UNIVERSITÉ CATHOLIQUE DE LOUVAIN
ÉCOLE POLYTECHNIQUE DE LOUVAIN
INSTITUT DE MÉCANIQUE, MATÉRIAUX ET GÉNIE CIVIL

DEVELOPMENT OF THE DISCONTINUOUS GALERKIN METHOD FOR HIGH-RESOLUTION, LARGE SCALE CFD AND ACOUSTICS IN INDUSTRIAL GEOMETRIES

DOCTORAL DISSERTATION PRESENTED BY

KOEN HILLEWAERT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR IN APPLIED SCIENCES

THESIS COMMITTEE:

Prof. Rémi Abgrall, Université de Bordeaux et INRIA
Prof. Philippe Chatelain, Université catholique de Louvain
Dr. Nicolas Chevaugeon, École Centrale de Nantes
Prof. Erik Dick, Universiteit Gent
Dr. Philippe Geuzaine, Cenaero
Prof. Vincent Legat, Université catholique de Louvain
Prof. Jean-François Remacle, Université catholique de Louvain (Advisor)
Prof. Grégoire Winckelmans, Université catholique de Louvain (President)

Louvain-la-Neuve, February 15^{th}, 2013

DANK U WEL / MERCI

Nu dit werk er op zit besef ik eens te meer hoeveel ik te danken aan mijn familie, vrienden en collega's, waarvan ik alleen maar kan hopen dat ze weten hoezeer ik hen apprecieer en graag zie. Omdat ik jammer genoeg dit niet altijd makkelijk laat blijken, is dit een unieke gelegenheid om dat wel te doen.

Samly is sinds lang mijn steun en toeverlaat, mijn beste vriendin, en nog altijd mijn lief. Samen met Maarten en Beatrice zorgt ze ervoor dat thuiskomen en zijn altijd leuk is, ook al ga ik nogal op in mijn werk, en had ik de laatste tijd niet altijd veel tijd voor hen. Alhoewel het niet altijd even plezierig moet geweest zijn, hebben ze mij de tijd gelaten om dit werk af te maken.

Mijn ouders ben ik zeer dankbaar voor meer dan 40 jaar liefde, vertrouwen, interesse, aanmoedigingen en goede raad. Ze hebben mij de mogelijkheid gegeven om me ten volle ontplooien, niet alleen door mij vrij te laten in mijn keuzes, maar vooral door me telkens ten volle te steunen, ook al hadden we niet altijd dezelfde mening. Ik hoop dat Maarten en Beatrice later hetzelfde over ons kunnen zeggen.

J'aimerais tout particulièrement remercier Jean-François pour m'avoir encadré tout au long de ma recherche, pour ses précieux conseils, pour le partage de ses connaissances et surtout son amitié. Je garderai toujours des beaux souvenirs à nos discussions - techniques et autres - qui ont infailliblement résulté dans des nouvelles idées et pistes de recherche, presqu'invariablement combiné avec l' historique du heavy metal (dont je maintiens que les pères sont les Zeps). Je suis sûr que notre collaboration perdurera encore longtemps, tout comme mon amitié envers lui.

Ook Prof. Dick en Prof. Van den Braembussche wil ik van harte danken, niet in het minst om me de passie voor en het inzicht in numerieke methodes, stromingsmechanica en turbomachines bij te brengen. De ervaring en kennis die zij mij doorgaven, hebben mij toegelaten om verder te gaan in dit boeiende vakdomain, en in het bijzonder om deze thesis tot een goed einde te brengen.

Michel et Philippe m'ont donné la possibilité de poursuivre cette recherche à Cenaero, malgré que l'applicabilité industrielle de DGM était encore à démontrer à l'époque. Je leur suis très reconnaissant d'avoir eu confiance dans mon intuition, et de m'avoir accordé le temps et soutien pour mener ce projet à bien. J'espère en revanche qu'ils sont contents du travail accompli par toute l'équipe Argo.

Depuis toujours, j'ai été entouré de bons compagnons de route, tous aussi amiables que compétents. Je pense tout d'abord à l'équipe Argo - Corentin, François, Bastien, Marcus, Guillaume et Pierre - dont l'enthousiasme et les efforts ont contribué largement aux succès de léquipe, et encore plus qu'ils ne s'en rendent compte, à ma motivation et mon plaisir dans le travail. Je pense également à l'équipe UCL -Jean-François, Nicolas, Émilie, Gaëtan, Cécile, Brian, Jon et Paul-Émile - avec qui

j'ai pris mes premiers pas en DGM, et qui m'ont appris tant de choses, non seulement au niveau technique, mais également au niveau humain.

Mes voisins de bureau, Cécile et Corentin, ont égayé mes jours à Cenaero par leur humour, leur écoute et leurs conseils, qu'aucun chouffomètre saura mesurer à sa juste valeur. Je pense également à tous mes collègues de longue date à Cenaero: Sandrine, Séverine, Éric, François, Ingrid, Arnaud, Nicolas, Josué, Anne, Ashvin, Landry, Thibaut, Danielle, Laurent, Serge ... et tant d'autres qui ont, chacun à sa manière, m'ont rendu la vie à Cenaero plus que agréable et intéressante.

Au cours des années l'équipe Argo a bénéficié de collaborations avec différentes équipes de recherche dans les universités wallonnes, à travers de la recherche collaborative, des doctorats ainsi que d'un grand nombre travaux de fin d'études et de stages. J'aimerais profiter de l'occasion pour remercier Prof. Bricteux (UMons), Prof. Chatelain (UCL), Prof. Coussement (UMons), Prof. Degrez (ULB), Prof. Essers (ULg), Prof. Geuzaine (ULg), Prof. Terrapon (ULg) et Prof. Winckelmans (UCL) pour leur confiance et soutien.

I would also like to acknowledge the financial support by the European projects Sinus (FEDER), Adigma (FP6) and IDIHOM (FP7), in which a large part of the developments of Argo were done. Furthermore I would like to thank the DEISA and PRACE consortium for the HPC grants which permitted more advanced HPC developments and test cases.

Finalement, je remercie vivement les membres du jury, pour avoir pris le temps de lire mon texte et évaluer le travail, ainsi que pour les précieux conseils et remarques. Ceci doit être la seule section qui n'a pas été vérifiée et commentée par leurs soins. J'espère de ne pas avoir laissé de coquilles ici, ni introduit des nouvelles ailleurs. Dès lors je vous souhaite bonne lecture.

CONTENTS

Contents **i**

1 Introduction **1**
1.1 Motivation 1
1.2 Overview 8

2 The discontinuous Galerkin method **11**
2.1 Model equations 11
2.2 Elements and functional spaces 12
2.3 Variational formulation 14
2.4 Shape functions 23

3 Extending the DG variational formulation **27**
3.1 Stability of the interior penalty method on hybrid meshes 27
3.2 Non-conformal formulation 35
3.3 Frequential formulation of the homentropic LEE 44

4 Iterative methods **57**
4.1 Newton methods 59
4.2 Multigrid methods 63
4.3 Concluding remarks 71

5 Efficient data structures **75**
5.1 Algebraic primitives on the computer 76
5.2 Data Structures 84
5.3 Efficient assembly 89
5.4 Conclusions 102

6 noFUDGe: a first industrial application **105**
6.1 Description of the flow 106
6.2 Computational setup 106
6.3 Comparison of computed flow fields 108
6.4 Validation 114
6.5 Comparison of computational cost 116
6.6 Scaling tests 116
6.7 Conclusions 118

7 Current status and prospects **119**
7.1 Conclusions 119
7.2 Current status of the Argo group 120
7.3 Prospects 121

A Elements of functional analysis **A.3**
A.1 Hilbert spaces A.3
A.2 Solvability of variational problems A.4
A.3 The Lax-Milgram theorem A.5
A.4 The most simple example A.5

B Function spaces, reference elements and quadrature **A.7**
B.1 Construction of Lagrange interpolants A.7
B.2 Interpolation on the boundary A.8
B.3 Specific elements A.9
B.4 Quadrature rules A.12

C Sharp values for the trace inverse inequality **A.15**
C.1 Simplices A.16
C.2 Outline of Warburton's method A.16
C.3 Tensor product elements A.17
C.4 Wedges A.19
C.5 Lagrange interpolation on pyramids A.21
C.6 The Pascal space on the pyramid A.25

D Nonlinear instability of quadrature-free methods **A.27**
D.1 Original formulation A.27
D.2 Extension to non-linear equations of state A.30
D.3 Spurious modes A.30

Bibliography **i**

CHAPTER 1

INTRODUCTION

1.1 Motivation

Throughout the development of *Computational Fluid Dynamics (CFD)*, as in any other field of computational analysis, there is an ever growing need for increased *accuracy* of the simulations. This increased accuracy is addressed by the use of better models on the one hand, and an enhanced precision of the solution of the model equations on the other [1]. This demand for high-fidelity simulations is driven by ever more stringent design requirements, and fueled by both an increased availability of computational resources as well as the scientific progress in the physical comprehension of fluid dynamics phenomena, in particular turbulence.

A second important industrial need is the *reliability* of simulations. In the long run, research and development teams would like to dispose of methods that can be used by design engineers that obviously have a detailed understanding of the physics, but are not necessarily experts on numerical methods. Ideally this reliability would be provided by computing strategies, in which resolution is adapted in function of the solution and performance criteria.

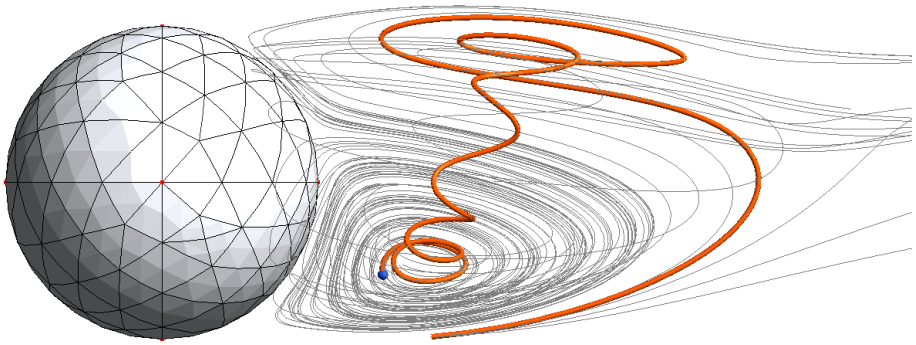
Finally the further improvement of an already impressive level of performance, currently obtained in industrial design, necessitates a very thorough and systematic exploration of design parameters, as currently the comprehension of the flow physics alone is no longer sufficient to further increase performance. Such a systematic exploration has to be performed by *automa-*

[1] Accuracy and precision should not be confused. In numerical modeling, precision refers to the error made by the code with respect to the model due to a lack of resolution. Accuracy refers to the closeness of the results to experiments, and hence encompasses both the quality of the model and the precision with which the model is solved.

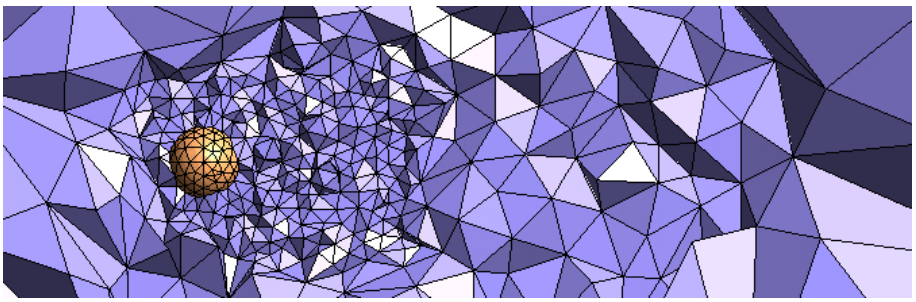
tised optimisation chains, such as developed at Cenaero, that autonomously launch computations for parametric variations of the geometry. The engineer only intervenes in the setup of the optimisation process, where he defines the parametrisation of the geometry, the performance goals and design constraints, and during the exploitation of the results. It is clear that the use of computational methods within such an automated design chain exacerbates greatly the importance of reliability.

Starting from the conviction that new high-order discretisations can provide enabling technology for these thematics, Cenaero is bringing such a technology up to par with an industrial use within the CFD platform called *Argo*. Although these thematics are equally important, their order is not innocent: it corresponds to our perception of the maturity and short-term applicability of high-order methods. This thesis is inscribed in this effort.

The following subsections further clarify these opportunities and challenges related, after the concept of high-order methods is clarified.



(a) Stream lines



(b) Mesh in the vicinity and wake of the sphere

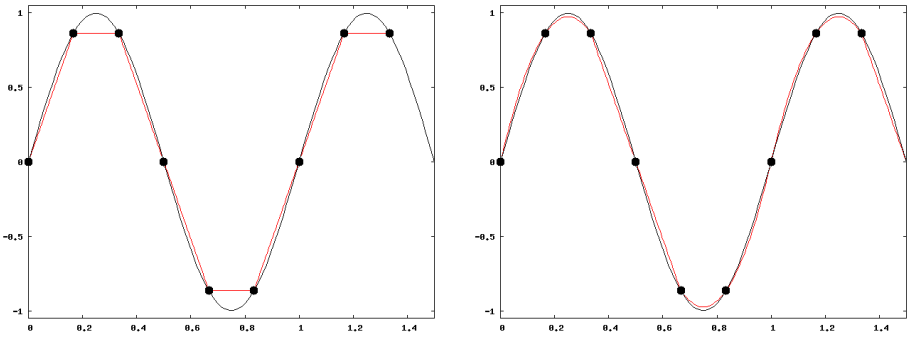
Figure 1.1: 3D steady computation of the flow around a sphere at Re=270 by the 4th order accurate version of Argo, illustrated by stream lines. The mesh used for computing the solution is defined all around the sphere, up to a relatively large distance. This computation illustrates the complexity of the flow which is already found for very simple geometrical configurations and low flow speeds.

Mesh based computational methods

In most fluid dynamic models a continuous solution is sought, which is by consequence defined everywhere in space. Obviously it is impossible to find this solution, defined in an infinity of points, unless a closed mathematical expression exists. Since this is generally not the case, approximate solutions are sought. Such approximate solutions are parametrised by a finite set of *discrete* variables, for which a corresponding number of equations are defined, based on a modified form of the original equations. Many of *discretisation* methods exist, each with a particular approach and domain of application.

Finite volume (FVM) and *finite element (FEM)* methods - currently the pervasive simulation technologies in aeronautic industry - subdivide the computational domain into a collection of small *control volumes*, also called *elements* in FEM, as illustrated in Fig. 1.1. The discrete variables are attached to these elements, which are then used for the interpolation of the approximated solution, as well as the definition of the discretised equations. The precision of this approximate solution depends obviously on the typical element size or *resolution* of this mesh on the one hand, but also on the *order of convergence* on the other. This order indicates the evolution of the error with h. Typical state-of-the-art industrial CFD codes have second order of accuracy, meaning that the error decreases as h^2.

The impact of order of convergence is shown in Fig. 1.2. A sinusoidal function is interpolated with linear and quadratic functions, parametrised by groups of 2 respectively 3 control points from the same set. Clearly the quadratic interpolation is of much higher quality, although the resolution - characterised by the distance between successive interpolation points - is exactly the same. The order of convergence on the solution is - as a rule of thumb - one order higher than the interpolation order p.



(a) Linear interpolation, second order accuracy (b) Quadratic interpolation, third order accuracy

Figure 1.2: The sinusoidal function (black) is interpolated with linear resp. quadratic functions (red) through 2 resp. 3 successive control points. The total number of control points for both interpolations is the same, yet much higher precision is obtained by the quadratic functions.

Due to their structure, the extension of classical state of the art finite volume and finite element methods to higher orders of convergence is difficult and reduces efficiency and robustness. The current thesis work concerns the industrialisation of a relatively novel discretisation method, the *discontinuous Galerkin method (DGM)* within the numerical platform Argo. DGM is a mix between the finite volume and finite element methods, and allows to obtain arbitrary high orders of convergence on unstructured meshes, thereby greatly surpassing the methods currently used in industry. The high precision provided by DGM is needed for a number of thematics, detailed in the following section.

High-fidelity simulations

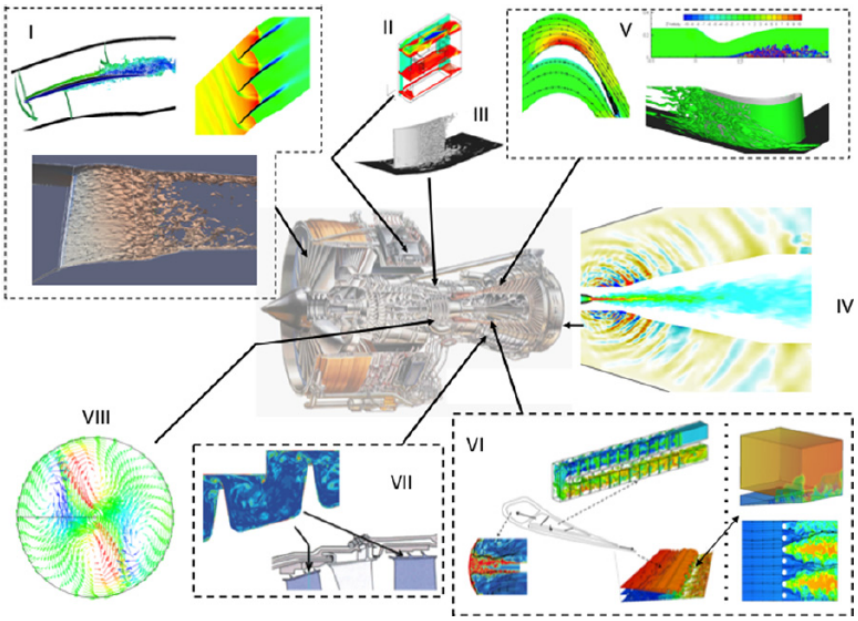


Figure 1.3: Overview of flow regimes characterized by large scale turbulence and transition in jet engines. From Tucker (100)

Within the day-to-day contact with industry and research centres, Cenaero has identified industrial needs for high-fidelity computations for the following thematics.

Large scale turbulence. For a considerable number of industrial applications turbulence manifests itself at scales comparable to the size of the geometry. This includes transitional flows at moderate Reynolds as well as geometrical induced separation in the case of bluff bodies, as shown in Fig. 1.1, flows at high angles of attack, transitional flow ... This large scale turbulence is also

very present in many parts of the jet engine, as illustrated in Fig. 1.3, thereby not even considering off-design regimes and large-scale flow instabilities.

The current industrial practice in CFD is largely based on *Reynolds Averaged Navier-Stokes (RANS)* approaches. These approaches solve for an averaged flow thereby modelling the statistical impact of the stochastic turbulent flow features. For large scale turbulent flows however, RANS is not capable to predict even the time-averaged performance with sufficient precision to support the advanced design of today. It is clear that in these cases, *scale resolving* approaches will be needed, that at least represent the most energetic turbulent features directly, and model the impact of only the smallest *subgrid* structures. Within these techniques we can broadly distinguish three classes of methods, in order of decreasing precision and computational cost: the model-free *Direct Numerical Simulation (DNS)*, large-scale resolving *Large Eddy Simulation (LES)* and finally *hybrids* between statistical models (RANS or boundary layer models) and LES.

The current universal adoption of RANS for design has been motivated by a more or less adequate prediction for well behaved flows and design conditions, combined with tractable computational resource requirements. Typically the computational cost of LES and its derivatives will be orders of magnitude higher. With the increasing availability of computational power, and ever increasing stringency with respect to performance, industry starts to consider LES and hybrids for their routine application within the next 10 years. In particular this would extend the prediction capabilities to a larger set of operating regimes, including instabilities, as well as accurate noise prediction.

In academia, a large effort has been devoted in last decade to develop the physical models for the subgrid scales. In this community, it has since long been recognized that scale-resolving approaches require extremely low dispersion and dissipation errors, in order to avoid contamination of the model with numerical error. Since these models were developed on simple benchmarks, Most of this effort was therefore based on highly structured high-order discretisation technology in order to reduce the computational cost. Unfortunately these discretisations do not have much geometrical flexibility, and are therefore difficult to apply to the complex geometries found in industry.

Much less effort has been dedicated to development of flexible high-order discretisation methods, capable of providing the same precision on complex geometry. This is probably one of the reasons why up to now good results with scale-resolving methods have mainly been obtained for academic cases whilst the few industrial applications, for which state of the art industrial methods were used, have not been quite as successful (see Tucker (101)).

It is clear that the near future of unstructured high-order methods is precisely scale-resolving simulations, as they would provide the necessary combination of high-precision and geometrical flexibility. Apart from the technological considerations, there is obviously a unique window of opportunity. As industry still needs to choose its tools for scale-resolving methods, it is open to the adoption of new numerical tools and discretisations, in complement to the state-of-the-art methods which remain the best tools for RANS. In order to

exploit this momentum, a further convergence between both the high-order and turbulence modeling community is desirable.

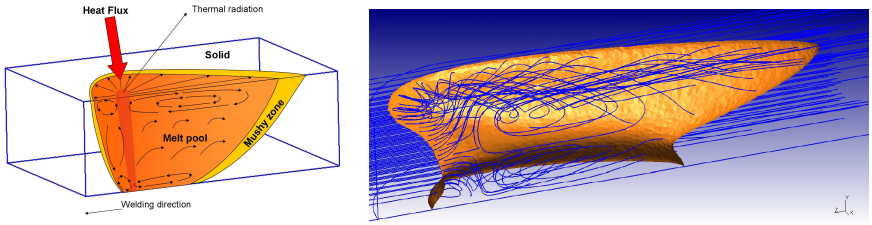
Noise generation and propagation. The reduction of flow induced noise has become a very important design criterion for many applications. These include in particular aircraft, where noise is generated by either the external structure (landing gear, flaps, slats) or the engines. A good illustration of this problematic is the research on *counter rotating (transonic) open rotors (CROR)*. Although this type of engine can easily obtain a specific fuel consumption which is 10 to 15% lower than that of conventional jet engines, its adoption for aircraft propulsion is blocked by the high level of noise that it generates.

As aerodynamic noise *generation* mechanisms are intimately related to turbulence, their prediction requires the use of the scale-resolving approaches discussed in the previous section. Another important issue is the simulation of the *propagating* acoustic signal signal. Noise propagation is also an aerodynamic phenomenon, which consists of coupled pressure and velocity perturbations. The amplitude of these waves is typically many orders of magnitude smaller than the pressure variations occurring due to the mean flow. At the same time its typical spatial frequency is much higher, thus requiring a much higher resolution than what is required to capture the flow. Therefore the acoustic signal and the flow are not easily captured by the same code. Currently dedicated high precision methods, based on a specific sets of linearised equations, are used in complement to CFD codes to predict sound propagation. Within this domain, unstructured high-resolution methods such as the discontinuous Galerkin method have already been widely adopted.

However, the simulation of acoustic waves propagating within complex geometry, possibly with moving boundaries, and their interaction with turbulent flow features is currently not very well mastered. In particular for CROR, where the rotor blades are apparent to the outside world and hence no clear separation between the source and the propagation zone is possible, the latter is extremely important. Moreover, for a number of applications the flow features and noise propagation are difficult to separate. Examples are transonic flows, highly receptive boundary layer or shear layer flows, ...

It is foreseeable that for such applications the propagation of noise in the direct neighbourhood of the engine or aircraft will also need to be taken up directly by the fluid solver, again stressing the need for high-resolution CFD methods.

Production processes. The flow of metal, glass or polymers in many production processes is modeled by non-Newtonian or visco-elastic equations of state, often in combination with phase transition models. The high nonlinearity of these constitutive laws leads to very localised and rapidly varying flow features. To capture these intricate structures, high resolution is required. High-order methods could therefore provide a significant gain, not only in terms of computational cost, but more importantly in terms of computational reliability. Fig. 1.4 for instance illustrates a simulation of a fusion welding process performed at Cenaero with the finite volume version of Argo.



(a) Principle (b) Computed stream traces and melting surface

Figure 1.4: Fusion welding processes proceed by injecting heat in to the weld seam, thereby melting metals on either side. The large temperature variations result in important buoyancy and surface tension effects, leading to very complex flow patterns. Computations were performed with the finite volume version of Argo. The metal is modeled as an incompressible fluid with temperature-dependent viscosity and phase transition. (Poletz *et al.* (87))

State of the art in numerical technology and new developments

There is a growing consensus that state of the art industrial CFD technology, consisting mainly of 2^{nd} order FVM, will require too extensive computational resources to provide the high precision for above mentioned thematics, even at the rate that the available computational power increases. The evolution towards ever more reliable quantitative predictions then naturally leads us to consider methods which have a higher order of grid convergence.

The extension of FVM to higher order of convergence compromises their computational robustness and efficiency. Recently, new unstructured high-order discretisations have emerged, which combine high precision similar to that of spectral and finite difference methods to the geometric flexibility characteristic of commercial codes, whilst providing computational efficiency. These discretisations seem much more plausible candidates for providing the core of new high-resolution CFD codes; an overview was recently published by Vincent *et al.* (106).

The *discontinuous Galerkin method (DGM)*, considered in this thesis, is one of these methods. It belongs to an important subclass, which is composed of discretisations that are based on cellwise independent or discontinuous interpolation. Other important methods in this class are the *spectral volume (SVM)* (107; 76; 97), the *spectral difference (SDM)* (75; 79) and the *flux reconstruction method (FRM)* (64). The latter provides to some extent a unifying framework for the DGM, SDM and SVM (see Vincent *et al.* (105)).

Due to the locality of the data, typical for discontinuous interpolation, and the algorithmic density, characteristic of high-order methods, all of the methods in this class can be implemented very efficiently on most architectures and can be made to scale very well on parallel machines. Furthermore, due to the discontinuity of the interpolation, both mesh resolution and order of interpolation can be chosen locally, allowing non-conforming connections (*i.e.* between incompatible interpolations) without the need for modifying the ap-

proach and hence without degradation of the quality of the solution. This important feature provides the practical framework for adaptive strategies, which will be undoubtedly needed to further reduce the computational cost and enhance solution reliability of future large scale simulations.

Of these methods, DGM is currently the most mature. It provides a stable discretisation on all element types, which is not the case for any of the other methods. It is furthermore underpinned by a complete and rigorous mathematical framework, within which the necessary tools for defining both order and mesh sise adaptation criteria have already been developed. On the basis of these qualities, it was chosen to further develop this method.

Towards automated design ?

Industrial design relies already today often on automated optimisation chains. Since human intervention is absent in the process, the quality of the numerical technology is crucial. First of all, the computational meshes are generated automatically, without any user intervention to check and assure the quality and resolution of the mesh. Secondly, no *a posteriori* verification of the validity and quality of the results is possible. Since the optimisation relies on quantitative results only, (the quantification of) the grid convergence of the computation becomes of paramount importance. In conclusion one needs to provide methods, which converge fast in terms of resolution, and are robust with respect to (severely) distorted meshes.

Unstructured high-order methods, in combination with adaptive resolution, can in theory go a very long way towards that goal. However, automatised design chains will rely mostly on RANS simulations for a long time to come, given the large disparity in computational cost with LES and the large number of computations that need to be run. It is clear that in the case of RANS, high-order methods are to date still not competitive with industrial state of the art unstructured finite volume codes.

This is to some extent due to the relatively low solution precision that is currently accepted, which is in practice far from grid independence. These low precision requirements are not surprising, given the uncertainty on the turbulence and transition models. However, in particular in combination with adaptative strategies, unstructured high-resolution methods can already drastically reduce the mesh dependence of the solution, thereby reducing the number of anomalous results. Further acceptance will both require a huge increase in computational efficiency through the development of more efficient iterative strategies and adaptation, and a considerable increase in the accuracy of the turbulence models, justifying the increase in solution precision.

1.2 Overview

This thesis describes the development and industrialisation of a high-order discontinuous Galerkin method based code, called Argo, for its application to fluid dynamic problems.

The first section discusses the main ingredients of the discontinuous Galerkin method. A very important concept is the reinterpretation of the method as a combination of elementwise defined finite element problems coupled by internal boundary conditions. This reinterpretation is exploited in many of the developments of the method, either in the development of efficient data structures and assembly or the application of high-quality transfer operators for multilevel iterative methods, and filters for non-conforming methods.

Chapters 3 through 5 discuss the contributions of this thesis. Although the outset of the work is a practical one, a number of more fundamental developments were undertaken:

- the definition and study of a generic framework for hp-multigrid transfer operators and the proof of the optimality of the latter. This is detailed in section 4.2 and publications (57; 59). This work has been applied as one of the ingredients of agglomeration multigrid for DGM by P. Tesini (99);
- the proposal of an instability mechanism for quadrature-free and simplified quadrature procedures, described in appendix D and (57). A further investigation of this instability mechanism was recently undertaken by Bassi *et al.* (11);
- the study and definition of memory and time efficient data structures and assembly routines for the discontinuous Galerkin method, based on the optimal use of BLAS operator primitives, described in chapter 5 and the publications (58; 56). Further refinements of this assembly have been elaborated in the doctoral thesis of J. Lambrechts (73);
- the extension of the stability analysis of the interior penalty method to hybrid meshes with optimal penalty parameters, described in section 3.1 and appendix C. These developments have been submitted for publication (60) and applied in joint publications with M. Drosson (40; 41) which have recently been accepted for publication;

Next to these developments, a number of more practical developments were made

- one of the first Jacobian-free Newton-GMRES method for DGM (see section 4.1);
- a post-processing methodology to implement a non-conforming approach for low Reynolds flows, both in Navier-Stokes and Stokes formulations. These flow regimes are typical for production processes featuring the flow of liquefied metal, glass, polymers ... This is detailed in 3.2 and used in (86).
- the implementation of frequential acoustic solvers based on the homentropic *linearized Euler Equations (LEE)* and associated *perfectly matched layers (PML)*, including the proof of stability of the temporal derivative terms. This is detailed in section 3.3.

Chapter 6 then discusses the direct numerical simulation of the transitional flow around a low pressure turbine blade, which was the first industrial application. Chapter 7 summarises the results obtained during the thesis and discusses subsequently the research directions that will have been identified and will be pursued in the near future.

CHAPTER 2

THE DISCONTINUOUS GALERKIN METHOD

This chapter summarises the mathematical background of the *discontinuous Galerkin / Interior penalty method (DGM)*. The main idea is the reinterpretation of the method as a combination of element-wise finite element problems, coupled by internal boundary conditions. This reinterpretation is key to the understanding the precision, flexibility, computational efficiency and solution-adaptivity of the discretisation.

A first section will discuss the nature of the equations to be solved. The following sections will elaborate the basic ingredients of the numerical technique used to find an approximate solution to these equations.

Appendix A provides a summary background in functional analysis needed for the comprehension of this chapter. A more comprehensive treatment is found in Reddy (90), Ciarlet (34) or Braess (22).

2.1 Model equations

To start the discussion the relevant physical models are cast into the following generic system of partial differential equations

$$\frac{\partial \tilde{u}_m}{\partial t} + \nabla \cdot \vec{f}_m\left(\tilde{u}\right) + \nabla \cdot \vec{d}_m\left(\tilde{u}, \nabla \tilde{u}\right) + s_m = 0\ ,\ m = 1\ \ldots\ N_v \tag{2.1}$$

The solution $\tilde{u}$ has N_v components and is defined on the domain Ω. Appropriate boundary conditions are prescribed on its boundary Γ.

The physical significance of these equations can be seen more clearly when considering the integral on a generic volume V, with outward normal $\vec{n}$.

$$\frac{\partial}{\partial t}\int_V \tilde{u}_m dV + \oint_{\partial V}\left(\vec{f}_m\left(\tilde{u}\right) + \vec{d}_m\left(\tilde{u}, \nabla \tilde{u}\right)\right)\cdot \vec{n}\, dS + \int_V S_m dV = 0 \tag{2.2}$$

In this form one sees that the equations (2.1) describe the conservation of $\tilde{u}$ on the volume V, whereby $\vec{f}$ and $\vec{d}$ describe different components of the *flux* or *flow* of $\tilde{u}$ through the boundary ∂V of the volume V. However, both fluxes describe fundamentally different phenomena. This difference in nature will reflect itself in the way boundary conditions are imposed as well as in the discretisation and its analysis.

Convective terms The terms $\vec{f}$ describe the transport of the state vector $\tilde{u}$. The most simple example is the advection of a scalar quantity $\tilde{u}$ with given velocity $\vec{a}$

$$\vec{f} = \vec{a}\tilde{u} \tag{2.3}$$

The convective terms introduce directionality in the solution and by consequence in the model equations, which will need to be taken into account in the numerical method. The most conspicuous consequence is that boundary conditions will only be imposed if the advection velocity $\vec{a}$ is entering the domain.

Diffusive terms The terms $\vec{d}$ describe diffusion, a process which tends to uniformise the value of the field $\tilde{u}$ across the domain. Thereto it introduces a flux opposing the gradients of the solution, from high to lower values. The most simple example is the Fourier heat conduction equation, stating that the flow of heat flux goes from high to low temperature, opposite but proportional to its gradient

$$\vec{d} = -k\nabla\tilde{u} \tag{2.4}$$

This form of the heat flux introduces a more omnidirectional character, which tends to uniformise the temperature on the whole domain. As a consequence conditions will be required on all of the boundaries.

