinett, Segalman, and Feddema[15]. Several performance indices were minimized to compare, in simulation, which resulting open-loop control provided the most satisfactory rest-to-rest, minimum-vibration motion. The ensuing discussion describes the structural dynamics model of the manipulator, the optimal control problem and parameterization of the controls, and the numerical simulation results.

The manipulator structure modeled in this study is the Sandia two-link flexible manipulator (see Figure 5.24), which has a two-link, cantilever arrangement constrained to slew in the horizontal plane. Tall, thin links are used to minimize vertical plane droop. The hub or joint-1 actuator, slews both links, an elbow motor, and a tip payload. The elbow or joint-2, actuator is located at the end of link-1 and slews the second link and the tip payload. The complete manipulator is about 0.5 meters (m) tall, and 1.2 m long.

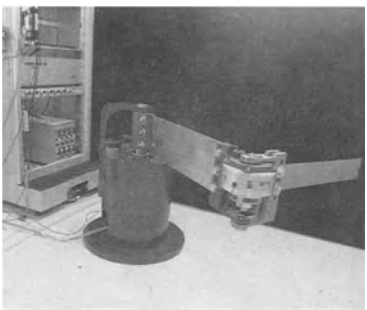

Figure 5.24: Sandia flexible two-link manipulator

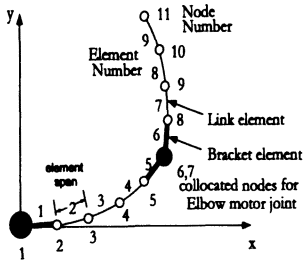

Figure 5.25: Finite element description

The dynamic equations of motion are developed by using an efficient finite element scheme for a multi-link, multi-joint system with bending only in the horizontal-plane. Following the techniques outlined in Section 3.4., the general manipulator structural dynamics model, which is based on the finite element arc length, are expressed as

$$[\mathbf{M}(\boldsymbol{\theta})]\ddot{\boldsymbol{\theta}} + [\mathbf{C}(\boldsymbol{\theta})]\dot{\boldsymbol{\theta}}^2 + [\mathbf{K}(\boldsymbol{\theta})]\mathbf{I}_1 = \boldsymbol{\tau}. \tag{5.12}$$

The Sandia manipulator brackets and links are constructed of aluminum. The brackets were modeled with 1 element and considered rigid ($EI = 10^5$), and links were modeled with 3 elements for a total of 9 elements (see Figure 5.25). The elbow joint is modeled by collocating two nodes. For the brackets, point moments of inertia were used to define mass distribution. No payload was used in this comparison. The joints were assumed to have no compliance (which means that the motors do not have any dynamics or they have a unity transfer function). Joint viscous damping was estimated from modal testing to be 0.5 and 0.05 newton-meter/(rad/sec), respectively for the hub and elbow[16]. Link structural damping was considered negligible. The finite element structural model parameters are given in Table 5.1.

Table 5.1. Finite element structural model parameters

Part	Length m	Mass kg	EI $N-m^2$	FE elem
hub bracket	0.635	0.545	10^5	1
link-1	0.504	0.640	10^2	3
elbow bracket/motor	0.107	5.415	10^5	1
2nd elbow bracket	0.104	0.830	10^5	1
link-2	0.489	0.313	2	3

The method of RQP coupled with a homotopy method was used to generate approximate minimum-time and minimum tracking-error tip trajectories for the two-link flexible manipulator movements in the horizontal plane. Constraints on these trajectories are completing a rest-to-rest maneuver, tracking a specified path $(x(t), y(t))_{tip}$, slewing between specified endpoints,$[(x(t_o), y(t_o)),\ (x(t_f), y(t_f))]_{tip}$, and not exceeding actuator torque limits $[\tau_1, \tau_2]_{max}$. Driving a flexible structure to rest at the final time t_f necessitates end constraints on both velocities and accelerations of the joints, and the final torque values. The chosen path is a straight line and the actuator torque limits are constants. Torque limits can be integrated naturally into the controls as discussed in Section 5.4.8. via

$$\tau_{1,2}(t) = |\tau_{1,2_{max}}| \sin \alpha_{1,2}(t)$$

where $\alpha_{1,2}(t)$ is a free variable. This form assumes that the two-sided limits on $\tau_{1,2}(t)$ are of the same magnitude. Unequal magnitude limits can be applied with the form $\tau(t) = 0.5[(\tau_{max} + \tau_{min}) + (\tau_{max} - \tau_{min}) \sin \alpha(t)]$. Assuming the configuration initially starts at rest, restate the optimization problems as minimizing final time or tracking error as

$$J = t_f \quad \text{or} \quad \int_{t_0}^{t_f} [y_{tip}(x_{tip}(t)) - y_{line}(x_{tip}(t))]^2 \, dt,$$

subject to

1. The finite element model (see Eq. 5.12)
2. The input actuator torques $\tau_{1,2}(t)$
3. The known initial conditions, and
4. Constrained by

$$\Psi(t_f) = \begin{bmatrix} x_{\text{tip}}(t_f) - x_{\text{specified}} \\ y_{\text{tip}}(t_f) - y_{\text{specified}} \\ \dot{\theta}_{1,2}(t_f) \\ \ddot{\theta}_{1,2}(t_f) \\ t_f - t_{\text{desired}} \\ \text{or}^* \\ \int_{t_0}^{t_f} [y_{\text{tip}}(x_{\text{tip}}(t)) - y_{\text{line}}(x_{\text{tip}}(t))] dt \\ \tau_{1,2}(t_f) \end{bmatrix} = \mathbf{0} \tag{5.13}$$

where $y_{tip}(x_{tip}(t))$ is dependent upon $x_{tip}(t)$ at time t because the second-link tip is tracking a straight line. (Note, $*$ = depending on the choice of J)

The tip-tracking criteria includes an integral constraint for following the line and a *point* constraint for acquiring the end condition while the constraints needed to bring the structure to rest are simply point constraints. These items affect the allowable behaviors of the structure during slewing maneuvers in characteristic fashions.

No constraint was placed on link vibration during slew maneuvers and the structure is allowed to *ring* during the trajectories. The integral constraint on tracking can allow pronounced, but brief excursions from tracking the path. The final torque values $\tau_{1,2}(t_f)$ could also have been assigned to zero and removed as formal constraints. Next, a discussion of parameterization and initialization of the controls is presented.

To approximate optimum system performance (i.e., solve the suboptimal problem) from the aforementioned structural model (see Eq. 5.12), a suitable parameterization of the controls $\tau_{1,2}(t)$ via $\alpha_{1,2}$ is necessary. For this study, the simplest case, using tabular values for $\alpha_{1,2}$, was chosen. These values were specified for both joint torques at nondimensional-time *node* points $\xi_j = t_j/t_f$ as

$$\alpha_1(\xi_j), \ \alpha_2(\xi_j), \quad j = 1, \ldots n_p \ \ 0 \leq \xi_j \leq 1$$

where t_j is the actual time corresponding to the j^{th} time-node point, n_p is the number of points for each motor, and t_f is the current value of the final time. This formulation is a variant of the ideas presented in Section 5.4.7. and also allows the torque histories to *stretch* naturally over the current trajectory length as t_f is varying continuously (i.e., the minimum time case) until convergence is achieved. Linear interpolation is used to compute $\alpha_{1,2}(\xi)$ between the time-node values and was found to be superior to spline interpolation from the standpoint of achieving high-frequency content in the torque results. This parameterization requires t_f to be a parameter and results in $2n_p + 1$ parameters to be optimized. For the minimum-tracking error case, normalization of the time could have been removed since trajectory integration occurs over a fixed time.

Numerical derivatives for the performance indices and the constraints $\Psi(t_f)$ provided to VF02AD are central finite-difference approximations. In computing these approximations, complete trajectories (or integrations of Eq. 5.12) are computed by using the current (or fixed) nominal t_f to produce perturbed J and $\Psi(t_f)$ values. As parameter optimization solutions can be very initial-condition sensitive, it was decided to use a *continuation* (homotopy) approach to acquire the solution[17,18]. Performance optimization for a complex model $\mathbf{f}_c(\mathbf{x}, \boldsymbol{\alpha}, t)$ can be accomplished by using the solution of a simpler model $\mathbf{f}_s(\mathbf{x}, \boldsymbol{\alpha}, t)$ and parameterizing the dynamics $\dot{\mathbf{x}}$ with a single parameter β such that the new dynamics model is

$$\dot{\mathbf{x}} = (1 - \beta)\mathbf{f}_s(\mathbf{x}, \boldsymbol{\alpha}, t) + \beta \mathbf{f}_c(\mathbf{x}, \boldsymbol{\alpha}, t).$$

At $\beta = 1$, the model of interest is realized. In this study, $\mathbf{f}_s$ are the dynamics of a two-link rigid manipulator while for the actual problem, $\mathbf{f}_c$ is the two-link flexible manipulator. The approximate optimal solution for the rigid manipulator is used to initialize the continuation process for the flexible manipulator. Since the solutions of the two problems are very similar, it only took one step ($\beta = 0 \rightarrow 1$). Results of the optimization study are discussed next.

Trajectory comparisons demonstrate the impact of torque input smoothness on structural mode excitation. Applied torques retain much of the qualitative character of rigid-body slewing motion with alterations for energy dissipation. The $\alpha_{1,2}(\xi)$ histories for each joint were composed of 27 tabular values where $\Delta\xi$ was varied from .025 in the midpoint and end-time regions to .05 elsewhere. Torque bounds were chosen as $\pm 6, \pm 1.5$ (n-m) for the hub (joint 1) and elbow (joint 2), respectively. The path tracked for this study was a straight line.

The four trajectory simulations shown in this work are differentiated by the final time requirement and include (with plot legend names)

1. A minimum time/minimum-integral-tracking error trajectory for a rigid link structure of identical mass properties and joint characteristics to those described previously. (*min time-rigid*)

2. A minimum time/minimum-integral-tracking-error trajectory for the flexible link system. (*min time-flexible*)

3. A fixed final time (t_f=2.5s)/minimum-integral-tracking-error trajectory for the flexible link system. (*t_f=2.5/min track error*)

4. A fixed final time (t_f=3.0s)/minimum-integral-tracking-error trajectory for the flexible link system. (*t_f=3.0/min track error*)

All position, velocity, acceleration, and torque constraints at t_f as stated in Eq. (5.13) are the same.

The results of the simulations are presented in Figures 5.26–5.28. The *min time* trajectories finished in essentially identical times of just over 2 seconds.

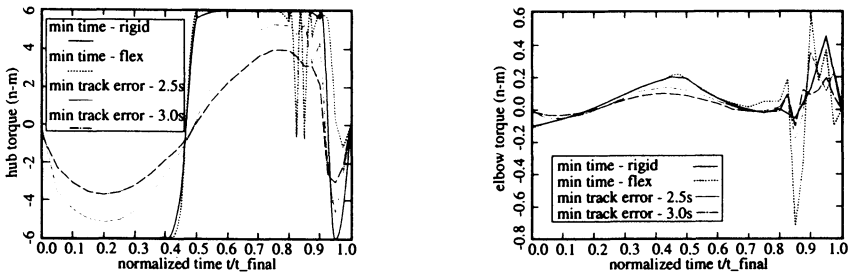

Figure 5.26. Hub and elbow joint torque histories

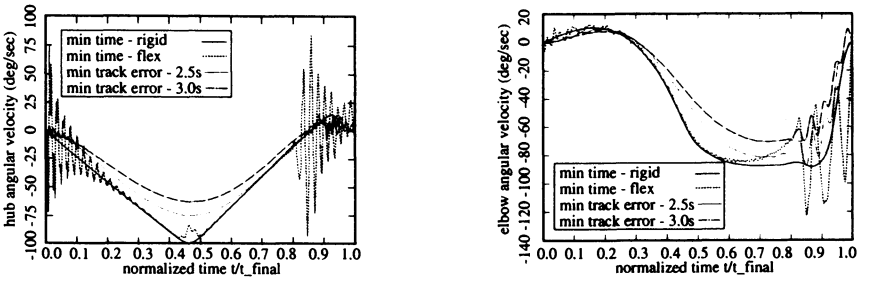

Figure 5.27. Hub and elbow joint velocity histories

From the torque profiles (see Figure. 5.26), it is obvious that the single homotopy step is sufficient to obtain a minimum-time flexible-link solution. The general characteristics of the *min time* trajectories (i.e., switching times, torque magnitudes, etc.) are almost identical. In particular, the approximate minimum-time flexible manipulator is identical to the rigid one up to the first 80% of the trajectory. Note, the *min time* curves for the hub torque initially jump to the boundary while the *min track error* curves, with the relaxed time constraints, start close to zero.

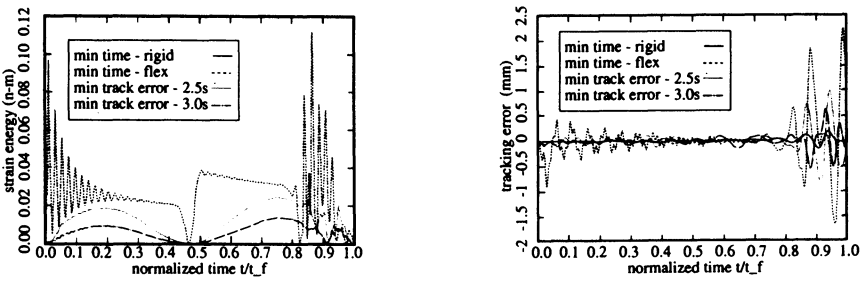

Figure 5.28. Strain energy and tracking error histories

The final 20% of this trajectory is unique because it demonstrates the combined optimization of the hub/elbow torque-switching necessary for a rigid two-link system to complete the minimum-time maneuver with the need to arrest the flexible body dynamics. Note, for rigid-link systems, additional switches appear in regions of nearly singular control where the stronger hub motor needs to wait for the elbow to *catch up*[19]. The fact that the additional switches after $t/t_f = 0.8$ are not fully bang-bang for the minimum-time case would suggest energy dissipation considerations for the flexible-link system. The *min track error* trajectories also show this behavior, but only to a limited extent since the t_f constraint is not as severe. (The finer nodal grid $\Delta\xi = .025$ was necessary to get sufficient definition of these additional switches.) Note, the final switch for the hub torque (τ_1) is nearly aligned in dimensionless time for all trajectories, but only truly bang-bang for the rigid system.

Although the *min time* flexible-link torque profiles are interesting in a comparative sense, they are probably not conducive to implementation on real actuators. A Fourier analysis of the torque data in Figure. 5.26 showed that power was concentrated in the 0.5-20 Hz range and the upper end of which may be difficult to achieve. The ringing of the structure is particularly evident in the joint velocities (see Figure. 5.27), and the strain energy (see Figure 5.28). As an alternative, one could use near minimum-time slewing to smooth the torque profiles[20]. This is essentially another approach to the fixed final-time/minimum tracking error problem.

The second-link tip trajectory tracking error $(y_{tip}(x_{tip}) - y_{line}(x_{tip}))$ in millimeters is presented in Figure. 5.28. Note, due to the suboptimal formulation proposed here none of the trajectories, including the rigid manipulator, produces a continuously zero error. However, because the model has reasonable fidelity, the tracking performance is sufficient for minimizing feedback control authority, which is a primary goal in open-loop trajectory design. The *min track* error trajectories show essentially zero error up to $t/t_f = 0.85$, thus demonstrating that the tracking error resides primarily in *unfolding* the links to complete the maneuver while accommodating the rest state. The use of the finer $\Delta\xi$ grid near the end not only resulted

in the additional torque switches being more clearly defined, but also reduced the maximum path deviation (after $t/t_f = 0.9$) by an order of magnitude. It is clear that additional parameterization of the torque profiles can produce better results.

Since this is an open loop technique for generating control histories, it is expected that a certain lack of robustness will be present from model parameter and initial condition differences in the translation to hardware. In Chapter 6, these issues are addressed for the Sandia flexible two-link manipulator.

In conclusion, employing parameter optimization successfully generated actuator torque histories for approximate, minimum-time and minimum tracking-error tip slewing maneuvers containing continuous and point control and trajectory constraints for a two -link flexible manipulator. It was demonstrated that final velocity, acceleration, and torque constraints do not preclude vibrations during the slew and that true minimum-time motions will have to endure them. Overall, the minimum-tracking error trajectories were better behaved.

5.4.11. Open Loop Input Shaping for a Slewing Flexible Rod

In this section, open loop input shaping control techniques are investigated for slewing long flexible boom links[21]. The robot configuration includes single links with low first natural frequencies ($\sim$ 1 Hz) typical for heavy hydraulic lift operations. The goals of the open loop control are as follows:

1. To develop simplified input acceleration profiles to slew, in the vertical plane, a 1 Hz rod in a symmetric rest-to-rest maneuver.

2. In the trajectory design, to properly account for the hydraulic servo dynamics.

For this particular problem, general steps are discussed to achieve open loop control design by using RQP optimization.

The research hardware configuration consists of a thin aluminum rod with a tip mass driven by a hydraulically-powered universal joint, a hydraulic pump, a motor, and drive circuitry (see Figure 5.29). The system geometry is also defined in Figure 5.30. The rigid link is cylindrical with a rigid mounting bracket located a distance d above the hydraulic joint. The flexible aluminum rod has length L, radius R, mass density ρ_0, and modulus E. The flexible rod is attached to the mounting bracket a distance r from the center of the cylindrical rigid link. In order to measure in-plane and out-of-plane tip acceleration, the tip mass m_T fixed to the end of the flexible rod is instrumented with accelerometers. The physical parameters are listed in Table 5.2.

The rotating rod's equations of motion are derived by using the method of quadratic modes, which was introduced in Chapter 3. For the maneuvers, all flexible body motion $u = u(\phi(x), q(t))$ occurs in-the-plane. Therefore, the out-of-plane flexible body equations of motion are not required. Following the procedure outlined in Chapter 3, the equations of motion are of the form

$$\mathbf{M}\ddot{\mathbf{q}} + \left[\mathbf{K} + \dot{\theta}^2\left[2\mathbf{N} - \mathbf{M}\right] - 2\mathbf{G}_2 \sin\theta\right]\mathbf{q} - \mathbf{G}_1 \cos\theta = -\mathbf{C}\ddot{\theta} \qquad (5.14)$$

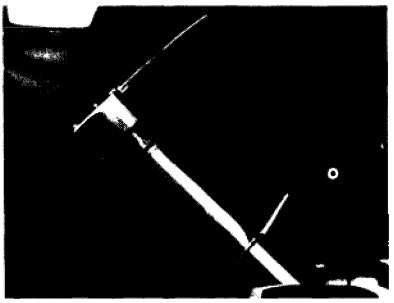

Figure 5.29: Hydraulic robot hardware

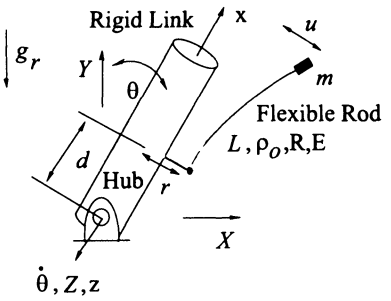

Figure 5.30: Relevant physical parameters

Table 5.2. Rod physical parameters

Parameter	Unit	Value
r	m	0.2286
d	m	0.6096
m_T	kg	0.9070
L	m	0.7112
ρ_o	kg/m^3	2768.0
R	mm	3.175
E	N/m^2	6.895x10^{10}

where the matrices definitions can be determined by closely examining each term in Eq. (3.49) and is left as an exercise for the reader.

For the hydraulic-drive robot, the analytical input shaping scheme discussed in Section 5.2. of this chapter is modified. It is desirable to derive an open loop control input based on the acceleration profile shown in Figure 5.31. For this open loop control to work, it is required that the actuator dynamics are included. A PD servo controller is used. Although nonlinear, the input/output relationship of the servo actuator dynamics are approximated (for nominal maneuvers) by using a third-order transfer function of the form

$$\frac{\theta_a}{\theta_c} = \frac{n_0}{s^3 + d_2 s^2 + d_1 s + d_0} \tag{5.15}$$

where θ_a is the actual joint angle and θ_c is the commanded joint angle. As will be explained later, the coefficients of the transfer function, n_0, d_0, d_1, and d_2, depend on the type of commanded maneuver. Conceptually, one could *reverse filter* the desired pulse output θ_a through the transfer function (see Eq. 5.15). That is, solve for θ_c to get the desired pulse acceleration $\ddot{\theta}$ corresponding to θ_a. However, for a pulse waveform, this would result in impulses. For implementation on the real

hardware, this is not desirable. Therefore, a modified control input based on the verisine function is introduced

$$\ddot{\theta} \sim A\,[1 - \cos\Omega[t - t^*]]$$

where A is the amplitude, Ω is the natural frequency, t is time, and t^* locates the first half-cycle and is shown in Figure 5.32. The verisine acceleration profile allows a reverse-filtering result that is achievable by a physical actuator. The methodology is described next.

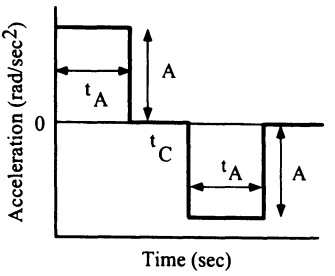

Figure 5.31: Conventional acceleration profiles

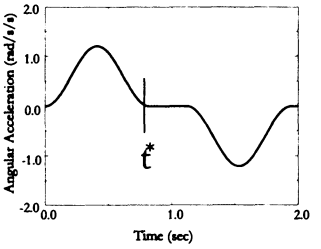

Figure 5.32: Smoothed acceleration profiles

The general steps required to establish an open-loop control trajectory for the slewing flexible rod with an hydraulic actuator are as follows:

1. Calibrate the actuator dynamics transfer function by using optimization as in Section 4.5.. Minimize the following integrated square error:

$$\int_{\theta_0}^{\theta_f} [\theta(t)_{measured} - \theta(t)_{modeled}]^2\, dt,$$

as a function of the parameters n_0, d_0, d_1, and d_2 in Eq. (5.15). The angles are defined in Eq. (5.15) as $\theta_{modeled} = \theta_a$ and $\theta_{input} = \theta_c$, respectively. $\theta(t)_{measured}$ is the measured data from the angle encoders. The measured data is initially collected by exciting the system with step inputs. Once the cost function is minimized, the transfer function model is fitted to the measured data.

2. Introduce a cost function that minimizes residual oscillation (post maneuver) by optimizing the versine acceleration parameters A, Ω, and t_c, to form, according to the quadratic modes dynamics model, a symmetric $\pm\theta_{desired}$ rest-to-rest maneuver. The final time t_f is fixed and is one cycle. The following constrained nonlinear optimization problem is suggested:

Minimize

$$J = \phi(\xi) = \int_{t_C+2t_A}^{t_f+t_C+2t_A} [u(x_{tip}, t)]^2\, dt,$$

subject to the constraints

$$\Psi_1 = \theta(t_f) - \theta_{desired_f} = 0$$
$$\Psi_2 = \dot{\theta}(t_f) = 0$$
$$\Psi_3 = \ddot{\theta}(t_f) = 0,$$

with the initial conditions

$$\theta(t_0) = \theta_{desired_0}$$
$$\dot{\theta}(t_0) = 0,$$

and the dynamic model given by Eq. (5.14). Since $\ddot{\theta}$ is prescribed, then $\dot{\theta}$ and θ are known. Only the flexible motion degrees-of-freedom $q_i(t)$ are treated as variables. By using the optimized parameters, a smoothed acceleration profile that minimizes residual oscillations is achieved.

3. By integrating $\ddot{\theta}(t)$ computed in Step 2 twice, the angle $\theta(t)_{modeled}$ coming out of the hydraulic joint is determined. The angle $\theta(t)_{input}$ going into the joint is determined by reverse filtering through the transfer function computed in Step 1. Reverse filtering is accomplished by transforming the s-domain Laplace transform result from Step 1 into a discrete or digital form, and using past and present values of $\theta(t)_{modeled}$ (determined from Step 2) to get the necessary $\theta(t)_{input}$ history).

Next, the symmetric maneuver for $\pm\theta_{desired} = \pm 16°$ is examined. By utilizing the identified actuator dynamics, the inputs to the hydraulic joint servo actuator are obtained via computer simulation and reverse filtering. For the flexible rod, the experimental results of this input history exhibited small residual oscillations. The acceleration profile parameters were adjusted to give better residual vibration performance. The adjustments were made to accommodate the unsimulated influence of the accelerometer cabling, use of an average tip mass center of gravity, and the use of a single clamped-free mode shape for modeling the flexible body behavior. It is time to step through the specifics of generating the $\pm\theta_{desired} = \pm 16°$ maneuver.

The computer simulation developed to integrate the equations of motion was parameterized by A, t_c, and t_A. A RQP optimization code was *wrapped around* this simulation for the purpose of choosing the acceleration profile parameters that will minimize the rod vibration at the end of the maneuver. In theory, the resulting acceleration profile can be reverse filtered by using the servo actuator dynamics of Eq. (5.15) to obtain the input needed to drive the hydraulic actuator. However, as stated earlier, the third-order form of the actuator dynamics results in impulses if a pulse acceleration profile is used. So that the necessary derivatives of the acceleration profile are smooth and to avoid this implementation problem, the pulses of the acceleration profile are replaced with $(1 - \cos\Omega t)$ terms. In order to achieve the maneuver for this type of acceleration profile, the RQP simulation now uses

the amplitude A, the frequency Ω, and the coast time t_C as control variables. The optimized acceleration profile for the $\pm 16°$ maneuver is shown in Figure 5.32 and the corresponding optimal acceleration profile parameters are given in Table 5.3. As mentioned earlier, for a given trajectory, the linearized representation of the

Table 5.3. Optimized acceleration profile parameters

Parameter	Unit	Theoretical Value	Experimental Value
A	rad/s^2	0.5486	0.6038
Ω	rad/s	7.278	7.640
t_C	sec	0.3161	0.3000
Resid. Osc.	cm	8.0	< 1.0

nonlinear actuator servo dynamics must be tuned (or linearized). This tuning is performed again by using an RQP technique. Once the desired trajectory is obtained by reverse filtering through the nominal actuator dynamics, the hardware is run with this trajectory and the encoder data is obtained that describes the actual joint angle time history. Since this is typically not the case, the RQP code is used to minimize the integrated squared error between the experimental encoder output and the actuator dynamics filtered command input. The control variables for this optimization problem are the actuator transfer function coefficients n_0, d_0, d_1, and d_2. The optimized actuator transfer function coefficients are shown in Table 5.4. The curves in Figure 5.33 show the results of tuning the actuator model to match the experimental data for the $\pm 16°$ maneuver. The predicted joint angle (dashed line) corresponds well with the experimental data (solid line). The reverse filtered input command is also shown (dotted line). A joint servo input is created based on

Table 5.4. Optimized filter coefficients

Filter Coefficients	Value
n_0	4373.7
d_1	3975.1
d_2	1213.9
d_3	84.2

the optimal parameters of Table 5.3 and the optimal filter coefficients of Table 5.4. Maneuvering the robot by using these commands results in residual vibratory oscillations with a peak-to-peak amplitude of 8 cm. The acceleration profile parameters are adjusted to achieve residual oscillations with a peak-to-peak amplitude of less than 1 cm. The filtered in-plane tip mass accelerometer results are shown in Figure 5.34.

Procedures for calculating input command histories to perform residual oscillation free symmetric maneuvers of a flexible rod by using a hydraulically actuated heavy-lift robot were developed and experimentally verified. The main features of these procedures are the use of a smoothed joint acceleration profile, an optimization calculation for obtaining the acceleration profile parameters, and an optimal hydraulic joint actuator System ID calculation.

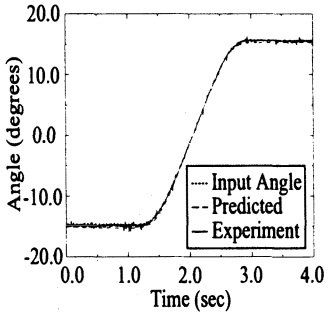

Figure 5.33: Tuned actuator model system identification results

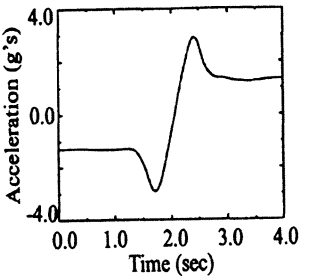

Figure 5.34: Filtered in-plane tip mass accelerometer 1 cm peak-to-peak results

5.5. Dynamic Programming

Dynamic programming is a powerful mathematical tool used to solve problems in a variety of different areas. One such area is optimal control. Input shaping problems in robotics can often be expressed as an optimal control problem. Accordingly, dynamic programming is useful for input shaping. Details of a dynamic programming algorithm for input shaping are presented in this section for linear problems. Constraints are dealt with approximately by using a simple penalty approach. Dynamic programming algorithms for unconstrained nonlinear problems[22], nonlinear problems with equality constraints[23], and problems with a combination of equality and inequality constraints[24] are also available.

The contents of this section are as follows:

- Introduction to Dynamic Programming.
- Application of Dynamic Programming to a data smoothing problem.
- Application of Dynamic Programming to a class of discrete-time, linear, optimal control problems.
- Application of Dynamic Programming to optimal control in robotics.

5.5.1. The Principle of Optimality

Dynamic programming is founded on the principle of optimality[25–27]. An optimal sequence of decisions in a multistage decision process problem has the property that whatever the initial stage, state, and decision, the remaining decisions must constitute an optimal sequence of decisions for the remaining problem, with the stage and state resulting from the first decision considered as initial conditions.

As an example, consider a problem dealing with the discrete-time optimal control of a rigid-link robot (see Figure 5.35). In this example, stage refers to a discrete

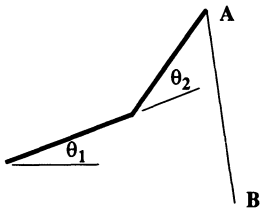

Figure 5.35. Discrete-time optimal control of a rigid link

point in time. For example, the i^{th} stage refers to time t_i, the $(i+1)^{th}$ stage to time t_{i+1}, etc. State refers to the configuration of the system at a particular point in time. In this example, the state at the i^{th} stage consists of the joint angles and joint angular rates at the time, t_i. Decisions are the actions taken at a given stage which transform the system from one state to another. In this example, the decision at the i^{th} stage consists of the torques applied to the robot's joints at time t_i.

5.5.2. Simple Application of Dynamic Programming

Consider the problem of traveling from point A to point B in Figure 5.36. Movement is only allowed from left to right and the cost of traveling from one point on the grid to another is given by the number at the edge connecting the two points. The goal is to find the path from A to B that minimizes the total cost.

The number by each point in the grid in Figure 5.37 is the cost of the lowest-cost path from that point to B. These numbers are obtained recursively by moving backwards from B to A and applying the principle of optimality. The arrows in the Figure 5.37 indicate the direction to be taken from each point to minimize the total cost of getting to B. The best path from A to B is seen to have a cost of 13. The path moves *uddduud* where u denotes up to the right and d denotes down to the right[28].

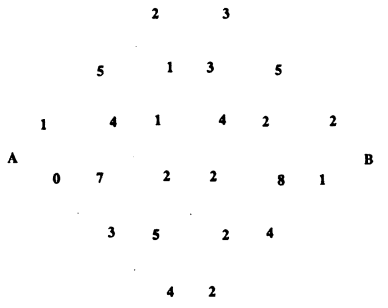

Figure 5.36: Simple example illustrating dynamic programming concepts

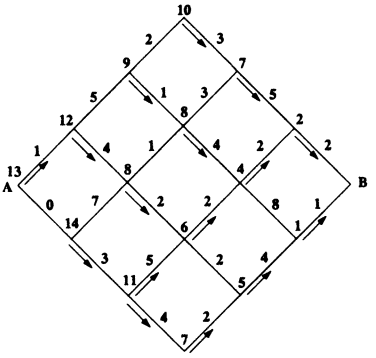

Figure 5.37: Dynamic programming solution to the problem in Figure 5.36

5.5.3. Application of Dynamic Programming to Data Smoothing

This section considers with the minimization of the objective function

$$J(x_k, u_k) = \frac{1}{2}\sum_{k=1}^{N}\left((x_k - d_k)^2 + \lambda u_k^2\right), \tag{5.16}$$

subject to

$$x_{k+1} = x_k + h_k u_k \qquad (k = 1, \ldots, N-1). \tag{5.17}$$

In the Eqs. (5.16) and (5.17), x_k is the value of the approximating function (a linear spline) evaluated at time t_k. d_k is the value of the data point at time t_k. $\lambda \geq 0$ is a smoothing parameter. u_k is the input at time t_k and $h_k = t_{k+1} - t_k$. The input u_k can be interpreted as the first derivative of the linear spline for $t_k \leq t < t_{k+1}$ (see Figure 5.38). The number of stages equals N. The state and decision at the k^{th} stage are given by x_k and u_k, respectively. Define the optimal value function as

$$f_i(x_i) = \min_{(u_i, \ldots, u_N)}\left[\frac{1}{2}\sum_{k=i}^{N}\left((x_k - d_k)^2 + \lambda u_k^2\right)\right]. \tag{5.18}$$

That is, $f_i(x_i)$ is the value of Eq. (5.18) for an optimal sequence of decisions (inputs) in terms of the *initial* state x_i. Here, the number of stages is equal to $N - i + 1$. The solution strategy will be to obtain $f_i(x_i)$ from $f_{i+1}(x_{i+1})$ by using dynamic programming. In other words, the solution to the $N - k + 1$ stage problem will be obtained from that for the $N - k$ stage problem. This is where the idea of

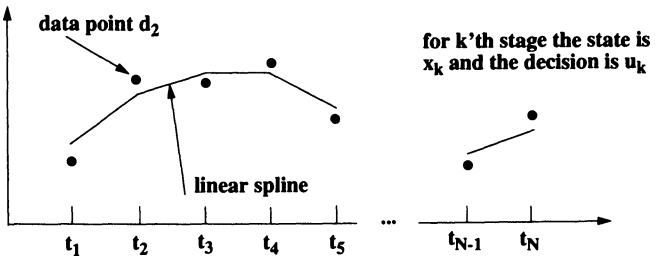

Figure 5.38. Linear spline curve fit

an imbedded solution is made manifest. In mathematical form, the *principle of optimality* states that

$$f_i(x_i) = \min_{u_i} \left[\frac{(x_i - d_i)^2 + \lambda u_i^2}{2} + f_{i+1}(x_i + h_i u_i) \right]. \tag{5.19}$$

One postulates that the optimal value function can be expressed as

$$f(x_i) = \zeta_i + v_i x_i + w_i \frac{x_i^2}{2}. \tag{5.20}$$

Substituting Eq. (5.20) into Eq. (5.19) yields

$$\zeta_i + v_i x_i + \frac{w_i x_i^2}{2} = \min_{u_i} \left[\frac{(x_i - d_i)^2 + \lambda u_i^2}{2} + \zeta_{i+1} + v_{i+1}(x_i + h_i u_i) + \frac{w_{i+1}(x_i + h_i u_i)^2}{2} \right]. \tag{5.21}$$

The value of u_i that minimizes Eq. (5.21) is readily obtained by differentiating the right hand side of Eq. (5.21) with respect to u_i and equating the result to zero. This leads to

$$u_i = \frac{-h_i(v_{i+1} + w_{i+1} x_i)}{\lambda + h_i^2 w_{i+1}}. \tag{5.22}$$

Substituting Eq. (5.22) into Eq. (5.21) and equating terms of equal degree in x_i yields the recursive equations

$$\zeta_i = \zeta_{i+1} + \frac{d_i^2}{2} - \frac{h_i^2 v_{i+1}^2}{2[\lambda + h_i^2 w_{i+1}]} \tag{5.23}$$

$$v_i = -d_i + \frac{\lambda v_{i+1}}{\lambda + h_i^2 w_{i+1}} \tag{5.24}$$

$$w_i = 1 + \frac{\lambda w_{i+1}}{\lambda + h_i^2 w_{i+1}}. \tag{5.25}$$

Given, the values for ζ_N, v_N, and w_N, one can recursively work backwards by using Eq. (5.23)–(5.25) to obtain

$$f_1(x_1) = \zeta_1 + v_1 x_1 + w_1 \frac{x_1^2}{2}. \tag{5.26}$$

Examination of Eq. (5.18) reveals, that Eq. (5.26) provides the minimum value of the objective function given by Eq. (5.16) in terms of the initial state x_i. Minimizing f_i with respect to the initial state yields

$$x_1 = \frac{-v_1}{w_1}. \tag{5.27}$$

This is, the value for the initial state that minimizes Eq. (5.16).

To determine ζ_N, v_N, and w_N, Eq. (5.18) and Eq. (5.20) are combined giving

$$\zeta_N + v_N x_N + w_N \frac{x_N^2}{2} = \min_{u_i} \left[\frac{(x_N - d_N)^2 + \lambda u_N^2}{2} \right]. \tag{5.28}$$

The right hand side of Eq. (5.28) is minimized for $u_N = 0$. Substituting $u_N = 0$ into Eq. (5.28) and equating terms of equal degree in x_N yields

$$\zeta_N = \frac{d_N^2}{2} \tag{5.29}$$

$$v_N = -d_N \tag{5.30}$$

$$w_N = 1. \tag{5.31}$$

Summary

- Calculate and store v_N and w_N by using Eq. (5.30) and Eq. (5.31).
- Calculate and store v_i and w_i for $i = N-1, \ldots, 1$ by using Eq. (5.24) and Eq. (5.25).
- Calculate the initial state x_1 by using Eq. (5.27).
- Calculate u_i and x_{i+1} for $i = 1, \ldots, N-1$ by using Eq. (5.22) and Eq. (5.17).

Example 5.3. Given, $N = 501$, $t_k = (k-1)/500$, $\lambda = 0.001$, and $d_k = \sin(2\pi t_k) + e_k$ where e_k is a uniformly distributed random variable between -0.2 and 0.2. What happens as λ goes to 0 and λ goes to ∞?

Solution

The results of this example are shown in Figure 5.39 (see Homework problem 5.8 for more details). The dynamic programming algorithm is applied to the noisy sinusoidal data shown along with the data smoothed results in Figure 5.39. As the parameter λ goes to 0, the smoothed data fit will approach the noisy sinusoidal data. λ going to ∞ will cause u_i to go to 0 (see Eq. 5.22), which results in a least squares fit to the data.

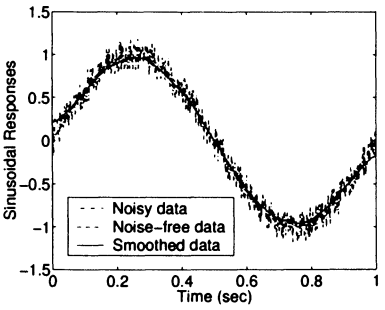

Figure 5.39: Dynamic programming for data smoothing application results for Example 5.3

5.5.4. Application of Dynamic Programming to Discrete-Time Optimal Control Problems

Consider the initial value problem

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t) \qquad \mathbf{x}(t_1) = \mathbf{x}_1$$

where $\mathbf{x} \in R^n$ is the state, $\mathbf{u} \in R^m$ is the input, and the dot denotes differentiation with respect to time t. The input is assumed to be discretized temporally as

$$\mathbf{u}(t) = \mathbf{u}_k \qquad t_k \leq t < t_{k+1}$$

for $k = 1, \ldots, N-1$. Provided the existence and uniqueness of the solution to the initial value problem, adjacent states in time can be related as

$$\mathbf{x}_{k+1} = g_k(\mathbf{x}_k, \mathbf{u}_k) \tag{5.32}$$

where $\mathbf{x}_k = \mathbf{x}(t_k)$.

The discrete-time optimal control problem is stated as follows: Given the initial state $\mathbf{x}_1$ find the inputs $\mathbf{u}_k$ that minimize the objective function

$$\Gamma(\mathbf{x}_k, \mathbf{u}_k) = \sum_{k=1}^{N} \Gamma_k(\mathbf{x}_k, \mathbf{u}_k), \tag{5.33}$$

subject to Eq. (5.32).

