NOTES ON NUMERICAL FLUID MECHANICS AND MULTIDISCIPLINARY DESIGN · VOLUME 113

ADIGMA – A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications

Results of a collaborative research project funded by the European Union, 2006–2009

Norbert Kroll Heribert Bieler
Herman Deconinck Vincent Couaillier
Harmen van der Ven
Kaare Sørensen (Editors)

Springer

ADIGMA – A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications

Results of a collaborative research project funded by the European Union, 2006–2009

Norbert Kroll, Heribert Bieler,
Herman Deconinck, Vincent Couaillier,
Harmen van der Ven, and Kaare Sørensen
(Editors)

Springer

Prof. Dr. Norbert Kroll
German Aerospace Center (DLR)
Institute of Aerodynamics and
Flow Technology
Lilienthalplatz 7
38108 Braunschweig, Germany
E-mail: norbert.kroll@dlr.de

Dr. Heribert Bieler
Airbus Deutschland GmbH
Flight Physics
Airbusallee 1
28199 Bremen, Germany
E-mail: heribert.bieler@airbus.com

Prof. Dr. Herman Deconinck
von Karman Institute for
Fluid Dynamics (VKI)
Chausee de Waterloo, 72
1640 Rhode-St-Genese, Belgium
E-mail: deconinck@vki.ac.be

Dr. Vincent Couaillier
ONERA
29, avenue de la Division Leclerc
92322 Châtillon, France
E-mail: vicent.couaillier@onera.fr

Dr. Harmen van der Ven
NLR
Anthony Fokkerweg 2
1059 CM Amsterdam
The Netherlands
E-mail: venvd@nlr.nl

Dr. Kaare Sørensen
EADS Deutschland
Military Air Systems
85077 Manching, Germany
E-mail: kaare.sorensen@eads.com

ISBN 978-3-642-03706-1 e-ISBN 978-3-642-03707-8

DOI 10.1007/978-3-642-03707-8

Notes on Numerical Fluid Mechanics
and Multidisciplinary Design ISSN 1612-2909

Library of Congress Control Number: 2010931446

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

5 4 3 2 1 0

springer.com

This page intentionally left blank

NNFM Editor Addresses

Prof. Dr. Wolfgang Schröder
(General Editor)
RWTH Aachen
Lehrstuhl für Strömungslehre und
Aerodynamisches Institut
Wüllnerstr. 5a
52062 Aachen
Germany
E-mail: office@aia.rwth-aachen.de

Prof. Dr. Kozo Fujii
Space Transportation Research Division
The Institute of Space
and Astronautical Science
3-1-1, Yoshinodai, Sagamihara
Kanagawa, 229-8510
Japan
E-mail: fujii@flab.eng.isas.jaxa.jp

Dr. Werner Haase
Höhenkirchener Str. 19d
D-85662 Hohenbrunn
Germany
E-mail: office@haa.se

Prof. Dr. Ernst Heinrich Hirschel
(Former General Editor)
Herzog-Heinrich-Weg 6
D-85604 Zorneding
Germany
E-mail: e.h.hirschel@t-online.de

Prof. Dr. Ir. Bendiks Jan Boersma
Chair of Energytechnology
Delft University of Technology
Leeghwaterstraat 44
2628 CA Delft
The Netherlands
E-mail: b.j.boersma@tudelft.nl

Prof. Dr. Michael A. Leschziner
Imperial College of Science
Technology and Medicine
Aeronautics Department
Prince Consort Road
London SW7 2BY
U.K.
E-mail: mike.leschziner@ic.ac.uk

Prof. Dr. Sergio Pirozzoli
Università di Roma “La Sapienza”
Dipartimento di Meccanica e Aeronautica
Via Eudossiana 18
00184, Roma, Italy
E-mail: sergio.pirozzoli@uniroma1.it

Prof. Dr. Jacques Periaux
38, Boulevard de Reuilly
F-75012 Paris
France
E-mail: jperiaux@free.fr

Prof. Dr. Arthur Rizzi
Department of Aeronautics
KTH Royal Institute of Technology
Teknikringen 8
S-10044 Stockholm
Sweden
E-mail: rizzi@aero.kth.se

Dr. Bernard Roux
L3M – IMT La Jetée
Technopole de Chateau-Gombert
F-13451 Marseille Cedex 20
France
E-mail: broux@l3m.univ-mrs.fr

Prof. Dr. Yurii I. Shokin
Siberian Branch of the
Russian Academy of Sciences
Institute of Computational
Technologies
Ac. Lavrentyeva Ave. 6
630090 Novosibirsk
Russia
E-mail: shokin@ict.nsc.ru

This page intentionally left blank

Preface

Within the ACARE Vision 2020, ambitious goals have been set for air traffic of the next decades. These include a reduction of emissions by 50% and a decrease of the perceived external noise level by 10–20 dB. Continuous improvement of conventional technologies will not be sufficient to achieve these goals, however, a technological leap forward is required. The combined efforts of industry and academia will be necessary to harness new flow control technologies and entirely new configurations for use in future aircraft design and development. This will require significant improvement and enhancement of the capabilities and tools of numerical simulation, which has become a key technology in recent years.

Although numerical simulations of entire aircraft configurations are routinely performed in industry today, the time required is still on the order of hours and days, posing a significant obstacle to aerodynamic design and keeping it at a conservative level. Consequently, enhanced CFD capabilities for reducing design cycle and cost are indispensable for industry. The majority of the aerodynamic simulation tools currently used in the aeronautical industry for routine applications are based on second-order finite volume methods. Being bound to second-order numerical schemes the simulation of real-life applications requires a large amount of mesh points and computing time. Significantly fewer degrees of freedom are required for higher-order methods than for classical second-order schemes to reach the same level of accuracy. Hence the development of higher-order methods for Euler and Navier-Stokes equations is currently a hot research topic all over the world. However, in order to be competitive, these methods have to be designed in such a way that the associated increased computational complexity is more than balanced.

In order to add a major step towards the development of next generation CFD tools with significant improvements in accuracy and efficiency the specific targeted research project ADIGMA (Adaptive Higher-Order Variational Methods for Aerodynamic Application in Industry) was initiated within the 3^{rd} Call of the 6^{th} European Research Framework Programme. The goal of ADIGMA was the development and utilization of innovative adaptive higher-

order methods for the compressible flow equations enabling reliable, mesh independent numerical solutions for aerodynamic applications in aircraft design. A critical assessment of the newly developed methods for industrial aerodynamic applications should demonstrate their potential compared to the classical approaches and should identify the best numerical strategies for integration as major building blocks for the next generation of industrial flow solvers With the help of a highly skilled consortium, well balanced between upstream research, applied research and aerospace industry, the ADIGMA project was aiming at scientific results and algorithms/methods which are novel in an industrial environment.

After a general project overview (Chapter 1), this volume compiles technical papers of the ADIGMA partners, in which the major research activities and achievements are discussed (Chapter 2). Research areas covered are the development of continuous and discontinuous higher-order finite element methods, the development and enhancement of corresponding solution algorithms as well as the development of innovative adaptation techniques. The results and findings of the industrial assessment are discussed in chapter 3. Finally, the major conclusions drawn from the collaborative European research effort and recommendations for future research paths are presented.

The editors would like to express their particular thanks to Dr. D. Knörzer, the European Commissions scientific officer of the ADIGMA project, for his constant technical interest and administrative help.

Thanks are due to all partners who have contributed in the context of the ADIGMA project in a very open and collaborative manner. The knowledge and engagement of each individual contributed to the success and world wide appreciation of the ADIGMA project.

Finally, the editors would like to express gratitude to M. Wagler and Dr. J. Held for technical support in compiling this book. Acknowledgements are due to Prof. Dr. W. Schröder, the general editor of the Springer Series "Notes on Numerical Fluid Mechanics and Multidisciplinary Design", and to his colleague A. Hartmann for their help and editorial advice.

February 2010

Norbert Kroll, Braunschweig
Heribert Bieler, Bremen
Herman Deconinck, Rhode-St-Genése
Vincent Couaillier, Châtillon Cedex
Harmen van der Ven, Amsterdam
Kaare Sørensen, Manching

Contents

II.2 Solution Strategies

II.3 Adaptation and Error Estimation

III Industrial Assessment of Newly Developed Technologies

IV Conclusion and Recommendations

Chapter 1
The ADIGMA Project

Norbert Kroll

Abstract. Computational Fluid Dynamics is a key enabler for meeting the strategic goals of future air transportation. However, the limitations of today's numerical tools reduce the scope of innovation in aircraft development, keeping aircraft design at a conservative level. Within the 3rd Call of the 6th European Research Framework Programme, the strategic target research project ADIGMA was initiated. The goal of ADIGMA was the development and utilization of innovative adaptive higher-order methods for the compressible flow equations, enabling reliable, mesh independent numerical solutions for large-scale aerodynamic applications in aircraft design. A critical assessment of the newly developed methods for industrial aerodynamic applications allowed the identification of the best numerical strategies for integration as major building blocks for the next generation of industrial flow solvers. In order to meet the ambitious objectives, a partnership of 22 organizations from universities, research organizations and aerospace industry from 10 countries with well proven expertise in CFD was set up, guaranteeing high level research work with a clear path to industrial exploitation. The project started September 2006 and finished at the end of 2009.

1 Introduction

Computational Fluid Dynamics (CFD) has become a key technology in the development of new products in the aeronautical industry. During the last years the aerodynamic design engineers have progressively adapted their way-of-working to take advantage of the possibilities offered by new CFD capabilities based on the solution of the Euler and Reynolds averaged Navier-Stokes (RANS) equations. Significant improvements in physical modelling and solution algorithms have been as important as the enormous increase of computer power to enable numerical simulations in all

Norbert Kroll
German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology,
Lilienthalplatz 7, 38108 Braunschweig, Germany
e-mail: Norbert.Kroll@dlr.de

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 1–9, 2010.
springerlink.com

stages of aircraft development. In particular, better automation of mesh generation techniques due to unstructured mesh technology and a generalized block-structured grid approach with non-matching and overlapping grids resulted in the ability to predict the flow physics and aerodynamic data of highly complex configurations.

However, despite the progress made in CFD, in terms of user time and computational resources, large aerodynamic simulations of viscous high Reynolds number flows around complex aircraft configurations are still very expensive and time consuming. The requirement to reliably achieve results at a sufficient level of accuracy within short turn-around time places severe constraints on the application of CFD for aerodynamic data production, and the integration of high-fidelity methods in multidisciplinary simulation and optimization procedures. The limitations of today's numerical tools used in industry reduce the scope of innovation in aircraft development, keeping aircraft design at a conservative level. Consequently, enhanced CFD capabilities for reducing design cycle and cost are indispensable for industry.

2 State of the Art

The majority of the aerodynamic simulation tools used in the aeronautical industry for routine applications are based on second-order finite volume methods [4]. In the case when complex configurations are considered very often the accuracy of these methods ranges between first and second-order due to irregular and highly stretched meshes. The results of the AIAA Drag Prediction Workshops DPW I, DPW II, DPW III and DPW IV (see e.g. [3, 2, 5]) indicate that CFD technology currently in use may not produce sufficiently accurate results on meshes with typical grid sizes that are used in an industrial environment. Nearly mesh independent solutions are required to achieve confidence in numerical simulations. The separation of numerical and modelling errors allows systematic statements concerning the correctness or deficits of the physical modelling (turbulence and transition) for the flow problem under consideration. For meeting these objectives with second-order methods, very fine meshes with a large number of grid points are required which, in the case of complex applications, lead to enormous computing times. Higher-order methods and reliable adaptation techniques are the appropriate strategies to significantly reduce computational effort while maintaining accuracy. The advantage of higher-order methods are expected to become even more profound for DES/LES simulations due to the very high mesh resolution demands required for this type of simulation.

Over the last years the development of higher-order methods for Euler and Navier-Stokes equations has been a hot research topic all over the world. Significantly fewer degrees of freedom are required for higher-order methods than for classical second-order schemes to reach the same level of accuracy. However, in order to be competitive, these methods have to be designed in such a way that the associated increased computational complexity is more than balanced. Various approaches are known in the literature including ENO/WENO reconstruction schemes for finite volume methods, Discontinuous Galerkin finite element methods, spectral

difference methods, spectral finite volume and difference methods and residual distribution schemes. Recent reviews of the state of art concerning higher-order accurate methods for aerodynamics can be found for example in [1] and [6]. Despite the advantages and capabilities, the higher-order methods are not yet mature and current implementations are subject to strong limitations for their applications to large scale industrial problems related to aerodynamic aircraft design. Crucial aspects are efficiency (both in terms of memory storage and computing time) and robustness in particular for turbulent high Reynolds number flows, higher-order boundary representation and the preservation of monotonicity over discontinuities. Significant further research is required to overcome current obstacles and to push higher-order methods into industrial design processes.

Another well known strategy for minimizing the cost of a computational simulation while achieving a given level of accuracy is adaptive mesh refinement. The basic idea is to locally refine the mesh in regions which most adversely affect the accuracy of the solution and to coarsen the mesh in more benign areas. Local mesh refinement is available in many of the finite volume codes used by the aeronautical industry. A number of adaptation techniques have been developed to refine and de-refine isotropic volume meshes driven by feature-based sensors. They have clearly demonstrated their capability for relevant industrial applications. However, continuous local refinement of the dominant features of the flow does not necessarily guarantee that certain measures of the global error will be simultaneously reduced. Research is still ongoing to find computationally efficient and reliable adaptation sensors and error estimators. Recent work on adjoint methods has shown a lot of promise regarding reliable mesh adaptation. Indeed, adjoint methods have enabled the development of error estimators for general functionals of the solution such as lift or drag. However, further research is necessary in order to utilize this novel adaptation approach for industrial use. This in particular includes applications to turbulent high Reynolds number flows around 3D complex configurations and to multiple target quantities. Moreover, in order to fully explore the capabilities of adaptation, significant development activities with respect to the hp-refinement as well as anisotropic refinement for compressible flows are required.

3 Goals and Major Objectives

In order to add a major step towards the development of next generation CFD tools with significant improvements in accuracy and efficiency the specific target research project ADIGMA was initiated within the 3rd Call of the 6th European Research Framework Programme. The main objective of the ADIGMA project was the development and utilization of innovative adaptive higher-order methods for the compressible flow equations, enabling reliable, mesh independent numerical solutions for large-scale aerodynamic applications in aircraft design. A critical assessment of the newly developed methods for industrial aerodynamic applications allowed the identification of the best numerical strategies for integration as major building blocks for the next generation of industrial flow solvers.

The ADIGMA project concentrated on technologies showing the highest potential for efficient higher-order discretizations. These are Discontinuous Galerkin (DG) methods and Continuous Residual Distribution (CRD) schemes. The main scientific objectives of the ADIGMA project are summarized as follows:

- Further development and improvement of key parts of higher-order space discretization methods for compressible Euler, Navier-Stokes and RANS equations
- Development of higher order space-time discretizations for unsteady flows including moving geometries
- Development of novel solution strategies to improve efficiency and robustness of higher-order methods, enabling large-scale aerodynamic applications
- Development of reliable adaptation strategies including error estimation, goal-oriented isotropic and anisotropic mesh refinement and the combination of mesh refinement with local variation of the order of accuracy (hp-refinement)
- Utilization of innovative concepts in higher-order approximations and adaptation strategies for industrial applications
- Critical assessment of newly developed adaptive higher-order methods for industrial aerodynamic applications; measurement of benefits compared to state-of-the-art flow solvers currently used in industry
- Identification of the best strategies for the integration as major building blocks for the next generation industrial flow solvers.

4 Partner Consortium

The ADIGMA consortium was comprised of 22 organizations which included the main European aircraft manufacturers, the major European research establishments and several universities, all being well recognized for playing an active role in the development and utilization of advanced high fidelity CFD methods for aerodynamic applications. The role of the different organizations was quite complementary. Universities were dealing with upstream research and their main objective was to provide new technologies with improved capabilities. The national research centers were addressing applied research and thus closing the gap between upstream research and industry. In terms of computational methods, new algorithms and technologies developed at universities are adapted and enhanced for large scale applications. The role of industry covered the specification of requirements for future CFD tools and the final assessment of newly developed technologies based on industry relevant application.

Industrial partners in ADIGMA were Alenia Aeronautica (Italy), Airbus (France and Germany), Dassault Aviation (France), EADS-MAS (Germany) and CENAERO (Belgium). Research organizations involved were ARA (United Kingdom), DLR (Germany), INRIA (France), NLR (The Netherlands), ONERA (France), and VKI (Belgium). Participating Universities were Universit degli Studi di Bergamo (Italy), Ecole Nationale Suprieure d'Arts et Mtiers Paris (France), University of Nottingham (United Kingdom), Charles University Prague (Czech Republic), University of Wales Swansea (United Kingdom), University of Stuttgart (Germany), Uppsala

University (Sweden), University of Twente (The Netherlands), Warsaw University of Technology (Poland), Nanjing University (China). The project was co-ordinated by DLR.

5 Technical Project Description

As mentioned above the project focused on the discretization and solver parts of compressible CFD codes, modelling issues were not explicitly treated. The technical work in ADIGMA was split into 5 work packages (WP) (see Fig. 1).

In WP2 industrial partners specified the requirements and the evaluation procedure for the newly developed methods. A test case suite of increasing complexity was specified including the necessary data in order to provide a firm basis for comparison at midterm and the end of the project.

Work package WP3, the core of the ADIGMA project, aimed at the improvement and enhancement of higher-order discretizations for the solution of the Navier-Stokes equations at high Reynolds numbers, covering the flow regimes of aeronautical applications. From the state-of-the-art it became clear that two variational technologies have the highest chance for success, namely Discontinuous Galerkin (DG) methods and Continuous Residual-Based discretizations. The first category uses a discontinuous polynomial representation in space, generalizing to finite elements the well known first and second-order finite volume methods. The second approach is a class of continuous-in-space-finite element methods including stabilized finite element methods like Galerkin Least Squares, Residual Distribution Schemes and Residual-Based Compact finite volume methods. Although these methods had shown their potential for improved accuracy, many aspects are incomplete and need

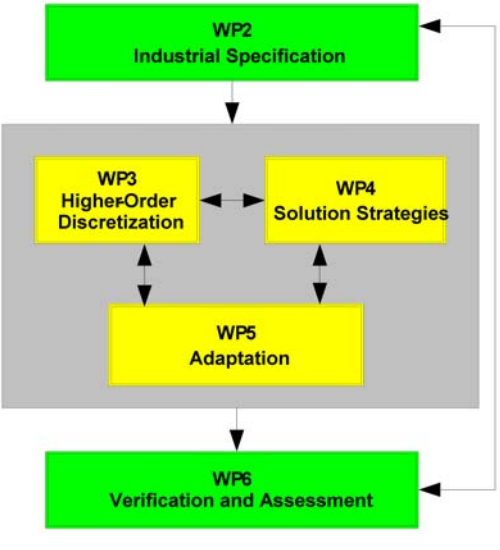


Fig. 1 Work package structure of the ADIGMA project

further development, especially with respect to efficiency and robustness for complex applications. The objective of this work package was therefore to improve key parts of higher-order spatial discretization related to stabilization and monotone shock capturing for hyperbolic conservation laws, discretization of turbulent high Reynolds number Navier-Stokes equations, ensuring higher-order accuracy in presence of complex bodies with curved boundaries and in cases of highly stretched meshes needed in high Reynolds number boundary layers.

Since computational efficiency is a crucial aspect for higher-order methods, work package WP4 was dedicated to the development of solution strategies which meet the industrial requirements in terms of memory storage, computing time and efficient utilization of parallel low cost computers. Research activities included improvements and further developments with respect to multigrid strategies (h-p multigrid) and fully implicit schemes based on Newton-type methods.

Work package WP5 addressed the effectiveness and reliability of adaptation techniques in combination with higher-order methods developed in the previous work packages. New approaches were developed in order to achieve accurate flow features and flow quantities with minimal amount of degrees of freedom and computation time. Main emphasis was on refinement indicators, error estimation and combination of h- and p-refinement. The novel adaptation algorithms should be extended to industrial relevant applications.

Finally, work package WP6 dealt with the critical assessment of the methods and technologies developed in ADIGMA under the specific aspect of a later industrial use for complex aerodynamic problems. The assessment was based on the evaluation plan and the test case suite defined in WP2. As reference results obtained with well established state-of-the-art industrial codes were provided. Identification of the best strategies and best practice guidelines should ensure technology transfer to industry.

Details of the project structure are given in Fig. 2.

6 Dissemination and Exploitation

In general, dissemination of the project achievements had been strongly encouraged both inside as well as outside the consortium. Inside the partnership an open communication strategy had been adopted to keep all partners informed. The knowledge gained in the ADIGMA project, the computational methods and the particular results generated had been disseminated in various forms. Important means are detailed technical reports, publications in journals and presentation at national and international conferences. In particular, two open ADIGMA/VKI Lecture Series courses ([7, 8]), the public closing workshop and the final report published as a dedicated book in the Springer Series “Notes on Numerical Fluid Mechanics and Multidisciplinary Design Notes” are seen as important channels to disseminate the project results.

The ADIGMA objectives enabled a strong co-operation between universities (upstream research), aircraft industry (end user) and research establishments (bridge

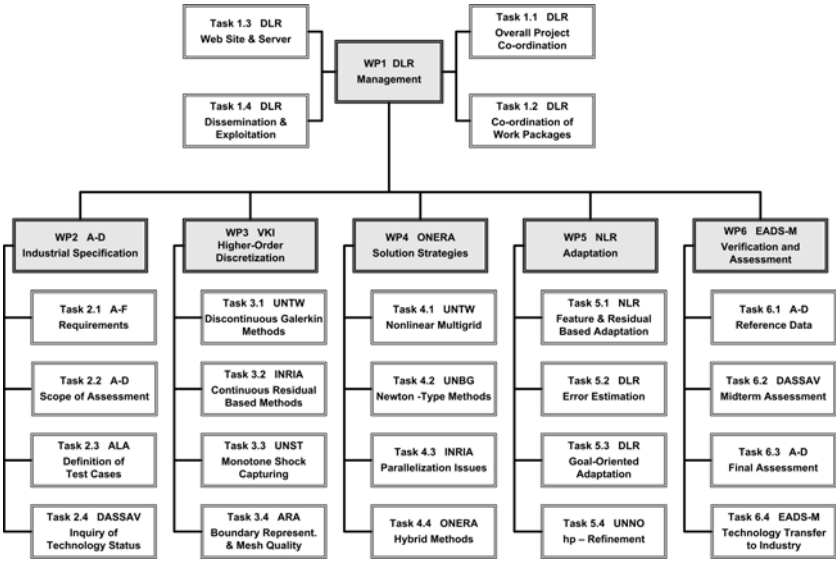


Fig. 2 Flow chart describing the structure of ADIGMA

between basic research and application). According to their role the organizations applied different dissemination and exploitation strategies.

The aircraft industry is directly involved in the transfer process from basic research and development into industrially applicable simulation methods and tools. The newly developed discretization schemes and numerical solution algorithms were explored on industrially relevant test cases. With the help of the research establishments as central providers of highly sophisticated CFD simulation tools for the aircraft industry the most promising methods will be further explored on even more complex cases from daily aerodynamic work.

The research organizations participating in ADIGMA directly exploited the knowledge gained in the project by improving their numerical tools. In particular, ADIGMA had been an important step towards the establishment of the next generation of CFD tools which can cope with the future requirements of the aeronautical industry. By providing the improved CFD methodologies to their customers and partners in industry and academia, the research organizations actively contribute to the dissemination of the ADIGMA results.

The universities taking part in ADIGMA exploited the ADIGMA findings of advanced numerical algorithms and procedures for teaching and training students and researchers. The close co-operation with industry had led to the training of qualified personnel with knowledge of the industrial requirements, thereby increasing the potential of graduates for employment within industry. The project outcome allows universities to pursue their goals in the field of applied mathematics and computational fluid dynamics, both in research and education.

In summary, on the one hand the ADIGMA project fostered the scientific cooperation between the universities, research establishments and the aeronautical industry resulting in numerous significant and novel research contributions, an improved transfer from innovative upstream CFD technologies into the industrial design cycle as well as considerable improvements and enhancements in education of highly qualified personnel. On the other hand the novel higher-order adaptive methods developed within ADIGMA should yield essential progress on several items like

- Improved simulation accuracy in reduced time and at lower cost
- Enabling automatic and reliable shape optimization and multi-disciplinary simulation and optimization through improved flow solvers
- Enabling accurate flow control simulations based on advanced physical modeling of flow control phenomena e.g. controlled flow, receptivity issues
- Mesh independent predictions of aerodynamic forces through error estimation and goal-oriented adaptation
- Automatic and reliable resolution of physical effects that have become relevant to aerodynamic design (confluent boundary layers, vortex sheets, trailing vortices, etc.)
- Support and exploitation of advanced physical models (DES, LES)
- Provision of highly accurate aerodynamic input for aero-acoustic simulations.

7 Conclusion

Within the 6th European Research Framework Programme the project ADIGMA was set up to significantly improve the capabilities of the aerodynamic simulation tools for aircraft design. The project focused on the development and utilization of innovative higher-order variational methods in combination with reliable adaptive solution strategies. The project gathered well known partners from academia and research organizations in Europe with proven expertise in this particular field of CFD. The involvement of the European aircraft industry ensured that the research work has a clear path to industrial exploitation. The project started end of 2006 and ran for 36 month wit a 4-month prolongation. The project has proven to be very successful. It has enabled the European partners to improve and enhance their knowledge on advanced adaptive higher-order methods. The results achieved demonstrate the high potential of the new methodology but they also indicate limitations and open issues still to be tackled for full industrialization of the new methods.

In summary, the ADIGMA project is seen as an important corner stone to support the competitiveness of both the European research community and European aircraft manufactures. It has to be stated that the collective outcome of the European project ADIGMA is seen to be far greater than what could have been expected by each individual partner. In the following chapters the developments and results achieved within the project are discussed in detail.

References

1. Ekaterinaris, J.A.: High-order accurate. Low diffusion methods for aerodynamics. Progress in Aerospace Sciences 41, 192–300 (2005)
2. Laflin, K.R., Vassberg, J., Wahls, R.A., Morrison, J.H., Brodersen, O., Rakowitz, M., Tinoco, E.N., Godard, J.: Summary of data from the Second AIAA CFD Drag Prediction Workshop. AIAA 2004-0555 (2004)
3. Levy, D.W., Zickuhr, T., Vassberg, J., Agrawal, S.K., Wahls, R.A., Pirzadeh, A., Hemsch, M.J.: Data summary from the First AIAA Computational Fluid Dynamics Drag Prediction Workshop. Journal of Aircraft 40, 875–882 (2003)
4. Vos, J.B., Rizzi, A., Darracq, D.: Navier-Stokes solvers in European aircraft design. Progress in Aerospace Sciences 38, 601–697 (2002)
5. Vassberg, J., Tinoco, E., Mani, M., Brodersen, O., Eisfeld, B., Wahls, R., Morrison, H., Zickuhr, T., Laflin, K., Mavripilis, D.: Abridged Summary of the Third AIAA Computational Fluid Dynamics Drag Prediction Workshop. Journal of Aircraft 45(3), 781–793 (2008)
6. Wang, Z.J.: High-order methods for the Euler and Navier-Stokes equations on unstructured grids. Progress in Aerospace Sciences 43, 1–41 (2007)
7. 35th CFD/ADIGMA VKI Course: High-Order Discretization Methods, VKI, Brussels, Belgium, October 13-17 (2008)
8. 36th CFD/ADIGMA VKI Course: hp-adaptive methods and hp-multigrid, VKI, Brussels, Belgium, October 26-30 (2009)

This page intentionally left blank

Chapter 2
Exploiting Data Locality in the DGM Discretisation for Optimal Efficiency

Koen Hillewaert

Abstract. Near-optimal CPU efficiency for the basic operations of the Newton-Krylov-BILU iterations for the discontinuous Galerkin method (DGM) can be obtained by exploiting its data locality, as it allows to rewrite these operations in terms of BLAS/LAPACK operations on contiguous vectors and matrices of considerable size. Further significant enhancements are obtained by using single precision preconditioners and by exploiting data alignment to improve cache efficiency. The underlying data structures and operations are explained, followed by the comparison of the obtained floating point operation (FLOP) efficiency to the theoretical optimum.

1 The Discontinuous Galerkin Method

We use the following notation for a system of N_v convective-diffusive equations

$$\begin{aligned} &\frac{\partial u_m}{\partial t} + \nabla \cdot \mathbf{f}_m(u) + \nabla \cdot \mathbf{d}_m(u, \nabla u) = 0 \ , \ m = 1 \ldots N_v \\ &\mathbf{d}_m^k = \sum_{l=1}^{d} \mathsf{D}_{mn}^{kl} \frac{\partial u_n}{\partial x^l} \end{aligned} \tag{1}$$

This equation is to be solved on the computational domain Ω, with appropriate boundary conditions imposed at the boundary Γ. Here u, $\mathbf{f}$ and $\mathbf{d}$ denote the solution, the convective and the diffusive flux respectively whilst the sub- and superscript indices relate to variables and coordinates respectively.

The *discontinuous Galerkin method (DGM)* approximates the solution components u_m by functions $\tilde{u}_m$ defined in the function space Φ. As illustrated in Fig. 1 Φ is composed of functions that are polynomial functions on each of the elements ω of the tesselation, but are potentially discontinuous across the element interfaces.

Koen Hillewaert
Cenaero, Bâtiment Éole, 1er étage, 29, Rue des Frères Wright, B6041 Gosselies, Belgium
e-mail: koen.hillewaert@cenaero.be

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 11–23, 2010.
springerlink.com

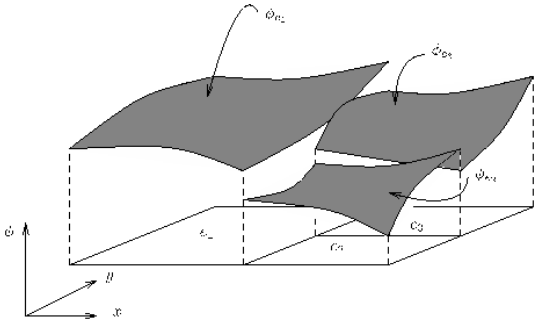


Fig. 1 Discontinuous finite element interpolation

Here the polynomials are defined elementwise in terms of *parametric* coordinates ξ defined in a reference element. Both the solution $\tilde{u}$ and the coordinates $\mathbf{x}$ are then defined as a linear combination of the respective basis functions ϕ_i and χ_j:

$$\begin{aligned} \tilde{u}_m(\xi) &= \sum_{i=1}^{N_\phi} \mathbf{u}_{im}\phi_i(\xi)\ ,\ \phi_i \in \boldsymbol{\Phi} \\ \mathbf{x}(\xi) &= \sum_{k=1}^{N_\chi} \mathbf{X}_i\chi_i(\xi)\ ,\ \chi_i \in \Xi \end{aligned} \tag{2}$$

As suggested by the notation, the *degrees-of-freedom (dof)* $\mathbf{u}_{im}$ are stored in matrix format, see [4]; the first index refers to the associated shape function, and the second to the variable. We choose both ϕ_i and χ_i to be Lagrangian interpolants based on nodes, equidistantly spaced in the reference element. A major advantage is that on any of the element faces only the functions associated to nodes on that boundary are non-zero, thus greatly reducing the work associated to boundary contributions.

The variational formulation combines a classical DGM discretisation for the convective terms with the *Symmetric Interior Penalty* method for the discretisation for the viscous terms (see Arnoldi et al. [2] for an overview of viscous discretisations). The weighted residual for variable m, associated to shape function i, is given by:

$$\begin{aligned} \mathbf{r}_{im} = &\underbrace{-\int_\omega \nabla\phi_i \cdot \mathbf{f}_m\, dV}_{CV} + \underbrace{\oint_\gamma \phi_i\, \mathscr{H}_m\left(\tilde{u}^-, \tilde{u}^+, \mathbf{n}\right) dS}_{CI} - \underbrace{\int_\omega \nabla\phi_i \cdot \mathbf{d}_m\, dV}_{DV} \\ &+ \underbrace{\oint_\gamma [[\phi_i]]^k \left\langle \mathsf{D}^{kl}_{mn} \cdot \frac{\partial \tilde{u}_n}{\partial x^l} \right\rangle dS}_{DI} + \underbrace{\oint_\gamma [[\tilde{u}_n]]^k \left\langle \mathsf{D}^{kl}_{nm} \cdot \frac{\partial \phi_i}{\partial x^l} \right\rangle}_{DT} + \underbrace{\oint_\gamma \sigma\, [[\tilde{u}_m]]\, [[\phi_i]]\, dS}_{DP} \\ &[[u]] = (u^+\mathbf{n}^+ + u^-\mathbf{n}^-)\ ,\ \langle u \rangle = (u^+ + u^-)/2 \end{aligned} \tag{3}$$

The superscripts $^+$ and $^-$ denote the outward resp. inward limit of the solution when approaching the interface γ, while $\mathbf{n}$ is the local unit normal; $\mathscr{H}$ is an (approximate) Riemann solver, defining an upwind flux across the face in function of both states on either side of the interface. The Dirichlet boundary conditions are applied to the convective interface (CI) and the penalty terms (DT, DP) by providing an appropriate external state; the Neumann boundary conditions for the diffusive terms (DI) replace the average diffusive fluxes in DI appropriately.

The resulting system of non-linear equations is solved using a combination of damped inexact Newton iterations with block ILU preconditioned, Jacobian-free GMRES iterations, see ao. Chisholm et al. [5]. Although the Jacobian matrix is not used for the Krylov iterations, the ILU preconditioner necessitates its construction and storage. We will show in the next sections how we can implement the related operations efficiently, relying on the data localisation of the DGM.

Since any of the shape functions ϕ_i is associated to one element only, we can reinterprete the discretisation as a set of high-order finite element problems defined per element ω; the element solutions are coupled across element interfaces γ by enforcing appropriate "boundary conditions", either through a Riemann solver for the convective or penalty terms for the diffusive part of the equations. It is further important to notice that within each element the organisation of its associated *dof* is the same: DGM is in fact a hybrid between a *structured* method on the element level and an *unstructured* method at the interface level. This has important implications for efficiency: all of the elemental data can be stored in dense matrix format and the bulk of the computational work can be recast as linear algebra operations on dense vectors and matrices of relatively large size.

2 BLAS and LAPACK Reference Performances

We compare the performance of different implementations (native, Atlas and MKL) of BLAS and LAPACK on an Intel Core2$^{\text{TM}}$ machine clocked at 2.5GHz, with a L1 data cache of 32KB and a unified L2 cache of 6MB. All of the operations are performed on contiguous data (ie. no strides). We want to test these operations in configurations that are close to their use within the code, in order to provide an estimate of the ideal performance. The performances are measured in GFlops, ie. in billions of *Floating point Operations per Second*. By aligning data, this CPU can perform up to 4 double precision floating operations per cycle, corresponding to a maximal speed of 10 GFlops. Due to further data alignment, single precision arithmetic can be performed twice as fast, as on most modern processors - obviously a very appealing feature, currently under investigation for accelerating double precision linear algebra [1, 3].

2.1 Scaled Vector Addition

Level 1 blas defines the scaled vector addition operation *axpy* as:

$$\begin{gathered} \mathbf{y} \leftarrow \alpha\mathbf{x} + \mathbf{y} \\ \alpha \in \mathbb{R},\ \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \end{gathered} \tag{4}$$

As only 2 floating point operations are performed per pair of entries in the vectors, the *axpy* operation is very sensitive to cache size. Figure 2 shows the performance for a realistic scenario, corresponding to the use of *axpy* during the Jacobian assembly (see section 4); in this case we systematically replace the output vector **y** by the next vector in memory whilst keeping the addendum **x** constant.

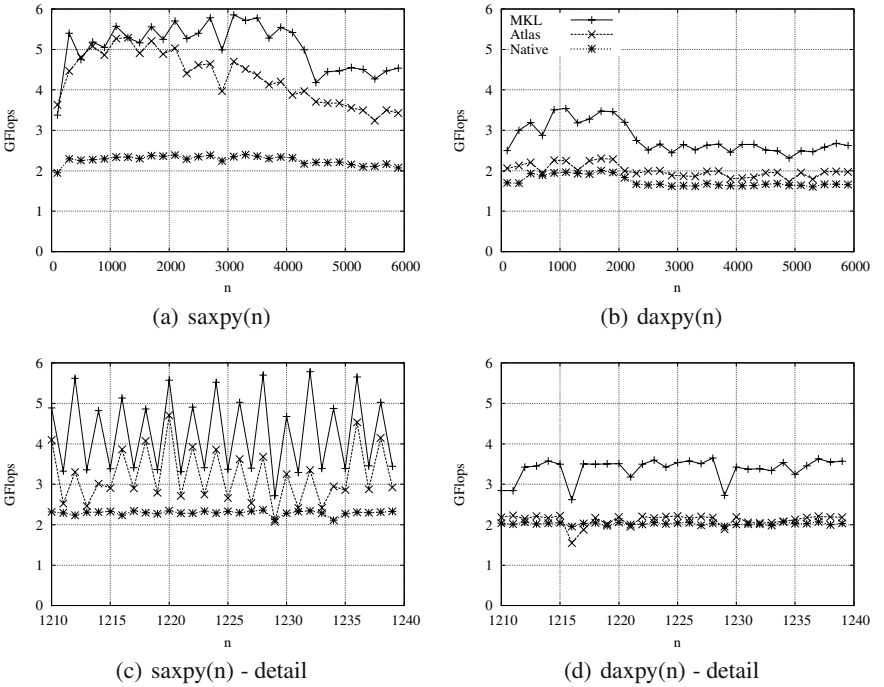


Fig. 2 CPU measurements for vector sum operation when y is systematically replaced - single (*saxpy*) versus double precision (*daxpy*) for different BLAS implementations

In figures 2(a) and 2(b) we sample the performance per increment of 200 of the vector size. However if we perform more detailed measurements we get the results shown in Figs. 2(c) and 2(d): for single precision operations, we found up to 50% difference between the performance for odd- and even-sized vectors; moreover, multiples of 4 are usually most efficient. This could be a data alignment issue, yet it is surprising that the performance is impacted so heavily, and only in single precision.

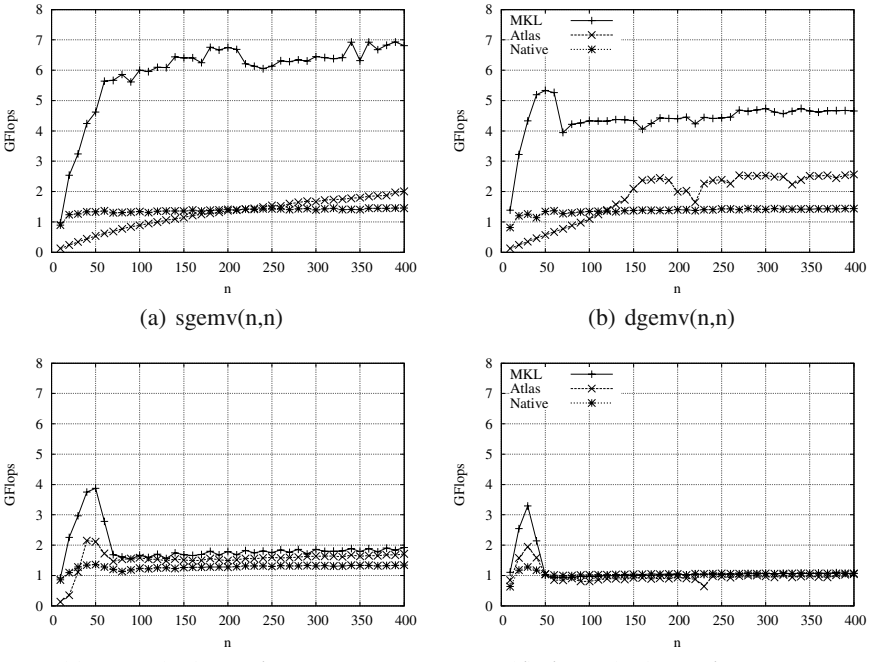
2.2 Scaled Matrix Vector Multiplication

Level 2 BLAS defines the matrix-vector product *gemv* as

$$\begin{gathered} \mathbf{y} \leftarrow \alpha\, \mathbf{A} \cdot \mathbf{x} + \beta \mathbf{y} \\ \alpha, \beta \in \mathbb{R},\ \mathbf{y} \in \mathbb{R}^m,\ \mathbf{x} \in \mathbb{R}^n,\ \mathbf{A} \in \mathbb{R}^{m \times n} \end{gathered} \tag{5}$$

The number of operations for *gemv* is given by $2mn$, which is proportional to the number of data involved; hence this operation is again very sensitive to cache miss.

We will use *gemv* in the back-and forward substitution steps (see section 3), where the memory access is almost random; the distance between successive accesses to the vector only being limited by the bandwidth of the Jacobian. In practice this means that we will systematically run into cache miss. In Fig.3 we compare the speeds for two configurations. Thereto contiguous blocks of 400 matrices and vectors are allocated; in the first configuration we access both matrices and vectors in a sequential manner. In the second realistic scenario, memory is accessed randomly within those two blocks. We see a very dramatic loss of performance for the



(a) sgemv(n,n) (b) dgemv(n,n)

(c) sgemv(n,n) - random access (d) dgemv(n,n) - random access

Fig. 3 CPU measurements for the scaled matrix-vector multiplication and addition - single (*sgemv*) vs double (*dgemv*) precision (left-right) and organised versus random acces (top-bottom)

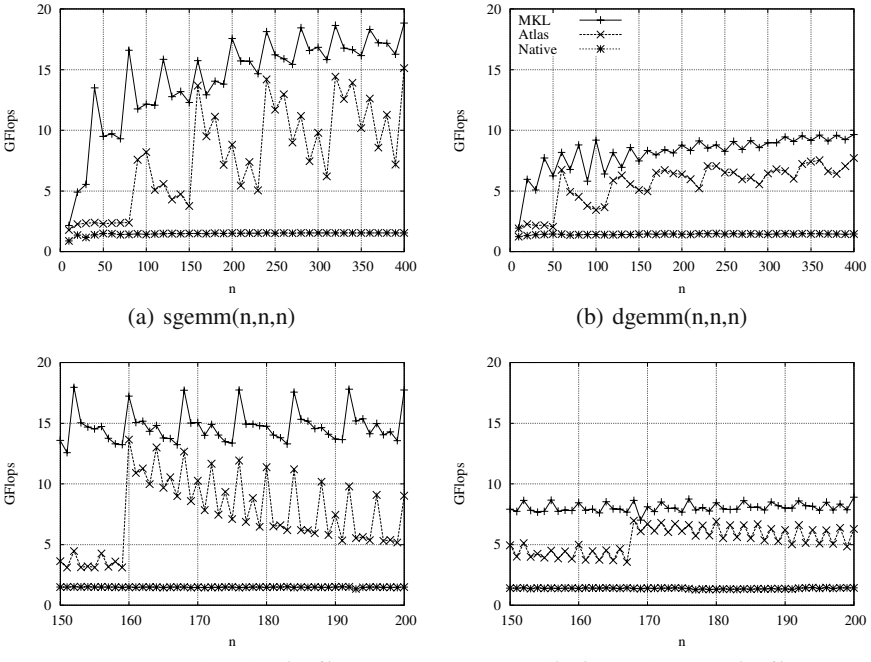
second scenario from the point where the level 2 cache (6 MB) is no longer capable of containing all matrices and vectors.

2.3 Scaled Matrix-Matrix Multiplication and Addition

Level 3 BLAS defines the matrix-matrix multiplication *gemm* as follows:

$$\begin{gathered} \mathbf{A} \leftarrow \alpha\, \mathbf{B} \cdot \mathbf{C} + \beta\, \mathbf{A} \\ \alpha, \beta \in \mathbb{R},\ \mathbf{A} \in \mathbb{R}^{m \times o},\ \mathbf{B} \in \mathbb{R}^{m \times n},\ \mathbf{C} \in \mathbb{R}^{n \times o} \end{gathered} \tag{6}$$

The number of floating point operations is $2mno + 2mo$. The number of operations is high with respect to data transfer, and as a consequence the operation is not very sensitive to cache. Typical timings for $(m = n = o)$ are shown in figure 4, this time only for a random access to memory (cfr.2.2). The performance is dependent on matrix size, this time both in single and double precision, with a particular good efficiency for multiples of 4 in double, and multiples of 8 in single precision.



(a) sgemm(n,n,n) (b) dgemm(n,n,n)

(c) sgemm(n,n,n) - detail (d) dgemm(n,n,n) - detail

Fig. 4 CPU measurements for the scaled matrix-matrix multiplication and addition - single (sgemm) vs double precision (dgemm); bottom graphs are zooms

2.4 Matrix Inversion

Measurements for the matrix inversion show similar trends and dependency on vector lengths as for matrix-matrix multiplications, although the maximum speed is about twice as low, probably due to pivoting.

3 Jacobian Matrix

The Jacobian matrix $\mathbf{L} = \frac{\partial \mathbf{r}}{\partial \mathbf{u}}$ is stored in blocked sparse format. Each diagonal block entry $\mathbf{L}_{aa}$ corresponds to the coupling between all variables within the element a while the off-diagonal entries $\mathbf{L}_{ab}$ and $\mathbf{L}_{ba}$ correspond to the direct neighbour coupling induced by the interface between elements a and b. Each block is a fully dense matrix of size $n = N_\phi N_v$, stored contiguously in memory. We further partition $\mathbf{L}_{ab}$ in a quadrilateral raster of submatrices $\mathbf{L}_{ab}^{kl}$ corresponding to the coupling between variables k and l. Within each subblock, the variables are further arranged according to shape function indices i and j. We will denote scalar entries in the block $\mathbf{L}_{ab}$, with indices k, l refering to variable and i, j to shape function combinations, as

$$(\mathbf{L}_{ab})_{ij}^{kl} = \frac{\partial \mathbf{r}_{im}^{a}}{\partial \mathbf{u}_{jn}^{b}} \tag{7}$$

3.1 Matrix Operations

All matrix operations use dense block operations from BLAS and LAPACK. During (incomplete) block LU decomposition, the row reduction operation is rewritten as a combination of dense matrix inversions and matrix-matrix products (*gemm*):

$$\mathbf{L}_{bc} := \mathbf{L}_{bc} - \mathbf{L}_{ba} \cdot \mathbf{L}_{aa}^{-1} \cdot \mathbf{L}_{ac} \ , \ \forall c > a \tag{8}$$

Matrix-vector operations such as backward substitution are then recast as dense matrix-vector products (*gemv*):

$$\mathbf{a}_c := \mathbf{a}_c - \sum_{a>c} \mathbf{L}_{ab} \cdot \mathbf{a}_b \tag{9}$$

Given the large dimensions of the blocks, any cost related to block indexing will be small with respect to the effective operations, and as a consequence we can use a very flexible datastructure to store the matrix structure. It consists of maps for each row / column a, providing links between the off-diagonal column resp. row index b and the corresponding dense block $\mathbf{L}_{ab}$ resp. $\mathbf{L}_{ba}$. This structure allows for dynamic block allocation, dynamic decomposition strategies such as ILUt, dynamic renumbering strategies and even variable block size without noticeable overhead.

3.2 Storage Requirements and Precision

In 3D all elements except those on the boundary are connected to 4 other elements. Hence the storage requirements amount to slightly less than $5n^2$ floating point values per element, with $n = N_\phi N_v$. Considering that N_ϕ scales as the cube of the interpolation order, it is not surprising that the memory required for the Jacobian is much larger than that for all other data; eg. for 4th order interpolation we need 1.2MB per element in double precision, as compared to the solution vector which only requires 1.4 kB. The memory footprint is quite terrifying, so the prospect of halving it by storing $\mathbf{L}$ in single precision is quite appealing; however, the important question is whether this will not deteriorate or even impede convergence ?

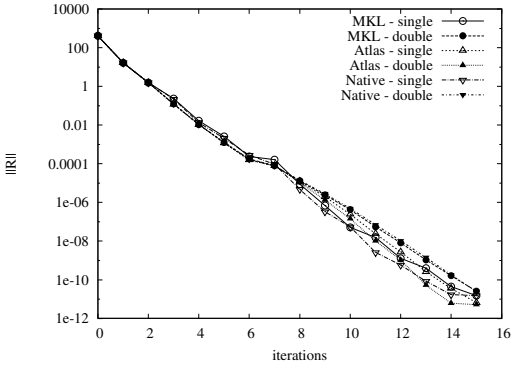


Fig. 5 Effect of preconditioner precision on Newton-GMRES-ILU(1) convergence

Figure 5 shows the impact of the precision of the ILU(1) preconditioner on the convergence of the Newton-GMRES iterations when using the different linear algebra librairies. We see that in this particular case, but also in general, that the precision has only as much impact as the choice of linear algebra library; the differences between libraries are due to operation order and a different pivoting strategy during the inversion of the diagonal blocks. Although we cannot prove that single precision is sufficient, nor provide conditions for convergence, we can put the following elements forward:

- the GMRES iterations are used for inexact Newton iterations, and hence a full convergence is not required - typically only two orders of magnitude will do. This related to the *iterative refinement* procedure (see ao. Golub and Van Loan [6], section 3.5.3 and Baboulin et al. [3]), a method that provides double precision solutions to a linear system of equations, using Newton steps with single precision linear updates.
- since we use a *Jacobian-free* GMRES, the matrix-vector product needed for the Krylov iterations is formed by finite differences and the Jacobian matrix is only used for preconditioning. Hence the product retains full precision. Arioli and

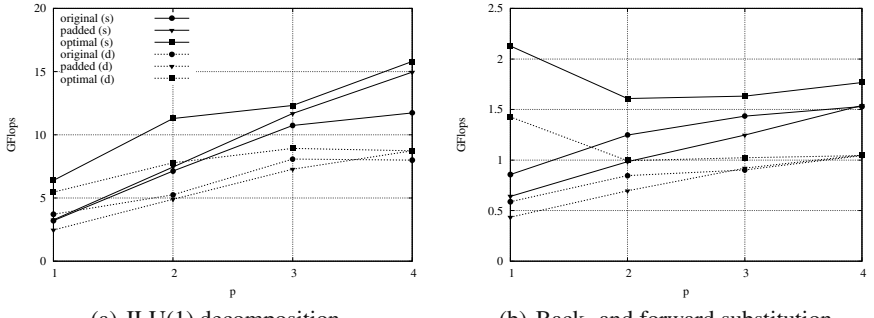
Duff [1] prove convergence of FGMRES iterations to double precision accuracy, preconditioned using a single-precision LU factorisation of the matrix, as long as the condition number of the matrix does not exceed the inverse of the relative single precision round-off error; numerical experiments seem to indicate that the algorithm works reasonably well even beyond this point.

3.3 Efficiency

The storage of the matrix in single precision is also good for performance: referring to section 2 we know that the basic operations such as pivot inversion, row reduction and back and forward substitution are about twice as fast in single precision as in double precision. This is not specific to the processor used for this study, but holds for most modern processors. We can further speed up the computation by *padding* the blocks, i.e. by artificially increasing the block sizes to the nearest multiple of 8 in single, and 4 in double precision.

The effects of precision and block padding on the floating performance as a function of the interpolation order p are illustrated in Fig.6(a). The reference optimal performance has been computed from the measured flop rates for the *gemm* and inversion operations, taking into account the number of times they were called and the number of flops involved in each, not counting the extra work due to padding. The performance is close to the optimal performance of the underlying BLAS/Lapack operations; the performance of the decomposition is close to the peak flop rate of the processor, as it is mainly based on *gemm*.

In Fig. 6, the speed of the substitution steps is compared to predicted optimal performance. The flop rates are low due to the random access to a large number of blocks, resulting in a memory typically outside of the L2 cache.



(a) ILU(1) decomposition (b) Back- and forward substitution

Fig. 6 Solver flop rate (MKL)

4 Optimising Assembly

The convective (CV) and diffusive (DV) volume terms are assembled as

$$\begin{aligned} CV_{im} &= \int_\omega \nabla\phi_i \cdot \mathbf{f}_m \, dV = \sum_{q=1}^{N_\mu} w_q \left(|\mathbf{J}| \sum_{k=1}^{d}\sum_{u=1}^{d} \frac{\partial \phi_i}{\partial \xi^u}\frac{\partial \xi^u}{\partial x^k}\mathbf{f}_m^k \right)_{\mu_q} \\ DV_{im} &= \int_\omega \nabla\phi_i \cdot \mathbf{d}_n \, dV = \sum_{q=1}^{N_\mu} w_q \left(|\mathbf{J}| \sum_{k=1}^{d}\sum_{u=1}^{d} \frac{\partial \phi_i}{\partial \xi^u}\frac{\partial \xi^u}{\partial x^k}\mathbf{d}_m^k \right)_{\mu_q} \end{aligned} \tag{10}$$

where μ_q defines the location of the q-th integration point, w_q the corresponding weight and $|\mathbf{J}|_{\mu_q}$ the determinant of the mapping Jacobian. We require an integration to be accurate up to order $2p+1$ in the element, and $2p$ on the element boundaries. The integration rules and taken from Segeth et al. [7].

The associated linearisations read

$$\begin{aligned} \frac{\partial CV_{in}}{\partial \mathbf{u}_{jm}} &= \sum_{q=1}^{N_\mu}\sum_{u=1}^{d} \underbrace{w_q \left(\frac{\partial \phi_i}{\partial \xi^u}\phi_j \right)_{\mu_q}}_{\mathfrak{C}^u_{q,ij}} \cdot \underbrace{\left(|\mathbf{J}| \sum_{k=1}^{d} \frac{\partial \xi^u}{\partial x^k}\frac{\partial \mathbf{f}_n^k}{\partial u_m} \right)_{\mu_q}}_{\kappa^u_{q,mn}} \\ \frac{\partial DV_{im}}{\partial \mathbf{u}_{jn}} &= \sum_{q=1}^{N_\mu}\sum_{u=1}^{d}\sum_{v=1}^{d} \underbrace{w_q \left(\frac{\partial \phi_i}{\partial \xi^u}\frac{\partial \phi_j}{\partial \xi^v} \right)_{\mu_q}}_{\mathfrak{D}^{uv}_{q,ij}} \cdot \underbrace{\left(|\mathbf{J}| \sum_{k=1}^{d}\sum_{l=1}^{d} \frac{\partial \xi^u}{\partial x^k}\mathsf{D}^{kl}_{mn}\frac{\partial \xi^v}{\partial x^l} \right)_{\mu_q}}_{\delta^{uv}_{q,mn}} \end{aligned} \tag{11}$$

Equation 11 implies that for each quadrature point q and variable combination (m,n), we add *precomputed parametric* convection $\mathfrak{C}^u_q$ and stiffness $\mathfrak{D}^{uv}_q$ contributions to the Jacobian matrix subblocks L^{aa}_{mn} with respective weights $\kappa^k_{q,mn}$ and $\delta^{uv}_{q,mn}$. This procedure is illustrated in Fig.7. We can get a close estimate of the computational cost by only counting the number of operations involved in the addition of the influence matrices. Taking into account the sparsities of the three convective flux

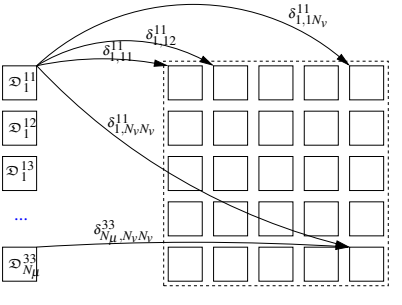


Fig. 7 Naive linearisation of the volume terms CV and DV. $\frac{\partial DV}{\partial \mathbf{u}}$ is taken as example.

Jacobians and of the diffusive flux Jacobians D_{mn}^{kl}, denoted f_c and f_d respectively, we have for the convective and diffusive term a total of $2\cdot(f_c+d\cdot f_d)\cdot d\cdot N_\mu\cdot {N_v}^2\cdot N_\phi^2$ operations. For an interpolation order of 4, using a $2p+1$-accurate quadrature rule, this amounts to 38.10^6 floating point operations per element.

Adding quadrature point contributions directly to the subblocks of the Jacobian matrix leads to N_ϕ vector additions (*axpy*) of very limited dimension (N_ϕ), since the none of the blocks $\mathbf{L}_{aa}^{mn}$ are stored contiguously in memory and have to be added row by row. Consequently the assembly is not very efficient.

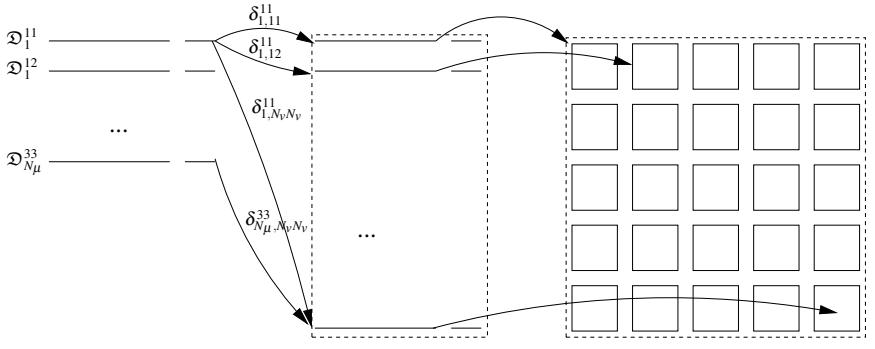


Fig. 8 Linearisation of volume terms using padded intermediate vectors. $\frac{\partial DV}{\partial \mathbf{u}}$ is taken as example.

If we preassemble all quadrature point contributions in an intermediate matrix (see Fig. 8) that vertically aligns all N_v^2 subblocks unrolled as vectors of size N_ϕ^2, we can add $\mathfrak{C}_q^u$ and $\mathfrak{D}_q^{uv}$ in a single step. This results in a much more efficient assembly, as *axpy* is applied to vectors of size N_ϕ^2. Only at the very end the assembled subblocks are added to the subblocks of the final Jacobian matrix. This final addition needs to be done only once for every $N_q\cdot(f_c+f_d d)\cdot d$ quadrature steps. We can further take advantage of the large differences in speed between odd and even sized vectors, by padding the vectors to the next multiple of 4.

Figure 9 compares the different versions to the underlying *axpy* operation; for the padded operations only the effective work is counted. We see that the assembly in single precision is up to 30% faster than the same operation in double precision, except for the original implementation. Padding is necessary to maintain the advantage of single precision operations with respect to double for $p=4$.

Similar optimisations have been applied to both interface and boundary terms. Figure 10 shows the evolution of the computational time for the different terms. The optimised version is globally 3 times than the original, and the increase in computational complexity as a function of interpolation order is compensated to a large extent by the increase in computational efficiency, especially for the lower orders.

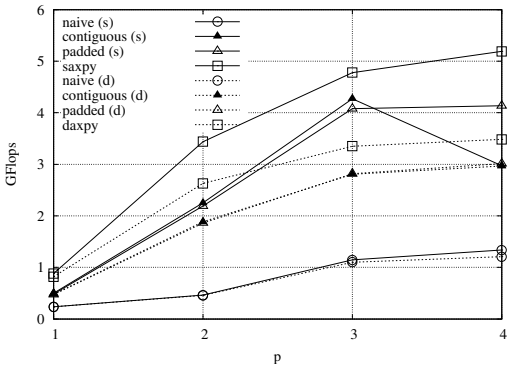
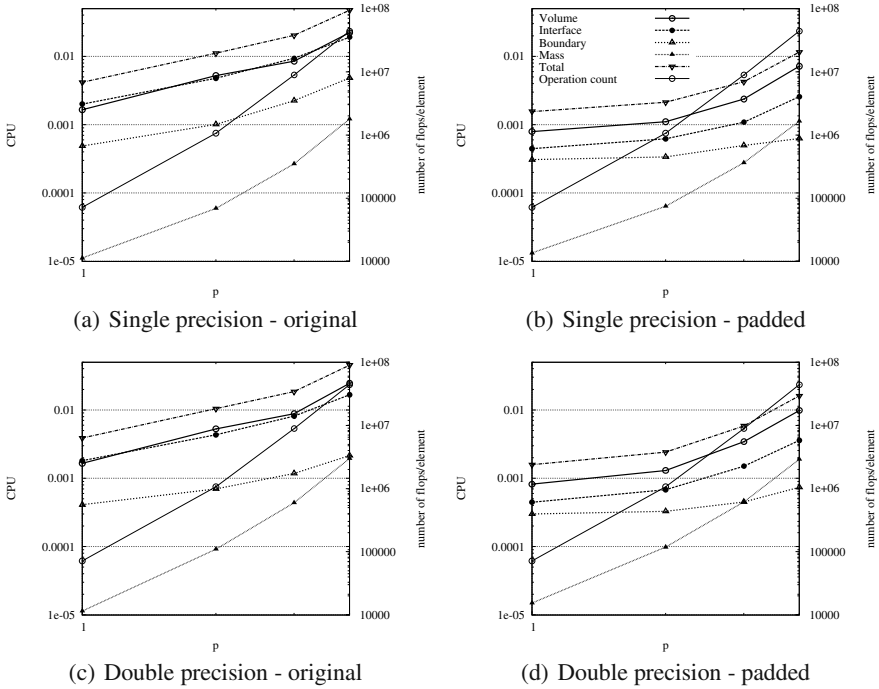


Fig. 9 Volume term assembly flop rate using MKL comparing naive, contiguous and padded versions with respect to the speed of vector additions (*axpy*) in single (s) and double (d) precision



(a) Single precision - original

(b) Single precision - padded

(c) Double precision - original

(d) Double precision - padded

Fig. 10 Assembly time per element as a function of interpolation order. Computational complexity is illustrated in terms of flop count.

5 Conclusions

The Newton-Krylov-ILU strategy can be implemented very efficiently for DGM, reaching very high and sometimes near-peak flop rates, especially matrix decomposition but also for matrix assembly. Near-optimal performance, with low overhead with respect to the relevant basic operations, has been obtained. This allows one to take advantage of the efficient implementation of BLAS and LAPACK implementation and the speed of single precision arithmetic.

The use of a Jacobian-free GMRES with single precision preconditioner for DGM thus leads to an economical method, both in terms of memory and CPU time, in comparison to a more classical implementation.

The significant efficiency that has been obtained for the assembly of the matrix and its decomposition, particularly with respect to the hard-to-optimise residual assembly, shift the balance somewhat towards matrix-based methods. Without pronouncing ourselves as yet on the outcome, these results in any case indicate the need for an optimised implementation in the assessment of iterative techniques for DGM.

Acknowledgements. Part of this work was supported by the Walloon Region and the European funds ERDF and ESF under contract N° EP1A122030000102.

References

1. Arioli, M., Duff, I.: Using FGMRES to obtain backward stability in mixed precision. Electronic Transactions on Numerical Analysis 33, 31–44 (2009)
2. Arnold, D., Brezzi, F., Cockburn, B., et al.: Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems. SIAM J. Num. Anal. 39, 1749–1779 (2002)
3. Baboulin, M., Buttari, A., Dongarra, J., et al.: Accelerating Scientific Computations with Mixed Precision Algorithms. Computer Physics Communications (2008) (in press)
4. Chevaugeon, N., Hillewaert, K., Gallez, X., et al.: Optimal numerical parameterization of discontinuous Galerkin method applied to wave propagation problems. Journal of Computational Physics 223, 188–207 (2007)
5. Chisholm, T., Zingg, D.: A Jacobian-free Newton-Krylov algorithm for compressible turbulent fluid flows. Journal of Computational Physics 228, 3490–3507 (2009)
6. Golub, G., Van Loan, C.: Matrix Computations. The Johns Hopkins University Press, Baltimore (1996)
7. Solin, P., Segeth, K., Dolezel, I.: Higher Order Finite Element Methods. Studies in Advanced Mathematics. Chapman and Hall/CRC (2004)

This page intentionally left blank

Chapter 3
Very High-Order Accurate Discontinuous Galerkin Computation of Transonic Turbulent Flows on Aeronautical Configurations

F. Bassi, L. Botti, A. Colombo, A. Crivellini, N. Franchina,
A. Ghidoni, and S. Rebay

Abstract. This chapter presents high-order DG solutions of the RANS and k-ω turbulence model equations for transonic flows around aeronautical configurations. A directional shock-capturing term, proportional to the inviscid residual, is employed to control oscillations around shocks. Implicit time integration is applied to the fully coupled RANS and k-ω equations. Several high-order DG results of 2D and 3D transonic turbulent test cases proposed within the ADIGMA project demonstrate the capability of the method.

1 Introduction

The combination of high-order discretization, equations stiffness and flow discontinuities makes the DG solution of the RANS and k-ω turbulence model equations for high Reynolds number transonic flows very challenging. The discretization should preserve high-order accuracy on highly stretched and curved grids, the shock-capturing technique should be able to provide sub-cell resolution of discontinuities while having minimal impact away from shocks, time integration should be robust and efficient. In this chapter we summarize the progress on such topics achieved within the ADIGMA project and demonstrate the capability of the DG code MIGALE on several challenging test cases.

F. Bassi · L. Botti · A. Colombo · A. Crivellini · N. Franchina
Facoltà di Ingegneria, Università degli studi di Bergamo, Viale Marconi 5,
24044 Dalmine (BG), Italy

A. Ghidoni · S. Rebay
Dipartimento di Ingegneria Meccanica e Industriale, Università degli Studi di Brescia,
Via Branze 38, 25123 Brescia, Italy
e-mail: francesco.bassi@unibg.it, lorenzo.botti@unibg.it,
alessandro.colombo@unibg.it, a.crivellini@univpm.it,
nicoletta.franchina@unibg.it, antonio.ghidoni@ing.unibs.it,
stefano.rebay@ing.unibs.it

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 25–38, 2010.
springerlink.com

2 DG Solution of the RANS and k-ω Equations

Turbulent flow solutions presented in this chapter have been computed by using the linearly implicit Euler method to solve the system of ODEs in time deriving from the DG space discretization of the RANS and k-ω turbulence model equations. The governing equations and the realizability constraints used in the implementation of the k-ω turbulence model have already been reported in [4] and here we only present an improved solid-wall boundary condition for ω that has been implemented in the course of the ADIGMA project.

2.1 Solid Wall Boundary Condition for ω

A popular approach to prescribe the wall boundary condition ω_w is that proposed by Menter [10], whereby the prescribed wall value ω_w is related to the first cell height y_1 according to the relation

$$\omega_w = \frac{6\nu}{\beta(\alpha_M y_1)^2}, \tag{1}$$

with $\alpha_M = 1/\sqrt{10}$. Instead, according to the results of DG computations of flat plate flows on differently refined grids reported in [4], a good agreement between experimental and numerical skin friction distributions of flat plate flows was obtained using $\alpha = 0.5366\times10^{-1}$. However, solutions presented in [4] were only computed up to $\mathbb{P}^2$ polynomial approximation. As higher degree polynomials can follow closer and closer the exact near wall distribution of ω, it seems reasonable, as also suggested by Hartmann [9], to make ω values set at solid walls dependent on the degree of polynomial approximation. For this purpose we define

$$\int_{y_1} \phi\,\widetilde{\omega}\,\mathrm{d}y = \int_{y_1} \phi\,\widetilde{\omega}_{ex}\,\mathrm{d}y, \tag{2}$$

where $\widetilde{\omega}_{ex}$ is the near-wall analytical behavior of $\widetilde{\omega}$, *i.e.*,

$$\widetilde{\omega}_{ex} = \log\left(\frac{6\nu}{\beta}\right) - 2\log y, \tag{3}$$

and ϕ is the one-dimensional polynomial basis adopted to define $\widetilde{\omega}$. From Eq. (2) we can then compute $\widetilde{\omega}_w^k = \widetilde{\omega}(0)$ for any desired polynomial degree k.

These values can be compared to that of Menter in terms of the first cell height fraction, α_k, by setting

$$\widetilde{\omega}_w^k = \log\frac{6\nu}{\beta(\alpha_k y_1)^2}, \tag{4}$$

and thus obtaining

$$\alpha_k = \frac{1}{y_1}\sqrt{\frac{6\nu}{\beta}\frac{1}{e^{\widetilde{\omega}_w^k}}}. \tag{5}$$

k	α_k
0	0.5
1	0.1304
2	0.6150×10^{-1}
3	0.3604×10^{-1}
4	0.2375×10^{-1}
5	0.1685×10^{-1}
6	0.1259×10^{-1}
7	0.9765×10^{-2}
8	0.7797×10^{-2}
9	0.6370×10^{-2}
10	0.5303×10^{-2}

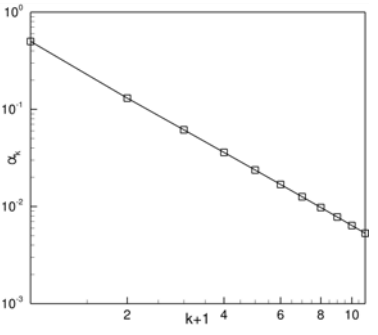


Fig. 1 Cell height fraction, α_k, vs. polynomial degree k

Figure 1 shows the values of the cell height fraction, α_k for $k = 0,\ldots,10$. In order to be roughly consistent with the value of α used in the computations reported in [4], for a $\mathbb{P}^k$ computation we have employed the α corresponding to $k+1$. The boundary condition for ω has been tested on the flat plate flow reported by Wieghardt [13]. The computational grid, taken from the NPARC Alliance Validation Archive, [12], is the coarsest one used for the validation of the WIND code and corresponds to $y^+ = 30$ for the first grid point off the wall. The Figure 2 displays the skin-friction distribution along the plate and the profiles of u velocity component and of turbulence quantities at $x/L = 0.923$ resulting from the $\mathbb{P}^4$ and $\mathbb{P}^5$ solutions. The difference of near wall behavior of k^+ between DG results and "average" experimental data is an effect produced by the high-Reynolds number k-ω model here employed, that disappears using the modified coefficients of the low-Reynolds number version of the model.

2.2 DG Space Discretization

The governing equations can be written in compact form as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla\cdot\mathbf{F}_c(\mathbf{u}) + \nabla\cdot\mathbf{F}_v(\mathbf{u},\nabla\mathbf{u}) + \mathbf{s}(\mathbf{u},\nabla\mathbf{u}) = \mathbf{0}, \tag{6}$$

where $\mathbf{u},\mathbf{s}\in\mathbb{R}^M$ denote the vectors of the M conservative variables and source terms, $\mathbf{F}_c,\mathbf{F}_v\in\mathbb{R}^M\otimes\mathbb{R}^N$ denote the inviscid and viscous flux functions, respectively, and N is the space dimension.

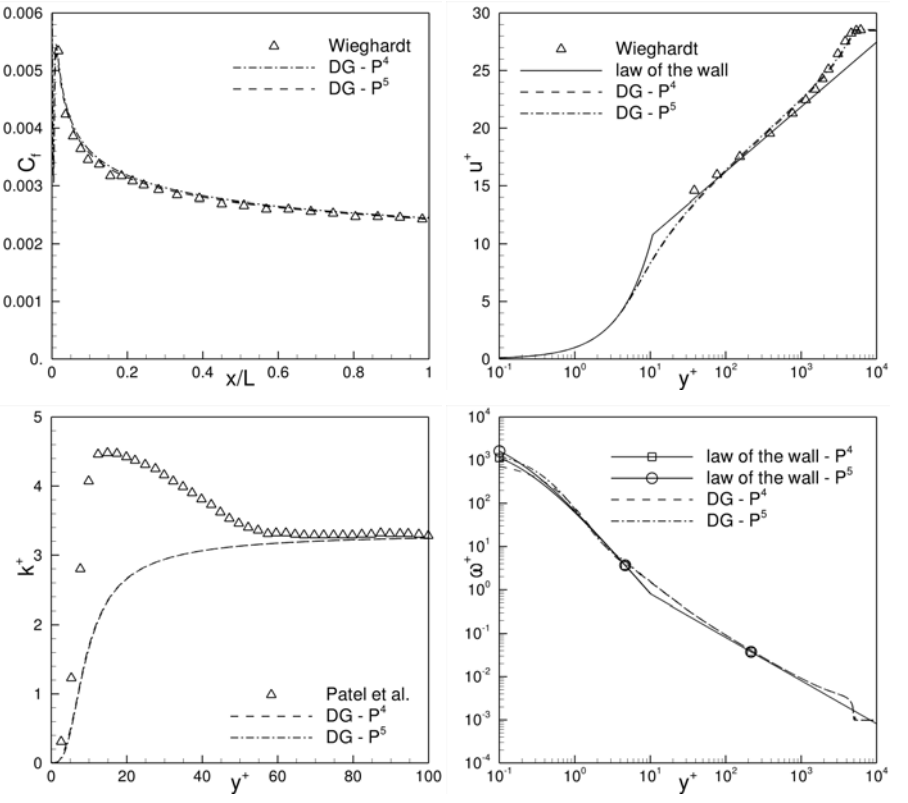


Fig. 2 Flat plate: skin-friction, velocity profiles and turbulence quantities

The weak form of Eq. (6) reads

$$\int_\Omega \phi \frac{\partial \mathbf{u}}{\partial t}\,\mathrm{d}\mathbf{x} - \int_\Omega \nabla\phi \cdot \mathbf{F}(\mathbf{u},\nabla\mathbf{u})\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} \phi \mathbf{F}(\mathbf{u},\nabla\mathbf{u})\cdot\mathbf{n}\,\mathrm{d}\sigma + \int_\Omega \phi \mathbf{s}(\mathbf{u},\nabla\mathbf{u})\,\mathrm{d}\mathbf{x} = \mathbf{0}, \quad (7)$$

where ϕ denotes any arbitrary, sufficiently smooth, test function and $\mathbf{F}$ is the sum of the inviscid and viscous fluxes. The DG discretization of Eq. (7) is defined on a triangulation $\mathscr{T}_h = \{K\}$ of an approximation Ω_h of Ω, consisting of a set of non-overlapping hybrid-type elements. The following space setting of discontinuous piecewise polynomial functions for each component $u_{h_i} = u_{h_1}, \ldots, u_{h_M}$ of the numerical solution $\mathbf{u}_h$ is assumed:

$$u_{h_i} \in \Phi_h \stackrel{\text{def}}{=} \left\{ \phi_h \in L^2(\Omega) : \phi_h|_K \in \mathrm{P}^k(K)\ \forall K \in \mathscr{T}_h \right\} \quad (8)$$

for some polynomial degree $k \geq 0$, being $\mathrm{P}^k(K)$ the space of polynomials of global degree at most k on the element K.

The discontinuous approximation of the numerical solution requires introducing a special treatment of the inviscid interface flux and of the viscous flux. For the former it is common practice to use suitably defined numerical flux functions which ensure conservation and account for wave propagation. For the latter we employ the BR2 scheme, presented in [6, 5] and theoretically analyzed in [7, 2] (where it is referred to as BRMPS), to obtain a consistent, stable and accurate discretization of the viscous flux. Accounting for these aspects, the DG formulation of problem (7) then requires to find $u_{h_1}, \ldots, u_{h_M} \in \Phi_h$ such that

$$\begin{aligned}\int_{\Omega_h} \phi_h \frac{\partial \mathbf{u}_h}{\partial t}\, \mathrm{d}\mathbf{x} - \int_{\Omega_h} \nabla_h \phi_h \cdot \mathbf{F}\left(\mathbf{u}_h, \nabla_h \mathbf{u}_h + \mathbf{r}(\llbracket \mathbf{u}_h \rrbracket)\right)\, \mathrm{d}\mathbf{x} \\ + \int_{\Gamma_h} \llbracket \phi_h \rrbracket \cdot \widehat{\mathbf{f}}\left(\mathbf{u}_h^{\pm}, (\nabla_h \mathbf{u}_h + \eta_e \mathbf{r}_e(\llbracket \mathbf{u}_h \rrbracket))^{\pm}\right)\, \mathrm{d}\sigma \\ + \int_{\Omega_h} \phi_h \mathbf{s}\left(\mathbf{u}_h, \nabla_h \mathbf{u}_h + \mathbf{r}(\llbracket \mathbf{u}_h \rrbracket)\right)\, \mathrm{d}\mathbf{x} = \mathbf{0}, \quad (9)\end{aligned}$$

for all $\phi_h \in \Phi_h$. In Eq. (9) we have introduced the jump $\llbracket \cdot \rrbracket$ and average $\{\cdot\}$ trace operators as well as the lifting operators $\mathbf{r}$ and $\mathbf{r}_e$, as defined in [6, 5]. The inviscid and viscous parts of the numerical flux $\widehat{\mathbf{f}}$ are treated independently. For the former we usually employ the Godunov flux or, alternatively, the van Leer-Hänel flux-splitting scheme, [8]. The numerical viscous flux is given by

$$\widehat{\mathbf{f}}_v\left(\mathbf{u}_h^{\pm}, (\nabla_h \mathbf{u}_h + \eta_e \mathbf{r}_e(\llbracket \mathbf{u}_h \rrbracket))^{\pm}\right) \stackrel{\text{def}}{=} \{\mathbf{F}_v\left(\mathbf{u}_h, \nabla_h \mathbf{u}_h + \eta_e \mathbf{r}_e(\llbracket \mathbf{u}_h \rrbracket)\right)\}, \quad (10)$$

where, according to [7, 2], the penalty factor η_e must be greater than the number of faces of the elements. The BR2 viscous flux discretization is as compact as possible because, for each element K, it only involves the nearest neighbor elements. This feature is obviously very attractive for the implicit implementation of the method.

2.3 Time Integration

The DG space discretized Eq. (9) represents a system of (nonlinear) ODEs in time for the global vector of degrees of freedom of the solution. By applying the linearly implicit Euler method to this system of ODEs, the linear system to be solved at each time step until convergence to steady state can be written as

$$\left[\frac{\mathbf{M}}{\Delta t} + \frac{\partial \mathbf{R}(\mathbf{U}^n)}{\partial \mathbf{U}}\right]\left(\mathbf{U}^{n+1} - \mathbf{U}^n\right) + \mathbf{R}(\mathbf{U}^n) = \mathbf{0}, \quad (11)$$

where $\mathbf{M}$ is the global block diagonal mass matrix and $\mathbf{J} = \partial \mathbf{R}(\mathbf{U}^n)/\partial \mathbf{U}$ is the Jacobian matrix of the residual $\mathbf{R}(\mathbf{U})$ resulting from the DG discretized space operators of Eq. (9). The matrix-based or the matrix-free GMRES algorithm can be used to actually solve Eq. (11), see [3] for a comparative discussion. In both cases

system preconditioning is required to make the convergence of the GMRES solver acceptable in problems of practical interest. The ILU(0) factorization of the Jacobian matrix $\mathbf{J}$ turns out to be one of the more effective preconditioning approaches for the implicit solver here employed. The Jacobian matrix implemented in our code has been derived analytically and takes full account of the dependence of the residual on the unknown vector and on its derivatives, including the implicit treatment of the lifting operators and of the boundary conditions. If coupled with a suitably accurate time integration scheme, this allows to employ the implicit solver also for accurate unsteady computations.

The choice of the time step can significantly affect both the efficiency and the robustness of the method. For steady computations we have implemented the pseudo-transient continuation strategy with the local time step given by

$$\Delta t_K = CFL \frac{h_K}{c+d},$$

where

$$c = |\mathbf{v}| + a, \qquad d = 2\frac{\mu_e + \lambda_e}{h_K}, \qquad h_K = N\frac{\Omega_K}{S_K},$$

define convective and diffusive velocities and the reference dimension of the generic element K, respectively. The coefficients μ_e and λ_e are the effective dynamic viscosity and conductivity, while Ω_K and S_K denote the volume and the surface of K. All quantities depending on $\mathbf{u}_h$ in the above relations are computed from mean values of $\mathbf{u}_h$. Devising an effective and robust strategy to increase the CFL number as the residual decreases is not an easy task, especially for turbulent computations. The rule here proposed is essentially the result of intensive numerical experimentation and aims at controlling the evolution of CFL number on the basis of both the L_∞ and the L_2 norms of the residual. Denoting with y the CFL number, the rule is as follows

$$\begin{cases} y = \frac{y_0}{x^\alpha} & \text{if } x \le 1 \\ y = y_e + (y_0 - y_e) e^{\alpha \frac{y_0}{y_0 - y_e}(1-x)} & \text{if } x > 1 \end{cases} \tag{12}$$

where, denoting by $x_{L_2} = \max\left(|R_i|_{L_2}/|R_{i0}|_{L_2}\right)$ and $x_{L_\infty} = \max\left(|R_i|_{L_\infty}/|R_{i0}|_{L_\infty}\right)$ for $i = 1, \ldots, M$,

$$\begin{cases} x = \min\left(x_{L_2}, 1\right) & \text{if } x_{L_\infty} \le 1 \\ x = x_{L_\infty} & \text{if } x_{L_\infty} > 1, \end{cases}$$

and $y_0 = \mathrm{CFL}_{min}$, $y_e = \mathrm{CFL}_{exp}$ and α are the user-defined minimum CFL number, the maximum CFL number of explicit schemes and the exponent (usually ≤ 1) governing the growth rate of the CFL number, respectively. The strong CFL number control based on the L_∞ norm of residual has been found useful to prevent sudden breakdown of computations once the CFL number has already reached quite high values. For relatively simple test cases, such as the BTC0 problem shown in

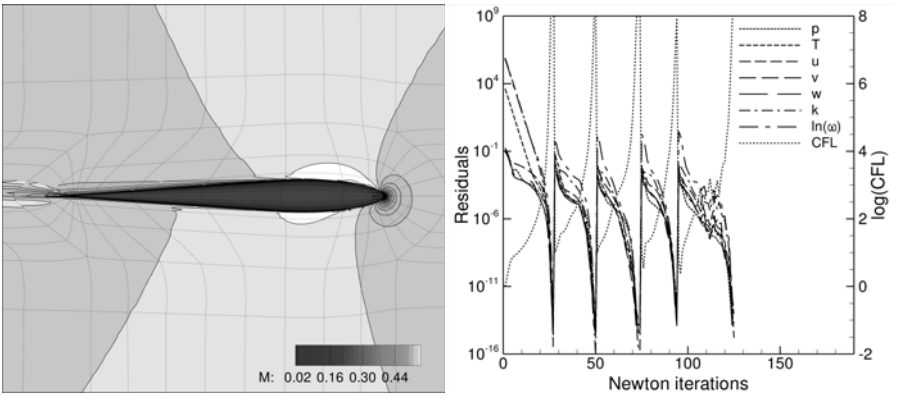


Fig. 3 BTC0: Mach number contours of $\mathbb{P}^4$ solution and residuals convergence history $\mathbb{P}^{0\to4}$

Figure 3 (232969 DOFs), the implicit time integration combined with the above CFL number rule provides quadratic Newton convergence to machine accuracy.

2.4 Shock-Capturing Approach

The shock-capturing approach consists of adding to the DG discretized equations an artificial viscosity term that aims at controlling the high-order modes of the numerical solution within elements while preserving as much as possible the spatial resolution of discontinuities. The shock-capturing term is local and active in every element, but the amount of artificial viscosity is proportional to the (inviscid) residual of the DG space discretization and thus it is almost negligible except than at locations of flow discontinuities. The shock-capturing term added to Eq. (9) reads

$$\sum_K \int_K \varepsilon_p(\mathbf{u}_h^{\pm}, \mathbf{u}_h)\,(\nabla_h \phi_h \cdot \mathbf{b})\,(\nabla_h \mathbf{u}_h \cdot \mathbf{b})\;\mathrm{d}\mathbf{x}, \tag{13}$$

with the shock sensor and the pressure gradient unit vector defined by

$$\varepsilon_p(\mathbf{u}_h^{\pm}, \mathbf{u}_h) = Ch_K^2 \frac{|s_p(\mathbf{u}_h^{\pm}, \mathbf{u}_h)| + |d_p(\mathbf{u}_h)|}{p(\mathbf{u}_h)} f_p(\mathbf{u}_h), \qquad \mathbf{b}(\mathbf{u}_h) = \frac{\nabla_h p(\mathbf{u}_h)}{|\nabla_h p(\mathbf{u}_h)| + \varepsilon}, \tag{14}$$

and

$$s_p(\mathbf{u}_h^{\pm}, \mathbf{u}_h) = \sum_{i=1}^{M} \frac{\partial p(\mathbf{u}_h)}{\partial u_{h_i}} s_i(\mathbf{u}_h^{\pm}), \qquad d_p(\mathbf{u}_h) = \sum_{i=1}^{M} \frac{\partial p(\mathbf{u}_h)}{\partial u_{h_i}} (\nabla_h \cdot \mathbf{F}_c(\mathbf{u}_h))_i\,. \tag{15}$$

The components s_i of the function $\mathbf{s}$, defined by the solution of the problem

$$\int_{\Omega_h} \phi_h \mathbf{s}(\mathbf{u}_h^{\pm})\, \mathrm{d}\mathbf{x} = \int_{\Gamma_h} [\![\phi_h]\!] \cdot \left(\widehat{\mathbf{f}}_c(\mathbf{u}_h^{\pm}) - \mathbf{F}_c(\mathbf{u}_h)\right)^{\pm} \mathrm{d}\sigma, \tag{16}$$

are actually the lifting of the interface jump in normal direction between the numerical and internal inviscid flux components. The further factor $f_p(\mathbf{u}_h)$ in Eq. (14) is a pressure sensor defined by

$$f_p(\mathbf{u}_h) = \frac{|\nabla_h p(\mathbf{u}_h)|}{p(\mathbf{u}_h)} \left(\frac{h_K}{k}\right), \tag{17}$$

which improves the accuracy of solutions in regions with high but otherwise smooth gradients and allows using the same value of the user-defined parameter C (typically $C = 0.2$) for different degrees of polynomial approximation. Finally, the element dimension h_K is defined as

$$h_K = \frac{1}{\sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}}}, \tag{18}$$

where $\Delta x, \Delta y$ and Δz are the dimensions of the hexahedral enclosing K, scaled in such a way that their product matches the volume of K. The shock-capturing technique outlined above is highly non-linear and residuals convergence of steady state solutions can be quite difficult, even implementing a fully (linearized) implicit discretization of the shock-capturing term (13). This is in fact the case for the solution of the transonic flow around the RAE 2822 airfoil ($\mathrm{M}_\infty = 0.730$, $\alpha = 3.19°$, $\mathrm{Re}_\infty = 6.5 \times 10^6$, 80860 DOFs), shown in Figure 4, that requires quite a large number of Newton iterations for convergence.

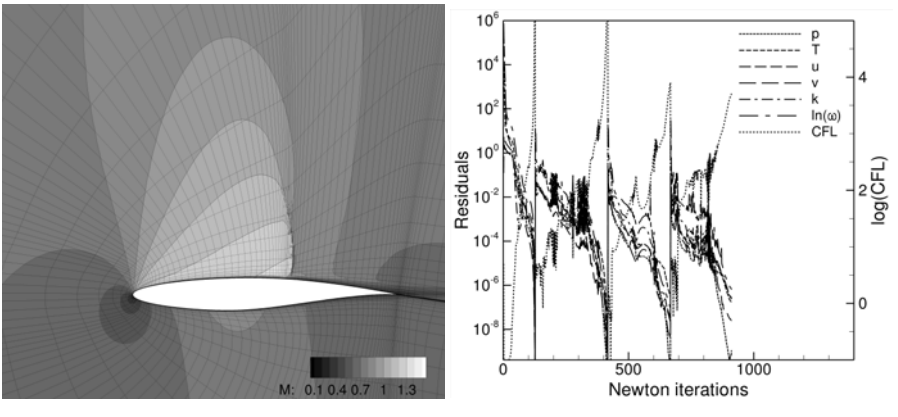


Fig. 4 RAE2822: Mach number contours of $\mathbb{P}^3$ solution and residuals convergence history $\mathbb{P}^{0\to 3}$

3 Numerical Results

In this section we present the results of high-order DG solutions around two wing models, the ONERA M6 and the DPW-W1, and around the DLR-F6 wing-body transport configuration. All the computations have been performed in parallel, starting higher-order solutions from the lower-order ones. Solutions have been advanced in time using the linearly implicit backward Euler method and the linear system (11) has been solved using the default solver available in PETSc, *i.e.*, the restarted GMRES algorithm preconditioned with the block Jacobi method with one block per process, each of which is solved with ILU(0). The computational grids here employed consist of hexahedrons with curved, eight-node faces.

The flow around the ONERA M6 wing has been computed up to $\mathbb{P}^2$ approximation for the conditions of Test 2308, [11], *i.e.*, $M_\infty = 0.8395$, $\alpha = 3.06°$, chord-based $Re_\infty = 11.72 \times 10^6$, on a grid with 215632 hexahedral elements. Figure 5 and Table 1 summarize the computational results of this test case. In particular, the shock-capturing technique proves capable of providing accurate resolution of the lambda shock structure all along the suction surface of the wing. The flow around

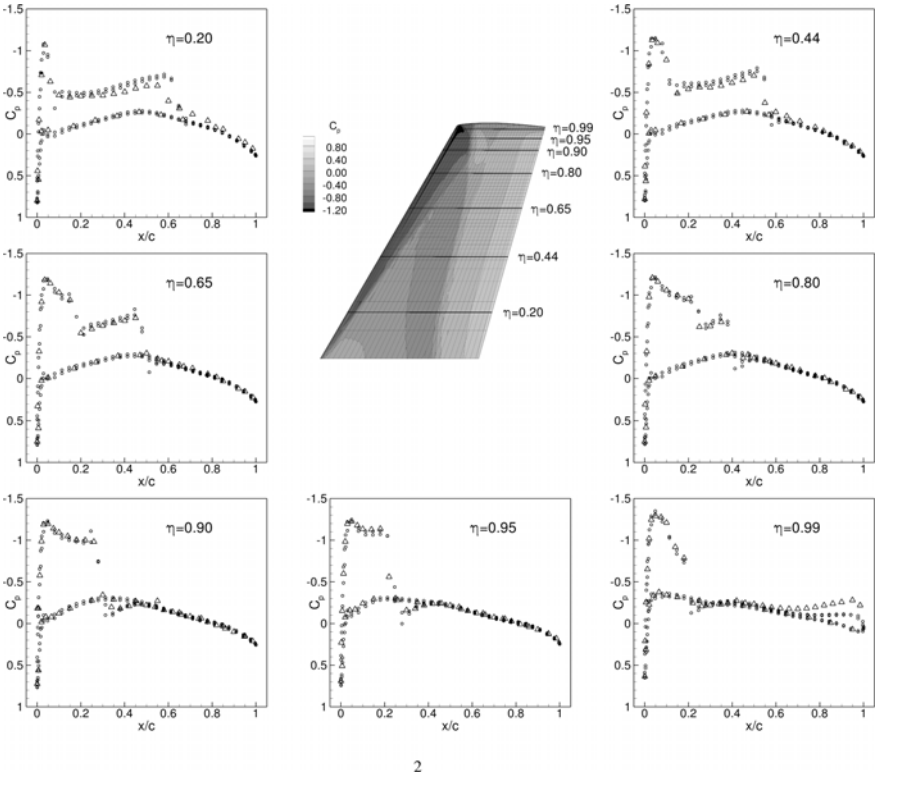


Fig. 5 ONERA M6: pressure coefficient of $\mathbb{P}^2$ solution (○ 2156320 DOFs) compared with the experiments (△)

Table 1 ONERA M6: lift and drag coefficients of DG solutions

	C_l	C_d	C_{d_p}	C_{d_f}
$\mathbb{P}^0$	0.231900	0.0555007	0.0502764	0.00522416
$\mathbb{P}^1$	0.274433	0.0184980	0.0133475	0.00515066
$\mathbb{P}^2$	0.275279	0.0180224	0.0123096	0.00571281

the DPW-W1 wing, [1], has been computed up to $\mathbb{P}^2$ polynomial approximation for the conditions $M_\infty = 0.76$, $\alpha = 0.5°$, chord-based $Re_\infty = 1\times10^7$ on a grid with 188928 hexahedral elements. In Figure 6 the pressure coefficient distribution of the $\mathbb{P}^2$ solution is compared with solutions of the TAU and FUN3D codes, [1], computed on finer and adapted grids for $Re_\infty = 5\times10^6$. Despite the relatively coarse grid employed, the shock resolution of the high-order DG solution appears also in this case remarkably good.

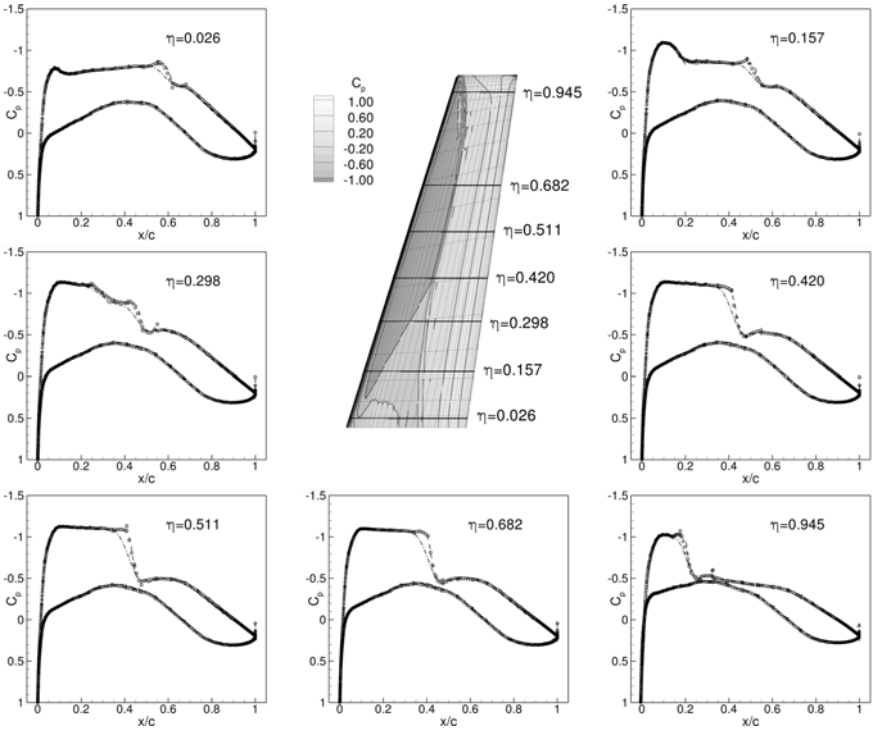


Fig. 6 DPW-W1: pressure coefficient of $\mathbb{P}^2$ solution (○ 1889280 DOFs) compared with TAU (– – – 17053510 DOFs) and FUN3D (– · – 11459041 DOFs)

The flow around the DLR-F6 wing-body transport configuration, [1], has been computed for the conditions $M_\infty = 0.75$, $C_l = 0.5$, chord-based $Re_\infty = 5 \times 10^6$. The DG solutions have been computed up to $\mathbb{P}^3$ and up to $\mathbb{P}^2$ polynomial approximation on two nested grids with 50618 and 404944 hexahedral elements, respectively. The coarse and fine grid computations have been run in parallel using, respectively, 128 and 512 cores of the CASE cluster facility at DLR in Braunschweig. Pressure contours and residuals convergence for the corse grid solution are shown in Figure 7, while Table 2 reports the force and pitching moment coefficients on the two grids. The discrepancy between the more accurate results on the two grids is still to be understood and no conclusion can be drawn about the asymptotic convergence of the two solutions. One issue could be the poor geometrical approximation of solid surfaces when using only quadratic mappings for the faces of very coarse meshes. Figure 8 shows how the pressure coefficient distribution of the $\mathbb{P}^3$ solution

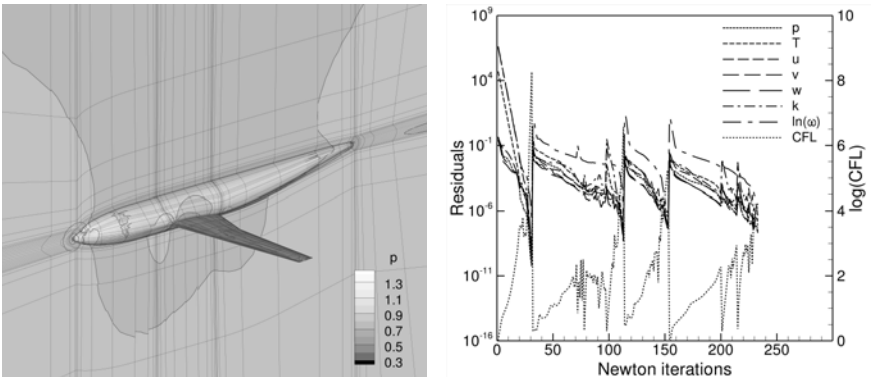


Fig. 7 DLR-F6: pressure contours of $\mathbb{P}^3$ solution and residuals convergence history $\mathbb{P}^{0\to3}$ on the coarse grid

Table 2 DLR-F6: force and pitching moment coefficients of DG solutions

(a) coarse grid

	$\mathbb{P}^0$	$\mathbb{P}^1$	$\mathbb{P}^2$	$\mathbb{P}^3$
DOFs	50618	202472	506180	1012360
α	2.34000	0.22500	-0.07000	-0.07000
C_l	0.49973	0.49996	0.49998	0.49986
C_d	0.16745	0.04232	0.02822	0.02832
C_{d_p}	0.15201	0.02905	0.01672	0.01531
C_{d_f}	0.01544	0.01327	0.01151	0.01301
C_m	0.03812	-0.12468	-0.14526	-0.14642

(b) fine grid

	$\mathbb{P}^0$	$\mathbb{P}^1$	$\mathbb{P}^2$
DOFs	404944	1619776	4049440
α	1.34600	0.10600	0.35700
C_l	0.50002	0.50005	0.49994
C_d	0.11738	0.03045	0.02890
C_{d_p}	0.10306	0.01874	0.01727
C_{d_f}	0.01432	0.01171	0.01163
C_m	-0.03781	-0.13714	-0.12528

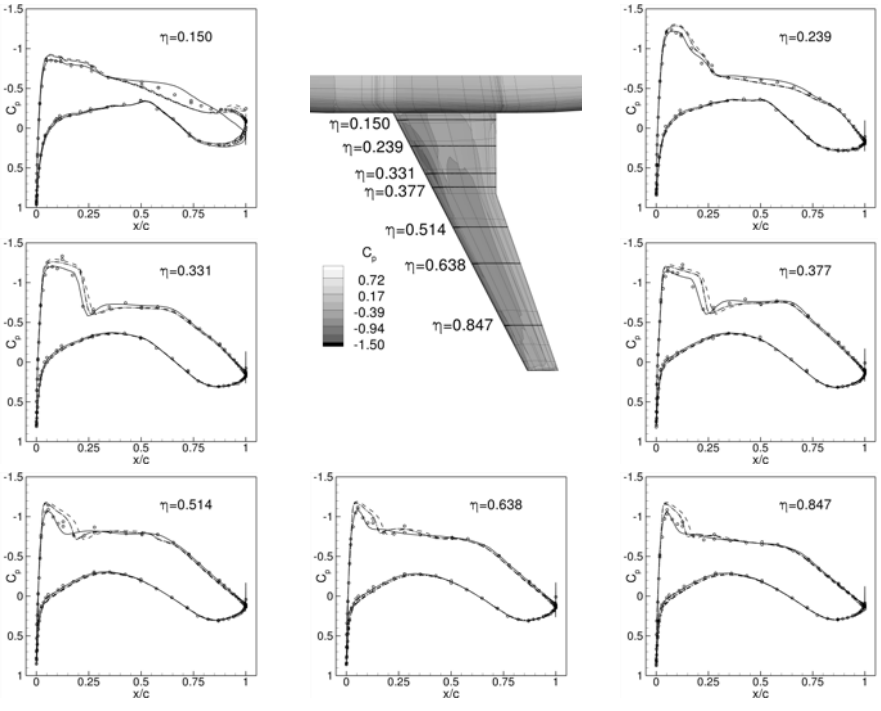


Fig. 8 DLR-F6: pressure coefficient of $\mathbb{P}^3$ solution (○ 1012360 DOFs) compared with TAU (—— 5102446 DOFs) and CFL3D (– – – 2256896 DOFs, – · – 7689088 DOFs, – ·· – 26224640 DOFs)

on the coarse grid compares with reference results of the TAU and CFL3D codes taken from [1]. Finally, we remark the capability of higher order methods to capture complex fluid features on coarse meshes by showing in Figure 9 the detail of flow separation near the tip of the ONERA-M6 wing and at the wing-root juncture of the DLR-F6 wing-body configuration.

3.1 Conclusion

In this chapter we have demonstrated the capability of the DG code MIGALE to provide high-order solutions of complex transonic turbulent flows. Progresses on time integration and on the implicit implementation of shock capturing achieved within the ADIGMA project have been crucial for being able to obtain convergence of residuals for shocked flows. Subsonic turbulent flow solutions can now be computed quite efficiently but a fully satisfactory compromise between efficiency and robustness is still lacking. The shock-capturing approach turned out to be robust and accurate. However, the highly non-linear character of the shock-capturing term has

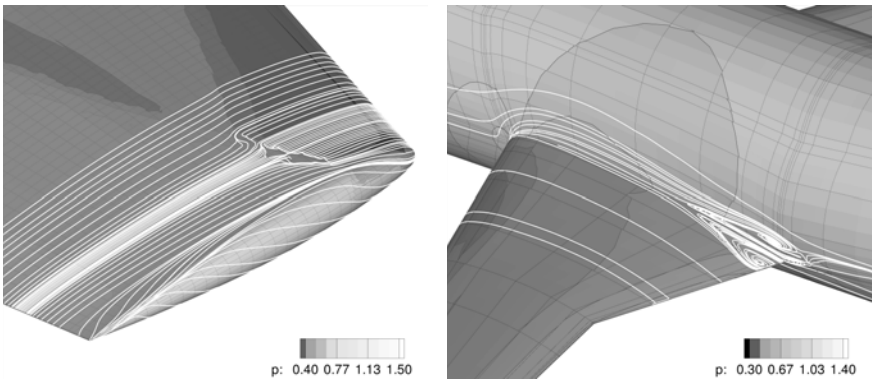


Fig. 9 Flow separation near the tip of the ONERA M6 wing and at the wing-root juncture of the DLR-F6 wing-body configuration

an adverse impact on the regularity of convergence of residuals and hence on the computational efficiency.

References

1. Third AIAA Computational Fluid Dynamics Drag Prediction Workshop (June 2006), http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/Workshop3/
2. Arnold, D.N., Brezzi, F., Cockburn, B., Marini, D.: Unified analysis of discontinuous Galerkin methods for elliptic problems. SIAM J. Numer. Anal. 39(5), 1749–1779 (2002)
3. Bassi, F., Botti, L., Crivellini, A., Ghidoni, A., Rebay, S.: D4.2.2–Investigation of Jacobian and Jacobian-free Newton-Krylov methods for implicit DG methods. Technical report, ADIGMA (2009), http://www.dlr.de/as/adigma
4. Bassi, F., Crivellini, A., Rebay, S., Savini, M.: Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and k-ω turbulence model equations. Comput. & Fluids 34, 507–540 (2005)
5. Bassi, F., Rebay, S.: A high order discontinuous Galerkin method for compressible turbulent flows. In: Cockburn, B., Karniadakis, G.E., Shu, C.-W. (eds.) Discontinuous Galerkin Methods. Theory, Computation and Applications, Lecture Notes in Computational Science and Engineering, vol. 11, pp. 77–88. Springer, Heidelberg (2000)
6. Bassi, F., Rebay, S., Mariotti, G., Pedinotti, S., Savini, M.: A high-order accurate discontinuous finite element method for inviscid and viscous turbomachinery flows. In: Decuypere, R., Dibelius, G. (eds.) Proceedings of the 2nd European Conference on Turbomachinery Fluid Dynamics and Thermodynamics, Antwerpen, Belgium, March 5-7, pp. 99–108. Technologisch Instituut (1997)
7. Brezzi, F., Manzini, M., Marini, D., Pietra, P., Russo, A.: Discontinuous Galerkin approximations for elliptic problems. Numer. Methods Partial Differential Equations 16, 365–378 (2000)
8. Hänel, D., Schwane, R., Seider, G.: On the accuracy of upwind schemes for the solution of the Navier–Stokes equations. AIAA Paper 87-1105 CP, AIAA (1987)

9. Hartmann, R.: Private discussion on solid wall boundary conditions for the k-ω turbulence model in the framework of Discontinuous Galerkin methods. DLR, Braunschweig (March 2009)
10. Menter, F.R.: Two-equation eddy-viscosity turbulence models for engineering applications. AIAA Journal 32(8), 1598–1605 (1994)
11. Schmitt, V., Charpin, F.: Pressure distributions on the ONERA-M6-wing at transonic Mach numbers. Advisory Report 138, AGARD (1979)
12. Slater, J.W.: NPARC Alliance CFD Verification and Validation Web Site (2003), `http://www.grc.nasa.gov/WWW/wind/valid/archive`
13. Wieghardt, K., Tillman, W.: On the turbulent friction layer for rising pressure. Technical Memorandum 1314, NACA (1951)

Chapter 4
Incorporating a Discontinuous Galerkin Method into the Existing Vertex-Centered Edge-Based Finite Volume Solver Edge

Sven-Erik Ekström and Martin Berggren

Abstract. The discontinuous Galerkin (DG) method can be viewed as a generalization to higher orders of the finite volume method. At lowest order, the standard DG method reduces to the cell-centered finite volume method. We introduce for the Euler equations an alternative DG formulation that reduces to the *vertex-centered* version of the finite volume method at lowest order. The method has been successfully implemented for the Euler equations in two space dimensions, allowing a local polynomial order up to $p = 3$ and supporting curved elements at the airfoil boundary. The implementation has been done as an extension within the existing edge-based vertex-centered finite-volume code Edge.

1 Introduction

The finite volume (FV) method is presently the most widely used approach to discretize the Euler and Navier–Stokes equations of aerodynamics. A basic choice when implementing a finite volume method is whether to employ a *cell-centered* or *vertex-centered* approach. The control volumes coincide with the mesh cells in the cell-centered approach (left in Figure 1), whereas in the vertex-centered approach, the control volumes are constructed from a dual mesh, consisting in two dimensions of polygons surrounding each vertex in the original primal mesh (right in Figure 1).

The vertex-centered approach has a number of features that helps to explain its current popularity. A vertex-centered scheme has about half the memory footprint on the same mesh as a cell-centered scheme, and has more fluxes per unknown,

Sven-Erik Ekström
Department of Information Technology, Uppsala University,
Box 337, SE-751 05 Uppsala, Sweden
e-mail: sven-erik.ekstrom@it.uu.se

Martin Berggren
Department of Computing Science, Umeå University, SE-901 87 Umeå, Sweden
e-mail: martin.berggren@cs.umu.se

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 39–52, 2010.
springerlink.com

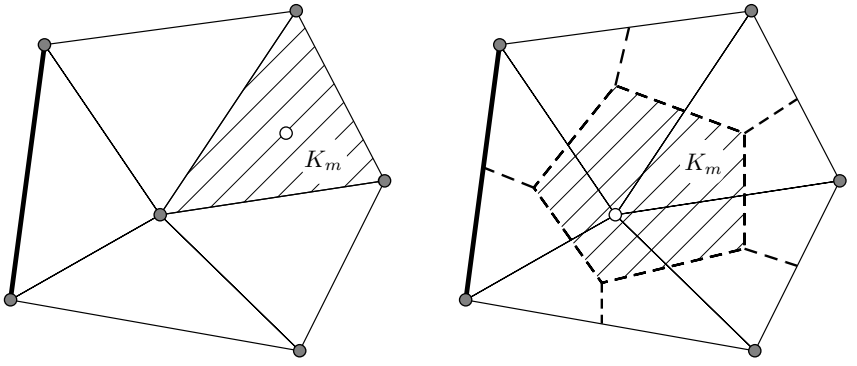


Fig. 1 Control volume choices for finite volume methods. A computational node (white circle) is either associated with a mesh cell center (left) or a mesh vertex (right), giving a cell-centered or a vertex-centered control volume respectively.

as discussed by Blazek [7]. Also, for instance Abgrall [1] argues that reconstruction schemes are more easily formulated for vertex-centered control volumes. Moreover, as opposed to cell-centered schemes, the treatment of boundary conditions are facilitated by the fact that control volume centers are located precisely on the boundary. The main computational effort in a typical finite volume code concerns the residual computations. A solver using a vertex-centered scheme may be implemented to support what Haselbacher *et al.* [16] call *grid transparency*: to assemble the residual, the solver loops over all edges in the mesh, regardless of the space dimension or the choice of mesh cell type (triangles, quadrilaterals, tetrahedrons, prisms, hexahedrons). Note, however, that an analogous construction is also possible for cell-centered methods, by looping over a list of cell surfaces. For a detailed discussion on cell-centered versus vertex-centered methods, we refer to Blazek's book [7] and the review by Morton and Sonar [20]. Also, there are other numerical schemes, besides finite volumes, that use vertex-centered control volumes or loop over edges. Some examples are residual distribution schemes [2, 10], edge-based finite elements [19], and edge-based SUPG [8].

Since the discontinuous Galerkin (DG) method constitutes a higher-order generalization of the finite volume method, it is tempting to reuse and extend existing finite volume codes to higher orders to avoid a costly rewrite of a complex and familiar software system. A hurdle for such an approach is that the standard discontinuous Galerkin method is a higher-order version of the cell-centered finite-volume method whereas many of today's codes, such as DLR-Tau [22], Edge [12, 14], Eugenie [13], Fun3D [15], and Premo [23], are vertex-centered.

The contribution of Uppsala University to the ADIGMA project consists of a case study in which a vertex-centered version of the discontinuous Galerkin method is implemented within the Edge system [12, 14] so the user, depending on the need for accuracy, can choose whether to use a classical finite volume scheme or a discontinuous Galerkin discretization. The vertex-centered discontinuous Galerkin

method we use was first introduced by Berggren [5] in the context of a linear first-order hyperbolic model problem. A further analysis of the method was presented by Berggren, Ekström, and Nordström [6] for higher-order elements and for a nonlinear problem with a shock. Here, we present for the first time the use of the scheme for the Euler equations. We first introduce the basic scheme of the method and then further discuss the method as developed within the Edge system.

2 The Discontinuous Galerkin Method

The target application for the proposed scheme is aeronautical CFD with the Euler and Navier–Stokes equations, where computations of steady states are particularly prominent. We here present the scheme for the steady Euler equations, written in the compact form

$$\nabla \cdot \mathscr{F}(U) = 0 \quad \text{in } \Omega, \tag{1}$$

where $U = [\rho, \rho\mathbf{u}, \rho E]^T$ is the solution vector of conservative variables (mass density ρ, momentum density $\rho\mathbf{u}$, and total energy density ρE), $\mathscr{F}$ is the Euler flux function, and Ω the computational domain. We also need to impose appropriate boundary conditions in the far field and at solid walls. The flux function for the Euler equations is

$$\mathscr{F}(U) = \begin{pmatrix} \rho\mathbf{u} \\ \rho\mathbf{u}\otimes\mathbf{u} + \mathbb{I}p \\ \rho\mathbf{u}H \end{pmatrix}, \tag{2}$$

where $\rho H = (\rho E + p)$ is the total enthalpy density, and the pressure p satisfies the equation of state

$$p = (\gamma - 1)\left[\rho E - \frac{|\rho\mathbf{u}|^2}{2\rho}\right], \tag{3}$$

in which $\gamma = 1.4$ for air.

We divide the domain, Ω, into a set of non-overlapping control volumes K_m such that $\overline{\Omega} = \bigcup_{m=1}^{M} \overline{K}_m$. Denote by $\mathscr{V}_h^{d+2}$, where d is the space dimension, the space of numerical solutions to equation (1). Each component of a vector function in $\mathscr{V}_h^{d+2}$ belongs to the space $\mathscr{V}_h$, which comprises functions that are continuous inside each K_m but in general discontinuous across boundaries between control volumes. The restriction on each K_m of functions in $\mathscr{V}_h$ are polynomials for the standard DG method. Here we consider a different choice of functions, as described in Secion 3. Each $V_h \in \mathscr{V}_h^{d+2}$ can be expanded as

$$V_h(\mathbf{x}) = \sum_{m=1}^{M} \sum_{i=1}^{N_m} V_i^m \phi_i^m(\mathbf{x}) = \sum_{i=1}^{N_{\text{dof}}} V_i \phi_i(\mathbf{x}), \tag{4}$$

where M is the number of control volumes, N_m is the number of local degrees of freedom in control volume K_m, $\{\phi_i^m\}_{i=1}^{N_m}$ is the set of basis functions associated with control volume K_m, N_{dof} is the total number of degrees of freedom, and ϕ_i are the basis functions in a global numbering.

The DG method is obtained by multiplying equation (1) by a test function $V_h \in \mathcal{V}_h^{d+2}$, integrating over each control volume, integrating by parts, and introducing the numerical flux $\mathscr{F}^*$ on the boundaries. This procedure yields that $U_h \in \mathcal{V}_h^{d+2}$ solves the variational problem

$$\int_{\partial K_m} V_h \cdot \mathscr{F}^*(U_L, U_R, \hat{\mathbf{n}})\,\mathrm{d}s - \int_{K_m} \nabla V_h \cdot \mathscr{F}(U_h)\,\mathrm{d}V = 0, \quad \forall K_m \subset \Omega, \quad \forall V_h \in \mathcal{V}_h^{d+2}, \tag{5}$$

where subscripts L and R denote local ("left") and remote ("right") values on the boundary ∂K_m of control volume K_m, and $\hat{\mathbf{n}}$ is the outward unit normal. The remote values are either taken from the neighboring control volume's boundary or, when ∂K_m intersects the domain boundary, from the supplied boundary condition data. Thus, the boundary conditions are imposed weakly through the numerical flux.

In our implementation, we use the Roe numerical flux by default,

$$\mathscr{F}^*_{\mathrm{ROE}}(U_L, U_R, \hat{\mathbf{n}}) = \frac{1}{2}\hat{\mathbf{n}} \cdot \big(\mathscr{F}(U_L) + \mathscr{F}(U_R)\big) - \frac{1}{2}\bar{R}|\bar{\Lambda}^*|\bar{L}^{-1}(W_R - W_L), \tag{6}$$

where the last term in expression (6) is a dissipation term whose factors are described in the following. Let matrices Λ and R be the diagonal eigenvalue matrix and the right eigenvector matrix in the eigendecomposition of the Jacobian with respect to the conservative variables, that is,

$$\frac{\partial(\hat{\mathbf{n}} \cdot \mathscr{F})}{\partial U} = R\Lambda R^{-1}. \tag{7}$$

Moreover, let $W = [\rho, \mathbf{u}, p]^T$ be the primitive variables vector, and let matrix L satisfy

$$L^{-1}(W_R - W_L) = R^{-1}(U_R - U_L). \tag{8}$$

Explicit expressions of L, R, and Λ can for example be found in the books by Blazek [7] and Hirsch [17]. The elements of matrices $\bar{R}$, $\bar{\Lambda}^*$, and $\bar{L}^{-1}$ in expression (6) are evaluated using the Roe averages

$$\begin{aligned} \bar{\rho} &= \sqrt{\rho_L \rho_R}, \quad \bar{\mathbf{u}} = \frac{\mathbf{u}_L\sqrt{\rho_L} + \mathbf{u}_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\ \bar{H} &= \frac{H_L\sqrt{\rho_L} + H_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \quad \bar{c} = \sqrt{(\gamma - 1)\,[\bar{H} - |\bar{\mathbf{u}}|^2]}. \end{aligned} \tag{9}$$

Also, to prevent the eigenvalues to vanish, we utilize a standard entropy fix

$$|\bar{\lambda}_i^*| = \begin{cases} \dfrac{\bar{\lambda}_i^2 - \delta^2}{2\delta}, & |\bar{\lambda}_i| \le \delta, \\ |\bar{\lambda}_i|^2, & |\bar{\lambda}_i| > \delta. \end{cases} \tag{10}$$

Also available in our implementation is the much simpler Local Lax–Friedrichs flux

$$\mathscr{F}^*_{\mathrm{LLF}}(U_L,U_R,\hat{\mathbf{n}}) = \frac{1}{2}\hat{\mathbf{n}}\cdot\left(\mathscr{F}(U_L)+\mathscr{F}(U_R)\right) - \frac{C}{2}(U_R-U_L), \tag{11}$$

where C is an upper bound for the absolute values of the eigenvalues of the Jacobian, $\partial(\hat{\mathbf{n}}\cdot\mathscr{F}(U))/\partial U$. We use

$$C = \max\left(|\hat{\mathbf{n}}\cdot\mathbf{u}_L|+c_L, |\hat{\mathbf{n}}\cdot\mathbf{u}_R|+c_R\right), \tag{12}$$

where $c=\sqrt{\gamma p/\rho}$ is the speed of sound. The choice of Roe or Local Lax–Friedrichs numerical flux appears not to cause any significant differences performance wise, neither regarding computational effort nor accuracy.

Boundary conditions are usually imposed simply by appropriate modifications of the remote condition U_R. At the outer free stream, we apply $U_R = U_\infty$ where U_∞ is the free stream values of the solution vector. To impose the solid wall boundary condition, we have implemented two options: the first imposes symmetry and the second uses an explicitly modified flux to enforce non penetration. Both give similar results, both with respect to computational effort and accuracy.

The first option uses the the same Roe or Local Lax–Friedrichs flux at the wall as everywhere else, but imposes symmetry by defining U_R from U_L using a reflected momentum vector:

$$\rho_R = \rho_L, \quad (\rho\mathbf{u})_R = (\rho\mathbf{u})_L - 2(\hat{\mathbf{n}}\cdot(\rho\mathbf{u})_L)\hat{\mathbf{n}}, \quad (\rho E)_R = (\rho E)_L. \tag{13}$$

The second option uses the numerical flux

$$\mathscr{F}^*_{\mathrm{W}}(U,\hat{\mathbf{n}}) = \begin{pmatrix} 0 \\ p\hat{\mathbf{n}} \\ 0 \end{pmatrix}, \tag{14}$$

that is, $\mathscr{F}^*_{\mathrm{W}}(U,\hat{\mathbf{n}}) = \hat{\mathbf{n}}\cdot\mathscr{F}(U)|_{\hat{\mathbf{n}}\cdot\mathbf{u}=0}$ at the wall, instead of the Roe or Local Lax–Friedrichs flux.

The discussion above covers both the usual cell-centered formulation of the DG method and our vertex-centered scheme. In the next sections, we introduce the particularities with our approach: the construction of a macro element on the dual mesh and the use of curved boundaries for these elements.

3 The Vertex-Centered Macro Element

The finite elements of the standard (cell-centered) DG method use polynomials defined separately on each mesh cell, which means that the control volumes K_m discussed in Section 2 coincide with the mesh cells, typically triangles or quadrilaterals in two space dimensions. In our vertex-centered DG method we use another choice of control volumes K_m, defined on a so-called dual mesh.

A preprocessor constructs the dual mesh and necessary data structures; Figure 2 illustrates the procedure. From the primal mesh, top left in Figure 2, the preprocessor constructs the dual mesh shown in top right in Figure 2. A dual mesh can be constructed in several ways, as discussed by Barth [3]; here we choose just to connect the centroids of adjacent triangles to each other with a new edge. Although Figure 2 shows a uniform mesh, dual meshes can be constructed for any nondegenerate mesh. Next we triangulate the dual mesh (bottom left in Figure 2) and define our finite element on each dual cell as the macro element consisting of standard triangular Lagrange elements of order p. That is, the functions are continuous on each K_m, and piecewise polynomials of degree p on each sub triangle of K_m. Note that we allow discontinuities in the solution between adjacent dual cells, but that the solution within each dual cell is continuous. Thus, no flux evaluations are necessary at the internal edges between the sub triangles in the dual cells. Indeed, any internal flux contribution would vanish since the left and right states are identical due to the continuity of the approximating functions across such edges. Note also that the element type and order may be different on different dual cells. At the bottom right in Figure 2, we see an example where all boundary macro elements and the leftmost of the three inner macro elements are of constant type ($p = 0$), which corresponds to the vertex-centered FV method, whereas the center and right interior macro elements are linear ($p = 1$) and quadratic ($p = 2$), respectively. Note the multiple nodes, associated with the possibility of jump discontinuities, occurring at boundaries of the dual cells.

A common method to solve a problem such as the discrete version of equation (1) is to march an unsteady version of the equation to steady state using an explicit

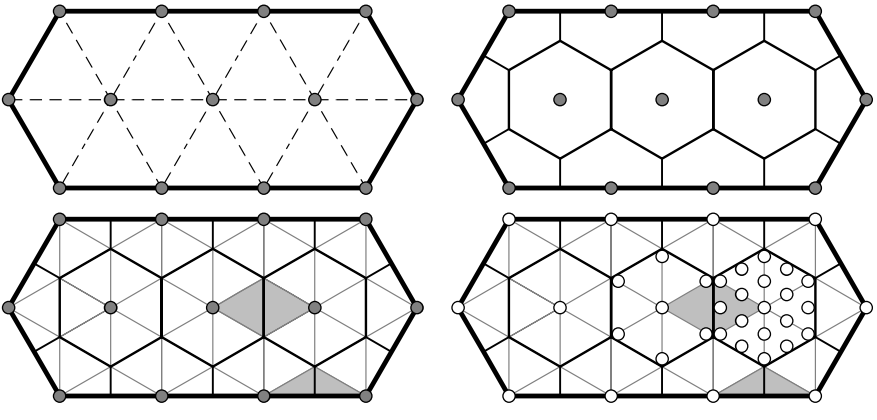


Fig. 2 Preprocessing stages for generating macro elements on the dual mesh. From the primal mesh (top left) the preprocessor constructs a dual mesh (top right) that contains as many polygonal dual cells K_m as the number of mesh vertices in the primal mesh. We triangulate each dual cell (bottom left) and define a macro element on each dual cell (bottom right), where the white circles are the computational nodes.

Runge–Kutta scheme. This strategy is often combined with convergence acceleration strategies such as local time stepping and multigrid; both have been implemented and the multigrid approach is discussed in our other contribution to this volume [11]. A crucial step in such an algorithm is the computation of the residual, which is a vector of dimension equal to the number of degrees of freedom for $\mathcal{V}_h$ for each equation that is solved.

As mentioned in the introduction, implementations of the residual calculation for vertex-centered FV methods are typically *edge-based*, and we now shortly indicate how to extend this approach to the current DG method.

Pointers to the degrees of freedom for $\mathcal{V}_h^{d+2}$ as well as geometric information are stored in a list associated with the edges in the primal mesh. An additional list associated with the domain boundary edges is also required to set boundary conditions.

For each edge in the primal mesh, we associate the two primal mesh vertices i and j connected by the edge, and the normal vector $\mathbf{n}_{ij}$ (with $|\mathbf{n}_{ij}| = |\partial T_{ij}|$) to the intersection of the boundaries of control volumes K_i and K_j. To compute the contribution to the residual associated with this edge, the values of the unknowns at nodes i and j, and the normal $\mathbf{n}_{ij}$ constitute all the information that is needed for $p = 0$ (the finite volume case). For higher orders, we also associate two sub triangles, T_i and T_j, of control volumes K_i and K_j, and a larger set of nodes indices $(i_1, i_2, \ldots; j_1, j_2, \ldots)$ associated with the added degrees of freedom (left in Figure 3). Nodes i_1 and j_1 coincide with primal mesh vertices i and j of the FV method.

To impose boundary conditions, we utilize a list of boundary edges. For each such edge that is not curved, as illustrated to the right in Figure 3, we associate boundary vertices i and j connected by the edge and boundary normals $\mathbf{n}_{ib}$ and $\mathbf{n}_{jb}$ (of equal length, so stored as one normal $\mathbf{n}_{ijb} = \mathbf{n}_{ib} + \mathbf{n}_{jb}$) associated with the intersection of the boundaries of control volumes K_i and K_j with the domain boundary. For curved boundaries, more nodes are needed along the boundary together with miscellaneous additional information, as discussed in Section 4.

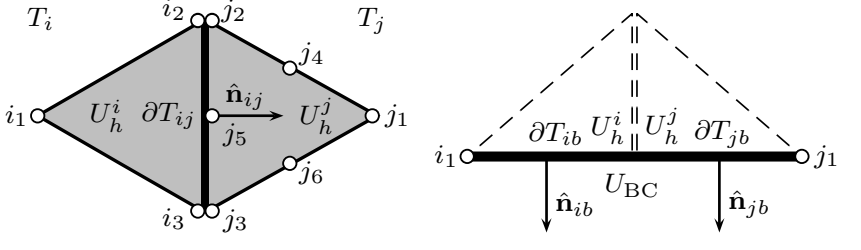


Fig. 3 Edge data structures with geometric data and spatial placement of the computational nodes. Left: The triangles T_i and T_j share the boundary ∂T_{ij}, and the outward unit normal of T_i on ∂T_{ij} is $\hat{\mathbf{n}}_{ij}$. Restrictions of the function $U_h \in \mathcal{V}_h^{d+2}$ onto T_i and T_j are denoted U_h^i and U_h^j. Right: The boundary edge is the union of ∂T_{ib} and ∂T_{jb}, and the domain boundary outward unit normals are $\hat{\mathbf{n}}_{ib}$ and $\hat{\mathbf{n}}_{jb}$. The domain boundary data, which is given by the boundary condition, is denoted u_{BC}.

A pseudo-code version of the algorithm can be found in the the article by Berggren *et al.* [6]; in essence, it is the same loops as for the FV code plus the volume integral of the DG scheme. Standard tabulated Gauss quadrature rules, as given, for example, in the book by Solin *et al.* [24], are used to evaluate the integrals, and the evaluation of the basis functions at these points are done once and for all in the beginning of the computation.

The numerically observed convergence rate s in $\|u-u_h\|_{L^2(\Omega)} \leq Ch^s$ for the model problem investigated in [6] is of optimal order for the vertex-centered scheme, that is, $s = p+1$. Also, as expected, the number of degrees of freedom necessary to attain a given error bound is substantially decreased by increasing the order of the method.

4 Curved Geometries

A higher order scheme, such as DG, also needs a higher-order representation of the geometry, and curved elements have to be introduced when curved objects, for example airfoils, are present in the computations. Bassi and Rebay [4] and Cockburn *et al.* [9], for example, has showed that a piecewise-linear approximation of the boundary does not yield satisfactory results. We also observed strong oscillatory solutions in our scheme when using piecewise-linear approximations of the boundary in combination with orders $p \geq 1$.

To account for the curved boundaries, we use a common finite-element strategy, namely polynomial approximations of the curved element edges combined with evaluations of the integrals on reference elements using coordinate transformations. The nodes marked a, b, and c to the left in Figure 4 are vertices in the primal mesh.

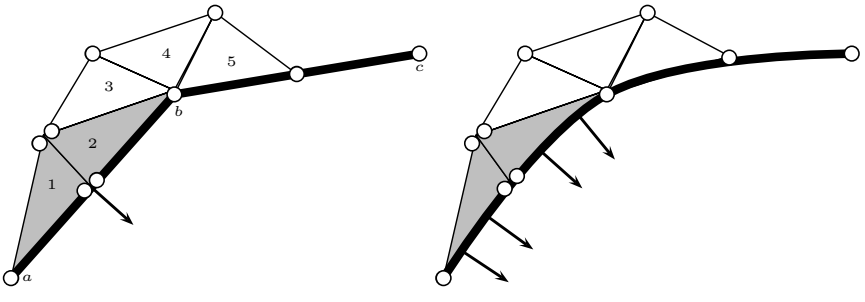


Fig. 4 Boundary elements. Left: The piecewise-linear boundary as obtained by constructing the dual mesh similarly as in the interior. Points a, b, and c are vertices on primal mesh. Triangle 1 is associated to macro element a and triangles $2, 3, 4$, and 5 are associated to macro element b. A boundary edge is marked in gray together with its associated normal. Right: The same part of the boundary as in the left image but taking into account the curvature of the boundary. Now a specific normal, each precomputed, is associated with each quadrature point on the boundary.

If dual meshes are constructed at the boundary in the same way as in the interior, the result will be as depicted to the left in Figure 4. This piecewise-linear boundary generated massive spurious oscillations at the boundary for $p \geq 1$. A simple strategy that yields a somewhat more accurate approximation of the boundary is to move the nodes on the dual mesh that are located between primal nodes (for instance the double nodes located in between nodes a and b in the left of Figure 4) so that they lie at the underlying curved boundary. This change did not significantly reduce the oscillations. Krivodonova's [18] simplified approach to represent a curved boundary was not successful either. However, a polynomial approximation of the boundary and associated normals at a sufficiently high order, as visualized in the right of Figure 4 and as shortly outlined below, worked well.

There will be at most one side that is curved in any sub triangle, since the other two triangle sides are located inside the domain. Now consider a boundary edge and the two associated curved triangles, T_i and T_j (for instance the gray triangles to the right in Figure 4). Denote by $\hat{T}$ the reference triangle with corners at $(0,0)$, $(1,0)$, and $(0,1)$. If we use the polynomial order p_b to approximate the boundary shape, and we use the polynomial order p for the basis functions of the elements, we will have *subparametric* elements if $p_b < p$, *isoparametric* elements if $p_b = p$, and *superparametric* elements if $p_b > p$. Our mesh generator is implemented such that any order p_b can be chosen, regardless of element order p. Numerical experiments indicate that a $p_b \geq 2$ is qualitatively sufficient for the calculations we have made for $p = 0,1,2,3$.

Denote by Φ the mapping from reference triangle $\hat{T}$ to any curved triangle T (T_i or T_j) and by Φ^{-1} the inverse mapping,

$$\Phi(\xi,\eta) = \begin{pmatrix} x \\ y \end{pmatrix}, \quad \Phi^{-1}(x,y) = \begin{pmatrix} \xi \\ \eta \end{pmatrix}. \tag{15}$$

Points $(\xi,\eta) \in \hat{T}$ are transformed into points $(x,y) \in T$ according to

$$x = \sum_{m=0}^{p_b} \sum_{n=0}^{p_b-m} \gamma_{mn} \xi^m \eta^n, \quad y = \sum_{m=0}^{p_b} \sum_{n=0}^{p_b-m} \delta_{mn} \xi^m \eta^n, \tag{16}$$

where p_b is the polynomial order of the boundary and γ_{mn}, δ_{mn} are constants that are computed in the preprocessor. The Jacobian J of Φ,

$$\begin{aligned} J(\xi,\eta) &= \nabla\Phi(\xi,\eta) \\ &= \sum_{m=0}^{p_b} \sum_{n=0}^{p_b-m} \begin{pmatrix} \frac{\partial}{\partial\xi}\gamma_{mn}\xi^m\eta^n & \frac{\partial}{\partial\eta}\gamma_{mn}\xi^m\eta^n \\ \frac{\partial}{\partial\xi}\delta_{mn}\xi^m\eta^n & \frac{\partial}{\partial\eta}\delta_{mn}\xi^m\eta^n \end{pmatrix} \\ &= \sum_{m=0}^{p_b} \sum_{n=0}^{p_b-m} \begin{pmatrix} \gamma_{mn} m \xi^{m-1}\eta^n & \gamma_{mn} n \xi^m \eta^{n-1} \\ \delta_{mn} m \xi^{m-1}\eta^n & \delta_{mn} n \xi^m \eta^{n-1} \end{pmatrix}, \end{aligned} \tag{17}$$

will not be constant for $p_b > 1$. The Jacobians are precomputed at each quadrature point, and are stored and used during computations. For a function F on a curved triangle T, we have

$$\int_T F \,\mathrm{d}V \;=\; \int_{\hat{T}} F \circ \boldsymbol{\Phi}\, |J| \,\mathrm{d}V, \tag{18}$$

where $|J|$ denotes the determinant of J. The gradient of a function g on T is transformed to corresponding function $\hat{g} = g \circ \boldsymbol{\Phi}$ on $\hat{T}$ according to

$$\nabla g = J^{-T} \nabla \hat{g}, \tag{19}$$

where the left- and right-hand sides are evaluated at corresponding points related by mapping $\boldsymbol{\Phi}$. Finite element basis functions ϕ_i on the curved triangle T are defined by the mapping $\phi_i = \hat{\phi}_i \circ \boldsymbol{\Phi}$, where $\hat{\phi}_i$ are the standard Lagrangian basis functions on the reference triangle $\hat{T}$. Let $\mathscr{F}_m$ be row m ($m = 1$, 2, or 3) of flux function (2)[1]. By expressions (18) and (19), it follows that row m of the second term (the volume integral term) in equation (5) can be written

$$\begin{aligned} \int_T \nabla \phi_i \cdot \mathscr{F}_m(U_h) \,\mathrm{d}V &= \int_{\hat{T}} J^{-T} \nabla \hat{\phi}_i \cdot \mathscr{F}_m(U_h \circ \boldsymbol{\Phi}) |J| \,\mathrm{d}V \\ &\approx \sum_{k=1}^{N_{ep}^T} w_k J(\xi_k, \eta_k)^{-T} \nabla \hat{\phi}_i(\xi_k, \eta_k) \cdot \mathscr{F}_m(U_h \circ \boldsymbol{\Phi}(\xi_k, \eta_k)) \, |J(\xi_k, \eta_k)| \end{aligned} \tag{20}$$

where w_k, (ξ_k, η_k), and N_{ep}^T are the weights, coordinates in $\hat{T}$, and number of evaluation points for some appropriate integration rule for triangles.

Let the curved side $\partial T \cap \partial \Omega$ be the image of the restriction of the map $\boldsymbol{\Phi}$ to $\hat{I} = (0,1) \times \{0\}$ (the intersection of $\hat{T}$ with the x-axis). Also, let $\mathscr{F}_m^*(U_L, U_R, \hat{\mathbf{n}})$ be row m in the numerical flux function. The curved-boundary contribution from the first term in equation (5) can then be written

$$\begin{aligned} \int_{\partial T \cap \partial \Omega} \phi_i \mathscr{F}_m^*(U_L, U_R, \hat{\mathbf{n}}) \,\mathrm{d}s &= \int_{\hat{I}} \hat{\phi}_i \mathscr{F}_m^*(U_L \circ \boldsymbol{\Phi}, U_R \circ \boldsymbol{\Phi}, J^{-T} \hat{\mathbf{n}}) |J| \,\mathrm{d}s \\ &\approx \sum_{k=1}^{N_{ep}^I} z_k \hat{\phi}_i(\chi_k, 0) \, |J(\chi_k, 0)| \, \mathscr{F}_m^*(U_L \circ \boldsymbol{\Phi}(\chi_k, 0), U_R \circ \boldsymbol{\Phi}(\chi_k, 0), J(\chi_k, 0)^{-T} \hat{\mathbf{n}}_k) \end{aligned} \tag{21}$$

where N_{ep}^I is the number of evaluation points for the chosen integration rule on the unit interval, and z_k, χ_k are the corresponding weights and coordinates. Note that a specific normal $\mathbf{n}_k = |J(\chi_k, 0)| J(\chi_k, 0)^{-T} \hat{\mathbf{n}}_k$ is used for each evaluation point, and that all normals are precomputed in the preprocessor. Since the curved boundary part of the domain is typically much smaller than the rest of the domain, this extra storage and computational work is mostly negligible.

[1] The rows correspond to the mass, momentum, and energy equations, respectively. Note that the first and third rows are scalar equations, whereas the second row is a system of d equations.

5 Results

In Figure 5 we present results from one of the Mandatory Test Cases (MTC) of the ADIGMA project, MTC1. The figures depict the pressure coefficients using successively higher order elements, $p = 0, 1, 2, 3$, on the same mesh. The element sizes around the airfoil are visible in the $p = 0$ solution at the top left in Figure 5. Some overshoots are visible for the linear element solution, $p = 1$ (top right of Figure 5). At the bottom left of Figure 5, the $p = 2$ solution, small kinks are visible in the element on the leading edge, which vanish for the $p = 3$ solution at the bottom right of Figure 5.

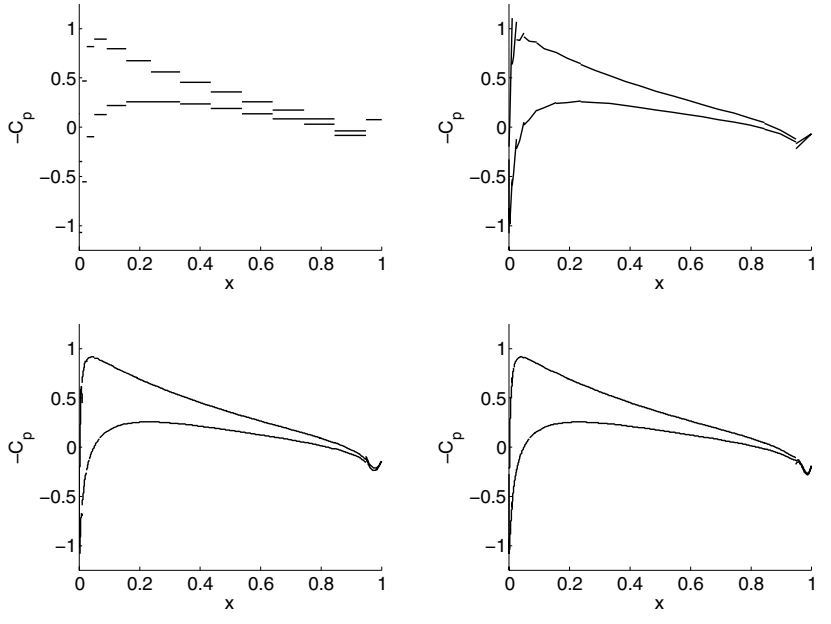


Fig. 5 MTC1 of the ADIGMA project: Pressure coefficients, 2𝔻 Euler, NACA0012, $M = 0.5$, $\alpha = 2.0^\circ$. The actual discontinuous functions, of order $p = 0, 1, 2, 3$, are plotted without any postprocessing or smoothing.

In Figure 6 we show two solutions for MTC2 on the same mesh, but with different order elements. The left picture of Figure 6 depicts the pressure coefficient for constant, $p = 0$ elements, solved with a second order central flux, i.e. a standard finite volume solution provided by Edge. Both shocks are rather poorly resolved and a finer mesh would be required. The right picture of Figure 6 depicts the pressure coefficient for linear, $p = 1$, elements. Note that the oscillations due to shocks are localized to elements adjacent to the shock, and that the oscillations do not spread out into the domain. No shock capturing technique is used. An *hp*-adaption along the lines previously developed for model problems [6] could be used to better resolve

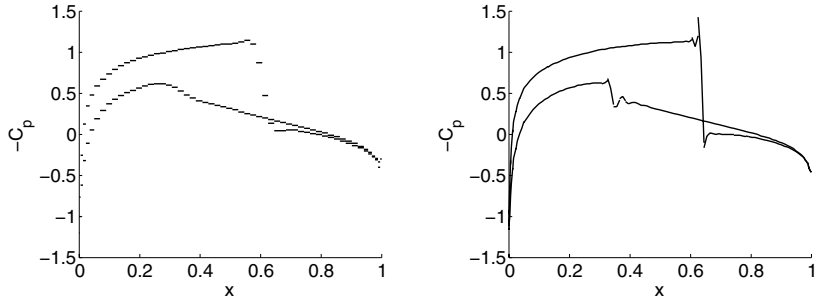


Fig. 6 MTC2 of the ADIGMA project: Pressure coefficients, 2𝔻 Euler, NACA0012, $M = 0.8$, $\alpha = 1.25^\circ$. Left: $p = 0$, central flux (2^{nd} order). Right: $p = 1$, upwind flux (2^{nd} order). The actual discontinuous functions, of order $p = 0, 1$, are plotted without any postprocessing or smoothing.

the shock; another strategy would be to use the sub-cell shock capturing with artificial viscosity by Persson and Peraire [21]. The use of higher orders, $p = 2, 3$, for this transonic case and on this mesh appears to require a shock capturing technique; the solution blows up when approaching a steady-state solution.

6 Conclusions

The effort reported here constitutes a pioneering, first-ever implementation for the Euler equations of the current vertex-centered DG scheme. To the best of our knowledge, this is also the first time that a DG scheme has been implemented within the framework of an edge-based FV code of the type very common in the aeronautical industry.

Acceleration techniques such as local time stepping and agglomerated multigrid has been implemented successfully; the latter is further discussed in our other contribution to this volume [11].

The data structures needed for the current scheme are complex, but compact and memory efficient. The local polynomial approximations take place on the sub elements within the macro elements. Thus, the appropriate grid-size parameter h is the size of the sub elements, and *not* the size of the primal mesh elements. Since no fluxes need to be computed at the boundaries occurring strictly inside the macro elements, the current implementation will be significantly more memory efficient compared to a standard DG based on a mesh with comparable interpolation bound.

Artificial oscillations have been shown to stay localized around shocks. Nevertheless, shock handling will still need to be implemented in order to stabilize higher order approximations and improve the convergence properties of the solver. Another research issue is h-adaption in the current context, which may be more complicated to implement compared to the case for the standard DG scheme.

Although our results are encouraging, more research is needed to decisively determine the benefit of the vertex-centered DG approach. In particular, the method needs to be evaluated on viscous, $3\mathbb{D}$ and turbulent calculations. In addition, particular effort should be guided towards unsteady and aeroacoustic calculation, since the need for high accuracy everywhere in the computational domain may be even more crucial in such cases, compared to steady aerodynamic calculations.

References

1. Abgrall, R.: On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation. J. Comput. Phys. 114(1), 45–58 (1994)
2. Abgrall, R.: Toward the ultimate conservative scheme: following the quest. J. Comput. Phys. 167(2), 277–315 (2001)
3. Barth, T.J., Jespersen, D.C.: The design and application of upwind schemes on unstructured meshes. In: 27th Aerospace Sciences Meeting, AIAA 89-0366, Reno, Nevada (1989)
4. Bassi, F., Rebay, S.: High-order accurate discontinuous finite element solution of the 2d Euler equations. J. Comp. Phys. 138, 251–285 (1997)
5. Berggren, M.: A vertex-centered, dual discontinuous Galerkin method. J. Comput. Appl. Math. 192(1), 175–181 (2006)
6. Berggren, M., Ekström, S.-E., Nordström, J.: A discontinuous Galerkin extension of the vertex-centered edge-based finite volume method. Commun. Comput. Phys. 5, 456–468 (2009)
7. Blazek, J.: Computational Fluid Dynamics, 2nd edn. Elsevier, Amsterdam (2005)
8. Catabriga, L., Coutinho, A.: Implicit SUPG solution of Euler equations using edge-based data structures. Comput. Meth. Appl. Mech. Eng. 191(32), 3477–3490 (2002)
9. Cockburn, B., Karniadakis, G.E., Shu, C.W.: Discontinuous Galerkin Methods: Theory, Computation and Applications. Springer, Heidelberg (1999)
10. Csík, A., Ricchiuto, M., Deconinck, H.: A Conservative Formulation of the Multidimensional Upwind Residual Distribution Schemes for General Nonlinear Conservation Laws. J. Comput. Phys. 179(1), 286–312 (2002)
11. Ekström, S.-E., Berggren, M.: Agglomeration multigrid for the vertex-centered dual discontinuous Galerkin method. In: Kroll, N., et al. (eds.) ADIGMA. NNFM, vol. 113, pp. 301–308. Springer, Heidelberg (2010)
12. Eliasson, P.: EDGE, a Navier–Stokes solver, for unstructured grids. Technical Report FOI-R-0298-SE, Swedish Defence Research Agency (2001)
13. Fezoui, L., Stoufflet, B.: A class of implicit upwind schemes for Euler simulations with unstructured meshes. J. Comput. Phys. 84(1), 174–206 (1989)
14. FOI. Edge - Theoretical Formulation. Technical Report FOI dnr 03-2870, Swedish Defence Research Agency (2007) ISSN-1650-1942
15. Fun3D. Fully Unstructured Navier–Stokes, `http://fun3d.larc.nasa.gov/`
16. Haselbacher, A., McGuirk, J.J., Page, G.J.: Finite volume discretization aspects for viscous flows on mixed unstructured grids. AIAA J. 37(2), 177–184 (1999)
17. Hirsch, C.: Numerical Computation of Internal and External Flows, vol. 2. Wiley, Chichester (1990)
18. Krivodonova, L., Berger, M.: High-order accurate implementation of solid wall boundary conditions in curved geometries. J. Comp. Phys. 211(2), 492–512 (2006)
19. Luo, H., Baum, J.D., Löhner, R.: Edge-based finite element scheme for the Euler equations. AIAA J. 32, 1182–1190 (1994)

20. Morton, K.W., Sonar, T.: Finite volume methods for hyperbolic conservation laws. Acta Numer. 16, 155–238 (2007)
21. Persson, P.-O., Peraire, J.: Sub-Cell shock Capturing for Discontinuous Galerkin Methods. AIAA Paper, 2006-112 (2006)
22. Schwamborn, D., Gerhold, T., Heinrich, R.: The DLR TAU-Code: Recent Applications in Research and Industry. In: Wesseling, P., Onate, E., Périaux, J. (eds.) ECCOMAS CFD 2006 (2006)
23. Smith, T.M., Ober, C.C., Lorber, A.A.: SIERRA/Premo–A New General Purpose Compressible Flow Simulation Code. In: AIAA 32nd Fluid Dynamics Conference, St. Louis (2002)
24. Solin, P., Segeth, K., Dolezel, I.: Higher Order Finite Element Methods. Taylor & Francis Ltd., Abington (2003)

Chapter 5
Explicit One-Step Discontinuous Galerkin Schemes for Unsteady Flow Simulations

Arne Taube, Gregor Gassner, and Claus-Dieter Munz

Abstract. In the following, we describe an explicit discontinuous Galerkin scheme for the compressible Navier-Stokes equations. The scheme is of arbitrary order of accuracy by choosing the polynomial degree of the approximation. It is kept very local so that the solution in each cell only depends on the von Neumann neighbors. Apart from the standard DG framework the computational efficiency is increased by the use of a mixed approach using a modal and a nodal set of basis functions. For the approximation of the viscous fluxes an approximation based on local Riemann solutions is used. At the end we show a high order approximation of the unsteady laminar flow over a NACA0012 profile.

1 Introduction

The aim of our ongoing research is the construction of explicit schemes to simulate unsteady fluid flow in complex geometries. Our explicit space-time expansion discontinuous Galerkin (STE-DG) scheme as described in [15, 9], is formulated in the space-time domain by using a space-time Taylor expansion as a predictor. The discretization is arbitrary order accurate in space and time and due to the scheme's local character enables us to do p-refinement in a very efficient manner. For the Navier-Stokes equations, we solve the diffusive generalized Riemann problem (dGRP) with the flux presented in [8].

In Sect. 2, we list the Navier-Stokes equations and introduce the diffusion matrices. Section 3 contains the description of the proposed DG scheme based on a space-time expansion. The nodal approach increasing the computational efficiency

Arne Taube · Gregor Gassner · Claus-Dieter Munz
Institut für Aerodynamik und Gasdynamik, University of Stuttgart, Pfaffenwaldring 21, 70569 Stuttgart, Germany
e-mail: taube@iag.uni-stuttgart.de, gassner@iag.uni-stuttgart.de, munz@iag.uni-stuttgart.de

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 53–65, 2010.
springerlink.com

of the scheme is described in Sect. 4 followed by an example for our high order numerical scheme in Sect. 5. We draw our conclusion in Sect. 6.

2 Compressible Navier-Stokes Equations

The three dimensional unsteady compressible Navier-Stokes equations written in the flux formulation are given by:

$$U_t + \nabla \cdot \mathbf{F}^a(U) - \nabla \cdot \mathbf{F}^d(U, \nabla U) = S, \tag{1}$$

where U is the vector of the conservative variables $U = (\rho, \rho v_1, \rho v_2, \rho v_3, \rho e)^T$. The Euler fluxes are written in a short hand notation as a row vector $\mathbf{F}^a := \left(F_1^a, F_2^a, F_3^a\right)^T$ which has as coefficients the fluxes into x, y- and z-direction:

$$F_l^a(U) = \begin{pmatrix} \rho v_l \\ \rho v_1 v_l + \delta_{1l} p \\ \rho v_2 v_l + \delta_{2l} p \\ \rho v_3 v_l + \delta_{3l} p \\ \rho e v_l + p v_l \end{pmatrix}, \; l = 1,2,3. \tag{2}$$

Here, we use the usual notation of the physical quantities: ρ, $\mathbf{v} = (v_1, v_2, v_3)^T$, p, and e denote the density, the velocity vector, the pressure, and the specific total energy, respectively. The system is closed with the equation of state of a perfect gas

$$p = \rho RT = (\gamma - 1)\rho \left(e - \frac{1}{2}\mathbf{v}\cdot\mathbf{v}\right) \quad \text{and} \quad e = \frac{1}{2}\mathbf{v}\cdot\mathbf{v} + c_v T \tag{3}$$

with the specific gas constant $R = c_p - c_v$. The adiabatic exponent is $\gamma = \frac{c_p}{c_v}$ with the specific heats c_p, c_v depending on the fluid and assumed to be constant. The diffusion terms are also written in this flux formulation $\mathbf{F}^v := \left(F_1^v, F_2^v, F_3^v\right)^T$ with

$$F_l^v(U, \nabla U) = \begin{pmatrix} 0 \\ \tau_{1l} \\ \tau_{2l} \\ \tau_{3l} \\ \sum_j \tau_{lj} v_j - q_l \end{pmatrix}, \; l = 1,2,3, \tag{4}$$

in which τ_{ij} denotes the components of the viscous stress tensor and $\mathbf{q} = (q_1, q_2, q_3)^T$ denotes the heat flux,

$$\underline{\tau} := \mu \left(\nabla \mathbf{v} + (\nabla \mathbf{v})^T - \frac{2}{3}(\nabla \cdot \mathbf{v})\underline{I}\right), \quad \mathbf{q} := -k\nabla T, \quad \text{with} \quad k = \frac{c_p \mu}{Pr}. \tag{5}$$

Here, the viscosity coefficient μ and the Prandtl number Pr depend on the fluid and are assumed to be constant. S denotes a source term.

The diffusion fluxes may be re-written with the help of the diffusion matrices. For the two-dimensional case, we get

$$\begin{aligned}
\underline{\underline{D}}_{11} &= \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ -\frac{4}{3}v_1 & \frac{4}{3} & 0 & 0 \\ -v_2 & 0 & 1 & 0 \\ -\left(\frac{4}{3}v_1^2+v_2^2+\frac{\gamma}{Pr}\left(e-v^2\right)\right) & \left(\frac{4}{3}-\frac{\gamma}{Pr}\right)v_1 & \left(1-\frac{\gamma}{Pr}\right)v_2 & \frac{\gamma}{Pr} \end{pmatrix} \\
\underline{\underline{D}}_{12} &= \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{2}{3}v_2 & 0 & -\frac{2}{3} & 0 \\ -v_1 & 1 & 0 & 0 \\ -\frac{1}{3}v_1v_2 & v_2 & -\frac{2}{3}v_1 & 0 \end{pmatrix}, \underline{\underline{D}}_{21} = \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ -v_2 & 0 & 1 & 0 \\ \frac{2}{3}v_1 & -\frac{2}{3} & 0 & 0 \\ -\frac{1}{3}v_1v_2 & -\frac{2}{3}v_2 & v_1 & 0 \end{pmatrix} \\
\underline{\underline{D}}_{22} &= \frac{\mu}{\rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ -v_1 & 1 & 0 & 0 \\ -\frac{4}{3}v_2 & 0 & \frac{4}{3} & 0 \\ -\left(v_1^2+\frac{4}{3}v_2^2+\frac{\gamma}{Pr}\left(e-v^2\right)\right) & \left(1-\frac{\gamma}{Pr}\right)v_1 & \left(\frac{4}{3}-\frac{\gamma}{Pr}\right)v_2 & \frac{\gamma}{Pr} \end{pmatrix},
\end{aligned} \tag{6}$$

in the form

$$F_l^v = \underline{\underline{D}}_{l1}U_x + \underline{\underline{D}}_{l2}U_y, \quad l = 1,2\,. \tag{7}$$

This form allows the application of 'two integration by parts' to the diffusion terms. For the definition of the numerical fluxes, the rotational invariance of both, the inviscid Euler fluxes and the diffusion fluxes, are used to reduce the multidimensional problem into a one dimensional problem in interface normal direction. For this one dimensional approximation, the fluxes derived in Sect. 3.4 are used. For the approximation of the Euler flux, the HLLC flux of Toro is used, see [20] for details.

3 The STE-DG Scheme

We take the Navier-Stokes equations (1) as an example for a system of advection diffusion equations which is formulated in conservation form.

3.1 Variational Formulation

For the approximation, the domain Ω is subdivided in non-overlapping spatial grid cells Q_i with surfaces ∂Q_i. As usual the intersection of two grid cells is an edge, a point, or empty in two space dimensions or a surface, an edge or empty in three dimensions.

To obtain a weak formulation of the Navier-Stokes equations (1), we first multiply each component with a spatial test function $\phi = \phi(\mathbf{x})$ and integrate over an arbitrary space-time cell $Q_i^n := Q_i \times [t^n, t^{n+1}]$ to obtain

$$\int_{Q_i^n} \left(U_t + \nabla \cdot \left(\mathbf{F}^a(U) - \mathbf{F}^d(U, \nabla U)\right)\right) \phi \, d\mathbf{x}dt = 0. \tag{8}$$

Next, in the finite element framework we apply integration by parts with respect to the space variables in order to get a coupling between the grid cells:

$$\begin{aligned}\int_{Q_i^n} U_t \phi \, d\mathbf{x}dt - \int_{Q_i^n} \mathbf{F}^a(U) \cdot \nabla\phi \, d\mathbf{x}dt + \int_{Q_i^n} \mathbf{F}^d(U, \nabla U) \cdot \nabla\phi \, d\mathbf{x}dt \\ + \int_{\partial Q_i^n} \mathbf{F}^a(U) \cdot \mathbf{n}\phi \, dsdt - \int_{\partial Q_i^n} \mathbf{F}^d(U, \nabla U) \cdot \mathbf{n}\phi \, dsdt = 0,\end{aligned} \tag{9}$$

where $\mathbf{n}$ denotes the normal vector into the outer direction of the space-time cell's side faces $\partial Q_i^n := \partial Q_i \times \left[t^n, t^{n+1}\right]$.

Using this variational formulation as a basis for the discontinuous Galerkin scheme results in non-optimal convergence behavior with respect to the diffusion terms, as the scheme is not adjoint consistent, [1]. To overcome this problem, Bassi and Rebay [3] introduced a mixed finite element approach, where they reformulate the second order problem into a first order system. However, the disadvantage of this approach is that auxiliary variables are introduced, resulting in an increase of the computational effort, especially for systems of equations. In Gassner *et al.* [8] a new variational formulation for diffusion problems is introduced, where the need for auxiliary variables is circumvented by the application of a second integration by part. We recognize that the volume integral of the diffusion term in (9) still allows a second integration by parts to the viscous volume integral, using the homogeneity property (7) of the viscous flux with respect to the gradient.

Inserting this into (9) yields the weak formulation of the Navier-Stokes equations as

$$\begin{aligned}\int_{\Omega_i^n} U_t \cdot \Phi \, d\mathbf{x}dt + \int_{\partial\Omega_i^n} (F_n^a - F_n^v) \cdot \Phi \, dsdt - \int_{\Omega_i^n} (\mathbf{F}^a - \mathbf{F}^v) \cdot (\nabla\Phi) \, d\mathbf{x}dt \\ + \int_{\partial\Omega_i^n} h(U, \mathbf{n}, \nabla\Phi) \, dsdt = 0,\end{aligned} \tag{10}$$

with the additional diffusion flux

$$h(U, \mathbf{n}, \nabla\Phi) := U \cdot F_n^{vv}(U, \nabla\Phi) - \left[U \cdot F_n^{vv}(U, \nabla\Phi)\right]_{INT}. \tag{11}$$

To get this diffusion flux in (11) we use the artifice of Bassi and Rebay [3], who use partial integration back and forth to introduce the global lifting operator. The expression in brackets, $[:]_{INT}$, in (11) denotes that only the state from inside the cell is used. Considering the discontinuous Galerkin discretization in the next section this generates an additional diffusion flux, $h(U, \mathbf{n}, \nabla\Phi)$, which gives adjoint consistency and is similar to the 'symmetric term' in the Symmetric Interior Penalty (SIP) method as proposed by Hartmann and Houston [12] and the term of the BR2 scheme of [3] involving the global lifting operator. If the function U is continuous, then this flux is zero. With the definition of suitable numerical fluxes and a suitable

numerical state for the second diffusion surface integral we get an adjoint consistent formulation and thus a discretization with optimal order of convergence. We come back to this point within the section about numerical diffusion fluxes 3.4.

3.2 The Fully Discrete Scheme

At each fixed time level, the approximative solution $U_h = U_h(\mathbf{x},t)$ is a piecewise polynomial in space. In the grid cell Q_i it is represented by

$$U_i(\mathbf{x},t) = \sum_{l=0}^{\mathscr{N}(d,p)} \hat{U}_{i,l}(t)\,\phi_{i,l}(\mathbf{x})\,, \tag{12}$$

where $\phi_{i,l} = \phi_{i,l}(\mathbf{x})$ are basis functions which span the space of polynomials of degree p with support Q_i and $\hat{U}_{i,l}(t)\,, l = 1,..,\mathscr{N}$, are the time dependent degrees of freedom (DOF) in the grid cell Q_i. The number of DOF depends on the space dimension d and on the polynomial degree p and is given by

$$\mathscr{N}(d,p) = \frac{1}{d!}\prod_{j=1}^{d}(p+j)\,. \tag{13}$$

We use a set of orthonormal basis functions which are constructed using the Gram-Schmidt orthogonalization algorithm yielding a diagonal mass matrix, even for elements with curved boundaries. The temporal evolution of the degrees of freedom is represented by discrete values at the different time levels, e.g. at the time level t_n, we have

$$U_i(\mathbf{x},t_n) = \sum_{l=0}^{\mathscr{N}(d,p)} \hat{U}_{i,l}(t_n)\,\phi_{i,l}(\mathbf{x}) =: \sum_{l=0}^{\mathscr{N}(d,p)} \hat{U}_{i,l}^{n}\phi_{i,l}(\mathbf{x})\,. \tag{14}$$

We insert this trial function into the weak formulation (10) and choose the test functions equal to the basis functions $\phi_{i,l} = \phi_{i,l}(\mathbf{x})$.

As the global approximation U_h may be discontinuous across element interfaces, we have to introduce suitable numerical flux functions, to guarantee stability and consistency of the discretization. The numerical fluxes are denoted by G_n, G_n^v, and g_n^{vv} for the Euler flux F_n^a and the two diffusion fluxes F_n^v and h, respectively. They are defined and discussed in detail in Sect. 3.4.

3.3 Prediction Based on a Local Approximate Solution

To keep the whole scheme explicit, we use a predictor that approximates the time evolution of the data in the considered time step, but only within the grid cell Q_i. This may be considered as to solve the Cauchy problem

$$\begin{aligned} U_t + \nabla\cdot\mathbf{F}^a(U) &= \nabla\cdot\mathbf{F}^v(U,\nabla U) \quad \text{in} \quad \mathbb{R}^d\times[0,\Delta t] \\ U(\mathbf{x},0) &= U_i(\mathbf{x},t_n) \quad \forall\mathbf{x}\in\mathbb{R}^d, \end{aligned} \tag{15}$$

where $U_i(\mathbf{x},t_n)$ is the DG polynomial at time $t=t_n$ in grid cell Q_i extended to $\mathbb{R}^d$. In this predictor no values from the neighboring grid cells are included. We use this solution in the space-time grid cell Q_i^n to determine in the variational formulation the space-time volume integrals and to get the arguments for the numerical fluxes from the interior for the surface integrals. There are different approaches to get an approximate solution of this Cauchy problem (15) in Q_i^n or better at the Gauss points of the time quadrature formulae.

3.3.1 Space-Time Expansion and the Cauchy-Kovalevskaya Procedure

For our scheme, it is important to get approximate values at the space-time Gauss points. The basis of the proposed time discretization approach is a Taylor expansion in space and time,

$$U(\mathbf{x},t) = U(\mathbf{x}_i,t_n) + \sum_{j=1}^{p} \frac{1}{j!} \left((t-t_n)\frac{\partial}{\partial t} + (\mathbf{x}-\mathbf{x}_i)\cdot\nabla \right)^j U(\mathbf{x}_i,t_n), \tag{16}$$

about the element Q_i's barycenter $\mathbf{x}_i$ at time t_n. This space-time Taylor expansion provides approximate values for U and ∇U at all space-time points $(\mathbf{x},t) \in \Omega_i^n$, if the values of the space-time derivatives at the state of the expansion are known.

While the pure space derivatives at $(\mathbf{x}_i,t_n)$ are readily available within the DG framework, the time derivatives and mixed space-time derivatives can be computed using the so-called Cauchy-Kovalevskaya (CK) procedure. The use of a Taylor expansion in space-time at the barycenter has already been proposed by Harten *et al.* [11] within the ENO finite volume framework. Later Toro *et al.* used the CK procedure in [19, 21] for the multi-dimensional Euler equations within the ADER-FV (Arbitrary order using DERivatives Finite Volume) schemes to get a high order approximation of the generalized Riemann problem. This idea was extended to the ADER-DG schemes by Dumbser and Munz [4, 5].