For further use $\bar{D}$ represents the Jacobian of the diffusive flux $\vec{d}$ with respect to the solution gradients. A linear diffusive flux vector $\vec{d}$ would then read

$$d_m^k = \bar{D}_{mn}^{kl} \frac{\partial \tilde{u}_n}{\partial x^l} \tag{2.5}$$

Here one should note that Einstein notation has been used, *i.e.* summation occurs on repeated indices.

Source terms Finally the source terms s_m group all effects that cannot be expressed as fluxes. The most important consideration is that, in contrast to the fluxes that can be treated in a generic fashion, a particular treatment will be required depending on the physics at hand.

2.2 Elements and functional spaces

One needs an infinite number of values to characterise the solution $\tilde{u}$ of the equations described in the previous section. This is of course untractable, so approximate solutions will be computed. The discontinuous Galerkin method,

as any *finite element method (FEM)*, proposes an approximate solution belonging to a finite-dimensional *trial space* $\mathcal{V}$. The elements of $\mathcal{V}$ are vectors of N_v functions, one for each variable.

To construct $\mathcal{V}$, the domain Ω is subdivided into a finite number of *elements* which serve as a support for the definition of its member functions. The collection of all elements is noted as $\mathcal{E}$ and referred to as *mesh* or *grid*.

$$\Omega \approx \mathcal{E} = \cup e \tag{2.6}$$

In theory all types of elements are allowed, as long as their union covers the entire domain without overlapping. In Argo the element types are restricted to triangles and quadrilaterals in two dimensions, and tetrahedra, prisms, hexahedra and pyramids in three dimensions. Moreover conformal meshes are used, meaning that any boundary face of the element can only connect to a single other element.

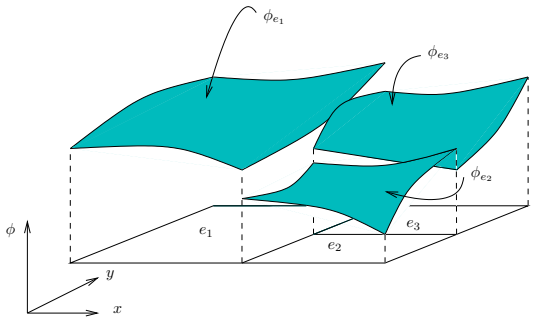


Figure 2.1: Non-conformal discontinuous finite element interpolation.

The trial space is then composed of functions that are regular on the interior of each of the elements e, but not necessarily continuous across the element boundaries f, see Fig. 2.1. Since the formulation allows for discontinuous interpolation, the use of non-conformal connections is trivial, and does not incur the loss of precision.

In this work the trial space $\mathcal{V}$ is composed of vectors of polynomial functions of *interpolation order* p, defined in the *parametric coordinates* ξ. These parametric coordinates are defined in a reference element, which is the same for each of the elements of the same type and interpolation order. For each of the elements a particular transformation or *mapping* needs to be defined to convert the physical coordinates (x_1, x_2, x_3) into to the parametric coordinates (ξ_1, ξ_2, ξ_3) in the standard element as illustrated in Fig. 2.2. Again this is a choice specific to Argo. In general the absence of continuity at the element interfaces allows for a fully arbitrary choice of interpolation functions. This can be exploited to integrate particular solutions of the equations of state (*e.g.* plane wave solutions in acoustics or electromagnetics) or to allow an element independent interpolation for agglomeration multigrid by Tesini (99). For a detailed description of the elements and corresponding functional spaces within Argo, the reader is referred to appendix B.

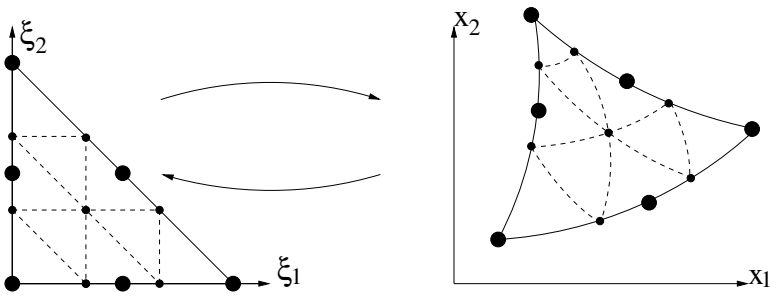


Figure 2.2: Third order mapping (small control points) of a triangle with second order interpolation (large control points) – or vice versa. The control points can be used to define Lagrange polynomials or splines.

It is clear that $\mathcal{V}$ is a *vector space*: any linear combination of its elements belongs again to $\mathcal{V}$ (see appendix A). By choosing an inner product operator, $\mathcal{V}$ can be promoted to a Hilbert space. Classically the broken Sobolev space of order s is chosen. In this case the inner product is defined as

$$(u, v) = \sum_e \sum_{|\alpha| \leq s} \int_e D^\alpha u D^\alpha v dV \tag{2.7}$$

The broken Sobolev space is defined in analogy with the classical Sobolev space: the inner product is constructed as the sum of Sobolev inner products evaluated per element. Due to the regularity of the functions within each element, D^α is a regular partial derivative, instead of its weak counterpart

$$\begin{aligned} D^\alpha u &= \frac{\partial^{|\alpha|} u}{\partial {x_1}^{\alpha_1} \dots \partial {x_d}^{\alpha_d}} \\ |\alpha| &= \sum_{k=1}^{d} \alpha_k \end{aligned} \tag{2.8}$$

which is used for classical finite element spaces. Furthermore we should note that for at most second order partial derivatives, such as found in classical convection-diffusion problems treated in this text, the Sobolev order 1 is sufficient to define and analyse the discontinuous Galerkin variational form, which is detailed in the following section.

2.3 Variational formulation

The last ingredient is the definition of a well-posed set of equations that will identify the approximate solution within $\mathcal{V}$. In the case of DGM, this set is provided by a *Galerkin variational formulation*: the residual of Eq. 2.1, expressed with the approximate solution should be formally orthogonal to any of the functions within the trial space itself.

Generic variational formulation

First a further generalisation of Eq. 2.1 is introduced:

$$\frac{\partial u_m}{\partial t} + \nabla\cdot\vec{g}_m = 0 \tag{2.9}$$

The Galerkin variational form can then be rewritten, using an elementwise decomposition and integration by parts

$$\begin{aligned}
&\int_\Omega v_m \frac{\partial u_m}{\partial t} dV + \int_\Omega v_m \nabla\cdot\vec{g}_m dV = 0\ ,\ \forall v \in \mathcal{V} \\
&\qquad\qquad \Downarrow \\
&\sum_e \int_e v_m \frac{\partial u_m}{\partial t} dV + \sum_e \int_e v_m \nabla\cdot\vec{g}_m dV = 0\ ,\ \forall v \in \mathcal{V} \\
&\qquad\qquad \Downarrow \\
&\sum_e \int_e v_m \frac{\partial u_m}{\partial t} dV - \sum_e \left(\int_e \nabla v_m \cdot \vec{g}_m dV + \int_{\partial e} v_m \vec{g}_m \cdot \vec{n}\, dS \right) = 0\ ,\ \forall v \in \mathcal{V}
\end{aligned} \tag{2.10}$$

where e denote an element and $f \in e$ its facets.

Trace operators and interface fluxes Converting the sum on the element faces by the sum on all of the element interfaces f the formulation (2.10) becomes

$$\begin{aligned}
\sum_e \int_{\partial e} v_m \vec{g}_m \cdot \vec{n}\, dS &= \sum_{f\in e} \int_f v_m \vec{g}_m \cdot \vec{n}\, dS \\
&= \sum_f \int_f \left(v_m^+ \vec{g}_m^+ - v_m^- \vec{g}_m^- \right) \cdot \vec{n}\, dS \\
&= \sum_f \int_f \left(v_m^+ \vec{g}_m^+ \cdot \vec{n}^+ + v_m^- \vec{g}_m^- \cdot \vec{n}^- \right) dS
\end{aligned} \tag{2.11}$$

The $+$ and $-$ distinguish limit values of the discontinuous quantities on either side of an the interface, depending on the element from which the interface is approached. The $+$ sign corresponds to the element of which its exterior normal corresponds to that of the oriented face, whilst $-$ corresponds to the other. The exterior normals of these elements on the face f are denominated as $\vec{n}^+$ and $\vec{n}^-$ respectively. Defining the jump $[[.]]$ and the average operator $\langle . \rangle$ as

$$\begin{aligned}
[[a]] &= a^+\vec{n}^+ + a^-\vec{n}^- \\
[[\vec{a}]] &= \vec{a}^+ \cdot \vec{n}^+ + \vec{a}^- \cdot \vec{n}^- \\
\langle a \rangle &= \left(a^+ + a^- \right)/2
\end{aligned} \tag{2.12}$$

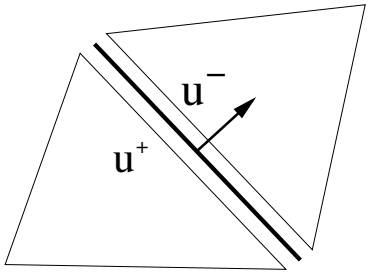


Figure 2.3: Convention for suffixes $+$ and $-$ when approaching the interface (shown by the thick line). These indicate the limit values of discontinuous quantities when approaching the interface along the direction and opposite to the direction of its normal, indicated by the arrow, from within the adjacent cells.

we find

$$\begin{aligned}\sum_e \oint_{\partial e} v_m \vec{g}_m \cdot \vec{n}\, dS &= \sum_f \int_f [[v_m g_m]] dS \\ &= \sum_f \int_f \left([[v_m]] \langle g_m \rangle + \langle v_m \rangle [[g_m]] \right) dS\end{aligned} \tag{2.13}$$

All discontinuous Galerkin methods, both for the convective and diffusive equations, replace the term (2.13) by providing suitable interface flux functions γ, such that the final variational formulation is formally given by

$$\begin{aligned}\sum_e \int_e v_m \frac{\partial u_m}{\partial t} dV - \sum_e \int_e \nabla v_m \cdot \vec{g}_m dV \\ + \sum_f \int_f \gamma_m(u^+, u^-, v^+, v^-, \vec{n})\, dS = 0\ ,\ \forall v \in \mathcal{V}\end{aligned} \tag{2.14}$$

Local reinterpretation In order to obtain that the variational formulation of Eq. 2.14 is satisfied $\forall v \in \mathcal{V}$, of course one does not need to test all functions. Any set of test functions v_m can be chosen, if this set forms a basis for the vector space $\mathcal{V}$. A particular class of such sets are composed of functions with elementwise support, *i.e.* which are non-zero only on one of the elements:

$$\int_e v_m \frac{\partial u_m}{\partial t} dV - \int_e \nabla v_m \cdot \vec{g}_m dV + \sum_{f \in e} \int_f \gamma_m(u^+, u^-, v^+, 0, \vec{n})\, dS = 0 \tag{2.15}$$

On each of these elements, again a local basis can be chosen. Pursueing this logic, it is clear that this formulation can be reinterpreted as a local Galerkin finite element problem for the solution u defined on the element e only. γ then provides the flux boundary conditions that link e to its neighbours. This interpretation of γ as implementing "internal" Dirichlet-type boundary conditions directly provides a guiding principle for their definition. Furthermore

the interface fluxes will be chosen that two generic conditions are satisfied: the resulting method should be consistent and conservative.

Condition 1 - consistency In order to find a unique solution, at least for linear problems, the Lax equivalence theorem states that it suffices that the variational formulation is *stable* and *consistent*. At this stage, we will not go into stability as this issue is closely related to the nature of the flux term and will be discussed in the following subsections. Consistency on the other hand means that the interface flux is such that for exact solution

$$\lim_{h\to 0} \gamma_m(u^+, u^-, v^+, v^-, \vec{n}) = [[v_m]]\langle \vec{g}_m \rangle \tag{2.16}$$

Notice that for this reason, we can remove the term $\langle v_m \rangle [[\vec{g}]]$, hence imposing implicitly the continuity of the flux through the face[1].

Condition 2 - conservativity Starting from the elementwise formulation (2.15) one can see that the DGM formulation can be made elementwise conservative. Plugging in $v_m = 1$ on the corresponding element one finds

$$\int_e \frac{\partial u_m}{\partial t} dV - \sum_{f\in e} \int_f \gamma_m(u^+, u^-, 1, 0, \vec{n})\, dS = 0 \tag{2.17}$$

In case

$$\gamma_m(u^+, u^-, 1, 0, \vec{n}) = -\gamma_m(u^+, u^-, 0, 1, \vec{n}) \tag{2.18}$$

the flux leaving the element through a face, is completely recuperated by its neighbour.

Both the convective and diffusive flux are discretised separately, up to the point that the convergence and stability analysis are radically different. In particular the convective equations follow an approach which is very akin to the methods used in finite volumes, whereas for the diffusive terms a typical finite element framework is used.

Convective variational form

The discretisation of the convective part of the equations is always stabilised by choosing an (approximate) Riemann solver (see Cockburn (35)) for γ.

Starting from the interpretation of DGM as local FEM coupled by boundary conditions (2.15), the obvious choice is to use characteristic boundary conditions. This can be provided by the use of a (n approximate) Riemann solver flux or even a monotone flux $\mathcal{H}$. Then the interface flux in (2.15)

$$\gamma_m(u^+, u^-, v^+, 0, \vec{n}) = v^+ \mathcal{H}(u^+, u^-, \vec{n}) \tag{2.19}$$

This is only a heuristic interpretation. In the slightly more rigorous following paragraphs DGM will be shown to be a straightforward extension of a

[1]Continuous finite element methods, although most often not stated, implicitly impose the same continuity by omission of the interface terms during the formal integration by parts.

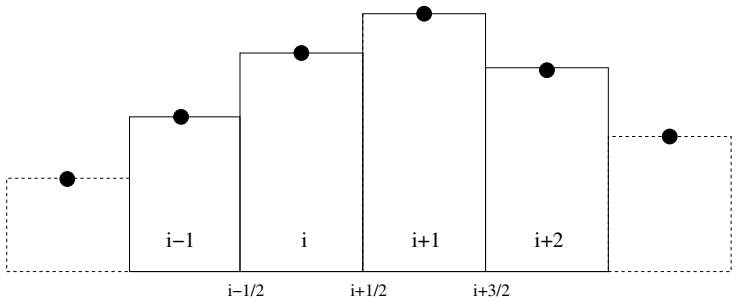


Figure 2.4: The one-dimensional Godunov first order finite volume scheme solves for the cell-wise averages. The discrete equations are found by expressing a flux balance on each of the cells. Thereto flux functions are defined on the interfaces between elements which are computed from the left and right state. This method can be considered as a DG method with constant or zeroth order polynomial interpolants.

finite volume method. As a consequence energy stability can be obtained by upwind fluxes. These paragraphs do not intend to give a thorough analysis, only to give the basic feel of the method.

Finite volumes The most classical finite volume schemes is the first order upwind scheme. It solves for the constant state in each of the elements of the mesh, as illustrated in Fig. 2.4 for the one-dimensional version. Here volumes correspond to integer and interfaces to fractional indices. The Godunov scheme computes the evolution of the volume average using the cell flux balances over the boundaries of the element. Thereto a common flux on the interfaces in between two elements is defined. It is a function of the constant states on either side. For a scalar equation the semi-discrete scheme then reads

$$\frac{du_m^e}{dt} = -\frac{1}{V^e}\sum_f \mathcal{H}_m(u^e, u^f, \vec{n}^f) \ , \ \forall e \tag{2.20}$$

where the notation u^e is obviously used for the internal state of the element e, and u^f is used for the outside state (u^-) at interface f. The flux is required to be

- *consistent:* $\mathcal{H}_m(u, u, \vec{n}) = \vec{f}_m(u) \cdot \vec{n}$
- *conservative:* $\mathcal{H}_m(u^-, u^+, -\vec{n}) = -\mathcal{H}_m(u^+, u^-, \vec{n})$

In case of linear problems, the Lax equivalence theorem states that *stability* is the only requirement in addition to *consistency* to find a *unique* and *mesh converging* solution.

For illustrating stability, a scalar problem is considered for simplicity. Since each of the elements in the mesh is closed, and hence $\sum_f \vec{n}^f = 0$ one finds the

flux

$$\begin{aligned}\frac{du^e}{dt} &= -\frac{1}{V^e}\sum_f \mathcal{H}(u^e, u^f, \vec{n}^f) - \vec{f}(u^e)\cdot\vec{n}^f \\ &= \frac{1}{V^e}\sum_f \left(\frac{\partial \mathcal{H}}{\partial u^-}(u^e, \tilde{u}, \vec{n}^f)\right)(u^e - u^f)\ ,\ \tilde{u}\in[u^e, u^f]\end{aligned} \tag{2.21}$$

since consistency of $\mathcal{H}$ and the midpoint rule give

$$\begin{aligned}\mathcal{H}(u^e, u^f, \vec{n}^f) - \vec{f}(u^e)\cdot\vec{n}^f &= \mathcal{H}(u^e, u^f, \vec{n}^f) - \mathcal{H}(u^e, u^e, \vec{n}^f) \\ &= \frac{\partial \mathcal{H}}{\partial u^-}(u^e, \tilde{u}, \vec{n}^f)(u^e - u^f)\end{aligned} \tag{2.22}$$

A *monotone* flux for a scalar equation [2] is then defined as

$$\begin{aligned}&\frac{\partial \mathcal{H}}{\partial u^-}(u^+, u, \vec{n}) < 0\ ,\ \forall u\in[u^+, u^-] \\ &\frac{\partial \mathcal{H}}{\partial u^+}(u, u^-, \vec{n}) > 0\ ,\ \forall u\in[u^+, u^-]\end{aligned} \tag{2.23}$$

If such a flux is used the Godunov scheme is positive, *i.e.* can be rewritten as

$$\frac{du^e}{dt} = \frac{1}{V^e}\sum_{f\in e} C^{fe}\left(u^f - u^e\right) \tag{2.24}$$

for which all C^{fe} are positive. Hence can choose the time integration such that each new value u^e is a convex combination of its neighbours and its previous value, implying that no new extrema can be created. In case of a steady solution, one also finds that each element value is a convex combination of its neighbours. Positivity also immediately implies energy stability and a discrete entropy inequality. The latter implies the correct choice in case of multivalued solutions in case characteristics diverge (see Leveque (74)).

Discontinuous Galerkin Using the monotone flux $\mathcal{H}$ as interface fluxes

$$\gamma_m\left(u^+, u^-, v^+, v^-, \vec{n}\right) = \left([[v]]\cdot\vec{n}\right)\mathcal{H}_m\left(u^+, u^-, \vec{n}\right) \tag{2.25}$$

the discontinuous Galerkin scheme becomes

$$\begin{aligned}\sum_e \int_e v_m \frac{\partial u_m}{\partial t} dV - \sum_e \int_e \nabla v_m\cdot\vec{f}_m dV \\ + \sum_f \int_f \left([[v_m]]\cdot\vec{n}\right)\mathcal{H}_m\left(u^+, u^-, \vec{n}\right) dS = 0\ \ \forall v\in\mathcal{V}\end{aligned} \tag{2.26}$$

Obviously the method reduces to the Godunov scheme for interpolation order $p = 0$, and hence the DGM scheme can be seen as a higher order extension of

[2] System monotone fluxes have positive resp. negative semi-definite Jacobians.

the latter. Looking at the properties of the scheme, *consistency* and *conservation* are immediately recuperated. The scheme is however not positive for an interpolation order ($p \geq 1$), but energy stability [3] and elementwise entropy inequalities can be obtained, as shown by Jiang and Shu (66).

To show stability, again a scalar problem is considered. First the primitive of the flux function is defined

$$\vec{F}(u) = \int^u \vec{f} du \tag{2.27}$$

Chosing $v = u$ in the DGM formulation (2.26) and repeatedly applying the midpoint rule

$$\begin{aligned}
&\sum_e \int_e u \frac{\partial u}{\partial t} dV = \sum_e \int_e \nabla u \cdot \vec{f} dV - \sum_f \int_f [[u]] \cdot \vec{n} \mathcal{H}(u^+, u^-, \vec{n}) dS \\
&\qquad \Downarrow \\
&\sum_e \int_e \frac{\partial}{\partial t} \frac{u^2}{2} = \sum_e \int_e \nabla \cdot \vec{F} dV - \sum_f \int_f [[u]] \cdot \vec{n} \mathcal{H}(u^+, u^-, \vec{n}) dS \\
&\qquad \Downarrow \\
&\sum_e \int_e \frac{\partial}{\partial t} \frac{u^2}{2} = -\sum_f \int_f [[u]] \cdot \vec{n} \mathcal{H}(u^+, u^-, \vec{n}) - \left[\left[\vec{F}\right]\right] dS \\
&\qquad \Downarrow \exists \hat{u} \in [u^e, u^f] : \left[\left[\vec{F}\right]\right] = \vec{f}(\hat{u}) \cdot \vec{n} \left(u^e - u^f\right) \\
&\sum_e \int_e \frac{\partial}{\partial t} \frac{u^2}{2} = -\sum_f \int_f [[u]] \cdot \left(\vec{n} \mathcal{H}(u^+, u^-, \vec{n}) - \vec{f}(\hat{u})\right) dS \\
&\qquad \Downarrow \\
&\sum_e \int_e \frac{\partial}{\partial t} \frac{u^2}{2} = -\sum_f \int_f (u^+ - u^-) \left(\left(\frac{\partial \mathcal{H}}{\partial u^+} \right)_{\tilde{u}} (u^+ - \hat{u}) - \left(\frac{\partial \mathcal{H}}{\partial u^-} \right)_{\bar{u}} (\hat{u} - u^-) \right) dS
\end{aligned} \tag{2.28}$$

with $\hat{u} \in [u^+, u^-]$, $\tilde{u} \in [u^+, \hat{u}]$ and $\bar{u} \in [\hat{u}, u^-]$. Evidently, equation (2.28) shows that monotone fluxes guarantee energy stability.

Diffusive variational form

Through the years, a large number of methods has been proposed for the discretisation of the viscous terms, independently or as an addition to the DGM method for convective problems. Consistent with this outset, one can broadly speaking distinguish two corresponding design philosophies. *Interior penalty methods* have been developed indepently for pure elliptic/parabolic problems, whilst methods based on lifting operators emanate originally from the convective DGM community. A historic overview as well as a common framework

[3] Positivity would guarantee L_∞ stability, meaning that solution extrema cannot be increased. This feature is a primary ingredient for shock capturing strategies.

for the study and classification, was elaborated by Arnold *et al.* (7) [4]. Only the basic philosophy of the approach chosen for Argo, namely the *interior penalty method (IP)*, will be elaborated.

The main reason for choosing the interior penalty method is its compacity: only the direct neighbours of the element are used to evaluate the residual. This greatly simplifies the structure and evaluation of the Jacobian. The second scheme of Bassi and Rebay (BR2) has the same compacity, but since it is based on *lifting operators* the interface term is more complicated to implement, in particular for systems. The main drawback of the IP approach with respect to the BR2 scheme is its dependence on a tunable, seemingly arbitrary parameter, for which in literature very few precise expressions exist. However, during this thesis very adequate and strict values for this parameter have been elaborated, as described in section 3.1.

The starting point of the method corresponds very well with the reinterpretation of DGM as elementwise FEM problems, as it corresponds to the use of a typical weak imposition method of Dirichlet boundary conditions.

Boundary penalty methods Consider a simple elliptic problem on domain Ω with Dirichlet boundary conditions on $\partial\Omega$

$$\begin{aligned} \nabla\cdot(\mu\nabla\tilde{u}) &= 0 \ \ \forall x \in \Omega \\ \tilde{u} &= u^* \ \ \forall x \in \partial\Omega \end{aligned} \tag{2.29}$$

for which an approximate finite element solution $u \in \mathcal{V}$ is sought using the following Galerkin variational formulation:

$$\int_\Omega \nabla v \mu \nabla u dV = 0 \ , \ \forall v \in \mathcal{V} \tag{2.30}$$

Obviously, if the imposed boundary data is "rough", the direct imposition of the value at the boundary, *e.g.* as illustrated by the red line in Fig. 2.5 for a nodal finite element problem, is not very appropriate. One would rather prefer an approach that minimises the average interpolation error between u and u^*. This can be obtained by adding an additional variational term DP, which penalizes the difference $u - u^*$

$$\begin{aligned} \int_\Omega \nabla v \mu \nabla u dV + \underbrace{\int_{\partial\Omega} \sigma v (u - u^*) dS}_{DP} - \underbrace{\int_{\partial\Omega} v \mu \nabla u \cdot \vec{n}^f dS}_{DD} \\ \underbrace{- \theta \int_{\partial\Omega} (u - u^*) \, \mu \nabla v \cdot \vec{n}^f dS}_{DT} = 0 \end{aligned} \tag{2.31}$$

Since the value of u is now also computed on the boundary, the third term DD needs to be added for consistency. The last term is consistent irrespective of

[4] One should note that since then non-conformal or so-called hybridised DGM schemes have been developed. However, these fall outside of the class of methods used for this work.

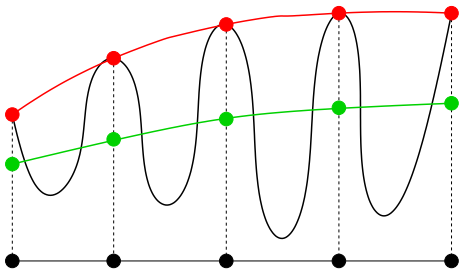


Figure 2.5: In case rough data are imposed at the boundary, the standard node-wise imposition (red) of the Dirichlet boundary condition will lead to a large error and potentially very rapidly varying values. The weak imposition through a penalty (green), as introduced by Nitsche, will in this case result in a much smoother value distribution, with a lower global error.

the value of θ. For $\theta = 1$ a symmetric formulation is found, which is attributed to Nitsche (81). According to the Lax-Milgram theorem (see section A.3) the solvability of the variational problem (2.31) is guaranteed if the associated bi-linear form $a(.,.)$

$$a(u,v) = \int_\Omega \nabla v \cdot \mu \nabla u dV + \underbrace{\int_{\partial\Omega} \sigma v u dS}_{DP} - \underbrace{\int_{\partial\Omega} v \mu \nabla u \cdot \vec{n}^f dS}_{DD} - \underbrace{\theta \int_{\partial\Omega} u \mu \nabla v \cdot \vec{n}^f dS}_{DT} = 0 \tag{2.32}$$

is coercive

$$\exists C > 0 : a(v,v) \geq C||v|| \ , \ \forall v \in \mathcal{V} \tag{2.33}$$

Developing $a(v,v)$ one finds

$$a(v,v) = \int_\Omega \mu \nabla v \cdot \nabla v + \underbrace{\int_{\partial\Omega} \sigma v^2 dS}_{DP} \underbrace{- (1+\theta) \int_{\partial\Omega} v \mu \nabla v \cdot \vec{n}^f dS}_{DD+DT} \tag{2.34}$$

The contribution of DD+DT is not consistently positive, and hence needs to be dominated by the first two terms. Therefore the penalty parameter σ needs to be larger than a certain critical value, unless $\theta = -1$. In the latter case, $\sigma > 0$ suffices. However, the symmetric version is usually preferred, not only to allow the use of more performant solvers for symmetric problems, but also because the formulation is adjoint consistent, leading to *a.o.* better convergence of a posteriori error estimates for functional outputs (see Hartmann (52) and Harriman *et al.* (50) for a discussion in the framework of DGM).

Interior penalty methods Following Wheeler (111), Nitsches approach can be used to couple the solution between elements. Naive application to equation (2.15) would result in

$$\gamma(u^+, u^-, v^+, 0, \vec{n}^f) = \sigma[[u]]v^+\vec{n}^f - v^+\mu\nabla u^+ \cdot \vec{n}^f - \theta\nabla v^+ \mu[[u]] \tag{2.35}$$

and hence

$$\begin{aligned}\gamma(u^+,u^-,v^+,v^-,\vec{n}^f) &= \gamma(u^+,u^-,v^+,0,\vec{n}^f)+\gamma(u^+,u^-,0,v^-,\vec{n}^f)\\ &= \sigma[[u]][[v]]-[[\mu v\nabla u]]-2\theta\langle\nabla v\rangle\cdot\mu[[u]]\\ &= \sigma[[u]][[v]]-\mu[[v]]\cdot\langle\nabla u\rangle-\mu\langle v\rangle[[\nabla u]]-2\theta\langle\nabla v\rangle\cdot\mu[[u]]\end{aligned} \tag{2.36}$$

No choice of θ will result in a symmetric formulation. However, both $[[\tilde{u}]]=0$ and $[[\nabla\tilde{u}]]=0$, such that the following formulation is also consistent

$$\gamma(u^+,u^-,v^+,v^-,\vec{n}^f)=\sigma[[u]][[v]]-\mu[[v]]\cdot\langle\nabla u\rangle-\theta\langle\nabla v\rangle\cdot\mu[[u]] \tag{2.37}$$

Notice that the introduction of $\mu\langle v\rangle[[\nabla u]]$ implicitly imposes continuity of the gradient of the approximate solution, much in the same way as the omission of the interface terms in a continuous finite element method. As for the boundary penalty method, the stability of the method is governed by the *penalty parameter* σ, which has to be higher than a certain critical value.

The choice for θ is arbitrary, however typically three choices are made

- $\theta=1$ leads to the *symmetric interior penalty (SIPDG)* method, which has optimal convergence properties;
- $\theta=-1$ leads to the *non-symmetric interior penalty (NIPDG)* method. This method eliminates the destabilizing effect of the interface terms, and hence only requires $\sigma>0$.
- $\theta=0$ corresponds to the *incomplete interior penalty (IIPDG)* method. This method only has the advantage of simplicity, and is therefore rarely used in practice.

For a system of equations, (2.37) is generalised to

$$\gamma_m(u^+,u^-,v^+,v^-,\vec{n}^f)=\underbrace{\Sigma_{mn}[[v]]_m\cdot[[u]]_n}_{DP}-\underbrace{[[v_m]]\cdot\langle d_n\rangle}_{DD}-\underbrace{\theta[[v_m]]^k\left\langle\bar{D}^{kl}_{mn}\frac{\partial u_n}{\partial x^l}\right\rangle}_{DT} \tag{2.38}$$

where $[[\tilde{u}_n]]^k$ is component k of the jump, while Σ is in theory a symmetric positive definite matrix. In practice a single penalty parameter is chosen for all equations, such that $\Sigma=\sigma\mathbf{I}$.

At the external boundary faces, the same formulation is used, thereby using the jumps in DP and DT to impose Dirichlet conditions, whilst the average of the diffusive flux in DD is replaced, depending on the Neumann conditions.

2.4 Shape functions

At this point the method requires a way to parametrise the approximate solution u, as well as a simple way to express the variational formulation (2.14).

Since $\mathcal{V}$ is a vector space, we can choose a set of linearly independent functions ϕ_{im}, called *shape functions*, to form a basis for $\mathcal{V}$. Each function f can then be written as a unique combination of ϕ_{im}

$$\forall f \in \mathcal{V}, \exists!\ \alpha_{im} : f_m = \sum \alpha_{im}\phi_{im} \tag{2.39}$$

Hereby m is again the index running on the variables. Of course the same applies to the approximation u of $\tilde{u}$, which can be expanded as

$$\tilde{u}_m \approx u_m = \sum_i \mathbf{u}_{im}\phi_{im} \tag{2.40}$$

The variational formulation (2.14) can then be stated as

$$\begin{aligned}\sum_e \int_e \phi_{im}\frac{\partial u_m}{\partial t}dV - \sum_e \int_e \nabla\phi_{im}\cdot\vec{g}_m dV \\ - \sum_f \int_f \gamma_m(u^+, u^-, \phi_i^+, \phi_i^-, \vec{n}^f)\ dS = 0\ ,\ \forall\phi_i\end{aligned} \tag{2.41}$$

since only the variational formulation only needs to be tested with respect to the shape functions.

Since no inter-element continuity is required, one will choose shape functions that are supported on a single element only. The basis functions ϕ_{im} used here are Lagrange interpolants based on equispaced interpolation points μ_i in the reference element. This choice is further elaborated in appendix B.

The main advantage is that on an given face of the element, only shape functions corresponding to interpolation points on this face are non-zero, thereby significantly reducing the computational work for the integration of the interface terms.

Going to very high order

This locality of the boundary interpolation is not a unique feature of the Lagrange interpolants. For instance, although they are not interpolatory, Bézier curves (see Ferguson (44) and Farin (43) for an introduction) are also localised on the boundary. This is a consequence of their definition on the basis on barycentric coordinates. For instance in 1D, the Bézier curve on the interval $[-1, 1]$ and associated to control point i on a total of m, is given by

$$\mathcal{B}_m^i(\xi) = \frac{1}{2^m}\begin{pmatrix} i \\ m \end{pmatrix}(1-\xi)^i(1+\xi)^{m-i} \tag{2.42}$$

It is easy to see that on $\xi = -1$ only $\mathcal{B}_m^0$ will be different from zero, whilst on $\xi = 1$, only $\mathcal{B}_m^m$ will be. Hence these functions are, from an implementation point of view, almost the same as regular Lagrange interpolants.