Consider a subset of discrete-time optimal control problems in which the dynamics are linear and the objective function is quadratic in the inputs and states. That is,

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k \tag{5.34}$$

$$\Gamma(\mathbf{x}_k, \mathbf{u}_k) = \sum_{k=1}^{N} \eta_k + \mathbf{x}_k^T \mathbf{y}_k + \mathbf{u}_k^T \mathbf{z}_k + \frac{\left[\mathbf{x}_k^T \mathbf{Q}_k \mathbf{x}_k + 2\mathbf{x}_k^T \mathbf{R}_k \mathbf{u}_k + \mathbf{u}_k^T \mathbf{S}_k \mathbf{u}_k\right]}{2} \tag{5.35}$$

As was done previously for the data smoothing problem, one defines the optimal value function as

$$\Lambda_i(\mathbf{x}_i) = \min_{(\mathbf{u}_i,\ldots,\mathbf{u}_N)} \sum_{k=1}^{N} \eta_k + \mathbf{x}_k^T\mathbf{y}_k + \mathbf{u}_k^T\mathbf{z}_k + \frac{\left[\mathbf{x}_k^T\mathbf{Q}_k\mathbf{x}_k + 2\mathbf{x}_k^T\mathbf{R}_k\mathbf{u}_k + \mathbf{u}_k^T\mathbf{S}_k\mathbf{u}_k\right]}{2}. \tag{5.36}$$

Application of the principle of optimality yields

$$\Lambda_i(\mathbf{x}_i) = \min_{\mathbf{u}_i}\left\{\eta_i + \mathbf{x}_i^T\mathbf{y}_i + \mathbf{u}_i^T\mathbf{z}_i + \frac{\left[\mathbf{x}_i^T\mathbf{Q}_i\mathbf{x}_i + 2\mathbf{x}_i^T\mathbf{R}_i\mathbf{u}_i + \mathbf{u}_i^T\mathbf{S}_i\mathbf{u}_i\right]}{2} + \Lambda_{i+1}[\mathbf{A}_i\mathbf{x}_i + \mathbf{B}_i\mathbf{u}_i]\right\}. \tag{5.37}$$

Consistent with the use of a quadratic objective function one expresses the optimal value function as

$$\Lambda_i(\mathbf{x}_i) = \zeta_i + \mathbf{x}_i^T\mathbf{v}_i + \frac{\mathbf{x}_i^T\mathbf{W}_i\mathbf{x}_i}{2}. \tag{5.38}$$

Substituting Eq. (5.38) into Eq. (5.37) one obtains

$$\zeta_i + \mathbf{x}_i^T\mathbf{v}_i + \frac{\mathbf{x}_i^T\mathbf{W}_i\mathbf{x}_i}{2} = \min_{\mathbf{u}_i}\left\{\eta_i + \zeta_{i+1} + \mathbf{x}_i^T\mathbf{h}_{4i} + \mathbf{u}_i^T\mathbf{h}_{5i} + \frac{\left[\mathbf{x}_i^T\mathbf{H}_{1i}\mathbf{x}_i + 2\mathbf{x}_i^T\mathbf{H}_{2i}\mathbf{u}_i + \mathbf{u}_i^T\mathbf{H}_{3i}\mathbf{u}_i\right]}{2}\right\} \tag{5.39}$$

where

$$\begin{aligned}
\mathbf{H}_{1i} &= \mathbf{Q}_i + \mathbf{A}_i^T\mathbf{W}_{i+1}\mathbf{A}_i \\
\mathbf{H}_{2i} &= \mathbf{R}_i + \mathbf{A}_i^T\mathbf{W}_{i+1}\mathbf{B}_i \\
\mathbf{H}_{3i} &= \mathbf{S}_i + \mathbf{B}_i^T\mathbf{W}_{i+1}\mathbf{B}_i \\
\mathbf{h}_{4i} &= \mathbf{y}_i + \mathbf{A}_i^T\mathbf{v}_{i+1} \\
\mathbf{h}_{5i} &= \mathbf{z}_i + \mathbf{B}_i^T\mathbf{v}_{i+1}.
\end{aligned}$$

Differentiating the right hand side of Eq. (5.39) with respect to $\mathbf{u}_i$ and setting the result equal to zero yields

$$\mathbf{u}_i = -\mathbf{H}_{3i}^{-1}\left[\mathbf{H}_{2i}^T\mathbf{x}_i + \mathbf{h}_{5i}\right]. \tag{5.40}$$

Finally, substitution of Eq. (5.40) for $\mathbf{u}_i$ into Eq. (5.39) and equating terms of equal degree in $\mathbf{x}_i$ results in the following recursive equations:

$$\zeta_i = \zeta_{i+1} + \eta_i - \frac{\mathbf{h}_{5i}^T \mathbf{H}_{3i}^{-1} \mathbf{h}_{5i}}{2}$$

$$\mathbf{v}_i = \mathbf{h}_{4i} - \mathbf{H}_{2i} \mathbf{H}_{3i}^{-1} \mathbf{h}_{5i} \tag{5.41}$$

$$\mathbf{W}_i = \mathbf{H}_{1i} - \mathbf{H}_{2i} \mathbf{H}_{3i}^{-1} \mathbf{H}_{2i}^T. \tag{5.42}$$

The *initial* values for the recursive equations above are given by

$$\zeta_N = \eta_N$$

$$\mathbf{v}_N = \mathbf{y}_N \tag{5.43}$$

$$\mathbf{W}_N = \mathbf{Q}_N. \tag{5.44}$$

If the initial state $\mathbf{x}_i$ is unspecified, minimization of Eq. (5.33) yields

$$\mathbf{x}_1 = -\mathbf{W}_1^{-1} \mathbf{v}_1. \tag{5.45}$$

Summary
The procedure for solving the discrete-time optimal control problem under consideration is summarized as follows:

1. Calculate $\mathbf{v}_N$ and $\mathbf{W}_N$ by using Eq. (5.43) and Eq. (5.44).
2. Calculate $\mathbf{v}_i$ and $\mathbf{W}_i$ recursively for $i = N - 1$ to $i = 1$ by using Eq. (5.41) and Eq. (5.42) and store the matrices $\mathbf{H}_{3i}^{-1}$, $\mathbf{H}_{2i}^T$, and $\mathbf{H}_{3i}^{-1} \mathbf{h}_{5i}$ in the process.
3. If the initial conditions are unspecified, calculate $\mathbf{x}_1$ by using Eq. (5.45).
4. Calculate $\mathbf{u}_i$ and $\mathbf{x}_{i+1}$ recursively for $i = 1$ to $i = N - 1$ by using Eq. (5.40) and Eq. (5.34), respectively.

The computational features of dynamic programming algorithm are as follow:

1. Order Nn^3 operations are required to solve a problem.
2. Order Nnm storage is required to solve a problem.
3. Constraints can be accommodated by using penalty functions.

5.5.5. Practical Issues

Issue 1: Determination of state transition matrices appearing in Eq. (5.34).

Consider a linear, time-invariant system written in first-order form as

$$\dot{\mathbf{x}} = \mathbf{E}\mathbf{x} + \mathbf{F}\mathbf{u}. \tag{5.46}$$

Recall, the solution to Eq. (5.46) can be written in terms of the matrix exponential[29] as

$$\mathbf{x}(t) \equiv \phi(t; t_0, \mathbf{x}_0, \mathbf{u}) = e^{\mathbf{E}(t-t_0)}\mathbf{x}_0 + \int_{t_o}^{t} e^{\mathbf{E}(t-\tau)}\mathbf{F}\mathbf{u}(\tau)d\tau$$

where $\mathbf{x}_0$ is the state at time t_0. Setting

$$\begin{aligned} t_0 &= t_k \\ t &= t_{k+1} \\ h &= t_{k+1} - t_k, \end{aligned}$$

one obtains

$$\mathbf{x}_{k+1} = e^{\mathbf{E}h}\mathbf{x}_k + \int_0^h e^{\mathbf{E}(h-\tau)}d\tau\mathbf{F}\mathbf{u}_k. \tag{5.47}$$

Comparison of Eq. (5.34) and Eq. (5.47) implies

$$\mathbf{A}_k = e^{\mathbf{E}h} \tag{5.48}$$

$$\mathbf{B}_k = \left[\int_0^h e^{\mathbf{E}(h-\tau)}d\tau\mathbf{F}\right]. \tag{5.49}$$

Example 5.4. Find the state transition matrices for a cubic spline

$$\mathbf{x}_k = \begin{bmatrix} f(t_k) & \dot{f}(t_k) & \ddot{f}(t_k) \end{bmatrix}^T \qquad u_k = \dddot{f}(t_k)$$

The matrices $\mathbf{E}$, and $\mathbf{F}$ for the continuous-time system are

$$\mathbf{E} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Solution

$$e^{\mathbf{E}t} = \begin{bmatrix} 1 & t & \frac{t^2}{2} \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix}$$

$$\left[\int_0^h e^{\mathbf{E}(h-\tau)}d\tau\mathbf{F}\right] = \begin{bmatrix} \frac{h^3}{6} \\ \frac{h^2}{2} \\ h \end{bmatrix}.$$

Thus,

$$\mathbf{A}_k = \begin{bmatrix} 1 & h & \frac{h^2}{2} \\ 0 & 1 & h \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{B}_k = \begin{bmatrix} \frac{h^3}{6} \\ \frac{h^2}{2} \\ h \end{bmatrix}.$$

Comments

- The MATLAB® function *c2d* of the Control System Toolbox can be used to convert state-space models from continuous-time to discrete-time for linear, time-invariant models. That is, *c2d* can be used to calculate the transition matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ (see Eqs. 5.48 and 5.49) based on the continuous-time state-space model (see Eq. 5.46).
- The transition matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ can also be calculated directly from closed-form solutions. This point is illustrated in Example 5.5.
- Calculation of the transition matrices for time-varying or nonlinear systems may require the use of a numerical integration scheme.

Example 5.5. Find the state transition matrices for a single-link rigid robot

$$\mathbf{x}_k = \begin{bmatrix} \theta(t_k) & \dot{\theta}(t_k) \end{bmatrix}^T \qquad u_k = M_k$$

The equation of motion for this system is given by

$$J\ddot{\theta} = M.$$

Solution

For $t_k \leq t < t_{k+1}$,

$$\dot{\theta}(t) = \dot{\theta}(t_k) + \frac{u_k(t - t_k)}{J} \tag{5.50}$$

$$\theta(t) = \theta(t_k) + \dot{\theta}(t_k)(t - t_k) + \frac{u_k}{2J}(t - t_k)^2. \tag{5.51}$$

Substituting $t_k = t_{k+1}$ into Eq. (5.50) and Eq. (5.51), gives

$$\mathbf{A}_k = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \tag{5.52}$$

$$\mathbf{B}_k = \frac{1}{J}\begin{bmatrix} \frac{h^2}{2} \\ h \end{bmatrix} \tag{5.53}$$

where $h = t_{k+1} - t_k$.

Issue 2: Enforcement of constraints with penalty functions
Consider the problem of minimizing the function

$$G(x, y) = x^2 + y^2,$$

subject to the constraint

$$\Psi_1 = x + y - 1 = 0.$$

One can propose to solve the constrained minimization problem by solving a related unconstrained problem, which includes the constraint equation in the objective function.

Define the augmented function as

$$\begin{aligned} \bar{G} &= G + p\Psi_1 \\ &= x^2 + y^2 + p(x + y - 1)^2. \end{aligned}$$

Minimization of $\bar{G}$ leads to the solution

$$x = y = \frac{p}{2p + 1}.$$

Comment
Penalty functions provide a convenient means for converting constrained optimization problems to unconstrained problems. The major drawback of penalty functions is that their use may lead to numerical difficulties (ill-conditioning) as the penalty parameter takes on *large* values.

Example 5.6. Discrete-time optimal control of a single-link rigid robot

Minimize

$$\Gamma = \frac{1}{2} \sum_{k=1}^{N} u_k^2,$$

subject to

$$\begin{aligned} \theta(0) &= 0 \\ \dot{\theta}(0) &= 0 \\ \theta(T) &= \theta_F \\ \dot{\theta}(T) &= 0. \end{aligned}$$

Solution

Defining $h = T/(N-1)$ and $t_k = (k-1)h$, one can use the state transition matrices given in Eq. (5.52) and Eq. (5.53). In order to enforce the constraints on the final states, consider the problem of minimizing

$$\bar{\Gamma} = \frac{1}{2}\sum_{k=1}^{N} u_k^2 + p[\mathbf{x}_N - \mathbf{x}_F]^T[\mathbf{x}_N - \mathbf{x}_F] \tag{5.54}$$

where

$$\mathbf{x}_F = \begin{bmatrix}\theta_F & 0\end{bmatrix}^T. \tag{5.55}$$

Examination of Eq. (5.54), and Eq. (5.55) reveals that the only non-zero terms to be used in the dynamic programming algorithm are

$$\begin{aligned} S_k &= 1. \qquad \text{for } k = 1, \ldots, N \\ \eta_N &= p\frac{\theta_F^2}{2} \\ \mathbf{y}_N &= -p\mathbf{x}_F \\ \mathbf{W}_n &= p\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}. \end{aligned}$$

For this example, consider the problem where $J = 1.5, T = 1.2, N = 201, \theta_F = \pi/2$, and $p = 1 \times 10^6$. The input (torque) and states (joint angle and joint angle rates) are plotted as functions of time (see Figure 5.40 and 5.41, respectively). Note, the constraints on the states at the final time are met satisfactorily by using the penalty function.

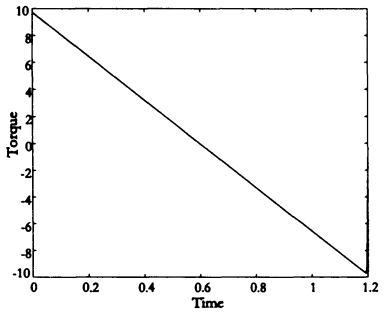

Figure 5.40: Torque time history for Example 5.6

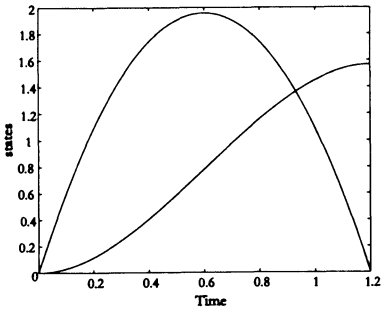

Figure 5.41: Joint angles and joint angle rates for Example 5.6

Example 5.7. Discrete-time optimal control of a single-link rigid robot with a modified objective function

$$\mathbf{x}_k = \begin{bmatrix} \theta(t_k) & \dot{\theta}(t_k) & M_k \end{bmatrix}^T \qquad u_k = \dot{M}_k$$

Minimize

$$\Gamma = \frac{1}{2}\sum_{k=1}^{N} u_k^2,$$

subject to

$$\begin{aligned} \theta(0) &= 0 \\ \dot{\theta}(0) &= 0 \\ M(0) &= 0 \\ \theta(T) &= \theta_F \\ \dot{\theta}(T) &= 0 \\ M(T) &= 0. \end{aligned}$$

Solution

This problem is solved in the same manner as Example 5.6 with the following differences:

$$\mathbf{A}_k = \begin{bmatrix} 1 & h & \frac{h^2}{(2J)} \\ 0 & 1 & \frac{h}{J} \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{B}_k = \begin{bmatrix} \frac{h^3}{(6J)} \\ \frac{h^2}{(2J)} \\ h \end{bmatrix}$$

$$\mathbf{x}_F = \begin{bmatrix} \theta_F \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{W}_N = p \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The results for this example (plotted in Figure 5.42 and Figure 5.43) were obtained by using a value of $p = 1 \times 10^8$ for the penalty parameter. Note, the torque time history is smoother than the one for Example 5.6.

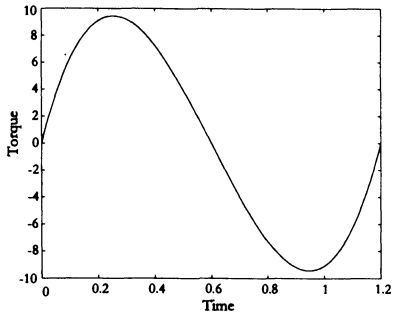

Figure 5.42: Torque time history for Example 5.7

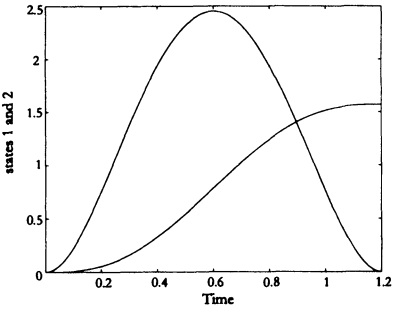

Figure 5.43: Joint angles and joint angle rates for Example 5.7

5.5.6. What Drove us to Dynamic Programming?

The motivation to utilize dynamic programming[30] was the need for a trajectory optimizer or path planner that could generate arbitrarily smooth input profiles with no initial guess (i.e., zero values) for the input parameters. The last three examples (see Examples 5.8, 5.9, and 5.10) of this chapter demonstrate the ability of dynamic programming to meet these needs.

Traditional rigid robot trajectory planning approaches are typically based on inverse kinematics techniques[31]. Use of these techniques is appropriate in many situations, but there are important instances when it is not. Such instances arise whenever robot flexibility or dynamics are of importance. For example, inverse kinematics schemes cannot accommodate structural flexibility in high payload-to-robot weight ratios[15]. Another feature of inverse kinematics solutions is that trajectories are not unique for redundant robots. The purpose of this section is to present an alternative approach for trajectory planning based on the method of dynamic programming that is applicable to rigid, flexible, and redundant robots.

The optimal trajectory planning problem for rigid robots is posed as a linear time-invariant system for both non-redundant and redundant configurations. Examples 5.8 and 5.9 demonstrate this approach by generating a pair of trajectories for end-effector tracking of a straight line with rest-to-rest motions of planar two-link and three-link rigid robots. The generality of the dynamic programming method is presented in Example 5.10. A single-link manipulator with a flexible payload is optimally slewed through a set of rest-to-rest maneuvers. The problem is posed as a nonlinear system with state variable equality constaints and input inequality constaints. The simulation results of the single-link manipulator display interesting symmetries characteristic of such motions.

Examples 5.8 and 5.9 deal with straight line tracking, rest-to-rest maneuvers of rigid two-link and three-link robots moving in a plane (see Figure 5.44). The goals are to move the tip of the terminal link from point A to point B while tracking a straight line connecting the two points and bringing the robot to rest in time T.

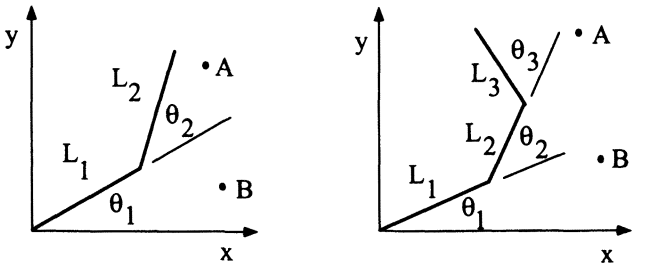

Figure 5.44. Sketch of planar robots considered in Examples 5.8 and 5.9

Joint trajectories for such maneuvers are not unique, that is there are an infinite number of joint time histories that meet the above stated goal. The optimal trajectory is defined as the one that minimizes the function C, or

$$C = \sum_{k=1}^{N} \sum_{j=1}^{N_j} (u_k^j)^2$$

where N is the number of time steps, N_j is the number of joints, $u_k^j = \ddot{\theta}_j$, and $\tau_k = (k-1)t/T$. Note, differentiation is with respect to $\tau = t/T$. The function C can be thought of as a measure of the smoothness of the trajectory and the tip position is defined as $\bar{x} = x/L_1$ and $\bar{y} = y/L_1$.

Example 5.8. Consider a two-link robot with $\theta_{1A} = \pi/4$, $\theta_{2A} = \pi/12$, $\bar{x}_B = 1.8$, $\bar{y}_B = -0.2$, $L_2/L_1 = 1$, and $N = 201$. The states for this problem are the joint angles and the joint velocities, and the inputs are the joint accelerations.

Solution

The state vector is $\mathbf{x}_k = \left[\theta_1(t_k), \theta_2(t_k), \dot{\theta}_1(t_k), \dot{\theta}_2(t_k)\right]^T$ for a massless two-link robot that is linear since one is integrating the input vector, $\mathbf{u}_k = \left[\ddot{\theta}_1(t_k), \ddot{\theta}_2(t_k)\right]^T$. The constraints are defined via the forward kinematics, which gives the tip motion that is suppose to track a straight line. Plots of the states and inputs as functions of τ are shown in Figure 5.45. Also shown in the Figure 5.45 is the path followed by the tip of the second link. Note, the straight line path is tracked and the robot is at rest at the conclusion of the maneuver. It is worth mentioning again that the trajectory shown in Figure 5.45 is one of many that can accomplish the goal of the stated maneuver.

Example 5.9. Consider a three-link robot with $\theta_{1A} = \pi/4$, $\theta_{2A} = \pi/12$, $\theta_{3A} = \pi/6$, $\bar{x}_B = 2.6$, $\bar{y}_B = -0.4$, $L_2/L_1 = 1$, $L_3/L_1 = 1$, and $N = 201$.

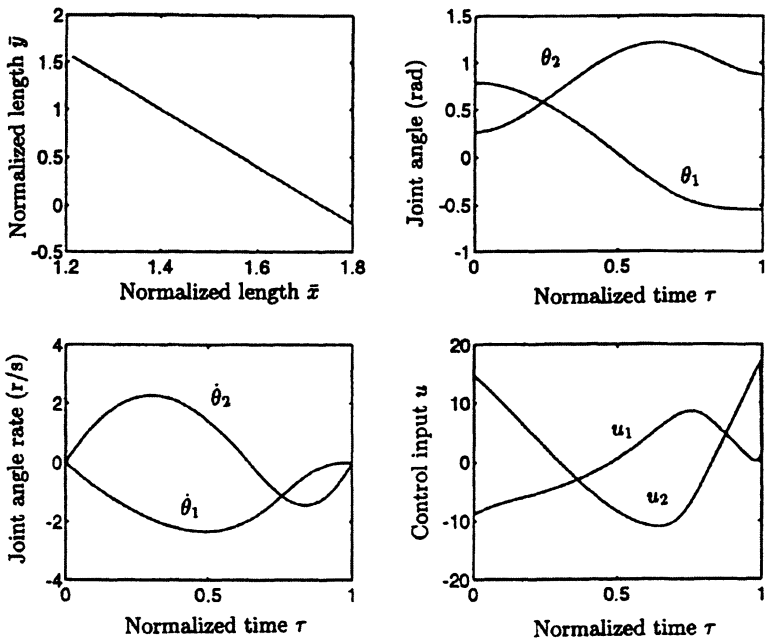

Figure 5.45: Straight line tracking rest-to-rest motion of two-link robot considered in Example 5.8

Solution

Plots of the states,inputs, and tip path are shown in Figure 5.46. Note again, the straight line path is tracked and the robot comes to rest at the endpoint.

It is worthwhile mentioning that there is not a one-to-one correspondence between the joint angles for straight line tracking in the case of the three-link robot. This is in contrast to the situation for the two-link robot in which there is only a single value of θ_2 for each value of θ_1. The redundant degree-of-freedom introduced by the additional link in this example poses no computational difficulties since the minimization of C leads to a unique solution.

Example 5.10. Single-link robot with a flexible payload.

This example is concerned with rest-to-rest maneuvers of a single-link robot with a flexible payload. A sketch of the model showing the relevant parameters is given in Figure 5.47. The equations of motion in dimensionless form are given by

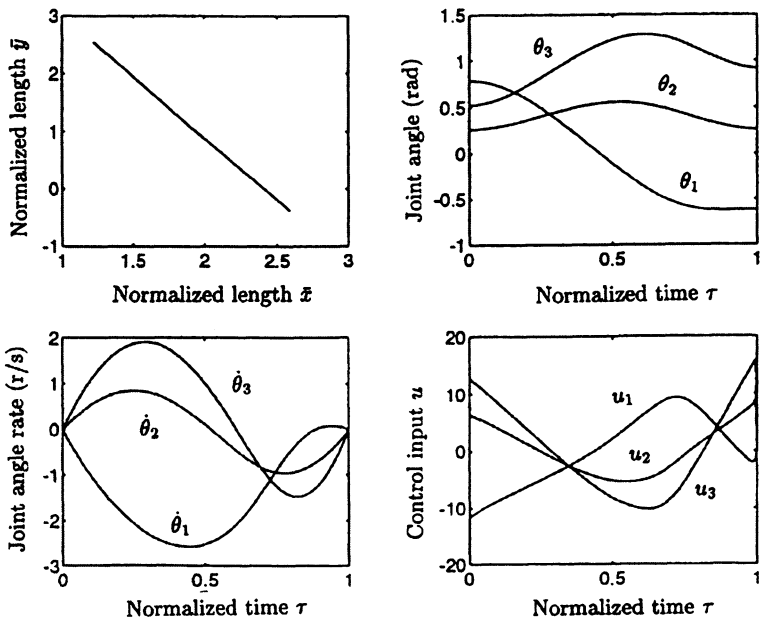

Figure 5.46: Straight line tracking rest-to-rest motion of three-link robot considered in Example 5.9

$$\begin{bmatrix} c_1 + (1+q_1)^2 + q_2^2 & -q_2 & 1+q_1 \\ -q_2 & 1 & 0 \\ 1+q_1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} c_2 u - 2\left[(1+q_1)\dot{q}_1 + q_2\dot{q}_2\right]\dot{\theta} \\ 2\dot{q}_2\dot{\theta} + (1+q_1)(\dot{\theta})^2 - c_3 q_1 \\ -2\dot{q}_1\dot{\theta} + q_2(\dot{\theta})^2 - c_3 q_2 \end{bmatrix} \quad (5.56)$$

where $c_1 = I/(mL^2), c_2 = AT^2/(mL^2), c_3 = (k/m)T^2$, and $u = M/A$ is the input subject to the constraints $-1 \leq u \leq 1$. For this example, the optimal rest-to-rest trajectory is defined as one that minimizes the function D defined as

$$D = \sum_{k=1}^{N} u_k^2$$

where $u_k = u(\tau_k)$.

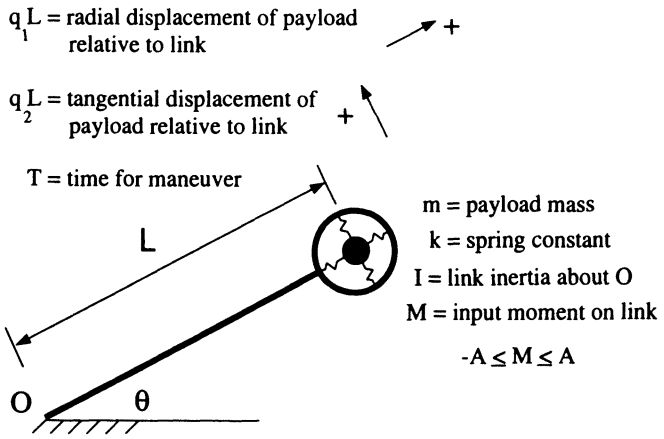

Figure 5.47: Sketch of single-link manipulator with a flexible payload considered in Example 5.10

Solution

Since closed-form solutions of Eq. (5.56) are generally not available, the dynamic programming algorithm made use of a numerical integration scheme to determine the g_k's and their first two derivatives (see Eq. 5.32). In this example, a fixed-step, fourth-order Runge-Kutta numerical integration scheme with $N = 201$ was used.

Results presented in Figure 5.48 show the states and inputs as functions of τ for a maneuver with $\theta(0) = 0$, $\theta(1) = 1/2$ radian, $c_1 = 1$, $c_2 = 5$, and $c_3 = (8\pi)^2$. The symmetries of the inputs and states about $\tau = 1/2$ are evident in Figure 5.48. Similar symmetries have been observed by for the control of a flexible rod[32] and for planar, time-optimal, rest-to-rest maneuvers of undamped flexible satellites[33].

The effect of varying payload stiffness on the optimal torque profile is also shown in Figure 5.48. Results are presented for the two values of c_3 indicated in Figure 5.48. As c_3 is increased, the torque profile approaches the one for a rigid payload (i.e., single-link rigid robot). As c_3 decreases, a critical value is reached below which the rest-to-rest maneuver is not possible. This lower bound is an important constraint to consider for the control of flexible payloads where the final angle and final time are specified (i.e., may not be attainable).

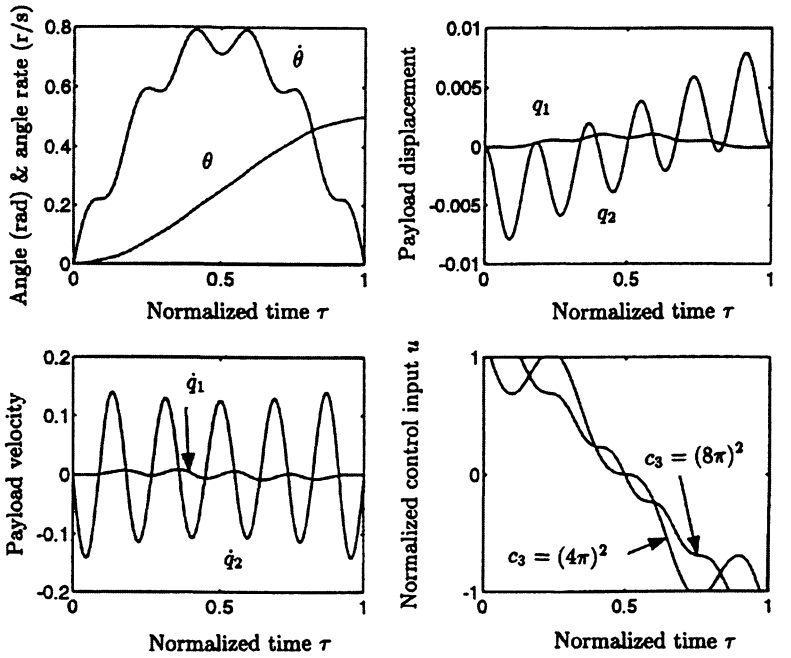

Figure 5.48: Slewing results for the single link robot with a flexible payload in Example 5.10

5.6. Chapter 5 Summary

This chapter reviewed several input shaping techniques. Input shaping techniques are a cost-effective way to minimize residual vibrations of flexible systems. Initially, an analytical solution was developed for a bang-coast-bang profile. The solution provided fundamental insight into open-loop slewing techniques. Next, both the FIR and IIR input shaping filter designs were introduced. Both of these methods were consolidated into prescriptive steps that could be used during the design of an input shaping profile for flexible systems. Further developments were included for the IIR filter that involved both continuous and discrete time-domains. Constrained optimization techniques that were used to develop parameterized open-loop control profiles were introduced. These profiles are considered suboptimal with respect to their defined performance index. Several optimization problem definitions were discussed that considered minimum effort, minimum effort with bounded control, and minimum time configurations. Examples were solved to demonstrate their effectiveness. Finally, dynamic programming techniques were developed as an efficient alternative for numerical optimization. Both optimal control and input shaping

problems were investigated. Because many real world applications that use open loop techniques often do not account for unmodeled effects, in Chapters 6–8 feedback control techniques are developed to address this problem.

5.7. Chapter 5 References

1. N. C. Singer, "Residual vibration reduction in computer controlled machines," Technical Report AI-TR 1030, MIT Artificial Intelligence Laboratory, Cambridge, MA, January 1989.
2. A. D. Christian, "Design and implementation of a flexible robot," Technical Report AI-TR 1153, MIT Artificial Intelligence Laboratory, Cambridge, MA, August 1989.
3. S. Yurkovich, A. Tzes, I. Lee, and K. Hillsley, "Control and system identification of a two-link flexible manipulator," *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pp. 1626-1631, Cincinnati, OH, May 13-19, 1990.
4. B. R. Murphy, and I. Watanabe, "Digital shaping filters for reducing machine vibration," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 2, pp. 285-289, April 1992.
5. R. Khorrami, "Analysis of multi-link flexible manipulators via asymptotic expansions," *Proceedings of the 28th IEEE Conference on Decision and Control*, pp. 2089-2094, Tampa, FL, December 1989.
6. E. Bayo, P. Papadopoulos, J. Stubbe, and M. A. Serna, "Inverse dynamics and kinematics of multi-link elastic robots: An iterative frequency domain approach," *The International Journal of Robotics Research*, Vol. 8, No. 6, December 1989.
7. J. T. Feddema, "Digital filter control of remotely operated flexible robotic structures," *Proceedings of the 1993 American Control Conference, San Francisco*, pp. 2710-2715, June 2-4, 1993.
8. G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems*, 2nd Edition, Addison-Wesley Publishing Company, 1990.
9. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetteling, *Numerical Recipies in C*, Cambridge University Press, Cambridge, MA, 1988, pp. 540-547.
10. P. E. Gill, et al., *Practical Optimization*, Academic Press, 1981.
11. M.J. Hopper, Harwell Subroutine Library, AERE-R9185, UKAEA, Harwell, UK, 1978.
12. M. A. Branch, and A. Grace, MATLAB®:Optimization Toolbox, Version 1.5, User's Guide, The MathWorks Inc., Natick, MA.
13. M. J. D. Powell, "A fast algorithm for nonlinearly constrained optimization calculations," *Numerical Analysis*, ed. G.A. Watson, Lecture Notes in Mathematics, Springer-Verlag, Vol. 630, 1978.
14. C. Bischof, et al., *ADIFOR 2.0 User's Guide. Revision C*, Technical Report CRPC-95516-S, August 1995, Argonne National Laboratory, Argonne, IL.
15. G. R. Eisler, et al., "Approximate optimal trajectories for flexible-link manipulator trajectories for flexible-Link manipulator slewing using recursive quadratic programming," Journal of Dynamic Systems, Measurement, and Control, Vol. 115, pp. 405-410, 1993.

16. R. L. Mayes, (1991), "Experimental characterization of flexible two link robot arm," Internal Memorandum, Sandia National Laboratories.
17. R. D. Robinett, (1987), "A unified approach to vehicle design, control, and flight path optimization," The Center for Strategic Technology, Texas A&M Univeristy, SS87-1, 1987.
18. J. L. Junkins, and J. P. Dunyak, (1982), "Continuation methods for enhancement of optimization algorithms," 19th Annual Meeting of the Society of Engineering Science, University of Missouri-Rolla, October, 1982.
19. E. B. Meier, and A. E. Bryson, (1990), "Efficient algorithm for time-optimal control of a two-link manipualtor," Journal of Guidance, Controls, and Dynamics, Vol. 13, Sept.-Oct., 1990, pp. 859-866.
20. N. K. Hecht, and J. L. Junkins, (1991), "Time-optimal reference maneuvers for a flexible arm," 1991 AIAA Guidance, Navigation, and Control Conference, New Orleans, Vol. 3, pp. 1592-1598.
21. G. G. Parker, G. R. Eisler, J. Phelan, and R. D. Robinett, III, "Input shaping for vibration-damped slewing of a flexible beam using a heavy-lift hydraulic robot," SAND report 94-1702C, Sandia National Laboratories, 1994
22. J. C. Dunn, and D. P. Bertsekas, "Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems," Journal of Optimization Theory and Applications, Vol. 63, No. 1, pp. 23-38, 1989.
23. C. R. Dohrmann, and R. D. Robinett, "Efficient sequential quadratic programming implementations for equality-constrained discrete-time optimal control," Journal of Optimization Theory and Applications, Vol. 95, No. 2, pp. 323-46, 1997.
24. C. R. Dohrmann, and R.D. Robinett, "Dynamic programming method for discrete-time optimal control," Journal of Optimization Theory and Applications, Vol. 101, No. 2, pp. 259-283, 1999.
25. R. E. Bellman, *An Introduction to the Theory of Dynamic Programming*, Rand Corporation, 1953.
26. R. E. Bellman, *Dynamic Programming*, Princeton University Press, New Jersey, 1957.
27. R. E. Bellman, *Applied Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1962.
28. S. E. Dreyfus, *Dynamic Programming and the Calculus of Variations*, Academic Press, New York, 1965.
29. C. T. Chen, *Linear System Theory and Design*, Holt, Rinehart & Winston, Inc. p. 142, 1984.
30. C.R. Dohrmann, and R. D. Robinett, "Robot trajectory planning via dynamic programming," Proceedings of the Fifth International Symposium on Robotics and Manufacturing, and Applications, Maui, HI, pp. 75-81, 1994.
31. M. W. Spong, and M. Vidyasagar, *Robot Dynamics and Control*, John Wiley and Sons, New York, 1989.
32. B. J. Petterson, and R. D. Robinett, "Model-based damping of coupled horizontal and vertical oscillations in a flexible rod," Journal of Intelligent and Robotic Systems, Vol. 4, pp. 285-299, 1991.
33. J. Ben-Asher, J. A. Burns, and E. M. Cliff, "Time-optimal slewing of flexible space-

craft," Journal of Guidance, Control, and Dynamics, Vol. 15, No. 2, pp. 360-367, March-April 1992.

5.8. Chapter 5 Problems

Homework 5.1. From Section 5.2.

1. Find the constraints on amplitude and frequency of a sinusoidal input profile for a swing-free, gantry robot maneuver (see Figure 5.49).

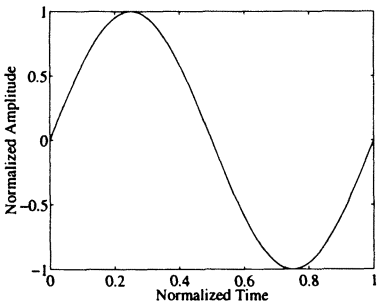

Figure 5.49. Input shaping profile with a sine wave

$$\ddot{x} = A \sin \Omega t. \tag{5.57}$$

$$\Omega = \frac{2\pi}{T}. \tag{5.58}$$

Homework 5.2. From Section 5.3.1.

1. Derive the FIR filter presented in Figure 5.6.

2. Derive a four-impulse shaping filter that compensates for a linear oscillatory system with two modes (ω_1 and ω_2) and no dampening. Assume the plant's impulse response is given by

$$y(t) = \frac{\omega_2^2}{(\omega_2^2 - \omega_1^2)} \omega_1 \sin(\omega_1 t) + \frac{\omega_1^2}{(\omega_1^2 - \omega_1^2)} \omega_2 \sin(\omega_2 t).$$

Show the output of the filter and the response of the oscillatory system to a step input. Assume the modes are at 1 Hz and 10 Hz, and the sampling period is 1 millisecond.

Homework 5.3. From Section 5.3.2.

Design an IIR filter that compensates for the oscillatory system described in problem 1 by cascading two IIR filters. Use the design by emulation approach and the desired transfer function

$$G_d(s) = \frac{p_1 p_2 p_3}{(s+p_1)(s+p_2)(s+p_3)}$$

where p_1, p_2 and p_3 are negative real numbers ($p_i \neq p_j$). Show the output of the filter and the response of the oscillatory system to a step input. Assume the modes are at 1 Hz and 10 Hz, and the sampling period is 1 millisecond. Choose p_1, p_2, and p_3 to your satisfaction.

Homework 5.4. From Section 5.4.3.

Solve Rosenbrock's problem by using Newton's optimization method. Compare iterations to converge to the gradient scheme used previously (see Homework 2.7 problem, #1). State your observations to the differences.

Homework 5.5. From Section 5.4.6.

1. Solve the gantry robot problem in Section 5.2. with a RQP method. Assume the problem has four (4) states θ, x, $\dot{\theta}$, and $\dot{x}$ (pendulum angle, linear distance, and their respective velocities), and three (3) parameters t_A, t_C, and A (acceleration and coast times, and amplitude of linear acceleration). There are five (5) active constraints $\theta(t_f) = 0$, $\dot{\theta}(t_f) = 0$, $x(t_f) = x_{\text{desired}}$, $\dot{x}(t_f) = 0$, and final time, $t_f = t_{\text{desired}} (= 2t_A + t_C)$. Since there are only 3 parameters and 5 constraints, solve the problem as a penalty function scalar performance index with a single equality constraint

$$\phi(\boldsymbol{\xi}) = W_1\theta(t_f)^2 + W_2\dot{\theta}(t_f)^2 + W_3\dot{x}(t_f)^3 + W_4(t_f - t_{desired})^2$$
$$\Psi(\boldsymbol{\xi}) = (x(t_f) - x_{desired})$$

where W_{1-4} are user specified weights (for example, set to inverse of maximum values).

Hint: Use a fixed-final time integration scheme and provide a method to test your initial parameter guesses to see that you are in the *ballpark* of a solution.

2. Solve the gantry problem for the input acceleration form $\ddot{x} = A\sin(\Omega t)$ where $\Omega = 2\pi/T$ and $t_f = t_{desired} = T$. This is now only a two (2) parameter problem.

3. Minimize the control effort for a single rigid link oriented in the vertical plane to execute a rest-to-rest maneuver to a given angle in a given final time. This is an example of a *parameterized* control history (refer to Section 5.4.8.). Minimize

$$\phi(\boldsymbol{\xi}) = x_3(t_f) = \int_{t_0}^{t_f} u^2 dt,$$

subject to

$$\frac{ml^2\ddot{\theta}}{3} + \frac{mgl\sin\theta}{2} = \tau = u$$
$$\theta(t_0) = \dot{\theta}(t_0) = 0$$
$$\Psi = \theta(t_f) - \theta_d = 0$$
$$\Psi_2 = \dot{\theta}(t_f) = 0.$$

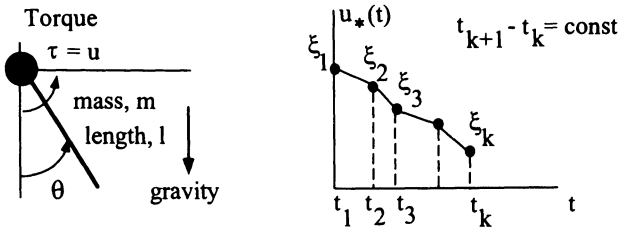

Figure 5.50. Problem definition with parameterized control template

Hints: Space the discrete times over the desired time interval of the slew and associate one (1) parameter with each discrete time t_i. Use a linear interpolation scheme to interpolate the control effort $u(t)$ for non-discrete time values during the integration of the state equations (see Section 2.4.1.). In addition to state equations $\mathbf{x}$ for θ and $\dot{\theta}$, a third state equation of the form $\dot{x}_3 = u^2$ will be appended to the differential equations to compute $\phi(\boldsymbol{\xi})$.