3.4 The Numerical Fluxes

For the approximation of the fluxes normal to a grid cell's edge, the Navier-Stokes equations' rotational invariance is used. The following description is done for two space dimensions, but its implementation in 3D is straightforward. In a first step, every Gauss point χ_j^k on the edge $S_k \in \partial Q_i$, the state vector U and the gradient of the state vector ∇U are rotated from the global (x,y)-system into the edge aligned $(\widetilde{x},\widetilde{y})$-system. With the two dimensional rotation matrix

$$\underline{R} := \underline{R}\left(\chi_j^k\right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & n_1\left(\chi_j^k\right) & n_2\left(\chi_j^k\right) & 0 \\ 0 & -n_2\left(\chi_j^k\right) & n_1\left(\chi_j^k\right) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{17}$$

the transformation reads as,

$$\mathscr{U} := \underline{R}U, \quad \mathscr{U}_{\mathbf{n}} := \underline{R}\left(U_x n_1 + U_y n_2\right) \quad \text{and} \quad \mathscr{U}_{\mathbf{t}} := \underline{R}\left(-U_x n_2 + U_y n_1\right) \tag{18}$$

for every surface Gauss point χ_j^k. We denote the derivative into normal and tangential direction in the local coordinate system with $\mathscr{U}_{\mathbf{n}}$ and $\mathscr{U}_{\mathbf{t}}$, respectively.

For the inviscid Euler flux, a number of different numerical fluxes have been proposed in the past, see e.g. [20]. A recent investigation of different numerical flux functions within the RKDG scheme is given by Qiu *et al.* [18]. In our calculations, we chose the HLLC flux,

$$G_n\left(U^{\pm}\left(\chi_j^k, \tau_m\right)\right) := \underline{R}^{-1}\mathscr{G}_{HLLC}\left(\mathscr{U}^{-}\left(\chi_j^k, \tau_m\right), \mathscr{U}^{+}\left(\chi_j^k, \tau_m\right)\right), \tag{19}$$

evaluated at every edge's space-time Gauss point $\left(\chi_j^k, \tau_m\right)$. Here, $\mathscr{U}^{\pm}$ denotes the values of the limit at the grid cell edge from the two sides with respect to the outer normal direction.

For the approximation of the diffusion fluxes, we also use the viscous fluxes' rotational invariance

$$\begin{aligned} F_n^v\left(U, \nabla U\right) &= F_1^v\left(U, \nabla U\right) n_1 + F_2^v\left(U, \nabla U\right) n_2 \\ &= \underline{R}^{-1} F_1^v\left(\mathscr{U}, \left(\mathscr{U}_{\mathbf{n}}, \mathscr{U}_{\mathbf{t}}\right)^T\right), \qquad \forall \mathbf{n} \in \mathbb{R}^2, \end{aligned} \tag{20}$$

to reduce the two dimensional problem to a one dimensional problem into the direction normal to the edge:

$$G_n^v\left(U^{\pm}\left(\chi_j^k, \tau_m\right), \nabla U^{\pm}\left(\chi_j^k, \tau_m\right)\right) := \underline{R}^{-1}\mathscr{G}^v\left(\mathscr{U}^{\pm}, \left(\mathscr{U}_{\mathbf{n}}^{\pm}, \mathscr{U}_{\mathbf{t}}^{\pm}\right)^T\right). \tag{21}$$

We further split the flux $\mathscr{G}^v$ into the normal and tangential part

$$\mathscr{G}_{\mathbf{n}}^v := \underline{D}_{11}\left(\mathscr{U}^{\pm}\right)\mathscr{U}_{\mathbf{n}}^{\pm}, \quad \mathscr{G}_{\mathbf{t}}^v := \underline{D}_{12}\left(\mathscr{U}^{\pm}\right)\mathscr{U}_{\mathbf{t}}^{\pm}, \tag{22}$$

respectively.

In [8], Gassner *et al.* construct a diffusion flux in one space dimension which is based on the solution of the generalized Riemann problem for systems of linear diffusion equations and called it the dGRP-flux (diffusive Generalized Riemann Problem). They also show in [8] how to extend this numerical flux to nonlinear diffusion problems. An important difference to the hyperbolic case is that for diffusion problems piecewise constant initial data leads to an inconsistent flux approximation. At least piecewise linear initial data are necessary to take the character of the second order differential equation into account. In the following, we adopt this approach from [8] to the approximation of the viscous flux (22) of the Navier-Stokes equations.

The first step of the approximation is to linearize the nonlinear diffusion flux $\mathscr{G}_n^v$ at a suitable state. As diffusion does not prefer any direction, a natural choice for the trace of the approximation on the edge S_k is the arithmetic mean

$$\mathscr{U}\big|_{S_k} \approx \{\mathscr{U}\} := \frac{1}{2}\left(\mathscr{U}^+ + \mathscr{U}^-\right). \tag{23}$$

The part of the numerical diffusion flux into *normal* direction is then based on the initial value problem for the linearized diffusion system with piecewise linear data

$$\begin{aligned} \frac{\partial}{\partial t} V &= \underline{D}_{11}\left(\{\mathscr{U}\}\right) \frac{\partial^2}{\partial \widetilde{x}^2} V, \\ V\left(\widetilde{x},0\right) &= \begin{cases} \mathscr{U}^+ + \widetilde{x}\mathscr{U}_{\mathbf{n}}^+, \text{ for } \widetilde{x} > 0, \\ \mathscr{U}^- + \widetilde{x}\mathscr{U}_{\mathbf{n}}^-, \text{ for } \widetilde{x} < 0, \end{cases} \end{aligned} \tag{24}$$

where $\widetilde{x}$ denotes the coordinate into the normal direction. For physically meaningful states $\mathscr{U}$, the diffusion matrix can be diagonalized and the system (24) can be transformed into a decoupled system of equations. The exact solution of the scalar diffusion problem for piecewise linear initial values can be calculated and used to define the time averaged diffusion flux in the diagonalized form. The transformation back leads to the diffusion flux approximation for the Navier-Stokes equations, for more details see [8].

The two numerical fluxes can be written in the form

$$\mathscr{G}^v_{dGRP^*}\left(\mathscr{U}^{\pm}, \left(\mathscr{U}_{\mathbf{n}}^{\pm}, \mathscr{U}_{\mathbf{t}}^{\pm}\right)^T\right) = F_1^v\left(\{\mathscr{U}\}, \left(\frac{\eta}{\Delta x_k}[\![\mathscr{U}]\!] + \{\mathscr{U}_{\mathbf{n}}\}, \{\mathscr{U}_{\mathbf{t}}\}\right)^T\right) \tag{25}$$

and

$$g_n^{vv}\left(U^{\pm}, \mathbf{n}, \nabla\Phi^-\right) \approx \frac{1}{2}\underline{R}^{-1}\left(\underline{D}_{11}\left(\{\mathscr{U}\}\left(\chi_j^k, \tau_m\right)\right)\left[\!\left[\mathscr{U}\left(\chi_j^k, \tau_m\right)\right]\!\right]\right) \cdot \left(\Phi_{\mathbf{n}}^-\left(\chi_j^k\right)\right). \tag{26}$$

The parameter η depends on the parabolic time step restriction and is specified in Sect. 2.1 in Chapter 30. It may be considered as a penalization parameter of the jump. Hence, equation (25) has a similar form as the symmetric interior penalty flux as proposed in [12]. A main difference between the SIP-DG flux and the dGRP flux is a jump term of $U[1] = \rho$ in the continuity equation. Numerical experiments show that this causes the SIP-DG approach to be only sub-optimal for trial functions with even order p.

4 Nodal Integration of Fluxes

In this section we focus our considerations to different sets of basis functions on a grid cell $Q \subset \mathbb{R}^d$.

4.1 The Nodal Elements

The elements of the *monomial basis* $\{\pi_i\}_{i=1,\ldots,\mathscr{N}}$ of the space of polynomials with degree less or equal than p can be written as

$$\pi_i(\mathbf{x}) = x_1^{\alpha_1^i} \cdot \ldots \cdot x_d^{\alpha_d^i} \quad \text{with} \quad 0 \leq \alpha_1^i + \ldots + \alpha_d^i \leq p\,. \tag{27}$$

The dimension $\mathscr{N}$ of this space depends on the polynomial degree p and on the spatial dimension d and is given by (13). Based on the monomial basis $\{\pi_i\}_{i=1,\ldots,\mathscr{N}}$, a construction of an orthonormal basis $\{\varphi_i\}_{i=1,\ldots,\mathscr{N}}$ with property

$$\int_Q \varphi_i(\mathbf{x})\,\varphi_j(\mathbf{x})\,d\mathbf{x} = \delta_{ij}, \tag{28}$$

can be done for any grid cell shape using the Gram-Schmidt orthogonalization procedure. With this *modal* basis we are able to define also sets of *nodal* basis functions. Given a set of interpolation points $\{\xi_j\}_{j=1,\ldots,\mathscr{M}_I} \subset Q_i$ we construct the nodal Lagrange basis $\{\psi_j\}_{j=1,\ldots,\mathscr{M}_I}$ and the nodal degrees of freedom $\underline{\hat{U}}$ from the conditions

$$\begin{aligned} &\psi_j(\xi_i) = \delta_{ij}, \\ &U(\mathbf{x}) := \sum_{j=1}^{\mathscr{N}} \hat{U}_j \varphi_j(\mathbf{x}) \overset{!}{=} \sum_{i=1}^{\mathscr{M}_I} \widetilde{U}_i \psi_i(\mathbf{x})\,. \end{aligned} \tag{29}$$

Combining these conditions yields the transformations

$$\underline{\underline{V}}\,\underline{\hat{U}} = \underline{\widetilde{U}} \text{ and } \underline{\underline{V}}^T\,\underline{\psi} = \underline{\varphi}, \tag{30}$$

where $\underline{\underline{V}}$ denotes the generalized Vandermonde matrix with entries

$$V_{ij} = \varphi_j(\xi_i), \quad i = 1,\ldots,\mathscr{M}_I;\ j = 1,\ldots,\mathscr{N}, \tag{31}$$

and where $\underline{\hat{U}}$, $\underline{\varphi}$, $\underline{\widetilde{u}}$, and $\underline{\psi}$ denote the vectors with components $\hat{U}_i$, φ_i for $i = 1,\ldots,\mathscr{N}$ and $\widetilde{U}_j$, ψ_j for $j = 1,\ldots,\mathscr{M}_I$, respectively. For clarity we introduced an underline for these vectors to differ them from the spatial vectors.

For $\mathscr{M}_I \neq \mathscr{N}$ the inverse of the Vandermonde matrix is not uniquely defined. To avoid this problem, one has to extend the modal basis from dimension $\mathscr{N}$ to dimension $\mathscr{M}_I$. We refer to Lörcher and Munz [16] for a strategy to find basis extensions for non-tensor product interpolation on a Cartesian grid. The extension of this approach to the general case is not straightforward, since the non-singularity of the Vandermonde matrix is not guaranteed. To overcome this issue a singular value decomposition based strategy is used to define the following pseudo-inverse transformations

$$\underline{\hat{U}} = \underline{\underline{V}}^{-1}\,\underline{\widetilde{U}} \text{ and } \underline{\psi} = \underline{\underline{V}}^{-T}\,\underline{\phi}. \tag{32}$$

Using the pseudo-inverse Vandermonde matrix, condition (29) is only satisfied in the *least squares* sense.

4.2 The Nodal Integration Approach

We restrict our attention to sets of interpolation points $\Omega_I := \{\xi_i\}_{i=1,\ldots,\mathscr{M}_I}$ which have first been described in [10] and discuss the nodal approach to increase the computational efficiency of the scheme.

In the standard modal DG implementations, the evaluation of the integrals is usually done with Gauss integration. For instance we get the following approximation for the first volume integral in the variational formulation (9)

$$\int_{Q_i} F_1(U_h) \frac{\partial \phi_j}{\partial x_1}(\mathbf{x})\, d\mathbf{x} \approx \sum_{j=1}^{p^d} F_1(U_h(\chi_j,t)) \frac{\partial \phi_j}{\partial x_1}(\chi_j)\, \omega_j, \tag{33}$$

where ω_j and χ_j are the Gauss weights and Gauss positions, respectively. If we consider a hexahedron with a $p = 5$ approximation, we get $p^d = 125$ evaluation points with this strategy for the first volume integral, and, following [10] with $\pi = (0,0)$, $\mathscr{M}_I = 80$ evaluation points for the nodal DG scheme. Keeping in mind that the Gauss type approach needs additionally $(p+1)^{(d-1)}$ Gauss evaluation points for each of the six sides to approximate the surface integral, it is obvious that the nodal framework enhances the efficiency of *modal DG* implementations.

We again focus for simplicity on the approximation of the first volume integral of the modal formulation (33). But instead of using Gauss integration, we use the nodal elements of Sect. 4.1 to build a high order interpolation of the flux F_1

$$F_1(U_h) \approx \underline{\psi}^T \underline{\widetilde{F}}_1, \tag{34}$$

where again $\left(\widetilde{F_1}\right)_j := F_1\left(U_h\left(\xi_{\mathbf{j}}\right)\right)$. Inserting this into the volume integral yields

$$\int_{Q} F_1(U_h) \frac{\partial \phi_j}{\partial x_1}(\mathbf{x})\, d\mathbf{x} \approx \underline{\underline{K}}^1 \underline{\widetilde{F}}_1, \tag{35}$$

where the general stiffness matrix is given by

$$\underline{\underline{K}}^1 := \int_{Q} \frac{\partial \underline{\phi}}{\partial x_1}(\mathbf{x})\, \underline{\psi}^T(\mathbf{x})\, d\mathbf{x} = \underline{\underline{K}}^{1,M} \underline{\underline{V}}^{-1}. \tag{36}$$

The evaluation of the stiffness matrix can be done with Gauss integration in an initial phase of the simulation, yielding a quadrature free approach. The surface integrals are treated in a similar manner. Comparing with the 'traditional' modal implementation, we have now with this approach approximately the computational complexity of the nodal DG scheme, but with only $\mathscr{N}$ modal degrees of freedom per grid cell, compared to $\mathscr{M}_I$ nodal degrees of freedom.

5 High Order Numerical Results

Gassner and Lörcher already demonstrated the STE-DG scheme's experimental order of convergence with an inhomogeneous problem for the two- and three-dimensional Navier-Stokes equations in [6, 14, 7], where an exact solution for the inhomogeneous problem is constructed in a simple way by inserting some reasonable functions into the homogeneous equations. The obtained residuum is then prescribed as a source term and an exact solution of the non-homogeneous problem is found. This model problem has been solved with the modal STE-DG scheme with nodal integration and the results show that the optimal order of convergence $EOC = p + 1$ is achieved.

5.1 Laminar Flow past a NACA0012 Profile

The following section deals with the numerical simulation of the fluid flow around a symmetric two dimensional NACA0012 airfoil. The flow has the following parameters, angle of attack $\alpha = 2^\circ$, free stream Mach number $Ma_\infty = 0.5$ and the Reynolds number is $Re_\infty = 5000$ based on the length of the airfoil $c = 1.0$. For the high order results, the treatment of curved boundaries is an important issue in DG schemes as shown for example by [2, 13, 17].

It is a laminar but yet unsteady flow computation till $t_{end} = 0.4$. The simulation has been carried out with fixed approximation order $\mathscr{O}6$ and with 6^{th}-order boundary representation as described by Lörcher [14]. In the following Fig. 1, 20 equally spaced Mach contour lines are plotted from $Ma = 0.05 \ldots 0.6$ along with the mesh.

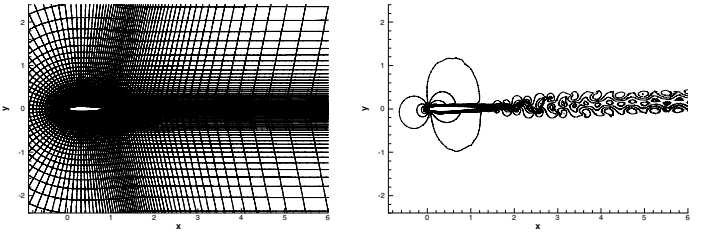


Fig. 1 Fluid flow around NACA0012 for $Re_\infty = 5000$, $Ma_\infty = 0.5$. C-type mesh (left) and Mach contour lines (right) at 6^{th}-order. STE-DG: DOF=134400, $95.58 \frac{[piu]}{ms}$.

The test case shows non-stationary behavior and the mesh's structured character yields to a rather fine zone in the profile's wake which is giving way to a nice resolution of the occurring vortex street in that region. Nevertheless, a minimum resolution or number of DOF is required for the vortex street to show up. The increase in order and accuracy goes hand in hand with an increase in degrees of freedom (DOF) and eventually CPU-time, measured in ADIGMA performance index units

[piu]. It could be shown that the high order STE-DG scheme can produce much higher resolved results with less DOF on coarser meshes.

6 Conclusion

We have described our explicit space-time expansion discontinuous Galerkin (STE-DG) scheme and its building blocks in detail. The space-time discretization is kept explicit by starting with a predictor which gives the approximate values at the intermediate time levels needed for higher order accuracy in time. The numerical fluxes at the grid cell interfaces are based on approximative Riemann solvers for the hyperbolic Euler fluxes as well as for the parabolic diffusion terms and are integrated using nodal elements. This explicit approximation leads to a compact discretization for which data is only needed from the direct adjacent grid cells.

Compared to the state of the art, the nodal type integration allows for a reduction of interpolation and flux evaluation points and thus increases efficiency.

Furthermore, it is possible to run the scheme on parallel computers very efficiently. Therefore, it is an ideal building block towards high fidelity simulation of turbulent flow.

Acknowledgements. We gratefully acknowledge funding of this work by the target research project ADIGMA within the 6^{th} European Research Framework Programme.

References

1. Arnold, D.N., Brezzi, F., Cockburn, B., Marini, L.D.: Unified analysis of discontinuous Galerkin methods for elliptic problems. SIAM J. Numer. Anal. 39(5), 1749–1779 (2002)
2. Bassi, F., Rebay, S.: High-order accurate discontinuous finite element solution of the 2D Euler equations. J. Comput. Phys. 138(2), 251–285 (1997)
3. Bassi, F., Rebay, S.: Numerical evaluation of two discontinuous Galerkin methods for the compressible Navier–Stokes equations. International Journal for Numerical Methods in Fluids 40, 197–207 (2002)
4. Dumbser, M., Munz, C.-D.: Arbitrary High Order Discontinuous Galerkin Schemes. In: Cordier, S., Goudon, T., Gutnic, M., Sonnendrücker, E. (eds.) Numerical Methods for Hyperbolic and Kinetic Problems. IRMA Series in Mathematics and Theoretical Physics, pp. 295–333. EMS Publishing House (2005)
5. Dumbser, M., Munz, C.-D.: Building blocks for arbitrary high order discontinuous Galerkin schemes. Journal of Scientific Computing 27(1-3), 215–230 (2006)
6. Gassner, G.: Discontinuous Galerkin Methods for the Unsteady Compressible Navier–Stokes Equations. Dissertation, University of Stuttgart (2009)
7. Gassner, G., Lörcher, F., Dumbser, M., Munz, C.-D.: Explicit Space-Time Discontinuous Galerkin Schemes for Advection Diffusion Equations. In: 35th CFD/ADIGMA course on very high order discretization methods, VKI LS 2008-08 (2008)
8. Gassner, G., Lörcher, F., Munz, C.-D.: A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes. J. Comput. Phys. 224(2), 1049–1063 (2007)

9. Gassner, G., Lörcher, F., Munz, C.-D.: A discontinuous Galerkin scheme based on a space-time expansion II. Viscous flow equations in multi dimensions. J. Sci. Comput. 34(3), 260–286 (2008)
10. Gassner, G., Lörcher, F., Munz, C.-D., Hesthaven, J.S.: Polymorphic nodal elements and their application in discontinuous Galerkin methods. J. Comput. Phys. 228(5), 1573–1590 (2009)
11. Harten, A., Engquist, B., Osher, S., Chakravarthy, S.: Uniformly high order essentially non-oscillatory schemes, III. J. Comput. Phys. 71, 231–303 (1987)
12. Hartmann, R., Houston, P.: Symmetric interior penalty DG methods for the compressible Navier–Stokes equations I: Method formulation. Int. J. Num. Anal. Model. 3(1), 1–20 (2006)
13. Krivodonova, L., Berger, M.: High-order accurate implementation of solid wall boundary conditions in curved geometries. J. Comput. Phys. 211(2), 492–512 (2006)
14. Lörcher, F.: Predictor-Corrector Discontinuous Galerkin Schemes for the Numerical Solution of the Compressible Navier–Stokes Equations in Complex Domains. Dissertation, University of Stuttgart (2009)
15. Lörcher, F., Gassner, G., Munz, C.-D.: A discontinuous Galerkin scheme based on a space-time expansion I. Inviscid compressible flow in one space dimension. J. Sci. Comput. 32(2), 175–199 (2007)
16. Lörcher, F., Munz, C.-D.: Lax–Wendroff–type schemes of arbitrary order in several space dimensions. IMA J. Num. Anal. 27, 593–615 (2007)
17. Luo, H., Baum, J.D., Lohner, R.: On the computation of steady-state compressible flows using a discontinuous Galerkin method. International Journal for Numerical Methods in Engineering 73(5), 597–623 (2008)
18. Qiu, J., Khoo, B.C., Shu, C.-W.: A numerical study for the performance of the Runge–Kutta discontinuous Galerkin method based on different numerical fluxes. J. Comput. Phys. 212, 510–565 (2006)
19. Titarev, V.A., Toro, E.F.: ADER: Arbitrary high order Godunov approach. Journal of Scientific Computing 17(1-4), 609–618 (2002)
20. Toro, E.F.: Riemann solvers and numerical methods for fluid dynamics, 2nd edn. Springer, Heidelberg (1999)
21. Toro, E.F., Titarev, V.A.: Solution of the generalized Riemann problem for advection-reaction equations. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 458(2018), 271–281 (2002)

This page intentionally left blank

Chapter 6
RKDG with WENO Type Limiters

Jianxian Qiu and Jun Zhu

Abstract. The discontinuous Galerkin (DG) method is a spatial discretization procedure for convection dominated equations, which employs useful features from high resolution finite volume schemes, such as the exact or approximate Riemann solvers serving as numerical fluxes and limiters, which is termed as RKDG when TVD Runge-Kutta method is applied for time discretization. It has the advantage of flexibility in handling complicated geometry, *h*-*p* adaptivity, and efficiency of parallel implementation and has been used successfully in many applications. However, the limiters used to control spurious oscillations in the presence of strong shocks are less robust than the strategies of essentially non-oscillatory (ENO) and weighted ENO (WENO) finite volume and finite difference methods. In this chapter, we will describe the procedure of using WENO and Hermite WENO finite volume methodology as limiters for RKDG methods on unstructure meshes, with the goal of obtaining a robust and high order limiting procedure to simultaneously obtain uniform high order accuracy and sharp, non-oscillatory shock transition for RKDG methods.

1 Introduction

The discontinuous Galerkin (DG) method for solving hyperbolic conservation laws and its extension to time-dependent convection dominated equations [8, 7, 6, 4, 9, 1, 10, 11] are high order finite element methods employing the useful features from high resolution finite volume schemes, such as the exact or approximate Riemann solvers serving as numerical fluxes, and total variation bounded (TVB) limiters [20].

Jianxian Qiu · Jun Zhu
Department of Mathematics, Nanjing University, Nanjing, Jiangsu, China 210093
e-mail: jxqiu@nju.edu.cn

Jun Zhu
College of Science, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China 210016
e-mail: zhujun@nuaa.edu.cn

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 67–80, 2010.
springerlink.com

The DG method is a spatial discretization procedure for convection dominated equations, which is termed as RKDG when TVD Runge-Kutta method is applied for time discretization. DG method has the advantage of typical finite element methods in an easy handling of complicated geometry, arbitrary triangulations, and also the added advantage due to the discontinuous nature of the solution and the test function space, in an explicit time marching, local communications hence high efficiency in parallel implementation [2], and easy *h-p* adaptivity. For these reasons, they have been widely used in applications, see for example the the survey paper [5], and other papers in that Springer volume, which contains the conference proceedings of the First International Symposium on Discontinuous Galerkin Methods held at Newport, Rhode Island in 1999. The review paper [11] is a good reference for many details.

An important component of RKDG methods for solving hyperbolic conservation laws (1):

$$\begin{cases} u_t + f(u)_x + g(u)_y = 0 \\ u(x,y,0) = u_0(x,y) \end{cases} \tag{1}$$

with strong shocks in the solutions is a nonlinear limiter, which is applied to detect discontinuities and control spurious oscillations near such discontinuities. Many such limiters have been used in the literature on RKDG methods. For example, we mention the *minmod* type TVB limiter [8, 7, 6, 4, 9], which is a slope limiter using a technique borrowed from the finite volume methodology; the moment based limiter [2] and an improved moment limiter [3], which are specifically designed for discontinuous Galerkin methods and work on the moments of the numerical solution. These limiters tend to degrade accuracy when mistakenly used in smooth regions of the solution. However, the limiters used to control spurious oscillations in the presence of strong shocks are less robust than the strategies of essentially non-oscillatory (ENO) and weighted ENO (WENO) finite volume and finite difference methods. In [16], Qiu and Shu initiated a study of using the WENO methodology as limiters for RKDG methods on structured mesh. The following framework has been adopted:

Step 1: First, identify the "troubled cells", namely those cells which might need the limiting procedure.

Step 2: Then, replace the solution polynomials in those troubled cells by reconstructed polynomials using the WENO methodology which maintain the original cell averages (conservation), have the same orders of accuracy as before, but are less oscillatory.

This technique works quite well in one and two-dimensional test problems in [16] and in the followup work [15] and [17] where the more compact Hermite WENO methodology was used in the troubled cells.

In this chapter based on [23, 24, 25] which were partially supported by the European project ADIGMA on the development of innovative solution algorithms for aerodynamic simulations, we will describe the procedure of using WENO and Hermite WENO finite volume methodology as limiters for RKDG methods on unstructure meshes, with the goal of obtaining a robust and high order limiting procedure to

simultaneously obtain uniform high order accuracy and sharp, non-oscillatory shock transition for RKDG methods.

2 Description of RKDG Methods with WENO Type Limiter

In this section we give the details of the procedure using the WENO and Hermite WENO reconstruction as a limiter for the RKDG method.

2.1 Description of RKDG Methods

Given a triangulation consisting of cells $\triangle_j$, $P^k(\triangle_j)$ denotes the set of polynomials of degree at most k defined on $\triangle_j$. Here k could actually change from cell to cell, but for simplicity we assume it is a constant over the whole triangulation. In the DG method, the solution as well as the test function space is given by $V_h^k = \{v(x,y) : v(x,y)|_{\triangle_j} \in P^k(\triangle_j)\}$. We emphasize that the procedure described below does not depend on the specific basis chosen for the polynomials. We adopt a local orthogonal basis over a target cell, such as $\triangle_0$: $\{v_l^{(0)}(x,y), l = 0,\ldots,K; K = (k+1)(k+2)/2-1\}$:

$$
\begin{aligned}
v_0^{(0)}(x,y) &= 1,\\
v_1^{(0)}(x,y) &= \frac{x-x_0}{\sqrt{|\triangle_0|}},\\
v_2^{(0)}(x,y) &= a_{21}\frac{x-x_0}{\sqrt{|\triangle_0|}} + \frac{y-y_0}{\sqrt{|\triangle_0|}} + a_{22},\\
v_3^{(0)}(x,y) &= \frac{(x-x_0)^2}{|\triangle_0|} + a_{31}\frac{x-x_0}{\sqrt{|\triangle_0|}} + a_{32}\frac{y-y_0}{\sqrt{|\triangle_0|}} + a_{33},\\
v_4^{(0)}(x,y) &= a_{41}\frac{(x-x_0)^2}{|\triangle_0|} + \frac{(x-x_0)(y-y_0)}{|\triangle_0|} + a_{42}\frac{x-x_0}{\sqrt{|\triangle_0|}} + a_{43}\frac{y-y_0}{\sqrt{|\triangle_0|}} + a_{44},\\
v_5^{(0)}(x,y) &= a_{51}\frac{(x-x_0)^2}{|\triangle_0|} + a_{52}\frac{(x-x_0)(y-y_0)}{|\triangle_0|} + \frac{(y-y_0)^2}{|\triangle_0|} + a_{53}\frac{x-x_0}{\sqrt{|\triangle_0|}}\\
&\quad + a_{54}\frac{y-y_0}{\sqrt{\triangle_0}} + a_{55},\\
&\quad \cdots
\end{aligned}
$$

where (x_0,y_0) and $|\triangle_0|$ are the barycenter and the area of the target cell $\triangle_0$, respectively. Then we would need to solve a linear system to obtain the values of $a_{\ell m}$ by the orthogonality property:

$$\int_{\triangle_0} v_i^{(0)}(x,y)\, v_j^{(0)}(x,y)\, dxdy = w_i \delta_{ij} \tag{2}$$

with $w_i = \int_{\triangle_0} \left(v_i^{(0)}(x,y)\right)^2 dxdy$.

The numerical solution $u^h(x,y,t)$ in the space V_h^k can be written as:

$$u^h(x,y,t) = \sum_{l=0}^{K} u_0^{(l)}(t)\, v_l^{(0)}(x,y), \qquad (x,y) \in \triangle_0,$$

and the degrees of freedom $u_0^{(l)}(t)$ are the moments defined by

$$u_0^{(l)}(t) = \frac{1}{w_l} \int_{\triangle_0} u^h(x,y,t)\, v_l^{(0)}(x,y) dxdy, \qquad l = 0, \cdots, K.$$

In order to determine the approximate solution, we evolve the degrees of freedom $u_0^{(l)}(t)$:

$$\begin{aligned} \frac{d}{dt} u_0^{(l)}(t) = \frac{1}{w_l} \Bigg(& \int_{\triangle_0} \left(F(u^h(x,y,t)) \frac{\partial}{\partial x} v_l^{(0)}(x,y) + g(u^h(x,y,t)) \frac{\partial}{\partial y} v_l^{(0)}(x,y) \right) dxdy \\ & - \int_{\partial \triangle_0} (f(u^h(x,y,t)), g(u^h(x,y,t)))^T \cdot n\, v_l^{(0)}(x,y)\, ds \Bigg), l = 0, \ldots, K. \quad (3) \end{aligned}$$

where n is the outward unit normal of the triangle boundary $\partial \triangle_0$.

In (3) the integral terms can be computed either exactly or by suitable numerical quadratures which are exact for polynomials of degree up to $2k$ for the element integral and up to $2k+1$ for the edge integral. In this paper, we use A_G Gaussian points ($A_G = 6$ for $k = 1$ and $A_G = 7$ for $k = 2$) for the element quadrature and E_G Gaussian points ($E_G = 2$ for $k = 1$ and $E_G = 3$ for $k = 2$) for the edge quadrature:

$$\begin{aligned} & \int_{\triangle_0} \left(f(u^h(x,y,t)) \frac{\partial}{\partial x} v_l^{(0)}(x,y) + g(u^h(x,y,t)) \frac{\partial}{\partial y} v_l^{(0)}(x,y) \right) dxdy \quad (4) \\ & \approx |\triangle_0| \sum_G \sigma_G \left(f(u^h(x_G,y_G,t)) \frac{\partial}{\partial x} v_l^{(0)}(x_G,y_G) + g(u^h(x_G,y_G,t)) \frac{\partial}{\partial y} v_l^{(0)}(x_G,y_G) \right) \end{aligned}$$

$$\begin{aligned} & \int_{\partial \triangle_0} (f(u^h(x,y,t)), g(u^h(x,y,t)))^T \cdot n\, v_l^{(0)}(x,y)\, ds \quad (5) \\ & \approx |\partial \triangle_0| \sum_G \bar{\sigma}_G \left(f(u^h(\bar{x}_G,\bar{y}_G,t)), g(u^h(\bar{x}_G,\bar{y}_G,t)) \right)^T \cdot n\, v_l^{(0)}(\bar{x}_G,\bar{y}_G) \end{aligned}$$

where $(x_G, y_G) \in \triangle_0$ and $(\bar{x}_G, \bar{y}_G) \in \partial \triangle_0$ are the Gaussian quadrature points, and σ_G and $\bar{\sigma}_G$ are the Gaussian quadrature weights. Since the edge integral is on element boundaries where the numerical solution is discontinuous, the flux $(f(u^h(x,y,t)), g(u^h(x,y,t)))^T \cdot n = F(u)\dot{n}$ is replaced by a monotone numerical flux. The simple Lax-Friedrichs flux:

$$F(u(G_k,t)) \cdot n \approx \frac{1}{2} \left[\left(F(u^-(G_k,t)) + F(u^+(G_k,t)) \right) \cdot n - \alpha \left(u^+(G_k,t) - u^-(G_k,t) \right) \right]$$

is used in all of our numerical tests. where α is taken as an upper bound for $|F'(u)\cdot n|$ in the scalar case, or the absolute value of eigenvalues of the Jacobian in the n direction for the system case, and u^- and u^+ are the values of u inside the cell $\triangle_0$ and outside the cell $\triangle_0$ (inside the neighboring cell) at the Gaussian point G_k. The semi-discrete scheme (3) is discretized in time by a non-linear stable Runge-Kutta time discretization, e.g. the third-order version

$$\begin{aligned} u^{(1)} &= u^n + \Delta t L(u^n), \\ u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t L(u^{(1)}), \\ u^{n+1} &= \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t L(u^{(2)}). \end{aligned} \tag{6}$$

The method described above can compute solutions to (1), which are either smooth or have weak shocks and other discontinuities, without further modification. If the discontinuities are strong, however, the scheme will generate significant oscillations and even nonlinear instability. To avoid such difficulties, we borrow the technique of a slope limiter from the finite volume methodology and use it after each Runge-Kutta inner stage (or after the complete Runge-Kutta time step) to control the numerical solution.

2.2 Troubled Cell Indicators

In the first step of limiter procedure, we will use the TVB type limiter adopted in [9] only to detect "troubled cells". The main procedure is as follows. We use (x_{m_ℓ}, y_{m_ℓ}), $\ell = 1,2,3$, to denote the midpoints of the edges on the boundary of the target cell $\triangle_0$, and (x_{b_i}, y_{b_i}), $i = 1,2,3$, to denote the barycenters of the neighboring triangles $\triangle_i$, $i = 1,2,3$, as shown in the left of Figure 1.

We then have

$$x_{m_1} - x_{b_0} = \alpha_1(x_{b_1} - x_{b_0}) + \alpha_2(x_{b_3} - x_{b_0}),\ y_{m_1} - y_{b_0} = \alpha_1(y_{b_1} - y_{b_0}) + \alpha_2(y_{b_3} - y_{b_0}) \tag{7}$$

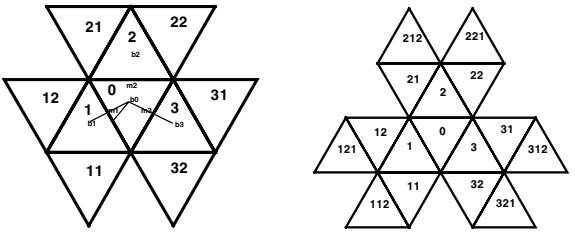


Fig. 1 The limiting diagram (left); The big stencil T for reconstruction (right)

with nonnegative $\alpha_1,\ \alpha_2$, which depend only on (x_{m_1}, y_{m_1}) and the geometry. We then define

$$\tilde{u}^h(x_{m_1}, y_{m_1}, t) \equiv u^h(x_{m_1}, y_{m_1}, t) - u_0^{(0)}(t) \tag{8}$$

$$\triangle u(x_{m_1}, y_{m_1}, t) \equiv \alpha_1(u_1^{(0)}(t) - u_0^{(0)}(t)) + \alpha_2(u_3^{(0)}(t) - u_0^{(0)}(t)) \tag{9}$$

Using the TVB modified *minmod* function [20] defined as

$$\tilde{m}(a_1, a_2) = \begin{cases} a_1 & \text{if } |a_1| \le M|\triangle_0| \\ \begin{cases} s\min(|a_1|, |a_2|) \text{ if } s = sign(a_1) = sign(a_2) \\ 0 \qquad\qquad \text{otherwise} \end{cases} & \text{otherwise} \end{cases} \tag{10}$$

where $M > 0$ is the TVB constant whose choice is problem dependent, we can compute the quantity

$$\tilde{u}^{mod} = \tilde{m}(\tilde{u}^h(x_{m_1}, y_{m_1}, t), \gamma\triangle u(x_{m_1}, y_{m_1}, t)) \tag{11}$$

with $\gamma > 1$ (we take $\gamma = 1.5$ in our numerical tests). If $\tilde{u}^{mod} \neq \tilde{u}^h(x_{m_1}, y_{m_1}, t)$, $\triangle_0$ is marked as a "troubled cell" for further reconstruction. This procedure is repeated for the other two edges of $\triangle_0$ as well. Since the WENO or Hermite WENO reconstruction maintains high order accuracy in the troubled cells, it is less crucial to choose an accurate M. There are several troubled cell indicators which can be used for the purpose, the detail discussion about these troubled cell indicators refers to [18].

2.3 WENO Limiter on Two Dimensional Unstructured Meshes

In the second step of limiter procedure, we reconstruct the polynomial solutions while retaining their cell averages in the troubled cells. In other words, we reconstruct the degrees of freedom $u_0^{(l)}(t),\ l = 1, \ldots, K$ and retain only the cell average $u_0^{(0)}(t)$.

For the $k = 1$ case, we summarize the procedure to reconstruct the first order moments $u_0^{(1)}(t)$ and $u_0^{(2)}(t)$ in the troubled cell $\triangle_0$ using the WENO reconstruction procedure. For simplicity, we relabel the "troubled cell" and its neighboring cells as shown in the right of Figure 1.

Step 1. We select the big stencil as $S = \{\triangle_0, \triangle_1, \triangle_2, \triangle_3, \triangle_{11}, \triangle_{12}, \triangle_{21}, \triangle_{22}, \triangle_{31}, \triangle_{32}\}$. Then we construct a quadratic polynomial $P(x, y)$ to obtain a third order approximation of u by requiring that it has the same cell average as u on the target cell $\triangle_0$, and matches the cell averages of u on the other triangles in the set $S \setminus \{\triangle_0\}$ in a least-square sense, see [12].

Step 2. We divide S into nine smaller stencils:

$$S_1 = \{\triangle_0, \triangle_1, \triangle_2\}, \quad S_2 = \{\triangle_0, \triangle_2, \triangle_3\}, \quad S_3 = \{\triangle_0, \triangle_3, \triangle_1\},$$
$$S_4 = \{\triangle_0, \triangle_1, \triangle_{11}\}, \quad S_5 = \{\triangle_0, \triangle_1, \triangle_{12}\}, \quad S_6 = \{\triangle_0, \triangle_2, \triangle_{21}\},$$
$$S_7 = \{\triangle_0, \triangle_2, \triangle_{22}\}, \quad S_8 = \{\triangle_0, \triangle_3, \triangle_{31}\}, \quad S_9 = \{\triangle_0, \triangle_3, \triangle_{32}\}.$$

We then construct nine linear polynomials $q_i(x,y)$, $i=1,\ldots,9$, satisfying

$$\frac{1}{|\triangle_\ell|}\int_{\triangle_\ell} q_i(x,y)dxdy = \bar{u}_\ell, \qquad \text{for } \triangle_\ell \in S_i. \tag{12}$$

Remark: When some triangles merge in the stencils, we can always use the next layer of triangles to overcome this situation.

Step 3. We find the combination coefficients, also called linear weights, denoted by $\gamma_1^{(l)},...,\gamma_9^{(l)}$, $l=1,2$, satisfying

$$\int_{\triangle_0} P(x,y)v_l^{(0)}(x,y)dxdy = \sum_{i=1}^{9}\gamma_i^{(l)}\int_{\triangle_0} q_i(x,y)v_l^{(0)}(x,y)dxdy, \qquad l=1,2 \tag{13}$$

for the quadratic polynomial $P(x,y)$ defined before. The linear weights are achieved by asking for

$$\min\left(\sum_{i=1}^{9}(\gamma_i^{(l)})^2\right), \qquad l=1,2. \tag{14}$$

By doing so, we can get the linear weights uniquely but can not guarantee their positivity. We use the method introduced in [12, 19] to overcome this difficulty.

Step 4. We compute the smoothness indicators, denote by β_i, $i=1,\ldots,9$, for the smaller stencils S_i, $i=1,\ldots,9$, which measure how smooth the functions $q_i(x,y)$, $i=1,\ldots,9$ are in the target cell $\triangle_0$. The smaller these smoothness indicators, the smoother the functions are in the target cell. We use the same recipe for the smoothness indicators as in [14]:

$$\beta_i = \sum_{|\ell|=1}^{k}|\triangle_0|^{|\ell|-1}\int_{\triangle_0}\left(\frac{\partial^{|\ell|}}{\partial x^{\ell_1}\partial y^{\ell_2}}q_i(x,y)\right)^2 dxdy \tag{15}$$

where $\ell=(\ell_1,\ell_2)$.

Step 5. We compute the non-linear weights based on the smoothness indicators:

$$\omega_i = \frac{\bar{\omega}_i}{\sum_{\ell=1}^{9}\bar{\omega}_\ell}, \qquad \bar{\omega}_\ell = \frac{\gamma_\ell}{(\varepsilon+\beta_\ell)^2}. \tag{16}$$

Here ε is a small positive number to avoid the denominator to become zero. We take $\varepsilon = 10^{-6}$ in our computation.

The moments of the reconstructed polynomial are then given by:

$$u_0^{(l)}(t) = \frac{1}{\int_{\triangle_0}(v_l^{(0)}(x,y))^2\,dxdy}\sum_{i=1}^{9}\omega_i^{(l)}\int_{\triangle_0} q_i(x,y)\,v_l^{(0)}(x,y)dxdy, \qquad l=1,2. \tag{17}$$

For the $k=2$ case, the procedure to reconstruct the first and second order moments $u_0^{(1)}(t)$, $u_0^{(2)}(t)$, $u_0^{(3)}(t)$, $u_0^{(4)}(t)$ and $u_0^{(5)}(t)$ in the troubled cell $\triangle_0$ is analogous to that

for the $k = 1$ case. The troubled cell and its neighboring cells are shown in the right of Figure 1.

Step 1. We select the big stencil as $T = \{\triangle_0, \triangle_1, \triangle_2, \triangle_3, \triangle_{11}, \triangle_{12}, \triangle_{21}, \triangle_{22}, \triangle_{31}, \triangle_{32}, \triangle_{112}, \triangle_{121}, \triangle_{212}, \triangle_{221}, \triangle_{312}, \triangle_{321}\}$. Then we construct a fourth degree polynomial $Q(x,y)$ to obtain a fifth order approximation of u by requiring that it has the same cell average as u on the target cell $\triangle_0$ and matches the cell averages of u on the other triangles in the set $T \setminus \{\triangle_0\}$ in a least-square sense.

Step 2. We divide T into nine smaller stencils:

$$
\begin{aligned}
&T_1 = \{\triangle_0, \triangle_1, \triangle_{11}, \triangle_{12}, \triangle_3, \triangle_{32}\}, && T_2 = \{\triangle_0, \triangle_1, \triangle_{11}, \triangle_{12}, \triangle_2, \triangle_{21}\},\\
&T_3 = \{\triangle_0, \triangle_2, \triangle_{21}, \triangle_{22}, \triangle_1, \triangle_{12}\}, && T_4 = \{\triangle_0, \triangle_2, \triangle_{21}, \triangle_{22}, \triangle_3, \triangle_{31}\},\\
&T_5 = \{\triangle_0, \triangle_3, \triangle_{31}, \triangle_{32}, \triangle_2, \triangle_{22}\}, && T_6 = \{\triangle_0, \triangle_3, \triangle_{31}, \triangle_{32}, \triangle_1, \triangle_{11}\},\\
&T_7 = \{\triangle_0, \triangle_1, \triangle_{11}, \triangle_{12}, \triangle_{112}, \triangle_{121}\}, && T_8 = \{\triangle_0, \triangle_2, \triangle_{21}, \triangle_{22}, \triangle_{212}, \triangle_{221}\},\\
&T_9 = \{\triangle_0, \triangle_3, \triangle_{31}, \triangle_{32}, \triangle_{312}, \triangle_{321}\}.
\end{aligned}
$$

We can then construct quadratic polynomials $q_i(x,y)$, $i = 1, \ldots, 9$, which satisfy the following conditions

$$\frac{1}{|\triangle_\ell|}\int_{\triangle_\ell} q_i(x,y)dxdy = \bar{u}_\ell, \qquad \text{for } \triangle_\ell \in T_i. \tag{18}$$

The remaining steps 3, 4 and 5 are the similar to those for the $k = 1$ case. Finally, the moments $u_0^{(l)}(t), l = 1,2,3,4,5$ of the reconstructed polynomial are given by:

$$u_0^{(l)}(t) = \frac{1}{\int_{\triangle_0}(v_l^{(0)}(x,y))^2 dxdy}\sum_{i=1}^{9}\omega_i^{(l)}\int_{\triangle_0} q_i(x,y)v_l^{(0)}(x,y)dxdy. \tag{19}$$

2.4 Hermite WENO Limiter on Two Dimensional Unstructured Meshes

We summarize the HWENO reconstruction procedure to reconstruct the first order moments $u_0^{(1)}(t)$ and $u_0^{(2)}(t)$ in the troubled cell $\triangle_0$ for the case $k = 1$.

Step 1. We select the big stencil as $S = \{\triangle_0, \triangle_1, \triangle_2, \triangle_3\}$. Then we construct polynomial $P(x,y)$ to approximate u by requiring that it has the same cell average as $u^{(0)}(t)$ on the target cell $\triangle_0$, and matches the cell averages of $u^{(0)}(t)$, $u^{(1)}(t)$ or $u^{(2)}(t)$ on the other triangles in the set $S \setminus \{\triangle_0\}$ in a least square sense.

Step 2. We then construct six linear polynomials $q_i(x,y)$, $i = 1, \ldots, 6$, satisfying:

$$\frac{1}{|\triangle_\ell|}\int_{\triangle_\ell} q_i(x,y)dxdy = u_\ell^{(0)}, \tag{20}$$

$$\frac{1}{\int_{\triangle_{\ell_x}}(v_1^{(\ell_x)}(x,y))^2 dxdy}\int_{\triangle_{\ell_x}} q_i(x,y)v_1^{(\ell_x)}(x,y)dxdy = u_{\ell_x}^{(1)}, \tag{21}$$

$$\frac{1}{\int_{\triangle_{\ell_y}}(v_2^{(\ell_y)}(x,y))^2dxdy}\int_{\triangle_{\ell_y}} q_i(x,y)v_2^{(\ell_y)}(x,y)dxdy = u_{\ell_y}^{(2)}. \tag{22}$$

For

$$i=1,\ell=0,1,2; i=2,\ell=0,2,3; i=3,\ell=0,3,1; i=4,\ell=0,\ell_x=1,\ell_y=1;$$
$$i=5,\ell=0,\ell_x=2,\ell_y=2; i=6,\ell=0,\ell_x=3,\ell_y=3.$$

The remaining steps 3, 4 and 5 are the similar to those of $k=1$ case in the procedure WENO reconstruction.

The moments of the reconstructed polynomial are then given by:

$$u_0^{(l)}(t) = \frac{1}{\int_{\triangle_0}(v_l^{(0)}(x,y))^2\,dxdy}\sum_{i=1}^{6}\omega_i^{(l)}\int_{\triangle_0} q_i(x,y)\,v_l^{(0)}(x,y)dxdy, \qquad l=1,2. \tag{23}$$

For the $k=2$ case, the procedure to reconstruct the first and second order moments $u_0^{(1)}(t), u_0^{(2)}(t), u_0^{(3)}(t), u_0^{(4)}(t)$ and $u_0^{(5)}(t)$ in the troubled cell $\triangle_0$ is analogous to that for the $k=1$ case. The troubled cell and its neighboring cells are shown in Figure 1.

Step 1. We select the big stencil as $S=\{\triangle_0,\triangle_1,\triangle_2,\triangle_3,\ \triangle_{11},\triangle_{12},\triangle_{21},\ \triangle_{22},\ \triangle_{31},\triangle_{32}\}$. Then we construct polynomial $Q(x,y)$ to approximate u by requiring that it has the same cell average as $u^{(0)}$ on the target cell $\triangle_0$ and matches the cell averages of $u^{(0)}$, $u^{(1)}$ or $u^{(2)}$ on the other triangles in the set $S\setminus\{\triangle_0\}$ in a least square sense.

Step 2. We can then construct quadratic polynomials $q_i(x,y)$, $i=1,\ldots,9$, which satisfy the following conditions:

$$\frac{1}{|\triangle_\ell|}\int_{\triangle_\ell} q_i(x,y)dxdy = u_\ell^{(0)}, \tag{24}$$

$$\frac{1}{\int_{\triangle_{\ell_x}}(v_1^{(\ell_x)}(x,y))^2dxdy}\int_{\triangle_{\ell_x}} q_i(x,y)v_1^{(\ell_x)}(x,y)dxdy = u_{\ell_x}^{(1)}, \tag{25}$$

$$\frac{1}{\int_{\triangle_{\ell_y}}(v_2^{(\ell_y)}(x,y))^2dxdy}\int_{\triangle_{\ell_y}} q_i(x,y)v_2^{(\ell_y)}(x,y)dxdy = u_{\ell_y}^{(2)}. \tag{26}$$

For

$$i=1,\ell=0,1,11,12,3,32; i=2,\ell=0,1,11,12,2,21; i=3,\ell=0,2,21,22,1,12;$$
$$i=4,\ell=0,2,21,22,3,31; i=5,\ell=0,3,31,32,2,22; i=6,\ell=0,3,31,32,1,11;$$
$$i=7,\ell=0,1,11,12,\ell_x=1,\ell_y=1; i=8,\ell=0,2,21,22,\ell_x=2,\ell_y=2;$$
$$i=9,\ell=0,3,31,32,\ell_x=3,\ell_y=3.$$

The remaining steps 3, 4 and 5 are the similar to those of $k = 1$ case in the procedure WENO reconstruction. Finally, the moments $u_0^{(l)}(t), l = 1,2,3,4,5$ of the reconstructed polynomial are given by:

$$u_0^{(l)}(t) = \frac{1}{\int_{\triangle_0}(v_l^{(0)}(x,y))^2 dxdy} \sum_{i=1}^{9} \omega_i^{(l)} \int_{\triangle_0} q_i(x,y) v_l^{(0)}(x,y) dxdy. \tag{27}$$

3 Numerical Results

In this section we provide numerical results to demonstrate the performance of the WENO and HWENO reconstruction limiters for the RKDG methods on unstructured meshes described in section 2.

We consider Mandatory Test Case 2 (MTC 2) and MTC 3 in ADIGMA project. The MTC 2 and MTC 3 are Euler and Navier-Stokes transonic flow past a single NACA0012 airfoil configuration with $M_\infty = 0.8$, angle of attack $\alpha = 1.25^\circ$ and Mach number $M_\infty = 0.5$, angle of attack $\alpha = 2^\circ$, Reynolds number $Re = 5000$, respectively. For MTC 3, we use local DG method [1, 10, 25] to treat the diffusion term. The mesh used in the computation is Mesh 4 in Unstructured MTC-Grids-v1.2-070212 mesh which was provided by ARA, one of the ADIGMA partners. The second and the third order DG schemes with the WENO and HWENO limiters with the TVB constant $M = 100$ are used in the numerical experiments. All the computations are performed on the DELL-1950, CPU Xeon-3.0GHz with 16GB ram.

The convergence criteria is $||E_{C_l}|| \leq 5.0 \times 10^{-3}, ||E_{C_d}|| \leq 5.0 \times 10^{-4}, ||E_{C_m}|| \leq 5.0 \times 10^{-4}$, for the both test cases, convergence parameter values are shown in Table 1 and 2, respectively. Distributions of mach number and pressure are shown in Figures 2–5, respectively.

Table 1 MTC2 settings and results. RKDG with the WENO and HWENO limiters

limiter	$\mathscr{O}$	c_l	c_d	c_m	CPU time [Sec.][piu]	*#Cell*	*#DOF*
WENO	2	1.50E-1	1.33E-2	5.63E-3	1078 266	2402	7206
WENO	3	1.64E-1	1.19E-2	5.82E-3	2358 582	2402	14412
HWENO	2	1.64E-1	1.42E-2	5.96E-3	1074 265	2402	7206
HWENO	3	1.58E-1	1.80E-2	4.20E-3	2165 535	2402	14412

Table 2 MTC3 settings and results. RKDG with the WENO and HWENO limiters

limiter	$\mathscr{O}$	c_l	c_d	c_m	CPU time [Sec.][piu]	*#Cell*	*#DOF*
WENO	2	7.35E-2	4.60E-2	5.90E-4	6153 1519	2402	7206
WENO	3	4.25E-2	5.55E-2	-4.42E-3	10476 2587	2402	14412
HWENO	2	7.57E-2	4.56E-2	1.09E-3	5996 1480	2402	7206
HWENO	3	4.14E-2	5.53E-2	-4.51E-3	10437 2577	2402	14412

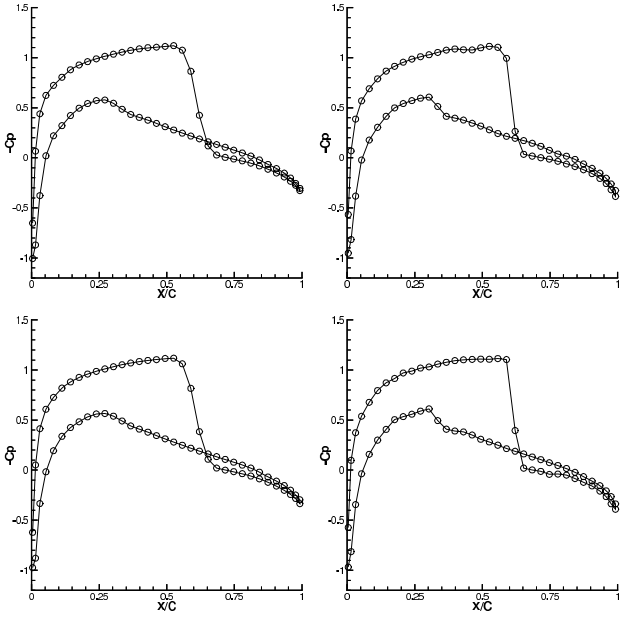


Fig. 2 Transonic Flow around the NACA0012 Airfoil (MTC 2). Pressure distribution. RKDG with the WENO limiter, second order (top left), third order (top right) and RKDG with the HWENO limiter, second order (bottom left), third order (bottom right).

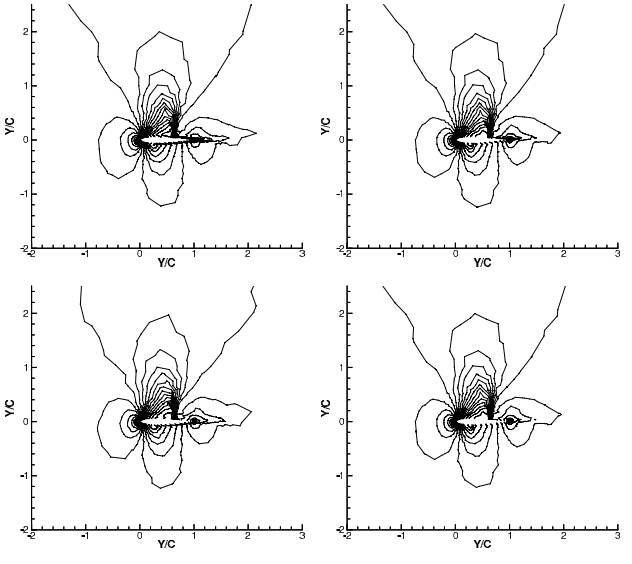


Fig. 3 Transonic Flow around the NACA0012 Airfoil (MTC 2). Thirty equally spaced Mach number contours from 0.172 to 1.325. RKDG with the WENO limiter, second order (top left), third order (top right) and RKDG with the HWENO limiter, second order (bottom left), third order (bottom right).

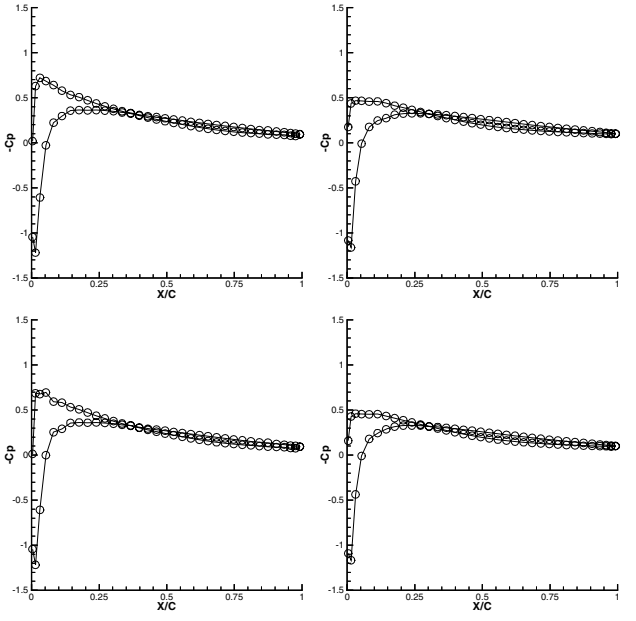


Fig. 4 Transonic Flow around the NACA0012 Airfoil (MTC 3). Pressure distribution. RKDG with the WENO limiter, second order (top left), third order (top right) and RKDG with the HWENO limiter, second order (bottom left), third order (bottom right).

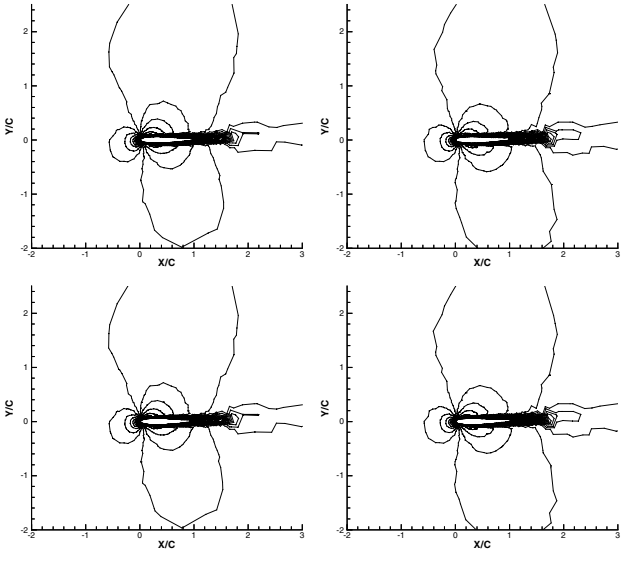


Fig. 5 Transonic Flow around the NACA0012 Airfoil (MTC 3). Thirty equally spaced Mach number contours from 0.020 to 0.629. RKDG with the WENO limiter, second order (top left), third order (top right) and RKDG with the HWENO limiter, second order (bottom left), third order (bottom right).

4 Concluding Remarks

We have developed the limiters for the RKDG methods solving convection-diffusion problems using finite volume high order WENO and HWENO reconstructions on unstructured meshes. The idea is to first identify troubled cells subject to the WENO or HWENO limiting, using a TVB *minmod*-type limiter, then reconstruct the polynomial solution inside the troubled cells by the WENO or HWENO reconstruction using the cell averages of neighboring cells or the cell averages and cell derivative averages of neighboring cells, while maintaining the original cell averages of the troubled cells. Numerical results are provided to show that the method is stable, accurate, and robust in maintaining accuracy.

References

1. Bassi, F., Rebay, S.: A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. J. Comput. Phys. 131, 267–279 (1997)
2. Biswas, B., Devine, K.D., Flaherty, J.: Parallel, adaptive finite element methods for conservation laws. Applied Numer. Math. 14, 255–283 (1994)
3. Burbeau, A., Sagaut, P., Bruneau, C.H.: A problem-independent limiter for high-order Runge-Kutta discontinuous Galerkin methods. J. Comput. Phys. 169, 111–150 (2001)
4. Cockburn, B., Hou, S., Shu, C.-W.: The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case. Math. Comput. 54, 545–581 (1990)
5. Cockburn, B., Karniadakis, G., Shu, C.-W.: The development of discontinuous Galerkin methods. In: Cockburn, B., Karniadakis, G., Shu, C.-W. (eds.) Discontinuous Galerkin Methods: Theory, Computation and Applications. Part I. Lecture Notes in Computational Science and Engineering, vol. 11, pp. 3–50. Springer, Heidelberg (2000) (Overview)
6. Cockburn, B., Lin, S.-Y., Shu, C.-W.: TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one dimensional systems. J. Comput. Phys. 84, 90–113 (1989)
7. Cockburn, B., Shu, C.-W.: TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework. Math. Comput. 52, 411–435 (1989)
8. Cockburn, B., Shu, C.-W.: The Runge-Kutta local projection P1-discontinuous Galerkin finite element method for scalar conservation laws. Math. Model. Numer. Anal. 25, 337–361 (1991)
9. Cockburn, B., Shu, C.-W.: The Runge-Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems. J. Comput. Phys. 141, 199–224 (1998)
10. Cockburn, B., Shu, C.-W.: The local discontinuous Galerkin method for time-dependent convection diffusion systems. SIAM J. Numer. Anal. 35, 2440–2463 (1998)
11. Cockburn, B., Shu, C.-W.: Runge-Kutta discontinuous Galerkin methods for convection-dominated problems. J. Sci. Comput. 16, 173–261 (2001)
12. Hu, C., Shu, C.-W.: Weighted essentially non-oscillatory schemes on triangular meshes, J. Comput. Phys. 150, 97–127 (1999)
13. Luo, H., Baum, J.D., Lohner, R.: A Hermite WENO-based limiter for discontinuous Galerkin method on unstructured grids. J. Comput. Phys. 225, 686–713 (2007)

14. Jiang, G., Shu, C.-W.: Efficient implementation of weighted ENO schemes. J. Comput. Phys. 126, 202–228 (1996)
15. Qiu, J., Shu, C.-W.: Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method: one dimensional case. J. Comput. Phys. 193, 115–135 (2004)
16. Qiu, J., Shu, C.-W.: Runge-Kutta discontinuous Galerkin method using WENO limiters. SIAM J. Scient. Comput. 26, 907–929 (2005)
17. Qiu, J., Shu, C.-W.: Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method II: two dimensional case. Computers Fluids 34, 642–663 (2005)
18. Qiu, J., Shu, C.-W.: A comparison of trouble cell indicators for Runge-Kutta discontinuous Galerkin method using WENO limiters. SIAM J. Scient. Comput. 27, 995–1013 (2005)
19. Shi, J., Hu, C., Shu, C.-W.: A technique of treating negative weights in WENO schemes. J. Comput. Phys. 175, 108–127 (2002)
20. Shu, C.-W.: TVB uniformly high-order schemes for conservation laws. Math. Comput. 49, 105–121 (1987)
21. Shu, C.-W., Osher, S.: Efficient implementation of essentially non-oscillatory shock-capturing schemes. J. Comput. Phys. 77, 439–471 (1988)
22. Woodward, P., Colella, P.: The numerical simulation of two-dimensional fluid flow with strong shocks. J. Comput. Phys. 54, 115–173 (1984)
23. Zhu, J., Qiu, J., Shu, C.-W., Dumbser, M.: Runge-Kutta discontinuous Galerkin method using WENO limiters II: unstructured meshes. J. Comput. Phys. 227, 4330–4353 (2008)
24. Zhu, J., Qiu, J.: Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method III: Unstructured meshes. J. Scient. Comput. 39, 293–321 (2009)
25. Zhu, J., Qiu, J.: Local DG method using WENO type limiters for convection-diffusion problems (preprint)

Chapter 7
IPG Discretizations of the Compressible Navier-Stokes Equations

Vít Dolejší, Martin Holík, and Jiří Hozman

Abstract. We deal with the numerical solution of the system of the compressible (laminar) Navier-Stokes equations with the aid of the discontinuous Galerkin method. Particularly, we employ the so-called interior penalty Galerkin (IPG) discretizations, namely symmetric, non-symmetric and incomplete variants. We described the space semi-discretization and discuss the choice of stabilization terms. Then a set of numerical experiments, demonstrating the stability, convergence and accuracy of the method are presented.

1 Introduction

Our aim is to develop a sufficiently robust, efficient and accurate numerical scheme for the simulation of unsteady compressible flows. The *discontinuous Galerkin method* (DGM), using discontinuous piecewise polynomial approximation, perfectly suits for this task. Among several variants of DGM we prefer the so-called *interior penalty Galerkin* (IPG) discretizations. This approach is based on the primal formulation where any auxiliary variable are not introduced. The inter-element continuity is replaced by the interior penalty terms. We deal with three variants of IPG, namely the *symmetric interior penalty Galerkin* (SIPG), *non-symmetric interior penalty Galerkin* (NIPG) and *incomplete interior penalty Galerkin* (IIPG) techniques, which were studied and analysed in many papers for a scalar nonlinear problems, e.g. [2], [16], [5], [10], [9]. A generalization from the scalar case to the system of the Navier-Stokes equations is not straightforward, it requires a little heuristic approach, which is discussed within this chapter.

Vít Dolejší · Martin Holík · Jiří Hozman
Charles University Prague, Faculty of Mathematics and Physics, Sokolovská 83, Prague, Czech Republic
e-mail: dolejsi@karlin.mff.cuni.cz, marthelh@seznam.cz, jhozmi@volny.cz

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 81–93, 2010.
springerlink.com

2 Problem Formulation

2.1 Compressible Flow Problem

Let $\Omega \subset R^d$, $d = 2,3$ be a bounded domain and $T > 0$. We set $Q_T = \Omega \times (0,T)$ and by $\partial\Omega$ denote the boundary of Ω which consists of several disjoint parts. We distinguish inlet $\partial\Omega_i$, outlet $\partial\Omega_o$ and impermeable walls $\partial\Omega_w$, i.e. $\partial\Omega = \partial\Omega_i \cup \partial\Omega_o \cup \partial\Omega_w$. The system of the Navier-Stokes equations describing a motion of viscous compressible fluids can be written in the dimensionless form

$$\frac{\partial \mathbf{w}}{\partial t} + \nabla \cdot \mathsf{f}(\mathbf{w}) = \nabla \cdot \mathsf{R}(\mathbf{w}, \nabla \mathbf{w}) \quad \text{in } Q_T, \tag{1}$$

where $\mathbf{w} = (\rho, \rho v_1, \dots, \rho v_d, e)^{\mathrm{T}}$ is the *state vector*, $\mathsf{f}(\mathbf{w}) = (\mathbf{f}_1(\mathbf{w}), \dots, \mathbf{f}_d(\mathbf{w}))$ with $\mathbf{f}_s(\mathbf{w}) = (\rho v_s, \rho v_s v_1 + \delta_{s1} p, \dots, \rho v_s v_d + \delta_{sd} p, (e+p)v_s)^{\mathrm{T}}$, $s = 1, \dots, d$ are the so-called *inviscid (Euler) fluxes* and $\mathsf{R}(\mathbf{w}) = (\mathbf{R}_1(\mathbf{w}), \dots, \mathbf{R}_d(\mathbf{w}))$ with

$$\mathbf{R}_s(\mathbf{w}, \nabla \mathbf{w}) = \left(0, \tau_{s1}, \dots, \tau_{sd}, \sum_{k=1}^{d} \tau_{sk} v_k + \frac{\gamma}{Re\, Pr} \frac{\partial \theta}{\partial x_s}\right)^{\mathrm{T}}, \quad s = 1, \dots, d \tag{2}$$

are the so-called *viscous fluxes*. Symbols ∇ and $\nabla\cdot$ mean the gradient and divergence operators. We consider the Newtonian type of fluid, i. e., the viscous part of the stress tensor has the form

$$\tau_{sk} = \frac{1}{Re}\left[\left(\frac{\partial v_s}{\partial x_k} + \frac{\partial v_k}{\partial x_s}\right) - \frac{2}{3}\sum_{i=1}^{d} \frac{\partial v_i}{\partial x_i}\delta_{sk}\right], \quad s,k = 1, \dots, d. \tag{3}$$

We use the following notation: ρ – density, p – pressure, e – total energy, $\mathbf{v} = (v_1, \dots, v_d)$ – velocity, θ – temperature, γ – Poisson adiabatic constant, Re – Reynolds number, Pr – Prandtl number.

In order to close the system, we consider the state equation for perfect gas and the definition of the total energy

$$p = (\gamma - 1)(e - \rho|\mathbf{v}|^2/2), \quad e = c_V \rho \theta + \rho|\mathbf{v}|^2/2, \tag{4}$$

where c_V is the specific heat at constant volume which we assume to be equal to one in the dimensionless case. The system (1) – (4) is of *hyperbolic-parabolic* type. It is equipped with a suitable initial and boundary conditions, see, e.g. [13]. We only mention that we prescribe Dirichlet conditions for density and velocity on the inlet, the vanishing velocity on the impermeable walls. Otherwise we prescribe homogeneous Neumann boundary conditions.

The problem to solve the compressible Navier-Stokes equations (1) – (2) with constitutive relations (3) – (4), equipped with the initial and boundary conditions will be denoted by (CFP) (compressible flow problem).

Finally, we present some properties of the inviscid and viscous fluxes $\mathsf{f}(\cdot)$ and $\mathsf{R}(\cdot,\cdot)$, respectively. These properties are fundamental for introducing the linearization of the nonlinear fluxes, which is the base of semi-implicit time discretization schemes.

The inviscid fluxes $\mathbf{f}_s$, $s=1,\ldots,d$ satisfy (see [13, Lemma 3.1]) $\mathbf{f}_s(\mathbf{w}) = \mathsf{A}_s(\mathbf{w})\mathbf{w}$, where $\mathsf{A}_s(\mathbf{w}) \equiv D\mathbf{f}_s(\mathbf{w})/D\mathbf{w}$, are the Jacobi matrices of the mappings $\mathbf{f}_s$. Then, we define a matrix

$$\mathsf{P}(\mathbf{w},\mathbf{n}) \equiv \sum_{s=1}^{d} \mathsf{A}_s(\mathbf{w}) n_s, \tag{5}$$

where $\mathbf{n} = (n_1,\ldots,n_d) \in \boldsymbol{R}^d$, $n_1^2+\ldots+n_d^2 = 1$, which plays a role in the definition of a numerical flux and the choice of boundary conditions.

Furthermore, the viscous terms $\mathbf{R}_s(\mathbf{w},\nabla\mathbf{w})$ can be expressed in the form

$$\mathbf{R}_s(\mathbf{w},\nabla\mathbf{w}) = \sum_{k=1}^{d} \mathsf{K}_{s,k}(\mathbf{w}) \frac{\partial \mathbf{w}}{\partial x_k}, \quad s=1,\ldots,d, \tag{6}$$

where $\mathsf{K}_{s,k}(\cdot)$ are $(d+2)\times(d+2)$ matrices nonlinearly dependent on $\mathbf{w}$, see, e.g., [13] or [8].

3 Discretization

3.1 Triangulations

Let $\mathscr{T}_h$ ($h>0$) be a partition of the domain Ω into a finite number of closed d-dimensional mutually disjoint (convex or non-convex) polyhedra K i.e., $\overline{\Omega} = \bigcup_{K\in\mathscr{T}_h} K$. We call $\mathscr{T}_h = \{K\}_{K\in\mathscr{T}_h}$ a *triangulation* of Ω and do not require the conforming properties from the finite element method. In 2D problems, we choose usually $K\in\mathscr{T}_h$ as triangles or quadrilaterals. In 3D, $K\in\mathscr{T}_h$ can be, e. g., tetrahedra, pyramids or hexahedra, but we can construct even more general elements K. By ∂K we denote the boundary of element $K\in\mathscr{T}_h$ and set $h_K = \mathrm{diam}(K)$, $h = \max_{K\in\mathscr{T}_h} h_K$. Symbol ρ_K is the radius of the largest d-dimensional ball inscribed into K and $|K|$ is the d-dimensional Lebesgue measure of K.

By $\mathscr{F}_h$ we denote the smallest possible set of all open $(d-1)$-dimensional faces (open edges when $d=2$ or open faces when $d=3$) of all elements $K\in\mathscr{T}_h$. Further, $\mathscr{F}_h^I$ means the set of all $\Gamma\in\mathscr{F}_h$ that are contained in Ω (inner faces). Moreover, we denote by $\mathscr{F}_h^w$, $\mathscr{F}_h^i$ and $\mathscr{F}_h^o$ the set of all $\Gamma\in\mathscr{F}_h$ such that $\Gamma\subset\partial\Omega_w$, $\Gamma\subset\partial\Omega_i$ and $\Gamma\subset\partial\Omega_o$, respectively. Furthermore, $\mathscr{F}_h^D$ is the set of all $\Gamma\in\mathscr{F}_h$ where the Dirichlet type of boundary conditions is prescribed at least for one component of $\mathbf{w}$ (i.e., $\mathscr{F}_h^D \equiv \mathscr{F}_h^w \cup \mathscr{F}_h^i$) and by $\mathscr{F}_h^N$ the set of all $\Gamma\in\mathscr{F}_h$ where the Neumann type of boundary conditions is prescribed for all components of $\mathbf{w}$ (i.e., $\mathscr{F}_h^N \equiv \mathscr{F}_h^o$). Obviously, $\mathscr{F}_h = \mathscr{F}_h^I \cup \mathscr{F}_h^D \cup \mathscr{F}_h^N$. For a shorter notation we put $\mathscr{F}_h^{io} \equiv \mathscr{F}_h^i \cup \mathscr{F}_h^o$, $\mathscr{F}_h^{ID} \equiv \mathscr{F}_h^I \cup \mathscr{F}_h^D$ and $\mathscr{F}_h^{DN} \equiv \mathscr{F}_h^D \cup \mathscr{F}_h^N = \mathscr{F}_h^w \cup \mathscr{F}_h^i \cup \mathscr{F}_h^o$.

Finally, for each $\Gamma \in \mathscr{F}_h$ we define a unit normal vector $\mathbf{n}_\Gamma$. We assume that $\mathbf{n}_\Gamma$, $\Gamma \in \mathscr{F}_h^{DN}$ has the same orientation as the outer normal of $\partial\Omega$. For $\mathbf{n}_\Gamma$, $\Gamma \in \mathscr{F}_h^I$ the orientation is arbitrary but fixed for each edge.

3.2 Discontinuous Finite Element Spaces

To each $K \in \mathscr{T}_h$, we assign a positive integer p_K (local polynomial degree) . Then we define the vector $\mathsf{p} \equiv \{p_K, K \in \mathscr{T}_h\}$.

Over the triangulation $\mathscr{T}_h$, we define the finite dimensional space of discontinuous piecewise polynomial functions associated with the vector p by

$$S_{h\mathsf{p}} \equiv \{v;\ v \in L^2(\Omega),\ v|_K \in P_{p_K}(K)\ \forall K \in \mathscr{T}_h\}, \tag{7}$$

where $P_{p_K}(K)$ denotes the space of all polynomials on K of degree $\leq p_K$, $K \in \mathscr{T}_h$. We seek the approximate solution in the space of vector-valued functions

$$\mathbf{S}_{h\mathsf{p}} \equiv S_{h\mathsf{p}} \times \ldots \times S_{h\mathsf{p}} \qquad (d+2 \text{ times}). \tag{8}$$

For each $\Gamma \in \mathscr{F}_h^I$ we denote by symbols $\langle v \rangle_\Gamma$ and $[v]_\Gamma$ the mean value and the jump with respect to $\mathbf{n}_\Gamma$ of $v \in S_{h\mathsf{p}}$. The value $[v]_\Gamma$ depends on the orientation of $\mathbf{n}_\Gamma$ of course but the value $[v]_\Gamma \mathbf{n}_\Gamma$ does not. For $\Gamma \in \mathscr{F}_h^{DN}$ we introduce the notation $\langle v \rangle_\Gamma = [v]_\Gamma = v|_\Gamma$.

In case that $[\cdot]_\Gamma$ and $\langle \cdot \rangle_\Gamma$ are arguments of $\int_\Gamma \ldots \mathrm{d}S$, $\Gamma \in \mathscr{F}_h$ we omit the subscript Γ and write simply $[\cdot]$ and $\langle \cdot \rangle$, respectively.

4 System of the Navier-Stokes Equations

Within this section, we apply the discontinuous Galerkin finite element method to the system of the compressible Navier-Stokes equations.

The crucial item of the IPG formulation of (CFP) is the treatment of the viscous terms. Let $\mathbf{w} \in \mathbf{S}_{h\mathsf{p}}$, then multiplying the viscous term $\nabla \cdot \mathsf{R}(\mathbf{w}, \nabla\mathbf{w})$ from (1) by $\varphi \in \mathbf{S}_{h\mathsf{p}}$, integrating over $K \in \mathscr{T}_h$, summing over all $K \in \mathscr{T}_h$ and using the boundary conditionsand (6), we obtain

$$\sum_{\Gamma \in \mathscr{F}_h^{ID}} \int_\Gamma \sum_{s=1}^d \langle \mathbf{R}_s(\mathbf{w}, \nabla\mathbf{w}) \rangle\, n_s \cdot [\varphi]\, \mathrm{d}S - \sum_{K \in \mathscr{T}_h} \int_K \sum_{s=1}^d \mathbf{R}_s(\mathbf{w}, \nabla\mathbf{w}) \cdot \frac{\partial \varphi}{\partial x_s}\, \mathrm{d}x \tag{9}$$

$$= \sum_{\Gamma \in \mathscr{F}_h^{ID}} \int_\Gamma \sum_{s,k=1}^d \left\langle \mathsf{K}_{s,k}(\mathbf{w}) \frac{\partial \mathbf{w}}{\partial x_k} \right\rangle n_s \cdot [\varphi]\, \mathrm{d}S - \sum_{K \in \mathscr{T}_h} \int_K \sum_{s,k=1}^d \left(\mathsf{K}_{s,k}(\mathbf{w}) \frac{\partial \mathbf{w}}{\partial x_k} \right) \cdot \frac{\partial \varphi}{\partial x_s}\, \mathrm{d}x$$

In virtue of numerical analysis from [9], [10], we add to this expression a *stabilization term* which we obtain by the formal exchange of arguments $\mathbf{w}$ and φ in the linear part of last term of (9), i.e.,

$$\theta \sum_{\Gamma \in \mathscr{F}_h^{ID}} \int_\Gamma \sum_{s=1}^{d} \left\langle \sum_{k=1}^{d} \mathsf{K}_{s,k}(\mathbf{w}) \frac{\partial \varphi}{\partial x_k} \right\rangle n_s \cdot [\mathbf{w}] \, \mathrm{d}S, \tag{10}$$

where $\theta = -1, 0, 1$ depending on the type of stabilization, i.e., NIPG, IIPG or SIPG variants of DGM. However, numerical experiments indicate that this choice of stabilization is not suitable. It is caused by that fact that for $\varphi = (\varphi_1, 0, \ldots, 0)^{\mathrm{T}}$, $\varphi_1 \in S_{h\mathsf{p}}$, $\varphi_1 \neq const$, we obtain a non-vanishing term (10) whereas both terms in (9) are equal to zero since the first rows of R_s, $K_{s,k}$, $s,k = 1, \ldots, d$, vanish, see (2). Therefore, in virtue of [4], [14], we employ the stabilization term

$$\theta \sum_{\Gamma \in \mathscr{F}_h^{ID}} \int_\Gamma \sum_{s=1}^{d} \left\langle \sum_{k=1}^{d} \mathsf{K}_{s,k}^{\mathrm{T}}(\mathbf{w}) \frac{\partial \varphi}{\partial x_k} \right\rangle n_s \cdot [\mathbf{w}] \, \mathrm{d}S, \tag{11}$$

which avoids the drawback mentioned above. Here, K^{T} denotes the matrix transposed to K.

Therefore, we define for $\mathbf{w}, \varphi \in \mathbf{S}_{h\mathsf{p}}$ the forms

$$(\mathbf{w}, \varphi) = \sum_{K \in \mathscr{T}_h} \int_K \mathbf{w} \cdot \varphi \, \mathrm{d}x, \tag{12}$$

$$\begin{aligned} \tilde{\mathbf{a}}_h(\mathbf{w}, \varphi) = & \sum_{K \in \mathscr{T}_h} \int_K \sum_{s=1}^{d} \left(\sum_{k=1}^{d} \mathsf{K}_{s,k}(\mathbf{w}) \frac{\partial \mathbf{w}}{\partial x_k} \cdot \frac{\partial \varphi}{\partial x_s} \right) \mathrm{d}x \\ & - \sum_{\Gamma \in \mathscr{F}_h^{ID}} \int_\Gamma \sum_{s=1}^{d} \left\langle \sum_{k=1}^{d} \mathsf{K}_{s,k}(\mathbf{w}) \frac{\partial \mathbf{w}}{\partial x_k} \right\rangle n_s \cdot [\varphi] \, \mathrm{d}S \\ & - \theta \sum_{\Gamma \in \mathscr{F}_h^{ID}} \int_\Gamma \sum_{s=1}^{d} \left\langle \sum_{k=1}^{d} \mathsf{K}_{s,k}^{\mathrm{T}}(\mathbf{w}) \frac{\partial \varphi}{\partial x_k} \right\rangle n_s \cdot [\mathbf{w}] \, \mathrm{d}S \\ & + \theta \sum_{\Gamma \in \mathscr{F}_h^{D}} \int_\Gamma \sum_{s=1}^{d} \sum_{k=1}^{d} \mathsf{K}_{s,k}^{\mathrm{T}}(\mathbf{w}) \frac{\partial \varphi}{\partial x_k} n_s \cdot \mathbf{w}_B \, \mathrm{d}S, \end{aligned} \tag{13}$$

$$\tilde{\mathbf{b}}_h(\mathbf{w}, \varphi) = \sum_{K \in \mathscr{T}_h} \left\{ \int_{\partial K} \sum_{s=1}^{d} \mathbf{f}_s(\mathbf{w}) n_s \cdot \varphi \, \mathrm{d}S - \int_K \sum_{s=1}^{d} \mathbf{f}_s(\mathbf{w}) \cdot \frac{\partial \varphi}{\partial x_s} \mathrm{d}x \right\}, \tag{14}$$

$$\tilde{\mathbf{J}}_h^{\sigma}(\mathbf{w}, \varphi) = \sum_{\Gamma \in \mathscr{F}_h^{ID}} \int_\Gamma \sigma [\mathbf{w}] \cdot [\varphi] \, \mathrm{d}S - \sum_{\Gamma \in \mathscr{F}_h^{D}} \int_\Gamma \sigma \mathbf{w}_B \cdot \varphi \, \mathrm{d}S, \tag{15}$$

where the penalty parameter σ is chosen by

$$\sigma|_\Gamma = \frac{C_W}{d(\Gamma) Re}, \quad \Gamma \in \mathscr{F}_h^{ID}, \tag{16}$$

where $d(\Gamma) = \mathrm{diam}(\Gamma)$ and $C_W > 0$ is a suitable constant.

The state vector $\mathbf{w}_B$ prescribed on $\partial\Omega_i \cup \partial\Omega_w$ is given by the boundary conditions, in particular, we have

$$\begin{aligned}\mathbf{w}_B &= (\rho|_{\partial\Omega_w}, 0, \ldots, 0, \rho|_{\partial\Omega_w}\theta|_{\partial\Omega_w}) \text{ on } \partial\Omega_w, \\ \mathbf{w}_B &= (\rho_D, \rho_D(\mathbf{v}_D)_1, \ldots, \rho_D(\mathbf{v}_D)_d, \rho|_{\partial\Omega_i}\theta|_{\partial\Omega_i} + \frac{1}{2}\rho_D|\mathbf{v}_D|^2) \text{ on } \partial\Omega_i,\end{aligned} \tag{17}$$

where $\rho_D, \mathbf{v}_D$ and θ_D are given functions from the boundary conditions and $\rho|_\Gamma$ and $\theta|_\Gamma$ are the values of density and temperature extrapolated from interior of Ω on the appropriate boundary part, respectively. More detailed determination of (13) – (17) is given in [7].

Let $\mathbf{w}(t)$ denotes the function on Ω such that $\mathbf{w}(t)(x) = \mathbf{w}(x,t), x \in \Omega$. Then with the aid of (12) – (15) the IPG formulation for the Navier-Stokes equations reads

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{w}(t), \varphi) + \tilde{\mathbf{a}}_h(\mathbf{w}(t), \varphi) + \tilde{\mathbf{b}}_h(\mathbf{w}(t), \varphi) + \tilde{\mathbf{J}}_h^\sigma(\mathbf{w}(t), \varphi) = 0 \tag{18}$$

for $\mathbf{w}(t), \varphi \in \mathbf{S}_{h\mathsf{p}}$, $t \in (0,T)$.

In order to evaluate the boundary integrals in (14) we use the ("finite volume") approximation

$$\sum_{s=1}^{d} \mathbf{f}_s(\mathbf{w})(\mathbf{n}_{\partial K})_s \cdot \varphi\Big|_{\partial K} \approx \mathbf{H}\left(\mathbf{w}|_\Gamma^{(\mathrm{in})}, \mathbf{w}|_\Gamma^{(\mathrm{out})}, \mathbf{n}_{\partial K}\right) \cdot \varphi\Big|_{\partial K}, \tag{19}$$

where $\mathbf{w}|_\Gamma^{(\mathrm{in})}$, $\mathbf{w}|_\Gamma^{(\mathrm{out})}$ are the traces of $\mathbf{w}$ on ∂K from the interior and the exterior of element $K \in \mathscr{T}_h$, respectively and $\mathbf{H}(\cdot,\cdot,\cdot)$ is a *numerical flux*, for details, see, e.g. [12] or [13]. Then with the aid of (14) and (19) we define the form

$$\bar{\mathbf{b}}_h(\mathbf{w}, \varphi) \equiv \sum_{\Gamma \in \mathscr{F}_h} \int_\Gamma \mathbf{H}\left(\mathbf{w}|_\Gamma^{(p)}, \mathbf{w}|_\Gamma^{(n)}, \mathbf{n}_\Gamma\right) \cdot [\varphi]_\Gamma \,\mathrm{d}S - \sum_{K \in \mathscr{T}_h} \int_K \sum_{s=1}^{d} \mathbf{f}_s(\mathbf{w}) \cdot \frac{\partial \varphi}{\partial x_s} \,\mathrm{d}x, \tag{20}$$

where $\mathbf{w}|_\Gamma^{(p)}$ and $\mathbf{w}|_\Gamma^{(n)}$ are the traces of $\mathbf{w} \in \mathbf{S}_{h\mathsf{p}}$ on Γ from positive and negative orientation of $\mathbf{n}_\Gamma$. In order to employ the concept of semi-implicit schemes we need that the numerical flux $\mathbf{H}$ has a form suitable for a linearization. Hence, in our applications we employ the Vijayasundaram numerical flux, see [17], [12], Section 7.3 or [13], Section 3.3.4. The matrix $\mathsf{P}(\mathbf{w}, \mathbf{n})$ defined by (5) is diagonalizable, i.e., there exist matrices Λ and T such that

$$\mathsf{P}(\mathbf{w}, \mathbf{n}) = T\Lambda T^{-1}, \; \Lambda = \mathrm{diag}\,(\lambda_1, \ldots, \lambda_{d+2}), \tag{21}$$

where $\lambda_1, \ldots, \lambda_{d+2}$ are the eigenvalues of P. We define the "positive" and "negative" part of P by

$$\mathsf{P}^\pm(\mathbf{w}, \mathbf{n}) = T\Lambda^\pm T^{-1}, \; \Lambda^\pm = \mathrm{diag}\,(\lambda_1^\pm, \ldots, \lambda_{d+2}^\pm). \tag{22}$$

Then the Vijayasundaram numerical flux reads

$$\mathbf{H}(\mathbf{w}_1,\mathbf{w}_2,\mathbf{n}) \equiv \mathsf{P}^+\left(\frac{\mathbf{w}_1+\mathbf{w}_2}{2},\mathbf{n}\right)\mathbf{w}_1 + \mathsf{P}^-\left(\frac{\mathbf{w}_1+\mathbf{w}_2}{2},\mathbf{n}\right)\mathbf{w}_2. \tag{23}$$

It is necessary to specify the meaning of $\mathbf{w}|_\Gamma^{(n)}$ for $\Gamma \in \mathscr{F}_h^{DN}$. We use the approach known from the inviscid flow simulation, see, e.g., [13], [18]. For $\Gamma \in \mathscr{F}_h^{io}$ we prescribe m_n components of $\mathbf{w}$ on Γ and extrapolate $m_p = d+2-m_n$ components of $\mathbf{w}$ from K to Γ ($\Gamma \subset \partial K$) where m_n is the number of negative eigenvalues of matrix $\mathsf{P}(\mathbf{w},\mathbf{n})$. Thus, we define $\mathbf{w}|_\Gamma^{(n)} = LRP(\mathbf{w}|_\Gamma^{(p)}, \mathbf{w}_D, \mathbf{n}_\Gamma)$, where $LRP(\cdot,\cdot,\cdot)$ represents a solution of the *local Riemann problem* considered on edge $\Gamma \in \mathscr{F}_h^{io}$ and $\mathbf{w}_D$ is a given state vector (e.g. from far-field boundary conditions). For details, see [11].

For $\Gamma \in \mathscr{F}_h^{w}$, the impermeability condition

$$\mathbf{v}\cdot\mathbf{n} = 0 \tag{24}$$

is prescribed. Then in virtue of (19) we put

$$\int_\Gamma \mathbf{H}(\mathbf{w}(t)|_\Gamma^{(p)}, \mathbf{w}(t)|_\Gamma^{(n)}, \mathbf{n}_\Gamma)\cdot\varphi\,\mathrm{d}S := \int_\Gamma F_W(\mathbf{w}(t),\mathbf{n}_\Gamma)\cdot\varphi\,\mathrm{d}S, \quad \Gamma \in \mathscr{F}_h^W, \tag{25}$$

where $F_W(\mathbf{w},\mathbf{n}) \equiv (0, pn_1, \ldots, pn_d, 0)^{\mathrm{T}}$.

The approximate solution of (CFP) is sought in the space of discontinuous piecewise polynomial functions $\mathbf{S}_{h\mathsf{p}}$ defined by (8). We introduce the *semi-discrete problem*.

Definition 1. Function $\mathbf{w}_h$ is a *semi-discrete solution* of (CFP), if

$$\begin{aligned}
&\text{a)}\quad \mathbf{w}_h \in C^1(0,T;\mathbf{S}_{h\mathsf{p}}), \qquad (26)\\
&\text{b)}\quad \left(\frac{\partial \mathbf{w}_h(t)}{\partial t},\varphi_h\right) + \tilde{\mathbf{a}}_h(\mathbf{w}_h(t),\varphi_h) + \bar{\mathbf{b}}_h(\mathbf{w}_h(t),\varphi_h) + \tilde{\mathbf{J}}_h^\sigma(\mathbf{w}_h(t),\varphi_h) = 0\\
&\qquad\qquad \forall\,\varphi_h \in \mathbf{S}_{h\mathsf{p}},\ \forall\, t\in(0,T),\\
&\text{c)}\quad \mathbf{w}_h(0) = \mathbf{w}_h^0,
\end{aligned}$$

where $\mathbf{w}_h^0 \in \mathbf{S}_{h\mathsf{p}}$ denotes an $\mathbf{S}_{h\mathsf{p}}$-approximation of the initial condition $\mathbf{w}^0$ from initial condition.

Here $C^1(0,T;\mathbf{S}_{h\mathsf{p}})$ is the space of continuously differentiable mappings of the interval $(0,T)$ into $\mathbf{S}_{h\mathsf{p}}$. The problem (26), a) – c) exhibits a system of ordinary differential equations (ODEs) for $\mathbf{w}_h(t)$ which has to be discretized by a suitable method.

We employ the so-called *semi-implicit time discretization* method, which is based on a formal linearization of viscous and inviscid terms, and the linear terms are treated implicitly and the nonlinear ones explicitly. The resulting scheme has a high order of accuracy with respect to the time and space, it is practically unconditionally stable and at each time level we have to solve only one linear algebra system. This approach is described in Section II.2 of this book.

5 Numerical Examples

In this section we numerically study the stability, convergence and accuracy of the proposed IPG methods. We present two steady-state two-dimensional numerical examples. The first one is a basic benchmark of steady viscous flow around a flat plate. The second one represents a flow around NACA 0012 profile.

5.1 Blasius Problem

We consider the laminar flow on the adiabatic flat plate $\{(x_1,x_2);\ 0 \le x_1 \le 1, x_2 = 0\}$ characterised by the freestream Mach number $M = 0.1$ and the Reynolds number $Re = 10^4$. The computation domain is viewed in Figure 1, where two used triangular grids are plotted together with their details around the leading edge. We prescribe the adiabatic boundary conditions at the flat plate, the outflow boundary conditions at $\{(x_1,x_2);\ x_1 = 1, -1.5 \le x_2 \le 1.5\}$ and the inflow boundary conditions on the rest of the boundary.

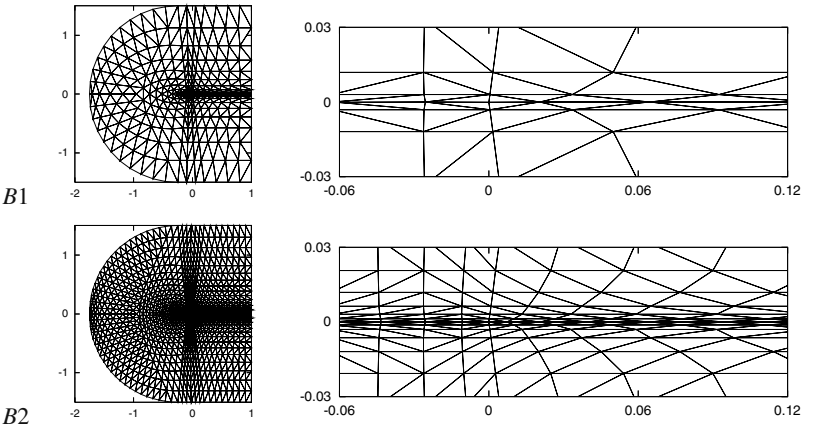


Fig. 1 Blasius problem, computational grids $B1$ and $B2$, the coarser one $B1$ having 662 elements (top) and the finer one $B2$ having 2648 elements (bottom), the whole computational domain (left) and their details around the leading edge (right)

5.1.1 Stability of the Method

We compared the NIPG, IIPG, SIPG variants of the DGM using piecewise linear, quadratic and cubic space approximation. Our aim is to find a suitable value of the constant C_W in (16) which ensures the stability of the scheme, i.e., a convergence to the steady-state solution. Firstly, we carried out computations for the values of $C_W = 1, 5, 25, 125, 625, 3\,125$ and consequently, several additional values of C_W were chosen in order to find a limit value of C_W. These results obtained on the grid

$B1$ are shown in Table 1, where an indication of a convergence of the appropriate variants of the IPG with a given value C_W is marked, namely,

- "convergence" (C), i.e., the stopping condition was achieved after less than 200 time steps,
- "quasi-convergence" (qC), i.e., the stopping condition was achieved after more than 200 time steps,
- "no-convergence" (NC), i.e., the stopping condition was not achieved after 500 time steps.

The "quasi-convergence" in fact means that the appropriate value C_W is just under the limit value ensuring a reasonable convergence to the steady-state solution.

Table 1 Blasius problem, the convergence (C), non-convergence (N) or quasi-convergence (qC) of the NIPG, IIPG and SIPG methods for P_1, P_2 and P_3 approximations for different values of C_W (symbol "-" means that the appropriate combination of the method, the degree of approximation and the value of C_W was not tested)

	1	5	10	25	100	125	250	300	400	500	625	1 000	3 125
P_1	C	C	-	C	-	-	-	-	-	-	C	-	C
NIPG P_2	C	C	-	C	-	-	-	-	-	-	C	-	C
P_3	C	C	-	C	-	-	-	-	-	-	C	-	C
P_1	C	C	-	C	-	-	-	-	-	-	C	-	C
IIPG P_2	N	C	C	C	-	-	-	-	-	-	C	-	C
P_3	N	N	C	C	-	-	-	-	-	-	C	-	C
P_1	N	N	-	N	N	C	-	-	-	-	C	-	C
SIPG P_2	N	N	-	N	-	N	N	qC	C	C	C	-	C
P_3	N	N	-	N	-	N	-	-	N	N	qC	C	C

From Table 1 we observe that

- NIPG variant converges for any $C_W \geq 1$ independently on the degree of polynomial approximation,
- IIPG variant requires higher values of C_W for P_2 and P_3 approximations, namely $C_W = 5$ and $C_W = 10$ are sufficient, respectively. On the other hand, P_1 approximation converges for any $C_W \geq 1$.
- SIPG variant requires significantly higher values of C_W. We observe that $C_W \geq 125$ for P_1, $C_W \geq 400$ for P_1 and $C_W \geq 1\,000$ for P_3. This is in a good agreement with the theoretical results from [15] carried out for a scalar quasilinear elliptic problem, where the dependence $C_W = cp^2,\ c > 0$ is employed (p denotes the polynomial degree of approximation).

5.1.2 Accuracy of the Method

We compared the numerical solutions with the "theoretical" one, which can be obtained from the well-known Blasius problem represented by an incompressible flow

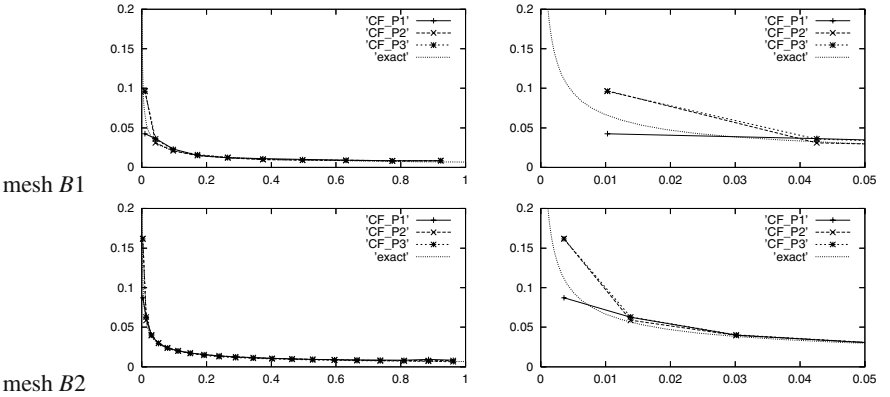


Fig. 2 Blasius problem, skin friction coefficient computed on meshes $B1$ and $B2$ by P_1 ('P_1'), P_2 ('P_2') and P_3 ('P_3') approximation in comparison with the Blasius formula ('exact'), distributions along the whole plate (left), their details around the leading edge (right)

along a flat plate. Figure 2 shows the comparison of the computed skin friction coefficient c_f achieved by P_1, P_2 and P_3 approximations on meshes $B1$ and $B2$ with the exact solution given by the Blasius formula. We observed a good agreement with the Blasius solution. However, P_2 and P_3 approximations give in fact the same value of c_f at the first element on the flat plate. Similar results were obtained in [3, Fig. 2] where the difference among P_1, P_2 and P_3 approximations on the first cell of the flat plate is almost negligible. We suppose that it can be caused by the singularity of the solution at $x_1 = x_2 = 0$ which decreases the local order of accuracy of the IPG method. This phenomenon was numerically verified for a scalar nonlinear convection-diffusion equation in [9].

5.2 Steady-State Flow around NACA0012 Profile

In Section 5.1, we studied the influence of the value of the penalty parameter C_W introduced in (16) to the stability of the NIPG, IIPG and SIPG variants (26). We do not observed any essential influence of C_W to, e.g., the skin friction coefficient. Nevertheless, the influence of C_W to the numerical solution should be investigated by a quantitative characteristic of the flow. Hence, we consider a flow around the profile NACA0012 at the free stream Mach number $M = 0.5$, the angle of attack $\alpha = 0^\circ$ and Reynolds number $Re = 5\,000$. The walls of the profile are adiabatic. The Reynolds number is near to the upper limit for the steady laminar flow. A characteristic feature of this flow problem is the linear separation of the flow occurring near to the trailing edge.

We carried out computations on a set of six successively generated ADIGMA grids $N1$ – $N6$ from [1]. The numbers of elements ($= \#\mathcal{T}_h$) and mesh sizes ($= 1/\sqrt{\#\mathcal{T}_h}$) of grids $N1$ – $N6$ are shown in Table 2 (top). We investigated a "convergence" of the drag coefficient c_D for "$h \to 0$" for the NIPG variant with

Table 2 NACA 0012 profile ($M = 0.5$, $\alpha = 0^\circ$, $Re = 5000$), numbers of elements ($= \#\mathcal{T}_h$) and mesh sizes ($= 1/\sqrt{\#\mathcal{T}_h}$) of grids $N1$ – $N6$ and the corresponding values of the drag coefficient c_D computed by the NIPG method for different values of C_W

mesh	$N1$	$N2$	$N3$	$N4$	$N5$	$N6$
$\#\mathcal{T}_h$	1148	2262	4216	8482	17888	40440
$1/\sqrt{\#\mathcal{T}_h}$	2.95E-02	2.10E-02	1.54E-02	1.09E-02	7.48E-03	4.97E-03

P_k	C_W	$N1$	$N2$	$N3$	$N4$	$N5$	$N6$
P_1	1	0.03322	0.04913	0.05288	0.05429	0.05470	0.05492
P_1	5	0.04289	0.04945	0.05150	0.05356	0.05459	0.05488
P_1	25	0.04692	0.04749	0.04910	0.05203	0.05379	0.05448
P_1	250	0.04157	0.04217	0.04605	0.05093	0.05271	0.05379
P_2	1	0.05538	0.05548	0.05489	0.05482	0.05486	–
P_2	5	0.05431	0.05423	0.05436	0.05467	0.05473	–
P_2	25	0.05167	0.05199	0.05373	0.05458	0.05473	–
P_2	250	0.04796	0.05137	0.05337	0.05428	0.05459	–
P_3	1	0.05939	0.05599	0.05500	0.05492	–	–
P_3	5	0.05783	0.05523	0.05467	0.05468	–	–
P_3	25	0.05475	0.05393	0.05374	0.05471	–	–
P_3	250	0.05178	0.05232	0.05477	0.05480	–	–

several choices of C_W. (We observed the same behavior as well as for the IIPG and SIPG techniques.) The values of c_D are presented in Table 2 and also visualized in Figure 3. We easily observe a non-negligible dependence of c_D on C_W on coarser

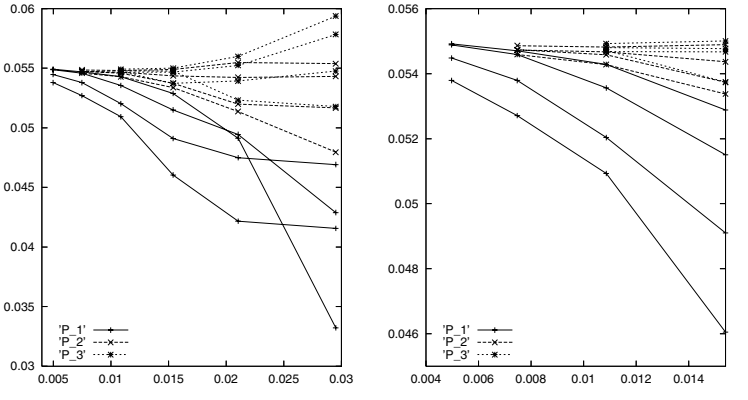


Fig. 3 NACA 0012 profile ($M = 0.5$, $\alpha = 0^\circ$, $Re = 5000$), visualization of results from Table 2, the dependencies of the drag coefficient c_D on mesh size ($= 1/\sqrt{\#\mathcal{T}_h}$) obtained by the NIPG method for different values of C_W and P_1, P_2 and P_3 approximations on meshes $N1$ – $N6$ (left) and its detail on meshes $N3$ – $N6$ (right)

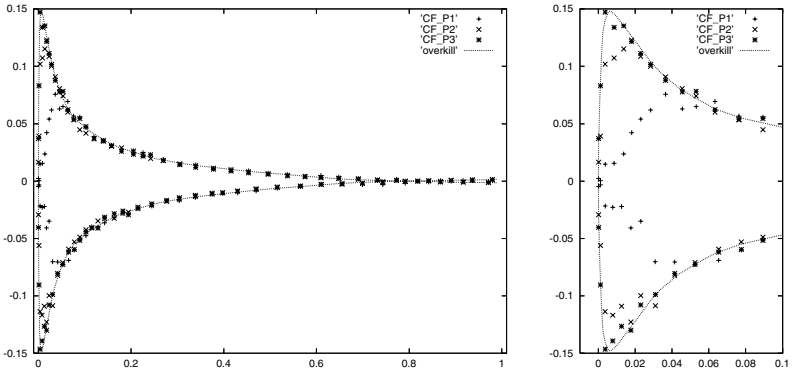


Fig. 4 NACA 0012 profile ($M = 0.5$, $\alpha = 0^\circ$, $Re = 5\,000$), distribution of the skin friction coefficient (left) with a detail around leading edge (right) obtained by piecewise linear (CF_P1), quadratic (CF_P2) and cubic approximations (CF_P3) on mesh $A1$ in comparison with "an exact" solution obtained by an "overkill" computation

grids but for increasing number of elements $\#\mathcal{T}_h$ the influence of C_W to c_D decreases and c_D converges to an asymptotic value. All values of c_D obtained on the finest employed meshes are within the range size 0.00113 ($\approx 2\%$ of the value c_D) for P_1 approximation, 0.00027 ($\approx 0.5\%$) for P_2 approximation and 0.00024 ($\approx 0.5\%$) for P_3 approximation.

Finally, we carried out additional computations on an adaptively refined grid $A1$ (obtained by the anisotropic adaptation technique [6] having 2 600 elements. Figure 4 show the corresponding distributions of the skin friction coefficient in comparison with an "exact" solution obtain by an "overkill" computation.We observe an increase of accuracy for increasing polynomial degree of approximation.

6 Conclusion

We presented the IPG space semi-discretization of the system of the Navier-Stokes equations. We discuss the choice of stabilization terms and presented a set of numerical experiments demonstrating the stability, convergence and a high order of accuracy of the proposed method.

References

1. ADIGMA. Adaptive higher-order variational methods for aerodynamic applications in industry, Specific Targeted Research Project no. 30719 supported by European Ccommison, `http://www.dlr.de/as/en/Desktopdefault.aspx/tabid-2035/2979_read-4582/`
2. Arnold, D.N.: An interior penalty finite element method with discontinuous elements. SIAM J. Numer. Anal. 19(4), 742–760 (1982)

3. Bassi, F., Rebay, S.: A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. J. Comput. Phys. 131, 267–279 (1997)
4. Baumann, C.E., Oden, J.T.: A discontinuous hp finite element method for the Euler and Navier-Stokes equations. Int. J. Numer. Methods Fluids 31(1), 79–95 (1999)
5. Dawson, C.N., Sun, S., Wheeler, M.F.: Compatible algorithms for coupled flow and transport. Comput. Meth. Appl. Mech. Engng. 193, 2565–2580 (2004)
6. Dolejší, V.: Anisotropic mesh adaptation for finite volume and finite element methods on triangular meshes. Comput. Vis. Sci. 1(3), 165–178 (1998)
7. Dolejší, V.: On the discontinuous Galerkin method for the numerical solution of the Navier–Stokes equations. Int. J. Numer. Methods Fluids 45, 1083–1106 (2004)
8. Dolejší, V.: Semi-implicit interior penalty discontinuous Galerkin methods for viscous compressible flows. Commun. Comput. Phys. 4(2), 231–274 (2008)
9. Dolejší, V., Feistauer, M., Kučera, V., Sobotíková, V.: An optimal $L^\infty(L^2)$-error estimate of the discontinuous galerkin method for a nonlinear nonstationary convection-diffusion problem. IMA J. Numer. Anal. 28(3), 496–521 (2008)
10. Dolejší, V., Feistauer, M., Sobotíková, V.: Analysis of the discontinuous galerkin method for nonlinear convection diffusion problems. Comput. Methods Appl. Mech. Eng. 194, 2709–2733 (2005)
11. Dolejší, V.: Discontinuous Galerkin method for the numerical simulation of unsteady compressible flow. WSEAS Transactions on Systems 5(5), 1083–1090 (2006)
12. Feistauer, M.: Mathematical Methods in Fluid Dynamics. Longman Scientific & Technical, Harlow (1993)
13. Feistauer, M., Felcman, J., Straškraba, I.: Mathematical and Computational Methods for Compressible Flow. Oxford University Press, Oxford (2003)
14. Hartmann, R., Houston, P.: Symmetric interior penalty DG methods for the compressible Navier-Stokes equations I: Method formulation. Int. J. Numer. Anal. Model. 1, 1–20 (2006)
15. Houston, P., Robson, J., Süli, E.: Discontinuous Galerkin finite element approximation of quasilinear elliptic boundary value problems I: The scalar case. IMA J. Numer. Anal. 25, 726–749 (2005)
16. Rivière, B., Wheeler, M.F., Girault, V.: Improved energy estimates for interior penalty, constrained and discontinuous Galerkin methods for elliptic problems. I. Comput. Geosci. 3(3-4), 337–360 (1999)
17. Vijayasundaram, G.: Transonic flow simulation using upstream centered scheme of Godunov type in finite elements. J. Comput. Phys. 63, 416–433 (1986)
18. Wesseling, P.: Principles of Computational Fluid Dynamics. Springer, Berlin (2001)

This page intentionally left blank

Chapter 8
Development of Discontinuous Galerkin Method for RANS Equations on Multibloc Hexahedral Meshes

F. Renac

Abstract. This article concerns the implementation of a Discontinuous Galerkin method for laminar and turbulent flow computations. The compressible Reynolds-averaged Navier-Stokes equations coupled with the $k-\omega$ turbulence model of Wilcox are discretized using the BR1 scheme [Bassi and Rebay, JCP, **131**, 1997] on structured meshes with hexahedral elements and an explicit Runge-Kutta technique for the time integration. The boundary condition treatment is performed through a reconstruction of the solution at the physical boundary which avoids the use of a ghost cell technique. This method allows space discretization with overlapping and non-matching multidomains as well as high order polynomial approximations of the solution. Numerical results concern laminar and turbulent flow computations on two-dimensional and three-dimensional domains.

1 Introduction

Over the last decade there has been a strong interest in the development of Discontinuous Galerkin (DG) methods for the discretization of compressible fluid flows (see Ref. [1] and references cited therein for a review). Many DG methods have been applied to solve non-linear convection-diffusion equations: the local Discontinuous Galerkin method [2], the BR1 and BR2 schemes [3, 4], the compact Discontinuous Galerkin method [5], the symmetric interior penalty method [6] (see Ref. [7] for an unified analysis of some of these schemes).

This article concerns the development of a Discontinuous Galerkin method for laminar and turbulent flow computations performed in the context of the European project ADIGMA [8]. We present the numerical simulation of the compressible Navier-Stokes and Reynolds-averaged Navier-Stokes (RANS) equations using a DG space discretization. We reformulate the equations as a first-order system in space

F. Renac
ONERA BP 72 - 29 avenue de la Division Leclerc FR-92322 Châtillon Cedex
e-mail: Florent.Renac@onera.fr

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 95–106, 2010.
springerlink.com

using the BR1 scheme introduced by Bassi and Rebay [3]. The two-equation $k-\omega$ turbulence model of Wilcox [9] is considered for the RANS equations.

The numerical approach is introduced in Sect. 2 and numerical results are presented in Sect. 3 for inviscid, laminar and turbulent test cases from the ADIGMA project. The convergence of global functions with mesh refinement is then explored. Finally, we end with some concluding remarks in Sect. 4.

2 Numerical Approach

2.1 RANS Equations

Let $\Omega \subset \mathbb{R}^d$ with $d=2$ or $d=3$ be a bounded domain, $\partial\Omega$ denotes the boundary of Ω. In the following, we introduce the DG discretization for turbulent flow simulations. A similar approach is applied for the discretization of laminar flow equations.

We consider the system of RANS equations coupled with the Wilcox's $k-\omega$ two-equation turbulence model. The system in conservative form reads:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla\cdot\mathbf{F_c}(\mathbf{u}) + \nabla\cdot\mathbf{F_v}(\mathbf{u},\nabla\mathbf{u}) = \mathbf{S}(\mathbf{u},\nabla\mathbf{u}), \tag{1}$$

where $\mathbf{u} = (\rho, \rho u, \rho v, \rho w, \rho E, \rho k, \rho\omega)^t$ denotes the conservative variable vector and t is the transpose operator, ρ stands for the density, $\mathbf{V} = (u,v,w)^t$ denotes the velocity vector, and $E = p/(\gamma-1)\rho + \mathbf{V}^2/2$ is the total energy per unit of mass with p the static pressure and γ the specific heat ratio. The turbulent variables are the kinetic energy of turbulence k and the specific rate of dissipation ω. The convective fluxes $\mathbf{F_c}$, the viscous fluxes $\mathbf{F_v}$, and the source terms a are defined by (see Ref. [9] for details on the parameters):

$$\mathbf{F_c} = \begin{pmatrix} \rho\mathbf{V} \\ \rho\mathbf{V}\otimes\mathbf{V} + p\mathbf{I} \\ (\rho E + p)\mathbf{V} \\ \rho k\mathbf{V} \\ \rho\omega\mathbf{V} \end{pmatrix}, \quad \mathbf{F_v} = \begin{pmatrix} 0 \\ -\tau - \tau_r \\ \mathbf{q} + \mathbf{q_t} - (\tau+\tau_r)\cdot\mathbf{V} \\ -(\mu+\sigma_k\mu_t)\nabla k \\ -(\mu+\sigma_\omega\mu_t)\nabla\omega \end{pmatrix},$$

and

$$\mathbf{S} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \tau_r : \nabla\mathbf{V} - \beta^\star\rho k\omega \\ \frac{\gamma^\star\omega}{k}\tau_r : \nabla\mathbf{V} - \beta\rho\omega^2 \end{pmatrix},$$

where μ stands for the kinematic viscosity. We use the model of Newtonian fluid for the viscous stress tensor and the Fourier's law for the heat transfer vector:

$$\tau = -\frac{2}{3}\mu(\nabla\cdot\mathbf{V})\mathbf{I} + \mu(\nabla\mathbf{V} + \nabla\mathbf{V}^T)\,,$$
$$\mathbf{q} = -\frac{C_p\mu}{Pr}\nabla T.$$

The system is completed by the Boussinesq assumption:

$$\tau_r = -\frac{2}{3}(\rho k + \mu_t\nabla\cdot\mathbf{V})\mathbf{I} + \mu_t(\nabla\mathbf{V} + \nabla\mathbf{V}^T)\,,$$
$$\mathbf{q_t} = -\frac{C_p\mu_t}{Pr_t}\nabla T.$$

where the turbulent viscosity is defined by $\mu_t = \rho k/\omega$. Finally, numerical values of the constants of the model are set as follows

$$\beta^\star = 0.09\,,\ \beta = 0.075\,,\ \sigma^\star = \sigma = 0.5\,,\ \gamma^\star = \frac{\beta}{\beta^\star} - \sigma\frac{\kappa^2}{\sqrt{\beta^\star}} = 0.5532\,,$$

and $\kappa = 0.41$ denotes the von Kàrmàn constant.

2.2 Space Discretization

The domain Ω is discretized with hexahedral elements: $\Omega_h = \cup_{i=1}^{N} K_i$. The spatial discretization of the diffusive and source terms is constructed by regarding the gradient of the conservative variables as additional unknowns of the problem

$$\mathbf{G}(\mathbf{u}) = \nabla\mathbf{u} \tag{2}$$

so we have $\mathbf{F_v} = \mathbf{F_v}(\mathbf{u},\mathbf{G})$ and $\mathbf{S} = \mathbf{S}(\mathbf{u},\mathbf{G})$.

Multiplying equations (1) and (2) by a test function ϕ and integrating by parts element by element leads to the weak formulation of the problem. The discrete version of the weak formulation of the system (1-2) in each element is

$$\int_{K_i}\phi\mathbf{G}_h d\Omega - \oint_{\partial K_i}\phi\hat{\mathbf{u}}_h\otimes\mathbf{n}dS + \int_{K_i}\nabla\phi\otimes\mathbf{u}_h d\Omega = 0\,, \tag{3a}$$

$$\int_{K_i}\phi\frac{\partial\mathbf{u}_h}{\partial t}d\Omega + \oint_{\partial K_i}\phi(\hat{F}_c + \hat{F}_v)dS - \int_{K_i}(\mathbf{F_c} + \mathbf{F_v})\cdot\nabla\phi d\Omega - \int_{K_i}\phi\mathbf{S}d\Omega = 0\,, \tag{3b}$$

where $\mathbf{n}$ is the outward unit normal vector and

$$\mathbf{G}_h = \sum_{j=1}^{n}\mathscr{G}_j\phi_j(\mathbf{x})\,,\quad \mathbf{u}_h = \sum_{j=1}^{n}\mathscr{U}_j\phi_j(\mathbf{x}) \tag{4}$$

represent approximate solutions of the initial equations (1-2) where $\mathscr{U}_j$ stand for the degrees of freedom of the problem. The functions ϕ_j represent a basis of the function space of piecewise discontinuous polynomials of degree k inside each cell:

$$V_h = \{\phi \in L^2(\Omega_h) : \phi|_{K_i} \in P_k(K_i), 1 \leq i \leq N\} \tag{5}$$

where $P_k(K_i)$ represents the space of polynomials in element K_i of degree at most k. We use the monomials 1, $(x-x_i)$, $(x-x_i)^2$, $(x-x_i)(y-y_i)$, etc. as basis of the function space V_h where $\mathbf{x}_i = (x_i, y_i, z_i)^t$ represents the centroid of the element K_i.

The numerical fluxes in contour integrals of equation (3) are chosen so as to be uniquely defined at the boundary ∂K_i of each element and to satisfy the consistency and conservativity conditions. We use a centred scheme for the gradient construction:

$$\hat{\mathbf{u}}_h = \frac{\mathbf{u}_h^- + \mathbf{u}_h^+}{2}\,, \tag{6}$$

where $\mathbf{u}_h^+$ denotes the trace of the internal variable $\mathbf{u}_h$ evaluated on the interface and $\mathbf{u}_h^-$ denotes the trace on the interface of the variable in the neighbouring element.

The numerical flux at a boundary is evaluated through an appropriate value $\hat{\mathbf{u}}_h = \mathbf{u}_{BC}$ which ensures that the boundary condition is verified. The boundary value $\mathbf{u}_{BC}$ is computed by imposing physical boundary data for Dirichlet conditions and the Riemann invariant associated to outgoing characteristics evaluated from $\mathbf{u}_h^+$.

The convective fluxes of the mean flow quantities are discretized by using a local Lax-Friedrichs flux with artificial dissipation:

$$\hat{F}_c = \mathbf{F_c}\Big(\frac{\mathbf{u}_h^- + \mathbf{u}_h^+}{2}\Big)\cdot\mathbf{n} - k_2\rho_s(\mathbf{u}_h^+ - \mathbf{u}_h^-)\cdot\mathbf{n}\,, \tag{7}$$

where the spectral radius is defined by $\rho_s = \max\{\|\mathbf{V}_h^+\| + c_h^+, \|\mathbf{V}_h^-\| + c_h^-\}$, c_h denotes the speed of sound and the artificial viscosity parameter is set at $k_2 = 0.25$ for viscous calculations and at $k_2 = 0.05$ for inviscid computations. The numerical flux at a boundary is evaluated by $\hat{F}_c = \mathbf{F_c}(\mathbf{u}_{BC})\cdot\mathbf{n} - k_2\rho_s(\mathbf{u}_h^+ - \mathbf{u}_{BC})\cdot\mathbf{n}$ where $\mathbf{u}_{BC}$ has been defined above.

The viscous fluxes are replaced by a centred scheme

$$\hat{F}_v = \Big(\frac{\mathbf{F_v}(\mathbf{u}_h^-, \mathbf{G}_h^-) + \mathbf{F_v}(\mathbf{u}_h^+, \mathbf{G}_h^+)}{2}\Big)\cdot\mathbf{n}. \tag{8}$$

The boundary treatment for $\mathbf{u}_h$ is similar to the convective flux: $\hat{F}_v = \mathbf{F_v}(\mathbf{u}_{BC}, \mathbf{G}_{BC})\cdot\mathbf{n}$ and $\mathbf{G}_{BC}$ is imposed if there are boundary conditions on $\nabla\mathbf{u}\cdot\mathbf{n}$, otherwise we set $\mathbf{G}_{BC} = \mathbf{G}_h^+$.

The surface and volume integrals in equation (3) are evaluated by means of Gauss quadrature formulae in the brick reference element $K_B = \{\xi = (\xi_1, \xi_2, \xi_3)^t : -1 \leq \xi_j \leq 1, 1 \leq j \leq 3\}$ associated to a linear mapping from the reference element to the physical cell. The integrand is evaluated at each Gauss point by using the polynomial approximation of the conservative variables $\mathbf{u}_h$ and of their gradients $\mathbf{G}_h$.

Finally, the time integration of the system is accomplished with an explicit four-stage Runge-Kutta method of second order accuracy in time. The same time scheme is used for mean and turbulent variables. In order to keep the modular feature of the solver, the time integration procedure is based on a decoupling between RANS and $k-\omega$ systems of equations. This allows the use of specific numerical flux for each system and we use either the local Lax-Friedrichs flux (7) or a Roe flux for the turbulent quantities.

3 Applications

3.1 Inviscid Test Cases

Figure 1 presents an Euler validation of the DG method for a 3D transonic flow around the Onera M6-wing set at $\alpha = 3.06^\circ$ of incidence and with a freestream Mach number of $M = 0.84$. The space is discretized with a 3D structured grid with one block. Results of the DG method are obtained with P_1 elements as approximation of the solution. The volume and contour integrals in (3) are calculated with second-order Gauss quadrature formulae.

Results are shown in Fig. 1. As shown by the residual history, the computation converges to machine accuracy. The C_p-distribution, obtained with the P_1 DG method, is in agreement with that obtained with a classical second-order Jameson finite volume (FV) method and demonstrates the shock capturing capability of the DG method.

We also computed two inviscid flows around the NACA0012 airfoil using a structured C-grid with 225×33 nodes: a subsonic flow at a freestream Mach number of $M = 0.5$ and with an angle of attack of $\alpha = 2^\circ$ and a transonic flow at a freestream Mach number of $M = 0.8$ and with an angle of attack of $\alpha = 1.25^\circ$. Figures 2 and 3 display the Mach number contours in both regimes. The upper and lower shocks are well captured in the transonic regime.

Convergence of global functions with mesh refinement is evaluated on four grids with different mesh sizes. The global functions are the pressure contributions to lift, drag and pitching moment coefficients. The computations are summarized in Tabs. 1 and 2 for both flows and for each grid. Grid convergence has been achieved for lift and pitching moment coefficients, while the drag moment coefficient presents poor convergence properties. Note that this coefficient has low amplitude compared to other coefficients.

The extension of the DG discretization to higer order approximation P_2 is shown in Fig. 2 and Tab. 1. Results highlight similar trends with a faster convergence of global quantities with mesh refinement compared to the P_1 approximation.

3.2 Viscous Test Cases

Figure 4 presents the computation of the laminar flow on an adiabatic flat plate at a freestream Mach number of $M = 0.5$ and a Reynolds number of $Re = 5 \times 10^3$

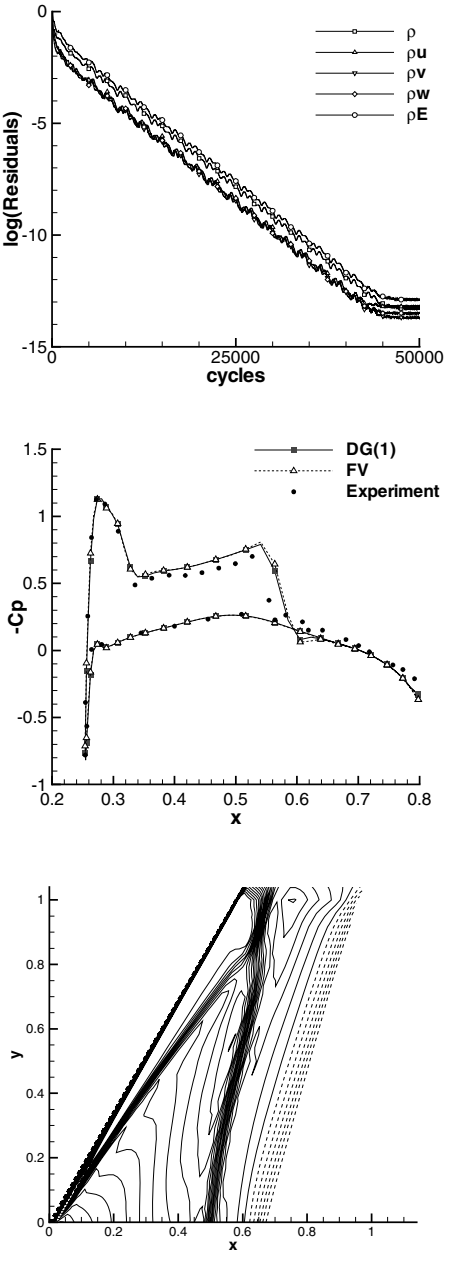


Fig. 1 Euler computation of the Onera M6-wing: (top) logarithmic plot of the residuals; (middle) computations (CFL number of 0.4) of static pressure coefficients along the wing surface at $y = 0.44$ and comparison with experiments [10]; (bottom) iso-C_p distributions on the upper side of the wing ($\Delta C_p = 0.05$ between two levels and dashed lines refer to negative values).

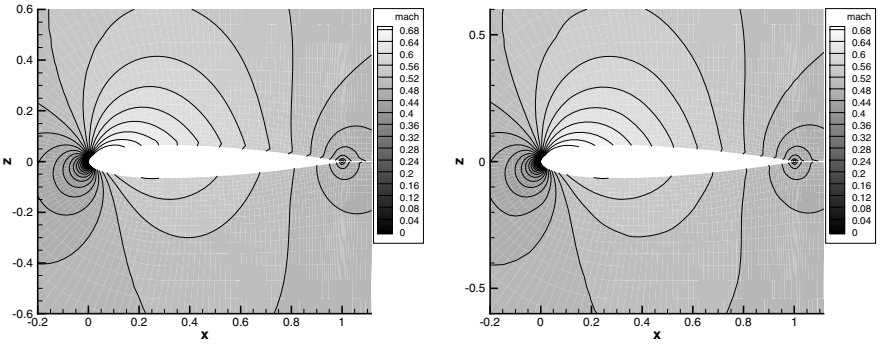


Fig. 2 Mach number contours of the inviscid subsonic flow around a NACA0012 airfoil with $M = 0.5$ and $\alpha = 2^\circ$ (225×33 grid): $P1$ approximation (left) and P_2 approximation (right)

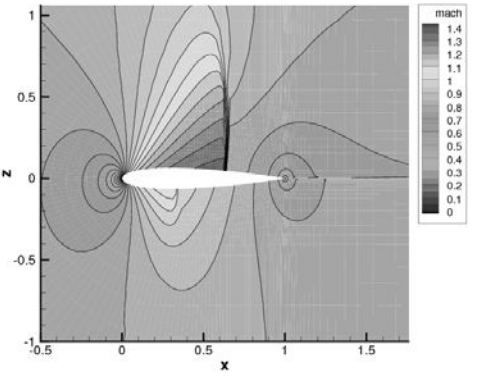


Fig. 3 Mach number contours of the inviscid transonic flow around a NACA0012 airfoil with $M = 0.8$ and $\alpha = 1.25^\circ$ (225×33 grid, P_1 approximation)

Table 1 Drag, lift and pitching moment coefficients for the Euler subsonic flow around a NACA0012 airfoil for P_1 (top) and P_2 approximations (bottom)

Test case	Grid	$C_{D,p}$	$C_{L,p}$	$C_{M,p}$
$M = 0.5, \alpha = 2^\circ$	113×17	$5.295e-3$	$2.775e-1$	$-2.488e-3$
	225×33	$1.297e-3$	$2.816e-1$	$-2.528e-3$
	449×65	$4.360e-4$	$2.831e-1$	$-2.680e-3$
	897×129	$3.890e-4$	$2.831e-1$	$-2.525e-3$
$M = 0.5, \alpha = 2^\circ$	113×17	$3.570e-3$	$2.761e-1$	$-6.880e-4$
	225×33	$8.900e-4$	$2.795e-1$	$-1.610e-3$
	897×129	$6.690e-4$	$2.741e-1$	$-1.040e-3$

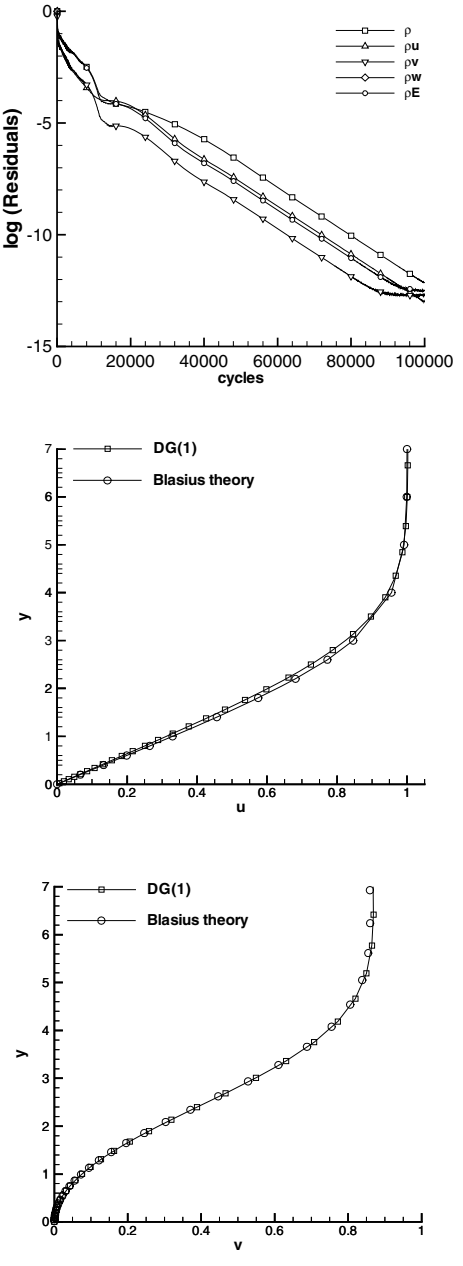


Fig. 4 Laminar flow on an adiabatic flat plate ($M = 0.5$ and $Re = 5 \times 10^3$): (top) logarithmic plot of the residuals; computations of the non-dimensional profiles of axial (middle) and vertical (bottom) velocity components and comparison with the Blasius theoretical solution.

Table 2 Drag, lift and pitching moment coefficients for the Euler transonic flow around the NACA0012 airfoil; P_1 approximation

Test case	Grid	$C_{D,p}$	$C_{L,p}$	$C_{M,p}$
$M = 0.8$, $\alpha = 1.25^\circ$	113×17	$2.518e-2$	$3.361e-1$	$-3.457e-2$
	225×33	$2.267e-2$	$3.465e-1$	$-3.670e-2$
	449×65	$2.224e-2$	$3.476e-1$	$-3.825e-2$
	897×129	$2.088e-2$	$3.522e-1$	$-3.478e-2$

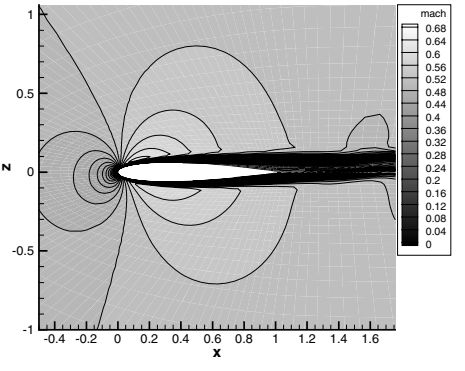


Fig. 5 Laminar flow around the NACA0012 airfoil with $M = 0.5$, $Re = 5 \times 10^3$ and $\alpha = 2^\circ$ (161×41 grid, P_1 approximation): Mach number contours

based on freestream quantities and plate length. A Cartesian grid with 81×57 nodes is used as space discretization and P_1 elements are considered. Computations of non-dimensional axial and vertical velocities agree well with the theoretical Blasius solution.

The second viscous test case concerns the 2D laminar flow around the NACA0012 airfoil at $\alpha = 2^\circ$ of incidence with a freestream Mach number of $M = 0.5$ and a Reynolds number, based on freestream quantities and chord length, of $Re = 5 \times 10^3$. A non slip condition is applied on adiabatic wall. Computations have been performed on a structured C-grid with 161×41 nodes. Results with the P_1 approximation are depicted in Fig. 5. The method allows describing the recirculation bubble in the wake region of the airfoil. Table 3 presents the grid convergence of global coefficients with pressure and viscous contributions. Pressure parts of the coefficients achieve better convergence than the corresponding viscous parts.

3.3 Turbulent Test Case

The DG method has been applied to a turbulent test case. This test case consists in a turbulent flow around a RAE2822 airfoil at an incidence of $\alpha = 2.79^\circ$ with a

Table 3 Drag, lift and pitching moment coefficients for the NACA0012 airfoil; P_1 approximation. Laminar subsonic flow ($M = 0.5$, $Re = 5000$, $\alpha = 2^\circ$).

Grid	$C_{D,p}$	$C_{D,v}$	C_D
161×41	$2.797e-2$	$2.807e-2$	$5.604e-2$
321×81	$2.518e-2$	$2.948e-2$	$5.466e-2$
641×161	$2.478e-2$	$3.126e-2$	$5.603e-2$
Grid	$C_{L,p}$	$C_{L,v}$	C_L
161×41	$3.653e-2$	$-1.260e-4$	$3.640e-2$
321×81	$3.806e-2$	$7.800e-4$	$3.814e-2$
641×161	$4.217e-2$	$3.070e-4$	$4.248e-2$
Grid	$C_{m,p}$	$C_{m,v}$	C_m
161×41	$1.682e-2$	$-3.020e-4$	$1.652e-2$
321×81	$1.639e-2$	$-2.370e-4$	$1.616e-2$
641×161	$1.783e-2$	$-1.980e-4$	$1.709e-2$

freestream Mach number of $M = 0.73$ and a Reynolds number, based on freestream quantities and chord length, of $Re = 6.5 \times 10^6$. The wall is assumed to be adiabatic with non slip condition. The domain is discretized by using a structured C-grid with 369×89 nodes. The RANS equations are coupled to the $k-\omega$ turbulence model of Wilcox as detailed in Sect. 2.1.

Figure 6a presents the convergence history of the residuals associated to the mean flow equations and Fig. 6b depicts the evolution of the global lift coefficient C_L as a function of the iteration number. Despite of the low convergence levels of the

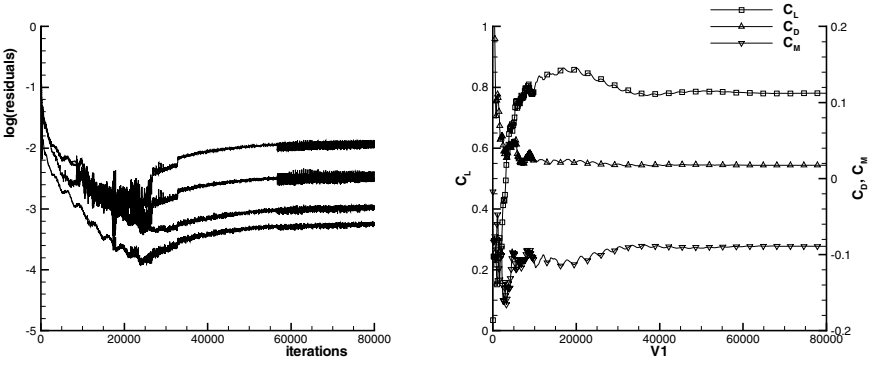


Fig. 6 Turbulent flow around the RAE2822 airfoil ($M = 0.73$, $Re = 6.5 \times 10^6$, $\alpha = 2.79^\circ$): logarithmic plot of the residuals (left) and convergence history of the aerodynamic coefficients (right)

Table 4 Drag, lift and pitching moment coefficients for the NACA0012 airfoil; P_1 approximation. Turbulent flow ($M = 0.73$, $Re = 6.5 \times 10^6$, $\alpha = 2.79^\circ$).

Grid	C_D	C_L	C_M
185×45	$7.078e-1$	$1.500e-2$	$-7.637e-2$
369×89	$7.806e-1$	$1.749e-2$	$-8.897e-2$

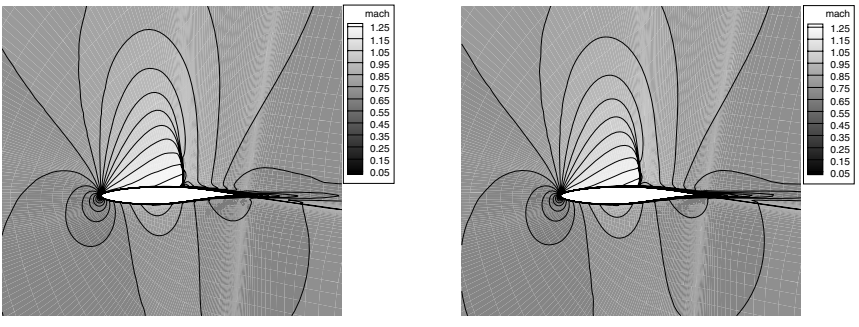


Fig. 7 Mach number contours around the RAE2822 airfoil ($M = 0.73$, $Re = 6.5 \times 10^6$, $\alpha = 2.79^\circ$): comparison between a finite volume computation (left) and a DG P_1 computation (right)

residuals, the global coefficient achieves a quick convergence to a steady value. Table 4 presents the evolution of global coefficients with mesh refinement.

Figure 7 displays the Mach number distribution obtained with the P_1 approximation and compares results to a second order FV computation. Compared to the FV computation, the shock structure is well defined with the P_1 approximation. Likewise, the separation region in the trailing edge region and the recirculation bubble are better described.

4 Conclusion

Flow computations with a discontinuous Galerkin scheme for 3D laminar and turbulent flows on multidomain hexahedral meshes with polynomial expansion order of $p \leq 2$ have been reported in this work. The compressible RANS equations coupled with the two-equation $k - \omega$ turbulence model of Wilcox are considered. The DG scheme is based on the introduction of an auxiliary variable for the gradient of the conservative variable vector in order to reduce the problem to a first order differential system [3]. An explicit Runge-Kutta scheme is applied for the time integration.

Results show the capability of computing 2D Euler flows with P_1 and P_2 approximations and 3D Euler flows with P_1 approximations. 3D laminar and 2D turbulent flow computations give also satisfactory results with P_1 approximation. The resolution of rotational regions and shock structure are improved by the DG method compared to second order finite volume computations.

References

1. Cockburn, B., Shu, C.W.: Runge-Kutta Discontinuous Galerkin Methods for convection dominated problems. J. Sc. Comput. 16, 173–261 (2001)
2. Cockburn, B., Shu, C.W.: The local discontinuous Galerkin method for convection-diffusion systems. SIAM J. Numer. Anal. 35, 2440–2463 (1998)
3. Bassi, F., Rebay, S.: A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. J. Comput. Phys. 131, 267–279 (1997)
4. Bassi, F., Crivellini, A., Rebay, S., Savini, M.: Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and k $-$ ω turbulence model equations. Comput. Fluids. 34, 507–540 (2005)
5. Peraire, J., Persson, P.-O.: The Compact Discontinuous Galerkin (CDG) Method for Elliptic Problems. SIAM J. Sci. Comput. 30, 1806–1824 (2008)
6. Hartmann, R., Houston, P.: Symmetric Interior Penalty DG Methods for the Compressible Navier-Stokes Equations I: Method Formulation. Int. J. Numer. Anal. Model. 3, 1–20 (2006)
7. Arnold, D.N., Brezzi, F., Cockburn, B., Marini, L.D.: Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems. SIAM J. Numer. Anal. 39, 1749–1779 (2001)
8. Kroll, N.: ADIGMA - A European project on the development of adaptative higher order variational methods for aerospace applications. In: Proceedings of ECCOMAS CFD 2006, TU Delft, The Netherlands (2006)
9. Wilcox, D.C.: Reassesment of the scale-determining equation for advanced turbulence models. AIAA J. 26, 1299–1310 (1979)
10. Schmitt, V., Charpin., F.: Pressure distributions on the ONERA M6-wing at Transonic Mach. numbers. AGARD TR. 138 (1988)

Chapter 9
Construction of High-Order Non Upwind Distribution Schemes

R. Abgrall, A Larat, and M. Ricchiuto

Abstract. In this paper we consider the very high order approximation of solutions of the Euler equations. We present a systematic generalization of the Residual Distribution method of [8] to very high order of accuracy, by extending the preliminary work discussed in [17]. We present extensive numerical validation for the third and fourth order cases with Lagrange finite elements. In particular, we demonstrate that we an both have a non oscillatory behavior, even for very strong shocks and complex flow patterns, and the expected accuracy on smooth problems. We also extend the scheme to laminar viscous problems.

1 Introduction

We are interested in the numerical approximation of steady hyperbolic problems

$$\text{div } \mathbf{f}(\mathbf{u}) = S(\mathbf{u}) \tag{1}$$

which are defined on an open set $\Omega \subset \mathbb{R}^d$, $d = 2,3$ with weak Dirichlet boundary conditions $u = g$. defined on the inflow boundary[1]

$$\partial\Omega^- = \{x \in \partial\Omega, \mathbf{n} \cdot \nabla_{\mathbf{u}}\mathbf{f} < 0\}.$$

R. Abgrall · M. Ricchiuto
INRIA and IMB, Team Bacchus, bat A29 bis, 341 Cours de la Libération,
33 405 Talence Cedex, France
e-mail: Remi.Abgrall@inria.fr

A. Larat
INRIA, Team Bacchus, bat A29 bis, 341 Cours de la Libération,
33 405 Talence Cedex, France
e-mail: Adam.Larat@inria.fr, Mario.Ricchiuto@inria.fr

[1] $\mathbf{n}$ is the inward normal.

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 107–128, 2010.
springerlink.com

In (1), the vector of unknown $\mathbf{u}$ belongs to $\mathbb{R}^p$, and the flux $\mathbf{f}$ is

$$\mathbf{f} = (f_1, \ldots, f_d).$$

In (1), S is a source term which here only depends on the unknown $\mathbf{u}$.

The main target example we are interested in is the system of the Euler equations with the vector of unknown

$$\mathbf{u} = (\rho, \rho\mathbf{u}, E)^T.$$

The density is ρ, $\mathbf{u}$ is the velocity and E is the total energy. The flux, in the case $d = 2$ to make things simple, is given by, using $\mathbf{u} = (u_1, u_2)^T$,

$$f_1 = \begin{pmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ u_1(E+p) \end{pmatrix}, \qquad f_2 = \begin{pmatrix} \rho u_2 \\ \rho u_1 u_2 \\ \rho u_2^2 + p \\ u_2(E+p) \end{pmatrix}. \tag{2}$$

The system is closed by an equation of state that relates the pressure p to $\mathbf{u}$. Here we assume a perfect gas equation of state. In the case of the Euler equations, we have several types of boundary conditions, we come back to this in due time.

In the recent years, there have been many researches to develop really robust and high order schemes for equations of the type (1) and in particular for (2). In this paper we are concerned with the approximation of these equations on conformal unstructured meshes. We restrict our-self to the case of two dimensional triangular type meshes, even-though things can be made more general [1].

We have chosen to develop a different strategy than Discontinuous Galerkin methods, the Residual Distribution schemes, where the stencil stays very local, as in the DG methods, but the number of degrees of freedom grows less quickly, even in 3D. The price to pay is to impose the continuity of the approximation $\mathbf{u}$ as in standard finite element methods. Indeed, the RD schemes can be seen as finite elements where the test functions may depend on the sought solution. This class of scheme is having a growing interest (see [2, 3, 4, 5, 6, 7, 8], etc.) but has mainly been developed for second order accuracy only, see however [9] for different but related approach on structured meshes. In this paper, we are interested in showing how the methods we have developed in previous papers can be extended to very high accuracy, even in the case of the Euler equation, at a relatively moderate price.

In the first section, we present the general form of RD schemes that are formally high order accurate. In the second section, we explicitly construct some examples of schemes. Then numerical illustrations for scalar problems are presented, and we show which type of difficulties our first solution leads to. Following a previous paper, we present a simple and efficient fix, and show on numerical examples which quality of solution we can reach. Then extension to the Euler equations are given, in the steady case as well as in the unsteady one. We also develop an implicit version of the scheme which is tested in several well known cases. A conclusion follows.

2 High Order Residual Distribution Schemes for Inviscid Flows

2.1 Introduction

We consider the problem (1) on a domain Ω. On this domain, we assume a tessellation τ_h which elements are triangles in 2D and tetrahedrons in 3D.[2] A generic element is denoted by T, there are n_t such elements. The list of vertices of the mesh τ_h is $\{M_i\}_{i=1,\ldots,n_s}$. When dealing with an element T, we denote when there is no ambiguity the list of its vertices by $1,\ldots,n_d$, n_d being the number of vertices in T.

In the residual distribution schemes, the degrees of freedom are associated to points, and not to control volumes as for finite volume methods or DG methods. We denote by $\{\sigma_l\}_{l=1\ldots,n_{dof}}$ the lists of degrees of freedom. In the case of a second order RD scheme, they are exactly the vertices of the mesh, so that $\sigma_l = M_l$ for any l. If one wants to construct a higher order accurate scheme, there are two options :

1. To update a solution in an element T, one can use information out of the element T. This option have been followed by [12, 7]. The compactness of the computational stencil is destroyed and need many indirections, which is not very computationally efficient. From a more theoretical view point, the complexity of codes becomes more and more important as one wishes to increase the order of accuracy.
2. On may add a supplementary constraint to the scheme : in order to locally update a degree of freedom belonging to the element T, only the degrees of freedom inside T are needed. The scheme is very compact, there is no indirection, but the situation is a bit more tricky.

We have chosen to follow the second option. For this we need to interpolate the unknown $\{\mathbf{u}_\sigma\}_{\sigma\in T}$ in T and we have chosen to use a continuous piecewise polynomial of degree k Lagrange interpolation. We denote by $\mathbb{P}^k(T)$ the set of polynomials of degree k defined on T.

The degrees of freedom are thus all the Lagrange points. To make things specific, we need for:

- Quadratic interpolation : the vertices and the mid–point edges. This yields $3+3$ points per triangle in 2D and $4+6$ points per tetrahedron in 3D.
- Cubic interpolation: in the 2D case, we need the vertices, 2 points per edge and the centroid, i.e. $3+2\times 3+1$ points per element. In the 3D case, we need the 2D dof per edge, the centroid of each face, i.e. $4+6\times 2+4=20$ dofs
- etc.

A residual distribution scheme for (1) writes, for an internal degree of freedom σ, as

$$\text{for all } \sigma \in T, \qquad \sum_{T\ni\sigma} \Phi_\sigma^T = 0. \tag{3}$$

[2] The parameter h denotes the maximum of the radius of C_T, $T\in\tau_h$, the circumscribed circles/sphere to T.

The residual in (3) must, following [13], satisfy the following conservation constraints

$$\text{for any } T, \sum_{\sigma\in T} \Phi_\sigma^T = \int_{\partial T} \mathbf{f}^h(\mathbf{u}^h)\cdot\mathbf{n}dl - \int_T S^h(\mathbf{u}^h)dx := \Phi^T \tag{4}$$

where $\mathbf{f}^h(\mathbf{u}^h)$ and $S^h(\mathbf{u}^h)$ are high order accurate approximations of the flux $\mathbf{f}(\mathbf{u})$ and the source term $S(\mathbf{u})$. Natural choices are either $\mathbf{f}^h(\mathbf{u}^h)$ is a Lagrange interpolant of $\mathbf{f}(\mathbf{u})$ at the degrees of freedom defining $\mathbf{u}^h$ or the true flux evaluated for $\mathbf{u}^h$.

Moreover, we also assume that the residuals Φ_σ^T depend continuously of their arguments. Indeed, we impose a more severe constraint, we assume that Φ_σ^T only depend on the values of $\{\mathbf{u}_{\sigma'}\}_{\sigma'\in T}$.

If Γ is any edge/face of the inflow boundary of Ω, we consider a numerical flux $\mathscr{F}$ which depends on the boundary condition $\mathbf{u}_-$, the inward normal $\mathbf{n}$ and the local state $\mathbf{u}^h$. Then we assume that we have boundary residuals Φ_σ^Γ which satisfy the following conservation relation

$$\text{for any } \Gamma\subset\partial\Omega, \sum_{\sigma\in\Gamma}\Phi_\sigma^\Gamma = \int_{\partial\Gamma}\Big(\mathscr{F}(\mathbf{u}^h,\mathbf{u}_-,\mathbf{n}) - \mathbf{f}^h(\mathbf{u}^h)\cdot\mathbf{n}\Big)dl := \Phi^\Gamma, \tag{5}$$

the residuals Φ_σ^Γ are assumed to be only dependant on the $\{\mathbf{u}_{\sigma'}\}_{\sigma'\in\Gamma}$, continuous, and the relation (4) has to be red, for any boundary node σ, as

$$\text{for all } \sigma\in\partial\Omega, \qquad \sum_{T\ni\sigma}\Phi_\sigma^T + \sum_{\Gamma\subset\partial\Omega^-,\Gamma\ni\sigma}\Phi_\sigma^\Gamma = 0. \tag{6}$$

Then following [13], it is easy to show that if the sequence $\mathbf{u}^h$ is bounded in L^∞ when $h\to 0$, and if there exists $\mathbf{v}$ such that $\mathbf{u}^h\to\mathbf{v}$ when $h\to 0$, then $\mathbf{v}$ is a weak solution of (1). Additional constraints can be set to fulfil entropy inequalities.

Before going further, let us introduce some simplifications. Instead of the conservation relations (4) which *a priori* need exact integration, one can of course use approximated quadrature. Hence, we replace the relations (4) and (5) by

$$\text{for any } T, \sum_{\sigma\in T}\Phi_\sigma^T = \oint_{\partial T}\mathbf{f}^h(\mathbf{u}^h)\cdot\mathbf{n}dl - \oint_T S^h(\mathbf{u}^h)dx := \widetilde{\Phi^T}. \tag{7a}$$

and

$$\text{for any } \Gamma\subset\partial\Omega^-, \sum_{\sigma\in\Gamma}\Phi_\sigma^\Gamma = \oint_{\partial\Gamma}\Big(\mathscr{F}(\mathbf{u}^h,\mathbf{u}_-,\mathbf{n}) - \mathbf{f}^h(\mathbf{u}^h)\cdot\mathbf{n}\Big)dl, \tag{7b}$$

where the symbol $\oint$ denotes an integral evaluated by mean of a quadrature formula.

2.2 Accuracy Constraints

Which quadrature formula should we use in practice ? This question is related to the formal accuracy of the scheme. We first explore this for the scheme (3), (6) with the conservation relations (4) and (5), and then we move to the more practical one with approximated quadrature formula (3), (6), (7).

Following the results of [13], we define the following truncation error

$$\mathscr{E}(\mathbf{u}^h, \varphi_h) = \sum_{\sigma \in \Omega} \varphi(\sigma) \left(\sum_{T \ni \sigma} \Phi_\sigma^T + \sum_{\Gamma \subset \partial\Omega^-, \Gamma \ni \sigma} \Phi_\sigma^\Gamma \right) \tag{8}$$

where φ_h is the Lagrange interpolant of $\{\varphi(\sigma)\}_\sigma$. The scheme is k-th order accurate if the truncation error is $O(h^k)$ when $\mathbf{u}^h$ is an interpolant of the exact solution, assumed to be smooth enough. We have the following result:

Proposition 1. *If the solution* $\mathbf{u}$ *is smooth enough and the residual, applied to the* $\mathbb{P}_k$ *interpolant of* $\mathbf{u}$ *satisfy*

$$\Phi_\sigma^T(\mathbf{u}^h) = \mathscr{O}(h^{k+d}) \tag{9a}$$

and

$$\Phi_\sigma^\Gamma = \mathscr{O}(h^{k+d-1}), \tag{9b}$$

if moreover the approximation $\mathbf{f}^h(\mathbf{u}^h)$ *is* $k+1$*-order accurate, then the truncation error satisfies*

$$|\mathscr{E}(\mathbf{u}^h, \varphi^h)| \leq C(\varphi, \mathbf{f}, \mathbf{u})\, h^{k+1}.$$

The constant $C(\varphi, \mathbf{u})$ *depends only on* φ *and* $\mathbf{u}$.

From the previous analysis, if there exists a constant (in the scalar case) or a matrix (in the system case) β_σ^T such that

$$\Phi_\sigma^T = \beta_\sigma^T \left(\int_{\partial T} \mathbf{f}^h(\mathbf{u}^h) \cdot \mathbf{n} dl - \int_T S^h(\mathbf{u}^h) dx \right), \tag{10a}$$

$$\Phi_\sigma^\Gamma = \beta_\sigma^\Gamma \left(\int_{\partial T} \left(\mathbf{f}^h(\mathbf{u}^h) \cdot \mathbf{n} - \mathscr{F}(\mathbf{u}^h, \mathbf{u}^-, \mathbf{n}) \right) dl \right), \tag{10b}$$

or more generally we keep the condition (9b), then the condition (9) is fulfilled provided that β_σ^T is uniformly bounded. In practice, the conditions (10) are replaced (if $\mathbf{u}^h$ is any polynomial of degree k) by

$$\Phi_\sigma^T = \beta_\sigma^T \left(\oint_{\partial T} \mathbf{f}^h(\mathbf{u}^h) \cdot \mathbf{n} dl - \oint_T S^h(\mathbf{u}^h) dx \right), \tag{11a}$$

$$\Phi_\sigma^\Gamma = \beta_\sigma^\Gamma \left(\oint_{\partial T} \left(\mathbf{f}^h(\mathbf{u}^h) \cdot \mathbf{n} - \mathscr{F}(\mathbf{u}^h, \mathbf{u}^-, \mathbf{n}) \right) dl \right), \tag{11b}$$

It can be seen by using exactly the same arguments that the constraints on the quadrature formula are the following :

- In (11a), we must have

$$\int_{\partial T} \mathbf{f}^h(\mathbf{u}^h)\cdot\mathbf{n}dl = \oint_{\partial T} \mathbf{f}^h(\mathbf{u}^h)\cdot\mathbf{n}dl + \mathscr{O}(h^{k+d}) \tag{12a}$$

  and

$$\int_{\partial T} S^h(\mathbf{u}^h)dx = \oint_{\partial T} S^h(\mathbf{u}^h)dx + \mathscr{O}(h^{k+d}) \tag{12b}$$

- In (11b), we must have for the integrated quantity

$$\int_{\partial T} g(\mathbf{u}^h)dl = \oint_{\partial T} g(\mathbf{u}^h)dl + \mathscr{O}(h^{k+d-1}) \tag{12c}$$

In order to obtain these errors, there are two possible ways. Either the quadrature formula is exact on the approximated flux $\mathbf{f}^h(u^h)$ which basically means that $\mathbf{f}(u^h)$ is a polynomial of degree k at least since we need that $\mathbf{f}(u) - \mathbf{f}^h(u^h) = O(h^{k+1})$, or the quadrature formula is not exact but provides this error. In the paper, we have followed the first method : in each element, the flux is reconstructed by the Lagrange interpolation of the exact flux evaluated at the degrees of freedom in the element. This is equivalent to a quadrature free approach. We come back to this point in section 3.3 to discuss our actual implementation of the boundary conditions.

2.3 Getting High Order Accuracy and Monotonicity Preservation

All the schemes we are aware of have residual that can be written, when $S \equiv 0$, as

$$\Phi_\sigma^T = \sum_{\sigma'\in T} c_{\sigma\sigma'}(\mathbf{u}_\sigma - \mathbf{u}_{\sigma'}) \tag{13}$$

so that that the relation (4) becomes for any σ

$$\sum_{T\ni\sigma}\sum_{\sigma'\in T} c_{\sigma\sigma'}(\mathbf{u}_\sigma - \mathbf{u}_{\sigma'}) = 0.$$

In general, this is a very complex set of non linear equation that can be solved by iterative methods. The simplest one is a Jacobi–like iteration.

$$\mathbf{u}_\sigma^{n+1} = \mathbf{u}_\sigma^n - \omega_\sigma\left(\sum_{T\ni\sigma}\sum_{\sigma'\in T} c^T_{\sigma\sigma'}(\mathbf{u}_\sigma - \mathbf{u}_{\sigma'})\right) \tag{14}$$

where ω_σ is a relaxation parameter. Note that the coefficient $c_{\sigma\sigma'}$ may depend on $\mathbf{u}$.

In the scalar case, we drop the bold symbol for the unknown u. It is easy to see that if the initial condition is such that $u_\tau^0 \in [a,b]$, then $u_\tau^n \in [a,b]$ provided that for any previous iteration we have

- $\sum_{T\ni\sigma,T'\ni\sigma'} c_{\sigma\sigma'} \geq 0$ for any σ, σ',
- for any σ, $1-\omega_\sigma\left(\sum_{T\in\sigma}\sum_{\sigma'\in T} c^T_{\sigma\sigma'}\right)\geq 0$.

A simpler set of conditions can be written, they are local in T. They are obtained by analogy with what is done for second order RD schemes. We first introduce the area $\mathscr{C}_\sigma$ and $\mathscr{C}^T_\sigma$:

$$\mathscr{C}_\sigma = \sum_{T\ni\sigma} \mathscr{C}^T_\sigma, \qquad\qquad \mathscr{C}^T_\sigma = \frac{|T|}{n_d}$$

and set

- $c_{\sigma\sigma'} \geq 0$ for any σ, σ',
- for any σ, ω_σ is such that

$$\omega_\sigma \max_{T\ni\sigma}\left[\frac{\mathscr{C}_\sigma}{\mathscr{C}^T_\sigma}\left(\sum_{\sigma'\in T} c^T_{\sigma\sigma'}\right)\right] \leq 1.$$

These conditions do not imply that the iterative scheme (14) is convergent, but only that a L^∞ bound is preserved. A scheme that preserves L^∞ bounds is said to be monotonicity preserving in the rest of the paper.

It is known that a scheme that is monotonicity preserving with coefficients $c_{\sigma\sigma'}$ constant cannot satisfy (9). This is true for second order RD scheme, see [14], and the proof of [14] does not use the type of Lagrange interpolation and hence can be extended to higher degree elements. This is a version of Godunov's theorem. Thus the scheme must be non linear.

There is a systematic way of constructing schemes that are both monotonicity preserving and satisfy (9). We first start from a monotone (first order scheme) which residuals are (for $S=0$) $\Phi^L_\sigma = \sum_{\sigma'\in T} c^L_{\sigma\sigma'}(u_\sigma - u_{\sigma'})$. The coefficients $c_{\sigma\sigma'}$ are *all* positive. Since there is no ambiguity, we drop the superscript T. We assume that $\sum_{\sigma\in T}\Phi^L_\sigma = \Phi^T (= \int_{\partial T} f^h(u^h)\cdot \mathbf{n} dl)$ where the integral is evaluated with the $\mathscr{P}^k(T)$ interpolant u^h. Then, if Φ^H_σ denote high order residuals, they also satisfy the same conservation relation $\sum_{\sigma\in T}\Phi^H_\sigma = \Phi^T$ and

$$\Phi^H_\sigma = \beta_\sigma \int_{\partial T} f^h(u^h)\cdot \mathbf{n} dl. \tag{15}$$

Clearly $\sum_\sigma \beta_\sigma = 1$ and this leads to introduce the parameters x_σ defined by $x_\sigma = \dfrac{\Phi^H_\sigma}{\Phi^T}$ for which, thanks to the conservation relation, we also have $\sum_\sigma x_\sigma = 1$.

The next step is to write the formal identity $\Phi^H_\sigma = \dfrac{\Phi^H_\sigma}{\Phi^L_\sigma}\Phi^L_\sigma = \sum_{\sigma'}\dfrac{\Phi^H_\sigma}{\Phi^L_\sigma} c^L_{\sigma\sigma'}(u_\sigma - u_{\sigma'})$ and we get a monotonicity preserving constraint if for each $\sigma\in T$ we have

$\frac{\Phi_\sigma^H}{\Phi_\sigma^L} \geq 0$ because then $c^H_{\sigma\sigma'} = \frac{\Phi_\sigma^H}{\Phi_\sigma^L} c^L_{\sigma\sigma'} \geq 0$. All this can be rephrased in term of the x_σs and β_σs :

1. Conservation. We have $\sum_\sigma \beta_\sigma = 1$ and $\sum_\sigma x_\sigma = 1$.
2. Monotonicity preservation. For any $\sigma \in T$, $x_\sigma \beta_\sigma \geq 0$.