The quality of a set of shape functions ϕ_i on an element e is determined by the *Lebesgue constant* Le

$$Le = \max_e \sum |\phi_i| \tag{2.43}$$

As Bézier splines are positive on the whole element and sum to 1 everywhere, they will remain bounded functions up to arbitrary order. In particular, the Lebesgue constant will be bounded by the number of functions, This tight control on interpolation error is in fact the main reason that Bézier splines are the work horse for computational geometry.

Classical FEM Lagrange polynomials based on equidistant points have extremal values that grow exponentially with interpolation order. The position of the interpolation points for Lagrange interpolants can be optimised to reduce the Lebesgue constant. It is well-known that for tensor-product elements the Gauss-Lobatto-Legendre quadrature points provide an optimal set of interpolation points. For most elements however these optimal points are not explicitly known and have to be defined by optimisation procedures as proposed by Chen and Babuška (32; 31) and Taylor *et al.* (98), or heuristic approaches as proposed by Hesthaven (54) or Luo and Pozrikidis (77). These approaches have been defined only for simplices.

Bézier splines offer therefore a viable alternative, providing a direct and systematic way of defining interpolation functions up to arbitrary order *for any type of element*. The positivity of the functions furthermore opens possibilities for solution verification and correction with respect to the physics.

CHAPTER 3

EXTENDING THE DG VARIATIONAL FORMULATION

This chapter discusses a number of generic ingredients of the variational formulation that have been developed during the thesis. These ingredients have allowed to tackle specific sets of equations, although they have a much broader scope of application. These extensions are:

- optimal stability parameters for the interior penalty method on hybrid, high-aspect ratio meshes as discussed in section 3.1;
- the implementation of a non-conforming discretisation in a conformal code by post-processing of the residual. This development allows for the computation of low Reynolds incompressible flows, such as used for welding applications. This development is presented in section 3.2;
- finally section 3.3 discusses the implementation of a frequential formulation and *perfectly matched layers (PML)* for acoustic simulations.

3.1 Stability of the interior penalty method on hybrid meshes

Argo was designed to support fully unstructured hybrid conformal meshes, which implies that different element types are supported in a single mesh. This choice is motivated by the geometrical complexity of industrial applications. To date, only tetrahedral mesh generators combine high quality, flexibility and robustness, and are ideal to fill the complex space between the different boundaries of the domain. In the proximity of the solid walls however,

wall-normal extruded meshes are required for precision, in particular to represent the sharp gradients in the boundary layer. Therefore a hybrid mesh is required, including quadrilaterals and triangles in 2 dimensions, and prisms, pyramids and tetrahedra in 3 dimensions.

The discretisation of the convective terms does not explicitly depend on the element type. The stability of the interior penalty method on the other hand is controlled by the stabilisation parameter σ, of which the critical value depends on element size and shape. Although it is often acknowledged that the condition number of the discretized equations is impacted heavily by σ, up to now only in the case of pure triangular or tetrahedral meshes sharp values have been put forward. This section proposes a straightforward extension to other element types and therefore to hybrid meshes. During the development, an alternative formulation for σ was also found, which seems more appropriate to anisotropic meshes, typical for *Reynolds-averaged Navier-Stokes* computations, which is the subject of the doctoral research of Marcus Drosson (40; 41).

Introduction

Consider the simple Poisson problem

$$\begin{aligned} \nabla\cdot(\mu\nabla\tilde{u}) &= f \ , \ \forall x \in \Omega \\ \tilde{u} &= u^* \ , \ \forall x \in \Gamma \end{aligned} \tag{3.1}$$

For this problem, usually the following generic expression for the minimal penalty coefficient σ_f^* is given (see Hartman and Houston (51), Epshteyn and Rivière (42), ...):

$$\sigma_f^* = Cp^2 \cdot \frac{\mu^*}{h^*} \tag{3.2}$$

There are however very few precise definitions for the length scale h^*, and the functional dependence on the order of interpolation p is only an asymptotic value. These incertainties are therefore compensated by the fudge factor C.

In case μ is variable, an appropriate diffusivity scale μ^* needs to be chosen as well. This scale depends on the functional dependence of μ with respect to the solution (and potentially its gradients) and is hence specific for the equations of state. In the remainder, μ is considered constant. Although one could remove μ from the equations 3.1, it is kept to include its impact on σ, such that the expression remains valid for mixed convection-diffusion problems.

Shahbazi (93) proposes sharp values for the minimal penalty coefficient σ_f^* in the case of triangular and tetrahedral meshes:

$$\begin{aligned} \sigma_f^* &= \mu\frac{(p+1)(p+d)}{d} \max_{e\ni f}(c_e) \\ c_e &= \frac{1}{\mathcal{V}(e)}\left(\frac{1}{2}\sum_{f\in e\setminus\Gamma}\mathcal{A}(f) + \sum_{f\in e\cap\Gamma}\mathcal{A}(f)\right) \end{aligned} \tag{3.3}$$

In this expression, $\mathcal{V}(e)$ is the volume (3D) or surface (2D) of the element e, while $\mathcal{A}(f)$ is the length (2D) or the surface (3D) of the face f. Finally d is the dimension of the mesh.

In the following sections this analysis will be extended to hybrid meshes, and an alternative length scale for anisotropic meshes will be proposed. The first part concerns the development of sharp expressions for a trace inverse inequality for all element types that will be used in the meshes; this work has been submitted to SINUM (Hillewaert *et al.* (60)). The second part generalises Shahbazi's coercivity analysis (93) to hybrid meshes and proposes an alternative, anisotropic length scale for high aspect ratio meshes.

The trace inverse inequality

The following inequality is used during the coercivity analysis of the interior penalty method:

$$\oint_{\partial e} v^2 dS \leq \frac{C(p)}{h} \cdot \int_e v^2 dV \; \forall v \in \mathcal{P}_p(e) \tag{3.4}$$

with $\mathcal{P}_p(e)$ a polynomial function space of order p defined on the element e. In Ciarlet (34) an asymptotic estimate $C(p) \sim p^2$ is given which clearly links to Eq. 3.2.

Warburton and Hesthaven (109) provide sharp values for both the length scale h and $C(p)$:

$$\oint_{\partial e} u^2 dS \leq \frac{(p+1)(p+d)}{d} \cdot \frac{\mathcal{A}(\partial e)}{\mathcal{V}(e)} \int_e u^2 dV \tag{3.5}$$

Their analysis was however restricted to the Pascal polynomial function space of order p and dimension d on simplicial elements (see appendix B for the definition of function spaces and element types). These bounds are sharp since for some of the polynomials in this set equation 3.5 reverts to an equality. As shown by Warburton, this inequality does not only hold for the integration on the full boundary, but also for each face individually. This inequality was used subsequently by Shahbazi to find the optimal expression 3.3 for the penalty parameter.

In the case of tensor product elements (lines, quadrilaterals and hexahedra) of dimension d Burman and Ern (26) proved the following trace inequality:

$$\int_f u^2 dS \leq \frac{p(p+1)}{2} \left(1 + \frac{1}{p}\right)^d \frac{\mathcal{A}(f)}{\mathcal{V}(e)} \int_e u^2 dV \; , \; \forall u \in \mathcal{P}_p(e) \tag{3.6}$$

however without claiming the sharpness of the bound.

In the general case the inequality depends the type of face $\mathfrak{f}$ and element $\mathfrak{e}$

$$\int_f u^2 dS \leq C_{\mathfrak{e},\mathfrak{f}}(p) \frac{\mathcal{A}(f)}{\mathcal{V}(e)} \int_e u^2 dV \; , \; \forall u \in \mathcal{P}_p(e) \tag{3.7}$$

The constants $C_{\mathfrak{e},\mathfrak{f}}(p)$ are listed in table 3.1 for all element types in conformal hybrid meshes. This set of constants was computed for both the Pascal polynomial space, as well as the classical polynomial spaces which are needed to

construct the standard Lagrange interpolators for the element. Both functional spaces are proven to yield exactly the same value. Since the mathematical development is rather tedious, only the final results are shown here, whilst the derivation is detailed in appendix C It is obvious that the values obtained for

$\mathfrak{e}$ / $\mathfrak{f}$	edge	triangle	quadrilateral
triangle*	$(p+1)(p+2)/2$	-	-
tetrahedron*	-	$(p+1)(p+3)/3$	-
quadrilateral	$(p+1)^2$	-	-
hexahedron	-	-	$(p+1)^2$
wedge	-	$(p+1)^2$	$(p+1)(p+2)/2$
pyramid	-	$1.05(p+1)(2p+3)/3^\dagger$	$(p+1)(p+3)/3$

Table 3.1: Compilation of sharp constants $C_{\mathfrak{e},\mathfrak{f}}(p)$ in the trace inverse inequality of Eq. (3.4). Values obtained by (109) marked by *. All of the values are determined analytically, except for those marked with †; on these values a security margin of 5% was taken.

quadrilaterals and hexahedra are much smaller than those obtained by Burman and Ern 3.6, and are moreover independent of dimension.

These bounds only apply to elements with a constant mapping. This precludes obviously curved elements, but also straight-sided non-simplicial elements. The derivation of bounds for curved elements is obviously much more complex, and will depend on the precise mapping. One can however assume that the values will be very similar, provided the elements are not too much distorted.

Optimal penalty coefficients for hybrid meshes

The interior penalty is now applied to the simple Poisson problem of Eq. 3.1 with constant diffusivity μ. This results in the following variational formulation for the approximate solution $u \in \mathcal{V}$

$$\begin{aligned}\sum_e \int_e \nabla v \cdot (\mu \nabla u)\, dV + \sum_f \sigma_f \int_f [[v]] \cdot [[u]] dS \\ - \sum_f \int_f [[v]] \cdot \langle \mu \nabla u \rangle + \theta [[u]] \cdot \langle \mu \nabla v \rangle dS = 0\ ,\ \forall v \in \mathcal{V}\end{aligned} \tag{3.8}$$

This expression is first cast as a generic variational formulation

$$a(u,v) = f(v)\ ,\ \forall v \in \mathcal{V} \tag{3.9}$$

with the following bilinear

$$\begin{aligned} a(u,v) = &\sum_e \int_e \nabla v \cdot (\mu \nabla u)\, dV \\ &+ \sum_{f \notin \Gamma} \sigma_f \int_f [[v]] \cdot [[u]] dS - \sum_{f \notin \Gamma} \int_f [[v]] \cdot \langle \mu \nabla u \rangle + \theta [[u]] \cdot \langle \mu \nabla v \rangle dS \\ &+ \sum_{f \in \Gamma} \sigma_f \int_f vu\, dS - \sum_{f \in \Gamma} \int_f v\vec{n} \cdot (\mu \nabla u) + \theta u \vec{n} \cdot (\mu \nabla v)\, dS \end{aligned} \tag{3.10}$$

and linear form

$$f(v) = \sum_{f \in \Gamma} \sigma_f \int_f v u^* dS + \sum_{f \in \Gamma} \int_f \theta u^* \vec{n} \cdot (\mu \nabla v)\, dS \tag{3.11}$$

Since the discontinuous Galerkin trial space $\mathcal{V}$ is a Hilbert space, the Lax-Milgram (theorem 2 in appendix A) states that a unique solution of 3.9 can be found if both forms are bound - which is a trivial condition - and if the bilinear form $a(.,.)$ is coercive:

$$\exists C > 0 : a(v,v) > C||v||^2\ ,\ \forall v \in \mathcal{V} \tag{3.12}$$

As $\mathcal{V}$ has finite dimensions, all of the norms defined on this space are equivalent. Therefore, one can also check for coercivity using the much more convenient DGM energy norm

$$||v||^2_{DG} = \sum_e \int_e |\nabla v|^2 dV + \sum_{f \notin \Gamma} \int_f |[[v]]|^2 dS + \sum_{f \in \Gamma} \int_f v^2 dS \tag{3.13}$$

instead of the standard broken Sobolev norm, based on the inner product defined in Eq. 2.7:

$$||u|| = \sum_e \sum_{|\alpha| \leq 1} \int_e \left(D^\alpha u\right)^2 dV \tag{3.14}$$

Further elaborating the bilinear form, one finds

$$\begin{aligned} a(v,v) = &\sum_e \int_e \mu |\nabla v|^2\, dV + \sum_{f \notin \Gamma} \sigma_f \int_f |[[v]]|^2\, dS + \sum_{f \in \Gamma} \sigma_f \int_f v^2\, dS \\ &- (1+\theta) \sum_{f \notin \Gamma} \int_f \langle \mu \nabla v \rangle \cdot [[v]]\, dS - (1+\theta) \sum_{f \in \Gamma} \int_f v \vec{n} \cdot \mu \nabla v dS \end{aligned} \tag{3.15}$$

The terms on the first line are obviously contributing to the coercivity, whilst the terms on the second line need to be dominated by a calibrated penalty term. In case of the *non-symmetric interior penalty method (NIPDG; $\theta = -1$)* the contributions on the second line do not impact the coercivity, and $\sigma_f > 0$

suffices for coercivity. Since the *incomplete IP (IIPDG) $\theta = 0$)* is in practice never used, we continue with the only other relevant case, namely the *symmetric variant SIPDG ($\theta = 1$)*. Young's inequality states that for any a, b and arbitrary $\epsilon > 0$

$$a^2/\epsilon + b^2\epsilon \geq 2ab \tag{3.16}$$

The application of this inequality to the terms on the second line results in

$$\begin{aligned} a\left(v,v\right) \geq & \sum_e \int_e \mu |\nabla v|^2 \, dV + \sum_{f\notin\Gamma} (\sigma_f - \epsilon_f) \int_f \left[\left[v\right]\right]^2 dS + \sum_{f\in\Gamma} (\sigma_f - \epsilon_f) \int_f v^2 \, dS \\ & - \sum_{f\notin\Gamma} \frac{1}{\epsilon_f} \int_f \mu |\langle \nabla v \rangle|^2 \, dS - \sum_{f\in\Gamma} \frac{1}{\epsilon_f} \int_f \mu |\nabla v|^2 dS \end{aligned} \tag{3.17}$$

A first condition for coercivity is then clearly that whatever ϵ_f is chosen, one needs

$$\sigma_f > \epsilon_f \ \ \forall f \tag{3.18}$$

Further applying the inequality $2a^2 + 2b^2 > (a+b)^2$ to the face integrals one finds

$$\begin{aligned} a\left(v,v\right) \geq & \sum_e \int_e \mu |\nabla v|^2 \, dV + \sum_{f\notin\Gamma} (\sigma_f - \epsilon_f) \int_f \left[\left[v\right]\right]^2 dS + \sum_{f\in\Gamma} (\sigma_f - \epsilon_f) \int_f v^2 \, dS \\ & - \sum_{f\notin\Gamma} \frac{1}{2\epsilon_f} \int_f \mu \left(|\nabla v^+|^2 + |\nabla v^-|^2 \right) \, dS - \sum_{f\in\Gamma} \frac{1}{\epsilon_f} \int_f \mu |\nabla v|^2 dS \end{aligned} \tag{3.19}$$

Finally the trace inequality 3.7 applied to ∇v leads to:

$$\begin{aligned} a\left(v,v\right) \geq & \underbrace{\sum_e \left(\mu - \sum_{f\in e} \frac{c_{f,e}}{\epsilon_f} \right) \int_e (\nabla v)^2 dV}_{a_I(v,v)} + \underbrace{\sum_f \int_f (\sigma_f - \epsilon_f) [[v]]^2 dS}_{a_{II}(v,v)} \\ c_{f,e} = & C_{\mathfrak{e},\mathfrak{f}}(p) \cdot \frac{\mathcal{A}(f)}{\mathcal{V}(e)} \ , \ \forall f \in \Gamma \\ = & C_{\mathfrak{e},\mathfrak{f}}(p) \cdot \frac{\mathcal{A}(f)}{2\mathcal{V}(e)} \ , \ \forall f \notin \Gamma \end{aligned} \tag{3.20}$$

Therefore $a\left(.,.\right)$ is coercive if

$$\begin{aligned} & \sum_{f\in e} \frac{c_{f,e}}{\epsilon_f} < \mu \\ & \sigma_f > \epsilon_f \end{aligned} \tag{3.21}$$

A straightforward generalisation of the proposition of Shahbazi to hybrid meshes is then

$$\begin{aligned} &\sigma_f > \epsilon_f > \mu \max_{e \ni f}(c_e) \\ &c_e = \frac{1}{\mathcal{V}(e)} \left(\frac{1}{2} \sum_{f \in e \setminus \Gamma} C_{\mathfrak{e},\mathfrak{f}}(p) \mathcal{A}(f) + \sum_{f \in e \cap \Gamma} C_{\mathfrak{e},\mathfrak{f}}(p) \mathcal{A}(f) \right) \end{aligned} \tag{3.22}$$

Alternatively one also obtains coercivity if

$$\sigma_f > \epsilon_f > \max_{e \ni f}(n_e c_{f,e}) \tag{3.23}$$

with n_e the number of faces in the element e.

The difference of both propositions can be seen by considering faces and elements away from the boundary on a mesh composed of a single element. Shahbazi's proposition then reduces to

$$\begin{aligned} \sigma_f &> C(p) \sum_{e \ni f} \frac{1}{h_e} \\ h_e &= \frac{\mathcal{V}(e)}{\mathcal{A}(\partial e)} \end{aligned} \tag{3.24}$$

and hence a length scale defined as the inscribed radius of the element. The alternative reduces to

$$\begin{aligned} \sigma_f &> C(p) \sum_{e \ni f} \frac{n_e}{h_{f,e}} \\ h_{f,e} &= \frac{\mathcal{V}(e)}{\mathcal{A}(f)} \end{aligned} \tag{3.25}$$

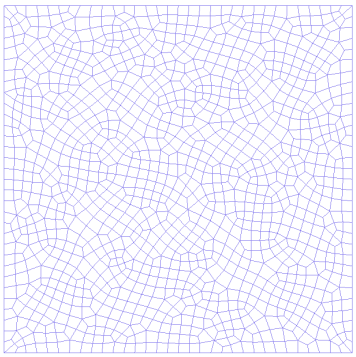
which defines a length scale attributed to the face itself, which is approximately proportional to the distance to the opposing face/node in the element. The latter seems more appropriate for high aspect ratio meshes.

Verification of stability and convergence

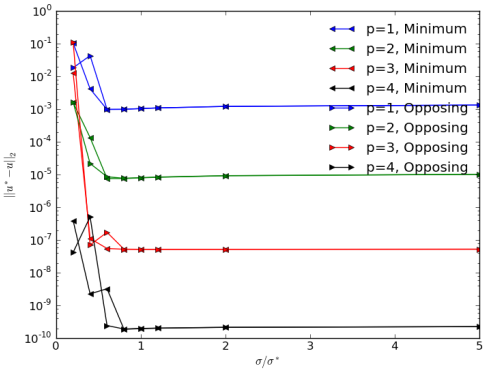
Both propositions for the penalty parameter have been tested by solving a Poisson problem on different mesh types, resolutions and aspect ratios. Thereto the value of the penalty parameter σ is varied with respect to the optimal value σ^*. The variation of the L_2 norm of the error allows to detect the onset of numerical instability.

In order to have full flexibility concerning the reference solution $\tilde{u}$, the *method of manufactured solutions* is used. Starting from the reference analytical solution

$$\tilde{u} = \prod_{i=1}^{d} e^{x_i} \tag{3.26}$$

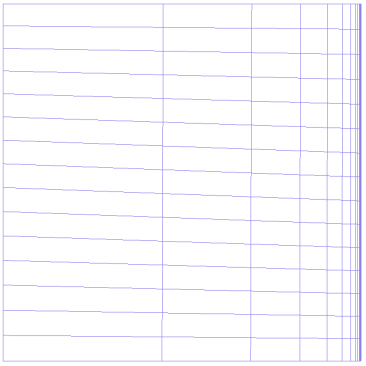


(a) mesh

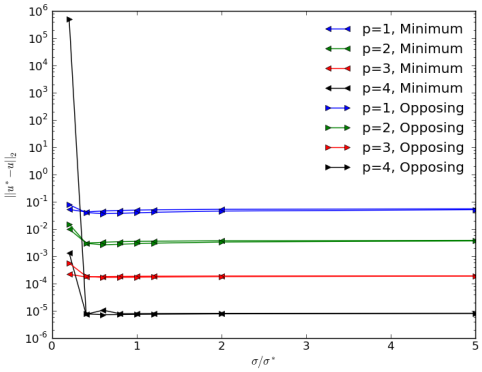


(b) variation of L_2 error with σ

Figure 3.1: Stability verification of the penalty parameter on mixed meshes. The error evolution as a function of σ is shown for both definitions of the length scale. The loss of stability is clearly noticeable for values $\sigma < \sigma^*$.



(a) mesh



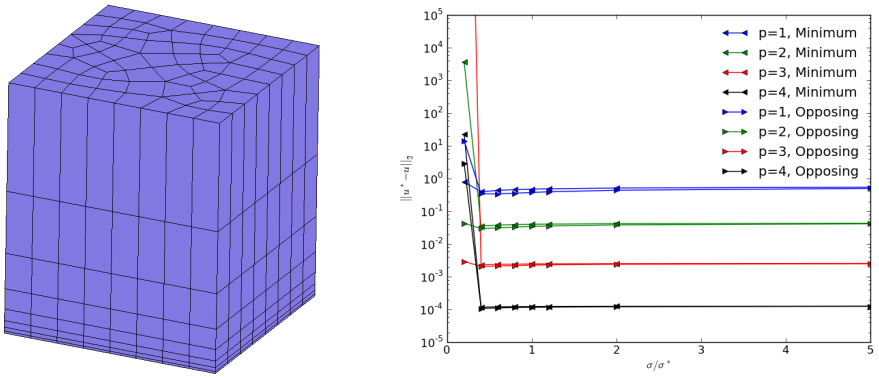
(b) variation of L_2 error with σ

Figure 3.2: Stability verification of the penalty parameter on stretched quadrilateral mesh.The error evolution as a function of σ is shown for both definitions of the length scale. The loss of stability is clearly noticeable for values $\sigma < \sigma^*$.

the d-dimensional Poisson problem is defined as

$$\begin{aligned} \Delta u &= \Delta \tilde{u} \ , \ \forall x \in \Omega \\ u &= \tilde{u} \ , \ \forall x \in \partial\Omega \end{aligned} \tag{3.27}$$

The exponential form is chosen such that the Taylor expansion of the solution has an infinite number of spectrally decreasing contributions. A representative sample of the results is shown in figures 3.1, 3.2, and 3.3.



(a) mesh

(b) variation of L_2 error with σ

Figure 3.3: Stability verification of the penalty parameter on mixed extruded meshes.The error evolution as a function of σ is shown for both definitions of the length scale. The loss of stability is clearly noticeable for values $\sigma < \sigma^*$.

Although one would naively expect non-convergence to occur from $\sigma < \sigma^*$, in practice it only appears around $\sigma \approx 0.5\sigma^*$. About the same "delay" is found by (93) in the case of triangles. It can be explained by the subsequent inequalities that have been used in the derivation of σ^*. Since this delay is systematically the same for all cases, mesh resolutions and aspect ratios considered, it can therefore be concluded that both estimates for σ^* are sharp. There is no significant difference however between both formulations for this simple problem. The anisotropic definition increases the stability for under-resolved RANS computations though, as shown by Drosson *et al.* (40; 41), indicating that this probably a result from the interaction with the anisotropic variation of the diffusivity.

3.2 Mixed formulation for the incompressible Navier-Stokes equations

This section is dedicated to the practical implementation of mixed methods, in particular DGM that combine trial spaces of different order. Since Argo is designed and optimised for conformal methods, in which all variables are interpolated with the same polynomials, the mixed discretisation would have to implemented separately.

However, in this section a simple technique is described that allows the reuse of the original, conformal discretisation routines. The transformation to the mixed formulation is then obtained by a simple post-processing of the residual vector, combined with the filtering of the pressure variable. This idea stems from discussions with Harald van Brummelen (25), who proposed a

similar approach to provide divergence free interpolation spaces for incompressible flows.

The most obvious application is of course the discretisation of the incompressible Navier-Stokes equations, but the method was also used to study the impact of the order of interpolation of the Spalart-Allmaras variable by Drosson *et al.*(40). The discretisation of the incompressible Navier-Stokes equations was used by Pochet in combination with the level-set approach for multiphase flows (86).

An important application is the simulation of welding procedures, where the molten or strained metal is modeled as a non-Newtonian incompressible flow. Although the simulation of welding procedures is one of the important applications of the Morfeo group, there is an interest for integrating this capacity in the flow solver Argo. This would allow us to use the free surface capacity of the solver to track the interface with the air, or two different metals in the weld.

Governing equations and DGM variational formulation

The incompressible Navier-Stokes equations are given by:

$$\begin{aligned} &\nabla\cdot\mathbf{u} = 0 \\ &\frac{\partial \mathbf{u}}{\partial t} + \nabla\cdot(\mathbf{u}\mathbf{u}) + \nabla p = \nabla\cdot(\nu\nabla\mathbf{u}) \end{aligned} \tag{3.28}$$

As for continuous finite element methods, this saddle point problem cannot be discretised in a straightforward manner, as this would allow spurious pressure modes. Girault and Rivière (48; 91) have shown that a stable DG formulation can be found by solving for a solution $(p, \mathbf{u})$ which belongs to the mixed DGM trial space $\mathcal{V}$

$$(p, \mathbf{u}) \in \mathcal{V} = \mathcal{V}_p \times \mathcal{V}_\mathbf{u} \tag{3.29}$$

where the interpolation order of the DGM pressure space $\mathcal{V}_p$ is one order lower than the DGM velocity space $\mathcal{V}_\mathbf{u}$:

$$\begin{aligned} \mathcal{V}_p &= \oplus_e \mathcal{P}_{p-1}(e) \\ \mathcal{V}_\mathbf{u} &= \oplus_e \left(\mathcal{P}_p(e)\right)^d \end{aligned} \tag{3.30}$$

Here d denotes the dimension of the problem, while $\mathcal{P}_p(e)$ is the set of polynomials up to order p on the element e, while $\oplus_e$ designates a direct sum on all the elements. We will denote the corresponding test functions as $(q, \mathbf{v})$.

In the description of the variational formulation the boundary terms have been omitted for simplicity. The continuity equation is then discretised as:

$$\sum_e \int_e \nabla q \cdot \mathbf{u} dV - \sum_f \int_f [[q]] \cdot \langle \mathbf{u} \rangle dS = 0 \ , \ \forall q \in \mathcal{V}_p \tag{3.31}$$

The diffusive terms can be discretised with any of the interior penalty methods. The physical interpretation, in particular the (lack of) preferential directions, of the two first order terms leads to a central pressure flux, and an

upwind flux for the velocity respectively

$$\begin{aligned}\sum_e \int_e \mathbf{v} \cdot \frac{\partial \mathbf{u}}{\partial t} dV - \sum_e \int_e \nabla \mathbf{v} : (\mathbf{u}\mathbf{u} + pI)\, dV \\ + \sum_f \int_f [[\mathbf{v}]] : \left(\mathcal{H}_{\mathbf{u}}(\mathbf{u}^+, \mathbf{u}^-, \vec{n})\vec{n}\right) dS + \sum_f \int_f [[\mathbf{v}]]\langle pI \rangle dS \\ - \sum_e \int_e \nabla \mathbf{v} : \nu \nabla \mathbf{u} dV \\ + \sum_f \int_f \left([[\mathbf{v}]] : \langle \nu \nabla \mathbf{u} \rangle + [[\mathbf{u}]] : \langle \nu \nabla \mathbf{v} \rangle\right) dS \\ - \sum_f \sigma_f \int_f [[\mathbf{v}]] : [[\mathbf{u}]] dS = 0\ ,\ \forall \mathbf{v} \in \mathcal{V}_{\mathbf{u}}\end{aligned} \tag{3.32}$$

Note that for convenience, an adapted jump is used for the velocity space, which is formed by the use of the external, rather than the internal, product with the normal.

$$[[\mathbf{u}]] = \mathbf{u}^+ \vec{n}^+ + \mathbf{u}^- \vec{n}^- \tag{3.33}$$

and that the upwind flux is a vector, which should satisfy the following consistency constraint

$$\mathcal{H}_{\mathbf{u}}(\mathbf{u}, \mathbf{u}, \vec{n}) = (\mathbf{u}\mathbf{u}) \cdot \vec{n} \tag{3.34}$$

Note that this formulation is only stable for limited Reynolds numbers, and is hence only suited for very viscous flows, as encountered in production processes. Furthermore, as a consequence of the non-conformity of the trial space, one expects that the norm of the error varies as h^p for the pressure, and as h^{p+1} for the velocity.

Adaptation of a conforming discretisation

The straightforward integration of this mixed formulation would necessitate specific assembly routines for the residual and the tangent matrix. This can be circumvented by the following approach which reuses all of the operations, as if the interpolation was conformal, and then converts to the result to the mixed counterpart by filtering the resulting equations.

The basic idea is the following. Throughout the computation of the residual, a fully conforming space $\mathcal{V}^*$ is used for the interpolation of $(p^*, \mathbf{u})$ and the test functions $(q^*, \mathbf{v})$

$$\mathcal{V}_p^* = \oplus \mathcal{P}_p(e) \tag{3.35}$$

We choose a set of basis functions ϕ_i^* for $\mathcal{V}_p^*$ and ϕ_i for $\mathcal{V}_p$ on the elements. Since $\mathcal{V}_p \subset \mathcal{V}_p^*$, we can expand any $\phi_i \in \mathcal{V}_p$ as a combination of $\phi_j^* \in \mathcal{V}_p^*$

$$\phi_i = \sum_j \mathbf{P}_{ij} \phi_j^* \tag{3.36}$$

where $\mathbf{P}$ is the restriction operator from $\mathcal{P}_p$ to $\mathcal{P}_{p-1}$. Due to the discontinuity of the interpolating functions, this operator is defined elementwise. It is the same as the restriction operators of the p-multigrid iterative method (see 4.2, and can be found by the Galerkin projection.

Replacing the test functions $\mathbf{v}$ by the shape functions ϕ_i in the discretised continuity equation 3.31. we find that we can recuperate the mixed formulation of the continuity equation by filtering the conformal formulation as:

$$\begin{aligned} -\sum_e \int_e \nabla\phi_i \cdot \mathbf{u} dV + \sum_f \int_f [[\phi_i]] \cdot \langle \mathbf{u} \rangle dS \\ = \sum_j \mathbf{P}_{ij} \left(-\sum_e \int_e \nabla\phi_j^* \cdot \mathbf{u} dV + \sum_f \int_f \left[[\phi_j^*] \right] \cdot \langle \mathbf{u} \rangle dS \right) \end{aligned} \tag{3.37}$$

The pressure p^* is still not necessarily in the space $\mathcal{V}_p$, and that the momentum equation residual has been computed with this a priori higher order pressure. One then enforces a posteriori that $p^* \in \mathcal{V}_p$ by demanding that it is orthogonal to the kernel of the projection operator $\mathbf{P}$. Since $\mathcal{V}_p = \text{range}(\mathbf{P})$, this restores the correspondence between the number of residual equations and the degrees of freedom. Both the restriction operator $\mathbf{P}$ and its kernel can be expressed elementwise; the latter is found by the *singular value decomposition (SVD)* of the elementwise restriction operator.

Kovasznay flow

Kovasznay (72) described an exact solution to the two-dimensional incompressible Navier-Stokes equations, closely resembling the steady flow behind a row of cylinders at low Reynolds number. This flow solution is given by the stream function

$$\begin{aligned} \psi(x,y) &= U\left(y - \frac{M}{2\pi} \exp\left(\frac{\alpha x}{M}\right) \sin\left(\frac{2\pi y}{M}\right) \right) \\ \alpha &= \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2} \end{aligned} \tag{3.38}$$

M is the distance between two successive cylinders and U the free stream velocity; hence the Reynolds number is given by $Re = \frac{UM}{\nu}$. In primitive variables we find

$$\begin{aligned} p^* &= p_0 + \frac{\rho U^2}{2}\left(1 - \exp\left(\frac{2\alpha x}{M}\right)\right) \\ u_x^* &= \frac{\partial\psi}{\partial y} = U\left(1 - \exp\left(\frac{\alpha x}{M}\right)\cos\left(\frac{2\pi y}{M}\right)\right) \\ u_y^* &= -\frac{\partial\psi}{\partial x} = U\frac{\alpha}{2\pi}\exp\left(\frac{\alpha x}{M}\right)\sin\left(\frac{2\pi y}{M}\right) \end{aligned} \tag{3.39}$$

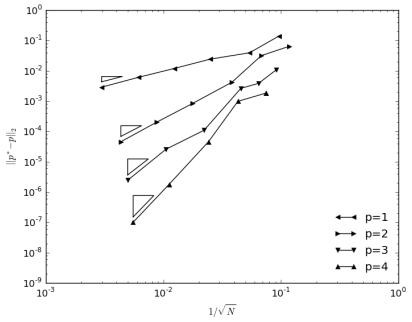
This flow is computed for $Re = 40$ on the domain $[0, M/2] \times [0, M/2]$. At the left boundary the theoretical velocity, whilst at the right boundary both theoretical pressure and tangential velocity are imposed; symmetry conditions are applied at top and bottom boundaries.