4. Find the minimum time for a two-link robot to complete a rest-to-rest maneuver to a new orientation in the horizontal plane.

   Parameters $\xi = [\xi_1, \xi_2, t_f]$ where ξ_1, ξ_2 are sets of torque values at discrete times (not necessarily the same times, except for the initial and final) for two joint motors, and t_f is the problem final time (see Figure 5.51).

   Minimize $\phi(\boldsymbol{\xi}) = t_f$

   Initial conditions $\theta_1(t_0), \theta_2(t_0)$, given $\dot{\theta}_1(t_0) = \dot{\theta}_2(t_0) = 0$.

   Constraints $\Psi(\boldsymbol{\xi}) = \begin{bmatrix} \theta_1(t_f) - \theta_{1_{desired}} \\ \theta_2(t_f) - \theta_{2_{desired}}) \\ \dot{\theta}_1(t_f) \\ \dot{\theta}_{2(t_f)} \end{bmatrix} = 0,$

physical properties

$$\begin{aligned} m_1, m_2 &= 2, 1 \text{ kg} \\ l_1, l_2 &= 1, 0.5 \text{ meters (m)} \\ I_1, I_2 &= 1, 0.5 \text{ kg-m}^2 \\ c_1, c_2 &= 1.0, 1.0 \text{ N-m-sec (joint damping)} \end{aligned}$$

boundary conditions

$$\begin{aligned} \theta_1(t_0), \theta_2(t_0) &= -20, 10 \text{ deg} \\ \theta_1(t_f), \theta_2(t_f) &= -30, 20 \text{ deg} \end{aligned}$$

torque limits

$$\tau_{1_{bound}}, \tau_{2_{bound}} = \pm 5, \pm 3 \text{ N-m}$$

and equi-sided bounded torque controls τ_1, τ_2.

Hints: Use Lagrange's method to develop the state equations for the two-link system as shown in Figure 5.51. For this problem, since t_f is both a parameter and the performance index, the derivatives are

$$\begin{aligned} \left.\frac{\partial \phi(\xi)}{\partial \xi}\right|_{\xi \neq t_f} &= 0, \\ \left.\frac{\partial \phi(\xi)}{\partial \xi}\right|_{\xi = t_f} &= 1. \end{aligned}$$

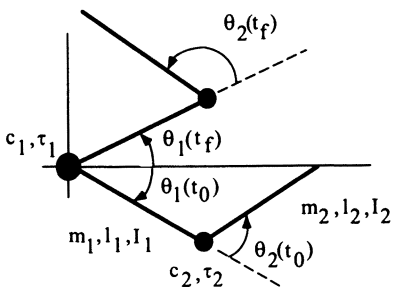

Figure 5.51. Two-link rigid robot

Homework 5.6. From Section 5.4.6.

Add a cubic spring to the basic single-link robot model and perform an input shaping design (i.e., swing-free) (see Figure 5.52).

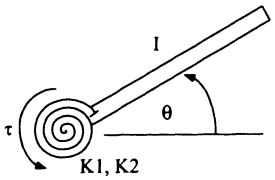

Figure 5.52. Slewing single-link with a nonlinear spring

a. Utilize an imbedded homotopy parameter.

b. Utilize Davidenko's method.

Hints:

i. The equation of motion is

$$u = I\ddot{\theta} + K_1\theta + K_2\theta^3. \tag{5.59}$$

ii. One homotopy approach is

$$u = I\ddot{\theta} + K_1\theta + \lambda K_2\theta^3. \tag{5.60}$$

iii. Pick $I = 1$, $K_1 = 2$, $K_2 = 1/2$.

Homework 5.7. From Section 5.5.2.

Determine the minimum cost to go from point A to point B (see Figure 5.53). Show the path that corresponds to your solution. Is this path unique?

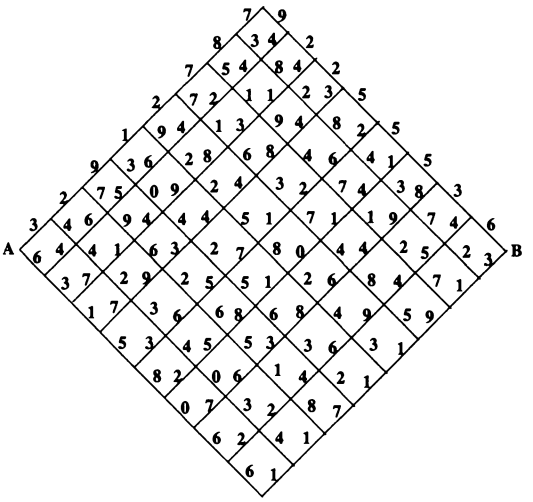

Figure 5.53. Dynamic programming combinatorial model for Homework 5.7

Homework 5.8. From Section 5.5.3.

Program the dynamic programming solution to the data smoothing problem in MATLAB® or some other programming language. Generate a noisy data set and test your algorithm on it. Plot your results as was done in Example 5.3. If your data set consists of an array of values that are all equal, what should your solution be? Explain.

Homework 5.9. From Section 5.5.5.

Solve this same problem (see Example 5.6) by using the Euler-Lagrange equations with constraints (these are the state/costate equations in Section 2.7.8.) and compare the results.

Homework 5.10. From Section 5.5.5.

The equation of motion for a beam cantilevered to a rotating hub was derived in Section 3.3.2. (see Eq. 3.39) and is given by

$$\ddot{q} + \left(\frac{140EI}{11\rho L^4} + \frac{1}{22} \left[\frac{35r}{L} + 5 \right] \omega^2 \right) q = -\frac{\dot{\omega}}{L^2} \left(\frac{35r}{44L} + \frac{7}{12} \right). \tag{5.61}$$

For rotation rates $\omega \ll \sqrt{140EI/11\rho L^4}$ Eq. (5.61) is approximated as

$$\ddot{q} + \left(\frac{140EI}{11\rho L^4} \right) q = -\frac{\dot{\omega}}{L^2} \left(\frac{35r}{44L} + \frac{7}{12} \right). \tag{5.62}$$

The parameters defining the system for this problem are $L = 10$, $r = 0$, $EI = 1.4e6$, and $\rho = 1.2$.

Part 1 Defining the state and input as

$$\mathbf{x}(t) = [q(t) \quad \dot{q}(t) \quad \theta(t) \quad \omega(t)]^T \qquad u(t) = \dot{\omega}(t) \tag{5.63}$$

determine

a. The state transition matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ in terms of the parameters defining the system and the fixed time step $h = t_{k+1} - t_k$.

b. The angular acceleration time history $\dot{\omega}(t)$ that minimizes the objective function

$$\Gamma = \frac{1}{2}\sum_{k=1}^{N} u_k^2, \tag{5.64}$$

subject to

$$\begin{aligned} q(0) &= 0 \\ \dot{q}(0) &= 0 \\ \theta(0) &= 0 \\ \omega(0) &= 0 \end{aligned}$$

and

$$\begin{aligned} \theta(T) &= \frac{\pi}{2} \\ q(T) &= 0 \\ \dot{q}(T) &= 0 \\ \omega(T) &= 0 \end{aligned}$$

where $T = 2$, $N = 201$, $h = T/N - 1$, and $t_k = (k-1)h$. Plot q, $\dot{q}$, θ, ω, and $\dot{\omega}$ as functions of time.

Part 2 Defining the state and input as

$$\mathbf{x}(t) = [q(t) \quad \dot{q}(t) \quad \theta(t) \quad \omega(t) \quad \dot{\omega}]^T \qquad u(t) = \ddot{\omega}(t) \tag{5.65}$$

determine

a. The state transition matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ in terms of the parameters defining the system and the fixed time step $h = t_{k+1} - t_k$.

b. The input time history $\ddot{\omega}(t)$ that minimizes the objective function

$$\Gamma = \frac{1}{2}\sum_{k=1}^{N} u_k^2 \tag{5.66}$$

subject to

$$q(0) = 0$$
$$\dot{q}(0) = 0$$
$$\theta(0) = 0$$
$$\omega(0) = 0$$

and

$$\theta(T) = \frac{\pi}{2} \tag{5.67}$$
$$q(T) = 0 \tag{5.68}$$
$$\dot{q}(T) = 0 \tag{5.69}$$
$$\omega(T) = 0 \tag{5.70}$$

where $T = 2$, $N = 201$, $h = T/N - 1$, and $t_k = (k-1)h$. Plot q, $\dot{q}$, θ, ω, and $\dot{\omega}$ as functions of time.

Part 3 By using the inputs obtained from Parts 1 and 2, plot q and $\dot{q}$ versus time for a system identical to the one specified with the exception that $EI = 1.38e6$. Comment on the behavior of $q(t)$ for $t > T$. Which of the two approaches (**Part 1** or **Part 2**) would you consider more *robust* to modeling errors?

Chapter 6

Linear Feedback Control

6.1. Introduction

This chapter describes several linear feedback control techniques that can be used to robustly control flexible dynamic systems. As with any dynamic system, it is often difficult to accurately model the system with enough fidelity that open loop control performs as intended. Because modeling errors are often unavoidable, linear feedback is often used to compensate for these modeling uncertainty. Even though many of the flexible dynamic systems are nonlinear, their models can be adequately linearized about operating points and standard linear feedback control techniques can be applied with satisfactory results.

As in the other chapters, this chapter presents many of the techniques used to control flexible systems in the form of examples. The first section describes a simple Proportional-Derivative (PD) control law that stabilizes the pendulum problem described in the previous chapters. In the next section, Lag-stabilization is presented as a method of reducing the oscillation of a flexible system. This is followed by a non-collocated control example of a simple flexible system. Finally, Proportional-Integral-Derivative (PID) and Linear Quadratic Gaussian (LQG) control techniques are presented for the planar two-link flexible robot arm.

6.2. PD Control of a Gantry Robot

It is most instructive to begin with a well-known example. In this section, a PD controller for the gantry robot with suspended payload as described in the previous chapters is developed. Figure 6.1 shows the parameters of the gantry. The payload mass is denoted by m, the length of the pendulum is l, the direction of motion is along x, and the angle of the pendulum with respect to vertical is θ. From Lagrange's equation, one can derive the nonlinear equation of motion, which

R. D. Robinett III et al., *Flexible Robot Dynamics and Controls*

provides the linearized dynamic model in the s-domain

$$\frac{\theta(s)}{s^2 X(s)} = \frac{\frac{-1}{l}}{s^2 + \omega^2}$$

where the natural frequency is $\omega = \sqrt{g/l}$ and g is gravity. The resulting impulse response to acceleration in x is

$$\theta(t) = \frac{-1}{l\omega} \sin(\omega t).$$

Note, this response is marginally stable with no damping. Therefore, if the pendulum is disturbed in x, it will oscillate forever. How can feedback control be used to dampen the response?

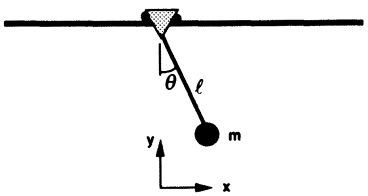

Figure 6.1. Diagram of a payload suspended from a gantry robot

Consider the PD feedback controller for disturbance rejection shown in Figure 6.2. The placement of the disturbance can be derived from Lagrange's equation.

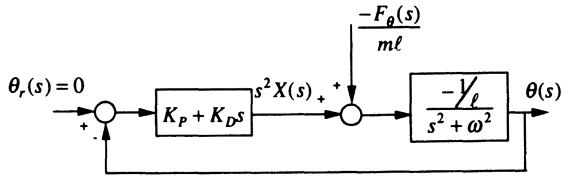

Figure 6.2. PD controller for disturbance rejection

Remember,

$$F_\theta = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0 \quad \text{if no external forces}$$

$$F_\theta = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = f_\theta(t) \quad \text{if external force such as a disturbance.}$$

After some mathematics, one finds that the Lagrange's equation for a disturbance in θ is $f_\theta(t) \approx ml^2(\ddot{\theta} + g/l\,\theta) + ml\ddot{x}$ for small θ. This can be written in the s-domain as

$$\theta(s) = \frac{\frac{-1}{l}}{s^2 + \omega^2}\left[s^2 X(s) - \frac{F_\theta(s)}{ml}\right]. \tag{6.1}$$

From Figure 6.2, one can see that $s^2X(s) = -(K_P + sK_D)\theta(s)$ where K_P is the proportional gain and K_D is the derivative gain. Substituting into Eq. (6.1) gives the closed-loop transfer function for a disturbance as

$$\frac{\theta(s)}{F_\theta(s)} = \frac{\frac{1}{ml^2}}{s^2 - \frac{K_D}{l}s + \left(\omega^2 - \frac{K_P}{l}\right)}.$$

From the denominator of this equation, one see that the closed-loop poles are at

$$s_{1,2} = 0.5\frac{K_D}{l} \pm 0.5\sqrt{\left(\frac{K_D}{l}\right)^2 - 4\left(\omega^2 - \frac{K_P}{l}\right)}. \tag{6.2}$$

First, one should consider when $K_D = 0$ and $K_P < l\omega^2$ (i.e., no damping). Note, the system is unstable for $K_P > l\omega^2$. Figure 6.3 shows the location of the poles.

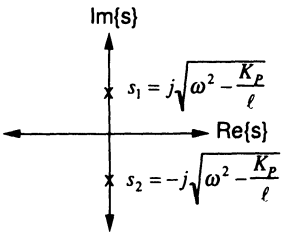

Figure 6.3. Pole locations for $K_D = 0$ and $K_P < l\omega^2$.

Increasing K_P decreases the natural frequency ω, but does not stop the oscillation.

Now, consider $K_D < 0$. Figure 6.4 shows that the new locations of the poles have moved into the left half plane. Decreasing K_D makes the exponential term decay faster, but also decreases the natural frequency (which implies a longer rise time). In general, there is a trade-off between the rise time and settling time. Increasing the proportional gain will shorten the rise time, but lengthen the settling time. Increasing the magnitude of the derivative gain will shorten the settling time, but lengthen the rise time. In practice, the proportional and derivative gains terms are often tuned by hand. First, adjusting the proportional gain to get a fast response and then adjusting the derivative gain to make the response critically damped.

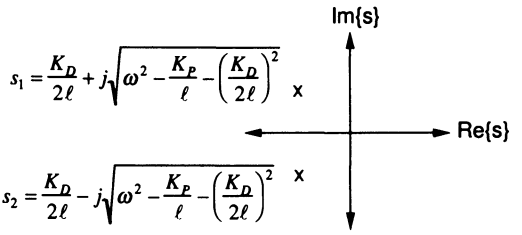

Figure 6.4. Pole locations for $K_D < 0$ and $K_P < l\omega^2$.

Now, it is time to consider a more formal method of determining these two gains. The proportional and derivative gains will be chosen for a fast critically damped response. If the system is critically damped, then the damping factor is one and the expression inside the square root of Eq. (6.2) is zero. Choosing only $K_D < 0$ gives

$$\frac{K_D}{l} = \pm 2\sqrt{\omega^2 - \frac{K_P}{l}}.$$

If $K_D = -2l\sqrt{\omega^2 - K_P/l}$, then

$$\frac{\theta(s)}{F_\theta(s)} = \frac{\frac{1}{ml^2}}{\left(s + \sqrt{\omega^2 - \frac{K_P}{l}}\right)^2}.$$

If $f_\theta(t) = \delta(t)$, then ω reaches a maximum when $t_s = 1/\sqrt{\omega^2 - K_P/l}$. This maximum value is

$$\theta(t_s) = \frac{1}{ml^2\sqrt{\omega^2 - \frac{K_P}{l}}} e^{-1}.$$

If one wants the system to stabilize within 5% of this maximum value in a settling time t_s, then

$$\theta(t_s) = \frac{1}{ml^2} t_s e^{-t_s\sqrt{\omega^2 - \frac{K_P}{l}}} - \frac{0.05}{ml^2} \frac{1}{\sqrt{\omega^2 - \frac{K_P}{l}}} e^{-1}.$$

This nonlinear equation can be solved for K_P by using a solver, such as ***fsolve*** in MATLAB®. For example, if $\omega = (2\pi/2.75)$ rad/sec, $l = 1.8792$ meters, $m = 20$ kg,

and $t_s = 2$ seconds, then

$$K_P = -5.134$$
$$K_D = -10.589$$
$$\frac{\theta(s)}{F_\theta(s)} = \frac{0.014193}{s^2 + 5.6415s + 7.9565} = \frac{0.014193}{(s + 2.8207)^2}.$$

Therefore, the impulse response is

$$\theta(t) = 0.014193te^{-2.8207t} \quad \text{when} \quad f_\theta(t) = \delta(t) .$$

The resulting control in acceleration is

$$s^2X(s) = -(K_P + K_Ds)\theta(s) = \frac{0.15028s + 0.072865}{s^2 + 7.44s + 13.84}.$$

Figures 6.5–6.7 show the disturbance impulse response and the resulting control in acceleration and in velocity, respectively.

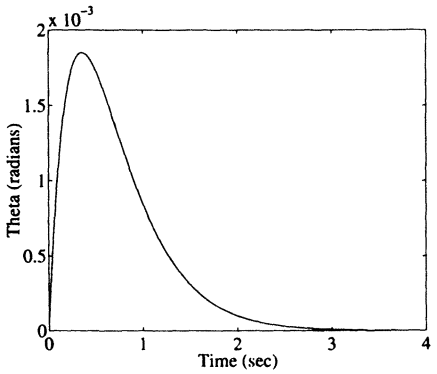

Figure 6.5: Angular response to an impulse disturbance for $K_P = -5.134$ and $K_D = -10.589$

Note, the PD control correctly controls θ, but it leaves the gantry moving at a constant velocity. Therefore, add a position feedback loop as shown in Figure 6.8. The open loop transfer function is

$$G(s)H(s) = \frac{K_{Px}(K_P + K_Ds)}{ls^2\left(s^2 - \frac{K_D}{l} + \left(\omega^2 - \frac{K_P}{l}\right)\right)}.$$

By using the same values of K_P and K_D gives the root locus as shown in Figure 6.9. Note, the poles move into the right hand plane and the system becomes unstable

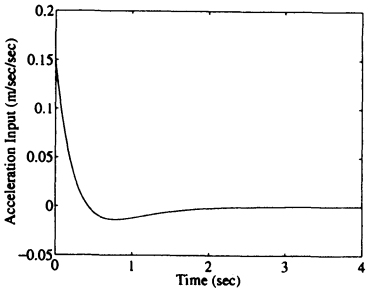

Figure 6.6: Resulting linear acceleration caused by PD feedback loop

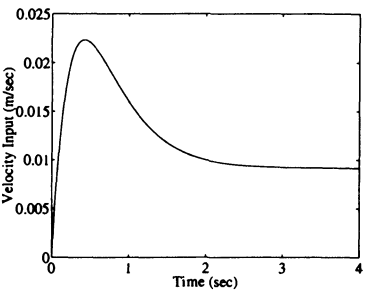

Figure 6.7: Resulting linear velocity caused by PD feedback loop

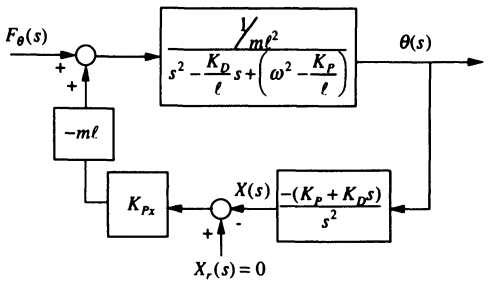

Figure 6.8: Position feedback loop added to stop motion caused by disturbance rejection feedback loop

when K_{Px} becomes too large. The closed loop transfer functions in position and angle are

$$\frac{\theta(s)}{F_\theta(s)} = \frac{\frac{1}{ml^2}s^2}{s^4 - \frac{K_D}{l} + \left(\omega^2 - \frac{K_P}{l}\right)s^2 + \frac{K_{Px}K_D}{l}s + \frac{K_{Px}K_P}{l}}$$

$$\frac{X(s)}{F_\theta(s)} = \frac{-\frac{1}{ml^2}(K_D s + K_P)}{s^4 - \frac{K_D}{l} + \left(\omega^2 - \frac{K_P}{l}\right)s^2 + \frac{K_{Px}K_D}{l}s + \frac{K_{Px}K_P}{l}}.$$

The impulse time responses are shown in Figures 6.10 and 6.11. Note, the position feedback introduces more oscillation into the system and the settling time is no longer 2 seconds.

In some situations, it is impossible to add an angle measuring sensor, such as

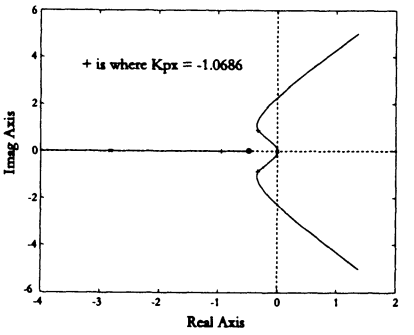

Figure 6.9. Root locus with additional position feedback loop

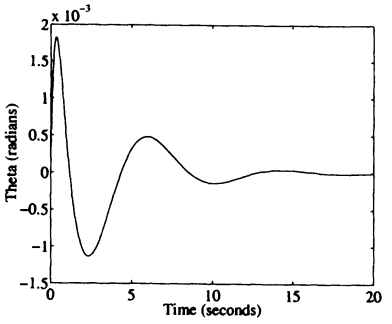

Figure 6.10: Angular response to impulse disturbance with additional position feedback loop

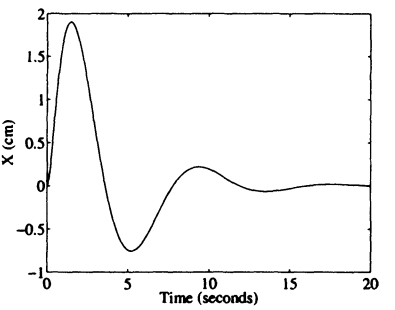

Figure 6.11: Position response to impulse disturbance with additional position feedback loop

an encoder for feedback. One alternative is to add a force sensor at the top of the cable[1]. As shown in Figure 6.12, the forces parallel and perpendicular to the mass motion are

$$F_{\|} = mg\sin\theta = ml\ddot{\theta}$$

$$F_{\perp} = T - mg\cos\theta = ml\dot{\theta}^2.$$

For small θ, the tension on the cable is $T = mg\cos\theta + ml\dot{\theta}^2 \approx mg$. Therefore, the forces felt by a force sensor are

$$F_x = T\sin\theta \approx mg\theta \tag{6.3}$$

$$F_y = -T\cos\theta \approx -mg. \tag{6.4}$$

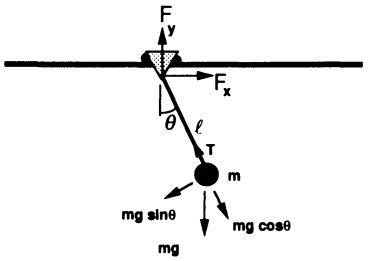

Figure 6.12. Notation for force sensing

Since the force sensor reading in the x-direction is proportional to the angle, it can be used to estimate the angle in the previously derived equations (see Eq. 6.3 and 6.4).

6.3. Lag-Stabilized Feedback Control

Most typical texts on control system theory[2,4] discuss the destabilizing effects of delay or lag in a system. In particular, controllers are often designed without regard to the lag in the system and then later *tuned* to accommodate this unmodeled behavior. In this section, a controller that relies on lag in the system for stability is described[3]. The lag generates rate feedback from position and/or acceleration measurements.

To illustrate the negative effects of lag, a simple spring-mass-damper system shown in Figure 6.13 (that could be used as a simple approximation for the first mode of a flexible arm) is stabilized by a PD controller. The dynamical model of the system is

$$m\ddot{x} + b\dot{x} + kx = u$$

where x is the position to be controlled, m is the mass, b is the damping term, k is the stiffness constant, and u is the control input. By using the Laplace transformation,

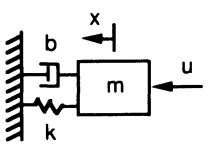

Figure 6.13. Single d.o.f. spring-mass-damper system

the s-domain representation is

$$(ms^2 + bs + k)x(s) = u(s).$$

A conventional PD control system can be defined as

$$u(s) = (K_P + K_D s)(x_r(s) - x(s))$$

where K_P is the proportional control gain, K_D is the derivative control gain, and x_r is the reference position. The closed-loop transfer function then becomes

$$\frac{x(s)}{x_r(s)} = \frac{K_P + K_D s}{ms^2 + (b + K_D)s + (k + K_P)}.$$

This closed-loop system is stable for all $K_P > 0$ and $K_D > 0$.

Now, suppose there is a delay T represented in the feedback path as e^{-Ts} (see Figure 6.14). By using a Taylor's series expansion the delay can be represented as

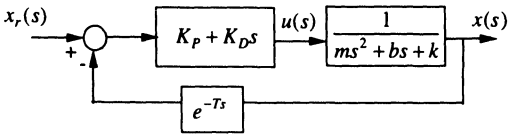

Figure 6.14. PD controller with delay in the feedback loop

$$e^{-Ts} = 1 - sT + \frac{(-sT)^2}{2!} + \frac{(-sT)^3}{3!} + \ldots .$$

By using only the first two terms, the closed loop transfer function becomes

$$\frac{x(s)}{x_r(s)} = \frac{K_P + K_D s}{(m - K_D T)s^2 + (b + K_D - K_P T)s + (k + K_P)}.$$

By using Routh-Hurwitz stability criterion, one can show that the system is stable if

$$K_P > -k \text{ and } -b + K_P T < K_D < m/T.$$

Note, the delay puts a limit on the derivative gain, above which the system becomes unstable. Earlier K_D could be selected as any positive value. Because of this, delay is typically thought of as undesirable. However, it can now be shown that delay can be stabilizing for oscillatory systems.

Once again, consider the pendulum problem, or

$$\frac{\theta(s)}{s^2 X(s)} = \frac{\frac{-1}{l}}{s^2 + \omega_n^2}$$

where $\omega_n = \sqrt{g/l}$, l is the length of the pendulum, g is the gravitational constant, and ω_n is the natural frequency of the system. By using a proportional controller with an intentional delay the modified control block diagram is shown in Figure 6.15.

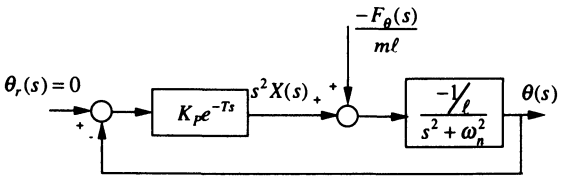

Figure 6.15. Lag stabilized controller with an input disturbance

The closed loop transfer function between the disturbance and the angle is

$$\frac{\theta(s)}{F_\theta(s)} = \frac{\dfrac{1}{ml^2}}{s^2 + \left(\omega_n^2 - \dfrac{K_P}{l}e^{-Ts}\right)}. \tag{6.5}$$

To obtain design equations for T and K_P, one substitutes $s = \sigma + j\omega$ into the characteristic equation derived from Eq. (6.5), or

$$(\sigma + j\omega)^2 + \left(\omega_n^2 - \frac{K_P}{l}e^{-\sigma T}e^{-j\omega T}\right) = 0. \tag{6.6}$$

By making the substitution $e^{-j\omega T} = \cos(\omega T) - j\sin(\omega T)$, the characteristic Eq. (6.6) becomes

$$\left(\sigma^2 - \omega^2 + \omega_n^2 - \frac{K_P}{l}e^{-\sigma T}\cos(\omega T)\right) + j\left(2\sigma\omega + \frac{K_P}{l}e^{-\sigma T}\sin(\omega T)\right) = 0.$$

Setting the real and imaginary parts, respectively, to zero and reassembling results in

$$\frac{K_P}{l}e^{-\sigma T}\cos(\omega T) = \sigma^2 - \omega^2 + \omega_n^2 \quad \text{and} \quad \frac{K_P}{l}e^{-\sigma T}\sin(\omega T) = -2\sigma\omega. \tag{6.7}$$

Dividing the imaginary part by the real part and taking the arc tangent defines the period as

$$T = \frac{1}{\omega}\tan^{-1}\left(\frac{2\sigma\omega}{\omega^2 - \omega_n^2 - \sigma^2}\right). \tag{6.8}$$

Substituting T back into the real part of Eq. (6.7) and solving for the proportional gain yields

$$K_P = \frac{-l(\omega^2 - \omega_n^2 - \sigma^2)}{e^{-\sigma T}\cos(\omega T)}.$$

Changing the period T and the proportional gain K_P will move the poles of the original system as indicated in Figure 6.16. The stability and performance of

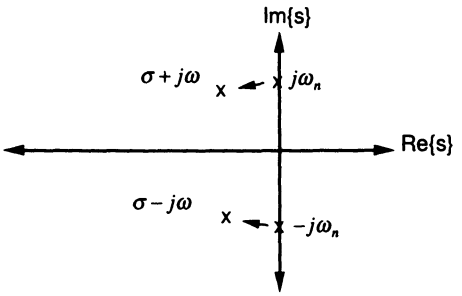

Figure 6.16. Root locus for the lag stabilized controller

the system can be evaluated for the disturbance frequencies ω's close to the natural frequency of the system ω_n. Approximate the lag term[5] with

$$e^{-sT} = \frac{e^{-0.5sT}}{e^{+0.5sT}} \approx \frac{1 - 0.5sT}{1 + 0.5sT}, \tag{6.9}$$

and substitute Eq. (6.9) into the characteristic equation derived from Eq. (6.5) to produce a rational function approximation, as

$$(1 + 0.5sT)s^2 + (1 + 0.5sT)\omega_n^2 - \frac{K_P}{l}(1 - 0.5sT) = 0,$$

or

$$s^3 + \frac{2}{T}s^2 + \left(\omega_n^2 + \frac{K_P}{l}\right)s + \frac{2}{T}\left(\omega_n^2 - \frac{K_P}{l}\right) = 0.$$

By using Routh-Hurwitz Criterion, the system is stable if $0 < K_P < l\omega_n^2$.

For the pendulum system, the approximate transfer function becomes

$$\frac{\theta(s)}{F_\theta(s)} = \frac{\frac{1}{ml^2}}{s^2 + \left(\omega_n^2 - \frac{K_P}{l}e^{-Ts}\right)} \approx \frac{\frac{1}{ml^2}\left(s + \frac{2}{T}\right)}{s^3 + \frac{2}{T}s^2 + \left(\omega_n^2 + \frac{K_P}{l}\right)s + \left(\omega_n^2 - \frac{K_P}{l}\right)}.$$

One further simplification in determining σ is setting $\sigma = -\zeta\omega$ where ζ is the desired damping factor. In most fundamental control system designs, second-order systems with natural frequencies and damping factors are usually well understood and have familiar characteristics. For this example, the controller design and physical parameters are given in Table 6.1 for both a marginally stable and stable control design. Each design's time domain simulations are shown in Figure 6.17. Substituting the parameters into the Routh-Hurwitz Stability Criterion results in $0 < K_P < 9.81$. Note, this is positive position feedback! The approximate numerical transfer function for the stable control design in Table 6.1 is

$$\frac{\theta(s)}{F_\theta(s)} = \frac{(14.192 \times 10^{-3})(s + 3.7126)}{s^3 + 3.7126s^2 + 8.5017s + 7.2053}$$
$$= \frac{(14.192 \times 10^{-3})(s + 3.7126)}{(s + 1.3586)(s + 1.1770 + j1.9794)(s + 1.1770 - j1.9794)}.$$

Table 6.1. Lag stabilized design example parameters

Case	ω	ω_n	ζ	σ	T	K_P	m	l
Marginal Stable	2.286	2.286	0.0	0.0	0.0	6.162	20.0	1.88
Stable	2.286	2.286	0.707	−1.616	0.538	6.162	20.0	1.88

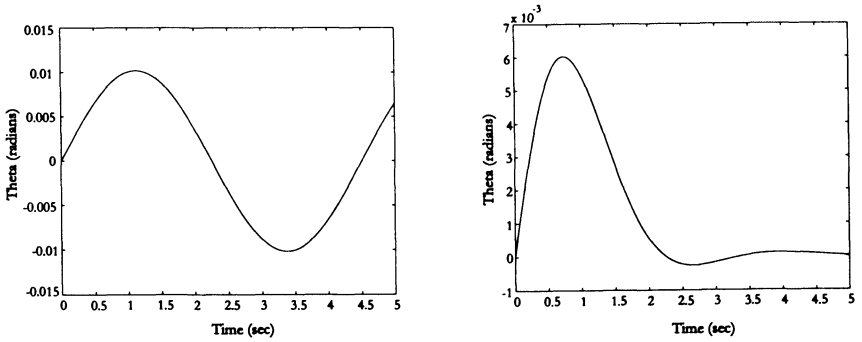

Figure 6.17. Impulse response, lag controller marginal stable and stable cases

The system stability and performance are tunable by determining the desired values of ω and ζ. Note, ω must be kept close to ω_n in order to obtain highly damped performance and maintain stability. The control scheme is based on phase shifting or differentiating sinusoidal signals (system responses) with respect to time,

with time delay to obtain rate feedback (i.e., no time delay = no damping). Again, this implies the desired frequency must be kept close to the natural frequency in order to time shift the correct signal and generate a rate signal that damps the system.

Typically, feedback delay in control systems is minimized as much as possible in order to achieve favorable stability margins. An atypical controller was presented that derives its stability from the lag in the feedback loop. A design example was completed for the gantry robot modeled as a simple pendulum model. Sandia has successfully applied this technique by using a force feedback controller to damp initial and residual oscillations of a slewing flexible link. The controller implements rate feedback for damping the oscillation by utilizing force sensor measurements that are delayed in time. This time delay created a lag-stabilized control system. The lag-stabilizing effects are discussed in detail and demonstrated on a commercial Cincinnati Milacron T3-786 robot[6]. In particular, vibration suppression is accomplished by feeding back force sensor measurements that are delayed in time and proportional to the displacement, or acceleration of the flexible link sinusoids. Nyquist stability criterion theoretical developments were added for stabilizing oscillatory systems with positive delayed feedback[7].

6.4. Non-collocated Controls

Traditionally, robotic manipulators are independently joint controlled, with each joint having collocated sensors and actuators. This results in good control stability characteristics. Normally, many of the tasks associated with manipulation involve end point control. For a rigid link manipulator this is achieved by using forward kinematics to process end point information. If the calibration is poor, the accuracy may not be acceptable, thus mandating additional end-effector sensing and non-collocated control. For flexible link manipulators, the dynamics due to the flexibility of the links introduce additional difficulties with non-collocated control, which take the form of non-minimum phase characteristics. To maintain end-effector performance, a better understanding of the dynamics and how to maintain stable control requires additional analysis techniques. The effects of non-collocated control and non-minimum phase can be visualized with an example. Consider a single slewing horizontal link with a collocated sensor/actuator pair at the joint. By moving the link from some initial location to a final location, the tip of the link will follow and be accurately positioned. By replacing the link with a flexible member and performing the same slew, one will observe that the tip of the flexible member will initially move in the opposite direction to the joint motion. To maintain tip position accuracy, further information about the dynamics is needed.

By processing sensor information about the tip with the actuator still located at the joint, the system becomes non-collocated. In addition, it becomes difficult to achieve satisfactory performance while maintaining stability with conventional low-order compensation. This problem is more challenging for the control engineer

and requires the use of advanced controller design techniques. The structural resonances found in many different systems, such as flexible space structures, flexible mechanisms, flexible manipulators, etc. fall into this category. The goal of this section is to help identify non-collocated control characteristics.

As an example, the structural resonances inherent in lightweight manipulator components can be investigated with a simple model. Each component is made-up of rigid and flexible portions. As an initial approach consider the following model constructed by using two-discrete components with a mass-spring-damper modeled at the node (see Figure 6.18). The system has one input $u = \tau$ at the joint and two outputs. The first output is the joint angle θ_1. The second is the angular displacement θ_2, which represents the relative motion of the second discrete component to the first. For the purposes of this analysis, the input/output relationship between u and θ_1 is collocated, but between u and $\theta_{2_I} = \theta_1 + \theta_2$ is non-collocated. θ_{2_I} is defined as the inertial angle. The effects of collocated and non-collocated control are explored in the following control system design and synthesis.

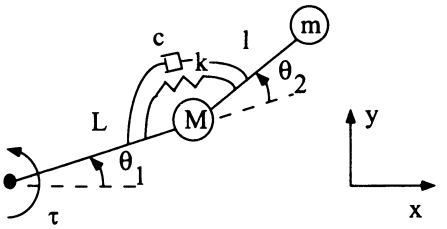

Figure 6.18. Simple two discrete component dynamic model

Applying Lagrange's equations from Chapter 2 gives the following nonlinear dynamic equations of motion result:

$$\tau = \left[(M+m)L^2 + ml^2 + 2mLlcos\theta_2\right]\ddot{\theta}_1 + \left[ml^2 + mLlcos\theta_2\right]\ddot{\theta}_2 - mLl(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2)\sin\theta_2$$

$$0 = \left[ml^2 + mLlcos\theta_2\right]\ddot{\theta}_1 + ml^2\ddot{\theta}_2 + mLl\dot{\theta}_1^2 sin\theta_2 + k\theta_2 + c\dot{\theta}_2.$$

Next, the equations of motion are linearized about an operating point $\theta_1 = \theta_2 = 0$ where the approximations $\sin\theta \approx \theta$ and $\cos\theta \approx 1$ are employed. All nonlinear terms are set to zero. The linearized equations of motion become

$$u = h_{11}\ddot{\theta}_1 + h_{12}\ddot{\theta}_2 \tag{6.10}$$

$$0 = h_{12}\ddot{\theta}_1 + h_{22}\ddot{\theta}_2 + c\dot{\theta}_2 + k\theta_2 \tag{6.11}$$

where

$$h_{11} = (M+m)L^2 + ml^2 + 2mLl$$
$$h_{12} = ml^2 + mLl$$
$$h_{22} = ml^2.$$

Taking the Laplace transform of Eq. (6.10) and Eq. (6.11) yields

$$u(s) = h_{11}s^2\theta_1(s) + h_{12}s^2\theta_2(s)$$
$$0 = h_{12}s^2\theta_1(s) + h_{22}s^2\theta_2(s) + cs\theta_2(s) + k\theta_2(s).$$

Solving for the transfer functions of interest yields the collocated transfer function as

$$\frac{\theta_1(s)}{u(s)} = \frac{h_{22}s^2 + cs + k}{s^2\left[(h_{11}h_{22} - h_{12}^2)s^2 + h_{11}cs + h_{11}k\right]}, \tag{6.12}$$

the relative angle transfer function as

$$\frac{\theta_2(s)}{u(s)} = \frac{-h_{12}s^2}{s^2\left[(h_{11}h_{22} - h_{12}^2)s^2 + h_{11}cs + h_{11}k\right]},$$

and the non-collocated transfer function as

$$\frac{\theta_{2_I}(s)}{u(s)} = \frac{(-h_{12} + h_{22})\,s^2 + cs + k}{s^2\left[(h_{11}h_{22} - h_{12}^2)s^2 + h_{11}cs + h_{11}k\right]}. \tag{6.13}$$

The compensator investigated is based on a PD control system and is represented in the s-domain as

$$u(s) = k_p e(s) + k_v s e(s),$$

or

$$G_c(s) = \frac{u(s)}{e(s)} = k_v(s + \alpha)$$

where $\alpha = k_p/k_v$ is a selected control system parameter. The error is defined as $e = r - \theta_l$ where $l = 1$ or 2_I. From Eq. (6.12), the plant transfer function for the collocated system is defined as

$$G_{p_1}(s) = \frac{\theta_1(s)}{u(s)}$$

where the combined control/plant block diagram is shown in Figure 6.19. From Eq. (6.13), the non-collocated system is defined as

$$G_{p_{2_I}}(s) = \frac{\theta_{2_I}(s)}{u(s)}$$

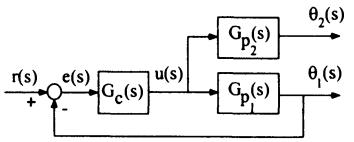

Figure 6.19: System block diagram using collocated feedback

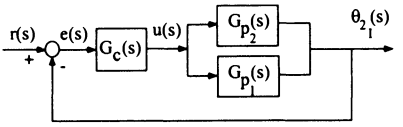

Figure 6.20: System block diagram using non-collocated feedback

where the sum of both transfer functions becomes the feedback signal. The new control/plant block diagram is shown in Figure 6.20.

In the first design only the collocated plant is considered. The control system performance requirements are a critically damped response with minimum overshoot and a settling time of $t_s \leq 1.0$ second. The physical parameters are $M = 20, m = 1, L = l = 0.25, k = 10$, and $c = 0.25$. A root locus design (see Figure 6.21) was performed by using the PD compensator with the controller zero set to $\alpha = 0.05$ and a gain of $k_v = 7.8521$ was selected from the plot. The closed-loop response to a 1 rad step input is shown in Figure 6.22 (left plot). For this case, the non-collocated signal is not considered in the design. The tip response is shown in Figure 6.22 (right plot). If the design required the tip response to be controlled to certain specifications, then structural resonance could become a problem.