These relations can be interpreted geometrically. Since there is no ambiguity, we can assume that the degrees of freedom can be numbered from 1 to n_d, an we identify the dof σ to its number ℓ in $[1,\ldots,n_d]$.

Let us consider in $\mathbb{R}^{n_d}$ n_d linearly independent points $\mathscr{S} = \{A_\ell\}_{\ell=1,\cdots,n_d}$. Note they do not have connections with any physical points in the mesh. We can introduce for any point $M \in \mathbb{R}^{n_d}$ its barycentric coordinates $\{\lambda_\ell(M)\}$ with respect to $\mathscr{S}$: $M = \sum_{\ell=1}^{n_d} \lambda_\ell(M) A_\ell$ or equivalently, for any $O \in \mathbb{R}^{n_d}$ $\mathbf{OM} = \sum_{\ell=1}^{n_d} \lambda_\ell(M)\mathbf{OA}_\ell$. We have by definition $\sum_{\ell=1}^{n_d} \lambda_\ell(M) = 1$. Thus, we can interpret $\{x_l\}$ and $\{\beta_l\}$ as the barycentric coordinates of the points L and H such that $L = \sum_{\ell=1}^{n_d} x_\ell A_\ell$ and $H = \sum_{\ell=1}^{n_d} \beta_\ell A_\ell$. The problem reduces to defining a mapping onto $\mathbb{R}^{n_d} : L \mapsto H$ such that the constraints $x_\ell \beta_\ell \geq 0$ are true : the advantage of that interpretation is that the conservation properties are automatically satisfied.

There are many solution to that problem, one particularly simple one is an extension of the PSI "limiter" of Struijs :

$$\beta_\ell = \frac{x_\ell^+}{\sum_{\ell'} x_{\ell'}^+}. \tag{16}$$

There is no singularity in the formula since $\sum_{\ell'} x_{\ell'}^+ = \sum_{\ell'} x_{\ell'} - \sum_{\ell'} x_{\ell'}^- \geq 1$. Throughout the paper, we use (16).

2.4 An Example of First Order Scheme

We describe several examples for $\operatorname{div} f(u) = 0, x \in \Omega$. Using the $\mathbb{P}^k$ interpolant in T, $u^h = \sum_{\sigma\in T} u_\sigma \psi_\sigma$, the total residual Φ^T can be written as, using the Green formula,

$$\Phi^T = \int_T \operatorname{div} f(u^h) dx = \int_T \nabla_u f(u^h)\cdot\nabla u^h \, dx = \sum_{\sigma\in T} u_\sigma \int_T \nabla_u f(u^h)\cdot\nabla\psi_\sigma dx.$$

We define $k_\sigma = \int_T \nabla_u f(u^h)\cdot\nabla\psi_\sigma dx$. The example that will be used here consists in the extension of Rusanov's scheme. It writes

$$\Phi_\sigma^T = \frac{1}{n_d}\left(\Phi^T + \alpha_T \sum_{\sigma' \in T} (u_\sigma - u_{\sigma'})\right) \tag{17}$$

and a better and simpler formulation is

$$\Phi_\sigma^T = \frac{\Phi^T}{n_d} + \alpha_T (u_\sigma - \bar{u}) \tag{18a}$$

with

$$\bar{u} = \frac{\sum_{\sigma' \in T} u_{\sigma'}}{n_d}. \tag{18b}$$

From (17), the LxF/Rusanov's residual have the form (13) with $c_{\sigma\sigma'} = \dfrac{k_\sigma - \alpha_T}{n_d}$ and $c_{\sigma\sigma'} \geq 0$ if $\alpha_T \geq \max_{\sigma \in T} |k_\sigma|$.

This scheme is extremely dissipative, but this is the one from which we start for two reasons : it is very cheap and simple to code, even for systems; it is a good candidate to demonstrate the efficiency of our approach.

Before going further, we present numerical illustrations in order to illustrate the behavior of the scheme (11)–(16).

2.5 Preliminary Numerical Results and Fixes

2.5.1 Preliminary Numerical Results

We test the scheme on two simple linear advection problems of the form

$$\lambda \cdot \nabla u = 0 \qquad \text{with inflow boundary conditions} \tag{19a}$$

on $\Omega = [0,1]^2$ with

$$\lambda = (1,1)^T \text{ and } u(x,y) = \begin{cases} 1 & \text{if } x = 0 \text{ and } y > 0 \\ 0 & \text{if } y = 0 \text{ and } x > 0 \end{cases} \tag{19b}$$

and

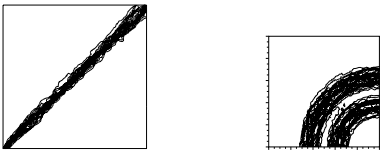
$$\lambda = (y,-x)^T \text{ and } u(x,y) = \varphi_0(x) \text{ if } x = 0 \tag{19c}$$

We have set

$$\varphi_0(x) = \begin{cases} \cos^2(2\pi x) & \text{if } x \in [0.25, 0.75] \\ 0 & \text{else} \end{cases}$$

The results are displayed in figure 1.

The behavior of the solutions is similar to what has been discussed in [8]. Indeed, the solution is non oscillatory, quite clean in the case of the problem (19a)–(19b). In the case of problem (19a)–(19c), the isolines should be circles. Instead of circles, the isolines are wiggly. This is not a manifestation of any instability : the local maximum principle is satisfied both theoretically and numerically. If one makes a



problem (19a)–(19b) problem (19a)–(19c)

Fig. 1 Results obtained for the scheme (15)==(16) for $\mathbb{P}^2$ interpolation. The first order scheme is (18).

cross–section, one can see that instead of a cos^2 profile, one has a succession of *plateaux*. In fact the scheme is much too over-compressive. The same results have been obtained with the extension of the N scheme. Another manifestation of this phenomena can be observed on the convergence history of the iterative schemes : after a very quick drop of about two orders of magnitudes, the iterative residual stagnates for ever, see [8] for example. Hence, the equation (3) is not solve exactly but with an error of the order of h, so that the formal accuracy is destroyed.

In the case of second order schemes, this wiggly behavior has been corrected by adding to the residuals (11) a term of the form

$$h_T \int_T \left(\lambda \nabla \psi_\sigma\right) \left(\lambda \nabla u^h\right) dx. \tag{20}$$

This term has a dissipative nature. Since this term is $\mathscr{O}(h^3)$ when it is evaluated for the interpolant of the exact solution, it does not destroy the formal accuracy of the scheme. In fact, it removes the spurious modes that are existing, and improves the quality of the solution. Details can be found in [8].

In the $\mathbb{P}^k$ case a similar strategy can be followed, but some difficulties arise. In the $\mathbb{P}^1$ case, the gradients are constant so that the evaluation of this integral is obvious. In the $\mathbb{P}^k$ case, the two terms $\lambda \nabla \psi_\sigma$ and $\lambda \cdot \nabla u^h$ are *a priori* polynomials of degree $k-1$. Hence, a strict application of the same method would require the exact integration of a polynomial of degree $2(k-1)\ldots$ which becomes quite expensive. In particular, the evaluation of this term would become much more important than that of the residual themselves. Let us describe a simple trick that reduces a lot the work load. A better analysis of the structure and the role of the dissipative term helps to reduce substantially the computational cost. In this section, we develop an abstract form of the scheme (10)–(16). The precise formula (16) does not play any role. We also ignore the role of boundary terms, knowing of course they have an important role, but not for the purpose of this analysis.

We first write $\Phi_\sigma^T = \Phi_\sigma^{T,c} + \Phi_\sigma^T - \Psi_\sigma^{T,c}$, multiply (3) by $\varphi(\sigma)$ and add all the equality. Using the conservation relations, (5) is equivalent to

$$\int_\Omega \varphi^h \text{ div } f(u^h) dx + \sum_T q_T(\varphi^h, u^h) = 0 \tag{21a}$$

with

$$q_T(\varphi^h,u^h)=\frac{1}{n_d!}\sum_{\sigma,\sigma'\in T}(\varphi(\sigma)-\varphi(\sigma'))\left(\beta_\sigma^T\Phi^T-\Psi_\sigma^{T,c}\right) \tag{21b}$$

The modification introduced in [8] amounts to adding to the quadratic forms q_T the term (20). The question is to know which quadratic form can be added to q_T such that the scheme is dissipative, keeps the same formal accuracy, and preserves the non oscillatory behavior of (21a).

The most natural way of proceeding is to consider a quadrature formula applied to (20), say

$$d_T(\varphi^h,u^h)=|T|\sum_{x_{\text{quad}}}\omega_{\text{quad}}\left[\left(\lambda\nabla\varphi_\sigma\right)(x_{\text{quad}})\left(\lambda\nabla u^h\right)(x_{\text{quad}})\right] \tag{22}$$

such that

$$(\varphi^h,u^h)\mapsto\int_\Omega\varphi^h\ \text{div}\ F(u^h)dx+\sum_T\left(q_T(\varphi^h,u^h)+\theta_T h_T d_T(\varphi^h,u^h)\right)$$

is dissipative. Here, h_T is a the radius of the circumscribed circle/sphere, and θ_T is a parameter that is of the order of 0 in discontinuities and 1 elsewhere.

Let us first explain the role of θ_T. Looking at figure 1, we see that the original scheme behaves very well in discontinuities. There is no reason to modify it. Thus, the optimal choice would be $\theta_T=0$ in discontinuities. This choice is not convenient elsewhere as the figure 1 again shows it.

It is complicated to find general criteria on the d_Ts to fulfill our goal. Using the compactness of the stencil, it is sufficient to ask that the quadratic form

$$(\varphi^h,u^h)\mapsto q_T(\varphi^h,u^h)+\theta_T h_T d_T(\varphi^h,u^h)$$

fulfill our goal.

The key remark is to see that, whatever the quadrature formula we use, and thanks to the term $\theta_T h_T$ in front of d_T, we see that is u^h is an interpolant of a function such that $\lambda\cdot\nabla u=0$, then

$$\left|\theta_T h_T d_T(\varphi^h,u^h)\right|\le C(u)||\nabla\varphi||\ h_T^{k+d+1},$$

so that the *formal* accuracy is not spoiled. Then, following the analysis of [8, 17], it is enough to ask that $\omega_{\text{quad}}>0$ and that the form

$$(\varphi^h,u^h)\mapsto d_T(\varphi^h,u^h)$$

is positive definite when $\lambda\cdot\nabla u^h\neq 0$. Hence, one quadrature point is enough for $k=1,3$ for $k=2,6$ for $k=3$ and so on. Note that there is no need that the "quadrature" formula be consistent with the integral $\int_T(\lambda\cdot\nabla\varphi)\ (\lambda\cdot\nabla u)dx$. We choose the

"quadrature" points so that the formula are independent of the numbering of the mesh points. In our examples, for $k=2$ we choose the vertices of T, for $k=3$ we add to these points the mid edge points: since these points are degrees of freedom, the additional cost is minimized. The weights are respectively $\omega = 1/3$ and $1/6$. In the following, we denote $\Psi_\sigma^T = |T| \sum_{x_{\text{quad}}} \omega_{\text{quad}} \left[\left(\lambda \nabla \psi_\sigma\right)(x_{\text{quad}}) \left(\lambda \nabla u^h\right)(x_{\text{quad}}) \right]$.

The last point is about the choice of the parameter θ. In [8], several good formula have been given, they all are very local and only depend on the residual and the values if u^h in T. However the best choice seems to be the following

$$\theta_T = 1 - \max_{\sigma \in T} \left[\max_{T' \ni \sigma} \left(\frac{|u_\sigma - \bar{u}_T|}{|u_\sigma| + |\bar{u}_T| + \varepsilon} \right) \right] \tag{23}$$

with ε of the order of machine zero and $\bar{u}_T = \left(\sum_{\sigma \in T} u_\sigma \right)/n_d$. Typically, $\theta = \mathcal{O}(h_T)$ in a smooth region and $\theta \equiv 1$ in a discontinuity. The relation (23) depends on values of u that do not lie in T, thus it seems that the formula is not compact. Indeed this is true, but from an algorithmic point of view, what is important is that the implementation can be made compact. The algorithm 1 show the main operations that are done (on an explicit algorithm) and show that the compactness of the method is not destroyed. This can easily be generalized to other type of iterative scheme.

Algorithm 1. Sketch of the general algorithm showing how the compactness of the algorithm is not destroyed by the evaluation of θ_T.

1: Initialize by $\theta_\sigma^0 = 1$ for all dofs.
2: **for** Do for $k = 1$ to k_{max} (maximum number of iterations) **do**
3: Set $\tilde{\theta}_{\sigma^T} = 0$ for each σ and $Res_\sigma = 0$
4: **for** For each T **do**
5: evaluate

$$\Phi_\sigma^T = \beta_\sigma^T \Phi^T + \theta_\sigma h_T \Psi_\sigma^T, \tag{24a}$$

6: evaluate

$$\xi_\sigma = \max(\tilde{\theta}_\sigma, \frac{|u_\sigma - \bar{u}_T|}{|u_\sigma| + |\bar{u}_T| + \varepsilon}) \tag{24b}$$

7: set $\tilde{\theta}_\sigma = \xi_\sigma$,
8: update

$$Res_\sigma = Res_\sigma + \Phi_\sigma^T.$$

9: **end for**
10: Swap : $\theta_\sigma = \tilde{\theta}_\sigma$,
11: Update : $u_\sigma^{n+1} = u_\sigma^n - \omega_\sigma Res_\sigma$
12: **end for**

2.6 Numerical Illustrations

To begin with, we rerun the same examples as those of figure 1. The accuracy is as expected, as shown on table 1 The orders are fitted by least square.

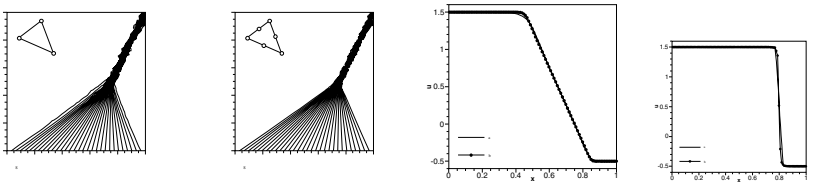
Table 1 L^2 errors for (19a)–(19b) with $u(x) = \varphi_0(x)$ on the inflow

h	$\varepsilon_{L^2}(P^1)$	$\varepsilon_{L^2}(P^2)$	$\varepsilon_{L^2}(P^3)$
1/25	0.50493E-02	0.32612E-04	0.12071E-05
1/50	0.14684E-02	0.48741E-05	0.90642E-07
1/75	0.74684E-03	0.13334E-05	0.16245E-07
1/100	0.41019E-03	0.66019E-06	0.53860E-08
	$\mathscr{O}^{\mathbf{ls}}_{L^2}$ =**1.790**	$\mathscr{O}^{\mathbf{ls}}_{L^2}$ =**2.848**	$\mathscr{O}^{\mathbf{ls}}_{L^2}$ =**3.920**

The method also works well for non linear problems. To give an example, let us consider the Burgers's equation

$$\frac{\partial u}{\partial y} + \frac{1}{2}\frac{\partial u^2}{\partial x} = 0 \qquad \text{if } x \in [0,1]^2$$
$$u(x,y) = 1.5 - 2x \text{ on the boundary.}$$

The exact solution consists in a fan that merges into a shock which foot is located at $(x,y) = (3/4, 1/2)$. Some results are displayed on figure 2. They are obtained on the same mesh as for the previous example. For the sake of comparison, we give the second and third order results on the same mesh (hence the $\mathbb{P}^2$ results have more degrees of freedom). There are no spurious oscillation accros the shock. The resolution of the fan is better also.



$\mathbb{P}^1$ $\mathbb{P}^2$ cut at $y = 0.3$ at $y = 0.6$

Fig. 2 Burger equation, solution obtained with a $\mathbb{P}^1$ and $\mathbb{P}^2$ lagrange interpolant and the scheme

3 The System Case

In this section, we describe the scheme for the system of the steady Euler equations described by (1) with the flux (2) and the conserved variables $\mathbf{u} = (\rho, \rho\mathbf{u}, E)^T$.

We assume a perfect gas equation of state, and $\gamma = 1.4$ in the applications. We denote by A (resp. B) the Jacobian matrix of the flux f_1 (resp. f_2) with respect to the state $\mathbf{u}$.

The scheme is a direct extension of what is done in the scalar case, with a major modification because the natural unknown is a vector, not a scalar. We follow essentially the procedure of [14, 8]. We provide the details of the scheme description on a single element T since there is no ambiguity.

3.1 The First Order Scheme

The first order scheme is constructed on the Lax–Friedrichs scheme, i.e., for any degree of freedom $\sigma \in T$,

$$\Phi_\sigma = \frac{1}{N} \oint_{\partial T} \mathbf{f}(\mathbf{u}^h) \cdot \mathbf{n} dl + \alpha_T (\mathbf{u}^h - \bar{\mathbf{u}}). \tag{25}$$

Here N is the number of degrees of freedom in T, hence $N = 6$. The total residual $\oint_{\partial T} \mathbf{f}(\mathbf{u}^h) \cdot \mathbf{n} dl$ is evaluated by Simpson formula: if $\Gamma = [a,b]$ is an edge of T and $c = \frac{a+b}{2}$, we set

$$\int_\Gamma f(x) dl \equiv \frac{1}{6} \Big(f(a) + 4f(c) + f(b) \Big),$$

which amounts, in our case, to use a quadratic interpolant of $\mathbf{f}$ in T. The average state is $\bar{\mathbf{u}} = \sum_{\sigma \in T} \mathbf{u}_\sigma / N$, and α_T is larger than the spectral radius of the flux Jacobians at the degrees of freedom. In practice, it is set to twice this maximum.

3.2 Controlling the Oscillations

In the scalar part, the control of oscillations is achieved by "limiting" the ratios Φ_σ / Φ. In the system case, this quantity has no meaning. Hence, we adapt the procedure presented in [14]. Using the average state $\bar{\mathbf{u}}$, we compute the average flow direction, i.e. $\bar{\mathbf{n}} = \dfrac{\bar{\mathbf{u}}}{||\bar{\mathbf{u}}||} = (n_1, n_2)$. Then we evaluate the Jacobian matrix

$$K_{\bar{\mathbf{n}}} = A(\bar{\mathbf{u}}) n_1 + B(\bar{\mathbf{u}}) n_2 \tag{26}$$

which is diagonalizable in $\mathbb{R}$. The eigenvectors are r_p for $p = 1, \cdots, 4$ associated to the eigenvalues

$$\lambda_{1,2} = \bar{\mathbf{u}} \cdot \bar{\mathbf{n}} = ||\bar{\mathbf{u}}||, \lambda_3 = ||\bar{\mathbf{u}}|| - \bar{c}, \lambda_4 = ||\bar{\mathbf{u}}|| + \bar{c}.$$

Last, we denote by ℓ_p the right eigenvectors of the system, i.e. the linear forms such that any state vector $X \in \mathbb{R}^4$ can be decomposed as $X = \sum_{p=1}^4 \ell_p(X) r_p$. Our method is then the following:

1. We decompose the first order residuals Φ_σ into "characteristic" residuals, for each $\sigma, \Phi_\sigma = \sum_{p=1}^{4} \ell_p(\Phi_\sigma) r_p$. We denote the characteristic residual by $\varphi_\sigma^p = \ell_p(\Phi_\sigma)$, they satisfy the conservation relation: for each $p \in \{1, \cdots, 4\}$, $\sum_{\sigma \in T} \varphi_\sigma^p = \ell_p(\Phi) := \varphi^p$
2. For any $p = 1, \cdots, 4$, we "limit" the characteristic sub-residual by the same procedure as in the scalar case: $\beta_\sigma^p := \left(\frac{\varphi_\sigma^p}{\varphi^p}\right)^+ / \sum_{\sigma' \in T} \left(\frac{\varphi_{\sigma'}^p}{\varphi^p}\right)^+$
3. We construct the limited residual by

$$\Phi_\sigma^\star := \sum_{p=1}^{4} \beta_\sigma^p \varphi^p r_p. \tag{27}$$

The property (9) is satisfied in a suitable norm. Indeed, if A_0 denotes the Hessian of the mathematical entropy evaluated at $\bar{\mathbf{u}}$, we know that we can find a set of eigenvectors r_l that are are orthogonal for the metric defined by A_0. Indeed, if we denote by $(\,.\,,\,.\,)_{A_0}$ the scalar product associated to A_0. We have $\varphi_\sigma^p = (r_p, \Phi_\sigma)_{A_0}$ so that $||\Phi_\sigma^\star||_{A_0}^2 = \sum_p |\beta_\sigma^p|^2 |; |\varphi^p|^2 \le \sum_p |\varphi^p|^2 \le ||\Phi||_{A_0}^2$ where of course we have assumed that the eigenbasis $\{r_p\}$ is orthogonormal.

The matrix A_0 is not uniform, but we can nevertheless state that if the conserved state is such that the density and the pressure are bounded from above and below, all the norms defined by the $A_0(\bar{\mathbf{u}})$ are equivalent and the LP property is uniformly satisfied.

The last step is to get an explicit form of such a basis. The standard eigenvectors of the Euler equations are simple and good candidates for that since it can easily be shown that they are orthogonal for the quadratic form defined by the entropy. Hence, this is our practical choice. They are evaluated as the eigenvectors of (26) where $\mathbf{n}$ is the normalized averaged velocity. It the case of a stagnation point, we choose the x direction.

As for the scalar case, the scheme (7b)–(27) produces very good results in discontinuous regions, and very poor one in the smooth parts of the flow. Indeed the iterative convergence is very poor, and the results are at most first order accurate.

To the same problem, we use the same cure. We add to $(\Phi_\sigma)^\star$ a correction of the type

$$h_T \int_T \left((A,B)\cdot\nabla\varphi_\sigma\right) \tau \left((A,B)\cdot\nabla\mathbf{u}\right) dx \tag{28}$$

where the matrix τ is a scaling matrix.

Several choices have been tested. A good choice seems to be $\tau = h_T^{-1} N$ where the N matrix is $N = \left(\sum_{i=1}^{3} K_i^+\right)^{-1} = 2\left(\sum_{i=1}^{3} |K_i|\right)^{-1}$. In this relation, the summation is on the three vertices of the triangle The Jacobian matrix are evaluated at the averaged state $\bar{\rho} = \frac{1}{6}\sum_{\sigma\in T} \rho_\sigma$, $\overline{\mathbf{u}} = \frac{1}{6}\sum_{\sigma\in T} \mathbf{u}_\sigma$, $\bar{p} = \frac{1}{6}\sum_{\sigma\in T} p_\sigma$. It is shown in [5] that if the velocity $\overline{\mathbf{u}} \neq 0$, the matrix $\sum_{i=1}^{3} K_i^+$ is always invertible. To avoid this situation, we slightly

modify the eigenvalues λ^+ using Harten's entropy fix. The second thing we do is to simplify the expression (28) in the spirit of (22). Namely, the residual is

$$\Phi_\sigma^{\star\star} = \Phi_\sigma^\star + \theta_T \Psi_\sigma \tag{29a}$$

with

$$\Psi_\sigma = |T| \sum_{x_{\text{quad}}} \omega_{\text{quad}} \left[\left((A,B)(x_{\text{quad}}) \nabla \psi_\sigma \right)(x_{\text{quad}}) N \left((A,B)(x_{\text{quad}}) \nabla u^h \right)(x_{\text{quad}}) \right]. \tag{29b}$$

As in the scalar case, we have taken the smallest set of quadrature points, i.e. the vertices of T. This also simplifies the evaluation of the Jacobian matrices. The last feature is the parameter θ_T. It has to be of the order of unity in the smooth regions, and of the order of zero when the gradient of the solution is large. In the numerical experiments, we have chosen a sensor the entropy $\theta_T = 1 - \max_{\sigma\in T} \max_{T',\sigma\in T'} \max_{\sigma'\in T'} \frac{|s_{\sigma'} - \overline{s}_{T'}|}{s_{\sigma'} + \overline{s}_{T'}}$. where $s = p\rho^{-\gamma}$ (in order to have a positive quantity). Conservation is guarantied automatically.

3.3 Boundary Conditions

We have used a simplified version of the boundary conditions. If an element T has an edge, Γ_T, on the boundary, we need to add to the degrees of freedom on Γ_T a boundary residual. We denote it by $\Phi_\sigma^{\Gamma_T}$. These residuals should satisfy the conservation relation $\sum_{\sigma\in\Gamma_T} \Phi_\sigma^{\Gamma_T} = \int_{\Gamma_\sigma} \left(\mathscr{F}_n(u^h) - f(u^h)\cdot\mathbf{n} \right) dl$ where $\mathscr{F}_n$ is a boundary flux. In the examples of this paper, two types of boundary are considered:

- Wall boundary where $\mathbf{u}\cdot\mathbf{n} = 0$ is weakly imposed so that $\mathscr{F}_n(u^h) = \left(0, p(u^h)n_x, p(u_h)n_y, 0\right)^T$
- Inflow/outflow boundary conditions. The state at infinity is U_∞ and we take here the modified Steger-Warming flux

$$\mathscr{F}_n(u^h) = \left(A(u^h)\cdot\mathbf{n}\right)^+ u^h + \left(A(u^h)\cdot\mathbf{n}\right)^+ u_\infty.$$

By analogy with what is done in [5], we have chosen a 'centered' version of the boundary residuals, namely

$$\Phi_\sigma^{\Gamma_T} = \int_{\Gamma_T} \left(\mathscr{F}_n(u^h) - f(u^h)\cdot\mathbf{n} \right) \psi_\sigma(x) dl$$

where again ψ_σ is the Lagrange basis function defined in T for σ. This is approximated by a quadrature formula with positive weights. The quadrature formula should be of order $k+d-1$, i.e. 3 for a third order scheme in 2D. The actual residual is

$$\Phi_\sigma^{\Gamma_T} = |\Gamma_T| \sum_{\text{quadrature points}} \omega_{quad} \left(\mathscr{F}_n(u^h) - f(u^h) \right)(x_{\text{quad}})\cdot\mathbf{n}.$$

In the case of interest ($\mathbb{P}^2$ interpolation), we approximate these relation with Simpson's formula : only one term appears in the sum.

3.4 Numerical Results

3.4.1 A Convection Problem

Our first example is the solution of the Euler system on $[0,1]\times[0,1]$ with the following inflow condition at $y=0$, $x=1$ and $x=0$: $\rho = 2+\sin(\pi x), u = 10, p = 1$. The flow is assumed to be supersonic at $y=1$. The exact solution is $\rho(x,y) = 2+\sin(\pi x), u(x,y) = 10, p(x,y) = 1$ the problem is a simple convection one. However, we use the full Euler system and the scheme developped above to compute the solution. The scheme is really third order accurate as it can be seen from the L^2 errors on Figure 3.

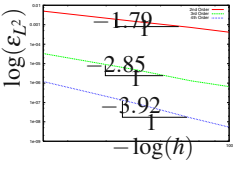


Fig. 3 L^2 error for the second, third order and fourth order version of the scheme

3.4.2 Two NACA0012 Airfoils Cases

Our two examples are the flow over a NACA012 airfoil. The first one, which is subsonic, has the following conditions at infinity: $M = 0.5,$ Angle of attack 2° and is referred as MTC Case 1. The second one, that is transonic, has the following conditions at infinity: $M = 0.8,$ Angle of attack 1.25° and is referred as MTC case 2. In both cases, the mesh has 10959 points and 21591. This corresponds to 43509 degrees of freedom.

- Subsonic test case. The isolines of the Mach number isolines, density, pressure and entropy are displayed in figure 4, for the first test case. On figure Figure 4, we have displayed the Mach number, the pressure coefficients en relative entropy deviation for the third order versions of the scheme.
- Transonic test case. The isolines of the Mach number isolines, density, pressure and entropy are displayed in figure for the first test case. On Figure 6, we have displayed the Mach number, the pressure coefficients en relative entropy deviation for the third order version of the scheme.

  The solutions are fine. Note however a non physical overshoot in the entropy accros the upper shock. If we compare th second order solution run with a mesh constructed from the mesh we have used where the element would have been subtriangulated so that we would have had the same number of degrees of freedom

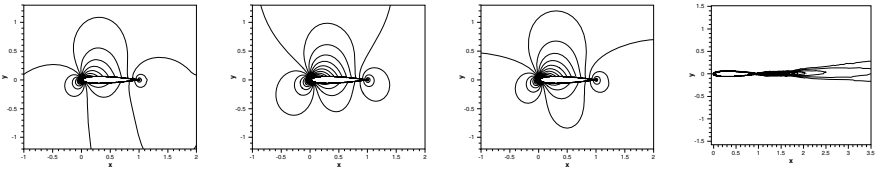


Fig. 4 Isolines of the Mach number, pressure, density and entropy for the cases 1

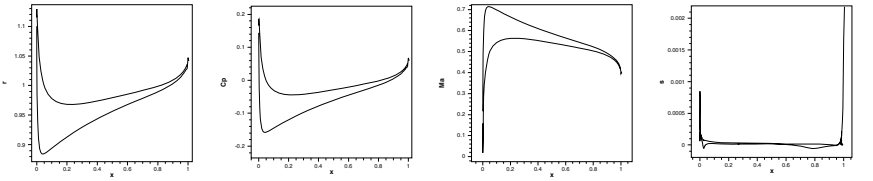


Fig. 5 Plots on the profile, subsonic test case

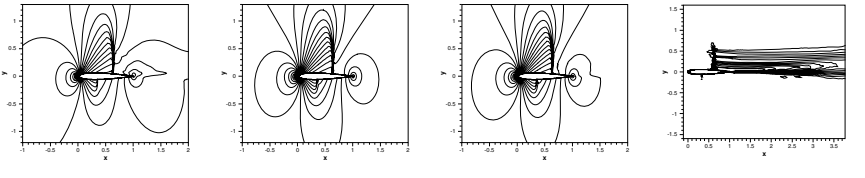


Fig. 6 Isolines of the Mach number, pressure, density and entropy for the cases 2

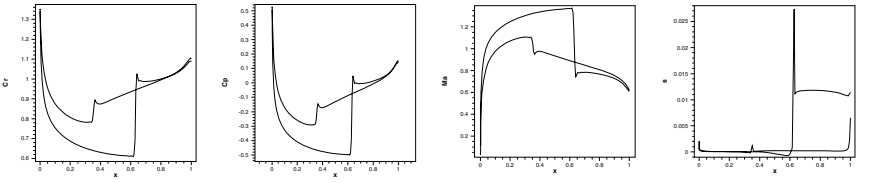


Fig. 7 Plots on the profile, subsonic test case

(not shown), we would see an excellent agreement between the solutions with a main difference however. In both cases, the shock with is one element, but one element for the third order solution is roughly twice as large as an element for the second order one. Hence, the shock look more diffused in the third order case. However, the entropy levels are much lower, as we could see in the two sphere subsonic case.

4 The Viscous Case

The viscous terms are approximated by the standard Galerkin method. We yet consider a viscous flow around a NACA012 airfoil. The flow parameters are the following: Incidence: 0° of incidence; Mach: $Ma = 0.5$; Reynolds: $Re = 500$. This test case is known to be steady. We have run second and third order computations on 8 different meshes containing between 609 and 230×10^3 vertices. On Figure 8 are represented the horizontal velocity in color and the density isolines at third order for the finest mesh. We see that the global shape of the solution is the one expected, with the boundary layer around the airfoil and its wake. Now, because the incidence is null, the lift coefficient should be zero, but because the mesh is not symmetric, the numerical value of the lift coefficient is non zero. And it should converge to zero with the right order of convergence when the mesh gets finer. We have represented the value of the computed lift coefficients at steady state with respect to $h = \sqrt{\#\{\text{DoF}s\}}$ on Figure 9. Except for one strange value at second order for the

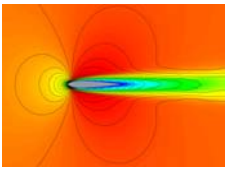


Fig. 8 Third order solution on the finest mesh for the steady viscous NACA012 test case. x-velocity in color and isolines of the density component.

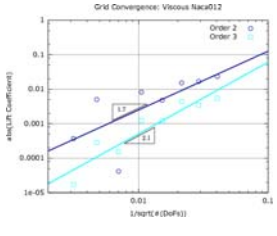


Fig. 9 Convergence of the lift coefficient with respect to the mesh characteristic size $h = \sqrt{\#\{\text{DoF}s\}}$ for 2^{nd} and 3^{rd} order simulation of the viscous NACA012 problem

6^{th} mesh, all the second order estimated lift coefficients are larger in absolute value than their associated third order lift coefficients. Furthermore, the slope of the least square line is larger for the 3^{rd} order simulation than for the 2^{nd} order one. This means the third order scheme is doing a better job for viscous simulation. But on the other hand the slope is not the one expected. If 1.7 is a good result for the expected slope 2, 2.1 is a bit far from the slope 3 expected and it is clear that the convergence is not regular at all. The mix between the residual formulation of the advective term and the Galerkin treatment of the second order diffusive term does not seem to provide the right convergence rate. This might be explained by looking at the variational formulation of the problem. However, the final result is not so bad, because the mesh convergence is still acceptable and the solution is pretty nice.

5 Extension to 3D

The extension to 3D does not involve any conceptual effort. We have run the BTC0 test cases, for the Euler and Laminar viscous problem. The characteristics of the flows are as follows :

- Euler BTC0 : $M_\infty = 0.5$, incidence=0.5°,
- Laminar BTC0 :$M_\infty = 0.5$, incidence=0.5°, $Re = 500$, Neuman conditions on the boundary.

The figure 10 represents the density isolines obtained for the second and third order scheme. The second order version uses two meshes, one with 176538 vertices and 1022928 tetrahedrons (designed by VKI), and a fine mesh that is obtained by refining regularly each tet using the mid points of the edges. Hence this mesh has exactly the same degrees of freedom than those used for the third order scheme. The calculation has been done on a 16 processor machine.

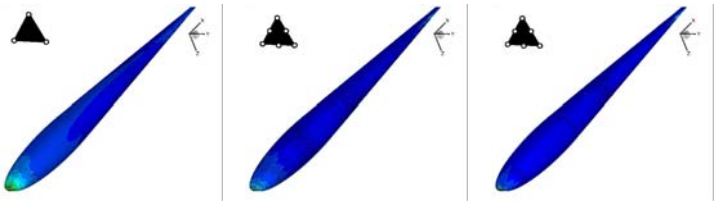


Fig. 10 Density isolines and entropy (in color) on the body for the second order scheme on the coarse and a fine mesh with the same degrees of freedom than the third order result

The figure 11 provides *preliminary* results obtained on the laminar BTC0 test case. The mesh has been designed using GmSh. The 3D results are not fully satisfactory, especially near the boundary. In our opinion, one of the reasons is that the

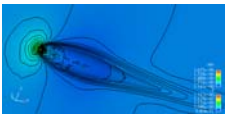


Fig. 11 Laminar BTC0 test case: the pressure is displayed in color, the density with isolines

mesh does not represent correctly the boundary geometry since we have no meshes with curve elements.

6 Conclusion

We have described a systematic way of construction high order Residual distribution schemes fo unstructured meshes. The construction use a continuous representation of data. From this information, we evaluate a total fluctuation and a high order mechanism for distribution these residual to the degrees of freedom is proposed and evaluated.

The accuracy of the scheme is evaluated on scalar problems and subsonic flows where it is known that the entropy stays uniform. The deviation of this quantity is a simple manner to check the accuracy. We have shown that for a given number of degree of freedom, our third order scheme is more accurate that a second order version of the scheme.

The robustness of the method has been evaluated on subsonic, transonic, supersonic problems, as well as on complex configurations where many flow interaction occur.

The extension to the Navier Stokes equations and to unsteady problem has also been considered. Several numerical tests seem to indicate that the solution has a correct behavior. However, it is not clear at all (and we believe this is not the case) that the scheme has a residual structure, so that the formal accuracy is questionable. A deeper investigation is currently beeing done, in a separate project.

References

1. Abgrall, R., Marpeau, F.: Residual distribution schemes on quadrilateral meshes. J. Sci. Comput. 30(1), 131–175 (2007)
2. Struijs, R., Deconinck, H., Roe, P.L.: Fluctuation splitting schemes for the 2D Euler equations. VKI-LS 1991-01, Computational Fluid Dynamics (1991)
3. Deconinck, H., Roe, P.L., Struijs, R.: A multidimensional generalization of Roe's difference splitter for the Euler equations. Computer and Fluids 22(2/3), 215–222 (1993)
4. van der Weide, E., Deconinck, H.: Positive matrix distribution schemes for hyperbolic systems. In: Computational Fluid Dynamics, pp. 747–753. Wiley, New York (1996)
5. Abgrall, R.: Toward the ultimate conservative scheme: Following the quest. J. Comput. Phys. 167(2), 277–315 (2001)

6. De Palma, P., Pascazio, G., Rossiello, G., Napolitano, M.: A second-order-accurate monotone implicit fluctuation splitting scheme for unsteady problems. J. Comput. Phys. 208(1), 1–33 (2005)
7. Rossiello, G., De Palma, P., Pascazio, G., Napolitano, M.: Third-order-accurate fluctuation splitting schemes for unsteady hyperbolic problems. J. Comput. Phys. 222(1), 332–352 (2007)
8. Abgrall, R.: Essentially non-oscillatory residual distribution schemes for hyperbolic problems. J. Comput. Phys. 214(2), 773–808 (2006)
9. Corre, C., Hanss, G., Lerat, A.: A residual-based compact scheme for the unsteady compressible Navier-Stokes equations. Comput. Fluids 34(4-5), 561–580 (2005)
10. Abgrall, R., Tavé, C.: Construction of very high order residual distribution schemes. In: Deconinck, H., Dick, E. (eds.) Preceedings of the ICCFD6, Ghent, Belgium, July 2006. Springer, Heidelberg (2006)
11. Larat, A., Abgrall, R., Ricchiuto, M.: Very high order residual distribution schemes for steady flow problems. In: Proceedings of the ICCDF8, Seoul, July 2008. Springer, Heidelberg (2008)
12. Caraeni, D., Fuchs, L.: Compact third–order multidimensional upwind scheme for Navier Stokes simulations. Theoretical and Computational Fluid Dynamics 15, 373–401 (2002)
13. Abgrall, R., Roe, P.L.: High-order fluctuation schemes on triangular meshes. J. Sci. Comput. 19(1-3), 3–36 (2003)
14. Abgrall, R., Mezine, M.: Construction of second-order accurate monotone and stable residual distribution schemes for steady problems. J. Comput. Phys. 195(2), 474–507 (2004)
15. Roe, P.L.: Fluctuations and signals - a framework for numerical evolution problems. In: Proc. Conf., Numerical methods for fluid dynamics, Reading, U.K., pp. 219–257 (1982)
16. Csík, Á., Ricchiuto, M., Deconinck, H.: A conservative formulation of the multidimensional upwind residual distribution schemes for general nonlinear conservation laws. J. Comput. Phys. 179(2), 286–312 (2002)
17. Abgrall, R., Ricchiutto, M., Larat, A.: A simple construction of very high order non oscillatory compact schemes on unstructured meshes. In: Computers and Fluids (2008) (in press)
18. Csík, Á., Ricchiuto, M., Deconinck, H.: A conservative formulation of the multidimensional upwind residual distribution schemes for general nonlinear conservation laws. J. Comput. Phys. 179(1), 286–312 (2002)

Chapter 10
High Order Residual Distribution Schemes Based on Multidimensional Upwinding

N. Villedieu, T. Quintino, M. Vymazal, and H. Deconinck

Abstract. We present an extension of the multidimensional upwind distributive schemes to high order solution spaces. We look into different high-order discretization issues such as: quadratic and cubic boundary curvature; monotonicity of the schemes in presence of solutions with discontinuities; discretisation of temporal terms for unsteady applications and discretization of diffusive fluxes. Results of test cases representative of all these issues are presented.

1 Introduction

We present the design of high order upwind schemes for systems of steady hyperbolic conservation laws given by the form

$$\nabla \cdot F(\mathbf{U}) = 0 \quad \forall (x,y) \in \Omega \tag{1}$$

where $\mathbf{U}$ is the m-vector of the conserved quantities, and $\mathbf{F}$ is a $m \times 2$-tensor: $\mathbf{F} = (\mathbf{f}^{\mathbf{x}}, \mathbf{f}^{\mathbf{y}})$($\mathbf{f}^{\mathbf{x}}$ and $\mathbf{f}^{\mathbf{y}}$ being m-vectors).

1.1 Generalities and Notations

For a given domain Ω we denote by τ_h a generic triangulation of Ω composed of a set of non-overlapping triangles $T \in \tau_h$. The elements of this triangulation are P^k finite elements having $M = (k+1)(k+2)/2$ degrees of freedom, as plotted on figure 1. Moreover, we sub-triangulate each triangle with a P^1 conformal triangulation. This means that $\forall k \geq 1$ on a given $T \in \tau_h$ we introduce $N = k^2$ sub-elements that we denote by $\{T_s\}_{s=1,N}$.

N. Villedieu · T. Quintino · M. Vymazal · H. Deconinck
Von Karman Institute, Chausse de Waterloo, 72, 1640 Rhode-St-Gense Belgium
e-mail: villedie@vki.ac.be, quintino@vki.ac.be, vymazal@vki.ac.be, deconinck@vki.ac.be

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 129–143, 2010.
springerlink.com

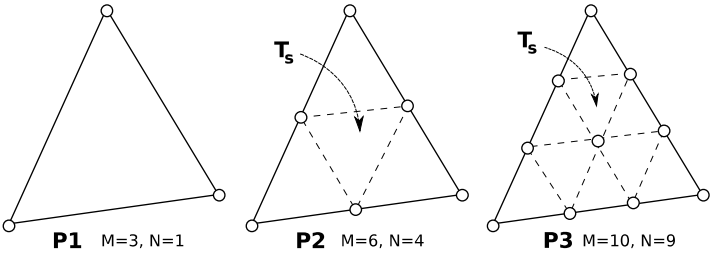


Fig. 1 Sub-triangulations of Lagrangian elements

The solution is approximated by: $\mathbf{U}^h = \sum_{i \in \tau_h} \psi_i^{P^k} \mathbf{U}_i$, where $\psi_i^{P^k}$, the basis function of node i, is a continuous piecewise polynomial of order k and $\mathbf{U}_i$ is defined by $\mathbf{U}_i = \mathbf{U}^h(x_i, y_i)$. On each $T \in \tau_h$ we also consider the set of vectors $\{\mathbf{n}_j\}_{j \in T}$, defined

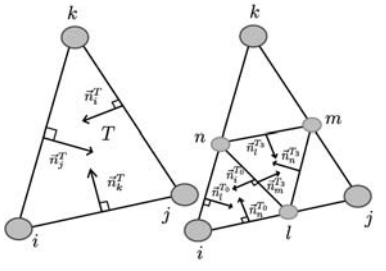


Fig. 2 Definition of the inward normals $\mathbf{n}_i$

by the inward normals to the edges of T facing each node $j \in T$. In the general P^k case, with $k > 1$, we will assume that the $\mathbf{n}_j$'s are defined on a local sub-element $T_s \in T$, as shown on figure 2 for the case $k = 2$. Finally, on each sub-element we define the following upwind parameters: $K_j = \frac{1}{2} \frac{\partial \mathbf{F}(\mathbf{U}^*)}{\partial \mathbf{U}} \cdot \mathbf{n}_i$ where $\mathbf{U}^*$ is a suitable arbitrary average of $\mathbf{U}$.

The residual distribution scheme consists of three steps:

1. Computation of the residual on each sub-element: $\phi^{T_s} = \int_{T_s} \nabla \cdot \mathbf{F} \, d\Omega = \oint_{\partial T_s} \mathbf{F} \, d\mathbf{l}$
2. Distribution of the residual to each node of the sub-element: $\phi_i^{T_s} = B_i^{T_s} \phi^{T_s}$
3. Resolution of a system of nodal equations: $\sum_{i, \in T_s} \phi_i^{T_s} = 0$

This system is solved implicitly or explicitly by pseudo-time (τ) iterations:
$\frac{\partial \mathbf{U}}{\partial \tau} - \sum_{i \in T_s} \phi_i^{T_s} = 0.$

In the hyperbolic case the schemes we consider are a particular case of the residual distribution (*RD*) schemes introduced by Abgrall [3].

1.2 Linear Schemes

1.2.1 LDA Scheme

The first linear scheme is LDA. This scheme is multidimensionally upwind[1] and order preserving[2]. Its distribution coefficient is defined by: $B_i^{T_s,\mathrm{LDA}(P^k)} = K_i^+ (\sum_{j\in T_s} K_j^+)^{-1}$ where $K_i^\pm = R\Lambda^\pm R^{-1}$ where R_i is the matrix of the right eigenvectors of K_i and $\Lambda_i^\pm$ the diagonal matrix of the positive (resp. negative) part of its eigenvalues.

1.2.2 N Scheme

The N scheme is 1^{st} order, monotone[3] when using P^1 elements and upwind. We directly define the distributed residual:

$$\phi_i^{\mathrm{N}(P^k)} = B_i^{\mathrm{LDA}(P^k)} \phi^{T_s} + d_i^{\mathrm{N},T_s} \quad \text{with} \quad d_i^{\mathrm{N},T_s} = \sum_{j\in T_s} K_i^+ N K_j^+ (\mathbf{U}_i - \mathbf{U}_j) \tag{3}$$

where d_i^{N,T_s} is a cross-wind dissipation involving only the nodes of the sub-elements whereas ϕ^{T_s} involves all the nodes of the element. That is the reason why the scheme is not rigorously monotone, but is quasi non-oscillatory.

2 How to Treat Curvature Issues?

Until now, we have considered high order discretisation only for the solution and from the geometrical point of view, the elements are linear. This causes some problems when we consider curved geometries as a NACA-0012. In this case all the mid-points of the faces lying on the airfoil will not be on the body. This creates some wiggles in the solution as shown on figure 3. On this figure, we consider a subsonic flow ($M = 0.5, \alpha = 2^\circ$) over a NACA-0012 and we look at the distribution

[1] A scheme is multidimensional upwind if $B_i = 0$ when $K_i \leq 0$.

[2] Order preserving and Accuracy: In the steady case the condition to get $k+1$-th order schemes is that (see [3] for details)

$$\Phi_j^{T_s} = \mathscr{O}(h^{k+2}) \tag{2}$$

For the k-th degree polynomial approximation (1) we get $\Phi^{K_s} = \mathscr{O}(h^{k+2})$, hence the accuracy condition is also expressed by $\Phi_j^{K_s} = \mathscr{O}(\Phi^{K_s})$ meaning that the distribution coefficient should be bounded (Order preserving condition).

[3] Monotonicity: The rigorous definition of monotonicity for RD schemes resorts to the theory of positive coefficients, see [3, 5] for details. In this paper we will define a scheme as being monotone if, in practical computations, it gives a non-oscillatory approximations of discontinuities. In particular, we are interested in schemes for which, across a discontinuity, $\Phi_j^{T_s} \times \Phi_j^{\mathrm{M}} \geq 0$, for some first order monotone splitting Φ_j^{M}.

of Cp on the NACA-0012. The first test was done using LDA(P^2). If we zoom in on the maximum of Cp, there are a lot of oscillations.

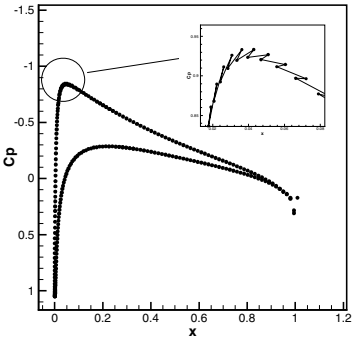


Fig. 3 Approximation of curved geometries by linear elements

2.1 Krivodonova Methodology Applied to RDS

To solve this problem we first follow the approach of Krivodonova [1]. In this article the author explains that if one uses the exact normals of the geometry in the integration points used to compute the corrective flux at the wall, then the results are improved.

The method is an intermediate between the use of curved elements and the use of P^1P^2 elements. Normally, when using a P^1P^2 element, the normal used to correct the flux at the wall is the one of the element and it is the same for all the Gauss points as illustrated on figure 4. However, for a P^2P^2 element the normals are defined per Gauss points (figure 4). Here, we still use a straight element but we use the normals of the geometry to compute the correction flux (figure 4).

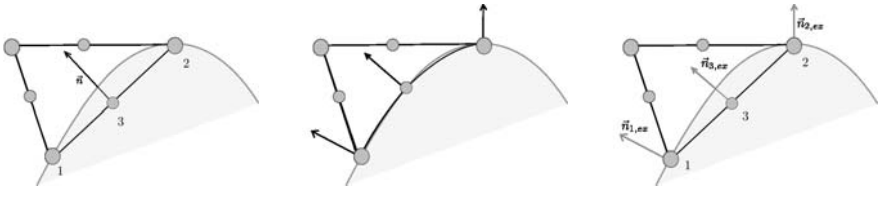


Fig. 4 Approximation of curved geometries P^1P^2 face (left), P^2P^2 face (middle), P^1P^2 face using exact normals (right)

2.2 P2P2 Elements

In this strategy, we use a P^2 discretisation both for the geometry and the solution. The geometry of the curvilinear P^2 element is defined by the following isoparametric transformation to a parent element in ξ–η space (as shown in figure 5):

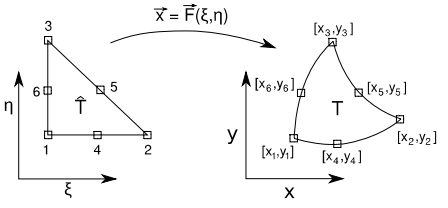


Fig. 5 Transformation of triangle geometry from reference to physical space using quadratic Lagrange shape functions

$$x(\xi,\eta) = \sum_{i=1}^{6} \psi_i^{h,2}(\xi,\eta)\cdot x_i \qquad y(\xi,\eta) = \sum_{i=1}^{6} \psi_i^{h,2}(\xi,\eta)\cdot y_i \tag{4}$$

where x_i and y_i are the known coordinates of the 6 nodes defined in physical space. We can construct a mapping that transforms a face of the reference triangle in ξ–η space to a face of a curvilinear triangle in x–y space (eq. (4)).

Since each shape function vanishes on a complete edge in ξ–η space, the boundaries of the triangle (defined by $\xi = 0$, $\eta = 0$ and $\xi + \eta = 1$ respectively) are quadratic functions of one parameter ζ:

$$\begin{aligned} x &= x(\zeta) \\ y &= y(\zeta) \end{aligned} \tag{5}$$

The sub-element residual $\phi^{\mathbf{T_s}}$ is computed by a contour integral:

$$\phi^{T_s} = \oint_{\partial T_s} \mathbf{F}(\mathbf{U}^{h,2})\cdot\hat{n}\,dl \tag{6}$$

The boundary ∂T_s is described by the transformation (4, 5). We integrate the fluxes in (6) numerically using Gauss quadrature, 3-points per face f of the sub-element as

$$\oint_f \mathbf{F}(\mathbf{U}^{h,2})\cdot\hat{n}\,dl \simeq \sum_{q=1}^{N} w_q[\mathbf{F}(\mathbf{U}_q)\cdot\hat{n}_q]J_q, \tag{7}$$

$$J_q = \sqrt{\dot{x}^2(\zeta_q) + \dot{y}^2(\zeta_q)} \tag{8}$$

where J_q is the Jacobian of transformation (5) at quadrature point q. We use the transformation also to compute the normal n_q.

2.3 Subsonic Flow over a NACA-0012 Airfoil

We compare entropy generated by the LDA scheme in an inviscid subsonic flow over an airfoil. The freestream Mach number is Ma = 0.5 and the angle of attack was set to $\alpha = 2°$. Figure (6) shows entropy iso-lines in the interval $\langle 11.200, 11.290 \rangle$ plotted with step $\Delta = 0.002$. We compare the results obtained when using P^1P^2 elements, P^1P^2 elements with Krivodonova approach for the wall boundary conditions and P^2P^2 elements. We can see that with simple P^1P^2 elements, lot of entropy is created all around the airfoil. When using Krivodonova approach, the entropy deviation is reduced but there is still an important layer at the trailing edge. Finally, the optimal result is obtained with P^2P^2 elements.

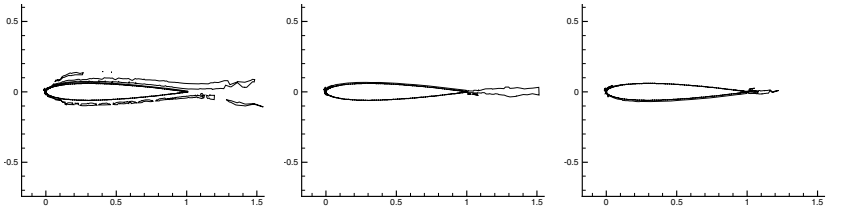


Fig. 6 Entropy iso-lines for different boundary discretizations. From left to right: P^1P^2 elements, P^1P^2 with Krivodonova boundary treatment and P^2P^2 elements.

3 Shock Capturing

3.1 RDS Shock-Capturing via Blending Schemes

To combine monotonicity and high order accuracy, the residual distribution method, in accordance to the Godunov theorem [4], requires a non-linear high-order scheme to be constructed. One way to construct such scheme is to blend the N scheme (1^{st} order monotone) with the LDA scheme (high order scheme) to obtain the residual in the following form:

$$\phi_i^B = \theta \phi_i^N + (1-\theta)\phi_i^{LDA} \tag{9}$$

This scheme relies on a good shock detection to properly capture the shock and obtain both accuracy and monotonicity. There are many choices of different detectors. Within this work, we will focus on an improved version of the Dobes shock-detector, particular for the system of Euler equations, and introduced by Bonanni [2]. It is based on the pressure gradients of the solution, and its definition is:

$$\theta = \min(1, sc_{\text{Bon}}{}^2 h), \quad sc_{\text{Bon}} = \left(\frac{\left(\int_{T_s} \nabla p \, dx \right) \cdot \mathbf{v}}{\delta_{pv^2} |T_s|} ||\mathbf{v}|| \right)^+ \tag{10}$$

where p is the static pressure, $\delta_{pv^2} = (p_{max} - p_{min})\bar{v}^2$ is a global pressure variation scale multiplied by the square of the magnitude of the mean velocity in the domain. This sensor is positive in a shock and compression, vanishes in expansions and is of order $\mathcal{O}(1)$ in discontinuous regions. The shock detection is illustrated on

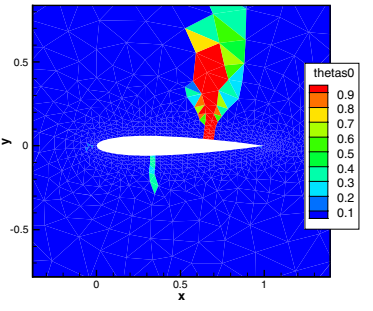


Fig. 7 Transonic flow over a NACA-0012: Shock detector repartition on the airfoil

figure 7 where we plotted the repartition of θ for a transonic flow over a NACA-0012 ($M = 0.8, \alpha = 1.25°$). We used linear discretisation of the solution to do this simulation.

3.2 Transonic Flow over a NACA-0012 Airfoil

In this section we test the monotonicity and the accuracy of the Bx scheme. We choose the transonic flow used to test the shock detector. We compare Bx(P^1) and Bx(P^2) on meshes having the same number of degrees of freedom (3820 DOF). On figure 8, we plot the Mach iso-lines. The solution of the Bx(P^1) is behaving as a monotone scheme. Bx(P^2) shows few spurious oscillations that are advected until the trailing edge. This phenomenon was expected since this scheme is not rigorously monotone. However the better accuracy of the quadratic discretisation is visible on figure 8. Indeed, the weakest shock is better resolved when we increase the order of the discretisation.

Finally, we want to check that blending with a low order scheme does not imply an increase of entropy at the leading edge. On figure 9 we can see that even if with Bx(P^1) there is a slight increase of entropy before the shock, with Bx(P^2) the main increase of entropy is located at the upper shock.

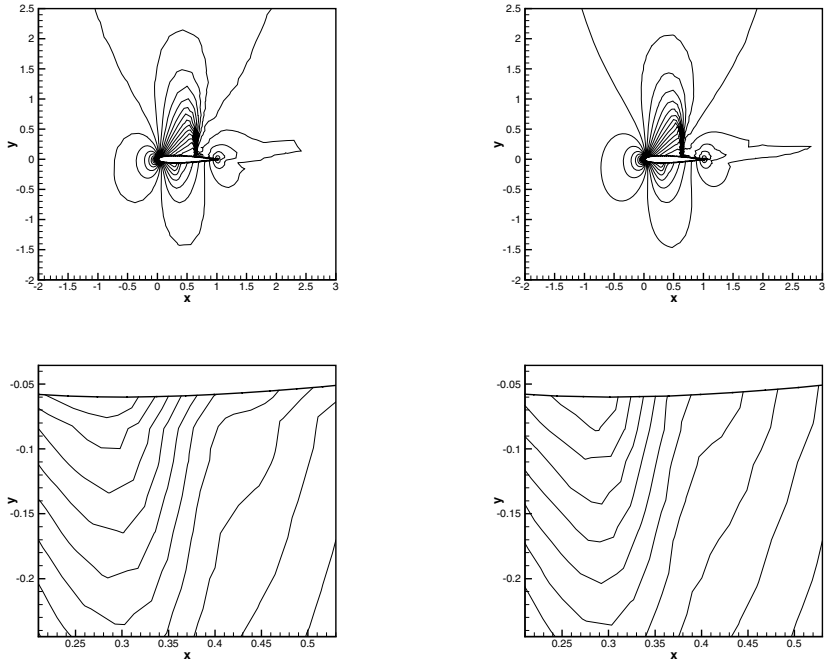


Fig. 8 Transonic flow over a NACA-0012 airfoil (same number of degrees of freedom): Mach iso-lines ($\Delta M = 0.03$) (up)- zoom at leading edge ($\Delta M = 0.02$) (down); P^1P^1 elements (left) and P^1P^2 elements (right)

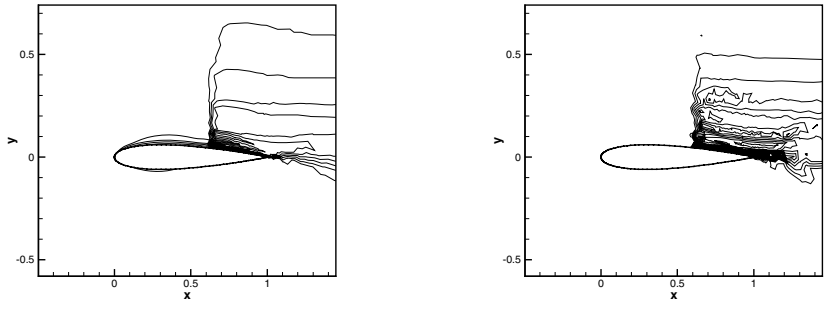


Fig. 9 Transonic flow over a NACA-0012 airfoil: entropy deviation ($\Delta S = 0.002$) using P^1P^1 (left) and P^1P2 (right) elements

4 Unsteady

4.1 Generalities and Notations

We describe a class of compact methods to approximate the unsteady solution of

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0 \quad \forall (x,y,t) \in \Omega_t = \Omega \times [0;t_f] \tag{11}$$

To solve the unsteady problem of (11) we consider that time is a third dimension. So, the domain Ω_t is discretised by a succession of prismatic elements. To get high order discretisation in time we use prisms with $k+1$ levels, each level being a P^k element of τ_h. As an example, on figure 10, P^1 and a P^2 prismatic elements are plotted. For any given function $\mathbf{U}$, its restriction on the prism K is defined by: $\mathbf{U}^h = \sum_{l=n-k+1}^{n+1} H^l(t) \sum_{i \in T} \psi_i^{P^k}(x,y)\, \mathbf{U}_i^l$ where $\mathbf{U}_i^l$ is the value of u^h at node i and time $t_l : \mathbf{U}_i^l = \mathbf{U}^h(x_i, y_i, t_l)$ and $\psi_i^{P^k}(x,y)$ denotes the (mesh dependent) continuous $k-th$ order Lagrangian basis function. H^l is the $1D\ k-th$ order basis function of level l.

In each space-time element the $(\mathbf{U}^{n-l})_{k-1 \le l \le 0}$ are considered as known and $\mathbf{U}^{n+1}$ is the unknown which we compute using the process of a steady problem:

1. We compute the residual on each space-time sub-prism between n and $n+1$: $\phi^{K_s} = \int_{t_n}^{t_{n+1}} \int_{T_s} \left(\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} \right) d\Omega\, dt$.
2. We distribute the residual to the nodes of the sub-prism K_s. To respect the physical meaning of time, we would like to distribute only to the nodes of the level $n+1$. The consistency of the scheme is ensured by a constaint on the time step called past-shield condition (for more details we refer to [14]). Under this condition it is possible to distribute the residual Φ^{K_s} only to the nodes of the level $n+1$: $\phi_i^{K_s} = \phi_i^{n+1} = B_i \phi^{K_s}$.
3. The following system is solved by pseudo-time iterations: $\sum_{K_s, i \in K_s} \phi_i^{n+1,K_s} = 0$.

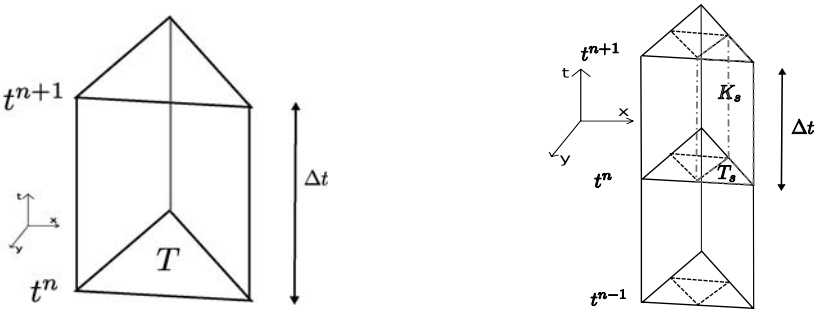


Fig. 10 Space-time P^1 element (left) and P^2 (right) with subdivision, definition of Δt

4.2 Space-Time Schemes

We extend the schemes of the steady case to the space-time framework by simply replacing the spacial upwind parameter by the space-time one: $\tilde{k}_i^{n+1} = \frac{\Delta t}{2} k_i + \frac{|T|}{3}$.

To combine high order of accuracy and monotonicity , we consider the non-linear limitation of the distribution coefficients of the ST-N scheme.

ST-NLim scheme. We limit the distribution coefficient of the N scheme[4]:

$$\begin{array}{l} \Phi_i^{T_s} = \quad \Phi_i^{\text{ST-Nlim}} = B_i^{\text{ST-Nlim}} \Phi^{T_s}, \\ B_i^{Nlim} = \max(0, B_i^{\text{ST-N}}) / \sum_{j \in T_s} \max(0, B_j^{\text{ST-Nlim}}) \end{array} \tag{12}$$

with $B_j^{\text{ST-N}} = \Phi_i^{\text{ST-N}} / \Phi_i^{T_s}$. The Nlim scheme verifies both the monotonicity requirement ($\Phi_i^{T_s} \times \Phi_i^{\text{ST-N}} \geq 0$), and the accuracy condition (2) ($B_i^{\text{ST-Nlim}}$ bounded).

4.3 Results

Now, we want to test the monotonicity and the accuracy of ST-Nlim(P^2). The double Mach reflection test case was first proposed in [6]. It is very interesting to test the accuracy and the robustness of a scheme. It consists of the interaction of a planar right-moving $M = 10$ shock with a 30° ramp. We consider that the ramp is aligned with the x-axis. The computational domain is $[0;3] \times [0;0.8]$ and the ramp start at $x = \frac{1}{6}$. The initial shock forms an angle of 60° with the x-axis as sketched on figure 11. On the top boundary, we impose the movement of the shock. We look at the solution at $t = 0.2$ The real challenge of this test case is to catch the Kelvin-Helmholtz instabilities. To see them, we zoom in on the triple point region and we

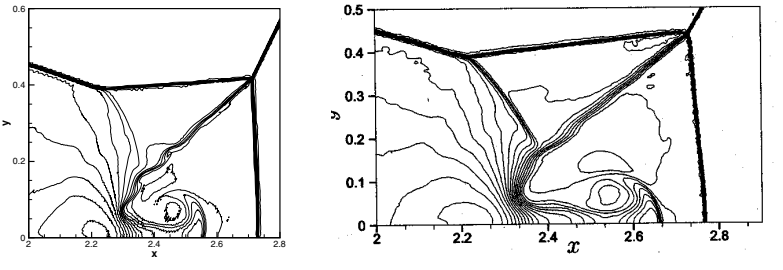


Fig. 11 Double mach reflection: zoom in on the triple point density iso-lines at $t = 0.2$, solution obtained with Nlim(P^2) (left) and Nlim(P^1) (right) with the same number of DOF ($h = 1/240$)

[4] In the case of a scalar problem the distribution coefficients of N scheme can be defined by $B_j^{\text{ST-N}} = \Phi_i^{\text{ST-N}} / \Phi_i^{K_s}$. In the case of non-linear system of equations, this definition is not any more valid. So, we use the wave decomposition proposed by Abgrall to demonstrate the monotonicity of N scheme [7]. For more details on this methodology we refer to [14].

use a mesh with mesh spacing of $h = 1/240$ (in the P^1 mesh). We compare the result obtained by Ricchiuto in [14] and by ST-Nlim(P^2). First, the result obtained with ST-Nlim(P^2) is quasi non-oscillatory. Moreover, the shocks and the slip line are better resolved with ST-Nlim(P^2). Similarly, the instabilities are more developed when using ST-Nlim(P^2).

5 Viscous

5.1 Analogy with Petrov-Galerkin Method (PG)

Now, that we have described the design of high order *RD* scheme for system of hyperbolic conservation laws, we want to link *RD* formulation with Petrov-Galerkin schemes. In both methods the solution is discretised by a combination of the continuous Lagrangian basis functions. However, upwind *RD* schemes such as LDA are set for hyperbolic system of conservation laws whereas *PG* can be used for more general conservation laws. Then, one idea to extend upwind *RD* to viscous term is to use a Petrov-Galerkin approach. To do so, we need to find a weight function such that *RD* and *PG* schemes are equivalent for hyperbolic conservation laws. Without loss of generality we consider the linear scalar advection equation:

$$\mathbf{a} \cdot \nabla u = 0 \tag{13}$$

Lets look for the weight function $\omega_i^{P^k}$ such that if u is linear on each sub-element *PG* and *RD* method would be equivalent. Then, we have:

$$\begin{cases} B_i^{T_s} \displaystyle\int_{T_s} \mathbf{a} \cdot \nabla u \, d\Omega = \int_{T_s} \omega_i^{P_k} \mathbf{a} \cdot \nabla u \, d\Omega & \text{if } i \in T_s \\ 0 \qquad\qquad\quad = \displaystyle\int_{T_s} \omega_i^{P_k} \mathbf{a} \cdot \nabla u \, d\Omega & \text{if } i \notin T_s \end{cases} \tag{14}$$

A weight function that verifies these conditions is:

$$\omega_i^{P_k}{}_{|T_s} = \psi_i^{P_k} + \alpha_i^{T_s} S^{T_s} \tag{15}$$

where S^{T_s} is the bubble function of the sub-element T_s and (x_g, y_g) is the gravity centre of the sub-element T_s.

$$S^{T_s}(x,y) = \begin{cases} 0 \text{ if } (x,y) \in \partial(T_s) \\ 1 \text{ if } (x,y) = (x_g, y_g) \end{cases} \tag{16}$$

Finally, $\alpha_i^{T_s}$ is defined by:

$$\alpha_i^{T_s} = \begin{cases} \dfrac{B_i^{T_s} |T_s| - \displaystyle\int_{T_s} \psi_i^{P_k} \, d\Omega}{\displaystyle\int_{T_s} S^{T_s} \, d\Omega} & \text{if } i \in T_s \\ \dfrac{-\displaystyle\int_{T_s} \psi_i^{P_k} \, d\Omega}{\displaystyle\int_{T_s} S^{T_s} \, d\Omega} & \text{if } i \notin T_s \end{cases} \tag{17}$$

Compared to the shape functions $\psi_i^{P_k}$, the local regularity (within T) of the bubble function S^{T_s} is quite low. However, ultimately both shape and bubble functions are in the same functional space $H_0^1(\Omega)$ and share simple C^0 continuity. In the next section, we use this weight function to construct a consistent scheme to solve system of conservation laws with a viscous term.

5.2 Extension to Viscous Terms

In this section, we consider the solution of a system of conservation laws with a viscous term:

$$\nabla \cdot \mathbf{F}(\mathbf{U}) = \nabla \cdot (\nabla \mathbf{G}(\mathbf{U})) \quad \forall (x,y) \in \Omega \tag{18}$$

To construct a consistent *RD* scheme to solve equation (18), we use the *PG* formulation of the last section:

$$\phi_i^{PG,T} = \sum_{T_s \in T} \underbrace{\int_{T_s} \omega_i^{PG} \nabla \cdot \mathbf{F}(\mathbf{U}) d\Omega}_{I} + \underbrace{\int_{T_s} \nabla \omega_i^{PG} \nabla \mathbf{G}(\mathbf{U}) d\Omega}_{II} \tag{19}$$

We previously saw that integral I is equivalent to *RD* so we will replace it by the nodal residual of *RD* method. Finally, the nodal residual of the *RD* is:

$$\begin{cases} \phi_i^{T_s} = B_i \oint_{\partial T_s} \mathbf{F}(\mathbf{U}) d\Omega + \int_{T_s} \nabla \omega_i^{PG} \nabla \mathbf{G}(\mathbf{U}) d\Omega & i \in T_s \\ \phi_i^{T_s} = \quad 0 \quad + \int_{T_s} \nabla \omega_i^{PG} \nabla \mathbf{G}(\mathbf{U}) d\Omega & i \notin T_s \end{cases} \tag{20}$$

After assembling all the contributions, we end up solving the following system:

$$\sum_{T, i \in T} \sum_{T_s, T_s \in T} \phi_i^{T_s} = 0 \tag{21}$$

With this method we achieve $(k+1)$ order scheme when using P^k elements. We denote by LDA(P^k) these schemes using P^k discretisation.

5.3 Results

Finally, we consider the flow over a NACA-0012 at Mach 0.5 and Reynolds 5000 without angle of incidence. For this test case the Reynolds number is close to the limit of steady laminar flow. The solution has a characteristic separation of the flow near the trailing edge. This forms two symmetrical recirculation bubbles in the near-wake region of the airfoil. The Mach number iso-lines computed using the quadratic discretisation are plotted in figure 13. This solution was obtained on the non-symmetrical mesh plotted on figure 12 which have 8564 degrees of freedom

(nodes on the mesh), from which 200 are on the airfoil. We also plot the repartition of the pressure coefficient (figure 15) and of the friction coefficient (figure 14) around the airfoil. These plots show that we obtain a symmetrical result.

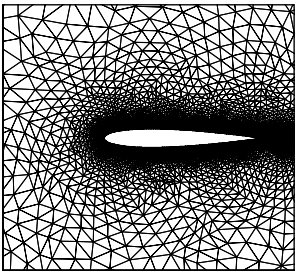


Fig. 12 Viscous flow over a NACA-0012: Type of mesh used

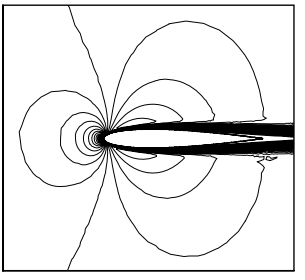


Fig. 13 Viscous flow over a NACA-0012: Mach number iso-lines ($\Delta M = 0.02$)

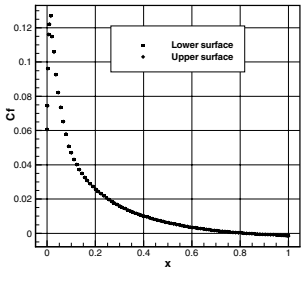


Fig. 14 Skin friction coefficient

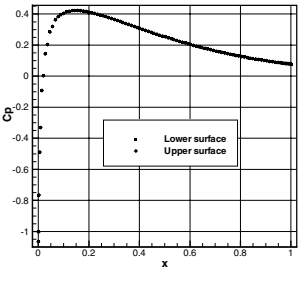


Fig. 15 Pressure coefficient

Finally, on table 1, we compare the drag coefficient and the separation points of literature to the ones obtained with LDA. It is difficult to drive any conclusion from the value of $C_{d,pres}$ because the value obtained with linear and quadratic discretisations are very similar. Concerning the viscous component, we can see that already with the second mesh, we obtain a value very close to the one of literature with LDA(P^2), whereas with LDA(P^1) even on the finest mesh it is still far from the references. Similarly, on the coarsest meshes, we obtain already a good value of the separation point when using a quadratic discretisation, although with the linear one even on the finest mesh the result is a bit far from the expected result. To run the 3^{rd} order scheme on the coarsest mesh 682s CPU time were necessary. To obtain a similar result with the 2^{nd} order scheme it is necessary to use the finest grid and then it takes 2272s CPU time. This shows that it is faster to obtain a good result with a high order scheme than with a low order one.

Table 1 Comparison of Drag coefficients and separation point with literature

Mesh	Cdp	Cdv	Separation point
4630 DOF (148 on wall) P^1	0.02239	0.03132	94.5
4630 DOF (148 on wall) P^2	0.02238	0.03414	82.5
8564 DOF (200 on wall) P^1	0.02296	0.03130	87
8564 DOF (200 on wall) P^2	0.02234	0.03294	82.3
17146 DOF (300 on wall) P^1	0.02284	0.03181	82.9
17146 DOF (300 on wall) P^2	0.02239	0.03274	82.1
Reference [11] 16384	0.0219	0.0337	81.9
Reference [12] 65536	0.0227	0.0327	81.4
Reference [13] 1024 (cubic elements)	0.02208	0.03303	

6 Conclusion

We have presented an extension of upwind residual distributive schemes to high order discretisations. In particular, we have described two ways to deal with curved boundaries: Krivodonova approach and the use of P^2P^2 elements. Both strategies decrease the creation of entropy. But the best result is obtained when using a high order discretisation of the geometries. Then, we have considered the construction of high order monotone schemes. In the steady case, we have used a blending method,that allows to use a high order scheme in smooth region and a monotone scheme (only 1^{st} order accurate) on discontinuities. This strategy allows to get a quasi non-oscillatory solution. In the unsteady case, to achieve accuracy and monotonicity, we have limited the distribution coefficients of N scheme to get an order preserving scheme. This scheme also give quasi monotone solutions. Finally, we have used an analogy between Petrov-Galerkin method and Residual Distributive method to extend RDS to viscous flow. We have compared the results obtained with linear and quadratic discretisations and we have shown that the quadratic discretisation were more efficient.

References

1. Krivodonova, L., Berger, M.: High-order accurate implementation of solid wall boundary conditions in curved geometries. J. Comput. Phys. 211, 492–512 (2005)
2. Bonanni, A.: Investigation of high-order residual distribution schemes for external aerodynamic applications. Diploma course report, Von Karman Indtitute for Fluid Dynamics (2009)
3. Abgrall, R., Roe, P.L.: High order fluctuation schemes on triangular meshes. J. Sci. Comput. 19(3), 3–36 (2003)
4. Godunov, S.K.: A finite difference method for the computation of discontinuous solutions of the equations of fluid dynamics. Mat. Sb 47, 357–393 (1959)

5. Ricchiuto, M., Villedieu, N., Abgrall, R., Deconinck, H.: High-order residual distribution schemes: discontinuity capturing crosswind dissipation and extension to advection-diffusion. In: VKI LS on Higher Order Discretization Methods for Computational Physics, Von Karman Institute for Fluid Dynamics (2005)
6. Woodward, W.A., Colella, P.: The numerical simulation of two-dimensional flows with strong shocks. J. Comput. Phys. 54, 115–173 (1984)
7. Abgrall, R., Mezine, M.: Construction of second-order accurate monotone and stable residual distribution schemes for steady flow problems. J. Comput. Phys. 195, 474–507 (2004)
8. Ricchiuto, M., Abgrall, R., Deconinck, H.: Construction of very high order residual distributive schemes for unsteady scalar advection: preliminary results. In: VKI LS 2003-05 33rd computational fluid dynamics-novel methods for solving convection dominated systems (2003)
9. Ricchiuto, M., Villedieu, N., Abgrall, R., Deconinck, H.: On uniformly high order accurate residual distribution schemes for advection-diffusion. Journal of Computational and Applied Mathematics 215, 547–556 (2007)
10. Ricchiuto, M., Dobes, J., Abgrall, R., Deconinck, H.: Analysis of hybrid residual distribution-Galerkin disretizations for advection-diffusion. Application to the solution of the steady and time-dependent laminar NavierStokes equations (to be published)
11. Martinelli, L.: Calculations of viscous flows with a multigrid method. PhD Thesis, Dept. of Mechanical and Aerospace Engineering, Princeton University (1987)
12. Radespield, R., Swanson, R.C.: An investigation of cell centered and cell vertex multigrid schemes for Naver-Stokes equation. AIAA paper 89-0543 (January 1989)
13. Bassi, F., Rebay, S.: A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. Journal of computational physics 131, 267–279 (1997)
14. Ricchiuto, M.: Construction and analysis of compact residual discretizations for conservation laws on unstructured meshes. PhD. Thesis, Université Libre de Bruxelles (2005)

This page intentionally left blank

Chapter 11
Higher-Order Stabilized Finite Elements in an Industrial Navier-Stokes Code

Frédéric Chalot and Pierre-Elie Normand

Abstract. This chapter covers Dassault Aviation's contribution to Workpackage 3 of the ADIGMA Project, which focuses on the extension of its stabilized finite element industrial Navier-Stokes code to higher-order elements. Mesh generation aspects are treated and especially the issue of highly-stretched curved elements close to the wall boundary of Navier-Stokes meshes. The high-order approach is carefully assessed using inviscid subsonic and transonic, laminar, and high Reynolds number turbulent flows.

1 Introduction

Over the past two decades, modern CFD has gone from producing pretty pictures to actually producing numbers which are crucial when improving the aerodynamic design of aircraft. Over this period of time, models have improved going from inviscid Euler calculations to laminar and then turbulent Navier-Stokes. Turbulent models have evolved from purely algebraic models to RANS models, to unsteady models like LES and DES which are slowly making their way into the industrial world [10]. The growth of computer power has also tremendously helped that change: the power of the vector supercomputers of the 80's is now available on laptop PC's, whereas the Top500 parallel computers are flirting with a few sustained Petaflops. Most industrial CFD codes and commercial packages have made the transition from early developments in finite differences to finite volumes, and are rapidly moving

Frédéric Chalot
Dassault Aviation, 78 quai Marcel Dassault, CEDEX 300, 92252 Saint-Cloud CEDEX, France
e-mail: frederic.chalot@dassault-aviation.com

Pierre-Elie Normand
Université de Bordeaux, 341 cours de la Libération, 33405 Talence CEDEX, France
e-mail: pierre-elie.normand@external.dassault-aviation.com

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 145–165, 2010.
springerlink.com

away from the apparent simplicity of structured and block-structured meshes to the flexibility of unstructured meshes, often at the price of a lesser efficiency.

A large amount of work was also performed to develop stable and accurate spatial numerical schemes for compressible flow calculations. Overly diffusive first-order schemes were rapidly abandoned for second-order accurate schemes with monotonic shock-capturing capabilities. Such a level of accuracy proved sufficient for most industrial codes, even for applications such as LES and DES which were at first developed with higher-order schemes [7]. Higher-order schemes seem reserved for specific fields (DNS, aeroacoustics) where the enhanced accuracy is mandatory.

Today's complex applications require an ever increasing number of grid points for which mesh convergence can seemingly never be attained.

The European project **ADIGMA** proposed a unique framework to address many of the accuracy and cost issues of current industrial CFD codes. The different partners have put together innovative higher-order methods which will constitute key ingredients for the next generation of industrial flow solvers. The participation of Dassault Aviation was twofold: higher-order stabilized finite elements described in the following sections, and adjoint-based adaptive mesh refinement detailed in Chapter 26.

2 Higher-Order Stabilized Finite Element Schemes for the RANS Equations

Although Dassault Aviation started from the beginning with unstructured meshes and a Navier-Stokes code based on a finite element formulation, the claim that finite elements can fairly effortlessly and in a straightforward manner go high in order was never fully exploited. We currently still use for all Navier-Stokes calculations linear elements which yield second-order accuracy [5, 9, 10]. A single but successful attempt was made to compute the flow past a supersonic ramp [2] using quadratic elements.

Higher-order (3^{rd} and 4^{th} order) finite elements in the SUPG/Galerkin-least squares framework will be revisited. We will present our numerical method in the following sections and highlight the adjustments required by higher-order elements.

2.1 General Description of Our Flow Solver

Dassault Aviation's Navier-Stokes code, called `AETHER`, uses a finite element approach, based on a symmetric form of the equations written in terms of entropy variables. The advantages of this change of variables are numerous: in addition to the strong mathematical and numerical coherence they provide (dimensionally correct dot product, symmetric operators with positivity properties, efficient preconditioning), entropy variables yield further improvements over the usual conservation variables, in particular in the context of chemically reacting flows (see [4, 5]).

The code can handle the unstructured mixture of numerous types of elements (triangles and quadrilaterals in 2-D; tetrahedra, bricks, and prisms in 3-D). In practice mostly linear triangular and tetrahedral meshes are used.

Different one- and two-equation Reynolds-averaged turbulence models are available: Spalart-Allmaras, K-ε, K-ω, K-ℓ, K-KL... These models are either integrated down to the wall, use a two-layer approach with a low-Reynolds modelization of the near wall region, or adopt a wall function treatment of the boundary layer. More advanced RANS models, such as EARSM and RSM, and extensions to LES and DES are also available (see [7], [8], and [10]).

Convergence to steady state of the compressible Navier Stokes equations is achieved through a fully-implicit iterative time-marching procedure based on the GMRES algorithm with nodal block-diagonal or incomplete *LDU* preconditioning (see [12]).

The code has been successfully ported on many computer architectures. It is fully vectorized and parallelized for shared or distributed memory machines using the MPI message passing library (IBM SP2 Series, IBM BlueGene, Itanium II- and Xeon-based Bull NovaScale) or native parallelization directives (NEC SX-4) (see [6]).

2.2 The Symmetric Navier-Stokes Equations

As a starting point, we consider the compressible Navier-stokes equations written in conservative form:

$$_{,t} + \quad ^{\text{adv}}_{i,i} = \quad ^{\text{diff}}_{i,i} \tag{1}$$

where $\quad$ is the vector of conservative variables; $\quad ^{\text{adv}}_{i}$ and $\quad ^{\text{diff}}_{i}$ are, respectively, the advective and the diffusive fluxes in the i^{th}-direction. Inferior commas denote partial differentiation and repeated indices indicate summation.

Equation (1) can be rewritten in quasi-linear form:

$$_{,t} + \quad _{i} \quad _{,i} = (\quad _{ij} \quad _{,j})_{,i} \tag{2}$$

where $\quad _{i} = \quad ^{\text{adv}}_{i,\boldsymbol{U}}$ is the i^{th} advective Jacobian matrix, and $\quad = [\quad _{ij}]$ is the diffusivity matrix, defined by $\quad ^{\text{diff}}_{i} = \quad _{ij} \quad _{,j}$. The $\quad _{i}$'s and $\quad$ do not possess any particular property of symmetry or positiveness.

We now introduce a new set of variables,

$$_{T} = \frac{\partial \mathscr{H}}{\partial}$$

where $\mathscr{H}$ is the generalized entropy function given by

$$\mathscr{H} = \mathscr{H}(\quad) = -\rho s$$

and s is the thermodynamic entropy per unit mass. Under the change of variables $\mapsto \quad$, (2) becomes:

$$\widetilde{\quad}_0\ _{,t} + \widetilde{\quad}_i\ _{,i} = (\widetilde{\quad}_{ij}\ _{,j})_{,i} \tag{3}$$

where

$$\begin{aligned} \widetilde{\quad}_0 &= \ _{,}\boldsymbol{V} \\ \widetilde{\quad}_i &= \ _i\ \widetilde{\quad}_0 \\ \widetilde{\quad}_{ij} &= \ _{ij}\ \widetilde{\quad}_0. \end{aligned}$$

The Riemannian metric tensor $\widetilde{\quad}_0$ is symmetric positive-definite; the $\widetilde{\quad}_i$'s are symmetric; and $\widetilde{\quad} = [\widetilde{\quad}_{ij}]$ is symmetric positive-semidefinite. In view of these properties, (3) is referred to as a symmetric advective-diffusive system.

For a general divariant gas, the vector of so-called (physical) entropy variables, , reads

$$= \frac{1}{T}\begin{Bmatrix} \mu - |\ \ |^2/2 \\ \\ -1 \end{Bmatrix}$$

where $\mu = e + pv - Ts$ is the chemical potential per unit mass; $v = 1/\rho$ is the specific volume. More complex equations of state are treated in [3]. We would like to stress the formal similarity between the conservation variables and the entropy variables , which can be made more apparent if we write the conservation variables in the following form:

$$= \frac{1}{v}\begin{Bmatrix} 1 \\ \\ e + |\ \ |^2/2 \end{Bmatrix}$$

where $v = 1/\rho$ is the specific volume.

Taking the dot product of (3) with the vector yields the Clausius-Duhem inequality, which constitutes the basic nonlinear stability condition for the solutions of (3). This fundamental property is inherited by appropriately defined finite element methods, such as the one described in the next section.

2.3 The Galerkin/Least-Squares Formulation

Originally introduced by Hughes and Johnson, the Galerkin/least-squares (GLS) formulation is a full space-time finite element technique employing the discontinuous Galerkin method in time (see [1, 13]). The least-squares operator ensures good stability characteristics while retaining a high level of accuracy. The local control of the solution in the vicinity of sharp gradients is further enhanced by the use of a nonlinear discontinuity-capturing operator.

Let Ω be the spatial domain of interest and Γ its boundary. The semi-discrete Galerkin/least-squares variational problem can be stated as:

Find $^h \in \mathscr{S}^h$ (trial function space), such that for all $^h \in \mathscr{V}^h$ (weighting function space), the following equation holds:

$$\int_{\Omega}\Big(W^h\cdot \quad_{,t}(\quad^h)-\quad^h_{,i}\cdot\quad^{\mathrm{adv}}_i(\quad^h)+\quad^h_{,i}\cdot\widetilde{\quad}_{ij}\quad^h_{,j}\Big)\,d\Omega$$
$$+\sum_{e=1}^{n_{\mathrm{el}}}\int_{\Omega^e}\Big(\mathscr{L}\quad^h\Big)\cdot\Big(\mathscr{L}\quad^h\Big)\,d\Omega$$
$$+\sum_{e=1}^{n_{\mathrm{el}}}\int_{\Omega^e}\nu^h g^{ij}\quad^h_{,i}\cdot\widetilde{\quad}_0\quad^h_{,j}\,d\Omega$$
$$=\int_{\Gamma}\quad^h\cdot\Big(-\quad^{\mathrm{adv}}_i(\quad^h)+\quad^{\mathrm{diff}}_i(\quad^h)\Big)n_i\,d\Gamma. \qquad (4)$$

The first and last integrals of (4) represent the Galerkin formulation written in integrated-by-parts form to ensure conservation under reduced quadrature integration.

The second integral constitutes the least-squares operator where $\mathscr{L}$ is defined as

$$\mathscr{L}=\widetilde{\quad}_0\frac{\partial}{\partial t}+\widetilde{\quad}_i\frac{\partial}{\partial x_i}-\frac{\partial}{\partial x_i}\Big(\widetilde{\quad}_{ij}\frac{\partial}{\partial x_j}\Big). \qquad (5)$$

is a symmetric time-scale matrix for which definitions can be found in [13].

The third integral is the nonlinear discontinuity-capturing operator, which is designed to control oscillations about discontinuities, without upsetting higher-order accuracy in smooth regions. g^{ij} is the contravariant metric tensor defined by

$$[g^{ij}]=[\quad_{,i}\cdot\quad_{,j}]^{-1}$$

where $\quad=\quad(\boldsymbol{x})$ is the inverse isoparametric element mapping and ν^h is a scalar-valued homogeneous function of the residual $\mathscr{L}\quad^h$. The discontinuity capturing factor ν^h used for linear elements is an extension of that introduced by Hughes, Mallet, and Shakib (see [11, 13]).

A key ingredient to the formulation is its consistency: the exact solution of (1) satisfies the variational formulation (4). This constitutes an essential property in order to attain higher-order spatial convergence.

2.4 Extension to Higher-Order Elements

In principle everything is contained in the weighted residual given by Eq. (4). There is no new term to code, no interpolation technique specific to higher order to derive: everything is already there. We just have to compute the integrals of (4), taking into account the new higher-order shape functions.

The volume and surface integrals are numerically evaluated with quadrature rules. All is needed is the values of the shape functions (and their gradients) at the integration points. Higher-order functions only require more precise integration rules. In general, we use 3-, 6-, and 12-point rules, respectively for linear, quadratic, and cubic triangles. They have orders of accuracy which integrate exactly polynomials of degrees 2, 4, and 6 respectively.

For a given number of degrees of freedom, higher-order meshes contain much fewer elements than P1 meshes. The ratio is 1/4th for quadratic elements, and 1/9th for cubic. Although more integration points are required, the higher-order computation of (4) is actually cheaper. The extra cost comes from the implicit linear system which possesses a much larger bandwidth. For a regular 2-D mesh with six triangles connected to a given node, each line of the implicit matrix contains 7, 19, and 37 non-zero blocks, respectively for P1, P2, and P3 elements.

Preliminary quadratic and cubic element results obtained with the original stabilization and discontinuity capturing term used for linear elements, appeared too diffusive especially for MTC 2. This is an indication that the intrinsic time scale matrix must be reduced for higher-order elements. Theoretical study of the 1-D scalar advection diffusion equation showed that the optimal must indeed be reduced in the advective limit for any higher-order element. The shock capturing operator must also be tuned in a similar fashion.

In fact one term in the weighted residual must be specially treated in the context of higher-order elements. The last term in (5) vanishes to zero for linear elements. It appears in the second integral of (4). This term must be computed with higher-degree shape and test functions in order to preserve consistency. In practice, it is evaluated using an L_2-projection.

One-dimensional studies showed that there was no significant differences between SUPG and Galerkin/least-squares. We have chosen to concentrate solely on SUPG which is easier to implement.

As a final remark, we want to stress the fact that whatever the order of the elements, all operations remain local (viz. at the element level). Consequently higher-order elements engender no implicitation nor parallelization issue (see [6]).

3 Isoparametric Meshes with Curved Boundaries

We have made the seemingly obvious choice of higher-order **isoparametric** elements. One of the advantages of these elements, besides the higher-order shape functions, is the use of higher-order polynomials to represent curved boundaries. They only ensure C^0 continuity across elements, but locate all the nodes on the actual surface.

We had thought at first that the slope discontinuity across element boundaries could be minimized by adjusting the location of the extra nodes along the sides and the faces of elements beyond P1. In practice it is very easy to generate negative elements with "shamrock"-like edges if one tries to play with node location along edges to optimize curvature. Consequently we sticked in this study to elements with equally distributed nodes along the edges and faces.

All higher-order meshes were obtained by adding nodes to a coarse initial P1 mesh. We had previously checked that quadratic and cubic triangular meshes would fit DASSAV data structure. Nested two-dimensional P1, P2, and P3 grids could be generated with equally distributed boundary nodes. Local node numbering was introduced into `AETHER` for cubic triangles, quadratic and cubic tetrahedra (linear

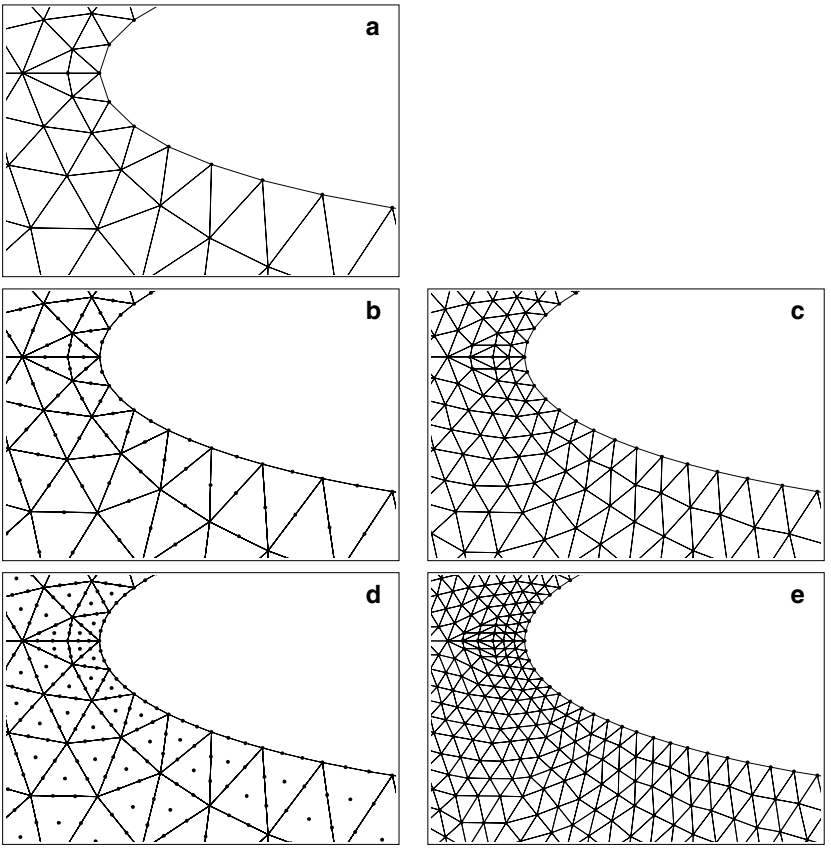


Fig. 1 Higher-order mesh generation for inviscid test cases: original P1 mesh (a); P2 mesh and corresponding P1 mesh (b and c); P3 mesh and corresponding P1 mesh (d and e)

and quadratic triangles, and linear tetrahedra were preexisting). Corresponding face numbering for boundary elements was also introduced.

ARA provided sets of unstructured linear triangular grids for the Mandatory Test Cases. We had to generate new series of higher-order P1, P2, and P3 meshes. The first meshes used for the inviscid MTC's (1 and 2) are depicted in Figure 1.

All inviscid higher-order meshes were obtained by adding nodes to a coarse 1106-node P1 mesh (see Fig. 1 a). This yields a 4336-node P2 mesh and a 9690-node P3 mesh. The first P2 mesh is shown in Figure 1 b, whereas its P1 counterpart, which contains exactly the same number of grid points, is shown in Figure 1 c. Figures 1 d and e show the first cubic mesh and the matching linear grid. Four finer quadratic grids (up to 1,088,896 nodes) and two finer cubic grids (up to 775,386 nodes) were generated. All new nodes are added along the actual profile. This produces boundary elements with curved edges. Elements not connected to the boundary have straight edges.

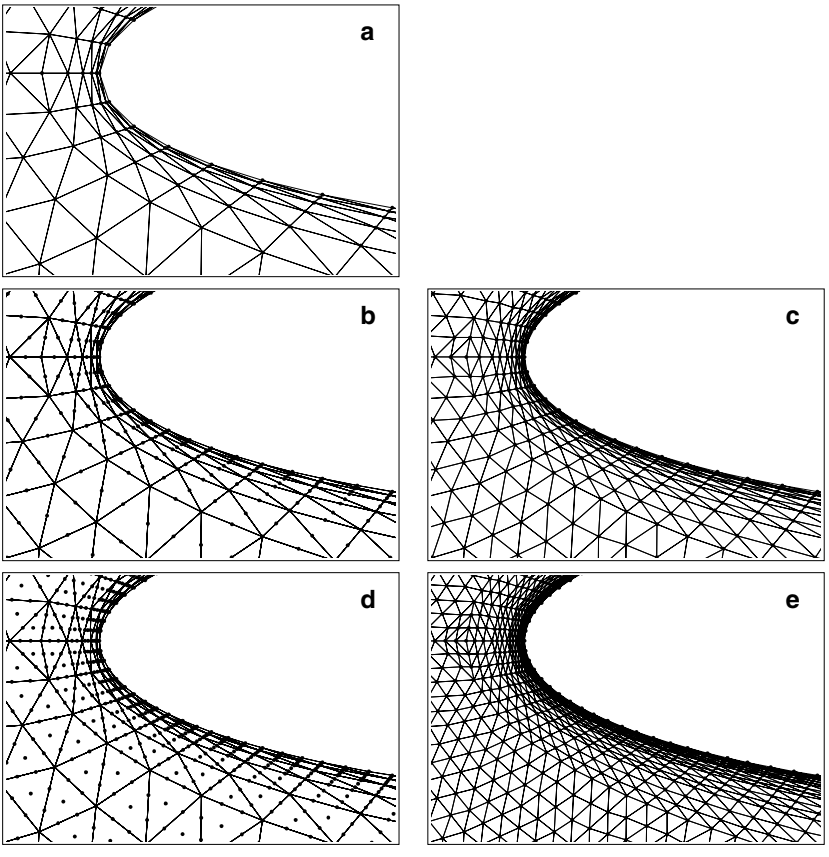


Fig. 2 Higher-order mesh generation for Navier-Stokes test cases: original P1 mesh (a); P2 mesh and corresponding P1 mesh (b and c); P3 mesh and corresponding P1 mesh (d and e)

Navier-Stokes meshes with their stretched elements along the boundary bring a specific difficulty: extra nodes added along the boundary may produce negative elements. Figure 2 a presents the initial coarse 1533-node mesh which is the starting point of all grids generated for MTC 3. The corresponding 6034-node P2 and P1 iso-P2 grids, and 13,503-node P3 and P1 iso-P3 grids are shown respectively in Figures 2 b–e. Four finer quadratic grids (up to 1,521,184 nodes) and two finer cubic grids (up to 1,083,159 nodes) were generated.

Figure 3 depicts the mesh deformation technique used to generate stretched and curved higher-order elements close to the airfoil boundary for the Navier-Stokes cases. The initial grid is shown in Figure 3 a. P1 iso-P2 and P1 iso-P3 grids are constructed with straight edges (see Figs. 3 b and d). The bold line represents the actual boundary. Figures 3 c and e presents the P1 meshes after deformation. The

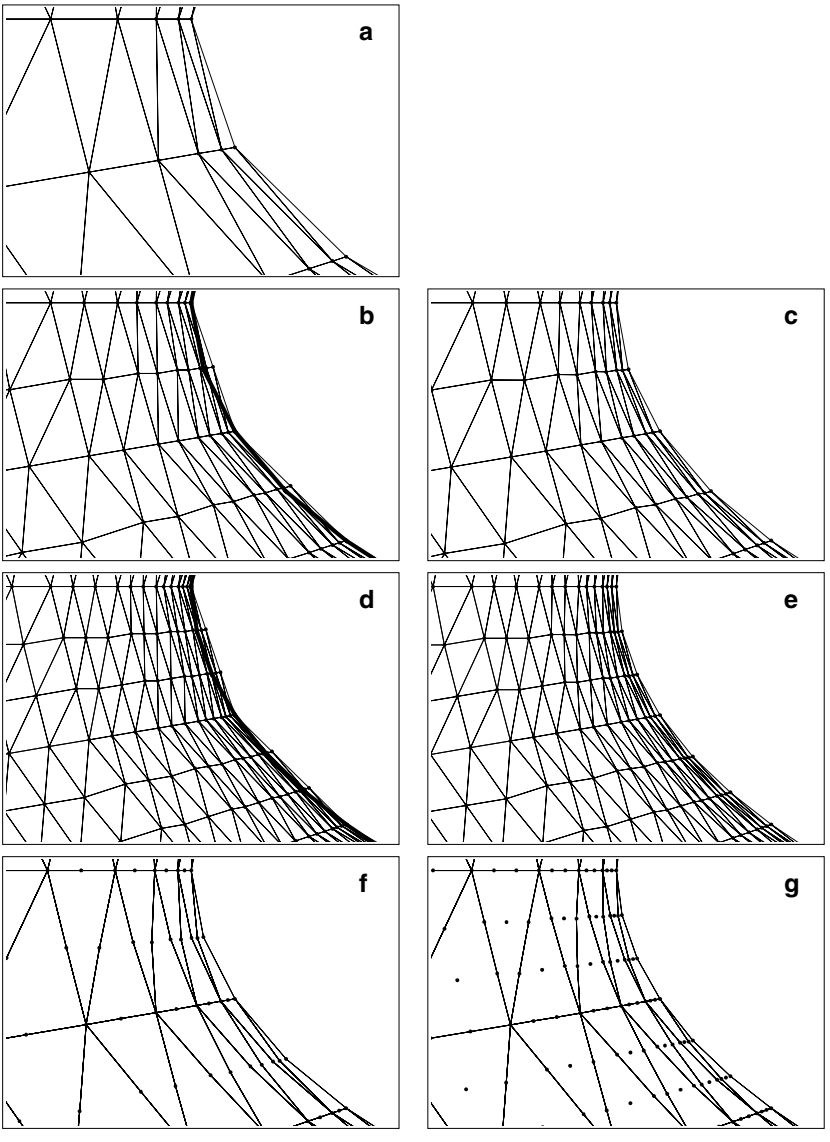


Fig. 3 Mesh deformation for isoparametric higher-order mesh generation for Navier-Stokes test cases: original P1 mesh (a); P1 iso-P2 mesh before (b) and after (c) deformation; P1 iso-P3 mesh before (d) and after (e) deformation; in (b) and (d) actual boundary is represented with bold line; final P2 and P3 grids (f and g).

corresponding P2 and P3 grids are shown in Figures 3 f and g. Unlike the meshes built for the inviscid test cases, these meshes contains elements with curved faces in the volume away from the airfoil surface.

A series of P1, P2, and P3 meshes was also generated for MTC 5. The same mesh deformation technique used for MTC 3 grids was applied to obtain stretched and curved higher-order elements close to the airfoil boundary. Highly stretched elements are present close to the airfoil surface and in the wake with aspect ratios up to 2×10^6!

4 Numerical Examples

Dassault Aviation computed four of the Mandatory Test Cases defined in Workpackage 2 of the ADIGMA Project. They cover a wide range of applications: from inviscid subsonic and transonic flows (MTC's 1 and 2), to laminar Navier-Stokes (MTC 3), and finally a profile in transonic turbulent conditions (MTC 5). All four test cases were run with the baseline second-order version of Dassault Aviation's industrial Navier-Stokes code AETHER and with the revisited or newly developed third and fourth order extensions.

4.1 MTC 1: NACA0012, $M = 0.50$, $\alpha = 2^\circ$, Inviscid

As an introductory comment, we should say that our code AETHER is really dedicated to Navier-Stokes applications. It can compute Euler flows but uses a strong slip boundary condition at the nodes with the true normals to the geometry. We impose a weak slip boundary condition at the trailing edge of airfoils and in regions where the definition of a single normal is tricky. A more natural way of imposing the inviscid slip condition in a finite element framework would be a weak condition through the boundary integral everywhere. Nevertheless inviscid test cases are valuable since they allow the assessment of the higher-order stabilization operator in the advection limit.

Higher-order MTC 1 results are compared with those obtained on the corresponding P1 mesh with the same number of nodes in Figures 4–5. They clearly show the advantage of the increased order of accuracy brought by quadratic and cubic elements. The entropy layer generated at the stagnation point is much reduced with quadratic elements and virtually disappears with cubic elements. This directly impacts the Mach number contours which traditionally present kinks near the wall on coarse P1 meshes. These kinks are removed from higher-order calculations, which also present much cleaner contours for the same number of degrees of freedom.

The kinks in Mach number contours observed in second-order solutions along the profile are not due to a lower degree of accuracy boundary condition or boundary integral computation as may have been suggested, but in fact to the level of spurious entropy generated at the leading edge. It is convected along the profile and affects the solution close to the airfoil. This fact will be confirmed in Chapter 26, where local mesh refinement in the sole leading edge region suppresses the spurious entropy production.

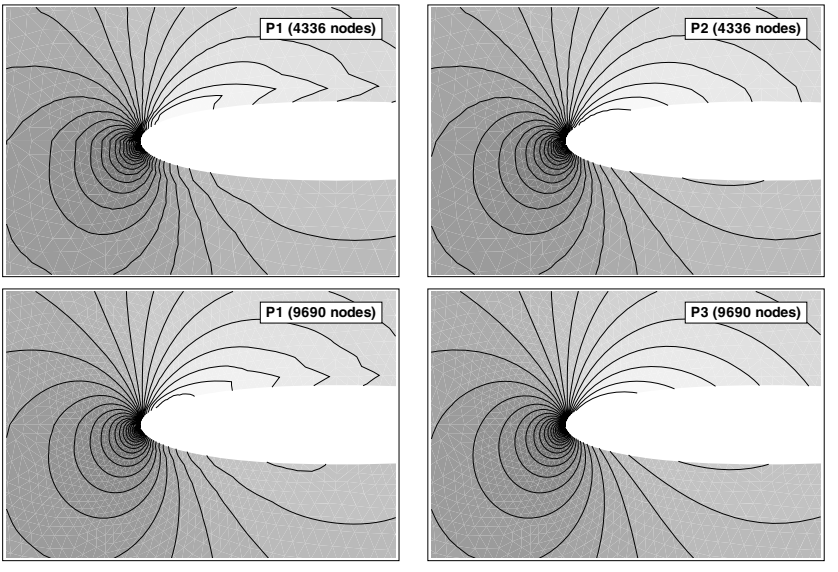


Fig. 4 MTC 1: NACA0012, $M = 0.50$, $\alpha = 2^\circ$, inviscid. Mach number contours on matching P1 iso-P2 and P2 grids, and P1 iso-P3 and P3 grids.

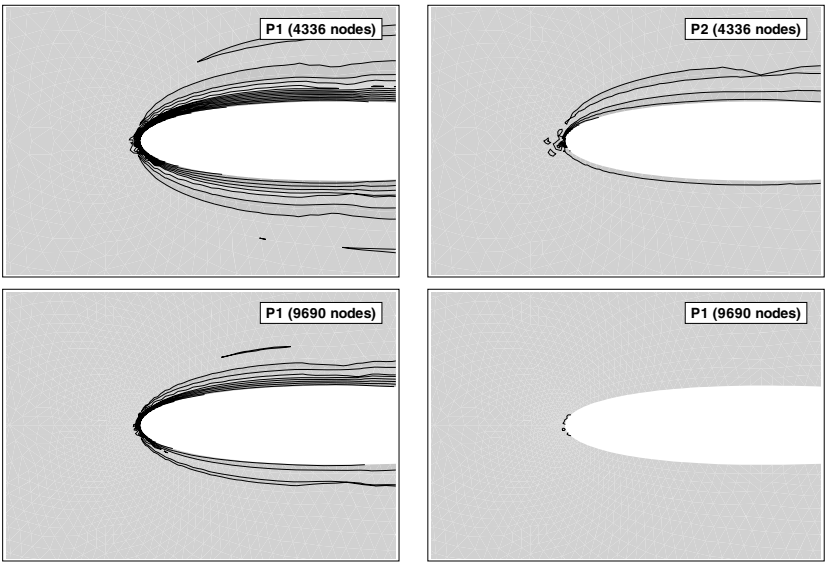


Fig. 5 MTC 1: NACA0012, $M = 0.50$, $\alpha = 2^\circ$, inviscid. Entropy contours on matching P1 iso-P2 and P2 grids, and P1 iso-P3 and P3 grids.

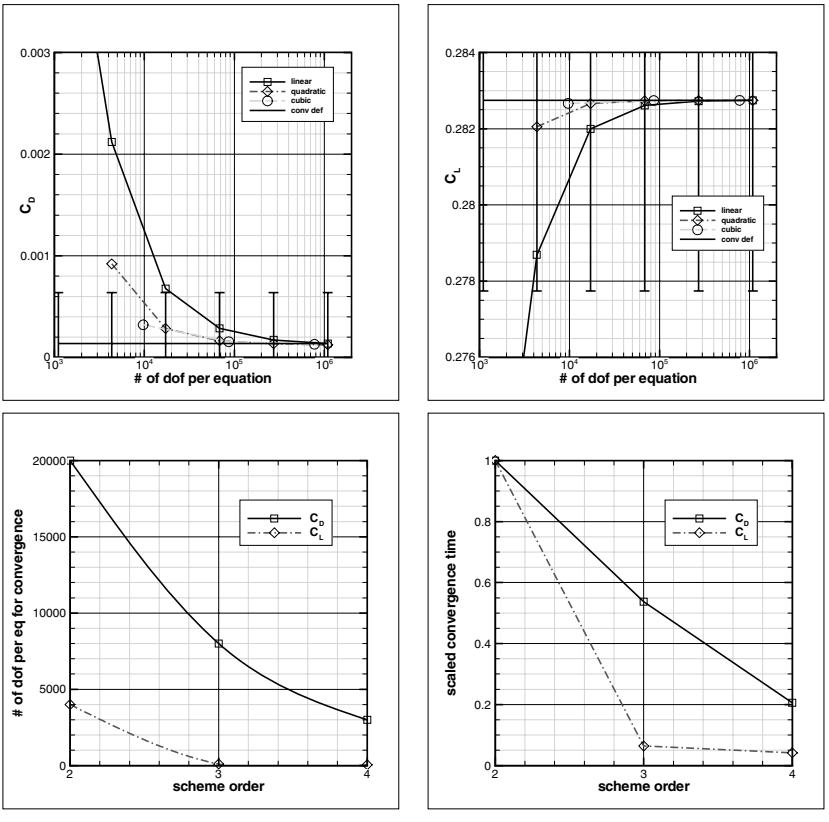


Fig. 6 MTC 1: NACA0012, $M = 0.50$, $\alpha = 2^\circ$, inviscid. Convergence of force coefficients for P1, P2, and P3 elements; estimated numbers of degrees of freedom and times for convergence.

Figure 6 presents the convergence of the drag and lift coefficients with respect to the grid size given by its node number or "number of degrees of freedom per equation." The error bars represent the convergence definitions provided for the test case: when a given coefficient reaches within the error bars, the solution is assumed converged for that particular coefficient.

We can notice a dramatic increase in convergence rate with the order of the scheme. Lift is converged for every tested higher-order mesh; drag requires more effort, and may still gain from an increase in scheme order beyond 4 as shown in the last plots of Figure 6. Even CPU time shows a gain with scheme order (note that a few higher-order values in these plots have been extrapolated). The times for convergence are scaled by the corresponding time for linear elements.

4.2 MTC 2: NACA0012, $M = 0.80$, $\alpha = 1.25°$, Inviscid

MTC 2 is a transonic inviscid test case. It is interesting in its own respect, since it can challenge the ability of higher-order elements to treat shocks with the help of the discontinuity capturing operator.

Figure 7 shows Mach number contours on the same set of meshes used for MTC 1. In spite of the presence of the shock wave, no obvious degradation in the solution quality can be observed. P3 elements even produce the best result with a well resolved slip line and a captured windward-side weak shock.

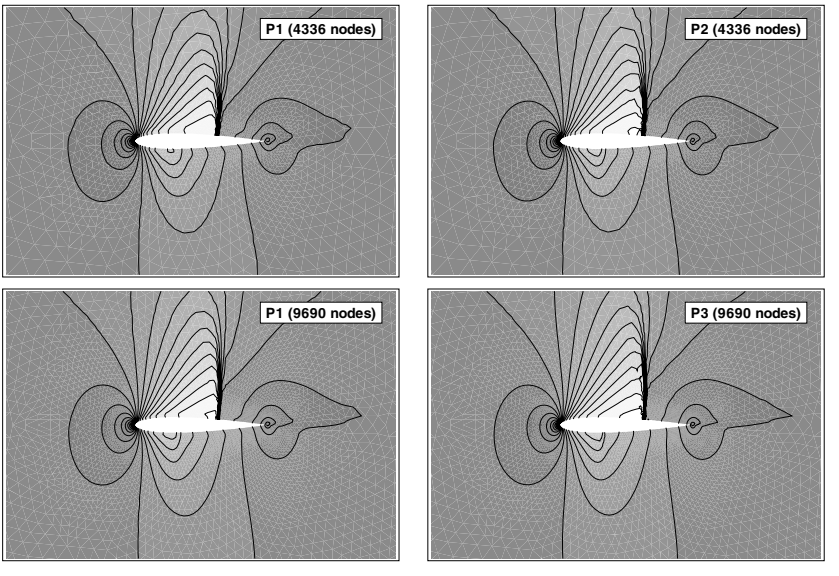


Fig. 7 MTC 2: NACA0012, $M = 0.80$, $\alpha = 1.25°$, inviscid. Mach number contours on matching P1 iso-P2 and P2 grids, and P1 iso-P3 and P3 grids.

Entropy contours displayed in Figure 8 show a reduction in the production of spurious leading-edge entropy similar to MTC 1. However the entropy rise through the normal shock does not look as controlled with higher-order elements. The perturbations remain local though, thanks to the SUPG operator. Note that all these contours are plotted on P1 meshes. Actual higher-order contours might be smoother.

Figure 9 presents the convergence of the drag and lift coefficients. As with MTC 1, all higher-order meshes display a converged lift coefficient, whereas drag requires more mesh points. The last two plots in Fig. 9 indicate that most of the gain is obtained with third order elements. On the average, CPU time to convergence is reduced by 80%.

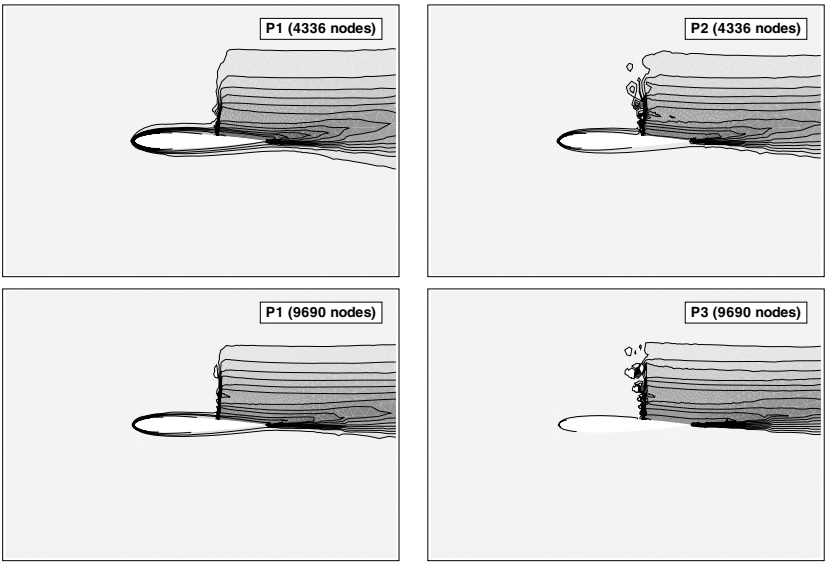


Fig. 8 MTC 2: NACA0012, $M = 0.80$, $\alpha = 1.25°$, inviscid. Entropy contours on matching P1 iso-P2 and P2 grids, and P1 iso-P3 and P3 grids.

4.3 MTC 3: NACA0012, $M = 0.50$, $\alpha = 2°$, $Re = 5,000$

We now come to MTC 3, one of the most interesting test cases in the selection. It concerns the laminar computation of an airfoil. Although a Navier-Stokes test case, it is still far from concrete industrial applications. We will see however that it exemplifies the difficulty of getting converged Navier-Stokes solutions. One can anticipate an even greater challenge with complex 3-D RANS computations.

Figure 10 presents pressure contours obtained on the coarsest quadratic and cubic meshes. They are compared with results computed on corresponding linear meshes containing the very same numbers of grid points. P1 results show the difficulty of preserving a constant pressure through an underresolved boundary layer and highly stretched elements. This difficulty is alleviated with the increasing order of the elements.

Figure 11 presents the convergence of force coefficients: pressure drag and lift, friction drag, and heat flux. The advantage of higher-order elements is even more blatant than for the inviscid test cases described previously. Pressure drag and lift converge faster with quadratic elements; cubic elements yield values close to the asymptotic limit for every computed grid, even the coarser ones.

Unexpectedly viscous fluxes appear as a real challenge for this laminar test case. Second order viscous drag is still not converged for the finest mesh which contains over 1.5 million nodes: the asymptotic value is provided by the quadratic results. The magnified plot is even more striking: linear elements have a hard time getting within one drag count of the asymptotic value of the friction drag, whereas as all

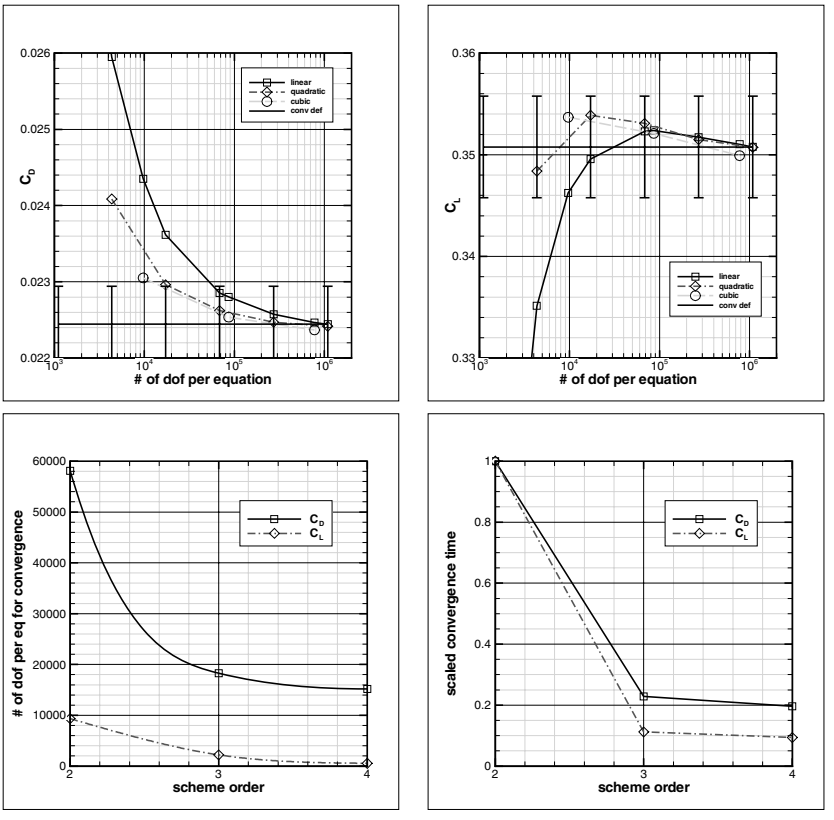


Fig. 9 MTC 2: NACA0012, $M = 0.80$, $\alpha = 1.25^\circ$, inviscid. Convergence of force coefficients for P1, P2, and P3 elements; estimated numbers of degrees of freedom and times for convergence.

higher-order results are within half of the same margin. Heat flux convergence plotted in log scale shows the substantial advantage of higher-order elements. The error in heat flux (which should be zero for an adiabatic wall condition) can be reduced by several orders of magnitude.

The number of nodes and the CPU time for convergence are again reduced with the order of the scheme used. Quadratic elements bring most of the reduction, except for lift which seems to converge at a slower rate and may benefit from an element order beyond 3.

Regarding CPU cost and memory requirements, we can be more specific for this particular test case. For the same number of degrees of freedom, the extra cost of P2 elements over P1 is only 30%; P3 elements are 2 to 2.5 times as expensive as P1 elements. The overhead due to the L_2 projection can be reduced. The CPU cost increase is overtaken by the drastic reduction in the number of nodes required for convergence. Consequently the CPU time for convergence decreases with the degree

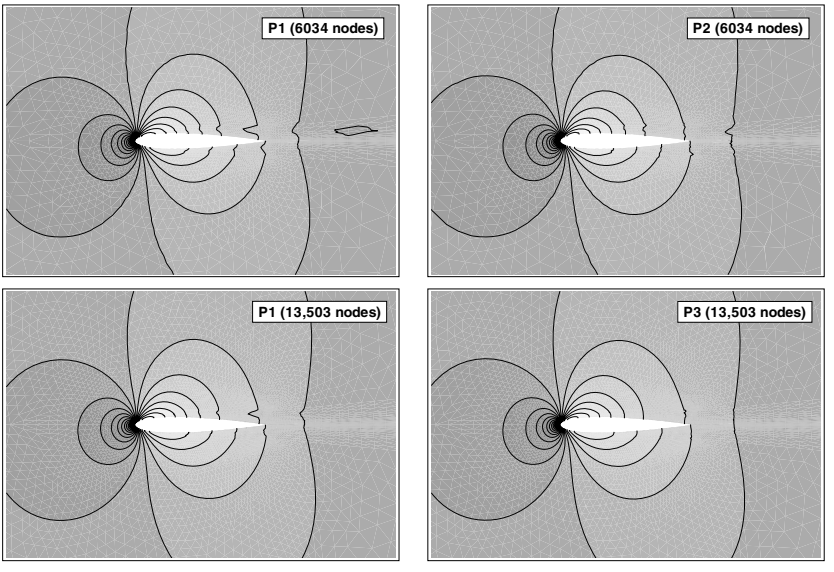


Fig. 10 MTC 3: NACA0012, $M = 0.50$, $\alpha = 2^\circ$, $Re = 5,000$. Pressure contours on matching P1 iso-P2 and P2 grids, and P1 iso-P3 and P3 grids.

of the scheme. Memory requirements are mostly due to the implicit Jacobian matrix. They respectively gain 30% and 70% for quadratic and cubic elements.

4.4 MTC 5: RAE2822, $M = 0.734$, $\alpha = 2.79^\circ$, $Re = 6,500,000$

The final test case deals with a transonic high Reynolds number RANS problem.

In the numerical method described in Section 2.1, the turbulence equations are solved in a staggered manner, with a second-order residual distribution scheme, and are weakly coupled to the Navier-Stokes field through the turbulent viscosity μ_t.

As a first step, for higher-order calculations, RANS turbulent equations are solved on an underlying P1 mesh, and thus remain second order accurate. These first results show the robustness of the SUPG finite element method. As for the more elementary MTC's (1, 2, and 3), the convergence of quadratic and cubic elements is similar to that obtained for linear elements with the same CFL settings. High aspect ratios (up to 2×10^6 in the considered set of meshes) do not seem to be an issue.

Figure 12 presents Mach number contours obtained with P1, P2, and P3 elements on matching grids. On these fairly coarse meshes, it's hard to see any difference between the solutions.

The force coefficient convergence plots are gathered in Figure 13. The open symbol curves represent the second-, third-, and fourth-order methods described above

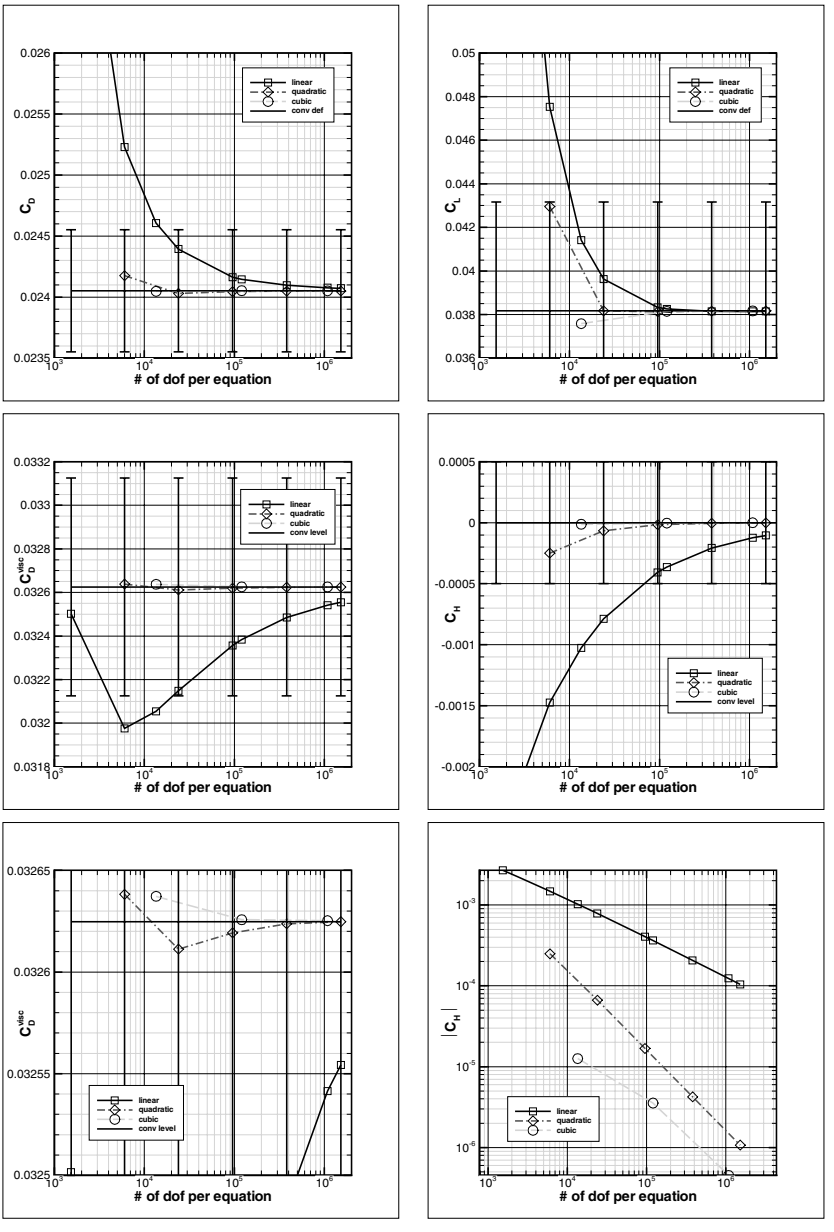


Fig. 11 MTC 3: NACA0012, $M = 0.50$, $\alpha = 2^\circ$, $Re = 5,000$. Convergence of force and heat flux coefficients for P1, P2, and P3 elements; estimated numbers of degrees of freedom and times for convergence.

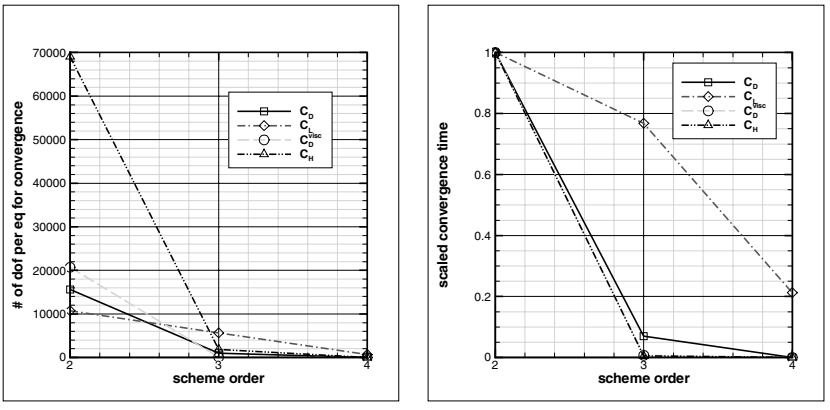


Fig. 11 (*continued*)

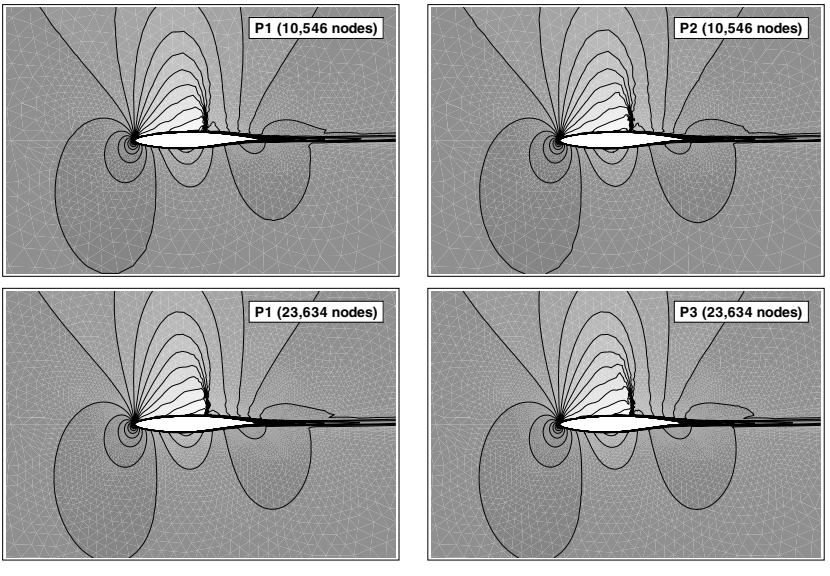


Fig. 12 MTC 5: RAE2822, $M = 0.734$, $\alpha = 2.79°$, $Re = 6,500,000$. Mach number contours on matching P1 iso-P2 and P2 grids, and P1 iso-P3 and P3 grids.

(with a second order turbulence model). There is no real distinction between the three schemes. They converge at the same rate toward the same asymptotic values. Nevertheless heat flux shows once more an indisputable advantage of higher-order elements over linear ones. The error is smaller by as much as three orders of magnitude. There is no additional benefit brought by cubic elements though.

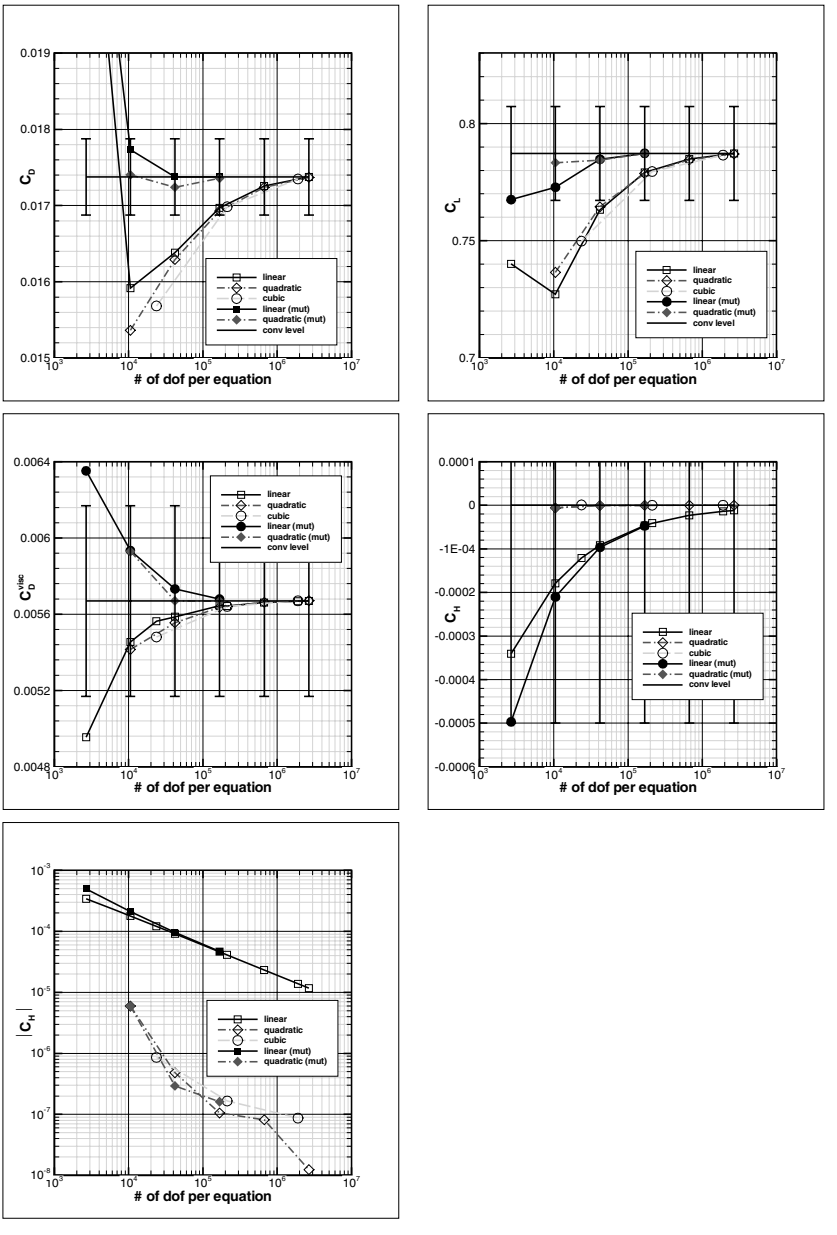


Fig. 13 MTC 5: RAE2822, $M = 0.734$, $\alpha = 2.79°$, $Re = 6,500,000$. Convergence of force and heat flux coefficients for P1, P2, and P3 elements

In an attempt to simulate a "higher-order" turbulence model, we used the interpolation of the μ_t field computed on the finest P1 mesh (2,669,536 nodes). The outcome of this test is indicated in the different convergence plots of Fig. 13 with filled symbols. We have only tested linear and quadratic elements. Results show that the turbulence model has a huge impact on the convergence of force coefficients. Quadratic elements have a slight edge over linear elements, especially for the coarsest meshes. Heat flux convergence is unaffected. This demonstrate the need for a higher-order turbulence model to fully exploit in RANS computations the benefit of higher-order elements observed in inviscid and laminar test cases.

4.5 Concluding Remarks on Numerical Test Cases

In this study, many firsts have been accomplished:

- the implementation of higher-order (quadratic and cubic) stabilized finite elements for compressible flows in an industrial code;
- the systematic convergence study of increasingly difficult test cases: inviscid, transonic, laminar, and turbulent flows;
- the proof that higher-order convergence can be achieved at a reasonable cost;
- the demonstration that higher-order elements are robust: same CFL rules where applied in our simulations with convergences similar to linear elements and sometimes significantly better;
- the verification that higher-order elements bring no particular complications in terms of implicitation nor parallel efficiency.

Difficulties were encountered with the RANS test cases. We believe they can be palliated with a stronger higher-order coupling between the Navier-Stokes solver and the turbulence model, or the use of a genuine higher-order scheme for solving the turbulence equations.

5 Towards Industrial Applications

As a conclusion we'll comment on the transition towards industrial applications. The extension to 3-D is readily available. To make it industrially viable, one needs a dedicated way to generate higher-order meshes. Enriching P1 meshes yields way too fine higher-order mesh sets in 2-D. This is even more true in 3-D.

The cost of higher-order elements is reasonable (at most a factor of 2 for P3 with the same number of dof's), and it can be worked upon.

Higher-order elements can handle high aspect ratios and same CFL's as the standard second-order scheme with convergences often better than with linear elements. They engender no implicit, nor parallel issue, which is mandatory for industrial applications.

The coupling with RANS turbulence model must be improved. In the mean time, higher-order elements might show a unique potential for Large Eddy Simulations.

References

1. Brooks, A.N., Hughes, T.T.R.: Streamline Upwind Petrov Galerkin formulation for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. Computer Methods in Applied Mechanics and Engineering 32, 199–259 (1982)
2. Chalot, F., Hughes, T.J.R., Johan, Z., Shakib, F.: Application of the Galerkin/least-squares formulation to the analysis of hypersonic flows. I: Flow over a two-dimensional ramp. In: Proceedings of a Workshop, Hypersonic Flows for Reentry Problems, Antibes, France, January 22-25. Test Cases – Experiments and Computations, vol. II, pp. 181–200. Springer, Heidelberg (1991)
3. Chalot, F., Hughes, T.J.R., Shakib, F.: Symmetrization of conservation laws with entropy for high-temperature hypersonic computations. Computing Systems in Engineering 1, 465–521 (1990)
4. Chalot, F., Hughes, T.J.R.: A consistent equilibrium chemistry algorithm for hypersonic flows. Computer Methods in Applied Mechanics and Engineering 112, 25–40 (1994)
5. Chalot, F., Mallet, M., Ravachol, M.: A comprehensive finite element Navier-Stokes solver for low- and high-speed aircraft design. Paper #94-0814. AIAA 32nd Aerospace Sciences Meeting, Reno, NV, January 10-13 (1994)
6. Chalot, F., Dinh, Q.V., Mallet, M., Naïm, A., Ravachol, M.: A multi-platform shared- or distributed-memory Navier-Stokes code. In: Parallel CFD 1997, Manchester, UK, May 19-21 (1997)
7. Chalot, F., Marquez, B., Ravachol, M., Ducros, F., Nicoud, F., Poinsot, T.: A consistent Finite Element approach to Large Eddy Simulation. Paper #98-2652. AIAA 29th Fluid Dynamics Conference, Albuquerque, NM, June 15-18 (1998)
8. Chalot, F., Marquez, B., Ravachol, M., Ducros, F., Poinsot, T.: Large Eddy Simulation of a compressible mixing layer: study of the mixing enhancement. Paper #99-3358. In: AIAA 14th Computational Fluid Dynamics Conference, Norfolk, VA, June 28-July 1 (1999)
9. Chalot, F.: Industrial aerodynamics. In: Stein, E., de Borst, R., Hughes, T.J.R. (eds.) Encyclopedia of Computational Mechanics. Computational Fluid Dynamics, vol. 3, ch. 12. Wiley, Chichester (2004)
10. Chalot, F., Levasseur, V., Mallet, M., Petit, G., Réau, N.: LES and DES simulations for aircraft design. Paper #2007-0723. 45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 8-11 (2007)
11. Hughes, T.J.R., Mallet, M.: A new finite element formulation for computational fluid dynamics: IV A discontinuity-capturing operator for multidimensional advective - diffusive systems. Comp. Meth. in Applied Mech. and Eng. 58, 329–336 (1986)
12. Shakib, F., Hughes, T.J.R., Johan, Z.: A multi-element group preconditioned GMRES algorithm for nonsymmetric systems arising in finite element analysis. Computer Methods in Applied Mechanics and Engineering 75, 415–456 (1989)
13. Shakib, F., Hughes, T.J.R., Johan, Z.: A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier-Stokes equations. Computer Methods in Applied Mechanics and Engineering 89, 141–219 (1991)

This page intentionally left blank

Chapter 12
A Third-Order Finite-Volume Residual-Based Scheme on Unstructured Grids

Xi Du, Christophe Corre, and Alain Lerat

Abstract. A residual-based compact (RBC) scheme originally designed on structured grids has been extended to unstructured grids. A second and third-order finite-volume (FV) formulations of the residual-based scheme have been proposed, which rely on a linear or quadratic least-square based solution reconstruction and an original dissipation flux. A simple matrix-free implicit scheme provides robustness and fast-convergence to a steady-state. The schemes are extended from a steady inviscid formulation to viscous and unsteady flows, keeping the same design principles and successively including new contributions to the residual. Computations of 2D and 3D external flows proposed in the ADIGMA project are presented and analyzed.

1 Introduction

The RBC scheme was proposed by Lerat and Corre in the early 2000s [1] [2]. The key idea is to take advantage of the residual r vanishing at steady-state to derive a compact and accurate discrete steady solution for the system of conservation laws $w_t + r = 0$, where for instance $r = f(w)_x + g(w)_y$ in the case of the $2D$ Euler equations. Another key ingredient is the design of matrix dissipation coefficients ensuring the scheme's dissipation hence robustness. A 3^{rd} order FV RBC was designed in [2] for solving the compressible Euler and Navier-Stokes equations and successfully applied to the computation of transonic flows around airfoils on curvilinear grids. The compactness of this third-order scheme (expressed on a $3 \times 3 \times 3$ stencil for the 3D Euler equations) allowed simple boundary treatments as well as enhanced

Xi Du · Alain Lerat
Arts et Métiers ParisTech, 151 Boulevard de l'hôpital, 75013 Paris, France
e-mail: xi.du-4@etudiants.ensam.eu, alain.lerat@paris.ensam.fr

Christophe Corre
LEGI UMR5519, Domaine universitaire BP 53, 38041 Grenoble Cedex 9, France
e-mail: christophe.corre@grenoble-inp.fr

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 167–180, 2010.
springerlink.com

convergence to steady-state when coupled with a simple first-order implicit stage. The scheme was further extended to unsteady flows within a dual or fictitious time approach, that is looking for steady solutions of $w_\tau + r = 0$, with τ the dual time and the residual $r = w_t + f_x + g_y$ now including the physical time-derivative [3] [5]. The scheme's structure was left unchanged with respect to the steady-case when expressed in terms of discrete expressions for the residual. Second- and third-order RBC scheme were applied to the RANS and URANS equations with a RSM turbulence modeling for computing flows with oscillating shocks in supersonic air intakes [6], in the frame of the French National Program "Recherche Aronautique sur le Supersonique". In the course of the DGAC AITEC program, these same schemes have also been implemented into the ONERA *elsA* code and applied to a variety of realistic steady and unsteady turbomachinery configurations. More recently, the RBC schemes have been extended to accuracy orders higher than 3 by constructing compact residual expressions deduced from particular Pade fractions [7] [8] [9]. For instance a 7^{th}-order scheme using a $5 \times 5 \times 5$ stencil has been applied to the $3D$ Euler equations. These very high order RBC scheme have been applied to aeroacoustic problems of the TurboNoise-CFD European Program, RANS simulations of oscillating shocks in nozzles and Euler simulations of spinning acoustic waves in aircraft engine intakes. Motivated by the successful development of the RBC schemes on structured grids, an effort toward their extension on unstructured grids was initiated in the late 2000s. A second-order RB scheme for computing compressible flows in general unstructured grids was proposed in [10]. Building on this first step, a third-order formulation and an improved second-order one were derived and systematically assessed within the ADIGMA project; these developments and some key results are summarized in this contribution.

2 Space Discretization

2.1 Linear and Quadratic Solution Reconstruction

The Euler and Navier-Stokes equations can be written in the form of a system of conservation laws:

$$\frac{\partial U}{\partial t} + \nabla \cdot \mathscr{F} = 0 \tag{1}$$

where U is the conservative variable vector, $\mathscr{F}$ is the flux tensor, such that $\mathscr{F} = \mathscr{F}^E(U)$ for inviscid flows and $\mathscr{F} = \mathscr{F}^E(U) - \mathscr{F}^V(U, \nabla U)$ for viscous flows. The FV discretization of (1) can be expressed as follows :

$$\frac{\partial U_i}{\partial t} + \frac{1}{|\Omega_i|} \sum_{k \in \mathscr{I}(\Omega_i)} \sum_g \omega_g (\mathscr{H}_{i,k})_g |\Gamma_{i,k}| = 0 \tag{2}$$

where U_i is the cell average value defined at the centroid of the control cell Ω_i with its volume $|\Omega_i|$; $\Gamma_{i,k}$ is the k-th face of this cell and $|\Gamma_{i,k}|$ its area. The set

$\mathscr{I}(\Omega_i)$ contains all the faces of cell i. A Gauss quadrature point along $\Gamma_{i,k}$ is indexed g, with ω_g its corresponding quadrature weight; $(\mathscr{H}_{i,k})_g$ denotes the numerical flux approximating the physical flux projected onto the $\Gamma_{i,k}$ face normal direction $\mathbf{n}_{i,k}$ at the Gauss point g. For the Euler equations, the inviscid numerical flux $(\mathscr{H}_{i,k})_g = (\mathscr{H}^E_{i,k})_g$ approximates $\mathscr{F}^E(U)\cdot\mathbf{n}_{i,k}$; for the Navier-Stokes equations, $(\mathscr{H}_{i,k})_g = (\mathscr{H}^E_{i,k})_g - (\mathscr{H}^V_{i,k})_g$, with the numerical viscous flux $(\mathscr{H}^V_{i,k})_g$ approximating $\mathscr{F}^V(U,\nabla U)\cdot\mathbf{n}_{i,k}$. The inviscid numerical flux of a conventional upwind scheme (Roe, AUSM+ ...) typically reads $(\mathscr{H}^E_{i,k})_g = \mathscr{H}^E((U^L_{i,k})_g,(U^R_{i,k})_g;\mathbf{n}_{i,k})$ where the numerical flux formula $\mathscr{H}^E$ depends on the scheme and $(U^L_{i,k})_g$, $(U^R_{i,k})_g$ are the reconstructed solutions at the Gauss-point g on face $\Gamma_{i,k}$, computed from polynomial representation in the left- and right-side cells. For a second-order scheme, the states $(U^{L/R}_{i,k})_g$ at the single Gauss-point (face center) are computed with the following linear polynomial :

$$(U^{L/R}_{i,k})_g = U_{i/o(i,k)} + \phi_{i/o(i,k)}(\mathbf{r}_g - \mathbf{r}_{i/o(i,k)})\cdot\nabla U_{i/o(i,k)} \tag{3}$$

where r is a position vector, with g the $\Gamma_{i,k}$ face Gauss point index, i and $o(i,k)$ the indices of the left and right cells sharing face $\Gamma_{i,k}$. The cell gradient $\nabla U_{i/o(i,k)}$ is evaluated at the cell centroid $C_{i/o(i,k)}$ using a linear least-square reconstruction with a fixed centered stencil; $\phi_{i/o(i,k)}$ is the Barth limiter as modified by Venkatakrishnan [11]. For a third-order scheme a quadratic polynomial is used to estimate states $(U^{L/R}_{i,k})_g$ at each Gauss-point g on the face $\Gamma_{i,k}$ (2 points per face in $2D$) :

$$\begin{aligned}(U^{L/R}_{i,k})_g = U_{i/o(i,k)} &+ ((1-\sigma_{i/o(i,k)})\phi_{i/o(i,k)} + \sigma_{i/o(i,k)})(\mathbf{r}_g - \mathbf{r}_{i/o(i,k)})\cdot\check{\nabla}U_{i/o(i,k)} \\ &+ \frac{1}{2}\sigma_{i/o(i,k)}(\mathbf{r}_g - \mathbf{r}_{i/o(i,k)})^T\cdot\mathbf{H}_{i/o(i,k)}\cdot(\mathbf{r}_g - \mathbf{r}_{i/o(i,k)})\end{aligned} \tag{4}$$

where the second-order approximation $\check{\nabla}U_{i/o(i,k)}$ of the cell gradient and the first-order approximation of the cell Hessian $\mathbf{H}_{i/o(i,k)}$ are computed at each cell centroid using a quadratic least-square reconstruction, which makes use of an extended fixed centered stencil [12]. The reconstruction formula also includes a sensor $\sigma_{i/o(i,k)}$ which allows a smooth transition between the limited linear reconstruction in high-gradient flow regions ($\sigma_{i/o(i,k)} \to 0$) and the quadratic reconstruction in smooth flow regions ($\sigma_{i/o(i,k)} \to 1$).

2.2 Numerical Flux Computation

The RB invisicd numerical flux takes the following form:

$$(\mathscr{H}^{RB}_{i,k})_g = \left(\mathscr{H}^c_{i,k}\right)_g - (d_{i,k})_g = \mathscr{H}^c((U^L_{i,k})_g,(U^R_{i,k})_g) - (d_{i,k})_g \tag{5}$$

where $\left(\mathscr{H}^c_{i,k}\right)_g$ is a purely centered (non-dissipative) approximation of the physical normal flux vector $(\mathscr{F}^E)^\perp$ computed at the Gauss-point g of face $\Gamma_{i,k}$; $(d_{i,k})_g$ is the

dissipation flux computed at the same Gauss point, which is fully specific to the RB scheme since it does not rely on reconstructed states, as for conventional upwind schemes, but on an estimate of the residual r associated with system (1). Note this dissipation flux is computed only once on a given face $\Gamma_{i,k}$ and shared by all the face Gauss-points; its expression reads :

$$(d_{i,k})_g = d_{i,k} = \frac{1}{2} L^{\perp}_{i,k} \Phi_{i,k} \mathscr{R}_{i,k} \tag{6}$$

where $L^{\perp}_{i,k} = \mathbf{r}_{C_i,C_{o(i,k)}} \cdot \mathbf{n}_{i,k}$ is the projection onto the face normal of the distance between the cell centroids C_i and $C_{o(i,k)}$, $\Phi_{i,k}$ is a matrix coefficient of order $\mathscr{O}(1)$ designed so as to ensure the dissipation of the scheme and $\mathscr{R}_{i,k}$ is an approximation of the system residual $R_{i,k}$ defined as :

$$R_{i,k} = \frac{1}{|\Omega_{i,k}|} \int_{\Omega_{i,k}} r \, d\Omega \tag{7}$$

The shift cell $\Omega_{i,k}$ on which the integral form $R_{i,k}$ of the residual r is defined is formed by the nodes $N^1_{i,k}$, $N^2_{i,k}$of face $\Gamma_{i,k}$ and the two cell centroids C_i and $C_{o(i,k)}$ (see Figure 1). Since $L^{\perp}_{i,k} = O(h)$, with h the characteristic mesh size, a second-order estimate for $R_{i,k}$ will lead to a third-order dissipation and the global accuracy (second- or third-order will be eventually controlled by the polynomial reconstruction used in the purely centered flux $\mathscr{H}^c$.

In the inviscid case, the residual r is given by $r = \nabla \cdot \mathscr{F}^E$, so that the second-order approximation of (7) can be expressed as:

$$\mathscr{R}_{i,k} = \frac{1}{|\Omega_{i,k}|} \sum_{l \in \mathscr{I}(\Omega_{i,k})} \left(\mathscr{H}^E_c\right)_l |\Gamma_l| \tag{8}$$

with $(\mathscr{H}^E_c)_l$ an approximation of the normal physical inviscid flux $(\mathscr{F}^E)^{\perp}_l$ at the center of face Γ_l of the shifted cell $\Omega_{i,k}$. Using the trapezoidal rule on a face Γ_l with vertices $(N^1_{i,k}, C_o)$ (see Figure 1), this flux is computed as $(\mathscr{H}^E_c)_l = \frac{1}{2}\left(\mathscr{F}^E(U_{N^1_{i,k}}) + \mathscr{F}^E(U_{C_o})\right) \cdot \mathbf{n}_l$. The required node values are computed from

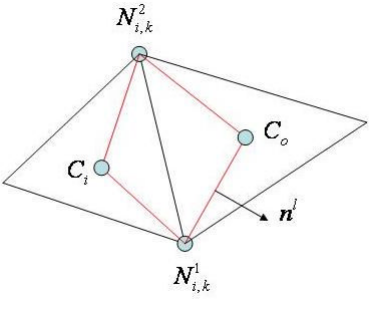


Fig. 1 Shift cell used for computing the RB dissipation in 2D

neighboring cell-center values using the average formula proposed by Holmes and Connell [13], which is found sufficient to yield a second-order dissipation. When building the third-order RB scheme, a more accurate node estimate technique is needed to ensure a second-order discretization of (8). Since the solution gradient and Hessian are available in each cell, the node value can be extrapolated by using (4) in each cell containing the node and averaging these reconstructed values.

In order to define the matrix dissipation coefficient $\Phi_{i,k}$ in (6), let us introduce the *normal* and *tangential* Jacobian matrices associated with face $\Gamma_{i,k}$, respectively $(J_\perp)_{i,k} = A^E_{i,k}(n_x)_{i,k} + B^E_{i,k}(n_y)_{i,k}$ and $(J_\parallel)_{i,k} = A^E_{i,k}(t_x)_{i,k} + B^E_{i,k}(t_y)_{i,k}$, where A^E, B^E are the Jacobian matrices of the inviscid fluxes f^E and g^E for the 2D Euler system, $\mathbf{t}_{i,k}$ is the unit vector tangent to $\Gamma_{i,k}$. Let us also denote $\lambda_\perp^{(l)}$ (resp. $\lambda_\parallel^{(l)}$) the l^{th} eigenvalue of $J_\perp$ (resp. $J_\parallel$), $T_\perp$ (resp. $T_\parallel$) the matrix whose columns are the eigenvectors associated with the eigenvalues $\lambda_\perp^{(l)}$ (resp. $\lambda_\parallel^{(l)}$). The dissipation matrix $\Phi_{i,k}$ has the same eigenvectors as $(J_\perp)_{i,k}$, that is:

$$\Phi_{i,k} = (T_\perp)_{i,k} \cdot Diag(\phi_{i,k}^{(l)}) \cdot (T_\perp^{-1})_{i,k} \tag{9}$$

with the diagonal matrix $Diag(\phi_{i,k}^{(l)})$ defined from the eigenvalues of $J_\perp$ and $J_\parallel$:

$$\phi_{i,k}^{(l)} = sign((\lambda_\perp^{(l)})_{i,k}) min(1, \frac{|\Gamma_{i,k}|}{|L_{i,k}^\perp|} \frac{|(\lambda_\perp^{(l)})_{i,k}|}{m(J_\parallel)_{i,k}}) \tag{10}$$

where $m(J_\parallel) = min_l(|\lambda_\parallel^{(l)}|)$. Eigenvalues and eigenvectors at face $\Gamma_{i,k}$ are computed by a Roe-average between the solutions in the cells sharing the face. More details on the design principles of the residual-based scheme can be found in [1] [2].

The calculation of the viscous numerical flux $\mathscr{H}_{i,k}^V$ requires an estimate of the solution and its gradients at each Gauss point. For a second-order scheme, these quantities are computed at the single Gauss point (face center M in Figure 2) using a simple average of node quantities:

$$(U_{i,k})_g = U_M = \tfrac{1}{2}(U_{N^1_{i,k}} + U_{N^2_{i,k}}) \ , \ (\nabla U_{i,k})_g = \nabla U_M = \tfrac{1}{2}(\nabla U_{N^1_{i,k}} + \nabla U_{N^2_{i,k}}) \tag{11}$$

using the aforementioned Holmes and Connell formula for node estimates. For the third-order scheme, the solution at each Gauss point on face $\Gamma_{i,k}$ is obtained by an average of the quadratically reconstructed states on both sides of the face $(U_{i,k})_g = \frac{1}{2}((U^L_{i,k})_g + (U^R_{i,k})_g)$. The second-order gradient estimate at the quadrature point g is computed as $(\breve{\nabla} U_{i,k})_g = \frac{1}{2}((\breve{\nabla} U_g)^L + (\breve{\nabla} U_g)^R)$ where the reconstructed gradients are obtained from the gradient and Hessian computed at each cell centroid :

$$(\breve{\nabla} U_g)^L = \breve{\nabla} U_i + \mathbf{H}_i \cdot \mathbf{r}_{i,g} \ , \ (\breve{\nabla} U_g)^R = \breve{\nabla} U_o + \mathbf{H}_o \cdot \mathbf{r}_{o,g} \tag{12}$$

A fundamental difference between the RB scheme and a conventional upwind scheme is the fact the residual used in the dissipation flux (6) adapts itself to the

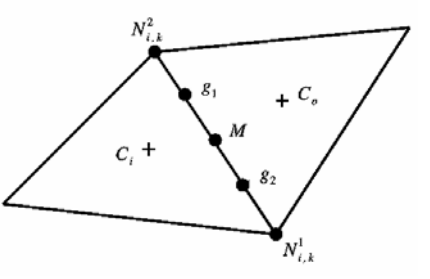


Fig. 2 Gauss quadrature points used for flux integral computation

problem solved. Since the residual associated to the Navier-Stokes equations reads $r = \nabla \cdot (\mathscr{F}^E - \mathscr{F}^V)$, the approximation of (7) becomes:

$$\mathscr{R}_{i,k} = \frac{1}{|\Omega_{i,k}|} \sum_{l \in \mathscr{I}(\Omega_{i,k})} \left(\mathscr{H}_c^E - \mathscr{H}_c^V\right)_l |\Gamma_l|, \tag{13}$$

where the inviscid numerical flux $\mathscr{H}_c^E$ is computed as described in the inviscid case. The numerical viscous flux $\mathscr{H}_c^V$ at each face center of the shift cell is also computed by the trapezoidal rule :

$$(\mathscr{H}_c^V)_l = \frac{1}{2}\left(F^V(U_{N^1_{i,k}}, \nabla U_{N^1_{i,k}}) + F^V(U_{C_o}, \nabla U_{C_o})\right) \cdot \mathbf{n}_l \tag{14}$$

Since the gradient at cell centroid is already available, only node gradients introduce extra computations. The node gradient is calculated by the linear least square reconstruction with a centered stencil including all the cells sharing the node. Note the definition of the dissipation matrix $\Phi_{i,k}$ in (6) remains unchanged with respect to the inviscid case : this simple choice does not impact the overall accuracy (since this term is $O(1)$) nor compromise the robustness.

2.3 Boundary Conditions

The specific design of the RB scheme requires to deal not only with the faces located on the computational domain boundaries but also with the faces containing a single boundary node (in $2D$) since the state at this boundary node is involved in the flux balance evaluated on the shift cell $\Omega_{i,k}$. Two types of boundaries have been specifically considered : far-field and wall boundary. On the far-field boundary the physical state is computed with a characteristic-based non-reflecting boundary condition, using the interior solution and the prescribed far-field state; the flux through the boundary face is then computed applying the physical flux formula with this boundary state. No specific treatment on boundary nodes is performed at the far-field. For inviscid flows, the slip-wall condition is imposed in the physical flux computed on a wall face and this boundary condition is also imposed for wall nodes. For viscous flows, the no-slip wall condition is similarly imposed.

3 Time Integration

3.1 Implicit Time Integration for Steady Problems

In order to speed up the convergence to a steady-state, the solution update is obtained from the following implicit scheme :

$$\frac{\Delta U_i^n}{\Delta t_i} + \frac{1}{|\Omega_i|} \sum_{k \in \mathscr{I}(\Omega_i)} (\Delta \mathscr{H}_{i,k}^{imp})^n \, |\Gamma_{i,k}| = -\mathscr{R}_i^n, \tag{15}$$

where the explicit stage $\mathscr{R}_i^n = \dfrac{1}{|\Omega_i|} \displaystyle\sum_{k \in \mathscr{I}(\Omega_i)} \sum_g \omega_g (\mathscr{H}_{i,k})_g |\Gamma_{i,k}|$ is either based on the RB scheme or on a conventional upwind scheme and $\Delta\phi^n = \phi^{n+1} - \phi^n$ is the time-increment of the grid quantity ϕ (solution U_i or implicit numerical flux $\mathscr{H}_{i,k}^{imp}$). Ideally, for maximal intrinsic efficiency, the implicit numerical flux should be taken equal to its explicit counterpart $\mathscr{H}_{i,k}$. However, such a choice would lead to a large computational cost per iteration, cancelling out the benefit of high intrinsic efficiency. Instead, the strategy followed is to retain for $\mathscr{H}^{imp}$ a simple implicit stage, sufficient to ensure robustness (unconditional stability for large time-steps, except for highly non-linear problems involving very strong discontinuities) and leading to an inexpensive solution of the implicit stage. Following ideas originally proposed in [14], the implicit numerical flux increment is computed as $(\Delta \mathscr{H}^{imp})^n = (\Delta \mathscr{H}^{E(imp)})^n - (\Delta \mathscr{H}^{V(imp)})^n$ with the inviscid and viscous contributions defined as :

$$\begin{cases} (\Delta \mathscr{H}_{i,k}^{E(imp)})^n = \dfrac{1}{2}[(\Delta \mathscr{F}_{i,k}^{E})^n \cdot \mathbf{n}_{i,k} - (\rho_\perp^E)_{i,k}^n (\Delta U_{o(i,k)}^n - \Delta U_i^n)] \\ \\ (\Delta \mathscr{H}_{i,k}^{V(imp)})^n = \dfrac{(\rho_\perp^V)_{i,k}^n}{|\mathbf{r}_{i,o} \cdot \mathbf{n}_{i,k}|} (\Delta U_{o(i,k)}^n - \Delta U_i^n) \end{cases} \tag{16}$$

where the scalar coefficients $(\rho_\perp^E)_{i,k}^n$, $(\rho_\perp^V)_{i,k}^n$ are the respective spectral radii of the normal inviscid and viscous Jacobian matrices. Inserting (16) into (15) leads to a so-called matrix-free implicit scheme of the form:

$$D_i^n \Delta U_i^n = -\mathscr{R}_i^n - \frac{1}{2|\Omega_i|} \sum_{k \in \mathscr{I}(\Omega_i)} \left((\Delta F_{i,k}^E)^n \cdot \mathbf{n}_{i,k} \; - C_{i,k}^n \Delta U_{o(i,k)}^n \right) |\Gamma_{i,k}| \tag{17}$$

with the scalar coefficients $C_{i,k}^n$ and D_i^n defined by:

$$C_{i,k}^n = (\rho_\perp^E)_{i,k}^n + \frac{2(\rho_\perp^V)_{i,k}^n}{|\mathbf{r}_{i,o} \cdot \mathbf{n}_{i,k}|} \;, \; D_i^n = \frac{1}{\Delta t_i} + \frac{1}{2|\Omega_i|} \sum_{k \in \mathscr{I}(\Omega_i)} C_{i,k}^n |\Gamma_{i,k}| \tag{18}$$

System (17) is then solved by a simple Point-Jacobi relaxation technique with a very low cost per iteration, which makes up for the limited intrinsic efficiency and yields a globally efficient implicit strategy.