The mesh resolutions are chosen for each order p such that an equivalent resolution, in terms of degrees of freedom per unit length, is maintained. This resolution varies from $h^* = M/80$ to $h^* = 2M/5$ by repeatedly doubling the size. The interpolation orders range from $p = 1 \dots 4$. The imposed mesh size is then determined as $h = ph^*$ as $p + 1$ interpolation points are needed to support a p-th order Lagrange polynomial, and hence every cell can be considered to be subdivided in p intervals. We follow both the RMS (L_2 norm) and maximum (L_∞ norm) error for the three components of the solution. The L_2 norm of the error is computed by quadrature on the element; the L_∞ norm however can not be computed exactly, and is approximated by the maximum error observed in the quadrature points.

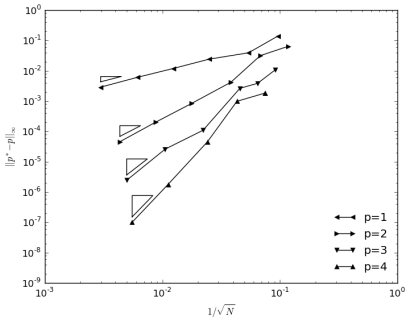
Two remarks concerning the resolutions are in order. First of all, more degrees of freedom are "duplicated" on the interfaces between elements at lower orders of interpolation, and hence the number of degrees of freedom for a similar resolution is higher at lower order. Secondly it was not possible to strictly observe the imposed mesh size, as for some cases it is to close to the size of the domain. For the most extreme case, *i.e.* $p = 4$ the same mesh was generated when imposing the mesh size corresponding to the two coarsest equivalent resolutions.

The error convergence is show in Fig. 3.4. The pressure error curves are accompanied by a triangle indicating h^p convergence for reference, whilst the triangles next to the velocity error curves show h^{p+1} convergence. Overall the error exhibits the expected convergence behaviour for both norms. One moreover sees that the error of the $p = 1$ computation on the finest mesh is always larger than the $p = 4$ computation on the coarsest mesh. This error reflects itself clearly in the computed fields as illustrated later on. We see that on any mesh, higher order implies lower error. It is worthwhile to note that the linear computation hasn't reached asymptotic convergence even for the finest resolutions.

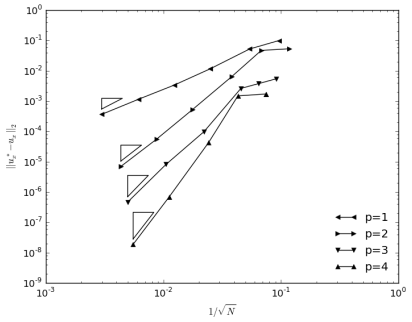
Figures 3.5, 3.6 and 3.7 illustrate the mesh convergence for p=4 for three successive refinements. The mesh corresponding to the coarsest resolution can not be generated as h is too large with respect to the domain, and thus the sequence starts from the second coarsest resolution. The quality of the solution can be visually inspected easily: the pressure should only be dependent on the axial coordinate, and the streamtrace on the upper surface should go straight forward up to the end of the domain. We see that for $p = 4$ the velocity distribution is already obtained on the coarsest mesh, and only minor problems are found with the pressure. The mesh corresponding to the 4th coarsest resolution already is sufficient to accurately capture both velocity and pressure, at least visually. Figures 3.9 and 3.8 show the same plots for $p = 1$ on the two finest resolutions $h = M/40$ and $h = M/80$. Unsurprisingly the pressure is not very well represented, as it is constant per element. Also the velocity is not converged, as can be seen from the stream traces.



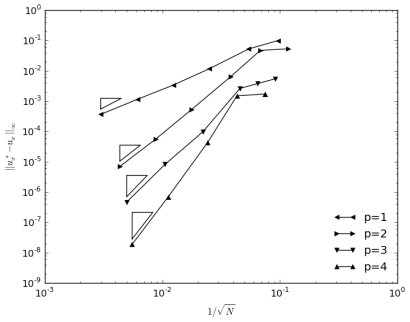
(a) L_2 error on p



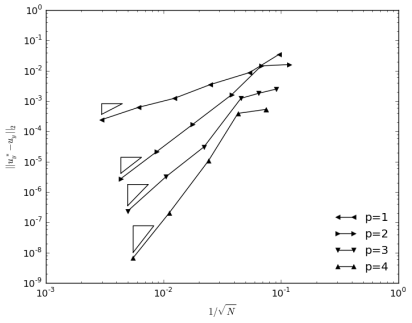
(b) L_∞ error on p



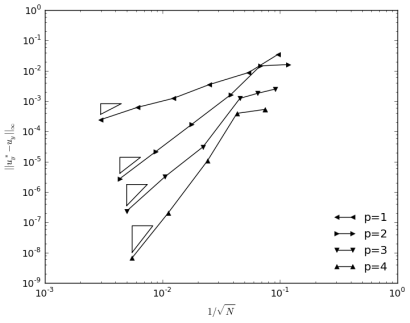
(c) L_2 error on u_x



(d) L_∞ error on u_x

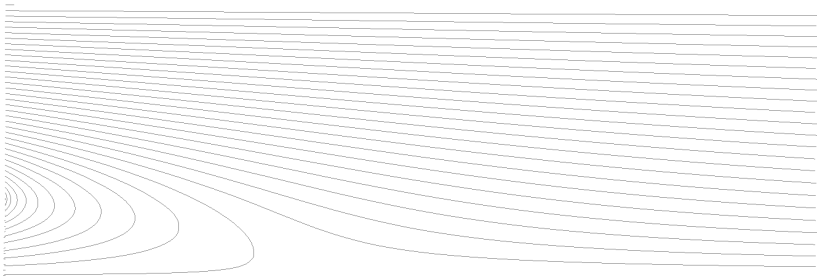


(e) L_2 error on u_y

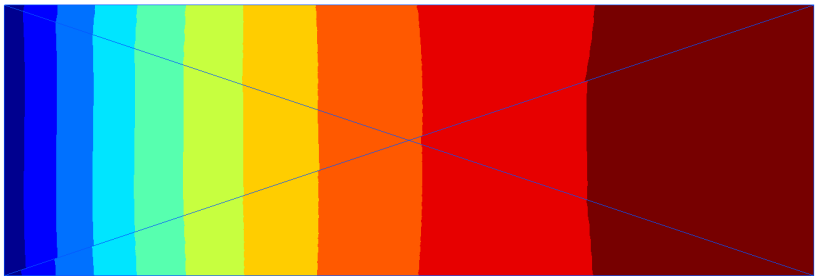


(f) L_∞ error on u_y

Figure 3.4: Kovasznay flow - mesh convergence of the pressure and velocity errors for different orders. The expected convergence rate - h^p for pressure and h^{p+1} for velocity - are indicated with triangles.

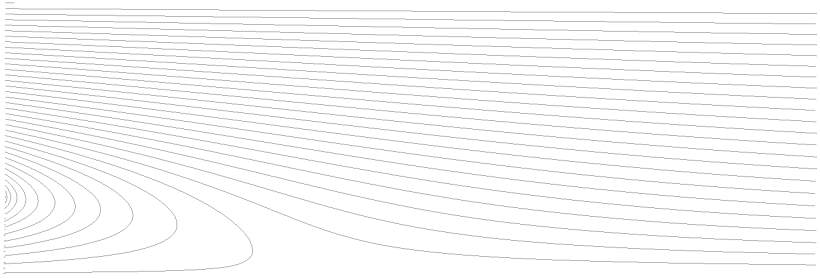


(a) Stream traces

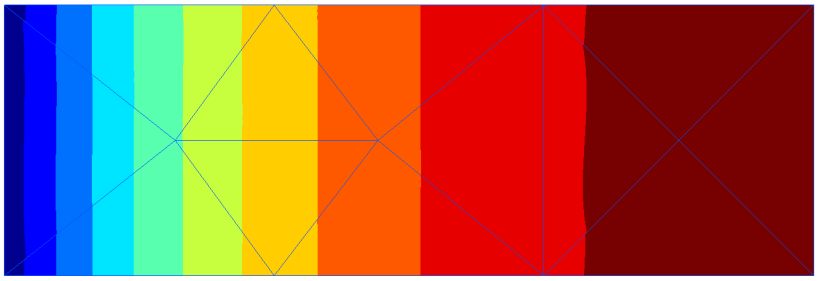


(b) Pressure isolines and mesh

Figure 3.5: Kovasznay flow - $p = 4$ computations on resolution $h = M/5$



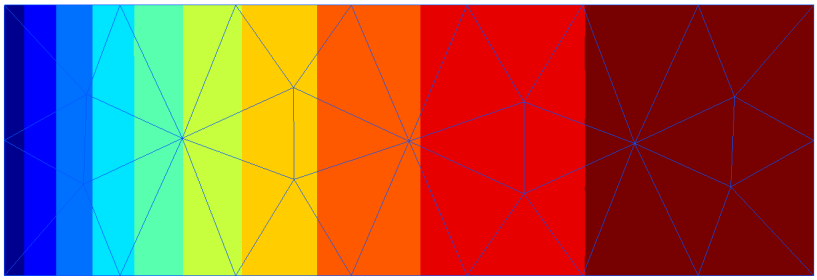
(a) Pressure isolines and mesh



(b) Pressure isolines and mesh

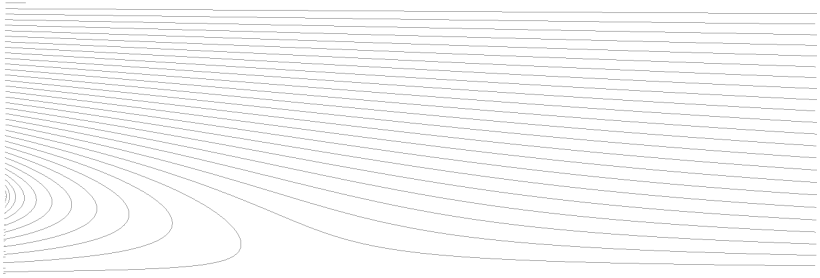
Figure 3.6: Kovasznay flow - $p = 4$ computations on resolution $h = M/10$

(a) Stream traces

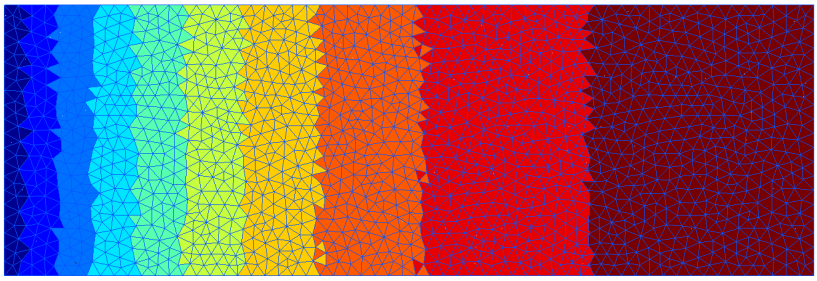


(b) Pressure isolines and mesh

Figure 3.7: Kovasznay flow - $p = 4$ computations on resolution $h = M/20$

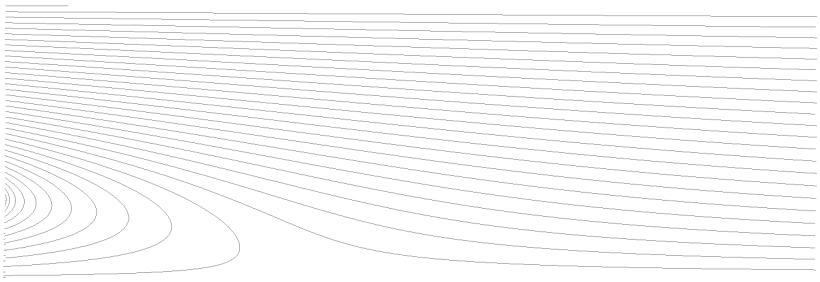


(a) Stream traces

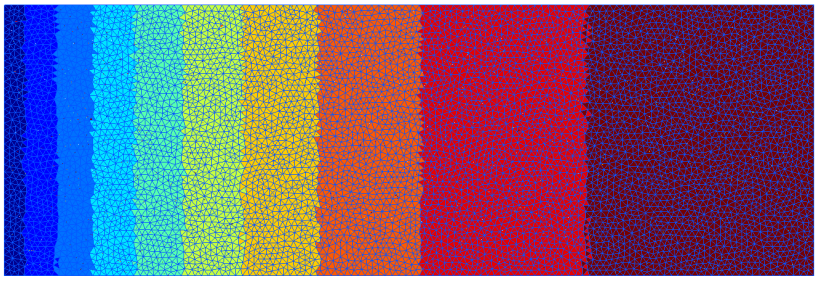


(b) Pressure isolines and mesh

Figure 3.8: Kovasznay flow - $p = 1$ computations on resolution $h = M/40$



(a) Stream traces



(b) Pressure isolines and mesh

Figure 3.9: Kovasznay flow - $p = 1$ computations on resolution $h = M/80$

3.3 Frequential formulation of the homentropic linearized Euler equations

In this section, we investigate the DGM formulation for a frequential formulation of the (cylindrical) *homentropic linearized Euler Equations (hLEE)*. Thereto the hLEE are cast in a conservative form. Subsequently the stability of the harmonic time derivative is shown; this analysis extends to frequential formulations of systems with first order time derivatives. Finally some validation examples are shown.

Governing equations

The linearized Euler Equations are solved for the pressure, density and velocity perturbations

$$\begin{aligned} p &= \bar{p} + p' \\ \rho &= \bar{\rho} + \rho' \\ \mathbf{u} &= \bar{\mathbf{u}} + \mathbf{u}' \end{aligned} \tag{3.40}$$

Based on the homentropy of flow, the following linearized isentropic relation between the density and the pressure perturbation is assumed:

$$p' = \left(\frac{\partial p}{\partial \rho}\right)_S \rho' = c^2 \rho' \tag{3.41}$$

where the speed of sound is noted as c. Further assuming incompressible base flow one finds the following set of equations:

$$\begin{aligned} &\frac{\partial p'}{\partial t} + \nabla\cdot\left(\bar{\mathbf{u}} p'\right) + \nabla\cdot\left(\bar{\rho} c^2 \mathbf{u}'\right) = 0 \\ &\frac{\partial \mathbf{u}'}{\partial t} + \nabla\left(\frac{p'}{\bar{\rho}}\right) + \nabla\cdot\left(\bar{\mathbf{u}}\mathbf{u}'\right) = -\frac{\bar{\mathbf{u}}\cdot\nabla\bar{\mathbf{u}}}{\rho c^2} p' - \mathbf{u}'\cdot\nabla\bar{\mathbf{u}} \end{aligned} \tag{3.42}$$

The DGM discretisation of the conservative form of the homentropic linearised Euler equations in Cartesian coordinates and time domain has been presented by Chevaugeon *et al.* (33). For axisymmetric problems one finds

$$\begin{aligned} &\frac{\partial p'}{\partial t} + \frac{\partial}{\partial r}\left(\bar{\rho} c^2 v_r' + \bar{v}_r p'\right) + \frac{\partial}{\partial a}\left(\bar{\rho} c^2 v_a' + \bar{v}_a p'\right) = -\frac{\bar{\rho} c^2 v_r' + \bar{v}_r p'}{r} \\ &\frac{\partial v_r'}{\partial t} + \frac{\partial}{\partial r}\left(\frac{p'}{\bar{\rho}} + \bar{v}_r v_r'\right) + \frac{\partial}{\partial a}\left(\bar{v}_a v_r'\right) = -\frac{p'}{\bar{\rho} c^2}\left(\bar{v}_r\frac{\partial \bar{v}_r}{\partial r} + \bar{v}_a\frac{\partial \bar{v}_r}{\partial a}\right) + v_r'\frac{\partial \bar{v}_a}{\partial a} - v_a'\frac{\partial \bar{v}_r}{\partial a} \\ &\frac{\partial v_a'}{\partial t} + \frac{\partial}{\partial r}\left(\bar{v}_r v_a'\right) + \frac{\partial}{\partial a}\left(\frac{p'}{\bar{\rho}} + \bar{v}_a v_a'\right) = -\frac{p'}{\bar{\rho} c^2}\left(\bar{v}_r\frac{\partial \bar{v}_a}{\partial r} + \bar{v}_a\frac{\partial \bar{v}_a}{\partial a}\right) + v_a'\frac{\partial \bar{v}_r}{\partial r} - v_r'\frac{\partial \bar{v}_a}{\partial r} \end{aligned} \tag{3.43}$$

As shown in the subsequent tests, the singularity in the source term of the continuity equation is not critical: it is well-behaved when approaching the axis, since

$$\lim_{r\to 0} v_r = 0\ ,\ \lim_{r\to 0} v_r' = 0 \tag{3.44}$$

As far as the numerical implementation is concerned, this term is only evaluated in the volume quadrature points, which are not located on the axis. Hence no particular precautions need to be taken.

Remark that the cylindrical formulation is similar to that of the 2D Cartesian counterpart (replace a by x, and r by y), except for a number of additional source terms. As these source terms are discretised as such, exactly the same convective formulation is found.

The DGM formulation stabilises the convective terms using an upwind flux. However we should indicate that the source terms in the momentum equations of 3.42 can (should?) a priori not be stabilised by the discretisation, since they correspond to linear hydrodynamic instabilities, such as the Kelvin-Helmholtz instability in shear layers. To avoid their occurence specific measures should be taken, *e.g.* by avoiding the excitation or filtering of terms that excite those the unstable modes. These instabilities should not be present in frequential computations, since usually the frequency is not compatible with the instability.

Stability of the harmonic time derivative

The time-harmonic formulation of the LEE at frequency f is written as

$$\begin{aligned} i\omega\widehat{p} + \nabla\cdot\left(\bar{\rho}c^2\widehat{\mathbf{u}} + \bar{\mathbf{u}}\widehat{p}\right) &= 0 \\ i\omega\widehat{\mathbf{u}} + \nabla\left(\frac{\widehat{p}}{\rho}\right) + \nabla\cdot\left(\bar{\mathbf{u}}\widehat{\mathbf{u}}\right) &= -\widehat{p}\frac{\bar{\mathbf{u}}\cdot\nabla\bar{\mathbf{u}}}{\rho c^2} - \widehat{\mathbf{u}}\cdot\nabla\bar{\mathbf{u}} \end{aligned} \tag{3.45}$$

with the angular frequency defined as $\omega = 2\pi f$, and $\widehat{p}$ resp. $\widehat{v}$ the Fourier transforms of the acoustic pressure and velocity perturbations. The Fourier transform of the time derivative results in the addition of a source term, providing the coupling between the real and imaginary components. It can easily be shown that this term does not impact on stability. Therefore first the frequency domain LEE is written down in generic notation as

$$i\omega\widehat{u} + \nabla\cdot f(\widehat{u}) = S(\widehat{u}) \tag{3.46}$$

The complex state vector is defined as $u = (u_R, u_I)$ so the governing equations read:

$$\begin{aligned} -\omega u_I + \nabla\cdot f(u_R) &= S(u_R) \\ \omega u_R + \nabla\cdot f(u_I) &= S(u_I) \end{aligned} \tag{3.47}$$

If $c(.,.)$ is the bilinear form corresponding to the DGM discretisation of the convective flux f and the source term S then the bilinear form of the discretisation of the coupled real and imaginary equations read

$$a\left(u,v\right) = \omega\sum_e\int_e\left(v_I u_R - v_R u_I\right)dV + a(u_R,v_R) + a(u_I,v_I) \tag{3.48}$$

Hence the stability of $a^*(.,.)$ depends only on the stability of $a(.,.)$ since

$$a^*(v,v) = a(v_R,v_R) + a(v_I,v_I) \tag{3.49}$$

This means that a stable formulation for the time-accurate equations, automatically provides a stable formulation for the corresponding harmonic problem.

Harmonic PML without mean flow

The freestream boundary condition imposes the absence of waves outside of the boundary. Since this is not compatible with the acoustic field approaching the boundary, reflections will be generated. In order to avoid these interfering with the physical field, we provide a damping zone around the boundary.

In case the mean flow velocity is zero, we can use a so-called *perfectly matched layer (PML)*. This technique was initially introduced for electromagnetic scattering problems by Bérenger (15), and subsequently developed by Hu (62) for acoustic problems without base flow. The extension of PML to generic base flows is not straightforward and has been discussed by Hu (63) for a number of representative cases. Within this thesis, only formulations for a medium at rest are considered.

The PML technique is defined for rectangular/parallellipedal domains, that are aligned with the coordinate axes. On the external boundaries of the domain, axis-aligned damping layers are introduced. Within these layers the equations are modified to provide sufficient directional damping of the waves and at the same time avoid spurious reflections on the transition.

Within this work the formulation proposed by Rahmouni (88), which allows the introduction of PML as a simple modification of the harmonic time-derivative source terms in 3.45. Since the convective part is the same exactly the same PML formulation can be used for both the two-dimensional Cartesian and the axisymmetric formulation. In two dimensions, the PML formulation reads:

$$\begin{aligned} i\omega\widehat{p} + \bar{\rho}c^2\left(D_x\frac{\partial\widehat{v}_x}{\partial x} + D_y\frac{\partial\widehat{v}_y}{\partial y}\right) &= 0 \\ i\omega\widehat{v_x} + \frac{D_x}{\rho}\frac{\partial\widehat{p}}{\partial x} &= 0 \\ i\omega\widehat{v_y} + \frac{D_y}{\rho}\frac{\partial\widehat{p}}{\partial y} &= 0 \end{aligned} \tag{3.50}$$

where

$$\begin{aligned} D_x &= \frac{i\omega}{i\omega + \sigma_x} \\ D_y &= \frac{i\omega}{i\omega + \sigma_y} \end{aligned} \tag{3.51}$$

σ_x and σ_y are the damping coefficients in the PML that are respectively orthogonal to the x and y (or r coordinates) and have the dimensions of a frequency. In case of 1D equations, it is easy to see that this set of equations corresponds to a damped convection.

One can show (see Hu (62)) that irrespective of the choice of σ_x and σ_y, no reflections are generated at the interfaces, provided σ_x only varies along x and σ_y only along y. Still following Rahmouni (88), this set of equations can

be reorganised as:

$$\begin{aligned} i\omega D_x^{-1} D_y^{-1} \widehat{p} + \bar{\rho} c^2 \left(\frac{\partial \widehat{v}_x}{\partial x} + \frac{\partial \widehat{v}_y}{\partial y} \right) = 0 \\ i\omega D_x^{-1} D_y \widehat{v_x} + \frac{1}{\rho} \frac{\partial \widehat{p}}{\partial x} = 0 \\ i\omega D_x D_y^{-1} \widehat{v_y} + \frac{1}{\rho} \frac{\partial \widehat{p}}{\partial y} = 0 \end{aligned} \tag{3.52}$$

This formulation is straightforward in a frequential formulation (in contrast to the temporal version), since one only needs to modify the Fourier transformed time derivatives. To provide an automatic and sufficiently high definition of σ_x and σ_y, the unbounded version by Bermudez et al. (17) is used. For instance in the PML of thickness δ located at the right side of the domain ($x = X$), σ_x is defined as:

$$\begin{aligned} \sigma_x &= 0, \ x < X - \delta \\ &= \frac{c}{X - x}, \ X - \delta < x < X \end{aligned} \tag{3.53}$$

The unboundedness of the damping factors σ_x and σ_y on the boundary of the domain is not problematic, since both are used in a source term which is integrated on the volume of the elements. Most of the element quadrature/cubature rules do not contain integration points outside or on the boundary of the element. If this is however the case, the singularity needs to be displaced far enough - typically over a distance comparable to the mesh size - out of the domain.

The source term in the pressure equation reduces to

$$\left(i\omega \frac{\omega^2 - \sigma_x \sigma_y}{\omega^2} + (\sigma_x + \sigma_y) \right) \widehat{p} \tag{3.54}$$

As shown previously the imaginary contribution is neutral with respect to stability. As both $\sigma_x, \sigma_y \geq 0$ the real contribution is always positive, and hence contributes to stability. The source terms in the momentum equations reduce to

$$\begin{aligned} \left(i\omega \frac{\omega^2 + \sigma_x \sigma_y}{\omega^2 + \sigma_y^2} + \frac{\omega^2}{\omega^2 + \sigma_y^2} (\sigma_x - \sigma_y) \right) \widehat{v}_x \\ \left(i\omega \frac{\omega^2 + \sigma_x \sigma_y}{\omega^2 + \sigma_x^2} + \frac{\omega^2}{\omega^2 + \sigma_x^2} (\sigma_y - \sigma_x) \right) \widehat{v}_y \end{aligned} \tag{3.55}$$

Here we see that stability is only diminished in case both σ_x and σ_y are different from zero, *i.e.* in the overlap of both PML.

Validation

Exact analytical solutions for acoustic problems are not easily found. Often only approximative expressions can be found, usually restricted to very specific regions. Therefore, most convergence assessments in the following sections will be rather qualitative than quantitative.

Circular loudspeaker

A circular loudspeaker with radius R, is uniformly oscillating along its central axis with angular frequency ω and maximal displacement 2δ. The amplitude of the acoustic pressure perturbation along the central axis is then given by the following analytic expression (Beyer (18))

$$2\rho\omega\delta \sin\left(\frac{k}{2}\left(\sqrt{x^2+R^2}-x\right)\right) \tag{3.56}$$

The acoustic field has been computed for two frequencies using the axisym-

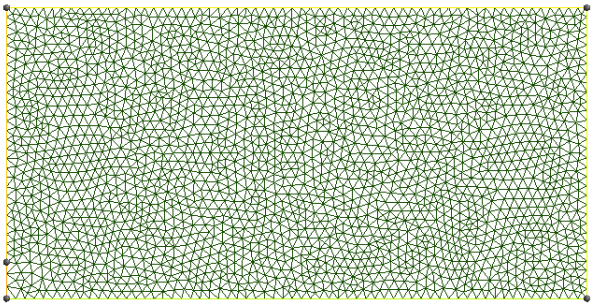


Figure 3.10: Circular loudspeaker - mesh and domain. The loudspeaker is located in the lower left corner, between the corner and point on the left vertical edge.

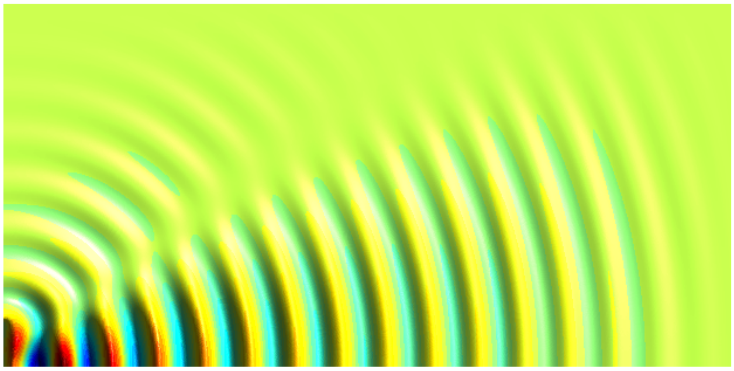
metric formulation. A single mesh in the axi-radial (*i.e.* (x, r)) plane is used and shown in Fig. 3.10. The size h of its elements ranges from $R/4$ to $R/3$. The loudspeaker is located at the lower left corner.

The *Helmholtz number* relates the typical size of the geometry, here obviously the radius of the speaker, to the wave length of the acoustic signal:

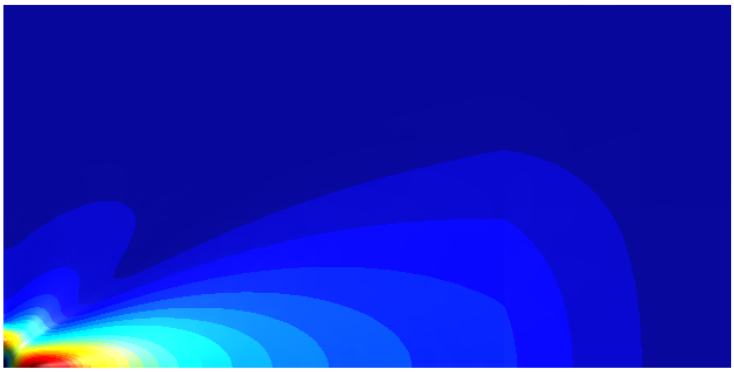
$$He = kR = \frac{2\pi R}{\lambda} = \frac{\omega R}{c} \tag{3.57}$$

The first computation corresponds to the Helmholtz number $He = 2\pi$. Hence the wave length of the acoustic wave corresponds to the radius, and the mesh resolution corresponds from a quarter up to a third of a wave length. The computed pressure field is shown in Fig. 3.11. One side-lobe is present next to the central one. The perfectly matching layer is 5 wave lengths wide, and its impact on the pressure field can be easily seen.

The second computation corresponds to the Helmholtz number $He = 4\pi$; hence the mesh resolution corresponds from a half up to two thirds of a wave length. The computed pressure field is shown in Fig. 3.12. This time three side lobes can be distinguished. The impact of the PML is illustrated in Fig. 3.13. If no PML are used, a very prominent diffraction pattern is formed by the interaction of the primary waves with spurious reflections on the freestream



(a) Real component

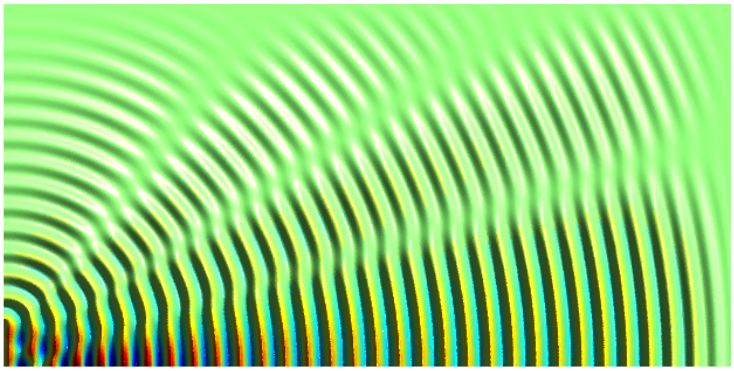


(b) Amplitude

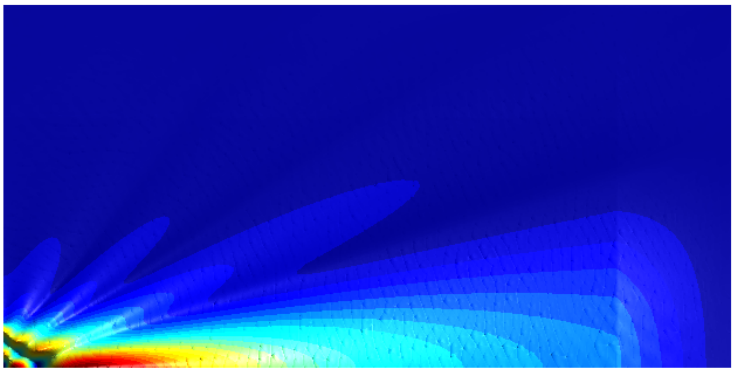
Figure 3.11: Circular loudspeaker actuated at $He = 2\pi$ - computed acoustic pressure field in the axi-radial plane. One side-lobe can be distinguished, and a single maximum on the axis.

boundary. On the other hand, when PML are used, the solution outside of the PML is not noticeably impacted.

Some small oscillations can still be noticed in the amplitude field for the computation with PML. These are however not due to reflections or under-resolution, but due to the fact that for the visualisation, the amplitude was



(a) Real component (with PML)

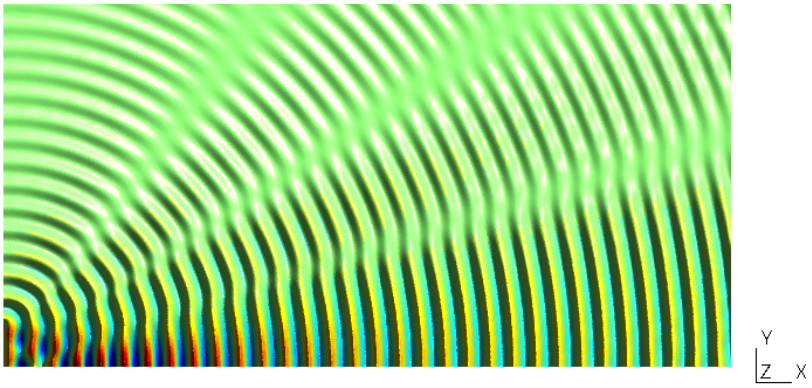


(b) Amplitude (with PML)

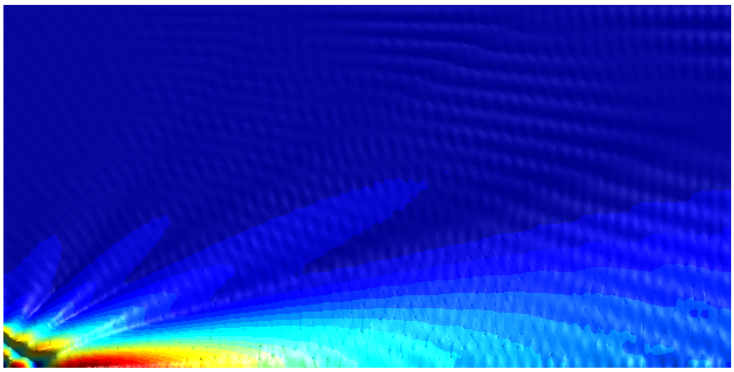
Figure 3.12: Circular loudspeaker actuated at $He = 4\pi$ - computed acoustic pressure field in the axi-radial plane. Three side lobes can be noticed, as well as two maxima along the axis.

interpolated with the same functions as the solution itself. Clearly this is not sufficient for a non-linear function of the solution, such as the amplitude.