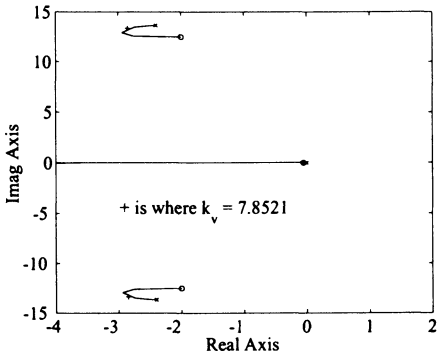

Figure 6.21. Root locus plot using collocated transfer function

In the second design, the non-collocated transfer function is considered in the control system design. The same controller zero ($\alpha = 0.05$) is retained. The root locus plot of the non-collocated system is shown in Figure 6.23. Note, the zero in the right-half plane is characteristic of non-minimum phase systems. A control gain of $k_v = 1.7733$ is selected with the corresponding step response shown in Figure 6.24 (left plot). The settling time has increased substantially to $t_s = 5.0$

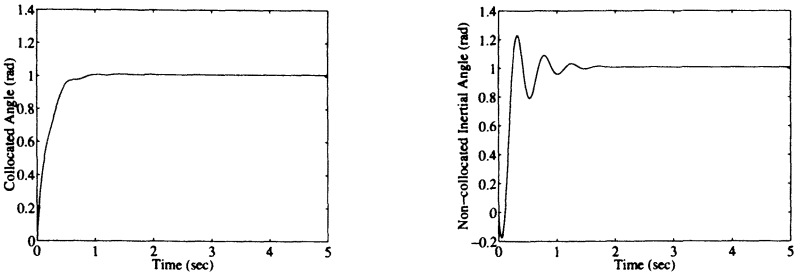

Figure 6.22. Step responses for collocated designed and non-collocated observed

seconds. Naturally, to improve the performance, the gain must be increased. When a gain of $k_v = 3.55$ (to the right of the $j\omega$ axis - see Figure 6.23) is selected, the step response is shown in Figure 6.24 (right plot). The response has become oscillatory and unstable, thus destabilizing the structural mode. For the non-collocated case, using PD compensation to stabilize the system becomes much more difficult. This type of system would require higher-order compensation to improve performance, resulting in more complexity during implementation (increase in analog components or discrete number of computations).

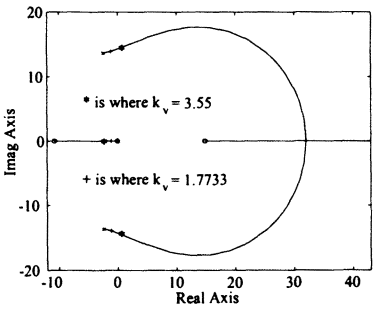

Figure 6.23. Root locus plot using non-collocated transfer function

Several successful designs have been described in the literature. Initial investigations demonstrated experimental end-point control of a slewing flexible link[8]. For a planar flexible two-link manipulator, end-point control by using a real-time camera as a tip position feedback sensor was demonstrated[9]. End-point control design by using convex optimization techniques has been demonstrated [10]. An alternative non-collocated measurement, called the virtual angle of rotations that resulted

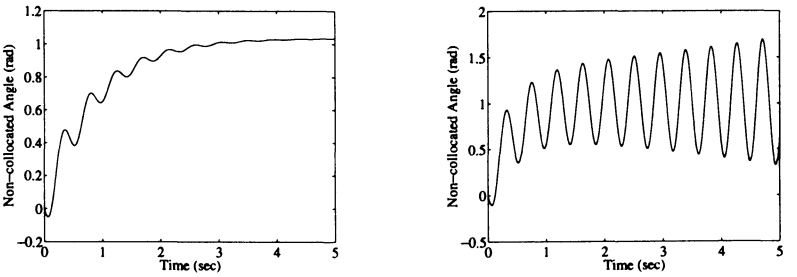

Figure 6.24. Step responses for non-collocated $k_v = 1.7733$ and $k_v = 3.55$

in several useful properties was introduced[11]. Several significant contributions of other researchers involving non-collocated non-minimum phase control were also identified. In a more recent study, μ synthesis techniques for flexible manipulators taking into consideration combinations of both collocated and non-collocated sensor/actuator locations was suggested[12]. The resulting controllers were shown to be robust to high frequency dynamics, actuator uncertainty, noise, and mode variations.

In this section several aspects of collocated and non-collocated control were demonstrated. The complexities involved in using non-collocated control were identified. The non-minimum phase characteristic present in a simple model is representative of flexibility effects found in non-rigid link manipulators.

6.5. Feedforward Control

In this section, a feedforward control scheme[13] is discussed that uses the torque profile created by the optimization technique of Section 5.4.10. and the finite element model in Section 3.4.. The optimization algorithm generated the motor torque profiles shown in Figure 6.25. The resulting tip trajectory is shown in Figure 6.26 as the curve labeled *model*. Note, at the (x,y) position (0.95m, 0.15m), the tip trajectory is not monotonic along the line and actually doubles back at the point where there is a large spike in the joint 2's torque. Since the tip still lies close to the straight line during the spike, this does not violate any of the optimization constraints. The large spike in joint 2's torque is to eliminate any residual oscillation at the final position. In the future, it may be desirable to pose the optimization constraints so that this doubling back does not occur.

The first experiment tested was an open loop torque controller (see Figure 6.27). Because of slight modeling errors, the joint positions and velocities did not follow the desired trajectories (see Figure 6.28). As shown in Figure 6.26, this resulted in a poor tip trajectory and the tip never reached the final desired position. While the model was not accurate enough for open loop control, a comparison of the strain

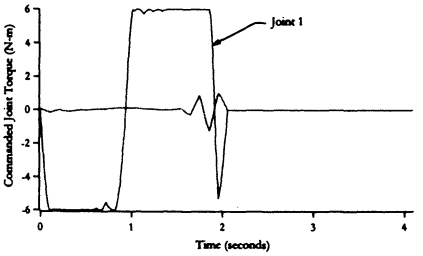

Figure 6.25. Modeled joint torque as specified by the minimum time optimization

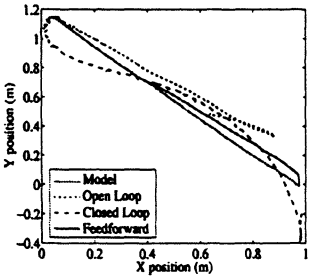

Figure 6.26: Tip position of the two-link arm for open loop torque profiles, closed loop PD joint control, and feedforward control. Rigid link kinematics used to generate tip position from joint angles

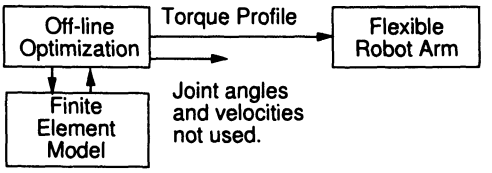

Figure 6.27. Open loop control

gauge readings to the estimated link curvatures (see Figure 6.29) shows that the model was not far from the actual system. In fact, Fourier transforms of these plots prove to be almost identical. The largest discrepancy is that the first modal frequency according to the model is about 10 Hz compared with about 8 Hz in the experiment. These are good results considering that nominal values were used to

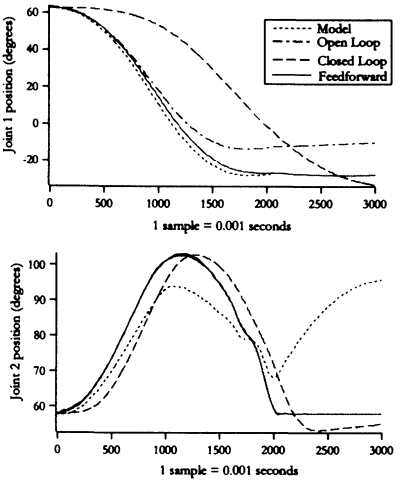

Figure 6.28: Joint positions for open loop torque profiles, closed loop PD joint control, and the feedforward control.

compute the system parameters, i.e. link inertias, stiffness, densities, etc.. The parameters of the model could be further refined by using a least squares parameter identification scheme. Repeating the optimization with these refined parameters would improve the open loop response.

The second experiment used a PD joint controller (see Figure 6.30). Here, the desired joint positions and velocities were those specified by the optimization. Although the joint positions and velocities do eventually converge on the desired values (convergence occurred after the 3 seconds as shown in Figure 6.28), this control approach will not produce a minimum time motion. Since the errors in joint positions and velocities start out small, the resulting torque at the start of the motion will be much smaller than the step torque input specified by the optimization (see Figure 6.31). The gain constants $\mathbf{K}_p$ and $\mathbf{K}_v$ were found experimentally. The torque's oscillation shown in Figure 6.31 is the result of the difference between the joints' desired velocity according to the optimization and the actual velocity according to the tachometers. Vibration in the links reflects back to the joints and is observed in the tachometer feedback. It was impossible to critically dampen the system because of this effect and increasing $\mathbf{K}_v$ resulted in instability. As shown in Figure 6.26 the tip position oscillated about the desired trajectory. The tip position did converge on the final position, but after the 3 seconds as is also shown in Figure 6.26.

In the final experiment a feedforward controller was used (see Figure 6.32). Compared to the previous two methods this method provided the best attributes.

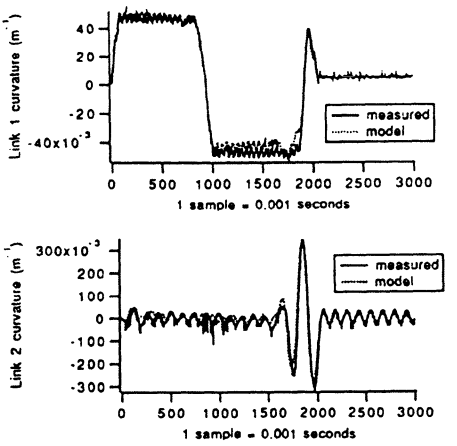

Figure 6.29: Link curvatures according to the model and during the open loop control experiment

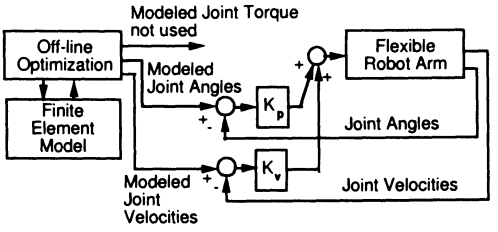

Figure 6.30. Closed loop joint control

First, the feedforward torque term provided the torque necessary for a minimum time motion. Second, the error in joint position and velocity were used to correct any modeling errors. A proportional error term in curvature was also used to dampen vibration transients. The desired curvature was specified from the optimization and the measured curvature was derived from strain gauges along the links. Strain gauges were located at positions on the link that corresponded to nodal positions of the finite element model. This curvature feedback is especially effective in damping out residual vibrations at the end of the trajectory when the desired curvature is zero. Accelerometer feedback was not used in these experiments since the waveforms were almost identical to the strain gauge waveforms. In addition, the strain gauge signals were less noisy than the accelerometers. The resulting joint torques of the

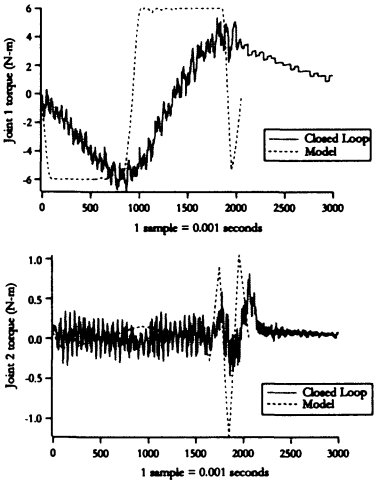

Figure 6.31: Comparison between modeled torque specified by optimization and torque generated from closed loop PD joint control.

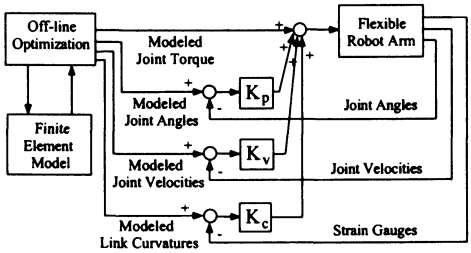

Figure 6.32. Feedforward control

feedforward controller are shown in Figure 6.33.

As shown in Figures 6.26 and 6.28, the performance of the feedforward approach is superior to the open-loop and PD joint control schemes. Although the tip positions in Figure 6.26 were computed with the rigid link kinematics, a rough comparison of the control schemes is achieved. More importantly, the measured joint angles, joint accelerations, and link curvatures were driven close to the modeled values with the feedforward control, thus indicating that the straight line trajectory determined by the model was performed.

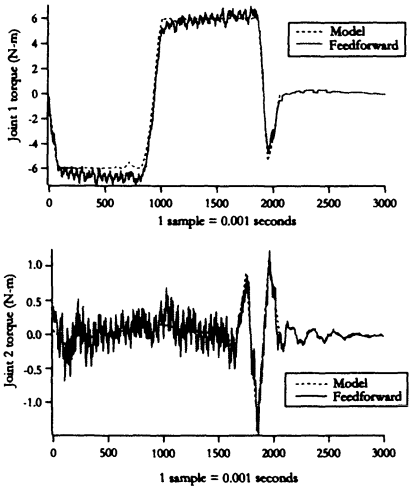

Figure 6.33: Comparison of modeled torque specified by optimization and torque generated from the feedforward control.

6.6. Linear Quadratic Regulator

To compliment the off-line optimization, one would like to develop an optimal controller that accommodates perturbations from the optimal path. From Figure 6.34, one would like to solve for the time varying control matrix $\mathbf{C}(t)$. The following subsections provide a brief overview of the theoretical background required to determine this optimal control gain. Further details can be found in any optimal control book[14–20].

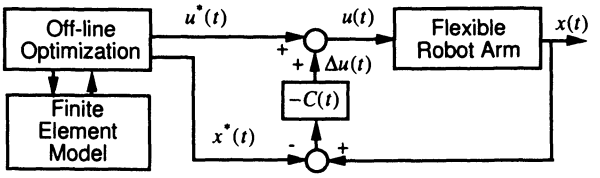

Figure 6.34. Linear quadratic regulator feedback

6.6.1. Necessary Conditions for Optimality

In Section 2.7.8., the following optimization problem was considered: Given the nonlinear, time-varying model

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t].$$

The goal is to minimize the augmented cost function

$$J = \phi[\mathbf{x}(t_f), t_f] + \int_{t_0}^{t_f} \langle L[\mathbf{x}(t), \mathbf{u}(t), t] + \boldsymbol{\lambda}^T(t)\{\mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t] - \dot{\mathbf{x}}(t)\}\rangle dt$$

where

$\mathbf{x}(t)$ = state vector
$\mathbf{u}(t)$ = control vector
$\mathbf{f}[\mathbf{x}, \mathbf{u}, t]$ = nonlinear system dynamics
$\phi[\mathbf{x}(t_f), t_f]$ = final cost function
$L[\mathbf{x}(t), \mathbf{u}(t), t]$ = integrand cost function
$\boldsymbol{\lambda}^T(t)$ = Lagrange multiplier—also called the adjoint vector because it adjoins the dynamic constraint to the cost integrand.

The Hamiltonian is defined as

$$H[\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t), t] = L[\mathbf{x}(t), \mathbf{u}(t), t] + \boldsymbol{\lambda}^T(t)\mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t].$$

One wishes to choose the adjoint vector time history $\boldsymbol{\lambda}(t)$ such that the system is stationary with respect to infinitesimal control variations. Setting the first variation of J to zero yields the following necessary conditions when the final time is fixed:

1. $\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t]$ with $\mathbf{x}(t_0)$ given

2. $$\dot{\boldsymbol{\lambda}} = -\left\{\frac{\partial H[\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t), t]}{\partial \mathbf{x}}\right\}^T \text{ for } \boldsymbol{\lambda}(t_f) = \left.\frac{\partial \phi[\mathbf{x}(t), t]}{\partial \mathbf{x}}\right|_{t=t_f}^T$$

3. $$\left\{\frac{\partial H[\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t), t]}{\partial \mathbf{u}}\right\}^T = \mathbf{0}. .$$

Once again, these are called the Euler-Lagrange equations. They form a two-point boundary value problem where there is a constraint on $\mathbf{x}(t)$ at the beginning and on $\boldsymbol{\lambda}(t)$ at the end. Solving the above equations simultaneously (given the boundary conditions) is difficult, and this solution provides the open loop control or input shaping command. Instead, one would like to find a feedback solution that is independent of $\mathbf{x}(t)$ and $\mathbf{u}(t)$.

6.6.2. Neighboring-Optimal Solutions

As seen in Section 6.5., the robot did not follow the optimal path exactly when using an open loop torque profile. The feedforward control scheme showed that one needs to modify the control input based on the perturbed states. Let

$$\begin{aligned}\Delta\mathbf{x}(t) &= \mathbf{x}(t) - \mathbf{x}^*(t)\\ \Delta\mathbf{u}(t) &= \mathbf{u}(t) - \mathbf{u}^*(t)\\ \Delta\boldsymbol{\lambda}(t) &= \boldsymbol{\lambda}(t) - \boldsymbol{\lambda}^*(t)\end{aligned}$$

where the optimal solution (i.e., the input shaped command) that is generated off-line will be denoted with a * superscript.

The linearized state equation about the off-line computed optimal path is

$$\Delta\dot{\mathbf{x}}(t) = \mathbf{F}(t)\Delta\mathbf{x}(t) + \mathbf{G}(t)\Delta\mathbf{u}(t), \quad \Delta\mathbf{x}(t_0) \text{ is given and}$$

where

$$\mathbf{F}(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \quad \text{and} \quad \mathbf{G}(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}.$$

The cost function can now be written as

$$\begin{aligned}J[\mathbf{x}^*(t) + \Delta\mathbf{x}(t), \mathbf{u}^*(t) + \Delta\mathbf{u}(t)] \cong\ & J[\mathbf{x}^*(t), \mathbf{u}^*(t)]\\ &+ \Delta J[\Delta\mathbf{x}(t), \Delta\mathbf{u}(t)] + \Delta^2 J\mathbf{L}[\Delta\mathbf{x}(t), \Delta\mathbf{u}(t)].\end{aligned}$$

It has already been shown that the first variation is zero. The second variation can be expressed as

$$\begin{aligned}\Delta^2 J = & \frac{1}{2}\Delta\mathbf{x}^T(t_f)\boldsymbol{\phi}_{\mathbf{xx}}(t_f)\Delta\mathbf{x}(t_f)\\ &+ \frac{1}{2}\int_{t_0}^{t_f}\left\{\begin{bmatrix}\Delta\mathbf{x}^T(t) & \Delta\mathbf{u}^T(t)\end{bmatrix}\begin{bmatrix}\mathbf{L}_{xx}(t) & \mathbf{L}_{xu}(t)\\ \mathbf{L}_{ux}(t) & \mathbf{L}_{uu}(t)\end{bmatrix}\begin{bmatrix}\Delta\mathbf{x}(t)\\ \Delta\mathbf{u}(t)\end{bmatrix}\right\}dt \qquad (6.14)\end{aligned}$$

where $\boldsymbol{\phi}_{\mathbf{xx}}$ is the second partial derivative of ϕ with respect to $\mathbf{x}(t)$, $\mathbf{L}_{xx}$ is the second partial derivative of $\mathbf{L}$ with respect to $\mathbf{x}(t)$, $\mathbf{L}_{ux} = \mathbf{L}_{xu}$ is the partial derivative of $\mathbf{L}$ with respect to $\mathbf{x}(t)$ and $\mathbf{u}(t)$, and $\mathbf{L}_{uu}$ is the second partial derivative of $\mathbf{L}$ with respect to $\mathbf{u}(t)$.

Ignoring the coupling terms between $\mathbf{x}(t)$ and $\mathbf{u}(t)$ and denoting the diagonal terms $\mathbf{L}_{xx}$ and $\mathbf{L}_{uu}$ as $\mathbf{Q}(t)$ and $\mathbf{R}(t)$, write the Hamiltonian as

$$\begin{aligned}H(t) = & \frac{1}{2}[\Delta\mathbf{x}^T(t)\mathbf{Q}(t)\Delta\mathbf{x}(t) + \Delta\mathbf{u}^T(t)\mathbf{R}(t)\Delta\mathbf{u}(t)]\\ &+ \Delta\boldsymbol{\lambda}^T(t)[\mathbf{F}(t)\Delta\mathbf{x}(t) + \mathbf{G}(t)\Delta\mathbf{u}(t)].\end{aligned}$$

The matrices $\mathbf{Q}(t)$, $\mathbf{R}(t)$, and $\mathbf{P}(t_f) = \boldsymbol{\phi}_{\mathbf{xx}}(t_f)$ are weighting matrices that are specified by the control system designer. The matrix $\mathbf{R}(t)$ must be positive definite

and the matrix $\mathbf{Q}(t)$ must be positive semi-definite. Increasing the matrix $\mathbf{Q}(t)$, increases the emphasis on following the desired states during the transient response. Increasing matrix $\mathbf{R}(t)$, decreases the control effort used to follow the desired states. Increasing the matrix $\mathbf{P}(t_f)$, increases the emphasis on achieving the desired final position at time t_f.

The resulting Euler-Lagrange equations for Eq. (6.14) are

$$\Delta\dot{\mathbf{x}}(t) = \mathbf{F}(t)\Delta\mathbf{x}(t) + \mathbf{G}(t)\Delta\mathbf{u}(t), \quad \Delta\mathbf{x}(t_0) \text{ is given}$$

$$\Delta\dot{\boldsymbol{\lambda}} = -\left\{\frac{\partial H}{\partial(\Delta\mathbf{x})}^T\right\} = -\mathbf{Q}(t)\Delta\mathbf{x}(t) - \mathbf{F}^T(t)\Delta\boldsymbol{\lambda}(t) \tag{6.15}$$

$$\left\{\frac{\partial H}{\partial(\Delta\mathbf{u})}\right\}^T = \mathbf{R}(t)\Delta\mathbf{u}(t) + \mathbf{G}^T(t)\Delta\boldsymbol{\lambda}(t) = \mathbf{0}. \tag{6.16}$$

where

$$\begin{aligned}\boldsymbol{\lambda}(t_f) &= \left\{\frac{\partial}{\partial(\Delta\mathbf{x})}\left[\frac{1}{2}\Delta\mathbf{x}^T(t)\mathbf{P}(t)\Delta\mathbf{x}(t)\right]\right\}\Bigg|_{t=t_f} \\ &= \mathbf{P}(t_f)\Delta\mathbf{x}(t_f) \\ \mathbf{P}(t_f) &= \boldsymbol{\phi}_{\mathbf{xx}}(t_f)\end{aligned} \tag{6.17}$$

Equation (6.16) implies $\Delta\mathbf{u}(t) = -\mathbf{R}^{-1}(t)\mathbf{G}^T(t)\Delta\boldsymbol{\lambda}(t)$.

If one assumes that $\Delta\boldsymbol{\lambda}(t) = \mathbf{P}(t)\Delta\mathbf{x}(t)$ for all t where $\mathbf{P}(t)$ is not yet known, then the control can be written as

$$\Delta\mathbf{u}(t) = -\mathbf{R}^{-1}(t)\mathbf{G}^T(t)\mathbf{P}(t)\Delta\mathbf{x}(t).$$

$\mathbf{R}(t)$, and $\mathbf{G}(t)$ are known, but what is $\mathbf{P}(t)$ for all t? From the Euler-Lagrange equations (see Eq. 6.15)

$$\begin{aligned}\Delta\dot{\boldsymbol{\lambda}}(t) &= -\mathbf{Q}(t)\Delta\mathbf{x}(t) - \mathbf{F}^T(t)\mathbf{P}(t)\Delta\mathbf{x}(t) \\ &= \frac{d}{dt}\{\mathbf{P}(t)\Delta\mathbf{x}(t)\} \\ &= \dot{\mathbf{P}}(t)\Delta\mathbf{x}(t) + \mathbf{P}(t)\Delta\dot{\mathbf{x}}(t) \\ &= \dot{\mathbf{P}}(t)\Delta\mathbf{x}(t) + \mathbf{P}(t)[\mathbf{F}(t)\Delta\mathbf{x}(t) + \mathbf{G}(t)\Delta\mathbf{u}(t)] \\ &= \left[\dot{\mathbf{P}}(t) + \mathbf{P}(t)\mathbf{F}(t) - \mathbf{P}(t)\mathbf{G}(t)\mathbf{R}^{-1}(t)\mathbf{G}^T(t)\mathbf{P}(t)\right]\Delta\mathbf{x}(t).\end{aligned}$$

The resulting matrix Riccati equation is

$$\mathbf{P}(t) = -\mathbf{Q}(t) - \mathbf{F}^T(t)\mathbf{P}(t) - \mathbf{P}(t)\mathbf{F}(t) + \mathbf{P}(t)\mathbf{G}(t)\mathbf{R}^{-1}(t)\mathbf{G}^T(t)\mathbf{P}(t)$$

where $\mathbf{P}(t_f)$ is the known boundary condition.

Note, $\mathbf{P}(t)$ is independent of $\Delta\mathbf{x}(t)$ and $\Delta\mathbf{u}(t)$. Therefore, variations in $\Delta\mathbf{x}(t_0)$ have no effect on the optimal feedback control law gains, or

$$\begin{aligned}\Delta\mathbf{u}^*(t) &= -\mathbf{R}^{-1}(t)\mathbf{G}^T(t)\mathbf{P}(t)\Delta\mathbf{x}(t) \\ &= -\mathbf{C}(t)\Delta\mathbf{x}(t).\end{aligned}$$

The control gain $\mathbf{C}(t)$ can be computed off-line by using the linearized dynamic system about the optimal trajectory $\mathbf{x}^*(t)$. Unfortunately, the computation of the integral in the Riccati equation is often difficult. Therefore, it is common practice to compute the steady state solution where $\dot{\mathbf{P}}(t) = \mathbf{0}$. This is referred to as the algebraic Riccati equation, or

$$\mathbf{0} = \mathbf{F}^T(t)\mathbf{P}(t) + \mathbf{P}(t)\mathbf{F}(t) - \mathbf{P}(t)\mathbf{G}(t)\mathbf{R}^{-1}(t)\mathbf{G}^T(t)\mathbf{P}(t) + \mathbf{Q}(t),$$

and there are several software packages, such as MATLAB® that can solve this equation.

6.7. Linear Optimal Estimation

The linear optimal regulator discussed in the previous section assumed that all states could be measured and controlled. However, often the state measurements are corrupted by noise. In these cases, a Kalman-Bucy filter is used to estimate the true state values. In this section, the derivation of the Kalman-Bucy filter closely follows that of Anderson-Moore[21]. There are other ways of deriving the filter[22–24].

Before introducing the Kalman-Bucy filter, one must review a few basic definitions. The mean of a random variable X is

$$\widehat{x} = E\{X\} = \int_{-\infty}^{\infty} x p_X(x) dx$$

where $p_X(x)$ is the probability density function. The covariance of X is

$$\Sigma_{XX} = E\{[X - \widehat{x}][X - \widehat{x}]^T\} = \int_{-\infty}^{\infty} [x - \widehat{x}][x - \widehat{x}]^T p_X(x) dx,$$

and is a symmetric positive definite matrix. The most used probability density function $p_X(x)$ is the Normal (also called Gaussian) probability function. Its statistics are uniquely defined by two parameters (mean and covariance). The probability density function for a Normal distribution is

$$p_X(x) = \frac{1}{(2\pi)^{0.5n} \mid \Sigma_{XX} \mid^{0.5}} \exp\left[-0.5(x - \widehat{x})^T \Sigma_{XX}^{-1}(x - \widehat{x})\right]$$

where n is the dimension of x.

Again one will consider the state-space model in the LQR section, except this time suppose that the measurement of the state vector is not perfect and that state noise $\mathbf{w}(t)$ is introduced, or

$$\Delta\dot{\mathbf{x}}(t) = \mathbf{F}(t)\Delta\mathbf{x}(t) + \mathbf{G}(t)\Delta\mathbf{u}(t) + \mathbf{w}(t). \tag{6.18}$$

Also assume that one cannot directly measure all the states and the observation vector is

$$\Delta \mathbf{z}(t) = \mathbf{H}(t)\Delta \mathbf{x}(t) + \mathbf{v}(t) \tag{6.19}$$

where $\mathbf{v}(t)$ is observation noise. Assume both the state and observation noise are *white, zero-mean Gaussian* random sequences that are uncorrelated with each other. In other words,

$$\begin{aligned} E\{\mathbf{w}(t)\} &= 0 \\ E\{\mathbf{v}(t)\} &= 0 \\ E\{\mathbf{w}(t)\mathbf{w}(\tau)^T\} &= \mathbf{W}(t)\delta(t-\tau) \\ E\{\mathbf{v}(t)\mathbf{v}(\tau)^T\} &= \mathbf{V}(t)\delta(t-\tau) \end{aligned}$$

where

$$\delta(t-\tau) = \begin{cases} 1 & t = \tau \\ 0 & t \neq \tau \end{cases}$$

$$E\{\mathbf{v}(t)\mathbf{w}(\tau)^T\} = 0 \text{ for all } t \text{ and } \tau.$$

The covariance matrix $\mathbf{V}(t)$ is positive definite, and the covariance matrix $\mathbf{W}(t)$ is positive semi-definite. Also assume that the initial state mean value and its covariance are known and given by

$$\begin{aligned} E\{\Delta \mathbf{x}(t_0)\} &= \Delta \hat{\mathbf{x}}(t_0) \\ E\{(\Delta \mathbf{x}(t_0) - \Delta \hat{\mathbf{x}}(t_0))(\Delta \mathbf{x}(t_0) - \Delta \hat{\mathbf{x}}(t_0))^T\} &= \mathbf{\Sigma}(t_0) \end{aligned} \tag{6.20}$$

The optimal linear estimate minimizes the error variance

$$J = E\{[\Delta \mathbf{x}(t_1) - \Delta \hat{\mathbf{x}}(t_1)]^T[\Delta \mathbf{x}(t_1) - \Delta \hat{\mathbf{x}}(t_1)]\}. \tag{6.21}$$

The solution to this minimization subject to the dynamics in Eqs. (6.18)–(6.20) is given by integrating

$$\Delta \dot{\hat{\mathbf{x}}}(t) = \mathbf{F}(t)\Delta \hat{\mathbf{x}}(t) + \mathbf{G}(t)\Delta \mathbf{u}(t) + \mathbf{K}(t)\left[\Delta \mathbf{z}(t) - \mathbf{H}(t)\Delta \hat{\mathbf{x}}(t)\right] \tag{6.22}$$

where the Kalman filter gain is

$$\mathbf{K}(t) = \mathbf{\Sigma}(t)\mathbf{H}^T(t)\mathbf{V}^{-1}(t). \tag{6.23}$$

The covariance matrix is determined by integrating the Ricatti equation

$$\dot{\mathbf{\Sigma}}(t) = \mathbf{F}(t)\mathbf{\Sigma}(t) + \mathbf{\Sigma}(t)\mathbf{F}^T(t) + \mathbf{W}(t) - \mathbf{K}(t)\mathbf{V}(t)\mathbf{K}^T(t) \tag{6.24}$$

starting with the initial condition $\mathbf{\Sigma}(t_0)$. As with the linear quadratic regulator, it is difficult to numerically solve the Ricatti equation. Therefore, it is common

practice to solve the algebraic Ricatti equation for the steady state solution of $\mathbf{\Sigma}(t)$, or

$$\mathbf{0} = \mathbf{F}(t)\mathbf{\Sigma}(t) + \mathbf{\Sigma}(t)\mathbf{F}^T(t) + \mathbf{W}(t) - \mathbf{K}(t)\mathbf{V}(t)\mathbf{K}^T(t).$$

Note, the form of the optimal estimation solution is nearly identical to the solution of the linear quadratic regulator. In fact, the optimal estimation problem given in Eq. (6.21) can be transformed into the optimal regulator problem by defining a set of dual equations. The dual dynamic system is given by

$$\dot{\mathbf{X}}(t;t_1) = -\mathbf{F}^T(t)\mathbf{X}(t;t_1) + \mathbf{H}^T(t;t_1)\mathbf{M}(t), \quad \mathbf{X}(t_1;t_1) = \mathbf{I} \tag{6.25}$$

where $\mathbf{X}(t;t_1)$ (costate) is a transition matrix (i.e. $\mathbf{X}^{-1}(t;t_1) = \mathbf{X}(t_1;t)$) of the same dimensions as the state $\Delta\mathbf{x}(t)$, and $\mathbf{I}$ is the identity matrix. The matrix $\mathbf{M}(t)$ is also a transition matrix and will be explained shortly. Notice that the boundary condition is at the final time t_1 and one is interested in the solutions for $t \leq t_1$. Also note, this equation is similar to the state equation except that $\mathbf{F}(t)$ has become $-\mathbf{F}^T(t)$ and $\mathbf{G}(t)$ has become $\mathbf{H}^T(t)$.

Differentiating the product of the costate and state gives

$$\begin{aligned}\frac{d}{dt}[\mathbf{X}^T(t;t_1)\Delta\mathbf{x}(t)] &= \dot{\mathbf{X}}^T(t;t_1)\Delta\mathbf{x}(t) + \mathbf{X}^T(t;t_1)\Delta\dot{\mathbf{x}}(t) \\ &= -\mathbf{X}^T(t;t_1)\mathbf{F}(t)\Delta\mathbf{x}(t) + \mathbf{M}^T(t;t_1)\mathbf{H}(t)\Delta\mathbf{x}(t) \\ &\quad + \mathbf{X}^T(t;t_1)\mathbf{F}(t)\Delta\mathbf{x}(t) + \mathbf{X}^T(t;t_1)\mathbf{w}(t) \\ &= \mathbf{M}^T(t;t_1)\Delta\mathbf{z}(t) - \mathbf{M}^T(t;t_1)\mathbf{v}(t) + \mathbf{X}^T(t;t_1)\mathbf{w}(t)\end{aligned}$$

assuming that $\Delta\mathbf{u}(t) = \mathbf{0}$. Integrating from t_0 to t_1 and applying the boundary condition on $\mathbf{X}(t;t_1)$ at time t_1 gives

$$\begin{aligned}\Delta\mathbf{x}(t_1) - \int_{t_0}^{t_1}\mathbf{M}^T(t;t_1)\Delta\mathbf{z}(t)dt = \mathbf{X}^T(t_0;t_1)\Delta\mathbf{x}(t_0) + \int_{t_0}^{t_1}\mathbf{X}^T(t;t_1)\mathbf{w}(t)dt \\ - \int_{t_0}^{t_1}\mathbf{M}^T(t;t_1)\mathbf{v}(t)dt.\end{aligned} \tag{6.26}$$

The second term on the left hand side is the optimal state estimate of Eq. (6.21), or

$$\Delta\hat{\mathbf{x}}(t_1) = \int_{t_0}^{t_1}\mathbf{M}^T(t;t_1)\Delta\mathbf{z}(t)dt \tag{6.27}$$

where the transition matrix function $\mathbf{M}(t;t_1)$ can be thought of as an impulse response matrix to the observation $\Delta\mathbf{z}(t)$.

Combining Eq. (6.26) with Eq. (6.21) yields

$$\begin{aligned}E\{[\Delta\mathbf{x}(t_1) - \Delta\hat{\mathbf{x}}(t_1)][\Delta\mathbf{x}(t_1) - \Delta\hat{\mathbf{x}}(t_1)]^T\} &= \mathbf{X}^T(t_0;t_1)\mathbf{\Sigma}(t_0)\mathbf{X}^T(t_0;t_1) \\ &\quad + \int_{t_0}^{t_1}[\mathbf{X}^T(t;t_1)\mathbf{W}(t)\mathbf{X}(t;t_1) + \mathbf{M}^T(t;t_1)\mathbf{V}(t)\mathbf{M}(t;t_1)]dt.\end{aligned}$$

Taking the trace of both sides gives

$$E\{[\Delta\mathbf{x}(t_1) - \Delta\hat{\mathbf{x}}(t_1)]^T[\Delta\mathbf{x}(t_1) - \Delta\hat{\mathbf{x}}(t_1)]\} = tr\{\mathbf{X}^T(t_0;t_1)\boldsymbol{\Sigma}(t_0)\mathbf{X}^T(t_0;t_1) + \int_{t_0}^{t_1}[\mathbf{X}^T(t;t_1)\mathbf{W}(t)\mathbf{X}(t;t_1) + \mathbf{M}^T(t;t_1)\mathbf{V}(t)\mathbf{M}(t;t_1)]dt\}. \quad (6.28)$$

Now, the problem is in the same form as the LQR problem with $\mathbf{Q}(t)$ and $\mathbf{R}(t)$ replaced by $\mathbf{W}(t)$ and $\mathbf{V}(t)$. As described in Section 6.6.2., the solution to Eq. (6.28) subject to Eq. (6.25) is

$$\mathbf{M}^*(t;t_1) = -\mathbf{V}^{-1}(t)\mathbf{H}(t)\boldsymbol{\Sigma}(t)\mathbf{X}(t;t_1), \quad (6.29)$$

and

$$\dot{\boldsymbol{\Sigma}}(t) = \mathbf{W}(t) + \mathbf{F}(t)\boldsymbol{\Sigma}(t) + \boldsymbol{\Sigma}(t)\mathbf{F}^T(t) - \boldsymbol{\Sigma}(t)\mathbf{H}^T(t)\mathbf{V}^{-1}(t)\mathbf{H}(t)\boldsymbol{\Sigma}(t) \quad (6.30)$$

where $\boldsymbol{\Sigma}(t_0)$ is the known initial condition. These are the Kalman gain and Ricatti matrix equations given in Eqs. (6.23) and (6.24). From Eq. (6.27) and (6.29), the optimal state estimate is

$$\Delta\hat{\mathbf{x}}^*(t_1) = \int_{t_0}^{t_1}\mathbf{X}^T(t;t_1)\boldsymbol{\Sigma}(t)\mathbf{H}^T(t)\mathbf{V}^{-1}(t)\Delta\mathbf{z}(t)dt.$$

Differentiating this expression yields

$$\frac{d}{dt_1}\Delta\hat{\mathbf{x}}^*(t_1) = \int_{t_0}^{t_1}\frac{d}{dt_1}\mathbf{X}^T(t;t_1)\boldsymbol{\Sigma}(t)\mathbf{H}^T(t)\mathbf{V}^{-1}(t)\Delta\mathbf{z}(t)dt + \boldsymbol{\Sigma}(t_1)\mathbf{H}^T(t_1)\mathbf{V}^{-1}(t_1)\Delta\mathbf{z}(t_1).$$

From Eq. (6.25) and (6.29), it can be shown that

$$\frac{d}{dt_1}\mathbf{X}^T(t;t_1) = [\mathbf{F}(t_1) - \boldsymbol{\Sigma}(t_1)\mathbf{H}^T(t_1)\mathbf{V}^{-1}(t_1)\mathbf{H}(t_1)]\mathbf{X}^T(t;t_1).$$

By combining these last two equations, one arrives at the solution given in Eq. (6.22).

6.8. Linear Quadratic Gaussian (LQG) Control

With this background theory on LQR control (deterministic control) and Kalman filtering (stochastic state estimation), one can solve the stochastic control problem. The derivation of the LQG controller will not be presented in this section, but the separation property and the certainty equivalence property will be used to state the control law. The separation property says that if a first-degree

expansion is adequate to describe perturbation dynamics and if the corresponding variational cost function is adequately modeled by a quadratic function of the state and control, then the control and estimation logic can be derived separately and the stochastic optimal control profile can be expressed as

$$\Delta\mathbf{u}(t_0,t) = \Upsilon\{\Delta\mathbf{x}(t_0,t)\}.$$

The certainty-equivalence property says that $\Upsilon\{\bullet\}$ will be the same control function as the deterministic optimal control function (e.g. LQR).