3.2 Dual-Time Integration for Unsteady Problems

Unsteady solutions of the 2*D* Euler equations are found as steady-state with respect to the fictitious or dual-time τ of the following implicit scheme :

$$\begin{cases} \dfrac{\Delta U_i^{n,m}}{\Delta \tau_i^{n,m}} + \dfrac{3}{2}\dfrac{\Delta U_i^{n,m}}{\Delta t} + \dfrac{1}{|\Omega_i|}\displaystyle\sum_{k\in\mathscr{I}(\Omega_i)} (\Delta \mathscr{H}_{i,k}^{imp})^{n,m}\,|\Gamma_{i,k}| = -\mathscr{R}_i^{n,m}, \\ \mathscr{R}_i^{n,m} = \mathscr{T}(U_i^{n,m}, U_i^n, U_i^{n-1}) + \dfrac{1}{|\Omega_i|}\displaystyle\sum_{k\in\mathscr{I}(\Omega_i)}\sum_g \omega_g \left(\mathscr{H}_{i,k}^E\right)_g^{n,m}\,|\Gamma_{i,k}|, \end{cases} \tag{19}$$

where m and n are respectively the dual-time and physical-time iteration counter, $\Delta U_i^{n,m} = U_i^{n,m+1} - U_i^{n,m}$ with $U_i^{n,0} = U_i^n$ and the three-level time-discretization operator $\mathscr{T}$ is chosen such that :

$$\mathscr{T}(U_i^{n,m}, U_i^n, U_i^{n-1}) = \frac{\frac{3}{2}(U_i^{n,m} - U_i^n) - \frac{1}{2}(U_i^{n-1} - U_i^n)}{\Delta t} = (U_t)_i^{n+1} + \mathscr{O}(\Delta t^2), \tag{20}$$

When the pseudo-time marching reaches a steady solution $U^{n+1} = U^{n,m+1} = U^{n,m}$, scheme (19)-(20) yields an approximation of the physical unsteady flow solution at order 2 in time and 2 or 3 in space, depending on the use of a linear or quadratic solution reconstruction for a conventional scheme. When the RB numerical flux formula (5) is used for computing $\mathscr{H}_{i,k}^E$ in (19), care must be taken to compute the third-order RB dissipation flux with a residual r that includes the physical time derivative : $r = U_t + \nabla\cdot\mathscr{F}^E$. The residual integral (7) is then discretized as $\mathscr{R}_{i,k} = \mathscr{R}_{i,k}^t + \mathscr{R}_{i,k}^E$ where $\mathscr{R}_{i,k}^E$ approximates the inviscid flux balance over the shift cell $\Omega_{i,k}$ and remains unchanged with respect to the steady case (using $U^{n,m}$ instead of U^n) while $\mathscr{R}_{i,k}^t$ approximates the physical time-derivative over the shift cell as :

$$\mathscr{R}_{i,k}^t = \frac{1}{2}\left(\mathscr{T}(U_i^{n,m}, U_i^n, U_i^{n-1}) + \mathscr{T}(U_{o(i,k)}^{n,m}, w_{o(i,k)}^n, U_{o(i,k)}^{n-1})\right). \tag{21}$$

4 Numerical Results

Some selected results taken from the ADIGMA Mandatory and Baseline Test Cases suites are now presented and analyzed from the viewpoint of assessing the interest of the third-order RB scheme over the baseline second-order RB scheme.

4.1 MTC1: Subsonic Inviscid Flow over a NACA0012 Airfoil

The inviscid flow at upstream Mach number $M_\infty = 0.5$ and angle of attack $\alpha = 2^\circ$ over the NACA0012 airfoil is first considered. A series of 9 increasingly refined unstructured meshes, provided within the ADIGMA project and mainly made of quadrilateral elements, is used to assess the grid convergence properties of the second- and third-order RB schemes. The coarsest grid, **Mesh1**, contains 206 **dof** and 16 faces on the airfoil while the finest grid, **Mesh9**, contains 41685 **dof** and 926

faces on the airfoil (see Fig. 3 for a view of **mesh9** along with the computed Mach contours on this fine grid). The evolution of the computed global aerodynamic coefficients (lift and drag - moment is not shown for space reason) is plotted in Fig. 4, along with a so-called convergence zone defined by the value obtained on the finest grid and a tolerance interval based on industrial experience and provided in the ADIGMA project. The tolerance intervals for the lift, drag and moment coefficient were respectively for $MTC1 : E_{Cl} = 1 \times 10^{-3}$, $E_{Cd} = 1 \times 10^{-4}$ and $E_{Cm} = 2 \times 10^{-4}$. A key observation is all coefficients are converged within the tolerance interval on mesh level 7 when RB $O3$ is used while mesh level 8 is needed when computing with RB $O2$. With on one hand a ratio of **dof** of more than 2 between **mesh7** and **mesh8**, a faster convergence on the coarser grid **mesh7** and on the other hand a cost per point per iteration for RB $O3$ about 25% higher than RB $O2$, the CPU gain offered by the third-order scheme with respect to the second-order one is about 60%

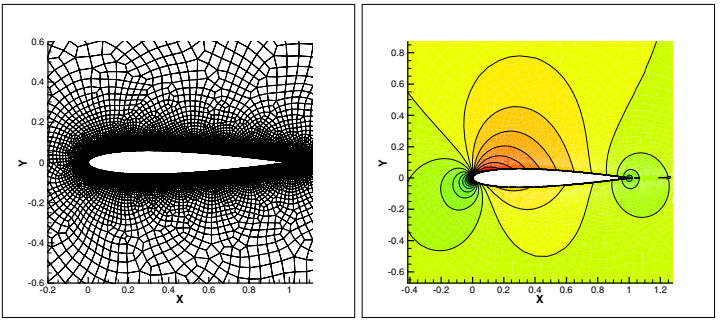


Fig. 3 MTC1. **Mesh9** (left) and Mach contours (30 levels from 0 to 0.7) computed with the $O3$ RB scheme.

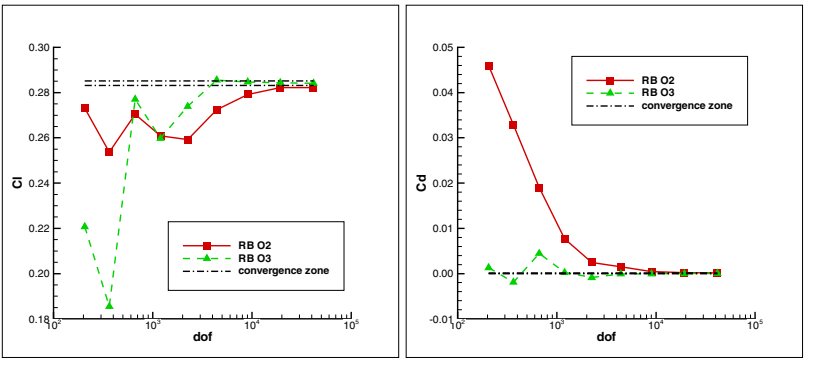


Fig. 4 MTC1. Aerodynamic coefficients convergence with **dof** based on meshes from **mesh1** to **mesh9** obtained by the second and third-order FV-RB scheme.

and the memory storage reduction goes up to 20%. Note the predicted lift coefficients on **mesh9** differ of 0.7% only ($C_L = 0.282181$ with RB $O2$, $C_L = 0.284150$ with RB $O3$); the computed RB $O3$ drag coefficient is closer to its ideal zero value ($C_D = 4.62 \times 10^{-5}$ for RB $O3$ against $C_D = 1.67 \times 10^{-4}$ for RB $O2$).

4.2 MTC2: Transonic Inviscid Flow Over a NACA0012 Airfoil

The FV RB schemes have also been applied to flow problems involving discontinuities, such as the steady flow at upstream Mach number $M_\infty = 0.8$ and angle of attack $\alpha = 1.25^\circ$ over the NACA0012 airfoil. The pressure coefficient distributions computed with RB $O2$ and RB $O3$ on a rather fine unstructured mesh with 26384 pure triangle elements are plotted, along with a view of the grid, in Fig.5. The single tuning parameter for the second-order scheme is the coefficient K used in the slope limiter; for the third-order scheme, the parameters S and β appearing in the switch from quadratic to limited linear reconstruction must also be tuned. The values eventually retained are summarized in Table 1; they have been systematically used for transonic flows and found to yield oscillation-free results. The second-order and third-order results are very close to each other and compare also well with a reference result obtained on a very fine grid by another ADIGMA partner.

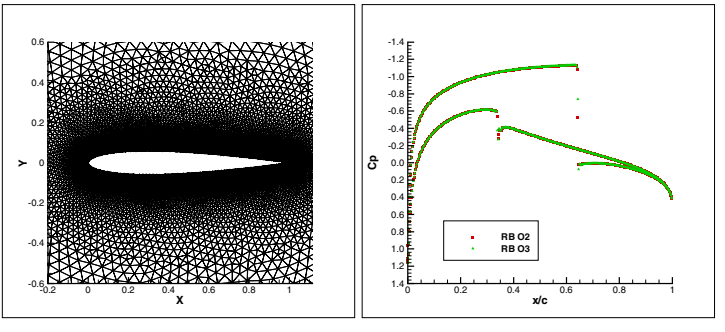


Fig. 5 MTC2. Computational mesh (left) and pressure coefficient distributions (left) computed with the $O2$ and $O3$ RB schemes.

Table 1 Tuning parameters used with the FV-RB schemes for transonic flows. Computed aerodynamic coefficients obtained on a triangular mesh with 26384 **dof**. The reference result is a second-order DLR computation on a very fine structured C-mesh with 115584 **dof**.

Scheme	K	S	β	C_L	C_D	C_M
RB $O2$	2	-	-	0.358254	0.023006	-0.040901
RB $O3$	2	160	0.03	0.360094	0.022934	-0.041374
Ref. result	-	-	-	0.357895	0.022736	-0.038646

4.3 MTC3S: Subsonic Laminar Flow over a NACA0012 Airfoil

The performance of the FV-RB schemes for solving the Navier-Stokes equations is assessed on a steady flow over the NACA0012 airfoil with upstream Mach number $M_\infty = 0.5$, zero angle of attack and Reynolds number (based on the airfoil chord and the far-field conditions) $Re_{\infty,c} = 500$. A series of 5 increasingly refined triangular meshes is used for the computations; the coarsest grid has 2262 **dof** and 16 faces on the airfoil while the finest grid **mesh5** contains 26384 **dof** and 700 faces on the airfoil (see Fig. 6 for an overview of this grid and the computed Mach contours with RB $O3$). The grid convergence analysis is performed following the methodology described for MTC1 with $E_{C_l} = \pm 1 \times 10^{-3}$, $E_{C_d} = \pm 5 \times 10^{-4}$ and $E_{C_m} = \pm 2 \times 10^{-4}$. The evolution of the global aerodynamic coefficients when the number of **dof** increases is plotted in Fig.7. The prescribed tolerance intervals are found a bit too loose; with more stringent requirements, the RB $O2$ scheme does not provide grid-converged coefficients before **mesh5** is used. Meanwhile, grid-convergence is achieved on all coefficients using RB $O3$ on the grid level 4, containing 17210 **dof**. Taking into account the extra-cost induced by the third-order scheme as well as the faster convergence on a coarser grid, the gain offered by RB $O3$ with respect to RB $O2$ amounts to a 24% CPU time reduction and 21% memory requirement reduction. The computed coefficients using RB $O2$ and RB $O3$ on the finest mesh **mesh5** are provided in Table 2; these grid-converged values remain very close to each other.

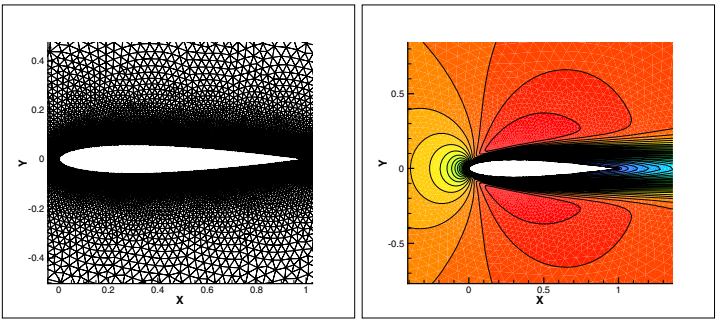


Fig. 6 MTC3S. **Mesh5**(left) and Mach number contours (right) computed with the FV-RB $O3$ scheme.

Table 2 MTC3S. Aerodynamic coefficients obtained with the FV-RB scheme on **mesh5**.

Scheme	C_l	C_d	C_{dp}	C_{dv}	C_m
$O2$	-1.56×10^{-4}	0.181513	0.048751	0.132762	-2.31×10^{-5}
$O3$	-1.04×10^{-4}	0.181808	0.049580	0.132228	-2.59×10^{-5}

4.4 BTC0_Euler: Subsonic Inviscid Flow over a 3D Airfoil Body

The RB schemes are extended to the 3D Euler equations and applied to the computation of the subsonic flow over a 3D streamlined body, with upstream Mach number $M_\infty = 0.5$ and angle of attack $\alpha = 1^\circ$. A grid convergence study is performed on a limited series of 3 unstructured meshes made of pure tetrahedral elements and provided within the ADIGMA project. The coarsest grid **Mesh1** contains 191753 **dof** while the finest grid **Mesh3** contains 440494 **dof**. The body geometry is displayed in Fig.8 along with the surface mesh corresponding to **Mesh3**. The evolution of the global aerodynamic coefficients computed with the RB $O2$ and $O3$ schemes for an increasing number of **dof** is plotted in Fig.9. These 3D results differ from the previous 2D analysis on a smooth inviscid flow since second- and third-order results remain systematically very close to each other, even on the coarsest grid. No benefit from using RB $O3$ can be perceived on this test-case since the level of accuracy similar to that of RB $O2$ is achieved for a higher cost, both in CPU and memory storage. Note also the drag coefficient is far from grid convergence; the value of C_D obtained by NLR on a fine structured grid containing 1572864 **dof** with a

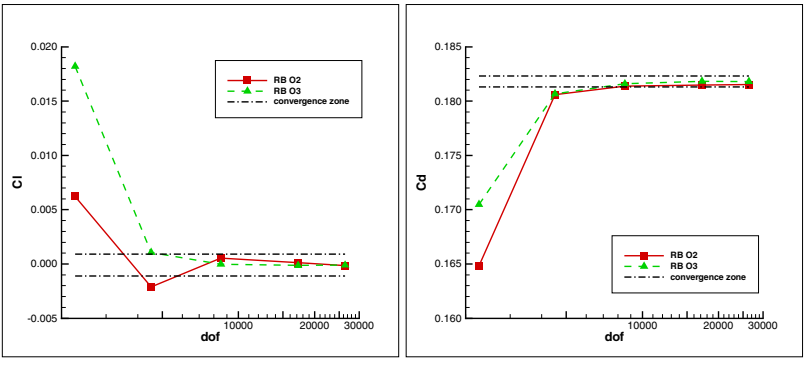


Fig. 7 MTC3S. Aerodynamic coefficients convergence with dof based on meshes from **mesh1** to **mesh5** obtained by the FV-RB scheme.

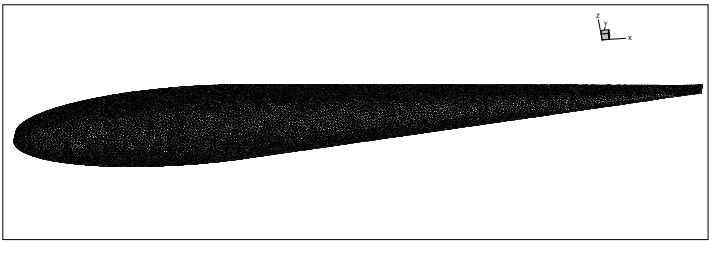


Fig. 8 BTC0. **Mesh3** surface mesh for the BTC0 body.

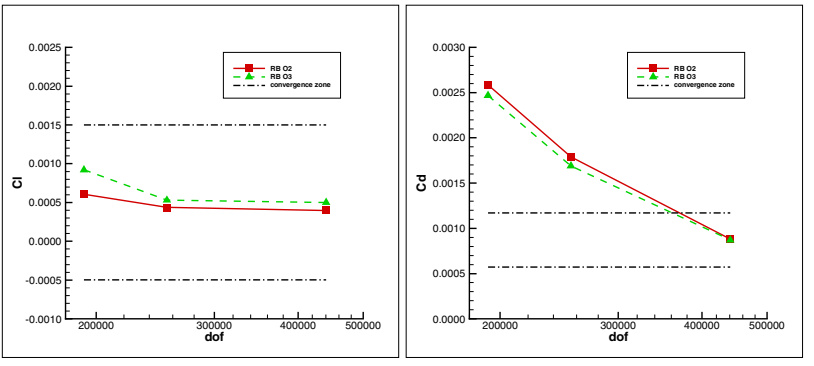


Fig. 9 BTC0. Aerodynamic coefficients convergence with **dof** based on meshes from **Mesh1** to **Mesh3** obtained by the second and third-order FV-RB scheme.

second-order discontinuous Galerkin method was found equal to $C_d = 5.3076 \times 10^{-4}$ (vs $C_d = 8.7147 \times 10^{-4}$ on **Mesh3** with the RB $O3$ scheme).

5 Conclusion

The improved efficiency of a third-order RB scheme over a second-order version has been demonstrated for $2D$ inviscid and viscous compressible flows through grid-convergence studies performed on series of increasingly refined unstructured grids made available in the ADIGMA project. The benefits of using the third-order scheme were not established however in the $3D$ case where both second- and third-order schemes yielded very close results, probably because of the excessively large stencils used to compute the cell gradient and Hessian estimates. A way of improvement would be to compute the non-dissipative part of the RB numerical flux using a face-centered least-square approach rather than the usual cell-reconstructed solutions - the dissipative part remaining computed as described in the present paper; this strategy will be investigated in the near future.

References

1. Lerat, A., Corre, C.: Residual based compact schemes for multidimensional hyperbolic systems of conservation laws. Computers and Fluids 31, 639–661 (2002)
2. Lerat, A., Corre, C.: A residual based compact scheme for compressible Navier-Stokes equations. Journal of Computational Physics 170, 642–675 (2001)
3. Hanss, G.: Schémas numériques compacts basés sur le résidu en maillage irrégulier pour les équations de Navier-Stokes en compressible. Phd thesis, ENSAM, Paris (2002)
4. Lerat, A., Corre, C., Hanss, G.: Efficient high-order schemes on non-uniform meshes for multi-dimensional compressible flows. In: Caughey, D.A., Hafey, M.M. (eds.) Frontiers of Computational Fluid Dynamics, pp. 89–112. World Scientific, Singapore (2002)

5. Corre, C., Hanss, G., Lerat, A.: A residual based compact scheme for the unsteady compressible Navier-Stokes equations. Computers and Fluids 34, 561–580 (2005)
6. Michel, B.: Contribution á la simulation numérique efficace des écoulements dans les prises d'air supersoniques. Phd thesis, ENSAM, Paris (2004)
7. Lerat, A., Corre, C.: High order residual-based compact schemes on structured grids. In: VKI Lecture Series 2006-01, Belgium (2006)
8. Corre, C., Falissard, F., Lerat, A.: High-order residual based compact schemes for compressible inviscid flows. Computers and Fluids 36, 1567–1582 (2007)
9. Corre, C., Lerat, A.: High-order residual based compact schemes for advection-diffusion problems. Computers and Fluids 37, 505–519 (2008)
10. Corre, C., Du, X.: A residual-based scheme for computing compressible flows on unstructured grids. Computers and Fluids 38, 1338–1347 (2009)
11. Venkatakrishnan, V.: Convergence to steady-state solutions of the Euler equations on unstructured grids with limiters. Journal of Computational Physics 118, 120–130 (1995)
12. Vaassen, J.M., Wautelet, P., Essers, J.A.: Application of a third-order reconstruction scheme to hypersonic reacting flows using unstructured meshes and multigrid techniques. J. of Comput. and Applied Math. 168, 481–489 (2004)
13. Holmes, D.G., Connell, S.D.: Solution of the 2D Navier-Stokes equations on unstructured adaptive grids. AIAA Paper, 89–1932 (1989)
14. Luo, H., Baum, J.D., Lohner, R.: An accurate, fast, matrix-free implicit method for computing unsteady flows on unstructured grids. Computers and Fluids 30, 137–159 (2001)

Chapter 13
Investigation of Issues Relating to Meshing for Higher-Order Discretizations

Craig Johnston and Jeremy Gould

Abstract. During the ADIGMA project ARA have investigated various aspects relating to grid generation for higher-order solvers. Within the industrial hybrid mesh generation code SOLAR we have both implemented a higher-order boundary representation, and have propagated the influence of this boundary into the volume mesh. Although subject to some minor limitations, this capability has a near industrial level of robustness and has successfully generated grids for a variety of geometries of industrial interest, including the ONERA M6 wing and DLR-F6 wing-body. A mesh-quality toolkit for analysis of the SOLAR higher-order grids has been developed. This code has a modular design, to allow forsimple implementation of new metrics, and currently includes a number of metrics which have been used to analyse the grids generated within the project.

1 Introduction

In order to fully exploit the developments in higher-order solver algorithms from within ADIGMA it is essential that appropriate meshes can be generated. Such a mesh must allow the algorithms to be applied robustly and efficiently for 3D complex industrial configurations. Within ADIGMA, ARA have investigated and addressed several of the issues that needed to be resolved in order to allow the generation of meshes which achieve this aim.

One of the key motivations in this work is the fact that a higher-order solver has a computational expense significantly higher than a standard second-order RANS solver running on the same mesh. This increased cost is mitigated by the higher-order scheme requiring fewer overall grid elements/points to achieve a solution accuracy equivalent to that of the RANS solver. However, such a coarse mesh is

Craig Johnston · Jeremy Gould
ARA Ltd, Manton Lane, Bedford, United Kingdom, MK41 7PF
e-mail: cjohnston@ara.co.uk, jgould@ara.co.uk

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 181–193, 2010.
springerlink.com

still required to accurately represent the underlying geometry and, in particular, resolve regions of high curvature. This in turn motivates the need for a higher-order boundary representation (HO-BREP) that will enable highly curved regions to be accurately resolved with a reduced number of grid points. We describe in Section 3 how a HO-BREP was successfully implemented within the SOLAR mesh generation system [2, 3, 4, 5].

In terms of the overall mesh generation process, the HO-BREP cannot however be viewed in isolation since it has a significant impact upon the volume mesh generation. Within our implementation the piecewise linear surface elements generated by the standard SOLAR grid generation capability are instead replaced by quadratic surface elements[1]. In the case of piecewise linear boundary elements it is a relatively simple process to grow a highly stretched boundary layer mesh away from a curved boundary in the underlying geometry; however once a quadratic (or higher-order) BREP is introduced then issues of grid crossover can be encountered. In Section 4 we discuss how such issues were handled within SOLAR.

Finally, in Section 5 we discuss the mesh quality toolkit developed to allow analysis of the meshes generated using the approach described in Sections 3 and 4. It is expected that higher-order solvers will impose more stringent demands upon mesh quality than current second-order methods. This toolkit is intended to allow analysis of these issues, and to also allow the use of specific metrics during the mesh generation process itself, so ensuring the suitability of the mesh for higher-order simulations.

2 Background to SOLAR

Before describing how a higher-order mesh generation capability has been implemented in SOLAR, it is worth briefly discussing the general operation of SOLAR, since the work described here builds upon some of the standard volume mesh generation algorithms used by it.

2.1 SOLAR Mesh Generation

Over the past nine years, Airbus, BAE SYSTEMS, ARA and QinetiQ have collaborated on development of the rapid-response RANS CFD system, SOLAR [2, 3, 4, 5]. Included within this system is an automatic mesh generation capability designed to produce high quality meshes for viscous flow simulations on aerospace applications. The nearfield mesh is created using an advancing-layer technique [6] to march away from an unstructured quadrilateral-dominant surface mesh. Edge-collapsing and face-enrichment algorithms alter the topology of the layer automatically to take into account the underlying concavity or convexity of the region being meshed,

[1] Note that although in this work quadratic surface elements were used, there is in principle no reason why yet higher-order elements could not be used in conjunction with the approaches we describe in Sections 3 and 4.

with the layer growth factor being varied locally to achieve a smooth outer layer. In three dimensions, the nearfield layer mesh is then coupled to either a Cartesian far-field mesh which uses cut cells to avoid overlap of the near and farfield meshes [2], or alternatively to a tetrahedral farfield mesh, generated using a Delaunay method, via a buffer layer comprising pyramids and tetrahedra [3]. The latter of these approaches has the key benefit of avoiding the arbitrary polyhedra which are inevitably produced by a cut cell interface, and is the capability upon which our work in ADIGMA is based. However, it is the nearfield meshing algorithm, common to both approaches, which is of principal interest here.

2.1.1 Advancing Layer Mesh Generation

The basic methodology of the advancing-layer technique [6] is that the nearfield mesh is produced as a result of the recursive generation of a layer of cells and its subsequent placement on top of the previously generated layers (see Figure 1), starting from a predefined surface mesh. At the simplest level this process is analogous to the inflation of a balloon, with the surface topology being maintained in each layer, but in practice the aforementioned edge-collapsing and face-enrichment algorithms alter the topology of some layers in order to ensure a high quality mesh in concave and convex regions. These algorithms are outside the scope of the current discussion and will not be considered here (for details see [4] and [5]). The remaining two key issues in the generation of each layer concern the evaluation of marching direction and distance.

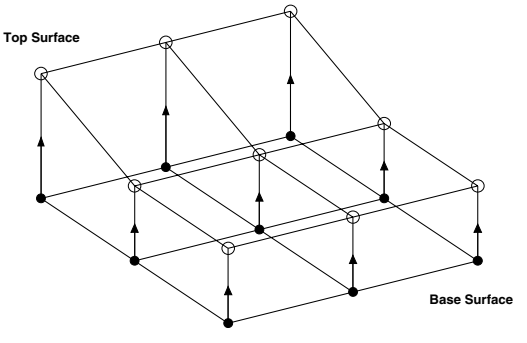


Fig. 1 Illustration of the advancing-layer approach to grid generation. The *base surface* is projected out into the flow domain to obtain the *top surface* of the layer. The top surface is then used as the base surface for generation of the subsequent layer.

Calculation of the marching direction is discussed in [4], but is essentially based upon the surface normal direction at each node, to ensure orthogonality of the mesh close to the surface, with a degree of smoothing applied to ensure (a) discontinuities in the direction of the marching vectors are removed; and (b) excessive mesh skewness is avoided in convex and concave regions. It is however the calculation of marching distance which is of most interest here since, as discussed later in Section 4, it is a extension to this algorithm which has enabled us to generate volume meshes in which the HO-BREP is accounted for.

For each surface node the marching distance is kept constant for the first n_0 layers and then increases at a local growth rate g according to the formula

$$h_n = h_0 g^{n-n_0} \quad (n > n_0), \tag{1}$$

where h_0 is the first cell height and h_n is the height at a node on the nth layer. The local growth rate is determined by a combination of the rate required to obtain cells of unit aspect ratio in the outer layer and a limitation on this rate in order to prevent intersection with another region of boundary layer mesh, such as can occur in regions of concavity or between separate components in close proximity to one another. The details of these algorithms are again outside the scope of the current article, but the approaches outlined have been found to perform very well for a range of complex problems within the aerospace sector. It is upon these algorithms that we have based the work described in the following sections.

3 Implementation of a Higher-Order Boundary Representation

In looking to implement a HO-BREP within the SOLAR code base, one of the main aims was to maximise, as far as reasonably possible, the reuse of existing code and algorithms. Having considered a variety of approaches to how this could be achieved, it was decided that the best overall approach was to restrict the code changes to the volume meshing routines and apply the following general algorithm to obtain the higher-order boundary representation:

- use the existing surface meshing functionality but tune the control parameters to generate a surface mesh of coarseness appropriate for higher-order solvers (coarser than would be used for 2nd order RANS solvers);
- use this surface mesh to provide *corner* nodes for the higher-order surface elements;
- use knowledge of the underlying CAD geometry plus the corner nodes to identify where to insert additional nodes on element edges and faces, so generating higher-order boundary elements. This was achieved by adding points at an appropriate position along the edges and faces and then projecting to the surface of the actual geometry;
- extract the HO-BREP from the volume mesh storage into an output file of appropriate format.

This process is illustrated schematically in Figure 2. Implementation of this approach required only relatively minor code changes, such as providing an interface to the underlying geometry via an existing API (previously only used in the surface meshing process), implementing a file writer for an appropriate mesh format, and adding code to insert the higher-order nodes at appropriate locations.

In generating outputs, the pragmatic decision was made to provide meshes in the format of the open-source tool Gmsh [1], since this was the only pre-existing format found that could support higher-order variants of all the volume element types SOLAR meshing can potentially generate (hexahedra, prisms, pyramids and

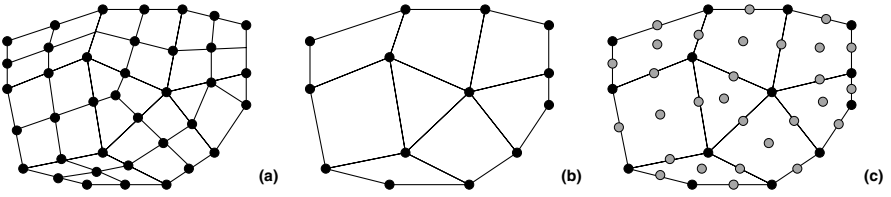


Fig. 2 Illustration of the process for generating a higher-order boundary representation in SOLAR. (a) illustrates the mesh that would be generated for use by a second order RANS solver. To obtain the higher-order boundary representation the coarser mesh in (b) is first generated using the existing surface meshing process. This mesh contains the corner nodes (in black) used to define the extent of the element. In (c) the higher-order nodes (in grey) are then added to provide the higher-order boundary discretization.

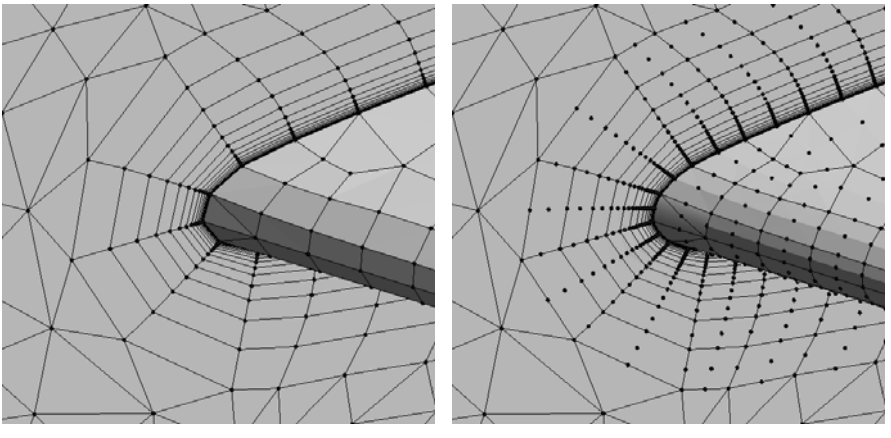


Fig. 3 Comparison of the surface meshes at the wing-symmetry plane junction of the ONERA-M6 wing obtained using regular and higher-order SOLAR meshing

tetrahedra). Utilising this format, meshes on a variety of standard test cases were generated. Results on the ONERA-M6 wing are illustrated in Figures 3 and 4, where meshes obtained using the above algorithm are compared to those obtained using regular SOLAR meshing. All user control parameters were identical, save for those used to switch between the standard and higher-order surface meshing capabilities. It can be seen from these figures that the above approach has successfully enabled the generation of a higher-order boundary representation on a relatively coarse surface mesh. In particular, the level of faceting observed in the highly curved regions of the geometry (the wing leading edge and tip regions) is vastly reduced in the higher-order mesh[2]. Similar results were obtained for significantly more complex

[2] Note that the clear lines between high-order nodes in these and subsequent figures are an artifact of the visualisation package representing higher-order faces by a set of regular surface elements (triangles and quadrilaterals).

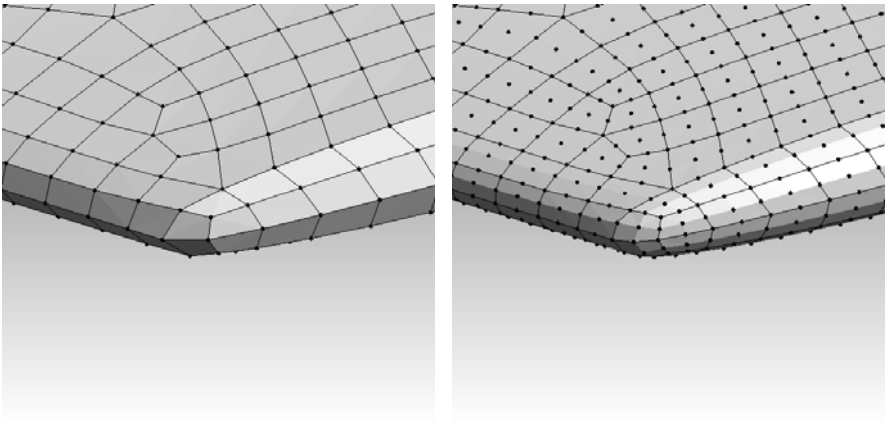


Fig. 4 Comparison of the surface meshes on the leading-edge/wing-tip region of the ONERA-M6 wing obtained using regular and higher-order SOLAR meshing

geometries (see Figure 12 for a further example). Considering these results, it is our opinion that the above algorithm provides a sound basis upon which to build a higher-order volume meshing capability.

4 Higher-Order Volume Meshing within SOLAR

Moving towards higher-order volume meshing based upon the advancing layer approach, the key issue is the need to prevent mesh crossover within the boundary layer mesh. The potential cause for this problem is illustrated in Figure 5. Given this possibility, is it clear that to be of practical use the layer meshing process must take account of the higher-order boundary. This is further illustrated by Figure 6 which shows the crossover that was obtained for the ONERA-M6 mesh when naively using the regular SOLAR layer meshing algorithms with the higher-order boundary representation from Section 3.

Again considering the pre-existing code algorithms, it was noted that the case illustrated in Figure 5(c) is essentially the same as would occur with the regular layer meshing, apart from the fact that there is no edge normal to the surface between the higher-order nodes. Taking this idea further we introduce the idea of *virtual* edges which are grown normal to the boundary from each of the higher-order nodes, in the same way as edges are grown from each of the *corner* nodes on a face. These virtual edges are not included in the output solver mesh, but are purely internal constructs used by the layer meshing algorithms. By using virtual edges we can essentially use the existing layer meshing algorithms to generate a higher-order layer mesh of the type illustrated in 5(c).

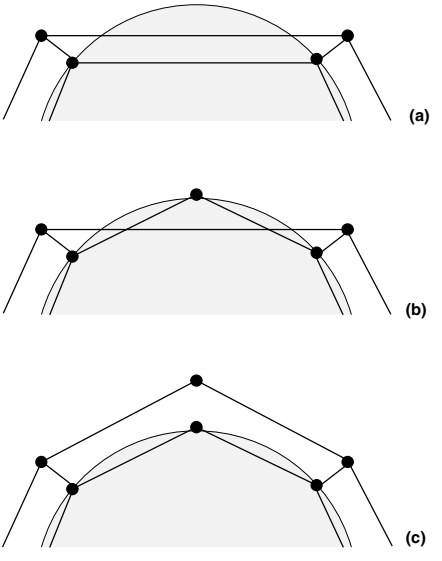


Fig. 5 Illustration of the consequences of not accounting for a higher-order boundary representation within the boundary layer mesh. (a) illustrates regular SOLAR meshing in which a piecewise linear boundary representation is used. In this case there is not an issue with mesh crossover, despite the poor resolution of the geometry. (b) illustrates the mesh crossover that can occur due to combining a higher-order boundary representation with piecewise linear volume elements. (c) illustrates the case when the higher-order boundary representation is accounted for in the layer mesh.

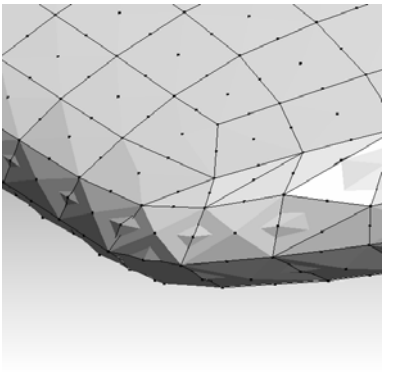


Fig. 6 Illustration of mesh crossover in the wing-tip region of the ONERA-M6 geometry if the higher-order boundary representation is not accounted for. The figure shows the surface of the first layer of the boundary layer mesh, through which the surface mesh can be seen to protrude.

At this stage it was decided, for pragmatic reasons, to limit the "true" higher-order elements to the nearfield region of the volume mesh[3]. By doing this, the overall higher-order volume meshing problem is reduced in scope, since algorithms for higher-order elements in the buffer and farfield regions do not have to be considered. We do however impose the artificial constraint that the outer surface of the layer mesh must have linear surface elements, to enable the buffer mesh to be generated using existing algorithms. We achieve this by varying the marching distance for virtual edges in an appropriate manner. Without applying such a modification, we obtain a higher-

[3] Linear volume elements can, of course, be treated as higher-order by inserting additional nodes on required edges and faces.

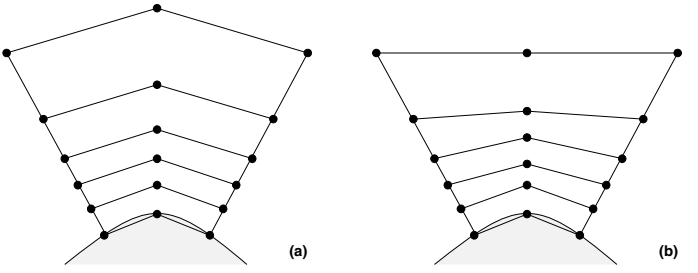


Fig. 7 Effect of the marching distance algorithm for higher-order nodes. (a) shows the layer mesh if we apply the same marching distance algorithm for all nodes. (b) illustrates the effect of applying a marching distance modification to the higher-order surface nodes (corresponding to the virtual edges).

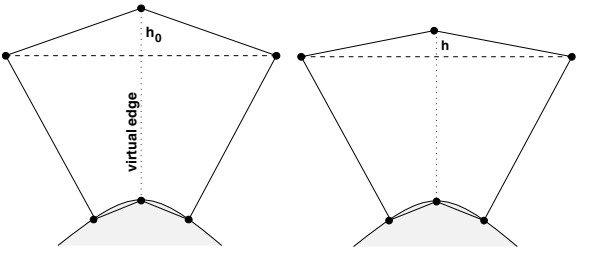


Fig. 8 Basic algorithm used in SOLAR to ensure a linear outer surface to the layer mesh. h_0 is first identified and then used to calculate h using Equations 2 and 3.

order layer mesh of the form illustrated in Figure 7(a). With appropriate changes to the marching distance we can instead generate a mesh as shown in Figure 7(b).

Within SOLAR, the marching distance modification was implemented following the algorithm illustrated in Figure 8. First the marching distance for a virtual edge is obtained using the standard meshing algorithms and used to provide an initial location for the higher-order node in the next layer. The corner nodes of the top surface are used to define a planar surface and the distance h_0 from this plane to the initial higher-order node position identified. The higher-order node is then placed along the vector described by the virtual edge, but at a distance h from where the virtual edge intersects the plane, where for layer n

$$h = f(n)h_0 \quad 0 \leq f(n) \leq 1. \tag{2}$$

Provided $f(N) = 0$, where N is the number of layers in the nearfield mesh, this ensures that linear faces are obtained on the outer surface of the layer mesh.

Through investigation it was identified that if $f(n)$ decreases too rapidly in the lower layers then issues with mesh crossover occur. In practice it was found that the performance of the algorithm was best if $f(n)$ remained at (or very near) zero until $n > N/2$, with best results being found when using an expression of the form

$$f(n) = \left\{ 1 + \exp\left(\frac{(n-p)}{\sigma} \right) \right\}^{-1}. \tag{3}$$

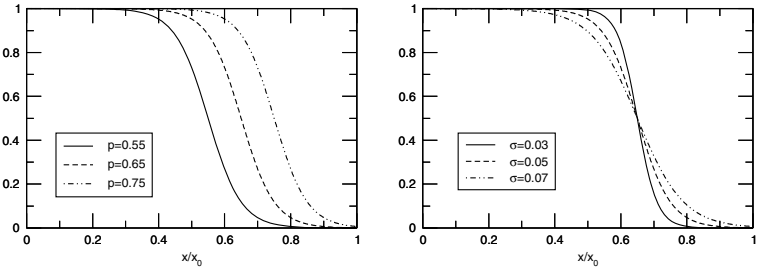


Fig. 9 Effects of varying the parameters p and σ in Equation 3

This equation essentially describes a smoothed version of the well known step function, in which $f(n)$ changes from 1 to 0 over a range determined by the value of σ and centred about p. The effects of varying σ and p are illustrated in Figure 9. For the implementation in SOLAR, values of $p = 0.85N_{layers}$ and $\sigma = 1$ were found to perform well.

Use of the above algorithms allowed generation of the meshes illustrated in Figures 10 and 11 for the ONERA-M6 geometry. In Figure 10 it can be seen that the chosen approach has successfully eliminated the issue of mesh crossover, the mesh protrusion, readily apparent in the left hand figure, disappearing when the algorithms described are used. Similarly, the outer surface of the nearfield mesh at various layers is illustrated in Figure 11. Again it can be seen that there is no protrusion of lower layers through the surface and it should be noted that the surface faces become more linear as the layer count increases, indicating that the higher-order node placement algorithm is working as intended.

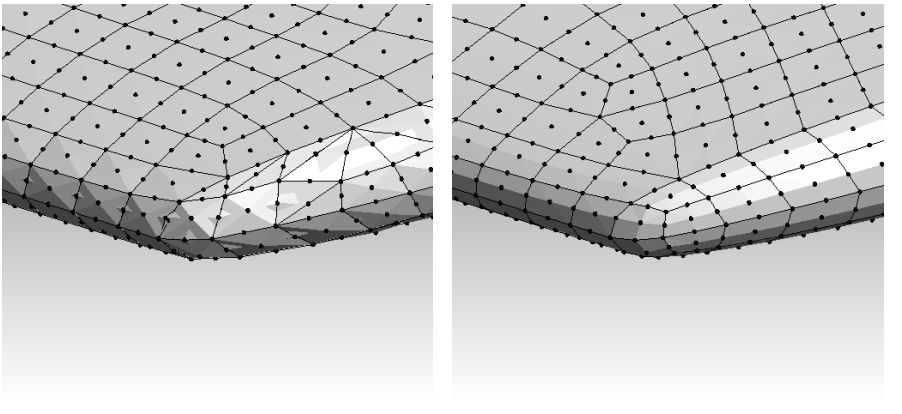


Fig. 10 Illustration of how the described algorithms have successfully resolved the issue of mesh crossover in the wing-tip region of the ONERA-M6 geometry. The figures show the surface of the first layer of the boundary mesh before and after the use of algorithms which allow for the higher-order boundary.

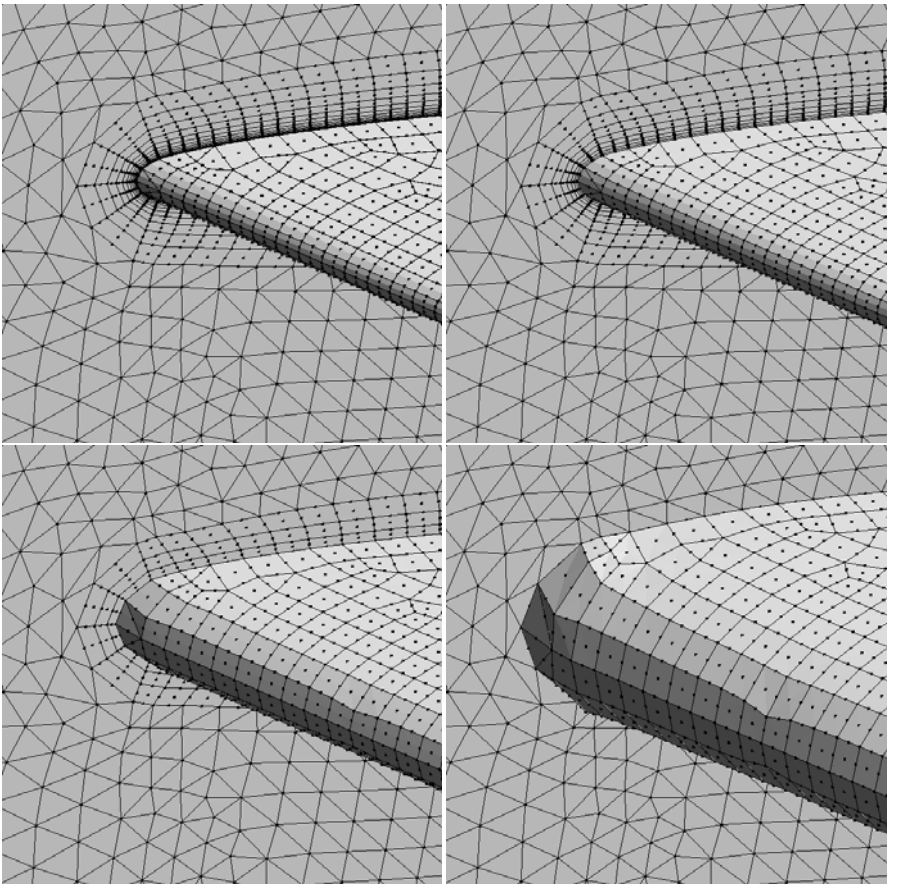


Fig. 11 Visualisation of various layers of the higher-order boundary layer mesh on the ONERA-M6 geometry. From top left: layer 15, layer 20, layer 25 and layer 28 (outermost layer). Note the transition from higher-order to linear faces as the layer increases.

Moving to a more complex geometry, meshes generated for the DLR-F6 test case are illustrated in Figures 12 and 13. As with the ONERA-M6 geometry it can be seen that a high-quality higher-order representation of the nearfield has been successfully obtained. Unlike the ONERA-M6 geometry however, this test case contains several of the geometric features that often prove problematic to mesh generators, such as the regions of high concavity in the the wing-body and wing-pylon junctions. Further work will be required to guarantee the successful generation of a higher-order grid on test cases containing features not considered here (for example the small gap regions between wing and slats/flaps in high-lift geometries), but the results presented here give us confidence that extension of the described approach will prove appropriate.

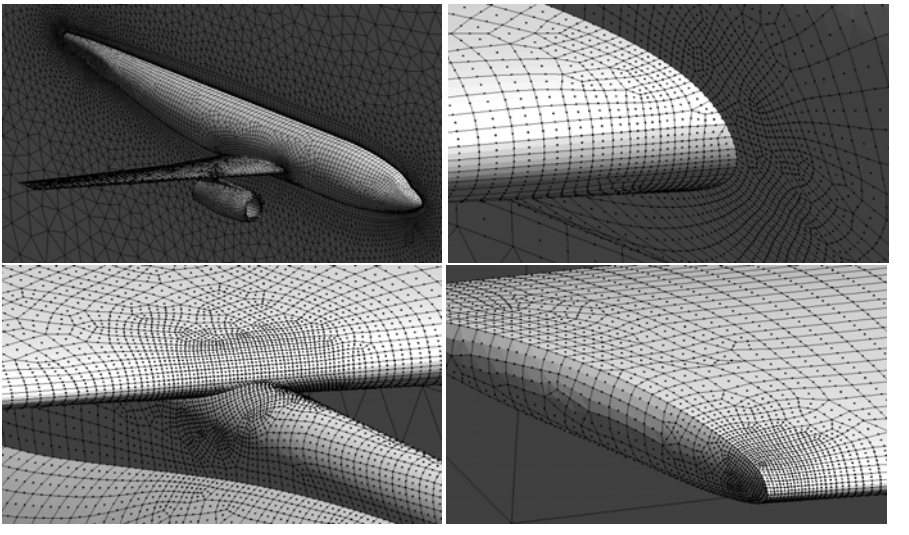


Fig. 12 Higher-order surface mesh on the DLR-F6 geometry. From top left: full surface mesh, wing-body junction region, wing-pylon junction and the wing tip.

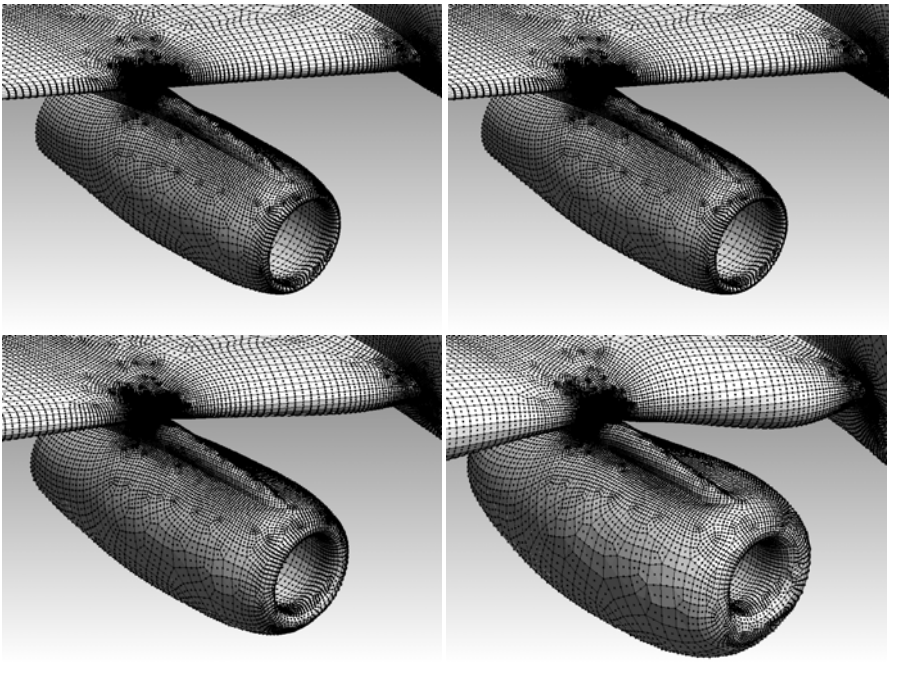


Fig. 13 Illustration of the higher-order nearfield mesh for the DLR-F6 geometry. From top left: the wing-pylon-nacelle region surface mesh, outer surfaces on layers 15 and 30, and the outer surface of the nearfield mesh.

5 Quality Toolkit for Higher-Order Meshes

The final part of our investigation within ADIGMA involved the development of a framework for analysing the quality of the generated meshes. The framework was implemented in a modular form, allowing relatively easy implementation of new metrics. As well as extracting information about the mesh, such as the number of cells of each type, this toolkit currently has metrics for evaluating:

- face area;
- face area ratio for mesh elements (maximum/minimum area for a given cell);
- edge curvature (perpendicular distance of the high-order node from vector joining end nodes/length of this vector);
- cell volume;
- ratio of cell diagonals for hexahedral elements (minimum diagonal/maximum diagonal).

The metrics for minimum face area and cell volume are self explanatory and are aimed at identifying potential causes of numerical error. The other metrics are instead aimed at identifying poor quality elements, such as those which are highly skewed.

Table 1 Mesh metrics for the higher-order ONERA-M6 and DLR-F6 meshes

Metric	ONERA-M6	DLR-F6
Minimum face area	6.293E-10	1.806E-10
Maximum face area ratio	25155	20362
Maximum edge curvature	0.203	0.203
Minimum cell volume	1.277E-8	2.423E-13
Maximum cell diagonal ratio	0.502	0.254

Lack of a suitably industrialised higher-order solver capable of running on meshes containing all the elements potentially generated by SOLAR has precluded a detailed study of how solution quality relates to that of the mesh, but the metrics have been used to ensure the validity of the meshes generated using the approaches detailed here. Example values for the ONERA-M6 and DLR-F6 meshes previously considered are given in Table 1. It is expected that this capability will be of significant utility in ongoing development of the higher-order meshing capability.

6 Conclusions

In the previous sections we have presented a novel approach for integrating a higher-order meshing capability into an existing advancing-layer based mesh generator and demonstrated that this technique can successfully generate such meshes on relatively complex geometries of industrial interest. However there still remain a number of ways in which this capability could be developed further.

Concerning the higher-order boundary representation discussed in Section 3, at present a relatively naive algorithm is used when inserting the higher-order nodes, in that these are simply placed at the midpoint between corner nodes and then projected onto the geometry. This approach can easily lead to significant discontinuities in the surface curvature between neighbouring faces and a poor approximation of the underlying geometry. Work should be undertaken to investigate how the placement of higher-order nodes can be optimised to minimise the error in the discretisation of the geometry.

Moving to the volume meshing, one potential issue lies with the lack of higher-order elements in the buffer and farfield regions. It would be a relatively trivial task to modify the existing capability so that elements outside the nearfield are transformed into higher-order equivalents. At a more practical level, the developments discussed here were implemented with little emphasis on optimising the relevant data structures and supporting algorithms within the code. As such, the capability is somewhat slower and more resource intensive than would be ideal. Again, improvements should be investigated, and implemented where appropriate.

Finally, in order to maximise the potential gains offered by higher-order solvers it is essential that a method of mesh adaptation is implemented for higher-order volume meshes, so that new nodes can be accurately inserted onto the surface of the geometry. Once an industrial higher-order mesh generation capability is established, the ability to adapt these meshes will assume a far higher level of importance, and is a requirement that will provide the focus for future work in this area.

Acknowledgements. The authors would like to thank our colleagues in the Computational Aerodynamics department at ARA, Prof. Paul Houston of Nottingham University and Dr. Ralf Hartmann of DLR for numerous useful discussions throughout the duration of this work.

References

1. Geuzaine, C., Remacle, J.-F.: Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. Int. J. Numer. Meth. Eng. 79, 1309–1331 (2009)
2. Leatham, M., Stokes, S., Shaw, J.A., Cooper, J., Appa, J., Blaylock, T.A.: Automatic Mesh Generation for Rapid-Response Navier-Stokes Calculations. AIAA Paper 2000-2247 (June 2000)
3. Martineau, D.G., Stokes, S., Munday, S.J., Jackson, A.P., Gribben, B.J., Verhoeven, N.A.: Anisotropic Hybrid Mesh Generation for Industrial RANS Applications. AIAA Paper 2006-0534 (January 2006)
4. Munday, S.J., Martineau, D.G., Verhoeven, N.A.: Automated Hybrid Mesh Generation for Industrial High-Lift Applications. AIAA Paper 2007-268 (January 2007)
5. Shaw, J.A., Stokes, S., Lucking, M.A.: The rapid and robust generation of efficient hybrid grids for rans simulations over complete aircraft. Int. J. Numer. Meth. Fluids 43, 785–821 (2003), doi:10.1002/fld.497
6. Pirzadeh, S.: Unstructured Viscous Grid Generation by Advancing Layers Method. AIAA Paper 93-3453-CP (1993)

This page intentionally left blank

Chapter 14
Synthesis Report on Shock Capturing Strategies

Arne Taube and Claus-Dieter Munz

Abstract. One key problem in higher-order methods is the preservation of monotonicity across discontinuities such as shock waves. This synthesis report gives an overview about the different approaches adopted by the ADIGMA partners, INRIA, NJU, SERAM, UNBG, UNPR, UNST and VKI, who are all involved in the shock capturing workpackage. It shows the current state-of-the-art of the shock capturing capabilities. After a brief description of the different methods, these are compared based on the results for two two-dimensional test cases, one unsteady strong $Ma = 10$ shock (DMR) and one steady solution of a transonic flow around a NACA0012 profile (MTC 2). The focus is put on a thin shock profile and the least dissipative representation of the small scale instabilities for the DMR problem and the accurate computation of the aerodynamic coefficients with the least entropy error in the smooth flow region for the MTC 2 case.

1 Introduction

ADIGMA deals with the development of high-order methods for the Euler and Navier-Stokes equations. These methods bear the potential to solve future CFD problems for aerospace applications more efficiently, by making use of three different advantages: the high order property, the fact that they can cope with unstructured meshes, hence complex geometries, and adaptivity in polynomial order as well as mesh refinement *(p-, h- and hp-adaption)*.

In order to preserve monotonicity across discontinuities such as shock waves different approaches are adopted by the ADIGMA partners. This synthesis report gives an overview and shows the current state-of-the-art of their shock capturing capabilities.

Arne Taube · Claus-Dieter Munz
Institut für Aerodynamik und Gasdynamik, University of Stuttgart, Pfaffenwaldring 21, 70569 Stuttgart, Germany
e-mail: taube@iag.uni-stuttgart.de, munz@iag.uni-stuttgart.de

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 195–207, 2010.
springerlink.com

First in Sect. 2, the test cases within the ADIGMA project to validate and to compare the different approaches are presented. Two simple transient test problems are chosen: a two-dimensional hypersonic problem containing strong shocks, and the transonic NAC0012 test case from the MTC catalogue. With the help of these proposed problems each partner demonstrates the shock-capturing property as well as the high-order resolution possibility. After the description of the test problems, a short overview about the different approaches that are used by the different partners for shock-capturing is given in Sect. 3.

The summary in Sect. 4 is founded on each parter's input as given in tabular form in the appendix and endeavors to give an evaluation with respect to the shock capturing capabilities developed within the ADIGMA project.

2 Numerical Test Cases for Shock Capturing Property

Apart from some one-dimensional test cases computed previously in the project, the comparison of the results with the other ADIGMA partners is limited to two two-dimensional test cases.

2.1 Two-Dimensional Problems

One of them is the double Mach reflection of a strong shock (DMR) and the other is the transonic flow around the NACA0012 airfoil (MTC 2). These test cases are well known and we have already gathered experience with them in the ADIGMA project.

2.1.1 Double Mach Reflection of a Strong Shock (DMR)

This test problem is presented by Woodward and Colella in [14]. The setup in their paper is as follows:

A Mach 10 shock in air which initially makes a 60° angle with a reflecting wall. The undisturbed air ahead of the shock has a density of $\rho = 1.4$ and a pressure of $p = 1$. The reflecting wall lies along the bottom of the problem domain, beginning at $x = 1/6$. The shock makes a 60° angle with the x-axis and extends to the top of the problem domain. The short region from $x = 0$ to $x = 1/6$ along the bottom boundary at $y = 0$ is always assigned values for the initial post-shock flow. This boundary condition forces the reflected shock to be 'attached' to the reflecting wall. The left-hand boundary is also assigned values for the initial post-shock flow, and at the right-hand boundary all gradients are set to zero. The values along the top boundary are set to describe the exact motion of the initial Mach 10 shock. This setup was chosen due to the fact that most numerical codes at that time were not able to calculate boundaries of complex geometry, thus the setup was kept as simple as possible. In order to facilitate the result comparison, this setup has been adopted and rectangular meshes with $\Delta x = \Delta y = \frac{1}{120}$ or derivatives of that are chosen. The

total simulation time is chosen appropriate to the original setup by Woodward and Colella [14] to $t_{end} = 0.2$.

2.1.2 Transonic Flow around the NACA0012 Airfoil (MTC 2)

In the following, we compute the flow around the symmetric NACA0012 airfoil as described in the ADIGMA list of test cases with an inflow Mach number of $M_\infty = 0.8$ and an angle of attack $\alpha = 1.25°$. The flow becomes locally supersonic on both sides of the profile leading to a relatively strong shock at the suction side and a very weak shock on the pressure side. Here, we use the unstructured meshes from the ARA mesh catalogue.

3 Main Approaches to Shock Capturing

In order to achieve the shock capturing property for higher order schemes, one has to take some action to avoid Gibbs-type oscillations. There are three main approaches: reconstruction with limiting, the residual distribution approach and artificial viscosity. Some remarks according to all methods are given in the following. In any case, the first step is to detect the grid cells in which a strong gradient is situated and which will produce spurious oscillations of the approximative piecewise polynomial. Only in this case a correction is applied. The detection algorithm may be considered as a separate step, but is often closely related to the form of the correction. An overview of existing detection strategies for DG methods is given in [12].

3.1 Reconstruction and Limiting

So called limiting procedures have been proposed within the finite volume framework. While the first order accurate flux calculations in FV schemes give a monotone behavior, the higher order versions generate spurious oscillations, if values from both sides, e.g. of a shock wave are used within the approximation. To avoid this, one has to look to both sides and to take the values from the side where the approximation is smoother. This has first been formulated in terms of the numerical flux values and the nomination flux limiter has been introduced (see, e.g. [13]). Later, the approach of reconstructing the arguments of the flux function has become widely accepted rather than the improvement of the flux values. The reconstruction is controlled in such a way that the stencil for the reconstructed polynomial is chosen into the direction in which the data are smoother. Here, the established approach is the WENO reconstruction (Weighted Essentially Non-Oscillatory). A comprehensive overview may be found in [6, 7].

The simplest version for a DG scheme to establish shock-capturing would be to locally reduce the discretization order to a monotone first order scheme. In this case, the DG scheme coincides with the first order finite volume scheme guaranteeing monotone shock resolution. In order to have a good shock resolution, it has to be combined with local grid refinement. Here, the question remains: Is the accuracy

preserved, if a smooth wave interacts with a discontinuity or if a narrow viscous profile such as a boundary layer has to be captured? In any case, such an first order approach should only be applied to the troubled cells. Hence, the first step is to detect the troubled cells. These cells are then refined and the order of their approximation polynomial is reduced. However, it might be, that a large number of low order cells are created to yield a satisfactory accuracy. This would lead to high computational costs.

An improvement of this FV based strategy is to use a reconstructed polynomial to limit the slopes or to replace the DG polynomial in the troubled zones. The reconstruction methods allow for stable discretizations near discontinuities while still maintaining a high order approximation. The problem in that case is that the designed high order accuracy for FV schemes has often not been observed in practical situations, especially on unstructured grids. Very recently, a more robust modification of the WENO reconstruction technique is presented in [5] which seems to perform better on unstructured grids. Nevertheless, there is another drawback: the big advantage of strong locality is lost. To keep the stencil for the reconstruction small the so-called HWENO approach seems to be more attractive. The reconstruction is based on Hermite interpolation combined with the WENO idea. It has been presented in [11] and found application in [8]. Here, the values of both the function and its first derivatives are used within the reconstruction. Thus, only the von Neumann neighborhood is required for a third order reconstruction.

3.2 Residual Distribution Approach

In standard approaches, such as the MUSCL or ENO/WENO approaches, one tries to control the Gibbs phenomena by tuning the amount of artificial dissipation so that in the end the solution is oscillation free without destroying the formal accuracy of the scheme as it can be obtained by truncation error analysis. In the residual distribution approach, the approach is different and does not use the structure of an equivalent equation, it uses a comparison principle and is the result of a series of simple remarks.

In a finite volume approach, for any control volume, the solution is updated by the way the sum of fluxes changes around it. The natural unknown is the average of the unknown quantities in these control volumes. In the RD approach, as for the finite element method, the natural quantity is the approximation of the unknowns at vertices and possibly other points in the mesh. Their values are updated by the way the sum of the fractions of residual of the elements surrounding these points evolves. Conservation is ensured if the scheme is stable and if, for any element, the sum of the fractions of residual evaluated for that element sums up to the normal flux over the boundary. This quantity is called the total residual.

There are many ways to construct these sub–residuals. Historically, one way is to consider the N scheme. Another (simpler) solution is a generalization of the Rusanov flux, re–interpreted in the RD framework. One can show that these schemes are L^2 and L^∞ bounded under a CFL–like condition, hence are oscillation free.

The second remark is that accuracy is obtained provided that for each element, the total residual scales correctly, and second that the sub–residual has the same scaling. The proof requires the problem to be steady. Unsteady problems can be rephrased into steady ones and the same line of arguments applies, see [1, 4, 3].

The third remark is that one can construct a fully automatic way of constructing a L^∞ stable and formally high order accurate scheme. This method is obtained by comparing the first-order sub-residual with the total residual, see [4]. Unfortunately, this scheme is over compressive, and an entropy dissipation mechanism needs to be added, see [2].

3.3 Artificial Viscosity

The third approach consists of locally adding some sort of dissipation, by which the shock wave is rendered into a steep viscous profile which can then be resolved by the numerical approximation. The basic principle dates back to von Neumann and Richtmyer [9] from the early '50s and gave way to resolve shock profiles with finite difference schemes. The drawback of this approach is that the artificial viscosity has to be tuned by a problem dependent parameter. Hence, the second order accurate Godunov-type schemes were celebrated more than 20 years ago as an approach to parameter-free shock-capturing. Nevertheless, there are schemes for practical problems in the transonic regime with a sophisticated choice of the coefficients which are quite successful.

For high order DG schemes, the artificial viscosity idea becomes attractive again. Here, we may have large grid cells and higher order polynomials. This in combination with artificial viscosity may give rise to a sub-cell resolution of strong gradients, i.e. shocks. Persson and Peraire [10] state that a steep gradient can be resolved by a piecewise polynomial approximation with degree p on a width $\delta_S \sim \frac{h}{p}$. Thus, if p is high, the shock can even be resolved at sub-cell level. Hence, the artificial viscosity needed becomes smaller with increasing polynomial degree. The hope is that with such a strategy of locally increasing the order at steep gradients and adding some small amount of artificial viscosity, one can still use coarse grid cells near shocks. Persson and Peraire apply this shock-capturing by p-refinement within an implicit scheme. They describe two strategies, called Laplacian and physical artificial viscosity [10]. Both approaches lead to local jumps in the viscosity distribution with some oscillating time behavior. This approach can also be combined with an explicit scheme, but the parabolic time step restriction may be more severe near shocks due to the local strong artificial viscosity term. Other open questions are the form of the artificial viscosity and keeping the method parameter-free.

4 Summary

The partner's computations of the two-dimensional test cases provide the basis for our evaluation and give a clear view on the current state-of-the-art of the numerical shock capturing methods within the project.

4.1 General Assessment

As stated before, different techniques are used for shock capturing and hence any comparison is difficult, especially, since for the issue of shock capturing exact reference solutions are rare. Even if they are available shocks are singularities and therefore it is not possible to assess a numerical scheme's quality for example by its rate of convergence.

4.1.1 DMR Results

With regard to the first test case, it is commonly agreed on in the ADIGMA project, that in a first step it is necessary to successfully compute the DMR's $Ma = 10$ shock in a stable manner. A very good result, however, consists not only of a thin shock profile, but also of the well resolved small scale structures and wiggles which are smooth phenomena behind the shock in the contact region. The shape and exact location of these instabilities, however, is not determined and therefore only a visual judgement can be applied. The numbers provided by the partners for the DMR results are dressed in Fig. 1.

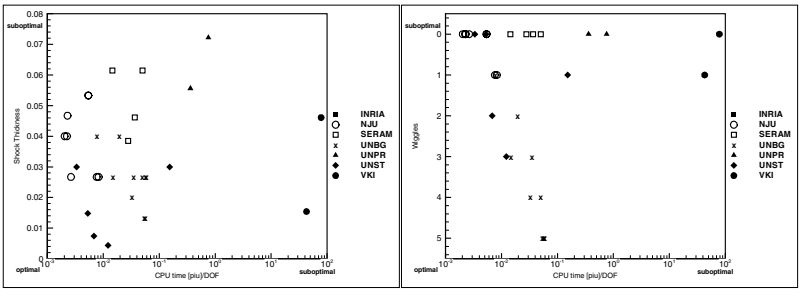


Fig. 1 Measured shock thickness (left) and 'wiggles' (right) versus DOF efficiency

Figure 2 shows some of the results obtained by the ADIGMA partners for this particular test case. A direct comparison of these figures, however, cannot be made immediately, because the meshes and orders are very different. Nonetheless, the plots give a flavor of the solution quality.

The shock thickness measured from the partners' provided density contour plots is put in relation to the CPU time in performance index units [piu] normalized per degree of freedom (DOF). An optimal computation requires the least computational effort per DOF and, in return, generates a very slim shock profile.

There are quite a few results with a rather thin shock and moderate CPU times, but this is not the whole story. For a truly genuine DMR solution, the unstable contact discontinuity roll-up behind the triple-point is required, as well. Therefore, it is tried to find an assessment for that in Fig. 1 right graph. The chosen strong criterion may be questionable, but if applied there are only few really efficient high order results left for this test case.

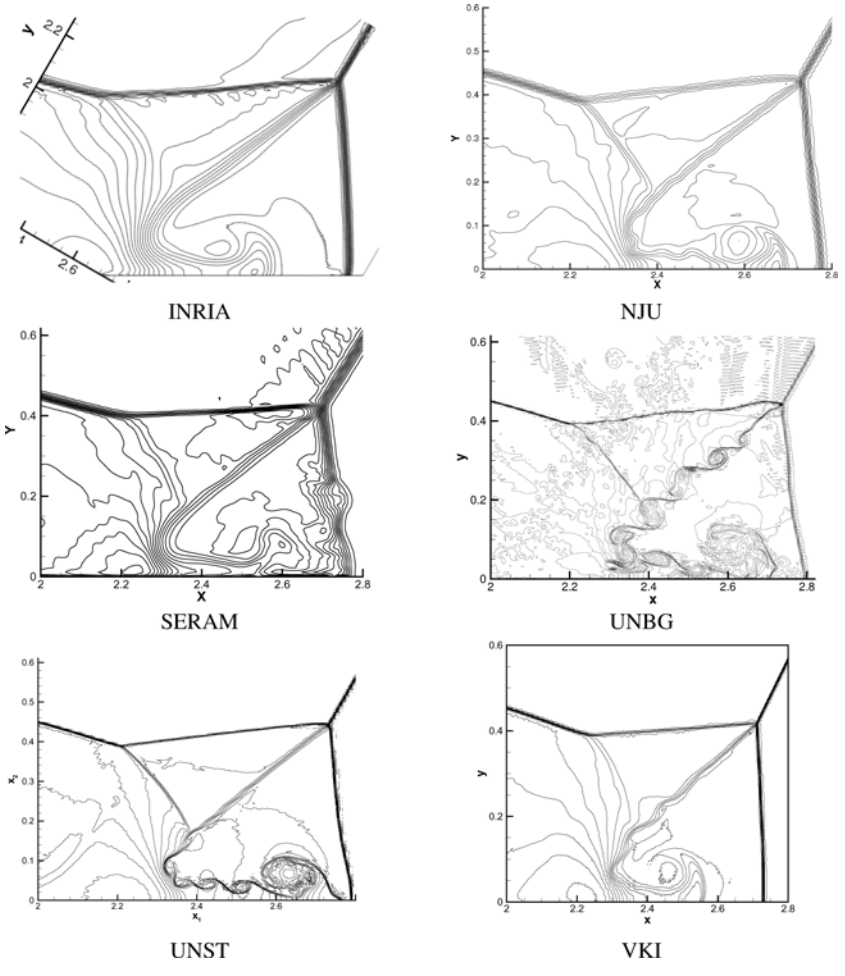


Fig. 2 Selected DMR results by the ADIGMA partners on different meshes

All in all, the most efficient and thus optimal shock capturing results for this particular test case are the ones in the bottom left corner of all plots. These results show a clear and slim shock profile as well as a good resolution of the fine structures in the rolled-up contact discontinuity.

4.1.2 MTC 2 Results

The MTC 2 test case bears a steady solution. There, the strong shock on the profile's suction and the weak one on the pressure side shall be captured and resolved well. In addition, there shall not be an entropy error in the nose region, but only behind the

shock. Besides that, this test case consists of an aerodynamic profile and therefore the aerodynamic coefficients shall match the reference data.

Figures 3 and 4 give a notion of each partner's capabilities for this steady test case. Figures 5 and 6 show some of the results obtained by the ADIGMA partners for this particular test case. A direct comparison of the plots, however, cannot be made immediately, because the meshes and orders are very different. Nonetheless, the plots give a taste of the solution quality.

For the coefficient diagrams, the best solutions are located around the line marked as TAU reference solution and most to the left, i.e. consuming the least CPU time. The estimation of the entropy error is again based on visual judgement. In that case, the best results are to be found in the bottom left corners of the plots at Fig. 4 right.

Clearly, the contributions focus on the shock capturing property. That may be the reason, why some of the obtained aerodynamic coefficients violate the ADIGMA convergence criterion. For the c_m value this might be a matter of sign rule, but some are nowhere near to the exact solution. Some of the entropy production in the nose area, may however not stem from the shock capturing or limiter but is due to the coarse approximation of the NACA0012 nose curvature by the unstructured ADIGMA meshes.

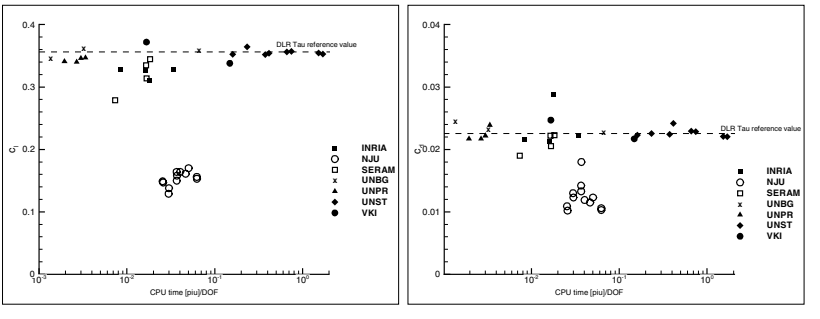


Fig. 3 Lift c_l (left) and drag c_d (right) coefficients versus DOF efficiency

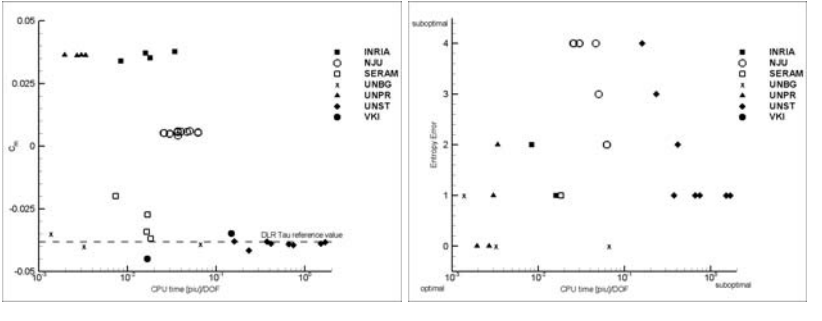


Fig. 4 Moment coefficient c_m (left) and entropy error in the profile's nose area (right) versus DOF efficiency

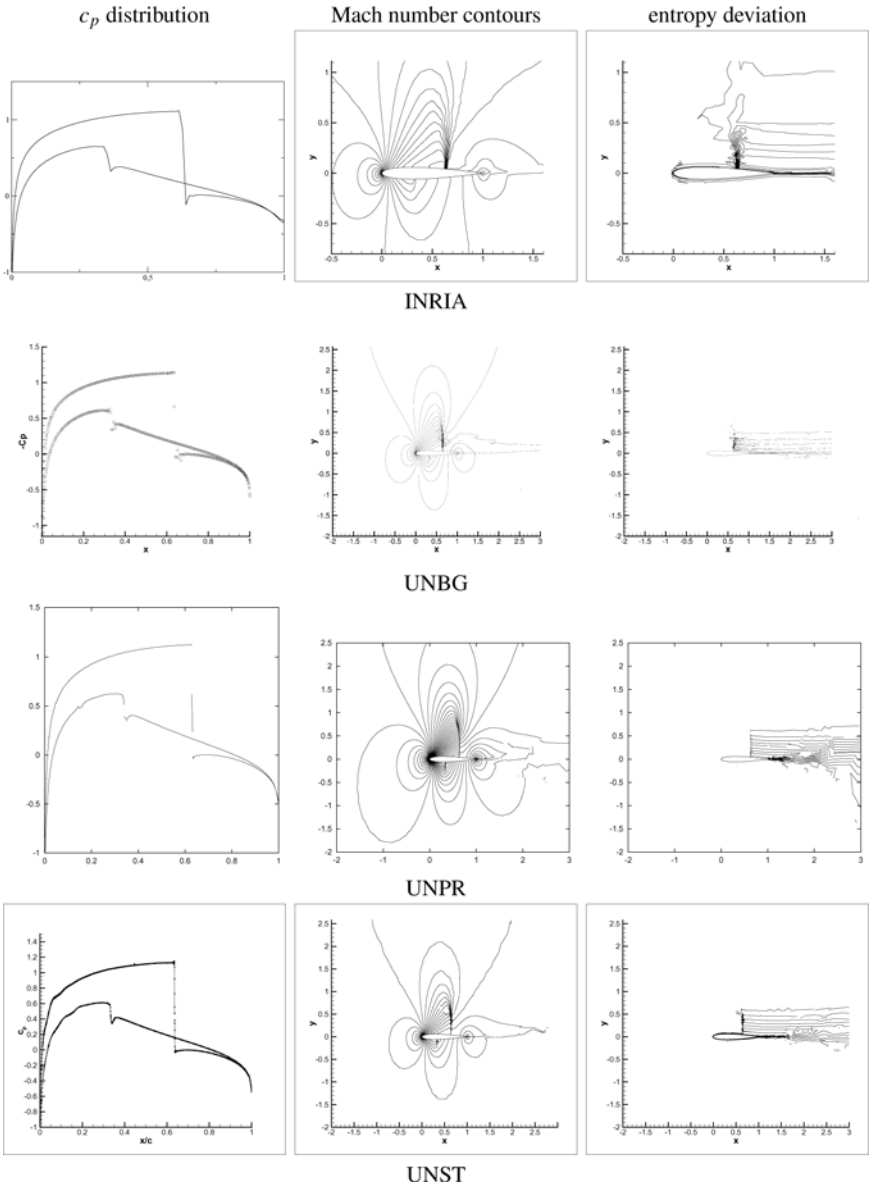


Fig. 5 Selected MTC 2 results by the ADIGMA partners on different meshes

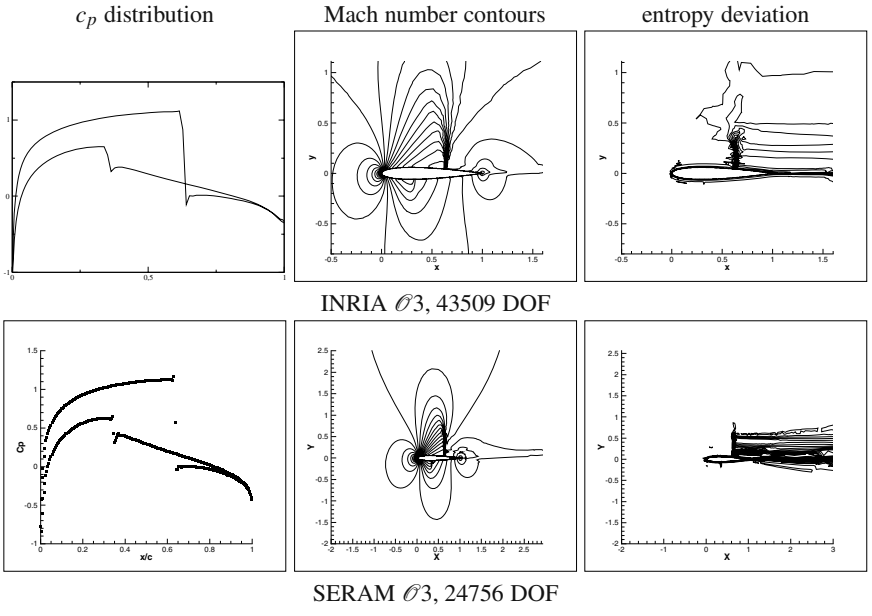


Fig. 6 Selected MTC 2 results by the ADIGMA partners on different meshes (continued)

4.2 Each Partner's Assessment

INRIA

INIRA's limitation technique is easy to handle without any experienced overshoots. Its extension to any order or accuracy on arbitrary meshes is straight forward. The technique can also be applied to DG problems because of a reformulation of DG-schemes in terms of RD schemes. However, the implementation of wall boundary conditions is not perfect and the best results are shown on the steady results, although the boundary conditions are not fully satisfactory. Future work is going to focus on a better implementation for the unsteady schemes.

NJU

NJU has developed the limiters for the RKDG methods solving hyperbolic conservation laws using finite volume high order WENO and HWENO reconstructions on unstructured meshes. The idea is to first identify troubled cells subject to the WENO or HWENO limiting, using a TVB *minmod*-type limiter, then reconstruct the polynomial solution inside the troubled cells by the WENO or HWENO reconstruction using the cell averages of neighboring cells or the cell averages and cell derivative averages of neighboring cells, while maintaining the original cell averages of the troubled cells. Numerical results show that the method is stable, accurate, and robust in maintaining accuracy.

SERAM

SERAM uses the Venkatakrishnan modified Barth limiter for the 2^{nd} order FV-RB scheme and the binary limiter for the 3^{rd} order scheme. The computation of an unsteady problem, DMR test case, shows that the combination of a temporal limiter with the spatial limiters can remove most of the oscillations. However, for the same number of degrees of freedom, the 3^{rd} order RB scheme produces almost the same quality of result as the 2^{nd} order scheme. The RB scheme with the present time residual limiter can compute a challenging test case such as the DMR, but another type of limiter has to be designed for an economic use of the method on such test-problems. The computational results obtained for the steady MTC 2 test case are satisfactory.

UNBG

The shock-capturing approach for DG methods developed by UNBG aims at achieving an optimal balance among several contrasting objectives, such as accuracy, robustness and no need of case-specific fine tuning of parameters. Moreover, its applicability to steady and unsteady flows has always been considered highly desirable. The results presented show that UNBG's approach is capable of resolving with excellent accuracy both shocks and contact discontinuities, while the topic of the numerical shock structure inside elements needs further investigation.

UNPR

UNPR's presented shock capturing technique (SCT) may produce some small overshoots and overshoots in the vicinity of discontinuities and the shock wave is spread into two elements, hence for coarser grids the shock waves are smeared. This drawback can be reduced with the use of *h*-mesh adaptation. Furthermore, their technique is robust, without having to change empiric constants for different test cases and is able to significantly reduces the non-physical oscillations.

UNST

UNST's approach depends on the exact shock detection and a good estimate for the artificial viscosity. Given that, we can capture a shock with increasing order within less grid cells until its full resolution within only one cell. However, once a shock is detected, the right amount of viscosity added is crucial in order to get a very sharp profile on one hand and not to spoil the solution on the other hand. Up to now, no truly parameter free implementation of this approach is known. The strategy works well for both test cases, whereas the numerical scheme simulates time accurately and thus is optimized for non-stationary problems.

VKI

The results obtained with Bx are quite satisfactory. With this scheme we have achieved the combination of high order of accuracy in the smooth parts with a non-oscillatory capture of the shock, keeping a good resolution of it. The main drawback of the Bx scheme is that it needs a good tuning of the parameters to capture well the shock. In particular, for the MTC 2 test case it is not easy to capture the shock of the lower surface without adding some dissipation on the leading edge where the pressure gradients are important. Anyway, the convergence of the scheme is quite good and both shocks are well captured and do not show big spurious oscillations.

Concerning the unsteady shock capturing, we used the limiting strategy. This method has the advantage of being parameter free. The results obtained are good in terms of monotonicity, but the code still needs to be optimized concerning the computational speed.

4.3 Outlook

There is no uniform solution to the problem of high order shock capturing. The test cases presented in this report are very different, i.e. a non-stationary strong shock and stationary weaker shock. Therefore, the different numerical methods and their approaches to shock capturing may perform well in one case and not so well in the other one. The assessment is solely based on the numerical values for the measured shock thickness in the DMR and on the aerodynamic coefficients in the MTC 2 case. These are put in relation to the CPU time. The rating for the DMR's contact resolution and the MTC's entropy error around the nose is rather soft, because it is only a visual judgement.

Besides that, the methods are constantly under development and being improved. Therefore, even not so successful candidates from the previously made analysis will improve or have their strengths elsewhere. For example factors like the time and effort taken for manually adjusting control parameters to obtain a stable and yet highly accurate solution are not taken into account in the CPU times plots. Since, the numerical methods differ, too, some may only play off their potentials when using additional features like mesh adaptation or a finer mesh.

Acknowledgements. We gratefully acknowledge funding of this work by the target research project ADIGMA within the 6^{th} European Research Framework Programme.

References

1. Abgrall, R.: Toward the ultimate conservative scheme: Following the quest. J. Comput. Phys. 167(2), 277–315 (2001)
2. Abgrall, R.: Essentially non-oscillatory residual distribution schemes for hyperbolic problems. J. Comput. Phys. 214(2), 773–808 (2006)

3. Abgrall, R., Andrianov, N., Mezine, M.: Towards very high-order accurate schemes for unsteady convection problems on unstructured meshes. Int. J. Numer. Methods Fluids 47(8-9), 679–691 (2005)
4. Abgrall, R., Roe, P.L.: High-order fluctuation schemes on triangular meshes. J. Comput. Phys. 19(1-3), 3–36 (2003)
5. Dumbser, M., Käser, M.: Arbitrary high order non-oscillatory finite volume schemes on unstructured meshes for linear hyperbolic systems. J. Comput. Phys. 221(2), 693–723 (2007)
6. Hu, C., Shu, C.-W.: Weighted essentially non-oscillatory schemes on triangular meshes. J. Comput. Phys. 150(1), 97–127 (1999)
7. Jiang, G.-S., Shu, C.-W.: Efficient implementation of weighted ENO schemes. J. Comput. Phys. 126(1), 202–228 (1996)
8. Luo, H., Baum, J.D., Lohner, R.: A Hermite WENO-based limiter for discontinuous Galerkin method on unstructured grids. J. Comput. Phys. 225(1), 686–713 (2007)
9. von Neumann, J., Richtmyer, R.D.: A method for the numerical calculation of hydrodynamic shocks. Journal of Applied Physics 21(3), 232–237 (1950)
10. Persson, P.-O., Peraire, J.: Sub-cell shock capturing for discontinuous Galerkin methods. In: Proc. of the 44th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2006-1253, Reno, Nevada (January 2006)
11. Qiu, J., Shu, C.-W.: Hermite WENO schemes and their application as limiters for Runge–Kutta discontinuous Galerkin method: One-dimensional case. J. Comput. Phys. 193, 115–135 (2004)
12. Qiu, J., Shu, C.-W.: A comparison of troubled-cell indicators for Runge–Kutta discontinuous Galerkin methods using weighted essentially nonoscillatory limiters. SIAM J. Sci. Comput. 27(3), 995–1013 (2005)
13. Sweby, P.K.: High resolutions schemes using flux limiters for hyperbolic conservation laws. SIAM J. Num. Anal. 21, 995–1011 (1984)
14. Woodward, P., Colella, P.: The numerical simulation of two–dimensional fluid flow with strong shocks. J. Comput. Phys. 54, 115–173 (1984)

This page intentionally left blank

Chapter 15
Implicit Strategy and Parallelization of a High Order Residual Distribution Scheme

R. Abgrall, R. Butel, P. Jacq, C. Lachat, X. Lacoste, A. Larat, and M. Ricchiuto

Abstract. This chapter describes the implicitation and parallelization strategies we have developed, as well as some development and results using high performance linear algebra softwares.

1 Introduction

The purpose of this chapter is to describe what has been the strategy we have used for the implicitation and the parallelization of the high order residual distribution scheme described in chapter 9. We first recall some elements on the numerical method described in details in chapter 9. Considering a steady problem with boundary conditions

$$\text{div } F(U) = 0 \tag{1}$$

R. Abgrall
INRIA and Institut de Mathématiques de Bordeaux, Team Bacchus, bat A29 bis,
341 Cours de la Libération, 33 405 Talence Cedex, France
e-mail: Remi.Abgrall@inria.fr

R. Butel
Institut de Mathématiques de Bordeaux, 341 Cours de la Libération,
33 405 Talence Cedex, France
e-mail: Remi.Butel@math.u-bordeaux1.fr

A. Larat · P. Jacq · X. Lacoste · M. Ricchiuto
INRIA, Team Bacchus, bat A29 bis, 341 Cours de la Libération,
33 405 Talence Cedex, France
e-mail: Adam.Larat@inria.fr, Pascal.Jacq@inria.fr,
Xavier.Lacoste@inria.fr, Mario.Ricchiuto@inria.fr

C. Lachat
INRIA, Team Pumas, 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex, France
e-mail: Cedric.Lachat@inria.fr

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 209–224, 2010.
springerlink.com

on the domain Ω, where the flux term is the sum of the Euler flux and the diffusive flux (if needed) and a partition of Ω made of triangle (in 2D) and tetrahedrons (in 3D), we have developed so far two types of residual distribution schemes. The second order version uses a continuous P_1 interpolation in each element, and a third order version which uses a continuous P_2 interpolation. In the first case, the degrees of freedom are the vertices of the mesh. In the second case (P_2), the degrees of freedom are the vertices of the mesh and the mid point of the edges. We denote by σ a generic degree of freedom, and by T a generic element of the tessellation of Ω.

In each case, the scheme writes, for any dof σ

$$\sum_{T,\sigma\in T} \Phi_\sigma^T = 0 \tag{2}$$

where the residuals Φ_σ^T satisfies

- In the case of the Euler equation, we have the conservation relation

$$\sum_{\sigma\in T} \Phi_\sigma^T = \int_{\partial T} F\cdot\mathbf{n}(U^h)dl \tag{3}$$

  where $\mathbf{n}$ is the outward unit normal to ∂T, U^h is the interpolant of U.
- In the case of the Navier Stokes equation, $F = F^e + F^v$, the viscous flux are discretised by a Galerkin approximation

$$\Phi_\sigma^{T,v} = -\int_T \nabla\varphi_\sigma \cdot F^v(U^h)dx$$

  with φ_σ being the Lagrange basis function at σ, we set

$$\Phi_\sigma^T = \Phi_\sigma^{T,e} - \Phi_\sigma^{T,v}.$$

The set of relations (2) are solved by a linearized implicit method to be described in section 2.

In practice, the residual Φ_σ are constructed so that the only arguments entering in the formulations are the other degrees of freedom in T. Thanks to this property, we see that if we partition the mesh, we only need an overlap of one element, whatever the degree of the interpolation. There are several ways of implementing this strategy, we explain the simplest one in the following sections. We first describe the implicitation strategy, then how the code has been parallelized.

2 Implicitation of the Scheme

For each degree of freedom, one has to solve the equation

$$\sum_{T,\sigma\in T} \Phi_\sigma^T = 0 \tag{4}$$

where nodal residuals are given in the previous paragraphs.

This is a set of complex non linear equations that are solved by an iterative scheme. The scheme can either be explicit:

$$\mathbf{u}_\sigma^{n+1} = \mathbf{u}_\sigma^n - \omega_\sigma \sum_{T,\sigma\in T} \Phi_\sigma^T(\mathbf{u}^n)$$

or implicit

$$\mathbf{u}_\sigma^{n+1} = \mathbf{u}_\sigma^n - \omega_\sigma \sum_{T,\sigma\in T} \Phi_\sigma^T(\mathbf{u}^{n+1}).$$

In the first case, the local relaxation parameter ω_σ is chosen to satisfy a CFL–like condition. For the implicit scheme, one first proceeds an approximate linearisation of the residuals Φ_σ^T. Defining $\delta\mathbf{u}_{\sigma'} := \mathbf{u}_{\sigma'}^{n+1} - \mathbf{u}_{\sigma'}^n$ for any σ', we have

$$\begin{aligned}\delta\mathbf{u}_\sigma^n &= -\omega_\sigma \sum_{T,\sigma\in T} \Phi_\sigma^T\left(\mathbf{u}_h^{n+1}\right) \\ &\approx -\omega_\sigma \sum_{T,\sigma\in T} \Phi_\sigma^T\left(\mathbf{u}_h^n\right) - \omega_\sigma \sum_{T,\sigma\in T} \sum_{\sigma'\in T} \frac{\partial \Phi_\sigma^T}{\partial \mathbf{u}_{\sigma'}} \cdot \delta\mathbf{u}_{\sigma'}\end{aligned}$$

and then we approximate the Jacobians of the residual by those of the LxF scheme, see chapter 9. We end up to :

$$\left(\frac{I}{\omega_\sigma} + \sum_{T,\sigma\in T} \frac{\partial \Phi_\sigma^{LxF,T}}{\partial U_\sigma}\right)\delta U_\sigma + \sum_{j\neq\sigma}\left(\sum_{T,\sigma\in T} \frac{\partial \Phi_\sigma^{LxF,T}}{\partial U_j}\right)\delta U_j = -\sum_{T,\sigma\in T} (\Phi_\sigma^T)^{\star\star}(u_h^n). \tag{5}$$

Another strategy has also been tested : the true Jacobian is evaluated by a finite difference method. This has proved to be efficient for smooth flows, but we have run into problems when a discontinuity exists in the flow.

The system 5 is solved by standard relaxation. We have chosen Gauss-Seidel or GMRES with ILU(0) preconditioning.

3 Scheme Parallelization

For 3D test cases, our sequential code suffers from two limitations. A high memory footprint and the computational time increase dramatically. So in this part, we will discuss how to speed up the calculation with multiple processors via MPI.

3.1 Theory

The main idea used in a parallel computation is “divide and conquer”. If we have n processors then we will split our computational domain into n sub-domains, this is called mesh partitioning. To be efficient the partitions have to correctly balance the computational cost, i.e. each domain should contains the same number of unknowns.

After the partitioning step we know which processor will compute which unknown (see Fig 1). Unfortunately, the dependency domain of some unknowns is not

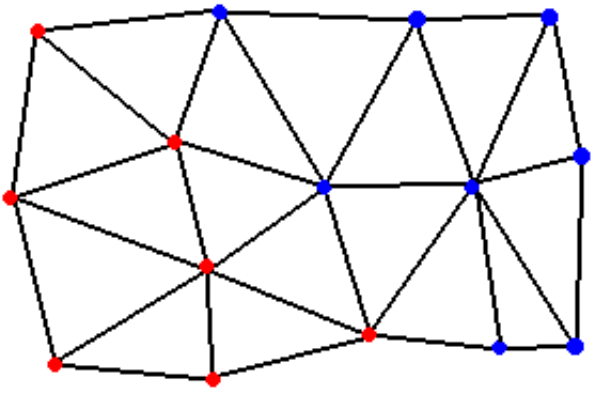


Fig. 1 2 domains partitioning (P1 scheme): Processor's 0 unknowns in red, Processor's 1 unknowns in blue

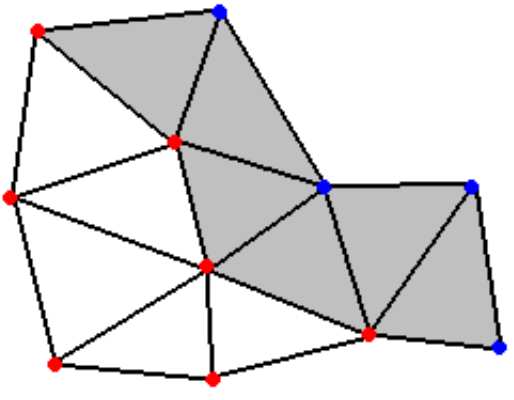


Fig. 2 2 domains partitioning - Overlaps creation (P1 scheme): Processor's 0 calculated unknowns in red, unknowns received from Processor 1 in blue, Overlap's cells in Grey

complete, the missing unknowns are calculated by an other processor. To solve this problem we have to build the "overlaps" by adding the missing unknowns to the domain. These unknowns are not evaluated by the current processor, they receive the missing information from the processor "owning" the unknowns (see Figures 2 and 3).

There are two cases to consider : P1 approximation (second order scheme) and P2 approximation (third order). It is essential, since the evaluation of the sub residuals associated to an element T is done using *only* values associated to that element, not to break this structure in the mesh partitioning.

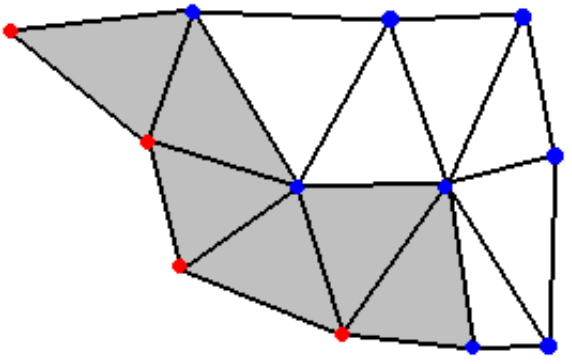


Fig. 3 2 domains partitioning - Overlaps creation (P1 scheme): Processor's 1 calculated unknowns in blue, unknowns received from Processor 0 in red, Overlap's cells in Grey

3.2 Implementation for P1 Scheme

Lets T be a triangulation, N be the nodes and $E \in (N,N)$ be the edges of the triangulation. Let us note P the partition, $P(i)$ with $i \in N$, contains the number of the processor computing the unknown i. The partitioning is done using the mesh partionner SCOTCH [1].

In the P1 scheme the unknowns are located at the mesh nodes. The dependency domain D_{P1} is composed of the first neighbors, i.e. the nodes the unknown shares an edge with.

$$D_{P1}(i) = \{j \in N / (i,j) \in E\} \text{ for } i \in N$$

The construction of the overlap is described by the algorithm 1.

Algorithm 1. Overlap creation algorithm

```
for e = (i, j) ∈ E do
  if P(i) ≠ P(j) then
    add i to the overlap of processor P(j)
    add j to the overlap of processor P(i)
  end if
end for
```

3.3 Implementation for P2 Scheme

In the P2 scheme, the unknowns are located at the mesh nodes and in the middle of the edges. Let us denote $V(i)$ the neighboring elements of node i ($V(e)$ the neighboring elements of edge e). The dependency domain D_{P2} is defined as follow:

$$\begin{cases} D_{P2}(i) = \{j \in N/(i,j) \in E\} \cup \{e \in E/e \in V(i)\} \text{ for } i \in N \\ D_{P2}(e) = \{i \in N/i \in V(e)\} \cup \{e' \in E/e' \in V(e)\} \text{ for } e \in E \end{cases}$$

For the P2 scheme, we keep the P1 partition, in particular for the partionning. the difference with the P1 scheme is in the overlap definition. At this point, the partition P is defined for the unknowns located at the nodes, we have to find a way to uniquely define the partition for the unknowns located at the edges. To do so we use algorithm 2.

Algorithm 2. Modifying P2 partition

```
for e = (i, j) ∈ E do
  if P(i) ≠ P(j) then
    P(e) ← min(P(i), P(j))
  else
    P(e) ← P(i)
  end if
end for
```

The construction of the P2 overlap is described by the algorithm 3.

Algorithm 3. P2 Overlap creation algorithm

```
for every element t ∈ T do
  for every edge e = (i, j) of t do
    if P(i) ≠ P(j) then
      add i to the overlap of processor P(j)
      add j to the overlap of processor P(i)
    end if
    for every node k of t do
      if P(e) ≠ P(k) then
        add e to the overlap of processor P(k)
        add k to the overlap of processor P(e)
      end if
    end for
  end for
end for
```

This partition is not optimal, we have balanced the number of nodes of the mesh, which is not the number of unknowns. For instance, in a NACA test case with 16 processors, the processor with the most nodes has 2% more nodes than the average number of nodes, and the processor with the most unknowns has 10% more unknowns than the average. A more efficient way of proceeding could be to consider the graph of the third accurate scheme, to partition this graph while keeping the locality in the residual definition. This point of view is examined in section 4.2.

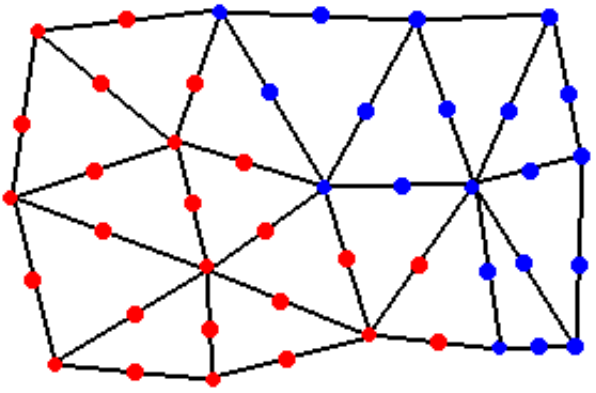


Fig. 4 2 domains partitioning (P2 scheme): Processor's 0 unknowns in red, Processor's 1 unknowns in blue

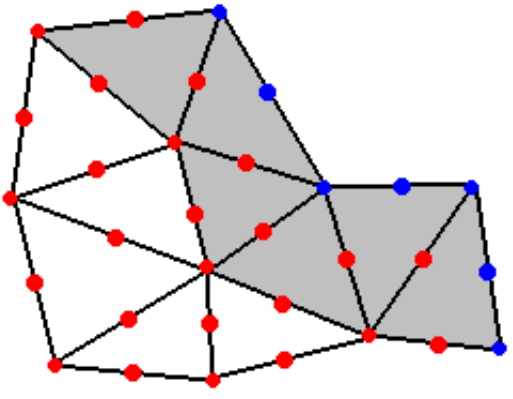


Fig. 5 2 domains partitioning - Overlaps creation (P2 scheme): Processor's 0 calculated unknowns in red, unknowns received from Processor 1 in blue, Overlap's cells in Grey

3.4 Results

In this part we show some results of 2D and 3D Euler P2 tests cases.

3.5 NACA 12

The first test we have run is a NACA 12 at Mach 0.4 with 0 degrees incidence (case MTC1). The Figure 7 shows the mesh partitioning on 16 domains.

The Figure 8 shows the convergence rate of the sequential and parallel code. The convergence rates are not exactly the same because of the iterative method used to solve the system. The results given by the Gauss-Seidel method depends on the

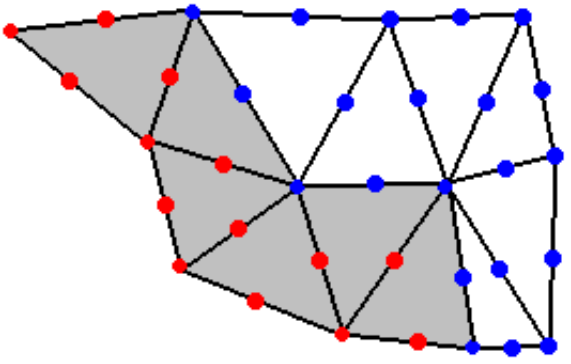


Fig. 6 2 domains partitioning - Overlaps creation (P2 scheme): Processor's 1 calculated unknowns in blue, unknowns received from Processor 0 in red, Overlap's cells in Grey

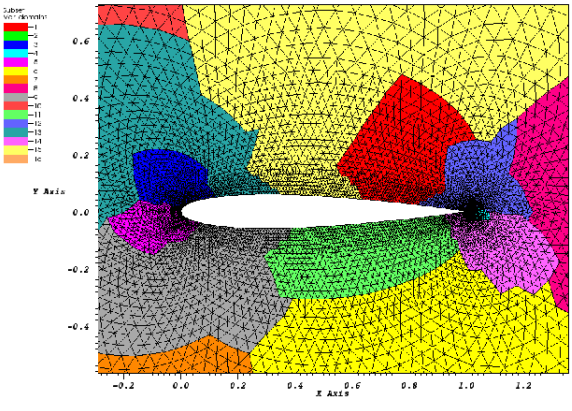


Fig. 7 NACA 12 - mesh partitioning : 16 domains

ordering of the unknowns, which is not the same in sequential and parallel. However the difference in the iterative convergence is quite small.

The figure 9 shows the density contours on the 16 domains at convergence. The figure 10 shows the speedup factor. For a small number of processors (less than 4), we obtain almost the ideal speedup factor, then this degrades. This loss in performance could be explained because of the architecture of the parallel computer we used to do the calculation. This computer is made of computational nodes of 4 processors, all the nodes are connected with infiniband. On a local node the time for data exchange can be neglected, which is no longer the case between 2 processors on different nodes.

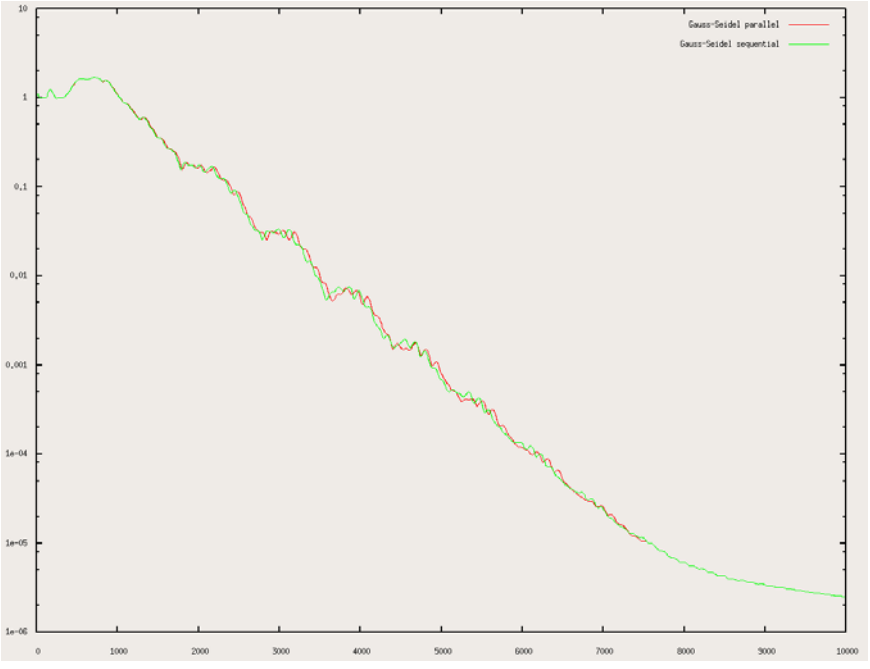


Fig. 8 NACA 12 - convergence

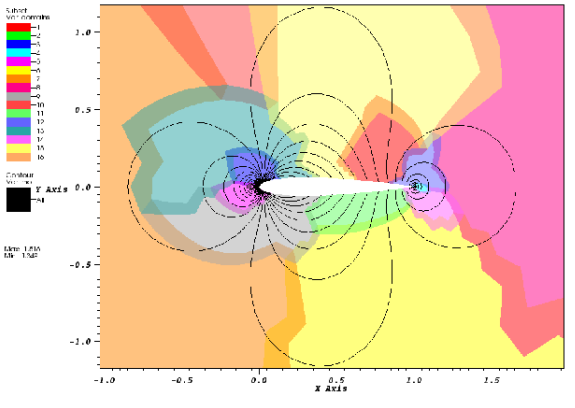


Fig. 9 NACA 12 - density contours : 30 levels

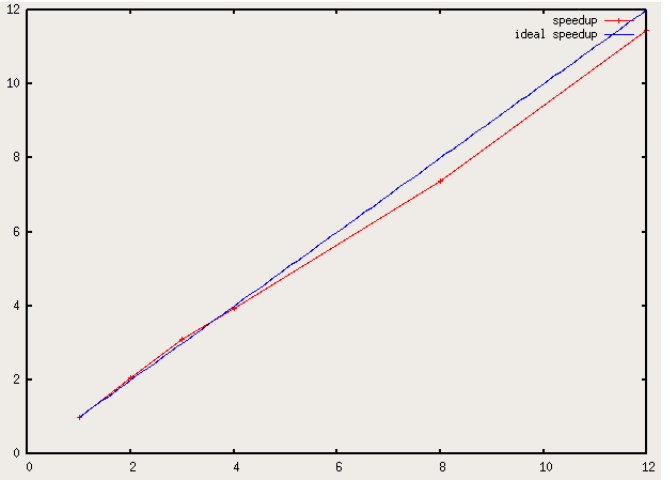


Fig. 10 NACA 12 - Speedup factor

3.6 Adigma Test Case BTC0

Here is some results of the Adigma test case BTC0 (inviscid case). The mesh was partitioned on 16 domains. Figure 11 shows the pressure on the boundaries, and Fig. 12 shows the density contours on the plane z=0. Theses results were obtained after 10 000 time steps. As we can see on Fig. 12 we have some spurious non-physical oscillations near the boundary. This problem was already existing in the sequential scheme and is likely due, as pointed out in chapter 9, to the boundary definition. In this simulation, each triangle of the P1 boundary was subdivided into four smaller sub-triangles by adding the new P2 degrees of freedom. This implies that the true geometry is not better approximated that the P1 one.

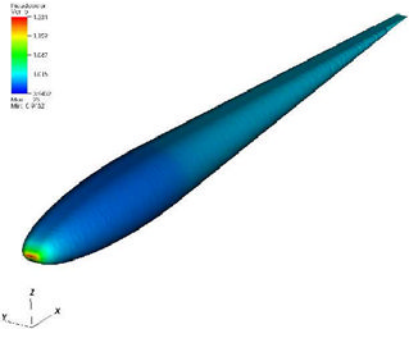


Fig. 11 BTC0 - Pressure

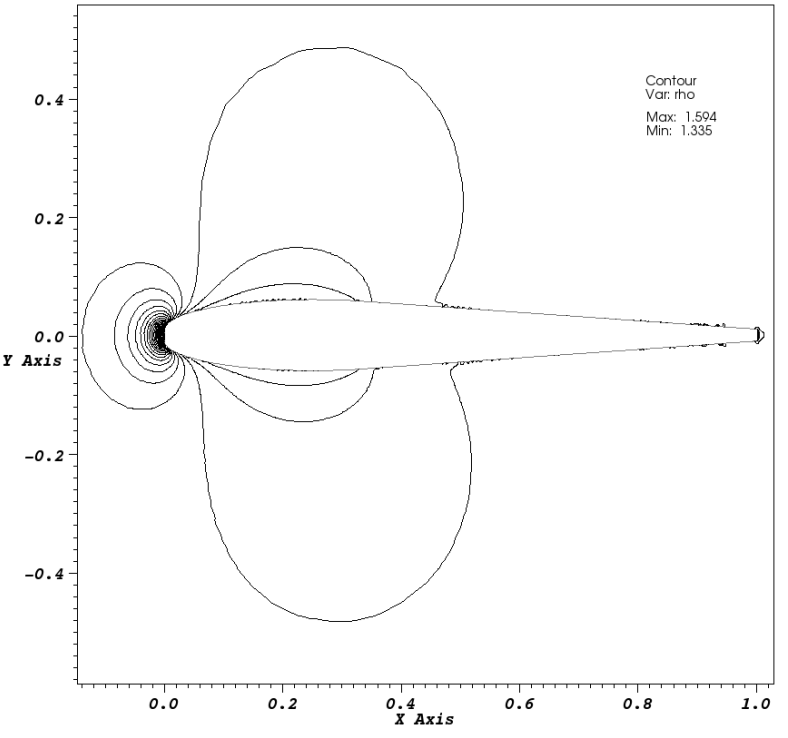


Fig. 12 BTC0 - Density contours : 30 levels on the plane z=0

4 Resolution of Linear Problems

Most of the computer time is spent in solving the implicit (non-)linear equation that describes the implicit phase. The implicit scheme would write

$$\mathscr{F}(U^{n+1}) = 0$$

where the i-th component of the operator $\mathscr{F}$ is (2). Assuming that $\mathscr{F}$ is differentiable, we make the assumption

$$\mathscr{F}(U^{n+1}) \approx \mathscr{F}(U^n) + \frac{\partial \mathscr{F}}{\partial U}(U^n)\left(U^{n+1} - U^n\right)$$

so that we are led to a linear system

$$\frac{\partial \mathscr{F}}{\partial U}(U^n)\left(U^{n+1} - U^n\right) = -\mathscr{F}(U^n) \tag{6}$$

where most of the computer time is spent. We have evaluated the following strategy. When solving (6) with "simple" methods, say Gauss-Seidel or GmRes like

techniques, a naive implementation at the boundary of sub-domains amounts basically to a Jacobi procedure at the interface. What about a truly parallel resolution of (6) assuming that $\frac{\partial \mathscr{F}}{\partial U}(U^n)$ is known ?

Building $\frac{\partial \mathscr{F}}{\partial U}(U^n)$ is trivial (because the scheme is very compact) but solving it efficiently can be tricky. Therefore we have chosen to use a library specialized in solving linear systems in parallel : PASTIX [2]. PASTIX (Parallel Sparse matriX package) is a scientific library that provides a high performance parallel solver for very large sparse linear systems based on direct and block ILU(k) iterative methods. We describe how we integrated this solver in our CFD code, and then we will present some results obtained with it.

4.1 *The* PASTIX *Solver*

PASTIX solves sparse systems of equations of the form $Ax = b$ using block LU (or LL^T for symmetric matrices) methods and block ILU(k) methods.

The LU method consists in writing the matrix A as the product of two matrices L and U, L being a lower triangular matrix and U an upper triangular matrix. Details on this method can be found in [3]. With $A = LU$, solving $Ax = b$ is equivalent as solving $LUx = b$ which is done in two steps:

1. We solve $Ly = b$
2. We solve $Ux = y$

This two steps called forward and backward substitution are trivial because L and U are triangular.

The $ILU(k)$ method consists in computing two triangular matrices L and U so that the residual matrix $R = LU - A$ satisfies certain constraints such as having a small fill-in. Reducing the fill-in means that reducing the amount of memory used by the decomposition and also reducing the computational time by reducing the number of operations. Details on theses methods can be found in [4].

To solve the system we first perform the forward and backward substitution as in the LU method, but because LU is not equal to A we need to refine the solution. This is done by performing an iterative method such as a GMRES method.

4.2 *Implementation*

PASTIX allows us to use its functionalities through 2 different interfaces. The PASTIX interface and the MURGE [5] interface. MURGE is a generic interface that allow the developers to use different solvers through the same function calls. Because of its genericity we chose to use the MURGE interface.

The first thing to do to use PASTIX is to send the graph of the matrix. To do so we have used the function MURGE_GRAPHEDGE. For every segment of the mesh we send the ID of its vertices.

```
DO i = 1, Mesh%Nsegmt
   CALL MURGE_GRAPHEDGE(Mesh%Nubo(1,i), Mesh%Nubo(2,i))
END DO
```

Mesh%Nsegmnt contains the number of segments in the mesh. Mesh%Nubo contains the identities of the vertices of the segments.

Then we can get the mesh partition generated by PaStiX (PaStiX uses the SCOTCH [1] partitionner). We can get the number of local nodes with MURGE_GETLOCALNODENBR, and then we can get the list of the local nodes with MURGE_GETLOCALNODELIST.

```
CALL MURGE_GETLOCALNODENBR(LocalNodes)
ALLOCATE(Nodelist(LocalNodes))
CALL MURGE_GETLOCALNODELIST(Nodelist)
```

After that we can send the values of our matrix to PaStiX with the function MURGE_ASSEMBLYSETNODEVALUES. It takes the coordinates (i,j) and the value we want to set in the matrix.

```
CALL MURGE_ASSEMBLYSETNODEVALUES(i, j, value)
```

The right hand side is sent with MURGE_SETLOCALRHS.

```
CALL MURGE_SETLOCALRHS(rhs)
```

The last step is to solve the system and get the solution. To do so we call the function MURGE_GETLOCALSOLUTION

```
CALL MURGE_GETLOCALSOLUTION(solution)
```

4.3 Results

In order to see the improvements brought by the PaStiX solver we need to define the efficiency of our code. Let n be the number of processors, T_{seq} the time needed for the computation to run on 1 processor, and T_n the time of the computation on n processors. We define the efficiency e of our program as:

$$e = \frac{T_{seq}}{nT_n}$$

$e = 1$ means that our parallelization is very efficient. It is really hard to get $e = 1$ in practice, because it is difficult to have the same load balancing on every processors, i.e. the processors do not do the same amount of work. The other problem when we work in parallel is that we need to exchange data between the processors, this operation cost time and is not present in the sequential algorithm.

An example of the efficiency we get with PaStiX for the MTC1 - level 9 test case is shown Fig. 14. The domain decomposition used for 16 processors is shown on Fig. 13. We can see that the efficiency is not very good, we get a bit more than 0.5 for 16 processors, this means we go only 8 times faster than the sequential

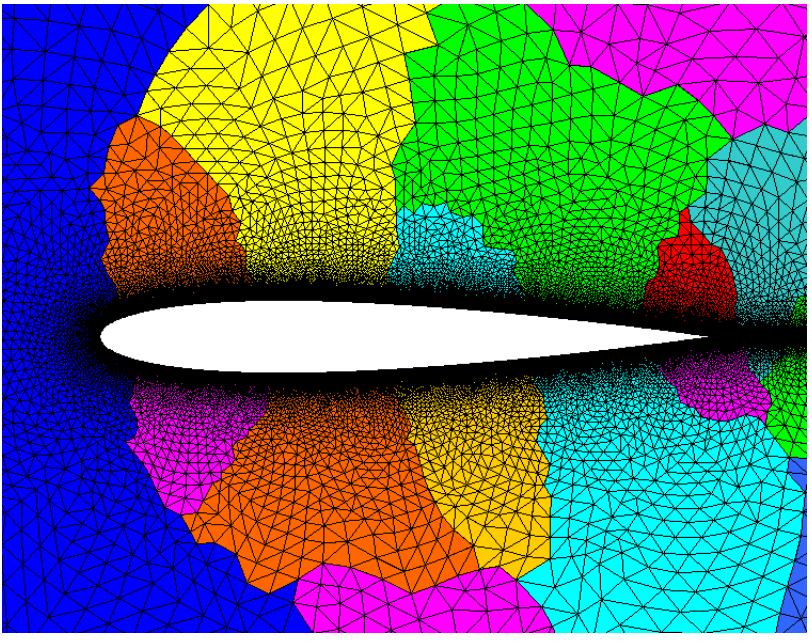


Fig. 13 MTC1 - level 9 - mesh partitioning : 16 domains

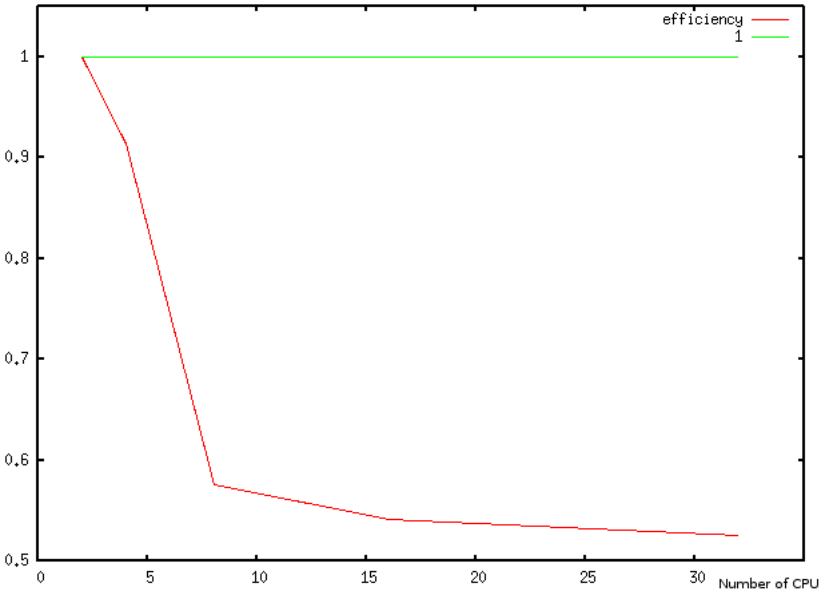


Fig. 14 MTC1 - level 9 - efficiency

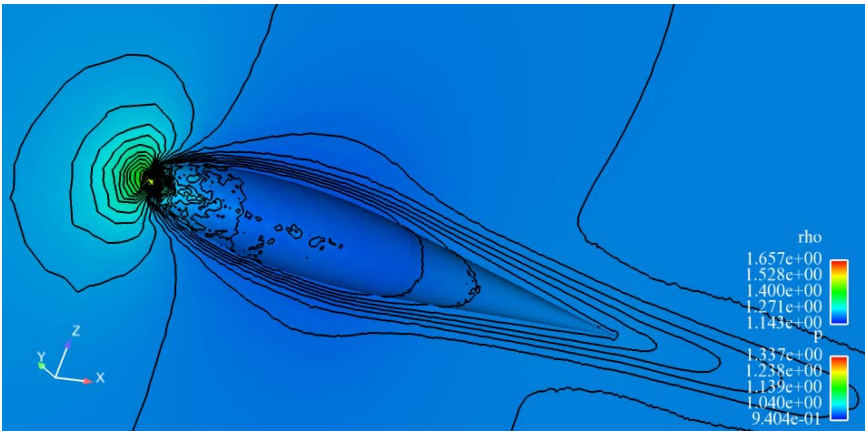


Fig. 15 BTC0 - P2 calculation - Density isolines - Colored by pressure

program. This happens because the matrix is not large enough to scale on a lot of processors. The time needed by each processor to complete its calculations is too short compared to the time needed to exchange data, so the efficiency is not good.

With theses developments we are now able to run some 3D tests cases as we can see in Fig. 15. This is a BTC0 laminar calculation, computed with a P2 Residual Distribution scheme.

5 Concluding Remarks

We developed a parallel version of or third order Residual distribution scheme. This code has been validated on smalls (and medium) 2D and 3D tests cases with satisfactory results. The code is efficient, we are located around 95% of the ideal speedup for computations with 16 processors, and we still have some leads on how to improve it ("true" P2 partition with SCOTCH). We also have explored some parallel strategies for solving the linear systems that need to be solved for implicit schemes. The efficiency is larger than 50%, but we have run into the following problem : the meshes are too small to use efficiently the PastiX and Hips libraries.

References

1. Pellegrini, F.: SCOTCH and LIBSCOTCH 5.1 user's guide. Projet ScAlApplix, INRIA Bordeaux Sud-Ouest ENSEIRB & LaBRI, UMR CNRS 5800 Université Bordeaux I 351 cours de la Libération, 33405 Talence, France (October 14, 2008), http://www.labri.fr/perso/pelegrin/scotch/scotch_en.html
2. Pastix: Solveur direct, http://pastix.gforge.inria.fr

3. Hénon, P., Ramet, P., Roman, J.: PaStiX: A High-Performance Parallel Direct Solver for Sparse Symmetric Definite Systems. Parallel Computing 28(2), 301–321 (2002)
4. Hénon, P., Ramet, P., Roman, J.: On finding approximate supernodes for an efficient ilu(k) factorization. Parallel Computing 34, 345–362 (2008)
5. MURGE, http://murge.gforge.inria.fr

Chapter 16
Hybrid Multigrid DG/FV Methods for Viscous Turbulent Flows

V. Couaillier, F. Renac, and M.C. Le Pape

Abstract. This chapter presents the development of a hybrid technique for coupling Discontinuous Galerkin (DG) and Finite Volume methods for compressible turbulent flow computations in a block-structured code with the RANS system of equations. This code developed at ONERA has been initially built from the following numerical tools : multi-domain approach using structured grids with patched or overlapping interfaces, cell-centered finite volume discretization, space-centered Jameson scheme, multigrid and implicit acceleration techniques (IRS or LDU), boundary condition treatment based on characteristic relations. It has been validated for the simulation of inviscid and turbulent complex internal and external flow configurations and the Finite Volume (FV) functionalities have been integrated in the elsA software environment developed at ONERA which is actually used both by aerospace manufacturers and research laboratories. The code structure enables to re-use as much as possible existing multiblock and multigrid functionalities for the DG method. The RANS system of equations coupled with a $k\omega$ turbulence model are reformulated as a first-order system in space using the mixed DG approach. The boundary condition treatment is performed through a reconstruction of the solution at the physical boundary which avoids the use of a ghost cell technique and improves stability. This method allows space discretization with overlapping and non-matching multidomains as well as high order polynomial approximations of the solution. Numerical examples for 3D inviscid and turbulent flows are presented to demonstrate the capacity of the method, as well as hybrid multidomain multigrid computations.

1 Introduction

The approach used by ONERA for implementing Discontinuous Galerkin (DG) schemes was to directly use a multi-block structured code [1, 2] with industrial capacities in order to take into account as much as possible a complex flow simulation

V. Couaillier · F. Renac · M.C. Le Pape
ONERA BP 72 - 29 avenue de la Division Leclerc FR-92322 Châtillon Cedex
e-mail: Vincent.Couaillier@onera.fr, Florent.Renac@onera.fr, Marie-Claire.Le_Pape@onera.fr

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 225–242, 2010.
springerlink.com

environment. This type of code is a standard for industrial applications due to its efficiency for solving practical aerodynamic problems around complex geometries. The DG method allows higher-order accuracy with a compact stencil well adapted to several implementation aspects. ONERA has combined these two strategies by developing a DG solver for 3D RANS/two-equation turbulence model simulations on hexahedral meshes. Domain decomposition methods have been devised in order to be able to combine the two strategies, FV and DG.

The compressible Reynolds-averaged Navier-Stokes (RANS) equations coupled with a $k-\omega$ turbulence model are considered using a DG space discretization and an explicit time integration based on a Runge-Kutta technique. We reformulate the RANS and turbulence model equations as a first-order system in space using the BR1 scheme [4]. The boundary condition treatment is performed through a reconstruction of the solution at the physical boundary which avoids the use of a ghost cell technique and improves stability. This method allows space discretizations with overlapping and non-matching multidomains as well as high order polynomial approximations of the solution.

Domain decomposition methods have been devised in particular within the distributed computation framework, in which ONERA is concerned. Techniques have been developed for the local grid refinement of block-structured grids, based on patched grids or Hierarchical Mesh Refinement (HMR). Classical methods consist in creating fictitious meshes and in interpolating values and derivatives. Their extension to the coupling of two different discretizations, such as a second-order finite volume scheme and a higher-order DG method, requires the definition of compact conditions which are based here on local fictitious cell reconstruction.

The H-multigrid method has been extensively used for practical 3D turbulent flow configurations for many years in block-structured finite volume codes. In particular, ONERA has studied different strategies of coupling/decoupling approaches for the solution of the RANS system associated to turbulent transport equation models, with various restriction and prolongation operators on hexahedral curvilinear grids, in the context of classical second-order finite volume methods. Based on this experience, the FV H-multigrid has been extended to DG H-multigrid by deriving the schemes on the coarse grids and re-using partially the transfer and prolongation operators.

The paper presents the FV and DG methods used as well as the hybrid coupling and multigrid technique. Some numerical examples are then introduced.

2 Numerical Approach

Let $\Omega \subset \mathbb{R}^3$ be a bounded domain, $\partial\Omega$ denotes the boundary of Ω. In the following, we introduce the DG discretization for turbulent flow simulations. A similar approach is applied for the discretization of laminar equations.

We consider the system of RANS equations coupled with the Wilcox's $k-\omega$ two-equation turbulence model in conservative form:

$$\frac{\partial u}{\partial t} + \nabla \cdot F_c(u) + \nabla \cdot F_v(u, \nabla u) = S(u, \nabla u), \tag{1}$$

where $u = (\rho, \rho u, \rho v, \rho w, \rho E, \rho k, \rho \omega)^t$ denotes the conservative variable vector and T is the transpose operator, ρ stands for the density, $V = (u, v, w)^t$ denotes the velocity vector, and E is the total energy per unit of mass. The turbulent variables are the kinetic energy of turbulence k and the specific rate of dissipation ω. The convective fluxes F_c, the viscous fluxes F_v and the source terms S are defined by (see [5] for details on the parameters):

$$F_c = \begin{pmatrix} \rho V \\ \rho V \otimes V + pI \\ (\rho E + p)V \\ \rho k V \\ \rho \omega V \end{pmatrix}, F_v = \begin{pmatrix} 0 \\ -\tau - \tau_r \\ q + q_t - (\tau + \tau_r) \cdot V \\ -(\mu + \sigma_k \mu_t)\nabla k \\ -(\mu + \sigma_\omega \mu_t)\nabla \omega \end{pmatrix}, S = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \tau_r : \nabla V - \beta^\star \rho k \omega \\ \frac{\gamma \omega}{k} \tau_r : \nabla V - \beta \rho \omega^2 \end{pmatrix},$$

where μ stands for the kinematic viscosity. The turbulent viscosity is defined by $\mu_t = \rho k / \omega$ and p is the static pressure. The system is completed by the Boussinesq assumption:

$$\tau = -\tfrac{2}{3}\mu(\nabla \cdot V)I + \mu(\nabla V + \nabla V^T), \qquad q = -\frac{C_p \mu}{Pr} \nabla T,$$

$$\tau_r = -\tfrac{2}{3}(\rho k + \mu_t \nabla \cdot V)I + \mu_t(\nabla V + \nabla V^T), \; q_t = -\frac{C_p \mu_t}{Pr_t} \nabla T.$$

2.1 Finite Volume Scheme

In a finite volume approach, the semi-discretized formulation of the system of equations reads :

$$\frac{du}{dt} = -\frac{1}{V_{ijk}} \left(\int_{\partial V} (F_C(u) - F_V(u, \nabla u)) n ds - D(u) \right) + S(u, \nabla u) = R(u) \tag{2}$$

where V_{ijk} is a volume element. The above residual operator $R(U)$ contains the contributions of convective fluxes $F_C(u)$, viscous fluxes $F_V(u, \nabla u)$, artificial dissipation fluxes $D(u)$ and source terms $S(u, \nabla u)$.

More precisely the explicit stage is defined by the following expression:

$$E_{F_C, F_V, S, D}\left(u^{(0)}, u^{(q)}\right)_{ijk} = -\frac{1}{V_{ijk}} \left(\int_{\partial V_{ijk}} (F_C^{(q-1)} - F_V{}^{(0)}) n ds - D_{ijk}^{(0)} \right) + S_{ijk}^{(0)}, \tag{3}$$

in which the "cell centered" spatial discretisation is performed with second order space centered scheme. In the explicit stage the convective fluxes are recomputed

at each Runge-Kutta step whereas the viscous fluxes, as well as the numerical dissipation term and the source term, are frozen at step $t^{(0)}$, leading to a second order time accurate scheme in the perfect fluid regions and to a first order time accurate scheme in the viscous regions.

The time step Δt is determined from the convective and viscous stabilitiy criteria by the following expression :

$$\Delta t_{ijk} \;=\; CFL\, min\left(\frac{V}{\lambda_E \overline{S}}, \frac{V^2}{\xi_\mu \lambda_N \overline{S}^2/2}\right)$$

where:

$$\overline{S}^{\,2} \;=\; max\left(S^2_{i+1/2jk}, S^2_{i-1/2jk}\right) + max\left(S^2_{ij+1/2k}, S^2_{ij-1/2k}\right) + max\left(S^2_{ijk+1/2}, S^2_{ijk-1/2}\right)$$

The quantities $S_{l_i l_j l_k}$ represent the oriented surface vector of faces from cell V_{ijk}. The spectral radius λ_E of the convective Jacobian matrix $dF_C(u)/du$ is defined by the following expression:

$$\lambda_E \;=\; |V| \;+\; C,$$

C being the sound speed. The spectral radius λ_N of the diffusive Jacobian matrix $dF_V(u)/du$ is defined by the following expression :

$$\lambda_N \;=\; \frac{\gamma}{\rho}\left(\frac{\mu}{Pr} \;+\; \frac{\mu_t}{Pr_t}\right)$$

An analysis of the $k-\omega$ system shows that the linear stability criterion defined by the convective and diffusive terms is less restrictive than that defined by the analoguous terms of the mean flow system. Nevertheless, this turbulent system of equations has strong non linear behaviour in regions where the level of turbulent kinetic energy k is high, and then the level of eddy viscosity μ_t is also high, so we introduced a coefficient ξ_μ greater than 1 in the time step evaluation. Numerical tests lead to typical values inbetween 2 and 4.

The artificial viscosity is analogous to that proposed by Jameson et al. [3] in a cell vertex approach, modified by Eriksson for the treatment of boundary conditions. It is composed of a non linear second order term for the shock capture and of a linear fourth oder term to ensure the Kreiss dissipative aspect of the scheme :

$$D(u)_{ijk} \;=\; \sum_{l=i,j,k} D(u)_l$$

The artificial viscosity terms in the "i" direction are defined as follows:

$$D(u)_i \;=\; \delta_i\left(\varepsilon^{(2)}\,\delta_i u\right)^n_{ijk} \;-\; \delta_i^2\left(\varepsilon^{(4)}\,\delta_i^2 u\right)^n_{ijk}$$

where the coefficients $\varepsilon^{(2)}_{i\pm1/2jk}$ and $\varepsilon^{(4)}_{ijk}$ depend on local gemetry and aerodynamic field:

$$\varepsilon^{(2)}_{i\pm 1/2jk} = K^{(2)} max\left(\nu^{(i)}_{ijk}, \nu^{(i)}_{i\pm 1jk}\right) \overline{\lambda}_{i\pm 1/2jk},$$

$$\varepsilon^{(4)}_{ijk} = max\left[0, K^{(4)} - \mu_i \varepsilon^{(2)}_{ijk}\right],$$

and

$$\overline{\lambda}_{i\pm 1/2jk} = \lambda_{i\pm 1/2jk}\left(1 + \zeta_1 max(l_i = i, i+1; l_j = j \pm 1/2, l_k = k \pm 1/2)\left(\frac{\lambda_{l_i l_j l_k}}{\lambda_{i\pm 1/2jk}}\right)^{\zeta_2}\right)$$

where the spectral radius of the matrix $(A^x S^x + A^y S^y + A^z S^z)$, is defined by the following relation :

$$\lambda \;=\; |V.S| \;+\; C|S|.$$

For meshes with high cell aspect ratio (this is typically the case for turbulent flows) the parameters ζ_1, ζ_2 are set respectively to 1 and to 1/5. The sensor $\nu^{(i)}_{ijk}$, is based on a combination of second order pressure difference and second order velocity difference :

$$\nu^{(i)}_{ijk} \;=\; \sigma g^{(i)}(p) \;+\; (1 - \sigma) g^{(i)}(|V|)$$

with

$$g^{(i)}(\zeta) \;=\; \frac{|\zeta_{i+1jk} \;-\; 2\zeta_{ijk} \;+\; \zeta_{i-1jk}|}{\zeta_{i+1jk} \;+\; 2\zeta_{ijk} \;+\; \zeta_{i-1jk} \;+\; \varepsilon}$$

The coefficients $K^{(2)}$, $K^{(4)}$ are set to usual values (from 0.25 to 1 for $K^{(2)}$, and from 0.016 to 0.064 for $K^{(4)}$), σ is a parameter allowing to balance the respective wheight of pressure and velocity, and $\varepsilon > 0$ is a small parameter.

2.2 Runge-Kutta Time Integration

The numerical integration is performed using a four-step Runge-Kutta scheme:

$$\begin{cases} u^{(0)} = u^n \\ \vdots \\ \Delta\tilde{u}^{(q)} = \tilde{u}^{(q)} - u^{(0)} = \alpha_q \Delta t E_{F_C, F_V, Q, D}\left(u^{(0)}, u^{(q)}\right) \\ I_{rs}\, \Delta u^{(q)} = \Delta\tilde{u}^{(q)} \\ u^{(q)} = u^{(0)} + \Delta u^{(q)} \\ \vdots \\ u^{n+1} = u^{(4)} \end{cases}$$

In these expressions, $E_{F_C, F_V, S, D}$ represents the explicit operator and I_{RS} the implicit operator. the coefficients α_q are defined as follows :

$$\alpha_1 = \frac{1}{4}\,,\; \alpha_2 = \frac{1}{3}\,,\; \alpha_3 = \frac{1}{2}\,,\; \alpha_4 = 1. \tag{4}$$

2.3 Discontinuous Galerkin Scheme

We recall here the hexahedral DG method developed in the ADIGMA project (see chapter "Development of Discontinuous Galerkin Method for RANS equations on Multibloc Hexahedral Meshes" foe mre details) .The domain Ω is discretized with hexahedral elements: $\Omega_h = \cup V_{ijk}$. The spatial discretizations of the diffusive and source terms are constructed by regarding the gradient of the conservative variables as additional unknowns of the problem

$$G(u) = \nabla u \tag{5}$$

so we have $F_v = F_v(u,G)$ and $S = S(u,G)$.

Multiplying the system equations by the same test function ϕ and integrating by parts element by element leads to the weak formulation of the problem. The discrete version of the weak formulation of the system in each element reads

$$\int_{V_{ijk}} \phi G_h d\Omega - \oint_{\partial V_{ijk}} \phi \hat{u}_h \otimes n dS + \int_{V_{ijk}} \nabla\phi \otimes u_h d\Omega = 0, \tag{6a}$$

$$\int_{V_{ijk}} \phi \frac{\partial u_h}{\partial t} d\Omega + \oint_{\partial V_{ijk}} \phi(\hat{F}_c + \hat{F}_v) \cdot n dS - \int_{V_{ijk}} (F_c + F_v) \cdot \nabla\phi d\Omega - \int_{V_{ijk}} \phi S d\Omega = 0, \tag{6b}$$

where n is the normal unit vector and

$$G_h = \sum_{j=1}^{n} \mathscr{G}_j \phi_j(x), \quad u_h = \sum_{j=1}^{n} \mathscr{U}_j \phi_j(x) \tag{7}$$

represent approximate solutions of the equations where $\mathscr{U}_j$ stand for the degrees of liberty of the problem. The functions ϕ_j represent a basis of the function space of piecewise discontinuous polynomials of degree k inside each element:

$$V_h = \{\phi \in L^2(\Omega_h) : \phi|_{K_i} \in P_k(V_{ijk}), 1 \le i \le N\} \tag{8}$$

where $P_k(K_i)$ represents the space of polynomials in element V_{ijk} of degree at most k. We use the monomials 1, $(x-x_i)$, $(x-x_i)^2$, $(x-x_i)(y-y_i)$, etc. as basis of the function space V_h where $x_i = (x_i, y_i, z_i)^T$ represents the centroid of the element V_{ijk}.

The numerical fluxes in contour integrals of equation (6) are chosen so as to be uniquely defined at the boundary ∂V_{ijk} of each element and to satisfy the consistency relations. We use a centered scheme for the gradient construction:

$$\hat{u}_h = \frac{u^- + u^+}{2}, \tag{9}$$

where u^- denote the internal state evaluated at the interface and u^+ denotes the neighboring interface state. Likewise, the convective fluxes of the mean quantities are discretized by using a Lax-Friedrichs flux with artificial dissipation:

$$\hat{F}_c = F_c\Big(\frac{u^- + u^+}{2}\Big) - k_2\rho_s(u^+ - u^-), \tag{10}$$

where the spectral radius is defined by $\rho_s = \max\{\|V^+\| + c^+, \|V^-\| + c^-\}$, c denotes the speed of sound and the artificial viscosity parameter is set at $k_2 = 0.25$ for viscous calculations and at $k_2 = 0.05$ for inviscid computations.

In order to keep the modular feature of the solver, the time integration procedure is based on a decoupling between RANS and $k-\omega$ systems of equations. This allows the use of specific numerical flux for each system and we use a Roe flux for the turbulent quantities.

The viscous fluxes are replaced by a centered scheme

$$\hat{F}_v = \frac{F_v(u^-, G^-) + F_v(u^+, G^+)}{2}. \tag{11}$$

The surface and volume integrals in equation (6) are evaluated by means of Gauss quadrature formulae in the brick reference element $K_B = \{\xi = (\xi_1, \xi_2, \xi_3)^T : -1 \leq \xi_j \leq 1, 1 \leq j \leq 3\}$ associated to a linear mapping form the reference element to the physical cell. The integrands are evaluated at each Gauss point by using the polynomial approximation of the conservative variables u_h and of their gradients G_h.

Finally, as for the FV scheme, the time integration of the system is accomplished with an explicit four-stage Runge-Kutta method of second order accuracy in time. The same time scheme is used for mean and turbulent variables.

3 Hybrid FV/DG Coupling

Domain decomposition methods have been devised in particular within the distributed computation framework, in which ONERA is concerned. Techniques have been developed for the local grid refinement of block-structured grids, based on patched grids or HMR. Classical methods consist in creating fictitious meshes and in interpolating values and derivatives. Its extension to the coupling of two different discretizations, such as a second-order finite volume scheme FV and a higher-order DG method, requires the definition of compact conditions.

Then the coupling technique used for FV/DG is based on a one raw of external cells at each boundaries to perform the coupling. It is based on a matching treatment initially developed for FV/FV coupling and it is presented hereafter. Let us first consider the different coupling techniques used for generalized matching conditions with a FV scheme.

3.1 Generalized Domain Matching Techniques

For the FV method the different stages in the explicit term evaluation in a current inner cell are the following :
1) Evaluation of $\nabla \cdot V$, ∇V, ∇T
2) Calculation of τ, τ_r, q, q_t, $F_c - F_v$
3) $\nabla \cdot (F_c - F_v)$
4) S

Phases 2) and 4) of the explicit stage discretization consist of simple algebra from quantities kwown at the centers of the cells. These calculations can be applied even for boundary cells. On the contrary, phases 1) and 3) have to be modified when applied to boundary cells, allowing to take into account the physical and matching boundary conditions. Let us consider the evaluation of the quantity

$$G_V = \frac{1}{V}\left(\int_{\partial V} (\phi(u,u_v)nds\right) \tag{12}$$

where $u_v = \nabla u$ and V is a cell C of domain D_1, having a face B which lies on the boundary of D_1 as shown in Fig. 1.

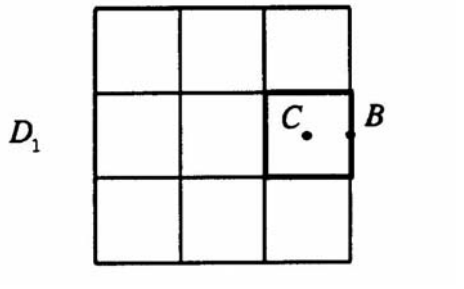
The flux density $\phi(u,u_v)$ on this face is calculated from values of u and u_v estimated on this face. Let us use the subscripts C and B to identify a value at the center of face B. Three boundary types have to be considered :

- for a physical boundary, u_B is estimated from u_C and physical conditions using a treatment based on characteristic relations, and u_{v_B} is a zero order approximation of u_{v_C}.
- for a boundary between two adjacent domains with coincident points, u_B (resp. u_vB) is the mean value of u (resp. u_v) kwown at each adjacent cell, on each domain,
- for a boundary between two adjacent domains D_1 and D_2 with non coincident points, or a boundary of D_1 overlapped by D_2, we define a fictitious cell A adjacent to cell C; u_B (resp. u_{v_B}) is the mean value of u_C (resp. u_{v_C}) and u_A (resp. u_{v_A}) is given by a trilinear interpolation of domain D_2.

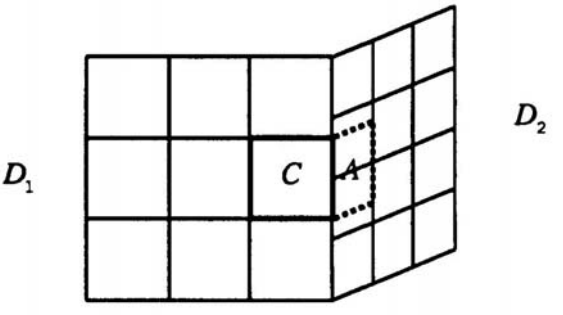
For this third type of boundary, the definition of fictitious cell A is different wether we consider adjacent or overlapping domains. Figure 1 (c) shows this definition for an overlapped domain D_1. We can notice that this approach is also usuable for adjacent domains.

Figure 1 (b) shows that in the case of adjacent domains the fictitious cell of domain D_1 (direction for mesh lines not included in the common plane and width in this direction).

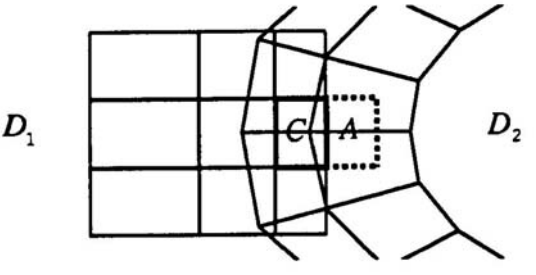
This type of construction of cell A has the nice property of giving the same boundary treatment in the case of adjacent domains with coincident points as the direct treatment without interpolation. Indeed in this case the cells A are identical to the adjacent cells of domain D_2.



(a) Boundary cell notation



(b) Fictitious cell for adjacent domains



(c) Fictitious cell for overlapping domains

Fig. 1 Treatment of domain coupling : coincident, adjacent and overlapping meshes

Concerning the artificial dissipation we use the treatment proposed by Eriksson to evaluate the non linear second order term and the linear fourth order term at the physical boundaries. For the matching boundaries we use information from the adjacent or overlapping domain to evaluate the various differerences appearing in both artificial viscosity terms. For this we use the fictitious cells already defined in the explicit stage.

3.2 Extension to Hybrid FV/DG Coupling

The techniques presented above have been extended to FV/DG coupling. In the case of a P0 approximation for the shape functions the reconstruction at cell faces is based on one point located at the face centers, and then we directkly use the techniques presented above, knowing that the artificial dissipation term in the fluxes is simpler than in the FV approach.

For a DG/FV coupling we have to reconstruct the states at different Gauss points of the boundary cell faces. In that case the technique used is to define on the FV domain interface side the values at the same Gauss point than those needed for the DG scheme. An example is presented in Fig. 2 for a 2D DGP1/FV coupling.

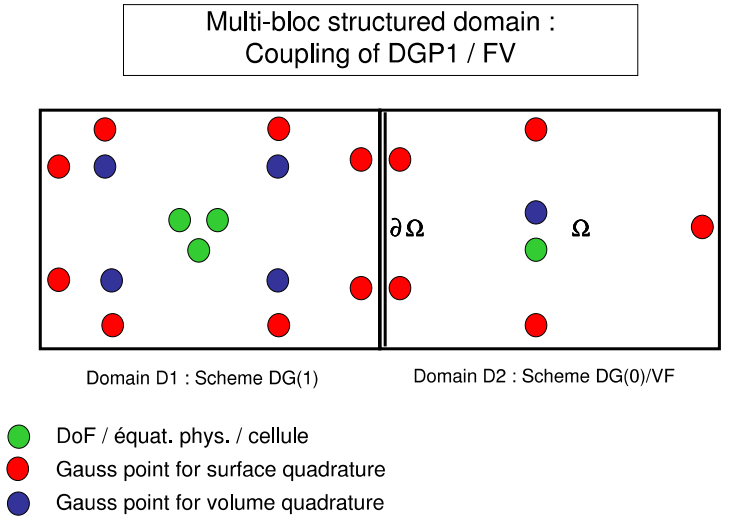


Fig. 2 Hybrid DGP1/FV coupling : Gauss point reconstruction at a common interface

4 Hybrid Multigrid Method

The time integration of this system of ordinary differential equations is carried out using multi-stage Runge-Kutta scheme. To enhance convergence to steady state, local time stepping as well as implicit residual smoothing are used.

The above mentionned iterative approach is well adapted for rapidly damping high frequency error components on a given grid. The remaining errors, associated with the smoother low frequency error components, are responsible for the slow convergence. These low frequency error components on the fine grid appear as higher frequencies on the coarser grid. Thus, to ensure fast convergence of the solution to steady state, a multigrid acceleration technique is developed in this study.

This technique uses a sequence of successively coarser grids to efficiently damp the perturbations. Denote the grid level by a subscript, a sequence of grids h_1,..., h_m, ..., h_M are then defined, where h_1 denotes the finest grid and h_M represents the coarsest grid.

The multigrid strategy employed is the Full Approximation Storage (FAS) scheme in conjunction with Runge-Kutta time stepping proposed by Jameson. This strategy is used to improve the convergence rate of a multi-block solver for the solution of Euler and Reynolds-Averaged Navier-Stokes equations. The Jamesons FAS algorithm for a simple V-Cycle can then be summarised as follows :

- Compute the residual Rh1 R_{h_1} and start with a q-stage Runge-Kutta time stepping to update the solution on the finest level.

The following steps are repeated right up to the coarsest level $m = 2,...,M$ which correponds to the Restriction step :

- Recompute the residual $R_{h_{m-1}}(u_{h_{m-1}})$ on the previous level and calculate the modified residual to be transferred from grid level h_{m-1} to the level h_m:

$$R^{(*)}_{h_{m-1}} = R_{h_{m-1}}(u_{h_{m-1}}) + P_{h_{m-1}},$$

where $P_{h_{m-1}}$ is the added forcing function defined below with P_{h_1} on the finest level.

- Transfer the solution and residual vectors from the previous grid h_{m-1} to the next coarser grid h_m using respectively the fine to coarse transfer operators $T^{h_m}_{h_{m-1}}$ and $\hat{T}^{h_m}_{h_{m-1}}$:

$$\overline{u}_{h_m} = T^{h_m}_{h_{m-1}}\, u_{h_{m-1}}$$
$$\overline{R}_{h_m} = \hat{T}^{h_m}_{h_{m-1}}\, R^{(*)}_{h_{m-1}}$$

- Compute the forcing function for the residuals on the grid level h_m which is the difference between aggregated residuals transferred from grid h_{m-1} and the residuals recalculated on h_m :

$$P_{h_m} = \overline{R}_{h_m} - R_{h_m}(\overline{u}_{h_m}),$$

- where $R_{h_m}(\overline{u}_{h_m})$ is the residual vector computed on the grid level h_m using the transferred solution vector $\overline{u}_{h_m}$ from the previous grid level h_{m-1}.

- Start Runge-Kutta time stepping on the coarse level h_m using the following reformulated version, to take into account the forcing function as well as to include subiterations if necessary, coupled with an implicit smoothing technique (IRS or LU) :

$$
\begin{cases}
u_{h_m}^{(0)} = \overline{u}_{h_m} \; ou \; u_{h_m}^{(q)} \\
\\
\Delta \tilde{u}_{h_m}^{(1)} = \alpha_1 \, \dfrac{\Delta t_{h_m}}{\Omega_{h_m}} \, [R_{h_m}(u_{h_m}^{(0)}) \; + P_{h_m}] \\
\Theta_{h_m} \, \Delta u_{h_m}^{(1)} = \Delta \tilde{u}_{h_m}^{(1)} \\
u_{h_m}^{(1)} = u_{h_m}^{(0)} + \Delta u_{h_m}^{(1)} \\
\vdots \\
\Delta \tilde{u}_{h_m}^{(q)} = \alpha_q \, \dfrac{\Delta t_{h_m}}{\Omega_{h_m}} \, [R_{h_m}(u_{h_m}^{(q-1)}) + P_{h_m}] \\
\Theta_{h_m} \, \Delta u_{h_m}^{(q)} = \Delta \tilde{u}_{h_m}^{(q)} \\
u_{h_m}^{(q)} = u_{h_m}^{(0)} + \Delta u_{h_m}^{(q)}
\end{cases}
$$

Note that upon convergence, when the residual on the finest level goes to zero, the term $R_{h_m}(u_{h_m}^{(0)}) \; + P_{h_m}$ in the above equation which can be rewritten as $R_{h_m}(u_{h_m}^{(0)}) \; + [\overline{R}_{h_m} - R_{h_m}(\overline{u}_{h_m})]$ goes as well to zero. Thus, no correction is computed on the coarser levels and driven back to the finest level.

- Updated solution on coarse grid h_m :

$$u_{h_m} = u_{h_m}^{(q)} \; .$$

The accumulated corrections from each coarser grids are then successively passed-back to finer levels by interpolation ($m = M, ..., 2$). This represents the prolongation step : Transfer the correction from the grid level h_m to the next finer one h_{m-1}.

- Transfer the correction from the grid level h_m to the next finer one h_{m-1}

$$u_{h_{m-1}}^{(+)} = u_{h_{m-1}} + I_{h_m}^{h_{m-1}} \, (u_{h_m}^{(+)} - \overline{u}_{h_m}) \quad avec \quad u_{h_M}^{(+)} \equiv u_{h_M} \; ,$$

where $I_{h_m}^{h_{m-1}}$ is the coarse to fine grid prolongation or interpolation operator from grid h_m to the next finer one h_{m-1}.

- Transfer grid operators

The implemented strategies include V and W cycles as well as options for full multigrid (FMG) versions. In the present multigrid approach, only full coarsening algorithms are employed. Thus, a sequence of coarser grids is extracted from the initial given fine grid by deleting every other grid line in each coordinate direction. Further, the boundary conditions on the coarse grids are treated in the same way as in the fine grid.

Special attention is given to the intergrid transfer operators in the cell-centered formulations in which the variables are located at cell centers.Thus, the transferred

variable locations change from one grid to another, which is not the case in a cell-vertex or node centered formulation. For the fine to coarse operators, the standard approach is used. The transfer of flow variables conserves mass, momentum and energy by the rule:

$$\overline{u}_{h_m} = \frac{\sum \Omega_{h_{m-1}} - u_{h_{m-1}}}{\sum \Omega_{h_{m-1}}}$$

and the residual transferred to grid h_m is the sum of the residuals computed on the eight cells of the fine grid :

$$\overline{R}_{h_m} = \sum R^{(*)}_{h_{m-1}}$$

where the summations range over the cells on the fine grid composing each cell on the coarser grid. For the coarse to fine operator, in order to damp the high frequency errors, an efficient prolongation operator possessing inherent smoothing properties and well adapted for multiblock computational mesh is introduced. The basic idea is to project the cell centered corrections, denoted here by the symbol ϕ, in a conservative manner to the nodes of the coarse grid by the relation :

$$\phi^{(N)}_{h_m} = \frac{\sum\limits_{\{C|N\in C\}} \Omega^{(C)}_{h_m} \phi^{(C)}_{h_m}}{\sum\limits_{\{C|N\in C\}} \Omega^{(C)}_{h_m}} \quad .$$

This conservative smoothing is found to be quite efficient in all our applications. For multi-block computational mesh, the exact interfaces are also taken into account in this cell-to-node projection process. For the finer four cells (in 2D) cells, which are then exactly included in the coarser grid cell, a volume weighted interpolation is used to compute the cell centered corrections :

$$\phi^{(a)}_{h_{m-1}} = \frac{\sum\limits_i \Omega_i \phi^{(i)}_{h_m}}{\Omega} \quad .$$

where Ω is the volume of the coarse grid cell hm given by
with (in 2D) :

$$\Omega = \Omega^{(a)}_{h_{m-1}} + \Omega^{(b)}_{h_{m-1}} + \Omega^{(c)}_{h_{m-1}} + \Omega^{(d)}_{h_{m-1}}$$

and

$$\begin{aligned}
\Omega_1 &= \tfrac{1}{4}\Omega^{(a)}_{h_{m-1}} + \tfrac{1}{2}\Omega^{(b)}_{h_{m-1}} + \Omega^{(c)}_{h_{m-1}} + \tfrac{1}{2}\Omega^{(d)}_{h_{m-1}} \\
\Omega_2 &= \tfrac{1}{4}\Omega^{(a)}_{h_{m-1}} + \tfrac{1}{2}\Omega^{(d)}_{h_{m-1}} \\
\Omega_3 &= \tfrac{1}{4}\Omega^{(a)}_{h_{m-1}} \\
\Omega_4 &= \tfrac{1}{4}\Omega^{(a)}_{h_{m-1}} + \tfrac{1}{2}\Omega^{(b)}_{h_{m-1}}
\end{aligned}$$

which gives the usual linear interpolation stencil containing entries for the cell center a on uniform grids.

For inviscid computations, this leads to an efficient procedure and good convergence properties are obtained for a wide range of 3D applications. Further, to treat complex multi-block configurations with limited number of cells in one direction, the idea of using linear dissipation terms on the coarse grids is also implemented. This consists in using a simple constant coefficient second order dissipation term on the coarser grids instead of the nonlinear artificial dissipation model.

- Strategy for turbulent flows

In the case of the RANS equations, the approach adopted is to compute the viscous terms on the coarser grids too. Thus, their influences are also taken into account in the forcing functions on the coarser grids. Different turbulence models are available in the solver ranging from algebraic models to two equation models. These models are used to compute the turbulent quantities only on the finest grid level. On the coarser grids, they are obtained by interpolating the values from the finest level. This leads to a very direct approach with algebraic models, while with one or two equation models, the corresponding turbulence model equations are solved separately decoupled from the flow equations. In the solver, one Runge-Kutta iteration is carried out to update the turbulent quantitites on the fine grid. Thus, different new turbulence models can easily be included in the present environment.

- Strategy for multigrid cycles

In order to ensure robustness in V cycles without multigrid on turbulent quantities, sub-iterations are performed on the corresponding equations (let say 2 subiterations on $k-\omega$ system for a V cycle with 2 or 3 grids). Concerning the present multigrid DG implementation, we have used a P1 approximation on the fine grid and a P0 approximation on the coarse grid, leading to small overcost when using multigrid (20% additional cost per iteration).

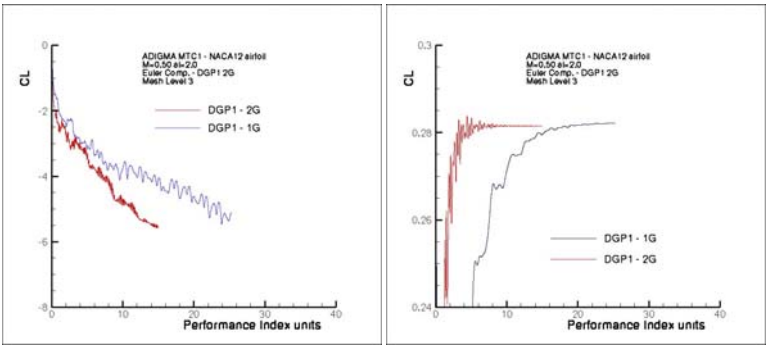
5 Applications

5.1 Inviscid Test Cases: Hybrid Multigrid Method

Figures 3 and 4 present Euler computations of the DG method for two inviscid flows around the NACA0012 airfoil using a DG discretization : a subsonic flow at freestream Mach numer $M = 0.5$ with an angle of attack of $\alpha = 2^\circ$, and a transsonic flow at freestream Mach number $M = 0.8$ and $\alpha = 1.25^\circ$. The space is discretized with a 2D structured C-grid with 225x33 nodes. Results of the DG method are obtained with a $P1$ approximation of the solution. The volume and contour integrals in (6) are calculated with second-order Gauss quadrature formulae.

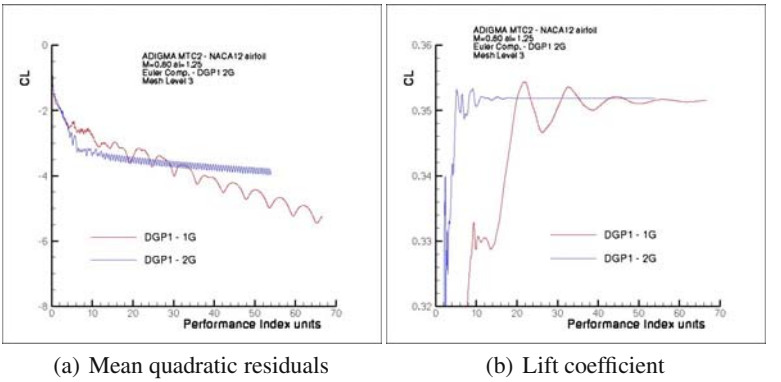
For the subsonic case, as shown by the residual history, both monogrid and multigrid computations converge well. In term of Performance Index Units the gain obtained in CPU reduction is good, especially when considering the lift coeffocient

evolution. Concerning the transonic case, the analysis of the L2 norm of the density residual shows that whereas the multigrid computation better converges in a first stage, it is not the case in the second stage. This residual evolution behaviour could be generated by some local limit cycle due to numerical parameters which have not been optimised. Nevertheless Fig. 4 shows a very good convergence for the multigrid computation, with a gain in CPU time better in this transonic case than that obtained in the subsonic case. Concerning accuracy and for this relatively coarse mesh the improvement obtained in shock wave representation by using the DGP1 scheme instead of the FV JST is clearly demonstrated in Fig. 5.