In Fig. 3.14 the evolution of the amplitude along the axis is compared to the theoretical profile. We can again clearly see the impact of the PML on the profile.



(a) Real component (without PML)



(b) Amplitude (without PML)

Figure 3.13: Circular loudspeaker actuated at $He = 4\pi$ - computed acoustic pressure field in the axi-radial plane without PML. The effect of the homogeneous boundary condition is clearly noticeable in the form of a diffraction pattern.

Finally, the mesh convergence rate on this testcases for $He = 2\pi$ is illustrated in figure 3.15. It shows the L_2 norm of the real and imaginary part of the radial velocity on the axis as a function of mesh resolution, based on the total number of degrees of freedom N. As $\lim_{r\to 0} w_r = 0$, this norm is a direct measure of the error. To provide "compatible" mesh resolutions for different

orders, the mesh size is computed as a base size h multiplied by the order p, thus resulting in the same number of (continuous) interpolation intervals per unit length. Since DGM duplicates degrees of freedom on the element interfaces, higher N are obtained for similar resolution as interpolation order decreases. One can see that the theoretical convergence rate h^{p+1} is obtained, in spite of the singularity of the equations near the axes. This reflects the fact that in spite of the singularity of the equations, the solution - or at least the radial velocity - is regular.

Acoustic waves propagating up-or downstream of Poiseuille flow

An approximate analytical formula describing the waves of the form

$$\begin{aligned} p' &= p(r)e^{i\omega t+\gamma a+n\theta} \\ \mathbf{u}' &= \bar{\mathbf{u}}(r)e^{i\omega t+\gamma a+n\theta} \end{aligned} \tag{3.58}$$

propagating on top of a Poiseuille flow in a circular tube of radius R has been developed by Boucheron *et al.* (21). The following non-dimensional parameters characterise the solution

- the reduced temporal frequency $\Omega = \frac{\omega R}{c0}$
- the reduced axial frequency $\Gamma = \frac{\gamma c_0}{\omega}$
- the centerline Mach number M_0 (with $v_a = M_0 c_0(R^2 - r^2)$)

Here the axial frequency Γ is found from a dispersion relation. For quiescent flow, clearly $\Gamma = 1$. For the cases of uniform flow at Mach number M, one finds

- downstream running wave: $\Gamma = 1/(1 + M)$
- upstream running wave: $\Gamma = 1/(1 - M)$

One sees that for the Poiseuille flow the local spatial frequency changes with radius. However, coherent wave group solutions can be found.

The validation starts by imposing the analytical radial profiles found by Boucheron *et al.* at the boundaries:

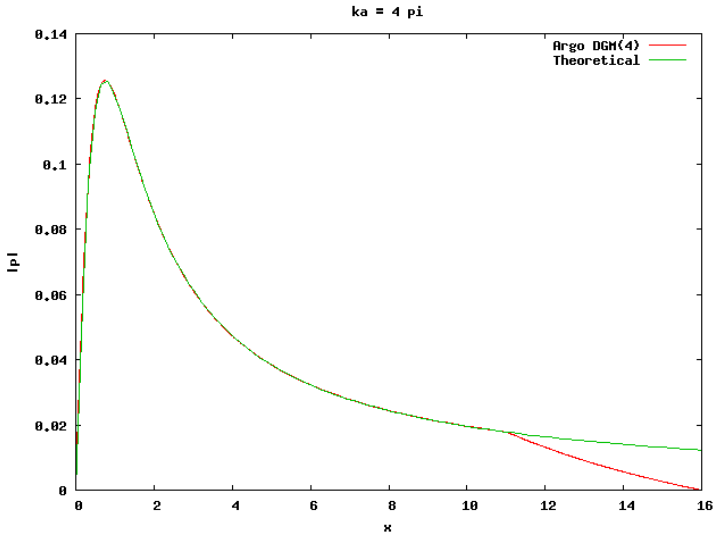
- for the downstream running waves, the radial profiles of pressure and radial velocity are imposed at the upstream boundary, whilst a PML is used at the downstream boundary;
- for the upstream running waves, the radial profile of pressure radial velocity is imposed at the downstream boundary, whilst a PML is used at the upstream boundary;

after which the computed group axial frequency is compared to the theoretical values (21). Nota bene: the formulation of the PML has not yet been modified to take the non-zero velocity into account, and therefore damping is not guaranteed, which prevented the computation of some conditions. To ensure

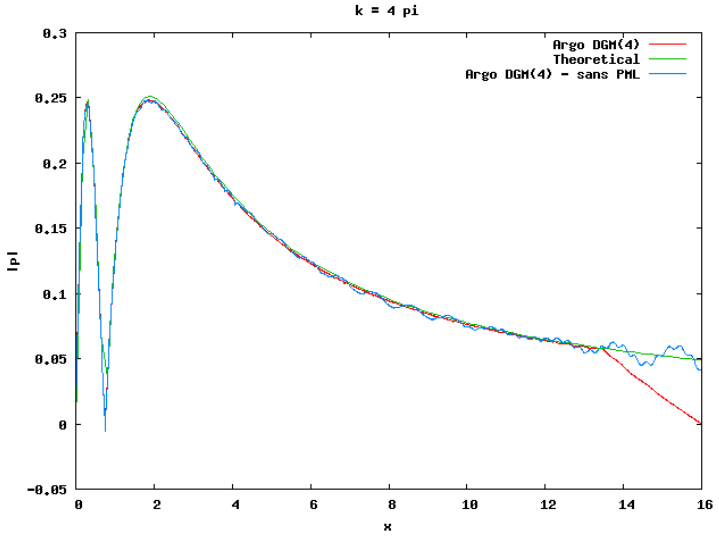
operation of the model for all conditions, modificied versions *e.g.* as proposed by Hu (63) should be implemented.

For the moment, only axisymmetric waves are implemented in the cylindrical formulation, *i.e.* $m = 0$. As an example, the upstream and downstream running wave for $\Omega = 3$ and $M_0 = 0.3$ are computed. The corresponding imposed radial pressure distributions are shown in Fig. 3.16. The computed axial variations of the pressure perturbation, at the center and the wall of the tube respectively, are shown in Fig. 3.17 and Fig. 3.18. One can see the impact of the PML, which become active at an axial distance of 3 of the non-specified boundary. Up to this distance the wave is maintained without distortion, indicating the correspondance between the numerically computed and theoretical wave form.

The group spatial frequency Γ is computed by locating the zeros of the real component of the pressure. The computed values are 1.1783 for the upstream and 0.8656 for the downstream running wave, whereas the theoretically predicted values are 1.1528 and 0.8563 respectively. One should consider that the theoretical study neglects terms in the differential equation for the pressure, that have an order of magnitude of $(M_0\Gamma)^2 \sim 1\%$ along the central axis, such that no error convergence can be done.

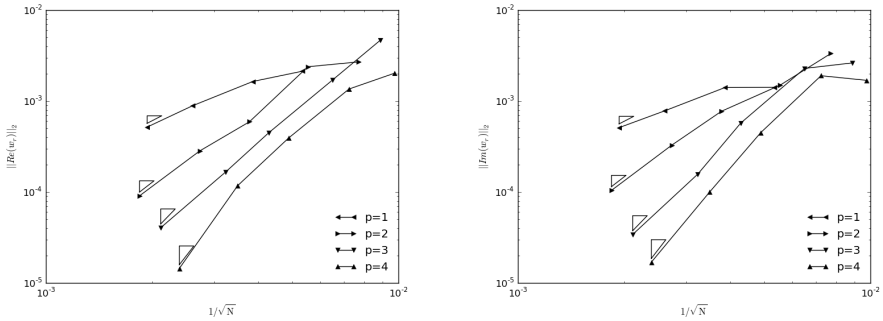


(a) $He = 2\pi$



(b) $He = 4\pi$

Figure 3.14: Circular loudspeaker - evolution of the amplitude of the acoustic pressure along the axis for $He = 2\pi$ and 4π compared to the theoretical expression. The axial distance is relative to R. The number of maxima and the extent of the near-field region are seen to increase with He. The influence of the PML is clearly noticeable in the rapid decrease of the amplitude towards the right boundary. For $He = 4\pi$ the pressure variation without PML is added for comparison, clearly showing reflections running up to the loudspeaker.



(a) Real part (b) Imaginary part

Figure 3.15: Circular loudspeaker actuated at $He = 2\pi$ - convergence of the solution error measured by the complex radial velocity on the axis. Triangles indicating the indicated convergence rate h^{p+1} are shown next to the corresponding convergence curve.

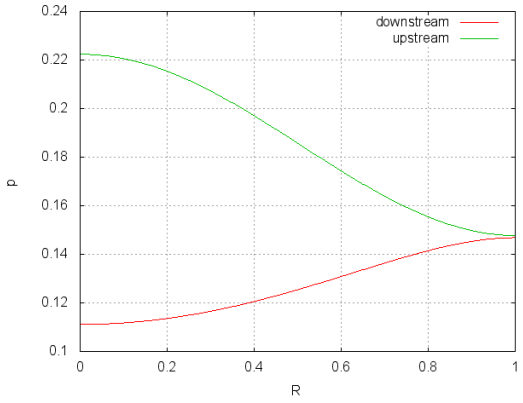
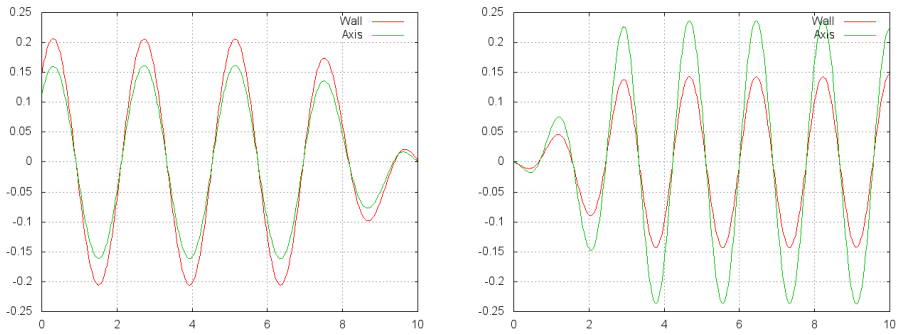
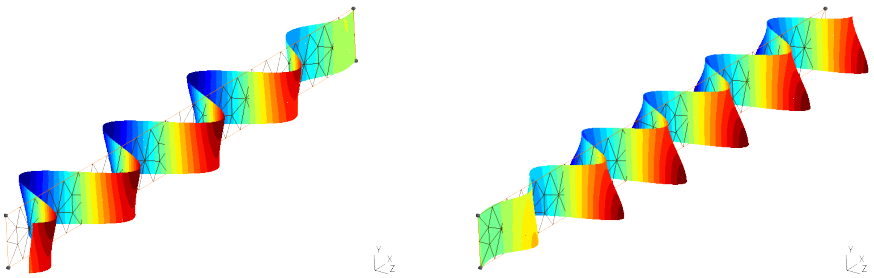


Figure 3.16: Acoustic wave propagating on top of Poiseuille flow - radial pressure profile imposed at upstream resp. downstream boundary for the computation of the downstream and upstream running wave at $M_0 = 0.3$, $m = 0$ and $\Omega = 3$



(a) Downstream running wave (b) Upstream running wave

Figure 3.17: Acoustic wave propagating on top of Poiseuille flow - computed axial variation of the pressure (real component) along the axis and the exterior wall for an axisymmetric acoustic perturbation for $m = 0$, $M_0 = 0.3$ and $\Omega = 3$.



(a) Downstream running wave (b) Upstream running wave

Figure 3.18: Acoustic wave propagating on top of Poiseuille flow - computed pressure field (real component) for an axisymmetric acoustic perturbation at $M_0 = 0.3$ and $\Omega = 3$. The x axis is aligned with the axial direction and located at the bottom, whilst y shows the radial direction. The instantaneous pressure amplitude is then visualised by a displacement in the z-direction. The flow proceeds from the bottom left to the top right corner of the figures. The PML are clearly visible near the non-specified boundary.

CHAPTER 4

ITERATIVE METHODS

From the moment the discontinuous Galerkin method has been applied to CFD, the development of efficient iterative techniques has been a very active topic of research. This is because often stationary solutions are sought, or implicit time-integration is required, mainly due to large variations in mesh size. Next to the iterative convergence, particularly important issues are the memory consumption and *operational complexity*, *i.e.* the number of elementary computation steps that need to be taken. Two iterative strategies have been developed, namely *Newton-Krylov* and *multilevel* methods.

Multilevel methods use additional iterations on "coarser" levels of discretisation in order to accelerate convergence. Indeed, most iterative methods are very efficient in removing the short wavelength errors, but perform very badly when it comes to the longer wavelengths. By reducing the resolution of the discretisation, we convert part of the long-wavelength errors with respect to the finer level to shorter wavelength errors on the coarser. These short wavelength errors are then efficiently removed by iterations on the coarser levels.

Since in the multilevel framework the optimal iterative techniques are efficient in removing high-frequency error, these methods are often referred to as *smoothers*. A very good introduction to the basic theory of multilevel methods is found in Briggs *et al.* (24), whilst Brandt (23) and Wesseling (110) provide more thorough discussions. Although this material is somewhat dated, and obviously applies to older numerical methods, it allows to understand most of the concepts and convergence issues.

In case of DGM we can distinguish two basic classes of multilevel methods. The classical *h-multigrid (hMG)* methods use coarser meshes whereas *p-multigrid (pMG)* methods use lower orders of interpolation to provide the

coarser discretisations. As the latter can only be used with higher-order methods, these methods have been developed only recently.

The use of multilevel methods allows in theory a considerable gain in memory requirements, in case matrix-free / semi-explicit iterative methods can be used on the finer levels. Although the design of such effective smoothers for CFD seems up to present an elusive goal, it is very likely that in the long iterative strategies will contain a multilevel component.

The development of multilevel methods for DGM is to date still an active topic of research in the community, initially mainly focused on the development of the definition of the coarser levels and the transfer operators, whilst only few authors have worked on the optimisation of the smoothers, namely Klaij *et al.* (70) and van der Vegt and Rheberg (102; 103).

For the discontinuous Galerkin method, the first developments of multilevel methods concern *h-Multigrid methods on nested meshes*, which were proposed by Bastian *et al.* (14), van der Vegt (104) and Kanschat (67; 68)). Subsequently *p-Multigrid* methods have been introduced by Helenbrook *et al.* (53), Oliver (82), Fidkowski (45; 46) and the author (57); this class of methods uses different orders of interpolation to provide the coarser levels, and is hence also based on hierarchically nested interpolation. Both approaches were later on combined to hp-multigrid approaches by Nastase *et al.* (80), Shahbazi *et al.* (94) and van der Vegt and Rhebergen (102; 103). Recently Tesini developed agglomeration multigrid techniques (99) for DGM which were further refined by Bassi *et al.* (10).

Newton-Krylov methods have been used in the context of discontinuous Galerkin methods from the late 90s onward. First of all a straightforward matrix-based implementation was proposed by Bassi and Rebay (12; 13).

Throughout the years, research has been dedicated to the reduction of the memory cost associated to the storage of the matrix. Early on Rasetarinera *et al.* (89) proposed a matrix-free version combined with LU-SGS as a preconditioner, whilst the comparison of matrix-based and matrix-free GMRES for RANS computations has been addressed by Crivellini *et al.* (36). Given the acceptance of matrix-free GMRES, the main memory cost is associated to the preconditioner. Most authors seem to agree on the fact that ILU is the most efficient, and use it as a reference to test new, less memory-intensive preconditioners. On the other hand, block-Jacobi is often considered too slow (83). One way to go is to reduce the weight of the off-diagonal blocks, such as proposed recently by Birken *et al.* (19).

Combined methods A significant reduction can be obtained by the combination of the Newton-Krylov method and multilevel method. Most authors consider multilevel methods as preconditioners. This is the case of Persson *et al.* (84; 85), Diosady *et al.* (37; 38; 39) and Shahbazi (94).

4.1 Newton methods

The first implementation of the Newton-GMRES method within Argo was developed and presented in (57; 59). The original implementation does not present any particularities with respect to the first published methods (*e.g.* Bassi and Rebay (13)), other than the combination with the p-multigrid cycle. The proposed method was also one of the first to feature both classical and matrix-free GMRES. Later developments include optimised data structures, single precision preconditioners and the additive Schwarz parallellisation as detailed in the VKI lecture series (58) and the ADIGMA book (56).

This section only aims at introducing the main ingredients of the method, whereas the real specificities of the implementation are discussed in chapter 5 which details the efficient data structures and assembly routines.

The damped inexact Newton method

The damped inexact Newton method defines a modified residual $\mathbf{r}^\star$ that incorporates a pseudo-temporal term:

$$\mathbf{r}^\star_{im} = \left(\phi_i, \frac{u^n_m - u^{n-1}_m}{\Delta\tau^n} + \mathcal{L}_m\left(u^n\right) \right) = 0 \ , \ \forall \phi_i \in \mathcal{V}. \tag{4.1}$$

To enhance the conditioning of the discretised system, the residual is premultiplied with the inverse of the mass matrix

$$(\mathbf{M}^{-1})_{ij}\mathbf{r}^\star_{jm} = 0 \ , \ \forall i, m \tag{4.2}$$

Only one Newton iteration is performed per timestep n - hence the denomination "inexact" - leading to the following linear system:

$$\mathbf{L}^\star \cdot \Delta\mathbf{u}^n = -\mathbf{r}^\star \tag{4.3}$$

where $\mathbf{L}^\star$ is the Jacobian matrix corresponding to the damped problem in equation 4.3, defined as

$$\mathbf{L}^\star = \mathbf{M}^{-1}\frac{\partial \mathbf{r}^\star}{\partial \mathbf{u}} = \frac{\mathfrak{I}}{\Delta\tau^n} + \mathbf{L} \tag{4.4}$$

and $\mathbf{L}$ is the Jacobian matrix associated to the original steady state problem. The pseudo-temporal term has three functions:

- it provides an appropriate underrelaxation in order to avoid unphysical states;
- the pseudo-timestepping converges linearly, and hence avoids possible divergence of the Newton approach;
- the temporal term provides an enhanced conditioning of the corresponding linear problem.

The pseudo-timestep $\Delta\tau$ is defined elementwise such that a constant CFL number is maintained across the domain. It is defined as a function of cell size h, flow conditions and interpolation order p:

$$\Delta\tau_i^n = \frac{\text{CFL}^n}{\lambda_C + \lambda_D} \tag{4.5}$$

Here λ_C and λ_D are estimates of the spectral radius of the convective and diffusive operator.

As the computation converges, we recuperate convergence by increasing the imposed CFL number as the residual is reduced:

$$\text{CFL}^n = max\left(\text{CFL}^0\left(\frac{\|\mathbf{r}^o\|_2}{\|\mathbf{r}^{n-1}\|_2}\right)^\alpha, \text{CFL}^\infty\right) \tag{4.6}$$

Here CFL^0 is the lower and CFL^∞ the upper bound, whilst $\mathbf{r}^i$ the residual vector at iteration i. The exponent α is typically chosen between $1/2$ and 1. For $\text{CFL}^\infty = \infty$ a pure Newton method is eventually obtained, hence recuperating full quadratic convergence.

Jacobian-free GMRES

The linear system 4.3 is solved using Krylov subspace methods. This class of iterative methods constructs subsequent updates p^n of the solution within the so-called *Krylov subspaces* $\mathcal{K}_n$ based on $\mathbf{r}^\star$ and $\mathbf{L}^\star$

$$\mathcal{K}_n\left(\mathbf{L}^\star, \mathbf{r}^\star\right) = \text{span}\{\mathbf{r}^\star, \mathbf{L}^\star \cdot \mathbf{r}^\star, ..., (\mathbf{L}^\star)^n \cdot \mathbf{r}^\star\} \tag{4.7}$$

Depending on the type of matrix, different Krylov subspace methods are defined. For general definite systems, the *generalised minimum residual (GMRES)* is used.

To construct $\mathcal{K}_n$ we only need an operator to construct the matrix vector products $\mathbf{L}^\star \cdot \mathbf{p}$ for arbitrary vectors $\mathbf{p}$, not the matrix $\mathbf{L}^\star$ itself. To avoid the storage of the Jacobian matrix, which is quite large for DGM, the matrix vector product is formed using a *Jacobian-free* approach, consisting of a one-sided finite difference approximation:

$$\mathbf{L}^\star \cdot \mathbf{p} \approx \frac{\mathbf{r}^\star(\mathbf{u}^n + \epsilon\,\mathbf{p}) - \mathbf{r}^\star(\mathbf{u}^n)}{\epsilon} \tag{4.8}$$

The scale factor ϵ should be small enough such that the first-order approximation is adequate, but not so small that the addition of the elements of p to $\mathbf{u}^n$ is corrupted by round-off. In practice we take

$$\epsilon = \sqrt{\mu}\frac{\|\mathbf{u}\|_2}{\|\mathbf{p}\|_2} \tag{4.9}$$

where μ is the relative round-off error associated to double precision.

This method usually does not remove the need for the assembly and storage of the tangent matrix though, because the system of equations 4.3 has a

high condition number. To improve the conditioning of the linear system, a preconditioner, $\mathbf{P}$, needs to be used. In case of a *right* preconditioner the following system is solved:

$$\begin{aligned} \mathbf{L}^\star \cdot \mathbf{P} \cdot \mathbf{y} &= -\mathbf{r}^\star \\ \Delta \mathbf{u}^n &= \mathbf{P} \cdot \mathbf{y} \end{aligned} \tag{4.10}$$

This preconditioner is not necessarily of matrix form, but can be any operator that approximates, for given $\mathbf{p}$

$$\mathbf{P} \cdot \mathbf{p} \approx \mathbf{L}^{*-1} \cdot \mathbf{p} \tag{4.11}$$

and hence any iterative method can be used. However in this work we use standard matrix preconditioners. BILU preconditioners are formed by a block incomplete LU decomposition of $\mathbf{L}^\star$ and are as such block generalisations of the ILU(k) and ILUt(k,τ) preconditioners described in (92). A second, much more economical preconditioner is the block-Jacobi preconditioner, which only retains the blocks on the diagonal.

Block ILU preconditioners

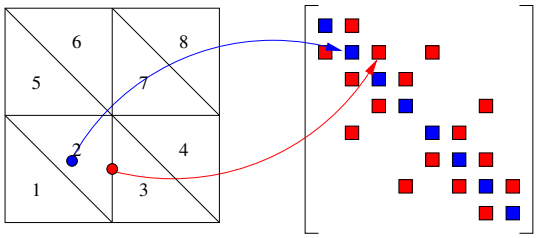


Figure 4.1: Jacobian matrix structure

Due to the fact that degrees of freedom are associated to one element only, $\mathbf{L}^\star$ is a block-sparse matrix, with large dense blocks of size $n = N_\phi N_v$. The diagonal elements correspond to the coupling between degrees of freedom of one element. As we use an interior penalty method, the off-diagonal elements can be linked directly to the coupling induced by a single interface.

The block ILU preconditioner P is formed by an incomplete LU factorisation of $\mathbf{L}^\star$ at the block level. This approximate inversion is coded using the LAPACK dense matrix block inversion on the diagonal block followed by BLAS matrix-matrix multiplication operations for the reduction of off-diagonal entries. For both BLAS and LAPACK very efficient implementations are available, which results in near-optimal inversion speed. Furthermore, the LAPACK inversion routines provide pivoting.

$$\mathbf{L}_{bc} := \mathbf{L}_{bc} - \mathbf{L}_{ba} \cdot \mathbf{L}_{aa}^{-1} \cdot \mathbf{L}_{ac} \ , \ \forall c > a \tag{4.12}$$

The approximate factorisation entails dropping some of the new blocks $\mathbf{L}_{bc}$, which would appear during the LU decomposition steps 4.12, according

to a predefined fill-in strategy. The fill-in strategy for the BILU(k) decomposition is based on a hierarchical argument, in which any block entry of the matrix is assigned a level. This level is 0 for blocks which are originally present in the matrix, whilst during the block row reduction step the level of a new entry is 1 higher than the level of $\mathbf{L}_{ik}$. *e.g.* BILU(1) allows new entries up to level 1. The main disadvantage of this method resides in the fact that we cannot control the number of additional entries.

BILU preconditioners are considered to be very efficient. On the other hand, their construction, inversion and application is quite costly, both in terms of computational time and memory footprint. Chapter 5 and in particular subsection 5.2 shows how to reduce memory footprint and computational time using single precision arithmetic, efficient data structures and optimal organisation of operations.

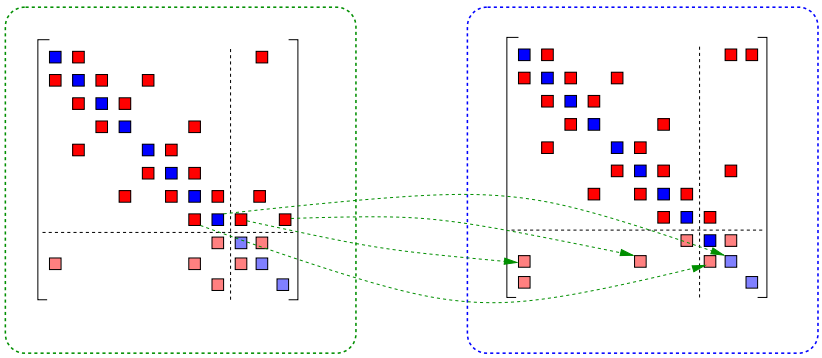


Figure 4.2: Additive Schwarz approach

The GMRES iterations are parallellised by first of all providing a parallellised internal product operator on distributed vectors. The only complicated part concerns the preconditioner; here the *Rational Additive Schwarz* approach is adopted.

As illustrated in Fig. 4.2 we first compute, for each partition, all of the block entries of the Jacobian that describe the dependence of the residual defined for the real elements with respect to all of the elements, including virtual elements. After that the diagonal and the off-diagonal blocks that couple the virtual to the real elements is communicated from the respective partitions from which the virtual elements are copied.

From there onward the decomposition and forward and backward substitution are performed on each partition separately. As no data are shared between partitions, no exchange needs to be done after the preconditioning, except during residual computation (ie. during the finite difference approximation of the matrix-vector product).

The main drawback of this approach is the relatively large memory overhead. Due to the fact that the tangent matrix has a large memory footprint, we cannot put many elements per processor, and hence the number of ghost elements is rather large with respect to the number of real elements. On the other hand, the iterations are nearly not impacted by the parallellisation.

Block-Jacobi preconditioners

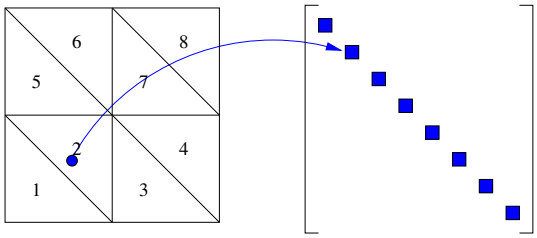


Figure 4.3: Block Jacobi preconditioners

For well-conditioned systems of equations, such as those resulting from the discretisation of a time-accurate problem, block Jacobi preconditioners are usually sufficient. These result from neglecting all the off-diagonal blocks. Thereby a very significant saving in terms of assembly time and especially memory is realised. Furthermore, as the element interaction is neglected, parallellisation is trivial, does not impact on convergence, and does not result in any memory overhead.

4.2 Multigrid methods

The first contribution of this work lies in the definition of a generic framework for the construction of interpolation operators for both h and p-multigrid, which supports moreover a generic definition of the coarser levels (57). Subsequently, a proof of the equivalence between *discretisation (DCGA)* and *Galerkin coarse grid approximation (GCGA)* for these operators, in case of nested interpolations, has been developed in (59). These contributions are detailed in the following sections, after an introduction to the theoretical background and ideas. The proposed framework for the transfer operators have subsequently been exploited for *agglomeration h-multigrid* by Tesini (99).

The Full Approximation Storage applied to DGM

The basic building block for multilevel method, as it is applied to discretisations of non-linear problems, is the two-cycle *Full Approximation Storage (FAS)* algorithm (Briggs *et al.* (24)), which is illustrated in Fig. 4.4.

The application of a multilevel method to a finite element discretisation such as DGM, entails the definition of both a "fine" and a "coarse" discretisation by introducing the corresponding function spaces $\mathcal{V}^p$ and $\mathcal{V}^q$. These will be based on two different meshes in the case of the classical multigrid, dubbed *h-multigrid (hMG)* in the remainder, or on two different polynomial orders in the case of *p-multigrid (pMG)*[1]

[1]The term p-multigrid, although misleading since both levels use the same grid, is now commonly accepted.

The FAS algorithm then solves a defect correction equation on the coarse level. This defect correction equation is driven by the fine grid residual and results in a coarse grid correction to the fine grid solution. Classically this defect correction equation is based on the discretised equations on the finest level. However, one can also define the defect correction equation as a partial differential equation for the correction, driven by the fine grid residual. This defect correction PDE is then discretised using the DGM variational formulation with the test functions on the coarse level. First the following shorthand notation for the generic form of the model equations is introduced:

$$\mathcal{L}_m(\tilde{u}) = \frac{\partial \tilde{u}_m}{\partial t} + \nabla\cdot\vec{f}_m\left(\tilde{u}\right) + \nabla\cdot\vec{d}_m\left(\tilde{u}, \nabla\tilde{u}\right) + s_m = 0\ ,\ m = 1\ \ldots\ N_v \tag{4.13}$$

Then the FAS algorithm proceeds as follows:

1. iterate on the fine level: $u^p \rightarrow u^{p\prime}$
2. restrict the fine level solution:

$$u^{q\prime} = \mathcal{T}^{qp}\left(u^{p\prime}\right) \tag{4.14}$$

3. solve the weighted defect correction equation

$$\left(\mathcal{L}\left(u^q\right) - \mathcal{L}\left(u^{q\prime}\right) + \mathcal{L}\left(u^{p\prime}\right), \phi_j^q\right) = 0\ \ \forall \phi_j^q \in \mathcal{V}^q \tag{4.15}$$

4. prolongate the correction

$$\begin{aligned} u^p &:= u^{p\prime} + \delta u^p \\ &:= u^{p\prime} + \mathcal{T}^{pq}\left(u^q - u^{q\prime}\right) \end{aligned} \tag{4.16}$$

5. iterate on the finest level to smooth the error

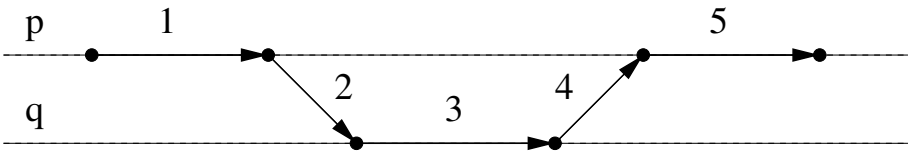


Figure 4.4: The *Full Approximation Storage (FAS)* two-level cycle accelerates the solution of the fine level problem (marked p), by combining iterations on this level (steps 1 and 5) by the iterative solution of a defect correction equation (step 3) on the coarser level (q). The impact of this defect correction equation is determined by the quality of the transfer operators (step 2 and 4) between both levels on the one hand, and the definition of a suitable discretisation and iterative scheme (step 3) on this coarser level on the other.

Transfer operators

One of the essential ingredients of the multilevel cycle are the *restriction* and *prolongation* operators. These operations define the transfer of respectively the solution and residual from the fine ($u^{p\prime}$) to the coarse level $u^{q\prime}$, and that of the coarse level correction δu^q to the fine level δu^p.