The objective of a continuous-time LQG controller is to minimize the cost function

$$J = \frac{1}{2}E\left\{\Delta\mathbf{x}^T(t_f)\mathbf{P}(t_f)\Delta\mathbf{x}(t_f) + \int_0^{t_f}\begin{bmatrix}\Delta\mathbf{x}^T(t) & \Delta\mathbf{u}^T(t)\end{bmatrix}\begin{bmatrix}\mathbf{Q} & 0\\ 0 & \mathbf{R}\end{bmatrix}\begin{bmatrix}\Delta\mathbf{x}(t)\\ \Delta\mathbf{u}(t)\end{bmatrix}\right\},$$

subject to the linear dynamic constraint

$$\Delta\dot{\mathbf{x}}(t) = \mathbf{F}(t)\Delta\mathbf{x}(t) + \mathbf{G}(t)\Delta\mathbf{u}(t) + \mathbf{w}(t)$$

and the observation equation

$$\Delta\mathbf{z}(t) = \mathbf{H}(t)\Delta\mathbf{x}(t) + \mathbf{v}(t).$$

Assume both state and observation noise are *white, zero-mean Gaussian* random sequences that are uncorrelated with each other, so

$$\begin{aligned}E\{\mathbf{w}(t)\} &= 0\\ E\{\mathbf{v}(t)\} &= 0\\ E\{\mathbf{w}(t)\mathbf{w}(\tau)^T\} &= \mathbf{W}(t)\delta(t-\tau)\\ E\{\mathbf{v}(t)\mathbf{v}(\tau)^T\} &= \mathbf{V}(t)\delta(t-\tau)\end{aligned}$$

where

$$\delta(t-\tau) = \begin{cases}1 & t=\tau\\ 0 & t\neq\tau\end{cases}$$

$$E\{\mathbf{v}(t)\mathbf{w}(\tau)^T\} = 0 \text{ for all } t \text{ and } \tau.$$

Also, assume that the initial state mean value and its covariance are known and given by

$$\begin{aligned}E\{\Delta\mathbf{x}(t_0)\} &= \Delta\hat{\mathbf{x}}(t_0)\\ E\{(\Delta\mathbf{x}(t_0) - \Delta\hat{\mathbf{x}}(t_0))(\Delta\mathbf{x}(t_0) - \Delta\hat{\mathbf{x}}(t_0))^T\} &= \mathbf{\Sigma}(t_0).\end{aligned}$$

The optimal stochastic control is given by the LQR equations

$$\Delta \mathbf{u}^*(t) = -\mathbf{C}(t)\Delta \mathbf{x}(t)$$

where

$$\mathbf{C}(t) = \mathbf{R}^{-1}(t)\mathbf{G}^T(t)\mathbf{P}(t),$$

and

$$\dot{\mathbf{P}}(t) = -\mathbf{Q}(t) - \mathbf{F}^T(t)\mathbf{P}(t) - \mathbf{P}(t)\mathbf{F}(t) + \mathbf{P}(t)\mathbf{G}(t)\mathbf{R}^{-1}(t)\mathbf{G}^T(t)\mathbf{P}(t).$$

The control gain $\mathbf{C}(t)$ and Ricatti matrix $\mathbf{P}(t)$ should be computed off-line, by working backwards starting with the known $\mathbf{P}(t_f)$. Computing off-line is fine if the system remains close to the optimal solution. If it does not, one should compute them on-line by using the dynamic programming approach. Unfortunately, this is not always an easy task to perform in real-time!

The state estimate is given by the Kalman-Bucy filter equations

$$\Delta \dot{\hat{\mathbf{x}}}(t) = \mathbf{F}(t)\Delta \hat{\mathbf{x}}(t) + \mathbf{G}(t)\Delta \mathbf{u}(t) + \mathbf{K}(t)\left[\Delta \mathbf{z}(t) - \mathbf{H}(t)\Delta \hat{\mathbf{x}}(t)\right]$$

where

$$\mathbf{K}(t) = \mathbf{\Sigma}(t)\mathbf{H}^T(t)\mathbf{V}^{-1}(t),$$

and

$$\dot{\mathbf{\Sigma}}(t) = \mathbf{F}(t)\mathbf{\Sigma}(t) + \mathbf{\Sigma}(t)\mathbf{F}^T(t) + \mathbf{W}(t) - \mathbf{K}(t)\mathbf{V}(t)\mathbf{K}^T(t)$$

With enough computing power, these equations could to be computed on-line since the initial condition on the covariance $\mathbf{\Sigma}$ is at t_0. Instead of integrating the above equation, the discrete time version of the Kalman filter is typically used[25]. A block diagram of the continuous time LQG controller is shown below in Figure 6.35.

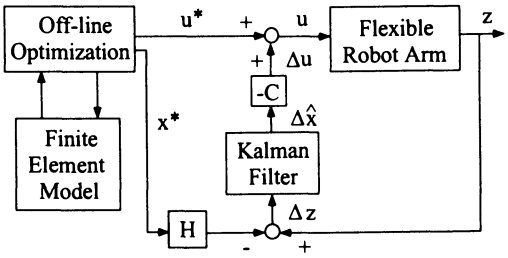

Figure 6.35. Continuous-time linear quadratic gaussian (LQG) controller

6.8.1. Experimental Results

This section describes the experimental results of the LQG control scheme applied to a two-link flexible robot[25]. An off-line optimization routine determined a minimum-time, straight-line tip trajectory that stays within the torque constraints of the motors and ends with no vibrational transients (see Section 5.4.10.). An efficient finite element model is used in the optimization to approximate the flexible arm dynamics (see Section 3.4.). The control strategy described in the previous section is used to determine the feedback gains for the position, velocity, and strain gauge signals from a quadratic cost criterion based on the finite element model linearized about the straight-line tip trajectory. These feedback signals are added to the modeled torque values obtained from the optimization routine and used to control the robot arm actuators. The results indicate that this combination of model-based and error-driven control strategies achieves a closer tracking of the desired trajectory and a better handling of modeling errors than either strategy alone.

A finite element model[26] (see Section 3.4.) is used to obtain a set of nonlinear ordinary differential equations. In the finite element model of the two-link flexible arm, the structure was divided into 9 elements and 2 hinge joints. Recall, Figure 5.24 shows the two-link manipulator and Figure 6.36 illustrates the geometry of the structure. The first node corresponds to the first joint or hub of the manipulator.

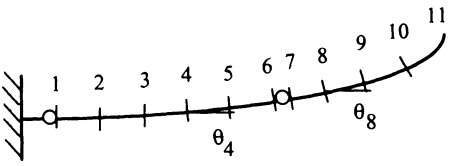

Figure 6.36. Finite element model geometry of the arm

The second node corresponds to the outer edge of the first bracket. The next three elements represent the first link divided into three equal segments with each segment having its own strain gauge. The next three nodes correspond to the second bracket, second joint, and third bracket. Finally, the last three elements are the second link divided into three equal segments, again with each segment having its own strain gauge. The brackets are modeled as very stiff links with large stiffness terms EI (Young's modulus times the area moment of inertia). The motion of the manipulator is assumed to take place entirely within the horizontal plane.

The dynamic equations of the finite element model can be compactly written in the following form:

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}^2 + \mathbf{K}(\boldsymbol{\theta})\mathbf{I}_1 = \mathbf{B}\mathbf{u} \tag{6.31}$$

where $\boldsymbol{\theta}$ is a column vector containing the angular positions of the 11 finite element nodes, and $\mathbf{u} = [\tau_1 \ \tau_2]^T$ is the external torque vector where τ_1 and τ_2 are the applied torques at the joints. The term $\dot{\boldsymbol{\theta}}^2$ is an 11x1 vector where each element is the square of $\dot{\theta}_i$. The mass matrix is $\mathbf{M}(\boldsymbol{\theta}) = [M_{i,j}\cos(\theta_i - \theta_j)]$, $i = 1,\ldots,11$, $j = 1,\ldots,11$, the centrifugal matrix is $\mathbf{C}(\boldsymbol{\theta}) = [M_{i,j}\sin(\theta_i - \theta_j)]$, $i = 1,\ldots,11$, $j = 1,\ldots,11$, and the stiffness matrix is $\mathbf{K}(\boldsymbol{\theta}) = [-K_{i,j}\sin(\theta_i - \theta_j)]$, $i = 1,\ldots,11$, $j = 1,\ldots,11$. The $M_{i,j}$ and $K_{i,j}$ in these equations are the mass and stiffness constants that are derived from the finite element model[26] and remain unchanged for a given structure (i.e., these elements do not depend on joint angles and link curvatures). The symbol $\mathbf{I}_1$ refers to a column vector of 11 ones. The matrix $\mathbf{B}$ is an 11x2 matrix whose purpose is to transform the 2x1 control vector $\mathbf{u}$ into the 11x1 angular acceleration space. It is defined as

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T .$$

The reader should note that the θ_i are absolute angles in the inertial coordinate system. Because of this absolute referencing, Eq. (6.31) does not contain Coriolis terms ($\dot{\theta}_i\dot{\theta}_j$, $i \neq j$).

This model is efficient from a computational point of view since it only requires one evaluation of the mass and stiffness elements $M_{i,j}$ and $K_{i,j}$, respectively. It also has proven to be quite accurate in modeling the behavior of the actual manipulator. Experimental results[13] show that the fundamental modal frequencies of the finite element model closely approximate those of the experimental apparatus (approx. 20 Hz and 8 Hz for the first and second links respectively). In addition, the torque, angular velocity, and joint position profiles obtained by the model closely approximate those of the experimental structure for corresponding maneuvers.

In order to apply LQG regulator theory, the dynamics in Eq. (6.31) must be put into state-space form and then linearized about the desired trajectory. The state variables are chosen to be the 11 nodal positions and the 11 nodal velocities

$$x_i = \theta_i \,, i = 1,\ldots,11 \,, \quad x_{i+11} = \dot{\theta}_i \,, i = 1,\ldots,11$$

where x_i are the new state variables. Rewriting Eq. (6.31) in state-space form gives

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_{11} \\ \cdots \\ \dot{x}_{12} \\ \vdots \\ \dot{x}_{22} \end{bmatrix} = \begin{bmatrix} x_{12} \\ \vdots \\ x_{22} \\ \cdots \\ \mathbf{M}(\mathbf{x})^{-1}\mathbf{S}(\mathbf{x}) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ \cdots & \\ \mathbf{M}(\mathbf{x})^{-1}\mathbf{B} \end{bmatrix} \mathbf{u},$$

or more compactly as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} \,. \tag{6.32}$$

The matrix $\mathbf{M}(\mathbf{x})$ is simply the matrix $\mathbf{M}(\boldsymbol{\theta})$ with the θ_i's replaced with the appropriately numbered states x_i. The control vector $\mathbf{u}$ is as described earlier (see Eq. 6.31), and the matrix $\mathbf{S}(\mathbf{x})$ is the matrix $\mathbf{S}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = -\mathbf{C}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}^2 - \mathbf{K}(\boldsymbol{\theta})\mathbf{I}_1$ with $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$ replaced by the state vector $\mathbf{x}$.

The state-space formulation described in Eq. (6.32) is a 22nd-order nonlinear ordinary differential equation. To perform the linearization about a reference trajectory, let $\Delta\mathbf{x}$ represent the perturbation in the state $\mathbf{x}$ about the operating point $\mathbf{x}^*$, and $\Delta\mathbf{u}$ represent the perturbation in the control input $\mathbf{u}$ about the operating point $\mathbf{u}^*$, or

$$\mathbf{x} = \mathbf{x}^* + \Delta\mathbf{x} \quad \text{and} \quad \mathbf{u} = \mathbf{u}^* + \Delta\mathbf{u}.$$

These operating points $\mathbf{x}^*$ and $\mathbf{u}^*$ are those from the off-line optimization presented in Figures 6.25, 6.28 ,and 6.29. Truncating second and higher-order terms gives the following linearized model:

$$\begin{aligned} \Delta\dot{\mathbf{x}} &= \left(\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*} + \left.\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*} \mathbf{u}^* \right) \Delta\mathbf{x} + \mathbf{g}(\mathbf{x}^*)\Delta\mathbf{u} \\ &= \mathbf{F}\Delta\mathbf{x} + \mathbf{G}\Delta\mathbf{u} \end{aligned} \tag{6.33}$$

where $\Delta\mathbf{x}$ represents deviations from the desired trajectory which one wants to drive to zero.

Next, the states are transformed into a more measurable form. In this system, joint encoders and tachometers are used to measure the angular position and velocity of each joint node with respect to the previous node. Also strain gauges on the links are used to measure the angular position of a node with respect to the previous node. Since all position and velocity measurements of one node are relative to the previous node, it is desirable to transform the state vector by a 22x22 matrix $\mathbf{T}$ such that

$$\Delta\tilde{\mathbf{x}} = \mathbf{T}\Delta\mathbf{x}$$

where

$$T_{i,j} = \begin{cases} 1 & \text{if } i = j \\ -1 & \text{if } i = j - 1 = 2, 3, ..., 11, 13, 14, ..., 22 \\ 0 & \text{otherwise} \end{cases}$$

Using $\Delta\tilde{\mathbf{x}}$ as the new state variable changes Eq. (6.33) to

$$\Delta\dot{\tilde{\mathbf{x}}} = \mathbf{T}\mathbf{F}\mathbf{T}^{-1}\Delta\tilde{\mathbf{x}} + \mathbf{T}\mathbf{G}\Delta\mathbf{u} = \tilde{\mathbf{F}}\Delta\tilde{\mathbf{x}} + \tilde{\mathbf{G}}\Delta\mathbf{u}. \tag{6.34}$$

As discussed earlier, the linear quadratic regulator portion of this problem is to minimize a quadratic cost criterion

$$J = \frac{1}{2}\int_0^\infty (\Delta\tilde{\mathbf{x}}^T\mathbf{Q}\Delta\tilde{\mathbf{x}} + \Delta\mathbf{u}^T\mathbf{R}\Delta\mathbf{u})dt,$$

with respect to the control vector $\Delta\mathbf{u}$ and subject to the system's dynamics in Eq. (6.34). The choice of weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ are up to the system designer. The solution to this problem involves solving an algebraic Riccati equation[27] for a matrix $\mathbf{P}$

$$\mathbf{0} = \tilde{\mathbf{F}}^T\mathbf{P} + \mathbf{P}\tilde{\mathbf{F}} - \mathbf{P}\tilde{\mathbf{G}}\mathbf{R}^{-1}\tilde{\mathbf{G}}^T\mathbf{P} + \mathbf{Q},$$

which results in the feedback equation

$$\Delta\mathbf{u} = -\mathbf{R}^{-1}\tilde{\mathbf{G}}^T\mathbf{P}\Delta\tilde{\mathbf{x}}. \tag{6.35}$$

Thus, the optimal *full* state feedback gain matrix is $\mathbf{C} = -\mathbf{R}^{-1}\tilde{\mathbf{G}}^T\mathbf{P}$.

Since one is unable to measure all of the states, a linear quadratic estimator design is used to determine an estimator gain matrix $\mathbf{K}$ that produces a linear quadratic Gaussian optimal estimate of $\Delta\tilde{\mathbf{x}}$ from the measurements $\Delta\mathbf{z}$. The problem is formally stated as follows: Given a continuous-time system with state and measurement equations

$$\begin{aligned}\Delta\dot{\tilde{\mathbf{x}}} &= \tilde{\mathbf{F}}\Delta\tilde{\mathbf{x}} + \tilde{\mathbf{G}}\Delta\mathbf{u} + \mathbf{w}\\ \Delta\mathbf{z} &= \mathbf{H}\Delta\tilde{\mathbf{x}} + \mathbf{v},\end{aligned}$$

and process and measurement noise means and covariances

$$E[\mathbf{w}] = E[\mathbf{v}] = 0, \quad E[\mathbf{w}\mathbf{w}^T] = \mathbf{W}, \quad E[\mathbf{v}\mathbf{v}^T] = \mathbf{V},$$

the optimal state estimate $\Delta\hat{\mathbf{x}}$ is found by integrating

$$\Delta\dot{\hat{\mathbf{x}}} = \tilde{\mathbf{F}}\Delta\hat{\mathbf{x}} + \tilde{\mathbf{G}}\Delta\mathbf{u} + \mathbf{K}[\Delta\mathbf{z} - \mathbf{H}\Delta\hat{\mathbf{x}}], \quad \Delta\hat{\mathbf{x}}(0) = \mathbf{0} \tag{6.36}$$

where

$$\mathbf{K} = \mathbf{\Sigma}\mathbf{H}^T\mathbf{V}^{-1}, \tag{6.37}$$

and $\mathbf{\Sigma}$ satisfies the algebraic Riccati equation

$$\mathbf{0} = \tilde{\mathbf{F}}\mathbf{\Sigma} + \mathbf{\Sigma}\tilde{\mathbf{F}}^T + \mathbf{W} - \mathbf{K}\mathbf{V}\mathbf{K}^T.$$

Similar to the linear quadratic gain in Eq. (6.35), the gain in Eq. (6.37) can be determined off-line and the gains scheduled for on-line control. However, at present there is insufficient on-line computing power in this system to solve the integration in Eq. (6.36) in real-time–even the discrete version is computationally intensive. If the integration in Eq. (6.36) is approximated in discrete time[14] as

$$\begin{aligned}\Delta\hat{\mathbf{x}}_k \approx \Delta\hat{\mathbf{x}}_{k-1} + [\tilde{\mathbf{F}}\Delta\hat{\mathbf{x}}_{k-1} + \tilde{\mathbf{G}}\Delta\mathbf{u}_{k-1}]\Delta t + \mathbf{K}_k\{\Delta\mathbf{z}_k - \mathbf{H}[\Delta\hat{\mathbf{x}}_{k-1}\\ + (\tilde{\mathbf{F}}\Delta\hat{\mathbf{x}}_{k-1} + \tilde{\mathbf{G}}\Delta\mathbf{u}_{k-1})\Delta t]\},\end{aligned} \tag{6.38}$$

one can reduce Eq. (6.38) to

$$\Delta\hat{\mathbf{x}}_k \approx \mathbf{K}_k \Delta\mathbf{z}_k \tag{6.39}$$

when the sampling time Δt is small and the previous estimated state $\Delta\hat{\mathbf{x}}_{k-1}$ reaches the desired state $\Delta\hat{\mathbf{x}}_{k-1} = 0$. Combining Equations (6.35) and (6.39) yields the approximate feedback gain at time k as $\mathbf{C}_k\mathbf{K}_k$.

MATLAB® was used to determine the approximate Kalman filter gain matrices $\mathbf{K}_k$ and the optimal feedback gain matrices $\mathbf{C}_k$ for one hundred set points along the optimal trajectory. At some points along the trajectory, the gains varied considerably (see Figure 6.37). When the set of gains was used to control the system, the results were less than satisfactory. Although the arm did reach the desired end point, there was considerable error in the tip position along the way. It appeared that some of the larger scheduled gains produced unwanted effects. Again, one must remember that these gains are computed from a model of the system, so the gains are only as good as the model itself. In an attempt to reduce the sensitivity of the feedback gains to modeling errors, a single gain matrix equal to the average of the time sequence of matrices was used. Figure 6.38 shows the resulting tip trajectory. The averaging process removed the abrupt changes in gains along the path. Although further testing is necessary, it is believed that an even better solution would be to use a set of three or four gains that would be scheduled to become active when major changes in the states occurred, e.g., when the applied torque on joint 1 changed from -6 N-m to +6 N-m.

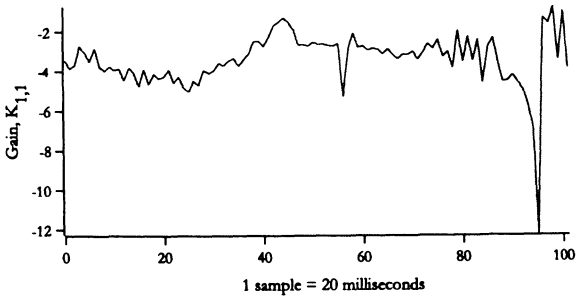

Figure 6.37: Feedback gain $K_{1,1}$ (contribution of joint 1 position error on joint 1 motor).

In the LQG design, the 10x22 observation matrix $\mathbf{H}$, was

$$H_{i,j} = \begin{cases} 1 & \text{if } (i,j) = (1,1),(5,7) \text{ (joint angles)} \\ 1 & \text{if } i = j-1 = 2,3,4 \text{ (strain gauges)} \\ 1 & \text{if } i = j-3 = 6,7,8 \text{ (strain gauges)} \\ 1 & \text{if } (i,j) = (9,12),(10,18) \text{ (joint velocities)} \\ 0 & \text{otherwise} \end{cases}$$

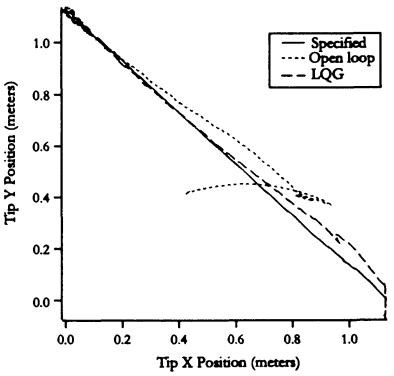

Figure 6.38. Specified, open loop, and LQG tip positions.

where i denotes the observation vector index and j denotes state vector index. The state and input weighting matrices were

$$Q_{i,j} = \begin{cases} 1000 & \text{if } i = j = 1,7 \text{ (joint angles)} \\ 0.1 & \text{if } i = j = 12,18 \text{ (joint velocities)} \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{R} = \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix}.$$

The state and measurement noise matrices were

$$W_{i,j} = \begin{cases} 0.01 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases},$$

and

$$\mathbf{V} = \begin{bmatrix} 0.1 & \text{if } i = j = 1,5 \text{ (joint angles)} \\ 1000 & \text{if } i = j = 2,3,4,6,7,8 \text{ (strain gauges)} \\ 5 \times 10^6 & \text{if } i = j = 9,10 \text{ (joint velocities)} \\ 0 & \text{otherwise} \end{bmatrix}.$$

In this experiment, $\mathbf{Q}$ and $\mathbf{V}$ were chosen so that the majority of the feedback came from the joint encoders and tachometers. This was done for two reasons. First, the strain gauges are slightly noisier than the joint encoders and tachometers. Second, for the specified straight-line slew, the tip position of the arm was dependent more on the joint positions than on the curvature of the links. It was found that penalizing the link curvature states often resulted in instability because

the controller put too much emphasis on the errors between the measured and modeled link curvatures. It is believed that the main problem was that the linearized model was not accurate enough to properly drive the link curvature measurements to those values determined by the model. With improved parameter estimation, it is likely that one would be able to put more emphasis on these curvature states.

Based on the experimental results, a promising strategy is to obtain an initial set of gains via the LQG method that is optimal for the given model and then improve on these gains by hand-tuning them on the physical structure. This combination strategy will converge to a desired performance level quicker than either the LQG method or hand-tuning by itself.

This method of control would be best applied to manipulators performing repetitive motion, such as assembly robots or high bandwidth micro-manipulators (e.g., disk drive positioning arms). While the optimization is time consuming, it would be cost effective if the motion was to be repeated hundreds of times.

6.9. Chapter 6 Summary

This chapter reviewed a general class of linear control design approaches for flexible dynamic systems. This class of linear control systems included both classical and modern control algorithms. The classical PD control system design was demonstrated for gantry robot problems. Next, an atypical control algorithm based on lag stabilization was introduced that demonstrated reduction in oscillations for a flexible system. In addition, a non-collocated control problem was reviewed. This problem showed non-minimum phase characteristics associated with flexible robotics systems. The modern control techniques included the development of LQR, linear optimal estimation, and LQG algorithms. Both feedforward and LQG control techniques were validated experimentally for a planar flexible robot arm. In the remaining chapters (see Chapters 7 and 8) advanced nonlinear control techniques are introduced in order to handle systems that cannot be effectively linearized.

6.10. Chapter 6 References

1. J. F. Jones, B. J. Petterson, and J. C. Werner, "Swing damped movement of suspended objects," Sandia Report 87-2189, Reprinted February 1993.
2. B. C. Kuo, *Automatic Control Systems*, 4th Ed., Prentice-Hall, 1982.
3. B. J. Petterson, R. D. Robinett, and J. C. Werner, "Lag-stabilized force feedback damping," SAND91-0194, May 1991.
4. J. Roskam, *Airplane Flight Dynamics and Automatic Flight Controls: Part II*, Roskam Aviation and Engineering Corporations, Lawrence, Kansas, 1979.
5. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Univ. Press, Cambridge, 1988.
6. R. D. Robinett, B. J. Petterson, and J. C. Fahrenholtz, "Lag-stabilized force feedback damping," Journal of Intelligent and Robotic Systems, 21: 277-285, 1998.
7. C. Abdallah, P. Dorato, J. Benitez-Read, and R. Byrne, "Delayed positive feedback

can stabilize oscillatory systems," Proceeding of the American Control Conference, San Francisco, CA, pp. 3106-7, 1993.

8. R. H., Jr. Cannon, and Schmitz, E., "Initial experiments on the end-point control of a flexible one-link robot," International Journal of Robotics Research, 3(3):62-75, 1984.
9. C. M. Oakley, and R. H. Cannon, "Anatomy of an experimental two-link flexible manipulator under end-point control," In proceedings of the IEEE Conference on Decision and Control, Honolulu, HI, Dec. 1990.
10. C. M. Oakley, and C. H. Barratt, "End-point controller design for an experimental two-link flexible manipulator using convex optimization," In proceedings of the American Control Conference, pages 1752-1759, San Diego, CA, May 1990.
11. P. A. Chodovarapu, and M. W. Spong, "On non-collocated control of a single flexible link," Proceedings of the 1996 IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, April 1996, pp. 1101-1106.
12. M. Karkoub, and K. Tamma, "Modelling and μ-synthesis control of flexible manipulators," Computers and Structures 79 (2001), pp. 543-551.
13. J. T. Feddema, G. R. Eisler, and D. J. Segalman, "Integration of Model-Based and Sensor-Based Control of a Two-link Flexible Robot Arm," *Proceedings of the 1990 IEEE Conference on Systems Engineering*, Pittsburgh, PA, August 9-11, 1990.
14. R. F. Stengel, *Stochastic Optimal Control: Theory and Application*, John Wiley & Sons, Inc., 1986.
15. P. S. Maybeck, *Stochastic Models, Estimation, and Control, Volume 1*, Academic Press, New York, 1979.
16. A. E. Bryson, and Y. Ho, *Applied Optimal Control*, Hemisphere Publishing Corporation, Washington, D.C., 1975.
17. B. D. O. Anderson, and J. B. Moore, *Optimal Control: Linear Quadratic Methods*, Prentice-Hall, New Jersey, 1990.
18. E. D. Sontag, *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, Springer-Verlag, New York, 1990.
19. F. L. Lewis, *Optimal Estimation: With an Introduction to Stochastic Control Theory*, John Wiley & Sons, Inc., New York, 1986.
20. K. J. Astrom, *Introduction to Stochastic Control Theory*, Academic Press, New York, 1970.
21. B. D. O. Anderson, and J. B. Moore, *Optimal Filtering*, Prentice-Hall Information and System Sciences Series, 1979.
22. J. L. Junkins, *An Introduction to Optimal Estimation of Dynamic Systems*, Sijhhof & Noordhoff International Publishers, The Netherlands, 1978.
23. R. E. Kalman, and R. S. Bucy, "New results in linear filtering and prediction theory," *Transactions of ASME, Journal of Basic Engineering*, Vol. 83, pp. 95-107, 1961.
24. P. A. Ruymgaart, and T. T. Soong, *Mathematics of Kalman-Bucy Filtering*, Springer-Verlag, New York, 1988.
25. D. A. Schoenwald, J. T. Feddema, G. R. Eisler, and D. J. Segalman, "Minimum-Time Trajectory Control of Two-Link Flexible Robotic Manipulator," *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 2114-2120.
26. D. J. Segalman, "A mathematical formulation for the rapid simulation of a flexible multilink manipulator," SAND89-2308, Sandia National Laboratories, Albuquerque, NM.

27. T. J. Tarn, A. K. Bejczy, and X. Ding, "On the modelling of flexible robot arms," Department of Systems Science and Mathematics, Washington University, St. Louis, MO, Robotics Laboratory Report SSM-RL-88-11, September 1988.

6.11. Chapter 6 Problems

Homework 6.1. From Section 6.2.
Validate the PD controller design discussed in Section 6.2. for the gantry robot by performing a numerical simulation. Plot the disturbance impulse response and resulting control in acceleration and in velocity responses.

Homework 6.2. From Section 6.2.
Add a position feedback loop as shown in Figure 6.8 to the previously designed PD controller of Homework 6.1. By using the same values for K_P and K_D perform a root locus design. Select a K_{P_x} that results in critically damped responses for both $\theta(t)$ and $X(t)$, respectively. Plot the time domain responses. For what value of K_{P_x} does the response become unstable?

Homework 6.3. From Section 6.3.
Validate the lag-stabilized design discussed in Section 6.3. for the gantry robot problem by performing a numerical simulation. Use the parameter values presented in Table 6.1. Plot the angular position for both marginally stable and stable critically damped responses. For what value of ω, varying from ω_n does the assumption that ω *needs to be close to* ω_n become invalid?

Homework 6.4. From Section 6.4.
Given the equations of motion for a slewing flexible link model that includes only a single flexible mode, one can obtain the following dynamic model:

$$I_0\ddot{\theta} + I_1\ddot{q} + C\dot{\theta} = \tau,$$

and

$$I_1\ddot{\theta} + I_2\ddot{q} + c\dot{q} + kq = 0.$$

a. Following the procedures given in Section 6.4., derive all the necessary transfer functions.

b. Perform a root-locus design by using the PD control transfer function and only the collocated transfer function for θ. In addition, plot the tip deflection response.

c. Perform a root-locus design as in b., but use the non-collocated transfer function for $\theta + \phi(L)q$. What do you observe? Is using conventional PD control (low-order compensation) sufficient to control the tip?

Homework 6.5. From Section 6.5.
As seen in Section 2.7.10., the time-optimal control for a rigid single link arm is the

bang-bang control input

$$u(t) = \begin{cases} +u_b & 0 \le t \le 0.5t_f \\ -u_b & 0.5t_f < t \le t_f \end{cases}$$

where u_b is the maximum torque input, the final time is $t_f = 2\sqrt{ml^2x_{1_f}/3u_b}$, m is the link mass, l is the link length, and x_{1_f} is the desired final angular position of the link. Assume that this time-optimal control was designed with $m = 0.5, l = 1$, and $x_{1_f} = \pi/2$; however, in the experiments the true link mass is $m = 0.6$.

a. Applying the time-optimal control to the actual link, simulate the open loop response of the system where

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ (\frac{3}{ml^2}) \end{bmatrix} u.$$

b. Add a proportional-derivative feedback loop to control the position of the rigid link. Try to achieve a 5% settling time of 1.25 seconds. Note, the control input will go beyond the maximum torque requirements, therefore the u_b chosen in the initial design should be well within the true maximum torque limits.

c. Remove the time-optimal feedforward path and use only the PD feedback loop designed in b. Is a settling time of 1.25 seconds still achieved?

Homework 6.6. From Section 6.8.
The equations of motion for a single assumed mode of a slewing flexible link is given by

$$M\ddot{\theta} + m\ddot{q} + C\dot{\theta} = \tau$$
$$m\ddot{\theta} + m_2\ddot{q} + c\dot{q} + kq = 0$$

where θ is the joint angle, and q is the perturbation of the flexible link. The parameters $M, m, C, m_2, c,$ and k are constants that depend of the moment of inertia of the hub, density of the link, damping of the joint, modulus of elasticity of the link, dampening of the link, and the length of the link.

a. Write these equations in state-space form where the state vector is $\mathbf{x} = \begin{bmatrix} \theta & q & \dot{\theta} & \dot{q} \end{bmatrix}$. Assume that you are only able to measure the joint angle θ.

b. Design a Kalman filter for estimating the joint angle θ, its velocity $\dot{\theta}$, flexible perturbation q, and its velocity $\dot{q}$. Because it is a linear system, the algebraic Ricatti equation can be used to find a constant Kalman filter gain. Use $M = 20, C = 0.1, c = 0.25, m = 1, m_2,$ and $k = 10$. Let the state and measurement noise matrices be $\mathbf{W} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$ and $\mathbf{V} = \begin{bmatrix} 0.01 \end{bmatrix}$.

c. Design a Linear Quadratic Gaussian Regulator for controlling the system. Again, the algebraic Ricatti equation can be used to find a constant control gain. Let the state and input regulator matrices be $\mathbf{Q} = \begin{bmatrix} 1000 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ and $R = [1000]$.

d. Simulate the performance of the LQG design. For the continuous time case, this can be accomplished by combining the Kalman filter equations with the LQR equations to give

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\hat{\mathbf{x}}} \end{bmatrix} = \begin{bmatrix} \mathbf{F} & \mathbf{GC} \\ \mathbf{KH} & \mathbf{F} + \mathbf{GC} - \mathbf{KH} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \hat{\mathbf{x}} \end{bmatrix}.$$

The first equation is the equation of motion of the system with control input $\mathbf{u} = \mathbf{C}\hat{\mathbf{x}}$ and the state vector is estimated in the second equation by using the Kalman filter. The Kalman filter gain is $\mathbf{K} = \mathbf{\Sigma H}^T\mathbf{V}^{-1}$ where the steady state covariance matrix $\mathbf{\Sigma}$ is determined by solving the algebraic Ricatti equation

$$\mathbf{0} = \mathbf{F\Sigma} + \mathbf{\Sigma F}^T - \mathbf{\Sigma H}^T\mathbf{V}^{-1}\mathbf{H\Sigma} + \mathbf{W}.$$

The control gain is given by the LQR design as $\mathbf{C} = -\mathbf{R}^T\mathbf{G}^T\mathbf{P}$ where the matrix $\mathbf{P}$ satisfies the algebraic Ricatti equation

$$\mathbf{0} = \mathbf{F}^T\mathbf{P} + \mathbf{PF} - \mathbf{PGR}^{-1}\mathbf{G}^T\mathbf{P} + \mathbf{Q}.$$

Chapter 7

Nonlinear Systems and Sliding Mode Control

7.1. Introduction

Many systems of practical interest are nonlinear, but sometimes it is possible to consider small motions about an operating and/or equilibrium point. In this case, a linear set of dynamic equations can be formulated, thus facilitating the use of linear analysis and design techniques. When it is inappropriate to linearize the system, the linear design tools cannot be applied and instead nonlinear analysis is required[1,2]. Two of the more important analyses that are often needed are stability determination and controller design. Several examples are presented to illustrate these situations.

Example 7.1. Simple pendulum

Consider a particle of mass m connected by a massless rod of length l to a frictionless joint at O. A motor at O can apply a torque τ to the rod. The angle between the rod and the vertical is denoted by θ as shown in Figure 7.1. Derive the

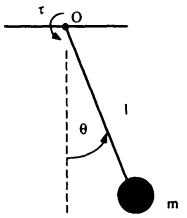

Figure 7.1. Simple pendulum with an applied torque

nonlinear dynamic equations of motion for the system and compare the linearized

R. D. Robinett III et al., *Flexible Robot Dynamics and Controls*

and nonlinear equations.

Solution

To this end, a free body diagram of the system is shown in Figure 7.2. The

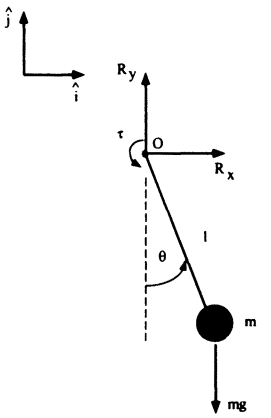

Figure 7.2. Free body diagram of simple pendulum

dynamic equation can be found by summing the external moments about the fixed point O. Using D'Alembert's Principle gives

$$\Sigma \mathbf{M}_O = (l \sin\theta \hat{\mathbf{i}} - l\cos\theta \hat{\mathbf{j}}) \times (-mg\hat{\mathbf{j}}) - ml^2\ddot{\theta}\hat{\mathbf{k}} + \tau\hat{\mathbf{k}} = 0.$$

Applying the cross product and simplifying gives

$$\ddot{\theta} + \frac{g}{l}\sin\theta = \frac{1}{ml^2}\tau. \tag{7.1}$$

Case 7.1. Small θ

If the motion of the particle is small (e.g., $-20° < \theta < 20°$), then Eq. (7.1) can be linearized about $\theta_O = 0$ resulting in

$$\ddot{\theta} + \frac{g}{l}\theta = \frac{1}{ml^2}\tau. \tag{7.2}$$

Since Eq. (7.2) is linear in θ and $\ddot{\theta}$, the Laplace transform can be applied and its transfer function found as

$$\frac{\theta(s)}{\tau(s)} = \frac{\frac{1}{ml^2}}{s^2 + \frac{g}{l}}. \tag{7.3}$$

Equation (7.3) can form the basis for assessing stability and controller design by using linear methods.

Case 7.2. Large θ

Since the angular motion is large, (e.g., $|\theta|_{\max} > 20°$), Eq. (7.1) must be used. Unfortunately, the Laplace transform does not apply because of the nonlinear term $\sin\theta$ and a transfer function is inappropriate. Nevertheless, methods for assessing stability and designing controllers are needed.

In the remainder of this chapter definitions are introduced leading to a discussion of nonlinear system stability. Sliding Mode Control (SMC) will be presented as one example of a systematic nonlinear controller design technique.

7.2. State-Space Representation of a Dynamic System

The development of a system's dynamic equations often results in one or more higher-order ordinary differential equations. An equivalent representation can be created consisting of a set of first-order differential equations. This is known as the state-space representation of the system and the n first-order equations are called the state equations. The independent variables (functions of time) are called the state variables.

Example 7.2. For a nonlinear pendulum find the state variables and the state equations.

Solution

From the previous example (see Example 7.1), a single second-order dynamic equation was developed as

$$\ddot{\theta} + \frac{g}{l}\sin\theta = \frac{1}{ml^2}\tau.$$

Together with the initial conditions $\theta(t_i) = \theta_i$ and $\dot{\theta}(t_i) = \dot{\theta}_i$, and the torque history the solution could be obtained. To convert this to state-space form, first define two independent state variables

$$\begin{aligned} x_1(t) &= \theta(t) \\ x_2(t) &= \dot{\theta}(t). \end{aligned} \tag{7.4}$$

The state equations are then formed by taking the time derivative of Eq. (7.4) and expressing the resulting equations in terms of the states $x_1(t)$ and $x_2(t)$

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -\frac{g}{l}\sin[x_1(t)] + \frac{1}{ml^2}\tau(t). \end{aligned} \tag{7.5}$$

Now, there are two first-order equations instead of one second-order equation. The initial conditions for Eq. (7.5) would similarly become $x_1(t_i) = \theta_i$ and $x_2(t_1) = \dot{\theta}_i$.

One distinguishing feature,of a dynamic system is its order. For a system represented in state-space form, the order is simply the number of first-order differential equations.

Later, in this chapter to assess stability, it will be necessary to consider state-space trajectories. That is, a plot of the state variable values during the evolution of the state equations to general forcing functions and initial conditions. This is easy to visualize for second-order systems, since the trajectory is planar as shown by the example in Figure 7.3. For higher-order systems the concept is valid, but difficult to visualize.

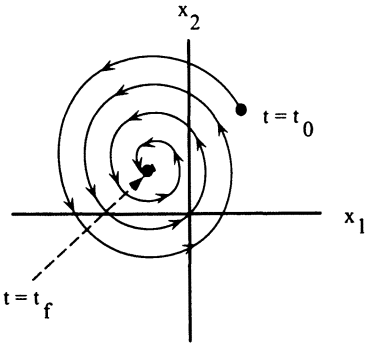

Figure 7.3. State-space trajectory

7.2.1. State Nonlinearities

A differential equation is linear if the state variables and their derivatives appear only to the first power, are not multiplied by other state variables, and do not appear as arguments of trigonometric functions. Some examples of state variables are

RCL Circuit	$\dot{x}_1 = -\frac{1}{RC}x_1 - \frac{1}{C}x_2$ $\dot{x}_2 = \frac{1}{L}x_1$	LINEAR
Large Motion Pendulum	$\dot{x}_1 = x_2$ $\dot{x}_2 = -\frac{K}{M}\sin(x_1) + \tau$	NONLINEAR
Van der Pol Oscillator	$\dot{x}_1 = x_2$ $\dot{x}_2 = -x_1 - x_1^2 x_2 + \mu x_2$	NONLINEAR.

7.2.2. Equilibrium Points

An equilibrium point is a point in state-space where the state derivatives are all zero when the input is zero. If a system has its initial condition at an equilibrium point, then it will not move from that point unless the input is changed from zero.

Example 7.3. For a large motion pendulum and the system's state equations

$$\begin{aligned}\dot{x}_1(t) &= x_2(t)\\ \dot{x}_2(t) &= -\frac{g}{l}\sin[x_1(t)] + \frac{1}{ml^2}\tau(t),\end{aligned}$$

find the equilibrium points.

Solution

First, set the input, $\tau(t)$ to zero. Next, set $\dot{x}_1(t)$ and $\dot{x}_2(t)$ to zero and solve for $x_1(t)$ and $x_2(t)$ as

$$\begin{aligned}\dot{x}_1(t) &= x_2(t) = 0\\ \dot{x}_2(t) &= -\frac{g}{l}\sin[x_1(t)] = 0.\end{aligned} \tag{7.6}$$

From Eq. (7.6) we see that there are an infinite number of solutions,

$$\begin{aligned}x_1(t) &= \pi \cdot n \qquad n = 0, \pm 1, \pm 2, \ldots\\ x_2(t) &= 0.\end{aligned}$$

However, physically, there are just two

$$x_1 = x_2 = 0, \quad \text{and} \quad \begin{aligned}x_1 &= \pi\\ x_2 &= 0,\end{aligned} \tag{7.7}$$

shown graphically in Figure 7.4.

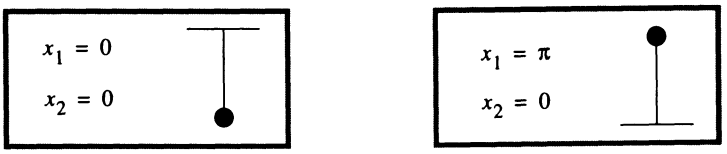

Figure 7.4. Pendulum equilibrium points

For definition purposes, equilibrium points are typically used as operating points when designing linear and nonlinear control systems. Also, equilibrium points are

used as the beginning and ending points of input shaping profiles. Equilibrium points will be considered to occur at $t = t_0$ and will be denoted as $\mathbf{x}_0$, that is

$$\mathbf{x}_0 = [x_1(t_0) \quad x_2(t_0) \ldots x_n(t_0)]^T .$$

7.3. Stability

The stability of a dynamic system is typically defined within the context of its equilibrium points. If a system has two equilibrium points, it may be that one equilibrium point is stable and the other equilibrium point is unstable.