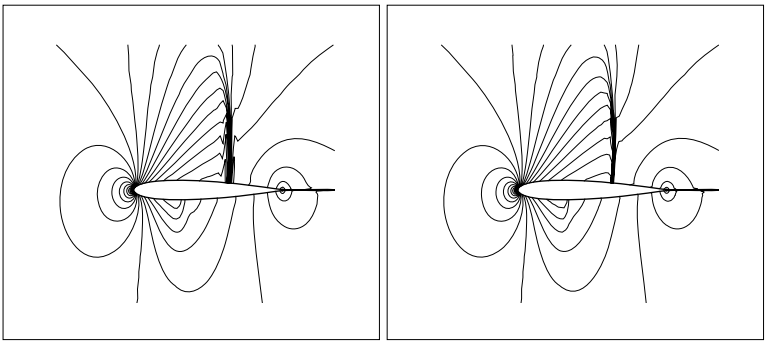
(a) Mean quadratic residuals (b) Lift coefficient

Fig. 3 Convergence history of the DGP1/FV hybrid multigrid computation - subsonic case - Mesh (225×33 grid, $P1$ approximation)



(a) Mean quadratic residuals (b) Lift coefficient

Fig. 4 Convergence history of the DGP1/FV hybrid multigrid computation - transonic case - (225×33 grid, $P1$ approximation)



(a) FV JST scheme (b) DGP1 LLF scheme

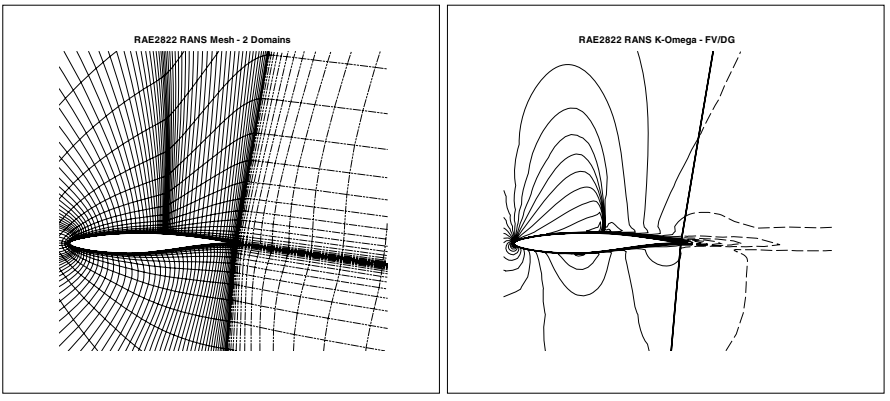
Fig. 5 Iso-Mach number lines - FV and DGP1 computations - transonic case - (225×33 grid)

5.2 Turbulent Test Case: FV/DGP1 Multidomain Coupling

The hybrid FV/DGP1 domain coupling method has been applied to a turbulent test case. This test case consists in a turbulent flow around a RAE2822 airfoil at an incidence of $\alpha = 2.79^\circ$ with a freestream Mach number of $M = 0.79$ and a Reynolds number, based on freestream quantities and chord length, of $Re = 2.5 \times 10^6$. The wall is assumed to be adiabatic and a no slip condition is applied. The domain is discretized by using a structured C-grid with 257×129 nodes. The RANS equations are coupled to the $k - \omega$ turbulence model of Wilcox. Figure 6 presents the mesh split into two domains and the iso-Mach number lines obtained for this hybrid calculation. The convergence in the DGP1 domain (downstream domain) is slow but reaches good level for a CFL number equal to 0.4 wich was impossible to use for a full DG computation. This highlight that the stability problems we have got in the full DGP1 turbulent computation were certainly due to boundary conditions treatment of the turbulent variables.

5.3 ONERA M6 Wing: DGP1 RANS Computation

In order to validate the capacity of the implemented DGP1 RANS scheme to compute 3D configurations the ONERA M6 wing [6] has been calculated. The flowfield is computed here at a free stream Mach number of 0.836, an angle of attack of 6.06 deg. and a free stream Reynolds number of $11.7x10^6$. The C-O mesh used for the computations is composed of 193x49x65 points, corresponding to $y^+ = 1$ almost everywhere on the wing. The DGP1 computation has been performed on the fine grid with a CFL number equal to 0.2 and one coarse grid has been used with the FV JST scheme to accelerate the convergence. Figure 7 presents the iso pressure lines on the upper surface.



(a) 2 domain mesh (b) FV/DGP1 computation

Fig. 6 RAE2822 airfoil: Hybrid FV/DGP1 RANS computation - (257x129$grid$)

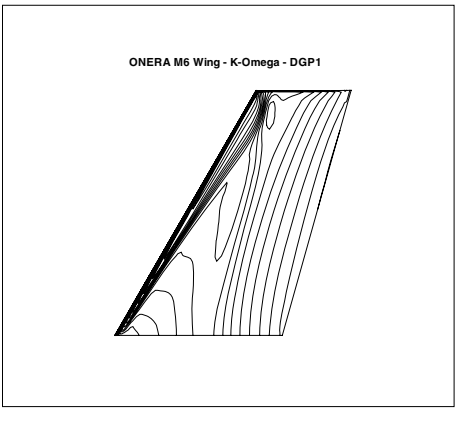


Fig. 7 ONERA M6 Wing: DGP1 RANS computation - Mesh 193x49x65

6 Conclusion

A hybrid DG/FV domain coupling method has been developed in the 3D DG multi-block structured solver. The method uses the DGP1 scheme on the fine grid developed by Onera in WP3.1 of the ADIGMA project. At the present time the DGP2 approximation has not been coupled to FV scheme but it is possible to implement it in an analogous way to that used for DGP1/FV coupling. It is based on a solution reconstruction on the FV interface side at the same Gauss points than those used in the DG domain. It has been tested on 2D turbulent simulations and presents good robustness properties. This technique needs now to be evaluated on more complex

configurations in order to analyze the benefits compared to a full DG approach. Moreover an hybrid multigrid technique has been implemented allowing the use of DGP1 scheme on the fine grid and FV scheme on the coarse grid. The efficiency obtained is higher than that obtained using DGP1 on the fine grid and DGP0 on the coarse grids. The output is a unique software environment for solving flows on multi-block meshes with the option of using classical Finite Volume (FV) methods or DG methods (P0, P1 and P2 approximations), with the capacity of using FV in particular domains and DG in other domains.

References

1. Vuillot, A.M., Couaillier, V., Liamis, N.: 3D Turbomachinery Euler and Navier-Stokes Calculations with Multidomain Cell-Centered Approach. AIAA Paper 93-2576 (1993)
2. Couaillier, V.: Numerical simulation of separated turbulent flow based on the solution of RANS/low Reynolds two-equation model. AIAA Paper 99-0154 (1999)
3. Jameson, A., Schmidt, W., Turkel, E.: Numerical Simulation of the Euler Equations by Finite Volume Method Using Runge-Kutta Time Stepping Schmes. AIAA Paper 81-1259 (1981)
4. Bassi, F., Rebay, S.: A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. J. Comput. Phys. 131, 267–279 (1997)
5. Wilcox, D.C.: Reassesment of the scale-determining equation for advanced turbulence models. AIAA J. 26, 1299–1310 (1979)
6. Schmitt, V., Charpin, F.: Pressure distributions on the ONERA M6-wing at Transonic Mach. numbers. AGARD TR 138 (1988)

Chapter 17
Semi-implicit Time Discretization of the Discontinuous Galerkin Method for the Navier-Stokes Equations

Vít Dolejší, Martin Holík, and Jiří Hozman

Abstract. We deal with the numerical solution of the compressible Navier-Stokes equations with the aid of the discontinuous Galerkin method. The space semi-discretization leads to a stiff system of ordinary differential equations. In order to accelerate a convergence to the steady-state solution we employ the semi-implicit time discretization which leads to the solution of linear algebra system at each time level. We focus on the solution of the arising linear algebra systems and propose a new efficient strategy for the steady-state solutions. The efficiency is demonstrated by a set of numerical experiments.

1 Introduction

In Section II.1 of this book, we presented the space semi-discretization of the compressible Navier-Stokes equations with the aid of the interior penalty Galerkin (IPG) method. This leads to a system of ordinary differential equations (ODEs) for $\mathbf{w}_h(t)$ which has to be discretized by a suitable method.

The explicit time discretization gives a strong restriction on the size of the time step. On the other hand, a full implicit scheme leads to a necessity to solve a nonlinear system of algebraic equations at each time step which is rather expensive. Therefore, we develop here a *semi-implicit method* which is based on a suitable linearization of the Euler fluxes. The linear terms are treated implicitly whereas the nonlinear ones explicitly which leads to a linear algebraic problem at each time step.

Vít Dolejší · Martin Holík · Jiří Hozman
Charles University Prague, Faculty of Mathematics and Physics, Sokolovská 83, Prague, Czech Republic
e-mail: dolejsi@karlin.mff.cuni.cz, marthelh@seznam.cz, jhozmi@volny.cz

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 243–255, 2010.
springerlink.com

Numerical solution of these linear algebra system by a "black box" iterative solver consume approximately 95-99% of the total computational time. Therefore, we focus on a significant reduction of computational time needed for the solution of the linear algebra problems. Moreover, the amount of the used computer memory has to be taken into account. We develop an efficient solution strategy for the mentioned algebraic problems, namely we deal with the choice of a stopping criterion and the size of the time step.

2 Compressible Flow Problem

Let $\Omega \subset \boldsymbol{R}^d$, $d = 2,3$ be a bounded domain and $T > 0$. We set $Q_T = \Omega \times (0,T)$ and by $\partial\Omega$ denote the boundary of Ω. The system of the Navier-Stokes equations describing a motion of viscous compressible fluids can be written in the dimensionless form

$$\frac{\partial \mathbf{w}}{\partial t} + \sum_{s=1}^{d} \frac{\partial \mathbf{f}_s(\mathbf{w})}{\partial x_s} = \sum_{s=1}^{d} \frac{\partial}{\partial x_s} \left(\sum_{k=1}^{d} \mathsf{K}_{sk}(\mathbf{w}) \frac{\partial \mathbf{w}}{\partial x_k} \right) \quad \text{in } Q_T, \tag{1}$$

where $\mathbf{w} = (\rho, \rho v_1, \ldots, \rho v_d, e)^{\mathrm{T}}$ is the *state vector*, $\mathbf{f}_s : \boldsymbol{R}^{d+2} \to \boldsymbol{R}^{d+2}$, $s = 1,\ldots,d$ are the inviscid (Euler) fluxes and $\mathsf{K}_{sk} : \boldsymbol{R}^{d+2} \to \boldsymbol{R}^{(d+2)\times(d+2)}$, $s,k = 1,\ldots,d$ represent the viscous terms. The forms of vectors $\mathbf{f}_s$, $s = 1,\ldots,d$ and matrices K_{sk} can be found, e.g., in [4] or [9, Section 4.3]. The system (1) is equipped with a suitable set of the initial and boundary conditions, see [3], [4].

Let us mention that the Euler fluxes $\mathbf{f}_s$, $s = 1,\ldots,d$ satisfy (see [9, Lemma 3.1]) $\mathbf{f}_s(\mathbf{w}) = \mathsf{A}_s(\mathbf{w})\mathbf{w}$, $s = 1,\ldots,d$ where $\mathsf{A}_s(\mathbf{w}) = \frac{D\mathbf{f}_s(\mathbf{w})}{D\mathbf{w}}$, $s = 1,\ldots,d$ are the Jacobi matrices of the mappings $\mathbf{f}_s$. Finally, we define a matrix

$$\mathsf{P}(\mathbf{w},\mathbf{n}) = \sum_{s=1}^{d} \mathsf{A}_s(\mathbf{w}) n_s, \qquad \mathbf{n} = (n_1,\ldots,n_d). \tag{2}$$

3 Interior Penalty Galerkin Discretization

3.1 Linearization

In Section II.1, we presented a discretization of inviscid, viscous and penalty terms represented by the forms $\tilde{\mathbf{a}}_h$, $\tilde{\mathbf{b}}_h$ and $\tilde{\mathbf{J}}_h^{\sigma}$, respectively. In order to proceed to the semi-implicit time discretization, we introduce the linearization of the inviscid and viscous terms.

3.1.1 Inviscid Terms

For $\mathbf{w}_h, \bar{\mathbf{w}}_h, \varphi_h \in \mathbf{S}_{h\mathsf{p}}$, we define the forms

$$\mathbf{b}_h(\bar{\mathbf{w}}_h,\mathbf{w}_h,\varphi_h) = -\sum_{K\in\mathscr{T}_h}\int_K\sum_{s=1}^{d}\mathsf{A}_s(\bar{\mathbf{w}}_h)\mathbf{w}_h\cdot\frac{\partial\varphi_h}{\partial x_s}\,dx \tag{3}$$

$$+\sum_{\Gamma\in\mathscr{F}_h^I}\int_\Gamma\left(\mathsf{P}^+\left(\langle\bar{\mathbf{w}}_h\rangle,\mathbf{n}\right)\mathbf{w}_h|_\Gamma^{(p)}+\mathsf{P}^-\left(\langle\mathbf{w}_h\rangle,\mathbf{n}\right)\mathbf{w}_h|_\Gamma^{(n)}\right)\cdot[\varphi_h]dS$$

$$+\sum_{\Gamma\in\mathscr{F}_h^{io}}\int_\Gamma\left(\mathsf{P}^+\left(\langle\bar{\mathbf{w}}_h\rangle,\mathbf{n}\right)\mathbf{w}_h|_\Gamma^{(p)}\right)\cdot[\varphi_h]\,dS+\sum_{\Gamma\in\mathscr{F}_h^{w}}\int_\Gamma F_W(\bar{\mathbf{w}}_h,\mathbf{w}_h,\mathbf{n})\cdot\varphi_h\,dS,$$

$$\mathbf{b}_h^B(\bar{\mathbf{w}}_h,\varphi_h) = -\sum_{\Gamma\in\mathscr{F}_h^{io}}\int_\Gamma\left(\mathsf{P}^-\left(\langle\bar{\mathbf{w}}_h\rangle,\mathbf{n}\right)\bar{\mathbf{w}}_h|_\Gamma^{(n)}\right)\cdot[\varphi_h]\,dS,$$

where $\mathsf{A}_s(\cdot)=1,\ldots,d$ are the Jacobi matrices of the mappings $\mathbf{f}_s,\ s=1,\ldots,d$ $\mathsf{P}^{\pm}(\cdot,\cdot)$ are the positive and negative parts of the matrix $\mathsf{P}(\cdot,\cdot)$ given by (2) which define the Vijayasundaram numerical flux [11] used for the approximation of inviscid fluxes though $\Gamma\in\mathscr{F}_h$. This numerical flux is suitable for the semi-implicit time discretization. Moreover, $\tilde{F}_W(\bar{\mathbf{w}}_h,\mathbf{w}_h,\mathbf{n})=(\gamma-1)DF_W(\bar{\mathbf{w}}_h,\mathbf{n})\mathbf{w}_h$, where $DF_W(\mathbf{w},\mathbf{n})$ is a $(d+2)\times(d+2)$ matrix obtained by a differentiation of $\mathbf{f}_s$, see [3], [4] or [7]. Finally, $\bar{\mathbf{w}}|_\Gamma^{(n)}=LRP(\bar{\mathbf{w}}|_\Gamma^{(p)},\mathbf{w}_D,\mathbf{n}_\Gamma),\ \Gamma\in\mathscr{F}_h^{io}$ where $LRP(\cdot,\cdot,\cdot)$ represents a solution of the *local Riemann problem* considered on edge $\Gamma\in\mathscr{F}_h^{io}$ and $\mathbf{w}_D$ is a given state vector (e.g. from far-field boundary conditions), see [6]. For more details, we refer to [7].

3.1.2 Viscous Terms

For $\bar{\mathbf{w}}_h,\mathbf{w}_h,\varphi_h\in\mathbf{S}_{hp}$, we define the forms

$$\mathbf{a}_h(\bar{\mathbf{w}}_h,\mathbf{w}_h,\varphi_h) = \sum_{K\in\mathscr{T}_h}\int_K\sum_{s,k=1}^{d}\left(\mathsf{K}_{s,k}(\bar{\mathbf{w}}_h)\frac{\partial\mathbf{w}_h}{\partial x_k}\right)\cdot\frac{\partial\varphi_h}{\partial x_s}\,dx \tag{4}$$

$$-\sum_{\Gamma\in\mathscr{F}_h^{ID}}\int_\Gamma\sum_{s=1}^{d}\left\langle\sum_{k=1}^{d}\mathsf{K}_{s,k}(\bar{\mathbf{w}}_h)\frac{\partial\mathbf{w}_h}{\partial x_k}\right\rangle n_s\cdot[\varphi_h]\,dS$$

$$-\theta\sum_{\Gamma\in\mathscr{F}_h^{ID}}\int_\Gamma\sum_{s=1}^{d}\left\langle\sum_{k=1}^{d}\mathsf{K}_{s,k}^{\mathrm{T}}(\bar{\mathbf{w}}_h)\frac{\partial\varphi_h}{\partial x_k}\right\rangle n_s\cdot[\mathbf{w}_h]\,dS,$$

$$\mathbf{a}_h^B(\bar{\mathbf{w}}_h,\varphi_h) = -\theta\sum_{\Gamma\in\mathscr{F}_h^{D}}\int_\Gamma\sum_{s,k=1}^{d}\mathsf{K}_{s,k}^{\mathrm{T}}(\bar{\mathbf{w}}_h)\frac{\partial\varphi_h}{\partial x_k}\,n_s\cdot\mathbf{w}_B\,dS.$$

The state vector $\mathbf{w}_B$ prescribed on $\partial\Omega_i\cup\partial\Omega_w$ is given by the boundary conditions, see [3] or [4]. The value of θ appearing in (4) defines different variant of IPG method, namely $\theta=1$ – *symmetric interior penalty Galerkin* (SIPG), $\theta=-1$ – *non-symmetric interior penalty Galerkin* (NIPG), or $\theta=0$ – *incomplete interior penalty Galerkin* (IIPG).

3.1.3 Interior and Boundary Penalties

For $\mathbf{w}_h, \varphi_h \in \mathbf{S}_{hp}$, we define the forms

$$\mathbf{J}_h^{\sigma}(\mathbf{w},\varphi) = \sum_{\Gamma\in\mathscr{F}_h^{ID}} \int_{\Gamma} \sigma[\mathbf{w}]\cdot[\varphi]\,dS, \qquad \mathbf{J}_h^{B,\sigma}(\varphi) = \sum_{\Gamma\in\mathscr{F}_h^{D}} \int_{\Gamma} \sigma \mathbf{w}_B\cdot\varphi\,dS, \tag{5}$$

where $\mathbf{w}_B$ is the boundary state and the penalty parameter σ is chosen by $\sigma|_{\Gamma} = C_W/(\mathrm{diam}(\Gamma)Re)$, $\Gamma\in\mathscr{F}_h^{ID}$, where Re is the Reynolds number and $C_W > 0$ is a suitable constant whose choice depends on the used variant of the IPG method (NIPG, IIPG or SIPG) and the degree of polynomial approximation, see [4] where a numerical study was presented.

3.2 Semi-implicit Time Discretization

In order to simplify the notation, for $\bar{\mathbf{w}}_h$, $\mathbf{w}_h$, $\varphi_h \in \mathbf{S}_{hp}$, we put

$$\begin{aligned} \mathbf{c}_h(\bar{\mathbf{w}}_h,\mathbf{w}_h,\varphi_h) &= \mathbf{a}_h(\bar{\mathbf{w}}_h,\mathbf{w}_h,\varphi_h) + \mathbf{b}_h(\bar{\mathbf{w}}_h,\mathbf{w}_h,\varphi_h) + \mathbf{J}_h^{\sigma}(\mathbf{w}_h,\varphi_h), \\ \mathbf{c}_h^B(\bar{\mathbf{w}}_h,\varphi_h) &= \mathbf{a}_h^B(\bar{\mathbf{w}}_h,\varphi_h) + \mathbf{b}_h^B(\bar{\mathbf{w}}_h,\varphi_h) + \mathbf{J}_h^{B,\sigma}(\varphi_h). \end{aligned} \tag{6}$$

It is possible to show (see, e.g., [3], [4]) that if $\mathbf{w}: \Omega\times(0,T)\to \boldsymbol{R}^{d+2}$ is a continuously differentiable function satisfying the Navier-Stokes equations (1) and the corresponding initial and boundary conditions then

$$\frac{d}{dt}(\mathbf{w},\varphi) + \mathbf{c}_h(\mathbf{w},\mathbf{w},\varphi) = \mathbf{c}_h^B(\mathbf{w},\varphi) \quad \forall\varphi\in\mathbf{S}_{hp}, \tag{7}$$

where $(\cdot,\cdot)$ denotes the L^2-scalar product over Ω.

In [4], we introduced the following method. Let $0 = t_0 < t_1 < t_2 < \ldots t_r = T$ be a partition of the time interval $(0,T)$ and $\mathbf{w}_h^k \in \mathbf{S}_{hp}$ denotes a piecewise polynomial approximation of $\mathbf{w}_h(t_k)$, $k = 0,1,\ldots,r$.

Definition 1. We define the *approximate solution* of (1) by the 1-step BDF-DGFE (backward difference formulae - discontinuous Galerkin finite element) scheme as functions $\mathbf{w}_{h,k}$, $k = 1,\ldots,r$, satisfying the conditions

a) $\mathbf{w}_{h,k}\in\mathbf{S}_{hp}$, (8)

b) $\dfrac{1}{\tau_k}(\mathbf{w}_{h,k},\,\varphi_h) + \mathbf{c}_h(\mathbf{w}_{h,k-1},\,\mathbf{w}_{h,k},\,\varphi_h) = \mathbf{c}_h^B(\mathbf{w}_{h,k-1},\,\varphi_h) \quad \forall\varphi_h\in\mathbf{S}_{hp}$

c) $\mathbf{w}_{h,0}\in S_{hp}$ is an approximation of $\mathbf{w}^0$.

Remark 1. The 1-step BDF-DGFE scheme (8), a) – c) has only the first order of accuracy with respect to time which is sufficient for the seeking of the steady-state solutions. For n-step BDF-DGFE scheme ($n\geq 2$) see [4], [5].

Remark 2. The resulting BDF-DGFE method is practically unconditionally stable, has a high order of accuracy with respect to the time and space coordinates and at each time step we have to solve only one linear algebra problem, which will be discussed in the following section.

4 Solution of Linear Algebra Problems

4.1 Linear Algebra Representations

Since $\mathbf{S}_{h\mathsf{p}}$ is a finite dimensional space of discontinuous piecewise polynomial functions, it is natural to construct its basis in such a way that the support of each basis function lies within one $K \in \mathscr{T}_h$. Then, let $B = \{\psi_j,\ \psi_j \in \mathbf{S}_{h\mathsf{p}},\ j = 1,\ldots,\mathrm{dof},\ \mu \in I\}$ denote a basis of $\mathbf{S}_{h\mathsf{p}}$ with dimension dof.

Therefore, a function $\mathbf{w}_{h,k} \in \mathbf{S}_{h\mathsf{p}}$ can be written in the form

$$\mathbf{w}_{h,k}(x) = \sum_{j=1}^{\mathrm{dof}} \xi_{k,j}\psi_j(x), \quad x \in \Omega,\ k = 0,1,\ldots,r, \tag{9}$$

where $\xi_{k,j} \in \boldsymbol{R},\ j = 1,\ldots,\mathrm{dof},\ k = 0,\ldots,r$. Moreover, for $\mathbf{w}_{h,k} \in \mathbf{S}_{h\mathsf{p}}$, we define a vector of its basis coefficients by $\xi_k = \left\{\xi_{k,j}\right\}_{j=1,\ldots,\mathrm{dof}} \in \boldsymbol{R}^{\mathrm{dof}},\ k = 0,1,\ldots,r$. Using (9) we have an isomorphism

$$\mathbf{w}_{h,k} \in \mathbf{S}_{h\mathsf{p}} \quad \longleftrightarrow \quad \xi_k \in \boldsymbol{R}^{\mathrm{dof}}. \tag{10}$$

Finally, if B is an *orthonormal basis* (which can be simply constructed by an orthogonalization procedure element-wise) then we have

$$\|\mathbf{w}_{h,k}\|_{L^2(\Omega)} = \|\xi_k\|_{\ell^2} \tag{11}$$

for any $\mathbf{w}_{h,k} \in \mathbf{S}_{h\mathsf{p}}$ and the corresponding $\xi_k \in \boldsymbol{R}^{\mathrm{dof}}$ via (10).

Then the problems (8) can be written in the matrix form:

$$\text{find } \xi_k \in \boldsymbol{R}^{\mathrm{dof}}:\ \left(\frac{1}{\tau_k}\mathsf{M} + \mathsf{C}(\xi_{k-1})\right)\xi_k = \frac{1}{\tau_k}\mathbf{m} + \mathbf{q}(\xi_{k-1}), \quad k = 1,\ldots,r, \tag{12}$$

where M is the block-diagonal *mass matrix* (if B is orthonormal basis with respect L^2 scalar product then M is the identity matrix) given by

$$\mathsf{M} = \{M^{i,j}\}_{i,j=1}^{\mathrm{dof}}, \quad M^{i,j} = (\psi_j, \psi_i), \tag{13}$$

the matrix $\mathsf{C}(\cdot)$ is the *"flux" matrix* corresponding to form $\mathbf{c}_h(\cdot,\cdot,\cdot)$ at t_k defined by

$$\mathsf{C}(\xi_{k-1}) = \{C^{i,j}(\xi_{k-1})\}_{i,j=1}^{\mathrm{dof}}, \quad C^{i,j}(\xi_{k-1}) = \mathbf{c}_h\left(\mathbf{w}_{h,k-1},\ \psi_j,\ \psi_i\right), \tag{14}$$

$\mathbf{q} \in \boldsymbol{R}^{\mathrm{dof}}$ represents the right-hand-sides of (8), b) given by

$$\mathbf{q}(\xi_{k-1}) = \{q^i(\xi_{k-1})\}_{i=1}^{\mathrm{dof}}, \quad q^i(\xi_{k-1}) = \mathbf{c}_h^B\left(\mathbf{w}_{h,k-1},\ \psi_i\right) \tag{15}$$

and

$$\mathbf{m}_k = \{m_k^i\}_{i=1}^{\mathrm{dof}}, \quad m_k^i = -\left(\mathbf{w}_{h,k-1},\ \psi_i\right). \tag{16}$$

In virtue of the local character of basis B it is easy to observe that the matrix C have a block structure.

Let us still mention that series of numerical experiments show that the Frobenius norm of the diagonal blocks of C is slightly higher than the norm of its off-diagonal blocks (in the same block-row). Moreover, the norm of the diagonal blocks of C is approximately 10^3 times higher than the Frobenius norm of the corresponding blocks of M.

4.2 General Solution Strategy

In case, when we seek the steady state solution, problem (12) has to be replaced by the problem:

$$\text{find } \xi \in \boldsymbol{R}^{\mathrm{dof}}: \quad \mathsf{C}(\xi)\xi = \mathbf{q}(\xi). \tag{17}$$

However, problem (17) represents a system of strongly nonlinear algebraic equations whose direct solution is impossible. Then it is natural employ an iterative solver. The relation (17) offer to us to define a formal iterative process:

$$\begin{aligned} &\text{i)} \quad \text{initiate } \xi_0 \in \boldsymbol{R}^{\mathrm{dof}} \\ &\text{ii)} \quad \text{find } \xi_k \in \boldsymbol{R}^{\mathrm{dof}}: \quad \mathsf{C}(\xi_{k-1})\xi_k = \mathbf{q}(\xi_{k-1}), \quad k = 1, \ldots, \\ &\text{iii)} \quad \xi = \lim_{k\to\infty} \xi_k. \end{aligned} \tag{18}$$

However, numerical experiments show that this iterative process often fails which is caused by the fact that usually we start from an unphysical initial state (represented here by ξ_0) and then negative density or pressure often appear.

A usual way how to avoid this obstacle is a use of the unsteady formulation (12) which can be also considered as a relaxation of method (18). It means that step ii) in (18) is replaced by

$$\text{find } \xi_k \in \boldsymbol{R}^{\mathrm{dof}}: \underbrace{\left(\frac{1}{\tau_k}\mathsf{M} + \mathsf{C}(\xi_{k-1})\right)}_{=:\mathsf{A}_k(\xi_{k-1})} \xi_k = \underbrace{\frac{1}{\tau_k}\mathbf{m} + \mathbf{q}(\xi_{k-1})}_{=:\mathrm{d}_k(\xi_{k-1})}, \quad k = 1, \ldots, \tag{19}$$

where $\tau_k > 0$, $k = 1, \ldots$, can be considered as the size of the time step as well as the relaxation parameter. The relation (19) represents a sequence of systems of linear algebraic equations which has to be solved by a suitable solver. There arise two fundamental questions:

1. How to choose $\tau_k,\ k = 1,\ldots,?$
2. How to solve (19)?

It seems to be suitable to use an iterative solver for the solution of (19) since the solution from the old step $k-1$ can be used as the initial solution in the new step k. Moreover, it is sufficient to know the solution of (19) only approximately since we are interested only in the limit vector $\xi = \lim_{k\to\infty} \xi_k$. Hence, the iterative solver for the solution of (19) can be stop after not too high number of iteration. In our case, we employ the restarted GMRES method ([10]) with the *block diagonal preconditioning* (BDP). This approach is simple for an implementation, it is fast and requires a small amount of additional memory.

Based on the previous consideration, we propose the following general solution procedure:

Algorithm (A)

1. let $\xi_0 \longleftrightarrow \mathbf{w}_h^0$ be given
2. for $k = 1$ to r
   a. set τ_k
   b. from ξ_{k-1} evaluate $\mathsf{A}_k(\xi_{k-1})$, $\mathrm{d}_k(\xi_{k-1})$
   c. solve $\mathsf{A}_k(\xi_{k-1})\xi_k = \mathrm{d}_k(\xi_{k-1})$ by restarted GMRES with BDP by
      i. $\xi_k^0 := \xi_{k-1}$
      ii. $\xi_k^{l+1} := \mathsf{GMRES\_iter}(\xi_k^l),\ l = 1,\ldots,s_k$
      iii. $\xi_k := \xi_k^{s_k}$
3. $\xi := \xi_r$.

In the previous algorithm r denotes the total number of used time steps and $s_k,\ k = 1,\ldots,r$ the number of inner iterative loops of the GMRES solver for the time level t_k. These values have to be chosen on the base of suitable stopping criteria which are discussed in the following sections.

4.2.1 Steady-State Criterion

Within this section we discuss the steady-state stopping criterion, i.e., when to stop the global loops in the algorithm **(A)** for $k = 1,\ldots,r$. The usual steady-state criterion, often used for explicit time discretization, is

$$\left\| \frac{\partial \mathbf{w}_h}{\partial t} \right\| \approx \eta_k := \frac{1}{\tau_k} \|\mathbf{w}_h^k - \mathbf{w}_h^{k-1}\|_{L^2(\Omega)} = \frac{1}{\tau_k} \|\xi_k - \xi_{k-1}\|_{\ell^2} \le \mathrm{TOL}, \tag{20}$$

where TOL is a given tolerance. However, this criterion makes not good sense for the semi-implicit time discretization when very large time steps can be employed. Then there exists a limit value of τ_k when $\mathsf{A}_k \approx \mathsf{C}_k$ and $\mathrm{d}_k \approx \mathbf{q}_k$ are independent of τ_k (in the finite precision arithmetic) whereas (20) depends on τ_k. Then by a very

large choice of τ_k we can achieve very small left-hand side in (20) although we are far from the steady-state solution.

Therefore, in virtue of (17) we employ the following *steady-state residual criterion*

$$\mathrm{SSres}(k) := \|\mathsf{C}(\xi_k)\xi_k - \mathbf{q}(\xi_k)\|_{\ell^2} \leq \mathrm{TOL}, \tag{21}$$

which is independent of τ_k.

Another possibility when to stop the global loops in **(A)** follows from the physical background of the considered problem. Many often, we are interested in the so-called *aerodynamic coefficients* of the considered flow, namely coefficients of *drag* (c_D), *lift* (c_L) and *momentum* (c_M). Then the natural choice is stop global iterative loops when these coefficients achieve a given tolerance, e.g.,

$$\frac{\Delta c_x(k)}{|c_x(k)|} \leq \mathrm{tol}, \qquad \Delta c_x(k) := \max_{l=0.9k,\ldots,k} c_x(l) - \min_{l=0.9k,\ldots,k} c_x(l), \tag{22}$$

where tol is a given relative tolerance, subscript x takes the value D, L and M (drag, lift, momentum), $c_x(k)$ is the value of the corresponding aerodynamical coefficient at k^{th}-time step and the minimum and maximum in (22) are taken over last 10% of the number of time steps.

Whereas the tolerance TOL in the preconditioned residuum (21) has to be chosen empirically, the tolerance tol in (22) can be chosen only on the base of our accuracy requirements (without any previous numerical experiments), e.g., $\mathrm{tol} = 0.01$.

4.2.2 GMRES Stopping Criterion

Within this section we deal with the stopping criterion of the inner loop in **(A)**, i.e., when to step the GMRES iterative process at each time step $k = 1, \ldots, r$. It is clear that too weak criterion can decrease accuracy and on the other hand, too strong criterion decreases the efficiency. Usually, one uses residuum criterion

$$\|\mathsf{A}_k(\xi_{k-1})\xi_k - \mathrm{d}_k(\xi_{k-1})\| \leq \mathrm{TOL} \tag{23}$$

or the preconditioned residuum criterion

$$\|\mathsf{Q}\mathsf{A}_k(\xi_{k-1})\xi_k - \mathsf{Q}\mathrm{d}_k(\xi_{k-1})\| \leq \mathrm{TOL}, \tag{24}$$

where Q is the matrix of preconditioning and TOL is a given tolerance. However, there is a problem how to choose TOL since there is no indication from the theory.

Hence, inspired by the so-called inexact Newton method from [2] we propose the following stopping criterion for GMRES method:

$$\|\mathsf{A}_k(\xi_{k-1})\xi_k - \mathrm{d}_k(\xi_{k-1})\| \leq \delta_k \|\mathsf{A}_k(\xi_{k-1})\xi_{k-1} - \mathrm{d}_k(\xi_{k-1})\|, \tag{25}$$

where $\delta_k \in (0,1)$ is a given value, the left hand-side is the residuum and the term on the right hand side can be considered either as *consistency residuum* from the previous time step or *initial residuum* since the solution of the previous time step is taken as an initial solution on the next time step. Concerning δ_k, two choices were proposed and analysed in [8]. However, numerical experiments presented in Section 5 show that for our purposes parameters δ_k can be chosen very simply.

4.2.3 Choice of the Time Step

The choice of the time step $\tau_k,\ k = 1,\dots,r$ exhibits another important issue in the efficient solution of the steady-state solution. At the beginning of computation, it is necessary to choose τ_k small in order to avoid fails of computations caused by the unphysical initial condition. Moreover, when the solution is approaching to the steady-state, we are increasing the size of τ_k in order to accelerate the computational process. In other words we are decreasing the relaxation parameter (τ_k^{-1}). In [5] we proposed the *adaptive backward difference formulae* technique which adapts the size of the time step in order to keep the local discretization error under a given tolerance and to minimize a number of time step. However, numerical experiments show that the size of τ_k is very often underestimate when we seek steady state solutions, i.e., the time step can be chosen larger.

Therefore, we propose here a new rather *heuristic* adaptive choice of the time step according to the formula

$$\tau_1 := \frac{1}{2\Lambda_k}, \qquad \tau_{k+1} := \frac{1}{2\Lambda_k}\left(\frac{\eta_k}{\eta_1}\right)^{-\omega}, \quad k = 1,\dots,r-1, \tag{26}$$

where $\eta_k,\ k = 1,\dots,r$ is given by (20), $\omega > 0$ is a given constant usually chosen as $\omega = 3/2$ or $\omega = 2$ and

$$\Lambda_k = \max_{K\in\mathcal{T}_h} |K|^{-1} \max_{\Gamma\in\partial K} \max_{l=1,\dots,d+2} \lambda_l(\mathbf{w}_h^k|_\Gamma)|\Gamma| \tag{27}$$

where $\lambda_l(\mathbf{w}_h^k|_\Gamma)$ is the spectral ration of the matrix (2) evaluated on $\Gamma \in \partial K,\ K \in \mathcal{T}_h$. This means that at the first step, τ_1 is chosen in the same way as for an explicit time discretization with CFL $= 0.5$, see [9]. Moreover, τ_k is exponentially increasing when η_k is decreasing.

5 Numerical Examples

In the previous section we presented the new solution strategy for the solving the steady-state solutions of the Navier-Stokes equations. However, there are still some undefined parameters whose choice will be discussed in the following. We show that these choices are very robust. Finally, we show a 3D illustrative example.

5.1 Numerical Study of Inexact Newton Method

Within this section we numerically study two still open questions:

- Choice of δ_k in (25),
- Choice of the number of restarts in GMRES solver.

Finally, we present a comparison the new strategy with the former BDF-DGFE method from [4].

We deal with a viscous compressible flow around NACA 0012 profile with inlet Mach number $M_{\text{inlet}} = 0.5$, angle of attack $\alpha = 2^\circ$ and Reynolds number $Re = 5\,000$. We employ a triangular grid having 2 394 elements (see Figure 1) and a piecewise cubic polynomial approximation. The computational processes are stopped if condition (21) is valid with $\text{TOL} = 10^{-3}$ and condition (22) is valid with $\text{tol} = 10^{-2}$ for drag, lift and momentum coefficients.

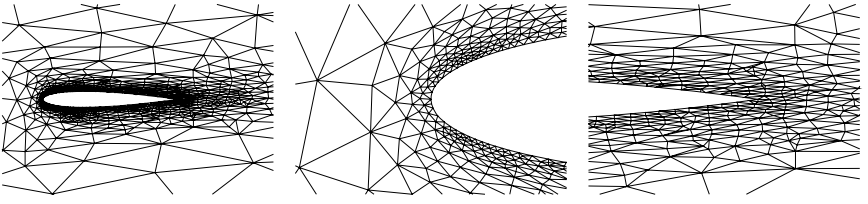


Fig. 1 The used triangular mesh around NACA0012 profile with details around leading and trailing edges

Figure 2 shows the dependence of SSres defined by (21) on the number of time steps and the computational time in second for $\delta = \delta_k = 0.9,\ 0.5,\ 0.1,\ 0.02,\ 0.005$, $k = 1, 2 \ldots$. We see that small values of δ increase the computational time whereas there is almost negligible difference in computational time for $\delta \in [0.1; 0.9]$. Hence, we can simply put, e.g., $\delta = 0.5$ and this value will be (almost) optimal.

Moreover, Figure 3 shows the dependence of SSres on the number of time steps and the computational time in second for different number of loops in GMRES method after which the GMRES is restarted (namely 20, 30, 40, 50, 60 loops). We observe that high number in inner loops within one restart is more efficient but the difference between 50 and 60 is again almost negligible. Hence, we use the restart after 50 loops in the following.

Furthermore, Table 1 shows a comparison of BDF-DGFE method presented in [4] with the new approach developed here, namely the number of time steps and computational time. The increase of efficiency (=decrease of computational time) is evident. This table also contains relative computational costs necessary for preparation and itself solution of linear algebra problems. For the new method, this ratio is equal almost to the optimal one 50 % : 50%.

Finally, Table 2 present the comparisons of relative computational costs necessary for preparation and itself solution of linear algebra problems carried out by P_1,

P_2 and P_3 polynomial approximations for the mesh from Figure 1 and a finner one. We simply observe that the ratios are close to the optimal one (50 % : 50%) and moreover, it is still better for finner grids (at least for P_1 and P_2) and higher degrees of polynomial approximations. Hence, our approach seems to be robust with respect to h and p.

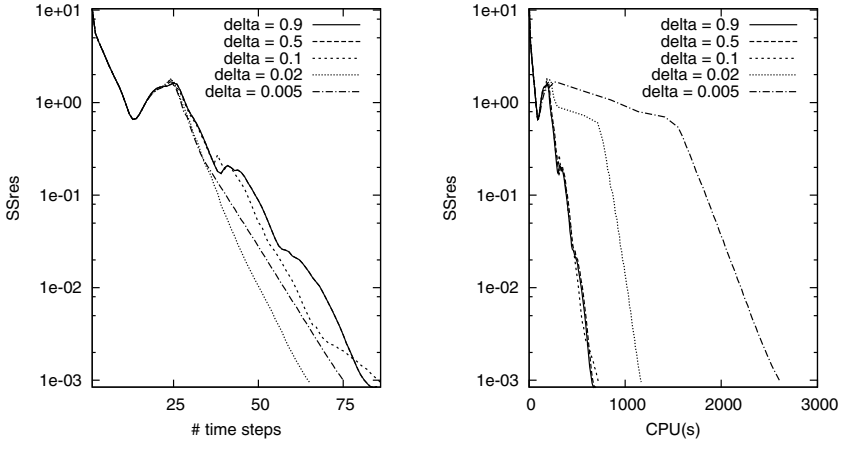


Fig. 2 Dependence of SSres on the number of time steps and the computational time for $\delta = 0.9,\ 0.5,\ 0.1,\ 0.02,\ 0.005$ in (21)

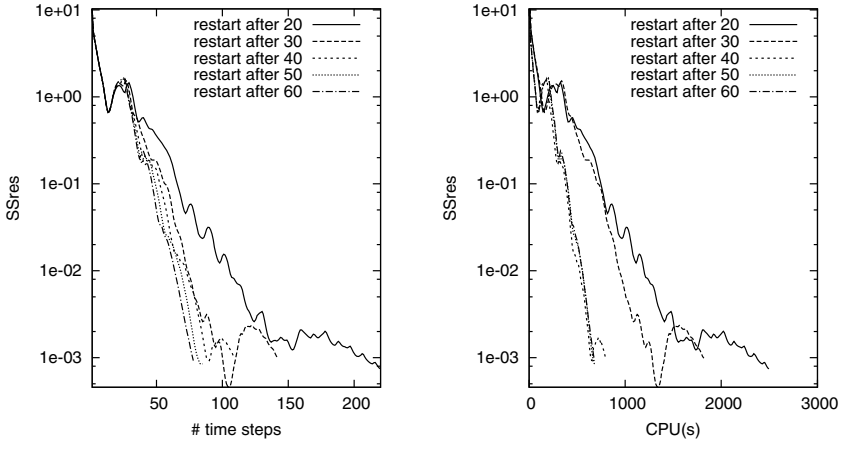


Fig. 3 Dependence of SSres on the number of time steps and the computational time for restart after 20,30,40,50,60 loops in GMRES method

Table 1 Comparison of BDF-DGFE method with inexact Newton, number of time steps, computational time and relative computational costs necessary for preparation and itself solution of linear algebra problems

method	# time steps	CPU (s)	preparing $\mathsf{A}_k, \mathrm{d}_k$	solving $\mathsf{A}_k\xi_k = \mathrm{d}_k$
BDF– DGFEM	273	10 774	12 % CPU	88 % CPU
inexact Newton	85	695	42 % CPU	58 % CPU

Table 2 Inexact Newton method, comparison of relative computational costs necessary for preparation and itself solution of linear algebra problems for two grids and different degree of polynomial approximations

P_k	$\#\mathcal{T}_h$	dof	preparing $\mathsf{A}_k, \mathrm{d}_k$	solving $\mathsf{A}_k\xi_k = \mathrm{d}_k$
P_1	2 394	28 728	31% CPU	69% CPU
P_1	4 214	50 568	33% CPU	67% CPU
P_2	2 394	57 456	36% CPU	64% CPU
P_2	4 214	101 136	37% CPU	63% CPU
P_3	2 394	95 760	42% CPU	58% CPU
P_3	4 214	168 560	41% CPU	59% CPU

5.2 3D Test Case

Finally, we show an illustrative 3D laminar viscous flow around the ONERA M6 wing with the inlet Mach number $M_{\text{inlet}} = 0.71$, angle of attack $\alpha = 3.06^\circ$ and the Reynolds number $Re = 5\,000$ which was solved within the project ADIGMA [1]. Figure 4 shows

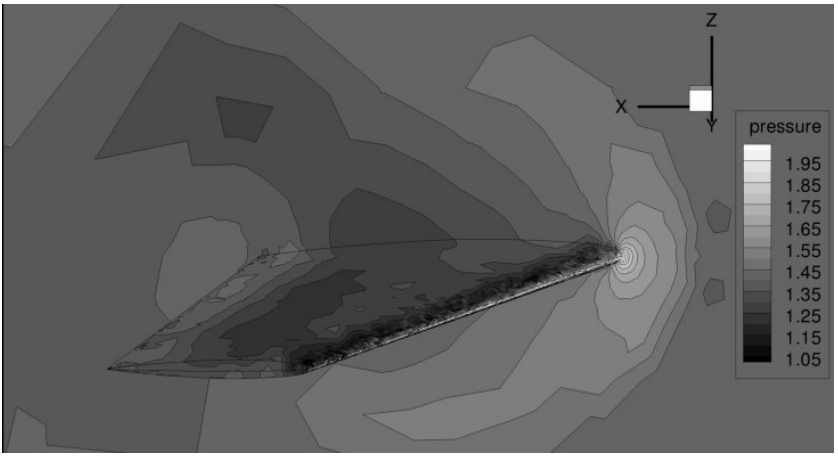


Fig. 4 ONERA M6 wing, distribution of the pressure

a distribution of the pressure around the profiles. In order to obtain better resolution an adaptive mesh refinement has to be employed.

6 Conclusion

We developed an efficient technique for the solution of steady state viscous compressible flows. The key feature is a week stopping criterion of the linear algebra systems arising from the semi-implicit time discretization. Numerical experiments show that solution of these systems requires approximately the same computational time as the setting of these systems itself. Moreover, this approach is robust with respect to h and p.

References

1. ADIGMA. Adaptive higher-order variational methods for aerodynamic applications in industry, Specific Targeted Research Project no. 30719 supported by European Commison, http://www.dlr.de/as/en/Desktopdefault.aspx/tabid-2035/2979_read-4582/
2. Dembo, R.S., Eisenstat, S.C., Steihaug, T.: Inexact newton methods. SIAM J. Numer. Anal. 19, 400–408 (1982)
3. Dolejší, V.: On the discontinuous Galerkin method for the numerical solution of the Navier–Stokes equations. Int. J. Numer. Methods Fluids 45, 1083–1106 (2004)
4. Dolejší, V.: Semi-implicit interior penalty discontinuous Galerkin methods for viscous compressible flows. Commun. Comput. Phys. 4(2), 231–274 (2008)
5. Dolejší, V., Kus, P.: Adaptive backward difference formula – discontinuous Galerkin finite element method for the solution of conservation laws. Int. J. Numer. Methods Eng. 73(12), 1739–1766 (2008)
6. Dolejší, V.: Discontinuous Galerkin method for the numerical simulation of unsteady compressible flow. WSEAS Transactions on Systems 5(5), 1083–1090 (2006)
7. Dolejší, V., Feistauer, M.: Semi-implicit discontinuous Galerkin finite element method for the numerical solution of inviscid compressible flow. J. Comput. Phys. 198(2), 727–746 (2004)
8. Eisenstat, S.C., Walker, H.F.: Choosing the forcing terms in inexact newton method. SIAM J. Sci. Comput. 17(1), 16–32 (1996)
9. Feistauer, M., Felcman, J., Straškraba, I.: Mathematical and Computational Methods for Compressible Flow. Oxford University Press, Oxford (2003)
10. Saad, Y., Schultz, M.H.: GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. 7, 856–869 (1986)
11. Vijayasundaram, G.: Transonic flow simulation using upstream centered scheme of Godunov type in finite elements. J. Comput. Phys. 63, 416–433 (1986)

This page intentionally left blank

Chapter 18
Multigrid Optimization for Space-Time Discontinuous Galerkin Discretizations of Advection Dominated Flows

S. Rhebergen, J.J.W. van der Vegt, and H. van der Ven

Abstract. The goal of this research is to optimize multigrid methods for higher order accurate space-time discontinuous Galerkin discretizations. The main analysis tool is discrete Fourier analysis of two- and three-level multigrid algorithms. This gives the spectral radius of the error transformation operator which predicts the asymptotic rate of convergence of the multigrid algorithm. In the optimization process we therefore choose to minimize the spectral radius of the error transformation operator. We specifically consider optimizing h-multigrid methods with explicit Runge-Kutta type smoothers for second and third order accurate space-time discontinuous Galerkin finite element discretizations of the 2D advection-diffusion equation. The optimized schemes are compared with current h-multigrid techniques employing Runge-Kutta type smoothers. Also, the efficiency of h-, p- and hp-multigrid methods for solving the Euler equations of gas dynamics with a higher order accurate space-time DG method is investigated.

1 Introduction

Space-time discontinuous Galerkin (DG) discretizations of time-dependent partial differential equations result in a system of (non)-linear algebraic equations which can be solved efficiently with multigrid methods. In this paper we will discuss the optimization of multigrid techniques for higher order accurate space-time DG discretizations describing advection dominated flows. This research is a continuation of

S. Rhebergen · J.J.W. van der Vegt
University of Twente, Department of Applied Mathematics, P.O. Box 217, 7500 AE, Enschede, The Netherlands
e-mail: {s.rhebergen,j.j.w.vandervegt}@math.utwente.nl

H. van der Ven
National Aerospace Laboratory NLR, P.O. Box 90502, 1006 BM, Amsterdam, The Netherlands
e-mail: venvd@nlr.nl

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 257–269, 2010.
springerlink.com

[3, 7] where we presented a multigrid algorithm in combination with a pseudo-time integration method for second order accurate space-time DG discretizations of the compressible Euler and Navier-Stokes equations. The main benefits of this multigrid algorithm are that no large global linear system needs to be solved and, through the use of Runge-Kutta type smoothers, the locality of the DG discretization is preserved. The algorithm is easy to implement and parallelize, even on locally refined meshes, and insensitive to initial conditions. For higher order accurate space-time DG discretizations the multigrid performance was, however, not satisfactory. The objective of this paper is to discuss improvements in the computational performance of space-time DG discretizations when higher order polynomial basis functions are used. The main tool to analyze the multigrid performance is three-level discrete Fourier analysis. This analysis tool is used to optimize the multigrid performance by minimizing the spectral radius of the multigrid error transformation operator. In particular, the focus will be on searching for better coefficients in the multigrid smoothing operator. More detailed information on the multigrid algorithms and the analysis techniques used in this paper can be found in e.g. [1, 6, 10, 11].

The outline of this paper is as follows. After a brief introduction in Section 2 on the multigrid error transformation operator, a summary of the discrete Fourier analysis of the multigrid algorithm will be given in Section 3. Next, we discuss the optimization of the multigrid algorithm in Section 4. Results of the optimization process will be given in Section 5 as well as a comparison in efficiency between h-, p- and hp-multigrid methods. Finally, conclusions are drawn in Section 6.

2 Multigrid Error Transformation Operator

The main goal of the multigrid algorithm is to iteratively solve in an efficient way a system of (non)-linear algebraic equations $L_h v_h = f_h$ on a mesh $\mathscr{M}_h$, with L_h a linear or non-linear discretization operator and f_h a given righthand side. In the h-multigrid method we use a finite sequence N_c of increasingly coarser meshes $\mathscr{M}_{nh}$, $n \in \{1, \cdots, N_c\}$ to generate approximations to the original problem. In addition, the data on the different meshes are connected with restriction operators $R^{mh}_{nh} : \mathscr{M}_{nh} \rightarrow \mathscr{M}_{mh}$ and prolongation operators $P^{nh}_{mh} : \mathscr{M}_{mh} \rightarrow \mathscr{M}_{nh}$, with $1 \leq n < m \leq N_c$. On these meshes a set of auxiliary problems is solved $\mathscr{M}_{nh}$, $1 < n \leq N_c$, namely $L_{nh} v_{nh} = f_{nh}$, in order to accelerate convergence. For non-linear problems we use the Full Approximation Scheme (FAS), see e.g. [6], but in the analysis of the multigrid performance we only consider linear problems.

In order to understand the performance of the multigrid algorithm we need to consider the multigrid error transformation operator. Given an initial error e^A_h, the error e^D_h after one full multigrid cycle with three grid levels is given by the relation

$$e^D_h = M^{3g}_h e^A_h$$

with

$$M^{3g}_h = S^{\nu_2}_h (I_h - P^h_{2h}(I_{2h} - M^{\gamma_c}_{2h}) L^{-1}_{2h} R^{2h}_h L_h) S^{\nu_1}_h \tag{1}$$

and

$$M_{2h} = S_{2h}^{\nu_4}(I_{2h} - P_{4h}^{2h}L_{4h}^{-1}R_{2h}^{4h}L_{2h})S_{2h}^{\nu_3}. \tag{2}$$

Here, S_{nh} and I_{nh} are, respectively, the smoothing and identity operator on the mesh $\mathscr{M}_{nh}$, ν_i, $i = 1,\cdots,4$, the number of pre- and post-smoothing iterations and γ_c the cycle index. In the multigrid analysis and computations we will also consider the effect of solving the algebraic system on the coarsest mesh approximately using ν_c smoother iterations instead of using an exact inverse. Next to h-multigrid also p-multigrid methods are possible in which on a single mesh coarser approximations are obtained by using lower order discretizations. Of course, combinations of both techniques are possible resulting in hp-multigrid methods.

3 Three-Level Multigrid Analysis

3.1 Discrete Fourier Analysis

Consider the infinite mesh G_h, which is defined as

$$G_h := \left\{x = (x_1,x_2) = (k_1h_1,k_2h_2) \mid k \in \mathbb{Z}^2, h \in \left(\mathbb{R}^+\right)^2\right\}.$$

On G_h we define for $v_h : G_h \to \mathbb{C}$ the norm

$$\|v_h\|_{G_h}^2 := \lim_{N\to\infty} \frac{1}{4N^2} \sum_{|k|\le m} |v_h(kh)|^2,$$

where $|k| = \max\{|k_1|,|k_2|\}$. In the theoretical analysis we only consider linear problems, where the linear systems on the various meshes are described using stencil notation

$$L_{nh}v_{nh}(x) = \sum_{k\in J_n} l_{n,k}v_{nh}(x+kh), \qquad x \in G_{nh}, \tag{3}$$

with stencil coefficients $l_{n,k} \in \mathbb{R}^{m_k\times m_k}$ and finite index sets $J_n \subset \mathbb{Z}^2$ describing the stencil. The restriction operators R_{nh}^{mh}, prolongation operators P_{mh}^{nh} and smoothing operators S_{nh} with $1 \le n < m \le N_c$ are also expressed using stencil notation, see e.g. [6, 10, 11].

On the infinite mesh G_h, we define for $x \in G_h$ the continuous Fourier modes with frequency $\theta = (\theta_1,\theta_2) \in \mathbb{R}^2$ as $\phi_h(\theta,x) := e^{i\theta\cdot x/h}$ with $\theta\cdot x/h := \theta_1x_1/h_1 + \theta_2x_2/h_2$, $h \in (\mathbb{R}^+)^2$ and $i = \sqrt{-1}$. We also define the space of bounded infinite grid functions by $\mathscr{F}(G_h) := \left\{v_h \,|\, v_h : G_h \to \mathbb{C} \text{ with } \|v_h\|_{G_h} < \infty\right\}$. For each $v_h \in \mathscr{F}(G_h)$ there exists a Fourier transformation, hence $v_h(x)$ can be written as a linear combination of Fourier components

$$v_h(x) = \int_{|\theta|\le\pi} \widehat{v_h}(\theta)e^{i\theta\cdot x/h}d\theta, \qquad x \in G_h, \tag{4}$$

with $x/h := (x_1/h_1, x_2/h_2) = j \in \mathbb{Z}^2$, and inverse transformation

$$\widehat{v_h}(\theta) = \frac{1}{4\pi^2} \sum_{x \in G_h} v_h(x) e^{-i\theta \cdot x/h}, \qquad -\pi \leq \theta_j < \pi,$$

see e.g. [1]. Due to aliasing, Fourier components with $|\hat{\theta}| := \max\{|\theta_1|, |\theta_2|\} \geq \pi$ are not visible on G_h. These modes coincide with $e^{i\theta \cdot x/h}$, where $\theta = \hat{\theta} \pmod{2\pi}$. Hence, the Fourier space $\mathscr{F} := \text{span}\left\{ e^{i\theta \cdot x/h} \mid \theta \in \Theta = [-\pi, \pi)^2, x \in G_h \right\}$ contains any bounded infinite grid function.

3.2 Three-Grid Fourier Analysis

For the three-grid Fourier analysis we define the Fourier harmonics $\mathscr{F}_{4h}(\theta)$ as

$$\begin{aligned}
&\mathscr{F}_{4h}(\theta) := \text{span}\left\{ \phi_h(\theta_\beta^\alpha, x) \mid \alpha \in \alpha_2, \beta \in \beta_2 \right\}, \quad \text{where} \\
&\theta = \theta_0^0 \in \Theta_{4h} := [-\pi/4, \pi/4)^2, \\
&\theta_\beta = \theta_0^0 - (\bar{\beta}_1 \,\text{sign}\,(\theta_1), \bar{\beta}_2 \,\text{sign}\,(\theta_2))\pi, \\
&\theta_\beta^\alpha := \theta_\beta - (\bar{\alpha}_1 \,\text{sign}\,((\theta_1)_\beta), \bar{\alpha}_2 \,\text{sign}\,((\theta_2)_\beta))\pi, \\
&\alpha_2 := \{\alpha = (\bar{\alpha}_1, \bar{\alpha}_2) \mid \bar{\alpha}_i \in \{0,1\}, i = 1,2\} \\
&\beta_2 := \{\beta = (\bar{\beta}_1, \bar{\beta}_2) \mid \bar{\beta}_i \in \{0, \frac{1}{2}\}, i = 1,2\}.
\end{aligned}$$

Note that we have 16 coupled Fourier harmonics, all related to θ_{00}^{00}. In the transition from G_{2h} to G_{4h} the modes $\theta_\beta = \theta_\beta^0$ are not visible due to aliasing.

The error e_h^D after one iteration of a three-grid multigrid cycle is determined by $e_h^D = M_h^{3g} e_h^A$, with e_h^A the initial error and M_h^{3g} the three-level multigrid error transformation operator defined by (1).

The properties of the error transformation operator can be investigated using discrete Fourier analysis. For this purpose we introduce the following matrices

$$\widehat{L}_h^{2g}(\theta_\beta) = \text{diag}\,(\widehat{L_h}(\theta_\beta^{00}), \widehat{L_h}(\theta_\beta^{11}), \widehat{L_h}(\theta_\beta^{10}), \widehat{L_h}(\theta_\beta^{01})) \in \mathbb{C}^{4m \times 4m} \tag{5}$$

$$\widehat{S}_h^{2g}(\theta_\beta) = \text{diag}\,(\widehat{S_h}(\theta_\beta^{00}), \widehat{S_h}(\theta_\beta^{11}), \widehat{S_h}(\theta_\beta^{10}), \widehat{S_h}(\theta_\beta^{01})) \in \mathbb{C}^{4m \times 4m} \tag{6}$$

$$\widehat{R}_h^{2g}(\theta_\beta) = (\widehat{R_h^{2h}}(\theta_\beta^{00}), \widehat{R_h^{2h}}(\theta_\beta^{11}), \widehat{R_h^{2h}}(\theta_\beta^{10}), \widehat{R_h^{2h}}(\theta_\beta^{01})) \in \mathbb{C}^{m \times 4m} \tag{7}$$

$$\widehat{P}_h^{2g}(\theta_\beta) = (\widehat{P_{2h}^h}(\theta_\beta^{00}), \widehat{P_{2h}^h}(\theta_\beta^{11}), \widehat{P_{2h}^h}(\theta_\beta^{10}), \widehat{P_{2h}^h}(\theta_\beta^{01}))^T \in \mathbb{C}^{4m \times m} \tag{8}$$

where diag refers to a diagonal matrix consisting of $m \times m$ blocks with $m \in \mathbb{N}$. The Fourier symbol of the linear operator L_{nh} is equal to $\widehat{L_{nh}}(\theta) = \sum_{k \in J_n} l_{n,k} e^{i\theta \cdot k}$. Similar expressions can be derived for the Fourier symbols of the restriction operator $\widehat{R_{nh}^{mh}}(\theta)$, the prolongation operator $\widehat{P_{mh}^{nh}}(\theta)$ and the smoothing operator $\widehat{S_{nh}}(\theta)$ on

the various mesh levels. For more details, see e.g. [1, 6, 11]. We also introduce the matrices

$$\widehat{L}_h^{3g}(\theta) = \text{bdiag}\left(\widehat{L}_h^{2g}(\theta_{00}), \widehat{L}_h^{2g}(\theta_{\frac{1}{2}\frac{1}{2}}), \widehat{L}_h^{2g}(\theta_{\frac{1}{2}0}), \widehat{L}_h^{2g}(\theta_{0\frac{1}{2}})\right) \in \mathbb{C}^{16m\times 16m}$$
$$\widehat{S}_h^{3g}(\theta) = \text{bdiag}\left(\widehat{S}_h^{2g}(\theta_{00}), \widehat{S}_h^{2g}(\theta_{\frac{1}{2}\frac{1}{2}}), \widehat{S}_h^{2g}(\theta_{\frac{1}{2}0}), \widehat{S}_h^{2g}(\theta_{0\frac{1}{2}})\right) \in \mathbb{C}^{16m\times 16m}$$
$$\widehat{R}_h^{3g}(\theta) = \text{bdiag}\left(\widehat{R}_h^{2g}(\theta_{00}), \widehat{R}_h^{2g}(\theta_{\frac{1}{2}\frac{1}{2}}), \widehat{R}_h^{2g}(\theta_{\frac{1}{2}0}), \widehat{R}_h^{2g}(\theta_{0\frac{1}{2}})\right) \in \mathbb{C}^{4m\times 16m}$$
$$\widehat{P}_h^{3g}(\theta) = \text{bdiag}\left(\widehat{P}_h^{2g}(\theta_{00}), \widehat{P}_h^{2g}(\theta_{\frac{1}{2}\frac{1}{2}}), \widehat{P}_h^{2g}(\theta_{\frac{1}{2}0}), \widehat{P}_h^{2g}(\theta_{0\frac{1}{2}})\right) \in \mathbb{C}^{16m\times 4m}$$
$$\widehat{Q}_h^{3g}(\theta) = \text{bdiag}\left(\widehat{L_{2h}^{-1}}(2\theta_{00}), \widehat{L_{2h}^{-1}}(2\theta_{\frac{1}{2}\frac{1}{2}}), \widehat{L_{2h}^{-1}}(2\theta_{\frac{1}{2}0}), \widehat{L_{2h}^{-1}}(2\theta_{0\frac{1}{2}})\right) \in \mathbb{C}^{4m\times 4m}.$$

The discrete Fourier transform of the error transformation operator for a three-level multigrid cycle $\widehat{M}_h^{3g}(\theta) \in \mathbb{C}^{16m\times 16m}$ then is equal to [11]

$$\widehat{M}_h^{3g}(\theta) = \left(\widehat{S}_h^{3g}(\theta)\right)^{\nu_2}\left(I^{3g} - \widehat{P}_h^{3g}(\theta)\widehat{U}^{3g}(\theta;\gamma_c)\widehat{Q}_h^{3g}(\theta)\widehat{R}_h^{3g}(\theta)\widehat{L}_h^{3g}(\theta)\right)\left(\widehat{S}_h^{3g}(\theta)\right)^{\nu_1} \tag{9}$$

with I^{3g} the $16m \times 16m$ identity matrix and $\theta \in \Theta_{4h} \setminus \Psi_{3g}$, where Ψ_{3g} is defined as $\Psi_{3g} := \left\{\theta \in \Theta_{4h} \mid \widehat{L_{4h}}(4\theta_0^0) = 0 \text{ or } \widehat{L_{2h}}(2\theta_\beta^0) = 0 \text{ or } \widehat{L_h}(\theta_\beta^\alpha) = 0\right\}$. We still need to obtain an explicit expression for $\widehat{U}^{3g}(\theta;\gamma_c) \in \mathbb{C}^{4m\times 4m}$. On the mesh G_{2h} the modes θ_β^α reduce after the restriction operator to modes $2\theta_\beta^0$, hence using the result of a two-level analysis the coarse grid error transformation operator is equal to

$$\widehat{M}_{2h}^{2g}(2\theta_\beta) = \left(\widehat{S}_{2h}^{2g}(2\theta_\beta)\right)^{\nu_4}\left(I^{2g} - \widehat{P}_{2h}^{2g}(2\theta_\beta)\widehat{L}_{4h}^{-1}(4\theta_0^0)\widehat{R}_{2h}^{2g}(2\theta_\beta)\widehat{L}_{2h}^{2g}(2\theta_\beta)\right)\left(\widehat{S}_{2h}^{2g}(2\theta_\beta)\right)^{\nu_3},$$

with I^{2g} the $4m \times 4m$ identity matrix and $\theta_\beta \in \Theta_{2h} := [-\pi/4, \pi/4)^2 \setminus \Psi_{2g}$, where Ψ_{2g} is defined as $\Psi_{2g} := \left\{\theta \in [-\pi/4, \pi/4)^2 \mid \widehat{L_{4h}}(4\theta_0^0) = 0 \text{ or } \widehat{L_{2h}}(2\theta_\beta^0) = 0\right\}$. The matrices $\widehat{L}_{2h}^{2g}$, $\widehat{S}_{2h}^{2g}$, $\widehat{R}_{2h}^{2g}$ and $\widehat{P}_{2h}^{2g}$ are given by (5)-(8), respectively, with h replaced by $2h$. The matrix $\widehat{U}^{3g}(\theta;\gamma_c)$ then is equal to

$$\widehat{U}^{3g}(\theta;\gamma_c) = I^{2g} - \left(\widehat{M}_{2h}^{2g}(2\theta_\beta)\right)^{\gamma_c}.$$

The spectral radius of the error transformation operator gives a prediction of the asymptotic rate of convergence of the multigrid method. This asymptotic convergence is expressed in terms of the asymptotic convergence factor per cycle, which is equal to

$$\mu = \sup_{\theta\in\Theta_{3g}\setminus\Psi_{3g}} \rho\left(\widehat{M}^{3g}(\theta)\right), \tag{10}$$

with ρ is the spectral radius. A requirement for convergence of the multigrid algorithm is that the spectral radius satisfies the condition $\mu < 1$. By minimizing the spectral radius of the three-level multigrid error transformation operator (9), we obtain optimized multigrid algorithms.

4 Optimizing Multigrid for Space-Time DG Discretizations

The theory of the previous sections holds for general linear discretizations and smoothing operators, but in this paper we are specifically interested in designing optimized multigrid methods for higher order accurate space-time DG discretizations. For the optimization, we will consider the 2D advection-diffusion equation as model problem

$$\partial_t u + \mathbf{a}\cdot\nabla u - \nabla\cdot(\bar{\bar{A}}\nabla u) = 0, \quad x \in \Omega \subset \mathbb{R}^2,\ t \in \mathbb{R}^+, \tag{11}$$

where we assume that the advection velocity $\mathbf{a} \in \mathbb{R}^2$ and diffusion matrix $\bar{\bar{A}} \in (\mathbb{R}^+)^2$ are constant, with $\bar{\bar{A}}_{11} = \nu_x$, $\bar{\bar{A}}_{22} = \nu_y$ and $\bar{\bar{A}}_{12} = \bar{\bar{A}}_{21} = 0$. We do not discuss the details of the space-time DG discretization for the advection-diffusion equation, but refer to [3, 5] for more details. In the multigrid optimization we consider a uniform space-time mesh with elements $\Delta t \times \Delta x \times \Delta y$ and periodic boundary conditions. The discretization depends on the following dimensionless numbers:

$$CFL = \frac{a\Delta t}{h}, \quad Re_x = \frac{a(\Delta x)^2}{\nu_x h}, \quad Re_y = \frac{a(\Delta y)^2}{\nu_y h}, \quad AR = \frac{\Delta y}{\Delta x},$$

in which $h = \Delta x\sqrt{1+AR^2}$ and $a = \sqrt{a_x^2 + a_y^2}$. Furthermore, we introduce the flow angle γ^{flow} with respect to the x-axis so that $a_x = \cos(\gamma^{flow})a$ and $a_y = \sin(\gamma^{flow})a$.

4.1 Pseudo-time Integration and Runge-Kutta Methods

The system of algebraic equations resulting from the space-time DG discretization of the 2D advection-diffusion equation can be represented as

$$\mathscr{L}(\hat{u}^n; \hat{u}^{n-1}) = 0, \tag{12}$$

with $\hat{u}^n$ the expansion coefficients of a polynomial approximation of u and n refers to the time index. To solve the system of coupled equations for the expansion coefficients $\hat{u}^n$ in (12), a pseudo time derivative is added to the system [7]:

$$\Delta x\Delta y\frac{\partial \hat{u}^*}{\partial \tau} = -\frac{1}{\Delta t}\mathscr{L}(\hat{u}^*; \hat{u}^{n-1}), \tag{13}$$

which is integrated to steady-state in pseudo-time. At steady state, $\hat{u}^n = \hat{u}^*$. For the pseudo-time integration we introduce the dimensionless number $\lambda = \Delta\tau/\Delta t$ and use the pseudo-time CFL number, defined as $CFL^\tau = \lambda CFL$. To solve (13) we consider N-stage Runge-Kutta methods. For notational purposes, we set $\mathscr{L}(\hat{V}^*; u^{n-1}) = \mathscr{L}(\hat{V}^*)$. Initialize $\hat{V}^0 = \hat{u}^{n-1}$. Then, an N-stage Runge-Kutta scheme is given by:

$$(1+\beta_j\lambda I)\hat{V}^j = \hat{V}^0 - \lambda\left(\sum_{l=1}^{j}\alpha_{j+1,l}\mathscr{L}(\hat{V}^{l-1})/(\Delta x\Delta y)\right) + \lambda\beta_j\hat{V}^{j-1}, \quad j=1,...,N,$$

with $\hat{u}^* = \hat{V}^N$. We see that there are a number of free parameters in the Runge-Kutta smoother. The smoother is therefore a good candidate for optimization. We will minimize the spectral radius (10) by optimizing the parameters α and β. In this paper only 5-stage Runge-Kutta schemes are considered for which we require that they are second order accurate in pseudo-time. This requirement gives constraints on the α coefficients. The β coefficients serve as the Melson correction to improve stability for small values of $\lambda \cong 1$, see Melson et al. [4].

4.2 Optimization Results

We now provide some examples of the optimization of the Runge-Kutta (RK) smoothers for multigrid. We distinguish between diagonal RK schemes (dRK5) and full RK schemes (fRK5) in which all coefficients $\alpha_{j+1,l}$, with $1 \le l \le j \le N$, are non-zero. We present optimized RK coefficients for the second ($p=1$) and third ($p=2$) order accurate space-time DG discretizations of the 2D advection-diffusion equation. For this we use the optimization procedures `fminsearch` and `fmincon`, available in Matlab. As constraint in the `fmincon` procedure, we require that both the spectral radius of the smoother and the three-level multigrid error transformation operator are less than 1. The optimization was performed for advection dominated steady flows in which we fix the Reynolds numbers $Re_x = Re_y = 100$ and the *CFL* number as $CFL = 100$. We also set the flow angle $\gamma^{flow} = \pi/4$, the aspect ratio $AR = 1$ and the number of pre- and post-smoothing steps $\nu_1 = \nu_2 = 1$. On the coarsest grid, we use four smoother steps instead of an exact inverse. Furthermore, $\gamma = 1$. As initial guess in the optimization procedure, we use the EXI RK method [7] for the optimized dRK5 scheme. We then use the dRK5 scheme as initial guess to obtain the fRK5 scheme. The optimized coefficients and spectral radii of the smoother $\rho^{\mathscr{S}}$ and the 3-level multigrid operator ρ^{MG} are given in Table 1. As a comparison, we also give the spectral radius of the 3-level multigrid operator with EXI-RK smoother, ρ^{EXI-MG} when using the given parameters. We see that for these parameters the multigrid algorithm with the EXI smoother is very unstable, while good convergence can be achieved with our optimized schemes.

5 Testing Multigrid Performance

In this section we test the multigrid performance. We start in Section 5.1 by comparing the optimized *h*-multigrid algorithms of the previous sections to the original EXI-EXV *h*-multigrid method [3]. For this we consider the 2D advection-diffusion equation. In Section 5.2 we consider a more complex test case in which we solve the Euler equations for inviscid flow over an NACA0012 airfoil. We will compare the performance of *h*-multigrid with *p*- and *hp*-multigrid.

Table 1 Optimized coefficients for the dRK5 and fRK5 smoothers for 3-level multigrid for steady flows

	dRK5 $p=1$	fRK5 $p=1$	dRK5 $p=2$	fRK5 $p=2$
α_{21}	0.05768995298	0.0578331573	0.04865009589	0.04877436325
α_{31}	-	-0.0002051554736	-	-0.0002188348438
α_{32}	0.1405960888	0.1403808301	0.130316854	0.1300906122
α_{41}	-	0.0003953470071	-	2.608884832e-05
α_{42}	-	-0.001195029164	-	2.444376496e-05
α_{43}	0.267958213	0.2681810517	0.2729621396	0.2734805705
α_{51}	-	0.0001441249202	-	-0.001250385487
α_{52}	-	-0.0002608610327	-	-0.0007838720635
α_{53}	-	-0.0003368070181	-	-0.0004890887712
α_{54}	0.5	0.8473374098	0.5	4.412139367
α_{61}	-	0.4115573097	-	0.8097217358
α_{62}	-	-0.003144851878	-	0.08435089009
α_{63}	-	-0.0001096455683	-	-0.01986799007
α_{64}	-	0.001555741114	-	0.01359815476
α_{65}	1.0	0.5901414466	1.0	0.1121972094
β_1	0.05768995298	0.04887040625	0.04865009589	0.5551936269
β_2	0.1405960888	0.1274785795	0.130316854	0.1333199239
β_3	0.267958213	0.2287556298	0.2729621396	-1.332263675
β_4	0.5	0.9547064029	0.5	-3.649588578
β_5	1.0	2.52621971	1.0	0.46771792
CFL^{τ}	0.8	0.8	0.4	0.4
$\rho^{\mathscr{S}}$	0.98812	0.98914	0.98974	0.9896
ρ^{MG}	0.89151	0.81762	0.90049	0.89903
ρ^{EXI-MG}	167.06	-	124.02	-

5.1 The 2D Advection-Diffusion Equation

In order to demonstrate the performance of the optimized algorithms we consider (11) on $\Omega = (0,1)^2$ with initial condition $u(x,y,0) = 1 - \frac{1}{2}(x+y)$ and boundary condition $u(x,y,t) = g(x,y)$. Here $g(x,y)$ equals at the domain boundary the exact steady state solution of (11) given by:

$$u(x,y) = \frac{1}{2}\left(\frac{\exp(a_1/\nu_x) - \exp(a_1 x/\nu_x)}{\exp(a_1/\nu_x) - 1} + \frac{\exp(a_2/\nu_y) - \exp(a_2 y/\nu_y)}{\exp(a_2/\nu_y) - 1}\right).$$

In the discretization we use a Shishkin mesh [3] which is suitable for dealing with boundary layers. The parameters in the test cases are the following: we consider a mesh with 32×32 elements, one physical time step, with $\Delta t = 100$, $a = \sqrt{2}$, $\nu_x = \nu_y = 0.01$ and a flow angle $\gamma^{flow} = \pi/4$. For the optimized RK schemes, we used a local pseudo-time scaling to deal with viscous flows [9]. For the multigrid

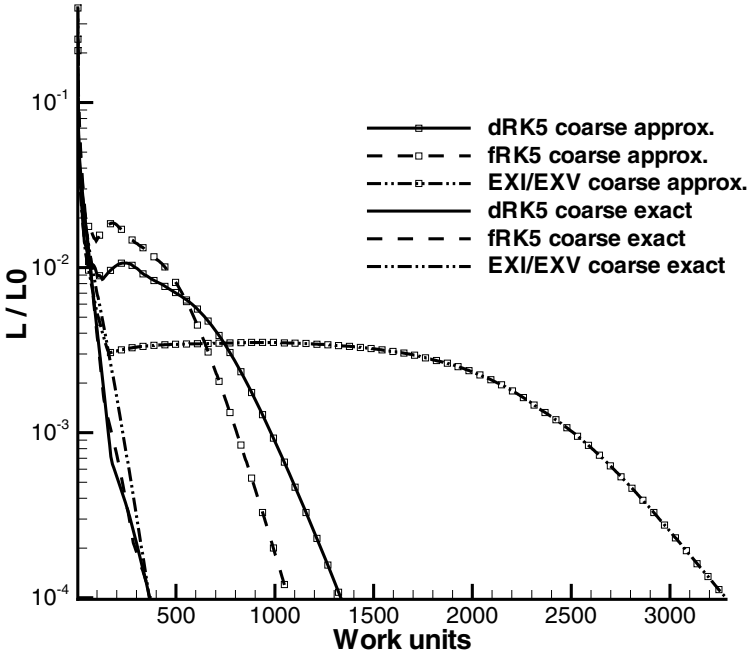


Fig. 1 Convergence results of second order space-time DG for three level multigrid algorithms with different Runge-Kutta smoothers. (dRK5, fRK5 and the EXI-EXV scheme [2], exact and approximate solution of equations on coarsest mesh).

computations we use $\nu_i = 1,\ i = 1,2,3,4$ and $\gamma = 1$. On the coarsest mesh we investigate the effect of using $\nu_C = 4$ smoother iterations or solving the discrete system exactly.

In Figures 1 and 2, we show the convergence results of the different smoothers for 3-level multigrid. We see that in all cases a big improvement is obtained with the optimized Runge-Kutta smoothers over the original EXI-EXV smoother. For a second order accurate space-time DG discretization the number of multigrid cycles to obtain 4 orders of reduction in the residual is reduced from 3283 to 371. For the third order accurate DG discretization the number of multigrid cycles reduces from 21254 to 184. Furthermore, comparing dRK5 with fRK5, we see that the differences for a second order accurate space-time DG discretization is negligible. For a third order accurate space-time DG discretization this difference is, however, significant. Using more Runge-Kutta coefficients enlarges the possibilities to optimize the smoother.

The effect of solving the equations on the coarsest mesh with high accuracy is very large. Without this the multigrid convergence significantly slows down after a rapid initial decrease of the residual. In particular, for nonlinear problems it is

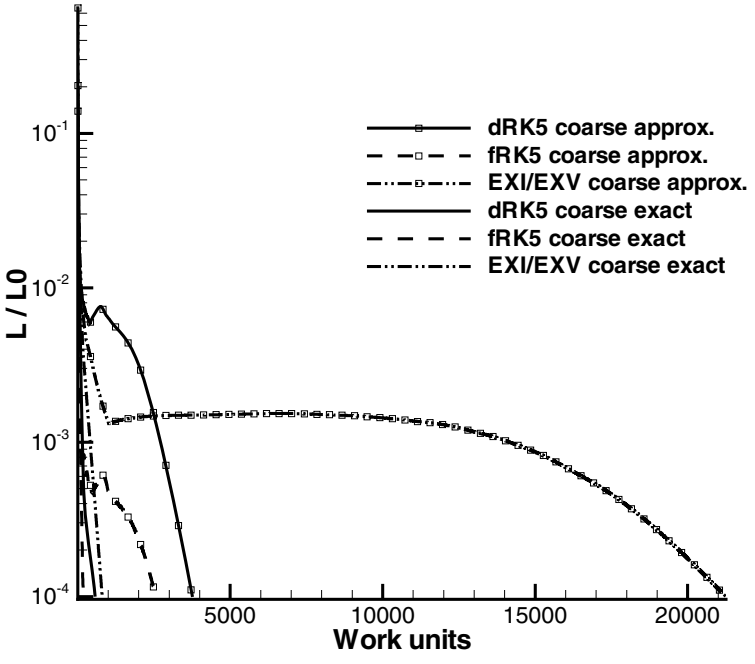


Fig. 2 Convergence results of third order space-time DG for three level multigrid algorithms with different Runge-Kutta smoothers. (dRK5, fRK5 and the EXI-EXV scheme [2], exact and approximate solution of equations on coarsest mesh).

tempting to solve the algebraic system on the coarsest mesh only approximately, because otherwise a global Newton solver is required. The effect of accurately solving the algebraic equations for the linear advection-diffusion equation on the coarsest mesh is, however, non-negligible.

5.2 The Euler Equations

We now compare the performance of an h-multigrid method with p- and hp-multigrid. Since the difference between EXI and the optimized RK smoothers for the Euler equations is small we will only show the EXI results. As test case we consider 2D steady subsonic flow around a NACA0012 airfoil with an angle of attack of $\alpha = 2^\circ$ and far-field Mach number $Ma = 0.5$ (MTC1 test case). Since this test case is a steady-state flow problem, we consider a space-time DG discretization which is only first-order accurate in time but third-order accurate in space. The grid around the airfoil has 448×64 elements.

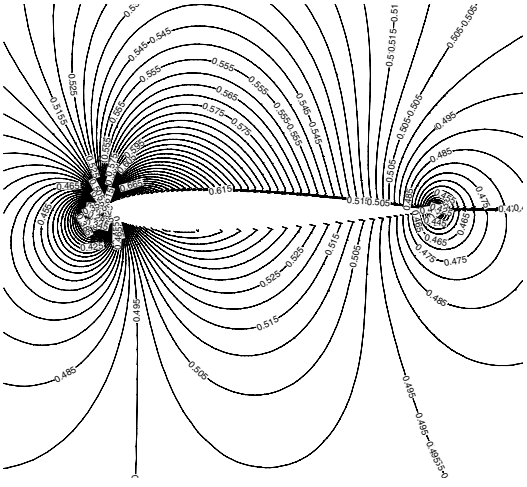


Fig. 3 Mach contours of inviscid flow around an NACA0012 airfoil ($\alpha = 2^\circ, Ma = 0.5$)

For single-grid, p- and hp-multigrid computations we used a pseudo-time CFL number of $CFL^\tau = 1.6$, while for h-multigrid $CFL^\tau = 0.8$. Larger pseudo-time CFL numbers for h-multigrid resulted in unstable calculations. For the p-multigrid method we solve the lowest order problem approximately taking $\nu_C = 20$. For the h- and hp-multigrid methods we solve the coarsest grid problem approximately, also taking $\nu_C = 20$. Furthermore, for the h-multigrid method, we also solve the coarse grid problem exactly using a matrix-free Newton method. In all cases, 5 pre- and post-smoothing steps were taken on each multigrid level. The Mach contours are given in Figure 3 while the convergence history plot is given in Figure 4.

We see that h-multigrid performs the worst while p- and hp-multigrid converge six orders in approximately the same amount of work units. We, however, had to take a twice as small CFL^τ number in the h-multigrid calculation compared to the other calculations. Furthermore, we see that after the high-frequency error modes have been smoothed, h-multigrid efficiency quickly deteriorates. A possible reason for this could be that the coarse-grid problem of the h-multigrid algorithm is not solved well with respect to the characteristic components, see [12]. We also see that there is hardly any difference in solving the coarse grid equations exactly with the Newton method or approximately by performing ν_C smoothing steps. This in contrary to the results obtained in Section 5.1, where we saw a large improvement when the coarse grid problem was solved exactly.

Regarding the hp-multigrid, where we first start with p-multigrid and continue at the lowest polynomial order with h-multigrid, we see that initially there is a significant improvement in reduction of the residual compared to the single-grid computation, but in the asymptotic regime single-grid and hp-multigrid have approximately the same residual reduction per work unit. The reason for this behavior is unclear

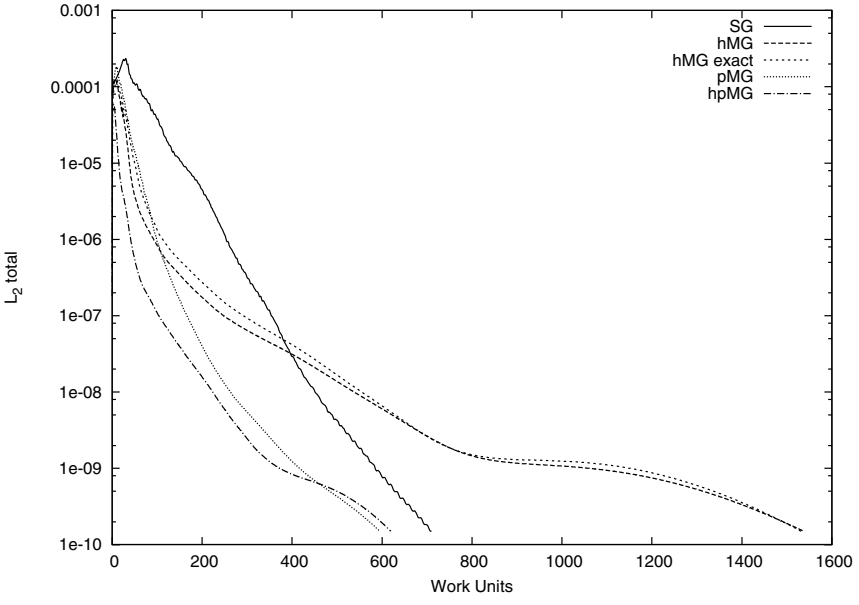


Fig. 4 Convergence history of single-grid, h-, p- and hp-multigrid techniques for the solution of inviscid flow around an NACA0012 airfoil ($\alpha = 2^\circ, Ma = 0.5$)

yet. For the p-multigrid method, initial convergence is significantly faster than for the single-grid computations, but in the asymptotic regime also a comparable convergence history is obtained.

6 Conclusions

Using discrete Fourier analysis, we have analyzed two- and three-level multigrid algorithms for the solution of linear algebraic systems originating from higher order accurate space-time DG discretizations. For the 2D advection-diffusion equation we have shown that by minimizing the spectral radius of the multigrid error transformation operator, a significant improvement in the multigrid performance can be achieved. The algorithms have been tested on a 2D problem containing boundary layers, where the optimized Runge-Kutta smoothers show a significant improvement compared to the original EXI-EXV Runge-Kutta smoother discussed in [2, 3]. Apart from optimizing the multigrid smoother, also the solution of the algebraic system on the coarsest mesh has a big impact on the multigrid performance.

We also compared the performance of h-multigrid with p- and hp-multigrid for solving the Euler equations. We considered subsonic inviscid flow around a NACA0012 airfoil. No significant difference was observed between the EXI scheme and the optimized Runge-Kutta smoothers. The main problem is the deterioration of the convergence rate after the high frequency error modes are smoothed, in

particular for h-multigrid. Also, the effect of solving the equations on the coarsest mesh exactly or approximately is small. This in contrast with the 2D advection-diffusion case. Furthermore, we saw that the p- and hp-multigrid methods show a better convergence rate than the h-multigrid method.

Acknowledgements. This research was partly funded by the ADIGMA project which was executed in the 6th Research Framework Work Programme of the European Union within the Thematic Programme Aeronautics and Space.

References

1. Brandt, A.: Rigorous quantitative analysis of multigrid, I: Constant coefficients two-level cycle with L_2-norm. SIAM J. Numer. Anal. 31, 1695–1730 (1994)
2. Klaij, C.M., van der Vegt, J.J.W., van der Ven, H.: Pseudo-time stepping methods for space-time discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. J. Comput. Phys. 219, 622–643 (2006)
3. Klaij, C.M., van Raalte, M.H., van der Ven, H., van der Vegt, J.J.W.: h-Multigrid for space-time discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. J. Comput. Phys. 227, 1024–1045 (2007)
4. Melson, N.D., Sanetrik, M.D., Atkins, H.L.: Time-accurate Navier-Stokes calculations with multigrid acceleration. In: Proc. 6th Copper Mountain Conference on multigrid methods. NASA Langley Research Center, pp. 423–437 (1993)
5. Sudirham, J.J., van der Vegt, J.J.W., van Damme, R.M.J.: Space-time discontinuous Galerkin method for advection-diffusion problems on time-dependent domains. Appl. Numer. Math. 56, 1491–1518 (2006)
6. Trottenberg, U., Oosterlee, C.W., Schüller, A.: Multigrid. Academic Press, London (2001)
7. van der Vegt, J.J.W., van der Ven, H.: Space-Time Discontinuous Galerkin Finite Element Method with Dynamic Grid Motion for Inviscid Compressible Flows I. General Formulation. J. Comput. Phys. 182, 546–585 (2002)
8. van der Vegt, J.J.W., Rhebergen, S.: Multigrid optimization using discrete Fourier analysis. Von Karman Institute Lecture Notes (2009)
9. van der Vegt, J.J.W., Rhebergen, S.: Multigrid optimization for higher order accurate space-time discontinuous Galerkin discretizations (in preparation)
10. Wesseling, P.: An introduction to multigrid methods. Wiley, Chicester (1991)
11. Wienands, R., Joppich, W.: Practical Fourier analysis for multigrid methods. Chapman & Hall/CRC (2005)
12. Yavneh, I.: Coarse-grid correction for nonelliptic and singular perturbation problems. SIAM J. Sci. Comput. 19, 1682–1699 (1998)

This page intentionally left blank

Chapter 19
COOLFluiD – A Collaborative Simulation Environment for Research in Aerodynamics

T. Quintino and H. Deconinck

Abstract. The COOLFluiD platform is a collaborative simulation environment where some parnters consolidated their developments during the ADIGMA project. We define the principles behind the design of the environment and how it is composed of components. We present this component architecture and explain how we use it to stimulate synergetic research collaborations. We provide implementation details of the building of components and doman representation.

1 Introduction

COOLFluiD is a Collaborative Simulation Environment (CSE) focused on complex multi-physics simulations. It is targetted at a wide variety of fields, from Aerodynamics to Structural Analysis, from Aeroacoustics to Heat Transfer. At the start of the ADIGMA project, COOLFluiD already existed, but it has since then been improved. Improvements came in many forms, but in this manuscript we will focus on what allows multiple discretizations to coexist within the same platform.

The CSE is based on components which plug into a Kernel. Components are solvers, equations, mesh formats, etc. Finite Volume, Finite Element methods and Navier-Stokes and Magnetohydrodynamics equations are examples of isolated components that collaborate together to form a simulation. The Kernel combines the components together as building blocks, trying to use the best methodology for each application. The aim is to create high-performant solvers, each dedicated to a specific application, while reusing the same components (algorithms). This approach differs from typical monolithic solvers which employ only one methodology, and thus are sub-optimal when applied to a variety of applications.

COOLFluiD is very flexible in the sence that every Physical Model or Numerical Method is a separate plug-in component. External developers contribute with

T. Quintino · H. Deconinck
Von Karman Institute, Chausse de Waterloo, 72, 1640 Rhode-St-Gense Belgium
e-mail: quintino@vki.ac.be, deconinck@vki.ac.be

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 271–285, 2010.
springerlink.com

their own components as plug-in's, in the form of numerical solvers, acceleration methods, new physical models, equation terms, etc. These external plug-ins can be loaded at run-time to plug into the already existing components.

2 Representing the Domain of Computation

The COOLFluiD environment was designed to support different types of discretizations, i.e. support the existence of multiple solvers that use different representations of the domain and the solution. Depending on the type of computation, different views of the domain are required. For example, computing the volume of the domain requires a global view, whereas computing the tangent to the surface point, requires a localised view of the domain surface. The first distinction we use is between topology and geometry.

Topology focuses on the intrinsic structure of geometric objects, by distinguishing qualitative geometry from ordinary geometry in which quantitative problems are treated. Topology deals with geometric problems that depend not on the exact shape of objects, but on how they are assembled together. For instance, the topological view of the domain will show that an aircraft has wings, engines and fuselage or that the landing gear is retracted.

Geometry focuses on questions related with size, shape and relative position of objects. The geometric view of the domain will show how much is the length of an aircraft and what is its wingspan.

Another distinction we use is between analytical information and numerical (discrete) information of the domain.

Analytical information is described in terms of functions or sets of functions. It provides information about the domain as a globally continuous object. For instance, the analytical view will show what is the normal vector to the aircraft surface or the curvature on the leading edge of the wing.

Numerical information is described in terms of isolated sets of numbers. It provides local information about points in the domain. For instance, the numerical view will show how big is the deflection of the aircraft wing tip or what are the coordinates of the aircraft nose.

Therefore, we have distinguished four different but complementary views of the domain. The topological view is global and identifies discrete parts of the domain, while the geometrical view provides information on locally continuous domain subdivisions. The analytical view is global and continuous, composed of surfaces and bodies connected together, while the numerical view is local and discrete, made of raw data in the form of collections of numbers. These views are depicted in figure 1, where they present different information about the same domain.

One Concept per View. For each view, we introduce a concept to represent it. Each concept generates one or more interfaces in the design that the numerical

methods use to access the domain. In figure 1, we show these concepts together with the views they represent, and we summarise them below.

Domain Model is the concept that represents the *Analytical View*. Its role is to provide an access to the continuous definition of the model surfaces, possibly providing an access to a CAD representation.

Topological Region is the concept that represents the *Topological View*. It defines regions on the domain, where the numerical methods perform their tasks. Regions are grouped in sets, for easier handling.

Geometric Entity is the concept that represents the *Geometrical View*. By using a generic connectivity storage, it provides geometrical information for the numerical methods to build the entities (cells, faces, edges). The numerical method uses these geometric entities to perform its actions.

Data Storage is the concept that provides *Numerical View*. It stores the arrays that contain all the numeric raw data. It also provides ways for different components to share data.

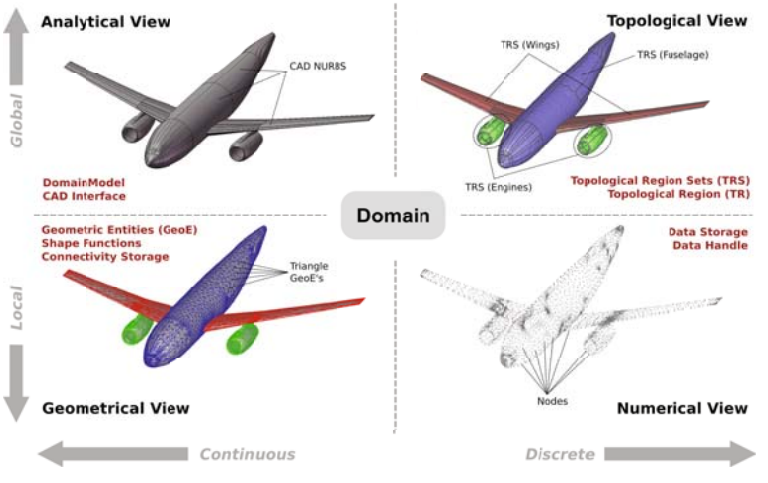


Fig. 1 Relation of Domain Views to Solution Concepts (in red)

Many Concepts, One Access. All these concepts are relative to the same domain, so it is sensible and convenient to group their access under the same generic interface, which we call *MeshData*. This interface provides the access to each of the underlying concepts, in a centralised manner. This allows to create multiple *MeshData* objects which coexist in the case of multi-domain simulations.

This *MeshData* follows an adapted *Facade* design pattern from [3]. The class diagram is represented in figure 2. All concepts can be accessed by proper functions, with the exception of *GeometricEntity* which is not explicitly present, instead

represented by the *ConnectivityStorage*. Geometric entities depend on the numerical method. Each method creates its own by using the connectivity information, therefore only the latter is present.

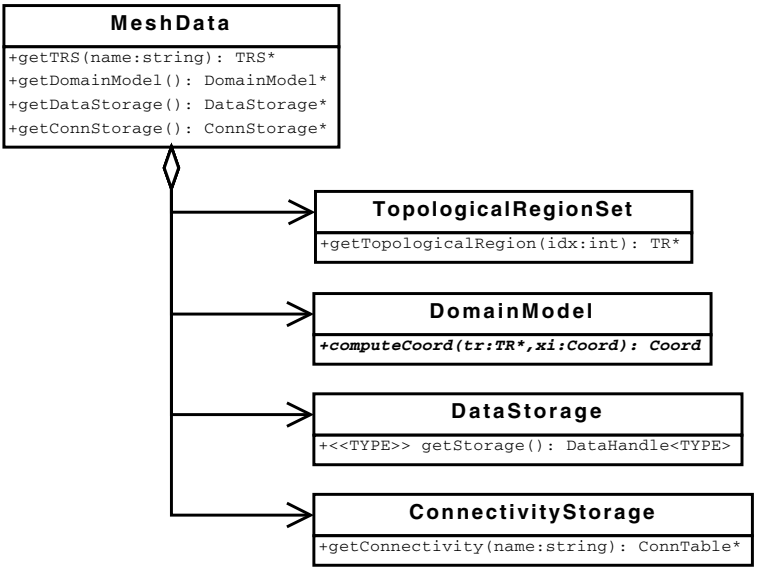


Fig. 2 Class Diagram of MeshData Concepts

We allow each numerical method to create its own specialised data-structures. Therefore, the framework loads the minimum mesh information into memory: the coordinates of each mesh point, the variables of the solution, and the cell to node connectivity. Then the *MeshCreator* delegates the rest of the building process to a *builder* object, called *MeshDataBuilder*. As described by the class diagram of figure 2, this object is provided by each numerical method and it builds the remaining structures according to the specific needs of the method. For example, the method may need a reverse connectivity from nodes to cell or maps from surfaces to boundary faces.

3 Representing the Topology

Topological Regions. To represent the topology of the domain we decompose the domain into regions of interest, which we call *TopologicalRegion*'s (*TR*). These regions represent the surfaces or volumes that form the domain. If the domain is a simple sphere, then we can imagine that the surface of the sphere is one *TR* and the volume is another *TR* albeit of different dimensionality. To these regions the numerical methods apply their computation actions. For instance, we can calculate the surface or the volume of the sphere by numerical integration.

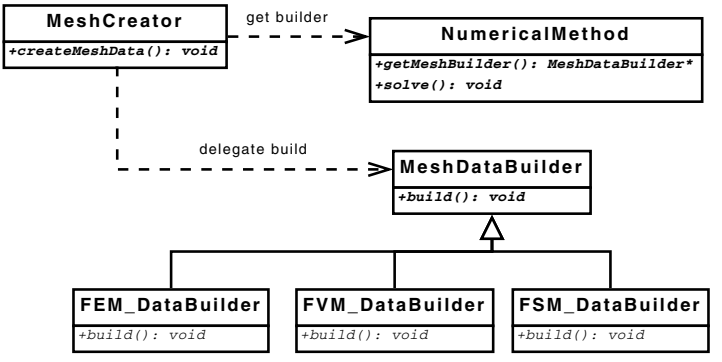


Fig. 3 Collaboration diagram of MeshBuilder

Sets of Topological Regions. It is convenient to group together the regions that have in common the same actions. Since order is not relevant, this group forms a *set*, called *TopologicalRegionSet* (*TRS*). In figure 4, one *TRS* is the "Wings" and is composed of all the CAD surfaces that define the aircraft wings. The same for the fuselage and engines. Each surface is a *TR*. Note that a *TRS* may form a discontinuous representation, in this case 2 separate wings.

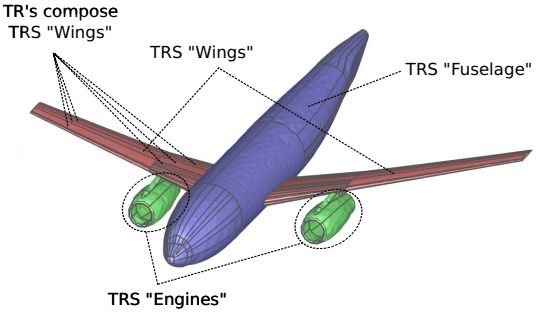


Fig. 4 TRS's are composed of TR's

In figure 5, we show the relation between the classes that implement these concepts. A *TRS* provides an interface to access each *TR*. It also gives access to all the states (variables) and nodes (coordinates) in that *TRS* as a global group. The *TR* provides the information that the numerical methods need to build *GeometricEntity*'s (see section 5.2). This information is stored in large *connectivity tables* (class *ConnTable*) and indicates which *Node*'s and *State*'s compose a given entity.

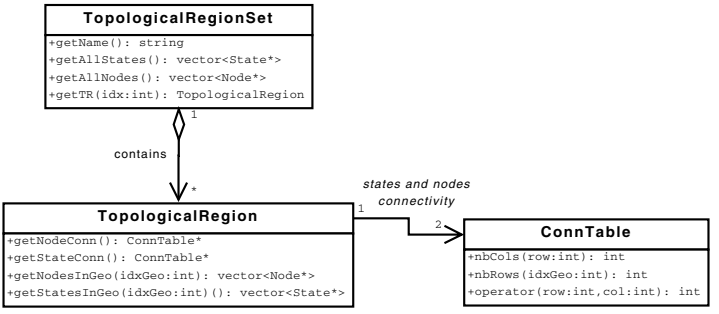


Fig. 5 Class diagram of TopologicalRegion's

4 Representing the Analytical Model

Within COOLFluiD the analytical definition of the domain represents the CAD model. This defines the form and shape of the surfaces of the boundary, typically the wing and fuselage of the aircraft. This analytical definition is provided by the *DomainModel* abstraction. This class, shown in Figure 6, provides functions to access the definition of the surfaces. This information is used to perform mesh adaptation near the boundary or to curve the high-order curvilinear elements that touch the boundary.

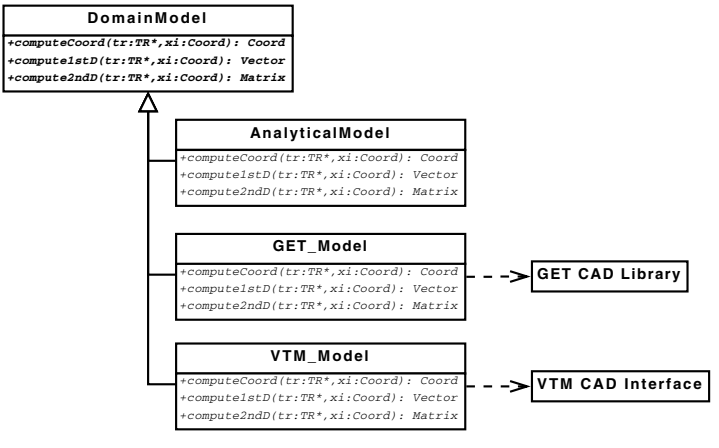


Fig. 6 Class diagram for the DomainModel

The functions present in the interface provide:

computeCoord transformation from parametric to physical coordinates
computeParamCoord inverse transformation from physical coordinates to parametric space
compute1stDeriv first derivatives on physical coordinates

compute2ndDeriv second derivatives on physical coordinates
computeAll physical coordinates and all derivatives in a single function call

Since each function will be a *C++* virtual call, a *computeAll* function is provided to compute all coordinates and derivatives in a single function call.

The model representation is usually piecewise analytical. As described in the previous sections, behind each surface there is a definition of each point in a parametric space (u, v) that maps to a point in the physical space. This parametric function may use primitives like planes, spline curves, NURBS surfaces, *etc*. Using these parametrisations we can compute not only positions, but also derivatives and curvatures of the surfaces.

Multiple Representations. For some definitions the underlying representation is not analytical, but an approximation with a fine geometry discretisation. This is the case of the STL [1] triangulated surface format, as used by the *VTM* CAD library of Dorochenko [2]. The functions on the interface assume that the underlying model is continuous and analytical. Its is up to the concrete implementation to mimic this behaviour in case of non-analytical representations. As there are many CAD systems on the market, one of the objectives of the COOLFluiD design is to be generic and independent from any specific CAD system. To enable the use of any system, the *DomainModel* class is abstract and leaves the work to the concrete implementations.

Within ADIGMA, two implementations have been developed: The *GETModel* and the *AnalyticalModel*. Majewski, from University of Warsaw, developed the *GET* library which follows the NURBS approach described another chapter of this book. The other implementation is the *AnalyticalModel* which allows the user to define analytical functions for each surface. It parses these functions and maps them to each boundary surface. However, in practice with the *AnalyticalModel* only very simple geometries can be defined, therefore this model is used in the most simple cases for testing and research (like cylinders, spheres, cubes, etc). The user defines explicitly the direct transformation from parametric space to physical space as well as the first and second derivatives (if necessary). The reverse transformation is computed numerically using a Newton method to solve a non-linear minimum distance problem.

5 Representing Geometry

Most numerical discretisations are performed on some geometric entity: compute the flux through a face, compute the heat source in a cell, compute the distance to a wall, *etc*. All these computations require geometry information, like coordinates, areas, volumes and solutions at certain points.

Geometric Entities. Within COOLFluiD design, *GeometricEntity* is the interface that provides such geometric information. Geometric entities decouple the numerical computations from the elements on which they are computed. This allows to vary the element characteristics without changing the numerical algorithms. Some examples are:

- program an algorithm that computes element distance to the nearest wall independent of dimension (it will work for 2D as well as 3D).

- program a finite element discretisation, independent of the element shapes: triangles, quadrilaterals, tetrahedra, prisms, *etc*.
- program the same finite element discretisation, independent of the order of approximation of the solution. For example, for first order (P1) or higher order elements ($P \geqq 2$).
- select different solution spaces, for instance Lagrange nodal functions or Chebyshev modal functions.

Presented in figure 7, *GeometricEntity* provides geometry and solution related functions. It is composed of *Node*'s and *State*'s. Nodes represent coordinates and States represent solution variables. Given an ordered set of *Node*'s and a function definition, it provides a continuous representation of the geometry. Given an ordered set of *State*'s and another function definition it provides a locally continuous solution representation.

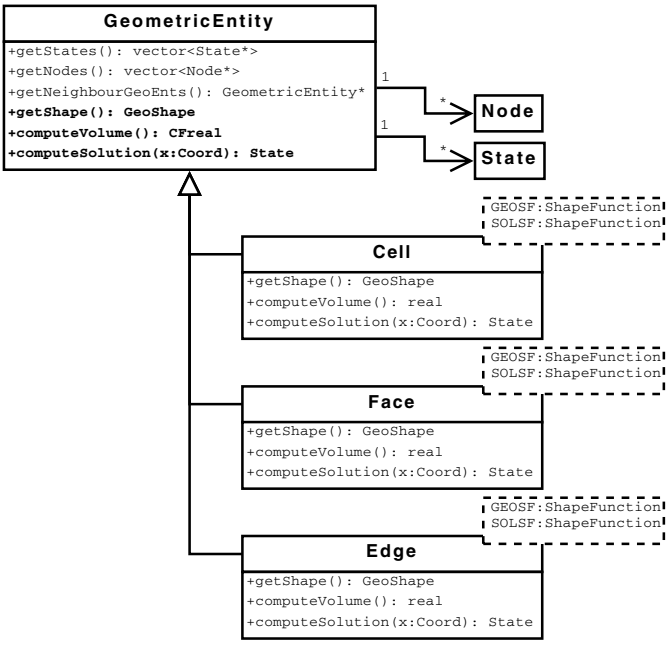


Fig. 7 Class diagram for Geometric Entity

This element representation can, for example, provide solution interpolation or compute the gradients on a finite element given the coordinates in the parametric space inside the element. This is done by mapping the values in the parametric space $\hat{K}$ to the physical space K via a transformation F, as shown in figure 8. To implement this, the concrete entities *Cell*, *Face* and *Edge* derive from *GeometricEntity* and are parametrised by the functions that define the solution and the geometry. These functions are *C++* template parameters to ensure minimal performace loss.

Only the function call across the *GeometricEntity* interface is virtual. Inside, the implementation is made of static calls to the functions which increase the probability of compiler optimisations. For this class this is very important since it is used in the innermost computation loops.

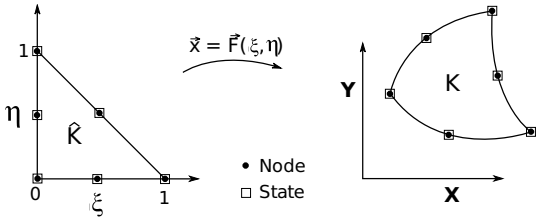


Fig. 8 Transformation from parametric $\hat{K}$ to physical space K

Using functions for describing the geometry and the solution independently is the standard Finite Element way of describing elements [8]. Because this element representation is very generic we assume it is enough to describe all elements for every numerical method present in COOLFluiD. For example, the finite volume cell-centered element is considered a P1 Lagrange function for the geometry and a P0 (constant) Lagrange function of the solution.

Shape Functions. Finite element representation is generic because it uses a very powerful mathematical abstraction: *shape functions*. They are used to describe the solution or the geometrical space. This allows to write a generic numerical method which can change its behaviour by choosing the proper function spaces. For example, in structural analysis continuous Lagrangian spaces are preferred, but for MHD applications the Raviart-Thomas [7] solution space solves the divergence free problem implicitly. As shown in figure 7, the concrete entities *Cell*, *Face* and *Edge* are generic templates which take two *ShapeFunction*'s as type parameters. By avoiding excessive subclassing, we reduce the size of the inheritance tree and simplify the code. Below we see an example code that implements the interface class *GeometricEntity* and the implementation of *Cell*. Note how the pure virtual functions, like *computeGeoShapeFunc*, are implemented in *Cell* by calling statically the function *computeShapeFunction* in the shape function template.

```
typedef std:valarray<double> DVector;
class GeometricEntity {
public:
  std::vector<State*>& getStates() { return m_states; }
  std::vector<Node*>& getNodes()  { return m_nodes;  }
  virtual double computeVolume() = 0;
  virtual DVector computeGeoShapeFunc (const DVector& mapcoord) = 0;
  virtual DVector computeSolShapeFunc (const DVector& mapcoord) = 0;
};

template <typename GEO_SHAPE_FUNCTION,
          typename SOL_SHAPE_FUNCTION>
class Cell : public GeometricEntity {
public:
  virtual double computeVolume ()
  {
```

```
      return GEO_SHAPE_FUNCTION::computeVolume(m_nodes);
    }
    virtual DVector computeGeoShapeFunc (const DVector& mapcoord)
    {
      return GEO_SHAPE_FUNCTION::computeShapeFunction(mappedCoord);
    }
    virtual DVector computeSolShapeFunc (const DVector& mapcoord)
    {
      return SOL_SHAPE_FUNCTION::computeShapeFunction(mappedCoord);
    }
  };
```

Together with the *self-registration* techniques described in [5, 6], the developer can register his own elements into the application by varying the parametrisation of the Geometric Entities. Below we see a code list of the providers that are registered by compiling and linking to this code.

```
/// Lagrange Triangle cell P1 geometry, P3 solution
ObjectProvider < Cell<LagrangeShapeFunctionTriagP1,
                      LagrangeShapeFunctionTriagP3> >
CellTriagLagrangeP1LagrangeP3("CellTriagLagrangeP1LagrangeP3");

/// Lagrange Quadrilateral cell P1 geometry, P2 solution
ObjectProvider < Cell<LagrangeShapeFunctionQuadP1,
                      LagrangeShapeFunctionQuadP2> >
CellQuadLagrangeP1LagrangeP2("CellTriagLagrangeP1LagrangeP2");

/// Lagrange Tetrahedron cell P2 geometry, P2 solution
ObjectProvider < Cell<LagrangeShapeFunctionTetraP2,
                      LagrangeShapeFunctionTetraP2> >
CellTetraLagrangeP2LagrangeP2("CellTetraLagrangeP2LagrangeP2");
```

The high-order shape functions, see figure 9, are commonly used to increase the solution accuracy by improving the finite element approximation. In space, high-order shape functions are used to define curved elements, suitable for curved geometries. By combining solution and space variations, we can construct many different types of finite elements.

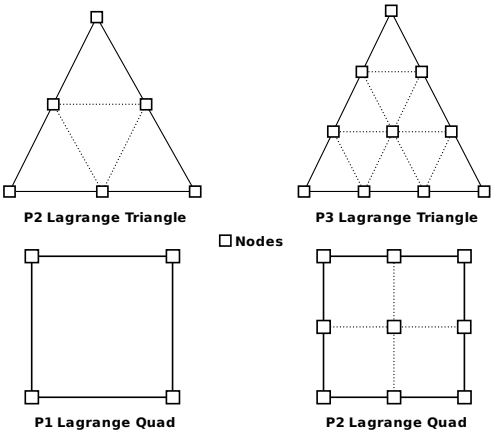


Fig. 9 Example of shape functions of multiple orders

The element type on the left of figure 10 is constructed with a first order polynomial shape function (P1) for defining the geometry and a second order (P2) for the solution. Because the space parametrisation is lower than the solution order, we call it a *subparametric* element. The element in the right has the inverse setup, a P2 space parametrisation and a P1 solution approximation, therefore is a *superparametric* element. The element in the middle we call *isoparametric* because it has the same order (P2) for space and solution. This technique allows the three types of parametrisations.

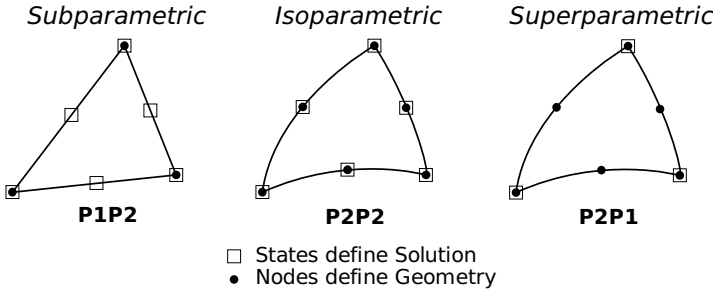


Fig. 10 Types of Element Parametrisations

The classes *Node* and *State* provide the access to the numerical data that the shape functions use to compute the concrete values. They are stored globally in large arrays of data but accessed locally via the *GeometricEntity* class. Using nodes and states we can interpolate respectively the geometry and the solution in the entity, following for instance expressions as presented below, where $N_i^{x,u}$, $\mathbf{x}_i$ and $\mathbf{u}_i$ are the shape functions, the nodal and the state values:

$$\mathbf{x}(\xi) = \sum_i N_i^x(\xi)\mathbf{x}_i \tag{1}$$

$$\mathbf{u}(\xi) = \sum_i N_i^u(\xi)\mathbf{u}_i \tag{2}$$

5.1 Storage of Connectivity

To correctly associate the nodes and states to each geometric entity, it is necessary to know the mesh connectivity information, which is stored in *ConnTable* objects. This information is one of the most memory consuming in an unstructured simulation, therefore *ConnTable* has been developed strictly for low memory consumption. Figure 11 exemplifies the construction of *ConnTable* for a small mesh. Note that we support hybrid meshes with elements of different types, therefore the table has this mixed information. Nevertheless, to optimise memory alignment and avoid indirections, we designed this table to be a rectangular matrix. The invalid entries are

marked with the maximum integer possible to represent by the computer. In principle, these entries are never accessed directly by the algorithms. The marker is useful to obtain the information of the number of valid columns per row of the matrix. This design has been tested against many other alternatives, like vector of vectors, or vector of pointers and it showed to have the fastest access time and the lighest memory footprint. The code that implements this class is presented in [5].

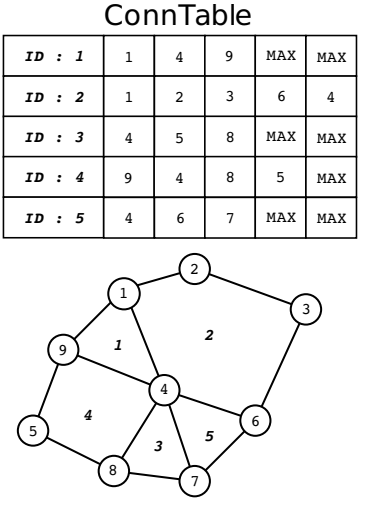


Fig. 11 Example of a connectivity table (ConnTable)

Each *ConnTable* object is stored in a *ConnectivityStorage* facility, accessed via a *MeshData* interface. This allows each numerical method to create the dedicated connectivity tables to correctly build their own geometric entities. The numerical method associates a name to the connectivity when placing it in the storage, which can then be reused by any other algorithm, like the mesh adaptation. Sharing the connectivity tables avoids unnecessary double storage of large objects.

5.2 Building the Geometric Entities

When we first implemented the *GeometricEntity* design and compared the memory usage to similar solvers, we were surprised to see that too much memory was used. With further investigations we learnt that storing all *GeometricEntity* objects in a simulation consumes too much memory. Therefore we introduced another concept in the design, the *GeoBuilder*. Its responsibility is to build the geometric entities on-the-fly, at the request of the numerical method, using the connectivity information. This avoids to store the *GeometricEntity*'s. In this design, we do not store the objects but just *what they are made of*.

In figure 12 we see that the client, in this example an algorithm named *FEMCommand*, accesses a *FEMGeoBuilder* object. This builder has a pool of GeometricEntities. These are pre-built and the algorithm accesses them by calling *buildGE*. When the command has done its job, it returns the entity to the pool by using the function *releaseGE*.

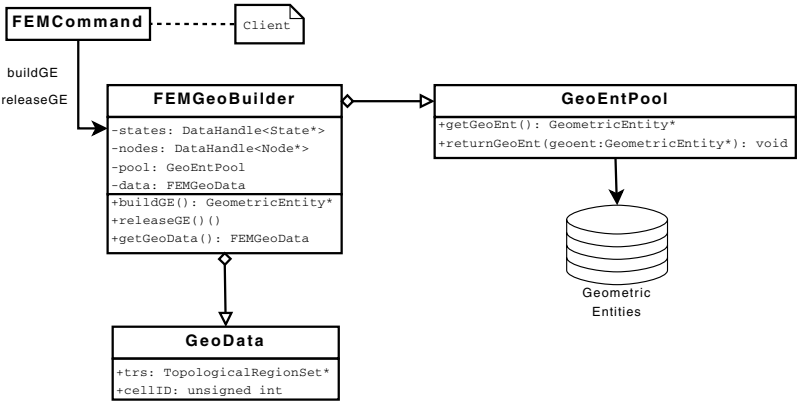


Fig. 12 Example of a GeoBuilder for the FEM method

In the next code listing we see how the client code uses the *GeoBuilder*. First the client gets the *GeoData* from the *GeoBuilder* where it sets the connectivity information (via the object *TRS*). This will inform the *GeoBuilder* from where to retrieve the connectivity information to build the *GeometricEntity*. When the loop starts, the client assigns each iteration the ID of the cell to build to the *GeoData*. The client then requests the builder to build the *GeometricEntity*. Once built, the client uses the entity for his computations. In the end, before continuing to the next entity he must release the current entity. This allows the builder to reuse it again.

```
/// these usually are given to the function
TopologicalRegionSet cells = MeshData::getInstance().getTrs("cells");
FEMGeoBuilder geoBuilder;

// **** LOOP OVER CELLS **** //
FEMGeoBuilder::GeoData& geoData = geoBuilder->getGeoData();
geoData.trs = cells; // choose on which TRS to loop

const CFuint nbGeos = cells->getNbGeoEnts();
for (unsigned int cellID = 0; cellID < nbGeos; ++cellID)
{
  // build the GeometricEntity
  geoData.idx = cellID;
  GeometricEntity& cell = *geoBuilder->buildGE();

  // ... computation on the cell goes here ... //
  vector<State*>& cell_states = cell.getStates();
  vector<Node*>&  cell_nodes  = cell.getNode();
  ...

  //release the GeometricEntity when done
  geoBuilder->releaseGE(); }
```

This design has shown not only to reduce memory usage, but in certain conditions also speed up of the loop over the elements in the mesh, by improving memory caching and reducing the memory that the processor needs to fetch.

Supporting multiple numerical methods. We have set all the ingredients to enable a flexible building of specialised elements for each numerical method. We can create a special *GeoBuilder* for each numerical method, which assembles the entities on-the-fly. This builder gets the nodes, states and connectivity via the *TRS* to build each entity. The builder selects the concrete type of entity according to the shape functions best suited for the problem. If special connectivity tables are needed, these can be created and registered in the *ConnectivityStorage* by the different numerical methods. This way, we can create specialised discrete entities, potentially very performant, without allocating unnecessary data structures. Moreover, the *GeometricEntity* provides convenient interpolation functions that allow the developer to program arbitrary high-order discretisations with less effort.

6 Representing the Numerical Data

Storing Numerical Data. To provide general access to stored data related to the geometry, like coordinates and normals, we created the interface *DataStorage*. DataStorage allows numerical methods to declare a data array of arbitrary type and size, and to store it with an associated name. This name works like a contract to a database. Any method that knows the name and the type of the data can access it. This allows for numerical methods to share data without having to explicitly declare it in their interfaces.

```
void MethodA::do_something() // Numerical method A
{
  DataStorage* ds = MeshData::getInstance().getDataStorage();
  DataHandle< real > volumes = ds->createData<real>("volumes", size);

  volumes[i] = ...  /* computes the volumes */
}

void MethodB::do_stuff() // Numerical method B
{
  DataStorage* ds = MeshData::getInstance().getDataStorage();
  DataHandle< real > volumes = ds->getData<real>("volumes");

  /* reuses the volumes without knowing who computed them */
}
```

Common Data Representation. More important than sharing, this design creates a standardised format of data that all components in the environment use, thus improving coexistence of components. To handle the data in a uniform way, the methods access it via the *DataHandle* interface which is explained in [5]. This interface also shields the methods from the concrete communication paradigm that synchronises the data in a parallel simulation.

This storage facility also decouples the way the data is placed in memory from the handling of the data. This way the storage algorithm may be changed to more efficient memory layouts without affecting the numerical methods. For example,

when supported by the operating system, we are able to use growable arrays, as described in [4].

7 Conclusion

In a multi-method platform each numerical method requires different information the domain of computation. In our design, this information is categorised in different views of the domain. To completely represent the domain we have selected four views. Each view is implemented by abstract interface. All four interfaces are connected to each other, both conceptually and in terms of code implementation. The *TR* links to the CAD definition that is provided by the *DomainModel*. The *TRS* groups the *TR*'s and holds the connectivity of the *GeometricEntity*'s that are created by a *GeoBuilder* which is method specific. The methods use the geometric entities together with numerical data, like coordinates and variables, stored in *DataStorage* and accessed via *DataHandle*'s to perform their computations. With this design COOLFluiD supports numerical methods as diverse as:

- Cell Centered Finite Volume - developed by the Von Karman Institute.
- Discontinuous Galerkin - developed by the Charles University of Prague within ADIGMA.
- Continuous Finite Element - developed by the Von Karman Institute.
- Residual Distribution - developed by the Von Karman Institute.
- Spectral Finite Difference and Finite Volume - developed by the Vrij Universiteit Brussels.

References

1. STL - Stereolithography Interface Definition (1989)
2. Dorochenko, A.: Development and evaluation of 3D computer integrated manufacturing and engineering tools (CIME tools) for electrochemistry. PhD thesis, Vrije Universiteit Brussel (2007)
3. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series. Addison-Wesley, Reading (1994)
4. Kimpe, D., Vandewalle, S., Poedts, S.: Evector: an efficient vector implementation. In: POOSC 2005 Workshop Notes (2005)
5. Quintino, T.: A Component Environment for High-Performance Scientific Computing, Design and Implementation. PhD thesis, Katoliek Universiteit Leuven (December 2008)
6. Quintino, T., Deconinck, H.: Run-time automatic instantiation of algorithms using C++ templates. Int. J. Comput. Sci. Eng. 4(4), 314–324 (2009)
7. Raviart, P.A., Thomas, J.M.: A mixed finite element method for second order elliptic problems. In: Galligani, I., Magenes, E. (eds.) Mathematical aspects of Finite Element Methods, vol. 606, pp. 292–315. Springer, Heidelberg (1977)
8. Zienkiewicz, O.C., Taylor, R.L.: The Finite Element Method: The Basis, 5th edn. Butterworth Heinemann, Butterworths (2000)

This page intentionally left blank

Chapter 20
Robust and Efficient Implementation of Very High-Order Discontinuous Galerkin Methods in CFD

F. Bassi, A. Colombo, N. Franchina, A. Ghidoni, and S. Rebay

Abstract. Discontinuous Galerkin (DG) methods are a very powerful numerical techniques, that offer high degree of robustness, accuracy and flexibility, nowadays necessary for the solution of complex fluid flows. The drawback is the relatively high computational cost and storage requirement. This work will focus on two approaches which can be adopted to enhance the computational efficiency of this class of methods: (i) a DG discretization based upon co-located tensor product basis functions, and (ii) a *p*-multigrid solution strategy. The effectiveness of the proposed approaches has been demonstrated by computing 3D inviscid and turbulent test cases.

1 Introduction

During the last two decades, the considerable advances in algorithm development and the huge increase of computer power have made CFD a key discipline for the industry. However the numerical technology used in standard industrial codes is still mainly based on formally second-order accurate finite volume or finite element schemes. In practice the accuracy provided by these methods is inadequate in applications such as large eddy simulation, direct numerical simulation, computational aeroacoustic unless using prohibitively large computational resources, that are beyond the available capabilities. Higher-order accurate methods, such as discontinuous Galerkin (DG) methods, are therefore needed to cope with this class of flow simulations. However the price to pay for their high degree of robustness, accuracy and flexibility is the relatively high computational cost and storage requirement.

F. Bassi · A. Colombo · N. Franchina
Facoltà di Ingegneria, Università degli studi di Bergamo, Via Marconi 5,
24044 Dalmine, Italy

A. Ghidoni · S. Rebay
Dipartimento di Ingegneria Meccanica e Industriale, Università degli Studi di Brescia,
Via Branze 38, 25123 Brescia, Italy

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 287–299, 2010.
springerlink.com

The most commonly considered time integration schemes used with DG space discretization are the explicit multistage Runge-Kutta (RK) methods and the implicit schemes [8]. However, the former shows extremely slow convergence rate for large scale simulations and/or for high-order polynomial approximations, while the latter, even reducing the number of iterations needed to reach the steady state solution, is characterized by an high computational demand, both in terms of the CPU time and of the memory required to store the Jacobian matrix, which may be prohibitive for a large scale problem and high-order solutions.

In the last years many solutions have been proposed to enhance the efficiency of DG methods. We can mention, for example, the techniques developed in the context of the spectral element methods (SEMs) [15] and p-multigrid (p-MG) [10, 13, 5, 4] solution strategies. In SEMs multidimensional basis functions are obtained as a tensor product of 1D basis with coincident interpolation and quadrature nodes (i.e. co-located bases). This technique allows to speed up the construction of the discrete DG space discretization for quadrilateral and hexahedral elements. The basic idea of the p-MG algorithm is instead to accelerate the convergence of standard iterative schemes by solving the equations considering a series of progressively lower-order approximations on the same grid.

In this paper the two approaches previously described have been analysed and extended. In particular the nodal polynomial approximation described above and presented in [3] has been extended to the 3D case and the effect of the coupling with p-MG strategy introduced in [4] has been investigated. Moreover the p-MG strategy proposed in [5] has been implemented in the implicit DG code MIGALE [2], and has been applied to turbulent flows computation.

The organization of this paper is as follows: in Section 2 the DG space discretization is briefly presented, in Section 3 Lagrangian polynomial approximations are discussed, Section 4 describes the p-multigrid algorithm, and Section 5 shows the computed results for the subsonic inviscid and turbulent flow around the ADIGMA [14] BTC0 three dimensional body.

2 DG Space Discretization

The compressible flow equations (inviscid,viscous and turbulent) can be written in compact form as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{F}_c = \nabla \cdot \mathbf{F}_v + \mathbf{s}, \tag{1}$$

where $\mathbf{u}, \mathbf{s} \in \mathbb{R}^M$ are the vectors of conservative variables and source terms, $\mathbf{F}_c, \mathbf{F}_v \in \mathbb{R}^M \otimes \mathbb{R}^d$ the inviscid and viscous flux functions, M and d the number of equations and the space dimension, respectively. Detailed description of these equations can be found, *e.g.* , in [7, 6, 2].

The weak form of Eq. (1) can be written as

$$\int_{\Omega} \phi \frac{\partial \mathbf{u}}{\partial t} \, \mathrm{d}\mathbf{x} - \int_{\Omega} \nabla \phi \cdot \mathbf{F}(\mathbf{u}, \nabla \mathbf{u}) \, \mathrm{d}\mathbf{x} + \int_{\partial \Omega} \phi \mathbf{F}(\mathbf{u}, \nabla \mathbf{u}) \cdot \mathbf{n} \, \mathrm{d}\sigma = \int_{\Omega} \phi \mathbf{s}(\mathbf{u}, \nabla \mathbf{u}) \, \mathrm{d}\mathbf{x}, \tag{2}$$

for any arbitrary, sufficiently smooth test function ϕ, where $\mathbf{F}$ is the sum of the inviscid and viscous contributions, $\partial\Omega$ the boundary of Ω, and $\mathbf{n}$ the unit outward normal vector to the boundary.

The DG discretization of Eq. (2) is defined on a mesh $\mathscr{T}_h = \{K\}$ of an approximation Ω_h of Ω, which consists of a set of non-overlapping elements K not necessarily simplexes. The functions ϕ and $\mathbf{u}$ are approximated on $\mathscr{T}_h$ as piecewise polynomial functions $\phi_h \in V_h$ and $u_{h_i} = u_{h_1}, \ldots, u_{h_M} \in V_h$, possibly discontinuous on element interfaces, where the discrete space V_h is defined as

$$V_h \stackrel{\text{def}}{=} \left\{ \phi_h \in \left(L^2(\Omega_h) \right)^M : \phi_h|_K \in \mathbb{P}^k(K)^M \ \forall K \in \mathscr{T}_h \right\}, \tag{3}$$

where $\mathbb{P}^k(K)$ are the polynomials of global degree at most k on K.

In order to introduce a coupling between adjacent elements and weakly impose the boundary conditions a suitable numerical flux function $\widehat{\mathbf{f}}$ has to be defined. The inviscid contribution is computed by means of the "exact" Godunov flux function, while the BR2 scheme is employed for the viscous part, see [11, 9, 12, 1].

Accordingly to these considerations, the DG formulation of Eq. (2) requires to find $u_{h_i} = u_{h_1}, \ldots, u_{h_M} \in V_h$ such that

$$\begin{aligned} \int_{\Omega_h} \phi_h \frac{\partial \mathbf{u}_h}{\partial t} \, \mathrm{d}\mathbf{x} - \int_{\Omega_h} \nabla_h \phi_h \cdot \mathbf{F}(\mathbf{u}_h, \nabla_h \mathbf{u}_h + \mathbf{r}(\llbracket \mathbf{u}_h \rrbracket)) \, \mathrm{d}\mathbf{x} \\ + \int_{\Gamma_h} \llbracket \phi_h \rrbracket \cdot \widehat{\mathbf{f}}\left(\mathbf{u}_h^{\pm}, (\nabla_h \mathbf{u}_h + \eta_e \mathbf{r}_e(\llbracket \mathbf{u}_h \rrbracket))^{\pm} \right) \, \mathrm{d}\sigma \\ + \int_{\Omega_h} \phi_h \mathbf{s}(\mathbf{u}_h, \nabla_h \mathbf{u}_h + \mathbf{r}(\llbracket \mathbf{u}_h \rrbracket)) \, \mathrm{d}\mathbf{x} = \mathbf{0}, \end{aligned} \tag{4}$$

for all $\phi_h \in V_h$. Γ_h is the set of boundary and internal faces, $\llbracket \cdot \rrbracket$ is the jump operator as defined in [1], $\mathbf{r}_e$ the lifting operator, which is assumed to act on the jumps of $\mathbf{u}_h$ componentwise, and η_e the penalty parameter, prescribed accordingly to [12, 1].

All integrals appearing in Eq. (4) are computed by means of Gauss quadrature formulae with a number of points consistent with the accuracy required.

The discrete problem corresponding to Eq. (4) can be written as

$$\mathbf{M} \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} + \mathbf{R}(\mathbf{U}) = 0, \tag{5}$$

where $\mathbf{U}$ is the global vector of unknown degrees of freedom, $\mathbf{M}$ is the global block diagonal mass matrix, and $\mathbf{R}$ the residuals vector.

3 Nodal Polynomial Expansion Basis

The choice of the expansion basis adopted for the discretization depends on several issues, namely, i) numerical efficiency, ii) conditioning of the DG discrete operator, and iii) capability of easily handling complex shaped grids.

Nodal basis functions defined on the reference space can well accomplish the first two objectives. Indeed, expansion basis computed by means of the tensor product technique on Gaussian nodes–usually exploited within spectral element methods–do not exhibit the oscillations typical of functions defined on equispaced nodes and many numerical operations can be performed efficiently. This approach, even if it allows to evaluate exactly only integrals up to the linear contribution of the weak form of the governing equations, does not show instability problems due to under-integration of the equations nonlinearities.

Multi-dimensional expansions are easily obtained by combining the one-dimensional functions. For the two-dimensional case, for example, the discrete quantity $\mathbf{u}_h$ for a polynomial approximation of degree $k-1$ on element K can be expressed as

$$\mathbf{u}_h(\mathbf{x},t)|_K = \sum_{i=1}^{k}\sum_{j=1}^{k} \phi_i(\xi)\phi_j(\eta)\mathbf{U}_{ij}(t), \quad \mathbf{u}_h \in \mathbb{P}^{k-1} \otimes \mathbb{P}^{k-1}, \tag{6}$$

where $\phi_{i,j}(\cdot)$ are the Lagrange polynomials along the coordinate axis of the reference quadrilateral $|(\xi,\eta)| \leq 1$. These functions satisfy the cardinality property, *i.e.* , $\phi_i(\xi_j) = \delta_{ij}$ and for this reason they are particularly useful as interpolation basis. In fact, any quantity of interest $\mathbf{u}_h$ is coincident with the expansion coefficient $\mathbf{U}_{ij}$ and thus straightforwardly available at each nodal point.

The effectiveness of the proposed basis functions has been demonstrated by comparison with nodal expansion bases defined on equispaced nodes, by considering the operation count of functional and derivatives evaluation. Results are summarized in Table 1. For further details see [3].

Table 1 Operation counts of functions and derivatives evaluation at node A for co-located tensor product basis functions (sDGm) and for nodal basis function defined on equispaced nodes (std-DGm). $k-1$ represents the degree of both polynomial approximations.

	sDGm	std-DGm
$\mathbf{u}_h(\mathbf{x},t)\|_A$	1	k^2
$\frac{\partial \mathbf{u}_h}{\partial \xi}\|_A$	k	k^2
$\frac{\partial \mathbf{u}_h}{\partial x}\|_A$	$2k$	$2k^2$
$\nabla \mathbf{u}_h\|_A$	$4k$	$4k^2$

4 The p-Multigrid Algorithm

Standard iterative solvers are very effective at eliminating the high-frequency or oscillatory components of the error in the first iterations, while leaving the low-frequency or smooth components relatively unchanged. This behaviour produces a fast deterioration of the convergence rate. The basic idea of the h-multigrid method is to accelerate the convergence of these methods by adopting a sequence of progressively coarser grids, which allow for an effective reduction of the solution error over the entire frequency field. The p-MG approach is based on the same concepts as the standard h-multigrid method except that lower-order approximations on a single grid serve as coarse levels.

The various levels can be visited following different paths (V-cycle, W-cycle). At each level, a number ν_1 of pre-smoothing iterations is performed prior to restricting the solution to the next coarser level (bullets), while, on the way back to finer level, a number of ν_2 post-smoothing iterations is performed after prolongation (circle). An improvement of the algorithm is the full multigrid (FMG) strategy which exploits the coarser level solution to obtain good initial guess to initialize the computation on the finer grid. In the proposed algorithm the FMG V-cycle depicted in Figure 1 has been adopted and the solution at each level is prolongated to the finer level when a selected residual-based criterion is met.

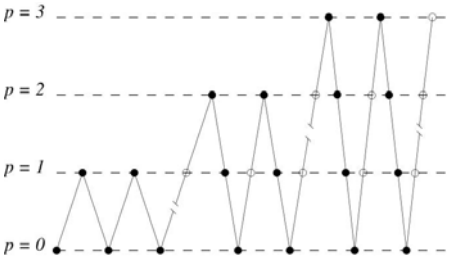


Fig. 1 V cycle full multigrid of $\mathbb{P}^3$ approximation (•: pre-smoothing; ○: post-smoothing)

4.1 The p-Multigrid FAS Scheme

The entire multigrid strategy is based on a recursive application of the so-called two-level algorithm—in which the "exact" solution on the coarser grid is used to accelerate the solution on the finer grid. In practice, to avoid the prohibitively expensive exact solution on the coarse grid, the two level algorithm is recursively applied to progressively coarser grids. The version of the two level scheme used for nonlinear problem is called Full Approximation Scheme (FAS). In order to illustrate FAS, a generic non linear problem $\mathbf{A}^p(\mathbf{u}^p) = \mathbf{b}^p$ is considered, where $\mathbf{u}^p$

is the exact solution vector of a given approximation order, $\mathbf{A}^p(\mathbf{u}^p)$ is the associated nonlinear algebraic operator and the superscript p indicates the polynomial degree of the approximation. Let $\mathbf{v}^p$ be an approximation to the exact solution $\mathbf{u}^p$ and $\mathbf{r}^p(\mathbf{v}^p) = \mathbf{b}^p - \mathbf{A}^p(\mathbf{v}^p)$ the discrete residual vector.

In the basic two level FAS algorithm, the exact solution on the coarse level is used to correct the solution on the fine level. The correction is performed according to the following steps:

1. restrict the solution and the residual to the coarse level,

$$\mathbf{v}_0^{p-1} = \widetilde{\mathbf{I}}_p^{p-1}\mathbf{v}^p, \quad \mathbf{r}^{p-1} = \mathbf{I}_p^{p-1}\mathbf{r}^p(\mathbf{v}^p), \tag{7}$$

   where $\widetilde{\mathbf{I}}_p^{p-1}$ and $\mathbf{I}_p^{p-1}$ are the solution and the residual restriction operators from level p to level $p-1$, respectively, to be defined in the following;
2. compute the forcing term for the coarse level:

$$\mathbf{s}^{p-1} = \mathbf{A}^{p-1}(\mathbf{v}_0^{p-1}) - \mathbf{r}^{p-1}; \tag{8}$$

3. solve the coarse level problem:

$$\mathbf{A}^{p-1}(\mathbf{v}^{p-1}) = \mathbf{I}_p^{p-1}\mathbf{b}^p + \mathbf{s}^{p-1}; \tag{9}$$

4. calculate the coarse grid error:

$$\mathrm{e}^{p-1} = \mathbf{v}^{p-1} - \mathbf{v}_0^{p-1}; \tag{10}$$

5. prolongate the coarse grid error and correct the fine level approximation:

$$\mathbf{v}^p = \mathbf{v}^p + \widetilde{\mathbf{I}}_{p-1}^p \mathrm{e}^{p-1} \tag{11}$$

   where $\widetilde{\mathbf{I}}_{p-1}^p$ is the error prolongation operator.

The solution/error restriction and prolongation operators, $\widetilde{\mathbf{I}}_p^{p-1}$ and $\widetilde{\mathbf{I}}_{p-1}^p$, are simply L^2 projections onto the low-order and high-order spaces, respectively. An explicit expression of the residual restriction operator $\mathbf{I}_p^{p-1}$ can be obtained following the approach proposed by Fidkowski, see e.g. [13], which shows that $\mathbf{I}_p^{p-1} = (\widetilde{\mathbf{I}}_{p-1}^p)^T$.

The orthonormal and hierarchical basis functions implemented in the MIGALE code allow to simply define the solution restriction and prolongation operators as:

$$\widetilde{\mathbf{I}}_p^{p-1} = \delta_{p,p-1}, \qquad \widetilde{\mathbf{I}}_{p-1}^p = \delta_{p-1,p}. \tag{12}$$

In practice the DOFs of the restricted solution are equal to the low-order subset of the high-order solution, and the low-order subset of the prolongated error are the same as the low-order error with null high-order DOFs.

4.2 Semi-implicit Runge-Kutta Smoother

At each level $p_{min} < p \leq p_{max}$ the p-MG scheme presented in this work employs as smoother the five stage semi-implicit RK scheme introduced in [5], which can be written as:

$$\begin{aligned}
&\mathbf{u}^0 = \mathbf{u}^n \\
&\texttt{DO } k = 1,\ m \\
&\qquad \left[\mathbf{M} + \alpha_k \Delta t D(\mathbf{u}^0)\right] \delta\mathbf{u}^k = -\mathbf{M}(\mathbf{u}^{k-1} - \mathbf{u}^0) - \alpha_k \Delta t \mathbf{r}(\mathbf{u}^{k-1}) \\
&\qquad \mathbf{u}^k = \mathbf{u}^{k-1} + \delta\mathbf{u}^k \\
&\texttt{END DO} \\
&\mathbf{u}^{n+1} = \mathbf{u}^m,
\end{aligned} \tag{13}$$

where $D(\mathbf{u}^0)$ is the block diagonal part of the full Jacobian matrix, which is computed only at the first stage.

4.3 Backward Euler Smoother

At level p_{min} a linearized backward Euler iterative smoother is employed,

$$\left(\frac{\mathbf{M}^{p_{min}}}{\Delta t} + \frac{\partial \mathbf{R}\left(\mathbf{u}^{p_{min}}\right)}{\partial \mathbf{u}}\right) \Delta \mathbf{u}^{p_{min}} + \mathbf{R}\left(\mathbf{u}^{p_{min}}\right) = 0, \tag{14}$$

where $\mathbf{M}^{p_{min}}$ denotes the mass matrix, $\mathbf{u}^{p_{min}}$ the vector of the unknowns and $\mathbf{R}^{p_{min}}$ the residuals vector at level p_{min}. The fully coupled linear system is solved by means of the GMRES algorithm and the incomplete LU factorization preconditioner.

5 Numerical Results

This section presents the results for two shockless test cases, the inviscid and turbulent flow around BTC0 three dimensional body. The inviscid test case has been computed with the spectral DG method based on the nodal basis functions described in Section 3 (sDGm). The fully coupled linear system is solved by means of the GMRES algorithm and the incomplete LU factorization preconditioner. The sDGM computational efficiency has been compared with a DG method based on nodal basis functions defined on equispaced points (std-DGm). The proposed basis functions have been also implemented in the p-MG algorithm presented in [4], and the effect of the coupling has been evaluated. The spectral p-MG is based on a FMG V-cycle, a five stage semi-implicit RK smoother for $\mathbb{P}^k$ polynomial approximations (if $p_{min} < k \leq p_{max}$) and the implicit backward Euler smoother for $\mathbb{P}^{p_{min}}$ (with $p_{min} = 0$) polynomial approximation (S-SIRK5+BE).

The turbulent test case has been computed to assess the performance of the p-MG algorithm described in Section 4 (SIRK5+BE). The coarsest level value p_{min}

has been taken equal to $\mathbb{P}^0$ and the coefficients α_i of the five stages semi-implicit RK smoother have been taken from [5], i.e., $\alpha_{i=1,\ldots,5} = \left\{\frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1\right\}$. The value of the pre/post smoothing iterations has been empirically determined in order to minimize the CPU time needed to reach a converged solution.

5.1 Spectral DG results

In this section the inviscid flow around BTC0 three dimensional body is computed for a farfield Mach number $M_\infty = 0.5$, and an angle of attack $\alpha = 1^\circ$. The geometry has been represented with bi-quadratic faces and a 768 hexahedral elements grid has been used. In Figure 2 the mesh (left) and the corresponding Mach isolines (right) of a $\mathbb{P}^5$ solution are depicted.

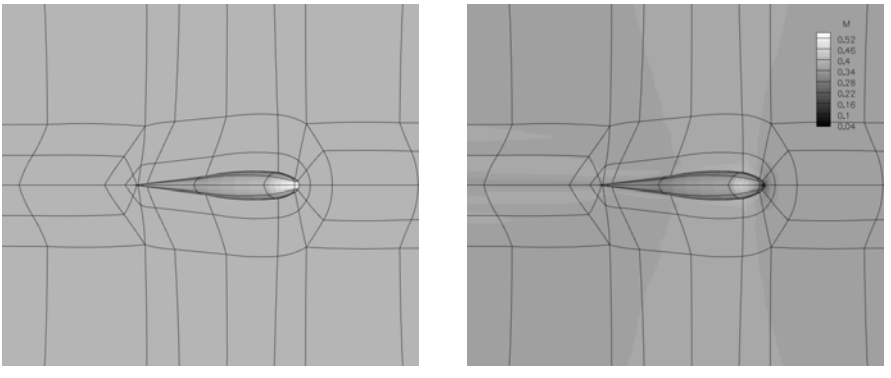


Fig. 2 Inviscid BTC0: 768 hexahedral $\mathbb{P}^2$ elements mesh (left) and Mach isolines of $\mathbb{P}^5$ solution (right)

The sDGm has been compared with std-DGm, evaluating the CPU time needed to converge. Table 2 reports the CPU time and memory requirement for different solution approximations $\mathbb{P}^k$ $(k = 2, \ldots, 4)$). It can be observed that the sDGm achieves a $10 - 15\%$ computational time reduction due to the speed-up of the matrices assembling. The effectiveness of the approximation based on co-located tensor product basis functions can be even more significant for methodologies where the time spent in the DG discrete operator assembly "dominates" the overall computational time effort, *e.g.* , *p*-MG solvers. The effect of the *p*-MG coupling with collocated tensor product basis functions has been investigated, comparing the spectral *p*-MG and the sDGm (S-IMP). Figure 3 (left) illustrates the density residual L^2 norm convergence history of both strategies for a $\mathbb{P}^4$ solution approximation, while Table 3 shows the CPU time and the memory required. The S-SIRK5+BE achieves a 25.4% CPU time reduction and 87% memory saving with respect to S-IMP.

Table 2 Inviscid BTC0: CPU time and memory requirement for $\mathbb{P}^k$ solutions using sDGm and std-DGm

$\mathbb{P}^k$	Memory [GB]	$t_{std-DGm}$ [s]	t_{sDGm} [s]
2	1.5	3.0×10^2	2.7×10^2
3	8.2	2.0×10^3	1.8×10^3
4	30	8.1×10^3	7.1×10^3

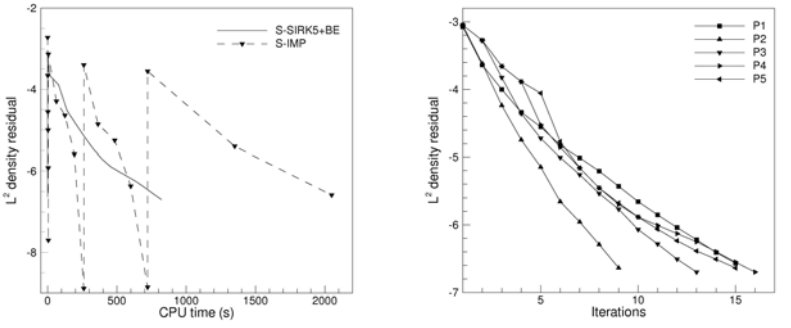


Fig. 3 Inviscid BTC0: density residual L^2 norm convergence history versus CPU time for a $\mathbb{P}^4$ solutions with S-IMP and S-SIRK5+BE (left). Density residual L^2 norm convergence history versus FMG cycle for different soluttion approximations $\mathbb{P}^k$ ($k = 1,\ldots,5$) with S-SIRK5+BE (right).

Table 3 Inviscid BTC0: convergence results obtained with S-SIRK5+BE and S-IMP. ν_c is the number of smoothing iterations at coarsest level p_{min}. $\nu_{1,2}$ is the number of pre/post smoothing iteration at level $p_{min} < p < p_{max}$. ν_f is the number of smoothing iteration at the finest level p_{max}. N_{it} is the number of iterations needed to reach a converged solution (it can represents both the FMG iterations or implicit scheme iterations). Mem is the memory requirement. t is the computational time.

Scheme	ν_c	$\nu_{1,2}$	ν_f	N_{it}	Mem [GB]	t [s]
S-SIRK5+BE	1	1-1	1-1	47	3.8	5.3×10^3
S-IMP	-	-	-	19	30	7.1×10^3

On the right side of Figure 3 the density residual L^2 norm convergence history obtained with the spectral p-MG algorithm for different solution approximation $\mathbb{P}^k$ ($k = 1,\ldots,5$) is illustrated. Table 4 resumes the number of multigrid (MG) iterations needed to converge and the slope of the linear regression of each convergence curve σ, showing the nearly p-independence property of the multigrid scheme.

Table 4 Inviscid BTC0: MG iterations needed to reach a converged solution and slope of the convergence curve of $\mathbb{P}^k$ solutions

$\mathbb{P}^k$	1	2	3	4	5
N_{MG}	15	8	11	13	11
σ	-0.23	-0.42	-0.27	-0.21	-0.23

5.2 p-Multigrid DG Results

In the second test case the turbulent flow around BTC0 three dimensional body is computed for a farfield Mach number $M_\infty = 0.5$, an angle of attack $\alpha = 5^\circ$ and a Reynolds number $Re = 10^7$. The geometry has been represented with bi-quadratic faces and two grids have been considered. The coarse grid is composed of 832 hexahedral elements with a maximum stretching factor value equal to 8850, while the fine grid has 7937 hexahedral elements with a maximum stretching factor value equal to 20000. Figure 4 shows details of the fine grid, while Figure 5 illustrates the corresponding pressure (left) and turbulence intensity (right) contours for a $\mathbb{P}^3$ spatial discretization. The comparison between the SIRK5+BE and the implicit MI-

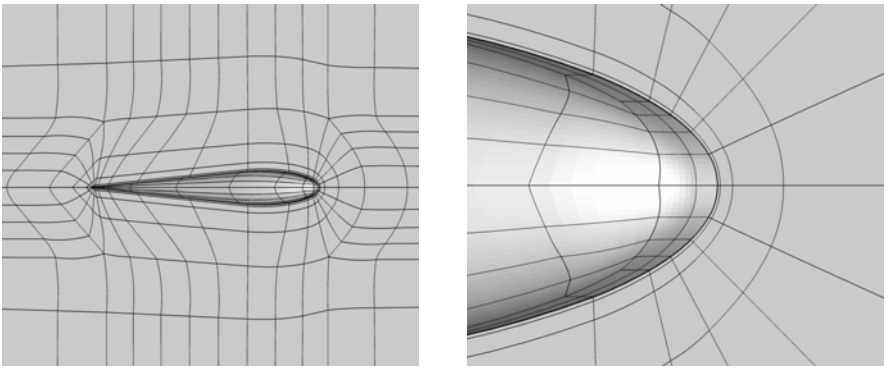


Fig. 4 Turbulent BTC0: 7937 hexahedral elements mesh

GALE code (IMP) on the coarse mesh has been presented in Figure 6 (left), while complete results have been resumed in Table 5. It can be seen that the SIRK5+BE algorithm turns out to be less efficient than the IMP scheme in term of CPU time but it shows a marked reduction of the memory requirement (70.7%). Figure 6 (right) illustrates the density residual L^2 norm convergence history as a function of the FMG iterations of S-SIRK5+BE scheme for different solution approximation $\mathbb{P}^k$ ($k = 1, \ldots, 3$), showing that the p-MG algorithm satisfies perfectly the polynomial order independent property.

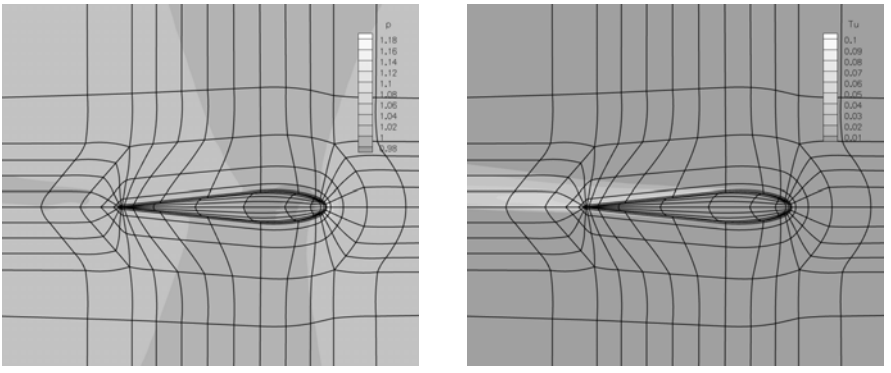


Fig. 5 Turbulent BTC0: pressure (left) and turbulence intensity (right) contours of a $\mathbb{P}^3$ solution on the fine mesh

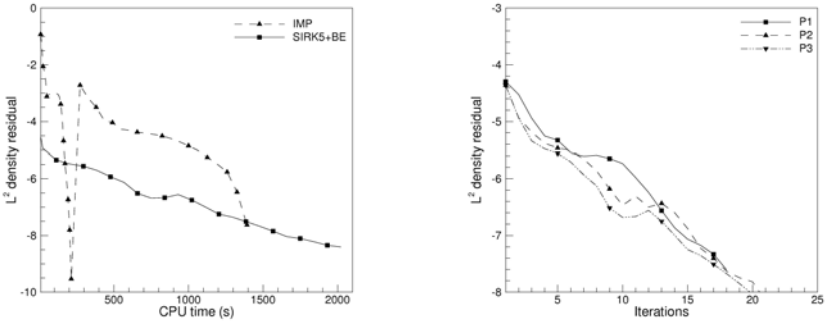


Fig. 6 Turbulent BTC0: density residual L^2 norm convergence history versus CPU time on the coarse mesh for different solution strategies. Dashed line: IMP scheme. Solid line: SIRK5+BE scheme (left). Density residual L^2 norm convergence history of the SIRK5+BE as a function of FMG cycles on the coarse mesh for different solution approximations $\mathbb{P}^k$ (right).

Table 5 Turbulent BTC0: convergence results obtained with p-multigrid algorithm (SIRK5+BE) and implicit MIGALE code (IMP). ν_c is the number of smoothing iterations at coarsest level p_{min}. $\nu_{1,2}$ is the number of pre/post smoothing iteration at level $p_{min} < p < p_{max}$. ν_f is the number of smoothing iteration at the finest level p_{max}. N_{it} is the number of iteration needed to reach a converged solution (it can represents both the FMG iteration and implicit scheme iteration). Mem is the memory requirement. t is the computational time.

Scheme	ν_c	$\nu_{1,2}$	ν_f	N_{it}	Mem [MB]	t [s]
SIRK5+BE	1	2-1	1	46	516	2004
IMP	-	-	-	44	1760	1392

6 Conclusions

Two different approaches to enhance the computational efficiency of DG methods have been analysed. In particular a DG discretization based on co-located tensor product basis functions has been presented, which allows to speed-up the DG discrete operator assembly. The *p*-multigrid algorithm is the second approach proposed, characterized by a marked reduction of the memory requirement with respect to implicit scheme. The coupling of these two strategies has shown great potentiality, exploiting the advantages typical of both approaches and making the spectral *p*-multigrid algorithm performances comparable with implicit schemes, both in term of memory requirement and CPU time.

References

1. Arnold, D.N., Brezzi, F., Cockburn, B., Marini, D.: Unified analysis of discontinuous Galerkin methods for elliptic problems. SIAM J. Numer. Anal. 39(5), 1749–1779 (2002)
2. Bassi, F., Crivellini, A., Rebay, S., Savini, M.: Discontinuous Galerkin solution of the Reynolds averaged Navier-Stokes and k – ω turbulence model equations. Computers & Fluids 34, 507–540 (2005)
3. Bassi, F., Franchina, N., Ghidoni, A., Rebay, S.: Spectral discontinuous Galerkin solution of the Navier-Stokes equations. Int. J. Numer. Meth. Fluids (2009) (submitted)
4. Bassi, F., Franchina, N., Ghidoni, A., Rebay, S.: Spectral p-multigrid discontinuous Galerkin solution of the Navier-Stokes equations. Int. J. Numer. Meth. Fluids (2009) (submitted)
5. Bassi, F., Ghidoni, A., Rebay, S., Tesini, P.: High-order accurate *p*-multigrid discontinuous Galerkin solution of the Euler equations. Int. J. Numer. Meth. Fluids 60(8), 847–865 (2009)
6. Bassi, F., Rebay, S.: A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. J. Comput. Phys. 131, 267–279 (1997)
7. Bassi, F., Rebay, S.: High-order accurate discontinuous finite element solution of the 2D Euler equations. J. Comput. Phys. 138, 251–285 (1997)
8. Bassi, F., Rebay, S.: GMRES discontinuous Galerkin solution of the compressible Navier–Stokes equations. In: First International Symposium on Discontinuous Galerkin Methods on Discontinuous Galerkin Methods. Theory, Computation and Applications, Newport, RI, USA, May 24–26. Lecture Notes in Computational Science and Engineering, vol. 11, Springer, Heidelberg (1999)
9. Bassi, F., Rebay, S.: A high order discontinuous Galerkin method for compressible turbulent flows. In: First International Symposium on Discontinuous Galerkin Methods on Discontinuous Galerkin Methods. Theory, Computation and Applications, Newport, RI, USA, May 24–26. Lecture Notes in Computational Science and Engineering, vol. 11, Springer, Heidelberg (2000)
10. Bassi, F., Rebay, S.: Numerical solution of the Euler equations with a multiorder discontinuous finite element method. In: Armfield, S., Morgan, P., Srinivas, K. (eds.) Computational Fluid Dynamics 2002: Proceedings of the Second International Conference on Computational Fluid Dynamics, Sydney, pp. 199–204. Springer, Heidelberg (2002)

11. Bassi, F., Rebay, S., Mariotti, G., Pedinotti, S., Savini, M.: A high-order accurate discontinuous finite element method for inviscid and viscous turbomachinery flows. In: Decuypere, R., Dibelius, G. (eds.) 2nd European Conference on Turbomachinery Fluid Dynamics and Thermodynamics, Antwerpen, Belgium, March 5-7, pp. 99–108. Technologisch Instituut. (1997)
12. Brezzi, F., Manzini, G., Marini, D., Pietra, P., Russo, A.: Discontinuous Galerkin approximations for elliptic problems. Numer. Meth. for Part. Diff. Eq. 16, 365–378 (2000)
13. Fidkowski, K.J., Oliver, T.A., Lu, J., Darmofal, L.: *p*-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. Journal of Computational Physics 207(1), 92–113 (2005)
14. Kroll, N.: ADIGMA-A European project on the development of adaptive higher-order varational methods for aerospace applications. AIAA Paper 2009-176, AIAA (2009)
15. Warburton, T.C., Lomtev, I., Du, Y., Sherwin, S.J., Karniadakis, G.E.: Galerkin and discontinuous Galerkin spectral/hp methods. Computer methods in applied mechanics and engineering 175, 343–359 (1999)

This page intentionally left blank

Chapter 21
Agglomeration Multigrid for the Vertex-Centered Dual Discontinuous Galerkin Method

Sven-Erik Ekström and Martin Berggren

Abstract. Agglomoration multigrid is used in many finite-volume codes for aerodynamic computations in order to reduce solution times. We show that an existing agglomeration multigrid solver developed for equations discretized with a vertex-centered, edge-based finite-volume scheme can be extended to accelerate convergence also for a vertex-centered discontinuous Galerkin method. Preliminary results for a subsonic as well as a transonic test case for the Euler equations in two space dimensions show a significant convergence acceleration for the discontinuous Galerkin equations using the agglomoration multigrid strategy.

1 Introduction

The particular discontinuous Galerkin (DG) method studied and implemented by Uppsala University within the ADIGMA project is vertex centered, in contrast to the standard cell-centered DG method. The method was introduced by Berggren *et al.* [1, 2] for model problems and for the Euler equation in our other contribution to this volume [4]. The method is designed to constitute a generalization to higher order of the edge-based, vertex-centered finite volume (FV) discretization that is particularly popular in many of the codes in engineering practice. Our implementation is done within a software system of this type, Edge [5, 6].

The use of multigrid is a common and often successful strategy for convergence acceleration in FV solvers. We here describe how an existing agglomerated

Sven-Erik Ekström
Department of Information Technology, Uppsala University, Box 337, SE-751 05 Uppsala, Sweden
e-mail: sven-erik.ekstrom@it.uu.se

Martin Berggren
Department of Computing Science, Umeå University, SE-901 87 Umeå, Sweden
e-mail: martin.berggren@cs.umu.se

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 301–308, 2010.
springerlink.com

multigrid facility in Edge is adapted to support multigrid also for the DG discretization. To explain the multigrid approach, we adopt a notation similar to the one used in the Edge documentation [6].

We want to solve the steady state problem

$$\nabla \cdot \mathscr{F}(U) = 0 \quad \text{in } \Omega, \tag{1}$$

where $\mathscr{F}$ is the standard flux function for the Euler equations [3, 4], and $U = [\rho, \rho\mathbf{u}, \rho E]^T$ of dimension $d+2$ (d is the space dimension) is the vector of conservative variables. Let each components of U_h, the numerical solution to equation (1), belong to the space V_h specified in [4]. The DG method is obtained by multiplying equation (1) by a test function vector $V_h \in \mathscr{V}_h^{d+2}$, integrating over each control volume, integrating by parts, and introducing the numerical flux $\mathscr{F}^*$ on the boundaries. This procedure yields that $U_h \in \mathscr{V}_h^{d+2}$ solves the variational problem

$$\int_{\partial K_m} V_h \cdot \mathscr{F}^*(U_L, U_R, \hat{\mathbf{n}})\,\mathrm{d}s - \int_{K_m} \nabla V_h \cdot \mathscr{F}(U_h)\,\mathrm{d}V = 0, \ \forall K_m \subset \Omega, \quad \forall V_h \in \mathscr{V}_h^{d+2}, \tag{2}$$

where subscripts L and R denote local ("left") and remote ("right") values on the boundary ∂K_m of control volume K_m, and $\hat{\mathbf{n}}$ is the outward unit normal.

Variational expression (2) defines a nonlinear equation $\mathscr{N}(U_h) = 0$. To solve this equation, a common approach in the the CFD community is to time march equation

$$\frac{\partial U_h}{\partial t} + \mathscr{N}(U_h) = 0 \tag{3}$$

to steady state using Runge–Kutta time stepping. For efficiency, this procedure needs to be accelerated by for example local time-stepping and multigrid, the latter which is the subject of this chapter. We start in Section 2 with a brief description of the existing agglomeration multigrid method as implemented in Edge. In Section 3, we explain how the method has been extended to encompass also the vertex-centered DG discretization. Sections 4 presents preliminary results of our DG agglomoration multigrid approach, and we end with some concluding remarks in Section 5.

2 Existing Finite Volume Agglomeration Multigrid in Edge

For use in the agglomoration multigrid process, the preprocessor of Edge generates a sequence of L coarser and coarser dual meshes. First, a dual mesh (top right in Figure 1) is generated from the primal mesh (top left in Figure 1). The data structures needed to represent the dual mesh is a list of edges connecting adjacent vertices in the primal mesh and a list, associated with these edges, of normals to the dual control volumes. (Additional boundary information is also required [4], but for simplicity not considered here).