For the discontinuous Galerkin method, these operations are most naturally based on Galerkin projection. Consider a solution $u^a \in \mathcal{V}^a$ and its projection $\mathcal{T}^{ba}u^a = u^b \in \mathcal{V}^b$

$$\begin{aligned} u^a_m &= \mathbf{u}^a_{im}\phi^a_i \ , \ \phi^a_i \in \mathcal{V}^a \\ \mathcal{T}^{ba}u^a_m = u^b_m &= \mathbf{u}^b_{jm}\phi^b_j \ , \ \phi^b_j \in \mathcal{V}^b \end{aligned} \tag{4.17}$$

Orthogonalising the difference $u^a - u^b$ to the space $\mathcal{V}^b$ defines the following set of equations for the expansion coefficients $\mathbf{u}^b_{im}$

$$\left(\phi^b_k, \phi^b_j\right) \mathbf{u}^b_{jm} = \left(\phi^b_k, \phi^a_i\right) \mathbf{u}^a_{im}, \ \forall \phi^b_k \in \mathcal{V}^b \tag{4.18}$$

The solution transfer operator $\mathcal{T}^{ba}$ has the discrete or matrix equivalent $\mathbf{T}^{ba}$ defining the transfer between the expansion vectors $\mathbf{u}^a$ and $\mathbf{u}^b$

$$\mathbf{u}^b = \mathbf{T}^{ba} \cdot \mathbf{u}^a = \left(\mathbf{M}^{bb}\right)^{-1} \cdot \mathbf{M}^{ba} \cdot \mathbf{u}^a \tag{4.19}$$

where the "mixed" mass or correlation matrix $\mathbf{M}^{ab}$ is defined as

$$\mathbf{M}^{ab}_{ij} = \left(\phi^a_i, \phi^b_j\right) \tag{4.20}$$

Defect correction forcing term

The way the residual vector is restricted follows from the weighted defect correction equation 4.15.

$$\mathbf{r}^{q\prime} = (\phi^q_i, \mathcal{L}(u^p)) \tag{4.21}$$

Conventionally one goes the other way around: first the equations are discretised on the finest level, restriction and prolongation operators are defined for residual, solution and correction vectors, and then - in the best of cases - the coarse grid operator is found by applying the discrete transfer operators to the fine grid operator. This approach is called *Galerkin coarse grid approximation (GCGA)*. More frequently one discretises using the same technique on the coarse representation and then optimises the transfer operators. The latter approach is the *Discrete coarse grid approximation (DCGA)*,

To compute the restricted residual explicitly one would need to redefine routines for each of the residual contributions defined in $\mathcal{V}^p$ with shape functions in $\mathcal{V}^q$. This explicit approach is quite cumbersome in terms of implementation, but fortunately it can be computed indirectly. First one expands the residual function $\mathcal{L}(u^p)$ in $\mathcal{V}^p$ using L_2 projection:

$$\begin{aligned} \mathcal{L}_m(u^p) &\approx \sum_i \mathbf{l}^p_{im}\phi^p_i \\ \sum_i \mathbf{l}^p_{im}\left(\phi^p_i, \phi^p_j\right) &\approx \left(\phi^p_j, \mathcal{L}(u^p)\right) = \mathbf{r}^p_{jm} \end{aligned} \tag{4.22}$$

This projection only requires the Galerkin weighted residual $\mathbf{r}^p$, which is already available. The expansion weights $\mathbf{l}^p$ are not constructed explicitly, since the "restricted residual" is computed as:

$$\mathbf{r}^{q\prime}_{im} = (\phi^q_i, \mathcal{L}_m(u^p)) \approx \left(\phi^q_i, \sum \mathbf{l}^p_{im}\phi^p_i\right) \tag{4.23}$$

and hence the restricted weighted residual can be directly computed from the fine level residual:

$$\begin{aligned} \mathbf{r}^{q\prime} &= \mathbf{M}^{qp} \cdot \left(\mathbf{M}^{pp}\right)^{-1} \mathbf{r}^p \\ &= \tilde{\mathbf{T}}^{qp}\mathbf{r}^p \end{aligned} \tag{4.24}$$

such that the residual restriction matrix $\tilde{\mathbf{T}}^{qp}$ is the transpose of the solution prolongation matrix $\mathbf{T}^{pq}$.

One sees that both solution and residual transfer operators use the inverse of the mass matrix. The inversion of the mass matrix is in practice only feasible for discontinuous interpolation methods such as used by DGM, since the mass matrix is block-diagonal.

Another computationally interesting feature is that the transfer operators are parametric operations, such that they can be recast in a matrix-matrix multiplication, applied to all elements at once. As a consequence, these transfer operators are very efficient, as explained in the section 5.1 on the efficiency of algebraic primitives.

Coarse grid approximation

For a linear operator $\mathcal{L}$ the discretised equations can be recast as follows:

$$\mathbf{L}^p \cdot \mathbf{u}^p = \mathbf{s}^p \tag{4.25}$$

where

$$\mathbf{L}^p_{ij} = \left(\phi^p_i, \mathcal{L}\left(\phi^p_j\right)\right) \tag{4.26}$$

and $\mathbf{s}^p$ contains all of the constant terms coming from forcing, boundary conditions etc. Suppose that the interpolation space $\mathcal{V}^q$ is nested into $\mathcal{V}^p$, *i.e.* every shape function in $\mathcal{V}^q$ is exactly represented in $\mathcal{V}^p$:

$$\phi^q_i = \alpha^{qp}_{ij} \cdot \phi^p_j \ , \ \forall \phi^q_i \in \mathcal{V}^q \tag{4.27}$$

In the case of p-multigrid this is obviously the case, as the lower order polynomial spaces are embedded in the higher-order ones. In the case of h-multigrid this requires a hierarchy of nested meshes which are found by recursive cell subdivision. Since the mapping between elements on the different levels is exact, the projection matrices α^{qp} found by any consistent projection method are equivalent. In particular one can use again the Galerkin projection:

$$\alpha^{qp} = \left(\mathbf{M}^{qp} \cdot \left(\mathbf{M}^{pp}\right)^{-1}\right) = \tilde{\mathbf{T}}^{qp} = \left(\mathbf{T}^{pq}\right)^T \tag{4.28}$$

The above choice of restriction and prolongation operators can now be shown to be "compatible" to the coarse grid operator:

$$\begin{aligned} \mathbf{L}^q_{ij} &= \left(\phi^q_i, \mathcal{L}(\phi^q_j\right) \\ &= \alpha^{qp}_{ik} \cdot (\phi^p_k, \mathcal{L}(\phi^p_l)) \cdot \alpha^{qp}_{jl} \\ \mathbf{L}^q &= \alpha^{qp} \mathbf{L} p \left(\alpha^{qp}\right)^T \\ &= \tilde{\mathbf{T}}^{qp} \cdot \mathbf{L}^p \cdot \mathbf{T}^{pq} \end{aligned} \tag{4.29}$$

This means that in this case the application of the same discretisation on the coarser level (*DCGA*) results in a discrete operator that coincides with the optimal Galerkin Coarse Grid Approximation (*GCGA*), which results from the Galerkin projection of the discretised fine level equations by the transfer operators (see Wesseling (110)).

Two-level cycle convergence analysis

A consequence of Eq. (4.29) is that the defect correction equation can be reorganised:

$$\tilde{\mathbf{T}}^{qp} \left(\mathbf{L}^p \cdot (\mathbf{u}^{p\prime} + \mathbf{T}^{pq} \cdot (\mathbf{u}^q - \mathbf{u}^{q\prime})) - \mathbf{s}^p\right) = 0 \tag{4.30}$$

Eq. (4.30) now explicitly states that the residual vector in the space $\mathcal{V}^p$, after prolongation of the coarse level correction, is in the kernel of the residual restriction matrix. This means that another two-level iteration starting from the corrected solution will no longer lead to any correction, as the coarse level defect correction equation is satisfied. This immediately precludes the appearance of limit-cycles due to the transfer operators, and hence is a sufficient condition for convergence. This property is sufficient but not required: the appearance of spurious residual contributions due to the transfer operators can be tolerated as long as this error converges faster than the problem itself (see Wesseling (110)). Notice that the choice of the solution restriction operator is not of importance for the linear case; obviously it remains important for the non-linear case.

Now the two-grid error propagation analysis from Wesseling (110) can be extended to the proposed framework. The error vector $\mathbf{e}^p$ is defined as the difference between the current solution vector $\mathbf{u}^p$ and the final solution $\mathbf{u}^{p\prime}$, and satisfies the following equation:

$$\mathbf{L}^p \mathbf{e}^p = \mathbf{L}^p \mathbf{u}^p - \mathbf{s}^p = \mathbf{r}^p \tag{4.31}$$

The defect correction equation now becomes:

$$\tilde{\mathbf{T}}^{qp} \left(\mathbf{L}^p \cdot (\mathbf{e}^{p\prime} + \mathbf{T}^{pq} \cdot (\mathbf{u}^q - \mathbf{u}^{q\prime}))\right) = 0 \tag{4.32}$$

The error $\mathbf{e}^p$ after one two-level iteration now becomes

$$\mathbf{e}^p = \left(\mathfrak{I} p - \mathbf{T}^{pq} \cdot \left(\mathbf{L}^q\right)^{-1} \cdot \tilde{\mathbf{T}}^{qp} \cdot \mathbf{L}^p\right) \cdot \mathbf{e}^{p\prime} \tag{4.33}$$

The error $\mathbf{e}^{p\prime}$ is now decomposed in a *smooth* part $\mathbf{e}^{p\prime}_S$ belonging to the range of $\mathbf{T}^{pq}$, and a *rough* part $\mathbf{e}^{p\prime}_R$ belonging to the kernel of $\left(\mathbf{T}^{pq}\right)^T = \tilde{\mathbf{T}}^{qp}$. Since the

matrices are each others transpose, $\mathbf{e}_R^{p\prime}$ is orthogonal to the range of $\mathbf{T}^{pq}$.

$$\begin{aligned}\mathbf{e}_S^{p\prime} &= \left(\mathbf{T}^{pq} \cdot \left(\tilde{\mathbf{T}}^{qp}\mathbf{T}^{pq}\right)^{-1} \cdot \tilde{\mathbf{T}}^{qp}\right) \cdot \mathbf{e}^{p\prime} \\ \mathbf{e}_R^{p\prime} &= \left(\mathfrak{I}p - \mathbf{T}^{pq} \cdot \left(\tilde{\mathbf{T}}^{qp}\mathbf{T}^{pq}\right)^{-1} \cdot \tilde{\mathbf{T}}^{qp}\right) \cdot \mathbf{e}^{p\prime}\end{aligned} \tag{4.34}$$

After the coarse grid correction the error only depends on the rough or high order part of the initial error

$$\mathbf{e}^p = \left(\mathfrak{I}p - \mathbf{T}^{pq} \cdot (\mathbf{L}^q)^{-1} \cdot \tilde{\mathbf{T}}^{qp} \cdot \mathbf{L}^p\right) \cdot \mathbf{e}_R^{p\prime} \tag{4.35}$$

This does not mean however that the resulting error is outside of the range of $\mathbf{T}^{pq}$, or - otherwise stated - is entirely made up of "rough" components.

Iteration strategy

To further reduce computational effort, the two-cycle algorithm is used in a recursive way: one replaces the direct solution of the defect correction equation on level q by one or two two-cycles implying yet a coarser representation. On this coarser level, one can again introduce another two-cycle and so on. A first important choice is the number of subsequent two-cycles to be run on each level. In case one two-cycle is performed on each level one obtains the *V-cycle*, while two lead to the *W-cycle*. A further evolution is *Full Multigrid (FMG)* in which, starting from the coarsest level, gradually the number of levels is increased. V- or W-cycles are then performed on the already available levels. The latter strategy is aimed at rapidly evacuating large scale errors, which are not affected by the finest meshes, thereby increasing robustness and convergence speed. A particular variant of FMG is *nested iteration* which is found by applying a recursion level 0.

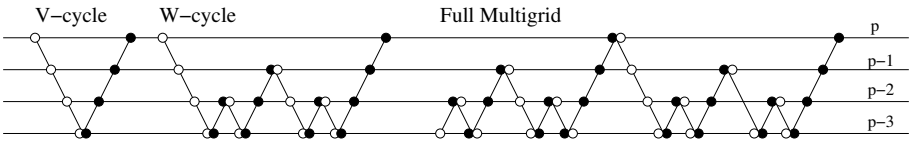


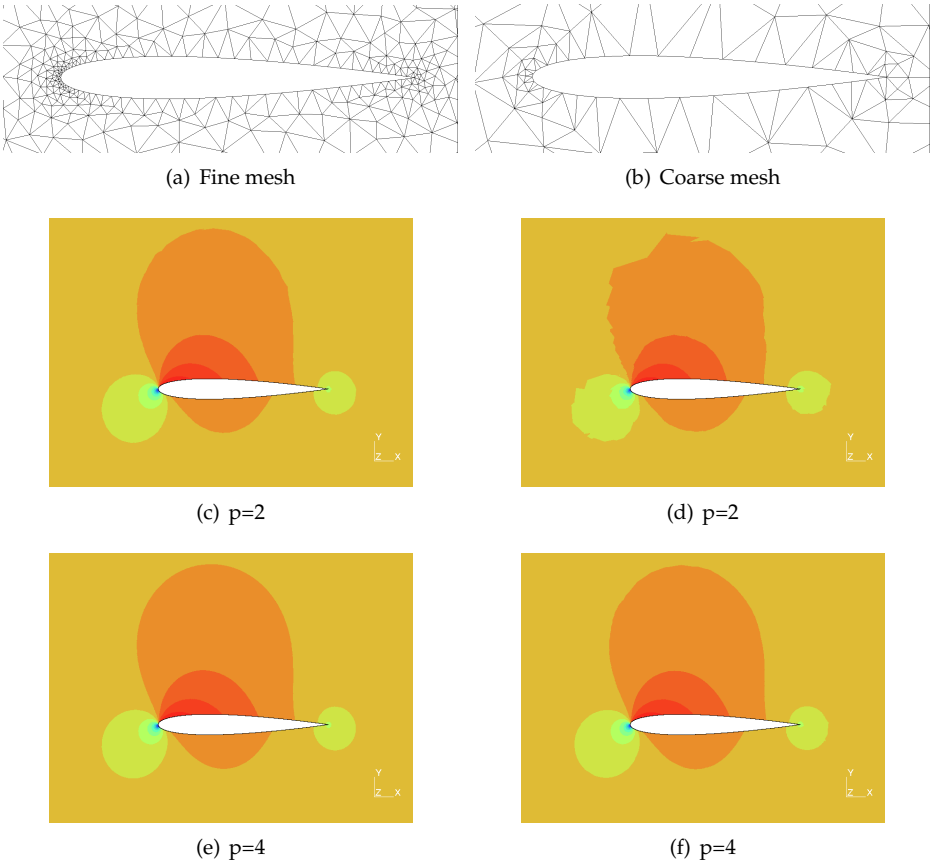
Figure 4.5: Multigrid strategies - depending on the number of recursive applications, typically 1 or 2, of the basic two-level cycle (see 4.4) on each of the levels leads respectively to the classical V and W-cycles. The *Full multigrid (FMG)* strategy gradually increases the number of levels, starting from the coarsest, to provide a rapid evolution of the transients and increase robustness of the computation. The latter is shown in combination with the W-cycle strategy

As the iterative method can be chosen freely, the defect correction equation on the coarsest level can be solved with an implicit iterative or even direct

solver. In this way one assures that the low frequency error on the coarsest level is still removed efficiently. Given the rapid decrease in computational effort when reducing interpolation order, and the limited number of elements, the overhead associated to this solution step is negligeable with respect to the fine level problem, especially for 3D computations.

Performance of the p-multigrid cycle

In this section the iterative strategies are applied to the computation of the inviscid flow around the NACA0012 profile, as illustrated in Fig. 4.6, for interpolation orders 2 and 4 on a coarse and a finer mesh. The following methods



(a) Fine mesh (b) Coarse mesh

(c) p=2 (d) p=2

(e) p=4 (f) p=4

Figure 4.6: Inviscid flow around NACA0012 - Mach number distribution

are compared:

- Explicit Runge-Kutta with local timestepping;
- Newton-Krylov iterations. The Krylov iterator is matrix-free GMRES preconditioned with ILU;

- p-multigrid V and W-cycles based on Runge-Kutta iterations;
- hybrid explicit-implicit p-multigrid V and W-cycles, using Runge-Kutta iterations on the finer, and Newton-Krylov solver on the coarsest level.

Fig. 4.7 compares the CPU time needed for the full convergence of the method.

First of all the bad performance of the Runge-Kutta method deteriorates even further with increasing order, as expected. Secondly the Newton-Krylov method outperforms any other strategy, as long as full convergence is expected. However it initially takes some time to recover quadratic convergence, especially as order or mesh resolution increases.

The most efficient p-multigrid strategy is a hybrid V-cycle, where a direct solver is used on the coarsest level. This method is competitive with the Newton-Krylov solver for the 4^{th} order computations, especially if only engineering precision, *i.e.* an iterative convergence of typically 10^{-4} up to 10^{-6}, is required. Moreover the CPU time for the fully explicit V-cycle is directly proportional to the number of unknowns per element for a given mesh, which is the best scaling one can hope for.

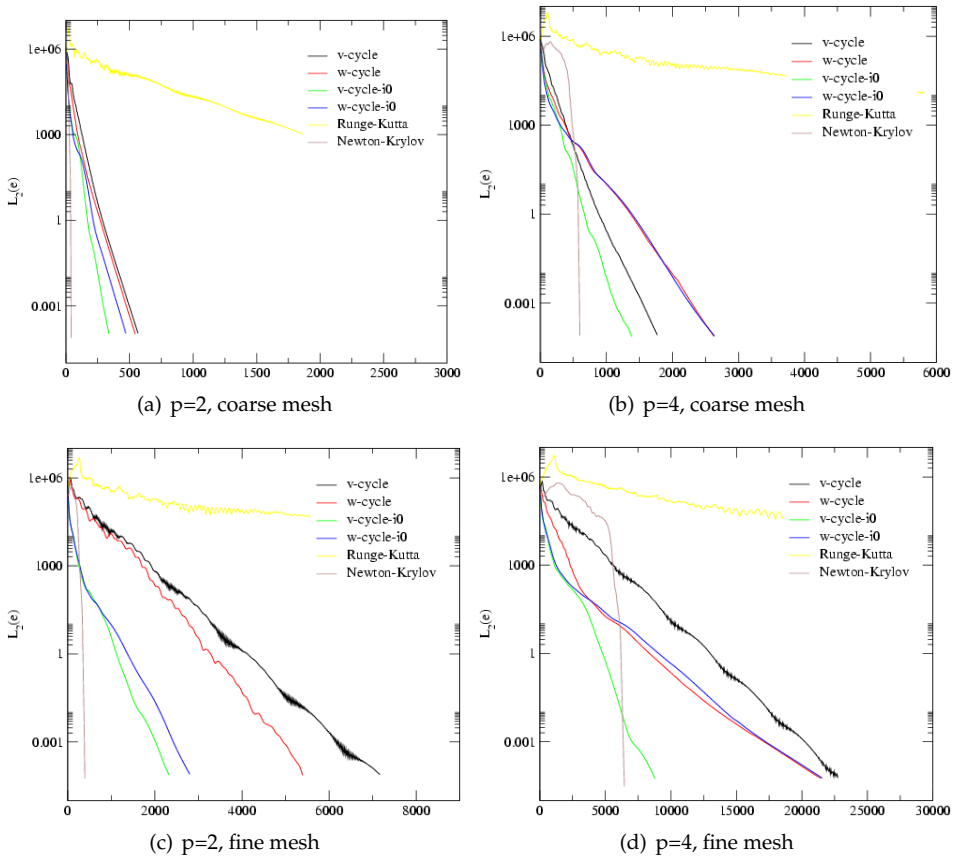
Now the different pMG strategies in terms of convergence per cycle are compared in Fig. 4.8. The fully explicit V-cycle has a convergence rate that is independent of order, meaning that *Textbook Multigrid efficiency* is obtained. This is reflected in the scaling of the CPU time as mentioned in the previous paragraph.

The scaling of the alternate strategies is less straightforward. In the case of the W-cycle, the coarser levels are visited far more often than the finest level, in contrast with the V-cycle strategy; this effect increases with interpolation order. In any case, the W-cycles perform better than the corresponding V-cycles, and this effect becomes more marked as order increases. This seems to indicate that convergence is still dominated by long wavelength error, and that a particular effort on this part of the error should pay off. This is further corroborated by the fact that the hybrid cycles always outperform their purely explicit counterpart.

The only important measure however remains the CPU time requirements. On the basis of this criterion, one sees that in case of the W-cycle the enhanced convergence rate is offset by the overhead induced by the coarse level iterations.

The important lesson is that for this 2D case we can make p-multigrid methods to be competitive with Newton-Krylov methods in terms of convergence - at a fraction of the memory cost. In 3D the comparison should even be more in favor of pMG due to the exponential scaling of the operation count and memory footprint of the Newton-Krylov strategy as a function of interpolation order.

Conversely this scaling of computational effort and memory footprint is extremely favorable with decreasing order. Hence it should normally be feasible to provide an implicit level, possibly even with a Newton-Gauss method. This is in my opinion a very attractive feature of the method, and a key ingredient of a future efficient multilevel strategy.



(a) p=2, coarse mesh

(b) p=4, coarse mesh

(c) p=2, fine mesh

(d) p=4, fine mesh

Figure 4.7: Comparison of convergence times for different iterative strategies

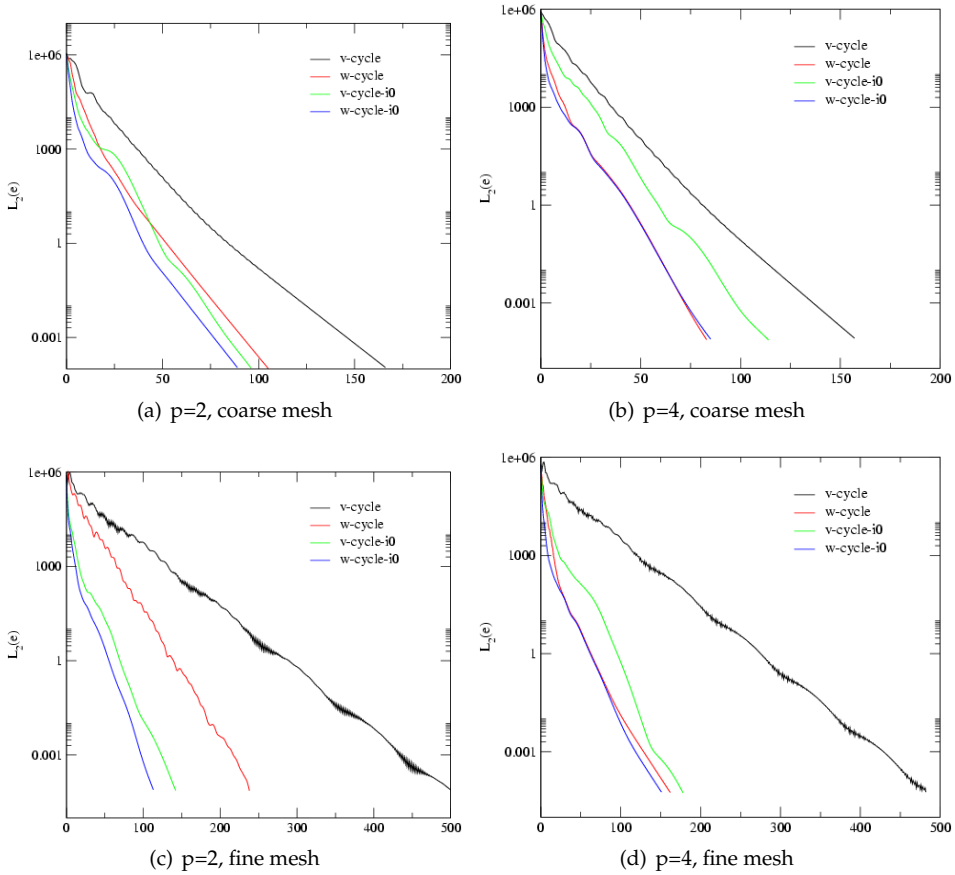
4.3 Concluding remarks

Multilevel methods

The practical application of the multilevel methods within Argo has, since some time, remained in a preliminary stage, and only the following theoretically interesting results have been obtained:

- The use of Galerkin projections and the continuous formulation of the defect correction equation results in optimal transfer operators for solution, residual and correction. In particular it was proven that in this case the *Discrete Coarse Grid Approximation* is equivalent to the optimal *Galerkin Coarse Grid Approximation* for linear problems. This optimal multigrid cycle is easy to obtain for DGM in contrast to other discretisations.

  This is only possible due to the discontinuous interpolation, which allows for an efficient Galerkin projection over the whole domain, because



(a) p=2, coarse mesh (b) p=4, coarse mesh

(c) p=2, fine mesh (d) p=4, fine mesh

Figure 4.8: Cycle efficiency for different pMG strategies

the mass matrix is block-diagonal matrix, and hence invertible element-wise.

- The proposed framework generalises the p-multigrid approaches presented earlier in (53; 82; 45). In this case the coarse level space is exactly included in the finer level $\mathcal{V}^q \subset \mathcal{V}^p$. The current approach does not need this assumption, and extends naturally to non-embedded p- and h- multigrid, providing high-quality yet very simple transfer operators.

  The framework is however also applicable to non-embedded levels. It has for instance been used for agglomeration h-multigrid by Tesini (99). An extension to generic h-multigrid is envisageable at relatively low complexity. As the projection operator is evaluated numerically on the destination level, the only required functionality is the location of integration points defined on the destination mesh in the departure mesh.

Although sufficient for purely convective problems, the current implementation has proven less successful for convection-diffusion problems. This does not reflect deficiencies in the transfer operators, but rather poor smoothing properties of the iterators. These are certainly due to large variation of the nature of the spatial operator on the domain and the large disparities between convective and diffusive contributions to the eigenspectrum.

Since there is no direct correspondence between the eigenspaces of the spatial operators on different order interpolations, as well as of the transfer operators, no clear distinction between "coarse" and "fine" level modes can be made. This entails that the only possible analysis requires the eigendecomposition of the full cycle. The choice of smoother algorithms is therefore usually only based on numerical experiments. Only very few rigorous smoother optimisations have been performed by Klaij (70) and van der Vegt and Rhebergen (102; 103), who performed optimisation on the basis of the eigendecomposition of a full cycle respectively in the case of h-multigrid and hp-multigrid. The optimised smoothers seem however very dependent on the particularities of the discretisation and the local physics of the flow.

Newton-GMRES

Currently Newton-GMRES is the standard solver for Argo. Although no fundamental developments were done, a number of practical steps have allowed to drastically reduce computational cost of the method

- Argo featured one of the first published matrix-free GMRES methods (57);
- the use of single precision matrix preconditioners has been shown to be an effective way to reduce memory consumption of the ILU, without changing iterative convergence (58). An additional advantage is the higher computational throughput due to additional vectorisation.
- the block-Jacobi preconditioner is usually sufficiently performant for unsteady computations, and very cost-effective with respect to block-ILU. Moreover, its parallellisation is trivial and very effective;
- for unsteady computations, the preconditioner can usually be frozen throughout the Newton iterations.

The last two points have not been demonstrated in the text, since no detailed assessment has been performed up to now. It is however the standard setting for unsteady applications, and has been applied for the computations in noFUDGe in chapter 6.

CHAPTER 5

EFFICIENT DATA STRUCTURES

This chapter discusses the efficient implementation techniques for the discontinuous Galerkin method. Computational efficiency is extremely important, since high order methods require many more floating point operations *per degree of freedom* than low order methods, in particular when the Jacobian matrix needs to be assembled. High-order methods will only be competitive if the increase in computational cost is compensated by the decrease of the number of elements for a given precision. Recently Cantwell *et al.* investigated the trade-off between precision and computational cost (27). As expected, higher accuracy requirements favor the use of high-order methods.

To further increase the applicability of high order methods, a number of groups work on the development of simplified discretisation methods, such as spectral elements based on tensor product functional spaces by Sherwin and Karniadakis (95) and Kopriva and Gassner (71), as well as quadrature free methods by Atkins and Shu (8). These simplified discretisation methods can be shown however to suffer from aliasing, as discussed by Hesthaven and Warburton (55) and Bassi and coworkers (11). Another relevant non-linear instability mechanism was proposed by the author (57) and applied to the quadrature-free discretisation of the Euler equations, in one of the first implementations of Argo. The main line of thought is repeated in appendix D.

Computational complexity is however not directly proportional to computational cost. It will be shown in this chapter that the computational density and the elementwise independent interpolation allow for a very efficient implementation of both assembly and linear algebra. For practical interpolation orders, the structured nature of DGM can be exploited to compensate the increase of computational complexity with order by increased operation efficiency.

This optimisation is based on the increase of the efficiency of algebraic primitives, which are used for assembly operations and linear algebra, with matrix and vector size. Both obviously grow with interpolation order.

In the first two sections the dense algebra libraries BLAS (20) and LAPACK (5) are discussed. First the layout of the data in memory is presented, followed by reference efficiency of the different relevant operations. Although the absolute figures are architecture specific and somewhat dated, the conclusions of this section are important for the computational efficiency optimisation, discussed in the following sections. The third section discusses the layout of the data structure of global vectors and matrices, whilst the last section explains the organisation of the assembly routines. This work has been presented by the author at the VKI lecture series (58), as part of the Adigma book (56). Lambrechts has continued and has further refined the approach in his doctoral thesis (73).

5.1 Algebraic primitives on the computer

This section explains the layout of floating point vectors and matrices in computer memory, and illustrates the variability and evolution of computational performance of the algebraic primitives implemented by the standards *BLAS* and *LAPACK* which are relevant to this work.

Data layout for dense vectors and matrices

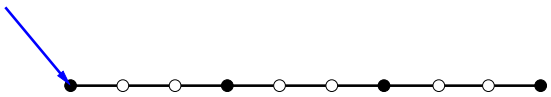


Figure 5.1: A vector is represented in computer memory by the triplet composed of pointer to the start indicated by the arrow, the total length and the stride.

A vector $\mathbf{y}$ (Fig. 5.1) is defined by the triplet $(\mathbf{y}^*, N, S)$, where $\mathbf{y}^*$ is the pointer to the data array, N the vector dimension, and S is the stride; an entry is then found as

$$y_j = \mathbf{y}^*[j \cdot S] \tag{5.1}$$

When a vector is allocated, typically $S = 1$. The stride is then used for the construction of *proxies*, *i.e.* reinterpretations of subsets of the vector. In this case only vector proxies are allowed, consisting of values at regular intervals in the original vector.

A matrix $\mathbf{A}$ is in reality stored by reinterpreting a linear array of values as sequence of columns (column-major ordering) or rows (row-major ordering) as shown in Fig. 5.2. Row-major ordering is used in this work. The matrix $\mathbf{A}$ is then specified by the quadruplet

$$(\mathbf{A}^*, N_I, N_J, N) \tag{5.2}$$

Obviously $\mathbf{A}^*$ is the pointer to the data array while N_I and N_J are the number of rows and columns. The rows are found by cutting the linear array in chunks

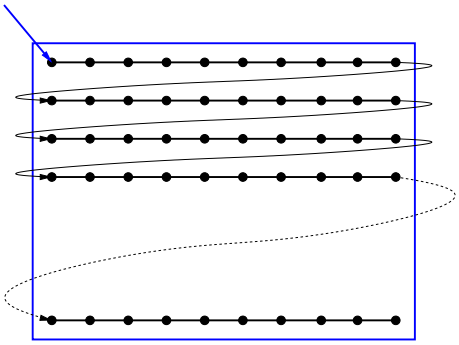


Figure 5.2: A matrix is described by the reinterpretation of a linear array. In the case of row-major ordering, this is done by cutting this array into subsequent rows. The quadruplet $(\mathbf{a}, N_I, N_J, L)$ characterises this matrix. Here $\mathbf{a}$ (indicated by the arrow) is the pointer to the start of the data, N_I and N_J are the matrix dimensions and L is the leading dimension, which indicates the interval at which the linear array needs to be cut. Here $L = N_J$.

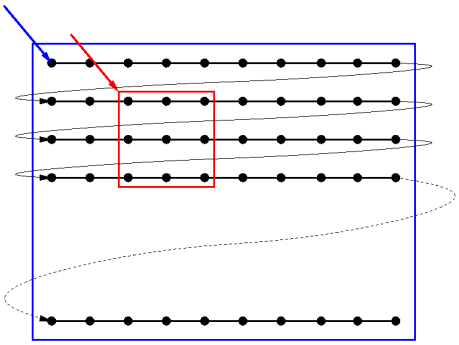


Figure 5.3: Suppose the matrix $\mathbf{A}$ is characterised by $(\mathbf{A}^*, N_I, N_J, L)$. The submatrix proxy starting from entry (i, j) and dimensions $N'_I \times N'_J$ is characterised by the modified quadruplet $(\mathbf{A}^* + i \cdot L + j, N'_I, N'_J, L)$. Notice that the leading dimension is still L.

of size L, the *leading dimension* of the matrix. In the simplest case where all of the values of the matrix are contiguous in memory, as shown in Fig. 5.2, both leading dimension and row size are equal $L = N_J$. A matrix entry A_{ij} is found as:

$$A_{ij} = \mathbf{A}[i \cdot L + j] \tag{5.3}$$

Several proxies can now be defined. A submatrix proxy A' of A, as illustrated in Fig. 5.3, starting at i', j' and of dimensions $N'_I \times N'_J$ is then defined by the quadruplet

$$(\mathbf{A}^* + +i' \cdot N + j', N'_I, N'_J, L) \tag{5.4}$$

Notice that the proxy retains the leading dimension of the original matrix. Vector proxies $\mathbf{r}$ and $\mathbf{c}$ of size n on respectively a matrix row or column are

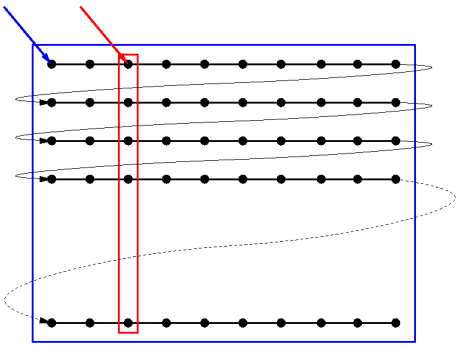


Figure 5.4: A column vector proxy of length n, starting on position i and column j of the matrix $\mathbf{A}$ is defined by the triplet $(\mathbf{A}^* + i \cdot L + j, n, L)$. Notice that the vector stride is equal to the leading dimension of the matrix.

defined by the vector triplets (see Fig. 5.4)

$$\begin{aligned} \mathbf{r} &\leftarrow (\mathbf{A} + i \cdot L, n, 1) \\ \mathbf{c} &\leftarrow (\mathbf{A} + j, n, L) \end{aligned} \tag{5.5}$$

Notice that in case of the column proxy the stride is equal to the leading dimension of the original matrix.