By definition, the equilibrium point $\mathbf{x}_0$ is stable if for all specified regions R_1 about $\mathbf{x}_0$ there exists a corresponding region R_2 about $\mathbf{x}_0$ such that if the initial condition state is within R_2, then the resulting state motion stays within R_1 (see Figure 7.5). This type of stability, called stability in the sense of Lyapunov, is the basis of Lyapunov's direct method. Figure 7.6 graphically presents several types of stability.

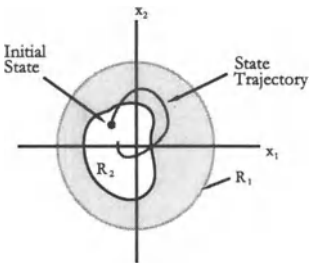

Figure 7.5: Equilibrium point stability

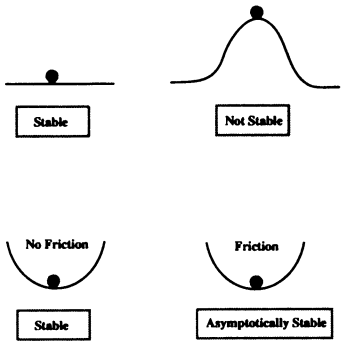

Figure 7.6: Several stability examples

7.3.1. Stability Determination - Lyapunov's Direct Method

Before stating the mathematical aspects of this approach, two simple examples are presented.

Example 7.4. Consider the parallel RLC circuit shown in Figure 7.7 with a state-space representation

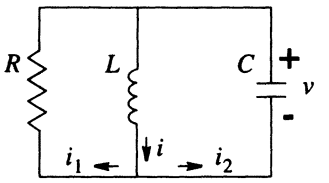

Figure 7.7. Parallel RLC circuit

$$\dot{x}_1 = -\frac{1}{RC}x_1 - \frac{1}{C}x_2$$
$$\dot{x}_2 = \frac{1}{L}x_1.$$

Is this system stable?

Solution

The first step towards assessing the stability of the circuit is to form the scalar, total energy function. It seems reasonable that if one can show that the total energy is always decreasing (when there is no input), then the system will approach a state of no motion (an equilibrium state) and is therefore stable.

The total energy of the parallel RLC circuit is

$$E = \frac{1}{2}Cx_1^2 + \frac{1}{2}Lx_2^2.$$

To determine if E is always decreasing, take its derivative with respect to time and substitute the state equations as needed.

$$\begin{aligned}\dot{E} &= Cx_1\dot{x}_1 + Lx_2\dot{x}_2 \\ &= Cx_1\left(-\frac{1}{RC}x_1 - \frac{1}{C}x_2\right) + Lx_2\left(\frac{1}{L}x_1\right) \\ &= -\frac{1}{R}x_1^2\end{aligned} \tag{7.8}$$

From Eq. (7.8) it is observed that $\dot{E}$ will always be negative as long as the resistance is positive, which it will be. Therefore, the system is expected to be stable.

Example 7.5. Consider the nonlinear pendulum of Example 7.2. Its total energy is

$$E = \frac{1}{2}ml^2\dot{\theta}^2 + mgl(1 - \cos\theta). \tag{7.9}$$

Is this system stable?

Solution

Again, its time derivative is examined (assuming the input τ is zero) as

$$\dot{E} = ml^2\ddot{\theta} + mgl\sin\theta = 0.$$

Since the total energy is not decreasing, it is not clear whether either of the equilibrium points are stable.

7.3.2. Formal Statement of Lyapunov Stability

Although the basis of Lyapunov's direct method for determining stability came from an energy argument, it was generalized to other functions (not just total energy). Before stating Lyapunov's stability conditions, a few definitions are required.

Definition 7.1. *A scalar function $V(\mathbf{x})$ is globally positive definite if and only if $V(\mathbf{x}_0) = 0$, and*

$$V(\mathbf{x}) > 0 \qquad \forall\, \mathbf{x} \neq \mathbf{x}_0. \tag{7.10}$$

This was certainly the case for the total energy function E derived for the parallel RLC circuit example considered earlier. A less restrictive definition is provided next by considering the requirement of Eq. (7.10) in some local region around an equilibrium point $\mathbf{x}_0$.

Definition 7.2. *A scalar function $V(\mathbf{x})$ is locally positive definite if and only if $V(\mathbf{x}_0) = 0$ and there exists a ball of radius r_0 about $\mathbf{x}_0$ (denoted as B_{r_0}) such that for any $\mathbf{x} \in B_{r_0}$, and*

$$V(\mathbf{x}) > 0 \qquad \forall\, \mathbf{x} \neq \mathbf{x}_0.$$

Of course, any function $V(\mathbf{x})$ that is globally positive definite is also locally positive definite at all of its equilibrium points. It is important to note, so far the function $V(\mathbf{x})$ can only be zero at the equilibrium points $\mathbf{x}_0$. The next two definitions are similar to the previous two except the function $V(\mathbf{x})$ will be allowed to be zero at points other than the equilibrium points $\mathbf{x}_0$.

Definition 7.3. *A scalar function $V(\mathbf{x})$ is globally positive semidefinite if and only if $V(\mathbf{x}_0) = 0$, and*

$$V(\mathbf{x}) \geq 0 \qquad \forall\, \mathbf{x} \neq \mathbf{x}_0.$$

Again, a less restrictive definition results when only a region around the equilibrium points is considered.

Definition 7.4. *A scalar function $V(\mathbf{x})$ is locally positive semidefinite if and only if $V(\mathbf{x}_0) = 0$, and there exists a ball of radius r_0 about $\mathbf{x}_0$ (denoted as B_{r_0}) such that for any $\mathbf{x} \in B_{r_0}$, and*

$$V(\mathbf{x}) \geq 0 \qquad \forall\, \mathbf{x} \neq \mathbf{x}_0.$$

With these definitions in place, it is time to consider various types of Lyapunov stability conditions.

7.3.3. Local Stability

The equilibrium point x_0 is stable if there exists a locally positive definite function V such that $-\dot{V}$ is locally positive semidefinite.

7.3.4. Global Stability

The equilibrium point x_0 is globally stable if there exists a globally positive definite function V such that $-\dot{V}$ is globally positive semidefinite.

7.3.5. Global Asymptotic Stability

The equilibrium point x_0 is globally asymptotically stable if there exists a globally positive definite function V such that $-\dot{V}$ is globally positive definite.

Example 7.6. Describe the spring-mass system's stability.

Solution

The kinetic energy T, and potential energy V of the spring-mass system are

$$T = \frac{1}{2}m\dot{x}^2$$
$$V = -mgx + \frac{1}{2}k(\Delta + x)^2$$
$$\Delta = m\frac{g}{k},$$

which can be used with Lagrange's equations to derive the following equation of motion:

$$m\ddot{x} + kx = u$$

and transformed to the state variable equations gives

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -\frac{k}{m}x_1 + u,$$

with the following equilibrium point:

$$x_1 = 0$$
$$x_2 = 0.$$

Pick the total energy as the Lyapunov function candidate, or

$$\begin{aligned} V &= \frac{1}{2}m\dot{x}^2 + \frac{1}{2}k(\Delta + x)^2 - mgx \\ &= \frac{1}{2}m\dot{x}^2 + \frac{1}{2}k(\Delta)^2 + \frac{1}{2}k(x)^2, \end{aligned}$$

and the time derivative of the Lyapunov function candidate is

$$\begin{aligned} \dot{V} &= m\dot{x}\ddot{x} + kx\dot{x} \\ &= m\dot{x}\left(-\frac{k}{m}x\right) + kx\dot{x} \\ &= 0. \end{aligned}$$

This system is therefore globally stable.

Example 7.7. Describe the mass-spring-damper system's stability.

Solution

The equation of motion of the mass-spring-damper system is

$$m\ddot{x} + c\dot{x} + kx = 0.$$

Choose a Lyapunov function candidate with the equilibrium point $x_0 = \dot{x}_0 = 0$ as

$$V = \frac{1}{2}\dot{x}^2 + \frac{1}{2}\frac{k}{m}x^2 > 0.$$

Taking the time derivative gives

$$\begin{aligned} \dot{V} &= \dot{x}\ddot{x} + \frac{k}{m}\dot{x}x \\ &= \dot{x}\left(-\frac{c}{m}\dot{x} - \frac{k}{m}x\right) + \frac{k}{m}\dot{x}x \\ &= -\frac{c}{m}\dot{x}^2 < 0. \end{aligned}$$

Therefore, this system is globally, asymptotically stable.

7.3.6. Comments

Two important comments regarding these stability conditions are worth noting. First, there is no universal method of selecting a Lyapunov function candidate $V(\mathbf{x})$. Using a system's total energy is a good starting point, but this approach may not always work. Second, if the conditions above are not satisfied for the $V(\mathbf{x})$ selected, it does not mean that the system is unstable. It only means that one still does not know because the Lyapunov criteria is only a sufficient condition for stability, not a necessary condition. There are, of course, two possibilities. One, the system is unstable or two your selection of $V(\mathbf{x})$ was poor. The nonlinear pendulum example with two Lyapunov function candidates should help solidify these concepts.

Example 7.8. Consider once again the large angle pendulum (see Eq. 7.5) with the equilibrium points of Eq. (7.7). Describe the system's stability.

Solution

Revisit the total energy function of Eq. (7.9) and obtain

$$V(\mathbf{x}) = \frac{1}{2}ml^2x_2^2 + mgl(1 - \cos x_1). \tag{7.11}$$

It is clear that this choice of $V(\mathbf{x})$ satisfies the conditions for being globally positive definite about the equilibrium point $x_1 = x_2 = 0$. Note, it is not positive definite for the equilibrium point $x_1 = \pi$. Next, construct the time derivative as

$$\dot{V} = ml^2\dot{x}_2x_2 + mgl\dot{x}_1 \sin x_1 = 0$$

that satisfies the condition for global stability ($-\dot{V}(\mathbf{x})$ is globally semidefinite). Therefore, the equilibrium point at $x_1 = x_2 = 0$ is globally stable. Physically, it means that one can find a range of initial conditions from which the system's motion, when it is released, will stay within a previously specified bound.

To analyze the other equilibrium point ($x_1 = \pi$), try a $V(\mathbf{x})$ different from Eq. (7.11). For example,

$$V(\mathbf{x}) = \frac{1}{2}ml^2x_2^2 + mgl(1 + \cos x_1), \tag{7.12}$$

which is also globally positive definite about the new equilibrium point. Taking the time derivative gives

$$\dot{V}(\mathbf{x}) = -2mglx_2 \sin x_1,$$

but it does not satisfy any of the stability conditions. So, nothing is known about the stability of this equilibrium point except that the Lyapunov function candidate of Eq. (7.12) is not a good one to use. It turns out that the equilibrium point $x_1 = \pi$ is indeed unstable.

Summary of Lyapunov's direct method (a five step process):

1. Derive the equations of motion.
2. Transform to state-space representation.
3. Choose a Lyapunov function candidate (start with the total energy).
4. Analyze the Lyapunov function candidate.
5. If stability conditions are indeterminate, return to Step 3, or give up.

Having developed some nonlinear system stability analysis tools, it is time to consider nonlinear system controller design.

7.4. Sliding Mode Control

Although the control design methods for nonlinear systems are not as numerous as for linear systems, some useful techniques do exist. Sliding Mode Control (SMC) is one such example that provides a systematic way to choose a Lyapunov function candidate and the nonlinear system performance. In this section, SMC is developed and initially applied to second-order nonlinear systems[3,4]. Afterwards, it will be generalized to higher-order systems and tracking control. More detailed descriptions, along with other nonlinear control approaches can be found in the references cited at the end of this chapter.

SMC starts with the definition of a stable *sliding surface* in the system's n-dimensional state-space and terminates at the equilibrium point of interest. Using Lyapunov's direct method will yield a control law that drives any initial state to the sliding surface. Once on the sliding surface, the control law drives the state trajectory to the equilibrium point (see Figure 7.8). Note, the equilibrium point can also be a reference trajectory or a series of equilibrium points.

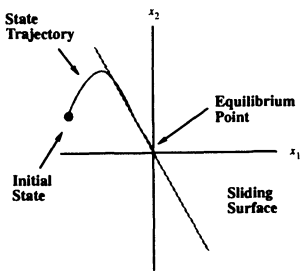

Figure 7.8. SMC graphical interpretation

7.4.1. SMC for Second-Order Systems

Consider the nonlinear system

$$\ddot{\theta} = f(\theta, \dot{\theta}) + u \tag{7.13}$$

where the function $f(\theta, \dot{\theta})$ is continuous and nonlinear and the input is u. A stable sliding surface can be defined as

$$s = \dot{\theta} + w\theta$$

where w is assumed to be a positive controller gain. The control law for tracking the sliding surface to the equilibrium point is found by enforcing the condition that the time rate of change of s is zero, or

$$\dot{s} = \ddot{\theta} + w\dot{\theta} = 0. \tag{7.14}$$

Next, Eq. (7.13) can be substituted into Eq. (7.14), which allows one to solve for the input as

$$u = -f - w\dot{\theta}.$$

Finally, a sign function is appended to the control law to accomplish the goal of moving the state trajectory to the sliding surface, or

$$u = -f - w\dot{\theta} - asgn(s).$$

This controller is known as a regulator since the reference input is zero.

Note 7.1. SMC is a way to use feedback linearization to convert the nonlinear control problem to an *equivalent* linear PD controller.

7.4.2. Stability Assessment

To determine the stability of the closed-loop system, one chooses a $V(\mathbf{x})$ as

$$V(\mathbf{x}) = \frac{1}{2}s^2 \tag{7.15}$$

which is positive definite. Next, one takes the time derivative of Eq. (7.15), or

$$\dot{V} = s\dot{s} = -s[asgn(s)] = -a|s|$$

which clearly satisfies the condition for global asymptotic stability as long as the controller design parameter a is positive. Note, $ssgn(s) = |s|$ is defined as the absolute value of s.

Example 7.9. For DC motor driven, large angle pendulum, find the SMC control law.

Solution

Combine the differential equations for the two separate systems

DC motor

$$J_m\ddot{\theta} + C_m\dot{\theta} = B_mE - \tau$$

pendulum

$$M\ddot{\theta} + K\sin\theta = \tau$$

(where $M = ml^2$ and $K = mlg$) to obtain a single, nonlinear differential equation relating motor voltage input E to pendulum rotation θ, or

$$\begin{aligned} J_m\ddot{\theta} + C_m\dot{\theta} &= B_mE - (M\ddot{\theta} + K\sin\theta) \\ (J_m + M)\ddot{\theta} + C_m\dot{\theta} + K\sin\theta &= B_mE \\ \ddot{\theta} + \widetilde{C}\dot{\theta} + \widetilde{K}\sin\theta &= \widetilde{B}E. \end{aligned}$$

Control Law and Results

Using the sliding mode control technique gives the control law as

$$E = -\frac{1}{\widetilde{B}}\left([W - C]\,\dot{\theta} - \widetilde{K}\sin\theta\right) - Asgn(s)$$

where

$$S = \dot{\theta} + W\theta.$$

The details are left to the reader.

The test case consists of an initial angular displacement 57° from the equilibrium point (vertically down, see Figure 7.9). The purpose of the controller is to bring the pendulum to the equilibrium point. Figures 7.10 and 7.11 show the simulation results for this case.

7.4.3. Sliding Mode Control for Tracking Control

In this section, the SMC approach is extended to multiple degree-of-freedom systems and to tracking control.

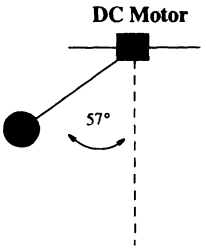

Figure 7.9. Simple pendulum controller with a DC motor

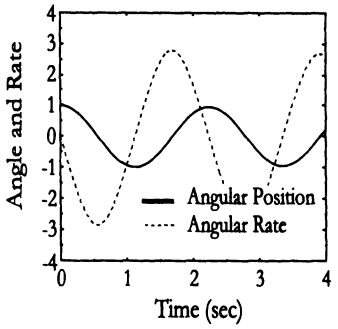

Figure 7.10: Uncontrolled simulation results

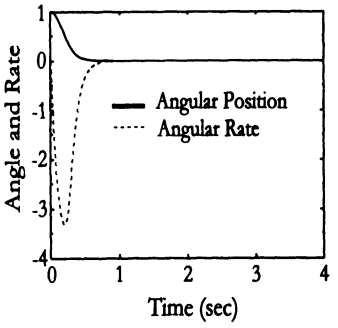

Figure 7.11: Controlled simulation results

Consider the system equations

$$\begin{aligned} \ddot{\mathbf{x}} &= \mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{B}\mathbf{u} \\ \mathbf{s} &= \dot{\mathbf{e}} + \mathbf{W}\mathbf{e} \end{aligned} \tag{7.16}$$

where $\mathbf{x}$ is a m-dimensional vector of physical degrees-of-freedom, $\mathbf{F}$ is a m-dimensional vector of nonlinear functions, $\mathbf{B}$ is a $m \times m$-dimensional input weighting matrix, and $\mathbf{u}$ is a m-dimensional vector of inputs. The sliding surface $\mathbf{s}$ is also a m-dimensional vector, but is now described in terms of the error vector $\mathbf{e} = \mathbf{r} - \mathbf{x}$. The $m \times m$-dimensional matrix $\mathbf{W}$ is positive definite and $\mathbf{r}$ is a m-dimensional reference input vector.

The sliding phase control law is found as before, that is, by forming the time derivative of $\mathbf{s}$, and solving for $\mathbf{u}$, or

$$\mathbf{u} = \mathbf{B}^{-1}(\mathbf{W}\dot{\mathbf{e}} - \mathbf{F} + \ddot{\mathbf{r}}).$$

Again, the sign function is appended to the control law to form

$$\mathbf{u} = \mathbf{B}^{-1}[\mathbf{W}\dot{\mathbf{e}} - \mathbf{F} + \ddot{\mathbf{r}} + \mathbf{A}sgn(\mathbf{s})]$$

where

$$sgn(\mathbf{s}) = [sgn(s_1)sgn(s_2)\ldots sgn(s_m)]^T$$

and the $m \times m$ matrix $\mathbf{A}$ is diagonal and positive.

To assess the closed stability, the Lyapunov function candidate is selected as

$$V = \frac{1}{2}\mathbf{s}^T\mathbf{s}$$

and its time derivative is

$$\dot{V} = -\mathbf{s}^T\mathbf{A}sgn(\mathbf{s}). \tag{7.17}$$

Equation (7.17) can be expanded as

$$\begin{aligned}\dot{V} &= -s_1A_{11}sgn(s_1 - s_2A_{22}sgn(s_2) - \ldots - s_mA_{mm}sgn(s_m)\\ &= -A_{11}|s_1| - A_{22}|s_2| - \ldots - A_{mm}|s_m| < 0\end{aligned}$$

where $|s_i| = s_isgn(s_i)$ and it is clear that the closed-loop system is asymptotically stable (i.e., meets the Lyapunov conditions) which enables the controller to track the reference trajectory $\mathbf{r}(t)$.

7.4.4. Sliding Mode Control for Systems with Parameter Uncertainty

Often there is some uncertainty in the parameters of a system dynamic model. If one knows the uncertainty bounds, then one can establish conditions on the controller design parameter A that will still guarantee global asymptotic stability. For ease of understanding, the following simple linear example demonstrates the approach.

Example 7.10. Consider the general, second-order linear system

$$\ddot{\theta} + 2\zeta\omega_n\dot{\theta} + \omega_n^2\theta = \omega_n^2\tau \tag{7.18}$$

where the damping ratio ζ is thought to be ζ_0 with an error of $\Delta\zeta$, that is,

$$\zeta = \zeta_0 + \Delta\zeta,$$

and the goal of the controller is to track the reference input θ_r. Establish conditions on the controller design parameter A.

Solution

Similar to Section 7.4.3. the control law for τ is

$$\tau = \frac{1}{\omega_n^2}[\ddot{\theta}_r + 2\zeta_0\omega_n\dot{\theta} + \omega_n^2\theta + W\dot{e} + Asgn(s)] \tag{7.19}$$

where ζ_0 has been used instead of the actual ζ, and W and A are the controller design parameters. In the process of assessing stability, a condition on A will arise that guarantees closed loop stability in the presence of the ζ uncertainty. Specifically, one uses the Lyapunov candidate function

$$V = \frac{1}{2}s^2$$

and takes its time derivative, or

$$\dot{V} = s\dot{s} = s(\ddot{\theta}_r - \ddot{\theta} + W\dot{e}), \tag{7.20}$$

which upon substitution of Eq. (7.18) into Eq. (7.20) and using the actual value of the damping ratio produces

$$\dot{V} = s(\ddot{\theta}_r + 2\zeta\omega_n\dot{\theta} + \omega_n^2\theta - \omega_n^2\tau + W\dot{e})$$

and substituting the control law Eq. (7.19) gives

$$= s[2\omega_n\dot{\theta}\Delta\zeta - Asgn(s)]. \tag{7.21}$$

Expanding the right side of Eq. (7.21) gives a new expression for $\dot{V}$ as

$$\dot{V} = 2\Delta\zeta\omega_n\dot{\theta}s - A|s| < 0,$$

which produces the desired inequality condition. Next, one solves for the control design parameter A

$$A > 2\Delta\zeta\omega_n\dot{\theta}sgn(s),$$

with the worst case being

$$A > \max(2\Delta\zeta\omega_n\dot{\theta}).$$

This approach can be applied equally well to nonlinear systems with uncertain parameters, but the conditions for stability on A may be more complicated.

7.4.5. Augmented Sliding Mode Control

Now, it is time to extend the SMC method[5,6] to handle flexible systems where the number of modes to be controlled may be greater than the number of available actuators. In the case of a flexible system with rigid body actuators only, the

controller can be shown to be globally, asymptotically stable in the rigid body motion while adding damping to the measured flexible body modes. This technique is useful for harmonic drives and large gear ratios where the flexible body modes cannot be measured by the encoders and tachometers.

The primary motivation for this technique is its application to flexible manipulators. Typically, rigid manipulators have as many actuators as modes. SMC has been shown to work well for the rigid manipulator case. In theory it will work equally well for flexible systems if there exist as many actuators as modes to be controlled. The goal is to increase the damping of the flexible modes *without* adding additional actuators, but additional sensors (i.e., strain gauges) are required to measure the flexible modes.

For a single flexible link robot modeled by m flexible modes, the nonlinear equations of motion are given by

$$\ddot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{B}U \tag{7.22}$$

where the control U is a scalar and the input weighting coefficient $\mathbf{B}$ is a $(m+1) \times 1$ vector. The $(m+1) \times 1$ vector of nonlinear functions $\mathbf{F}$ is arranged such that N_1 is the nonlinear rigid body equation of motion and N_2 through N_{m+1} are the nonlinear flexible body equations of motion. The vector $\mathbf{x}$ contains the quantities

$$\mathbf{x}^T = [\theta, q_1, \ldots, q_m]$$

where θ is the hub rotation and q_k is the k^{th} generalized flexible body coordinate.

The rotational equation of motion is extracted from Eq. (7.22) as

$$\ddot{\theta} = N_1 + B_1 U. \tag{7.23}$$

The sliding surface is chosen as

$$\begin{aligned} s &= W\left(\theta_e + \tilde{\mathbf{w}}^T \mathbf{q}_e\right) + \left(\dot{\theta}_e + \tilde{\mathbf{w}}^T \dot{\mathbf{q}}_e\right) = 0 \\ \theta_e &= \theta - \theta_{ref} \\ \mathbf{q}_e &= \mathbf{q} - \mathbf{q}_{ref} \end{aligned} \tag{7.24}$$

where W and $\tilde{\mathbf{w}}$ are weighting coefficients (W being positive), $\mathbf{q}$ is the $m \times 1$ vector of deformation generalized coordinates.

The objective of augmented SMC is to force the motion to follow that of the sliding surface. For this reason, the equilibrium point at the origin of s must be shown to be stable. This is done by using Lyapunov's direct method. A Lyapunov function candidate is

$$\begin{aligned} V &= \frac{1}{2}\left(\dot{\theta}_e + \tilde{\mathbf{w}}^T \dot{\mathbf{q}}_e\right)^T \left(\dot{\theta}_e + \tilde{\mathbf{w}}^T \dot{\mathbf{q}}_e\right) \\ \dot{V} &= \left(\dot{\theta}_e + \tilde{\mathbf{w}}^T \dot{\mathbf{q}}_e\right)\left(\ddot{\theta}_e + \tilde{\mathbf{w}}^T \ddot{\mathbf{q}}_e\right). \end{aligned} \tag{7.25}$$

The time derivative of the sliding surface is

$$\dot{s} = W\left(\dot{\theta}_e + \tilde{\mathbf{w}}^T\dot{\mathbf{q}}_e\right) + \left(\ddot{\theta}_e + \tilde{\mathbf{w}}^T\ddot{\mathbf{q}}_e\right) = 0.$$

Solving for the acceleration and substituting them into $\dot{V}$ of Eq. (7.25) gives

$$\dot{V} = -W\left(\dot{\theta}_e + \tilde{\mathbf{w}}^T\dot{\mathbf{q}}_e\right)^T\left(\dot{\theta}_e + \tilde{\mathbf{w}}^T\dot{\mathbf{q}}_e\right). \tag{7.26}$$

Since Eq. (7.26) is negative, the equilibrium point at the origin of Eq. (7.24) is globally asymptotically stable.

The *sliding phase* control is found by enforcing a condition of no motion off the sliding surface, or

$$\dot{s} = W\left(\dot{\theta}_e + \tilde{\mathbf{w}}^T\dot{\mathbf{q}}_e\right) + \left(\ddot{\theta}_e + \tilde{\mathbf{w}}^T\ddot{\mathbf{q}}_e\right) = 0. \tag{7.27}$$

Substituting Eq. (7.23) into Eq. (7.27) gives the *sliding phase* control law as

$$B_1U = -W\left(\dot{\theta}_e + \tilde{\mathbf{w}}^T\dot{\mathbf{q}}_e\right) - N_1 + \ddot{\theta}_{ref} - \tilde{\mathbf{w}}^T\ddot{\mathbf{q}}_e.$$

Treating the weighted sum of generalized accelerations as a disturbance yields in the new control law as

$$B_1U = -W\left(\dot{\theta}_e + \tilde{\mathbf{w}}^T\dot{\mathbf{q}}_e\right) - N_1 + \ddot{\theta}_{ref} - Asgn(s) \tag{7.28}$$

where A is a positive constant. Including the modal velocities in the sliding surface of Eq. (7.24) gives a control that reacts against this velocity, thus producing damping in the included modes. Substituting Eq. (7.28) into Eq. (7.23) gives the closed loop equation of motion for θ_e as

$$\ddot{\theta}_e = -W\left(\dot{\theta}_e + \tilde{\mathbf{w}}^T\dot{\mathbf{q}}_e\right) - Asgn(s).$$

This also has an equilibrium point at the origin. To check the validity of this control for the *reaching phase*, i.e., to check the stability of the equilibrium point a positive definite Lyapunov function candidate is

$$V = \frac{1}{2}s^2.$$

The *reaching phase* control law must satisfy

$$\dot{V} = s\left(W\left(\dot{\theta}_e + \tilde{\mathbf{w}}^T\dot{\mathbf{q}}_e\right) + N_1 + B_1U - \ddot{\theta}_{ref} + \tilde{\mathbf{w}}^T\ddot{\mathbf{q}}_e\right) < 0. \tag{7.29}$$

Substituting Eq. (7.28) into Eq. (7.29) results in

$$\dot{V} = s\left[\tilde{\mathbf{w}}^T\ddot{\mathbf{q}}_e - Asgn(s)\right].$$

This results in a constraint on A to ensure stability, or

$$A > \left| \tilde{\mathbf{w}}^T \ddot{\mathbf{q}}_e \right| .$$

Next, this method's robustness to model uncertainty, unmodeled dynamics, and input disturbances is examined.

Let the true model of the system be

$$\tilde{\mathbf{F}} = \mathbf{F} + \Delta\mathbf{F}$$

where $\Delta\mathbf{F}$ is a vector function describing differences between the true system and its model. These include parameter uncertainty and effects on the truncated system due to unmodeled dynamics.

The true system is assumed to satisfy the equation of motion given by

$$\ddot{\mathbf{x}} = \tilde{\mathbf{F}} + \mathbf{B}U + \mathbf{D}$$

where $\mathbf{D}$ is a vector of input disturbances.

For the sliding surface of Eq. (7.24) and the control law of Eq. (7.28), A must satisfy

$$A > \max_{\mathbf{x},\dot{\mathbf{x}},\ddot{\mathbf{x}}} \left| \Delta N_1 + \mathbf{D} + \tilde{\mathbf{w}}^T \ddot{\mathbf{q}}_e \right|$$

for stability. This shows that the bound on the weighted sum of generalized accelerations, unmodeled dynamics, and input disturbances is required to ensure stability.

The preceding discussion was concerned with the criteria for ensuring stability of the closed-loop system with no regard to performance. The control engineer can be defined system performance by a nonlinear cost function J. The controller parameters are then chosen to minimize J while satisfying the inequality constraints described above. The formal statement of this optimal control problem follows.

For the system given by Eq. (7.22), find the parameters $W, \tilde{\mathbf{w}}$, and A of the control law of Eq. (7.28) and Eq. (7.24) subject to the inequality constraints

$$W > 0$$
$$A > \max_{\mathbf{x},\dot{\mathbf{x}},\ddot{\mathbf{x}}} \left| \Delta N_1 + \mathbf{D} + \tilde{\mathbf{w}}^T \ddot{\mathbf{q}}_e \right| ,$$

which minimize the given cost function J.

This procedure enables the control designer to define the optimal system performance that meets the nonlinear stability requirements. The preceding discussions on numerical optimization (see Chapter 5) including homotopy (see Chapter 4) can be directly applied to this problem.

7.4.6. Output Feedback SMC

The SMC method can be extended further[7] to handle flexible robots with direct-drive actuators. In this case, the encoders and tachometers *see* or measure both the

rigid and flexible degrees-of-freedom. These types of systems can be described by

$$\begin{aligned} \ddot{\mathbf{x}} &= \mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{B}(\mathbf{x})\mathbf{U} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned} \tag{7.30}$$

where $\mathbf{x}$ is a $n \times 1$ vector of degrees-of-freedom, $\mathbf{F}(\mathbf{x}, \dot{\mathbf{x}})$ is a $n \times 1$ vector of nonlinear functions of the $\mathbf{x}$ and $\dot{\mathbf{x}}$, $\mathbf{B}(\mathbf{x})$ is a $n \times m$ matrix of control weighting coefficients that in general may be functions of $\mathbf{x}$, $\mathbf{U}$ is a $m \times 1$ vector of system inputs, $\mathbf{y}$ is a $r \times 1$ vector of measurable outputs, and $\mathbf{C}$ is a $r \times n$ matrix relating state variables to measurable outputs.

The sliding surfaces are designed in the output space, thus implying that sensor output regulation or tracking will yield the desired system motion. The sliding surface may be chosen as

$$\mathbf{s} = \mathbf{W}(\mathbf{y} - \mathbf{y}_r) + (\dot{\mathbf{y}} - \dot{\mathbf{y}}_r) = 0 \tag{7.31}$$

where $\dot{\mathbf{y}}_r$ is the desired sensor output time history and $\mathbf{W}$ is a positive definite matrix with real valued elements.

The equivalent control is found by enforcing a condition of stationarity on the sliding surface, or

$$\dot{\mathbf{s}} = \mathbf{W}(\dot{\mathbf{y}} - \dot{\mathbf{y}}_r) + (\ddot{\mathbf{y}} - \ddot{\mathbf{y}}_r) = 0. \tag{7.32}$$

Substituting Eq. (7.30) into Eq. (7.32) produces

$$\mathbf{CB}(\mathbf{x})\mathbf{U} = -\mathbf{CF}(\mathbf{x}, \dot{\mathbf{x}}) + \ddot{\mathbf{y}}_r - \mathbf{W}(\dot{\mathbf{y}} - \dot{\mathbf{y}}_r). \tag{7.33}$$

The control weighting matrix $\mathbf{B}(\mathbf{x})$, and the vector of nonlinear terms $\mathbf{F}(\mathbf{x}, \dot{\mathbf{x}})$ are approximated by $\mathbf{B}(\hat{\mathbf{x}})$ and $\mathbf{F}(\hat{\mathbf{x}}, \dot{\hat{\mathbf{x}}})$,respectively where

$$\begin{aligned} \hat{\mathbf{x}} &= \mathbf{C}^*\mathbf{y} \\ &\mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1} && n > r \\ \mathbf{C}^* &= \mathbf{C}^{-1} && n = r \\ &(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T && n < r \end{aligned} \tag{7.34}$$

Eq. (7.33) may now be written as

$$\mathbf{CB}(\hat{\mathbf{x}})\mathbf{U} = -\mathbf{CF}(\hat{\mathbf{x}}, \dot{\hat{\mathbf{x}}}) + \ddot{\mathbf{y}}_r - \mathbf{W}(\dot{\mathbf{y}} - \dot{\mathbf{y}}_r) - \mathbf{A}sgn(\mathbf{s})$$

where $\mathbf{A}$ is a $r \times r$ constant matrix. The $\mathbf{A}sgn(\mathbf{s})$ terms is added to drive the output to the stable sliding surface of Eq. (7.31). The control $\mathbf{U}$ is (for $r = m$)

$$\mathbf{U} = [\mathbf{CB}(\hat{\mathbf{x}})]^{-1}[-\mathbf{CF}(\hat{\mathbf{x}}, \dot{\hat{\mathbf{x}}}) + \ddot{\mathbf{y}}_r - \mathbf{W}(\dot{\mathbf{y}} - \dot{\mathbf{y}}_r) - \mathbf{A}sgn(\mathbf{s})]. \tag{7.35}$$

Stability is examined by using Lyapunov's direct method with a Lyapunov function candidate as

$$V = \frac{1}{2}\mathbf{s}^T\mathbf{s}.$$

The requirement for stability is

$$\mathbf{s}^T\Big\{\mathbf{W}(\dot{\mathbf{y}}-\dot{\mathbf{y}}_r)+\mathbf{CF}(\mathbf{x},\dot{\mathbf{x}})-\ddot{\mathbf{y}}_r-\mathbf{CB}(\mathbf{x})[\mathbf{CB}(\hat{\mathbf{x}})]^{-1}\mathbf{CF}(\hat{\mathbf{x}},\dot{\hat{\mathbf{x}}})+\mathbf{CB}(\mathbf{x})[\mathbf{CB}(\hat{\mathbf{x}})]^{-1}\ddot{\mathbf{y}}_r-\mathbf{CB}(\mathbf{x})[\mathbf{CB}(\hat{\mathbf{x}})]^{-1}\mathbf{W}(\dot{\mathbf{y}}-\dot{\mathbf{y}}_r)-\mathbf{CB}(\mathbf{x})[\mathbf{CB}(\hat{\mathbf{x}})]^{-1}\mathbf{A}sgn(\mathbf{s})\Big\}<0. \quad (7.36)$$

Establishing stability of the closed-loop system based on Eq. (7.36) when the input weighting matrix is a function of all the degrees-of-freedom is a formidable task. Fortunately, many real systems have special forms of $\mathbf{B}(\mathbf{x})$ that do facilitate a proof of stability. To this end, the input weighting matrix is written as a combination of the three matrices

$$\mathbf{B}(\mathbf{x})=\mathbf{B}_0+\mathbf{B}_y(\mathbf{y})+\mathbf{B}_{res}(\mathbf{x})$$

where $\mathbf{B}_0$ is a constant matrix, $\mathbf{B}_y(\mathbf{y})$ is a matrix whose elements are only a function of the measurable outputs $\mathbf{y}$, and $\mathbf{B}_{res}(\mathbf{x})$ is a matrix whose elements are only a function of the degrees-of-freedom different from $\mathbf{y}$. Clearly, if $\mathbf{B}_{res}(\mathbf{x})$ is zero or negligible, then there is no approximation in $\mathbf{B}(\hat{\mathbf{x}})$ and Eq. (7.36) simplifies to

$$\mathbf{s}^T[-\mathbf{A}sgn(\mathbf{s})+\mathbf{C}\{\mathbf{F}(\mathbf{x},\dot{\mathbf{x}})-\mathbf{F}(\hat{\mathbf{x}},\dot{\hat{\mathbf{x}}})\}]<0,$$

which is valid for $\mathbf{A}$ diagonal with elements satisfying

$$A_{ii}>||\mathbf{C}\{\mathbf{F}(\mathbf{x},\dot{\mathbf{x}})-\mathbf{F}(\hat{\mathbf{x}},\dot{\hat{\mathbf{x}}})\}||. \quad (7.37)$$

Again, the vector $\mathbf{F}(\mathbf{x},\dot{\mathbf{x}})$ may be composed of a constant part, a $\mathbf{y}$ only part, and a part dependent only on the degrees-of-freedom different from $\mathbf{y}$. Exploiting these relationships during control law implementation may result in the trivial stability constraint of $\mathbf{A}$ being strictly negative.

Example 7.11. Position regulation of two masses

This example presents the basic design procedure by using a system that exhibits both rigid body and flexible body motions. The system under consideration consists of two masses connected by a nonlinear hardening spring, and a linear damper. A force acting on the first mass is the sole input to the system. The system is shown in Figure 7.12 where z_1 and z_2 are the displacements of the individual masses m_1 and m_2.

The nonlinear hardening spring obeys the following relationship:

$$F_{spring}=K_l(z_1-z_2)+K_{nl}(z_1-z_2)^3$$

where K_l and K_{nl} are the linear and nonlinear spring constants, respectively. The damping coefficient is denoted by C and the force input to the system by F. The particular values used for this example are given in Table 7.1.

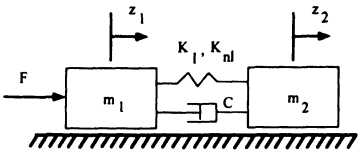

Figure 7.12. Two mass system connected by a nonlinear spring and a linear damper

Table 7.1. Physical parameters used for the two mass system of Example 7.11

Symbol	Units	Value
m_1	kg	1.0
m_2	kg	0.1
C	kg/s	0.2
K_l	kg/s/s	8.0
K_{nl}	kg/(m^{2}s^2)	50.0

The control objective is to move the center of mass of the system

$$x_1 = \frac{1}{2}(z_1 + z_2)$$

to a specified location while suppressing the reciprocating motion of the entire system

$$x_2 = (z_1 - z_2)$$

where the only measurable quantities are the position and velocity of the system center of mass, x_1 and $\dot{x}_1$.

Solution

By using the rigid body and flexible body coordinates x_1 and x_2, the equations of motion may be written in the form of Eq. (7.30) where

$$\mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}) = \begin{bmatrix} -\frac{1}{2}\left(\left(\frac{m_2 - m_1}{m_1 m_2}\right)(K_l x_2 + K_{nl} x_2^3 + C\dot{x}_2)\right) \\ -\frac{1}{2}\left(\left(\frac{m_2 + m_1}{m_1 m_2}\right)(K_l x_2 + K_{nl} x_2^3 + C\dot{x}_2)\right) \end{bmatrix}$$

$$\mathbf{B} = \frac{1}{2m_1}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

The output feedback sliding mode control (OFSMC) law is given by Eq. (7.31) and Eq. (7.35) with the stability constraint of Eq. (7.37). Since $n = 2$ and $r = 1$,

the vector $\mathbf{F}(\mathbf{x}, \dot{\mathbf{x}})$ must be approximated based on the measurable output y and $\dot{y}$ by using the pseudoinverse of Eq. (7.34) as

$$\hat{\mathbf{x}} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} y$$

$$\dot{\hat{\mathbf{x}}} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \dot{y}.$$

In this special case, the terms $\mathbf{F}(x_2, \dot{x}_2)$ are not dependent on the measured quantity x_1, therefore the approximation of $\mathbf{F}(x_2, \dot{x}_2)$ is exactly zero.

The output feedback sliding mode controller is compared to a simple PD compensator

$$u = K_p(y_r - y) + K_d(\dot{y}_r - \dot{y})$$

where K_p and K_d are the proportional and derivative error gains, respectively. The PD controller gains and the OFSMC gains W and A were chosen to limit the peak overshoot to 5.5%. These gains are given in Table 7.2.

Table 7.2: Controller parameters for OFSMC and PD compensation of the two mass system of Example 7.11

Symbol	Units	Value
A	1/sec	0.4
W	1/sec	1.9
K_p	kg/s/s	5.0
K_d	kg/s	7.0

Although both control schemes gave similar performance in tracking a reference command, the OFSMC exhibited enhanced disturbance accommodation characteristics. This is shown by the closed-loop performance of the system in response to an initial velocity applied to the second mass. The motion of the system center of mass x_1 and the measure of flexible body motion x_2 are shown in Figure 7.13 and Figure 7.14, respectively.