The meshes are indexed coarse to fine by $1, \ldots, L$, where mesh L refers to the dual mesh generated from the supplied primal mesh. To generate a coarser mesh, a set of adjacent dual cells are agglomerated into bigger cells, as shown in the bottom left of Figure 1; this mesh would be number $L-1$. The number of degrees of freedom is equal to the number of cells, hence less for the coarser mesh, and the location of each new node is computed by a weighted average of the nodes in the dual cells (of the finer mesh) that constitute the agglomerated cell. A new list of edges connecting nodes in the agglomerated mesh is generated, and new normals are constructed by vectorially adding normals from the finer mesh. In the bottom right of Figure 1, yet another agglomeration is shown, with even fewer elements, using the same procedure to generate a new list of edges and normals; this mesh would be numbered $L-2$.

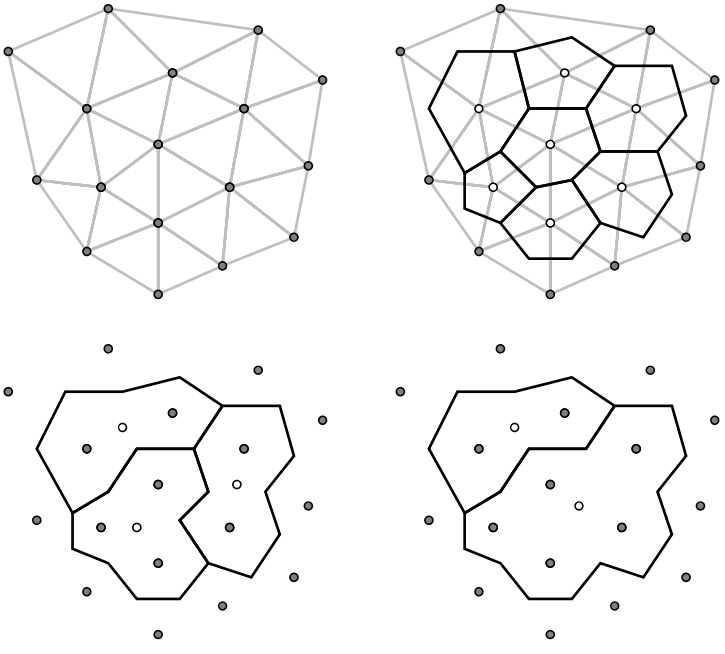


Fig. 1 The preprocessor stages to generate computational meshes. Top left: The primal mesh. Vertices are marked with gray circles. Top right: The dual mesh. The degrees of freedom for the unknowns are associated with the nodes (located at the primal mesh vertices) marked with white circles. The data structures involve lists, associated with the edges of the primal mesh, of (pair of) node numbers and the normals of the dual cells. This is level L in the agglomerated multigrid cascade of L meshes. Bottom left: Agglomeration one step, combining adjacent dual cells to bigger cells. The (white) node in each agglomerated cell is weighted together from the (gray) nodes in the finer cells constituting the agglomerated cell. A single normal for each edge in the agglomerated mesh is constructed by vectorially adding normals from the associated part of the cell boundary. This level would be $L-1$. Bottom right: Agglomeration one more level with even bigger cells, at level $L-2$.

The discrete Euler or Navier–Stokes equations on the finest mesh can be written

$$\frac{\partial U^L}{\partial t} + \mathcal{N}_L(U^L) = 0, \tag{4}$$

where $\mathcal{N}_L(U^L)$ is the spatial discretization of the equations on the finest mesh, here the FV discretization (or lowest order DG, $p = 0$). At each coarser mesh level $l < L$, we have

$$\frac{\partial U^l}{\partial t} + \mathcal{N}_l(U^l) = F_l, \tag{5}$$

with initial value $U^l = I^l_{l+1} U^{l+1}$, where I^l_{l+1} is the restriction operator for the unknowns, and F_l is a forcing function, defined recursively as

$$F_l = \mathcal{N}_l(I^l_{l+1} U^{l+1}) + \hat{I}^l_{l+1} \left[F_{l+1} - \mathcal{N}_{l+1}(U^{l+1}) \right], \tag{6}$$

with $F_L = 0$. Moreover, I^l_{l+1} and $\hat{I}^l_{l+1}$ are restriction operators of respectively the unknowns and the residuals from finer mesh $l+1$ to coarser mesh l. These are defined cell by cell by summing contributions from the subcells on mesh level $l+1$ that constitute the cell on mesh level l. The restriction of $\hat{I}^l_{l+1}$ to cell K_m at mesh level l is given by

$$\hat{I}^l_{l+1}|_m R^{l+1} = \sum_n R^{l+1}_n, \tag{7}$$

where R^{l+1} is the vector of residuals with components R^{l+1}_n associated with cell K_m. The restriction of I^l_{l+1} to cell K_m is similarly defined, but with a weighting involving the cell volumes:

$$I^l_{l+1}|_m U^{l+1} = \frac{\sum_n \mathcal{V}^{l+1}_n U^{l+1}_n}{\sum_n \mathcal{V}^{l+1}_n} \tag{8}$$

in which $\mathcal{V}^{l+1}_n$ is the cell volume of a subcell to K_m.

Smoothing is accomplished at each level by integrating (4) and (5) with a few Runge–Kutta steps, typically with local time stepping. When the solution has been smoothed on coarsest mesh, $l = 1$, the following prolongation scheme successively provides updated solutions $\bar{U}^l$, for $l > 1$:

$$\bar{U}^l = U^l + I^l_{l-1}(U^{l-1} - I^{l-1}_l U^l). \tag{9}$$

Operator I^l_{l-1} is the prologation operator (a simple injection operator), from mesh level $l-1$ to mesh level l. For the standard V, W, and F cycle schemes, smoothing is performed on the updated solution $\bar{U}^l$ before proceeding to even finer meshes.

Different choices of fluxes can be chosen by the user on coarser meshes to reduce the computational complexity for each multigrid sweep.

Full multigrid is also available within Edge. The calculations then start at the coarsest mesh and continue until convergence, that is, until the residual is lower

than a user-supplied threshold. After this, the solution is prolonged to the next finer mesh level. Two-grid cycles are now used until convergence is attained again. Then the solution is prolonged to the third mesh level and a three-grid cycle is used until convergence. This procedure is repeated until all meshes are involved and solution is given on the finest mesh. User-defined variables control the behavior of the solver, for example the maximum number of cycles spent in each stage of the full multigrid.

3 Discontinuous Galerkin Agglomeration Multigrid

In this section, we discuss how we adapted the Edge code to support multigrid also for the DG discretization.

The equation to solve at the finest mesh level L and for order p is

$$\frac{\partial U_p^L}{\partial t} + \mathscr{N}_L^p(U_p^L) = 0, \tag{10}$$

where $\mathscr{N}_L^p$ is the spatial discretization operator defined by equation (2). The multigrid process acts through repeated smoothing, that is, by applying a few Runge–Kutta iteration steps on equation (10), with successively modified initial conditions. These initial conditions are obtained recursively by similar smoothing procedures on lower-order approximations and on cruder meshes.

Assume now that $\bar{U}_p^L$ is the result of applying a few Runge–Kutta iteration steps to equation (10). At next multigrid level, we integrate on the same mesh, but at lowest order $p = 0$:

$$\frac{\partial U_0^L}{\partial t} + \mathscr{N}_L(U_0^L) = F_L^0, \tag{11}$$

where

$$F_L^0 = \mathscr{N}_L^0(J_p^0 \bar{U}_p^L) - \hat{J}_p^0 \left[\mathscr{N}_L^p(\bar{U}_p^L)\right], \tag{12}$$

and where J_p^0 and $\hat{J}_p^0$ are the restriction operators for the solution and the residual, respectively, as defined below. Since $p = 0$ is nothing else but an vertex-centered finite-volume scheme, equation (11) can be integrated using the agglomeration multigrid scheme in Edge outlined in section 2. Let $\bar{U}_0^L$ be the result of such an integration. Then we may define

$$\bar{\bar{U}}_p^L = \bar{U}_p^L + J_0^p \left(\bar{U}_0^L - J_p^0 \bar{U}_p^L\right) \tag{13}$$

and use $\bar{\bar{U}}_p^L$ as a new initial condition for equation (10). The prolongation operator J_0^p is specified below. For orders $p > 1$, it may also be advantageous to add a level of p-multigrid instead of projecting directly on $p = 0$ as described above. We have not yet implemented or tested such a general p multigrid strategy.

In order to specify the restriction and prolongation operators, let

$$U_q^L|_m = \underbrace{\begin{pmatrix} \rho_1^m & (\rho\mathbf{u})_1^m & (\rho E)_1^m \\ \rho_2^m & (\rho\mathbf{u})_2^m & (\rho E)_2^m \\ \vdots & \vdots & \vdots \\ \rho_{N_m}^m & (\rho\mathbf{u})_{N_m}^m & (\rho E)_{N_m}^m \end{pmatrix}}_{[N_m \times d+2]}, \tag{14}$$

be the piece, associated with macro element K_m, of the solution vector $U_q^L$of order q at the finest mesh level L. The number of degrees of freedom N_m depends on the order; $N_m = 1$ for $q = 0$, for instance. The restriction operator J_p^0 for the solution is block diagonal on each piece $U_p^L|_m$. The diagonal blocks $J_p^0|_m$ are defined by

$$U_0^L|_m = \underbrace{(\mathbf{M}^{00})^{-1}\mathbf{M}^{0p}}_{J_p^0|_m} U_p^L|_m \tag{15}$$

where $\mathbf{M}^{00}$ and $\mathbf{M}^{0p}$ are the mass matrices

$$\mathbf{M}^{00} = \int_{K_m} \mathrm{d}V, \tag{16}$$

$$\mathbf{M}^{0p} = \underbrace{\left(\int_{K_m} \phi_1 \,\mathrm{d}V, \ldots, \int_{K_m} \phi_{N_m} \,\mathrm{d}V \right)}_{[1 \times N_m]}. \tag{17}$$

The restriction operator for the residuals, $\hat{J}_p^0$, is also block diagonal. The diagonal blocks $\hat{J}_p^0|_m$ are defined by

$$\hat{J}_p^0|_m = \mathbf{M}^{0p} \left(\mathbf{M}^{pp}\right)^{-1}, \tag{18}$$

where

$$\mathbf{M}^{pp} = \underbrace{\begin{pmatrix} \int_{K_m} \phi_1\phi_1 \,\mathrm{d}V & \ldots & \int_{K_m} \phi_1\phi_{N_m} \,\mathrm{d}V \\ \vdots & & \vdots \\ \int_{K_m} \phi_{N_m}\phi_1 \,\mathrm{d}V & \ldots & \int_{K_m} \phi_{N_m}\phi_{N_m} \,\mathrm{d}V \end{pmatrix}}_{[N_m \times N_m]}. \tag{19}$$

Finally, each diagonal block of the prolongation operator J_0^p, is given by

$$J_0^p = (\mathbf{M}^{pp})^{-1}\mathbf{M}^{p0}, \tag{20}$$

where $\mathbf{M}^{p0} = \left(\mathbf{M}^{0p}\right)^T$.

4 First Results

In Figures 2 and 3, we show results for two Mandatory Test Cases of the ADIGMA project, MTC1 and MTC2. The left displays in Figures 2 and 3 depict the pressure coefficient for linear elements for the respective test cases. (The oscillations in the pressure coefficient at the leading edge for the subsonic case vanish when using higher-order elements, as shown in our other contribution to this volume [4].) The right displays in Figures 2 and 3 show the iteration histories, in terms of the mass conservation residuals, when solving MTC1 and MTC2 without and with multigrid. The multigrid computations are initiated with FV solutions obtained with full multigrid, and two grid levels are used in addition to the projection from $p = 1$ to $p = 0$. A clear reduction of number of iteration is attained in both test cases, although the code has not yet been tuned and optimized in any way.

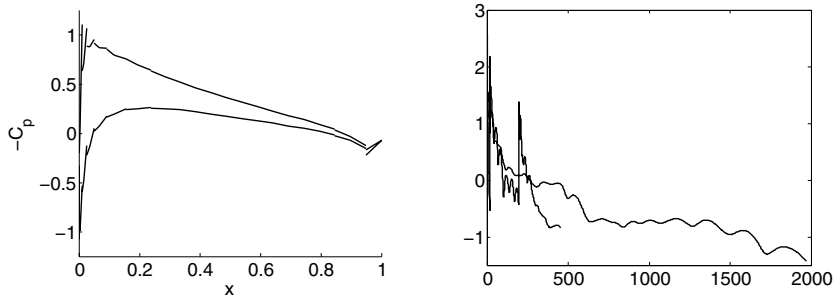


Fig. 2 MTC1 of the ADIGMA project. Left: Pressure coefficient, $p = 1$, $2\mathbb{D}$ Euler, NACA0012, $M = 0.5$, $\alpha = 2.0°$. Right: Mass conservation ("ρ") residual reduction without multigrid and with initial full FV multigrid followed by DG with agglomeration multigrid.

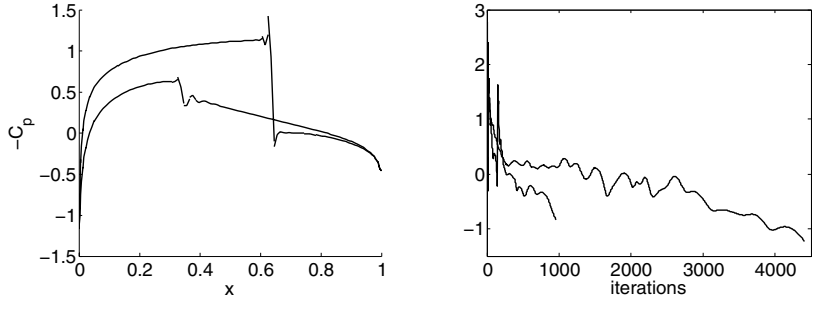


Fig. 3 MTC2 of the ADIGMA project. Left: Pressure coefficient, $p = 1$, $2\mathbb{D}$ Euler, NACA0012, $M = 0.8$, $\alpha = 1.25°$. Right: Mass conservation ("ρ") residual reduction without multigrid and with initial full FV multigrid followed by DG with agglomeration multigrid.

5 Conclusions

Agglomeration multigrid is a common strategy to accelerate convergence in vertex-centered finite-volume schemes. The implementation of a robust and effective agglomeration multigrid algorithm requires a substantial man-power investment. The availability of agglomeration multigrid in Edge was an important motivation to implement of our vertex-centered DG scheme inside Edge instead of attempting an implementation from scratch. To the best of our knowledge, this is the first time that such an extension effort has been attempted.

A central issue has been positively resolved: agglomeration multigrid, as developed for vertex-centered finite-volume schemes, is indeed effective as a converge acceleration also for the vertex-centered DG method. These first results certainly motivate further study of the method. First, the implementation needs to be completed to accept arbitrary number of mesh levels and orders $p > 1$. Also, for $p > 1$, it should be investigated whether a p-multigrid layer should be included before agglomoration multigrid is activated.

References

1. Berggren, M.: A vertex-centered, dual discontinuous Galerkin method. J. Comput. Appl. Math. 192(1), 175–181 (2006)
2. Berggren, M., Ekström, S.-E., Nordström, J.: A discontinuous Galerkin extension of the vertex-centered edge-based finite volume method. Commun. Comput. Phys. 5, 456–468 (2009)
3. Blazek, J.: Computational Fluid Dynamics, 2nd edn. Elsevier, Amsterdam (2005)
4. Ekström, S.-E., Berggren, M.: Incorporating a discontinuous Galerkin method into the existing vertex-centered edge-based finite volume solver Edge. In: Kroll, N., et al. (eds.) ADIGMA. NNFM, vol. 113, pp. 39–52. Springer, Heidelberg (2010)
5. Eliasson, P.: EDGE, a Navier–Stokes solver, for unstructured grids. Technical Report FOI-R-0298-SE, Swedish Defence Research Agency (2001)
6. FOI. Edge - Theoretical Formulation. Technical Report FOI dnr 03-2870, Swedish Defence Research Agency (2007) ISSN-1650-1942

Chapter 22
Higher–Order Aerodynamic Computations Using an Edge Based Finite Volume Scheme

G. Campagne, O. Hassan, K. Morgan, and K.A. Sørensen

Abstract. Methods of improving the computational performance of conventional low order, unstructured grid, cell vertex, finite volume codes, for the simulation of high speed compressible viscous flow, are considered. The objective is to improve the performance without requiring that major changes be made to existing codes. With this in mind, higher order discretisations and improved equation solution has been attempted, within wall boundary layer regions only. This is possible, using the structure that is present in the grids normally used to discretize the boundary layer regions. A number of examples are included to illustrate the improvement in computational performance that can be obtained in this way.

1 Introduction

In the industrial environment, inviscid flow simulations over complex aerodynamic configurations are often now effectively, and economically, performed using an unstructured tetrahedral mesh and a low order, edge based, finite volume solution algorithm. Popular steady state algorithms generally employ a cell vertex formulation, with centered or upwind stabilisation, and explicit multi–stage time stepping, with agglomerated multigrid acceleration.

However, when the same techniques are applied to viscous flow simulations on hybrid meshes, practical experience has show that, for many simulations of industrial relevance, the computational performance of these methods deteriorates significantly. This deterioration is believed to be due to the additional stiffness that is

G. Campagne · O. Hassan · K. Morgan
Civil & Computational Engineering Centre, School of Engineering,
Swansea University, Swansea SA2 8PP, Wales, UK
e-mail: o.hassan@swansea.ac.uk

K.A. Sørensen
EADS–MAS, Aerodynamics & Methods, Rechliner Straße, 85077 Manching, Germany
e-mail: kaare.sorensen@eads.com

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 309–325, 2010.
springerlink.com

introduced into the problem by the presence of the viscous boundary layers. To adequately resolve the solution, the meshes employed for viscous flow simulations are normally required to have a large number of nodes, with highly stretched elements, located within the boundary layer region.

Within the ADIGMA project, we have investigated two approaches designed to improve the computational performance for viscous flow simulations, by attempting to improve the treatment of the boundary layer. A major objective was to achieve improvements within the context of techniques that could be readily implemented in the current generation of industrial finite volume codes. To avoid duplication of work, an investigation aimed at improving the basic efficiency of the solution procedure was undertaken by EADS–MAS and involved solving the governing equations implicitly along the wall–normal mesh lines. At Swansea, effort was focused on the introduction of one dimensional higher–order discretisations in the wall–normal

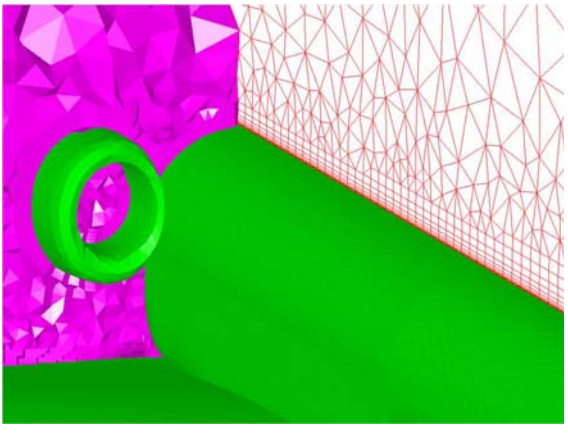


Fig. 1 Detail, on the symmetry plane, of a typical hybrid mesh generated for a viscous flow simulation over an aircraft configuration

direction. The objective here was to reduce the reduce the number of mesh points required to resolve the boundary layer [6]. At the conclusion of the project, the two developments were combined within a single 3D code, which was implemented by EADS–MAS.

2 Development of a Hybrid Solution Algorithm

2.1 Background

The FLITE system [18] at Swansea provides a facility for performing steady aerodynamic flow simulations. This system employs a cell vertex finite volume method for the solution of the conservative form of the unsteady compressible Navier Stokes equations on unstructured meshes. Stabilization, for transonic simulations, is accomplished by replacing the physical flux function with an unstructured mesh implementation of the well known JST consistent numerical flux function [15]. The

JST flux function consists of a blended combination of a second order pressure switched approximation to a scaled harmonic operator and a scaled approximation to a fourth order biharmonic operator [12]. The one equation model of Spalart and Allmaras [23] is employed for the simulation of turbulent flows. The resulting equation system is constructed using an edge based data structure and is solved by explicit iteration.

Within the Swansea FLITE system, the unstructured mesh generation process is controlled by means of a user specified function, that defines the desired distribution of the mesh spacing [17]. Following the discretisation of the surface of the computational domain [16], according to the requirements of the mesh control function, layers of tetrahedral elements are generated on viscous surfaces, using an advancing layers method [9]. This method employs advancing front concepts and generates points on smoothed normals emanating from the surface, with the thickness of the individual layers being specified by the user. This generation process stops when the number of layers generated reaches a specified value or when the size of the generated layer is comparable with the local mesh size specified by the mesh control function. Following this process, the structure of the generated layers is utilised to merge appropriate combinations of tetrahedra, to form a hybrid mesh of hexahedra, prisms and tetrahedral elements [19]. Prisms are located, where necessary on the outermost of the generated layers. The remainder of the domain is then discretised into tetrahedral elements, using a standard Delaunay procedure with point

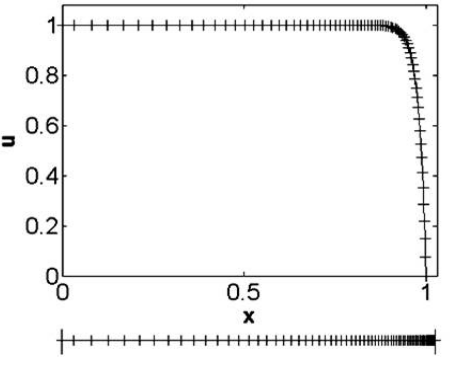


Fig. 2 Solution of a 1D steady state convection diffusion problem achieved using a fourth order finite volume method on the illustrated mesh

creation [24]. A detail of a mesh constructed in this fashion for flow over a 3D aircraft configuration is given in Figure 1. This clearly shows, on a symmetry plane, the consistent hybrid nature of the generated mesh.

In a series of papers, De Rango and Zingg [1, 2, 3] investigated the use of a globally fourth order accurate finite difference algorithm for computing steady flows over 2D aerofoils. They demonstrated that significant computational savings could be achieved, compared with the requirements of well established second order algorithms. The Swansea work under the ADIGMA project was intended to investigate

whether or not similar conclusions could be reached when a higher order finite volume discretisation was introduced into the Swansea FLITE code. The intention was that the higher order discretisation would be attempted, only within the boundary layers along solid surfaces, and would employ the structure present in the mesh generated by the advancing layers approach.

2.2 Initial 1D Testing

The practicality of the basic idea was initially tested in 1D. The problem considered required the steady state solution of the unsteady linear convection/diffusion equation. The problem was formulated on the spatial domain $0 \leq x \leq 1$ for values of the time $t \geq 0$ and boundary conditions $u = 1$ at $x = 0$, $u = 0$ at $x = 1$ for all $t > 0$ were imposed. For the convection dominated case, the steady state solution involves a boundary layer near $x = 1$. Initially, a uniform fourth order finite volume discretisation of all the terms in the governing equation was employed [8, 10, 11], with the resulting equation system solved by explicit multi–stage iteration. The form adopted for the initial condition only affected the number of explicit steps necessary for convergence. A uniform grid was used first and then a non uniform stretched grid was adopted. For the stretched grid, the discretisation was either achieved directly

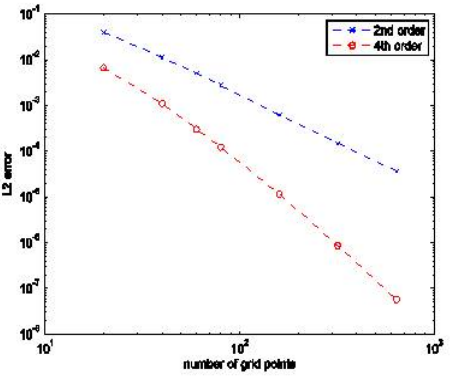


Fig. 3 The $\mathscr{L}_2$ error norm for the 1D steady state convection diffusion equation: comparison of the results computed with a second order (x) and a fourth order (o) discretisation on different meshes

in the physical space, with a non–uniform mesh spacing, or in a mapped space, with uniform mesh spacing. The use of compact differencing stencils [13, 4, 5] was not investigated, but this is an approach that would be worthy of further investigation. An additional sophistication, meant to replicate the approach that would be followed within the FLITE solver, was to employ the higher order discretisation within the boundary layer only and the standard discretisation elsewhere. No background dissipation was added.

In all cases, the method was found to work very well, as illustrated by the typical solution illustrated in Figure 2 in which the results, to the eye, are indistinguishable

from the exact. The results of a typical convergence study, comparing the variation of the $\mathscr{L}_2$ norm of the error, obtained with second order and the fourth order discretisations, with the number of grid points employed is shown in Figure 3. These results provided confidence in attempting to develop the method within the more complex environment of the FLITE code.

2.3 2D Implementation

The initial studies, directed at employing the proposed approach for practical simulations, were attempted in 2D. For 2D simulations, a detail of a typical hybrid mesh of triangles and quadrilaterals generated for the analysis of flow over an aerofoil is shown in Figure 4. When a mesh of this type is generated, the mesh generator is also

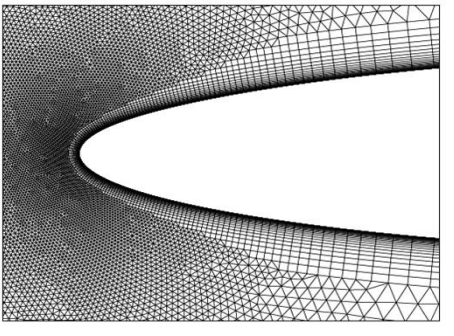


Fig. 4 Detail of a hybrid mesh generated by the advancing layers method for the simulation of transonic flow over an aerofoil

used to identify the normal lines running through the boundary layers and to provide the numbering of the consecutive nodes on each normal. Figure 5 indicates a typical situation, with the nodes numbered sequentially on one such line, and illustrates

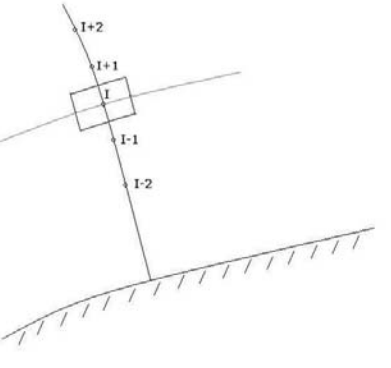


Fig. 5 Illustration of the consecutive numbering of the nodes in the boundary layer, on a normal to a 2D surface, and the dual mesh quadrilateral cell surrounding node *I*

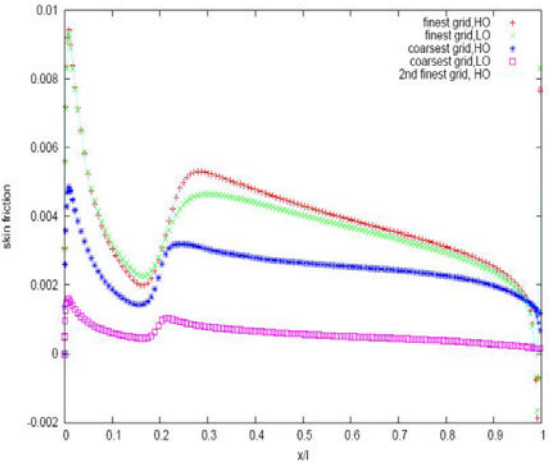


Fig. 6 Turbulent flow over a NACA0012 aerofoil at a free stream Mach number of 0.3, a Reynolds number of 1×10^6 and zero degrees angle of attack: skin friction distributions on the aerofoil surface computed with the new (HO) and the original (LO) schemes on different meshes

the finite volume, vertex centered, cell surrounding a given node I. Based upon the experience gained with the 1D simulations, it was decided to employ fourth order interpolation along these normal lines in the physical space to determine the flow variables, and the flow gradients, at the mid points of the faces of the finite volume cells which run parallel to the wall. With this implementation, the coefficients in the interpolation formulae are different at each point and, for computational efficiency, these coefficients are pre–computed and stored. These interpolated values are used to replace the simple averaged values, normally used to evaluate the flux vectors in the FLITE code, on these faces only.

The fourth order background stabilisation term also needs to be modified, to ensure that the projected benefits of the higher order discretisation are not swamped by this term. In the classical FLITE procedure, a stabilising diffusion term

$$\mathbf{AD}_I = \sum_{I \in \Lambda_I} \mathbf{A}_{IJ} \left(\mathbf{V}_J - \mathbf{V}_I \right) \qquad \text{where} \qquad \mathbf{V}_I = \sum_{I \in \Lambda_I} \left(\mathbf{U}_J - \mathbf{U}_I \right) \tag{1}$$

is added at each node I. Here Λ_I denotes the set of edges in the mesh that are connected to node I and $\mathbf{V}_I$ is formed by constructing edge differences in the solution. The simple scalar form $\mathbf{A}_{IJ} = \varepsilon \alpha_{IJ} \mathbf{I}$ is adopted for the dissipation matrix, where ε is a user specified constant. The coefficient $\alpha_{IJ}$is evaluated for each edge as

$$\alpha_{IJ} = \frac{1}{K_I + K_J} \min \left(\frac{\Omega_I}{\Delta t_I}, \frac{\Omega_J}{\Delta t_J} \right) \tag{2}$$

In this equation, K_I denotes the number of edges connected to node I, Δt_I is a local stability time step computed for node I and Ω_I is the area of the finite volume cell associated with node I.

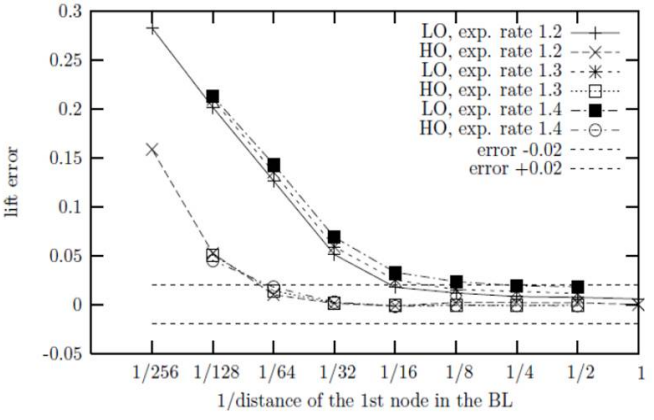


Fig. 7 Turbulent flow over an RAE2822 aerofoil at a free stream Mach number of 0.73, a Reynolds number of 6.5×10^6 and 3.19 degrees angle of attack: variation of the lift error computed by the new method (HO) and the original FLITE procedure (LO) on a sequence of meshes

The modified stabilisation is determined in a similar fashion, but is computed by recycling twice the edge differences in the solution at each node. This means that, in this case, the stabilization term added at node I is evaluated as

$$\mathbf{AD}_I = \sum_{I \in \Lambda_I} \mathbf{A}_{IJ} (\mathbf{W}_J - \mathbf{W}_I) \qquad \text{where} \qquad \mathbf{W}_I = \sum_{I \in \Lambda_I} (\mathbf{V}_J - \mathbf{V}_I) \tag{3}$$

and $\mathbf{V}_I$ is determined as before.

The procedure was initially tested for the case of turbulent subsonic flow over a NACA0012 aerofoil at a free stream Mach number of 0.3, a Reynolds number of 1 million and zero degrees angle of attack. A general indication of the improvements in solution quality offered by the use of the new approach are apparent in Figure 6, which shows the computed distribution of the skin friction obtained using the new and the original schemes on different grid levels.

2.4 Application to ADIGMA MTC5—Transonic RAE2822 Aerofoil

A better appreciation of the performance of the proposed approach can be gained by considering its application to the ADIGMA MTC5 test case. This requires the simulation of flow over an RAE2822 aerofoil at a Mach number of 0.73, a Reynolds number of 6.5 million and 3.19 degrees angle of attack. For this case, 46 computations were performed on a set of different meshes. These meshes only vary in the

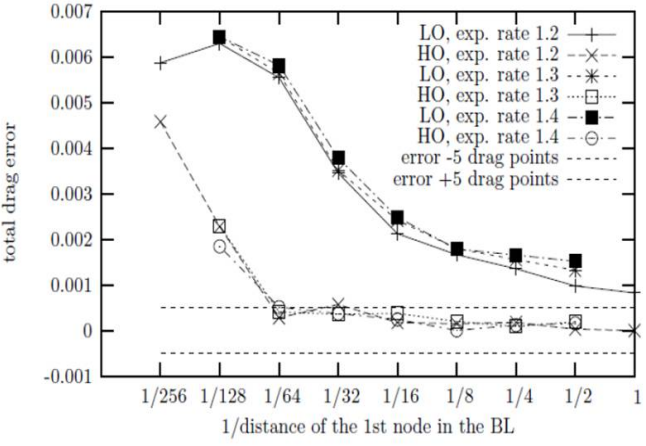


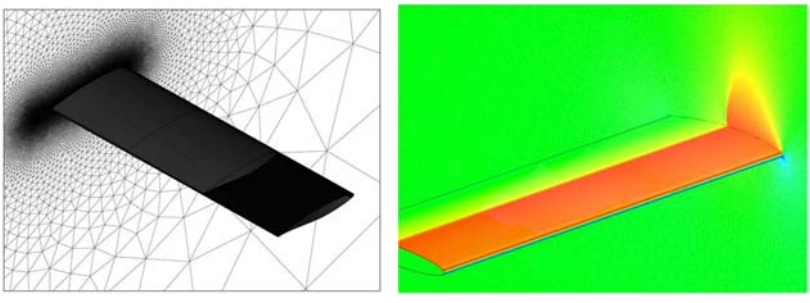
Fig. 8 Turbulent flow over an RAE2822 aerofoil at a free stream Mach number of 0.73, a Reynolds number of 6.5×10^6 and 3.19 degrees angle of attack: variation of the drag error computed by the new method (HO) and the original FLITE procedure (LO) on a sequence of meshes

expansion rate of the point distribution in the boundary layer, while the thickness of the boundary layer mesh remains constant in each case. The computed lift and drag errors have been plotted against the inverse of the distance of the first node from the wall in the boundary layer. Figure 7 shows the lift error for the different meshes. It can be observed that, for a given mesh, the original FLITE code (LO) gives a larger lift error than the new method (HO). The horizontal dashed lines represent the asymptotic value plus or minus the desirable engineering accuracy. The FLITE computations, on the coarsest meshes that remain within the engineering accuracy, require respectively 35, 29 and 26 layers. In comparison, the new approach produces results that remain within the engineering accuracy on meshes containing 27, 21 and 18 layers. This means that the new method allows a reduction of between 23% to 31% in the numbers of layers required in the boundary layer. Similar conclusions may be drawn from a study of the corresponding drag error plots, shown in Figure 8. On the same mesh, the new method gives a smaller drag error than the original FLITE procedure. The drag error increases faster in the results produced by the original FLITE procedure compared with the results of the new method, as the distance of the first node in the boundary layer increases.

This 2D study helps to underline the performance of the new method. It is possible to deduce that, as the boundary layer represents 60% of the total number of points of a mesh, the new approach allows for a reduction of about 20% in the total number of mesh points required. Simultaneously, only small modifications need to be made to the existing 2D FLITE code and the additional computation involved is fairly small, resulting generally in an increase of the order of 3% in the required cpu time.

2.5 3D Implementation

The necessary modifications are also readily implemented within the 3D FLITE solution procedure. However, the only application that was possible before the end of the project was to apply the new method to a pseudo–3D problem, consisting of flow over a wing of constant section. The wing section is taken to be in the form of the same RAE2822 aerofoil and the flow conditions were again those associated with the ADIGMA MTC5 test case. The simulations were performed on meshes that were constructed by stacking, in the span direction, meshes generated for the 2D aerofoil section. A detail of a typical discretization of this type is shown in Figure 9. Full analysis of the lift and drag results indicate that the 2D results are reproduced for this configuration. This gives confidence that the 3D implementation is correct. Further 3D analysis involving both a truly unstructured mesh for this configuration and an ONERA M6 turbulent test case are currently under investigation.



(*a*) (*b*)

Fig. 9 Pseudo–3D simulation of turbulent flow over a uniform wing, with RAE2822 aerofoil sections, at a free stream Mach number of 0.73, a Reynolds number of 6.5×10^6 and 3.19 degrees angle of attack: (a) indication of the structure of the stacked mesh used; (b) view of the distribution of the computed pressure contours on the symmetry plane and on the wing surface

3 Convergence Acceleration Developments

EADS-MAS modified two existing codes of industrial complexity in ADIGMA. For the two-dimensional research, the second order multigrid accelerated finite volume code developed at Swansea [18] was used as a basis. For this code, convergence acceleration techiques for the standard, second order, discretization scheme were developed. For 3D studies, EADS–MAS used the DLR Tau code [7] as a basis and, in this case, both the implicit preconditioning and the higher order boundary layer treatment were investigated in the same code. Here, the experiences obtained at Swansea for the higher order wall–normal boundary layer discretization were exploited. The particulars of the EADS–MAS pre–conditioner and the higher

order line discretizations will be described and a few examples will be presented that illustrate the application of the developed procedures to problems of different complexity.

3.1 Line Generation

As has been described, the boundary layer mesh is normally constructed by prisms or hexahedra in a layer–wise manner, resulting in a regular structure in a direction, more or less, normal to viscous walls. The thickness of this region varies but, typically, meshes employed to resolve boundary layers consist of around 30 layers. The

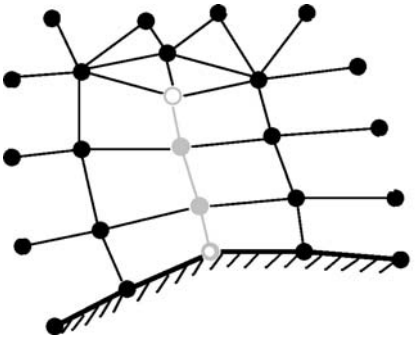


Fig. 10 Illustration of the line data structure in the boundary layer. The passive mesh is shown in black, the active mesh in grey.

elements closest to the wall are highly stretched and there is a gradual increase in the normal spacing as the distance to the wall increases. This means that the outer elements are much less stretched (optimally isotropic). Given this grid structure, it is relatively easy to construct lines emanating from wall nodes, following the edges of the elements and terminating at the outer boundary layer surface, as illustrated in Figure 1. The data structure for such lines can simply consists of registers, listing the node indexes of the lines. However, to improve the speed of the code, and for flexibility, a more complicated data structure was implemented in both the 2D and 3D versions of the code. The data structure implemented is illustrated in Figure 10, showing that for each line, two tiny, but complete, dual mesh structures were generated. One data structure, termed the active mesh for the line, contains the edges spanning internal line nodes. The other data structure, termed the passive mesh for the line, contains all nodes required to complete a two–level flux computation for the line node control volumes. The non–geometric data for the dual meshes are dynamically allocated, when needed, to reduce the memory penalty of this data structure.

3.2 Line Implicit Pre–conditioner

The theoretical peak performance of a boundary layer pre–conditioner can be estimated by completely solving, to convergence, for the boundary layer nodes before

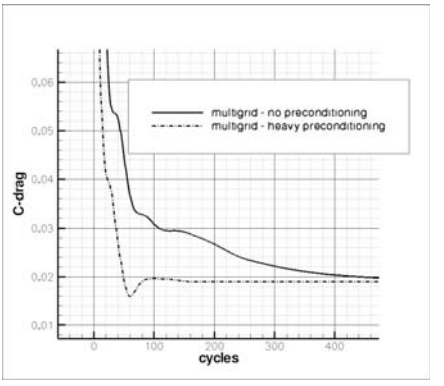


Fig. 11 Illustration of the preconditioning effect obtained by pre–solving the boundary layer before each multigrid cycle for the MTC5 test case

every solver iteration. This test was performed on the MTC5 test case [20] and is shown in Figure 11. The speed–up in convergence attained with respect to the number of cycles compared with the standard multigrid algorithm is more than five for this test case. For a subsonic version of the problem, the factor improves to around seven. Although this process does not reach the convergence rates normally achieved for Euler flow simulations, the improvements obtained are substantial and demonstrate that it is worth looking at the boundary layer for performance improvements for cases of this type.

To reduce the stiffness of the numerical system in the boundary layers, a pre–conditioning strategy has been developed. The pre–conditioner is separated from the main solver, which is an agglomerated multigrid approach, with Runge–Kutta and LU–SGS relaxation for the 2D and 3D codes respectively. In this way, the pre–conditioner is a separate solver, only inducing changes in the boundary layer, which can be applied in an inter–leaved manner at regular intervals. This approach is somewhat different from other methods found in the literature, where the boundary layer lines are used to provide a smoother for the multigrid solver [14]. Combining the two solvers in this manner did not, however, prove to be straightforward, as not all combinations run in a stable manner. This is mainly a problem if the pre–conditioner is applied too often. This is because it is constructed with aggressive directional relaxation along the lines and, thus, needs the multigrid solver to smooth the tangential components. On the other hand, if the pre–conditioning interval is too infrequent, the convergence improvement is reduced.

The pre–conditioner works by treating the boundary layer nodes with an implicit approach. For this, an analytic second order Jacobian matrix was implemented containing most of the terms stemming from the Navier–Stokes equations and the Spalart and Allmaras turbulence model. The Jacobian is stored as a block penta–diagonal matrix. For the line, the system

$$(\alpha \mathbf{D}_{IJ} + \mathbf{J}_{IJ})\,\Delta \mathbf{U}_J = \mathbf{R}_I \tag{4}$$

is solved directly for the unknown increment at node I, $\Delta \mathbf{U}_I$, using Gaussian elimination. In this equation, $\mathbf{J}_{IJ}$ denotes the discrete Jacobian matrix obtained by analytically differentiating the discrete flux at node I with the unknown vector $\mathbf{U}$ at node J, $\mathbf{R}$ is the flux residual, $\mathbf{D}$ is the unit diagonal vector, introduced for increasing the stability, and α is a stability parameter. The Jacobian is updated at each iteration, but this is probably not needed. This means that there remains a potential saving to be achieved by computing the inverse of the relaxed Jacobian once and, subsequently, only updating the fluxes in the iteration. This equation system is not completely solved and, usually, between 3 and 5 iterations are performed. After the iterations for the line are completed, a convergence check is performed to decide whether the pre–conditioning contribution should be included for the line or not. Typically, for optimal settings of the relaxation parameter, around 1% of the lines are rejected in the pre–conditioning. The result is a relatively robust approach, with the pre–conditioner influence added as a bonus in regions where it is well behaved. Here, a line is defined as well behaved if the residual of the Newton–Raphson iteration is reduced by at least a factor of three. There are various alternatives that can be adopted when updating the fluxes in equation (4). The approach chosen here is to only initialize the tangential fluxes at the first iteration and not to update them at later iterations. For this case, the data structure can be considerably reduced, as only the inner (active) mesh needs to be stored. The main advantage for this approach is that the line computations, while consistent, in effect decouple the tangential components of the fluxes in the pre–conditioning step. The effect of this is best illustrated by looking at the pre–conditioning performance on an initial velocity solution field consisting of free stream values for all internal nodes and the non–slip boundary condition at the wall. The task of the pre–conditioner can be thought of as generating a solution that looks like a classical boundary layer field. The lines are solved in a Jacobi manner, where the lines are decoupled and use the flow field at the beginning of the pre–conditioning step for the neighbouring (passive) nodes connected to the line. For the initial iteration, the tangential fluxes are close to zero, as there are almost no gradients in this direction for the initial free stream solution. However, due to the boundary condition, the normal gradient is very large and the pre–conditioner will attempt to smooth the velocity field in the normal direction. If the tangential fluxes are updated within each internal Newton–Raphson iteration, the tangential fluxes will influence the development of the boundary layer type velocity field. By only adding the original tangential fluxes, computed on the free stream solution, the boundary layer is allowed to evolve more freely. While this is a very aggressive approach, it is consistent as it will not change an already converged solution. It does, however, decouple the lines and introduces an instability effect which must be countered by appropriate smoothing in the tangential direction. In the current implementation, this is performed by the multigrid solver. Typically one pre–conditioning step is performed every 20–30 multigrid cycles. The relative cost of the pre–conditioning varies somewhat depending on the relative number of nodes in the boundary layer but, typically, is a factor of between one and three times as expensive as a multigrid cycle. The optimisation of the time taken to perform a pre–conditioning step has not been a primary priority to date, even though a clear potential is believed to

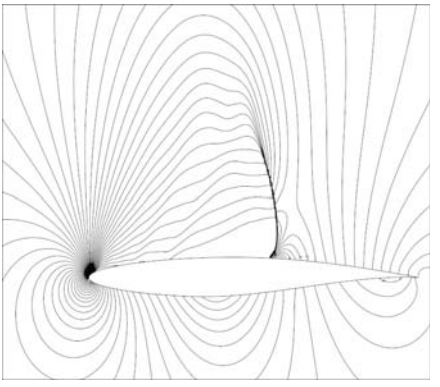


Fig. 12 Pressure plot for the MTC5 test case

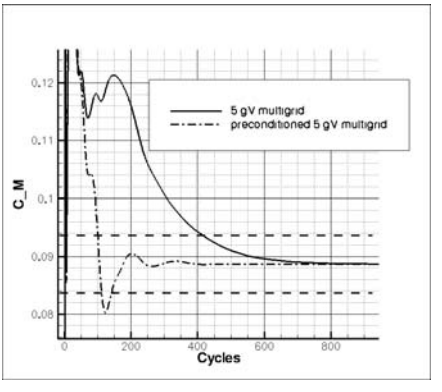


Fig. 13 Comparison of the convergence between the preconditioned and the standard multigrid methods for the MTC5 test case

exist. The effect of applying the pre–conditioner for the MTC5 test case is shown in Figures 12 and 13. For this test case, a CPU time speed–up in the region of 2–2.5 is achieved. For a subsonic version of this test case, the speed–up achieved improves to around a factor of 3.

3.3 Coupling with the Hybrid Algorithm

The boundary layer is characterized by relatively smooth solution fields and is, therefore, a good candidate region for the use of higher order discretizations. In particular, since the largest gradients are in the direction normal to the wall, a directional higher order discretization along the normal lines is likely to induce savings in the computational cost of resolving the boundary layers. The regular mesh structure of the lines can be used to apply simple 1D discretization schemes of finite difference type. To be able to use 1D stencils, either the lines normal to the wall have to be completely straight or they have to be mapped into a parametric space or ghost nodes have to be introduced in the stencil. For the EADS–MAS implementation,

the last approach was applied, mapping the outer node values in the four point flux stencil onto ghost nodes on the central edge line using the expression

$$\mathbf{U}_{I^*} = \mathbf{U}_I + \left.\frac{\partial \mathbf{U}}{\partial \mathbf{x}}\right|_I \mathbf{r}_{II^*} \tag{5}$$

where $\mathbf{U}_{I^*}$ denotes the ghost node value, and $\mathbf{r}_{II^*}$ is the offset vector of the ghost node I^* from node I.

For the gradients, the stencils

$$\mathbf{U}_{I+\frac{1}{2}} = -\frac{1}{12}\mathbf{U}_{I+2} + \frac{7}{12}\mathbf{U}_{I+1} + \frac{7}{12}\mathbf{U}_I - \frac{1}{12}\mathbf{U}_{I-1} \tag{6}$$

and

$$\Delta x \left.\frac{\partial \mathbf{U}}{\partial x}\right|_{I+\frac{1}{2}} = -\frac{1}{24}\mathbf{U}_{I+2} + \frac{15}{24}\mathbf{U}_{I+1} - \frac{15}{24}\mathbf{U}_I + \frac{1}{24}\mathbf{U}_{I-1} \tag{7}$$

for first and second order derivatives respectively were applied, which are fourth order accurate at the node for a regular mesh. Attempts were also made to include the stretching in these formulae, making the scheme fourth order also for lines with non–constant spacing. This was found to be unstable for the inviscid fluxes, but was applied for the viscous terms. A higher order discretization of the volume integration of the source terms in the turbulence model, using the trapezoidal rule, was attempted. Again, some instabilities were found using this approach and, due to time constraints preventing fuller investigation, the standard central formula was used for the results shown. For cases where the higher order source term treatment was stable, however, a relatively large effect was observed. This is something that should be further analyzed. The standard JST stabilization term [15] was kept for the current implementation in the TAU code, not transferring the work from Swansea on higher order stabilization terms, again due to time constraints. The higher order scheme was tested on the BTC0 testcase [21] and a view of the surface mesh employed is given in Figure 14. This yielded an allowable reduction in the boundary layers by

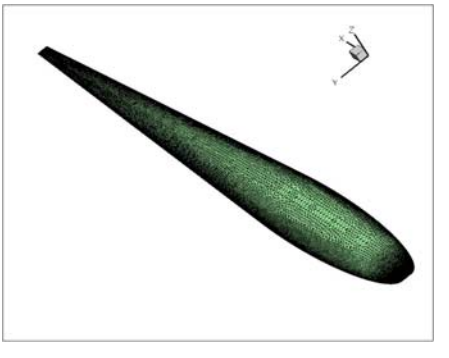


Fig. 14 View of the surface mesh for the BTC0 configuration

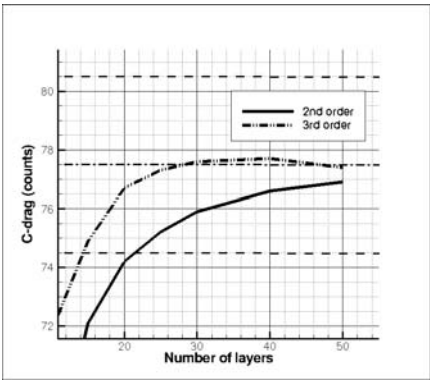


Fig. 15 Mesh convergence comparison between the standard and the higher order boundary layer discretization against the number of layers employed for the BTC0 test case

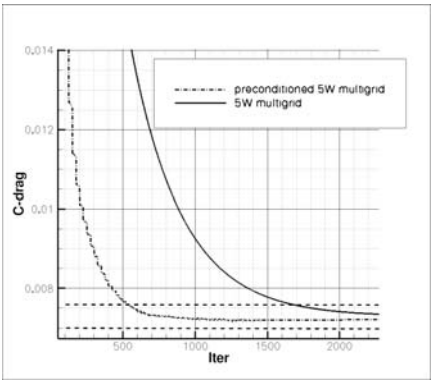


Fig. 16 Comparison of the convergence rates achieved with the preconditioned and the normal multigrid process for the BTC0 test case

a factor of around 1.6, as illustrated in Figure 15, resulting in a total reduction in the number of nodes in the mesh of around 30%. The added cost in CPU time of extending the order is insignificant and there appears to be no extra stiffness introduced. In addition, the implicit line preconditioning resulted in a speed–up factor of around 2.5, as shown in Figure 16. The combined effect of the two approaches can, thus, be said to result in a significant reduction, by around a factor of three, in the computational time. The study also showed that the second order Jacobian for the pre–conditioner works well in combination with the higher order discretization.

For cases where the boundary layer does not play a very important role for the integral values, the convergence rate is largely unaffected by the pre–conditioning. The reduction in the number of layers due to the higher order discretization does, however, represent a reduction in the numerical system size, regardless of flow condition. This also constitutes an improvement for cases where the boundary layer effects are secondary. An illustration of a case where the contribution of the boundary layer to the global integral values is minor is shown in Figure 17 and involves the simulation of transonic flow over the VFE2 delta wing at high angle of attack [22]. Due to the fact that the pre–conditioned higher order TAU code was not parallelized,

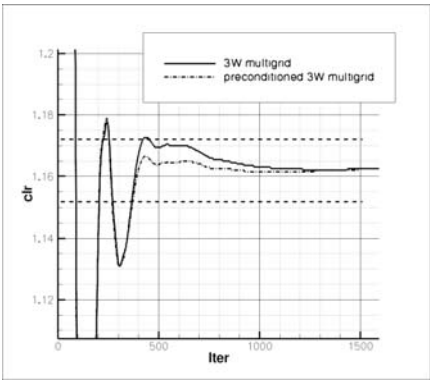


Fig. 17 Lift convergence curves for the preconditioned and the standard scheme for the CTC3 delta wing test case, using the higher order boundary layer discretization

this case had to be run on a coarse mesh. The runs illustrated that the approaches implemented can also run in a stable manner for more complex flows even though more experience is needed to enable general statements on the robustness and ease of use of the new code.

4 Conclusions

Research has been undertaken to investigate ways in which the computational performance of traditional low order unstructured mesh cell vertex finite volume codes, for viscous aerodynamic simulations, could be improved without major algorithmic changes. The research concentrated on two particular techniques. The first involved the use of a higher order discretization in the wall–normal direction, employing the structure present in the type of grids that are usually used to represent boundary layers. The second aimed at improving the performance of the agglomerated multigrid equation solver by adding a boundary layer pre–conditioner. The results of the research demonstrated that these approaches, when used in combination, offered the possibility of significantly reducing the computer time required to achieve the solution of a number of steady state viscous aerodynamic flow problems. However, further work is required before general conclusions can be drawn about the practical usefulness of these techniques for industrial flow simulations.

References

1. De Rango, S., Zingg, D.W.: Aerodynamic computations using a higher–order algorithm. AIAA Paper 99–0167 (1999)
2. De Rango, S., Zingg, D.W.: Further investigation of a higher–order algorithm for aerodynamic computations. AIAA Paper 2000–0823 (2000)
3. De Rango, S., Zingg, D.W.: Higher–order aerodynamic computations on multi–block grids. AIAA Paper 2001–2631 (2001)

4. Gaitonde, D.V., Shang, J.S.: Optimized compact–difference–based finite–volume schemes for linear wave phenomena. Journal of Computational Physics 138, 617–643 (1997)
5. Gaitonde, D.V., Shang, J.S., Young, J.L.: Practical aspects of higher–order numerical schemes for wave propagation phenomena. International Journal for Numerical Methods in Engineering 45, 1849–1869 (1999)
6. Gaitonde, D.V., Visbal, M.R.: High–order schemes for Navier–Stokes equations: algorithm and implementation into FDL3DI. US Air Force Research Laboratory Final Report AFRL–VA–WP–TR–1998–3060 (1998)
7. Gerhold, T., Friedrich, O., Evans, J., Galle, M.: Calculation of complex three-dimensional configurations employing the DLR–TAU code. AIAA Paper 97–0167 (1997)
8. Harten, A., Engquist, B., Osher, S., Chakravarthy, S.R.: Uniformly high order accurate essentially non–oscillatory schemes, III. Journal of Computational Physics 71, 231–303 (1987)
9. Hassan, O., Morgan, K., Probert, E.J., Peraire, J.: Unstructured tetrahedral mesh generation for three-dimensional viscous flows. International Journal for Numerical Methods in Engineering 39, 549–567 (1996)
10. Hernández, J.A.: High–order finite volume schemes for the advection–diffusion equation. International Journal for Numerical methods in Engineering 53, 1211–1234 (2002)
11. Hyman, J.M., Knapp, R.J., Scovel, J.C.: High order finite volume approximations of differential operators on nonuniform grids. Physica D 60, 112–138 (1992)
12. Jameson, A., Schmidt, W., Turkel, E.: Numerical simulation of the Euler equations by finite volume methods using Runge–Kutta timestepping schemes. AIAA Paper 81–1259 (1981)
13. Lele, S.K.: Compact finite difference schemes with spectral–like resolution. Journal of Computational Physics 103, 16–42 (1992)
14. Mavriplis, D.J.: On convergence acceleration techniques for unstructured meshes. ICASE Report 98-44 (1998)
15. Morgan, K., Peraire, J.: Unstructured grid finite element methods for fluid mechanics. Reports on Progress in Physics 61, 569–638 (1998)
16. Peiró, J., Peraire, J., Morgan, K.: The generation of triangular meshes on surfaces. In: Creasy, C., Craggs, C. (eds.) Applied Surface Modelling, pp. 25–33. Ellis–Horwood, Chichester (1989)
17. Peraire, J., Peiró, J., Morgan, K.: Advancing front grid generation. In: Thompson, J.F., Soni, B.K., Weatherill, N.P. (eds.) Handbook of Grid Generation, pp. 17–1:17–22. CRC Press, Boca Raton (1999)
18. Sørensen, K.A.: A multigrid procedure for the solution of compressible fluid flows on unstructured hybrid meshes. PhD Thesis University of Wales Swansea (2002)
19. Sørensen, K.A., Hassan, O., Morgan, K., Weatherill, N.P.: A multigrid accelerated hybrid unstructured mesh method for 3D compressible turbulent flow. Computational Mechanics 31, 101–114 (2003)
20. Sørensen, K.A.: ADIGMA Test Case Report MTC5, EADS–MAS (2008)
21. Sørensen, K.A.: ADIGMA Test Case Report BTC0, EADS–MAS (2009)
22. Sørensen, K.A.: ADIGMA Test Case Report, CTC3, EADS–MAS (2009)
23. Spalart, P.R., Allmaras, S.R.: A one–equation turbulence model for aerodynamic flows. AIAA Paper 92–0439 (1992)
24. Weatherill, N.P., Hassan, O.: Efficient three–dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. International Journal for Numerical Methods in Engineering 37, 2005–2039 (1994)

This page intentionally left blank

Chapter 23
Dynamic Load Balancing for Parallelization of Adaptive Algorithms

S. Gepner and J. Rokicki

Abstract. Mesh Adaptation is a technique allowing to solve of complex problems by limiting the use of computational resources. In parallel simulation, adaptivity causes perturbation in numerical load balance leading to decrease in overall parallel efficiency. It is therefore vital to allow for dynamic load balancing if adaptive algoritms are to be run in parallel. In this paper authors present results from implementation of a dynamic load balancing algorithm to parallel and adaptive compressible flow solver.

1 Introduction

Modern large scale flow computations require parallel approach to be effectively dealt with. It is common to use domain decomposition as it is a relatively simple and natural approach to parallelization. It requires the computational domain to be partitioned in to subdomains, prior to the simulations. Figure 1 shows an example of initial partitioning for a fighter plane like geometry.

If parallelization is to be used effectively, it is necessary that the partitioning satisfies certain conditions. Firstly the numerical load assigned to subdomains, should be well distributed. Commonly numerical effort is proportional to the amount of mesh entities assigned to a partition. As a consequence load balance might be achieved by even distribution of mesh elements between processors. Quality partitioning should also minimize the communication time spend on inter-processor communication. This, however might strongly dependent on both, the hardware being used and the problem being parallelized. In a general setting, it is difficult to speculate wether certain partitioning is optimal as far as the communication volume is concerned.

S. Gepner · J. Rokicki
Warsaw University of Technology, The Institute of Aeronautics and Applied Mechanics
Nowowiejska 24, 00-665 Warsaw, Poland
e-mail: sgepner@meil.pw.edu.pl, jack@meil.pw.edu.pl

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 327–338, 2010.
springerlink.com

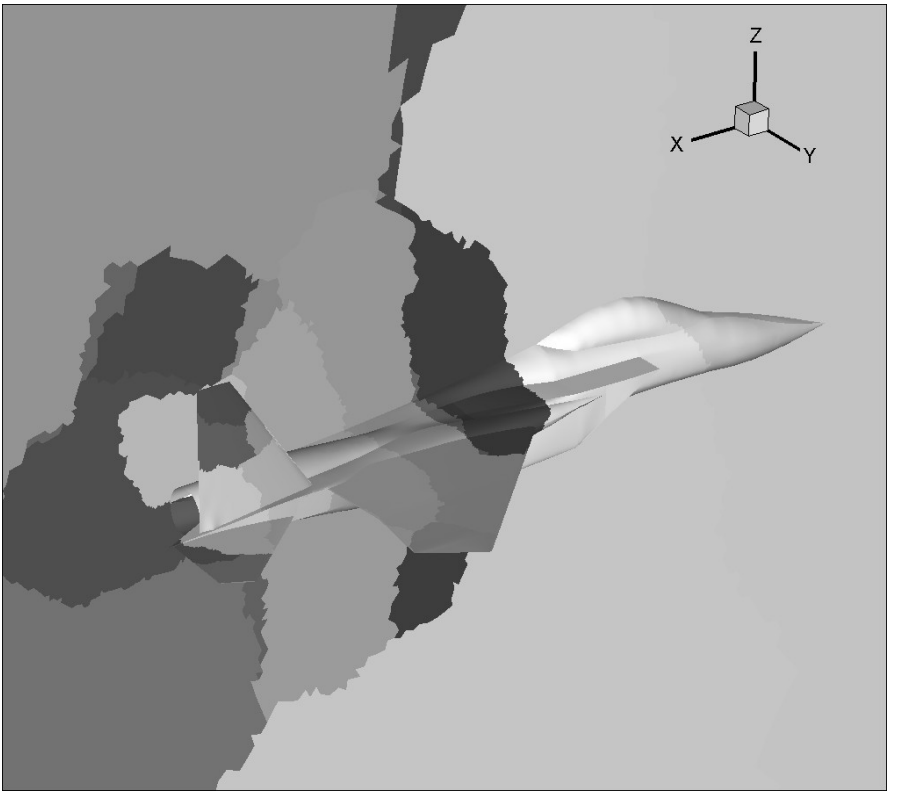


Fig. 1 Initial partitioning of a fighter airplane geometry - COOLFluiD project test-case [5]

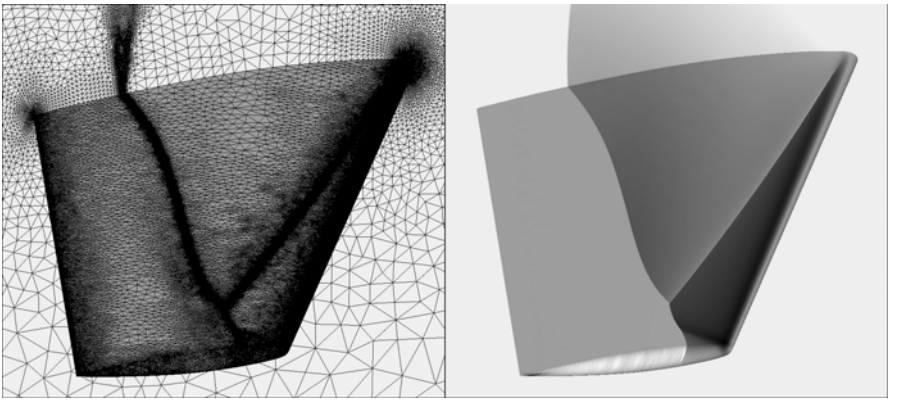


Fig. 2 Surface mesh for Onera M6 wing geometry (left) after a series of adaptations steps, and the resulting density field (right)

Adaptive techniques allow for computation of localized phenomena (boundary layers, shock waves, etc) with high resolution, while limiting the number of elements in less interesting regions. This increases the effectiveness of hardware use, as the necessary amount of resources is kept limited [2]. Figure 2 shows computational mesh of Onera M6 wing geometry adapted to capture a complex shockwave structure.

Adaptive algoritms introduce dynamic modifications to the computational mesh, and if used for parallel applications might have negative influence on the existing load balance. This in turn negatively affects the overall parallel effectiveness. If adaptivity is to be used in parallel applications it is vital to ensure proper load balancing throughout the simulation run time. This creates the need for a dynamically load balancing algorithm to be implemented.

Dynamic balancing of numerical load was considered in [6, 7, 8, 9, 10, 11, 12]. The present approach concentrates on merging the parallel anisotropic mesh refinement on unstructured meshes with a DLB algorithm. The benefits of such approach are evaluated in aerodynamically motivated applications. This paper presents results from implementation of the DLB algorithm into the in-house RED code [2] (a parallel residual distribution flow solver).

2 The Quality of Partitioning

Quality partitioning for domain decomposition based parallelization should, as mentioned in Section 1, follow some rules. It is not clear as to how to estimate the partitioning influence on the communication time [13, 14], since this might strongly depend on the hardware being used or the problem computed. Considering this, it is natural to measure the quality of partitioning by judging how far from the load balanced state the partitioning remains.

Let W be a total numerical load, equal to the total number of mesh entities. If p stands for the number of processor cores used, than the load considered optimal and assigned to an i'th processor would be: $w_{opt} = W/p$. The actual load assigned is w_i, so a load balance indicator is assumed in the following way:

$$\beta = \max_{0<i<p} \frac{w_i}{w_{opt}} \geqslant 1 \tag{1}$$

If defined in this way, the partitioning is of acceptable quality when load balance indicator is close to one.

3 Parallel Adaptivity through Anisotropic Mesh Refinement

A general, parallel adaptation for three dimensional complex geometries is a non trivial problem [2, 3, 4]. The algorithm implemented to enable parallel adaptivity in the RED code is illustrated in Fig. 3. It works in the following manner:

1. Gradient based error estimation is used to localize cell edges suitable for adaptations.
2. Nodes are introduced to the edges marked as adaptable.
3. Since work is performed in a parallel environment Processes communicate to ensure coherence in the overlap regions.
4. Cell topology is regenerated. The existing solution is interpolated on to the new mesh.
5. The communication rules are reset and the overlap is reconstructed.
6. The solution process is restarted on the adapted mesh.
7. When an appropriate convergence criteria are not met, process is returned to point one.

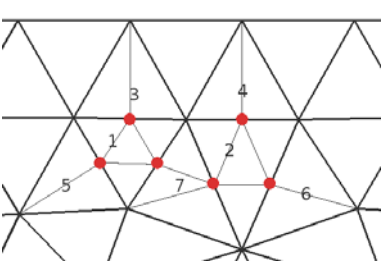


Fig. 3 Mesh refinement. Edges of the first and second cells have been marked as adaptable. New nodes are being created, and cell structure is remade.

Figure 4 shows the result of applying the described adaptive algorithm to a standard NACA0012 test case. The calculations ware performed for angle of attack 1.25 at 0.8 Mach number. On the left shown is the initial mesh and the resulting density contour. On the right hand side shown is the mesh and density contour after consecutive application of an adaptive algorithm. It might be observed that the regions near the shocks are better resolved.

4 Repartitioning Phase

At some point during the run of an adaptive parallel simulation it is necessary to perform a dynamic load balancing step. Commonly this event would be triggered when a threshold of acceptable load unbalance is breached. This might happen after the adaptation step is taken. Now it is necessary to calculate and apply a new better balanced partitioning. It is accomplished using one of many available partitioning tools. Subsequently the mesh entities are redistributed between processors according to the obtained coloring.

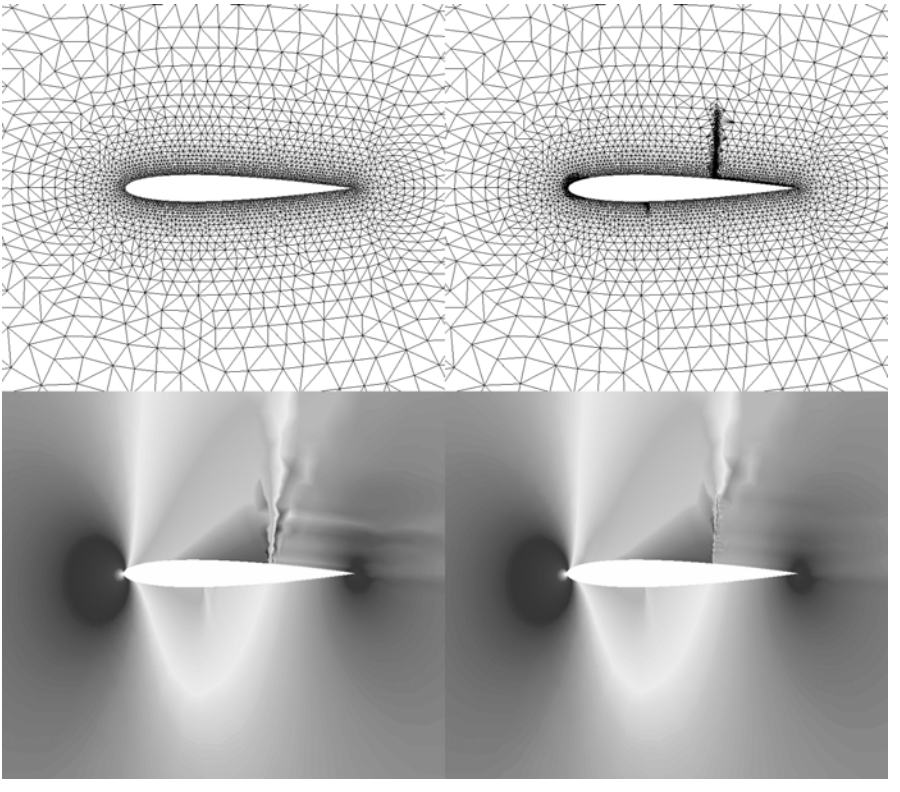


Fig. 4 NACA0012 test case. Initial mesh and resulting density field (left), case resulting from application of an adaptive algorithm.

4.1 Mesh Partitioning Tool Graph Partitioner

To calculate mesh decomposition an outside tool is used. In this work the ParMETIS [1] graph partitioner was used. It is based on the multilevel approach, and is capable of computing graph partitions of good quality. Main factor in it's selection was its parallel efficiency and ease of use.

4.2 Mesh Partitioning

To perform dynamic load balancing procedure, the following steps have to be undertaken:

1. Elements of the mesh being repartitioned have to be given unique numbering throughout the whole partitioned domain. This makes mesh elements easily distinguishable throughout the parallel environment.
2. Grid topology has to be described as a connectivity graph. When working with ParMETIS the so called distributed CSR format is used.

3. With graph connectivity ready it is possible for ParMETIS to calculate new mesh partitioning.
4. According to the calculated partitioning there are some data movement to be performed. During this phase mesh elements are selected for communication or deletion appropriately. This step is strongly dependent on the data structure used by the simulation software.

Applying the new partitioning involves massive changes to the simulation data structure. Some mesh entities have to be send to other computing nodes while some must be removed. Also the received data have to be added to the data structure accessible to a given CPU. It is a process strongly dependent on the data structure used within the solver. Once the information is exchanged the solver is ready to restart calculations.

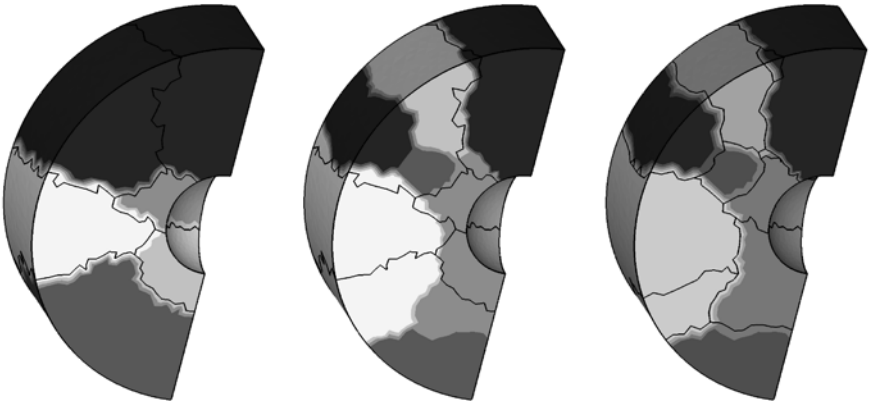


Fig. 5 Partitions are being modified according to the previously calculated coloring. On the left is the original partitioning and coloring, the middle picture shows original partitioning with new coloring. On the right is the partitioning after elements have been sent / deleted.

This process is illustrated on Fig. 5 where a three dimensional domain is being repartitioned. On the left shown is the initial partitioning. In the middle presented is the new coloring calculated during the graph partitioning phase plotted onto original partitioning. The rightmost picture shows the mesh after appropriate data migration has been carried out.

5 Impact of DLB on Parallel Efficiency

To judge the merits of applying the DLB to an adaptive flow computation the case shown on fig. 6 was considered. Adaptive algorithm was used in a parallel setting.

On the right shown is the case where no balancing is applied. The value measuring the quality (see Sec. 2 and [14]) is increasing with each adaptation step, thus indicating a drop in parallel efficiency. On the left, for the same adaptive simulation load balancing is applied. The load balance measure remains close to one, indicating that the high efficiency is maintained.

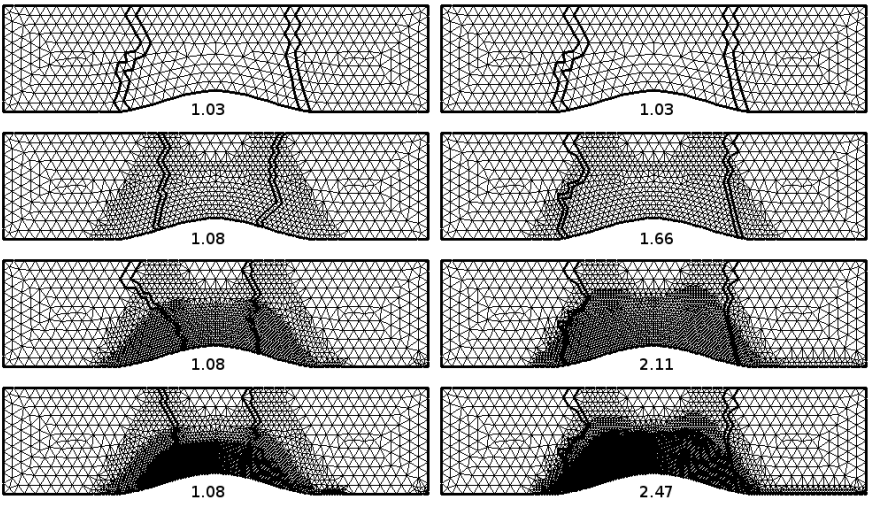


Fig. 6 Mesh after consecutive refinement. Dynamic load balancing has been applied on the left. The number next to the mesh denotes its load balance indicator (LBI).

Figure 7 shows convergence history as a function of computation time for both considered cases. It might be observed that the use of dynamic balancing allowes for convergence time to be reduced.

To investigate the cost of a single rebalancing step a three dimensional simulation of the flow around Oner M6 wing was selected. Mesh of 5078927 nodes (32622210 cells) is used. Results ware obtained by running the RED solver on the cluster containing 20 Quad-Core AMD Opteron processors (4 cores per processor) with 2 GB of memory per core, connected by Fast Ethernet connection.

Numerical cost of rebalancing is relatively small in comparison to total simulation time. Table 1 shows comparison of CPU time required to perform the simulation with explicit Euler solver. Compared is the time of a single iteration, time required to perform graph partitioning (ParMETIS run time) and a total time required to perform a full rebalancing step. It is worth noticing that total rebalancing time decreases as the number of CPU's grows.

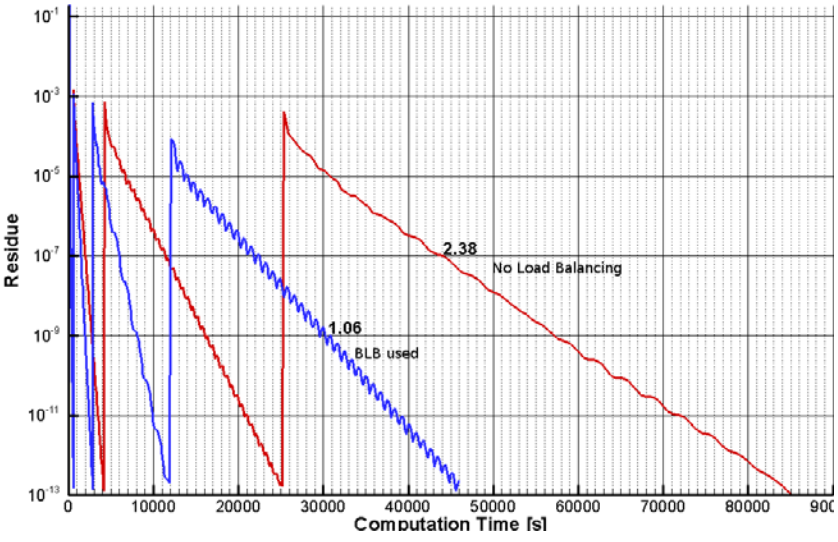


Fig. 7 Convergence as a function of computational time, with and without load balancing used. For a case from Fig. 6

Table 1 Comparison of repartition time, graph partitioning time, average iteration time; case of 5078927 nodes

No of cores	Simulation time	Total repartition time	Partitioner Run time
32	311500 s	663 s	5 s
72	157500 s	165 s	2 s
80	129500 s	128 s	3 s

6 Scramjet Testcase

To test the effectiveness of the presented approach Scramjet test case was selected. The geometry considered is shown on Fig. 8. The test was run in parallel using meshes of different resolution (from coarsest to finest: SCRAM 1-4). The coarser meshes ware adapted as the solution process was carried out. Residual distribution, explicit Euler solver was used.

For the finest mesh (SCRAM4) of 1600054 isotropic elements it took 4806 seconds to converge, using 80 computational cores. In this case no adaptation was used. The results are comparable in quality to results on coarser but adapted meshes. Figure 8 shows the Mach field resulting from calculation performed for an adaptive case. The original mesh of 18028 (SCRAM2) elements is adapted to capture the shock structure. Mesh after adaptation is composed of 159309 elements. The

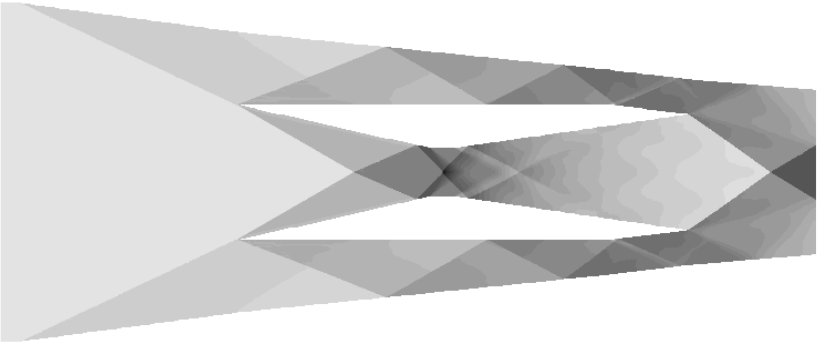


Fig. 8 Mach field distribution calculated for a Scramjet geometry (SCRAM2). Mesh was adapted from the original of 18028 elements to 159309 cells after final adaptation. Results of comparable quality ware obtained by using much finer meshes with no adaptation, but at higher numerical cost.

problem was run using 64 computational cores, and took 798 seconds to converge. Dynamic load balancing was used for this calculation. This situation is shown in Fig. 9. On the right shown is mesh configuration after consecutive adaptation steps. On the left picture presented are consecutive meshes. The picture on the right show partitioning resulting from use of a DLB algorithm. Data on the load balance measure with and without load balancing for the case shown on Fig. 9 are presented in table 2. It might be observed the algorithm significantly improves the load balancing.

Table 2 Load balance indicators (LBI) prior and after applying dynamic load balancing (DLB) for SCRAM2 mesh on 64 CPU's. See Fig. 9.

	LBI prior to DLB	LBI after DLB
Initial mesh	1.11	1.11
1'st adapt	1.94	1.10
2'nd adapt	1.90	1.11
3'rd adapt	1.74	1.10
4'th adapt	1.51	1.07
5'th adapt	1.33	1.08

Table 3 presents data on mesh sizes at specific moments of the simulation and the total time required to converge for all considered cases. Difference in mesh sizes is caused by doubled cells in the overlap regions.

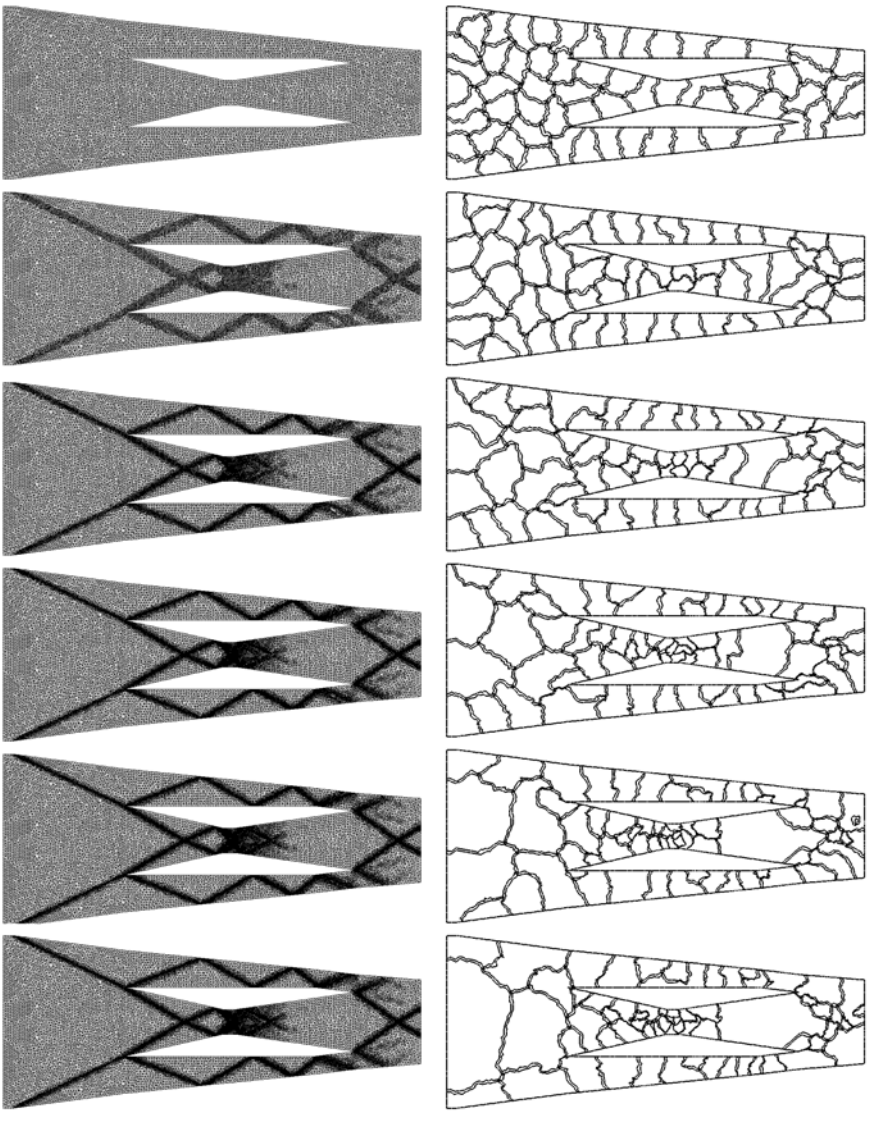


Fig. 9 Mesh adaptation and resulting partitioning for a case simulated on 64 CPU's. The initial mesh is SCRAM2.

Table 3 Initial mesh size, after completing first adaptation and after final adaptation compared with overall convergence time for a given case

Test:	No. cores	Init. mesh size	1'st adap.	final mesh size	Total sim. time
SCRAM1	16	2878	4681	22388	721 s
	32	3094	4937	22612	536 s
SCRAM2	16	16572	25945	157680	2920 s
	32	17159	26623	158904	1215 s
	64	18028	26745	159309	798 s
SCRAM3	16	64709	93221	554023	4579 s
	32	65816	94611	557671	2765 s
	64	67503	95014	563500	1931 s
SCRAM4	80	1600054	–	1600054	4806 s

7 Conclusions

Parallelization and adaptation are both techniques allowing for solving complex problem with time effectiveness. To use both it is necessary to ensure proper load balancing. It was shown that dynamic load balancing algorithm allowes to couple adaptivity with parallel processing. The rebalancing overhead is small in comparison to the overall simulation time (especially for larger problems).

References

1. Karypis, G., Schloegel, K., Kumar, V.: PARMETIS* Parallel Graph Partitioning and Sparse Matrix Ordering Library
2. Majewski, J.: Anisotropic solution-adaptive technique applied to simulation of steady and unsteady compressible flows. In: Proceedings of the International Conference on Computational Fluid Dynamics, Ghent, Belgium (2006)
3. Majewski, J.: Ph.D. Thesis Grid generation for flow problems in complex geometries (Rozprawa Doktorska, Generacja siatek dla symulacji przepywww obszarach o zoonej geometrii.) Warsaw University of Technology (2001) (in Polish)
4. Majewski, J.: Anisotropic adaptation for flow simulations in complex geometries. In: 36th Lecture Series on Computational Fluid Dynamics, ADIGMA course on hp-adaptive and hp-multigrid methods. von Karman Institute for Fluid Dynamics (2009)
5. COOLFluid project home page, `http://coolfluidsrv.vki.ac.be/trac/coolfluid`
6. Cybenko, G.: Dynamic load balancing for distributed memory multiprocessors. J. Parallel Distrib. Comput. 7, 279–301 (1989)
7. Devine, K., Boman, E., Heaphy, R., Hendrickson, B., Teresco, J., et al.: New challenges in dynamic load balancing. Applied Numerical Mathematics 52(23), 133–152 (2005)
8. Touheed, N., Selwood, P., Jimack, P.K., Berzins, M.: A comparison of some dynamic load-balancing algorithms for a parallel adaptive flow solver. Parallel Computing 26(12), 1535–1554 (2000)
9. Karypis, G., Schloegel, K., Kumar, V.: Dynamic Repartitioning of Adaptively Refined Meshes. Department of Computer Science and Engineering (1998)

10. Devine, K.D., et al.: New challenges in dynamic load balancing. Applied Numerical Mathematics 52, 133–152 (2004)
11. Karagiorgos, G., Missirlis, N.M.: Accelerated diffusion algorithms for dynamic load balancing. Information Processing Letters 84, 61–67 (2002)
12. Hendrickson, B., Devine, K.: Dynamic load balancing in computational mechanics. Comp. Meth. Appl. Mech. Engrg. 184, 485–500 (2000)
13. Hendrickson, B., Kolda, T.G.: Graph partitioning models for parallel computing. Parallel Computing 26, 1519–1534 (2000)
14. Rokicki, J., Tak, J., Drikais, D., Majewski, J.: Parallel Performance of Overlapping Mesh Techniques for Compressible Flows. In: Future Generation Computer Systems, vol. 19/1, pp. 3–15. Elsevier Science, Amsterdam (2001)

Chapter 24
Error Estimation and Adaptive Mesh Refinement for Aerodynamic Flows

Ralf Hartmann, Joachim Held, Tobias Leicht, and Florian Prill

Abstract. We consider the adjoint-based error estimation and goal-oriented mesh refinement for single and multiple aerodynamic force coefficients as well as residual-based mesh refinement applied to various three-dimensional laminar and turbulent aerodynamic test cases defined in the ADIGMA project.

1 Introduction

Important quantities in aerodynamic flow simulations are the aerodynamic force coefficients like the drag, lift and moment coefficients. In addition to the exact approximation of these quantities it is of increasing importance, in particular in the field of uncertainty quantification, to estimate the error in the computed quantities. By employing a duality argument error estimates can be derived for estimating the error measured in terms of the aerodynamic force coefficients. The error estimate includes primal residuals multiplied by the solution to an adjoint problem related to the force coefficient. The error estimate can be decomposed into a sum of local adjoint-based indicators which can be employed to drive a goal-oriented adaptive mesh refinement algorithm specifically tailored to the accurate and efficient approximation of the aerodynamic force coefficient.

Provided the adjoint solution related to an arbitrary target functional is sufficiently smooth the corresponding error representation can be bounded from above by an error estimate which includes primal residuals but is independent of the adjoint solution. By localizing this error estimate so-called residual-based indicators are obtained. Mesh refinement based on these

Ralf Hartmann · Joachim Held · Tobias Leicht · Florian Prill
Institute of Aerodynamics and Flow Technology, DLR (German Aerospace Center), Lilienthalplatz 7, 38108 Braunschweig, Germany
e-mail: {ralf.hartmann,joachim.held,tobias.leicht,florian.prill}@dlr.de

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 339–353, 2010.
springerlink.com

indicators leads to meshes which resolve *all* flow features irrespective of any specific target quantity.

Before the start of the ADIGMA project the techniques of error estimation, adjoint-based mesh refinement and residual-based mesh refinement were available for 2d laminar compressible flows around simple airfoil geometries, see e.g. [5, 11]. Within the ADIGMA project these techniques have been extended to 3d laminar compressible flows, see [13], as well as to 2d turbulent and 3d turbulent compressible flows. Furthermore, the error estimation and adjoint-based mesh refinement for single target quantities has been extended to the treatment of multiple target quantities, see [6]. The algorithms are applied to a range of test cases defined in the ADIGMA project.

2 Error Estimation and Adaptive Mesh Refinement

We consider the discontinuous Galerkin (DG) finite element discretization of the compressible flow equations, see e.g. [12, 13]: Find $\mathbf{u}_h \in \mathbf{V}_{h,p}$ such that

$$\mathcal{N}(\mathbf{u}_h, \mathbf{v}_h) = 0 \qquad \forall \mathbf{v}_h \in \mathbf{V}_{h,p}, \tag{1}$$

where the discrete function space $\mathbf{V}_{h,p}$ consists of discontinuous piecewise polynomial functions of degree $p \geq 0$. Given a target quantity $J(\mathbf{u})$ like for example the aerodynamic drag, lift or moment coefficient, a duality argument can be employed, see e.g. [3, 9], to obtain following error representation

$$J(\mathbf{u}) - J(\mathbf{u}_h) = -\mathcal{N}(\mathbf{u}_h, \mathbf{z} - \mathbf{z}_h) \equiv \mathcal{R}(\mathbf{u}_h, \mathbf{z} - \mathbf{z}_h) \approx \mathcal{R}(\mathbf{u}_h, \tilde{\mathbf{z}}_h - \mathbf{z}_h) \tag{2}$$

for any discrete function $\mathbf{z}_h \in \mathbf{V}_{h,p}$, where the exact adjoint solution $\mathbf{z}$ is replaced by the solution $\tilde{\mathbf{z}}_h$ to following discrete adjoint problem: Find $\tilde{\mathbf{z}}_h \in \tilde{\mathbf{V}}_{h,p}$ such that

$$\mathcal{N}'[\mathbf{u}_h](\mathbf{w}_h, \tilde{\mathbf{z}}_h) = J'[\mathbf{u}_h](\mathbf{w}_h) \quad \forall \mathbf{w}_h \in \tilde{\mathbf{V}}_{h,p}. \tag{3}$$

A possible choice of the adjoint discrete function space is $\tilde{\mathbf{V}}_{h,p} = \mathbf{V}_{h,p+1}$. The approximate error representation in (2) can be localized

$$J(\mathbf{u}) - J(\mathbf{u}_h) \approx \mathcal{R}(\mathbf{u}_h, \tilde{\mathbf{z}}_h - \mathbf{z}_h) \equiv \sum_{\kappa \in \mathcal{T}_h} \tilde{\eta}_\kappa, \tag{4}$$

where $\tilde{\eta}_\kappa$ are the so-called *adjoint-based indicators* which include the local residuals multiplied by the discrete adjoint solution. These indicators can be used to drive an adaptive mesh refinement algorithm tailored to the accurate and efficient approximation of the target quantity $J(\mathbf{u})$ under consideration.

The extension of the adjoint-based error estimation and mesh refinement approach to multiple target quantities has previously been considered for the inviscid Burgers' equation in [10] and has been extended within the ADIGMA

project to viscous compressible flows in [6]. Estimating the error in multiple quantities of interest, $J_i(\mathbf{u}), i = 1, \ldots, N$, would require the computation of the solutions $\tilde{\mathbf{z}}_{h,i} \in \tilde{\mathbf{V}}_{h,p}$ to N discrete adjoint problems:

$$\mathcal{N}'[\mathbf{u}_h](\mathbf{w}_h, \tilde{\mathbf{z}}_{h,i}) = J_i'[\mathbf{u}_h](\mathbf{w}_h) \quad \forall \mathbf{w}_h \in \tilde{\mathbf{V}}_{h,p}, \quad i = 1, \ldots, N, \tag{5}$$

and the evaluatation of the error representation for each of the quantities,

$$J(\mathbf{u}) - J(\mathbf{u}_h) \approx \mathcal{R}(\mathbf{u}_h, \tilde{\mathbf{z}}_{h,i} - \mathbf{z}_{h,i}), \quad i = 1, \ldots, N. \tag{6}$$

Instead, we compute the solution to following discrete error equation,

$$\mathcal{N}'[\mathbf{u}_h](\tilde{\mathbf{e}}_h, \mathbf{w}_h) = \mathcal{R}(\mathbf{u}_h, \mathbf{w}_h) \quad \forall \mathbf{w}_h \in \tilde{\mathbf{V}}_{h,p}, \tag{7}$$

and evaluate following approximation of $J_i(\mathbf{u}) - J_i(\mathbf{u}_h)$,

$$J_i(\mathbf{u}) - J_i(\mathbf{u}_h) \approx J_i'[\mathbf{u}_h](\mathbf{e}) \approx J_i'[\mathbf{u}_h](\tilde{\mathbf{e}}_h), \quad i = 1, \ldots, N, \tag{8}$$

where $\mathbf{e} = \mathbf{u} - \mathbf{u}_h$. Furthermore, defining a suitable combination $J_c(\mathbf{u})$ of the original target quantities, see [6], we compute the solution to following discrete adjoint problem

$$\mathcal{N}'[\mathbf{u}_h](\mathbf{w}_h, \tilde{\mathbf{z}}_{c,h}) = J_c'[\mathbf{u}_h](\mathbf{w}_h) \quad \forall \mathbf{w}_h \in \tilde{\mathbf{V}}_{h,p}, \tag{9}$$

and evaluate the error estimate

$$J_c(\mathbf{u}) - J_c(\mathbf{u}_h) = \mathcal{R}(\mathbf{u}_h, \mathbf{z}_c - \mathbf{z}_h) \approx \mathcal{R}(\mathbf{u}_h, \tilde{\mathbf{z}}_{c,h} - \mathbf{z}_h) \equiv \sum_{\kappa \in \mathcal{T}_h} \tilde{\eta}_\kappa^c. \tag{10}$$

The combined target quantity $J_c(\mathbf{u})$ can be defined, see [6], such that the error with respect to $J_c(\cdot)$ represents the sum of relative errors in the original target quantities, $\sum_{i=1}^N |J_i(\mathbf{u}) - J_i(\mathbf{u}_h)|/|J_i(\mathbf{u}_h)|$, or a weighted sum of absolute errors, $\sum_{i=1}^N \alpha_i |J_i(\mathbf{u}) - J_i(\mathbf{u}_h)|$ with weighting factors $\alpha_i > 0$. The adjoint-based indicators, $\tilde{\eta}_\kappa^c$, obtained by localizing the estimate (10) can be used to drive an adaptive algorithm for the accurate and efficient approximation of all the target quantities, $J_i(\mathbf{u}), i = 1, \ldots, N$, under consideration.

Finally, we note that for a target quantity $J(\mathbf{u})$ with a sufficiently smooth adjoint solution the error representation (2) can be bounded from above as follows

$$|J(\mathbf{u}) - J(\mathbf{u}_h)| \leq \left(\sum_{\kappa \in \mathcal{T}_h} \left(\eta_\kappa^{\text{res}}\right)^2 \right)^{1/2}, \tag{11}$$

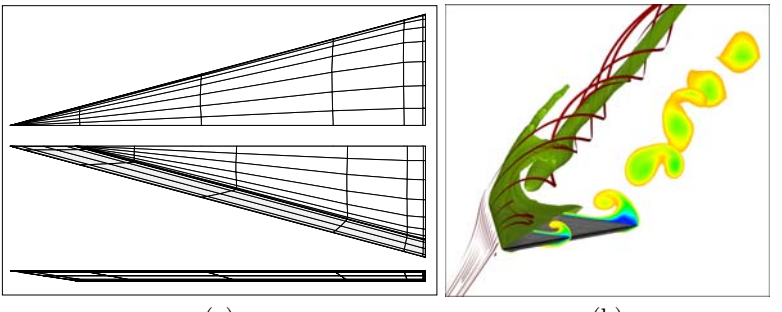
see [6, 8, 11, 13], where the so-called *residual-based indicators* η_κ^{res} include the primal residuals but are independent of the adjoint solution. Not depending on a particular target quantity the mesh refinement using residual-based indicators targets at resolving all flow features.

3 Numerical Results

In this section we demonstrate the performance of the adjoint-based error estimation, the goal-oriented and the residual-based mesh refinement for a range of aerodynamic test cases defined in the ADIGMA project. The computations have been performed with the DLR PADGE code [7] which is based on a modified version of the `deal.II` library [1].

3.1 ADIGMA BTC3: Laminar Flow around Delta Wing

First we consider a laminar flow around a delta wing. The delta wing has a sharp leading edge and a blunt trailing edge. A similar case has previously been considered in [15]. The geometry of the delta wing can be seen from the initial surface mesh in Figure 1(a). The delta wing is considered at laminar



(a) (b)

Fig. 1 Laminar delta wing: a) initial surface mesh: Top, bottom and side view of the half delta wing with straight leading edges, b) solution plot showing streamlines and a Mach number isosurface over the left half of the wing as well as Mach number slices over the right half, [13].

conditions with inflow Mach number equal to 0.3, at an angle of attack $\alpha = 12.5°$, and Reynolds number $\mathrm{Re} = 4000$ with isothermal no-slip wall boundary condition imposed on the wing geometry. As the flow passes the leading edge it rolls up, creates a vortex and a secondary vortex. The resulting vortex system remains over long distances behind the wing, see Figure 1(b). In the following the total drag, lift, and moment coefficients, C_d, C_l and C_m, will be computed up to a predefined error tolerance `TOL`. The following industrial accuracy requirements have been defined:

$$\begin{aligned} |J_{C_\mathrm{l}}(\mathbf{u}) - J_{C_\mathrm{l}}(\mathbf{u}_h)| &\leq \mathtt{TOL}_{C_\mathrm{l}} &&= 10^{-2}, \\ |J_{C_\mathrm{d}}(\mathbf{u}) - J_{C_\mathrm{d}}(\mathbf{u}_h)| &\leq \mathtt{TOL}_{C_\mathrm{d}} &&= 10^{-3}, \\ |J_{C_\mathrm{m}}(\mathbf{u}) - J_{C_\mathrm{m}}(\mathbf{u}_h)| &\leq \mathtt{TOL}_{C_\mathrm{m}} &&= 10^{-3}. \end{aligned} \tag{12}$$

By performing high order computations on fine meshes the following reference values of the force coefficients have been obtained: $J_{C_\mathrm{l}}(\mathbf{u}) = C_\mathrm{l}^\mathrm{ref} = 0.34865$, $J_{C_\mathrm{d}}(\mathbf{u}) = C_\mathrm{d}^\mathrm{ref} = 0.16608$, and $J_{C_\mathrm{m}}(\mathbf{u}) = C_\mathrm{m}^\mathrm{ref} = -0.03065$.

In the following we compare the performance of various refinement strategies in meeting these accuracy requirements. In particular, we consider the single-target error estimation and mesh refinement approach for each of the C_l, C_d, and C_m coefficients, separately. This results in three different sequences of locally refined meshes where on each mesh a flow problem (1) and a discrete adjoint problem (3) are solved and the error estimate (4) is evaluated. This is compared to residual-based and to global mesh refinement. Furthermore, we consider a multi-target error estimation and mesh refinement approach for reducing the sum of relative errors of the C_l, C_d and C_m coefficients. This results in one sequence of locally refined meshes which is targeted at reducing the error in all three coefficients, simultaneously. Here, on each mesh a flow problem (1), a discrete error equation (7), and a discrete adjoint problem (9) are solved and the error estimates (8) and (10) are evaluated.

In Figure 2(a)-(c) we see that for C_l and C_d the residual-based refinement is more efficient than global mesh refinement which, however, is not the case for C_m. Whereas the residual-based indicators target at resolving all flow features, see the resolution of the vortex system in Fig. 3, they do not necessarily result in meshes suitable for accurately approximating force coefficients. In contrast to that we see that the adjoint-based refinement is significantly more accurate than both, residual-based and global mesh refinement. Furthermore, we see that the accuracy of the single-target and the multi-target adjoint-based mesh refinement is comparable. Finally, we see that the enhanced force coefficients, $\tilde{C}_\mathrm{d/l/m} = C_\mathrm{d/l/m} + \sum_{\kappa \in \mathcal{T}_h} \tilde{\eta}_\kappa$, in case of the single-target algorithm and $\tilde{C}_\mathrm{d/l/m} = C_\mathrm{d/l/m} + J_i'[\mathbf{u}_h](\tilde{\mathbf{e}}_h)$ in case of the multi-target algorithm, are significantly more accurate than the original $C_\star$ values on the adjoint-based refined meshes. This demonstrates that the error estimation for single as well as for multiple target quantities is accurate and reliable.

Figure 2(d) shows the error in the drag coefficient vs. the computing time relative to the extrapolated time required for global mesh refinement to meet the tolerances (12). For meeting the tolerances (12) the residual-based mesh refinement requires about 10% of the time required for global mesh refinement. The adjoint-based mesh refinement requires about 2% and the adjoint-based mesh refinement including error estimation requires in the range of 0.1%. These time measurements include the time for solving the flow problem and possibly the adjoint problem and the discrete error accumulated for the solutions on coarser meshes. The time comparison clearly demonstrates the advantage of using error estimation and adjoint-based mesh refinement.

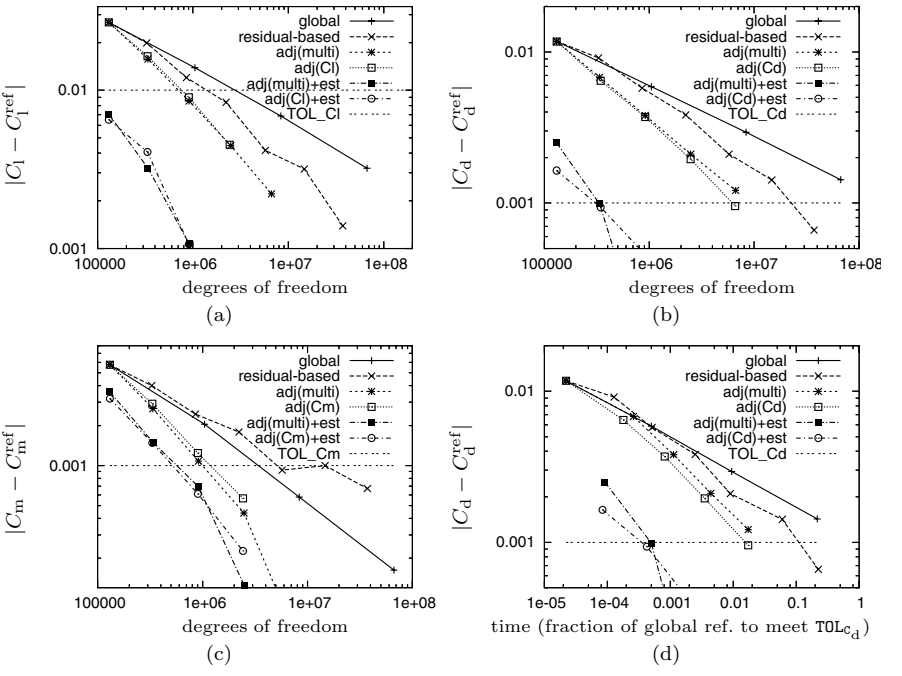


Fig. 2 Error in the (a) lift, (b) drag, and (c) moment coefficient for global, residual-based, adjoint-based(single-target) and adjoint-based(multi-target) mesh refinement vs. number of degrees of freedom. On the adjoint-based refined meshes also the enhanced coefficients $\tilde{C}_{l/d/m} = C_{l/d/m} + \text{est.}$ are given. (d) Error in the drag vs. computing time relative to the extrapolated time required for global mesh refinement to meet the tolerances (12).

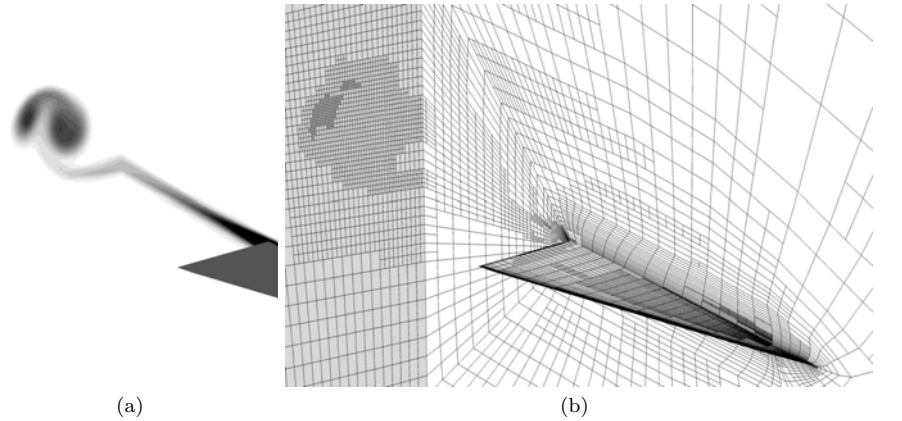


Fig. 3 (a) Mach number isolines of flow solution on (b) the last but one residual-based refined mesh

3.2 ADIGMA BTC1: L1T2 High-Lift Configuration

In this section we consider a turbulent flow around the L1T2 three-element airfoil, see Fig. 5(a), at a Mach number $M = 0.197$, a Reynolds number $Re = 3.52 \cdot 10^6$ and an angle of attack $\alpha = 20.18°$. This case has been documented extensively in the literature, see e. g. [4, 14]. In particular, there is data of two wind tunnel experiments (experiment 1 & 2) available, see [16].

A DG discretization of the RANS-$k\omega$ equations is used which represents a slight modification of the BR2 scheme proposed in [2]. Menter's wall boundary condition is used, where the first wall boundary layer grid spacing y_1 is chosen such that y_1^+ is in the range of one.

First, we compare numerical results generated by the PADGE code with results generated by the well validated finite volume code TAU [17] as well as with experimental data. The PADGE computations were performed with polynomial degrees $p = 3$ and $p = 4$, each on the same quadrilateral mesh with 4 740 curved elements, see Figure 4. This mesh emerged from an original 75 840 element mesh by two agglomeration steps. The curved mesh representation in this case is realized by piecewise polynomials of degree 4 based on additional points which have been extracted from the original mesh.

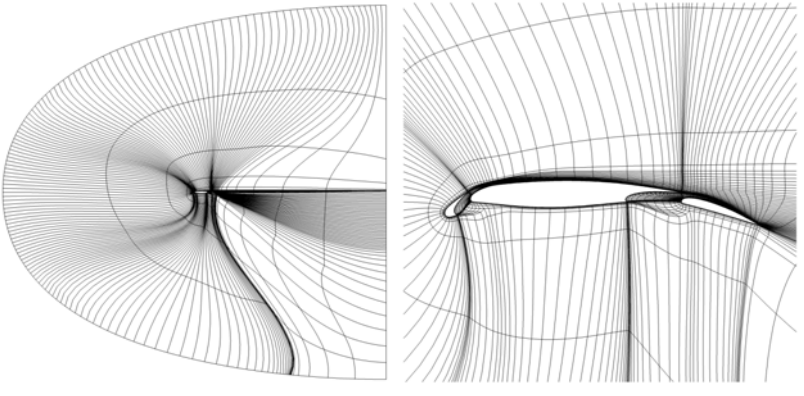


Fig. 4 L1T2 high lift configuration: Coarse grid of 4 740 curved elements

Figure 5(b) shows the pressure distribution over each of the airfoil elements, i.e., slat, main element and flap. Here, we see that the output by the PADGE code is in good agreement with the experimental data and with only minor differences compared to the TAU reference results. Furthermore, Figure 6 shows the comparison for the skin friction distribution. Whereas there are still considerable differences between the computed skin friction distribution for $p = 3$, the result for $p = 4$ is overall in good agreement with the TAU reference computation. We note that with a polynomial degree $p = 4$ on cells near the wall boundary and $p = 3$ everywhere else the c_p and

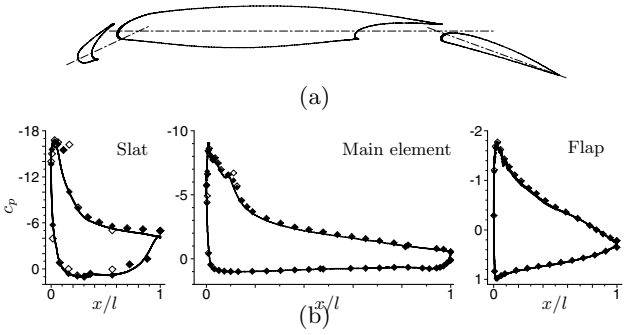


Fig. 5 a) Geometry of the L1T2 three-element airfoil. b) Pressure distributions for each L1T2 airfoil element computed by PADGE (solid line) compared to reference results by TAU (dotted) and data of experiment 1 (open symbols) and experiment 2 (filled), [7].

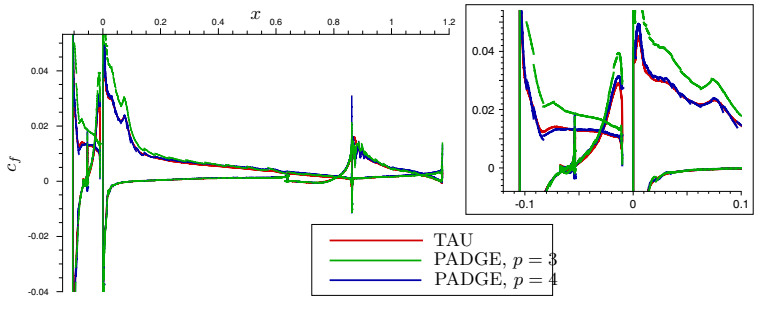


Fig. 6 L1T2 three-element airfoil. Comparison of computed skin friction distributions with details of the slat region, [7].

c_f distributions were almost identical to that with the globally high $p = 4$ polynomials. The $p = 4$ solution has almost as many degrees of freedom as the computation with the TAU code on the original mesh with 75 840 elements and required about the same computing time as the TAU code.

In the following, we investigate the performance of the adjoint-based error estimation and mesh refinement for this test case. Starting with a $p = 1$ solution on the coarse mesh of 4 740 curved elements, we consider the adjoint-based refinement targeted at efficiently approximating the drag coefficient C_{d}. In Figure 7(a) we compare the convergence of C_{d} for the global and the adjoint-based mesh refinement. We see that with the adjoint-based refinement the C_{d} value converges significantly faster to the C_{d} reference value than with global mesh refinement. Furthermore, we see that using the error estimation on the adjoint-based refined meshes for computing enhanced drag coefficients $\tilde{C}_{\mathrm{d}}$ further improves the C_{d} value which demonstrates that the

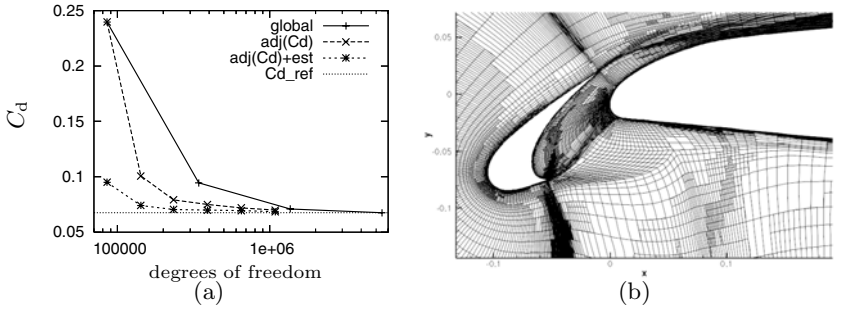


Fig. 7 L1T2 high lift configuration: (a) drag, C_d, coefficient values on globally and adjoint-based refined meshes; on the latter also the enhanced $\tilde{C}_d$ values are given; (b) Zoom of the adjoint-based refined mesh.

error estimation is accurate and reliable. Figure 7(b) shows a zoom of the final adjoint-based refined mesh. We see that the mesh has been refined in the neighborhood of the line which separates the recirculation zone behind the slat from the flow which passes between the slat and the main element. There is some refinement in the wake of the slat. Furthermore, the mesh has been refined in the neighborhood of the stagnation streamline of the main element. We note that, similarly, the stagnation streamlines of the slat and flap are refined. Here, the adjoint solution indicates that the exact position of the stagnation points, as well as the flow upstream of them is particularly important for an accurate prediction of the drag coefficient.

3.3 ADIGMA BTC0: Turbulent Flow around a Streamlined Body

We consider a turbulent flow around a streamlined body at a Mach number $M = 0.5$, an angle of attack $\alpha = 5°$, and a Reynolds number $\mathrm{Re} = 10 \cdot 10^6$ with adiabatic noslip wall boundary conditions. Reference values $J_{C_l}(\mathbf{u}) = 0.006612$ and $J_{C_d}(\mathbf{u}) = 0.0085646$ have been obtained based on higher order computations on very fine grids. The starting mesh of this computation, see Figure 9(a), has 6 656 curved elements. The edges are given by polynomials of degree 4 created by taking additional points from the nested finer grids.

In Figure 8(a) we compare the convergence of C_l for global, residual-based and adjoint-based mesh refinement. We see that within the first refinement step the C_l value for the adjoint-based refinement converges as fast as for the residual-based refinement but both significantly faster than global mesh refinement. However, from the second refinement step onwards the C_l values for the adjoint-based mesh refinement are significantly more accurate than for both residual-based and global mesh refinement. Furthermore, we see that the error estimation on the adjoint-based refined meshes further improves

the C_l value. In fact, computing the flow solution and its adjoint on the coarsest mesh results in an enhanced $\tilde{C}_\mathrm{l}$ value which almost coincides with the reference value. Figure 8(b) shows the corresponding error plot. Here we see that the enhanced C_l value already on the coarsest mesh is more accurate than the prescribed ADIGMA tolerance $\mathtt{TOL}_{C_\mathrm{l}} = 0.001$ and is even more accurate than the C_l value on the finest adjoint-based, residual-based and globally refined meshes. Also, we see that for a stricter convergence criterion, there is an increasing gain from using adjoint-based refinement in comparison to residual-based and global mesh refinement. Figures 8(a)&(b) show the corresponding plots for the C_d value. Here, we see that the enhanced C_d value meets the ADIGMA tolerance, $\mathtt{TOL}_{C_\mathrm{d}} = 0.0003$, already on the first adjoint-based refined mesh. Finally, Figure 9(b) shows the final residual-based refined mesh and Figures 9(c) & (d) show the final adjoint-based refined meshes targeted at the accurate and efficient approximation of the C_l and C_d values, respectively. Here, we see that the adjoint-based refinement is mainly concentrated near the body; indeed, the wake is almost unresolved. This corresponds to the fact, that the flow solution in and near the boundary layer is significantly more important for obtaining accurate aerodynamic force

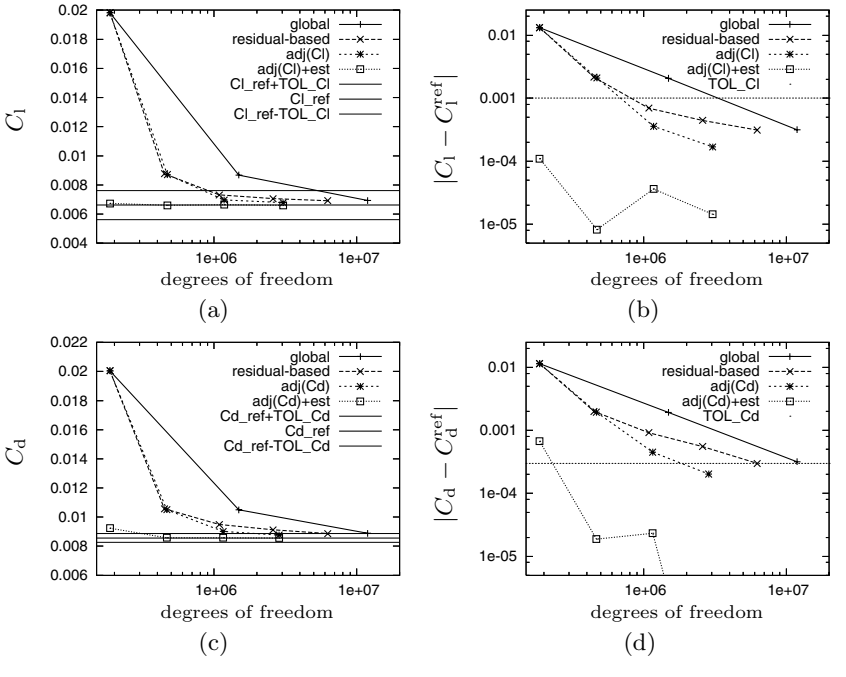


Fig. 8 ADGIMA BTC0 test case at turbulent conditions: (a) lift coefficients C_l on globally and residual-based refined meshes; C_l and the enhanced values, $\tilde{C}_\mathrm{l}$, on adjoint-based refined meshes vs. number of degrees of freedom; (b) the respective error plot. (c)&(d) the respective plots for the drag coefficient C_d.

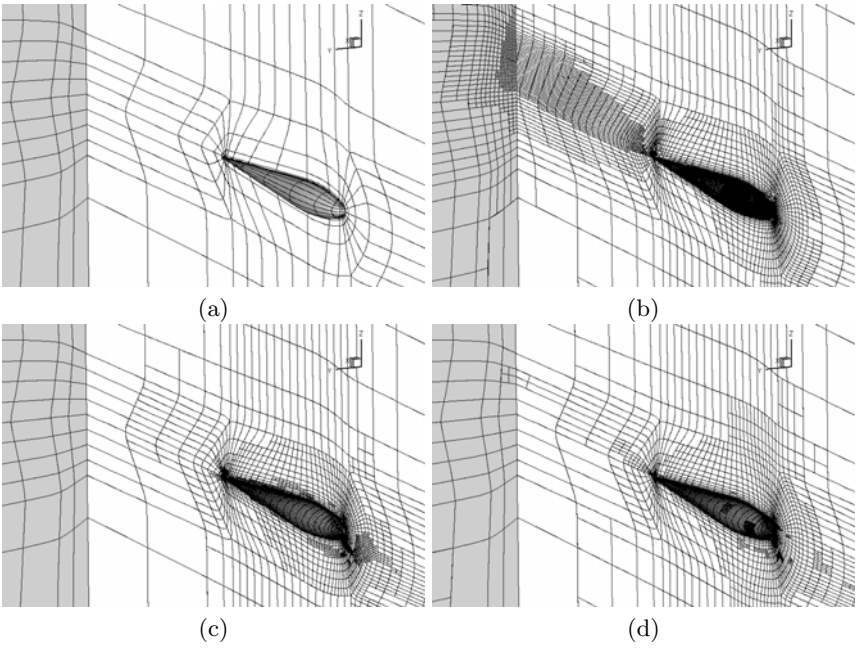


Fig. 9 ADGIMA BTC0 test case at turbulent conditions: (a) The coarse mesh with 6 656 curved elements. The edges are given by polynomials of degree 4. (b) Mesh after 4 residual-based refinement steps. (c)&(d) Meshes after 3 adjoint-based refinement steps targeted at C_l and C_d, respectively.

coefficients than the flow solution in the wake. In contrast to that the residual-based indicator which is targeted at resolving all flow features also refines elements in the vicinity of the wake.

3.4 Subsonic Turbulent Flow around the DLR-F6 Wing-Body Configuration

In this final example we consider a turbulent flow at Mach number $M = 0.5$, a Reynolds number $Re = 5 \cdot 10^6$ at an angle of attack $\alpha = -0.141$ around the DLR-F6 wing-body configuration without fairing. This is a modification of the drag prediction workshop (DPW) III test case. In fact, a fixed angle of attack has been assumed instead of a given target lift. Also, the Mach number has been reduced from originally $M = 0.75$ to $M = 0.5$ in order to ensure that the flow is subsonic. The original DPW mesh of 3.24 mio. hexahedral elements has been agglomerated twice resulting in a coarse mesh of 50 618 hexahedral elements. The additional points of the original mesh have been used to define 50 618 curved elements where the curved lines are represented by quartic polynomials. After some regularization this fifth order

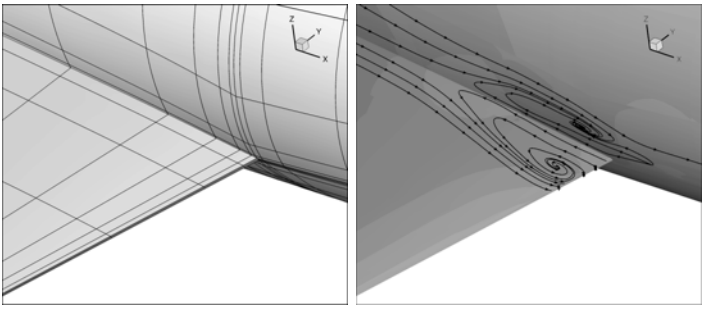


Fig. 10 DLR-F6 wing-body configuration: c_p distribution and wall stream lines of a 4^{th} order solution on the coarse mesh of 50 618 curved elements

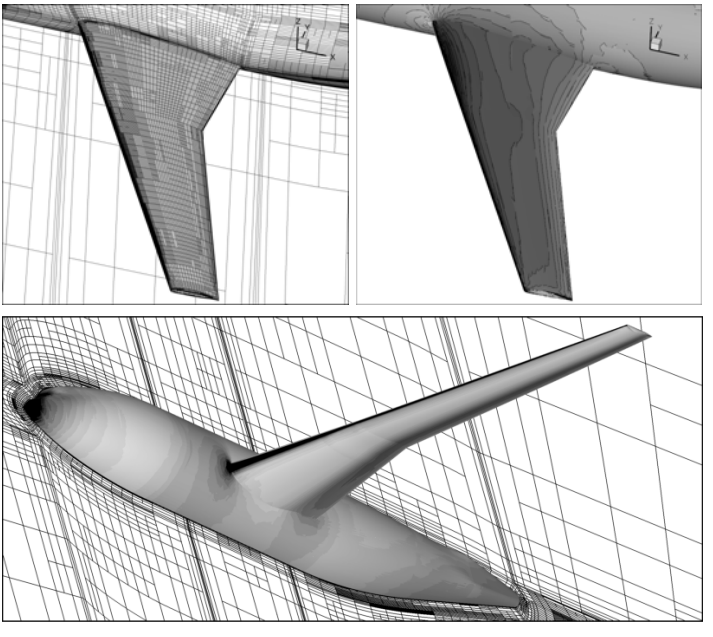


Fig. 11 DLR-F6 wing-body configuration: c_p distribution on mesh of 582 350 curved elements after 4 residual-based mesh refinement steps

mesh has been used in a residual-based and an adjoint-based mesh refinement algorithm.

In Table 1 we collect the C_{l} and C_{d} values obtained by PADGE for the $p = 2, 3$ solutions on the coarsest and for the $p = 2$ solutions on a once globally refined mesh in comparison with the values obtained by TAU on the original mesh. Figure 11 shows the surface mesh near the wing-body junction and the c_p distribution and wall stream lines of a 4^{th} order solution, i.e. for $p = 3$,

Table 1 Subsonic turbulent flow around the DLR-F6 wing-body configuration: Comparison of force coefficients by the PADGE code [7] for $p = 2, 3$ and the TAU code [17]

	coarse mesh		once globally	original mesh
	$p = 2$	$p = 3$	refined, $p = 2$	TAU
C_l	0.424	0.413	0.416	0.423
C_d	0.0270	0.0251	0.0249	0.0237
C_m	-0.122	-0.121	-0.123	-0.125

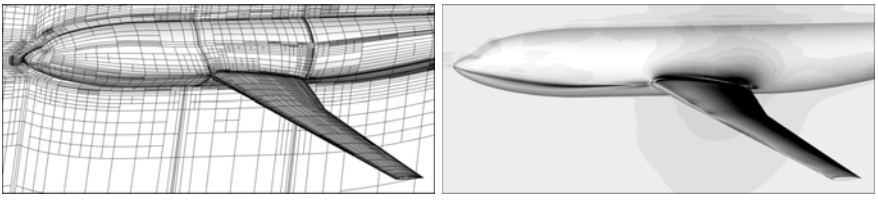


Fig. 12 DLR-F6 wing-body configuration: Density adjoint distribution, i.e., the first comp. of discrete adjoint solution $\tilde{\mathbf{z}}_h$ on a mesh of 202 314 curved elements after two adjoint-based mesh refinement steps targeted at C_l.

on the coarse mesh. We clearly recognize a separation of the flow. Figure 11 shows the c_p distribution on a mesh of 582 350 curved elements after four anisotropic residual-based mesh refinement steps. Finally, Figure 12 shows an example of an adjoint-based refined mesh; here for the target quantity C_l, together with the adjoint solution connected to the C_l value.

4 Summary

Within the EU-project ADIGMA the techniques of error estimation, residual-based and adjoint-based mesh refinement have been extended from 2d laminar flows to 3d laminar and turbulent flows. They have been implemented in the PADGE code [7] and successfully applied to various aerodynamic test cases including a vortex dominated laminar flow around a delta wing, a turbulent flow around the L1T2 three-element high lift configuration and a turbulent flow around the DLR-F6 wing-body configuration. Furthermore, the error estimation and adjoint-based mesh refinement have been extended from single target quantities to the treatment of multiple target quantities.

The residual-based indicators which are targeted at resolving all flow features have been shown to well resolve vortical systems over long distances. Furthermore, it has been shown that using error estimation and adjoint-based mesh refinement the aerodynamic force coefficients can be approximated significantly more accurate and more efficient than with residual-based and global mesh refinement.

Current and future research is dedicated to extending the adaptation algorithms from isotropic to anisotropic mesh refinement [13] as well as to *hp*-refinement. The flow solver PADGE will be extended from purely hexahedral to hybrid meshes. Furthermore, a *p*-multilevel algorithm for 3d turbulent flows will be developed in order to replace the current implicit solver which relies on the storage of the full Jacobian matrix.

References

1. Bangerth, W., Hartmann, R., Kanschat, G.: deal.II – A general purpose object oriented finite element library. ACM Trans. on Math. Software 33(4) (August 2007)
2. Bassi, F., Crivellini, A., Rebay, S., Savini, M.: Discontinuous Galerkin solution of the Reynolds-averaged Navier-Stokes and $k-\omega$ turbulence model equations. Computers & Fluids 34, 507–540 (2005)
3. Becker, R., Rannacher, R.: An optimal control approach to a posteriori error estimation in finite element methods. Acta Numerica 10, 1–102 (2001)
4. Fejtek, I.: Summary of code validation results for a multiple element airfoil test case. In: 28th AIAA fluid dynamics conference, AIAA Paper 97-1932 (1997)
5. Hartmann, R.: Adaptive discontinuous Galerkin methods with shock-capturing for the compressible Navier-Stokes equations. Int. J. Numer. Meth. Fluids 51(9-10), 1131–1156 (2006)
6. Hartmann, R.: Multitarget error estimation and adaptivity in aerodynamic flow simulations. SIAM J. Sci. Comput. 31(1), 708–731 (2008)
7. Hartmann, R., Held, J., Leicht, T., Prill, F.: Discontinuous Galerkin methods for computational aerodynamics – 3D adaptive flow simulation with the DLR PADGE code. Aerospace Science and Technology (2009) (Submitted)
8. Hartmann, R., Houston, P.: Adaptive discontinuous Galerkin finite element methods for nonlinear hyperbolic conservation laws. SIAM J. Sci. Comput. 24, 979–1004 (2002)
9. Hartmann, R., Houston, P.: Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations. J. Comput. Phys. 183(2), 508–532 (2002)
10. Hartmann, R., Houston, P.: Goal-oriented a posteriori error estimation for multiple target functionals. In: Hou, T.Y., Tadmor, E. (eds.) Hyperbolic problems: theory, numerics, applications, pp. 579–588. Springer, Heidelberg (2003)
11. Hartmann, R., Houston, P.: Symmetric interior penalty DG methods for the compressible Navier–Stokes equations II: Goal–oriented a posteriori error estimation. Int. J. Num. Anal. Model. 3(2), 141–162 (2006)
12. Hartmann, R., Houston, P.: An optimal order interior penalty discontinuous Galerkin discretization of the compressible Navier–Stokes equations. J. Comput. Phys. 227(22), 9670–9685 (2008)
13. Hartmann, R., Leicht, T.: Error estimation and anisotropic mesh refinement for 3d aerodynamic flow simulations. J. Comput. Phys (2009) (Submitted)
14. Hellsten, A.: New Two-Equation Turbulence Model for Aerodynamics Applications. Technical Report Report No. A-21, Helsinki University of Technology, Laboratory of Aerodynamics (2004)

15. Klaij, C.M., van der Vegt, J.J.W., van der Ven, H.: Space–time discontinuous Galerkin method for the compressible Navier-Stokes equations. J. Comput. Phys. 217(2), 589–611 (2006)
16. Moir, I.R.M.: Measurements on a two-dimensional aerofoil with high-lift devices. AGARD Advisory Report 303, Advisory Group for Aerospace Research & Development, Neuilly-sur-Seine, Test case A2 (1994)
17. Schwamborn, D., Gerhold, T., Heinrich, R.: The DLR TAU-code: Recent applications in research and industry. In: Wesseling, P., Oñate, E., Périaux, J. (eds.) Proceedings of European Conference on Computational Fluid Dynamics, ECCOMAS CDF 2006, Delft, The Netherland, pp. 91–100 (2006)

This page intentionally left blank

Chapter 25
Adjoint–Based Correction of Aerodynamic Coefficients on Structured Multiblock Grids

L. Tourrette, M. Meaux, and A. Barthet

Abstract. The work performed at AIRBUS Operations S.A.S on adjoint–based error estimation of functional outputs is presented. The ultimate goal of this technique aims at identifying areas where local mesh refinement and/or nodes displacement may be applied to get closer to grid convergence. The method also provides a correction term allowing to estimate the value of a given functional output if the grid were globally refined. The theoretical background is recalled and the two interpolation techniques implemented by way of prolongation operator are detailed. Also, improvements to the numerical method for the adjoint state are described. The adjoint–based correction is assessed for the lift and drag coefficients on two industrially relevant test cases, the DLR–F6 Wing–Body and Wing–Body–Pylon–Nacelle configurations.

1 Introduction

The work described herein falls within the framework of goal–oriented adaptation and is based on preliminary results compiled in the PhD thesis of A. Barthet ([1]). The ultimate goal of the implemented methodology aims at identifying areas where local mesh refinement and/or nodes displacement may be applied to get closer to grid convergence. As a by–product, the method also provides a correction term allowing to estimate the value of a given functional output if the grid were globally refined. The ability to compute the adjoint state relative to the chosen functional output is an essential pre–requisite of this approach.

L. Tourrette · M. Meaux
AIRBUS Operations S.A.S., Toulouse, France
e-mail: loic.tourrette@airbus.com, mathieu.meaux@airbus.com

A. Barthet
ALTRAN Sud–Ouest, Toulouse, France
e-mail: arnaud.barthet@altran.com

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 355–368, 2010.
springerlink.com

The basic principles of the method are presented in section 2.

For the coarse grid to fine grid prolongation operator, interpolation methods showed a much greater potential than local reconstruction techniques, especially with respect to high order accuracy. Two interpolation techniques, namely *trilinear interpolation* and *tricubic interpolation*, have been implemented. They are detailed in section 3.

Modifications of the numerical method devoted to the resolution of the adjoint state are described in section 4. They include a new artificial dissipation model for accuracy and an elliptic smoother to enhance the robustness of the iterative algorithm.

Section 5 is dedicated to the assessment of the adjoint–based correction for the lift and drag coefficients on the DLR–F6 Wing–Body and Wing–Body–Pylon–Nacelle configurations which are representative of the cases that are daily run at AIRBUS by design engineers.

Finally, conclusive remarks are drawn with an outlook on future work in section 6.

All flow solutions have been computed with the structured multi–block suite built around the *elsA* flow solver developed at ONERA and the adjoint solutions have been produced using the *Optalia* optimization suite based upon the same *elsA* version.

2 Fundamentals of the Approach

The theoretical approach is described in details in [1]. The fundamentals of the method are recalled below.

Let w_H be the solution of the RANS equations on grid G_H:

$$R_H(w_H) = 0 \tag{1}$$

and let w_h be the solution of the RANS equations on grid G_h ($h = H/2$):

$$R_h(w_h) = 0 \tag{2}$$

For a given functional output F (e.g. *lift* or *drag* coefficient), we wish to compute a correction term C_H that would bring F_H closer to F_h:

$$F_h \approx F_H + C_H \tag{3}$$

Let u_h be a perturbation of w_h. A Taylor expansion gives:

$$F_h(w_h) \approx F_h(u_h) + \frac{\partial F_h}{\partial w_h}(u_h)(w_h - u_h) \tag{4}$$

Similarly, a Taylor expansion of the residual gives:

$$R_h(w_h) \approx R_h(u_h) + \frac{\partial R_h}{\partial w_h}(u_h)(w_h - u_h) = 0 \tag{5}$$

The error $w_h - u_h$ is then given by:

$$w_h - u_h = -\left[\frac{\partial R_h}{\partial w_h}(u_h)\right]^{-1} R_h(u_h) \tag{6}$$

that can be substituted into the Taylor expansion of the functional output F. Introducing the adjoint state λ_h (based on u_h), defined by:

$$\left[\frac{\partial R_h}{\partial w_h}(u_h)\right]^T \lambda_h = -\left[\frac{\partial F_h}{\partial w_h}(u_h)\right]^T \tag{7}$$

the Taylor expansion of the functional output F finally reads:

$$F_h(w_h) \approx F_h(u_h) + \lambda_h^T R_h(u_h) \tag{8}$$

In the above correction term, the adjoint state should be computed on the fine grid. This is not viable from an industrial point of view and it is replaced by a prolongation of λ_H onto G_h:

$$\lambda_h \approx L_H^h \lambda_H \tag{9}$$

In a similar way, u_h is chosen as a prolongation of w_H onto G_h:

$$u_h \approx L_H^h w_H \tag{10}$$

3 Interpolation

The prolongation operator has to be selected with great care, in particular regarding the adjoint state, since the decision that consists in replacing the adjoint state on the fine grid by a prolongation of the adjoint state computed on the coarse grid may be risky.

In a first phase, the study was focused on local reconstruction techniques but extension to higher order appeared quite problematic. Considering the good results obtained with the trilinear interpolation used at that time in comparison to the linear local reconstruction technique, it has been decided to put the emphasis on interpolation methods instead.

In the present approach, the fine grid is defined through a global refinement of the coarse grid, in which each segment is divided into two equal–sized sub–segments. As a consequence, each coarse cell centre (resp. boundary face centre) coincides with a fine grid node. The interpolation methods rely on this assumption, which considerably simplifies the implementation.

For the two interpolation methods described below, the first step consists in transferring the values at coarse cell and boundary face centres onto the corresponding fine grid nodes. Then the interpolation proceeds in three steps: (1) direction I, (2) direction J and (3) direction K. As an essential pre–requisite to this three steps interpolation, for each fine grid block, values have to be prescribed for (a) every second node on each of the twelve edges and (b) the eight corners. This is done through an averaging based on the inverse distance to the closest nodes (2 neighbours for an edge node, 3 neighbours for a corner node).

Finally, on the fine grid, the values at cell (resp. boundary face) centres are obtained by an arithmetic averaging of the height (resp. four) nodal values.

3.1 Parameterization

Each interpolation method is based on a parameterization of the fine grid. Each node (I,J,K) in a given block is identified with three parameters $u_{I,J,K}$, $v_{I,J,K}$ and $w_{I,J,K}$. The parameter $u_{I,J,K}$ is defined in the following way:

$$u_{I,J,K} = \begin{cases} 0 & \text{for } I = 1 \\ \sum_{i=2}^{I} d(P_{i-1,J,K}, P_{i,J,K}) & \text{for } I \geq 2 \end{cases} \tag{11}$$

In equation 11, $d(P_{i-1,J,K}, P_{i,J,K})$ denotes the Euclidian distance between grid nodes $(i-1,J,K)$ and (i,J,K). Similar definitions hold for the two other parameters $v_{I,J,K}$ and $w_{I,J,K}$.

3.2 Trilinear Interpolation

On a given mesh line, let P_n and P_{n+2} be two nodes where the values ϕ_n and ϕ_{n+2} are known and let P_{n+1} be a node in–between where the value ϕ_{n+1} has to be interpolated. With t_n, t_{n+1} and t_{n+2} the corresponding values of the parameter ($t = u$, v or w depending on the nature of the mesh line), the interpolated value ϕ_{n+1} is defined through

$$\phi_{n+1} = (1-\alpha)\phi_n + \alpha\phi_{n+2}, \alpha = \frac{t_{n+1} - t_n}{t_{n+2} - t_n} \tag{12}$$

The great advantage of linear interpolation is that it inherently preserves positivity, which is of utmost importance for the turbulence variables.

3.3 Tricubic Interpolation

The tricubic interpolation is based on the use of cubic spline curves, which are particularly well presented in reference [2].

Suppose that we are given $N+1$ points $P_0, \ldots, P_N$ and associated parameter values $t_i \in [a,b]$ satisfying

$$a = t_0 < t_1 < \cdots < t_{N-1} < t_N = b \tag{13}$$

We denote the length of the subinterval $[t_i, t_{i+1}]$ by $\Delta t_i = t_{i+1} - t_i$. For each neighbouring pair of points (P_i, P_{i+1}), we wish to construct an interpolating cubic curve segment

$$X_i(t) = A_i(t-t_i)^3 + B_i(t-t_i)^2 + C_i(t-t_i),\ t \in [t_i, t_{i+1}],\ i = 0, \ldots, N-1 \qquad (14)$$

The collection of all curve segments will make up the cubic interpolating spline curve s. We require that s be twice continuously differentiable at each of the break points $P_1, \ldots, P_{N-1}$:

$$\begin{cases} X_i(t_{i+1}) = X_{i+1}(t_{i+1}) \\ X_i'(t_{i+1}) = X_{i+1}'(t_{i+1}) \\ X_i''(t_{i+1}) = X_{i+1}''(t_{i+1}) \end{cases} \qquad (15)$$

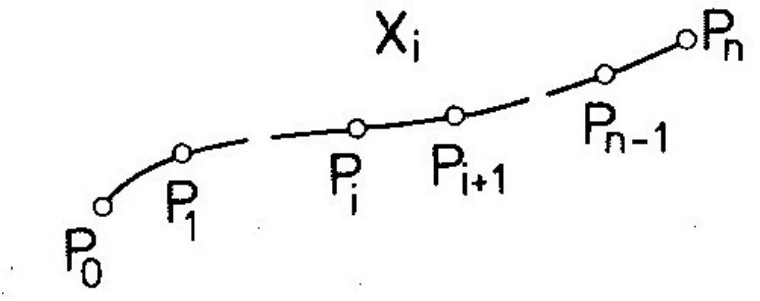


Fig. 1 The spline segment X_i

This leads to the so–called *Hermite* or *Ferguson* formula for a cubic spline curve:

$$X_i(t) = \phi_i(2u^3 - 3u^2 + 1) + \phi_{i+1}(-2u^3 + 3u^2) + \Delta t_i \left[\phi_i'(u^3 - 2u^2 + u) + \phi_{i+1}'(u^3 - u^2)\right] \qquad (16)$$

with $u = (t - t_i)/(\Delta t_i)$ and where ϕ_i and ϕ_i' (resp. ϕ_{i+1} and ϕ_{i+1}') are the value and slope at point P_i (resp. P_{i+1}) of the field being interpolated.

In general, the derivatives ϕ_i' are unknown. They can be estimated (i.e. explicitly prescribed) or alternatively, they can be determined from the requirement that s be C^2 continuous, which leads to the resolution of a tridiagonal linear system of N-1 equations in the N+1 unknowns $\phi_0', \ldots, \phi_N'$. For $i = 1, \ldots, N-1$, the i^{th} line of this system reads:

$$\Delta t_i \phi_{i-1}' + 2(\Delta t_{i-1} + \Delta t_i)\phi_i' + \Delta t_{i-1}\phi_{i+1}' = 3\frac{\Delta t_{i-1}}{\Delta t_i}(\phi_{i+1} - \phi_i) + 3\frac{\Delta t_i}{\Delta t_{i-1}}(\phi_i - \phi_{i-1}) \qquad (17)$$

There are several ways of choosing the two free boundary conditions to close the system (see [2]). For the test cases presented in section 5, the so–called *natural*

spline boundary condition has been adopted. It requires that the second derivatives vanish at the boundary points ϕ_0 and ϕ_N and leads to the following equations:

$$\begin{cases} \Delta t_0(2\phi'_0 + \phi'_1) = -3\phi_0 + 3\phi_1 \\ \Delta t_{N-1}(\phi'_{N-1} + 2\phi'_N) = -3\phi_{N-1} + 3\phi_N \end{cases} \tag{18}$$

4 Numerical Method for the Adjoint State

On the Wing–Body (WB) and Wing–Body–Pylon–Nacelle (WBPN) configurations, it has been observed during the first investigations that the lift correction was quite sensitive to the level of artificial dissipation in the numerical scheme for the adjoint state. With the prospect to reduce the amount of artificial dissipation, several numerical damping formulations already implemented in the *elsA* solver and using a sensor–based blending of first and third differences have been investigated, but all attempts were fruitless. A new artificial dissipation model based on Jameson's SLIP scheme (see [3], [4]) has been successfully implemented and is described in subsection 4.1.

In addition, the robustness of the iterative method for the resolution of the adjoint state, presented in [5], has been enhanced by introducing an elliptic smoothing, with constant coefficient, detailed in subsection 4.2.

4.1 Artificial Dissipation Model

The artificial dissipation model originally used involved second differences scaled by $k^{(2)}\rho$, where ρ denotes the spectral radius of the Jacobian matrix of the Euler equations,

$$\rho = |\mathbf{V} \cdot \mathbf{S}| + cS \tag{19}$$

and $k^{(2)}$ is a user defined coefficient usually set to 0.05. On the WB case, we tried to reduce the value of $k^{(2)}$ but the adjoint calculation could not converge anymore for values below 0.04. Moreover, with 0.04, the correction for the lift became worse than with 0.05, whereas with values higher than 0.05, it improved. Three other artificial dissipation models were available within the *elsA* solver, derived from Jameson's standard scalar centred scheme, characterized by different ways of blending the first and third differences in the numerical flux for the artificial dissipation. None of these approaches was able to converge on the wing body case and we decided to introduce a new model.

Derived from Jameson's work developed in references [3] and [4], the implementation of this new artificial dissipation model is based on a new implementation of limiters inspired from upwind schemes. In order to define the artificial dissipation flux at interface $i+1/2$ between cells i and $i+1$, let us introduce the left and right adjacent states λ_L and λ_R. The artificial dissipation flux is then defined by

$$d_{i+1/2} = \alpha\rho_{i+1/2}(\lambda_R - \lambda_L) \tag{20}$$

Let σ_i and σ_{i+1} be the slopes at the centres of cells i and $i+1$ respectively. In the absence of limitation, the left and right states are defined in the following way:

$$\lambda_L = \lambda_i + \frac{1}{2}\sigma_i,\ \lambda_R = \lambda_{i+1} - \frac{1}{2}\sigma_{i+1} \tag{21}$$

and the numerical flux in equation 20 becomes:

$$d_{i+1/2} = \alpha\rho_{i+1/2}\left[\Delta\lambda_{i+1/2} - A(\sigma_i, \sigma_{i+1})\right] \tag{22}$$

with

$$A(\sigma_i, \sigma_{i+1}) = \frac{1}{2}(\sigma_i + \sigma_{i+1}),\ \Delta\lambda_{i+1/2} = \lambda_{i+1} - \lambda_i \tag{23}$$

The final form of the artificial dissipation flux is obtained by substituting a limited average L to the arithmetic mean A:

$$d_{i+1/2} = \alpha\rho_{i+1/2}\left[\Delta\lambda_{i+1/2} - L(\sigma_i, \sigma_{i+1})\right] \tag{24}$$

Jameson suggests several limiters in [3]. In our case, we have selected Van Albada's limiter, since the equations for the adjoint state have been obtained by differentiating Roe's second order upwind scheme with Van Albada's limiter. The definition of this limiter is recalled below:

$$L(a,b) = \begin{cases} 0 & \text{if } ab \leq 0 \\ \frac{ab(a+b)}{a^2+b^2} & \text{otherwise} \end{cases} \tag{25}$$

The slopes σ_i and σ_{i+1} can be defined in two ways, giving rise to two variants of the artificial dissipation model. The first choice corresponds to Jameson's SLIP scheme (Symmetric LImited Positive):

$$\sigma_i = \sigma_{i+1} = L(\Delta\lambda_{i-1/2}, \Delta\lambda_{i+3/2}) \tag{26}$$

i.e. a unique slope is used in both adjacent cells, which is a limited average of the slopes across the two faces adjacent to face $i+1/2$. The second choice is coherent with the implementation in *elsA* of the MUSCL extrapolation for Roe's second order upwind scheme:

$$\sigma_i = L(\Delta\lambda_{i-1/2}, \Delta\lambda_{i+1/2}),\ \sigma_{i+1} = L(\Delta\lambda_{i+1/2}, \Delta\lambda_{i+3/2}) \tag{27}$$

i.e. the slope at the centre of the cell is a limited average of the slopes across its left and right faces.

It can be easily verified that the second variant is twice less dissipative than the first one. Regarding the value of α, Jameson demonstrates in [3] that the scheme will be LED (Local Extremum Diminishing) as soon as $\alpha \geq 1/2$.

4.2 Implicit Smoothing

Let us consider the three-dimensional parabolic equation

$$\frac{\partial R}{\partial t} = \nu \Delta R = \nu \left(\frac{\partial^2 R}{\partial x^2} + \frac{\partial^2 R}{\partial y^2} + \frac{\partial^2 R}{\partial z^2} \right) \tag{28}$$

An implicit discretization on a Cartesian grid (orthogonal grid with $\Delta x = \Delta y = \Delta z$) gives:

$$R_{i,j,k}^{n+1} = R_{i,j,k}^{n} + VNN \left(\delta_I^{(2)} + \delta_J^{(2)} + \delta_K^{(2)} \right) R_{i,j,k}^{n+1} \tag{29}$$

where $\delta_I^{(2)}$, $\delta_J^{(2)}$ and $\delta_K^{(2)}$ are the second difference operators along I, J and K directions respectively. VNN is the Von Neumann number:

$$VNN = \nu \frac{\Delta t}{\Delta x^2} = \nu \frac{\Delta t}{\Delta y^2} = \nu \frac{\Delta t}{\Delta z^2} \tag{30}$$

The linear system defined by equation 29 is solved using the symmetric Gauss–Seidel iterative method. Homogeneous Neumann boundary conditions are applied at block boundaries.

As explained in [5], the adjoint state λ is solution of a linear system $A\lambda = b$. An iterative process solves this linear system:

$$A' \Delta \lambda^n = b - A\lambda^n \tag{31}$$

where A' is a diagonal dominant approximation of A. The smoothing operator defined by equation 29 can be applied to $b - A\lambda^n$ (pre-smoothing) and/or $\Delta \lambda^n$ (post-smoothing). Experience has shown that pre– and post–smoothing were equivalently efficient and that it was not worthwhile to combine pre– and post–smoothing.

Initially, implicit smoothing aimed at replacing the introduction of numerical damping terms. Unfortunately, on complex cases, it was not possible to get rid of artificial dissipation. However, since convergence was noticeably improved, it has been decided to make a systematic use of implicit smoothing to enhance robustness by damping possible high frequency modes in the error.

5 Assessment on Complex Test Cases

The methodology described in section 2 has been assessed on a series of test cases with increasing complexity to compute the adjoint–based corrections of the lift and drag coefficients. This section presents the results obtained for the two most complex cases, namely the DLR–F6 Wing–Body and Wing–Body–Pylon–Nacelle configurations of the DPW2.

The numerical method is Roe's second order upwind scheme with Van Albada's limiter, since it is the numerical scheme underlying the discrete adjoint approach implemented in *elsA* and Menter's $k - \omega$ *BSL* turbulence model has been employed.

The "coarse" grid is the existing grid, the "fine" grid is generated by global refinement of the coarse grid and a converged flow solution is obtained on both grids. Then, the adjoint states for C_L and C_D are computed on the coarse grid. For a given aerodynamic coefficient, the coarse flow solution and adjoint state are interpolated on the fine grid and the flux differences based on the interpolated flow solution are evaluated. Finally, the adjoint–correction is computed according to equation 8. In the tables below, the ideal correction is defined as the difference between the values of the aerodynamic coefficient on the fine grid from the converged solution and from the prolongated flow field. The drag coefficient is expressed in drag counts.

The adjoint states relative to C_D and C_L have been computed using the first variant (SLIP scheme) of the new artificial dissipation model with $\alpha = 1/2$. Implicit post–smoothing has been switched–on, with two iterations and $VNN = 0.2$.

The geometry does not include the FX2B fairing designed to alleviate the side–of–body separation on the wing.

5.1 DLR–F6 Wing–Body Configuration

The aerodynamic configuration is defined as follows:

$M_\infty = 0.75,\ \alpha = 0.52^\circ,\ Re_1 = 21.25\ 10^6\ m^{-1}$.

The in–house generated grid contains 29 blocks and 3615401 nodes. A view of the mesh on the skin and in the symmetry plane can be seen on figure 2. The convergence history for the flow solution on the coarse grid displayed on figure 3 exhibits

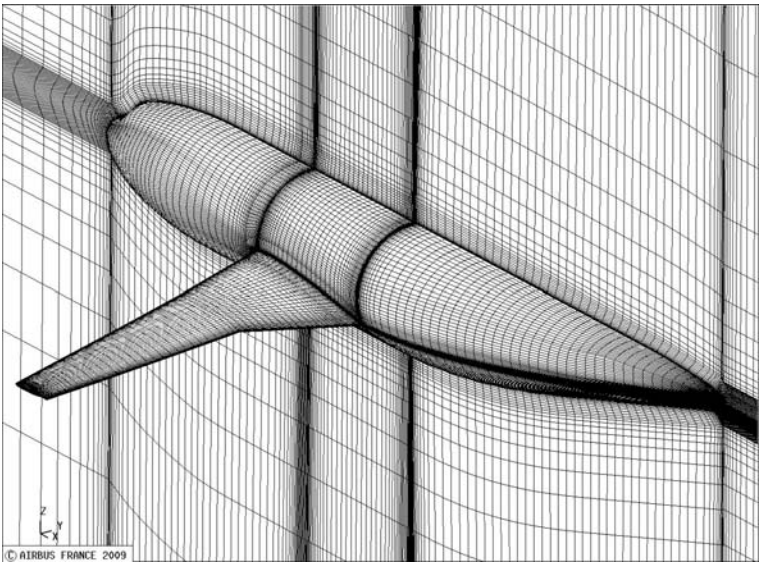


Fig. 2 Coarse grid – DLR–F6 Wing–Body

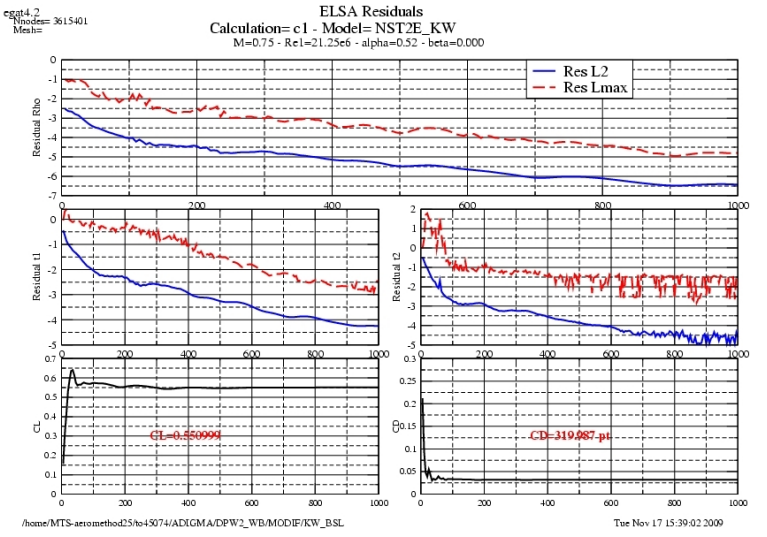


Fig. 3 Convergence history on the coarse grid – DLR–F6 Wing–Body

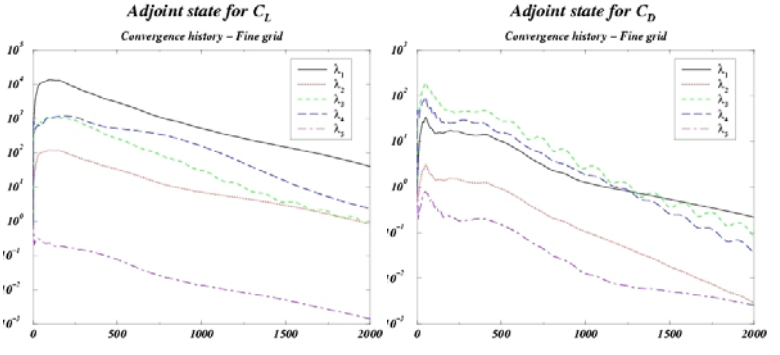


Fig. 4 Convergence history for the adjoint states – DLR–F6 Wing–Body

perfectly stabilized aerodynamic coefficients. On the fine grid, the flow solution, computed with 4 multigrid levels, is very well converged too (not shown). The convergence histories shown on figure 4 indicate that the numerical method for the adjoint state behaves quite satisfactorily. The adjoint–based error estimates for C_L and C_D are given in tables 1 and 2 for the tricubic and trilinear prolongation operators respectively. They have the proper sign. However, for the lift, the absolute value is fairly underestimated whereas for the drag, it is slightly overestimated. As regards the drag coefficient, the differences between the two interpolators are marginal while for the lift coefficient, the underestimation is more pronounced in the trilinear case.

Table 1 Corrections using the tricubic interpolation – DLR–F6 Wing–Body

	Ideal correction	Computed correction	Ratio (%)
C_L	$-10.41\ 10^{-3}$	$-2.47\ 10^{-3}$	23.7
C_D	-10.19	-11.65	114.3

Table 2 Corrections using the trilinear interpolation – DLR–F6 Wing–Body

	Ideal correction	Computed correction	Ratio (%)
C_L	$-10.43\ 10^{-3}$	$-1.56\ 10^{-3}$	15.0
C_D	-10.90	-12.30	112.8

5.2 DLR–F6 Wing–Body–Pylon–Nacelle Configuration

The aerodynamic configuration is given by:

$$M_\infty = 0.75,\ \alpha = 1^\circ,\ Re_1 = 21.25\ 10^6\ m^{-1}.$$

The in–house generated grid is made of 62 blocks, with a total of 5173658 nodes. A view of the grid on the skin and in the symmetry plane is shown on figure 5. The convergence history for the flow solution on the coarse grid, depicted on figure 6, demonstrates that the aerodynamic coefficients are perfectly stabilized. On the fine

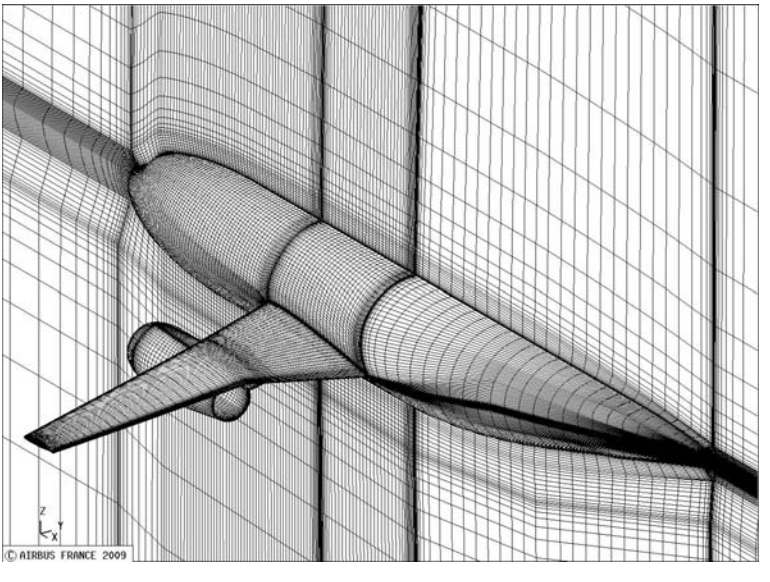


Fig. 5 Coarse grid – DLR–F6 Wing–Body–Pylon–Nacelle

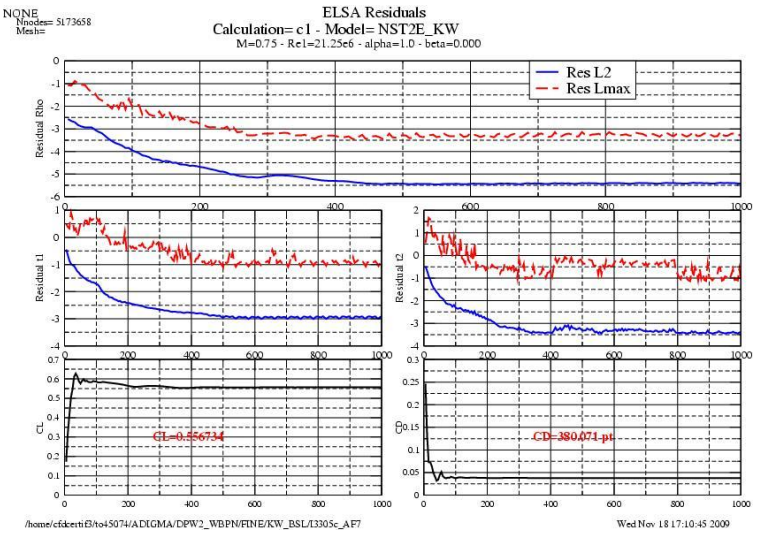


Fig. 6 Convergence history on the coarse grid – DLR–F6 Wing–Body–Pylon–Nacelle

grid, the flow solution, computed with 4 multigrid levels, is also extremely well converged (not shown). The convergence histories on figure 7 indicate again that the numerical scheme for the adjoint state is quite efficient and robust. The adjoint–

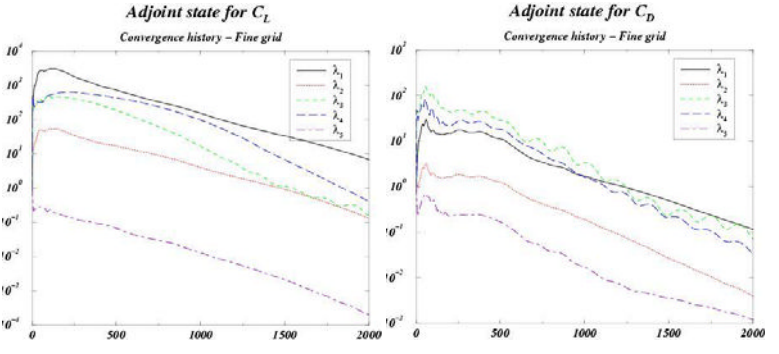


Fig. 7 Convergence history for the adjoint states – DLR–F6 Wing–Body–Pylon–Nacelle

based error estimates for the lift and drag coefficients are presented in tables 3 and 4 for the tricubic and trilinear interpolation techniques respectively. The computed corrections have the right sign. For the lift coefficient, however, the absolute value is again quite underestimated. The differences between the two prolongation operators are minor.

Table 3 Corrections using the tricubic interpolation – DLR–F6 Wing–Body–Pylon–Nacelle

	Ideal correction	Computed correction	Ratio (%)
C_L	$-1.16\ 10^{-2}$	$-3.88\ 10^{-3}$	33.4
C_D	-23.47	-21.68	92.4

Table 4 Corrections using the trilinear interpolation – DLR–F6 Wing–Body–Pylon–Nacelle

	Ideal correction	Computed correction	Ratio (%)
C_L	$-1.17\ 10^{-2}$	$-3.32\ 10^{-3}$	28.4
C_D	-23.75	-22.95	96.6

6 Conclusions and Outlook

A software package, acting as a post–processing tool, has been developed for evaluating adjoint–based error estimates of functional outputs, in the framework of structured multiblock grids. Two interpolation techniques have been implemented and assessed for the adjoint–based correction of the lift and drag coefficients on a series of test cases of increasing complexity, including a Wing–Body and a Wing–Body–Pylon–Nacelle. The results obtained for these two industrially relevant configurations have been presented herein.

First attempts with the WB and WBPN cases lead to erroneous error estimates for the lift coefficient and revealed a high sensitivity to the level of artificial dissipation in the numerical scheme for the adjoint state. With the prospect to reduce the amount of numerical damping, a new model inspired from Jameson's SLIP scheme has been implemented into the *elsA* solver. In addition, the robustness of the numerical method for the adjoint state has been enhanced by the introduction of an elliptic smoother. The new artificial dissipation model dramatically improved the lift correction.

The main objective of this contribution was to assess the potential of the adjoint–based correction method for relevant functional outputs on test cases sufficiently complex to be representative of industrial applications. If the interest of the approach for industry is established, it can then be used to improve the accuracy for the prediction of aerodynamic coefficients and design new sensors allowing more economical grid refinement or grid quality improvement. Predicting aerodynamic coefficients with higher accuracy would also have a positive impact on optimum design applications.