A first important point is that these proxies allow reinterpretation of the data without copying it. Furthermore these proxies can be defined recursively, *i.e* on top of other proxies. Finally due to the row-major ordering, only proxies that span all columns in the original matrix refer to data that are stored contiguously in memory; hence such proxies can be recast as vectors of size $N_I \times N_J$ during algebraic operations.

BLAS and LAPACK reference performances

BLAS and LAPACK are the oldest standards for the dense linear algebra operations used in scientific computing. BLAS provides basic matrix and vector operations (addition, multiplication, copy, norms, ...)[1] whilst LAPACK implements linear algebra, in particular several matrix decompositions and system solvers. Any Unix/Linux architecture has a native implementation of both standards, and quite a number of highly optimised freeware (ATLAS, GotoBLAS, ...) and proprietary implementations (MKL, ACML, ...) exist.

The performance of MKL, Atlas and finally the native GNU implementation of BLAS and LAPACK are compared on an Intel Core2^{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 (*i.e.* no strides). The operations are tested in configurations that are close to their use within the code, in order to provide an estimate of the ideal performance. During

[1] BLAS is organised following 3 levels: level 1 concerns vector-vector operations, level 2 matrix-vector and level 3 matrix-matrix operations

the measurements, the operation was repeated as many times as necessary to reduce the measurement error due to clock granularity to less than 0.5%, and the memory location was refreshed each time to avoid unrealistic cache effects. The flops (*Floating point Operations per Second*) rate was then computed based on the total number of additions and multiplications.

The layout of x86 processors (Intel/AMD) featured up until recently two floating point registers of two doubles each. Hence 4 double precision floating operations can be performed per cycle, leading to a optimal flops rate of 10 GFlops[2]. Single precision instructions are usually further packed, leading to 8 operations per cycle[3] and hence 20 GFlops. These optimal performances are obviously rarely obtained, since performance is limited by the rate at which data can be accessed. The data which is currently used is copied from standard memory (RAM) to fast memory close to the processor (cache). The more operations are done on the data in the cache, the less it needs to be refreshed, and the less data transfer will impact on performance.

Scaled vector addition

The level-1 BLAS operator *axpy* implements a scaled addition of two vectors:

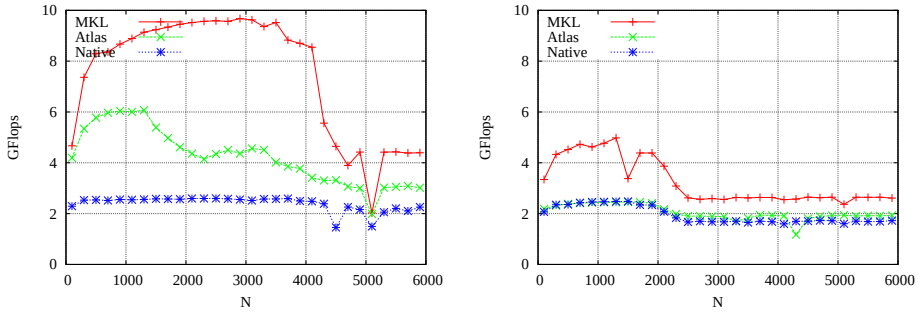
$$\begin{aligned} &\text{axpy}(n) : \mathbf{y} \leftarrow \alpha\mathbf{x} + \mathbf{y} \\ &\alpha \in \mathbb{R},\ \mathbf{x}, \mathbf{y} \in \mathbb{R}^n. \end{aligned} \tag{5.6}$$

The number of floating point operations on a vector of size n is $2n$. Since the number of operations is only proportional to the number of entries in the vectors, the axpy operation is very sensitive to cache size. The first measurements as shown in Fig. 5.5(a) for single and in Fig. 5.5(b) for double precision, recycle the same vectors time after time. For MKL a fast ramp up to a maximum speed plateau is found, which is sustained until the L1 cache size is no longer capable of holding both x and y (this corresponds to a vector size $n = 4096$ in single and $n = 2048$ in double precision). Atlas tails off much earlier on in single precision, for unclear reasons. This first scenario, which reuses the same vectors over and over again, is clearly not realistic, but is included for comparison. In a more realistic scenario, corresponding to the use of axpy during the Jacobian assembly (see subsection 5.3), the output vector $\mathbf{y}$ is continuously refreshed, whilst keeping the addendum $\mathbf{x}$ constant. More precisely, $\mathbf{y}$ is replaced by the next vector in memory. Fig. 5.6 shows the performance for this scenario. Since axpy is sensitive to cache, the performance deteriorates significantly. The performance of Atlas and MKL is similar, at least for the sizes encountered during Jacobian matrix assembly [4] (see subsection 5.3).

[2]For intel and AMD architectures local vectorisation (*instruction level parallellism / ILP)* is provided by the SSE part of the x86 instruction set. Until recently, x86 processors were equipped with 2 128bit wide floating point registers. In the latest Intel processors (from Sandy bridge onward), the register size is twice as large thereby doubling theoretical throughput.

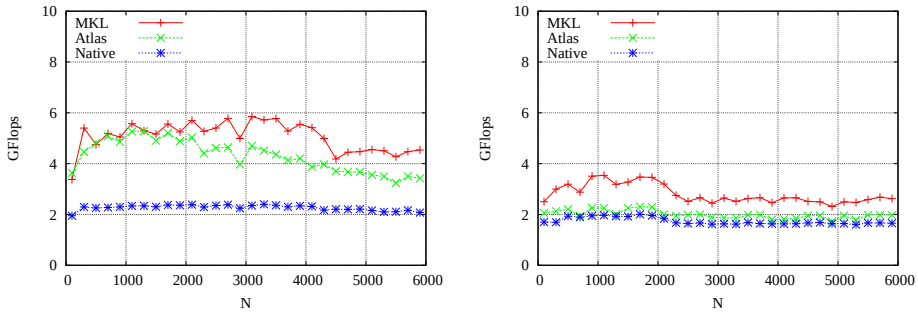
[3]*e.g.* BlueGene/P architectures before BG/Q have (theoretically) the same throughput for both single and double precision. x86 architectures on the other hand perform this packing since the introduction of SSE.

[4]Currently implicit computations are run with cubic polynomials, resulting in a vector size between $20 \times 20 = 400$ (tetrahedron) and $64 \times 64 = 4096$ (hexahedron)



(a) saxpy(n) (b) daxpy(n)

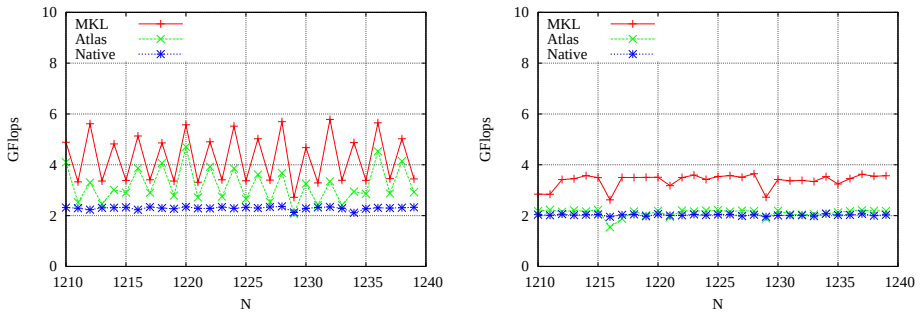
Figure 5.5: CPU measurements for the scaled vector sum operation recycling the same vectors - single (saxpy) versus double precision (daxpy).



(a) saxpy(n) (b) daxpy(n)

Figure 5.6: CPU measurements for vector sum operation when y is systematically replaced - single (saxpy) versus double precision (daxpy) for different BLAS implementations

The different graphs of Fig. 5.6 still hide an important feature, as the performance is shown per increment of 200 of the vector size. Surprisingly the graph Fig. 5.7, obtained with unit increment, show that for single precision operations, there is up to 50% difference between the performance for odd- and even-sized vectors; in fact the performance for single precision odd-size vectors is more or less the same as for double precision vectors. Moreover, multiples of 4 are usually most efficient. This is probably a data alignment issue. It is surprising though that the performance is impacted so heavily, considering that the vectors should be sufficiently long to amortize misalignment. It is equally surprising that only single precision is affected. An important lesson is that performance can be strongly enhanced by artificially lengthening the vector by an additional irrelevant entry - the additional effort of adding a single value is more than offset by the time gained due to the higher perfor-



(a) saxpy(n) (b) daxpy(n)

Figure 5.7: Detail of CPU measurements for the vector sum operation when y is systematically replaced - single (saxpy) versus double precision (daxpy) for different BLAS implementations

mance. This procedure is called *data padding*. Notice also that larger vectors have worse performance.

A similar operation for matrices

$$\begin{aligned} &\mathbf{A} \leftarrow \alpha\, \mathbf{B} + \mathbf{A} \\ &\alpha \in \mathbb{R},\ \mathbf{A}\ \mathbf{B} \in \mathbb{R}^{m \times n} \end{aligned} \tag{5.7}$$

is not provided in BLAS. Given the row-major ordering this operation can be implemented as a series of m contiguous axpy operations on a row per row basis. However if both matrices are either original matrices or proxies that span all columns of the original matrix, the sum can be reinterpreted as a single axpy of considerable size $m \times n$. This leads to a significant increase of the speed.

Scaled matrix vector multiplication

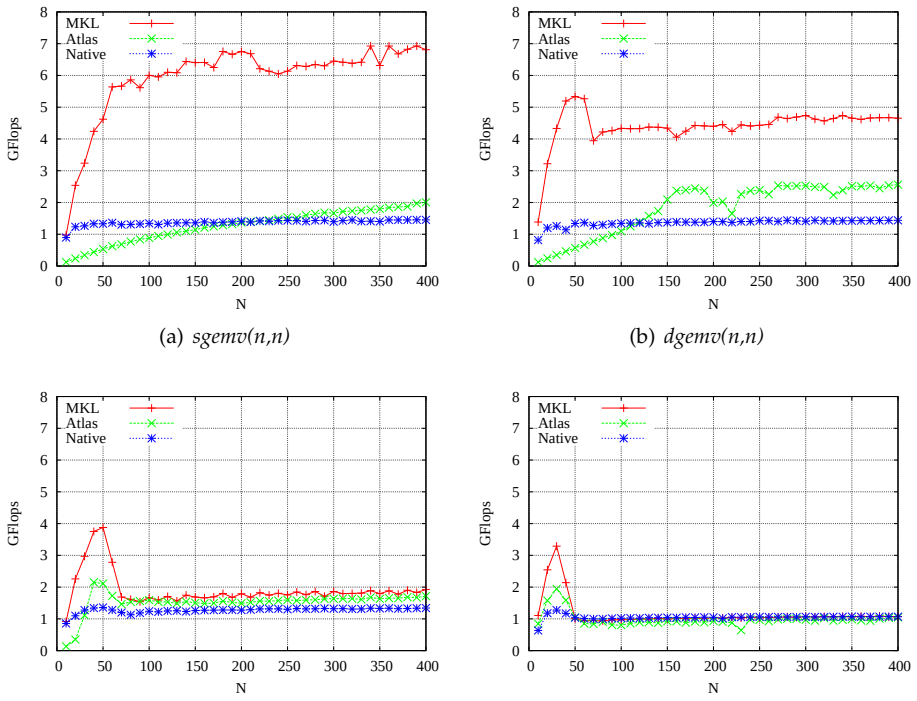
Level 2 BLAS defines the matrix-vector product *gemv* as

$$\begin{aligned} &\text{gemv}(m, n) : \mathbf{y} \leftarrow \alpha\, \mathbf{A} \cdot \mathbf{x} + \beta \mathbf{y} \\ &\alpha, \beta \in \mathbb{R},\ \mathbf{y} \in \mathbb{R}^m,\ \mathbf{x} \in \mathbb{R}^n,\ \mathbf{A} \in \mathbb{R}^{m \times n} \end{aligned} \tag{5.8}$$

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.

The problem is that gemv is used in the back-and forward substitution steps (see subsection 5.2), 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. Fig. 5.8 compares both configurations. Thereto contiguous blocks of 400 matrices and vectors are allocated; in the first configuration both matrices and vectors are accessed in a sequential manner. In the second

realistic scenario, memory is accessed randomly within those two blocks. A very dramatic loss of performance is seen for the second scenario from the point where the level 2 cache (8 MB) is no longer capable of containing all matrices and vectors.



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

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

Figure 5.8: CPU measurements for the scaled matrix-vector multiplication and addition - single (sgemv) vs double (dgemv) precision (left-right) and organised versus random access (top-bottom)

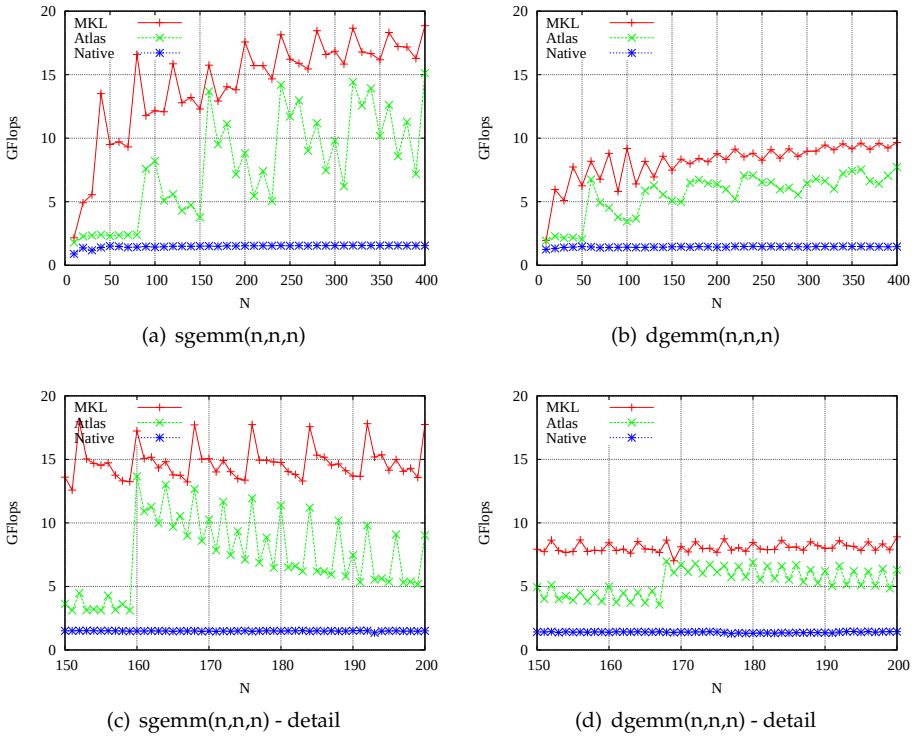
Scaled matrix-matrix multiplication and addition

Level 3 BLAS defines the matrix-matrix multiplication operator *gemm* as

$$\begin{aligned} &\text{gemm}(m,n,o) : \mathbf{A} \leftarrow \alpha\,\mathbf{B}\cdot\mathbf{C} + \beta\,\mathbf{A} \\ &\alpha,\beta \in \mathbb{R},\ \mathbf{A}\in\mathbb{R}^{m\times o},\ \mathbf{B}\in\mathbb{R}^{m\times n},\ \mathbf{C}\in\mathbb{R}^{n\times o} \end{aligned} \tag{5.9}$$

The number of floating point operations is $2mno + 2mo$. Typical timings for $\text{gemm}(n,n,n)$ are shown in Fig. 5.9, this time only for a random access to memory (*cfr.*. 5.1). The performance is again very dependent on matrix size, both in single and double precision, with a particular good efficiency for multiples of 4 in double and multiples of 8 in single precision. Again it is found that padding can enhance significantly performance.The timings are nearly

independent of memory access, and no notable cache size effect has been observed.



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

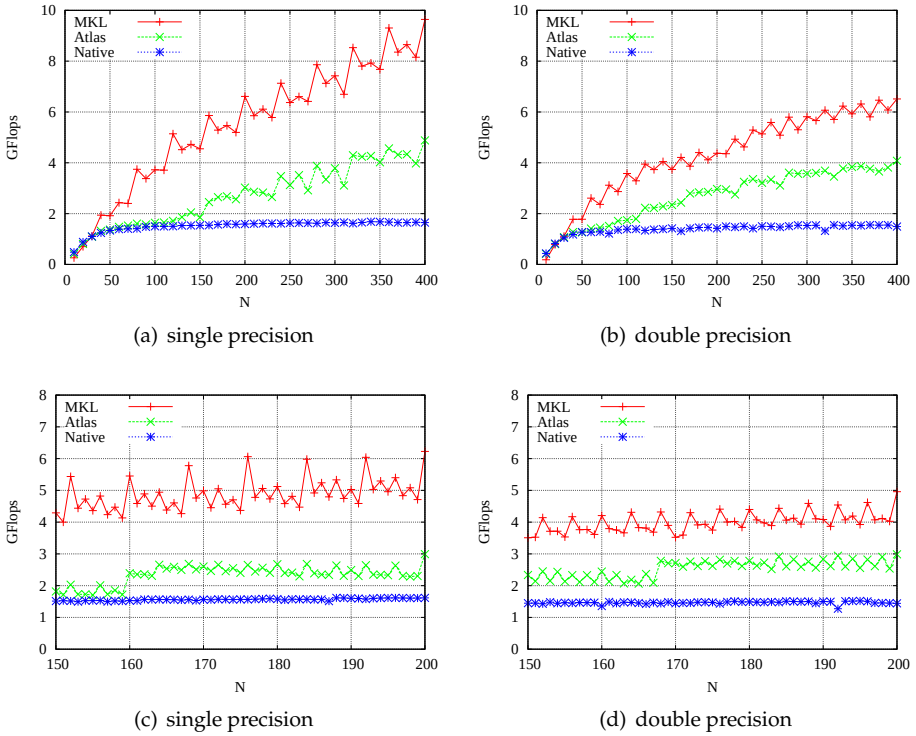
(c) sgemm(n,n,n) - detail (d) dgemm(n,n,n) - detail

Figure 5.9: CPU measurements for the scaled matrix-matrix multiplication and addition - single (sgemm) vs double precision (dgemm).

Matrix inversion

The LAPACK routine *gesv* implements the LU-decomposition of a matrix, followed by the subsequent solution by back and forward substitution of a set of vectors. By providing the columns of the identity matrix, the columns of the inverse of the matrix are found on output. This routine is used during the LU-decomposition of the Jacobian matrix (see 5.2), since the inverse of the diagonal block is stored.

The total number of operations of this inversion is given by $8/3n^3$. Timings are given in Fig. 5.10, again showing the strong dependence on matrix size. The speed is much lower than that of the matrix-matrix multiplication, probably due to the need for pivoting.



(a) single precision (b) double precision

(c) single precision (d) double precision

Figure 5.10: CPU measurements for the matrix inversion. Left graphs are using simple precision while right ones are for double precision.

5.2 Data Structures

Data vectors

As in DGM neither the solution nor the residual are shared between elements, it is possible to store all of the data belonging to one element in contiguous blocks. The solution and residual vectors are in fact stored as matrices. In this format elements are stored as successive blocks along the column index (see Fig. 5.11). Per element block the row index, i, corresponds to data for the i^{th} interpolation or quadrature point, while the variable index, m, corresponds to the column, as suggested by the matrix indices used throughout the text (*e.g.* $\mathbf{u}_{im}$). This data structure was described in (33).

This format has the advantage that common operations such as collocation (interpolation to the quadrature points) can be recast as matrix-matrix multiplications:

$$\mathbf{u}^*{}_{qm} = u_m(\alpha_q) = \sum_{i=1}^{N_\phi} \phi_i(\xi_q)\mathbf{u}_{im} = \sum_{i=1}^{N_\phi} C_{qi}\mathbf{u}_{im} \tag{5.10}$$

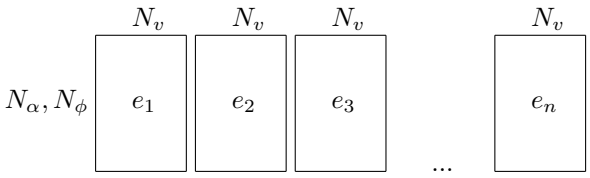


Figure 5.11: Data Structure of a data vector

Since a hybrid mesh is used, and the number of interpolation and quadrature points varies on the domain, depending on the type and order of the elements. Therefore the elements are grouped, and the data vector is then compartimented into matrices corresponding to each of these groups.

Jacobian matrix

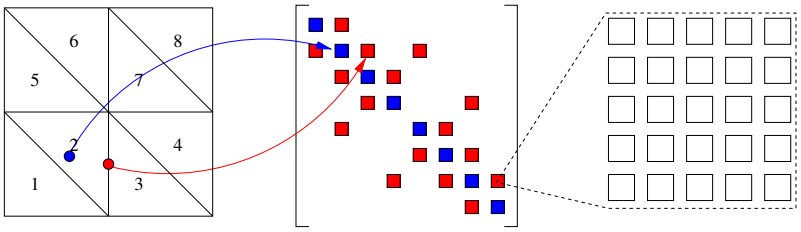


Figure 5.12: Data Structure of the Jacobian matrix

The Jacobian matrix $\mathbf{L}$

$$\mathbf{L} = \frac{\partial \mathbf{r}}{\partial \mathbf{u}} \tag{5.11}$$

is stored in blocked sparse format. Each diagonal block entry $\mathbf{L}_{aa} \in \mathbb{R}^{n \times n}$ corresponds to the coupling between all variables within the element a while the off-diagonal entries $\mathbf{L}_{ab}$ and $\mathbf{L}_{ba} \in \mathbb{R}^{n \times n}$ corresponds to the direct neighbour coupling induced by the interface between elements a and b. All entries are stored as square dense matrices of size $n = N_\phi \cdot N_v$, using row major ordering. The blocks are further partitioned 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. Scalar entries are then denoted as follows

$$(\mathbf{L}_{ab})_{ij}^{kl} \,. \tag{5.12}$$

Each submatrix $\mathbf{L}_{ab}^{kl}$ is accessed during assembly using submatrix proxy as shown in Fig. 5.3; hence it is accessed as a collection of N_ϕ rows of size N_ϕ.

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 LU decomposition (gesv) and matrix-matrix products (gemm):

$$\mathbf{L}_{bc} := \mathbf{L}_{bc} - \mathbf{L}_{ba} \cdot \mathbf{L}_{aa}^{-1} \cdot \mathbf{L}_{ac} \ , \ \forall c > a \tag{5.13}$$

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

$$\mathbf{a}_c := \mathbf{a}_c - \sum_{a>c} \mathbf{L}_{ab} \cdot \mathbf{a}_b \tag{5.14}$$

Before any decomposition, the block rows are reordered.

- in the case of an ILU decomposition, bandwidth is reduced using a reverse Cuthill-McKee procedure;
- in case of a complete inversion the fill-reducing ordering algorithms of the library Metis (Karypis *et al.* (69)) reduce the memory footprint.

During matrix-vector multiplication, or back and forward substitution, the solution vector needs to be copied in a more adapted linear format, and this operation is combined with renumbering of its entries; as the matrix-vector product or substitution scale as n^2, a copy which scales as n already has negligeable cost for relatively small values of n. The integration of the renumbering in the matrix-vector product allows for full flexibility of element numbering for the evaluation of the residual and the Jacobian.

Due to the large size of the blocks, and the associated cost of the dense matrix operations, a flexible data structure can be used to store off-diagonal blocks, consisting 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 tresholded ILU (see Saad (92)), and even variable block size without noticeable overhead.

Storage requirements and precision

In 3D all elements except those on the boundary are connected to 4 or more other elements. As such, the storage requirements amount to slightly less than $5n^2$ to $7n^2$ floating point values per element, with $n = N_\phi N_v$. Due to the structure of the equations, some sparsity in the off-diagonal blocks could in theory be exploited. However, keeping track of this would require storing the structure for each off-diagonal entry, and hence greatly complicate (and slow down) all proxies and operations.

Since in 3D N_ϕ scales as the cube of the interpolation order, the memory required for the Jacobian will be much larger than that for all other data from relatively low interpolation orders on; for instance in case of 4th order polynomials on tetrahedra, about 1.2MB per element is needed to store $\mathbf{L}$ in double precision, as compared to the solution vector itself which would require only about 1.4 kB per element. This memory footprint is quite impressive, so the prospect of halving it by storing $\mathbf{L}$ in single precision is quite appealing.

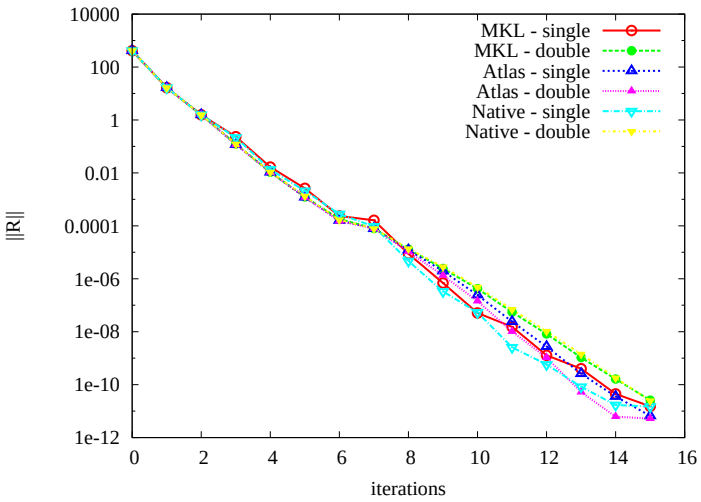


Figure 5.13: Effect of preconditioner floating precision on Newton-GMRES-ILU(1) convergence. It can be seen that single precision preconditioners perform as well as double precision versions, for lesser computational cost and memory.

Fig. 5.13 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 libraries. Apparently the precision has only as much impact as the choice of linear algebra library. The differences between libraries are probably due to the sequence of operations and pivoting strategy during the inversion of the diagonal blocks; these differences then get further amplified by the impact on the CFL evolution (see equation 4.5).

The use of a single precision preconditioner apparently has no significant impact on the convergence. This could be explained by the following considerations:

- GMRES is integrated into a Newton iteration, and in this framework a very high degree of linear convergence is not useful. Typically we only need about two orders of magnitude. In that case, single precision is more than sufficient to avoid significant round-off errors;
- the matrix-matrix product is formed by finite differences and retains hence full precision; the Jacobian matrix is only used for constructing an approximate inverse for preconditioning;
- the matrix inversion implemented by LAPACK includes a pivoting strategy which goes a long way towards solving potential near-singularity and bad conditioning.

Within the linear algebra community, some efforts are currently underway to exploit single precision efficiency to get double precision accurate solutions. A relatively old procedure is the so-called iterative refinement procedure (see

Golub and Van Loan(49), section 3.5.3). This is a method that provides double precision solutions of a linear system of equations, say $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$, using Newton steps with single precision approximation of the inverse

$$\mathbf{x}^{n+1} = \mathbf{x}^n - (\mathbf{A}^*)^{-1} \cdot (\mathbf{A} \cdot \mathbf{x}^n - \mathbf{b}) \tag{5.15}$$

This procedure converges provided that the condition number of the matrix does not exceed the inverse of the single precision relative round-off. Baboulin *et al.* (9) investigate the application of this algorithm for use within LAPACK. Arioli and Duff (6) prove that if flexible GMRES (FGMRES) is preconditioned with a single precision inverse of the matrix, the residual converges to double precision accuracy as long as the minimisation problem in GMRES has a solution. This application is very similar to the use of Jacobian-free GMRES, preconditioned with a single precision ILU decomposition.

Performance

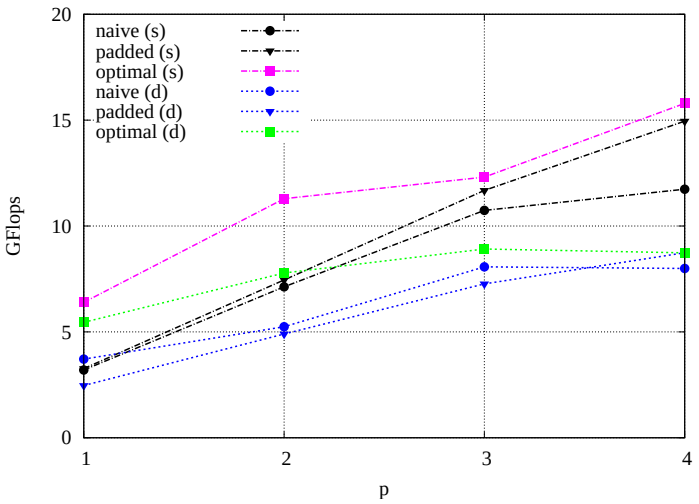


Figure 5.14: Performance of the ILU(1) decomposition (MKL)

The effects of precision and block *padding* on the floating performance of the ILU(1) matrix deomposition are illustrated in Fig. 5.14 for the MKL library, as a function of the DGM interpolation order p for a Navier-Stokes computation ($N_v = 5$). The reference optimal performance f_D^* has been computed from the measured flops rates for the gemm and inversion operations (f_G and f_I respectively), the corresponding number of operations (N_G and N_I) and the number of flops involved (F_G and F_I resp.).

$$f_D^* = \frac{f_G f_I \cdot (N_G F_G + N_I F_I)}{f_I N_G F_G + f_G N_I F_I} \tag{5.16}$$

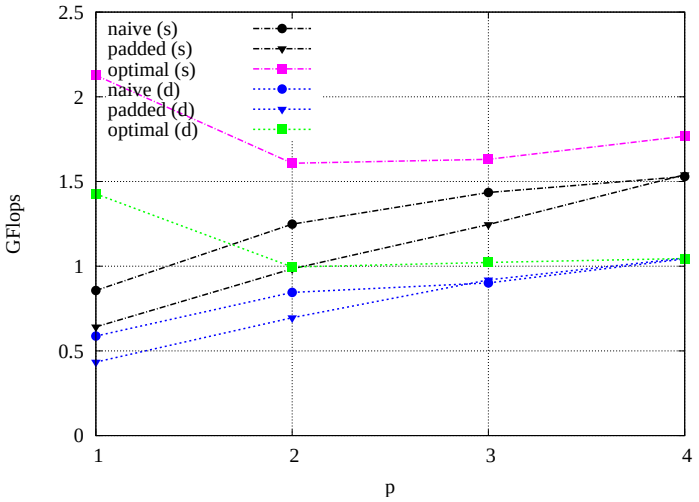


Figure 5.15: Performance of the ILU(1) backward/forward substitution steps (MKL)

During the evaluation of the performance only the useful work is taken into consideration for the actual performance, whilst for calculating the reference performance, the padded version is used.

The performance of ILU decompositions is close to the optimal performance as predicted by the underlying algebraic operations. Furthermore since gemm becomes the prevailing operation as more fill-in is allowed, near peak efficiency is approached.

In Fig. 5.15, the speed of the substitution steps is compared to predicted optimal performance for the MKL library. The flops rates are low due to the almost random access to a large number of blocks, typically outside of the L2 cache (the ILU(1) decomposition contains approximately 1600 blocks, so this is the case from p=2 onward). It is clear that in practice we will never use a matrix that fits in L2 cache, so these performances are not likely to be enhanced later on.

5.3 Efficient assembly

The evaluation of the integrals in the variational forms is done by numerical quadrature in the reference element:

$$\int_e a dV = \int_e a(\vec{\xi})\, |\mathbf{J}_e|\, d\xi \approx \sum_{q=1}^{N_\alpha} w_q\, a(\vec{\alpha}_q)\, |\mathbf{J}_e|_{\alpha_q} \tag{5.17}$$

Here $\vec{\xi}$ are the parametric coordinates, α_q the position of the i-th integration point, w_q the corresponding weight and $|\mathbf{J}_e|_{\alpha_q}$ the value of determinants of the mapping Jacobian in the integration point. For the element boundaries the corresponding entities are denoted β_q, v_q and $|\mathbf{J}_f|_{\beta_q}$.

The order of accuracy of the integration rule should be a function of the order p of the interpolation functions, the order of the flux functions c with respect to the state vector and the order k of the mapping of the reference to the physical element. In practice it is impossible to choose the exact order, since the inverse of the mapping Jacobian contains rational functions, and most non-trivial conservation laws feature rational or irrational functions of the state vector. Usually these influences are neglected, and hence the quadrature order is chosen to be $o = 2p + 1$. Some generic information on quadrature rules can be found in section B.4 for reference.