7.4.7. Control-Structure-Actuator Interaction

It is time to become specific about the applications of output feedback and augmented SMC methods[8,9]. Standard servo systems typically provide actuator position and velocity feedback. An ideal control design would use only servo sensors to achieve accurate position control while attenuating any residual vibrations. If structural sensors were also included, a structural sensor failure would still result in adequate performance. If the vibration of the structure can be sensed at the slewing

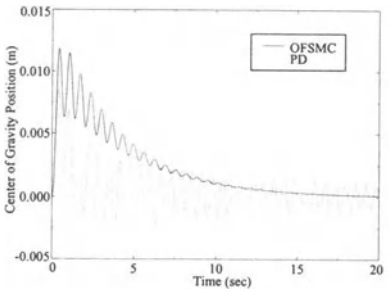

Figure 7.13: Closed-loop response of system cg to an initial velocity disturbance on mass 2

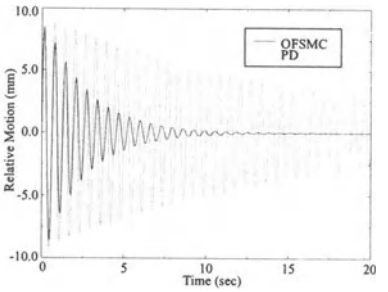

Figure 7.14: Closed-loop response of relative motion of the masses to an initial velocity disturbance on mass 2

axis, then both the rigid and flexible displacements will interact. This interaction occurs since vibration in the structure would cause rotation of the slewing axis. By taking advantage of the interaction between the actuator and the structure, vibration attentuation of the flexible modes can be achieved by using only the angular position and velocity sensors. The sliding surface for conventional SMC can be expressed as

$$s = \dot{e} + We \tag{7.38}$$

where the variables have been previously defined (see Eq. 7.16). For flexible systems, such as, a slewing flexible link, the error would be redefined as

$$e = \theta_{REF} - \left(\theta + \frac{\partial y(0,t)}{\partial x}\right) \tag{7.39}$$

where θ_{REF} is the reference trajectory, θ is the servo slewing angle, and the last term is the angular deflection of the flexible link evaluated at the root of the hub. The angular deflection represents the interaction between the actuator and the flexible link. When designing a SMC system (following the procedures of Section 7.4.6.) the goal is to suppress the filtered errors represented by sliding surfaces. Embedded within the definition are the flexible link deflections, which is equivalent to output feedback. The conventional SMC would not only provide robust tracking, but would automatically include the vibration suppression due to the inherent interaction. Typically, this is the case for direct-drive servo systems.

For gear-driven servo systems the effects of deadband on the flexible structure must be considered. If one considers a system with a harmonic drive or other gearbox, then these gear trains amplify the torque available from the motor and reduce the effect of the structure on the motor. Additionally, such devices are not ideal and they require a certain amount of torque simply to move the gearing. Fast slewing systems are often geared and have an inherent amount of deadband. In many cases, the gear-train deadband reduces the interaction between the servo actuator and the structure. This condition suggests an ideal solution using augmented SMC.

In the case of the slewing flexible link, the flexible link deflections would not interact with the hub servo system. The error would simply be

$$e = \theta_{REF} - \theta.$$

Using conventional SMC (with hub sensors only) results in robust tracking, but does not guarantee any suppression of the flexible link deflections. For a given servo tracking scenario, the hub may follow the trajectory perfectly while the flexible link may result in large deflections and residual oscillations. The use of structural sensors along with hub sensors would suppress the deflections by choosing an augmented sliding surface such as

$$s = W\left(e + \widetilde{W}^T \varepsilon_x\right) + \left(\dot{e} + \widetilde{W}^T \dot{\varepsilon_x}\right) \tag{7.40}$$

where $\widetilde{W}$ is an additional gain and ε_x is a strain gauge strategically placed along the flexible link. For a flexible link in bending, the longitudinal centroidal axis becomes an elastic curve. In this case, the strain of the fibers are proportional to the product of the distance from the neutral surface and the curvature. This is defined mathematically as

$$\varepsilon_x = z_{sg} \frac{\partial^2 y(x,t)}{\partial x^2}$$

where z_{sg} is the distance from the neutral axis to the strain gauge. Note, this structural sensor summarizes information about the many vibrational modes of the flexible link. By minimizing the sliding surface given by Eq. (7.40) with an augmented SMC architecture, one would provide both robust tracking and vibration suppression of the flexible link deflections. For these classes of flexible systems with gear-driven servo systems, augmented SMC offers an advantage over conventional control systems.

One last topic needs to be discussed when implementing SMC on flexible robots. The control term $-Asgn(s)$ compromises the SMC controller's robustness with respect to unmodeled, high frequency dynamics. Specifically, the high frequency content of this term could generate spill-over into the unmodeled flexible modes and create chattering. One way to eliminate this problem is to replace this term with $-A[atan(s)]$, which smooths the switching behavior and eliminates the high frequency content.

7.5. Chapter 7 Summary

This chapter introduced control techniques to deal with nonlinear systems. State-space representations were reviewed and stability concepts based on Lyapunov's direct method were introduced. A systematic method based on SMC was developed for nonlinear systems. This method was developed for both regulator

and tracking control of multiple degree-of-freedom systems. How SMC accommodates parameter uncertainty was also shown. Two important variations that specifically address flexible systems were developed. The first introduced the concept of augmented SMC in which the sliding surface is designed to suppress the additional degrees-of-freedom introduced by flexible systems. Control of harmonic drive or high gear ratio systems benefit from this technique. The second variation introduced output feedback SMC, which facilitates the design of the sliding mode controller based exclusively on output feedback variables and is used to control direct-drive systems. Stability was proven for both methods by using Lyapunov's direct method. The last section discussed the use of output feedback SMC and augmented SMC for both direct-drive and gear-driven servo systems with flexible links as aides in addressing control-structure-actuator interaction. In the final chapter (see Chapter 8) the SMC architecture is extended to adaptive control.

7.6. Chapter 7 References

1. M. Vidyasagar, *Nonlinear Systems Analysis*, 2^{nd} Edition, Prentice Hall, Inc., Englewood Cliffs, N.J., 1993.
2. H. K. Khalil, *Nonlinear Systems*, Macmillan Publishing Co., N.Y., 1992.
3. V. Utkin, *Sliding Modes in Control Optimization*, Springer-Verlag, N.Y., 1992.
4. J.-J.E Slotine, and W. Li, *Applied Nonlinear Control*, Prentice Hall, Inc., Englewood Cliffs, N.J., 1991.
5. G. G. Parker, *Control Techniques for Multibody Flexible Structures Modelled by a Method of Quadratic Modes*, Ph.D. Dissertation: State University of New York at Buffalo, April, 1994.
6. G. G. Parker, D. J. Segalman, R. D Robinett, and D. J. Inman, "Decentralized sliding mode control for flexible link robots," Journal of Intelligent and Robotic Systems, 17: pp. 61-79,1996.
7. G. G. Parker, and R. D. Robinett, "Output Feedback Sliding Mode Control with Application to Flexible Multibody Systems," Sandia National Laboratories, SAND95-2573J, 1995.
8. E. Garcia, and D. J. Inman, "Modeling of the slewing control of a flexible structure," Journal of Guidance Navigation, and Control, vol. 14, no. 4, pp. 736-742, July-August, 1991.
9. D. G. Wilson, *Nonlinear/Adaptive Control Architectures with Active Structures for Flexible Manipulators*, Ph.D. Dissertation: Mechanical Engineering Department, University of New Mexico, Albuquerque, NM, May, 2000.

7.7. Chapter 7 Problems

Homework 7.1. From Sections 7.2., and 7.3.
Find the equilibrium points and determine stability for the three dynamic systems. Determine whether each system is asymptotically stable, global asymptotically sta-

ble, or ?

a. $\dot{x} = -10x^3 + \sin^2 x$

b. $\ddot{x} = \sin x + 5x^2$

c. $\ddot{x} + 3\dot{x}^5 + 4x^3 = x^2 \cos^2 3x$

Homework 7.2. From Section 7.4.
Validate the sliding mode control design of Example 7.9 for the DC motor driven, large angle pendulum system. Develop a numerical simulation and implement the dynamic equations of motion, sliding surface, sliding mode control law, and stability. Plot the angular position, angular rate, and motor voltage requirement.

Homework 7.3. From Section 7.4.
Design and simulate a sliding mode control for a slewing horizontal link. Base your dynamic model on Example 3.7. Retain only a single mode. Use $L = 10, EI = 1.4e4$, and $\rho = 1.2$. Select controller gains for a critically damped response. Plot angular position, tip deflections and torque.

Homework 7.4. From Section 7.4.3.
Given the following 2^{nd} order system transfer function:

$$\frac{\theta}{\theta_c} = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2}$$

with

$$\theta_e = \theta - \theta_{ref},$$

design a sliding mode controller such that the system output $\theta(t)$ will track the reference input angle history $\theta_{ref}(t)$. The controller should have the form

$$\theta_c = F\left(\theta, \dot{\theta}, \ddot{\theta}_{ref}\right).$$

Homework 7.5. From Section 7.4.4.
For the transfer function presented in Homework 7.4, introduce the following uncertainty in ζ:

$$\zeta = \zeta_0 + \Delta\zeta.$$

Derive the sliding mode control law based on the model. What additional closed-loop stability condition is required? *Hint*: use the Lyapunov stability criterion.

Chapter 8

Adaptive Sliding Mode Control

8.1. Introduction

Traditionally, adaptive control is applied to dynamic systems that have constant or slowly-varying, uncertain or unknown parameters, such as, manipulator payloads. In the presence of changing plant dynamics, adaptive control design inherently adjusts control system parameters. Adaptive SMC is a specialized form of adaptive control algorithms that falls into the category of robust adaptive control design. A term is included in the control law development that ensures stability in the presence of disturbances, unmodeled dynamics, and modeling inaccuracies.

Sources of model imprecision[1] come from actual uncertainty or unknown plant parameters, and intentional simplification of system dynamics, such as, modeling nonlinear phenomena (like friction) as linear, or neglecting structural modes in a predominately rigid mechanical system. Modeling inaccuracies can be categorized into two types as follows:

1. Structured or parametric uncertainties including modeling inaccuracies.

2. Unstructured uncertainties, or unmodeled dynamics (underestimation of system order).

Two approaches dealing with modeling inaccuracies are robust control, such as, SMC and adaptive control. In the following adaptive control development[1-4], these two methods are merged.

Adaptive control development and application spans many decades and remains a major interest of the controls engineer. The concept of adapting to unknown environments is an attraction due to its improved performance over a larger operating regime. Several improvements to earlier shortcomings (persistent excitation and slowly varying parameters) make adaptive control a serious contender over traditional and other nonlinear control techniques. A correctly designed adaptive controller may provide faster transient decay while using the same control effort as

R. D. Robinett III et al., *Flexible Robot Dynamics and Controls*

conventional control systems (e.g., PD, and PID). One example involving an adaptive vibration isolation system for flexible structures[5] reported a 30% reduction in control effort over PID control with essentially the same performance.

In this chapter, an adaptive control architecture based on an extension of the SMC approach is developed. The intent is to provide a natural enhancement to the material presented in Chapter 7 and provide adaptive control algorithm examples. Adaptive control is one alternative to achieve explicit compensation of large parameter variations in the control law development. *A priori* knowledge of the structure of the dynamic system equations of motion is required. This is normally the case for both rigid and flexible manipulator systems. The development is limited to a specific adaptive control architecture that falls within the robust adaptive control family of controllers. There are many adaptive control schemes found in the literature that provide broader developments for dynamical systems. One resource[3] contains an excellent review (of many) of the most popular adaptive control algorithms. Most recently, another resource[6] introduced (from the theoretical aspect of adaptive control) a novel methodology that systematically constructs both feedback control laws and their corresponding Lyapunov functions. These authors introduce a recursive methodology, called backstepping. This methodology systematically constructs both feedback control laws and their corresponding Lyapunov functions. Global stability and tracking are automatically built into the nonlinear system. Other conventional adaptive control techniques rely on linearization methods that may cancel some of the useful properties associated with the nonlinearities. The interested reader is encouraged to explore the literature[6-15] for the latest developments in adaptive control algorithms and their applications to flexible manipulator systems.

8.2. Adaptive Sliding Mode Control

The class of systems considered have the form

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{N}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{B}\mathbf{u} \tag{8.1}$$

where $\mathbf{M}$ is a $n \times n$ constant matrix, $\mathbf{x}$ is a $n \times 1$ state vector, $\mathbf{N}(\mathbf{x}, \dot{\mathbf{x}})$ is a $n \times 1$ vector, $\mathbf{u}$ is a $n \times 1$ vector of inputs, and $\mathbf{B}$ is a $n \times n$ weighting matrix. These dynamic equations possess the *linear in the parameters property*[2]. Since some or all of the parameters may be unknown, this is important. Thus, the dynamics are linear in the unknown terms. This property is expressed as

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{N}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{Y}(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}})\varphi \tag{8.2}$$

where φ is the parameter vector of unknown terms, and $\mathbf{Y}(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}})$ is a known matrix of robot functions that depend on joint variables, velocities, and accelerations. Solve Eq. (8.1) for the accelerations as

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}\left[\mathbf{B}\mathbf{u} - \mathbf{N}(\mathbf{x}, \dot{\mathbf{x}})\right]. \tag{8.3}$$

Define the tracking sliding surface as

$$\mathbf{s} = \dot{\mathbf{x}}_r - \dot{\mathbf{x}} + \mathbf{W}(\mathbf{x}_r - \mathbf{x}) \tag{8.4}$$

where $\mathbf{W}$ is an $n \times n$ positive definite constant matrix. Take the derivative of Eq. (8.4) and equate to zero, or

$$\dot{\mathbf{s}} = \ddot{\mathbf{x}}_r - \ddot{\mathbf{x}} + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}}) = \mathbf{0}. \tag{8.5}$$

Substitute Eq. (8.3) for the acceleration to obtain

$$-\mathbf{M}^{-1}\mathbf{B}\mathbf{u} + \mathbf{M}^{-1}\mathbf{N} + \ddot{\mathbf{x}}_r + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}}) = \mathbf{0}.$$

Solve for the equivalent control as

$$\widehat{\mathbf{B}}\mathbf{u} = \widehat{\mathbf{M}}(\ddot{\mathbf{x}}_r + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}})) + \widehat{\mathbf{N}}. \tag{8.6}$$

Add the term $\mathbf{A}sgn(\mathbf{s})$, or

$$\mathbf{B}\mathbf{u} = \widehat{\mathbf{B}}\mathbf{u} + \mathbf{A}sgn(\mathbf{s}).$$

Later, it will be shown that this term ensures stability. Expand Eq. (8.6) as

$$\mathbf{B}\mathbf{u} = \widehat{\mathbf{M}}(\ddot{\mathbf{x}}_r + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}})) + \widehat{\mathbf{N}} + \mathbf{A}sgn(\mathbf{s}). \tag{8.7}$$

Substitute Eq. (8.7) (the control law) into Eq. (8.3) to obtain

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}\Big[\widehat{\mathbf{M}}(\ddot{\mathbf{x}}_r + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}})) - \mathbf{M}(\ddot{\mathbf{x}}_r + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}})) + \mathbf{M}(\ddot{\mathbf{x}}_r + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}})) + \widehat{\mathbf{N}} + \mathbf{A}sgn(\mathbf{s}) - \mathbf{N}\Big]$$

where the actual mass matrix is multiplied by the command acceleration, and added or subtracted to create a mass matrix parameter error term[15] as

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}\Big[(\widehat{\mathbf{M}} - \mathbf{M})(\ddot{\mathbf{x}}_r + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}})) + (\widehat{\mathbf{N}} - \mathbf{N}) + \mathbf{A}sgn(\mathbf{s})\Big] + (\ddot{\mathbf{x}}_r + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}})). \tag{8.8}$$

For stability investigations and to determine the adaptation update equations, the following Lyapunov function candidate is suggested[1,2]:

$$\mathcal{V} = \frac{1}{2}\mathbf{s}^T\mathbf{M}\mathbf{s} + \frac{1}{2}\tilde{\boldsymbol{\varphi}}^T\boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\varphi}} \tag{8.9}$$

where $\tilde{\boldsymbol{\varphi}} = \boldsymbol{\varphi} - \hat{\boldsymbol{\varphi}}$ and $\boldsymbol{\Gamma}^{-1}$ is a $n \times n$ positive definite, symmetric, constant matrix. Since $\dot{\mathcal{V}} \leq 0$ for stability, take the derivative of Eq. (8.9) to obtain

$$\dot{\mathcal{V}} = \mathbf{s}^T\mathbf{M}\dot{\mathbf{s}} + \tilde{\boldsymbol{\varphi}}^T\boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\varphi}}}. \tag{8.10}$$

Substitute Eq. (8.8) into Eq. (8.5) and the result into Eq. (8.10) as

$$\dot{\mathcal{V}} = \mathbf{s}^T\mathbf{M}\Big\{\ddot{\mathbf{x}}_r+\mathbf{W}(\dot{\mathbf{x}}_r-\dot{\mathbf{x}})-\mathbf{M}^{-1}\Big[(\widehat{\mathbf{M}}-\mathbf{M})(\ddot{\mathbf{x}}_r+\mathbf{W}(\dot{\mathbf{x}}_r-\dot{\mathbf{x}}))+(\widehat{\mathbf{N}}-\mathbf{N})+\mathbf{A}sgn(\mathbf{s})\Big] - (\ddot{\mathbf{x}}_r + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}}))\Big\} + \tilde{\boldsymbol{\varphi}}^T\boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\varphi}}}. \quad (8.11)$$

Simplifying gives

$$\dot{\mathcal{V}} \doteq \mathbf{s}^T\Big[(\mathbf{M} - \widehat{\mathbf{M}})(\ddot{\mathbf{x}}_r + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}})) + (\mathbf{N} - \widehat{\mathbf{N}}) - \mathbf{A}sgn(\mathbf{s})\Big] + \tilde{\boldsymbol{\varphi}}^T\boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\varphi}}}. \quad (8.12)$$

Use the linear in parameters property of Eq. (8.2) to introduce the following variation:

$$(\mathbf{M} - \widehat{\mathbf{M}})(\ddot{\mathbf{x}}_r + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}})) + (\mathbf{N} - \widehat{\mathbf{N}}) = \mathbf{Y}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{x}_r, \dot{\mathbf{x}}_r, \ddot{\mathbf{x}}_r)\tilde{\boldsymbol{\varphi}}. \quad (8.13)$$

Substituting Eq. (8.13) into Eq. (8.12) gives

$$\dot{\mathcal{V}} = \mathbf{s}^T\left[\mathbf{Y}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{x}_r, \dot{\mathbf{x}}_r, \ddot{\mathbf{x}}_r)\tilde{\boldsymbol{\varphi}} - \mathbf{A}sgn(\mathbf{s})\right] + \tilde{\boldsymbol{\varphi}}^T\boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\varphi}}},$$

and regrouping yields

$$\dot{\mathcal{V}} = -\mathbf{s}^T\mathbf{A}sgn(\mathbf{s}) + \mathbf{s}^T\mathbf{Y}(\cdot)\tilde{\boldsymbol{\varphi}} + \tilde{\boldsymbol{\varphi}}^T\boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\varphi}}}. \quad (8.14)$$

Take the transpose of the middle term in Eq. (8.14) and condense as

$$\dot{\mathcal{V}} = -\mathbf{s}^T\mathbf{A}sgn(\mathbf{s}) + \tilde{\boldsymbol{\varphi}}^T\left[\mathbf{Y}^T(\cdot)\mathbf{s} + \boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\varphi}}}\right]. \quad (8.15)$$

For an $\mathbf{A}$ selected as positive definite, the first term in Eq. (8.15) is negative definite. Determine the adaptation terms and guarantee stability by setting the term inside the bracket of Eq. (8.15) to zero, or

$$\mathbf{Y}^T(\cdot)\mathbf{s} + \boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\varphi}}} = \mathbf{0}. \quad (8.16)$$

Solve Eq. (8.16) for the adaptive update equations as

$$\dot{\hat{\boldsymbol{\varphi}}} = \boldsymbol{\Gamma}\mathbf{Y}^T(\cdot)\mathbf{s} \quad (8.17)$$

where $\tilde{\boldsymbol{\varphi}} = \boldsymbol{\varphi} - \hat{\boldsymbol{\varphi}}$, and $\boldsymbol{\varphi}$ = constant.

8.3. Examples

Two examples are developed to show the adaptive SMC algorithm implementation and performance.

Example 8.1. DC motor driven, large angle pendulum system

This discussion applies adaptive SMC to Example 7.9. The combined equations of motion for the dynamic system are

$$M_{tot}\ddot{\theta} + K\sin\theta = B_m E$$

where $M_{tot} = J_m + M$. The $C_m\dot{\theta}$ is neglected because it did not substantially contribute to the performance of the adaptive controller. Solve for the angular acceleration as

$$\ddot{\theta} = M_{tot}^{-1}\left(B_m E - K\sin\theta\right). \tag{8.18}$$

The stable tracking sliding surface is defined as

$$s = \dot{\theta}_r - \dot{\theta} + W(\theta_r - \theta) \tag{8.19}$$

where W is a positive definite controller gain. Solve for the control law by taking the derivative of Eq. (8.19) and equating to zero, or

$$\dot{s} = \ddot{\theta}_r - \ddot{\theta} + W(\dot{\theta}_r - \dot{\theta}) = 0. \tag{8.20}$$

Substitute Eq. (8.18) into Eq. (8.20) to obtain

$$-\frac{B_m E}{M_{tot}} + \frac{K\sin\theta}{M_{tot}} + \ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta}) = 0.$$

Solve for the approximate control law as

$$\widehat{B_m}E = \widehat{K}\sin\theta + \widehat{M}_{tot}\left(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})\right). \tag{8.21}$$

A robustifying term is added to satisfy the sliding condition, or

$$B_m E = \widehat{B_m}E + Asgn(s). \tag{8.22}$$

Substitute Eq. (8.21) into Eq. (8.22)

$$B_m E = \widehat{K}\sin\theta + \widehat{M}_{tot}\left(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})\right) + Asgn(s). \tag{8.23}$$

Update Eq. (8.18) by substituting Eq. (8.22) (the control law), and add/subtract the mass times the effective control acceleration term to obtain

$$\ddot{\theta} = M_{tot}^{-1}\Big[\widehat{M}_{tot}(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})) - M_{tot}(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})) + M_{tot}(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})) + \widehat{K}\sin\theta + Asgn(s) - K\sin\theta\Big].$$

To verify stability and determine the adaptive control laws, the following Lyapunov function candidate is proposed:

$$\mathcal{V} = \frac{1}{2}\mathbf{s}^T M_{tot}\mathbf{s} + \frac{1}{2}\tilde{\boldsymbol{\varphi}}^T \boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\varphi}}.$$

Take the derivative, or

$$\dot{\mathcal{V}} = \mathbf{s}^T M_{tot}\dot{\mathbf{s}} + \tilde{\boldsymbol{\varphi}}^T \boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\varphi}}} \leq 0.$$

Make substitutions and group terms as

$$\dot{\mathcal{V}} = s^T M_{tot}\Big\{\Big(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})\Big) - M_{tot}^{-1}\Big[\Big(\widehat{M}_{tot} - M_{tot}\Big)(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})) + \Big(\widehat{K} - K\Big)\sin\theta + Asgn(s) + M_{tot}\Big(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})\Big)\Big]\Big\} + \tilde{\boldsymbol{\varphi}}^T \boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\varphi}}}.$$

Simplify and obtain

$$\dot{\mathcal{V}} = s^T\{(M_{tot} - \widehat{M}_{tot})(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})) + (K - \widehat{K})\sin\theta\} - s^T Asgn(s) + \tilde{\boldsymbol{\varphi}}^T \boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\varphi}}}.$$

Introduce the parameterization partitions and expand the last term as

$$\dot{\mathcal{V}} = s^T Y_1\tilde{\varphi}_1 + s^T Y_2\tilde{\varphi}_2 - s^T Asgn(s) + \tilde{\varphi}_1^T\gamma_1^{-1}\dot{\tilde{\varphi}}_1 + \tilde{\varphi}_2^T\gamma_2^{-1}\dot{\tilde{\varphi}}_2$$

Grouping like parameterization terms yields

$$\dot{\mathcal{V}} = -s^T Asgn(s) + \tilde{\varphi}_1^T\left[Y_1^T s + \gamma_1^{-1}\dot{\tilde{\varphi}}_1\right] + \tilde{\varphi}_2^T\left[Y_2^T s + \gamma_2^{-1}\dot{\tilde{\varphi}}_2\right] \tag{8.24}$$

where

$$Y_1\tilde{\varphi}_1 = (\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta}))\left(M_{tot} - \widehat{M}_{tot}\right)$$
$$Y_2\tilde{\varphi}_2 = \sin\theta\left(K - \widehat{K}\right).$$

Note, $\tilde{\varphi}_1 = \left(M_{tot} - \widehat{M}_{tot}\right)$ and $\tilde{\varphi}_2 = \left(K - \widehat{K}\right)$ are the parameter vector components.

The goal is to achieve Lyapunov stability or $\dot{\mathcal{V}} \leq 0$. For $A > 0$, the first term of Eq. (8.24) meets these conditions. Setting the remaining bracketed terms to zero will determine the adaptation update equations and meet the Lyapunov stability condition. For the bracketed terms the result is

$$\dot{\widehat{M}}_{tot} = \gamma_1(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta}))s \tag{8.25}$$

$$\dot{\widehat{K}} = \gamma_2 \sin\theta s \tag{8.26}$$

where γ_1, γ_2 are positive adaptive control gains.

In summary the sliding surface is given as

$$s = \dot{\theta}_r - \dot{\theta} + W(\theta_r - \theta).$$

The adaptive SMC law is

$$B_m E = \widehat{M}_{tot}(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})) + \widehat{K}\sin\theta + Asgn(s)$$

where the adaptive control parameter update equations are Eq. (8.25) and Eq. (8.26).

Control laws that satisfy the sliding condition lead to *perfect* tracking in the presence of model uncertainty. The control laws are discontinuous across the surface $s(t)$ leading, in practice, to control chattering[1]. Chattering is highly undesirable as it leads to extremely high control actuation that may cause the excitation of the high-frequency dynamics that were neglected in the modeling step. The switching control law must change to eliminate chattering .

Note, any vector function satisfying $sgn(\mathbf{s}) = sgn[f(\mathbf{s})]$ when added to the control law will result in asymptotically stable reaching motion[3]. The high-frequency content of the $sgn(\mathbf{s})$ function could excite unmodeled dynamics. The sgn function can be smoothed to alleviate that phenomenon. By proper selection of the smoothed function the controller could be made asymptotically stable. In this example and Example 8.2, the $sgn(\mathbf{s})$ function is replaced with the hyperbolic tangent function $tanh(\beta\mathbf{s})$. The steepness of the slope is adjusted with the coefficient β.

A numerical simulation study was performed with the physical parameters given in Table 8.1. Both adaptive sliding mode control (ASMC) and SMC are compared to show performance benefits.

Table 8.1. Example 8.1 physical parameters

M_{tot}	K	B_m
$kg - m^2$	$\frac{N-m}{rad}$	$\frac{N-m}{Volt}$
0.0529	0.75	0.2026

Several test cases were investigated. In the first case, a SMC with low gains is used. In the second case, ASMC is used. In the third case, the sliding mode controller gains are increased. The controller gains for each case are shown in Table 8.2. For all cases the parameters are assumed to be poorly known (within only 10% of their actual values). For all three cases, the angular positions and velocities are shown in Figure 8.1. The voltage requirements are shown in Figure 8.2

Table 8.2. Controller simulation gains

CASE	Legend	W	A	β	γ_1	γ_2	γ_3
1	Low Gain SMC	2	0.1	10	0	0	0
2	ASMC	2	0.1	10	0.1	0.1	0.1
3	High Gain SMC	2	2	10	0	0	0

(left plot). The parameter estimates for the ASMC are shown for mass and stiffness estimates in Figure 8.2 (right plot).

Note, in terms of reducing the position response to zero while minimizing peak control voltage, the ASMC showed the best performance. For SMC, the A gain must be greater than the difference in parametric uncertainties. Thus, there is a potential for noise to creep into the system as this error grows with increasing A. A large A is not required with the ASMC as the adaptive portion *cancels* the parametric uncertainties. Consequently, reasonably less control authority is required to achieve the same or better performance (see the voltage responses in Figure 8.2).

Furthermore, the ASMC is not susceptible to the SMC requirements of high controller gains. However, more complexity during implementation is required for the ASMC. Theoretically, it has been shown[2] that the tracking error is asymptotically stable and the parameter estimates are bounded. Figure 8.2 (right plot) reveals that the parameter adaptation law does not require convergence to the actual parameters[1]. The adaptation algorithm converges to values that help to asmptotically drive the tracking error to zero.

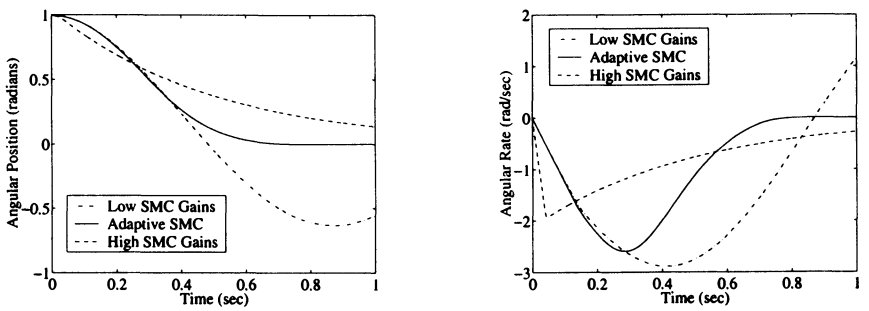

Figure 8.1. Adaptive SMC and SMC angular position and rate responses

Example 8.2. Slewing flexible link[4]

For the planar rotating beam of this example, the dynamic equations of motion were developed by using the methods of quadratic modes. The mathematical model was created by applying the methodology outlined in Chapter 3, the flexible

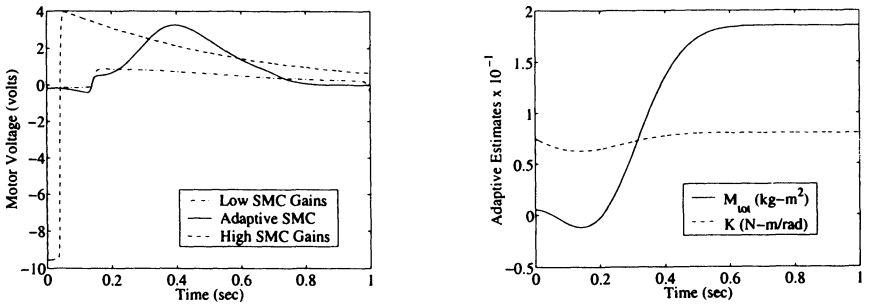

Figure 8.2. Motor voltage requirements and adaptive estimate responses

robot dynamic modeling chapter. A numerical simulation model was constructed by utilizing the MATLAB® simulation environment. By incorporating the numerical optimization techniques developed in Chapter 2 and Chapter 5, the simulation model was calibrated with actual hardware. From the calibrated model, numerical predictions were generated. The predictions were experimentally validated.

From the Section 3.3. developments, the rotation equation of motion was determined as

$$M_{rr}\ddot{\theta} + F(\theta,\dot{\theta}) + \sum_{i=1}^{N}\left\{\int_0^L \bar{m}_1(r+x)\phi^{(i)}(x)\,dx\right\}\ddot{q}^{(i)} = \tau \qquad (8.27)$$

where all the rigid inertial contributions are defined as,

$$M_{rr} = I_{hub} + \frac{1}{3}mL^3 + mrL^2 + mr^2L + M_{tip}(r+L)^2$$
$$\bar{m}_1 = m_1 + M_{tip}\delta(x-L)$$

and where m_1 is the mass per unit length, and $\delta()$ is the Dirac delta function.

Without any loss of generality, the effects of friction $F(\theta,\dot{\theta})$ were added into Eq. (8.27). A simple model that includes both viscous, and dry or Coulomb friction is defined as

$$F(\theta,\dot{\theta}) = C_{df}sgn(\dot{\theta}) + B_{vf}\dot{\theta}$$

where C_{df} is the Coulomb friction constant and B_{vf} is the viscous friction constant.

Based on Eq. (8.27) (the structural rotation expression) an adaptive sliding mode controller was designed. Only the linear friction term is retained to obtain

$$M_{rr}\ddot{\theta} + B_{vf}\dot{\theta} + \sum_{i=1}^{N}\left\{\int_0^L \bar{m}_1(r+x)\phi^{(i)}(x)\,dx\right\}\ddot{q}^{(i)} = \tau.$$

One can treat the flexible degree-of-freedom inertial acceleration effect as a torque disturbance, or

$$\tau = M_{rr}\ddot{\theta} + B_{vf}\dot{\theta} + \tau_d.$$

Solve for the angular acceleration to obtain

$$\ddot{\theta} = M_{rr}^{-1}\left[\tau - B_{vf}\dot{\theta} - \tau_d\right] = M_{rr}^{-1}\left[Bu - B_{vf}\dot{\theta} - \tau_d\right]. \tag{8.28}$$

Define the stable tracking sliding surface as

$$s = \dot{\theta}_r - \dot{\theta} + W(\theta_r - \theta) \tag{8.29}$$

where W is a positive definite controller gain. Take the derivative and enforce a condition of stationarity as

$$\dot{s} = \ddot{\theta}_r - \ddot{\theta} + W(\dot{\theta}_r - \theta) = 0. \tag{8.30}$$

Substituting Eq. (8.28) into Eq. (8.30) and ignoring the torque disturbance τ_d gives

$$\ddot{\theta}_r - M_{rr}^{-1}Bu + M_{rr}^{-1}B_{vf}\dot{\theta} + W(\dot{\theta}_r - \dot{\theta}) = 0. \tag{8.31}$$

Solve for the best approximate control law as

$$\widehat{B}u = \widehat{\tau} = \widehat{M}_{rr}\left(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})\right) + \widehat{B}_{vf}\dot{\theta}.$$

To satisfy the sliding condition and compensate for external disturbances as well as uncertainty in the dynamics, a robust term is added, or

$$\begin{aligned}\tau &= \widehat{\tau} + Asgn(s),\\ &= \widehat{M}_{rr}\left(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})\right) + \widehat{B}_{vf}\dot{\theta} + Asgn(s).\end{aligned} \tag{8.32}$$

By substituting Eq. (8.32) into Eq. (8.28) and following the steps outlined in Section 8.2. gives

$$\begin{aligned}\ddot{\theta} = M_{rr}^{-1}\Big[(\widehat{M}_{rr} - M_{rr})(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})) + (\widehat{B}_{vf} - B_{vf})\dot{\theta} + Asgn(s) - \tau_d\Big]\\ + \Big(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})\Big).\end{aligned} \tag{8.33}$$

Verify stability and determine the parameter adaptive control laws by choosing a Lyapunov function candidate as

$$\mathcal{V} = \frac{1}{2}M_{rr}s^2 + \frac{1}{2}\widetilde{\varphi}^T\Gamma^{-1}\widetilde{\varphi} \tag{8.34}$$

where $\widetilde{\varphi} = \varphi - \widehat{\varphi}$. Take the first derivative of Eq. (8.34) to obtain

$$\dot{\mathcal{V}} = M_{rr}s\dot{s} + \widetilde{\varphi}^T\Gamma^{-1}\dot{\widetilde{\varphi}} \;\leq\; 0. \tag{8.35}$$

Substitute Eq. (8.33) into Eq. (8.30), substitute that result into Eq. (8.35), and group terms to obtain

$$\dot{\mathcal{V}} = s^T M_{rr}\Big[\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta}) - M_{rr}^{-1}\Big[(\widehat{M}_{rr} - M_{rr})(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})) + (\widehat{B}_{vf} - B_{vf})\dot{\theta} + Asgn(s) - \tau_d\Big] - (\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta}))\Big] + \tilde{\varphi}^T\mathbf{\Gamma}^{-1}\dot{\tilde{\varphi}}.$$

Simplify as

$$\dot{\mathcal{V}} = s^T\Big[(M_{rr} - \widehat{M}_{rr})(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})) + (B_{vf} - \widehat{B}_{vf})\dot{\theta} - Asgn(s) + \tau_d\Big] + \tilde{\varphi}^T\mathbf{\Gamma}^{-1}\dot{\tilde{\varphi}}. \quad (8.36)$$

Introducing the linear in parameters property

$$(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta}))(M_{rr} - \widehat{M}_{rr}) + \dot{\theta}(B_{vf} - \widehat{B}_{vf}) = \mathbf{Y}(\theta, \dot{\theta}, \theta_r, \dot{\theta}_r, \ddot{\theta}_r)\tilde{\varphi},$$

and substituting into Eq. (8.36) gives

$$\dot{\mathcal{V}} = s^T\left[\mathbf{Y}\tilde{\varphi} - Asgn(s) + \tau_d\right] + \tilde{\varphi}^T\mathbf{\Gamma}^{-1}\dot{\tilde{\varphi}}.$$

Regroup terms as

$$\dot{\mathcal{V}} = s^T\left[-Asgn(s) + \tau_d\right] + s^T\mathbf{Y}\tilde{\varphi} + \tilde{\varphi}^T\mathbf{\Gamma}^{-1}\dot{\tilde{\varphi}}. \quad (8.37)$$

Take the transpose of the second term in Eq. (8.37) and regroup terms as

$$\dot{\mathcal{V}} = -s^T\left[Asgn(s) - \tau_d\right] + \tilde{\varphi}^T\left[\mathbf{Y}^T s + \mathbf{\Gamma}^{-1}\dot{\tilde{\varphi}}\right]. \quad (8.38)$$

To achieve Lyapunov stability ($\dot{\mathcal{V}} \leq 0$), the following conditions must be satisfied:

1. For disturbance rejection to be guaranteed,

$$A > max|\tau_d|$$

where A is a positive-definite gain term.

2. The second term of Eq. (8.38) is set to zero to determine the adaptive parameter update equations as

$$\dot{\hat{\varphi}} = \mathbf{\Gamma}\mathbf{Y}^T s. \quad (8.39)$$

Substitute all of the parameter estimates for the flexible link into Eq. (8.39) to obtain

$$\begin{Bmatrix} \dot{\widehat{M}}_{rr} \\ \dot{\widehat{B}}_{vf} \end{Bmatrix} = \begin{bmatrix} \gamma_1 & 0 \\ 0 & \gamma_2 \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta}) \\ \dot{\theta} \end{Bmatrix} s$$

where s is defined by Eq. (8.29).
Perform the multiplications to obtain the adaptive control laws

$$\dot{\widehat{M}}_{rr} = \gamma_1 \left(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})\right) s$$
$$\dot{\widehat{B}}_{vf} = \gamma_2 \dot{\theta} s.$$

Stability is checked by using Lyapunov's direct method. It can be shown that the tracking error e is asymptotically stable while the velocity tracking error $\dot{e}$, and the parameter estimates φ are bounded[2]. This step is left to the reader as an exercise.

Summarizing the adaptive control law gives

$$\tau = Bu$$
$$= \widehat{M}_{rr} \left(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})\right) + \widehat{B}_{vf}\dot{\theta} + A tanh(\beta s).$$

As recommended in the Example 8.1, use $sgn(s) \approx tanh(\beta s)$.

The update equations are given as

$$\dot{\widehat{M}}_{rr} = \gamma_1 \left(\ddot{\theta}_r + W(\dot{\theta}_r - \dot{\theta})\right) \left(\dot{\theta}_r - \dot{\theta} + W(\theta_r - \theta)\right)$$
$$\dot{\widehat{B}}_{vf} = \gamma_2 \dot{\theta} \left(\dot{\theta}_r - \dot{\theta} + W(\theta_r - \theta)\right).$$

Numerical simulation predictions

A MATLAB® simulation model was constructed for the ASMC system by incorporating the calibrated mathematical model's physical parameters. System identification was performed by using the numerical optimization techniques and the results explained in Example 4.3. The three parameters of interest were the hub inertia $\widetilde{I}_{hub}$, the viscous friction constant $\widetilde{B}_{vf}$, and the Coulomb friction constant $\widetilde{C}_{df}$. Controller gains were selected based on the previously developed stability criteria and are listed in Table 8.3.

Reference inputs (based on a cubic spline trajectory) were used to slew the flexible link from 0° to 45° in 0.2 seconds. The simulation results are shown in Figures 8.3–8.6. Figure 8.3 contains plots of the hub angle and hub velocity responses. There was a small amount of overshoot and a fast decay. Figure 8.4 contains the flexible link root strain and root strain rate responses. After the initial ring between $\pm 700\mu$ strain, a small amount of residual ($\pm 50\mu$ strain) persists. Figure 8.5 contains the tip acceleration response and torque input. The tip acceleration response showed similar characteristics to the root strain response, but with an additional amount of overlay chatter. Because a simple Coloumb friction model (consisting of a hard nonlinearity that in reality would not be present–see the experimental validation section) was used, the overlay chatter was present. Figure 8.6 shows the adaptive estimates for mass and viscous damping, respectively.