The results presented in section 5 look quite promising, but the problem of the underestimation of the correction for the lift remains to be solved.

Before the adjoint–based correction method becomes a fully industrial tool, several issues still need to be addressed:

- A separate tool should be implemented for computing the flux differences based on the prolongated flow field. It would work on a block–by–block basis, in order to minimize the memory requirement for large–scale problems. The *elsA* solver presently assumes this task.
- Other objective functions than C_L and C_D should be considered that are of great interest for design engineers as e.g. wave drag C_{D_w} or induced drag C_{D_i} but difficulties may be expected when the definition of the functional output gets more complex.
- The adjoint part of the solver should be extended to grids with non–matching block interfaces and chimera grids, which are more and more used for aircraft configurations. The developments are under way in research establishments.
- The multigrid technique should be applied to accelerate the convergence of the iterative method for the adjoint state. This would also benefit optimum design studies. The development has been scheduled and should start soon.
- Application of the adjoint–based correction to grid improvement (nodes displacement and/or grid refinement based on relevant sensors) should be considered. Concerning mesh enrichment, a necessary pre–requisite is that the *AMR* technique available in the *elsA* solver proves to be mature enough and easy to use, which may require a substantial effort. It would also be necessary to extend the numerical method for the adjoint state to *AMR* grids.

References

1. Barthet, A.: Amélioration de la prévision des coefficients aérodynamiques autour de configurations portantes par méthode adjointe, Arnaud Barthet, thèse de l'I.N.P de Toulouse (mai 31, 2007)
2. Hoschek, J., Lasser, D.: Fundamentals of computer aided geometric design. AK Peters, Ltd., Stanford (1993)
3. Jameson, A.: Analysis and design of numerical schemes for gas dynamics, 1: artificial diffusion, limiters and their effect on accuracy and multigrid convergence. International Journal of CFD 4, 171–218 (1995)
4. Jameson, A.: Analysis and design of numerical schemes for gas dynamics, 2: artificial diffusion and discrete shock structure. International Journal of CFD 5, 1–38 (1995)
5. DTP Optimization phase 2: final report T0+24, AIRBUS France technical report RP0624744, 10/09/2006

Chapter 26
Goal-Oriented Mesh Adaptation in an Industrial Stabilized Finite Element Navier-Stokes Code

Frédéric Chalot

Abstract. This chapter describes Dassault Aviation's contribution to Workpackage 5 of the ADIGMA Project. The adjoint operator developed in the framework of optimum design is used to estimate the error in the solution with respect to a given target quantity. Local values of this error estimation are used as a criterion to refine the mesh. This yields significant improvement over traditional criteria based on the residual or on gradients of physical quantities. The method is carefully tested using inviscid, transonic, laminar, and high Reynolds number turbulent flows.

1 Stabilized Finite Element Schemes for the RANS Equations

Dassault Aviation's Navier-Stokes code, called `AETHER`, uses a finite element approach, based on a symmetric form of the equations written in terms of entropy variables. The advantages of this change of variables are numerous: in addition to the strong mathematical and numerical coherence they provide (dimensionally correct dot product, symmetric operators with positivity properties, efficient preconditioning), entropy variables yield further improvements over the usual conservation variables, in particular in the context of chemically reacting flows (see [1, 2]).

The code can handle the unstructured mixture of numerous types of elements (triangles and quadrilaterals in 2-D; tetrahedra, bricks, and prisms in 3-D). In practice mostly linear triangular and tetrahedron meshes are used.

The code has been successfully ported on many computer architectures. It is fully vectorized and parallelized for shared or distributed memory machines using the MPI message passing library (IBM SP2 Series, IBM BlueGene, Itanium II- and Xeon-based Bull NovaScale) or native parallelization directives (NEC SX-4) (see [3]).

Frédéric Chalot
Dassault Aviation, 78, quai Marcel Dassault, CEDEX 300,
92252 Saint-Cloud CEDEX, France
e-mail: `frederic.chalot@dassault-aviation.com`

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 369–385, 2010.
springerlink.com

For details about the numerical method, the reader is referred to Chapter 11. We just recall the semi-discrete Galerkin/least-squares variational problem which can be stated as:

Find $\quad^h \in \mathscr{S}^h$ (trial function space), such that for all $\quad^h \in \mathscr{V}^h$ (weighting function space), the following equation holds:

$$\begin{aligned}\int_\Omega \Big(W^h \cdot \quad_{,t}(\quad^h) - \quad_{,i}^h \cdot \quad_i^{\mathrm{adv}}(\quad^h) + \quad_{,i}^h \cdot \widetilde{\quad}_{ij} \quad_{,j}^h \Big)\, d\Omega \\ + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \Big(\mathscr{L} \quad^h \Big) \cdot \Big(\mathscr{L} \quad^h \Big)\, d\Omega \\ + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \nu^h g^{ij} \quad_{,i}^h \cdot \widetilde{\quad}_0 \quad_{,j}^h \, d\Omega \\ = \int_\Gamma \quad^h \cdot \Big(- \quad_i^{\mathrm{adv}}(\quad^h) + \quad_i^{\mathrm{diff}}(\quad^h) \Big) n_i \, d\Gamma. \qquad (1)\end{aligned}$$

2 Adjoint-Based Indicators for Stabilized Finite Element Methods

In the following sections we describe how the adjoint operator developed in the framework of optimum design can be used to estimate the error in the solution with respect to a given target quantity. Local values of this error estimation are used as a criterion to refine the mesh. This yields significant improvement over traditional criteria based on the residual or on gradients of physical quantities.

2.1 Adjoint-Based Extrapolation

We assume that we have a solution of the Navier-Stokes equations V^H solution of (1): $R^H(V^H) = 0$ on a given mesh characterized by a mesh size parameter H, where R^H is the discrete residual associated with (1). We can compute some aerodynamic function f of the solution (for instance drag or lift): $f^H(V^H)$. Unfortunately, mesh M^H is too coarse to compute f^H accurately. A finer mesh h whose characteristic mesh size parameter h is smaller than H would yield a better estimate of f, viz., $f^h(V^h)$. The question is "how can we estimate f^h without explicitly computing V^h, which would require the solution of $R^h(V^h) = 0$?" We will seek an estimate of f^h in the form

$$f^h(V^h) \approx f^h(V_H^h) + \cdots$$

where V_H^h is the projection of V^H onto mesh M^h.

The first order Taylor expansion of $f^h(V^h)$ about V_H^h reads:

$$f^h(V^h) = f^h(V_H^h) + \left.\frac{\partial f^h}{\partial V^h}\right|_{V_H^h} (V^h - V_H^h) + O\left((V^h - V_H^h)^2\right) \qquad (2)$$

We can expand $R^h(V^h)$ in the same fashion

$$R^h(V^h) = R^h(V_H^h) + \left.\frac{\partial R^h}{\partial V^h}\right|_{V_H^h} (V^h - V_H^h) + O\left((V^h - V_H^h)^2\right) \tag{3}$$

Since V^h is the solution of (1) on M^h, we have

$$R^h(V^h) = 0 \tag{4}$$

Combining (2), (3), and (4), it results

$$f^h(V^h) \approx f^h(V_H^h) - \left.\frac{\partial f^h}{\partial V^h}\right|_{V_H^h} \left.\frac{\partial R^h}{\partial V^h}\right|_{V_H^h}^{-1} R^h(V_H^h) \tag{5}$$

Following the ideas of Giles [5], we introduce the adjoint problem

$$\left[\frac{\partial R^h}{\partial V}(V^h)\right]^T \Psi^h = \left[\frac{\partial f^h}{\partial V}(V^h)\right]^T \tag{6}$$

The adjoint of R^h with respect to the entropy variables V^h was obtained from the original FORTRAN code by automatic differentiation using TAPENADE [4] in reverse mode.

Eq. (5) can be rewritten as

$$f^h(V^h) \approx f^h(V_H^h) - \Psi^h \cdot R^h(V_H^h)$$

In practice, we do not solve the adjoint problem on mesh M^h. Instead, we replace Ψ^h with Ψ_H^h, the projection onto mesh M^h of Ψ^H, solution of the adjoint problem (6) on mesh M^H. This adjoint problem is solved with a preconditioned GMRES algorithm.

Finally we can write the approximate of $f^h(V^h)$ as

$$f^h(V^h) \approx f^h(V_H^h) - \Psi_H^h \cdot R^h(V_H^h) \tag{7}$$

which can be computed cheaply with the projected solution of the adjoint problem on the coarse mesh Ψ_H^h and a mere residual evaluation on the fine mesh using the projection of the coarse mesh solution V_H^h.

2.2 Error Estimation and Refinement Criterion

Similar ideas can be used to place error bounds on the estimated value of the aerodynamic function $f^h(V_H^h)$ with respect to $f^h(V^h)$:

$$f^h(V^h) - f^h(V_H^h) \approx -\Psi_H^h \cdot R^h(V_H^h) + (\Psi_H^h - \Psi^h) \cdot R^h(V_H^h) \tag{8}$$

Let T_H^h denote the correction term in (7):

$$T_H^h = \Psi_H^h \cdot R^h(V_H^h)$$

Then (8) becomes

$$f^h(V^h) - f^h(V_H^h) \approx -T_H^h + (\Psi_H^h - \Psi^h) \cdot R^h(V_H^h)$$

One can show that

$$\begin{aligned} \left|(\Psi_H^h - \Psi^h) \cdot R^h(V_H^h)\right| &\to 0, \quad \text{as } h \to 0 \\ &\leq \left|\Psi_H^h \cdot R^h(V_H^h)\right| \end{aligned}$$

Consequently,

$$\left|f^h(V^h) - f^h(V_H^h)\right| \leq C \left|T_H^h\right|$$

with

$$\begin{aligned} C &\to 0, \quad \text{as } h \to 0 \\ &\leq 2 \end{aligned}$$

Finally

$$f^h(V_H^h) - C\left|T_H^h\right| \leq f^h(V^h) \leq f^h(V_H^h) + C\left|T_H^h\right| \tag{9}$$

If M^h is a very fine mesh on which asymptotic convergence is reached, (9) represents the error bound on $f^h(V_H^h)$ with respect to the exact solution of (1).

As we will see shortly, T_H^h can also be used as a goal-oriented mesh adaptation criterion, which reveals area in the mesh where locals errors have the biggest impact on the value of the target function f^h. If we go back to equation (7), we can see that if the "coarse mesh" M^H (in fact the current mesh) produces a solution V^H which is accurate enough to give a satisfactory value of the target aerodynamic function f^h,

$$f^h(V^h) \approx f^h(V_H^h)$$

and

$$T_H^h \approx 0$$

The idea consists in refining the mesh where local values of $|T_H^h|$ are greater than some specified limit ε. The refinement algorithm goes as follows:

1. on mesh M^H, solve (1) for V^H and compute the solution of the adjoint problem Ψ^H with respect to some target function f^H;
2. generate a finer finer mesh M^h, typically obtained by a uniform (iso-P2) refinement of M^H;
3. project V^H and Ψ^H onto M^h and evaluate $R^h(V_H^h)$. Remark: (1) is not solved on M^h;

4. compute a local mesh adaptation parameter P_i^h at each node of mesh M^h

$$P_i^h = (\Psi_H^h)_i \cdot R_i^h(V_H^h)$$

   where the dot product is extended to the sole number of degrees of freedom at node i;
5. project P^h onto mesh M^H;
6. if $(P_h^H)_i < \varepsilon$, refine the mesh locally and go back to step 1; otherwise the mesh is fine enough and $f^H(V^H)$ is computed with an adequate accuracy.

3 Numerical Examples of Goal-Oriented Refinement for 2-D Flows

Dassault Aviation computed the same four Mandatory Test Cases defined in Workpackage 2 of the ADIGMA Project and already presented in Chapter 11. They cover a wide range of applications: from inviscid subsonic and transonic flows (MTC's 1 and 2), to laminar Navier-Stokes (MTC 3), and finally a profile in transonic turbulent conditions (MTC 5). For each test case we compare the baseline results obtained using Dassault Aviation's industrial Navier-Stokes code AETHER on a set of uniformly refined meshes with successive goal-oriented adaptation based on the same initial coarse grid.

Local isotropic mesh enrichment is used: triangles tagged for adaptation are split into four. Nodes added on the boundary are placed on the actual surface; if needed mesh deformation techniques are used to make all elements positive. On the border of a locally refined zone, hanging nodes are connected to the opposite vertex. In order to control the aspect ratios in the adapted meshes, we allow only subdivision of original triangles into four or two. A triangle with hanging nodes on two faces will be split into four, propagating a new hanging node further away. Memory of triangles split into two is kept to avoid later division. This technique ensures the quality of the adapted grids.

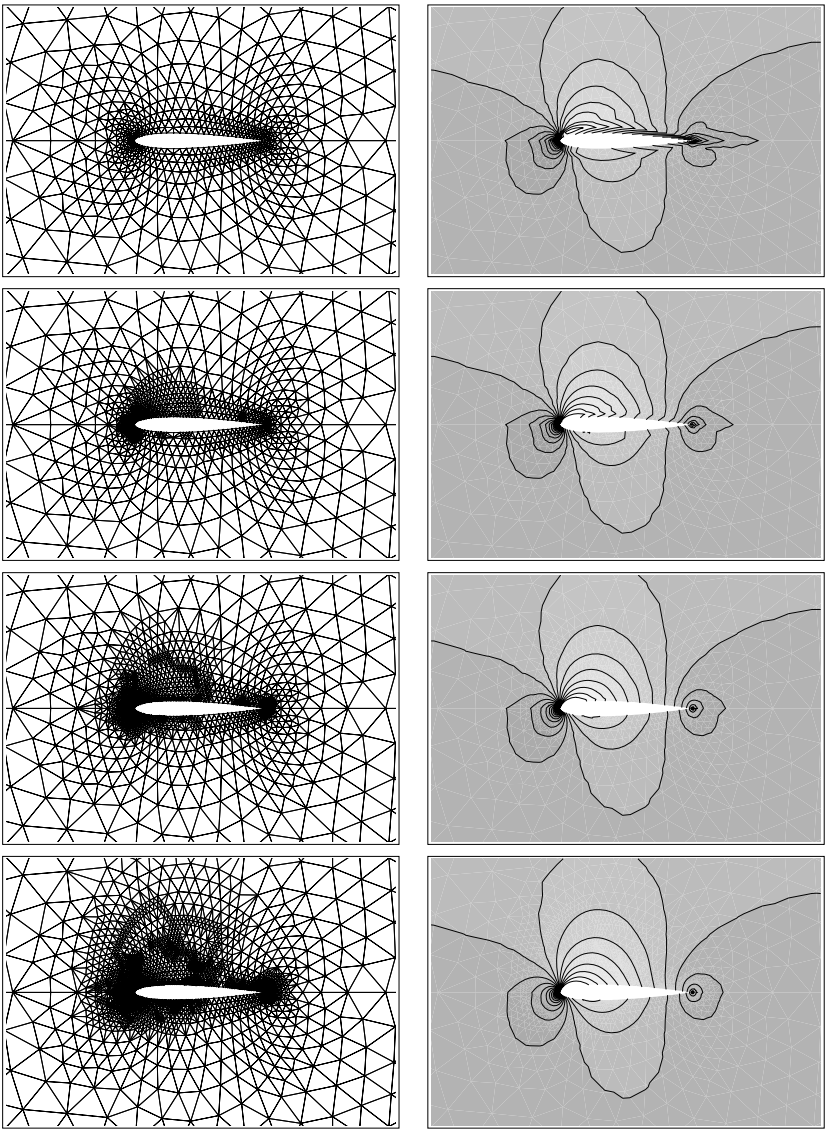
3.1 MTC 1: NACA0012, $M = 0.50$, $\alpha = 2°$, Inviscid

For this inviscid subsonic test case, the drag coefficient C_D was used as the target quantity. The initial unadapted mesh around the NACA0012 airfoil contains 1106 nodes. Figure 1 presents the original mesh and five successive levels of goal-oriented adaptive refinement together with the matching Mach-number contours.

Kinks in Mach number contours disappear after only two levels of adaptation although refinement mostly occurs in the stagnation, suction, and trailing edge regions. This is an indication that the entropy layer observed in coarse grid solutions is not due to lower order boundary conditions. Instead spurious entropy is generated at the leading edge and is convected along the profile.

Figure 2 shows the convergence of force coefficients obtained with goal-oriented mesh adaptation (mixed line) compared with those computed with global mesh refinement (solid line).

It is striking to see that C_D, which is the target, converges better than C_L as the sizes of adapted meshes increase. The gain in the required number of degrees of freedom for convergence is roughly a factor of 5. CPU gain is of the same order, with an additional advantage for adapted meshes: they have fewer points in the



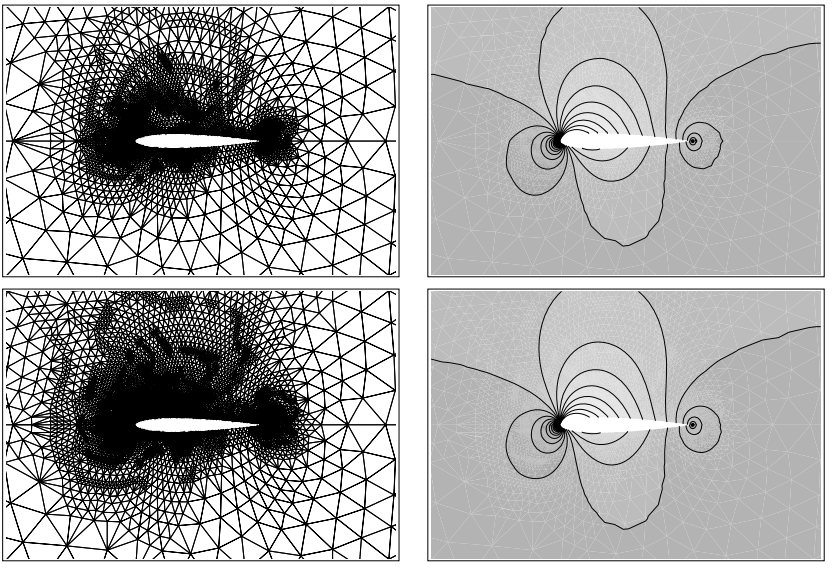


Fig. 1 MTC 1: NACA0012, $M = 0.50$, $\alpha = 2^\circ$, inviscid. Original 1106-node mesh and five successive levels of goal-oriented refinement based on drag with corresponding Mach number contours on the right.

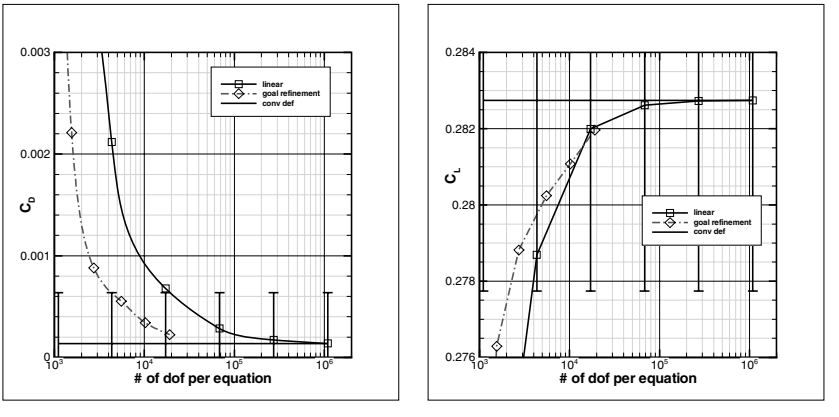


Fig. 2 MTC 1: NACA0012, $M = 0.50$, $\alpha = 2^\circ$, inviscid. Convergence of force coefficients compared with global mesh refinement.

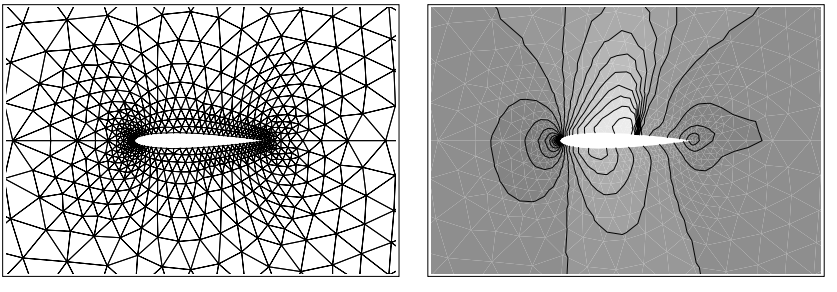
freestream, thus they require fewer time steps to reach convergence at a given CFL number.

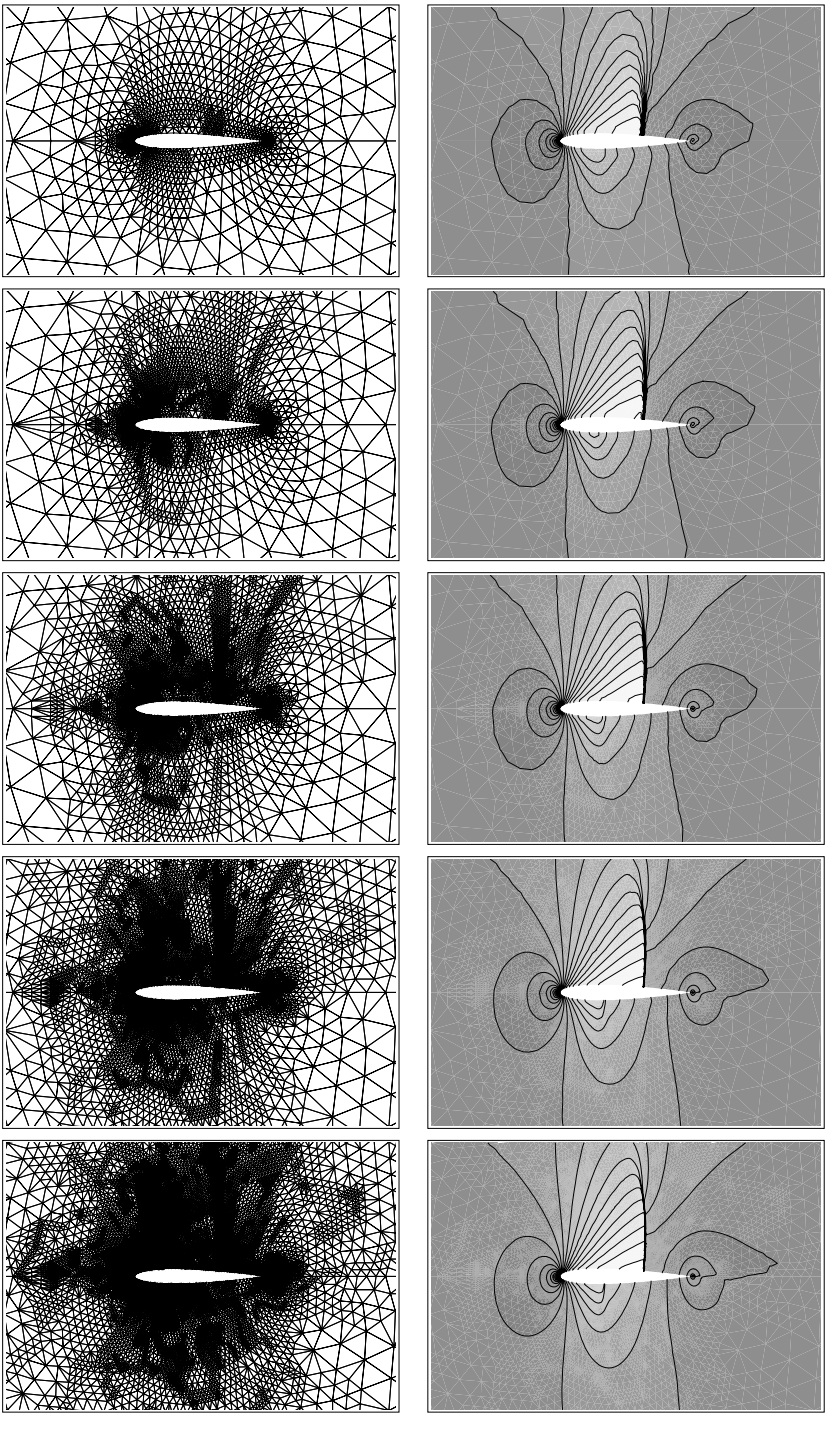
It must be noted that the threshold for refinement was fixed at 50% of the mean criterion value. This yields a series of adapted meshes which nearly double in size at each level of adaptation. This figure can certainly be reduced with a stricter refinement criterion limit.

3.2 MTC 2: NACA0012, $M = 0.80$, $\alpha = 1.25^\circ$, Inviscid

The pressure drag coefficient C_D was also used as the target quantity to produce the adapted meshes for the transonic case MTC 2 shown in Figure 3. Mach number contours corresponding to the initial grid and to the six subsequent levels of adaptation are displayed on the right. The initial mesh is the same as the one for MTC 1.

In the beginning, refinement occurs more or less uniformly in the dependency region. Only when the overall mesh size reaches a reasonable level, the criterion hits more selectively at the shocks. Both leeward and windward side shocks are accurately captured which would have been particularly challenging for a gradient-based adaptation. As in the subsonic case, points are added at the surface of the airfoil, in particular in the acceleration/expansion regions. The slip line at the trailing edge is also detected in the final meshes.





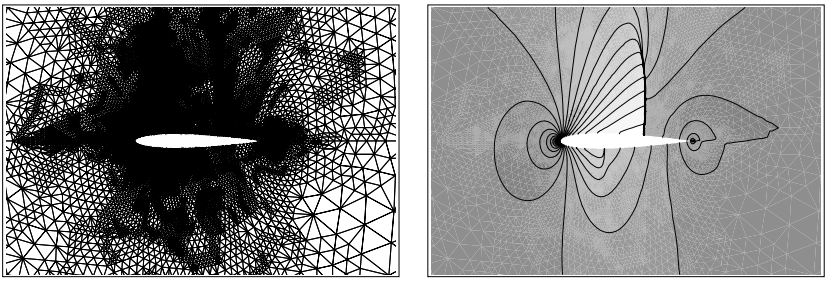


Fig. 3 MTC 2: NACA0012, $M = 0.80$, $\alpha = 1.25°$, inviscid. Original 1106-node mesh and six successive levels of goal-oriented refinement based on drag with corresponding Mach number contours on the right.

Figure 4 presents the convergence of force coefficients. The same criterion threshold as MTC 1 was used. Again it produces a series of adapted meshes which grow too fast in terms of degrees of freedom. Nonetheless the required number of nodes to converge the drag coefficient is reduced by a factor of 2.3. The behavior of lift coefficient is very disappointing. Even on the finest adapted grid (54,220 nodes), it does not meet the convergence criterion. Goal-oriented mesh refinement based on C_L should be tested.

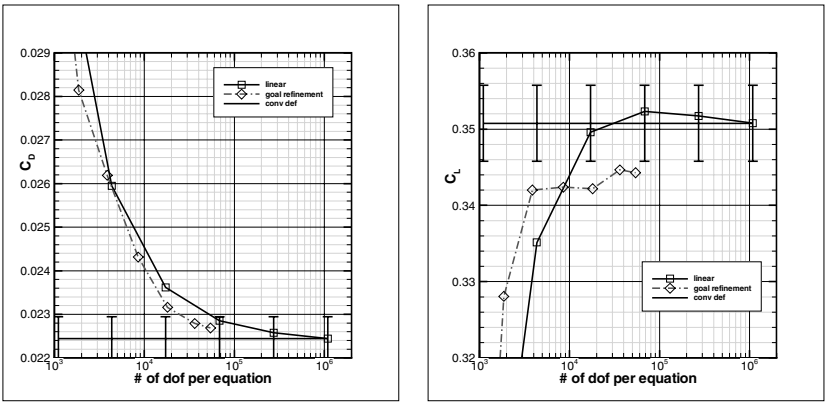


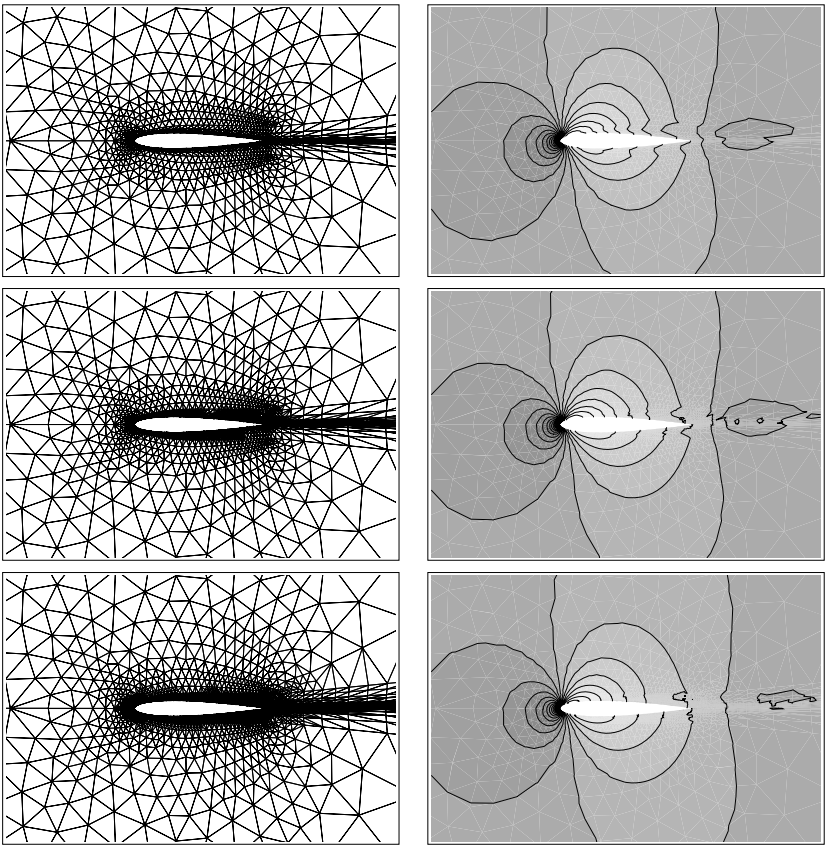
Fig. 4 MTC 2: NACA0012, $M = 0.80$, $\alpha = 1.25°$, inviscid. Convergence of force coefficients compared with global mesh refinement.

3.3 MTC 3: NACA0012, $M = 0.50$*,* $\alpha = 2^\circ$*,* $Re = 5,000$

Even though MTC 3 is a viscous test case, we have again used the pressure drag coefficient C_D as the target for goal-oriented mesh adaptation with the same criterion threshold chosen previously.

Figure 5 shows the original 1533-node mesh and four successive levels of goal-oriented refinement based on pressure drag together with the corresponding pressure contours on the right. Refinement occurs at the leading edge, along the profile, and in the wake. Mesh deformation was used at each level of adaptation to place the new nodes along the actual profile.

Only four levels of refinement could be applied. A fifth adapted mesh was generated, on which the computation revealed unsteady. This behavior was observed by other partners in the ADIGMA Project, although all our previous computations for this $Re = 5,000$ test cases always converged to a steady state.



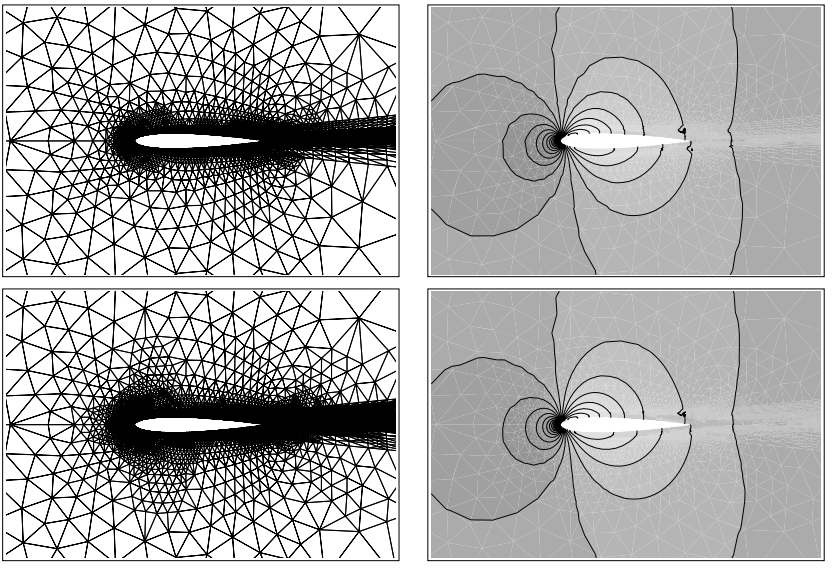
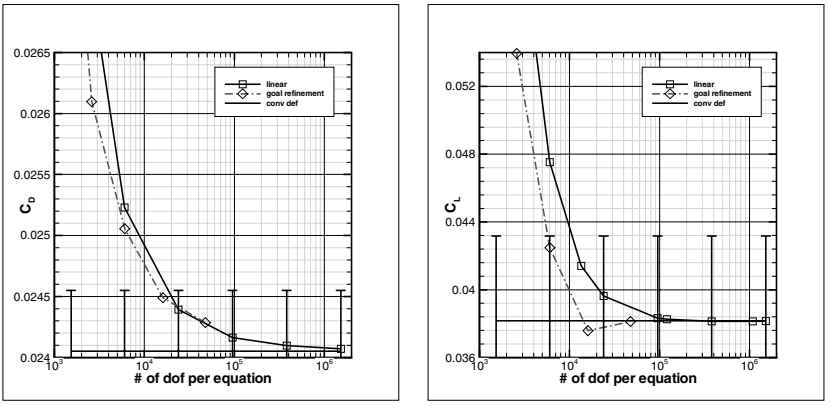


Fig. 5 MTC 3: NACA0012, $M = 0.50$, $\alpha = 2^\circ$, $Re = 5{,}000$. Original 1533-node mesh and four successive levels of goal-oriented refinement based on pressure drag with corresponding pressure contours on the right.

Figure 6 displays the convergence plots of the force and heat flux coefficients. Once more the criterion threshold seems too lenient. The mesh size increases too rapidly at each refinement step (by a factor of nearly 3). The required number of degrees of freedom to converge pressure drag is barely reduced by 10%.



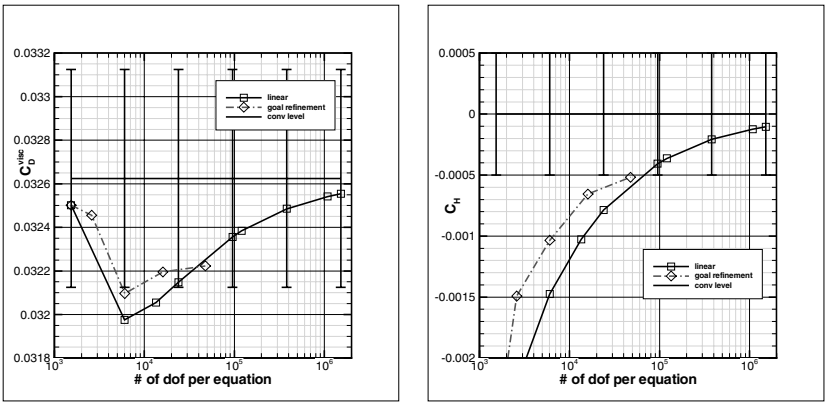


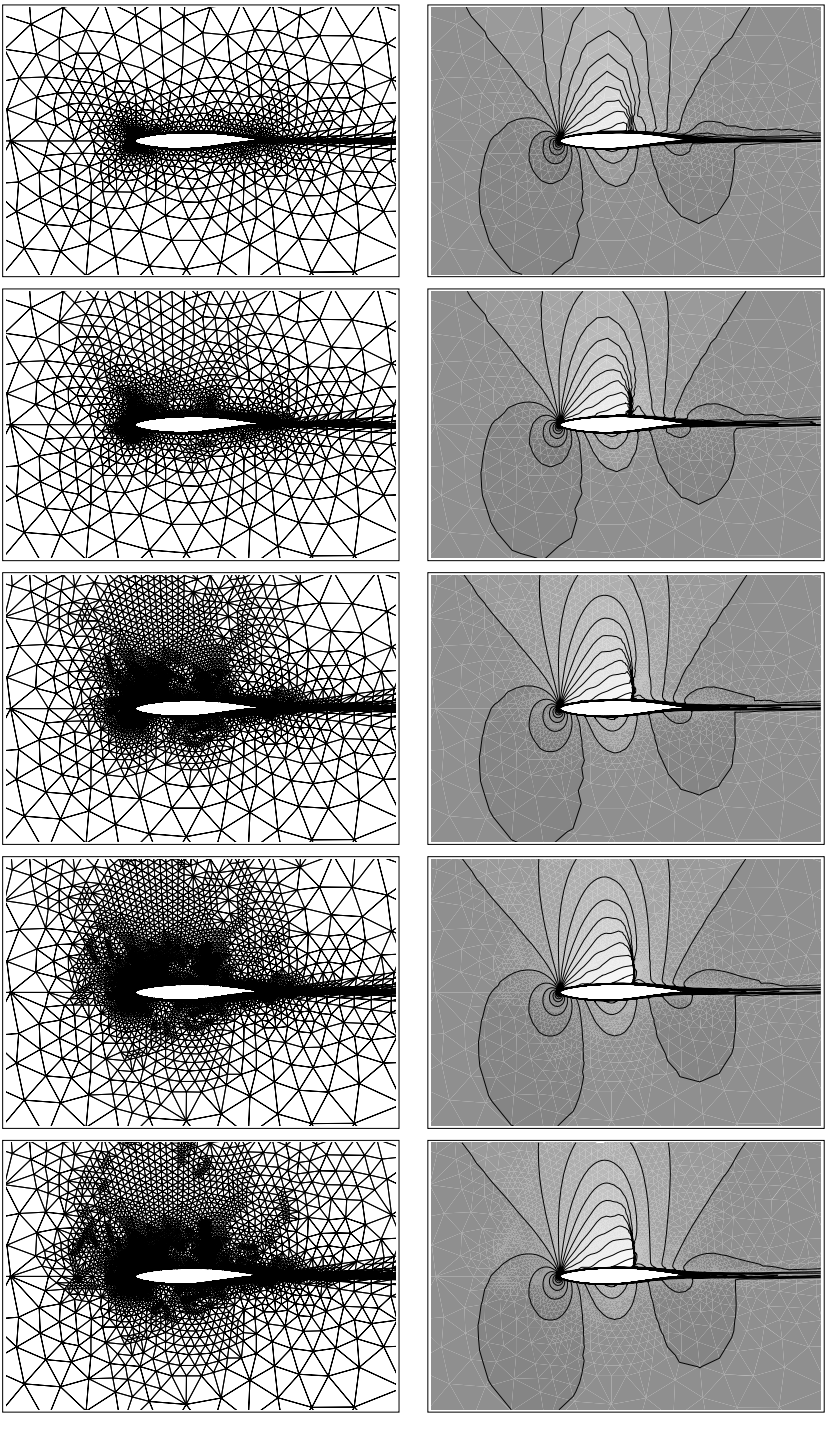
Fig. 6 MTC 3: NACA0012, $M = 0.50$, $\alpha = 2^\circ$, $Re = 5,000$. Convergence of force and heat flux coefficients compared with global mesh refinement.

Although not targeted, lift convergence requirements drop by about 40%, those of friction drag by about 20%. The convergence curves of friction drag and of heat flux tend to flatten out with the finest adapted meshes. This might be an indication of the forthcoming onset of unsteadiness.

3.4 MTC 5: RAE2822, $M = 0.734$, $\alpha = 2.79^\circ$, $Re = 6,500,000$

The final test case is a transonic high Reynolds number RANS calculation past an RAE2822 profile. We have still used the pressure drag coefficient C_D as the target for goal-oriented mesh adaptation. We have altered the criterion threshold from the previous MTC's though. It appeared that too heavy a refinement was applied at each level, which somehow reduced the potential benefit of mesh adaptation. For MTC 5 we have tried to limit the refinement to the top 20% of the elements where the criterion was the largest.

In doing so problems were encountered when refining under-resolved highly-curved boundaries near the leading edge. Too little local refinement would prevent mesh deformation from completing successfully: the locally refined region must be thick enough to allow the deformation of the thinnest elements into the volume. The refinement zone had to be extended respectively to the top 40 and 60% to permit deformation of the first two adapted meshes. For the next two refined grids, no criterion threshold value would yield no negative elements after deformation. We chose to stick to our 20% rule and to skip mesh deformation all together. The final adapted mesh was obtained with the top 20% criterion and with a successful mesh deformation. The five adapted meshes are presented in Figure 7 with the initial 2668-node mesh; Mach number contours are shown on the right next to each corresponding grid.



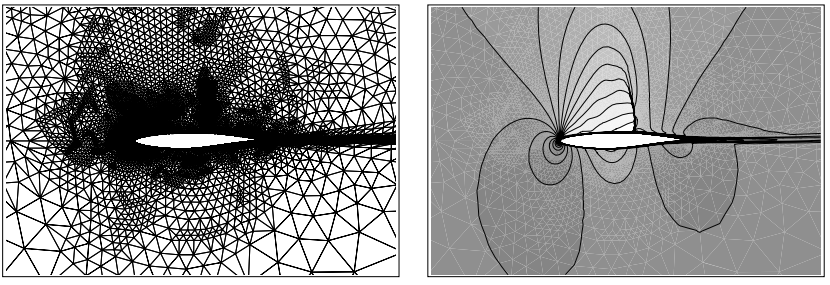


Fig. 7 MTC 5: RAE2822, $M = 0.734$, $\alpha = 2.79°$, $Re = 6,500,000$. Original 2668-node mesh and five successive levels of goal-oriented refinement based on pressure drag with corresponding Mach number contours on the right.

At first goal-oriented adaptation driven by pressure drag refined the leading edge, the suction region and the wake. Then it hit more specifically at the shock area. Although intrinsically inviscid the chosen target seems to have tackled this high-Reynolds number case rather well. Let's have a look at the convergence of force and heat flux coefficients presented in Figure 8 for a more quantitative analysis.

The number of degrees of freedom to converge pressure drag is reduced by a factor of 13 from 130,000 to about 10,000. This exemplifies the power of goal-oriented mesh adaptation when used with a controlled refinement criterion level. The requirement for lift is divided by a factor of nearly 3 (from 60,000 to about 23,000). Viscous coefficients are less successful. Friction drag converges only slightly faster than with a uniform grid refinement. Heat flux seems to reach an asymptotic value of 10^{-4} and not converge any further. Again it would be worth testing more Navier-Stokes specific target functions, such as friction drag, total drag, or heat flux.

In spite of the aforementioned mesh deformation difficulties, goal-oriented mesh adaptation, even based on a pressure target, can handle high Reynolds numbers. Feature based adaptation would probably miss some of the features of the flow.

4 Conclusion

Substantiated by various test cases, we have showed that goal-oriented adaptation coupled with local isotropic mesh refinement, works for diverse situations: inviscid, transonic, laminar, and high Reynolds number turbulent flows.

Adaptation based on local isotropic refinement requires a decent mesh to start with, that is for instance stretched elements along the wall for Navier-Stokes calculations. On that condition, it can handle high aspect ratios. One must pay attention though to underresolved curved boundary layer regions where mesh deformation to match the surface definition can be an issue.

In order to maximize the gain over uniform grid refinement, the criterion threshold needs to be tuned. Too much refinement at each adaptation level will slow down the whole process. Nevertheless there is always a slight CPU advantage for adapted

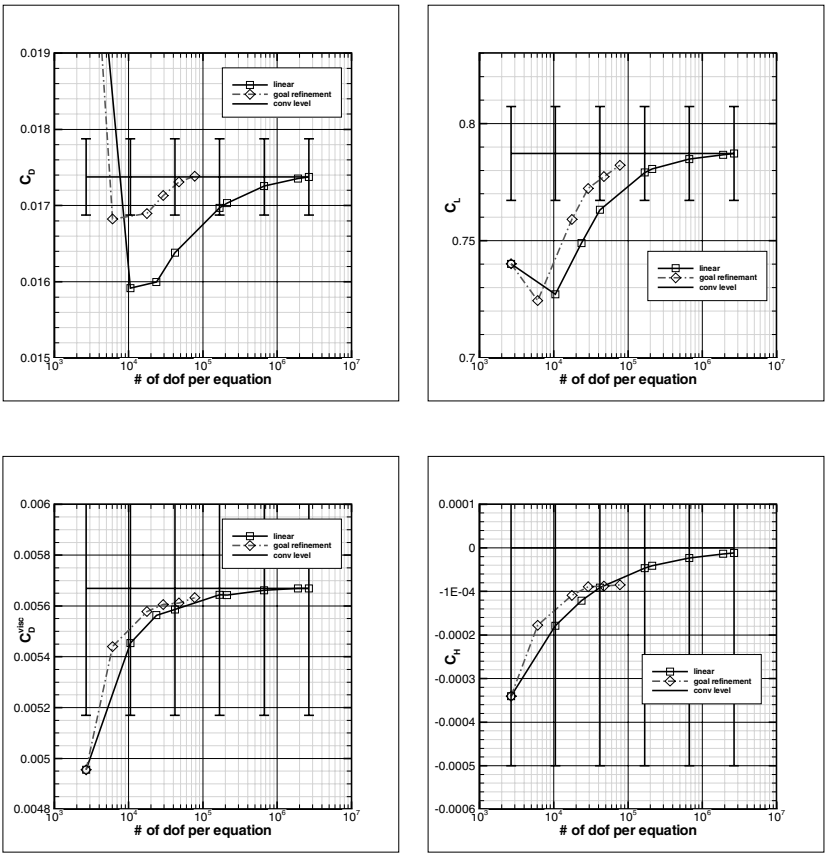


Fig. 8 MTC 5: RAE2822, $M = 0.734$, $\alpha = 2.79^\circ$, $Re = 6,500,000$. Convergence of force and heat flux coefficients compared with global mesh refinement.

meshes with an equivalent number of nodes: they have less points in the freestream and thus require fewer time steps to converge at a given CFL number.

The different force coefficients converge at different rates. This is even more true with goal-oriented adaptation. Only the targeted quantity tends to see the benefit of the refinement; the adjoint does not seem to improve the solution globally. Other cost functions should be tested, especially specific Navier-Stokes ones. Multiple targeted adaptation might be the solution. In any case, goal oriented adaptation is still more versatile than feature or residual based refinement: it works for shocks, boundary layers, and wakes, even with a simple pressure cost function.

Local isotropic refinement not viable in 3-D; the number of nodes grows too fast. Anisotropic refinement/derefinement or a remeshing capability are needed for complex 3-D applications. For industrial use, an automated procedure is needed.

Running several meshes for convergence is a turn down for the technique. Human cost must not overwhelm the CPU advantage.

The method should possibly be coupled with higher-order elements for augmented performance. In principle the adjoint obtained through automatic differentiation should work as is with higher order elements.

References

1. Chalot, F., Hughes, T.J.R.: A consistent equilibrium chemistry algorithm for hypersonic flows. Computer Methods in Applied Mechanics and Engineering 112, 25–40 (1994)
2. Chalot, F., Mallet, M., Ravachol, M.: A comprehensive finite element Navier-Stokes solver for low- and high-speed aircraft design. Paper #94-0814. AIAA 32nd Aerospace Sciences Meeting, Reno, NV, January 10-13 (1994)
3. Chalot, F., Dinh, Q.V., Mallet, M., Naïm, A., Ravachol, M.: A multi-platform shared- or distributed-memory Navier-Stokes code. In: Parallel CFD 1997, Manchester, UK, May 19-21 (1997)
4. Hascoët, L., Pascual, V.: TAPENADE 2.1 user's guide. Rapport Technique N° 300. INRIA (September 2004)
5. Pierce, N., Giles, M.: Adjoint and defect error bounding and correction for functional estimates. Journal of Computational Physics 200, 769–794 (2004)

This page intentionally left blank

Chapter 27
Application of Feature-Based Grid Adaptation to Helicopter Rotor Flow

H. van der Ven

1 Introduction

The simulation of rotorcraft aerodynamics is considerably more complex than the simulation of fixed wing aircraft aerodynamics. Rotorcraft flow is inherently dynamic, the inertial and elastic forces of the rotor blades interact with the aerodynamic forces, and aerodynamic interference of the rotor wake with the fuselage and tail rotor is important in many flight conditions. The flow condition known as Blade-Vortex Interaction (BVI) is an important example of such interactions. Especially in low-speed descent, the rotor blades fly in their own wake. The interaction of tip vortices and rotor blades may cause strong pressure fluctuations on the blade, responsible for the typical 'wopwop' sound of helicopters. Prediction of BVI is challenging: the blade motion under inertial, elastic, and aerodynamic forces must be predicted correctly and the convection of the tip vortices must be accurate enough to retain the vortices for, typically, one and a half rotor revolution.

The requirement to correctly represent the blade motion has led to the development of time-accurate flow solvers which are coupled with dynamics solvers or rotor comprehensive codes (Buchtala et al. [3], Pomin et al. [11], Altmikus et al. [1]; the reader is also referred to the excellent review paper on rotorcraft CFD by Datta et al. [5]). Several efforts have been undertaken to improve the vortex capturing capability of standard flow solvers, such as local grid refinement (Bottasso et al. [2]), Chimera techniques with specific vortex grid systems (Ochi et al. [10], Duraisamy et al. [7]), and high order methods (Wake et al. [16]). None of these techniques has been particularly successful or efficient, and successful BVI predictions have only been obtained by brute-force methods, using meshes of 100 million cells (Lim et al. [9]) and time-marching several rotor revolutions before structural dynamics, trim, and aerodynamics have balanced out. Lim et al. use a series of grids of which the fine grid contains 100 million elements and apply a time step of 0.05 azimuthal

H. van der Ven
National Aerospace Laboratory, A. Fokkerweg 2, Amsterdam, The Netherlands
e-mail: venvd@nlr.nl

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 387–398, 2010.
springerlink.com

degrees. Although the simulations exhibit BVI, the authors are not satisfied with the vortex resolution of the simulation as compared with experiment. They estimate a mesh containing 7 billion elements is necessary for the simulations to agree with experiment. This number is remarkably close to the estimates given by Caradonna [4] in his review article. Clearly, such simulations cannot be run in a routine way and despite the qualitative success so far, there is a need for more efficient algorithms.

In this paper, a grid adaptation strategy is described, which is specifically designed to generate efficient meshes for the simulation of rotor wakes. Combining local grid refinement with the four-dimensional solution algorithm of [14], allows meshes which have the required uniform resolution in space and time, only where and when a vortex is present. Such meshes may contain 20-50 times less degrees of freedom than meshes with uniform resolution.

2 Numerical Algorithm

The discretization of the Euler equations is based on a space-time discontinuous Galerkin method (Van der Vegt et al. [13]). Up to third order of accuracy can be obtained. Local grid refinement is applied for the generation of meshes which are uniform in the vortex core. A specific solution algorithm, called Multi-Time Multi-Grid, is used for the solution of time-periodic problems [14]. The solution method solves a time-periodic problem on a four-dimensional space-time mesh containing a discretization of the time period. As the mesh contains all time levels, the multigrid solution algorithim can be extended to the time dimension.

Apart from generating a periodic solution by construction, the main advantage of the solution algorithm lies in the fact that it transforms a time-dependent (*dynamic*) problem into a steady-state (*static*) problem. This has several advantages:

- as long as the solution process converges, the final solution is independent of the solution process. The underlying discretised equations are unmodified, and, moreover, there is no problem with the possible accumulation of numerical errors from preceding time steps;
- local grid refinement can be extended to the time dimension. Since there is no time direction in the solution algorithm, interpolation or the order of time steps in the case of hanging nodes are no issue;
- combining local grid refinement and parallel processing does not lead to dynamic load balancing problems. Since the local grid refinement no longer needs to be applied at each time step, the number of grid adaptations reduces significantly to about five times per simulation, which is the usual number for steady-state simulations.
- time-accurate coupling with other physics models is straightforward. As all simulation all simulation data is available at all time steps, and the trust and moments are readily available to trim the rotor. Moreover, a modification in the pitch schedule will require only a small number of pseudo-time iterations for the flow solution to conform to the new schedule. Another consequence of having the data available at all time steps is that the coupling between the aerodynamics and mechanics

modules can be made genuinely implicit, without the need of predictor-corrector mechanisms.

3 Grid Design and Expected Speedups

Combination of the adaptivity of the DG algorithm and four-dimensional nature of the MTMG algorithm allows the design of grids in which the required resolution in the vortex core is obtained only when and where a vortex is present. The accurate convection of the tip vortices requires a uniform mesh in both space and time within the vortex core. Let n be the number of cells which are required across the vortex core. For the third order DG scheme $n = 7$, and for the second order scheme $n = 15$, provide good estimates. With a core size of $R/50$ (Caradonna [4]), with R the rotor radius, the mesh width in the vortex core is $h = R/50n$. The time step is based on a physical, convective, CFL number of one, where the CFL number is based on the freestream velocity u_∞. Given the mesh width above, one easily computes that the time step shall satisfy $\Delta t = T/100n\pi\mu$, where T is the blade revolution period and μ is the advance ratio $\mu = \Omega R/u_\infty$ with $\Omega = 2\pi/T$ the angular speed of the rotor.

In order to allow an accurate representation of the induced velocity field of the vortex, the region near the vortex core has to satisfy certain resolution requirements as well. The approach followed here is that in a region at a distance of $R/25$ of the vortex core the resolution should be at least twice the resolution in the vortex core. This is repeated once more, in a region at a distance of $R/12.5$ the resolution should be at least four times the resolution in the vortex core. The attained resolution in the latter region is thus about $R/12n$. Note that the time step grows proportionally with the mesh width.

The total number of grid cells in these regions is determined by the length of the tip vortex. A good estimate of the convection velocity of the vortex is the freestream velocity u_∞. The average distance a vortex must travel to move outside the rotor disk area is R, and hence the average time a vortex remains in the disk area is R/u_∞. Since the vortices are created at the blade tips which travels at a velocity of ΩR, the average length of the vortices within the rotor disk area is $(\Omega R)(R/u_\infty) = R/\mu$.

So the required number of cells N_{core} within the vortex core is equal to (V_{core} is the space-time volume of the core area)

$$\begin{aligned} N_{\text{core}} &= V_{\text{core}}/(h^3\Delta t) = (R/\mu)(\pi(R/50)^2)T/((h^3\Delta t) \\ &= \frac{\frac{R}{\mu}\pi\left(\frac{R}{50}\right)^2 T}{\left(\frac{R}{50n}\right)^3\left(\frac{T}{100n\pi\mu}\right)} = 2\pi^2 50^2 n^4 \end{aligned}$$

So, for the third order scheme with $n = 7$ the number of cells in the four-dimensional mesh in the vortex core alone is 118 million. For the second order method with $n = 15$ it is 2.5 billion.

It is left to the reader to compute the number of cells in the two outer regions about the vortex. However, as both mesh width and time step are doubled, the

number of cells is considerably less: the first region contains about 22 million cells for the third order scheme and the second region about 6 million cells.

The resolution outside these regions is much coarser. A generic spatial mesh about a four-bladed rotor with geometry refinement contains about 200,000 cells. In order to capture the blade-vortex interaction the time resolution of the geometry must be the same as the time resolution in the vortex core. So the space-time mesh outside the vortex regions but including the geometry contains $200,000(100n\pi\mu)/n_b$ cells, where n_b is the number of blades (the time period is T/n_b). For $n = 7$ and realistic advance ratio's for a four-bladed rotor this number is roughly between 20 and 40 million cells.

Although these numbers may seem staggering, they should be compared with the resolution required for a uniform mesh. The numbers are tabulated in Table 1. The mesh size of the uniform spatial mesh size is obtained by a uniform mesh with resolution $R/50n$ in a computational domain of volume $R^3 = (\pi R^2)(R/\pi)$, the rotor disk with height R/π. Depending on the advance ratio, the four-dimensional meshes contain 20-50 times less cells than a uniform mesh contains. This is under the assumption that it is possible to generate such highly irregular meshes. This is the subject of the next chapter.

Table 1 Estimated grid sizes and efficiency gains for a four-bladed rotor in forward flight with different advance ratio's. Mesh sizes in millions.

Number of cells across vortex core	7			15		
Advance ratio	0.15	0.25	0.35	0.15	0.25	0.35
Mesh width (fraction of blade span)	0.0028	0.0028	0.0028	0.0013	0.0013	0.0013
Time steps (in degrees azimuth)	1.1	0.66	0.46	0.51	0.31	0.22
MTMG mesh (in millions)	183	172	162	3130	3150	3170
Average spatial mesh size	2.0	1.3	0.93	18	11	7.7
Uniform spatial mesh size	43	43	43	422	422	422
Efficiency gain	22	34	46	24	40	55

4 Adaptation Strategies

4.1 Introduction

For industrial applications the most common grid adaptation strategy is to use sensors based on flow features. Shock sensors based on gradients in the flow have in general been quite effective in resolving shocks. For tip vortices in the rotor wake the effectiveness of feature-based sensors is not so clear. The straightforward vorticity magnitude sensor does not discriminate between tip vortices, vortex sheets or numerically induced boundary layers. More advanced sensors, such as the λ_2 criterion (Jeong et al. [8]), require significant resolution of the vortex to detect it. Such a resolution is not present on the initial coarse meshes. Because of these findings, it was concluded to opt for pre-adaptation, where the mesh is refined before the actual

flow computation, based on the expected location of the tip vortices. This will be explained in detail later. In effect, the pre-adaptation strategy resembles unstructured grid generation more than conventional mesh refinement. It is comparable to the Chimera approach of Dietz et al. [6], where Chimera grids are constructed around the expected tip vortex locations. The benefit of our approach is that it is still based on the single-grid concept.

4.2 Adaptation on Streak Lines

By their nature, the location of the tip vortices is predominantly determined by the trajectory of the tip. So, to a very good approximation, the vortex trajectories can be taken to be the blade tip trajectories. The blade tip trajectories $\Gamma(t)$ at time t are cycloids, described by

$$\Gamma(t) = \{(R\cos(\Omega(t-s)), R\sin(\Omega(t-s)), w(t-s)) + u_\infty s\},$$

where $w = w(t)$ is the flapping motion of the tip.

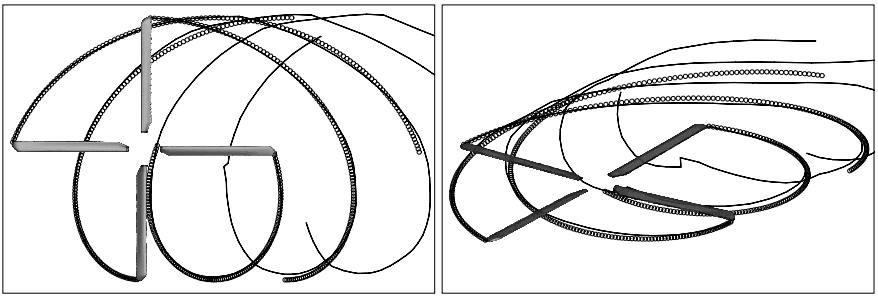
Based on this geometrical information, the mesh may be pre-adapted to increase the resolution near the tip trajectories. Whenever a cell is within a given distance of a tip trajectory and the mesh width or the time step is greater than a given threshold the cell is refined. Eventually a mesh is 'generated' with uniform space-time resolution in the expected vortex regions. The main benefit of this method is that there will be no refinement in other regions, for instance in the vortex sheets, hence the total number of grid cells is reduced compared to feature-based adaptation.

Of course, the assumption that the tip vortices follow the tip trajectories ignores the effects of downwash, contraction and interaction. Hence the free stream velocity in the definition of the tip trajectories $\Gamma(t)$ is replaced by the actual, computed velocity field. That is, the blade tip trajectories are replaced by the streak lines of particles released at the blade tips.

In this way, an iterative pre-adaptive procedure is constructed, where the accuracy of the predicted tip vortex locations is increased with each iteration. The adaptation strategy has been applied to the simulation of a four-bladed rotor in forward flight. Details of the flow will be described in Section 5, but the adaptation strategy will be illustrated in this section for this simulation.

In Figure 1, a comparison is made between the blade tip trajectories and the streak lines of particles released at the blade tips. The difference increases with the age of the vortex. The main difference between the two methods is that the second accounts for the downwash of the rotor.

Let Mesh G_0 be the initial mesh, which has local refinement near the blades in both space and time but without local refinement for the vortices. Typically, the resolution near the blades is four times finer than in the surrounding areas. Let G_1 be the mesh the mesh obtained by pre-adaptation of G_0 on the blade tip trajectories. On Mesh G_1 a high-order flow solution is computed, and from this flow solution streak lines are computed which are used for the next pre-adaptation. This next pre-adaptation is again executed on Mesh G_0, that is, the previous pre-adapted mesh is



(a) Perpendicular view (b) Perspective view

Fig. 1 Difference between the blade tip trajectories (dots) and streak lines (coloured lines) for a four-bladed rotor in forward flight

discarded. The mesh G_2 is generated in several adaptation steps, where each step zooms in on the expected vortex location by reducing both the target mesh width and the region in which the grid is refined (consistent with the grid design described in Section 3). First the cells at a distance of $0.1R$ are refined with the target mesh width $0.02R$. Both distance and mesh width are decreased proportionally, and the final step in the adaptation process has a target mesh width of $0.007R$ at a distance of $0.035R$. In the simulation described in Section 5, the adaptation on streak lines is repeated once, and the final mesh G_3 has 17.3 million elements. Due to memory restrictions on NLR's compute server, the resolution in the vortex core is restricted to three cells across the core. The simulation with a third-order DG scheme requires 60GB of memory.

The 17.3 million elements of Mesh G_3 are equivalent to 260 million degrees of freedom per equation. Since the space-time mesh contains 64 time slabs, the average number of degrees of freedom per time slab is 4 million. This clearly is a modest number when trying to resolve the rotor wake.

It should be remarked that the target mesh width is not necessarily attained. A cell is refined whenever the mesh width is more than two times the target mesh width. Hence, on the final mesh, mesh widths will be between once and twice the target mesh width. Moreover, out of practical considerations, the mesh adaptation is stopped before saturation of the refinement criterion, and the quality of the mesh is judged by inspection.

Figure 2 show Mesh G_3 at a horizontal cross section at a certain time level. The expected tip vortex locations are clearly visible as local refinement areas. Figure 4 illustrates the pre-adaptation algorithm in a grid plane through the rotor axis and approximately perpendicular to the vortices. This figure clearly demonstrates the adaptation strategy with its zooming characteristics.

It is clear from these figures that the mesh has been refined in a uniform way near the predicted vortex locations, and nowhere else. Hence the pre-adaptation algorithm is very effective in limiting the number of refined elements.

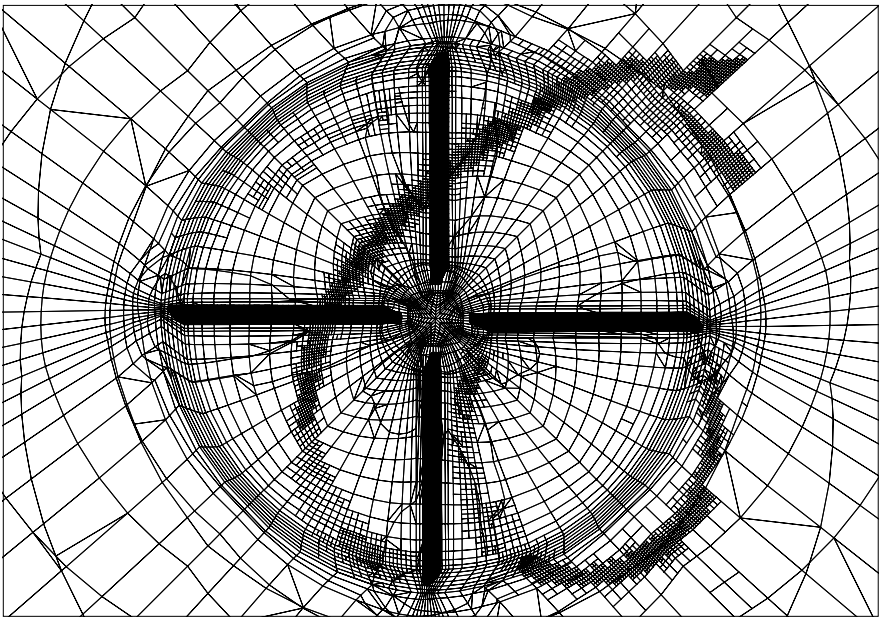


Fig. 2 Illustration of the pre-adaptation strategy. An arbitrary horizontal slice of Mesh G_3 is shown. Clearly visible are the refinement regions near the expected vortex locations.

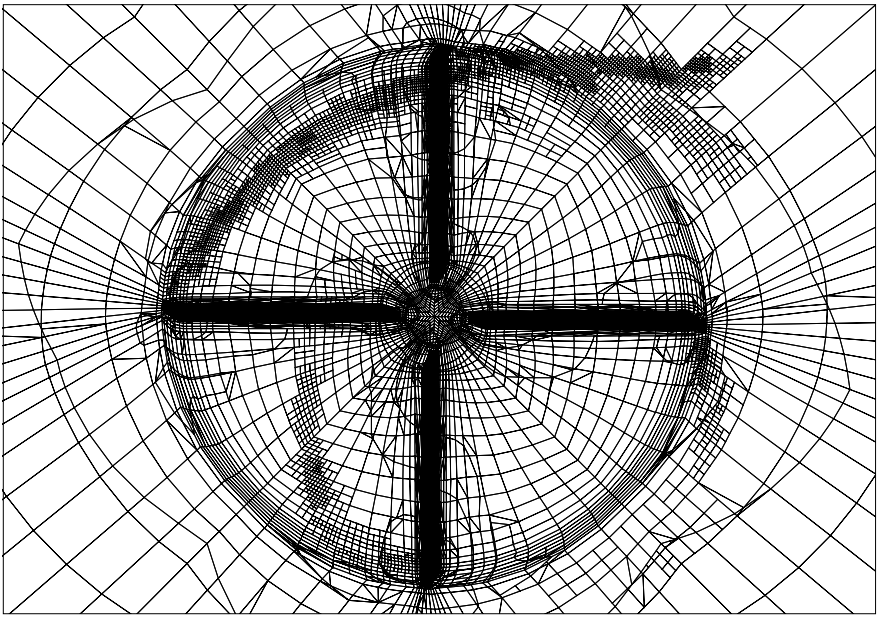


Fig. 3 As the previous figure, but a different horizontal plane, below the previous one

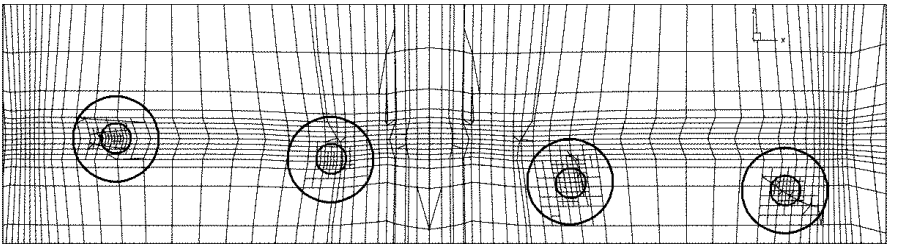


Fig. 4 Impression of the mesh obtained from pre-adaptation on streak lines. The grid plane shown is through the rotor axis (in the center of the figure) and almost perpendicular to the vortices. The larger circles have a diameter of $0.1R$ (within the circles the target mesh width is $0.02R$); the smaller circles have a diameter of $0.035R$ (within the circles the target mesh width is $0.007R$).

5 Results

Data point 135 of the Helishape wind tunnel program is simulated [12]. This data point concerns a high-speed level flight case at an advance ratio of 0.356 for the isolated 7AD1 rotor with parabolic and anhedral tip. The blade has a radius R of 2.1m and a chord c of 0.14m (aspect ratio $R/c = 15$). In the experiment the rotor is trimmed for a thrust coefficient of $C_T = \frac{F_z}{\pi\rho_\infty\Omega^2R^4} = 0.0071$ and zero flapping.

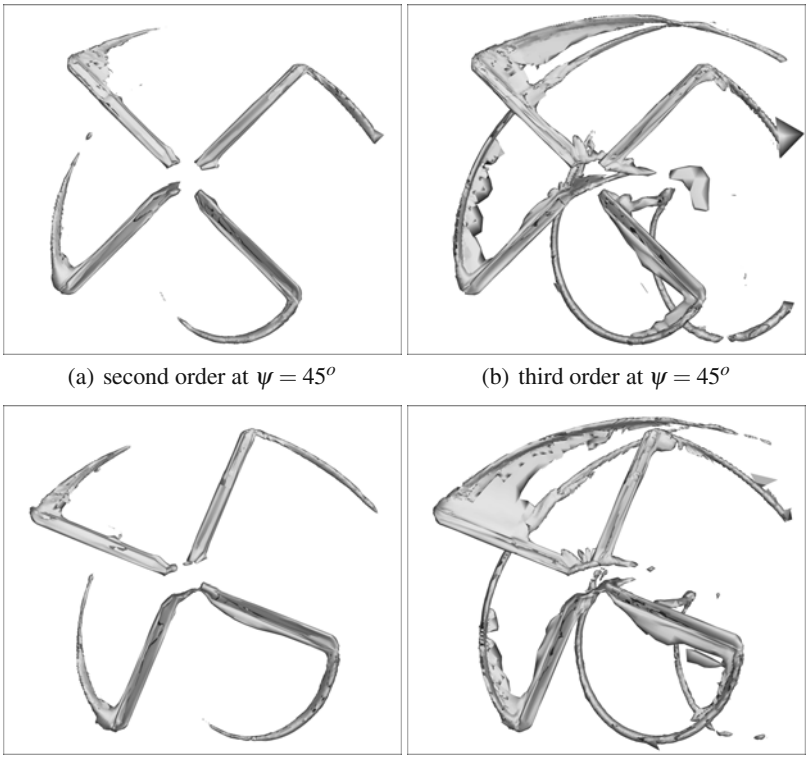
The flow displays (weak) shocks at the advancing side and the vortex wake contributes to the vibratory loads at the rotor hub. Hence a CFD mesh must be generated capable of resolving shocks and vortices.

The grid design for this simulation has been described in Section 4.2. In the following the vortex system as computed by the third order DG method on Mesh G_3 is described in detail. Other flow features, such as pressure distributions and forces, including validation, are described in [15].

Figure 5 shows the vortex systems at an azimuth angle of 45 and 67.5 degrees, for second and third order solutions. Clearly, the second order simulations show little evidence of vortices at the shown vorticity levels: the vortices are present but weak. The third order solutions show much improvement over the second order solutions.

In the following figures the vortex system is discussed in more detail. Each set of figures shows the vortex system from above (for example, Figure 6(b)), and in a cross-sectional vertical slice (for instance, Figure 7 and Figure 8). The figure with the vortex system viewed from above also presents the location of the cross-sectional plane, including numbered labels at locations where the vortices intersect the plane. The numbers are repeated in figures of the cross-sectional slice, with an additional letter, signifying whether the vortex is a tip vortex ('T') of root vortex ('R').

In Figure 7, tip vortices of four ages are visible in the third order solution: T1, T2, T4, and T5. Note that these vortices are barely visible in the second order solution. Tip vortex T6 is the tip vortex with about the same age of vortex T1, and hence is much clearer. The root vortices R3 and R4 are located near the tip vortices T2 and



(a) second order at $\psi = 45^o$ (b) third order at $\psi = 45^o$

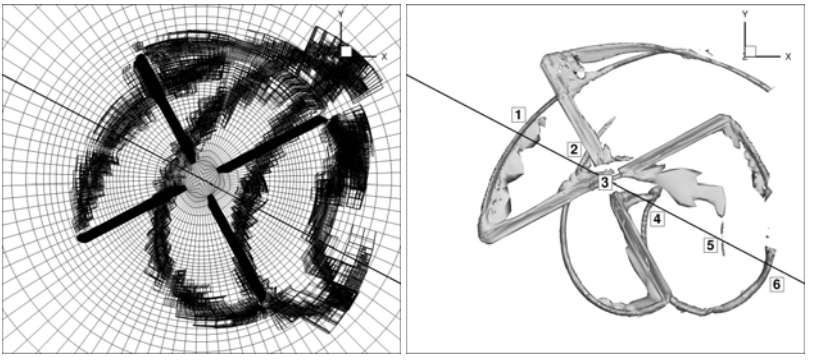
(c) second order at $\psi = 67.5^o$ (d) third order at $\psi = 67.5^o$

Fig. 5 Comparison of the vorticity contours for two orders of accuracy. Iso-contours are shown at vorticity magnitude level of $2a_\infty/R$ and $5a_\infty/R$, where a_∞ is the speed of sound.

T4, and since they are of opposite rotation they are likely to destroy the tip vortices. In reality the wake of the hub will have weaker interactional effects than the present root vortices. Comparing the vortex locations and grid resolution in Figure 8, shows that the grid has been refined in the correct locations.

Figure 10 shows the vortex system at a later stage and more on the advancing side of the rotor. The two tip vortices T1 and T2 are clearly visible in the third order simulation results, but the next tip vortex T3 is very weak. Note the different scale in the vorticity levels: the tip vortices emanating from the blades in the second quadrant are weaker than the vortices emanating from the blades in the third quadrant (compare Figure 7). As the blade loading is not significantly different for the two quadrants, this is most probably due by lack of resolution (see Figure 11).

For all the cross-sectional planes discussed above, the adaptation algorithm accurately refines the cells at the actual vortex locations. In general one can conclude that the pre-adaptation on streak lines is an efficient and accurate way of generating meshes with sufficient resolution in the vortex regions.



(a) grid (b) vortex system

Fig. 6 Definition of the cross-sectional slice, shown in red, at $\psi = 28^o$

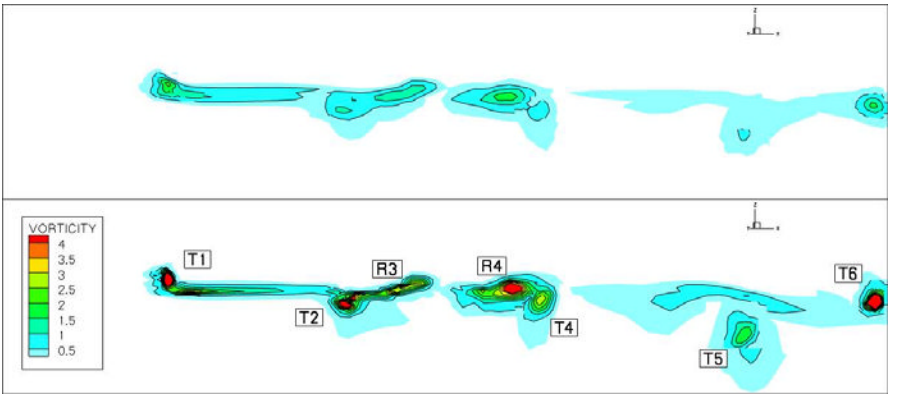


Fig. 7 Comparison of vortex resolution in the slice defined in Figure 6 with second order simulation (top) and third order simulation (bottom)

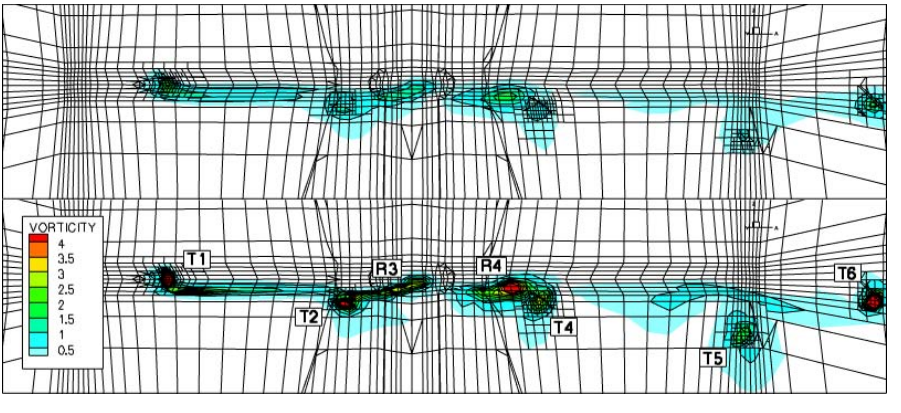
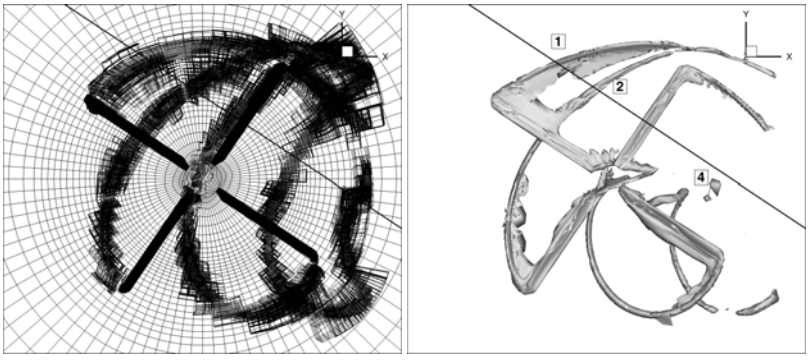


Fig. 8 Comparison of vortex resolution in the slice defined in Figure 6 with second order simulation (top) and third order simulation (bottom)



(a) grid (b) vortex system

Fig. 9 Definition of the cross-sectional slice, shown in red, at $\psi = 56^o$

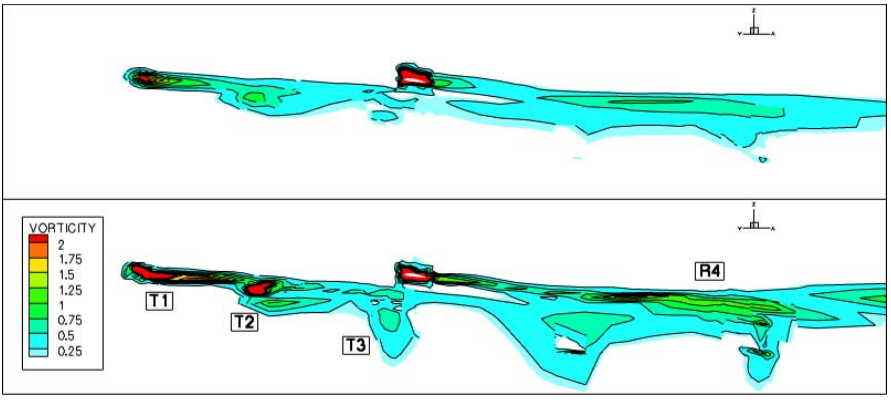


Fig. 10 Comparison of vortex resolution in the slice defined in Figure 9 with second order simulation (top) and third order simulation (bottom)

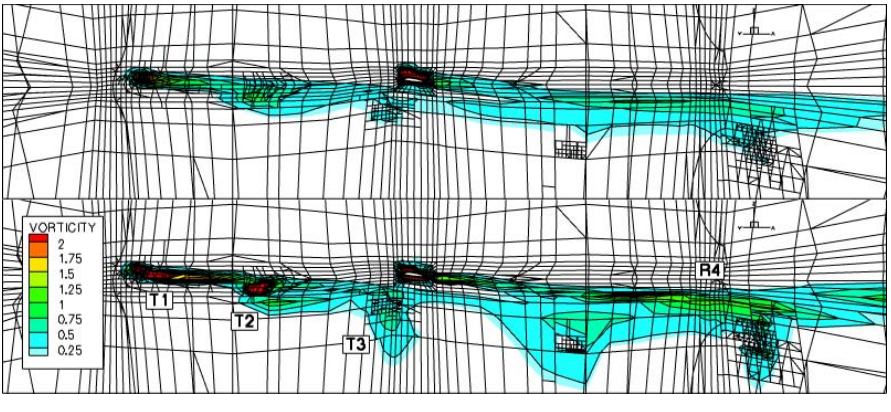


Fig. 11 Comparison of vortex resolution in the slice defined in Figure 9 with second order simulation (top) and third order simulation (bottom)

6 Conclusions

A feature-based pre-adaptation strategy has been described which aims at resolving the vortex wake of helicopter blades. The adaptation strategy is specifically designed to generate efficient space-time meshes for rotor simulations. The algorithm has been applied to the simulation of a rotor in forward flight. Given the limitations in computational capability, the necessary resolution cannot be obtained. Nonetheless, the vortex persistence is significantly improved by the local grid refinement.

References

1. Altmikus, A.R.M., Wagner, S., Servera, G., Beaumier, P.: A comparison: weak versus strong coupling for trimmed aeroelastic rotor simulations. In: Proceedings of the 29th European Rotorcraft Forum (September 2003)
2. Bottasso, C.L., Shephard, M.S.: A parallel adaptive finite element Euler flow solver for rotary wing aerodynamics. AIAA Journal 35(6), 937–944 (1997)
3. Buchtula, B., Wagner, S.: Rotory wing aeroelasticity in forward flight with refined wake modeling. In: Proceedings of the 24th European Rotorcraft Forum, Marseille (September 1998)
4. Caradonna, F.X.: Development and challenges in rotorcraft aerodynamics. AIAA paper, 2000-0109 (2000)
5. Datta, A., Nixon, M., Chopra, I.: Review of rotor loads prediction with the emergence of rotorcraft CFD. J. of the American Helicopter Society 52(4), 287–317 (2007)
6. Dietz, M., Kramer, E., Wagner, S.: Tip vortex conservation using on a main rotor in slow descent flight using vortex-adapted chimera grids. AIAA, 2006-3478 (2006)
7. Duraisamy, K., Baeder, J.D.: High resolution wake capturing methodology for hovering rotors. J. of the American Helicopter Society 52(2), 110–122 (2006)
8. Jeong, J., Hussain, F.: On the identification of a vortex. J. Fluid Mech. 285, 69–94 (1995)
9. Lim, J.W., Strawn, R.C.: Prediction of HART II rotor BVI loading and wake system using CFD/CSD loose coupling. AIAA paper, 2007-1281 (2007)
10. Ochi, A., Aoyama, T., Saito, S., Shima, E., Yamakawa, E.: BVI noise predictions by moving overlapped grid method. In: Proceedings of the AHS 55th Forum, Montreal, Canada, May 25-27, pp. 1400–1413 (1999)
11. Pomin, H., Wagner, S.: Aeroelastic analysis of helicopter rotor blades on deformable Chimera grids. J. of Aircraft 41(3), 577–584 (2004)
12. Schultz, K.-J., Splettstoesser, W., Junker, B., Wagner, W., Schoell, E., Arnaud, G., Mercker, E., Pengel, K., Fertis, D.: A parametric wind tunnel test on rotorcraft aerodynamics and aeroacoustics (helishape) — test documentation and representative results. Technical Report DLR-IB-129-96/25, DLR (1996)
13. van der Vegt, J.J.W., van der Ven, H.: Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows. Part I. General formulation. J. Comput. Phys. 182, 546–585 (2002)
14. van der Ven, H.: An adaptive multitime multigrid algorithm for time-periodic flows. J. Comput. Phys. 227, 5286–5303 (2008)
15. van der Ven, H., Boelens, O.J.: High-order simulation of a rotor in forward flight using a four-dimensional adaptive flow solver. In: Proceedings of the 34th European Rotorcraft Forum, Liverpool (September 2008)
16. Wake, B.E., Choi, D.: Investigation of high-order upwind differencing for vortex convection. AIAA Journal 34(2), 332–337 (1996)

Chapter 28
High–Order *hp*–Adaptive Discontinuous Galerkin Finite Element Methods for Compressible Fluid Flows

Stefano Giani and Paul Houston

Abstract. This article is concerned with the construction of general isotropic and anisotropic adaptive strategies, as well as *hp*–mesh refinement techniques, in combination with dual–weighted–residual *a posteriori* error indicators for the discontinuous Galerkin finite element discretization of compressible fluid flow problems.

1 Introduction

The development of Discontinuous Galerkin (DG) methods for the numerical approximation of the Euler and Navier-Stokes equations is an extremely exciting research topic which is currently being developed by a number of groups all over the world, cf. [1, 2, 5, 8, 9, 10, 14], for example. DG methods have several important advantages over well established finite volume methods. The concept of higher-order discretization is inherent to the DG method. The stencil is minimal in the sense that each element communicates only with its direct neighbors. In particular, in contrast to the increasing stencil size needed to increase the accuracy of classical finite volume methods, the stencil of DG methods is the same for any order of accuracy which has important advantages for the implementation of boundary conditions and for the parallel efficiency of the method. Moreover, due this simple communication at element interfaces, elements with so–called hanging nodes can be easily treated, a fact that simplifies local mesh refinement (*h*–refinement). Additionally, the communication at element interfaces is identical for any order of the method which

Stefano Giani
School of Mathematical Sciences, University of Nottingham,
University Park, Nottingham NG7 2RD, UK
e-mail: Stefano.Giani@nottingham.ac.uk

Paul Houston
School of Mathematical Sciences, University of Nottingham,
University Park, Nottingham NG7 2RD, UK
e-mail: Paul.Houston@nottingham.ac.uk

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 399–411, 2010.
springerlink.com

simplifies the use of methods with different polynomial orders p in adjacent elements. This allows for the variation of the order of polynomials over the computational domain (p–refinement), which in combination with h–refinement leads to so–called hp–adaptivity.

Mesh adaptation in finite element discretizations should be based on rigorous *a posteriori* error estimates; for hyperbolic/nearly–hyperbolic equations such estimates should reflect the inherent mechanisms of error propagation (see [12]). These considerations are particularly important when local quantities such as point values, local averages or flux integrals of the analytical solution are to be computed with high accuracy. Selective error estimates of this kind can be obtained by the optimal control technique proposed in [4] and [3] which is based on duality arguments analogous to those from the *a priori* error analysis of finite element methods. In the resulting *a posteriori* error estimates the element-residuals of the computed solution are multiplied by local weights involving the adjoint solution. These weights represent the sensitivity of the relevant error quantity with respect to variations of the local mesh size. Since the adjoint solution is usually unknown analytically, it has to be approximated numerically. On the basis of the resulting *a posteriori* error estimate the current mesh is locally adapted and then new approximations to the primal and adjoint solution are computed.

This article develops duality-based *a posteriori* error estimation of DG finite element methods, together with the application of these computable bounds within automatic adaptive finite element algorithms. Here, a variety of isotropic and anisotropic adaptive strategies, as well as hp–mesh refinement will be investigated.

2 Compressible Navier-Stokes Equations

In this article, we consider both two– and three–dimensional inviscid and laminar compressible flow problems. With this in mind, for generality, in this section we introduce the stationary compressible Navier-Stokes equations in three-dimensions:

$$\nabla \cdot (\mathscr{F}^c(\mathbf{u}) - \mathscr{F}^v(\mathbf{u}, \nabla \mathbf{u})) = 0 \quad \text{in } \Omega, \tag{1}$$

where Ω is an open bounded domain in $\mathbb{R}^d$ with boundary Γ; for the purposes of this section, we set $d = 3$. The vector of conservative variables $\mathbf{u}$ is given by $\mathbf{u} = (\rho, \rho v_1, \rho v_2, \rho v_3, \rho E)^\top$ and the convective flux $\mathscr{F}^c(\mathbf{u}) = \left(\mathbf{f}_1^c(\mathbf{u}), \mathbf{f}_2^c(\mathbf{u}), \mathbf{f}_3^c(\mathbf{u})\right)^\top$ is given by $\mathbf{f}_1^c(\mathbf{u}) = (\rho v_1, \rho v_1^2 + p, \rho v_1 v_2, \rho v_1 v_3, \rho H v_1)^\top$, $\mathbf{f}_2^c(\mathbf{u}) = (\rho v_2, \rho v_2 v_1, \rho v_2^2 + p, \rho v_2 v_3, \rho H v_2)^\top$, and $\mathbf{f}_3^c(\mathbf{u}) = (\rho v_3, \rho v_3 v_1, \rho v_3 v_2, \rho v_3^2 + p, \rho H v_3)^\top$. Furthermore, $\mathbf{f}_k^v(\mathbf{u}, \nabla \mathbf{u}) = (0, \tau_{1k}, \tau_{2k}, \tau_{3k}, \tau_{kl} v_l + \mathscr{K} T_{x_k})^\top$, $k = 1, 2, 3$. Here ρ, $\mathbf{v} = (v_1, v_2, v_3)^\top$, p, E and T denote the density, velocity vector, pressure, specific total energy, and temperature, respectively. Moreover, $\mathscr{K}$ is the thermal conductivity coefficient and H is the total enthalpy given by $H = E + \frac{p}{\rho} = e + \frac{1}{2}\mathbf{v}^2 + \frac{p}{\rho}$, where e is the specific static internal energy, and the pressure is determined by the equation of state of an ideal gas $p = (\gamma - 1)\rho e$, where $\gamma = c_p / c_v$ is the ratio of specific heat capacities at constant pressure, c_p, and constant volume, c_v; for dry air, $\gamma = 1.4$. For a Newtonian fluid,

the viscous stress tensor is given by $\tau = \mu\left(\nabla\mathbf{v} + (\nabla\mathbf{v})^\top - \frac{2}{3}(\nabla\cdot\mathbf{v})I\right)$, where μ is the dynamic viscosity coefficient; the temperature T is given by $\mathscr{K}T = \frac{\mu\gamma}{Pr}\left(E - \frac{1}{2}\mathbf{v}^2\right)$, where $\mathrm{Pr} = 0.72$ is the Prandtl number. For the purposes of discretization, we rewrite the compressible Navier–Stokes equations (1) in the following (equivalent) form:

$$\nabla\cdot(\mathscr{F}^c(\mathbf{u}) - G(\mathbf{u})\nabla\mathbf{u}) \equiv \frac{\partial}{\partial x_k}\left(\mathbf{f}_k^c(\mathbf{u}) - G_{kl}(\mathbf{u})\frac{\partial\mathbf{u}}{\partial x_l}\right) = 0 \quad \text{in } \Omega.$$

Here, the matrices $G_{kl}(\mathbf{u}) = \partial\mathbf{f}_k^v(\mathbf{u},\nabla\mathbf{u})/\partial u_{x_l}$, for $k,l = 1,2,3$, are the homogeneity tensors defined by $\mathbf{f}_k^v(\mathbf{u},\nabla\mathbf{u}) = G_{kl}(\mathbf{u})\partial\mathbf{u}/\partial x_l$, $k = 1,2,3$.

3 DG Discretization

In this section we introduce the adjoint-consistent interior penalty DG discretization of the compressible Navier–Stokes equations (1), cf. [11] for further details.

First, we begin by introducing some notation. We assume that $\Omega \subset \mathbb{R}^d$, $d = 2,3$, can be subdivided into a mesh $\mathscr{T}_h = \{\kappa\}$ consisting of tensor-product (quadrilaterals, $d = 2$, and hexahedra, $d = 3$) open element domains κ. For each $\kappa \in \mathscr{T}_h$, we denote by $\mathbf{n}_\kappa$ the unit outward normal vector to the boundary $\partial\kappa$. We assume that each $\kappa \in \mathscr{T}_h$ is an image of a fixed reference element $\hat{\kappa}$, that is, $\kappa = \sigma_\kappa(\hat{\kappa})$ for all $\kappa \in \mathscr{T}_h$, where $\hat{\kappa}$ is the open unit hypercube in $\mathbb{R}^d$, and σ_κ is a smooth bijective mapping. On the reference element $\hat{\kappa}$ we define the polynomial space $\mathscr{Q}_\mathbf{p}$ with respect to the anisotropic polynomial degree vector $\mathbf{p} := \{p_i\}_{i=1,\ldots,d}$ as follows: $\mathscr{Q}_\mathbf{p} := \mathrm{span}\{\Pi_{i=1}^d \hat{x}_i^j : 0 \le j \le p_i\}$. With this notation, we introduce the following (anisotropic) finite element space.

Definition 1. Let $\mathbf{p} = (\mathbf{p}_\kappa : \kappa \in \mathscr{T}_h)$ be the composite polynomial degree vector of the elements in a given finite element mesh $\mathscr{T}_h$. We define the finite element space with respect to Ω, $\mathscr{T}_h$, and $\mathbf{p}$ by $\mathbf{V}_{h,\mathbf{p}} = \{u \in L_2(\Omega) : u|_\kappa \circ \sigma_\kappa \in [\mathscr{Q}_{\mathbf{p}_\kappa}]^{d+2}\}$.

In the case when the elemental polynomial degree vector $\mathbf{p}_\kappa = \{p_{\kappa,i}\}_{i=1,\ldots,d}$, $\kappa \in \mathscr{T}_h$, is isotropic in the sense that $p_{\kappa,1} = p_{\kappa,2} = \ldots = p_{\kappa,d} \equiv p_\kappa$ for all elements κ in the finite element mesh $\mathscr{T}_h$, then we write $\mathbf{V}_{h,\mathbf{p}_{\mathrm{iso}}}$ in lieu of $\mathbf{V}_{h,\mathbf{p}}$, where $\mathbf{p}_{\mathrm{iso}} = (p_\kappa : \kappa \in \mathscr{T}_h)$. Additionally, in the case when the polynomial degree is both isotropic and uniformly distributed over the mesh $\mathscr{T}_h$, i.e., when $p_\kappa = p$ for all κ in $\mathscr{T}_h$, then we simply denote the finite element space by $\mathbf{V}_{h,p}$.

An *interior face* of $\mathscr{T}_h$ is defined as the (non-empty) $(d-1)$–dimensional interior of $\partial\kappa^+ \cap \partial\kappa^-$, where κ^+ and κ^- are two adjacent elements of $\mathscr{T}_h$, not necessarily matching. A *boundary face* of $\mathscr{T}_h$ is defined as the (non-empty) $(d-1)$–dimensional interior of $\partial\kappa \cap \Gamma$, where κ is a boundary element of $\mathscr{T}_h$. We denote by $\Gamma_\mathscr{I}$ the union of all interior faces of $\mathscr{T}_h$. Let κ^+ and κ^- be two adjacent elements of $\mathscr{T}_h$, and $\mathbf{x}$ an arbitrary point on the interior face $f = \partial\kappa^+ \cap \partial\kappa^-$. Furthermore, let $\mathbf{v}$ and $\underline{\tau}$ be vector- and matrix-valued functions, respectively, that are smooth inside each element $\kappa^\pm$. By $(\mathbf{v}^\pm, \underline{\tau}^\pm)$, we denote the traces of $(\mathbf{v}, \underline{\tau})$ on f taken from within the interior of $\kappa^\pm$, respectively. Then, the averages of $\mathbf{v}$ and $\underline{\tau}$ at $\mathbf{x} \in f$ are given by

$\{\{\mathbf{v}\}\} = (\mathbf{v}^+ + \mathbf{v}^-)/2$ and $\{\{\underline{\tau}\}\} = (\underline{\tau}^+ + \underline{\tau}^-)/2$, respectively. Similarly, the jump of $\mathbf{v}$ at $\mathbf{x} \in f$ is given by $\underline{[\![\mathbf{v}]\!]} = \mathbf{v}^+ \otimes \mathbf{n}_{\kappa^+} + \mathbf{v}^- \otimes \mathbf{n}_{\kappa^-}$, where we denote by $\mathbf{n}_{\kappa^\pm}$ the unit outward normal vector of $\kappa^\pm$, respectively. On $f \subset \Gamma$, we set $\{\{\mathbf{v}\}\} = \mathbf{v}$, $\{\{\underline{\tau}\}\} = \underline{\tau}$ and $\underline{[\![\mathbf{v}]\!]} = \mathbf{v} \otimes \mathbf{n}$, where $\mathbf{n}$ denotes the unit outward normal vector to Γ.

The DG discretization of (1) is given by: find $\mathbf{u}_h \in \mathbf{V}_{h,\mathbf{p}}$ such that

$$\begin{aligned}\mathscr{N}(\mathbf{u}_h,\mathbf{v}) \equiv & -\int_\Omega \mathscr{F}^c(\mathbf{u}_h) : \nabla_h \mathbf{v}\,\mathrm{d}\mathbf{x} + \sum_{\kappa\in\mathscr{T}_h}\int_{\partial\kappa\setminus\Gamma} \mathscr{H}(\mathbf{u}_h^+,\mathbf{u}_h^-,\mathbf{n}^+)\cdot\mathbf{v}^+\,\mathrm{d}s \\ & + \int_\Omega \mathscr{F}^v(\mathbf{u}_h,\nabla_h\mathbf{u}_h) : \nabla_h\mathbf{v}\,\mathrm{d}\mathbf{x} - \int_{\Gamma_\mathscr{I}} \{\{\mathscr{F}^v(\mathbf{u}_h,\nabla_h\mathbf{u}_h)\}\} : \underline{[\![\mathbf{v}]\!]}\,\mathrm{d}s \\ & - \int_{\Gamma_\mathscr{I}} \{\{G^\top(\mathbf{u}_h)\nabla_h\mathbf{v}\}\} : \underline{[\![\mathbf{u}_h]\!]}\,\mathrm{d}s + \int_{\Gamma_\mathscr{I}} \underline{\delta}(\mathbf{u}_h) : \underline{[\![\mathbf{v}]\!]}\,\mathrm{d}s + \mathscr{N}_\Gamma(\mathbf{u}_h,\mathbf{v}) = 0 \quad (2)\end{aligned}$$

for all $\mathbf{v}$ in $\mathbf{V}_{h,\mathbf{p}}$. The subscript h on the operator ∇_h is used to denote the discrete counterpart of ∇, defined elementwise. Here, $\mathscr{H}(\cdot,\cdot,\cdot)$ denotes the (convective) numerical flux function; this may be chosen to be any two–point monotone Lipschitz function which is both consistent and conservative. For the purposes of this article, we employ the Vijayasundaram flux.

In order to define the penalization function $\underline{\delta}(\cdot)$ arising in the DG method (2), we first introduce the local (anisotropic) mesh and polynomial functions $\mathtt{h}$ and $\mathtt{p}$, respectively. To this end, the function $\mathtt{h}$ in $L_\infty(\Gamma_\mathscr{I}\cup\Gamma)$ is defined as $\mathtt{h}(\mathbf{x}) = \min\{m_{\kappa^+}, m_{\kappa^-}\}/m_f$, if $\mathbf{x}$ is in the interior of $f = \partial\kappa^+ \cap \partial\kappa^-$ for two neighboring elements in the mesh $\mathscr{T}_h$, and $\mathtt{h}(\mathbf{x}) = m_\kappa/m_f$, if $\mathbf{x}$ is in the interior of $f = \partial\kappa\cap\Gamma$. Here, for a given (open) bounded set $\omega \subset \mathbb{R}^s$, $s \geq 1$, we write m_ω to denote the s–dimensional measure (volume) of ω. In a similar fashion, we define $\mathtt{p}$ in $L_\infty(\Gamma_\mathscr{I}\cup\Gamma)$ by $\mathtt{p}(\mathbf{x}) = \max\{p_{\kappa^+,i}, p_{\kappa^-,j}\}$ for κ^+, κ^- as above, where the indices i and j are chosen such that $\sigma^{-1}_{\kappa^+}(f)$ and $\sigma^{-1}_{\kappa^-}(f)$ are orthogonal to the ith–, respectively, jth–coordinate direction on the reference element $\hat{\kappa}$. For $\mathbf{x}$ in the interior of a boundary face $f = \partial\kappa\cap\Gamma$, we write $\mathtt{p}(\mathbf{x}) = p_{\kappa,i}$, when $\sigma^{-1}_\kappa(f)$ is orthogonal to the ith–coordinate direction on $\hat{\kappa}$. With this notation the penalization term is given by

$$\underline{\delta}(\mathbf{u}_h) = C_{\mathrm{IP}}\frac{\mathtt{p}^2}{\mathtt{h}}\{\{G(\mathbf{u}_h)\}\}\underline{[\![\mathbf{u}_h]\!]},$$

where C_{IP} is a (sufficiently large) positive constant, cf. [7].

Finally, we define the boundary terms present in the form $\mathscr{N}_\Gamma(\cdot,\cdot)$ by

$$\begin{aligned}\mathscr{N}_\Gamma(\mathbf{u}_h,\mathbf{v}) = & \int_\Gamma \mathscr{H}_\Gamma(\mathbf{u}_h^+,\mathbf{u}_\Gamma(\mathbf{u}_h^+),\mathbf{n}^+)\cdot\mathbf{v}^+\,\mathrm{d}s + \int_\Gamma \underline{\delta}_\Gamma(\mathbf{u}_h^+) : \mathbf{v}\otimes\mathbf{n}\,\mathrm{d}s \\ & - \int_\Gamma \mathbf{n}\cdot\mathscr{F}^v_\Gamma(\mathbf{u}_\Gamma(\mathbf{u}_h^+),\nabla_h\mathbf{u}_h^+)\,\mathbf{v}^+\,\mathrm{d}s - \int_\Gamma \left(G_\Gamma^\top(\mathbf{u}_h^+)\nabla_h\mathbf{v}_h^+\right) : \left(\mathbf{u}_h^+ - \mathbf{u}_\Gamma(\mathbf{u}_h^+)\right)\otimes\mathbf{n}\,\mathrm{d}s,\end{aligned}$$

where $\underline{\delta}_\Gamma(\mathbf{u}_h) = C_{\mathrm{IP}}\frac{\mathtt{p}^2}{\mathtt{h}}G_\Gamma(\mathbf{u}_h^+)\,(\mathbf{u}_h - \mathbf{u}_\Gamma(\mathbf{u}_h))\otimes\mathbf{n}$. Here, the viscous boundary flux $\mathscr{F}^v_\Gamma$ and the corresponding homogeneity tensor G_Γ are defined by

$$\mathscr{F}^v_\Gamma(\mathbf{u}_h,\nabla\mathbf{u}_h) = \mathscr{F}^v(\mathbf{u}_\Gamma(\mathbf{u}_h),\nabla\mathbf{u}_h) = G_\Gamma(\mathbf{u}_h)\nabla\mathbf{u}_h = G(\mathbf{u}_\Gamma(\mathbf{u}_h))\nabla\mathbf{u}_h.$$

Furthermore, on portions of the boundary Γ where adiabatic boundary conditions are imposed, $\mathscr{F}_\Gamma^v$ and G_Γ are modified such that $\mathbf{n}\cdot\nabla T = 0$. The convective boundary flux $\mathscr{H}_\Gamma$ is defined by $\mathscr{H}_\Gamma(\mathbf{u}_h^+, \mathbf{u}_\Gamma(\mathbf{u}_h^+), \mathbf{n}) = \mathbf{n}\cdot\mathscr{F}^c(\mathbf{u}_\Gamma(\mathbf{u}_h^+))$. Finally, the boundary function $\mathbf{u}_\Gamma(\mathbf{u})$ is given according to the type of boundary condition imposed; for details, we refer to [11], for example.

4 *A Posteriori* Error Estimation

In this section we consider the derivation of an adjoint-based *a posteriori* bound on the error in a given computed target functional $J(\cdot)$ of practical interest, such as the drag, lift, or moment on a body immersed within a compressible fluid, for example.

Assuming that the functional of interest $J(\cdot)$ is differentiable, we write $\bar{J}(\cdot;\cdot)$ to denote the mean value linearization of $J(\cdot)$ defined by

$$\bar{J}(\mathbf{u},\mathbf{u}_h;\mathbf{u}-\mathbf{u}_h) = J(\mathbf{u}) - J(\mathbf{u}_h) = \int_0^1 J'[\theta\mathbf{u}+(1-\theta)\mathbf{u}_h](\mathbf{u}-\mathbf{u}_h)\,\mathrm{d}\theta,$$

where $J'[\mathbf{w}](\cdot)$ denotes the Fréchet derivative of $J(\cdot)$ evaluated at some $\mathbf{w}$ in $\mathbf{V}$. Here, $\mathbf{V}$ is some suitably chosen function space such that $\mathbf{V}_{h,\mathbf{p}} \subset \mathbf{V}$.

Analogously, for $\mathbf{v}$ in $\mathbf{V}$, we define the mean–value linearization of $\mathscr{N}(\cdot,\mathbf{v})$ by

$$\mathscr{M}(\mathbf{u},\mathbf{u}_h;\mathbf{u}-\mathbf{u}_h,\mathbf{v}) = \mathscr{N}(\mathbf{u},\mathbf{v}) - \mathscr{N}(\mathbf{u}_h,\mathbf{v}) = \int_0^1 \mathscr{N}'[\theta\mathbf{u}+(1-\theta)\mathbf{u}_h](\mathbf{u}-\mathbf{u}_h,\mathbf{v})\,\mathrm{d}\theta.$$

Here, $\mathscr{N}'[\mathbf{w}](\cdot,\mathbf{v})$ denotes the Fréchet derivative of $\mathbf{u}\mapsto\mathscr{N}(\mathbf{u},\mathbf{v})$, for $\mathbf{v}\in\mathbf{V}$ fixed, at some $\mathbf{w}$ in $\mathbf{V}$. Let us now introduce the adjoint problem: find $\mathbf{z}\in\mathbf{V}$ such that

$$\mathscr{M}(\mathbf{u},\mathbf{u}_h;\mathbf{w},\mathbf{z}) = \bar{J}(\mathbf{u},\mathbf{u}_h;\mathbf{w}) \quad \forall\mathbf{w}\in\mathbf{V}. \tag{3}$$

With this notation, we may state the following error representation formula

$$J(\mathbf{u}) - J(\mathbf{u}_h) = \mathscr{R}(\mathbf{u}_h,\mathbf{z}-\mathbf{z}_h) \equiv \sum_{\kappa\in\mathscr{T}_h}\eta_\kappa, \tag{4}$$

where $\mathscr{R}(\mathbf{u}_h,\mathbf{z}-\mathbf{z}_h) = -\mathscr{N}(\mathbf{u}_h,\mathbf{z}-\mathbf{z}_h)$ includes primal residuals multiplied by the difference of the adjoint solution $\mathbf{z}$ and an arbitrary discrete function $\mathbf{z}_h\in\mathbf{V}_{h,\mathbf{p}}$, and η_κ denotes the local elemental indicators; see [8, 10] for details.

We note that the error representation formula (4) depends on the unknown analytical solution $\mathbf{z}$ to the adjoint problem (3) which in turn depends on the unknown analytical solution $\mathbf{u}$. Thus, in order to render these quantities computable, both $\mathbf{u}$ and $\mathbf{z}$ must be replaced by suitable approximations. Here, the linearizations leading to $\mathscr{M}(\mathbf{u},\mathbf{u}_h;\cdot,\cdot)$ and $\bar{J}(\mathbf{u},\mathbf{u}_h;\cdot)$ are performed about $\mathbf{u}_h$ and the adjoint solution $\mathbf{z}$ is approximated by computing the DG approximation $\bar{\mathbf{z}}_h\in\bar{\mathbf{V}}_{h,\mathbf{p}}$, where $\bar{\mathbf{V}}_{h,\mathbf{p}}$ is an *adjoint* finite element space from which the approximate adjoint solution $\bar{\mathbf{z}}_h$ is sought. For the purposes of this article, we set $\bar{\mathbf{V}}_{h,\mathbf{p}} = \mathbf{V}_{h,\mathbf{p}_d}$, where $\mathbf{p}_d = \mathbf{p}+\mathbf{1}$.

In the following sections we consider the development of a variety of adaptive mesh refinement algorithms in order to efficiently control the error in the computed target functional of interest.

5 Anisotropic Mesh Adaptation

In this section we first consider the automatic design of anisotropic finite element meshes $\mathcal{T}_h$, assuming that the underlying polynomial degree distribution is both uniform and fixed, i.e., when $\mathbf{u}_h \in \mathbf{V}_{h,p}$. To this end, elements are marked for refinement/derefinement according to the size of the (approximate) error indicators $|\bar{\eta}_\kappa|$, based on employing a fixed fraction strategy, for example. Here, $\bar{\eta}_\kappa$ is defined analogously to η_κ in (4) with $\mathbf{z}$ replaced by $\bar{\mathbf{z}}_h$.

To subdivide the elements which have been flagged for refinement, we employ a simple Cartesian refinement strategy; here, elements may be subdivided either anisotropically or isotropically according to the three refinements (in two–dimensions, i.e., $d = 2$) depicted in Figure 1. In order to determine the optimal refinement, we propose the following strategy based on choosing the most competitive subdivision of κ from a series of trial refinements, whereby an approximate local error indicator on each trial patch is determined.

Algorithm 5.1 *Given an element κ in the computational mesh $\mathcal{T}_h$ (which has been marked for refinement), we first construct the mesh patches $\mathcal{T}_{h,i}$, $i = 1,2,3$, based on refining κ according to Figures 1(a), (b), & (c), respectively. On each mesh patch, $\mathcal{T}_{h,i}$, $i = 1,2,3$, we compute the approximate error estimators $\mathscr{R}_{\kappa,i}(\mathbf{u}_{h,i}, \bar{\mathbf{z}}_{h,i} - \mathbf{z}_h) = \sum_{\kappa' \in \mathcal{T}_{h,i}} \eta_{\kappa',i}$, for $i = 1,2,3$, respectively. Here, $\mathbf{u}_{h,i}$, $i = 1,2,3$, is the DG approximation computed on the mesh patch $\mathcal{T}_{h,i}$, $i = 1,2,3$, respectively, based on enforcing appropriate boundary conditions on $\partial\kappa$ computed from the original DG solution $\mathbf{u}_h$ on the portion of the boundary $\partial\kappa$ of κ which is interior to the computational domain Ω, i.e., where $\partial\kappa \cap \Gamma = \emptyset$. Similarly, $\bar{\mathbf{z}}_{h,i}$ denotes the DG approximation to $\mathbf{z}$ computed on the local mesh patch $\mathcal{T}_{h,i}$, $i = 1,2,3$, respectively, with polynomials of degree p_d, based on employing suitable boundary conditions on $\partial\kappa \cap \Gamma = \emptyset$ derived from $\bar{\mathbf{z}}_h$. Finally, $\eta_{\kappa',i}$, $i = 1,2,3$, is defined in an analogous manner to η_κ, cf. above, with $\mathbf{u}_h$ and $\mathbf{z}$ replaced by $\mathbf{u}_{h,i}$ and $\bar{\mathbf{z}}_{h,i}$, respectively.*

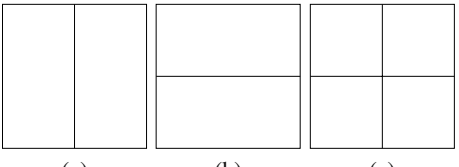


Fig. 1 Cartesian refinement in 2D: (a) & (b) Anisotropic refinement; (c) Isotropic refinement

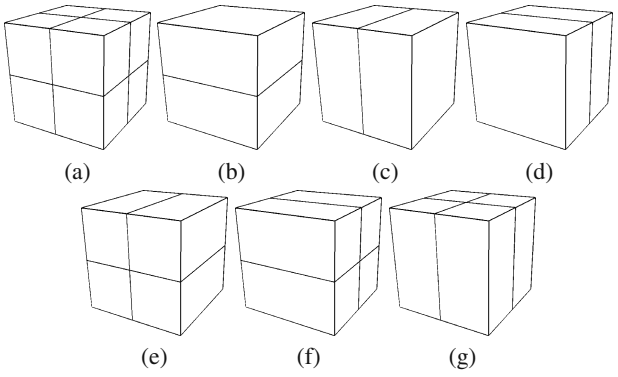


Fig. 2 Cartesian refinement in 3D

The element κ is then refined according to the subdivision of κ which satisfies

$$\min_{i=1,2,3} \frac{|\eta_\kappa| - |\mathscr{R}_{\kappa,i}(\mathbf{u}_{h,i}, \bar{\mathbf{z}}_{h,i} - \mathbf{z}_h)|}{\#dofs(\mathscr{T}_{h,i}) - \#dofs(\kappa)},$$

where $\#dofs(\kappa)$ and $\#dofs(\mathscr{T}_{h,i})$, $i = 1,2,3$, denote the number of degrees of freedom associated with κ and $\mathscr{T}_{h,i}$, $i = 1,2,3$, respectively, cf. [6].

The extension of this approach to the case when $\mathscr{T}_h$ is a hexahedral mesh in three-dimensions follows in an analogous fashion. Indeed, in this setting, we again employ a Cartesian refinement strategy whereby elements may be subdivided either isotropically or anisotropically according to the four refinements depicted in Figures 2(a)–(d). We remark that we assume that a face in the computational mesh is a complete face of at least one element. This assumption means that the refinements depicted in Figures 1(b)–(d) may be inadmissible. In this situation, we replace the selected refinement by either one of the anisotropic mesh refinements depicted in Figures 2(e)–(g), or if necessary, an isotropic refinement is performed.

5.1 Numerical Experiments

In this section we present a number of experiments to numerically demonstrate the performance of the anisotropic adaptive algorithm outlined in the previous section.

5.1.1 ADIGMA MTC3: Laminar Flow around a NACA0012 Airfoil

In this example, we consider the subsonic viscous flow around a NACA0012 airfoil. At the farfield (inflow) boundary we specify a Mach 0.5 flow at an angle of attack $\alpha = 2^\circ$, with Reynolds number $\mathrm{Re} = 5000$; on the walls of the airfoil geometry, we impose a zero heat flux (adiabatic) no-slip boundary condition. Here, we consider the estimation of the drag coefficient C_d; i.e., the target functional of interest

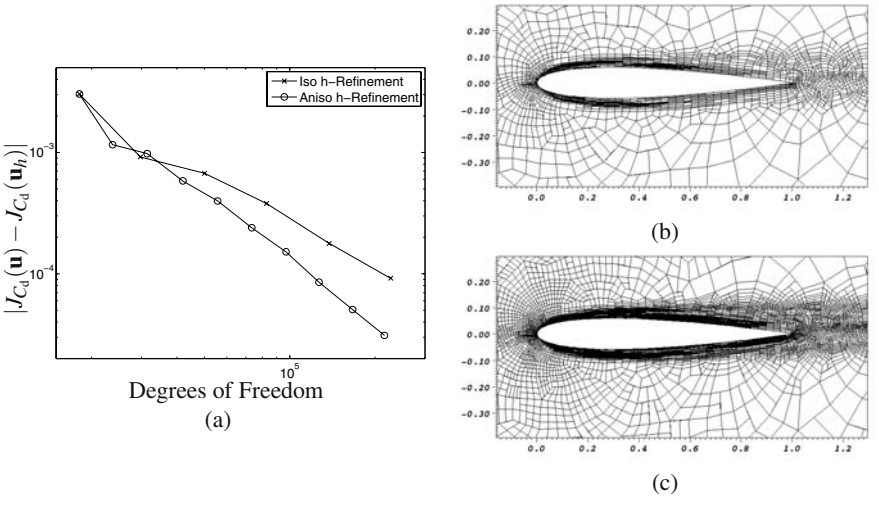


Fig. 3 ADIGMA MTC3 test case: (a) Comparison between adaptive isotropic and anisotropic mesh refinement; Anisotropic mesh after (b) 4 adaptive refinements, with 3485 elements; (c) 8 adaptive refinements, with 10410 elements

is given by $J(\cdot) \equiv J_{C_d}(\cdot)$. The initial starting mesh is taken to be an unstructured quadrilateral–dominant hybrid mesh consisting of both quadrilateral and triangular elements; here, the total number of elements is 1134. Furthermore, curved boundaries are approximated by piecewise quadratic polynomials. In Figure 3(a) we plot the error in the computed target functional $J_{C_d}(\cdot)$ using both an isotropic (only) mesh refinement algorithm, together with the anisotropic refinement strategy outlined in Section 5. From Figure 3(a), we observe the superiority of employing the anisotropic mesh refinement algorithm in comparison with standard isotropic subdivision of the elements. Indeed, the error $|J_{C_d}(\mathbf{u}) - J_{C_d}(\mathbf{u}_h)|$ computed on the series of anisotropically refined meshes designed using the proposed algorithm outlined in Section 5 is (almost) always less than the corresponding quantity computed on the isotropic grids. Indeed, on the final mesh anisotropic mesh refinement leads to an improvement in $|J_{C_d}(\mathbf{u}) - J_{C_d}(\mathbf{u}_h)|$ of over 60% compared with the same quantity computed using isotropic mesh refinement. The meshes generated after 4 and 8 anisotropic adaptive mesh refinements are shown in Figures 3(b) & (c), respectively. Here, we clearly observe significant anisotropic refinement of the viscous boundary layer, as we would expect.

5.1.2 ADIGMA BTC0: Laminar Flow around a Streamlined Body

In this second example we consider laminar flow past a streamlined three–dimensional body. Here, the geometry of the body is based on a 10 percent thick airfoil with boundaries constructed by a surface of revolution. The BTC0 geometry is considered at laminar conditions with inflow Mach number equal to 0.5, at an angle of

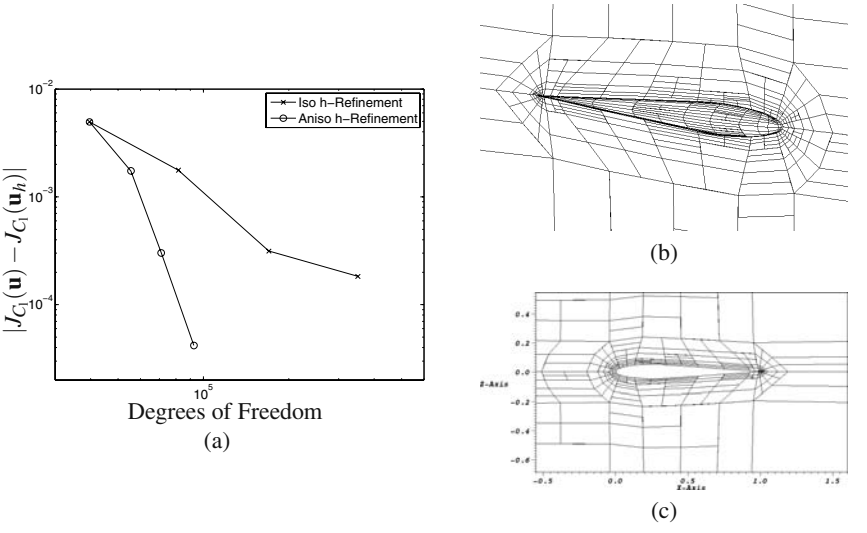


Fig. 4 ADIGMA BTC0 test case (laminar): (a) Comparison between adaptive isotropic and anisotropic mesh refinement; Anisotropic mesh after 3 adaptive refinements, with 2314 elements: (b) Boundary mesh; (c) Symmetry plane

attack $\alpha = 1^\circ$, and Reynolds number $\mathrm{Re} = 5000$ with adiabatic no-slip wall boundary condition imposed. Here, we suppose that the aim of the computation is to calculate the lift coefficient C_l; i.e., $J(\cdot) \equiv J_{C_l}(\cdot)$. In this example, the initial starting mesh is taken to be an unstructured hexahedral mesh with 992 elements. In Figure 4(a) we plot the error in the computed target functional $J_{C_l}(\cdot)$ using both an isotropic (only) mesh refinement algorithm, together with the anisotropic refinement strategy outlined in Section 5. From Figure 4(a), we again observe the superiority of employing the anisotropic mesh refinement algorithm in comparison with standard isotropic subdivision of the elements. Indeed, the error $|J_{C_l}(\mathbf{u}) - J_{C_l}(\mathbf{u}_h)|$ computed on the series of anisotropically refined meshes designed using Algorithm 5.1 is always less than the corresponding quantity computed on the isotropic grids. Indeed, on the final mesh the true error between $J_{C_l}(\mathbf{u})$ and $J_{C_l}(\mathbf{u}_h)$ using anisotropic mesh refinement is over an order of magnitude smaller than the corresponding quantity when isotropic *h*–refinement is employed alone. The mesh generated after 3 anisotropic adaptive mesh refinements is shown in Figures 4(b) & (c). Here, we again observe significant anisotropic refinement of the viscous boundary layer.

6 *hp*–Adaptivity on Isotropically Refined Meshes

In this section we now consider the case when both the underlying finite element mesh $\mathscr{T}_h$ and the polynomial distribution are isotropic; thereby, $\mathbf{u}_h \in \mathbf{V}_{h,\mathbf{p}_{\mathrm{iso}}}$. The extension to general anisotropic finite element spaces will be considered in the

following section. In this setting, once an element has been selected for refinement/derefinement the key step in the design of such an (isotropic) hp–adaptive algorithm is the local decision taken on each element κ in the computational mesh as to which refinement strategy (i.e., h-refinement *via* local mesh subdivision or p-refinement by increasing the degree of the local polynomial approximation) should be employed on κ in order to obtain the greatest reduction in the error per unit cost. To this end, we employ the technique for assessing local smoothness developed in the article [13], which is based on monitoring the decay rate of the sequence of coefficients in the Legendre series expansion of a square–integrable function.

6.1 ADIGMA MTC1: Inviscid Flow around a NACA0012 Airfoil

In this section we consider the performance of the goal–oriented hp–refinement algorithm outlined above for the ADIGMA MTC1 test case: inviscid compressible flow around a NACA0012 airfoil with inflow Mach number equal to 0.5, at an angle of attack $\alpha = 2^\circ$. Here, we suppose that the aim of the computation is to calculate the pressure induced drag coefficient $C_{\rm dp}$; i.e., $J(\cdot) \equiv J_{C_{\rm dp}}(\cdot)$.

In Figure 5 we plot the error in the computed target functional $J_{C_{\rm dp}}(\cdot)$, using both h– and hp–refinement against the square–root of the number of degrees of freedom on a linear–log scale in the case of both a structured and unstructured initial mesh. In both cases, we see that after the initial transient, the error in the computed functional using hp–refinement becomes (on average) a straight line, thereby indicating exponential convergence of $J_{C_{\rm dp}}(\mathbf{u}_h)$ to $J_{C_{\rm dp}}(\mathbf{u})$. Figure 5 also demonstrates the superiority of the adaptive hp–refinement strategy over the standard adaptive h–refinement algorithm. In each case, on the final mesh the true error between $J_{C_{\rm dp}}(\mathbf{u})$ and $J_{C_{\rm dp}}(\mathbf{u}_h)$ using hp–refinement is almost 2 orders of magnitude smaller than the corresponding quantity when h–refinement is employed alone. Finally, in Figure 6

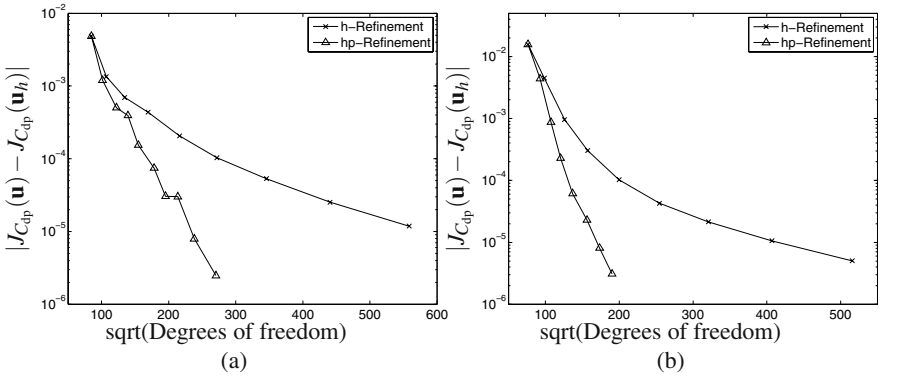
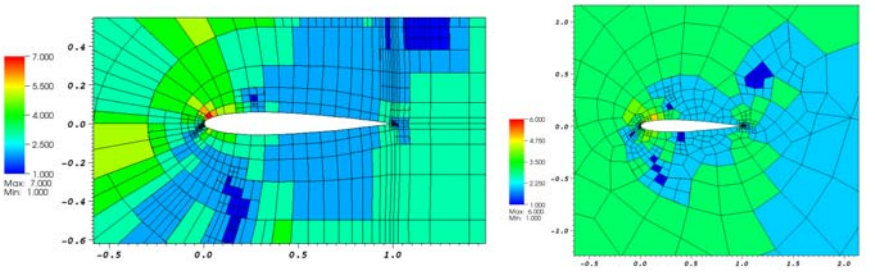


Fig. 5 ADIGMA MTC1 test case: Comparison between adaptive hp– and h–mesh refinement. (a) Structured initial mesh; (b) Unstructured initial mesh.



(a) (b)

Fig. 6 ADIGMA MTC1 test case: *hp*–Mesh distribution. (a) Structured initial mesh after 9 adaptive refinements; (b) Unstructured initial mesh after 7 adaptive refinements.

we show the *hp*–mesh distributions based on employing a structured and unstructured initial mesh after 9 and 7 adaptive refinement steps, respectively.

7 Anisotropic *hp*–Mesh Adaptation

Finally, in this section we consider the general case of automatically generating anisotropically refined computational meshes, together with an anisotropic polynomial degree distribution. With this in mind, once an element has been selected for refinement/derefinement a decision is first made whether to carry out an *h*-refinement/derefinement or *p*-enrichment/derefinement based on the technique outlined in Section 6, whereby the analyticity of the solutions $\mathbf{u}$ and $\mathbf{z}$ is assessed by studying the decay rates of their underlying Legendre coefficients. Once the *h*– and *p*–refinement flags have been determined on the basis of the above strategy, a decision regarding the type refinement to be undertaken — isotropic or anisotropic — must be made. Motivated by the work in Section 5, we employ a competitive refinement technique, whereby the "optimal" refinement is selected from a series of trial refinements. In the *h*–version setting, we again exploit the algorithm outlined in Section 5. For the case when an element has been selected for polynomial enrichment we consider the *p*–version counterpart of Algorithm 5.1 and solve local problems based on increasing the polynomial degrees anisotropically in one direction at a time by one degree, or isotropically by one degree; see [7] for details.

7.1 ADIGMA MTC3: Laminar Flow around a NACA0012 Airfoil

In this section we again consider the ADIGMA MTC3 test case and again suppose that the aim of the computation is to calculate the drag coefficient C_{d}, cf. Section 5.1.1. In Figure 7(a) we plot the error in the computed target functional $J_{C_{\mathrm{d}}}(\cdot)$, using a variety of *h*–/*hp*–adaptive algorithms against the square–root of the number of degrees of freedom on a linear–log scale in the case when an unstructured

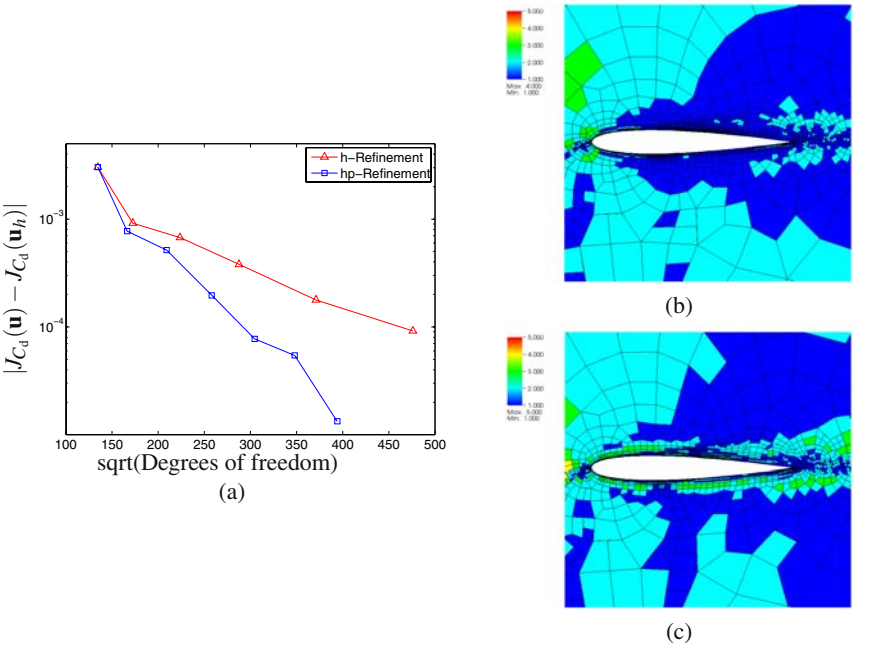


Fig. 7 ADIGMA MTC3 test case: (a) Comparison between different adaptive refinement strategies. Mesh distribution after 5 adaptive anisotropic hp–refinements, with 2200 elements and 52744 degrees of freedom: (b) h–/p_x–mesh distribution; (c) h–/p_y–mesh distribution.

initial mesh is employed. In particular, here we consider the performance of the following adaptive mesh refinement strategies: isotropic h–refinement, anisotropic h–refinement, isotropic hp-refinement, anisotropic h–/isotropic p–refinement, and anisotropic hp–refinement. Here, we clearly observe that as the flexibility of the underlying adaptive strategy is increased, thereby allowing for greater flexibility in the construction of the finite element space $V_{h,\mathbf{p}}$, the error in the computed target functional of interest is improved in the sense that the error in the computed value of $J_{C_d}(\cdot)$ is decreased for a fixed number of degrees of freedom. However, we point out that in the initial stages of refinement, all of the refinement algorithms perform in a similar manner. Indeed, it is not until the structure of the underlying analytical solution is resolved that we observe the benefits of increasing the complexity of the adaptive refinement strategy. Finally, we point out that the latter three refinement strategies incorporating p–refinement all lead to exponential convergence of $J_{C_d}(\mathbf{u}_h)$ to $J_{C_d}(\mathbf{u})$. Figures 7(b) & (c) show the resultant hp–mesh distribution when employing anisotropic hp–refinement after 5 adaptive steps; here, Figures 7(b) & (c) show the (approximate) polynomial degrees employed in the x– and y–directions, respectively. We observe that anisotropic h–refinement has been employed in order to resolve the boundary layer and anisotropic p-refinement has been utilized further inside the computational

domain. In particular, we notice that the polynomial degrees have been increased to a higher level in the orthogonal direction to the curved geometry, as we would expect.

Acknowledgements. The authors acknowledge the support of the EU under the ADIGMA project.

References

1. Bassi, F., Rebay, S.: A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. J. Comp. Phys. 131, 267–279 (1997)
2. Baumann, C., Oden, J.: A discontinuous hp finite element method for the Euler and Navier-Stokes equations. International Journal for Numerical Methods in Fluids 31, 79–95 (1999)
3. Becker, R., Rannacher, R.: An optimal control approach to a posteriori error estimation in finite element methods. Acta Numerica 10, 1–102 (2001)
4. Eriksson, K., Estep, D., Hansbo, P., Johnson, C.: Introduction to adaptive methods for differential equations. In: Iserles, A. (ed.) Acta Numerica, pp. 105–158. Cambridge University Press, Cambridge (1995)
5. Fidkowski, K.J., Darmofal, D.L.: A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations. J. Comput. Physics 225, 1653–1672 (2007)
6. Georgoulis, E., Hall, E., Houston, P.: Discontinuous Galerkin methods for advection–diffusion–reaction problems on anisotropically refined meshes. SIAM J. Sci. Comput. 30(1), 246–271 (2007)
7. Georgoulis, E., Hall, E., Houston, P.: Discontinuous Galerkin methods on hp–anisotropic meshes II: A posteriori error analysis and adaptivity. Appl. Numer. Math. 59(9), 2179–2194 (2009)
8. Hartmann, R., Houston, P.: Adaptive discontinuous Galerkin finite element methods for the compressible Euler equations. J. Comput. Phys. 183(2), 508–532 (2002)
9. Hartmann, R., Houston, P.: Symmetric interior penalty DG methods for the compressible Navier–Stokes equations I: Method formulation. Int. J. Num. Anal. Model. 3(1), 1–20 (2006)
10. Hartmann, R., Houston, P.: Symmetric interior penalty DG methods for the compressible Navier–Stokes equations II: Goal–oriented a posteriori error estimation. Int. J. Num. Anal. Model. 3(2), 141–162 (2006)
11. Hartmann, R., Houston, P.: An optimal order interior penalty discontinuous Galerkin discretization of the compressible Navier–Stokes equations. J. Comput. Phys. 227(22), 9670–9685 (2008)
12. Houston, P., Mackenzie, J., Süli, E., Warnecke, G.: A posteriori error analysis for numerical approximations of Friedrichs systems. Numerische Mathematik 82, 433–470 (1999)
13. Houston, P., Süli, E.: A note on the design of hp–adaptive finite element methods for elliptic partial differential equations. Comput. Methods Appl. Mech. Engrg. 194(2-5), 229–243 (2005)
14. van der Vegt, J., van der Ven, H.: Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows, I. General formulation. J. Comp. Phys. 182, 546–585 (2002)

This page intentionally left blank

Chapter 29
Treatment of the Non-polygonal Boundary with the Aid of NURBS

Vít Dolejší

Abstract. Within this chapter we deal with the description of the nonpolygonal boundaries of 3D domains which exhibits an important tool for a mesh adaptation and a higher order discretization in CFD. We define our requirements, namely the evaluation of the first and second order derivatives and the (pseudo)inverse mapping. Furthermore, we show that an extension of the approach used in 2D leads to enormous memory requirements. Hence, we propose the boundary description based on a NURBS representation. A detailed comparison of both approaches is presented.

1 Introduction

The reatment of non-polygonal boundaries is an important and complex issue in computational fluid dynamics. The most shape of wings, plains etc. are not described by simple analytical functions and hence, there arises a necessity to describe boundaries of computational domains by a sufficiently efficient and accurate way. Such treatment of non polygonal boundaries is employed is higher order methods as well as in a mesh adaptation. The aim is to describe the boundary shape by a finite number of patches in such a way that each patch is a results of a mapping of a reference rectangle, i.e.,

$$(u,v) \to \mathbf{f}(u,v) \in \boldsymbol{R}^3, \qquad (u,v) \in (0,T_1) \times (0,T_2), \tag{1}$$

where

$$\mathbf{f}(u,v) = (f_1(u,v), f_2(u,v), f_3(u,v)), \tag{2}$$

see Figure 1.

Mesh adaptation algorithms and higher order discretizations handling with non-polygonal boundaries require

Vít Dolejší
Charles University Prague, Faculty of Mathematics and Physics,
Sokolovská 83, Prague, Czech Republic
e-mail: dolejsi@karlin.mff.cuni.cz

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 413–425, 2010.
springerlink.com

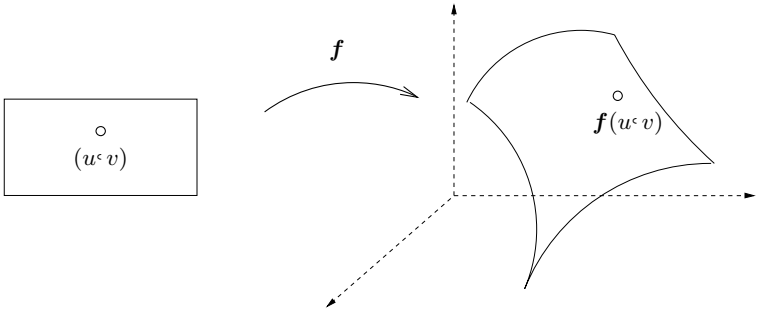


Fig. 1 Example of a mapping of the reference rectangle to a non-polygonal patch

- evaluation of the function **f** for each $(u,v) \in (0,T_1) \times (0,T_2)$,
- evaluation of the first order derivatives of **f**, i.e.,

$$\frac{\partial f_i}{\partial u}(u,v),\ \frac{\partial f_i}{\partial u}(u,v), \quad (u,v) \in (0,T_1) \times (0,T_2),\ i = 1,2,3, \tag{3}$$

- evaluation of the second order derivatives of **f**, i.e.,

$$\frac{\partial^2 f_i}{\partial u^2}(u,v),\ \frac{\partial^2 f_i}{\partial u \partial v}(u,v),\ \frac{\partial^2 f_i}{\partial v^2}(u,v), \quad (u,v) \in (0,T_1) \times (0,T_2),\ i = 1,2,3 \tag{4}$$

- evaluation of the "inverse mapping"

$$(u,v) = \mathbf{f}^{-1}(x_1,x_2,x_3), \quad (x_1,x_2,x_3) \in \{\mathbf{f}(u,v);\ (u,v) \in (0,T_1) \times (0,T_2)\}. \tag{5}$$

Here, we deal with two possible treatments of non-polygonal boundaries:

i) dense nodes representation (Section 2),
ii) NURBS representation (Section 3),

The first approach represents in fact a direct generalization of the approach developed in [1], [2] for the anisotropic mesh adaptation method for 2D domains. However, the studies and observations presented in Section 4 shows that this approach can not be applied with success to 3D problems. Hence, we develop the second technique which represents a more sofisticated approach which often used in engineering for description of nonpolygonal boundaries. We compare both approaches from the point of view of memory requirements and CPU time in Section 4.

2 Dense Nodes Representation

This approach represents in fact a direct generalization of approach developed in [1], [2] for the anisotropic mesh adaptation method for 2D domains. There, the nonpolygonal boundaries were approximated by a piecewise linear line over a large set of nodes lying on the boundary. Then the mesh adaptation was carried out with

the aid of this finite set. Hence, it was a semi-discrete representation. This approach can be directly extended to 3D in the following way.

2.1 Dense Nodes Definition

Let us assume that the computational domain $\Omega \subset \boldsymbol{R}^3$ has the boundary $\partial\Omega$ which consists of a finite number of surfaces $Q_k,\ k=1,\ldots,N$. These surfaces are images of mappings $F_k,\ k=1,\ldots,N$, of either reference square $\hat{S}=[0,1]\times[0,1]$ or a reference triangle $\hat{T}=\{(x,y),\ 0\le y\le 1-x,\ 0\le x\le 1\}$, i.e.

$$\partial\Omega=\cup_{k=1}^{N} Q_k, \qquad Q_K=F_k(\hat{T}) \text{ or } Q_K=F_k(\hat{S}), \tag{6}$$

where $F_k:\hat{T}\to\boldsymbol{R}^3$ or $F_k:\hat{T}\to\boldsymbol{R}^3,\ k=1,\ldots,N$. For the simplicity we will consider only Q_k which are images of reference squares.

Let M is a given (sufficiently large) integer number, we define a set of Lagrangian nodes of the reference element by

$$\{\hat{P}^{i,j};\ \hat{P}^{i,j}=(i/M,j/M),\ i,j=1,\ldots,M\}\in\hat{S}. \tag{7}$$

Then, for each surface Q_k we define the set of images of reference nodes by

$$\mathcal{N}_k:=\{P_k^{i,j}=F_k(\hat{P}^{i,j}),\quad i,j=1,\ldots,M_k\},\quad k=1,\ldots,N. \tag{8}$$

(It is possible consider different spacing with respect to x and y coordinates and also different M for each surface Q_k.)

Finally, we put

$$\mathcal{N}:=\{P_k^{i,j};\ P_k^{i,j},\ i,j=1,\ldots,M_k,\ k=1,\ldots,N\}. \tag{9}$$

Obviously, $\mathcal{N}\subset\partial\Omega$. We call $\mathcal{N}$ the *dense nodes* set and it represents a dense nodes representation of $\partial\Omega$.

2.2 Evaluation of the Derivatives

The functions $F_k,\ k=1,\ldots,N$, which map a reference element to the surfaces $Q_k,\ k=1,\ldots,N$ are not defined for any point of reference element but only for the finite (but large) number of nodes $\hat{P}^{i,j},\ i,j=1,\ldots,N$. Therefore, it is necessary to evaluate the first and second order derivatives with the aid of finite differences very well-known from the finite difference methods, i.e,

$$\begin{aligned}
\frac{\partial F_k}{\partial u}(\hat{P}^{i,j}) &\approx \frac{F_k(\hat{P}^{i+\delta,j})-F_k(\hat{P}^{i,j})}{\delta/M},\\
\frac{\partial F_k}{\partial v}(\hat{P}^{i,j}) &\approx \frac{F_k(\hat{P}^{i,j+\delta})-F_k(\hat{P}^{i,j})}{\delta/M},
\end{aligned} \tag{10}$$

$$\frac{\partial_k^F}{\partial u^2}(\hat{P}^{i,j}) \approx \frac{F_k(\hat{P}^{i-\delta,j}) - 2F_k(\hat{P}^{i,j}) + F_k(\hat{P}^{i+\delta,j})}{(\delta/M)^2},$$
$$\frac{\partial_k^F}{\partial v^2}(\hat{P}^{i,j}) \approx \frac{F_k(\hat{P}^{i,j-\delta}) - 2F_k(\hat{P}^{i,j}) + F_k(\hat{P}^{i,j+\delta})}{(\delta/M)^2},$$
$$\frac{\partial_k^F}{\partial u \partial v}(\hat{P}^{i,j}) \approx \frac{F_k(\hat{P}^{i+\delta,j+\delta}) - F_k(\hat{P}^{i-\delta,j+\delta}) - F_k(\hat{P}^{i+\delta,j-\delta}) + F_k(\hat{P}^{i-\delta,j-\delta})}{4(\delta/M)^2},$$

where δ is a suitable chosen nonzero integer number. The first order derivatives are approximated with the aid of forward (for $\delta > 0$) or backward ($\delta < 0$) difference formulae and the second order derivatives with the aid of central formulae. In case that we evaluate derivatives in $\hat{P}^{i,j}$ for i or j equal to 0 or N it is necessary to avoid a use of the central formulae. Smaller δ gives smaller discretization error but for very large N it can cause an increase of rounding errors. Therefore, in order to have a sufficiently robust evaluation of the derivatives, it is necessary to carried out several computations for different δ.

Hence, let $\partial_{\delta_l} F,\ l = 1,\ldots,L$ denote formally an approximation of an derivative given by (10) for a sequence of L different choices of δ such that $\delta_1 < \delta_2 < \ldots < \delta_L$. In practical implementations, it is natural to put $\delta_l = l,\ l = 1,\ldots,L$. Then we say that a sequence

$$\Delta\delta_l := |\partial_{\delta_l} F - \partial_{\delta_l+1} F|,\ l = L_0,\ldots,L_1,\ 1 \le L_0 < L_1 \le L-1 \tag{11}$$

is L_0, L_1 *reliable* if $\Delta\delta_l \le \Delta\delta_{l+1},\ l = L_0,\ldots,L_1$. The reliability means that the approximations of derivatives $\partial_{\delta_l} F,\ l = 1,\ldots,L$ numerically converge to a limit value. Then the resulting approximation of derivative is taken the value (10) obtained with δ_{L_0} where L_0 is the minimal value for which the sequence (11) is $L_0, L_0 + r$ reliable. The r is user defined value, e.g., $r = 3$.

The evaluation of the derivatives is not to much accurate. However, in the mesh adaptation algorithm, it is sufficient to have a reasonable approximation of these derivatives, hence the presented approach is sufficient.

2.3 Evaluation of the Inverse Mapping

Let $R \in \boldsymbol{R}^3$ be a node in a vicinity of surface Q_k. Our aim is to find a node $\hat{R}$ from an reference element such that $F_k(\hat{R}) = R$. If $R \notin Q_k$ then such $\hat{R}$ does not exist, hence the task should be taken from a point of some extrapolation.

In the case of the dense nodes representation it is natural (for a given R) to find the closest possible node $P_k^{i,j}$ from $\mathcal{N}$. Very naive technique (but always convergent) is to find the closest node by checking all nodes from $\mathcal{N}$.

The more efficient approach is a 2D generalization of the *half-splitting method*. In practical applications, the node R lies in a triangular or rectangular surface of a tetrahedral, hexahedral, etc. element.

Without any restriction we assume, that for a given R is a node lying in a triangle defined by a triple of nodes (T_1, T_2, T_3) lying on Q_k. Let $\hat{T}_i,\ i = 1,2,3$ be nodes of $\hat{T}$

such that $F_k(\hat{T}_i) = T_i,\ i = 1,2,3$. This assumption is fulfilled in practice since node R is usually a node of a given triangle surface. We describe the algorithm for a case when the dense node set $\mathscr{N}$ is infinitely dense. Then we introduce some remarks concerning implementation for finite $\mathscr{N}$.

Algorithm

1. let $s = 0$ and initiate $T_i^s = T_i,\ \hat{T}_i^s = \hat{T}_i,\ i = 1,2,3$
2. look up $i_s = 1,2,3$ such that

$$|T_{i_s}^s - R| \geq |T_i^s - R|, \qquad i = 1,2,3$$

3. put

$$\hat{T}_i^{s+1} = \begin{cases} \hat{T}_i^s & \text{if } i \neq i_s \\ (\hat{T}_1^s + \hat{T}_2^s + \hat{T}_3^s)/3 & \text{if } i = i_s \end{cases}, \qquad i = 1,2,3$$

4. **if** triangle $(\hat{T}_1^{s+1}, \hat{T}_1^{s+1}, \hat{T}_1^{s+1})$ is sufficiently small (i.e, between them there is only a few nodes from $\mathscr{T}$) **then** we stop the iteration **else**
   a. $T_i^{s+1} = F_k(\hat{T}_i^{s+1}),\ i = 1,2,3$
   b. put $s := s + 1$
   c. goto step 2.

Finally, we look up the node closest to R by checking all (a few) nodes from reference element lying within element $(\hat{T}_1^s, \hat{T}_1^s, \hat{T}_1^s)$. Figure 2 illustrates this algorithm. For finite set $\mathscr{N}$ it is necessary to slightly modify the second part of step 3 in order to ensure that $T_i^{s+1} \in \mathscr{N}$ and R lies within triangle $(T_1^{s+1}, T_1^{s+1}, T_1^{s+1})$.

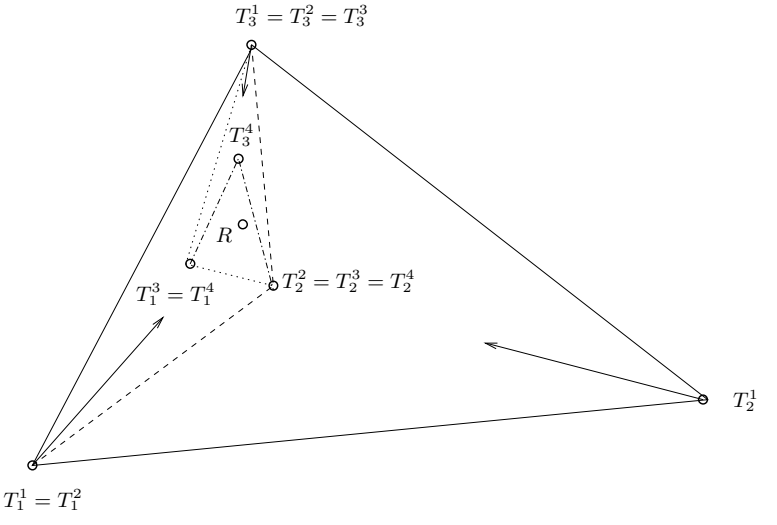


Fig. 2 Illustration of the evaluation of inverse mapping for dense nodes representation

3 NURBS

The so-called Non-Uniform Rational B-Splines, commonly referred as NURBS (see, e.g., [3]), have become in fact the industry standard for the representation, design, and data exchange of geometric information processing by computers. The enormous success behind NURBS is largely due the fact that

- NURBS provide a unified mathematical basis for representing both analytic shapes, such as conic sections and quadratic surfaces, as well as free-form entities such as car bodies and ship hulls;
- designing with NURBS is intuitive; almost every tool and algorithm has an easy-to-understand geometric interpretation;
- NURBS algorithms are fast and numerically stable;
- NURBS curves and surfaces are invariant under common geometric transformations, such as translation, rotation, parallel and perspective projections;
- NURBS are generalizations of non-rational B-splines and rational and non-rational Bézier curves and surfaces.

3.1 Definition of NURBS Surfaces

3.1.1 B-Spline

Let $U = \{u_0, \ldots, u_k\}$ be a non-decreasing sequence of real numbers. Each of them is called *knot* and U is the *knot vector*. The i-th B-spline basis function of p-degree denoted by $N_{i,p}(u)$ is defined recursively as follows

$$N_{i,0}(u) = \begin{cases} 1 \text{ if } \quad u_i \le u \le u_{i+1} \\ 0 \quad \text{otherwise} \end{cases}, \qquad i = 0, \ldots, k-1 \tag{12}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u), \quad i = 0, \ldots, k-p-1,$$

with the convention $0/0 \equiv 0$. These basis functions are key to define a NURBS surface. It is possible to derive the recursive formulae for basis function derivative

$$N'_{i,p}(u) = \frac{p}{u_{i+p} - u_i} N_{i,p-1}(u) - \frac{p}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u). \tag{13}$$

3.1.2 NURBS Surface

Let be given:

- $p \ge 1$ and $q \ge 1$ the degrees in the "u-direction" and the "v-direction", respectively,
- $n \ge p+1$ and $m \ge q+1$ corresponding to the number of used basic functions in the "u-direction" and the "v-direction", respectively,
- the real numbers:

$$0 < u_{p+1} \le u_{p+2} \le \ldots \le u_n < 1, \qquad (14)$$
$$0 < v_{q+1} \le v_{q+2} \le \ldots \le v_m < 1,$$

- the bidirectional control net

$$\{P_{i,j}, P_{i,j} \in \boldsymbol{R}^3,\ i = 0,\ldots,n, j = 0,\ldots,m\} \qquad (15)$$

- weights $\{w_{i,j}, w_{i,j} \in \boldsymbol{R}, i = 0,\ldots,n, j = 0,\ldots,m\}$

Using values (14) we define the knot vectors U and V by

$$U = \{\overbrace{0,\ldots,0}^{p+1}, u_{p+1},\ldots,u_n, \overbrace{1,\ldots,1}^{p+1}\} \in \boldsymbol{R}^{n+p+2}, \qquad (16)$$
$$V = \{\underbrace{0,\ldots,0}_{q+1}, v_{q+1},\ldots,v_m, \underbrace{1,\ldots,1}_{q+1}\} \in \boldsymbol{R}^{m+q+2},$$

over which we construct the corresponding B-spline basis functions of orders p and q

$$\{N_{i,p}(u), N_{i,p}(u) : (0,1) \to \boldsymbol{R},\ i = 0,\ldots,n\}, \qquad (17)$$
$$\{N_{j,q}(u), N_{j,q}(u) : (0,1) \to \boldsymbol{R},\ j = 0,\ldots,m\}$$

using (12).

Then we define a mapping $S(u,v) : (0,1) \times (0,1) \to \boldsymbol{R}^3$ by

$$S(u,v) \equiv \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}P_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \quad 0 \le u,v \le 1, \qquad (18)$$

which is called a *NURBS surface*.

3.2 Evaluation of the Derivatives

The direct evaluation of the (first and second order) derivatives of (18) is complicated. Therefore, we employ the so-called *homogeneous coordinates*. We represent three dimensional control net points $P_{i,j} = (x_{i,j}, y_{i,j}, z_{i,j})^{\mathrm{T}} \in \boldsymbol{R}^3$ with their weights $w_{i,j}$ by four-dimensional points $P^w_{i,j} = (w_{i,j}x_{i,j}, w_{i,j}y_{i,j}, w_{i,j}z_{i,j}, w_{i,j})^{\mathrm{T}} \in \boldsymbol{R}^4$. We introduce an equivalent definition of a NURBS surface. We define a mapping $A(u,v) : (0,1) \times (0,1) \to \boldsymbol{R}^4$ by

$$A(u,v) \equiv \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)P^w_{i,j}. \qquad (19)$$

Assembling the components of $A(u,v)$ by

$$\begin{aligned} A(u,v) &= (A(u,v)|_{3d}, A(u,v)|_{\omega})^{\mathrm{T}} \\ &= (A(u,v)|_{3d}, \omega(u,v))^{\mathrm{T}}, \qquad A(u,v)|_{3d} \in \boldsymbol{R}^3,\ \omega(u,v) \in \boldsymbol{R}, \end{aligned} \qquad (20)$$

the mappings $S(u,v)$ and $A(u,v)$ satisfy the relation

$$A(u,v)|_{3d} = \omega(u,v)\,S(u,v) \qquad \forall u,v \in (0,1)\times(0,1). \tag{21}$$

Now, we evaluate the partial derivatives of $A(u,v)$ with the aid of (13). By a simple computation we get

$$A^{(k,l)} \equiv \frac{\partial^{k+l} A(u,v)}{\partial^k u\,\partial^l v} = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}^{(k)}(u) N_{j,q}^{(l)}(v) P_{i,j}^{w}. \tag{22}$$

Finally, we evaluate the derivatives of $S(u,v)$. From (21) we derive by parts

$$A^{(k,l)}|_{3d} = (\omega S)^{(k,l)} = \left(\sum_{i=0}^{k}\binom{k}{i}\omega^{(i,0)} S^{(k-i,0)}\right)^{(l)} = \sum_{i=0}^{k}\binom{k}{i}\sum_{j=0}^{l}\binom{l}{j}\omega^{(i,j)} S^{(k-i,l-j)}. \tag{23}$$

By an elimination of (23), we derive the recurrent formulae

$$\begin{aligned}\omega S^{(k,l)} &= A^{(k,l)}|_{3d} - \sum_{i=1}^{k}\binom{k}{i}\omega^{(i,0)} S^{(k-i,l)} \\ &\quad - \sum_{j=1}^{l}\binom{l}{j}\omega^{(0,j)} S^{(k,l-j)} - \sum_{i=1}^{k}\binom{k}{i}\sum_{j=1}^{l}\binom{l}{j}\omega^{(i,j)} S^{(k-i,l-j)},\ k,l = 0,1,\ldots.\end{aligned} \tag{24}$$

Expressions (24) is used for the implementation exactly in the same manner as it was derived. At first, the homogeneous derivatives $A^{(k,l)}$ are computed by (22) with a recursive use of (13). This step gives us also all the necessary values of $w^{(i,j)} = A^{(i,j)}|_w$. Finally, with the aid of (24), we evaluate all desired derivatives of $S(u,v)$. For more details see [3].

3.3 Evaluation of the Inverse Mapping

Given a point P, assumed to lie on the NURBS surface $S(u,v)$, *point inversion* is the problem of finding the corresponding parameters $\tilde{u},\tilde{v}$ such that $S(\tilde{u},\tilde{v}) = P$. For numerical reasons, it is better to solve this problem as *point projection*. One can not expect that point P lies *precisely* on surface $S(u,v)$. Therefore, we look for the nearest point on the surface, i.e., point which minimises distance to P. In other words, we minimise length of vector $\mathbf{r}$ defined by

$$\mathbf{r} = r(u,v) = S(u,v) - P. \tag{25}$$

In what follows, we strictly distinguish between scalar product $\mathbf{a}\cdot\mathbf{b}$ and tensor product $\mathbf{a}\,\mathbf{b}$ of two vectors. The necessary condition for minimiser of (25) reads

$$g(\tilde{u},\tilde{v}) = \frac{\nabla(\mathbf{r}\cdot\mathbf{r})}{2} = \mathbf{r}(\tilde{u},\tilde{v})\cdot\nabla S(\tilde{u},\tilde{v}) = 0, \tag{26}$$

where

$$\nabla S(u,v) = (S'_u, S'_v)^{\mathrm{T}} = \left(\frac{\partial}{\partial u}S(u,v), \frac{\partial}{\partial v}S(u,v)\right)^{\mathrm{T}}. \tag{27}$$

We solve (26) by the Newton's method. It should converge well since the distance $|\mathbf{r}|$ is expected to be small (point P lies in the vicinity of NURBS surface). Let (u_i, v_i) be i-th approximation generated by the Newton's method. Obviously, the convergence means that $(u_i, v_i) \to (\tilde{u}, \tilde{v})$ for $i \to \infty$.

The Newton iteration process reads

$$(u_{i+1}, v_{i+1}) = (u_i, v_i) + (\delta u, \delta v), \tag{28}$$

where the difference $\delta = (\delta_u(u,v), \delta_v(u,v))$ is solution of linear system

$$\nabla g(u_i, v_i)\cdot\delta = -g(u_i, v_i). \tag{29}$$

For our specific choice of g (26), we get the following equations

$$[\nabla S\nabla S + \mathbf{r}\cdot\nabla\nabla S]\,\delta = -\mathbf{r}\cdot\nabla S. \tag{30}$$

All the involved functions here are evaluated in point (u_i, v_i). As we expected, $|\mathbf{r}|$ is rather small. It encourage us to omit corresponding term on the left hand side, hence

$$[\nabla S\nabla S]\,\delta = -\mathbf{r}\cdot\nabla S, \tag{31}$$

which means

$$\begin{bmatrix} S'_u(u_i,v_i)\cdot S'_u(u_i,v_i) & S'_u(u_i,v_i)\cdot S'_v(u_i,v_i) \\ S'_v(u_i,v_i)\cdot S'_u(u_i,v_i) & S'_v(u_i,v_i)\cdot S'_v(u_i,v_i) \end{bmatrix} \begin{bmatrix} \delta_u \\ \delta_v \end{bmatrix} = \begin{bmatrix} (P - S(u_i,v_i))\cdot S'_u(u_i,v_i) \\ (P - S(u_i,v_i))\cdot S'_v(u_i,v_i) \end{bmatrix}. \tag{32}$$

Sequential solution of this iteration step along with verification of intuitive stopping rules is the heart of the method. The initial guess $u_0, v_0)$ of the iterative process is taken randomly.

4 Comparison of Both Representations

Within this section we compare the computational performance of the both non-polygonal boundary representations presented above (dense nodes and NURBS), namely their memory and CPU requirements.

An exact comparison of boundary representations from the point of view of memory and CPU is very difficult task since it depends on the complexity of considered problem and the accuracy of the approximation of boundary representation. Therefore, we will try estimate these requirement only qualitatively. We will considered a simple test case whose geometry is similar to BTC0 test case of the ADIGMA

project. Let us consider a viscous compressible flow around a closed airfoil Γ with length $l_\Gamma = 1$ and an average intersection 0.008 (which correspond to a circle with diameter 0.1). Then the surface of Γ is $|\Gamma| = 0.008$.

Now, we try to estimate a number of triangle on the surface of profiles which should be used in order to obtain sufficiently accurate compressible flow simulation with the aid of discontinuous Galerkin method (DGM). The typical Reynolds number Re is about $10^6 - 10^7$ therefore the grid size in a direction perpendicular to the profile is $h_Y \approx c/\sqrt{Re} \approx 10^{-4}$ (c is a small positive number). Let us assume that DGM can treat with elements with aspect ratio 1:10, then the grid size in a direction parallel with the profile is $h_x \approx 10^{-3}$. Hence the number of surface elements on Γ can be estimated as

$$N_\Gamma \approx \frac{|\Gamma|}{h_x^2} \approx 8\,000. \tag{33}$$

Both considered boundary representations (dense nodes and NURBS) requires a division of the airfoil Γ into several simpler surfaces $Q_k,\ k = 1, N$ which will be images of reference elements. We estimate that simple closed airfoil can consist, e.g., of $N = 15$ simple surfaces. Then the number of surface elements for one surface is

$$N_{Q_k} = \frac{N_\Gamma}{N} \approx 500. \tag{34}$$

In the following we discuss the memory and CPU requirements of both boundary representations and then we compare both approaches.

4.1 Requirements of Dense Nodes

4.1.1 Memory Requirements

Based on our experiences from 2D computations with in-house code ANGENER [1] we expect that in order to generate sufficiently "smooth" grids the set $\mathcal{N}_k$ introduced in (9) should contain at least 100 nodes corresponding to one real triangle lying on Q_k, i.e, using (34), $\mathcal{N}_k$ should contains $M_k = 50\,000$ nodes. Each node form $\mathcal{N}_k$ is stores in memory by 3 real and (at least) 1 integer number.

4.1.2 CPU for Derivative Evaluation

The first and second order derivative are evaluated with the aid of finite differences introduced in (10). The first order derivatives need 4 operations (addition or multiplication) and the second order derivatives need 7 operations. We estimate that in order to find the "minimal"reliable difference (11) we need to evaluate each derivative 5 times. Moreover, the founding of the minimum (several repetition of `if then else` cycles needs about 5 times more operation than the evaluation itself. Then we estimate 100 operations for the first order deivative and 140 operations for the second order derivatives.

4.1.3 CPU for Inverse Mapping Evaluation

Evaluation of the inverse mapping, i.e., the founding of the closest node from $\mathcal{N}_k$ to a given node defined in Section 2.3 needs at most (in optimal case) $\log M/\log 2$ iterations, where M is the number of possible nodes from $\mathcal{N}_k$. Hence, the upper estimate is $M = M_k = 50\,000$ and the bellow estimate is $M = 100$ which corresponds to the case when the we know a good initial approximation of the solution. Hence we consider for simplicity $M = 1\,000$.

The number of operations for each step of the algorithm from Section 2.3 is given by 21 in 2nd step, 16 in 3rd step and approximately 20 in 4th step, which is totaly 57 operations. Then the total number of operations can be estimated by $57\log M/\log 2 \approx 570$.

4.2 Requirements of NURBS

4.2.1 Memory Requirements

Without a lost of accuracy we assume that for each surface $Q_k,\ k = 1,\ldots,N$ we have one $(p,q),(n,m)$ NURBS surface, see Section 3.1.2. Then we have to store $(n+1)(m+1)$ nodes of control net, weights and knot vectors, i.e., $4(n+1)(m+1)+(m-p)+(n-q)$ real numbers. In practice n,m are relatively small integer numbers, at most, let us say, 5 or 10. The integer numbers p,q are still smaller, at most, let us say 3 or 5.