Based on the assumed model the simulation demonstrated stable and acceptable tracking performances. Initially, the residual vibrations would be a cause for alarm, but the cantilever mode shape does not take into account the interaction between the flexible link and the servo hub assembly. Further refinements to the model are needed to capture the inherent damping present in the actual hardware setup (see the experimental validation section). The adaptive sliding mode controller's added benefits are presented next.

Table 8.3. Adaptive sliding mode controller gains

Controller	W	A	β	γ_1	γ_2
ASMC	30.0	30.0	0.5	0.001	0.0001

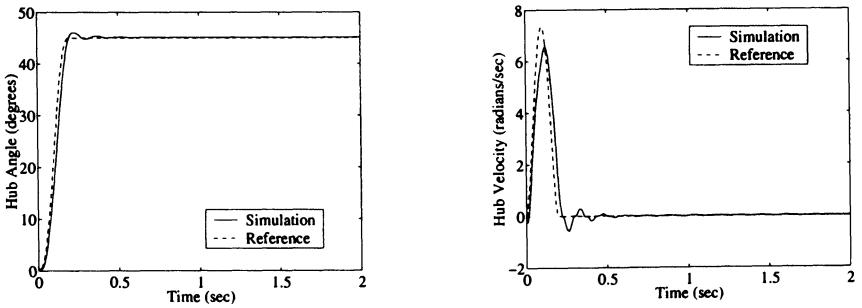

Figure 8.3. Simulation hub angle and angle rate responses

Experimental validation

The ASMC algorithm was tested on a slewing flexible link. The controller gains were empirically selected based on previously developed stability criteria. Figure 8.7 shows the direct-drive slewing structure experimental apparatus. A MATLAB$_{®}$/dSPACE$_{®}$ control/data acquisition architecture provided a rapid prototyping environment. The hub angle was measured with a standard angular encoder. The strain measurement was recorded with a half-bridge strain gauge set-up. An accelerometer was attached at the tip to monitor vibrations. The link properties are given in Table 8.4.

The slewing flexible link was subjected to a cubic spline trajectory reference input from 0° to 45° in 0.2 seconds. To test the robustness of the adaptive control

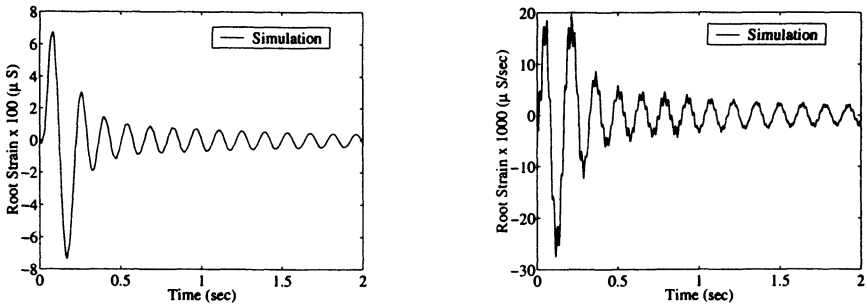

Figure 8.4. Simulation root strain and strain rate responses

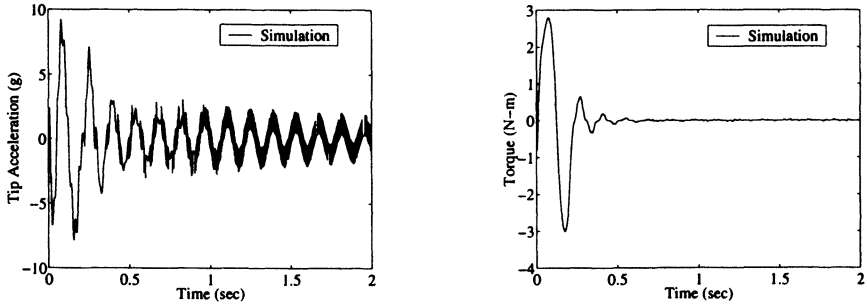

Figure 8.5. Simulation tip acceleration response and required torque input

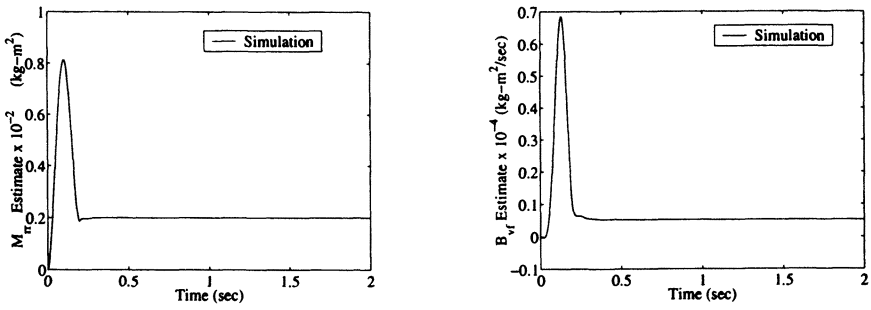

Figure 8.6: Simulation adaptation mass estimate and viscous damping estimate responses

algorithm, a large tip mass was added. Figure 8.8 contains plots of the hub angles with and without payload and the desired response is shown in the left set of plots. Although the adaptive control algorithm showed signs of deteriorating perfor-

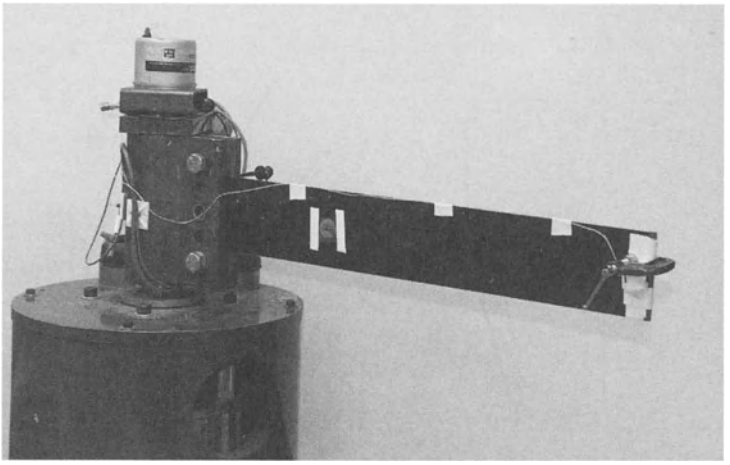

Figure 8.7. Adaptive SMC direct-drive slewing beam experimental set-up

mance, i.e., settling time increased from 0.22 seconds to 0.4 seconds, the algorithm still maintained stability with minimum overshoot. Figure 8.8 contains the tracking errors in the right set of plots. Note, the tracking errors increased from 11° to a peak of 25°. Also, the increased payload case showed a steady-state offset of approximately 1°. The root strain and tip acceleration responses are shown in Figure 8.9. Although the tip mass increased the peak strain (600μS to 1000μS), the peak tip acceleration showed a reduction (8.0G to 3.5G). For all link sensor responses, the settling times remain relatively constant. Figure 8.10 contains the torque input profiles for both cases, with and without payload, respectively. Figure 8.11 shows the adaptive parameter estimates $\widehat{M}_{rr}$ and $\widehat{B}_{vf}$ responses, respectively.

With the addition of a payload, both the mass and the damping estimates show an increase. These results demonstrate the robustness of the adaptive controller to parameter variations. The main focus of the adaptive controller is slewing the flexible link, that is, the rigid motion. Because of the robust term that was added in Eq. (8.32) and the passive damping in the hub, the link flexibility remained stable during the slew. Additionally, the adaptive controller did not excite any high-frequency modes because $sgn(s) \approx tanh(\beta s)$.

Table 8.4. Adaptive SMC direct-drive flexible link properties

Link Parameters	Symbol	Value	Unit
Length	L	45.72	cm
Width	w	7.62	cm
Thickness	t	0.1829	cm
Hub Radius	r	8.89	cm
Mass Density	ρ_m	1522	kg/m^3
Tip Mass	M_t	0.100	kg
Beam Stiffness	EI	1.497	$N \cdot m^2$
Hub Inertia	I_{hub}	0.0672	$kg \cdot m^2$
Viscous Damping	b_{vf}	1.34^{-4}	$kg \cdot m^2/s$

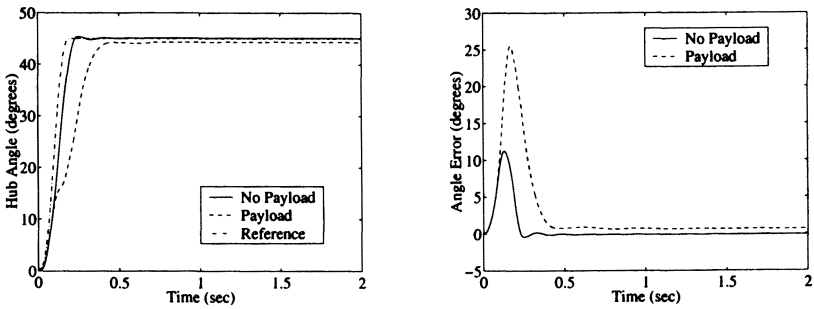

Figure 8.8. Hub angle and angle error responses

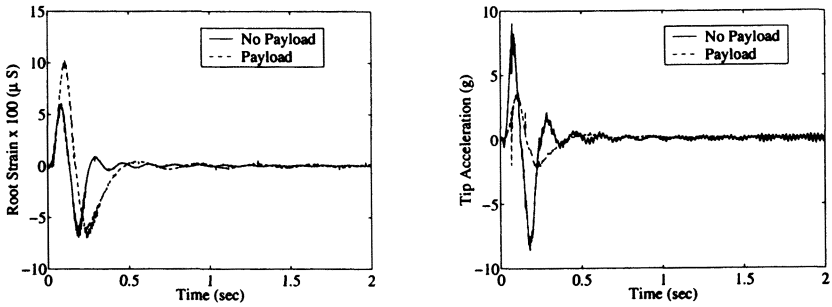

Figure 8.9. Root strain and tip acceleration responses

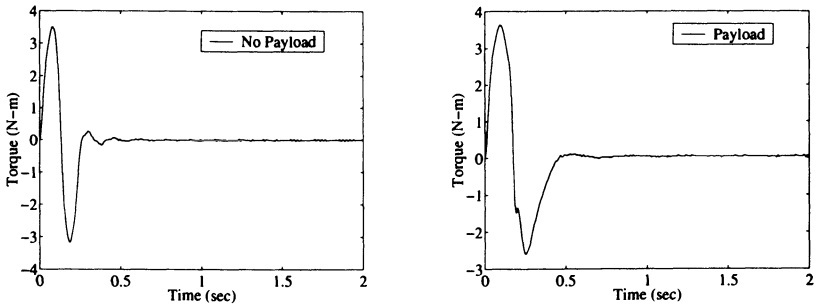

Figure 8.10. Command hub torque input

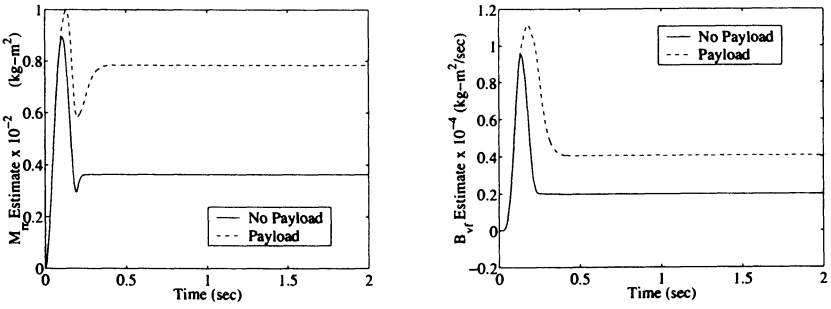

Figure 8.11. Adaptation parameter estimate responses

8.4. Chapter 8 Summary

In this final chapter an adaptive control technique that utilizes passivity concepts and incorporates SMC developments was introduced. By combining both the robust term (from the SMC development) with adaptation, both unmodeled dynamics/external disturbances and parameter variations are addressed. Again, Lyapunov's direct method was employed to prove stability and determine tracking criteria. Two examples were developed that illustrate this methodology. The final example included both a numerical simulation and an experimental validation discussion involving a slewing flexible link. For further developments in both nonlinear and adaptive control systems, as they are applied to flexible robotic systems, the interested reader is encouraged to consult the literature.

8.5. Chapter 8 References

1. J.-J.E. Slotine, and W. Li., *Applied Nonlinear Control*, Prentice Hall, Inc., Englewood Cliffs, N.J., 1991.
2. F. L. Lewis, C. T. Abdallah, and D. M. Dawson, *Control of Robot Manipulators*, Macmillan Publishing Co. New York, NY, 1993.
3. G. G. Parker, *Control Techniques for Multibody Flexible Structures Modelled by a Method of Quadratic Modes*, Ph.D. Dissertation: State University of New York at Buffalo, April, 1994.
4. D. G. Wilson, *Nonlinear/Adaptive Control Architectures with Active Structures for Flexible Manipulators*, Ph.D. Dissertation: Mechanical Engineering Department, University of New Mexico, Albuquerque, NM, May, 2000.
5. D. Ertur, Y. Li, and C. D. Rahn, "Adaptive Vibration Isolation for Flexible Structures," In *DE-Vol. 97/DSC-Vol. 65, Vibration and Noise Control, ASME 1998*, pages 49-55, Opryland, TN., Nov. 1998.
6. M. Kristic, I. Kanellakopoulous, and P. Kokotovic, *Nonlinear and Adaptive Control Design*, John Wiley & Sons, Inc., New York, NY, 1995.
7. Y. Landau, *Adaptive Control: The Model Reference Approach*, Marcel Dekker, New York, N.Y., 1979.
8. J. J. Craig, *Adaptive Control of Mechanical Manipulators*, Addison-Wesley Publishing Co., Inc., Reading, MA, 1985.
9. K. Astrom and B. Wittenmark, *Adaptive Control*, Addison Wesley Publishing Co., Inc., Reading, MA, 1985.
10. H. Kaufman, I. Bar-Kana, and K. Sobel, *Direct Adaptive Control Algorithms Theory and Applications*, Springer-Verlag, Inc., New York, NY, 1994.
11. P. A. Ioannou, and J. Sun, *Robust Adaptive Control*, Prentice-Hall, Inc., 1996.
12. G. Tao, P. V. Kokotovic, *Adaptive Control of Systems with Actuator and Sensor Nonlinearities*, John Wiley & Sons, Inc., New York, NY 1996.
13. A. Kelar, and S. Joshi, *Control of Nonlinear Multibody Flexible Structures*, Springer-Verlag, Inc., New York, N.Y., 1996.
14. C. Canudas de Wit, B. Siciliano, and G. Bastin, *Theory of Robot Control*, Springer-Verlag, Inc., New York, NY, 1997.
15. H. R. Asare, and D. G. Wilson, "Design of Computed Torque Model Reference Adaptive Control for Space-Based Robotic Manipulators," In *ASME Winter Annual Meeting*, Anaheim, CA., Dec. 1986.

8.6. Chapter 8 Problems

Homework 8.1. From Section 8.3.
Validate the adaptive SMC design in Example 8.1 for the DC motor driven, large angle pendulum system. Develop a numerical simulation and implement the dynamic equations of motion, sliding surface, adaptive SMC law, and the adaptive

control parameter update equations. Use the same relevant numerical values presented in Tables 8.1 and 8.2, respectively. Plot the angular position, angular rate, motor voltage requirement, mass estimate, and stiffness estimate responses.

Homework 8.2. From Section 8.3.
Design and simulate an adaptive SMC design for a slewing horizontal link whose dynamic equations of motion are given as

$$\tau = I\ddot{\theta} + \tau_d$$

where τ is the torque input, I is the moment of inertia about the hub, $\ddot{\theta}$ is the angular acceleration, and τ_d is the torque disturbance. Perform the design by following the steps given in Section 8.1. Select numerical values for I and τ_d, and perform a numerical simulation. Select controller gains for a critically damped response. Plot the angular position, velocity, torque, and inertial estimate. How well does the ASMC perform if τ_d becomes time varying? *Hint*: Should the controller gain A be adjusted? If τ_d represents the flexible coupling term identified in Example 8.2, then what problem was just solved?

Homework 8.3. From Section 8.3.
Consider a similar class of problems of the form

$$\mathbf{M}(\mathbf{x})\ddot{\mathbf{x}} + \mathbf{N}_m(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} = \mathbf{B}\mathbf{u} \tag{8.40}$$

where $\mathbf{M}(\mathbf{x})$ is now an *implicit* function of time rather than a constant, and the nonlinear dynamic matrix $\mathbf{N}_m$ is related to the original $\mathbf{N}$ vector (from Eq. 8.40) by the expression

$$\mathbf{N}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{N}_m(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}}$$

such that the matrix $\dot{\mathbf{M}} - 2\mathbf{N}_m$ is skew-symmetric. This property[2,14] is useful during the stability analysis. Verify that the same adaptation update equations (see Eq. 8.17) result. Show all of your steps.

The Lyapunov function candidate suggested by Slotine and Li[1] becomes

$$\mathcal{V} = \frac{1}{2}\mathbf{s}^T\mathbf{M}(\mathbf{x})\mathbf{s} + \frac{1}{2}\tilde{\boldsymbol{\varphi}}^T\boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\varphi}}.$$

Taking the derivative with respect to time yields

$$\dot{\mathcal{V}} = \mathbf{s}^T\mathbf{M}(\mathbf{x})\dot{\mathbf{s}} + \frac{1}{2}\mathbf{s}^T\dot{\mathbf{M}}(\mathbf{x})\mathbf{s} + \tilde{\boldsymbol{\varphi}}^T\boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\varphi}}} \tag{8.41}$$

Show that by determining $\mathbf{M}(\mathbf{x})\dot{\mathbf{s}}$ by rearranging Eq. (8.40) and constructing

$$\mathbf{M}(\mathbf{x})\dot{\mathbf{s}} = \mathbf{Y}(\cdot)\boldsymbol{\varphi} - \mathbf{B}\mathbf{u} - \mathbf{N}_m(\mathbf{x}, \dot{\mathbf{x}})\mathbf{s}$$

where

$$\mathbf{Y}(\cdot)\boldsymbol{\varphi} = \mathbf{M}(\mathbf{x})\left(\ddot{\mathbf{x}}_r + \mathbf{W}(\dot{\mathbf{x}}_r - \dot{\mathbf{x}})\right) + \mathbf{N}_m(\mathbf{x}, \dot{\mathbf{x}})\left(\dot{\mathbf{x}}_r + \mathbf{W}(\mathbf{x}_r - \mathbf{x})\right),$$

and substituting into Eq. (8.41) gives

$$\mathcal{V} = \mathbf{s}^T(\mathbf{Y}(\cdot)\boldsymbol{\varphi} - \mathbf{Bu}) + \frac{1}{2}\mathbf{s}^T\left(\dot{\mathbf{M}}(\mathbf{x}) - 2\mathbf{N}_m(\mathbf{x}, \dot{\mathbf{x}})\right)\mathbf{s} + \tilde{\boldsymbol{\varphi}}^T\boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\varphi}}}. \tag{8.42}$$

Note, by applying the skew-symmetric property to the second term, it becomes zero.

Next, select the control law as

$$\mathbf{Bu} = \mathbf{Y}(\cdot)\hat{\boldsymbol{\varphi}} + \mathbf{A}sgn(\mathbf{s}).$$

By substituting into Eq. (8.42) verifies that $\mathbf{A}$ must be positive definite and gives the same adaptive update equations (see Eq. 8.17).

Appendix A: VFO2AD Optimization

The VF02AD Fortran optimization programs are part of the Harwell Subroutine Library and are downloadable from *http://hsl.rl.ac.uk/archive/hslarchive.html.*

The VFO2AD program was written to calculate the least value of a function of several variables subject to general constraints on the values of the variables. An iterative method is used. Each iteration minimizes a quadratic approximation to the Lagrangian function subject to linear approximations to the constraints. The second derivative matrix of the Lagrangian function is estimated automatically. In order to force convergence, when the initial values of the variables are far from the solution set, a line-search procedure is used.

VF02AD is a set of FORTRAN subroutines for minimizing an arbitrary performance index $\phi(\boldsymbol{\xi})$ as a function of a vector of parameter constants $\boldsymbol{\xi}$. The performance index is subject to equality constraints of the form $\boldsymbol{\Psi}^{(1)}(\boldsymbol{\xi}) = \mathbf{0}$, and will also handle inequality constraints of the form $\boldsymbol{\Psi}^{(2)}(\boldsymbol{\xi}) \geq 0$. For example, if one were given an inequality constraint of the form $x_1(\boldsymbol{\xi}, t_f) \leq 10$, then in VF02AD it would be expressed as $\Psi_1^{(2)}(\boldsymbol{\xi}) = 10 - x_1(\boldsymbol{\xi}, t_f)$. If $\Psi_1^{(2)}(\boldsymbol{\xi}) < 0$, then VF02AD would continue to operate on it until the inequality is obeyed.

The VFO2AD subroutine requires a driver routine written by the user to do several tasks such as initialization, handle I/O, and call VF02AD. A first guess of your parameter vector $\boldsymbol{\xi}_0$ is also needed in the driver. Embedded inside the driver is the ***optimization loop*** that performs the following:

1. Call a subroutine to compute the scalar $\phi(\boldsymbol{\xi})$ and the vector $\boldsymbol{\Psi}(\boldsymbol{\xi}) = [\boldsymbol{\Psi}^{(1)}(\boldsymbol{\xi}) \;\; \boldsymbol{\Psi}^{(2)}(\boldsymbol{\xi})]$.

2. Call a subroutine to compute derivatives $\phi_{\boldsymbol{\xi}}(\xi_i)$, $\boldsymbol{\Psi}_{\boldsymbol{\xi}}(\xi_i)$ (typically finite differences).

3. Call VF02AD(*n*, *m*, *meq*, $\xi(=\xi_0), \phi, \phi_\xi, \Psi, \Psi_\xi^T$, *lcn*, *maxfun*, *acc*, *iprint*, *inf*, *w*, *lw*).
   (Implicitly VF02AD will check to see if input errors exist, too many function evaluations, i.e., ϕ's and Ψ's have been computed, or if the solution has been reached to a given tolerance). Once the given tolerance is met the optimization loop is completed successfully.

The final task of the driver is to perform output operations such as screen printouts and saving time domain results.

The formal call to VF02AD is

CALL VF02AD(*n, m, meq, parms, phi, phiz, psi, psizT, lcn, maxfun, acc, iprint, inf, w, lw*) where the following arguments are defined as:

- n - the number of parameters
- m - the length of Ψ (number of equality plus inequality constraints)
- *meq* - the length of $\Psi^{(1)}$ (number of equality constraints)
- *params,phi,phiz,psi,psizT* - variable names $\xi, \phi, \phi_\xi, \Psi, \Psi_\xi^T$ (all previously explained) Note, Ψ_ξ^T is specified for in the FORTRAN compiler specifications `psiT(i,j)`, the `i` index would correspond to the parameter ξ_i, and `j` would correspond to the constraint $\Psi_j(\xi)$
- *lcn* - set to at least $n + 1$
- *maxfun* - maximum number of function evaluations (i.e., ϕ and Ψ computations)
- *acc* - convergence tolerance on gradient norm (typically set to $\leq 10^{-3}$)
- *iprint* - output interval for intermediate information from VF02AD; when $= -\#$: no output; $= 0$: prints diagnostics only, and when $= +\#$: prints diagnostics and ϕ, Ψ, and ξ every $\#$ iteration
- *inf* - status variable, set to -1 initially outside of optimization loop
- w - vector work array and must be dimensioned $5n^2 + 19n + 14 + 4m + max(m, 3n + 3)$
- *lw* - length of work array $5n^2 + 19n + 14 + 4m + max(m, 3n + 3)$

The VF02AD software has been used to solve most of the optimization homework problems found throughout this book. VF02AD is normally considered more efficient with higher accuracy, less overhead and quicker running code than its MATLAB® Optimization Toolbox counterpart. The advantage of the MATLAB® Optimization code is ease of implementation.

Appendix B: MATLAB® Optimization

The MATLAB® Optimization Toolbox consists of both the *constr*, and more recently, the *fmincon* constrained nonlinear optimization functions. Both of these functions are used in this book. In the MATLAB® software environment, many support routines are readily available. For example, the *ode45* variable step integration routine requires a derivative subroutine to be written and dynamic equations of motion represented as differential equations are easily integrated. Another example is the *interp1* interpolation function that allows linear interpolation tables to be implemented. The discussion will be limited to the *fmincon* function. It will include a general description, with several implementation issues, and the inclusion of MATLAB® code, which solves several of the parameterized controls optimization examples presented in Chapter 5. Further details can be found in the Optimization Toolbox documentation. The majority of this general discussion comes directly from this Toolbox documentation.

The Optimization Toolbox is a collection of functions that extend the capability of the MATLAB® numeric computing environment. Routines for many types of optimization are included in the toolbox. Some of these features related to our book topics include:

- Unconstrained nonlinear minimization.
- Constrained nonlinear minimization.
- Quadratic and linear programming.
- Nonlinear least squares and curve-fitting.
- Constrained linear least squares.

For this discussion, constrained nonlinear minimization optimization problems are the focus. Optimization techniques involve the minimization or maximization of functions. The objective function to be minimized is contained in a M-file, which is required in the optimization routines. Optimization options passed to the routines change optimization parameters. The default optimization parameters are used extensively, but can be changed through an `options` structure. Gradients are calculated by using an adaptive finite-difference method unless they are supplied in

a function. The constrained nonlinear minimization function *fmincon* can be stated as

Minimize

$$f(\mathbf{x})$$

subject to

$$\begin{aligned} \mathbf{c}(\mathbf{x}) &\leq \mathbf{0} \\ \mathbf{ceq}(\mathbf{x}) &= \mathbf{0} \\ \mathbf{Ax} &\leq \mathbf{b} \\ \mathbf{Aeqx} &= \mathbf{beq} \end{aligned}$$

for

$$\mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub}$$

where $\mathbf{x}, \mathbf{b}, \mathbf{beq}, \mathbf{lb}$, and $\mathbf{ub}$ are vectors, $\mathbf{A}$ and $\mathbf{Aeq}$ are matrices, $\mathbf{c}(\mathbf{x})$ and $\mathbf{ceq}(\mathbf{x})$ are functions that return vectors, and $f(\mathbf{x})$ is a function that returns a scalar. $f(\mathbf{x}), \mathbf{c}(\mathbf{x})$, and $\mathbf{ceq}(\mathbf{x})$ can be nonlinear functions.

The syntax function calls used for most of this problems are

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options) .

The input arguments are defined as;

- *fun* - the function to be minimized. *fun* is a function that accepts a scalar x and returns a scalar f where the objective function is evaluated at x.
- *x0* - initial guess (scalar, vector, matrix)
- *A,b* - the matrix A and vector b are coefficients of linear inequality constraints $Ax \leq b$.
- *Aeq,beq* - the matrix *Aeq* and vector *beq* are coefficients of linear equality constraints $Aeqx = beq$.
- *lb,ub* - are lower and upper bound vectors. The arguments are normally the same size as x.
- *nonlcon* - the function that computes the nonlinear inequality and equality constraints.
- *options* - an optimization options parameter structure that defines parameters used by the optimization functions.

The output argument x is defined as the solution found by the optimization function.

The goal of *fmincon* is to find the constrained minimum of a scalar function of several variables starting at an initial estimate. This is referred to as constrained nonlinear optimization or nonlinear programming.

For x = *fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)* subjects the minimization to the nonlinear inequalities $c(x)$ or the equalities $ceq(x)$ defined in *nonlcon*. The optimization is performed such that $c(x) \leq 0$ and $ceq(x) = 0$. To customize some of the optimization procedure parameters, the options structure can be specified by using the Call x = *fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)*.

Further details of the user supplied functions *fun* and *nonlcon* are reviewed. *fun* is the function to be minimized. It accepts a scalar x, returns a scalar f and the objective function is evaluated at x. *fun* is subject to the nonlinear inequality $ceq \leq 0$ and the equality $ceq = 0$ constraints implemented in the *nonlcon* function. The vector c contains the nonlinear inequalities and ceq contains the nonlinear equalities. Both are evaluated at x. The functions *fun* and *nonlcon* are specified as a function handle
x = *fmincon(@myfun,x0,A,b,Aeq,beq,lb,ub,@mycon)* where both *myfun* and *mycon* are MATLAB® functions, such as,

function f = myfun(x) ... %Compute function, value at x,

and

function [c,ceq] = mycon(x) ... %Computes c and ceq at x.

These function handles are best realized with practical examples. Included below are the program listings that solve the minimum effort horizontal slewing link examples of Chapter 5.

The reader is encouraged to reproduce the results by studying the examples and corresponding MATLAB® code, and use the MATLAB® software and toolboxes . The optimization iterations are initiated, from the MATLAB® command line, by running *oneLinkDriver*, or *oneLinkDriver1*.

MATLAB® code for the minimum effort slewing horizontal link (see Example 5.1):

```
% one link driver routine oneLinkDriver.m
global c1;global tarray;global xo;global xdesired;global tfinal;
mass=0.5;link=1.0;c1=3/(mass*link*link);tfinal=1.0;deltat=.1;
tarray=[0:deltat:tfinal];xo=[0 0 0]; % theta0, thetadot0 u^2
xdesired=[pi/2 0];pvalue=.1;params0=pvalue*ones(1,length(tarray));
A=[];b=[];Aeq=[];beq=[];
%lowerBndValue=-5.  % use for unbounded control problem
%upperBndValue=5.   % use for unbounded control problem
lowerBndValue=-1.2   % use for bounded control problem
```

```
upperBndValue=1.2    % use for bounded control problem
lowerBnds=lowerBndValue*ones(1,length(tarray));
upperBnds=upperBndValue*ones(1,length(tarray));
%params=fmincon(@perfIndex,params0,A,b,Aeq,beq,lowerBnds,upperBnds,
%@constraints)
params=fmincon('perfIndex',params0,A,b,Aeq,beq,lowerBnds,upperBnds,
'constraints')

% fun function handle:  Performance Index matlab M-file -
%perfIndex.m
function f = perfIndex(params) global xo;global p;global
tfinal;p=params;
%[t,x]=ode45(@xprime,[0:tfinal],xo(:));
[t,x]=ode45('xprime',[0:tfinal],xo(:));
[rows,cols]=size(x);f=x(rows,3);return;

% nonlcon function handle:  constraint matlab M-file - constraints.m
function [c,coneq,GC,GCeq]=constraints(params) global xo;global
xdesired;global p;global tfinal;p=params;
%[t,x]=ode45(@xprime,[0:tfinal],xo(:));
[t,x]=ode45('xprime',[0:tfinal],xo(:));
[rows,cols]=size(x);coneq(1)=x(rows,1)-xdesired(1);
coneq(2)=x(rows,2);c=[];GC=[];GCeq=[];coneq  return;

% first order odes for horizontal link:  matlab file xprime.m
function xdot=xprime(t,x) global c1;global p;global tarray;
xdot(1)=x(2);u=interp1(tarray,p,t);
xdot(2)=c1*u;xdot(3)=u*u;xdot=xdot(:);  return;

% This file generates plot for horizontal link - postplt.m
[t0,x0]=ode45('xprime',[0:tfinal],xo(:));
tar0=tarray;par0=params;save hslew0 t0 x0 tar0 par0
plot(t0,x0(:,1),'-',t0,x0(:,2),'--',tar0,par0,'-.'),grid
legend('-','Angle','--','Angle Rate','-.','Torque') xlabel('Time
(sec)') ylabel('Minimum Effort Trajectories') title('Horizontal
Link Slew')
```

MATLAB® code for the minimum time slewing horizontal link (see Example 5.1):

```
% one link driver routine - minimum time, bounded controls -
%oneLinkDriver1.m
global c1;global xo;global xdesired;global tfinal; global
num_of_tvalues;
mass=.5;link=1;c1=3/(mass*link*link);tfinal=1;num_of_tvalues=17;
```

```
xo=[0 0]; % theta0, thetadot0 xdesired=[pi/2 0];pvalue=.5;
params0=[pvalue*ones(1,num_of_tvalues) tfinal];
A=[];b=[];Aeq=[];beq=[]; lowerBndValue=-1.2upperBndValue=1.2
lowerBnds=[lowerBndValue*ones(1,num_of_tvalues) .01];
upperBnds=[upperBndValue*ones(1,num_of_tvalues) 10];
%params=fmincon(@perfIndex1,params0,A,b,Aeq,beq,lowerBnds,upperBnds,
%@constraints1)
params=fmincon('perfIndex1',params0,A,b,Aeq,beq,lowerBnds,upperBnds,
'constraints1')

%fun function handle:  Performance Index matlab M-file -
%perfIndex1.m
function f = perfIndex1(params) global xo;global p;global tfinal;
lenparams=length(params);p=params(1:lenparams-1);
tfinal=params(lenparams);
%[t,x]=ode45(@xprime1,[0 tfinal],xo(:));
% tf= params(length(params))
[t,x]=ode45('xprime1',[0 tfinal],xo(:));
% tf= params(length(params))
[rows,cols]=size(x);f=tfinal; % final time is last parameter
return;

% nonlcon function handle:  constraint matlab M-file -
%constraints1.m
function [c,coneq,GC,GCeq]=constraints(params) global xo;global
xdesired;global p;global tfinal;
lenparams=length(params);p=params(1:lenparams-1);
tfinal=params(lenparams);
%[t,x]=ode45(@xprime1,[0 tfinal],xo(:));
[t,x]=ode45('xprime1',[0 tfinal],xo(:));
[rows,cols]=size(x);coneq(1)=x(rows,1)-xdesired(1);
coneq(2)=x(rows,2);c=[];GC=[];GCeq=[];coneq  return;

% first order odes for horizontal link min time:  matlab file
%xprime1.m
function xdot=xprime1(t,x) global c1;global p; global
tfinal;global num_of_tvalues;global tarray
xdot(1)=x(2);deltat=tfinal/(num_of_tvalues-1);
tarray=[0:deltat:tfinal];
u=interp1(tarray,p,t);xdot(2)=c1*u;xdot=xdot(:);  return;

% This file generates plot for horizontal link min time - postplt1.m
[t1,x1]=ode45('xprime',[0:tfinal],xo(:));
tar1=tarray;par1=params;save hslew1 t1 x1 tar1 par1
```

```
plot(t1,x1(:,1),'-',t1,x1(:,2),'--',tar1,par1,'-.'),grid
legend('-','Angle','--','Angle Rate','-.','Torque') xlabel('Time
(sec)') ylabel('Minimum Effort Trajectories') title('Horizontal
Link Slew with Bounded Control')
```

Appendix C: Hardware & Software Support

Flexible Manipulator Hardware Platforms

WAYA Research, Inc. and S. Enterprises can provide experimental hardware and software to support the material developed in this book.

For example, a multi-link modular planar flexible two-link manipulator was designed, manufactured and delivered to Sandia for flexible robot dynamic and control system studies. A key feature of the testbed is the encoder/DC motor/harmonic drive actuator assembly. The actuator is nonbackdrivable, therefore any residual vibrations are managed by flexible robot control techniques. The harmonic drive actuator is popular in many robotic applications where accurate positioning is required. For lightweight high-speed applications, link vibrations become significant and must be addressed. An external encoder is mounted on the link side. This encoder is used for both flexible robot control and actuator transmission characterization studies. The hub shaft is machined with a slot that facilitates the easy interchange of different types of flexible links. Currently shown (see Figure 1) are thin lightweight graphite/epoxy composite links. An accelerometer is mounted at the tip to monitor tip vibrations. A MATLAB$_{®}$/Simulink$_{®}$/Real time workshop$_{®}$/dSPACE$_{®}$ control/data acquisition architecture is used to provide a rapid prototyping environment. This environment accommodates efficient control algorithm implementation and validation.

WAYA Research, Inc. and S. Enterprises can provide custom software configurations compatible with similar hardware. The control system architecture and the planar two-link hardware are shown in Figure 1.

Shown in Figure 2 are additional flexible single-link configurations. The single-link configuration for the above described modular manipulator is shown on the left. Shown on the right is a flexible single-link direct-drive testbed. The direct-drive hardware was used for the final project of the UNM ME 562 course where the material presented in this book was first formally presented. The final project required the design and testing of input shaping open-loop control strategies. All of the control system software was programmed by using the C-language and implemented on a VME/VxWorks$_{®}$ platform. Other hardware platforms are possible.

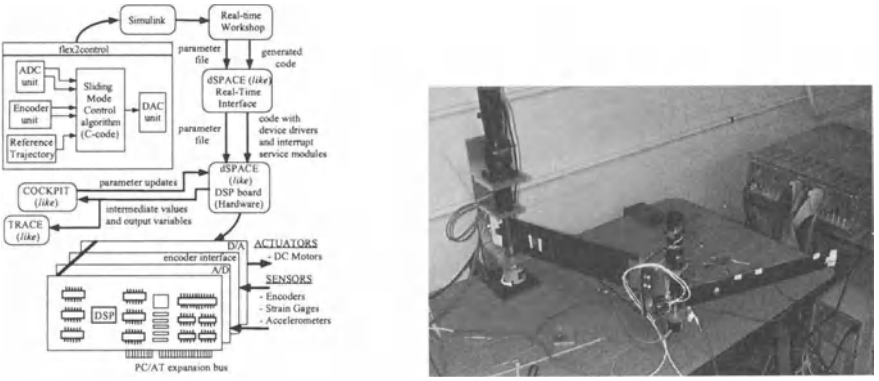

Figure 1: WAYA Research, Inc. designed, manufactured, and delivered flexible two-link robot for Sandia *By permission of WAYA Research, Inc.* ©*1995*

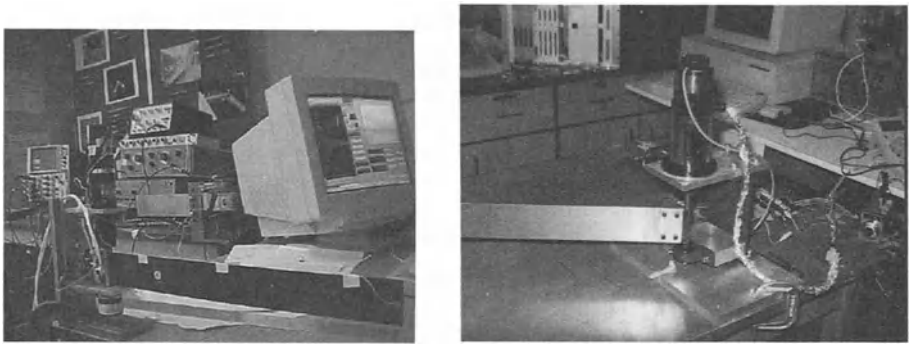

Figure 2: WAYA Research, Inc. single flexible link geared- (harmonic) drive (left) and direct-drive (right) testbed systems *By permission of WAYA Research, Inc.* ©*1995*

Control System Hardware/Software Architecture

The control system is based upon a Pentium® personal computer running the Windows® operating system. The MATLAB®, Simulink®, Real-Time Workshop®, and dSPACE®-*like* environment is employed to realize the flexible robot control law implementation (see Figure 1). The flexible robot control law is written in the C-language as a Simulink® S-function. The Simulink® S-function protocol is based on a general nonlinear state-space template. The control law accepts reference in-

puts either internally generated, such as, cubic spline, polynomial, or near-minimum time trajectories, or externally, such as, from a joystick user input. Feedback signals are provided through analog-to-digital (A/D) converters and encoder input ports. Control signals are calculated and sent out the digital-to-analog (D/A) port. A new control law model is created by the user in the Simulink® environment by using a component block diagram interface. The control law is compiled as a MEX® function and easily integrated into the remaining control architecture components. Other block components can be added to provide signal conditioning and scaling. Once the new model is completed, the entire Simulink® model is converted to a real-time executable code by using the Real-Time Workshop® module. This code also employs a dSPACE®-*like* module that provides the device drivers and real-time windows interrupt service routines. Once this step is complete, the controller software does not need to be rebuilt each time. The executable code is used for operating/commanding the flexible manipulator. The user interfaces with the controller software through the Simulink® environment running in *external mode*. This allows the user to tune parameters, such as, control system gains, select reference trajectories, start and stop locations, etc., through a COCKPIT®-*like* port. The user can also visualize signals in real-time by using a TRACE®-*like* port. This allows signal viewing by using the Simulink® oscilloscope outputs. Data acquisition is provided by using a TRACE®-*like* port. This allows the user to save the signals after a maneuver. The data is stored in a MATLAB® compatible format for later post-processing analysis. Included in this data are all of the time-domain variables available from the Simulink® model. The flexible manipulator interfaces with the Pentium® computer through a custom controller/data acquisition interface board. This board contains 8 channels of D/A, A/D, and encoder input. In addition, 32-bit digital I/O is provided. Each unique type of signal data is interfaced with the high-level controller software through device drivers. These device drivers are custom S-functions that initialize read/write to the boards D/A, A/D and encoder chips. These operations are transparent to the user. Figure 1 provides an overview of the control system hardware and software interaction.

Subject Index