4.2.2 CPU for Derivative Evaluation

Each derivative $N'_{i,p}(u)$ according (13) needs $7np$ operations. Then evaluation of $A^{(k,l)}$ according (22) needs approximately $op = 3(7np+7mq+3nm)$ operations. Finally, we estimate the number of operations needed for evaluation of $S^{(k,l)}$ according (24), which gives op operations for $S^{(0,0)}$, $2op$ operations for $S^{(1,1)}, S^{(0,1)}$ and $4op$ operations for $S^{(2,0)}, S^{(2,1)}, S^{(0,2)}$.

4.2.3 CPU for Inverse Mapping Evaluation

The inverse mapping is evaluated with the aid of the Newton method which practically converges in 3 – 5 steps if we have a good initial quest. The initialization of matrix problem (32) needs two evaluation of the first order derivatives, i.e., $4op$ operations (see previous section) and 8 additional operations. Moreover, the solution of linear system needs 14 operations and update (30) 2 operations. Summing, we have $4(4op+24)$ operations.

4.3 Comparison of Dense Nodes and NURBS

Within this section we compare the memory and CPU requirements of dense nodes and NURBS boundary representations. We assume that the airfoil Γ consists of 15

surface Q_k, $k = 1, \ldots, 15$ and we use the cubic NURBS with $p = q = 3$ and $N = M = 5$. Table 1 show the comparison of the memory and computational requirement for both approaches based on the previous analysis.

Table 1 Comparison of memory and computational requirements for the dense nodes and NURBS representations

	dense nodes	NURBS	(units)
memory	2 250 000	2 020	(reals stores)
$f(u,v)$	3	1 080	operations
$\partial^1 f(u,v)$	100	2 160	operations
$\partial^2 f(u,v)$	140	4 320	operations
$f^{-1}(x,y,z)$	570	17 376	operations

The results presented in Table 1 do not take into account memory manipulations which is more time consuming for the dense nodes representations since it requires significantly more memory. Hence, Table 2 shows the comparison of real computational times for evaluating of the zero and first order derivatives in more than 60 000 nodes, which were carried out by a simple test calculation. It shows that the difference of the real computational time between both approaches is not so drastic as in the number of operations.

Table 2 Comparison of real computational times for the dense nodes and NURBS representations

	dense nodes	NURBS
$f(u,v)$	0.008 s	0.296 s
$\partial^1 f(u,v)$	0.128 s	0.593 s

5 Conclusion

Based on the previous considerations and the numerical experiments, we conclude that

- evaluation of the derivatives and inverse mapping is several times faster with the aid of dense nodes boundary representations
- memory requirements for NURBS representations are many times smaller

Based on practical use where the mesh adaptation process takes a minor part of the total computational time in comparison with itself solution of the Navier-Stokes equations (see [2]) we prefer the significant save of memory than a small decrease of the computational time. Therefore, the proposed NURBS representation seems to be better technique for the treatment of nonpolygonal boundaries.

References

1. Dolejší, V.: ANGENER. Charles University Prague, Faculty of Mathematics and Physics, version 3.0 edition (2000), http://www.karlin.mff.cuni.cz/dolejsi/angen.html
2. Dolejší, V., Felcman, J.: Anisotropic mesh adaptation and its application for scalar diffusion equations. Numer. Methods Partial Differ. Equations 20, 576–608 (2004)
3. Piegl, L., Tiller, W.: The NURBS Book. Springer, Heidelberg (1997)

This page intentionally left blank

Chapter 30
HP-Adaption in Space-Time within an Explicit Discontinuous Galerkin Framework

Arne Taube, Gregor Gassner, and Claus-Dieter Munz

Abstract. Based on an explicit discontinuous Galerkin scheme for the compressible Navier-Stokes equations we describe an adaptation framework which consists of two building blocks. First, adaptation in time due to time accurate local time stepping and second, adaptation in space by mesh refinement and increase of the local polynomial order in the element during runtime based on a feature based cell resolution indicator – *(h-, p- and hp-adaption)*. At the end we show simulations of the unsteady laminar flow over a NACA0012 airfoil.

1 Introduction

The locality of our DG formulation allows for a drastic change in the usual time advancement by doing adaptation in time. Hence, our scheme features local time stepping and based on that, an adaptive approach allows for different discretizations to be used in different subregions depending on the local flow behavior. High-order accuracy on coarse grids may be applied in some subregions, while in other regions a fine grid in combination with lower order may be better. We also employ high order accuracy in order to capture strong gradients on relatively coarse grid cells.

The detection of under resolved regions is based on cell resolution indicators is described in Sect. 3. It is followed by an overview about our current adaptation strategy describing the three different means of adjustment – *(h-, p- and hp-adaptation)* – in Sect. 4. In Section 5, we show a numerical example of the adaptation strategy to the ADIGMA MTC 3 test case computed in parallel, which is a subsonic viscous flow with a laminar but unsteady vortex shedding. At the end, we draw our conclusion in Sect. 6.

Arne Taube · Gregor Gassner · Claus-Dieter Munz
Institut für Aerodynamik und Gasdynamik, University of Stuttgart, Pfaffenwaldring 21, 70569 Stuttgart, Germany
e-mail: taube@iag.uni-stuttgart.de, gassner@iag.uni-stuttgart.de, munz@iag.uni-stuttgart.de

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 427–439, 2010.
springerlink.com

2 Adaptation in Time

For an explicit scheme, the maximum permissible time step may strongly vary in the computational domain. It depends on the local element size, the local order of the trial function and the local value of the approximation. Standard explicit DG schemes use one global time step oriented at the smallest element size and highest wave speed to ensure stability as described in the following.

2.1 Time Step Restriction

As an explicit scheme, the STE-DG scheme has to satisfy a time step restriction for stability. For the two-dimensional compressible Navier-Stokes equations, we use the following estimation

$$\Delta t_i \leq \frac{1}{\sqrt{\frac{1}{\Delta t_{i,\alpha}^2} + \frac{1}{\Delta t_{i,\beta}^2}}}, \tag{1}$$

with the hyperbolic $(\Delta t_{i,\alpha})$ and parabolic $\left(\Delta t_{i,\beta}\right)$ time step restriction

$$\Delta t_{i,\alpha} = \alpha \frac{\Delta x_i}{\max\limits_{Q_i}\left(|v| + \sqrt{2}|c|\right)}, \quad \Delta t_{i,\beta} = \beta \frac{\Delta x_i^2}{\sqrt{2}\max\limits_{Q_i}\left(\frac{4\mu}{3\rho}, \frac{\mu\kappa}{\rho Pr}\right)}, \tag{2}$$

and the reference length Δx_i of grid cell Q_i which is two times the minimal distance between barycenter and cell boundary. The stability numbers α and β depend on the degree p of the polynomial approximation according to (3) and the maximal α^* and β^* are given in Table 1 as they were determined in numerical calculations for the STE-DG scheme using the HLLC and dGRP flux.

Table 1 STE-DG scheme stability numbers

p	1	2	3	4	5
α^*	1.3	1.1	0.9	0.7	0.7
β^*	1.5	0.7	0.35	0.25	0.13

$$\alpha := \frac{\alpha^*}{2p+1},\ \beta := \frac{\beta^*}{(2p+1)^2}, \tag{3}$$

The parameter η, which may be considered as the penalization parameter for jumps in the calculation of the diffusion flux, depends on the time step restriction. In all our calculations we choose

$$\eta := \eta(p) := \frac{1}{\sqrt{\pi\beta(p)}} = \frac{2p+1}{\sqrt{\pi\beta^*(p)}}, \tag{4}$$

as proposed in [5] for pure diffusion.

2.2 Time Consistent Local Time Stepping

To overcome the inefficiency of a global time stepping approach, Flaherty *et al.* [1] proposed a second order time accurate local time stepping algorithm within the RKDG schemes. In [9], Lörcher and Gassner show how to make use of the approximative space-time approach and the locality of the discontinuous Galerkin discretization to introduce a *natural* arbitrary high order accurate local time stepping. This local time stepping modification can be adopted to the multi-dimensional STE-DG discretization (see Sect. 3 in chapter 5) as shown in Lörcher and Gassner [6]. It starts with the introduction of the actual *local* time t_i in the grid cell Q_i. The degrees of freedom $\hat{U}_i(t_i)$ represent the solution at the local time t_i in this grid cell.

Figure 1 shows a four step sequence with four adjacent cells starting from the same time level, but with different time steps. The considered local time stepping algorithm minimizes the total number of time steps to reach the prescribed end time.

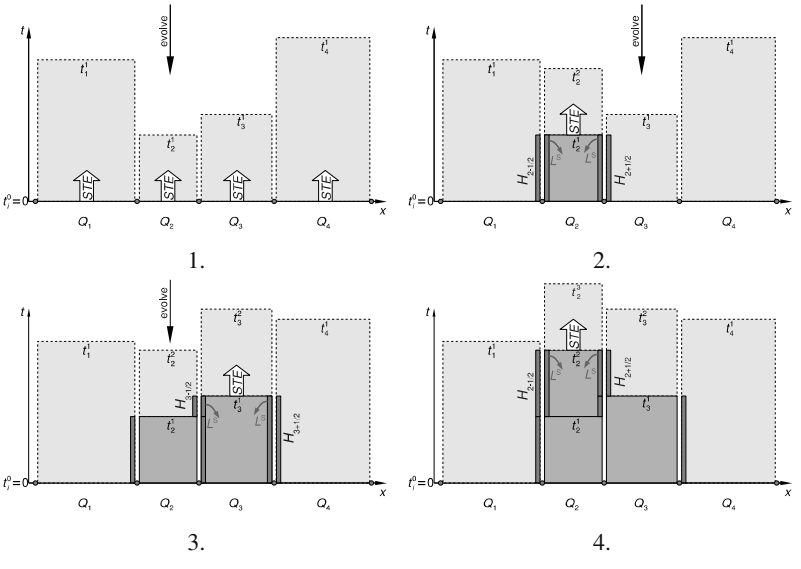


Fig. 1 Sequence of a 4 step computation with 4 different elements and local time stepping as described in Gassner *et al.* [3]

The time evolving process according to Fig. 1 takes the following steps:

1. First, the time increments Δt_i for the next step are calculated from the local stability criterion (1) for all grid cells. In addition, each grid cell's volume integral which is completely determined by the interior space-time approximate solution from the space time expansion (STE) is computed according to Sect. 3.3.1 in chapter 5 and the degrees of freedom are updated with this contribution. Cell Q_2 is detected as satisfying the so-called 'evolve condition',

$$t_i^{n+1} \leq \min\left\{t_j^{n+1}\right\}, \forall j : Q_j \cap Q_i \neq \emptyset, \tag{5}$$

and will be updated in time.

2. Q_2 is updated and the flux time integral contributions, $H_{2\pm1/2}$, are stored for the neighboring elements Q_1 and Q_3 with negative sign in order to ensure that the method is conservative. Now Q_3 can be 'evolved'.
3. In the evolution of Q_3 a part of the flux time integral has already been added and therefore only the missing contributions $H_{3\pm1/2}$ are added and stored for further use by the neighbors.
4. Then the next element to be updated is sought and the algorithm continues.

In the case where the difference of time levels of adjacent grid cells is very small, the number of flux calculations may be reduced and the efficiency of the method improved by locally synchronizing those time levels. A global synchronizing technique is used for the output at certain user-defined global time levels, as well.

3 Cell Resolution or Troubled Cell Indicators

Before adapting the mesh or employing a limiting strategy, the scheme's approximation error has to be estimated. Since an exact solution is only present within academic examples, the scheme's approximate solution itself has to give an estimation of the solution's quality. This process is called *a posteriori* error estimation. The solution behavior is different over the whole domain with well approximated regions and zones that require special treatment, because they are under-resolved or may contain singularities like shocks. Once detected, an adaptation or shock capturing/limiting strategy can be applied to those zones in order to improve the overall solution.

In order to locate such zones different indicators are applied. These can be divided into indicators that measure the solution's smoothness inside a cell or the jumps at the cell interface. The two main indicators used are described in the following, while the adaptation strategy is described in Sect. 4.

3.1 The Modified Spectral Decay Indicator

For a given local order p_i in the i^{th} element, the DG-approximation from (Eq. 12 in page 57) reads as

$$U_i^h(p_i) = \sum_{j=1}^{\mathcal{N}_i(p_i,d)} \hat{U}_{i,j} \cdot \phi_{i,j} \,. \tag{6}$$

In the case of an orthogonal hierarchical basis, one can easily calculate the differences between the approximation to the full order p_i and one order less p_{i-1}. Thus writing:

$$U_i^h(p_i) - U_i^h(p_i - 1, d) = \sum_{j=\mathcal{N}_i(p_i-1,d)+1}^{\mathcal{N}_i(p_i,d)} \hat{U}_{i,j} \cdot \phi_{i,j} =: \Delta U_i^h(p_i) \ . \tag{7}$$

This expression (7) represents the influence of the highest order moment to the solution. Analogously to a Fourier series, one may expect that,

$$\lim_{p_i \to \infty} \left(\Delta U_i^h(p_i) \right) = 0. \tag{8}$$

Taking Eq. (7) and the orthonormality of the basis functions, one can formulate the L_2-norm for the so-called spectral decay indicator,

$$\eta_i^{\mathrm{SDI}_{L_2}} = \frac{\int_{Q_i} \left(u_i^h(p_i) - u_i^h(p_i - 1) \right)^2 dx}{\int_{Q_i} \left(u_i^h(p_i) \right)^2 dx} = \frac{\sum_{j=\mathcal{N}_i(p_i-1,d)+1}^{\mathcal{N}_i(p_i,d)} (\hat{u}_{i,j})^2}{\sum_{j=1}^{\mathcal{N}_i(p_i,d)} (\hat{u}_{i,j})^2}. \tag{9}$$

This formulation does neither require function values at the Gauss points nor the basis functions, but only the i^{th} element's degrees of freedom $\hat{u}_{i,j}$. If this norm does not decay, then there should be some sort of under-resolution or shock. If the underlying exact solution is smooth, the coefficients of the approximation should decay fast. The smoothness sensor proposed in Persson and Peraire [10] is modified in the form,

$$\eta_i^{\mathrm{SDI}}(u) = \log_{10} \left\{ \max \left[\left(\frac{\sum_{j=\mathcal{N}_i(p_i-1,d)+1}^{\mathcal{N}_i(p_i,d)} (\hat{u}_{i,j})^2}{\sum_{j=1}^{\mathcal{N}_i(p_i,d)} (\hat{u}_{i,j})^2} \right), \left(\frac{\sum_{j=\mathcal{N}_i(p_i-2,d)+1}^{\mathcal{N}_i(p_i-1,d)} (\hat{u}_{i,j})^2}{\sum_{j=1}^{\mathcal{N}_i(p_i-1,d)} (\hat{u}_{i,j})^2} \right) \right] \right\}. \tag{10}$$

For $u = \rho$, $\eta_i^{\mathrm{SDI}}(\rho)$ is a measure for the rate of the decay of the magnitude of the two highest order density moments, which is further used in the cell flagging criterion (14) for the adaptation strategy described in Sect. 4.1. In contrast to the indicator by Persson and Peraire [10], Eq. (10) uses the maximum from the two highest order moments in order to address the different approximation properties of even and odd basis functions.

3.2 The Jump Indicator

The DG solution may have inter-element jumps in all cases, where the solution is not well resolved. In Krivodonova *et al.* [8], it is stated that for the jumps over the inflow boundary of the element Q_i,

$$I_i = \int_{\partial Q_i^-} \left(u_i^h - u_{i,nb}^h \right) ds \tag{11}$$

the following convergence criteria is valid:

$$I_i = \begin{cases} \mathscr{O}\left(\Delta x_i^{p_i+1} \right), & \text{if } u|_{\partial Q_i} \text{ is smooth} \\ \mathscr{O}\left(\Delta x_i \right) \quad , & \text{if } u|_{\partial Q_i} \text{ is discontinuous .} \end{cases} \tag{12}$$

From that assumption an indicator of the form

$$\eta_i^{\text{Jmp}}(u) = \frac{\left| \int_{\partial Q_i^-} \left(u_i^h - u_{i,nb}^h \right) ds \right|}{\Delta x_i^{\frac{p_i}{2}} \left| \partial Q_i^- \right| \left\| u_i^h \right\|}, \tag{13}$$

can be defined. It is based on a mean rate of convergence $\mathscr{O}(\Delta x^{P/2})$.
For regions with a smooth solution,

$$(\eta_i^{\text{Jmp}}(u) \to 0) \text{ for } \Delta x_i \to 0 \text{ or } p_i \to \infty,$$

whereas close to singularities/discontinuities it is

$$(\eta_i^{\text{Jmp}}(u) \to \infty).$$

4 Adaptation Strategies

In principle, the mesh employed has a huge impact on the computational result, its accuracy and efficiency. Using an inappropriate mesh can lead to the crash of the simulation or may produce false non-physical results or may increase CPU time unnecessarily. The flow phenomena to be investigated with a numerical simulation and the region of their occurrence may a priori be unknown or require a lot of experience from the user. Therefore, it is advisable to employ an adaptation strategy in order to automatically adjust the mesh or discretization according to the problem during the simulation run. Within our framework, we have three different means of adjustment:

- ***p*-adaptation**
  Our DG scheme features orthonormal and hierarchical polynomial basis functions with an approximation order that may be changed locally.
- ***h*-adaptation**
  The character h describes in this context a characteristic length scale within a grid cell. Thus h-adaptation means splitting a larger cell into smaller ones or merging smaller cells to a bigger one. Then the solution is projected from the old onto these newly created elements.
- ***hp*-adaptation**
  A combination of h- and p-adaptation yields hp-adaptation, where not only the elements are split or merged but the approximation order is changed as well. In such a case, the challenge is to decide, when which adaptation method is most

suitable. In our case, we usually apply p-adaptation up to a maximum order and then refine the grid.

4.1 Cell Flagging Using a Feature Based Indicator

The indicator have already been described in Sect. 3. The variable for the adaptation control is the density ρ, because – contrary to the shock capturing, where the pressure p would be advisable – the resolution around contact discontinuities shall be enhanced, as well. Then, finding an element i in an under-resolved region is guided by the following relation for its indicator value, $\eta_i(\rho)$:

$$\begin{cases} \eta_i(\rho) > hp_{\max}: & u \text{ is under resolved} & \rightarrow & \text{refine} \\ \eta_i(\rho) < hp_{\min}: & u \text{ is too smooth} & \rightarrow & \text{coarsen} . \end{cases} \tag{14}$$

The values $hp_{\max}$ and $hp_{\min}$ are user defined bounds for the adaptation indicator.

The technical terms 'refine' and 'coarsen' in (14) do not necessarily mean that in that case the mesh is adapted via h-adaptation. 'Refine' generally means that the resolution shall be increased by either increasing the order or splitting the element. The same applies for the expression 'coarsen', where the resolution may be decreased by either reducing the element's order or merging elements together. If found necessary by the above criterion, the adaptation strategy as described in the following is applied.

4.2 p-*Adaptation*

The degree of polynomial approximation, p, can easily be changed from one time step to the other. We use hierarchical and orthonormal polynomial approximation functions as described in 3.1. Hence, if according to the cell flagging criterion (14) more resolution is required, the sum in Eq. (6) is simply extended to a higher order or may be reduced in regions where less resolution may be acceptable. In that way more degrees of freedom or higher order moments are added or taken away. The mean value or conservation property of the approximation, i.e. the 1^{st} order moment, remains unchanged.

4.3 h-*Adaptation*

Compared to the previously described p-adaptation, adapting the mesh is considerably more complex. Due to our scheme's local character and the local time stepping feature, elements are as a general rule not on the same time level. This is a particular challenge to the coarsening strategy, i.e. the cell merging algorithm. However, splitting or merging elements together can lead to a much more optimized mesh. As each cell's time step depends on its size, h-adaptation may in combination with the local time steps drastically increase the performance. Furthermore, we can take into

account local gradients in order to refine the mesh in an anisotropic manner in the direction of such gradients.

Another advantage of our framework over classical CFD-implementations is the use of an 'object oriented' programming concept. The elements and their sides are stored as objects in interlinked lists. This approach enables us to easily manage newly formed or deleted elements. Classical CFD-implementations normally store the mesh information into data arrays which have to be buffered, deallocated and newly allocated when the mesh is manipulated. For our framework, however, an addition or removal of elements does not cause any problems due to the local and interlinked objects.

Nevertheless, independent of the fact if an element is split or other elements are merged together, a new Taylor series expansion (Eq. 14 in page 57) has to be performed to evaluate the surface and volume integrals.

4.3.1 Cell Splitting (Refinement)

If the indicator marks an element for refinement, its sides and their corresponding neighbor sides will be split. If each side of a triangle or quadrangle is simply split in half, then we have isotropic *h*-refinement as shown in Fig. 2 left part. If the indicator

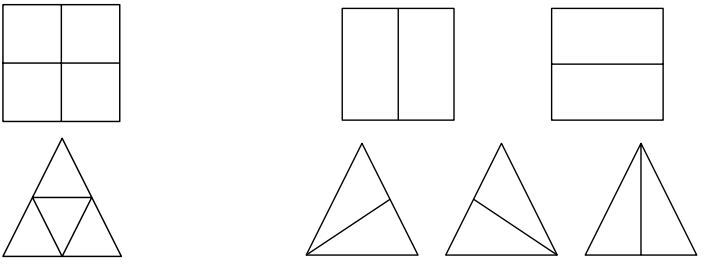


Fig. 2 Isotropic split rectangle and triangle (left) and anisotropic split rectangles and triangles (right)

is evaluated sidewise, we can perform a split only on those sides having the highest indicator value. This yields in an anisotropic split as shown in Fig. 2 right part. In a second step, the new sides are constructed. These as well as some of the old sides border the newly created elements. Finally, the solution is projected onto the new elements, the old (original) element is deleted and the so-called 'split-history' is increased. The split-history can be viewed as a family-tree of each element, which facilitates a possible cell merging at a later stage.

In our approach, every refinement stage is recorded in the split-history. With every split, the newly created elements receive a unique split index, which marks them as member of one group. Figure 3 is a schematic sketch of such a split-history. The first element is split in an isotropic way into four sub-elements with the same split index. Additionally, the split level $\left(N_{i,\text{split}}\right)$ is stored in order to reconstruct the number of splits to obtain a certain element. This may be used as a lower limit controlling

the adaptation process. In a second step, one of the newly formed elements is split in an anisotropic and another one in an isotropic manner. The resulting elements have the same split level, but different split indices.

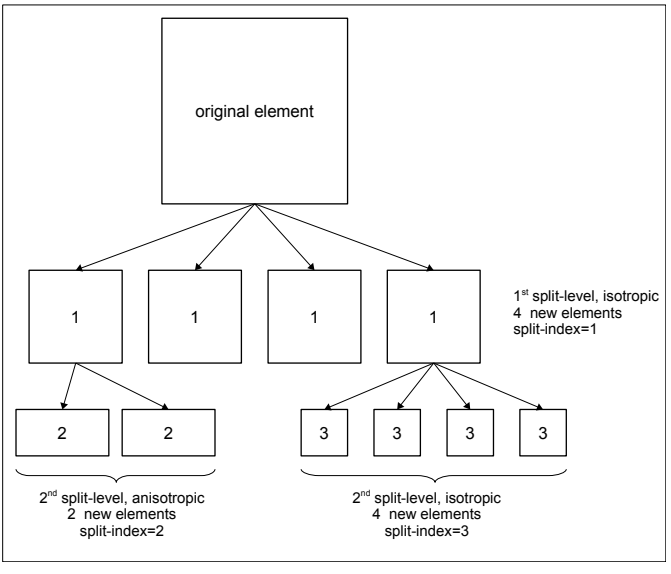


Fig. 3 An element's split-history with three refinements

4.3.2 Cell Merging (Coarsening)

For 2D problems, one could use geometrical operations to guide the coarsening or recombination process by finding neighboring elements which are tested if their roundup yields a valid element. This procedure, however, is very complex and to our mind impracticable for 3D applications. Furthermore, coarsening a once refined mesh may lead to a different mesh than at the beginning or to a deadlock in the adaptation process.

Therefore, we make use of the split-history for the coarsening process as shown in Fig. 3. When searching for elements to be joined together, only elements with the same split index will be taken. Hence, this history contains all information from previous element splits in order to be able to return in subsequent coarsening steps to the original mesh element. In fact, the coarsest mesh obtained may be the initial mesh as constructed from the mesh generator.

Having found all elements with the same split index, all the newly formed element surrounding sides are joined together. Provided that a few of the old sides form one new element side and where else it is possible, these sides as well as their connected neighboring sides are merged. In a second step, the inner sides and nodes

are deleted and the newly formed element is built. Finally, the solution from the old elements is projected onto the new one and the old elements are deleted.

4.3.3 Local Synchronization

Using local time stepping, it has to be ensured that when coarsening all elements involved are on the **same** time level. The reason for that is that the merged element has a different space time Taylor expansion and therefore surface flux integral. As a matter of fact, the neighboring elements have to be at least on the *same* or a *lower* time level than the synchronization time for the merged element.

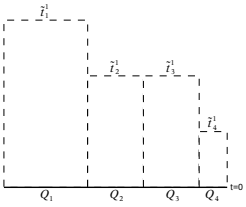


Figure 4 shows four elements Q_i being on the same time level (possibly at the start of a computation). Their space time elements $Q_i \times \left[t_i^n, \tilde{t}_i^{n+1}\right]$ are depicted with dashed lines. As the Taylor series expansion does not mark a full time step, the upper time level is named $\tilde{t}_i^{n+1}$. In the course of the simulation, elements Q_2 and Q_3 shall be united.

Fig. 4 Four elements before merging

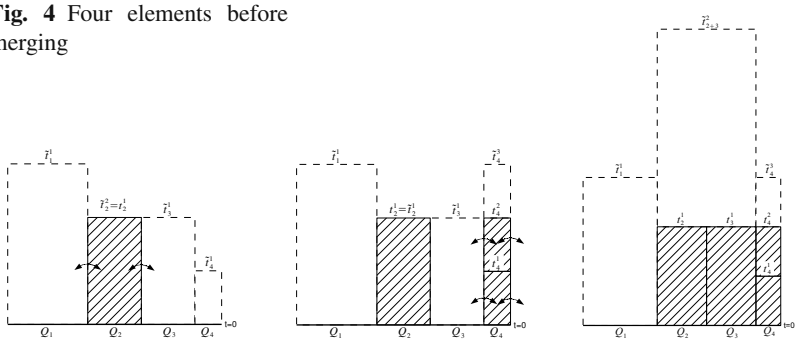


Fig. 5 Time step element Q_2 (left). Time steps element Q_4 with synchronization (middle). First time step element Q_3 with synchronization as a preparation to unite Q_2 and Q_3 (right).

According to the evolve condition (5), element Q_2 in Fig. 5 left part is the first to do its full time update by computing its surface integrals and making it available to its left and right neighbor. Usually the new Taylor expansion for Q_2 is computed right away, but this is suppressed by setting $\tilde{t}_2^2 = t_2^1$.

The next step consists of two time updates of the small element Q_4 in order to reach time level t_4^2, as in Fig. 5 middle.

Then, the synchronization steps in to prevent element Q_1 from doing its time update, as its neighbor element Q_2 does not have a space-time Taylor expansion. Normally element Q_1 would perform its time step based on the available information from its left and right side. Since, on the right side element Q_2 does not provide the necessary data, it is hindered. This enables the merge of cells Q_2 and Q_3 to a new element Q_{2+3} with a different time step and space time Taylor expansion $\tilde{t}_{2+3}^2$. Afterwards, we continue in the normal fashion as described in Sect. 2.2.

4.4 hp-*Adaptation*

Since both *h*- as well as *p*-adaptation are available within our framework, the question is which to use and when. In our case, the shock capturing strategy as described in Gassner *et al.* [2, 4], however, leaves us only one choice to use high order elements close to a shock which are required for the artificial viscosity approach. If, however, no shock capturing is required, then the choice of which adaptation $h-$ or $p-$ to do first is still open and requires further investigation.

5 High Order Numerical Results with *hp*-Adaptation

For each application a study finding the optimal settings is required. The ADIGMA MTC 3 test case is a numerical simulation of the fluid flow around a NACA0012 airfoil with the following parameters, angle of attack $\alpha = 2°$, free stream Mach number $Ma_\infty = 0.5$ and the free stream Reynolds number $Re_\infty = 5000$ based on the length of the airfoil $c = 1.0$. We compute it as a laminar but yet unsteady flow computation till $t_{end} = 0.4$. The mesh used is the coarsest unstructured quads mesh –level 1 for the viscous case– from the ARA mesh catalogue with 578 cells.

5.1 Laminar Flow Past a NACA0012 Profile with hp-*Adaptation*

For the *hp*-adaptation, the order varies between 2–8. The mesh used is rather coarse and therefore the vortex street formation can only be observed in the simulations with a resolution higher than second order. In addition, the simulations have been carried out in parallel on up to 6 CPUs using ParMETIS (see e.g. [7]) for dynamic load balancing at every synchronizing/output interval. Adapting the mesh and adjusting the order yields a good resolution of the vortex street.

Figure 6 shows the result in entropy as well as the order distribution around the airfoil with 13855 DOFs on 815 cells.

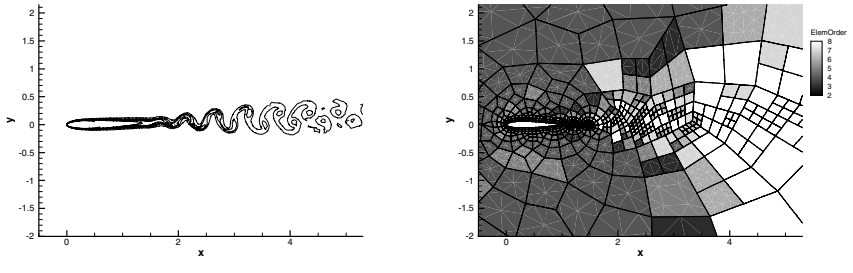


Fig. 6 Fluid flow around NACA0012 for $Re_\infty = 5000$, $Ma_\infty = 0.5$. 16 equally spaced Entropy contour lines from $[11.3, \ldots, 11.5]$ for computation using *hp*-adaptation (left). The order on the refined mesh varies between 2–8 (right).

6 Conclusion

The STE-DG's scheme explicit approximation leads to a compact discretization for which data is only needed from the direct adjacent grid cells. This locality and the space-time nature of our DG discretization allows to introduce local time steps and thus radically change the usual time advancement. Without introducing much computational overhead, every grid cell may run with its own time step adopted to the local stability restriction. This possibility of using local time steps strongly increases the efficiency for all problems which need different resolution in different regions of the computational domain.

The further adaption strategy is based on finding under resolved regions during the time dependant simulation with the help of a feature based indicator. There, the mesh or the approximation order will be adjusted until the resolution measured by the indicator lies within the user defined bounds. The mesh may be adapted regularly with isotropic adaption or even irregularly and more efficient using anisotropic adaption. For the coarsening process, the elements are synchronized and merged on a local level without disturbing the overall flow of the local time stepping. Using our split history for the coarsening process, the coarsest mesh obtained is the initial mesh. Hence, creating such a mesh is quite a challenge, because it should be rather coarse but yet should yield a good resolution of the geometry especially when it is curved like in the NACA0012 cases.

When running in parallel mode with adaption, the processor loads have to be readjusted from time to time. This is done at each synchronizing/output interval, when all cells are forced to reach a certain common time.

Especially in connection with our shock capturing, up to now some user experience is required in order to adjust the adaption control parameters for each individual test case. Thus, some preliminary studies like simulations with low order and on a coarse mesh have to be performed prior to a fully optimized run with adaption. In fact, however, this is the general approach in order to assess a simulation's outcome.

The current framework is limited two dimensional transonic and viscous cases. In three space dimensions, splitting and coarsening elements is much more difficult, because the resulting elements would not be of the same type. This is a topic of further studies. However, the *p*-adaptation in 3D is straight forward, because it does not involve the creation and connection of new elements to the mesh. Furthermore, depending on the test case to be simulated using *a posteriori error estimation* and *goal-oriented refinement for multiple target quantities* instead of a feature based indicator for the adaption would increase the efficiency of the adaption and thus of the simulation as well. For such an approach, Hartman and Houston have developed some promising strategies within the ADIGMA project.

Acknowledgements. We gratefully acknowledge funding of this work by the target research project ADIGMA within the 6^{th} European Research Framework Programme.

References

1. Flaherty, J.E., Loy, R.M., Shephard, M.S., Szymanski, B.K., Teresco, J.D., Ziantz, L.H.: Adaptive local refinement with octree load balancing for the parallel solution of three-dimensional conservation laws. J. Parallel Distrib. Comput. 47(2), 139–152 (1997)
2. Gassner, G.: Discontinuous Galerkin Methods for the Unsteady Compressible Navier–Stokes Equations. Dissertation, University of Stuttgart (2009)
3. Gassner, G., Altmann, C., Hindenlang, F., Staudenmeier, M., Munz, C.-D.: Explicit Discontinuous Galerkin Schemes with Adaptation in Space and Time. In: 36th CFD/ADIGMA course on hp-adaptive and hp-multigrid methods, VKI LS 2009 (2009)
4. Gassner, G., Lörcher, F., Dumbser, M., Munz, C.-D.: Explicit Space-Time Discontinuous Galerkin Schemes for Advection Diffusion Equations. In: 35th CFD/ADIGMA course on very high order discretization methods, VKI LS 2008-08 (2008)
5. Gassner, G., Lörcher, F., Munz, C.-D.: A contribution to the construction of diffusion fluxes for finite volume and discontinuous Galerkin schemes. J. Comput. Phys. 224(2), 1049–1063 (2007)
6. Gassner, G., Lörcher, F., Munz, C.-D.: A discontinuous Galerkin scheme based on a space-time expansion II. Viscous flow equations in multi dimensions. J. Sci. Comput. 34(3), 260–286 (2008)
7. Karypis, G., Kumar, V.: A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. J. Parallel Distrib. Comput. 48(1), 71–95 (1998)
8. Krivodonova, L., Xin, J., Remacle, J.-F., Chevaugeon, N., Flaherty, J.E.: Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws. Appl. Numer. Math. 48(3-4), 323–338 (2004)
9. Lörcher, F., Gassner, G., Munz, C.-D.: A discontinuous Galerkin scheme based on a space-time expansion I. Inviscid compressible flow in one space dimension. J. Sci. Comput. 32(2), 175–199 (2007)
10. Persson, P.-O., Peraire, J.: Sub-cell shock capturing for discontinuous Galerkin methods. In: Proc. of the 44th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2006-1253, Reno, Nevada (January 2006)

This page intentionally left blank

Chapter 31
Anisotropic Mesh Adaptation in the Presence of Complex Boundaries

Jerzy Majewski and Jacek Rokicki

Abstract. Grid adaptation is a very powerful tool for optimizing CFD calculations. However typical isotropic adaptation used for 3D flows still may result in excessive number of elements. This is especially the case when lower+dimensional features of the flow are dominant (boundary layers, shockwaves). The present paper investigates anisotropic adaptation for flows with complex boundaries. Of particular interest is the automatic adaptation for laminar / turbulent boundary layer for which case new error indicator is proposed.

1 Error Estimation for Anisotropic Adaptation

The anisotropic adaptation depends on the specific definition of the cell spacing provided by the means of a metric tensor field. This field is used for distance calculations inside grid generator in such a way that the generator tries to create a grid for which the length of all cell edges is equal 1. The length of an edge $\mathbf{e}$ is calculated according to the following approximated formula:

$$l = \sqrt{\mathbf{e}^T \mathscr{M}\, \mathbf{e}} \tag{1}$$

where $\mathscr{M}$ is a metric tensor averaged on $\mathbf{e}$ (the matrix $\mathscr{M}$ must be symmetric and positive definite).

The metric tensor $\mathscr{M}$ is used as a definition of the local spacing and should be provided by the external error estimator. The metric tensor $\mathscr{M}$ can be interpreted

Jerzy Majewski
Institute of Aeronautics and Applied Mechanics, Warsaw University of Technology, ul.Nowowiejska 27, 00-665 Warszawa
e-mail: jmajewsk@meil.pw.edu.pl

Jacek Rokicki
Institute of Aeronautics and Applied Mechanics, Warsaw University of Technology, ul.Nowowiejska 27, 00-665 Warszawa
e-mail: jack@meil.pw.edu.pl

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 441–453, 2010.
springerlink.com

as ellipsoid (ellipse in 2D) which is circumscribed on the optimal grid cell. Such ellipsoid has three main axes and their length (h_i) can be found using eigen-problem:

$$\mathscr{M} = R \cdot \begin{bmatrix} \frac{1}{h_1^2} & 0 & 0 \\ 0 & \frac{1}{h_2^2} & 0 \\ 0 & 0 & \frac{1}{h_3^2} \end{bmatrix} \cdot R^{-1} \tag{2}$$

The directions of main axes are defined by corresponding columns of the eigenvector matrix R.

In the following Sections two approaches used for anisotropic error estimator will be presented. The first one, which is typically used in literature for anisotropic adaptation, is based on a solution Hessian. The second one is based on a solution gradient and after blending with the Hessian metrics, it is used for viscous high Reynolds number simulations.

1.1 Calculation of the Metric - Hessian Based Approach

Details of the Hessian based approach can be found in ([7], [13], [3]) and here only the main concept will be described.

Assume that E denotes a grid cell inside which a function u is being interpolated and $\mathbf{x}_c$ is the center of E. Then after dropping terms of higher order, the interpolation error for E can be estimated as:

$$\varepsilon_E \leq \max_{\mathbf{x} \in E} \left(\mathbf{x} - \mathbf{x}_c\right)^T |\mathscr{H}| (\mathbf{x} - \mathbf{x}_c) \tag{3}$$

where $\mathscr{H}$ is a Hessian of u:

$$\mathscr{H} = \begin{bmatrix} \frac{\partial^2 u}{\partial x^2} & \frac{\partial^2 u}{\partial x \partial y} & \frac{\partial^2 u}{\partial x \partial z} \\ \frac{\partial^2 u}{\partial x \partial y} & \frac{\partial^2 u}{\partial y^2} & \frac{\partial^2 u}{\partial y \partial z} \\ \frac{\partial^2 u}{\partial x \partial z} & \frac{\partial^2 u}{\partial y \partial z} & \frac{\partial^2 u}{\partial z^2} \end{bmatrix} = R \cdot \Lambda \cdot R^{-1} \tag{4}$$

Consequently the $|\mathscr{H}|$ can be defined as:

$$|\mathscr{H}| = R \cdot \begin{bmatrix} |\lambda_1| & 0 & 0 \\ 0 & |\lambda_2| & 0 \\ 0 & 0 & |\lambda_3| \end{bmatrix} \cdot R^{-1} \tag{5}$$

The interpolation error in a given direction defined by the unit vector $\mathbf{w}$, is proportional to:

$$C = h^2 \, \mathbf{w}^T |\mathscr{H}| \mathbf{w} \tag{6}$$

where h denotes a length of a cell in the direction of unit vector $\mathbf{w}$. In the present approach $\mathbf{w}$ becomes a direction of a given edge and h becomes an edge length. The construction of an optimal grid is based on the equidistribution of the interpolation

error. This is equivalent to assuming that for every edge e_i, the constant C_i is equal to a global C. We can introduce now a scaled metric $\mathscr{M}$:

$$\mathscr{M} = C^{-1} \ |\mathscr{H}| \tag{7}$$

Additionally the eigenvalues in (5) are limited from above and below to avoid degeneration of cells (see Section 1.3).

The actual calculation of the Hessian follows the Green formula approach presented in [11].

1.2 Calculation of the Metric - Gradient Based Approach

When viscous flows are the subject to anisotropic adaptation the typical approach based on Hessian shows unwanted behavior. The adapted grid tries to resolve area with high second derivative solution. Unfortunately inside boundary layer the second derivative is high at a distance from the wall (see Fig. 1). In consequence mesh remains underrefined next to the wall boundary. This issue has been addressed elsewhere ([7], [9]) by adding a metric tensor field which is defined near the viscous boundary and relies on the user specified thickness of the viscous boundary layer. This approach works well, however additional user input is necessary and the adaptation is no longer fully automatic (especially for shear layers).

Therefore another method has been proposed which in addition to standard Hessian uses the gradient of magnitude of the fluid velocity $\mathbf{v}$. This additional metric tensor is defined in the following way:

$$\mathbf{w} = \nabla u \qquad u = |\mathbf{v}| \tag{8}$$

$$\mathscr{M} = \mathbf{w} \otimes \mathbf{w} = R \cdot \Lambda \cdot R^{-1} \tag{9}$$

$$\Lambda = diag(\mathbf{w} \cdot \mathbf{w}, 0, 0) \tag{10}$$

Such metric represents in correct way the thickness of a boundary layer in direction of the gradient (which for typical flows is close to the vector normal to the wall) but not in the tangent directions. Therefore this metric field has to be blended with metric obtained with the Hessian based approach.

In order to compare Hessian-based and gradient-based approaches the grids were generated using the same solution (NACA-0012 Re=5000 $\alpha = 2^\circ$) as an input for error estimator. Moreover the estimators were set in such a way that both grids have approximately the same number of nodes. The result can be seen on Fig. 1 and Fig. 2. The first one (Fig. 1) shows overview of the grid near NACA-0012 airfoil while the second one (Fig. 2) shows details of the grids inside boundary layer. It can be clearly seen that gradient-based approach produces the grid with much better resolution. Another example can be found in the Section 4.3 which shows results

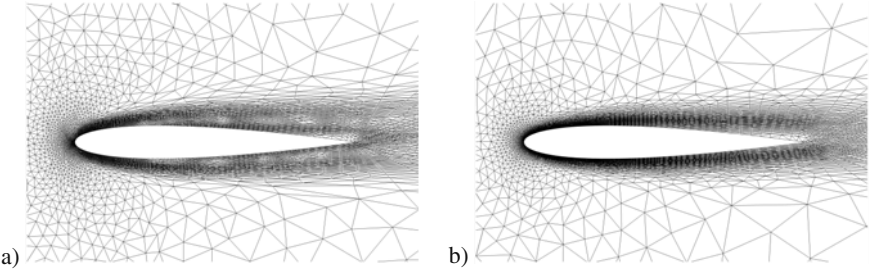


Fig. 1 Comparison of two approaches to the anisotropic error estimation applied for viscous flows. a) grid generated for hessian-based approach. b) grid generated for gradient-based approach.

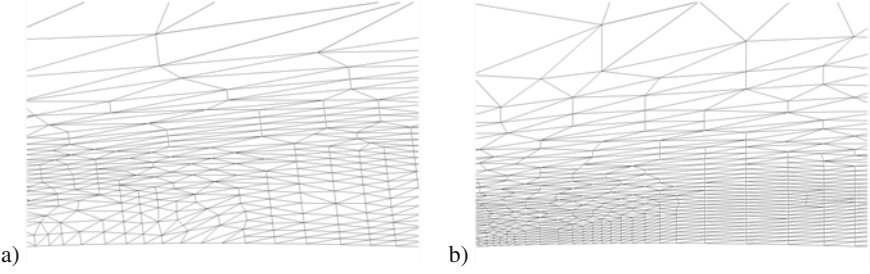


Fig. 2 Comparison of two approaches to the anisotropic error estimation applied for viscous flows (details of the grid for boundary layer).
a) grid generated for hessian-based approach. b) grid generated for gradient-based approach.

for turbulent flow with high Reynolds number past RAE-2822 as well as Section 4.4 presenting results of the flow past multi element airfoil.

1.3 Metric Limiting

The process of limiting is necessary to avoid unrealistic values of the grid cells (in fact the matrix defined by (5) is not yet positive definite). Assuming that values of h_{min} and h_{max} (which are provided by the user) define the maximum and minimum of the acceptable grid spacing the following modification ([13]) can be suggested:

$$\hat{\lambda}_i = \left(\frac{1}{\max(h_{min}, \min(h_{max}, h_i))} \right)^2 \tag{11}$$

where $h_i = 1/\sqrt{\lambda_i}$. The metric $\mathscr{M}$ is subsequently calculated for every node of the old grid and by means of interpolation is extended to form a continuous metric field.

2 Anisotropic Grid Generation for Complex Geometries

A new grid in adaptation loop is obtained by means of remeshing, i.e., the grid generator discards all information about the old grid and creates a new one from a scratch using metric tensor field as a definition of cell spacing. The generator in the presented approach relies on anisotropic Delaunay triangulation with point-placement strategy based on creation along edges approach [13] or cell center approach [17].

The main interest here is related to proper treatment of the boundary. Typically once the grid is created and sent to the solver, the information about domain boundary is discarded. However in order to perform adaptation, the boundary information must be present during the whole cycle.

The access to boundary information was accomplished through the specialized GeT (Geometry and Topology) library which forms the minimalistic interface to the CAD system. It handles:

- geometrical entities (e.g., Point, Curve, Surface)
- topological entities (e.g., Vertex, Edge, Face, BRep)

Another problem is the general definition of the so called Control Space which allows for consistent evaluation of metric tensor $\mathscr{M}$ on various levels of hierarchical mesh generation [13]. For this purpose various operators were defined and implemented [14], e.g.:

- slicing operator (which extracts tensor $\mathscr{M}$ for lower dimensional entities)
- superposition operator (when the composition of two metrics are required)
- blending operator (when more then one metric space is used for spacing definition - relies on metric intersection)

3 Anisotropic Adaptation Algorithm

Detailed description of the anisotropic grid generation algorithms can be found in [13], [12] and [4]. The present one can be described by the following steps:

I. Generate initial grid G^0 and set $k \leftarrow 0$
II. Solve flow equations on a grid G^k in order to obtain solution S^k.
III. Check the criterion to end the adaptation loop (e.g., whether the number of grid nodes between G^k and G^{k-1} does not vary significantly or k is equal to k_{max}). If this criterion is satisfied, adaptation loop is finished.
IV. Using error estimator find metric tensor field $\mathscr{M}^k$ for the solution S^k.
V. Generate a new grid G^{k+1} in a Riemann space using metric tensor field $\mathscr{M}^k$.
VI. Interpolate solution S^k on the new grid G^{k+1} (to obtain the initial guess for S^{k+1}
VII. return to point II setting $k \leftarrow k+1$.

4 Numerical Results and Verification

4.1 Transonic Flow for Onera m6 Wing

This testcase is a standard problem of transonic inviscid flow past Onera m6 wing. The Mach number is 0.8395 and angle of attack $\alpha = 3.06^\circ$. The solution to the problem is typical lambda shock wave formed on the upper surface of the wing.

The anisotropic adaptation was applied 6 times and results are shown on Figure 3. The adaptation was based on standard Hessian based error estimator with Mach number field used for input.

Surface grid was generated with in-house grid generator [10] coupled with GeT library used for modeling of the geometry of the wing. The volume grid for initial grid was generated with in-house volume generator which still is in the development stage while for adapted grids the VKI Meandros grid generator was used. Due to problems to properly recover highly anisotropic surface mesh inside volume grid the metric tensor was limited to force cell aspect ratio (AR) to be less then 1.2. However the volume grid was created without this restriction and as shown in section 4.5 the gain over fully isotropic approach is essential. Final grid consist of 572586 nodes and 3255350 cells.

The solution was calculated with Residual Distribution solver using LDAN scheme.

Figure 3 shows Mach number field and surface grid for the solution after 6 adaptations for top view for the wing surface. The solution shows the thin and well developed shock waves on the wing surface and also inside the domain.

4.2 Transonic Flow for Wing-Body-Nacelle Configuration DLR-F6

This testcase was used as a verification of capabilities of the developed tools used for simulation of complex geometries. This is the wing-body-nacelle configuration DLR-F6 for which the definition of the geometry can be freely available in the form of the STEP file [1]. The STEP file was imported with GeT library and was used as a base for geometry definition. Again it was necessary to fix few problems with topological connectivity and add definition for the external boundary. Subsequently it was used as an input for surface grid generator. The volume grid was created with VKI Meandros tool.

The solution was calculated with RDS LDAN solver (M=0.76 $\alpha = 0.5^\circ$). The grid was adapted 3 times. The Mach number field for grid after 3 adaptation steps (287915 nodes and 1679514 cells) is shown on Fig. 4. Some details of the grid and solution near the nacelle are shown on Fig. 5. Thanks to the adaptation it was possible to improve quality of the solution without increasing the computational cost in significant way - the final grid has only 2 times more degrees of freedom.

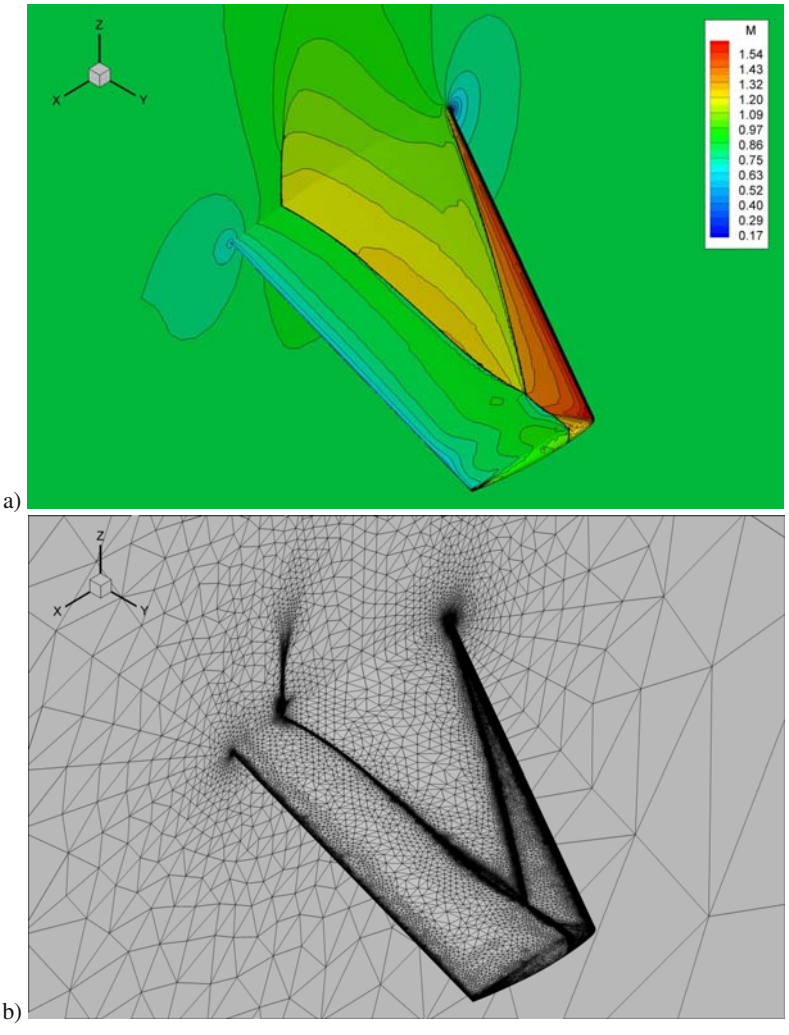


Fig. 3 Mach number field and surface grid (after 6 adpatation steps), Onera m6 wing, M=0.8395, $\alpha = 3.06^\circ$ - top view

4.3 Turbulent Transonic Flow for RAE-2822 Airfoil

The flow past RAE-2822 airfoil was chosen as a first testcase to test the gradient-based error estimator used for anisotropic adaptation for turbulent, high Reynolds number problems. The flow conditions are $Re = 6.5 \cdot 10^6$, $\alpha = 3.19^\circ$ and $M = 0.73$. Calculations were performed with THOR code developed at VKI (see [16]). It is based on Residual Distribution Scheme coupled with Spalart-Almaras turbulence

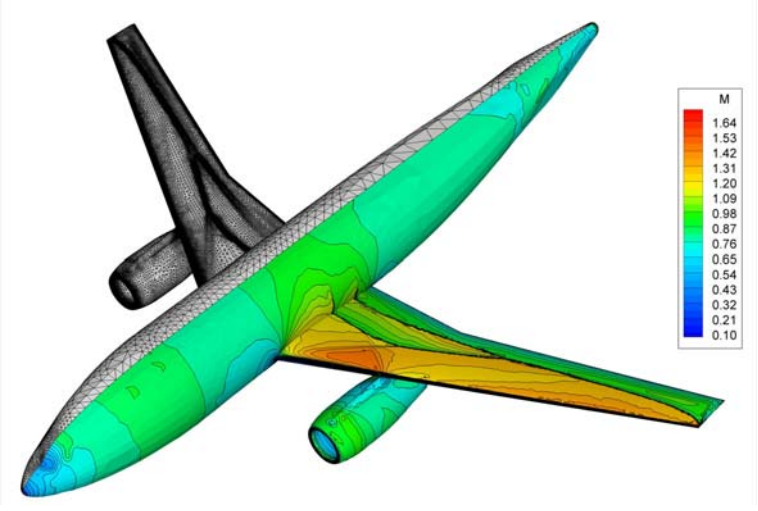


Fig. 4 Mach number field on grid after 3 adaptation steps, DLR-F6, M=0.76, $\alpha = 0.5^\circ$ - top view

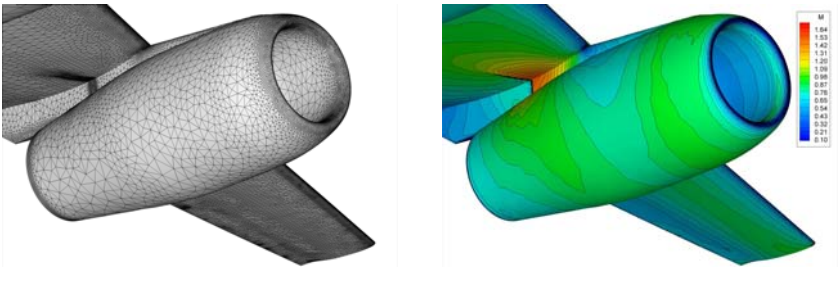


Fig. 5 Mach number field and grid after 3 adaptation steps, DLR-F6, M=0.76, $\alpha = 0.5^\circ$ - nacelle close-up

model. In order to receive the final solution 13 adaptations were performed. The initial grid was typical for inviscid calculations without capability to resolve the boundary layer. The boundary layer was eventually properly discretized thanks to the adaptation.

The sequence of adaptation steps (Mach number field and the corresponding grid) is shown on Figures 6–7. It can be seen that starting from grid with no boundary layer it was possible to obtain the grid which properly resolved the BL spacing

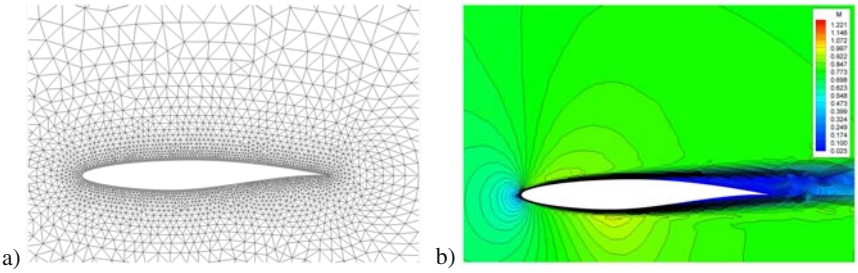


Fig. 6 Mach number field and grid without any adaptation (RAE-2822)

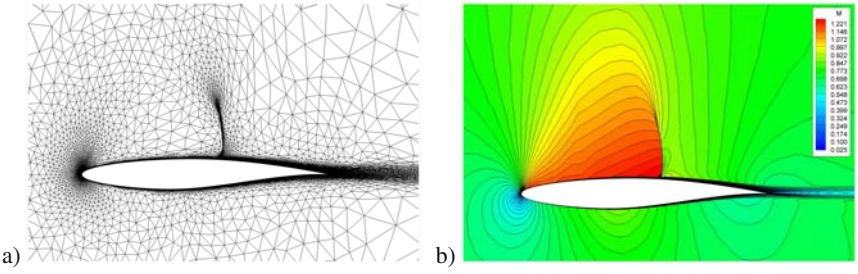


Fig. 7 Mach number field and grid after 13 adaptation steps (RAE-2822)

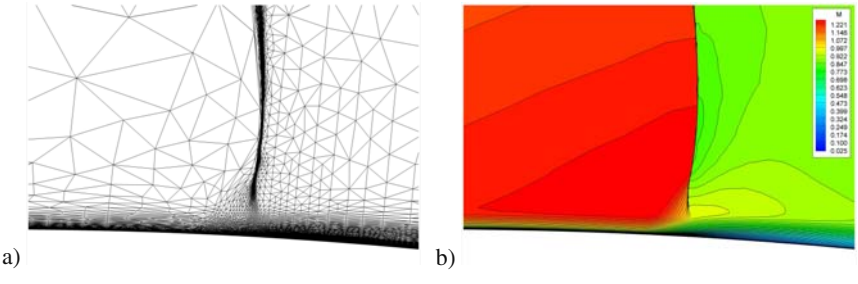


Fig. 8 Mach number field and grid after 13 adaptation steps - deatils near the shock wave root (RAE-2822)

(thickness of the cells at the wall for 9th adaption grid is of magnitude of 10^{-5}). Some details of the final grid can be seen on Figure 8. Finally the comparison of the computed pressure coefficient c_p with experiment is shown on Figure 9.

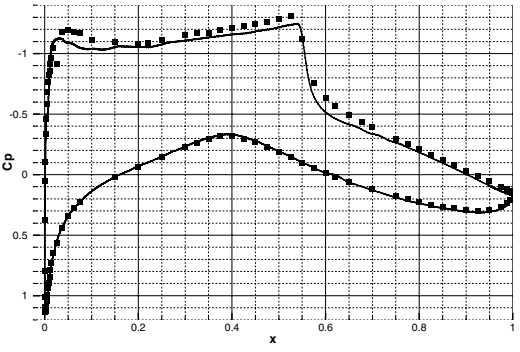


Fig. 9 Cp distribution obtained on grid after 13 adaptation steps (RAE-2822), compared with experimental values (diamonds) [8]

4.4 Turbulent Subsonic Flow for L1T2 Multi-Element Airfoil

The flow past L1T2 multi-element airfoil [2] was used as a second verification of the anisotropic adaptation applied to high Reynolds number turbulent flow. The flow conditions were $Re = 3.52 \cdot 10^6$, $\alpha = 20.18^\circ$ and $M = 0.197$. For this conditions the grid should adapt not only inside the boundary layer but also inside wakes which emanate from slat, main profile and also from the flap. The calculations were started once again with grid typical for inviscid computations. Then 8 steps of adaptation were performed resulting with proper resolution of the boundary layer near the wall together with all wakes.

The sequence of adaptation steps can be seen on Figures 10–11. The distribution of pressure coefficient for final adaptation can be seen on Figure 12.

a) b)

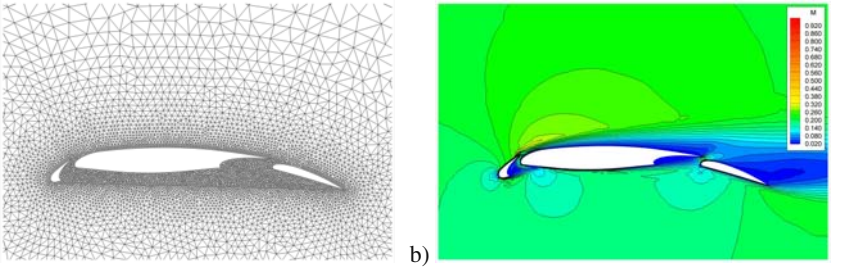


Fig. 10 Mach number field and grid without any adaptation (L1T2)

4.5 Estimation of the Computational Gain for Anisotropic vs Isotropic Adaptation

The final computations present difference between anisotropic and isotropic adaptation applied for 3D problems. The Onera m6 wing transonic flow was

a) b)

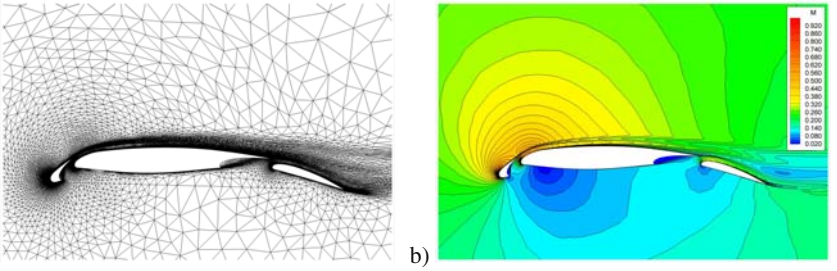


Fig. 11 Mach number field and the corresponding grid after 8 adaptation steps (L1T2)

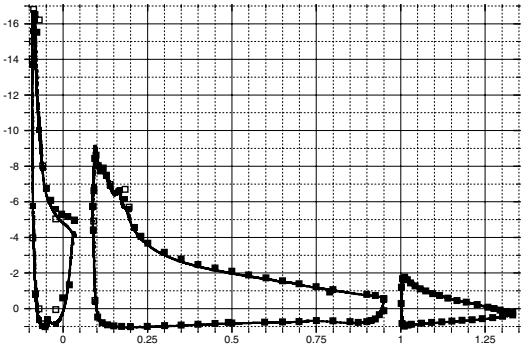


Fig. 12 Cp distribution obtained on grid after 8 adaptation steps (L1T2)

chosen for calculation and one step of adaptation was done using both approaches - anisotropic and isotropic. The only difference between both method is the modification of the metric tensor for isotropic case in such a way that all eigenvalues are set to their maximum. It is forcing the size of the cell to be the same in all directions.

Anisotropic grid consist of 574974 nodes and 3473850 cells while isotropic one has 5078927 nodes and 32622210 cells. The anisotropic grid has almost 9 times smaller number of cells then its isotropic counterpart. It is quite a significant ratio especially if one takes into account that the surface grid had to be limited to cell AR=1.2 in order to overcame problems with volume grid generation. Similar estimation of the computational advantage of the anisotropic adaptation over isotropic one was shown in [15] where authors received the gain factor close to 10. Details of the shockwave for both type of grids can be seen on Figures 13.

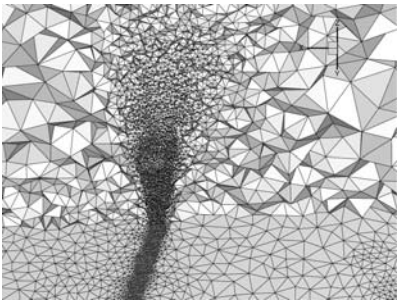
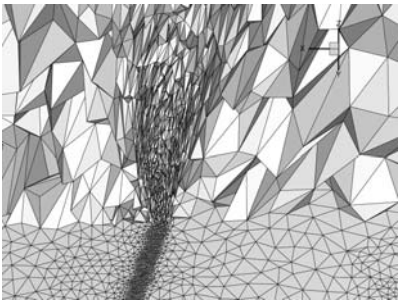
a)  b) 

Fig. 13 Grid details around the shock wave a) after isotropic adaptations (approximately 32.6 mln cells) b) after anisotropic adaptations (approximately 3.5 mln cells)

5 Conclusions and Future Work

In this paper an anisotropic approach to adaptation for complex domains was presented. It was also shown that the advantage of the grid adapted in anisotropic way over isotropic one is substantial (for Onera m6 wing the gain factor is 9). The anisotropic adaptation was also applied for high Reynolds number turbulent flows. After adding gradient-based component to the Hessian-based error estimator it was possible to discretize the boundary layer and wake in fully automatic way.

However promising the results are, there are also many problems to be addressed. One of the main issues is reliable 3D anisotropic grid generator. At the current stage of development there are some problems with proper reconstruction of the boundary grid especially for very stretched elements. Also the quality of inner cells could be improved (problem with *slivers*). Finally the adaptation for turbulent flows must be extended for 3D complex configurations (e.g., DLR-F6).

Acknowledgments

- The most of the work presented here were done in frame of the ADIGMA project AST5-CT-2006-030719 in close cooperation with the Von Karman Institute.
- All turbulent calculations were done using THOR code developed in VKI by Kurt Sermeus.
- The generator MEANDROS developed in VKI by Aristotelis Athansiadis was used for most of 3D cases for final tetrahedralizations of 3D domains.

References

1. 2nd AIAA CFD Drag Prediction Workshop, `http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/Workshop2/`
2. A Selection of Experimental Test Cases for the Validation of CFD Codes. Technical Report AR-303, AGARD (1994)
3. Alauzet, F.: Adaptation de maillage anisotrpe en trois dimensions. Application aux simulations instationnairesen Mécanique des Fluides. PhD thesis, de l'Université des Sciences et Techiques du Languedoc (2003)

4. Athanasiadis, A.: Three-Dimensional Hybrid Grid Generation with Application to High Reynolds Number Viscous Flows. PhD thesis, Université Libre de Bruxelles, Chaussée de Waterloo, 72, 1640 Rhode-St-Genèse, Belgium (2004)
5. Borouchaki, H., George, P.L., Hecht, F., Laug, P., Saltel, E.: Delaunay mesh generation governed by metric specifications. Part I. Algorithms. Finite Elem. Anal. Des. 25(1-2), 61–83 (1997)
6. Borouchaki, H., George, P.L., Mohammadi, B.: Delaunay mesh generation governed by metric specifications. Part II. Applications. Finite Elem. Anal. Des. 25(1-2), 85–109 (1997)
7. Castro-Díaz, M.J., Hecht, F., Mohammadi, B.: New progress in anisotropic grid adaptation for inviscid and viscous flows simulations. Technical Report RR-2671, INRIA (1995)
8. Cook, P.H., McDonald, M.A., Firmin, M.C.P.: Aerofoil RAE 2822 - Pressure Distributions, and Boundary Layer and Wake Measurements, Experimental Data Base for Computer Program Assessment. Technical Report AR-138, AGARD (1979)
9. Dolejší, V.: Anisotropic mesh adaptation technique for viscous flow simulation. East-West J. Numer. Math. 9, 1–24 (2001)
10. Dolejší, V., Majewski, J., Quintino, T.: Computational performance of diferent data structure models for grid adaptation. Technical Report ADIGMA/WP5/UNPR/D5.1.4 (2009)
11. Formaggia, L., Perotto, S.: Anisotropic error estimation for finite element methods. In: 31st Computational Fluid Dynamics VKI Lecture Series, von Karman Institute for Fluid Dynamics (2000)
12. Frey, P.J., George, P.L.: Mesh Generation - Application to Finite Elements, 2nd edn. ISTE/Wiley (2008)
13. George, P.L., Borouchaki, H.: Delaunay Triangulation and Meshing - Application to Finite Element. Hermes (1998)
14. Majewski, J.: Anisotropic adaptation for flow simulations in complex geometries. In: 36th Lecture Series on Computational Fluid Dynamics/ADIGMA course on hp-adaptive and hp-multigrid methods. von Karman Institute for Fluid Dynamics (2009)
15. Mesri, Y., Alauzet, F., Loseille, A., Hascoet, L., Koobus, B., Dervieux, A.: Continous mesh adaptation models for cfd. CFD Journal 12(3) (2008)
16. Sermeus, K., Deconinck, H.: Solution of steady euler and navier-stokes equations using residual distribution schemes. In: 33rd Lecture Series on Computational Fluid Dynamics - Novel Methods for Solving Convection Dominated Systems (LS2003-05). von Karman Institute for Fluid Dynamics (2003)
17. Weatherill, N.P., Hassan, O., Marcum, D.L., Marchant, M.J.: Grid generation by the delaunay triangulation. In: Lecture Series on Grid Generation (LS1994-02), vol. 2, von Karman Institute for Fluid Dynamics (1994)

This page intentionally left blank

Chapter 32
Requirements and Assessment Methodology

H. Bieler and K.A. Sørensen

Abstract. In this chapter the procedure followed in the industrial evaluation of the achievements in the ADIGMA project is described. This entails a specification of the assessment criteria used in the evaluation as well as methods employed for convergence analyses with the underlying goal of obtaining an as accurate and fair procedure as possible, within the resources made available for such studies in ADIGMA.

1 Introduction

The title of ADIGMA underlines the importance of industrial relevance in the project. This was reflected in the formulation of the assessment criteria where methods allowing for a fair and, as far as possible, accurate analysis, were sought. The scope of the project is the analysis of the industrial attractiveness of higher order methods and related technologies. This excludes the effects of modelling, the comparison of the flowfield results between partners is consequently of little importance for many cases as the partners use different turbulence models and versions thereof. For Euler and laminar flow, the comparison of results can be performed as a consistency check, this issue is however not central in the project.

The critical assessment of the newly developed methods for industrial aerodynamic applications was designed to allow for the identification of the best numerical strategies for the next generation of industrial flow solvers. The industrial partners in the project were given the task of specifying these assessment requirements and to give guidelines to the evaluation procedure for the new methods. A test case suite of increasing complexity was specified together with concise reporting templates

H. Bieler
AIRBUS Operations GmbH, Bremen, Germany
e-mail: heribert.bieler@airbus.com

K.A. Sørensen
EADS–MAS, Aerodynamics & Methods, Rechliner Str., 85077 Manching, Germany
e-mail: kaare.sorensen@eads.com

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 455–464, 2010.
springerlink.com

in order to put the comparison of newly developed methods with traditional industrial flow solvers on a firm basis. This industrial specification covered the following aspects:

- Requirements and scope of assessment
- Definition of test cases
- Inquiry of technology status at project start

2 Requirements and Scope of Assessment

An ADIGMA project goal has been to assess the level of attractiveness of a particular implementation or method for industry. For this to be done in a proper manner, the criteria of relevance to industry and their relative levels of importance must be clarified. A discussion was conducted in the beginning of the project, mainly involving the industrial partners and the research institutions treating this issue. The result was the following items

- Efficiency
- Applicability
- Robustness
- User friendliness
- Multidisciplinarity
- Implementational issues

Again, it is noted here that there is no direct reference to accuracy in this definition. This is a reflection of the fact that modelling errors are not an issue in ADIGMA, it is assumed that all implementations are consistent in that they converge to a unique solution as the mesh spacing decreases and that this solution fulfils the modelled governing equations.

2.1 Efficiency

In industry, a discrete system is traditionally defined as converged when the global integral values of relevance have reached a clear value in the solution iteration process. Since industry is always working with tolerances, and since it is well known that the modelled equations only represent the non-modelled equations to within a non-negligible tolerance, there is no gain in attempting to solve the numerical equations to within overly stringent convergence criteria. A common practice in industry is to converge to within "engineering accuracy", a level which is usually highly case dependent. The efficiency of a given code on a given machine and number of CPU's or cores from an industrial viewpoint is thus the wall-clock time required to converge to engineering accuracy on a mesh with resolution adequate for engineering accuracy. There is thus no direct interest from an industrial viewpoint in how many DOF's there are in the discrete problem. The number of DOF's are however relevant in comparisons between approaches that have not yet matured to a level that can be

considered state of the art, as an indication of the potential a code has to reduce the convergence time through a reduction in the discrete problem size.

The industrial definition of convergence is not a constant, even the entities used in the definition of convergence may vary. Normally however, the industrial aerodynamicist is focused on the full set of integral values for the forces and moments of the case. The usage of integral values was also considered useful for the ADIGMA project since they are relatively easy to evaluate and their implementation is normally performed at an early stage of a code development. This choice thus effectively has reduced the convergence criteria for each case down to a matter of selecting the proper error bounds for a particular test case, considered to be realistic accuracy expectations for industrial applications. In ADIGMA, the selection of these convergence criteria was usually performed by the partner proposing the test-case. Typically, engineering accuracy is of the order of 1-5% of a typical reference value of the particular integral value. For the drag integral value, it makes sense to report the pressure and viscous components as well as the total value. If the convergence history of a given integral value is such that no point in the convergence history can be found after which the value is bounded within the specified convergence criteria, this was to be reported, and the convergence of this integral value was to be set to the point in the convergence curve after which the convergence stagnates (i.e. the fluctuation amplitude does not decrease and there is no apparent change in the mean value). An effort was made to select cases and convergence criteria in a way to reduce the likelihood of such behaviour.

2.2 Applicability

The applicability of a code can be said to be defined as the a-priori specification of classes of problems in which a code can be expected to produce a solution. Some codes are only capable of subsonic flows as there are not means implemented to handle flow discontinuities. Other codes may not be applicable to high-Reynolds number flows since there are no turbulence models implemented. The applicability of a code can be reduced due to a low level of maturity of the implementation even though there is no technical reason why such constraints should be imposed on the underlying method. A more critical case is when the partner has not implemented a feature due to problems in introducing the capability to a sufficient degree.

2.3 Robustness

Defining the robustness of a code as the likelihood of being able to produce a solution to a given problem within the applicability domain, it is clear that this is a very important criterion of for industry. It must be noted that there is no reference in this parameter to the effort of obtaining this solution. The robustness definition does not overlap the definition of applicability, since applicability is an a-priori classification of the domain of application of the code, while the robustness is an indicator applied within the applicability domain.

2.4 User-Friendliness

The user-friendliness of a code is inversely proportional to the manual labour required to produce a solution. A code could be very robust if there is a high probability of obtaining a solution, but if this requires a lot of tweaking of parameters or the generation of several meshes, the user-friendliness of the code is reduced. If several meshes are indeed required, the efficiency analysis should be conducted only on the mesh that produces the solution. A key ingredient for the success of a numerical method in industry with end-users (design engineers) is the possibility to have a single set of numerical parameters appropriate to the majority of the computations to be performed.

2.5 Multidisciplinarity

As the codes in current industrial use are increasingly being applied to solve multidisciplinary problems, it is important that new approaches are also capable of such computations. For the industries represented in ADIGMA, the fields of aeroelasticity and aeroacoustics are of central interest, the new approaches should however also be capable of efficiently solving problems involving combustion and electromagnetic effects.

2.6 Implementational Issues

The implementational issues related to a code describes the required datastructure, parallelizability, implementational effort and hardware dependency of the approach. Optimally, a new approach would be implemented easily into the already existing industrial codes, so as to re-use highly optimized and tested components. However, if a sufficient gain in performance and/or stability is obtained at the price of writing a completely new code, this should of course not be ruled out.

The datastructure required is relevant with respect to code memory usage and efficiency issues, for example. with respect to caching. The efficiency issues are particularly important if the code is not optimized so as to be able to assess the theoretical performance of this part of the code. The memory usage of a code is important and should be compared with the current typical memory usage of a state of the art industrial code. If a new approach does not significantly exceed, say by a factor of two, the memory usage experienced in current state of the art codes, this issue is of second priority in comparisons.

The parallelizability of the code should be reported, either in the form of the experienced speed-up values if a parallel implementation exists, or the assessed parallelizability of the code if only a sequential version has been implemented. The latter entails a statement of whether the code demands procedures that are inherently sequential in nature and whether the parallelization has a direct influence on the solver performance (such as increasing the iterations for convergence for

multidomain solvers), as well as the amount of data that the approach requires to be communicated between processors or cores.

The implementational effort of a code is important to industry. As mentioned above, the optimal case would be a new approach that could be implemented into an existing industrial code with relatively small changes. This would allow for the reusage of highly optimized and tested code components that could otherwise require excessive resources to recreate. If a new code needs to be written to accommodate an improved approach, this is of course not critical if the performance gains are sufficiently high. If it turns out that a new scheme offers a relatively small performance improvement, but that it can be implemented into existing codes in a straightforward way, it is still industrially relevant. This is likely not to be the case if the small improvement requires excessive implementational effort.

3 Efficiency Analysis

As defined above, the efficiency of a method is dependent on first finding the smallest mesh size required for the solution to stay within the defined error tolerance, and then measure the time it takes to converge the discrete system on this mesh to within the engineering accuracy.

The mesh convergence analysis of a discrete problem is a procedure to identify how close a solution on a given mesh is to the solution resulting from the limit where the number of degrees of freedom in the problem approaches infinity in a regular way. Mesh convergence of a given mesh thus depicts how close the solution of the discrete system resulting from this mesh is to a solution from the same solution procedure on an infinitely fine, regular mesh. In practice, infinite meshes can not be considered, instead a series of successively finer meshes (with similar point distributions) can be created until a point is reached where the difference in the integral values between two consecutive meshes is much smaller than the convergence criteria and that an asymptotic analysis of the integral value confirms that the finest mesh is considerably closer to the asymptotic solution than the size of the convergence criteria. For this analysis, it is important to converge the discrete systems on each mesh to a higher degree than the engineering accuracy described above, if possible, to ensure that the errors related to the solution procedure itself do not interfere with the mesh convergence study. If a truly steady state solution is not achievable, average values can be used in the analysis. The coarsest mesh that is within the convergence criteria from the approximated asymptotic value is taken as the converged mesh. The distribution of the points on the meshes could be optimized for the discretization scheme, but if a high level of (non-automatic) mesh tailoring is needed for the code to be efficient, the user-friendliness of the approach will be reduced.

In addition to using mesh cascades for convergence, the usage of the adjoint error estimates can be applied on very fine meshes to increase the accuracy of the analysis. It must however always be kept in mind that, due to issues such as non-optimized meshes for a particular method, the asymptotic analysis is not of a very

high accuracy there is thus a limit after which the increased accuracy in finding asymptotic value is not necessary.

It must also be mentioned that a proper mesh convergence analysis requires a considerable amount of computational resources, even for relatively simple problems. The partners were therefore not forced to perform such analyses on every attempted test case. An impression of the mesh convergence properties of a code should however be possible on a basis of simpler cases. The more complex test cases are still relevant in the project for evaluations of characteristics such as robustness, ease of use, memory usage and multidisciplinarity.

3.1 Asymptotic Convergence Analysis

An important issue in ADIGMA is the determination of the number of degrees of freedom required for a given problem with a given accuracy level. Such questions are best answered using asymptotic analyses, where a series of meshes, with similar local resolution distributions, are analysed.

If convergence is measured in a physical variable (e.g. lift coefficient, pressure at a given point etc.), and the problem is well-posed, the assumption

$$V = V_\infty + cN^{-\alpha} \tag{1}$$

holds for classical schemes if N is large enough. Here V, V_∞, α and c are the asymptotic value of the variable, the number of degrees of freedom per equation, the order of the scheme with respect to the number of degrees of freedom per equation and an unknown constant respectively.

To find the unknowns for this equation, three nested solutions are needed. A cascade of meshes thus have to be made, fulfilling the following requirements:

- The resolution adequate for eq. 1 to be approximately valid
- The different mesh resolutions should be significantly different (a factor of $1.5-2$ recommended)
- The meshes should be nested, i.e. have a similar local refinement distribution

The meshes may be tailored for the code in question using an a priori approach. This means that knowledge of the scheme applied may be used in the meshing. If, for example, a discretization scheme is based on a classical second order FV method, but with increased spatial order in the boundary layer, it is known a-priori that the local spacings outside the boundary layer should equal those of standard FV schemes, while the spacings in the boundary layers could be made coarser. Each partner was to use common sense in deciding whether a point distribution strategy is generally applicable to a class of problems without knowing the solution in advance or whether the point distribution strategy represents a tailoring which is unrealistic for industrial computations.

Subsequently, a solution must be computed for each mesh. The solutions should be converged to a level where the convergence parameters have reached an asymptotic value (i.e. not to engineering accuracy). The nonlinear systems of equations

obtained by inserting for V and N in eq 1 for the convergence parameters of the three solutions are then solved by, for example, a Newton iteration approach. The term

$$E_V = c_V N^{-\alpha_V} \tag{2}$$

is thus a representation of the discrete system size dependency of the error of the integral value V, which can be used for approximating the number of degrees of freedom per equation needed to reduce the error below a certain level. The order of the scheme is defined using the local spacing, not the number of degrees of freedom. Since the mesh cascade has similar local point distributions, the following formula holds locally:

$$h = kN^{\frac{1}{n}} \tag{3}$$

where h, k and n are the local spacing, a local coefficient (independent of N) and the number of space dimensions respectively. The order of the scheme with respect to the spacing, β, is thus given by:

$$\beta = \alpha n \tag{4}$$

For time-accurate computations, the time resolution required for obtaining results within the accuracy limits should be identified. This is analogous to the mesh convergence analysis, looking at time as the third (2D) or forth (3D) mesh dimension.

If the complexity of the problem was found to render the above procedure excessively costly, the reduced approach of assuming the scheme order (based on experience on similar but smaller problems), could be applied. The solution of eq. 1 then results in the Richardson formula. The partners were also allowed to use other approaches to find the convergence characteristics of a method. Some partners applied curve-fitting approaches to determine the mesh convergence of their methods.

3.2 Performance Index

When comparing the wall-clock time two codes on different computers require for convergence, a way has to be found to eliminate, or at least reduce, the effect of the different machine speeds. This is a very complex subject since the performance ratio of two codes on two different machines can be far from constant, highly depending on issues such as cache-efficiency, vectorizability etc. In ADIGMA, the problem of machine independency was addressed by the introduction of a dummy performance index code, serving as an indicator of the hardware and compiler performance for the partners. This code was written in both C and Fortran and consists of a 2D FV Euler flux on a regularized mesh. The procedure works as follows:

- The performance index code is compiled on the same machine as the real computation is to be performed, with the same compiler options
- The performance index code is run on the machine (which is otherwise idling), replicating the running conditions of the real run (e.g. using the same amount of cores on a processor), say requiring 10 s of wall clock time

- The real code is run, say requiring 1000 s wall clock time for convergence
- The CPU time required for convergence is given in the performance index code units, in this case convergence takes 100 PI units

For parallel speed-up exercises, the usage of the performance index does not affect scaling issues and is thus straightforward.

3.3 Mesh Reporting Form

The mesh reporting form was created to enable a clear description of the meshes used in the convergence analysis of the schemes, and is detailed enough to enable the generation of similar meshes based on the form only. The form thus includes not only the relevant size parameters of the mesh, such as the total number of nodes and elements on the surface and in the volume, but also a coarse description of the boundary layer mesh generation and the local point resolutions on the mesh. The latter was typically described through characteristic spacings in the background and around specific features, such as leading and trailing edges, in the mesh. The mesh reporting forms are also important to remove ambiguities regarding the mesh identification, used in the reporting for the computations.

3.4 Computation Reporting Form

To ensure that the correct procedure for the reporting of the computations was followed, a computation reporting form template was supplied to the partners. This template includes a section for each computation performed on the case, as well as a heading and a conclusion part. The heading section requires the user to list the code version applied as well as the flow conditions and convergence criteria used to confirm that the parameters defined by industry were applied. The computation sections includes issues such as the discretization scheme, the turbulence model and the solver settings that were used. In addition, the identification of the mesh used for the computation, with reference to the corresponding mesh report, was required. The computational time and number of iterations required for convergence as well as the fully converged values on this mesh were also to be reported. The conclusions section treats the convergence analysis, asking for the computed scheme order and discrete system size required for convergence. From this value, the partner was to fill in the total sequential time required for convergence, measured in ADIGMA performance index units.

4 Definition of Test Cases

A central part in the project was the definition of suitable test cases to be computed within the project for the industrial analysis. Due to the different maturity levels of the codes in the project, a balance between industrially relevant and more basic cases was sought. The importance of a few simpler cases involving the basic

ingredients required for a realistic industrial analysis, mainly turbulence and shocks, was recognized, allowing for detailed efficiency studies which were not likely to be performed on test cases of industrial complexity. The result of the deliberations were three test case suites, namely

- MTC: 2D test cases of airfoils for inviscid, laminar, inviscid-time dependent and turbulent flows
- BTC: 2D and 3D test cases of multi-element airfoils, streamlined bodies and wings
- CTC: 3D test cases of aircraft, complicated wing flows and multidisciplinary applications

The listing of each test case, with the defined convergence criteria, is given in the next chapter.

5 Inquiry of Technology Status

In order to judge the progress made by ADIGMA in a fair manner, the technology status, i.e. a summary of the state of the art of adaptive higher-order methods in Europe was compiled at the beginning of the project. This work was mainly conducted by the academic partners, providing references and short descriptions of their current experience and capabilities. This technology status was compiled in a common report, reflecting the present status of higher-order methods (residual-based and DG type schemes) and adaptive capabilities. This technology survey contains a detailed technical description of the codes used by partners during the ADIGMA project, treating both the existing reference codes and the methods to be further developed in the project. The survey outcome was organized in tables, detailing for every code which equations are solved (2-D/3-D, steady/unsteady, Euler/Navier-Stokes...), the discretization method employed (finite volume/element, residual based/DG, order of the approximation, cell vertex/centered...), the type of grid used (structured/unstructured, hybrid), whether the code has adaptation capabilities, and the solution strategy (implicit/explicit, parallel/sequential, multigrid acceleration...). A specific section contains the current experience and capabilities with respect to adaptive higher-order methods.

6 Conclusions

In this chapter, the definition of the criteria relevant for an industrial assessment of an approach, as well as the procedure to be followed to obtain these characteristics have been treated. The industrial criteria decided upon were the efficiency, applicability, robustness, ease of use, multidisciplinarity and implementational issues of a method. To enable a rigorous study of the efficiency, the reasoning behind, and description of, the method of asymptotic analysis were presented. This method allows for both the description of the potential a method has to reduce the discrete system size, as well as the potential for reducing the overall time it takes to obtain a

convergence to within a certain level of accuracy for a given size of available computational resources. To improve the quality of comparisons between computations performed on different computers, a machine–neutral performance index was defined. In addition, the unified reporting forms supplied to the project were described, allowing for an as unbiased and quantitative industrial evaluation as possible for the resources allocated to the subject within ADIGMA.

Chapter 33
Verification and Assessment

K.A. Sørensen and H. Bieler

Abstract. The results of the industrial evaluation of the ADIGMA project are presented. The analysis is primarily based on the computational results reported by the project partners, where an attempt was made to perform a rigorous comparison with the industrial baseline results to enable clear conclusions regarding criteria such as efficiency and robustness. A short discussion treating the quality of the results and lessons learned are also included.

1 Introduction

A central issue in the ADIGMA project has been the analysis of the developed approaches in an industrial setting, based on a series of computations performed on testcases of varying complexity. The testcases were split into three suites, namely the Mandatory Testcase Suite (MTC), the Baseline Test Case Suite (BTC) and the Complex Test Case Suite (CTC). A short description of the various test cases are listed in Tables 1- 3, together with the convergence criteria used for the analysis [1], as discussed in the previous chapter. The evaluations were conducted within a dedicated work package in the project, where subtasks were assigned with the industrial baseline generation, the mid-term assessment of the new methods, the final evaluation of the new methods and the technology transfer to industry. The synthesis of these results are treated in [2, 3, 4]. In addition, several other cases were considered in the various technology work packages, providing more focused information on specific issues such as shock capturing, adaptation and solution methods. The partners of the project were also given the opportunity to give their own opinions

K.A. Sørensen
EADS–MAS, Aerodynamics & Methods, Rechliner Str., 85077 Manching, Germany
e-mail: kaare.sorensen@eads.com

H. Bieler
AIRBUS Operations GmbH, Bremen, Germany
e-mail: heribert.bieler@airbus.com

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 465–482, 2010.
springerlink.com

Table 1 MTC suite test cases with industrial convergence criteria

Case	Geometry	Flow Conditions	Convergence criteria
1	NACA0012	$M = 0.5$, $\alpha = 2^\circ$, inviscid	$\lvert\varepsilon_{c_l}\rvert < 5 \cdot 10^{-3}$, $\lvert\varepsilon_{c_d}\rvert < 5 \cdot 10^{-4}$, $\lvert\varepsilon_{c_m}\rvert < 5 \cdot 10^{-4}$
2	NACA0012	$M = 0.8$, $\alpha = 1.25^\circ$, inviscid	$\lvert\varepsilon_{c_l}\rvert < 5 \cdot 10^{-3}$, $\lvert\varepsilon_{c_d}\rvert < 5 \cdot 10^{-4}$, $\lvert\varepsilon_{c_m}\rvert < 5 \cdot 10^{-4}$
3	NACA0012	$M = 0.5$, $\alpha = 2^\circ$, $Re = 5000$	$\lvert\varepsilon_{c_l}\rvert < 5 \cdot 10^{-3}$, $\lvert\varepsilon_{c_d}\rvert < 5 \cdot 10^{-4}$, $\lvert\varepsilon_{c_m}\rvert < 5 \cdot 10^{-4}$
3s	NACA0012	$M = 0.5$, $\alpha = 2^\circ$, $Re = 500$	$\lvert\varepsilon_{c_l}\rvert < 5 \cdot 10^{-3}$, $\lvert\varepsilon_{c_d}\rvert < 5 \cdot 10^{-4}$, $\lvert\varepsilon_{c_m}\rvert < 5 \cdot 10^{-4}$
4	NACA0012	$M = 0.755$, $\bar{\alpha} = 0.016^\circ$, inviscid, $k = 0.1628$, $\Delta\alpha = 2.51^\circ$	$\lvert\varepsilon_{c_l}\rvert < 5 \cdot 10^{-3}$, $\lvert\varepsilon_{c_d}\rvert < 5 \cdot 10^{-4}$, $\lvert\varepsilon_{c_m}\rvert < 5 \cdot 10^{-4}$
5	RAE2822	$M = 0.73$, $\alpha = 3.19^\circ$, $Re = 6.5 \cdot 10^6$	$\lvert\varepsilon_{c_l}\rvert < 2 \cdot 10^{-2}$, $\lvert\varepsilon_{c_d}\rvert < 5 \cdot 10^{-4}$, $\lvert\varepsilon_{c_m}\rvert < 5 \cdot 10^{-3}$

Table 2 BTC suite test cases with industrial convergence criteria

Case	Geometry	Flow Conditions	Convergence criteria
0a	Analytic Streamlined Body	$M = 0.5$, $\alpha = 1^\circ$, inviscid	$\lvert\varepsilon_{c_l}\rvert < 1 \cdot 10^{-3}$, $\lvert\varepsilon_{c_d}\rvert < 3 \cdot 10^{-4}$, $\lvert\varepsilon_{c_m}\rvert < 5 \cdot 10^{-4}$
0b	Analytic Streamlined Body	$M = 0.5$, $\alpha = 1^\circ$, $Re = 5000$	$\lvert\varepsilon_{c_l}\rvert < 1 \cdot 10^{-3}$, $\lvert\varepsilon_{c_d}\rvert < 3 \cdot 10^{-4}$, $\lvert\varepsilon_{c_m}\rvert < 5 \cdot 10^{-4}$
0c	Analytic Streamlined Body	$M = 0.5$, $\alpha = 5^\circ$, $Re = 1.0 \cdot 10^7$	$\lvert\varepsilon_{c_l}\rvert < 1 \cdot 10^{-3}$, $\lvert\varepsilon_{c_d}\rvert < 3 \cdot 10^{-4}$, $\lvert\varepsilon_{c_m}\rvert < 5 \cdot 10^{-4}$
1	L1T2 3 Element Airfoil	$M = 0.197$, $\alpha = 20.18^\circ$, $Re = 3.52 \cdot 10^6$	$\lvert\varepsilon_{c_l}\rvert < 1 \cdot 10^{-2}$, $\lvert\varepsilon_{c_d}\rvert < 1 \cdot 10^{-3}$, $\lvert\varepsilon_{c_m}\rvert < 1 \cdot 10^{-3}$
2	ONERA M6	$M = 0.84$, $\alpha = 3.06^\circ$, $Re = 11.72 \cdot 10^6$	$\lvert\varepsilon_{c_l}\rvert < 1 \cdot 10^{-2}$, $\lvert\varepsilon_{c_d}\rvert < 1 \cdot 10^{-3}$, $\lvert\varepsilon_{c_m}\rvert < 1 \cdot 10^{-3}$
3	Delta Wing	$M = 0.3$, $\alpha = 12.5^\circ$, $Re = 4 \cdot 10^4$	$\lvert\varepsilon_{c_l}\rvert < 1 \cdot 10^{-2}$, $\lvert\varepsilon_{c_d}\rvert < 1 \cdot 10^{-3}$, $\lvert\varepsilon_{c_m}\rvert < 1 \cdot 10^{-3}$
4	DPW III Wing1	$M = 0.76$, $\alpha = 0.5^\circ$, $Re = 5 \cdot 10^6$	$\lvert\varepsilon_{c_l}\rvert < 1 \cdot 10^{-2}$, $\lvert\varepsilon_{c_d}\rvert < 1 \cdot 10^{-3}$, $\lvert\varepsilon_{c_m}\rvert < 1 \cdot 10^{-3}$

Table 3 CTC suite test cases with industrial convergence criteria

Case	Geometry	Flow Conditions	Convergence criteria
1	M219 Cavity	$M = 0.85$, $Re = 6.84 \cdot 10^6$	none defined
2a	ALA SMJ config	$M = 0.8$, $C_L = 0.45$, inviscid	$\lvert\varepsilon_{c_l}\rvert < 1 \cdot 10^{-5}$, $\lvert\varepsilon_{c_d}\rvert < 1 \cdot 10^{-4}$, $\lvert\varepsilon_{c_m}\rvert < 2 \cdot 10^{-4}$
2b	ALA SMJ config	$M = 0.8$, $C_L = 0.45$, $Re = 3.0 \cdot 10^7$	$\lvert\varepsilon_{c_l}\rvert < 1 \cdot 10^{-4}$, $\lvert\varepsilon_{c_d}\rvert < 1 \cdot 10^{-3}$, $\lvert\varepsilon_{c_m}\rvert < 2 \cdot 10^{-3}$
3	VFE2 Delta, medium rad LE	$M = 0.869$, $\alpha = 24.7^\circ$, $Re = 5.95 \cdot 10^7$	$\lvert\varepsilon_{c_l}\rvert < 1 \cdot 10^{-2}$, $\lvert\varepsilon_{c_d}\rvert < 5 \cdot 10^{-3}$, $\lvert\varepsilon_{c_m}\rvert < 2 \cdot 10^{-2}$
4	DLR F6	$M = 0.3$, $\alpha = 12.5^\circ$, $Re = 4 \cdot 10^4$	$\lvert\varepsilon_{c_l}\rvert < 1 \cdot 10^{-2}$, $\lvert\varepsilon_{c_d}\rvert < 5 \cdot 10^{-3}$, $\lvert\varepsilon_{c_m}\rvert < 2 \cdot 10^{-2}$
5	Helishape DP 135 rotor	$M = 0.76$, $\alpha = 0.5^\circ$, $Re = 5 \cdot 10^6$	none defined

of the potential of their codes, using a dedicated questionnaire [5]. The four major development directions are treated in separate sections in the following, the first two focusing on different discretization families and the last two looking into the subjects of solver methods and adaptation with error analysis.

2 Discontinous Space Discretization Methods

A large part of the research in ADIGMA has focused on higher order discretization approaches where the variables are allowed to be discontinous on interfaces between elements, the so called Discontinous Galerkin (DG) methods. The approach has reached a level of maturity which is arguably the highest of the non-industrially based methods, with three partners (UNBG, DLR,UNST) having codes which include turbulence modeling and the application of complex 3D configurations. The codes from UNBG and UNST are also capable of solving flows with shocks, where the latter is optimized for time-accurate computations. In addition, the partners UNNO, UNUP, UNPR, NJU, UNTW, NLR, ONERA and CENAERO all have codes in the DG family, representing a large collection of different variations of this approach.

For inviscid 2D subsonic flow (MTC1) the DG family of methods show a potential of reducing the number of DOFs for convergence with around a factor of 3, compared with the industrial baseline (IB), Fig. 1. Some partners show results which are closer to the IB. Even though the testcase is not of highest industrial relevance, it supports the assumption that higher order discretizations are efficient in solving

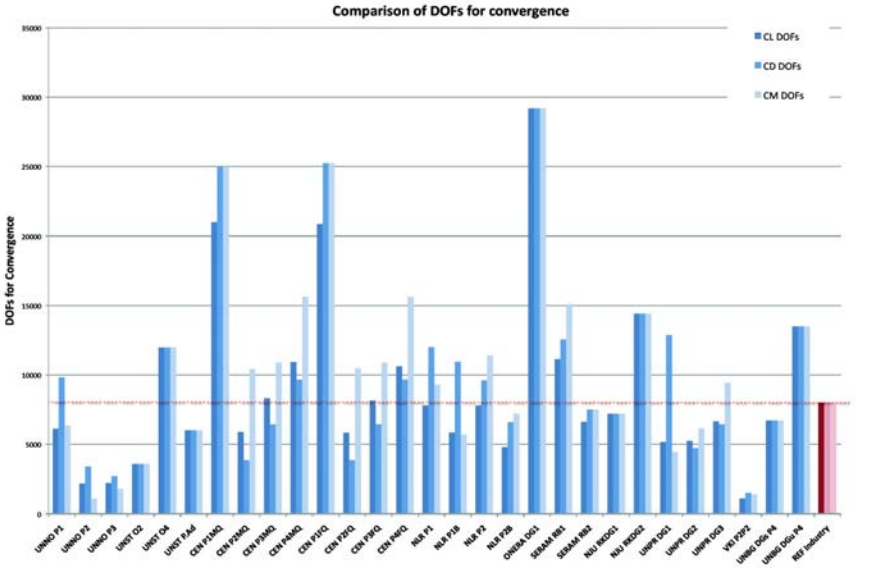


Fig. 1 Illustration of the number of DOFs per equation required for mesh convergence for various discretization schemes and the industrial standard for the MTC1 testcase

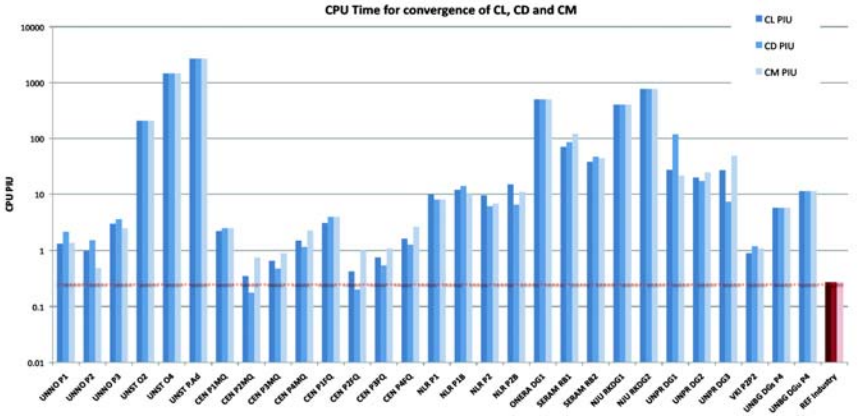


Fig. 2 Illustration of the equivalent sequential wall clock time (in ADIGMA performance index units) required to obtain a converged result for the MTC1 testcase

flow regions where no discontinuities are present and the inviscid components dominate. The solution times required for convergence are however consistently higher than the IB, Fig. 2. This is expected to be partly caused by the relatively low level of maturity in the DG solvers used, but also indicates that higher order numerical systems are stiffer to solve. This supports the assumption that, for direct global mesh refinement studies, the DOF reduction obtained by DG methods is a likely estimate of the best CPU time decrease that can be realistically hoped for. The memory usage of the DG schemes for this testcase vary a lot, but it appears that for the current solution methods a memory increase of a factor of more than 5 compared with the IB seems to be typical, Fig. 3.

The MTC2 testcase features inviscid transonic flow and was introduced to analyze the behaviour of the methods in the presence of shocks. The IB shows a threefold increase in the mesh size for convergence for this testcase. The required increase in DOFs for the DG methods seems to be roughly the same figure, if not less, for many codes. This is a very interesting result, indicating that shock-capturing is not as big a problem as is often assumed, at least by increasing the schemes to third order. The result is that some DG methods outperform the IB with a factor of typically around 3-4 w.r.t number of DOFs. The memory usage does not, as expected, seem to be influenced by the introduction of shocks in the solution field.

The MTC3 testcase is a subsonic laminar flow testcase of a NACA0012 airfoil at two degrees angle of attack and a Reynolds number of 5000. This testcase has been quite controversial as in the industrial baseline computations a separation bubble occurred for some partners, resulting in very large mesh requirements to converge to within the specified tolerances. This led to the introduction of another testcase, the MTC3s, with a Reynolds number of 500 and symmetric flow. Some partners also computed the case with the lower Reynolds number, but with a two degree angle of attack. Only the MTC3s results show any consistency, unfortunately no IB results are available for this testcase. The higher order results made available are however

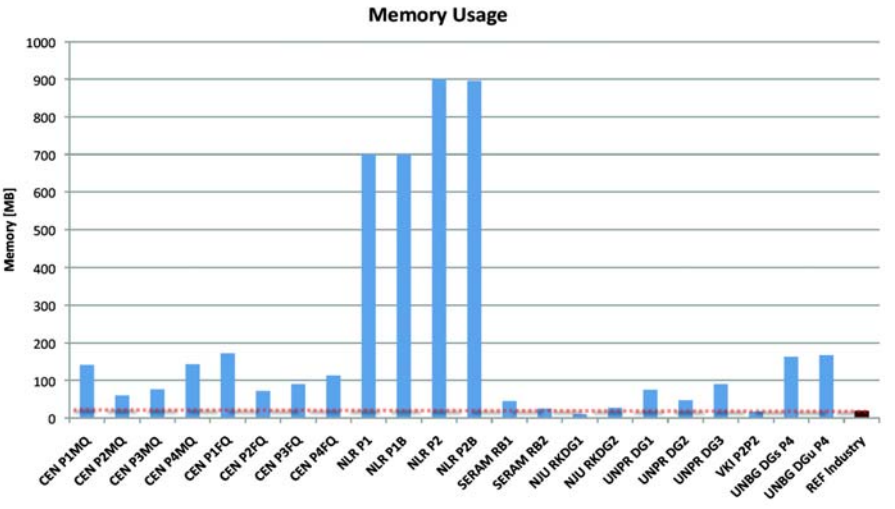


Fig. 3 Illustration of the memory usage required for a converged solution for the MTC1 test case

interesting, with the results from UNNO showing that the DG(2) discretization has better results than both the DG(1) and the DG(3), outperforming the DG(1) results with a factor of around 5 w.r.t. the discrete system size.

For the time-accurate MTC4 testcase, only one result was published, namely that of NLR. Here an attempt of a rigorous comparison with the IB was made using both global as well as adaptive mesh refinement strategies. Due to the time-space discretization used by the partner, and the non-nested meshes resulting from resolution differences in the time dimension, the analysis is not straightforward. The global refinement strategy appears to result in a time-space resolution of in the order of 10^8 DOFs per equation, the IB requires in the order of $5 \cdot 10^6$ DOFs per equation. There thus does not seem to be an advantage in the (non-adaptive) time-space approach for this testcase. It must be noted that basis functions of first order where applied, the nominal order of accuracy is thus the same as for the industrial standard. It thus appears that there is no direct gain in the time-space solution scheme for this particular application, but this picture might change if higher order elements are applied. In addition, the method stores a four dimensional mesh, resulting in very large memory requirements.

The MTC5 testcase is a transonic turbulent testcase and thus considered very important in the industrial evaluation of the new approaches in ADIGMA. Unfortunately, only two DG results are available for the analysis, namely those from UNBG and ONERA, both of which are not of the quality to enable rigorous performance studies. As a result, even though the industrial relevance of this case was repeatedly stated during the project, no efficiency statements for the DG method on this case are available.

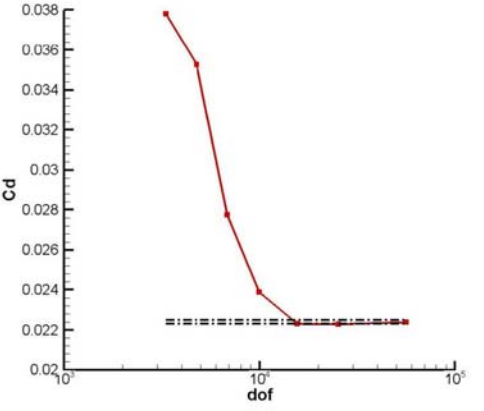


Fig. 4 Mesh convergence for the MTC2 testcase as reported by SERAM

Another important testcase is the BTC0, a streamlined body explicitly defined in ADIGMA for 3D analysis purposes, Fig. 5. Inviscid, laminar and turbulent testcases were defined. For the turbulent testcase, the unstructured computations performed by EADS-MAS showed a very bad mesh convergence behaviour compared with the structured grid results of DLR, where the unstructured results require around one order of magnitude more DOFs to converge, both using the DLR Tau code. This may be caused by problems related to the EADS-MAS unstructured meshes for the Tau code, indeed for the laminar case it was not possible to generate a satisfactory solution on these meshes, but the figures do not seem to be unrealistic considering the convergence criteria, for the drag coefficient roughly representing a value of 4%. The structured mesh results are however very good, even showing convergence rates much higher than the theoretical limit. Due to this large difference in results, it is

$$\begin{cases} 16\left(x-\frac{1}{4}\right)^2+400z^2=1, & x\in\left[0,\frac{1}{3}\right] \\ z=\frac{1}{10\sqrt{2}}(1-x), & x\in\left(\frac{1}{3},1\right],\ z>0 \\ z=-\frac{1}{10\sqrt{2}}(1-x), & x\in\left(\frac{1}{3},1\right],\ z<0 \end{cases} \quad y\in\left[0,\frac{1}{100}\right]$$

$$\begin{cases} 16\left(x-\frac{1}{4}\right)^2+400\left(z^2+\left(y-\frac{1}{100}\right)^2\right)=1, & x\in\left[0,\frac{1}{3}\right] \\ 200\left(z^2+\left(y-\frac{1}{100}\right)^2\right)-(1-x)^2=0, & x\in\left(\frac{1}{3},1\right] \end{cases} \quad y>\frac{1}{100}$$

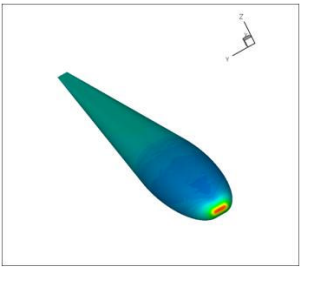


Fig. 5 The definition of a half-model BTC0 streamlined body testcase. This case was introduced to enable a simple 3D geometry for asymptotic mesh convergence studies, offering the possibility of analytic higher order meshes. By enlarging the central extrusion section, a primitive wing can be constructed.

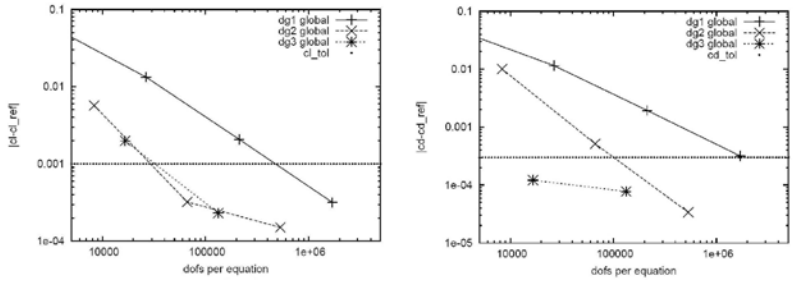


Fig. 6 Mesh convergence plots for the turbulent BTC0 case from DLR DG computations for different discretization orders, showing considerable reduction in problem size when using higher order methods

better to also discuss the structured and unstructured IB results separately. For the DG method, only structured results, from DLR, were supplied, Fig. 6. The computations show a decrease in the DOFs per equation of a factor of about two for DG(3), where DG(1) performs much worse than the IB for structured meshes. The situation is thus not very clear, but the possibility of the higher order methods to converge on meshes significantly smaller, say a factor around 2-3, seems to be possible for the higher order DG methods for 3D turbulent subsonic flow. The DG(1) results, which fit the unstructured IB results quite well also increase the suspicion that the structured IB results are, for some reason, unrealistically good. The computational time required for convergence is considerably slower than the industrial baseline, but this may very well be caused by the immaturity of the DG solver, as was also noted by DLR in their report. For the inviscid testcase, the results surprisingly indicate that the DG methods do not represent an improvement compared to the IB, computed by VKI. The inviscid BTC0 could be considered a 3D generalization of the MTC1 case, and one should expect at least a factor of 3 reduction in the DOFs per equation for convergence. Since more MTC1 data is available and the analysis shows a clearer trend, the inviscid BTC0 data are not given much weight in the conclusions. The laminar version of the BTC0 case is somewhat problematic since the IB results are not rigorous and feature integral values which are completely unrealistic. Ignoring the IB, and assuming that the DLR DG(1) DOF values are indicative for 2nd order FV codes, the speedup factor between DG(1) and DG(2) is more than 10, a very impressive value. For this testcase, UNPR also show results, unfortunately the integral values are one order of magnitude smaller than the reference, this is probably caused by a bug in the code. It must be noted that the unstructured meshes supplied by EADS-MAS proved a major challenge to the industrial Tau code, in effect no consistent analysis was possible. This was not reported a problem for the DG code used by DLR, showing that DG methods may have a stability advantage compared with the current industrial standard for meshes of reduced quality, something that has been demonstrated in the project for other testcases as well. It can be concluded that this 3D testcase has shown a clear potential for the DG methods with regard to the number of DOFs required to converge. It must however also be noted that this

# el.	# DoFs	$C_{\mathrm{d}}^{\mathrm{ref}} - C_{\mathrm{d}}$	$\mathcal{E}$	θ
6656	186368	-1.148e-02	-1.080e-02	0.94
16637	465668	-1.943e-03	-1.924e-03	0.99
41320	1156960	-4.497e-04	-4.263e-04	0.95
102087	2858436	-2.022e-04	-2.022e-04	1.00

Fig. 7 Results of the DLR adaptation study on the BTC0 testcase

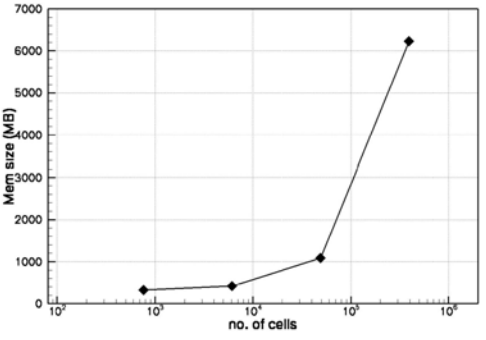


Fig. 8 Illustration of the memory usage of a the third order DG scheme of NLR showing the non-linear increase with the discrete problem on the laminar BTC0 testcase

testcase is characterized by a very simple flowfield and that all major flow structures appear in the solution of very coarse meshes. For more industrially relevant cases, involving separation, vortex burst, shocks etc. it is often the case that a certain level of refinement, often relatively fine, has to be obtained to enable all essential flow features to even appear in the solution. It is thus not given that the promising results, at least w.r.t. number of DOFs for convergence, are directly transferrable to more industrially relevant problems. The non–linear memory dependence of some higher order methods is illustrated in Fig. 8.

The L1T2 three-element airfoil of BTC1 was computed by DLR. There is some dispute regarding the quality of the IB analysis for this case. Taking the DG analysis for various discretization orders, the increase up to fifth order seems to reduce the number of DOFs by one order of magnitude for this case. The speedup obtained for the fifth order computation compared with the second order DG method is around a factor of 6. The cost for convergence is apparently not quite linearly dependent on the number of DOFs when increasing the order. Since no real convergence analysis was performed using IB codes, the significance of these values is unclear.

For the BTC2 testcase, a laminar delta wing configuration is considered. The IB work was performed by NLR, resulting in a requirement of around ten million DOFs per equation for convergence, with the drag requirements being the most critical. For lift and pitching moment, a mesh size requirement of magnitude one million is required. Comparing with the DLR PADGE results on the same structured meshes, there appears to be an inconsistency since for DG(1), all values are converged for

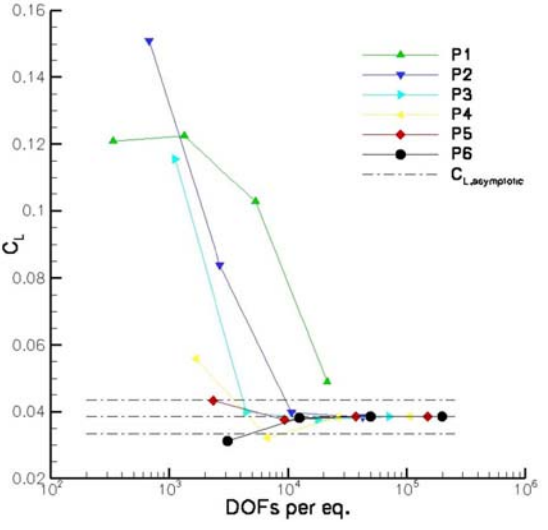


Fig. 9 Mesh convergence of the MTC3 case, computed by UNBG

less than one million DOFs per equation. It is probably unlikely that the second order DG method is ten times more efficient than the IB, the drag convergence requirements for the NLR computation is thus not given much weight in the comparison. Still, the DLR results show a significant improvement in the discrete system size by increasing the order to third (300 000 DOFs per equation) and fourth (100 000 DOFs per equation). This is a very promising result and in line with the general observation that higher order methods are very efficient for vortex-dominated flows. Solving the higher order systems proved to be much more difficult however, resulting in convergence times twice that of the second order method. Again, this can be a result of suboptimal solution strategies, as is also noted by the partner. Another important issue raised in the computational report is the dependency of the higher order speedup on the convergence criteria. Even though it is thought that the defined convergence levels reflect the industrial standard, an increased accuracy level would shift the higher order advantages with respect to the size of the discrete system considerably. The testcase is important in that thorough analyses have been performed and that the DG method shows potential of significant improvements for vortical flows. Also UNPR have contributed for this testcase, using the COOLFluid platform for a second order discretization. Again, there seems to be a problem with the coefficient evaluation, yielding values that are one order of magnitude too small. This results in convergence criteria that are too lenient, the approach used applying the finest mesh (with only 300000 DOFs per equation) as the asymptotic solution is thus not accurate enough. Still, the report shows that a computation can be performed for this case using the COOLFluid platform. The memory usage is however about 20 times larger than the IB baseline for the same number of DOFs per equation, a problem that should be addressed.

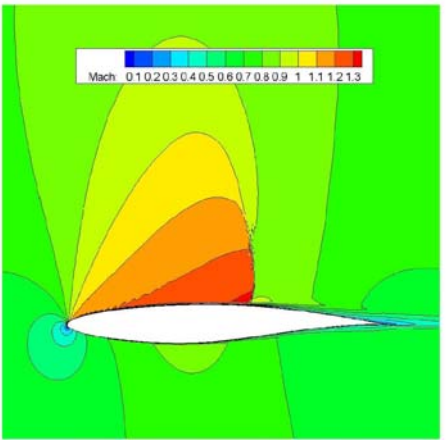


Fig. 10 Illustration of the Mach field computed by UNBG on the MTC5 test-case using a Discontinous Galerkin approach

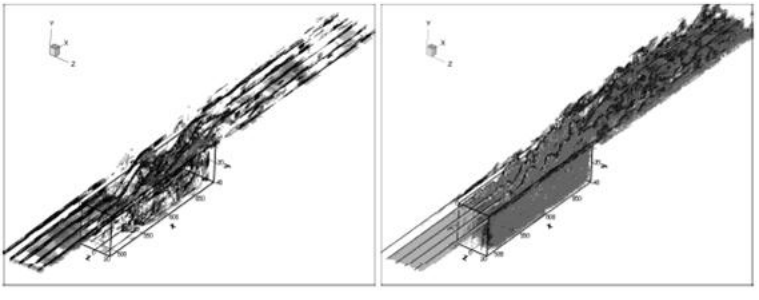


Fig. 11 Illustration of the UNST CTC1 computation

The CTC cases are normally too complex for rigid performance studies, they are included in the project to shed light upon other criteria such as applicability and robustness. There have been a few DG computations performed in the CTC suite, in general showing good behaviour of the methods. The CTC1 testcase, the unsteady cavity, Fig. 11, was computed by UNST, using higher order discretization schemes. The computations were performed with an LES approach for turbulence. The use of the explicit timestepping scheme for solving these types of problems is considered suboptimal. For the CTC2 testcase there were no contributions. For the CTC3 testcase, the turbulent high AoA delta wing, UNBG have showed some results, Fig. 12, illustrating the feasibility, albeit with some difficulties, of solving such problems with the DG methods. The CTC4 testcase, an F6 configuration, was solved by both UNBG and DLR. Even though neither the IB nor the DG results were rigorously analysed, it can be concluded that the use of higher order elements have the potential of significantly reducing the discrete system size, Fig. 13. The CTC5 testcase was performed by NLR, showing the potential of the DG method for multidisciplinary applications involving the simulation of elastic rotor blades.

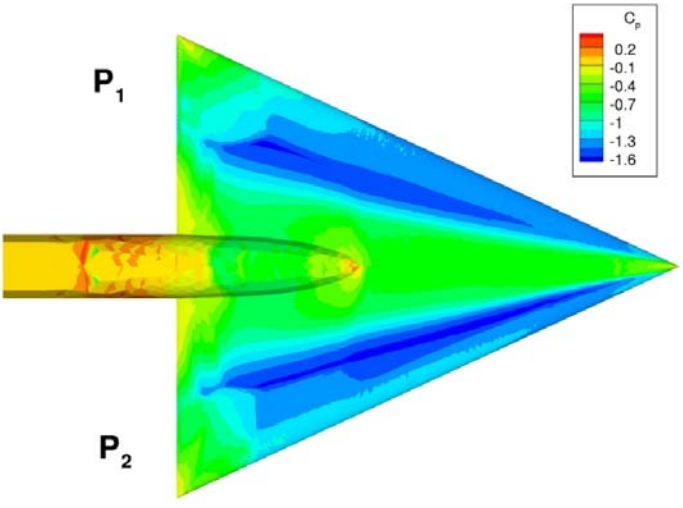


Fig. 12 Pressure coefficient plot comparison for the UNBG CTC4 computations for second and third order discretizations

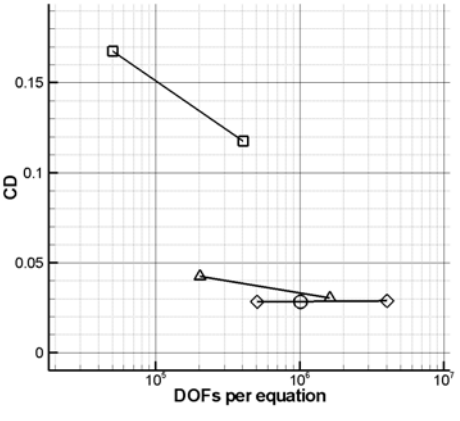


Fig. 13 Positive effects of higher order methods on the drag for the CTC4 case from the UNBG results. The squares, triangles, diamonds and circles denote the results for the first, second, third and fourth order DG schemes respectively.

Almost all reported computations were performed with standard solver settings. The DG codes thus seem to be easy to use within their respective area of application. A trend can be observed from the results, pointing to a potential reduction in the numerical system size. Typically, it appears that one can hope for a reduction factor of between 3 and 10 for the DG methods compared with the IB. It has however become clear that these figures are strongly dependent on the convergence

criteria. Unfortunately, there are however also indications that the resulting numerical system may be considerably more challenging to solve than for the current state of the art discretization schemes, the partners using the DG approach have thus not been able to prove a performance improvement stemming from the discretization only. Even though this may change as the solvers are further developed, there are no clear indications available that this will be successful, currently it can be said that only very few of the codes come close to the performance of the IB, let alone improving them. This issue thus has to be clarified before statements of any quality can be made on the potential of the DG method for industrial use.

Seven partners, UNST, UNTW, NJU, NLR, Cenaero, UNPR and DLR, have submitted the industrial potential questionnaire for the DG approach. Most partners cite the improvement potential with regard to efficiency as a major advantage of the DG approach. The favourable performance of higher order methods for vortical flows is pointed out as well as the expected performance increase for more stringent convergence criteria. Regarding robustness, there are different opinions whether the DG approach can be expected to run in a more stable manner than industrial codes, some claim that the insensitivity to bad meshes is a distinct advantage of the method, others cite from experience that the codes still exhibit convergence problems for complex flows. The importance of proper higher order mesh generation to enable the full advantages of the DG approach is also underlined in the questionnaires. For time-accurate computations with changing boundaries as well as for multidisciplinary problems, the time-space DG approach is claimed to be superior to the current state of the art. The usage of adaptation and the adjoint, with error estimation, is considered to be of critical importance by some partners. Most partners mention the solver as the weakest part of the code, requiring further development to enable efficient computations. The need for robust turbulence model implementations is also pointed out in the forms. Most partners expect to have a competitive code for 3D turbulent computations within the 3-10 year timespan.

3 Continuous Space Discretization Methods

For approaches where the representation of the unknowns is continuous on element interfaces, there are several approaches considered in ADIGMA. These continuous space methods are referred to as CS methods in the following. One approach is the Residual-Based Compact (RBC) schemes proposed by SERAM, introducing compact stencils for multidimensional upwind schemes reaching higher order at steady-state. Another method considered in the project is the Continuous Residual Based (CRB) approach, described as a mixture of finite volume and finite element approaches. This approach is developed or used in ADIGMA by the partners VKI, INRIA and UNWA. Also operating on a continuous basis is the Galerkin Least Squares (GLS) method developed by DASSAV, where the partner has based the developments on the in-house second order industrial code of the same approach. Unfortunately, the GLS computations reported were few in number and arrived at a very late stage of the project. Another continuous discretization scheme directly

implemented in codes of industrial standard is the approach investigated by UWS and EADS-MAS, attempting a higher order FD-like approach in the wall-normal direction in boundary layers.

For MTC1, most continuous space methods show little or no gain in the discrete problem size compared with the IB. The exceptions to this are the computations performed by VKI and DASSAV, which show a reduction factor of around 5 in the DOFs per equation needed. It must be noted that the convergence criteria are not very strict for this testcase. As for the DG methods, the convergence times and the memory usage are in general considerably higher than for the industrial codes. The exception to this is the higher order approach of DASSAV which shows a reduction factor of around 5 not only in the discrete problem size, but also in computational time required for convergence. This is a very interesting result, indicating that for this method, the discrete system stiffness reported for almost all other global higher order methods, is not present.

For the transonic inviscid flow case (MTC2), the continuous space methods from VKI and SERAM show a reduction in the discrete system size for convergence, Fig. 4. In the case of VKI, this reduction is dramatic, reaching a factor of 25. The partner does however point out that this reduction in discrete system size comes with a computational price. Again, as for the DG methods, these results show a surprising indifference of the higher order methods to the presence of shocks in the flowfield. The DASSAV results show similar impressive performance as for the MTC1 testcase.

For the MTC3s case, the results from INRIA and SERAM show a similar performance to the DG methods. The SERAM RB1 results are reported to be slightly better than the RB2 data. This effect was also observed for the DG approach. It thus seems that for this particular test case, there is little effect in increasing the discretization order beyond a value of around 3 with regard to the number of DOFs per equation. It is also observed that the higher order discretizations are more expensive to solve, per DOF, and that also the memory usage increases disproportionately with the scheme order. Again, the DASSAV computations show a significant speedup, measured in wall clock time, compared with the industrial baseline.

For the MTC4 testcase, no results were published for the higher order continuous methods. The MTC5 testcase has contributions from UWS, SERAM and DASSAV. UNSW, using the higher order boundary layer discretization, show an improvement compared with a standard boundary layer approach of a factor of around 1.4. The results from SERAM are inconclusive with regard to efficiency issues but shows a capability of the code for solving 2D turbulent transonic flows in a stable manner. The results from DASSAV were computed with higher order for all equations except for the turbulence field, for which a solution on a finer mesh were interpolated. This approach was chosen since the turbulence equation, which is treated differently in the code, has not yet received a higher order discretization. The result from the computation shows a decrease in the required discrete system size of around a factor of five. In addition, no stiffness problems are reported, resulting in a wall-clock speedup of a similar factor.

For the inviscid BTC0, 3D streamlined body flow, VKI and SERAM have reported. The mesh convergence results are however inconclusive, the data available do however indicate that the gain obtained by the use or higher order methods falls within the accuracy of the evaluation methods. It is thus not possible to identify improvements associated with the CS methods for this testcase. For the turbulent BTC0 case EADS-MAS shows results for the directional higher-order boundary layer discretization scheme. The studies conclude that the number of boundary layer levels can be reduced by around a factor of 1.6, resulting in a reduction of around 20-30% in the size of the numerical system compared with the IB.

The only CTC case computed by a higher order continuous approach is the CTC3 test case, the turbulent delta wing, which was treated by EADS-MAS illustrating the robustness of the higher order directional boundary layer discretization for a complex test case but not showing mesh convergence studies.

The continuous methods are in general observed to be in a less developed state than many of the DG implementations. In particular, there are not many CS codes capable of solving turbulent problems, making the analysis of the industrial applicability of the methods difficult. For the cases where results do exist, the globally higher order schemes mostly show similar performance, with respect to discrete system size, as the DG approach. The localized higher order approach of UWS and EADS-MAS show a small, typically 20-30%, but consistent, improvement compared with the IB, a major advantage of this method is however the relative ease of implementation into existing industrial codes. The results reported by DASSAV however show a much larger potential in the reduction of the both the discrete system size, but more importantly the convergence times. Due to the immaturity of these results, particularly for turbulent flows, there is some uncertainty whether these values can be realized for applications of industrial relevance. The method does however seem very promising.

Unfortunately, for CS methods only 3 partners, INRIA, EADS-MAS and UWS, have submitted the code industrial potential questionnaire. INRIA reports the issue of efficiency as the most critical issue with the residual distribution schemes, and since turbulence modeling is not yet available, it is difficult to assess the software for realistic industrial cases. The boundary layer developments from UWS and EADS-MAS have been applied to turbulent flow problems where the boundary layer plays an important role and are believed to represent an improvement of around 30% to existing industrial codes for such cases in the current form. Work still needs to be performed in the treatment of the turbulence models which might improve the results. Since the experience base is limited to only a few testcases, the statements are considered to be of medium certainty.

4 Advanced Solver Methods

The efficient solution of the discrete system of equations is an important subject and thus also under consideration in the ADIGMA project. Often, the solution scheme is strongly dependent on the discretization approach in use, it is thus not always

possible to evaluate a solution method without also taking into account the characteristics of the discretization scheme used. For many DG and CS approaches, the preferred solution method to date has been a Newthon-Rhapson type approach, with the linear subsystems solved either directly or in an iterative fashion. This method is suitable for the usually very stiff discrete problems arising from the higher order discretizations, but is usually considerably slower than the methods used for industrial codes and normally has much larger memory requirements. The partners UNTW, UNBG, ONERA, NLR and UNUP have been considering the usage of nonlinear multigrid approaches as solvers for high-order problems. The improvement of implicit approaches has been treated by UNBG and UNPR. For parallelization and other software issues, the partners INRIA, NLR, VKI, UNBG and UNWA have contributed. Other approaches have been considered by EADS-MAS, ONERA, UWS and UNST.

UNTW has considered methods for improving multigrid convergence. Unfortunately only results for the Euler equations are presented, where a reduction in the number of cycles for convergence is achieved for isentropic meshes. This does however not in the current implementation translate to reduced computational times as the cycles are more expensive. For stretched meshes, the performance is not improved w.r.t the number of cycles compared with the baseline code. UNBG has considered the usage of p-multigrid strategies, showing promising results for inviscid flow, where a small reduction in computational time is obtained compared with implicit solvers. The memory usage of this spectral p-multigrid scheme is however not considerably decreased. The performance of the scheme for transonic cases was not considered, the results for stretched meshes seem to indicate that the approach experiences the same stiffness problems as traditional multigrid schemes. The partner has also investigated the use of h-multigrid, resulting in considerable speed-up factors, even for viscous problems. The code is however new and the behaviour needs to be further analysed. The results from ONERA also show the potential of the multigrid method for higher order discretizations, where indications of improved performance are shown, albeit with some oscillations. In general, it can be said that there have been relatively little focus on the multigrid approach in ADIGMA which is unfortunate since this is a central technology in the attempt of increasing the performance of the higher order codes. For implicit solvers, some analysis regarding preconditioning and the solution of the linear subproblems have been conducted by UNBG and UNPR. These results show potential of improving the cost of the implicit solver somewhat, the memory usage is still a critical issue. Regarding parallelization, the results from INRIA, VKI, UNBG and UNWA show that the new approaches scale in an efficient manner.

The EADS-MAS/UWS approach of enhancing the multigrid strategy by using a line-implicit preconditioner has shown itself to be successful in cases where the boundary layer has a significant influence on the convergence indicators, typically yielding a speedup compared with the industrial standard of a factor between 2 and 3, as was seen in the MTC5 and BTC0 testcases. This comparison can be considered

of being of relatively high resolution since the same code forms a basis for both approaches. The preconditioner seems to work equally well for both lower and higher order discretizations, but if the number of layers required in the boundary layer is significantly reduced, it is expected that the approach becomes less effective simply because the stiffness in the boundary layer is reduced and the basic multigrid solver works more efficiently. It is believed that there is potential to further increase the efficiency of the preconditioner, the theoretical limit seems to be around a factor two larger than what has been obtained so far, approaching, but not quite reaching, typical Euler convergence rates. For cases where the boundary layer does not play an important role for the integral values, such as for the CTC3 turbulent delta wing case where a vortex is generated at the leading edge, the influence of the approach is reported to be negligible.

In general, the focus on the solution of the discrete system has been small in ADIGMA, resulting in uncertainties regarding the overall performance of the schemes. Looking at the published results, the impression remains that the higher order methods have, for most approaches, complicated the task of the solvers considerably and that this could prove to be a major problem in the industrialization of the higher order codes.

5 Adaptation and Error Estimation Strategies

Higher order methods introduce the possibility of locally adapting the discretization order of the scheme, p-adaptation, in addition to the h-adaptation normally used for industrial solvers. As for industrial codes, this can be performed on a local basis, as a function of the gradients or residuals, or using some form of error analysis on a global basis, usually through the computation of the adjoint problem. The advantage of these approaches is the potential of reducing the numerical system size for a given level of accuracy. The disadvantage of the approach is that, for h-refinement, new meshes need to be generated, or modified, and that significant computational time is often needed for analysing the current discretization, e.g. by computing the adjoint. These factors have to be taken into account for the overall performance analysis of the approaches. Contributors to the discipline in the ADIGMA projects were DLR, A-F, DASSAV, ARA, NLR, UNNO, UNPR, UNST and UNWA, the large number of participants underlining the importance of adaptation in the project.

The NLR CTC5 results for vortex flow show the positive effect of feature-based adaptation with pre-input where the user can exclude regions known a-priori not to require higher resolution for the problem in question. For the MTC4 case, the adaptation procedures applied show a tendency towards an asymptotic solution, the results are however not in a form allowing for a statement of the efficiency of the approaches. UNPR shows the application of using second order derivatives of primarily density and Mach number for refining shocks. UNWA shows the usage of anisotropic adaptation involving remeshing, where a mesh reduction factor

of around 10 is reported. This approach does however involve a strongly coupled interaction between solver and mesh generator, the cost of which must be further analysed. A-F shows results of mesh adaptation and deformation using the adjoint solution for 2nd order multiblock schemes, also resulting in error estimates. UNST has demonstrated the usage of h-p adaptation for the MTC1 and MTC3 testcases. The UNNO and DLR results on error estimation and goal oriented adaptation show the potential accuracy inherent in the adjoint, allowing for local and global error estimates for target functions. This method has the disadvantage of requiring a dual computation for each target function. By solving for the discretization error, the adjoint-adjoint problem, a single adjoint solution suffices for the entire error field description. Since this method involves two subsequent approximation steps, the accuracy level is somewhat reduced however. The methods was also not applied to problems of industrial complexity in ADIGMA. For the BTC2 testcase, UNWA have demonstrated the use of feature-based anisotropic mesh adaptation. For the BTC0 and BTC3 cases, the DLR results show convergence with a reduction factor of around 2-4 for the number of DOFs for convergence compared with global refinement. For the laminar BTC0 case, DLR show impressive DG(1) error analysis results, where the active usage of the error estimation in conjunction with the adjoint adaptation allows for a mesh-converged drag DOF reduction factor of over 5, Fig. 7. If the same factor would be realizable for DG(2), a combined speedup factor of third order and error analysis compared with the second order results of over 50 would result. The DLR turbulent BTC0 case shows similar, or even better results for adjoint refinement and error estimation, Fig. 5. For this case it is illustrated, taking the computational time required for the adjoint computation into account, a speedup compared with the standard DG(1) scheme of over 3 is obtained. It must however be noted that for this figure each integral value was considered separately. As previously noted, it is more usual to focus on all integral values, typically 6 in 3D, for the computations. For the CTC4 test case, several refinement strategies are attempted by DLR. The general conclusion is not clear, but there are no strong indications that the use of adaptation improves the mesh convergence significantly, even when the adjoint is used. The error estimate stemming from the adjoint does however seem to work well. It must here be noted that the error estimate was also applied by A-F with the elsA industrial code, albeit with less accuracy than demonstrated by the higher order code, presumably at least partly due to the early design stage of the method. The error estimation can thus not be considered a method only available for higher order FE schemes, but it might be that it is better suited and more accurate when used in conjunction with such approaches. Some partners have argued in ADIGMA that adjoint–based adaptation is not well suited for the FV schemes in industrial use. Also demonstrated in the project is the usage of h-p refinement strategies where the importance of first ensuring a sufficient h-resolution before p-adaptation is started is described. The methods show a large potential, especially if high accuracy requirements are posed on the solution. In general it appears that adaptation approaches applied in ADIGMA are robust and to a high degree automatic.

6 Conclusions

The large amount of testcases considered in the industrial validation was a result of the considerable differences in maturity levels of the codes developed in the project. While the aircraft industry is almost exclusively interested in 3D RANS computations involving geometries of realistic complexity, many of the partners of the project were limiting the scope of their studies to 2D and, more importantly, very few codes were capable of solving turbulent flow in a stable manner. In hindsight, it would probably have been more constructive to focus on a smaller amount of cases, allowing the partners to spend more time on the asymptotic analyses, resulting in clearer data for the industrial evaluation. It must however also be said that many partners were not prepared to invest an adequate amount of resources into the evaluation of testcases, choosing to focus on the development of their codes. As a result, the conclusions of the evaluation, treated in the next chapter, were of reduced accuracy compared with what was planned for at the onset of ADIGMA. Still, this was one of the first times an attempt has been made in an EU-funded project on CFD to conduct a rigorous comparison of new approaches with the current industrial standard, an approach that allows projects to keep an industrial perspective. This not only allows the industry to gain an impression of the attractiveness of the newest approaches developed in academia, but also informs the universities and research institutions of the needs of industry and the performance level they need to beat, as well as their current relative position to this baseline. Even though the execution of the industrial evaluation was partly unsatisfactory, it is thus believed that the exercise was a very valuable feature of ADIGMA that should be considered for future projects of industrial relevance.

References

1. ADIGMA Deliverable D2.2.2, Report of scope of assessment and assessment procedure (2006)
2. ADIGMA Deliverable D6.3.1, Synthesis report on MTC suite for final assessment (2009)
3. ADIGMA Deliverable D6.3.2, Synthesis report on BTC suite for final assessment (2009)
4. ADIGMA Deliverable D6.3.3, Synthesis report on CTC suite for final assessment (2009)
5. ADIGMA form, Method Industrial Potential Questionnaire (2009)

Chapter 34
Conclusions and Recommendations

Norbert Kroll, Heribert Bieler, Herman Deconinck, Vincent Couaillier, Harmen van der Ven, and Kaare Sørensen

Abstract. This chapter presents the major conclusions drawn from the collaborative European research effort ADIGMA on adaptive variational methods for aerodynamic applications in industry. Despite the fact that the ADIGMA project has significantly further matured higher-order solvers for the simulation of compressible flows, further research and improvements are still required to realize their full potential for industrial use. Recommendations for future research paths are given.

Norbert Kroll
German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology, Lilienthalplatz 7, 38108 Braunschweig, Germany
e-mail: norbert.kroll@dlr.de

Heribert Bieler
Airbus Operations GmbH, Airbusallee 1, 28199 Bremen, Germany
e-mail: heribert.bieler@airbus.com

Herman Deconinck
Von Karman Institute, Chausse de Waterloo, 72, 1640 Rhode-St-Gense, Belgium
e-mail: deconinck@vki.ac.be

Vincent Couaillier
ONERA BP 72-79 avenue de la Division Leclerc FR-92322 Chtillon Cedex
e-mail: vincent.couaillier@onera.fr

Harmen van der Ven
National Aerospace Laboratory, A. Fokkerweg 2, Amsterdam, The Netherlands
e-mail: venvd@nlr.nl

Kaare Sørensen
EADS-MAS, Aerodynamics & Methods, Rechliner Str., 85077 Manching, Germany
e-mail: kaare.sorensen@eads.com

N. Kroll et al. (Eds.): ADIGMA, NNFM 113, pp. 483–491, 2010.
springerlink.com

1 Higher-Order Discretization Methods

Major achievements

Throughout ADIGMA, the state-of-the-art of methods with discontinuous bases, represented mostly by the generic family of Discontinuous Galerkin (DG) Methods, has been further improved from the solution of viscous laminar flows for simple configurations to turbulent flows around aerodynamic configurations of moderate complexity (University of Bergamo, ONERA). For example, the University of Bergamo has demonstrated its capability to successfully conduct RANS simulation for the DLR-F6 wing/body configuration up to 4^{th}-order. Moreover, they have shown that in contrast to standard finite-volume solvers, DG methods can be used on high-aspect ratio meshes without severe convergence degradation as long as sufficient integration is employed. Some novel formulations for DG discretization methods have been investigated such as edge-based formulation (University of Uppsala), semi-implicit discretization (University of Prague) and space-time discretization strategies (University of Stuttgart, University of Twente). Since high-order DG discretization raises many efficiency issues in terms of CPU and memory usage, several aspects, that should have a significant impact on the code complexity and efficiency, have been investigated. These include efficient ways of implementing DG methods by mitigating the increased cost of higher-order discretizations by casting the method in terms of global linear algebra operations (CENAERO). Other partners have explored the type of shape functions, the choice of quadrature formulae or the use of reduced quadrature rules (zonal strategy, CENAERO). The results achieved have not always been conclusive. However in many test cases it has been shown that, although harder to optimize, higher-order DG methods are competitive to industrial base line simulation codes, when targeting the same accuracy threshold.

The continuous high-order formulations were less mature and understood at the start of the project than their discontinuous counterparts. Within ADIGMA the continuous high-order formulations have been further developed up to maturity for computation of aerodynamic configurations of moderate complexity. High-order accuracy has been demonstrated on a series of test cases on external aerodynamic configurations, from subsonic to transonic flow regimes, both for inviscid and laminar flows. Promising preliminary results have also been shown for RANS simulations (Dassault Aviation, SERAM), but these methods need a better understanding of the discretization of the turbulence equations. Both VKI and INRIA have developed Residual Distribution (RD) methods starting from scalar equations up to laminar Navier-Stokes equations and demonstrate some first RANS capabilities, with VKI focusing on multidimensional upwind schemes and INRIA on simpler, non-upwind, artificially stabilized methods.

One of the challenges in higher-order methods is the preservation of monotonicity over discontinuities. It is a crucial requirement for the methods concerning robustness for industrial aerodynamic applications. Many different

shock capturing techniques have been investigated within the project. Some partners have focused their activities on finding so-called "troubled cell indicators" (University of Stuttgart, Nanjing University, VKI), while others have relied on a global shock-sensing schemes (INRIA, Dassault Aviation). An extensive comparison of the different shock capturing procedures, both for continuous and discontinuous methods, has been performed under the same test case conditions. No uniform solution to the problem of high-order shock capturing has been found. However, the comparisons have helped to identify some techniques which perform better for steady cases while others perform better for unsteady cases.

In order to achieve the full potential of higher-order methods, the mesh needs to represent the underlying geometry and in particular to resolve regions of high curvature. However, several important bottlenecks exist in the creation of such a boundary representation, particularly in case of highly stretched boundary layer meshes around 3D configurations. Some aspects have been tackled within ADIGMA including the treatment of curved wall boundary conditions, as well as solution adaptive meshing, the development of mesh deformation strategies to provide curved higher-order elements close to the wall boundary and the specification of a set of mesh quality metrics for higher-order simulations. ARA has extended its hybrid grid generation system SOLAR to a higher-order meshing capability. An initial implementation of such a capability has been successfully demonstrated for the ONERA M6 wing. It is clear though, that this puts an increased onus on the mesh generation procedure.

In general, the partners developed their own independent software. There was, however, a limited effort where the numerical methods of Warsaw University of Technology, University of Prague and VKI were integrated into a collaborative software platform (COOLFluiD).

Challenges and recommendations

The competitiveness of higher-order methods to standard finite volume solvers has been demonstrated for airfoil computations and 3D inviscid or laminar flows around rather simple configurations. Only limited research activities have been devoted to the discretization of the RANS equations with higher-order methods, and it has become clear that this effort is in its infancy. A dedicated effort towards the industrialization of the different higher-order methods is required and in particular the understanding of the discretization procedure needs to mature. Moreover, although work has been carried out to mitigate the resource usage of higher-order methods, further research needs to be invested in the area of algorithm optimization and complexity reduction. Finally, as high-order methods target same accuracy levels with coarser and coarser meshes as the approximation order increases, the quality of the mesh generation becomes more important. Care must be placed into generating meshes where higher-order elements with high aspect ratio have

valid curved geometries. Achieving this on arbitrarily complex geometries is a challenge for the future. Additionally, the interface with the underlying CAD model information should not be lost, in order to support possible adaptation procedures.

2 Solution Strategies

Major achievements

Computational efficiency is a crucial aspect for higher-order methods. Within ADIGMA solution strategies have been identified and further developed towards having the potential to meet the industrial requirements in terms of memory storage, computing time and efficient utilization of parallel low cost computers. According to the literature two major research lines have been followed, namely capable multigrid strategies (p-multigrid, h-multigrid) on the one side and efficient and robust implicit Newton-type techniques with particular focus on linearization, linear solvers and suitable preconditioners on the other side. For the Discontinuous Galerkin discretizations a detailed two- and three-level h-multigrid analysis has been conducted both for the advection-diffusion and the linearized Euler equations in two space dimensions (University of Twente, NLR). These analysis tools have been successfully used to optimize Runge-Kutta type smoothers, which resulted in a significant improvement in convergence rate. A spectral p-multigrid DG algorithm for the solution of the steady state Euler and Navier-Stokes equations have been investigated by the University of Bergamo. Care has been taken to design efficient smoothers, in particular for viscous flows. Compared to classical implicit schemes this approach provides similar efficiency along with significant memory savings of about 75%. A hybrid multigrid approach has been proposed by ONERA in the frame of a 3D multiblock structured solver using second- or third-order Discontinuous Galerkin discretizations on the fine grid and classical second-order finite volume formulations on the coarser grids. NLR has extended the h-multigrid algorithm to four-dimensional (space-time DG discretization) time-accurate simulations targeted for helicopter applications including locally refined meshes. With respect to fully implicit methods based on Newton-type iterations, various aspects crucial for higher-order discretizations have been investigated. Particular focus has been put on linearization and efficiency of the linear solvers associated with each Newton step using GMRES (University of Bergamo, University of Prague). The research has covered e.g. the choice of preconditioners, the dimension of the Krylov subspaces and the number of iterations and the relative tolerance for GMRES. Although in many cases progress has been achieved in reducing the computational effort of higher-order methods, the findings have not always been conclusive, in particular with respect to viscous turbulent flow problems. Efficient parallelization of the higher-order methods has been another research topic within ADIGMA. Parallel versions of discontinuous Galerkin methods (e.g. University of Bergamo) and Residual

Based Distribution Schemes (INRIA, VKI) have been developed and analyzed. Research work has been devoted to dynamic load balancing allowing for proper parallel efficiency, particularly in case of local grid refinement (Warsaw University of Technology). It has been demonstrated that good parallel performance can be achieved with higher-order methods; however, the tests have been limited to rather small number of processors. Finally, hybrid techniques based on the heterogeneous domain approach coupling different discretization and integration strategies have been investigated within ADIGMA as a means to cut computational cost while maintaining improved accuracy of higher-order discretization. A hybrid Discontinuous Galerkin/finite volume solver has been investigated by ONERA. First results for 2D turbulent flows look promising but evaluation on more complex configurations is required in order to assess the benefit compared to a full discontinous Galerkin approach. A hybrid parallel framework connecting different classes of methods (DG, FV, FD) on structured and unstructured grids for the solution of different governing equations (linearized Euler, nonlinear Euler and Navier-Stokes equations) has been investigated by University of Stuttgart. This capability has been successfully demonstrated by the simulation of laminar vortex shedding. An alternative approach has been proposed by University of Wales Swansea and EADS-MAS. Higher-order discretizations and improved equation solution has been attempted within wall boundary layer regions based on standard finite volume solvers. The benefit of this methodology has been demonstrated for a number of examples. However, further work is required before general conclusions can be drawn about practical usefulness of this approach for industrial flow simulations.

Challenges and recommendations

Although within ADIGMA various methods and strategies have been investigated and further enhanced to improve the solver efficiency of higher-order methods, the development of memory and CPU efficient solvers for large-scale industrial relevant applications still remains a major challenge. The techniques developed have been mainly tested and adjusted for inviscid and laminar flows with moderate geometric complexity. The results achieved so far are not conclusive for turbulent complex flow problems. Further research work including efficient adaptation to the new type of processors and computer architectures is required in order to mature higher-order methods for industrial use.

3 Adaptation and Error Estimation

Major achievements

Local grid refinement is an essential ingredient for higher-order methods to be competitive with standard finite volume solvers. Traditionally, mesh adaptation for flow problems is based on feature-based sensors, which refine the

computational mesh based on flow features such as shocks and vortices. The mathematical framework of finite-element methods allows sensors targeted at reducing the error in the computation, so-called goal-oriented adaptation. Within ADIGMA existing goal-oriented algorithms for the Euler equations and single functionals have been extended to the laminar and turbulent Navier-Stokes equations and multiple functionals (DLR). The same holds for a posteriori error estimation algorithms and the extension to anisotropic refinement (DLR, University of Nottingham). The developed algorithms have successfully been applied to the 2D high-lift test case, a turbulent 3D streamlined body and wing-body configuration, clearly showing the fast convergence of the goal-oriented refinement strategy, especially when combined with error estimation. An important result is that the method is more efficient than performing a grid convergence study in terms of turnaround time, even though it requires an additional solution of the adjoint problem. As DG methods allow variation of the polynomial order on a cell-to-cell basis, *hp*-refinement strategies have been developed in the same context as the above *h*-refinement strategies. For *p*-refinement both feature-based sensors (University of Stuttgart) and sensors based on the smoothness of the solution have been developed (University of Nottingham). The efficiency of the anisotropic *hp*-refinement algorithm has been demonstrated for 2D laminar flow problems (University of Nottingham).

Challenges and recommendations

In order for local refinement strategies to be efficient, the starting mesh for the computations should be as coarse as possible, but also represent the geometry accurately. This poses different requirements on grid generation than the classical finite volume methods do. In case of turbulent simulations, an additional complication is that the curvature of the geometry must be extended into the flow domain in order to avoid grid folding. In ADIGMA only a small portion of the research effort has been devoted to this problem. There are currently no grid generators available which can be routinely used for this task. Thus the development of such grid generators is highly recommended. The goal-oriented sensors and error estimation developments have shown good progress. Nonetheless, maturation of the algorithms is required for routine industrial use, especially for turbulent flow and/or *p*-refinement. Another consideration in the application of goal-oriented adaptation and error estimation to turbulent flows is the balance between numerical and modeling error. The error estimation algorithms will reduce the numerical error, but will not improve upon the turbulence model. There will be a cross-over point where the numerical error drops below the modeling error and further reduction of the numerical error does not make sense. This cross-over point is unknown and application-dependent. It is recommended to develop best practices for stopping criteria for turbulent flows.

4 Industrial Assessment

Major achievements

The assessment of the developed methods and technologies in an industrial setting has been a central issue in the ADIGMA project. A specific activity has been devoted to the specification of requirements that industrial aerodynamic applications will be putting on future numerical simulation tools. Industrial partners have expressed their needs in terms of accuracy, efficiency, robustness, ease of use, applicability and implementation issues among others. A test case suite of increasing complexity has been specified together with clearly defined reporting templates in order to put the comparison of newly developed methods with traditional industrial flow solvers on a firm basis. The large amount of test cases considered in the industrial assessment activity has been a result of considerable differences in the maturity of the codes developed in ADIGMA. Many of the partners had to limit their studies to 2D flows and more importantly, very few codes were capable to treat turbulent flows. For some test cases (especially for the more complex ones) just one partner contributed with a newly developed method, and so the conclusion was somehow limited to this specific run. Furthermore, the thorough asymptotic analysis on order of accuracy has also been restricted to rather simple cases. In general, the higher-order methods investigated within ADIGMA show a potential for reducing the discrete system size by a factor of about 5–10 for most of the test cases and accuracy levels considered. At the same time, the memory usage of the examined codes is typically at least one order of magnitude higher. The gain in discrete system size, however, is not yet fully transferred to increased runtime performance as available solver technologies are not adequate for large scale applications. The treatment of flow discontinuities in transonic flow problems does not seem to be a major problem regarding robustness or efficiency and turbulence modeling does not represent a principal hindrance to the higher-order approaches. The codes that have been used in ADIGMA appear to be readily parallelizable and some of them appear to be suitable for enhancements to a competitive code for industrial use within the next few years. The solver part of the codes turned out to be the largest bottleneck and require future research activities. Although the evaluation has not been as rigorous as planned resulting in somewhat limited conclusion, a first attempt has been made in an EU-funded CFD project to conduct a careful and thorough assessment of new technologies with the current industrial standard. This approach on the one hand allows industry to gain insight in new CFD technologies developed in academia and on the other hand provides universities and research institutions industrial needs and performance levels to be beaten.

Challenges and recommendations

It has been realized during the execution of the test cases that, at the start of the project, rather ambitious test cases were intended to be investigated.

In order to keep the test case suites manageable for the majority of the consortium members, adjustments and redefinitions were necessary. In some cases this raised questions, whether the simplified test cases were still close enough to industrial needs to raise the industrial awareness for this project. Based on the experience gained in ADIGMA it is highly recommended to include a rigorous well defined industrial assessment in any future planning of adaptive higher-order R&T projects.

5 Dissemination and Exploitation

The knowledge gained in the ADIGMA project, the computational methods and the particular results generated have been disseminated in various forms. Important means are publications in journals and technical papers and presentation at national and international conferences. In particular, the two ADIGMA VKI Lecture Series courses, the publicly open final workshop and the final report published as a dedicated book in the Springer Series "Notes on Numerical Fluid Mechanics and Multidisciplinary Design Notes" are seen as important channels to disseminate the project results. Within the project a comprehensive data base for the evaluation of adaptive higher-order methods has been created. The data base includes specification of test cases, computational grids, solutions of higher-order obtained with various approaches as well as reference solutions of standard industrial second-order methods. This test case suite is of high interest to the CFD community to promote the proliferation of high-order CFD tools. The ADIGMA objectives enabled a strong cooperation between universities (upstream research), aircraft industry (end user) and research establishments (bridge between basic research and application). According to their role the organizations have applied different dissemination and exploitation strategies. The aircraft industry is directly involved in the transfer process from basic research and development into industrially applicable simulation methods and tools. The newly developed discretization schemes and numerical solution algorithms have been explored on industrially relevant test cases. With the help of the research establishments as central providers of highly sophisticated CFD simulation tools for the aircraft industry the most promising methods will be further explored on even more complex cases from daily aerodynamic work. The research organizations participating in ADIGMA will directly exploit the knowledge gained in the project by improving their numerical tools. In particular, ADIGMA has been an important step towards the establishment of the next generation of CFD tools which can cope with the future requirements of the aeronautical industry. By providing improved CFD codes to their customers and partners in industry and academia, the research organizations actively contribute to the dissemination of the ADIGMA results. The universities taking part in ADIGMA will directly exploit the ADIGMA findings of advanced numerical algorithms and procedures for teaching and training students and researchers.

The close cooperation with industry will lead to the training of qualified personnel with knowledge of the industrial requirements, thereby increasing the potential of graduates for employment within industry. The project outcome will allow universities to pursue their goals in the field of applied mathematics and computational fluid dynamics, both in research and education.

6 Summary

The main achievements of the collaborative research project funded by the European Union in the Sixth Framework Programme on Aeronautics and Space are (1) significant progress in the development of adaptive higher-order methods for aerodynamic applications with high scientific output (2) unique approach for critical assessment of innovative methods for industrial use, (3) creation of a comprehensive data base for performance assessment of advanced CFD methods, (4) successful demonstration of the potential and capabilities of higher-order methods, (5) identification of limitations and research directions for further industrialization of higher-order methods as well as (6) significant improvement of the collaboration between academia, research organizations and industry on advanced CFD methods. Despite the significant progress, it has to be mentioned that many achievements are still far from industrial use. In order to realize the full potential of adaptive higher-order methods, further concentrated research effort is required. Particular research areas to be addressed are generation of coarse higher-order meshes and memory-efficient solver strategies for large-scale simulations for turbulent flow problems.

This page intentionally left blank

Author Index

Notes on Numerical Fluid Mechanics and Multidisciplinary Design

Available Volumes

Volume 113: Norbert Kroll, Heribert Bieler, Herman Deconinck, Vincent Couaillier, Harmen van der Ven, and Kaare Sørensen (eds.): ADIGMA – A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications – Results of a collaborative research project funded by the European Union, 2006–2009. ISBN 978-3-642-03706-1

Volume 112: Andreas Dillmann, Gerd Heller, and Gerd Heller, Hans-Peter Kreplin, Wolfgang Nitsche, and Wolfgang Schröder (eds.): New Results in Numerical and Experimental Fluid Mechanics VII – Contributions to the 16th STAB/DGLR Symposium Aachen, Germany 2008. ISBN 978-3-642-14242-0

Volume 111: Shia-Hui Peng, Piotr Doerffer, and Werner Haase (eds.): Progress in Hybrid RANS-LES Modelling – Papers Contributed to the 3rd Symposium on Hybrid RANS-LES Methods, Gdansk, Poland, June 2009. ISBN 978-3-642-14167-6

Volume 110: Michel Deville, Thien-Hiep Lê, and Pierre Sagaut (eds.): Turbulence and Interactions – Proceedings the TI 2009 Conference. ISBN 978-3-642-14138-6

Volume 109: Wolfgang Schröder (ed.): Summary of Flow Modulation and Fluid-Structure Interaction Findings – Results of the Collaborative Research Center SFB 401 at the RWTH Aachen University, Aachen, Germany, 1997-2008. ISBN 978-3-642-04087-0

Volume 108: Rudibert King (ed.): Active Flow Control II – Papers Contributed to the Conference "Active Flow Control II 2010", Berlin, Germany, May 26–28, 2010. ISBN 978-3-642-11734-3

Volume 107: Norbert Kroll, Dieter Schwamborn, Klaus Becker, Herbert Rieger, Frank Thiele (eds.): MEGADESIGN and MegaOpt – German Initiatives for Aerodynamic Simulation and Optimization in Aircraft Design. ISBN 978-3-642-04092-4

Volume 106: Wolfgang Nitsche, Christoph Dobriloff (eds.): Imaging Measurement Methods for Flow Analysis - Results of the DFG Priority Programme 1147 "Imaging Measurement Methods for Flow Analysis" 2003–2009. ISBN 978-3-642-01105-4

Volume 105: Michel Deville, Thien-Hiep Lê, Pierre Sagaut (eds.): Turbulence and Interactions - Keynote Lectures of the TI 2006 Conference. ISBN 978-3-642-00261-8

Volume 104: Christophe Brun, Daniel Juvé, Michael Manhart, Claus-Dieter Munz: Numerical Simulation of Turbulent Flows and Noise Generation - Results of the DFG/CNRS Research Groups FOR 507 and FOR 508. ISBN 978-3-540-89955-6

Volume 103: Werner Haase, Marianna Braza, Alistair Revell (eds.): DESider – A European Effort on Hybrid RANS-LES Modelling - Results of the European-Union Funded Project, 2004–2007. ISBN 978-3-540-92772-3

Volume 102: Rolf Radespiel, Cord-Christian Rossow, Benjamin Winfried Brinkmann (eds.): Hermann Schlichting – 100 Years - Scientific Colloquium Celebrating the Anniversary of His Birthday, Braunschweig, Germany 2007. ISBN 978-3-540-95997-7

Volume 101: Egon Krause, Yurii I. Shokin, Michael Resch, Nina Shokina (eds.): Computational Science and High Performance Computing III - The 3rd Russian-German Advanced Research Workshop, Novosibirsk, Russia, 23–27 July 2007. ISBN 978-3-540-69008-5

Volume 100: Ernst Heinrich Hirschel, Egon Krause (eds.): 100 Volumes of 'Notes on Numerical Fluid Mechanics' - 40 Years of Numerical Fluid Mechanics and Aerodynamics in Retrospect. ISBN 978-3-540-70804-9

Volume 99: Burkhard Schulte-Werning, David Thompson, Pierre-Etienne Gautier, Carl Hanson, Brian Hemsworth, James Nelson, Tatsuo Maeda, Paul de Vos (eds.): Noise and Vibration Mitigation for Rail Transportation Systems - Proceedings of the 9th International Workshop on Railway Noise, Munich, Germany, 4–8 September 2007. ISBN 978-3-540-74892-2

Volume 98: Ali Gülhan (ed.): RESPACE – Key Technologies for Reusable Space Systems - Results of a Virtual Institute Programme of the German Helmholtz-Association, 2003–2007. ISBN 978-3-540-77818-9

Volume 97: Shia-Hui Peng, Werner Haase (eds.): Advances in Hybrid RANS-LES Modelling - Papers contributed to the 2007 Symposium of Hybrid RANS-LES Methods, Corfu, Greece, 17–18 June 2007. ISBN 978-3-540-77813-4

Volume 96: C. Tropea, S. Jakirlic, H.-J. Heinemann, R. Henke, H. Hönlinger (eds.): New Results in Numerical and Experimental Fluid Mechanics VI - Contributions to the 15th STAB/DGLR Symposium Darmstadt, Germany, 2006. ISBN 978-3-540-74458-0

Volume 95: R. King (ed.): Active Flow Control - Papers contributed to the Conference "Active Flow Control 2006", Berlin, Germany, September 27 to 29, 2006. ISBN 978-3-540-71438-5

Volume 94: W. Haase, B. Aupoix, U. Bunge, D. Schwamborn (eds.): FLOMANIA - A European Initiative on Flow Physics Modelling - Results of the European-Union funded project 2002 - 2004. ISBN 978-3-540-28786-5

Volume 93: Yu. Shokin, M. Resch, N. Danaev, M. Orunkhanov, N. Shokina (eds.): Advances in High Performance Computing and Computational Sciences - The Ith Khazakh-German Advanced Research Workshop, Almaty, Kazakhstan, September 25 to October 1, 2005. ISBN 978-3-540-33864-2

Volume 92: H.J. Rath, C. Holze, H.-J. Heinemann, R. Henke, H. Hönlinger (eds.): New Results in Numerical and Experimental Fluid Mechanics V - Contributions to the 14th STAB/DGLR Symposium Bremen, Germany 2004. ISBN 978-3-540-33286-2

Volume 91: E. Krause, Yu. Shokin, M. Resch, N. Shokina (eds.): Computational Science and High Performance Computing II - The 2nd Russian-German Advanced Research Workshop, Stuttgart, Germany, March 14 to 16, 2005. ISBN 978-3-540-31767-8

Volume 87: Ch. Breitsamter, B. Laschka, H.-J. Heinemann, R. Hilbig (eds.): New Results in Numerical and Experimental Fluid Mechanics IV. ISBN 978-3-540-20258-5

Volume 86: S. Wagner, M. Kloker, U. Rist (eds.): Recent Results in Laminar-Turbulent Transition - Selected numerical and experimental contributions from the DFG priority programme 'Transition' in Germany. ISBN 978-3-540-40490-3

Volume 85: N.G. Barton, J. Periaux (eds.): Coupling of Fluids, Structures and Waves in Aeronautics - Proceedings of a French-Australian Workshop in Melbourne, Australia 3-6 December 2001. ISBN 978-3-540-40222-0

Volume 83: L. Davidson, D. Cokljat, J. Fröhlich, M.A. Leschziner, C. Mellen, W. Rodi (eds.): LESFOIL: Large Eddy Simulation of Flow around a High Lift Airfoil - Results of the Project LESFOIL supported by the European Union 1998 - 2001. ISBN 978-3-540-00533-9

Volume 82: E.H. Hirschel (ed.): Numerical Flow Simulation III - CNRS-DFG Collaborative Research Programme, Results 2000-2002. ISBN 978-3-540-44130-4

Volume 81: W. Haase, V. Selmin, B. Winzell (eds.): Progress in Computational Flow Structure Interaction - Results of the Project UNSI, supported by the European Union 1998-2000. ISBN 978-3-540-43902-8

Volume 80: E. Stanewsky, J. Delery, J. Fulker, P. de Matteis (eds.): Drag Reduction by Shock and Boundary Layer Control - Results of the Project EUROSHOCK II, supported by the European Union 1996-1999. ISBN 978-3-540-43317-0

Volume 79: B. Schulte-Werning, R. Gregoire, A. Malfatti, G. Matschke (eds.): TRANSAERO - A European Initiative on Transient Aerodynamics for Railway System Optimisation. ISBN 978-3-540-43316-3