Volume term

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

$$\begin{aligned} CV_{im} &= \int_e \nabla\phi_i \cdot \vec{f}_m dV = \sum_{q=1}^{N_\alpha} w_q \left(|\mathbf{J}_e| \sum_{k=1}^{d} \sum_{u=1}^{d} \frac{\partial \phi_i}{\partial \xi^u} \frac{\partial \xi^u}{\partial x^k} f_m^k \right)_{\alpha_q} \\ DV_{im} &= \int_e \nabla\phi_i \cdot \vec{d}_n dV = \sum_{q=1}^{N_\alpha} w_q \left(|\mathbf{J}_e| \sum_{k=1}^{d} \sum_{u=1}^{d} \frac{\partial \phi_i}{\partial \xi^u} \frac{\partial \xi^u}{\partial x^k} d_m^k \right)_{\alpha_q} \end{aligned} \tag{5.18}$$

The efficient assembly of the residual proceeds in 3 steps

1. *Collocation*, *i.e.*the interpolation of the variables and their parametric gradients to the quadrature points:

$$\begin{aligned} \mathbf{u}^*{}_{qm} &= (u_m)_{\alpha_q} = \sum_{i=1}^{N_\phi} (\phi_i)_{\alpha_q} \cdot \mathbf{u}_{im} = \mathfrak{C}^*{}_{qi} \cdot \mathbf{u}_{im} \\ \mathfrak{g}^{*u}{}_{qm} &= \left(\frac{\partial u_m}{\partial \xi^u} \right)_{\alpha_q} = \sum_{i=1}^{N_\phi} \left(\frac{\partial \phi_i}{\partial \xi^u} \right)_{\alpha_q} \cdot \mathbf{u}_{im} = \mathfrak{G}^{*u}{}_{qi} \cdot \mathbf{u}_{im} \end{aligned} \tag{5.19}$$

2. Computation of the *parametric flux matrices* $\mathfrak{f}^{*u}$ and $\mathfrak{d}^{*u}$; this implies:
   - computation of the Cartesian components of the gradients:

$$\mathbf{g}^{*k}{}_{qm} = \sum_{u=1}^{d} \frac{\partial \xi^u}{\partial x^k} \mathfrak{g}^{*u}{}_{qm} \tag{5.20}$$

   - computation of the Cartesian components of the fluxes:

$$\begin{aligned} \mathbf{f}^{*k}{}_{qm} &= f_m^k(\mathbf{u}^*{}_{q*}) \\ \mathbf{d}^{*k}{}_{qm} &= d_m^k(\mathbf{u}^*{}_{q*}, \mathbf{g}^{*1}{}_{q*}, \ldots, \mathbf{g}^{*d}{}_{q*}) \end{aligned} \tag{5.21}$$

   - computation of the scaled parametric fluxes

$$\begin{aligned} \mathfrak{f}^{*u}{}_{qm} &= |\mathbf{J}_e| \sum_{k=1}^{d} \left(\frac{\partial \xi^u}{\partial x^k} \right)_{\alpha_q} \mathbf{f}^{*k}{}_{qm} \\ \mathfrak{d}^{*u}{}_{qm} &= |\mathbf{J}_e| \sum_{k=1}^{d} \left(\frac{\partial \xi^u}{\partial x^k} \right)_{\alpha_q} \mathbf{d}^{*k}{}_{qm} \end{aligned} \tag{5.22}$$

3. *Flux redistribution*, *i.e.*the premultiplication of the parametric fluxes $\mathfrak{f}^{*u}$ and $\mathfrak{d}^{*u}$ with the matrix $\mathfrak{R}^{*u}$ for $u = 1 \dots d$

As steps 1 and 3 are defined in the parametric space, they can be recast as a single matrix-matrix multiplication on all elements (of a given type and order), using the data structure defined in 5.2. The associated linearisations read

$$\begin{aligned}
\frac{\partial CV_{in}}{\partial \mathbf{u}_{jm}} &= \sum_{q=1}^{N_\alpha} \sum_{u=1}^{d} w_q \left(\frac{\partial \phi_i}{\partial \xi^u} \phi_j \right)_{\alpha_q} \cdot \left(|\mathbf{J}_e| \sum_{k=1}^{d} \frac{\partial \xi^u}{\partial x^k} \frac{\partial f_n^k}{\partial \tilde{u}_m} \right)_{\alpha_q} \\
&= \sum_{q=1}^{N_\alpha} \sum_{u=1}^{d} \mathfrak{C}_{q,ij}^{u} \cdot \kappa_{q,mn}^{u} \\
\frac{\partial DV_{im}}{\partial \mathbf{u}_{jn}} &= \sum_{q=1}^{N_\alpha} \sum_{u=1}^{d} \sum_{v=1}^{d} w_q \left(\frac{\partial \phi_i}{\partial \xi^u} \frac{\partial \phi_j}{\partial \xi^v} \right)_{\alpha_q} \cdot \left(|\mathbf{J}_e| \sum_{k=1}^{d} \sum_{l=1}^{d} \frac{\partial \xi^u}{\partial x^k} \bar{D}_{mn}^{kl} \frac{\partial \xi^v}{\partial x^l} \right)_{\alpha_q} \\
&= \sum_{q=1}^{N_\alpha} \sum_{u=1}^{d} \sum_{v=1}^{d} \mathfrak{D}_{q,ij}^{uv} \cdot \delta_{q,mn}^{uv}
\end{aligned} \tag{5.23}$$

Equation 5.23 means that, for each quadrature point q and variable combination (m, n), precomputed parametric convection $\mathfrak{C}_q^u$ and stiffness $\mathfrak{D}_q^{uv}$ contributions are added to the subblocks L_{mn}^{aa} of the diagonal block entry L^{aa}, after multiplication with respective weights $\kappa_{q,mn}^k$ and $\delta_{q,mn}^{uv}$. This is illustrated in Fig. 5.16, indicating quadrature point influence matrices and matrix additions in blue, and Jacobian matrix subblocks in black.

$$(\mathbf{L}_{aa})^{mn} \leftarrow \kappa_{q,mn}^u \cdot \mathfrak{C}_q^u + \delta_{q,mn}^{uv} \cdot \mathfrak{D}_q^{uv} + (\mathbf{L}_{aa})^{mn} \tag{5.24}$$

The number of operations involved in the scaled addition of the influence matrices provides a good approximation of the computational effort since relatively few weights need to be computed. Taking into account the sparsities f_c of c_q^u and f_d of d_q^{uv} respectively, the linearisation of both terms requires $2 \cdot (f_c + d \cdot f_d) \cdot d \cdot N_\alpha \cdot {N_v}^2 \cdot {N_\phi}^2$ operations. For an interpolation order of 4 on the tetrahedron, and $2p + 1$-accurate quadrature, this amounts to 38.10^6 floating point operations per element.

The direct addition of the quadrature point contributions the local stiffness matrix to the subblocks of the Jacobian matrix, leads to the addition of N_ϕ rows of size N_ϕ, since the none of the blocks $\mathbf{L}_{aa}^{mn}$ are stored contiguously in memory. To circumvent this problem all quadrature point contributions can be preassembled in an intermediate matrix (shown in red in Fig. 5.17) that vertically aligns all N_v^2 subblocks.

In this structure all of the subblocks are stored contiguously in memory. After the assembly the subblocks in the intermediate structures are added to the Jacobian matrix subblocks (green arrows). The overhead due the transfer from the temporary matrix to the Jacobian matrix is low since it has to be done once for any combination of variables, i.e. once for every $N_q \cdot (f_c + f_d d) \cdot d$ quadrature steps (blue arrows).

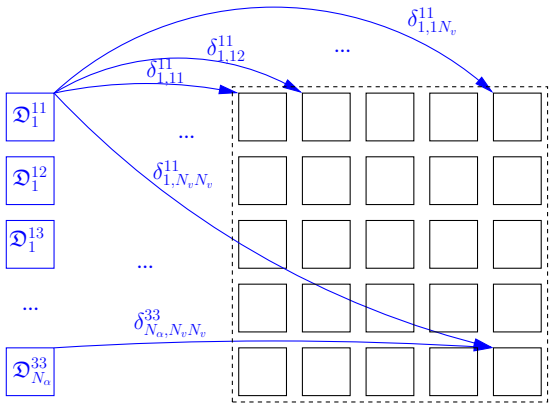


Figure 5.16: Linearisation of volume terms CV and DV. For each combination of variables and quadrature point, this naive version multiplies the influence matrices by the appropriate weights and adds them directly to the corresponding sub-block in the final matrix. As this sub-block is not contiguous in memory, suboptimal performance is obtained.

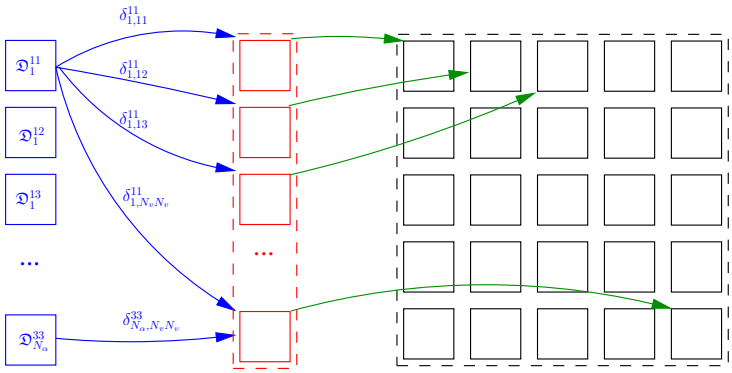


Figure 5.17: Linearisation of volume terms CV and DV. A first enhancement of the performance is obtained by assembling in an intermediate structure, where the subblocks are stored contiguously in memory. The subsequent copy to the final structure is only a fraction of the total cost.

To take advantage of the large differences in speed between odd and even sized vectors, the subblocks are further unrolled as vectors which are then padded to the next multiple of 4[5] as shown in Fig. 5.18.

Fig. 5.19 compares the different versions for single (s) in the figure) and double precision (d), to the optimal speed as found for the *axpy* operation. These counts are conservative in the sense that they only include the *axpy* operations but not the computation of the weights $\kappa^u_{q,mn}$ and $\delta^{uv}_{q,mn}$ on the one hand, and only the actual useful work for the padded version on the other.

[5]This number should be architecture dependent, but is taken 4 here to correspond to the measurements of algebraic primitives performance.

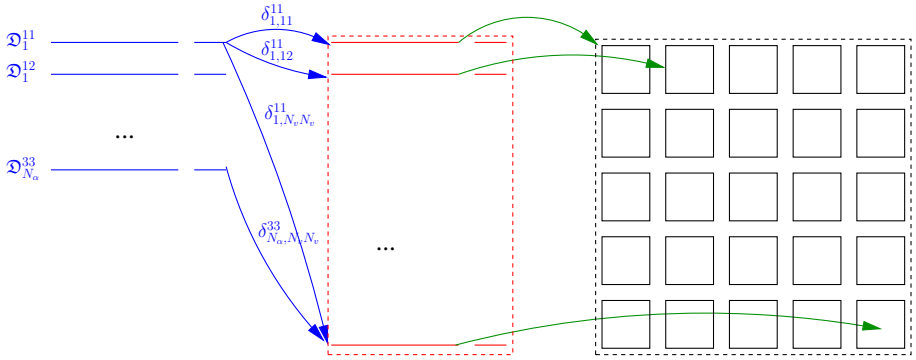


Figure 5.18: Linearisation of volume terms CV and DV. By reinterpreting the subblocks in the intermediate structure as vectors and padding them to the appropriate size, the assembly is further accelerated

The assembly in single precision is up to 30% faster than the same operation in double precision, except for the naive implementation. Padding is necessary to maintain the advantage of single precision operations with respect to double for $p = 4$; in double precision, the performance is nearly the same, as can be expected from the measurements shown in Fig. 5.7.

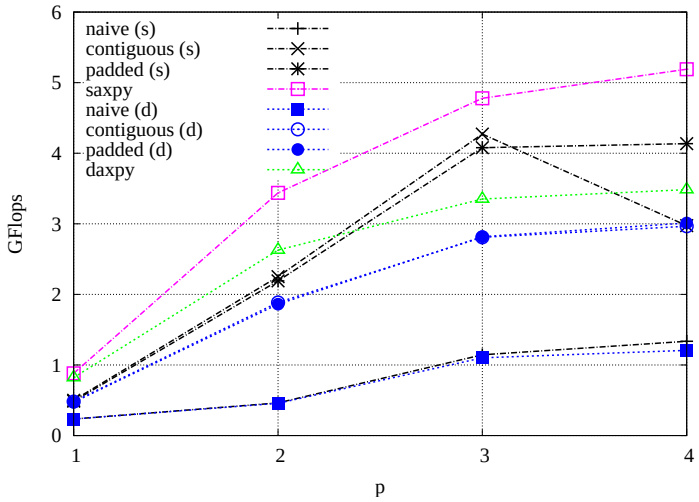


Figure 5.19: Volume term assembly efficiency comparing naive, contiguous and padded versions with respect to the speed of vector additions (axpy) in single (s) and double (d) precision (MKL)

Interface terms

Suppose an interface located between elements a and b. The assembly of the interface contribution to the residual and the tangent matrix is done in a lo-

cal frame of reference. In this frame local shape functions ψ_k, which are the

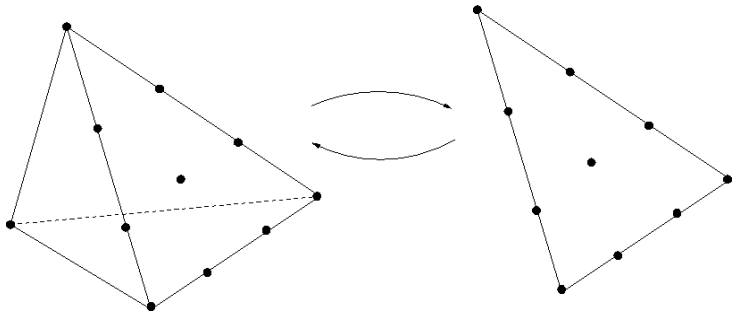


Figure 5.20: Local frame of reference for the interface

restrictions of ϕ_k, and quadrature rules (β_q, v_q) are defined.

In the following sections the linearisation cost is computed per element and not per interface, to allow a direct comparison of the operation counts for volume and interface terms.

Convective flux CI and diagonal penalty term DP

The convective flux term CI^a_{im} after quadrature is given by

$$CI^a_{im} = \sum_{q=1}^{N_\beta} v_q \left(\phi^a_i \, \mathcal{H}_m \left(u^a, u^b, \vec{n} \right) \, |\mathbf{J}_f| \right)_{\beta_q} \tag{5.25}$$

and consequently linearised as

$$\begin{aligned} \frac{\partial CI^a_{im}}{\partial \mathbf{u}^a_{jn}} &= \sum_{q=1}^{N_\beta} v_q \left(\phi^a_i \phi^a_j \right)_{\beta_q} \cdot \left(|\mathbf{J}_f| \, \frac{\partial \mathcal{H}_m}{\partial \tilde{u}^a_n} \right)_{\beta_q} = \sum_{q=1}^{N_\beta} \mathfrak{m}^{aa}_{q,ij} \cdot \rho_{q,mn} \\ \frac{\partial CI^a_{im}}{\partial \mathbf{u}^b_{jn}} &= \sum_{q=1}^{N_\beta} v_q \left(\phi^a_i \phi^b_j \right)_{\beta_q} \cdot \left(|\mathbf{J}_f| \, \frac{\partial \mathcal{H}_m}{\partial \tilde{u}^b_n} \right)_{\beta_q} = \sum_{q=1}^{N_\beta} \mathfrak{m}^{ab}_{q,ij} \cdot \rho_{q,mn} \end{aligned} \tag{5.26}$$

This linearisation requires $16 \, N_\beta \, {N_v}^2 \, {N_\psi}^2$ operations per *element* (not per face), not counting the effort of computing the Jacobian of $\mathcal{H}$.

The penalty parameter σ is a scalar, hence the linearisation of the penalty term DP only contributes to the diagonal subblocks $(\mathbf{L}_{aa})^{mm}$ and $(\mathbf{L}_{ab})^{mm}$:

$$\begin{aligned} DP^a_{im} &= \sum_{q=1}^{N_\beta} v_q\, \sigma \left(\phi^a_{im} \left(u^b_m - u^a_m\right) |\mathbf{J}_f|\right)_{\beta_q} \\ &= \sum_{q=1}^{N_\beta} v_q\, \sigma \left(\phi^a_i \left(\mathbf{u}^b_{jm}\phi^b_j - \mathbf{u}^a_{jm}\phi^a_j\right) |\mathbf{J}_f|\right)_{\beta_q} \\ \frac{\partial DP^a_{im}}{\partial \mathbf{u}^a_{jn}} &= -\sum_{q=1}^{N_\beta} v_q \left(\phi^a_i\phi^a_j\right)_{\beta_q} \cdot \left(|\mathbf{J}_f|\, \sigma\, \delta_{mn}\right)_{\beta_q} = \sum_{q=1}^{N_\beta} \mathfrak{m}^{aa}_{q,ij} \cdot \rho_{q,mn} \\ \frac{\partial DP^a_{im}}{\partial \mathbf{u}^b_{jn}} &= \sum_{q=1}^{N_\beta} v_q \left(\phi^a_i\phi^b_j\right)_{\beta_q} \cdot \left(|\mathbf{J}_f|\, \sigma\, \delta_{mn}\right)_{\beta_q} = \sum_{q=1}^{N_\beta} \mathfrak{m}^{ab}_{q,ij} \cdot \rho_{q,mn} \end{aligned} \tag{5.27}$$

resulting in an operation count of $16\, N_\beta N_v {N_\psi}^2$, again per element. Both terms CI and DP have the same influence matrices and are hence treated simultaneously. This influence matrices have very high sparsity, leading to a very inefficient quadrature if we directly add the entries to the blocks in the Jacobian matrix, as illustrated in Fig. 5.21. In the frame of reference associated to the

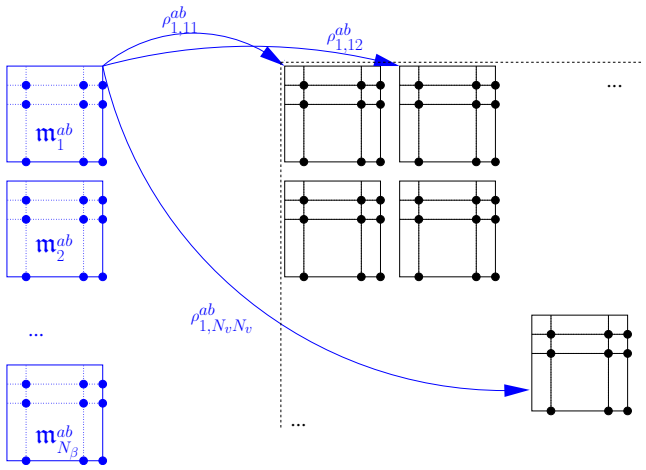


Figure 5.21: Linearisation of interface terms CI and DP. For each combination of variables and quadrature point, this naive version multiplies the influence matrices by the appropriate weights before adding it to the subblocks in the tangent matrix. Since only the points on the face are involved, both in terms of the fluxes as well as in the weight functions, the contributions need to be added point by point.

face, the shape functions of the face nodes of a and b coincide ($\psi^a = \psi^b = \psi$),

and the following antisymmetry can be found for both terms CI and DP:

$$\frac{\partial CI^a_{im}}{\partial \mathbf{u}^b_{jn}} = -\frac{\partial CI^b_{im}}{\partial \mathbf{u}^b_{jn}} = \sum_{q=1}^{N_\beta} v_q \left(\psi_i \psi_j\right)_{\beta_q} \cdot \left(|\mathbf{J}_f| \frac{\partial \mathcal{H}_m}{\partial \tilde{u}^b_n} \right)_{\beta_q} = \sum_{q=1}^{N_\beta} \mathfrak{m}_{q,ij} \rho_{q,mn}$$
$$\frac{\partial DP^a_{im}}{\partial \mathbf{u}^a_{jn}} = -\frac{\partial DP^b_{im}}{\partial \mathbf{u}^a_{jn}} = \sum_{q=1}^{N_\beta} v_q \left(\psi_i \psi_j\right)_{\beta_q} \cdot \left(|\mathbf{J}_f| \, \sigma \, \delta_{mn} \right)_{\beta_q} = \sum_{q=1}^{N_\beta} \mathfrak{m}_{q,ij} \pi_{q,mn} \tag{5.28}$$

If the assembly is performed in the frame of reference of the face first, contiguous matrix additions can be used. Thereto an intermediate structure is defined that vertically aligns N_v^2 blocks of size $N_\psi \times N_\psi$. The "explosion" of these blocks to the tangential blocks is done after all contributions have been added. Again, the procedure can further be optimised by unrolling the matri-

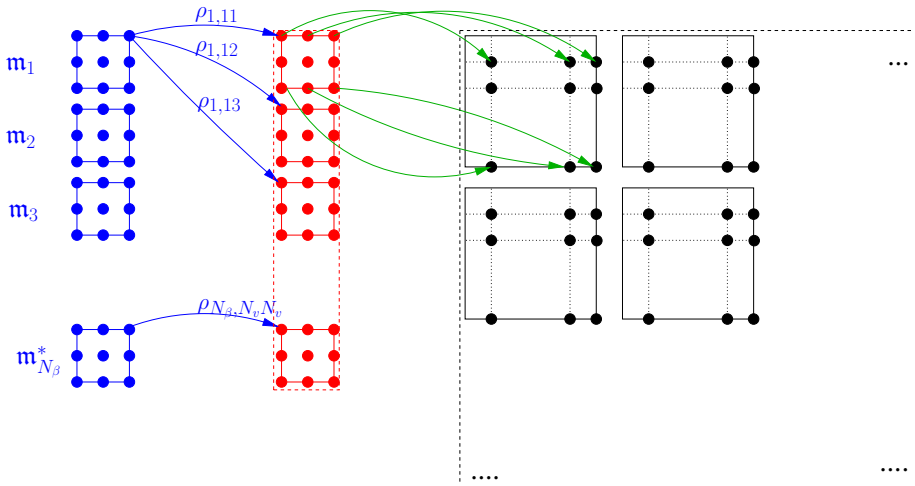


Figure 5.22: Linearisation of interface terms CI and DP. As for the volume term, the first enhancement of the performance is obtained by assembling in an intermediate structure, where the subblocks are stored contiguously in memory.

ces as padded vectors, as illustrated in Fig. 5.23.

Diffusive flux DI and transpose penalty term DT

DI^a_{im} and DT^a_{im} can be expanded as:

$$DI^a_{im} = \sum_{j=1}^{N_\phi} \sum_{q=1}^{N_\beta} \frac{v_q}{2} \left(\sum_{k=1}^{d} \sum_{l=1}^{d} \phi^a_i n^k_a \bar{D}^{kl}_{mn} \left(\frac{\partial \phi^a_j}{\partial x^l} \mathbf{u}^a_{jn} + \frac{\partial \phi^b_j}{\partial x^l} \mathbf{u}^b_{jn} \right) |\mathbf{J}_f| \right)_{\beta_q}$$
$$DT^a_{im} = \theta \sum_{j=1}^{N_\phi} \sum_{q=1}^{N_\beta} \frac{v_q}{2} \left(\sum_{k=1}^{d} \sum_{l=1}^{d} \left(\phi^a_j \mathbf{u}^a_{jn} n^k_a + \phi^b_j \mathbf{u}^b_{jn} n^k_b \right) \bar{D}^{kl}_{nm} \frac{\partial \phi^a_i}{\partial x^l} |\mathbf{J}_f| \right)_{\beta_q} \tag{5.29}$$

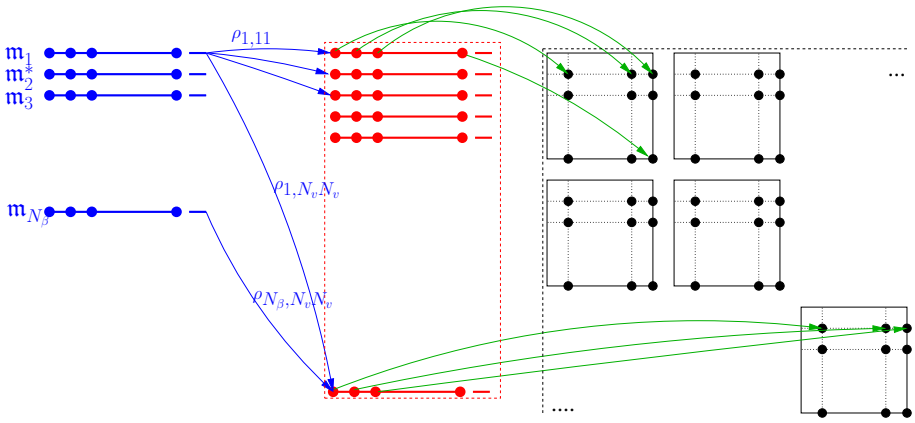


Figure 5.23: Linearisation of interface terms CI and DP. By reinterpreting the subblocks in the intermediate structure as vectors and padding them to the appropriate size, the assembly is further accelerated.

and are linearised as (bearing in mind that $n_b^k = -n_a^k$)

$$\begin{aligned}
\frac{\partial DT_{jn}^a}{\partial \mathbf{u}_{im}^a} &= \theta \frac{\partial DI_{im}^a}{\partial \mathbf{u}_{jn}^a} = \frac{\theta}{2} \sum_{q=1}^{N_\beta} \sum_{u=1}^{d} v_q \left(\phi_i^a \frac{\partial \phi_j^a}{\partial \xi^u} \right)_{\beta_q} \cdot \left(|\mathbf{J}_f| \sum_{k=1}^{d} \sum_{l=1}^{d} n_a^k \bar{D}_{mn}^{kl} \left(\frac{\partial \xi^u}{\partial x^l} \right)_a \right)_{\beta_q} \\
&= \theta \sum_{q=1}^{N_\beta} \sum_{u=1}^{d} \mathfrak{d}_{q,ji}^{aa,u} \cdot \gamma_{q,mn}^{u,a} \\
\frac{\partial DT_{jn}^b}{\partial \mathbf{u}_{im}^a} &= \theta \frac{\partial DI_{im}^a}{\partial \mathbf{u}_{jn}^b} = \frac{\theta}{2} \sum_{q=1}^{N_\beta} \sum_{u=1}^{d} v_q \left(\phi_i^a \frac{\partial \phi_j^b}{\partial \xi^u} \right)_{\beta_q} \cdot \left(|\mathbf{J}_f| \sum_{k=1}^{d} \sum_{l=1}^{d} n_a^k \bar{D}_{mn}^{kl} \left(\frac{\partial \xi^u}{\partial x^l} \right)_b \right)_{\beta_q} \\
&= \theta \sum_{q=1}^{N_\beta} \sum_{u=1}^{d} \mathfrak{d}_{q,ji}^{ab,u} \cdot \gamma_{q,mn}^{u,b}
\end{aligned} \tag{5.30}$$

The elementary contribution matrices $\mathfrak{d}_q^{ab,u}$ consist of rows corresponding to the shape functions which are non-zero on the boundary. As these matrices are parametric, they only need to be stored for every possible relative orientation of two adjacent elements; of course, $\mathfrak{d}_q^{ab,u}$ are stored in rectangular contiguous format of size $N_\psi \times N_\phi$, in which only non-trivial rows remain.

The direct terms DI are then linearised using direct row additions of $\mathfrak{d}_q^{ab,u}$, with factors $\gamma_{q,mn}^{u,b}$ and $\gamma_{q,mn}^{u,a}$. In practice this boils down to row additions to the submatrices $\mathbf{L}_{mn}^{ab}$. As expected the transpose terms add the transpose of the parametric matrices $\mathfrak{d}_q^{ab,u}$ to $\mathbf{L}_{nm}^{ba}$ with transpose weights $\theta\gamma_{mn}^{u,a}$ and $\theta\gamma_{mn}^{u,b}$. To keep Fig. 5.24 simple, we illustrate this process for $a = b$ and $m = 1, n = 2$. Counting only the scaled addition of influence vectors, the operation count per *element* for the computation of the interface linearisation amounts to $8\, d\, N_\beta\, N_\phi\, N_\psi\, f_d N_v^2$ per element.

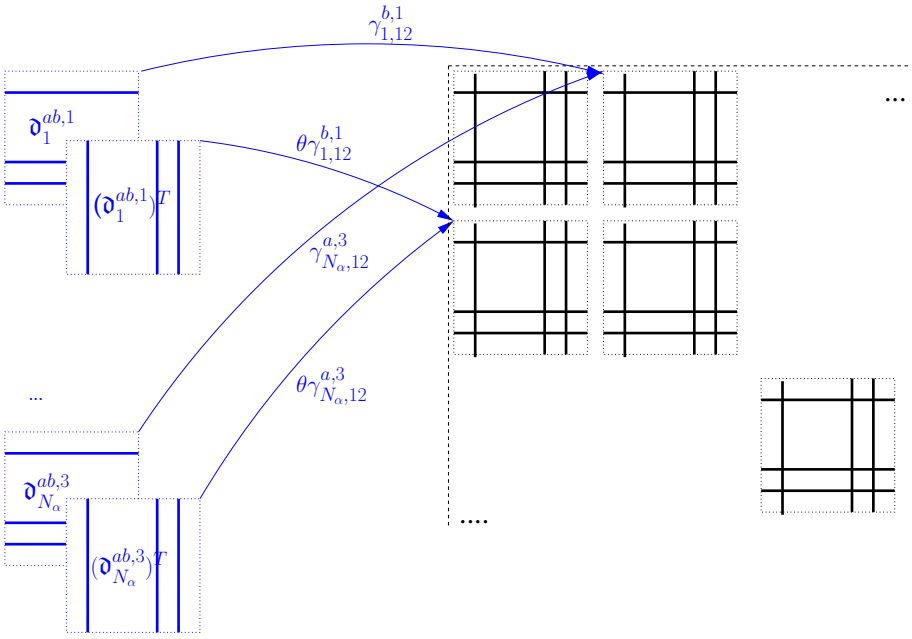


Figure 5.24: Linearisation of interface terms DI and DT. For each combination of variables and quadrature point, this naive version multiplies the influence matrices by the appropriate weights and adds it directly to the corresponding sub-block in the final matrix. Since for DI and DT the weighting resp. the jump operator only involve boundary interpolation points, the additions need to be done row resp. columnwise. This addition is not obviously not contiguous in memory, so suboptimal performance is obtained.

A naive implementation, illustrated in Fig. 5.24, uses row additions of size N_ϕ directly to the Jacobian matrix. This is not only very inefficient due to the small size of the vector (N_ϕ), but furthermore but this cannot exploit the symmetry of the terms during quadrature.

Now the condensed storage of $\mathfrak{d}^{ab,u}_q$ is again exploited by assembling in intermediate structures, composed of vertically aligned matrices in $\mathbb{R}^{N_\psi \times N_\phi}$, where we can add each scaled $\mathfrak{b}^{ab,u}_q$ in a single axpy step. This procedure is illustrated in Fig. 5.25, for the case $a = b$, the redistribution is only shown for $m = 1, n = 2$ as in the previous illustration. The main operation during assembly is then a far more efficient vector addition of size $N_\psi \times N_\phi$; the intermediate structure also allows to exploit the inherent (anti)symmetry in the terms DI and DT, such that only one of these two terms needs to be assembled. As before we can further optimise the operation by unrolling $\mathfrak{b}^{ab,u}_q$ to a vector, which is lengthened to a multiple of 4 (see Fig. 5.26).

Combined efficiency of the interface linearisation

The resulting efficiency for the combined interface terms is shown in Fig. 5.27. An efficiency of 60% is obtained with respect to the ideal performance, which

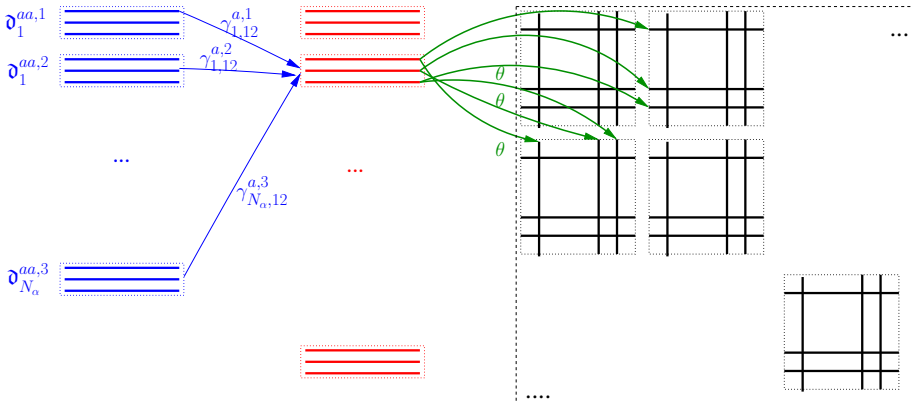


Figure 5.25: Linearisation of interface terms DI and DT. As always a first enhancement can be obtained by assembling in contiguous intermediate structures for the row/column contributions. It is particularly important that in this way the symmetries in the terms DI and DT can be exploited, so that in effect only one needs to be linearised.

is determined by the appropriate combination of $axpy(N_\phi \times N_\psi)$ and $axpy(N_\psi \times N_\psi)$, which are the dominant operations; this estimate takes the symmetry into account. The efficiency is lower than for the volume term since on the one hand the relative cost of the computation of the weights is larger; furthermore 4 block entries, namely L^{aa}, L^{ab}, L^{ba} and L^{bb} need to be treated at once, leading to a larger impact of the memory access. For double precision the gain in efficiency only slightly improves on the gain corresponding to the application of the symmetry; if padding is not used, the same applies to single precision.

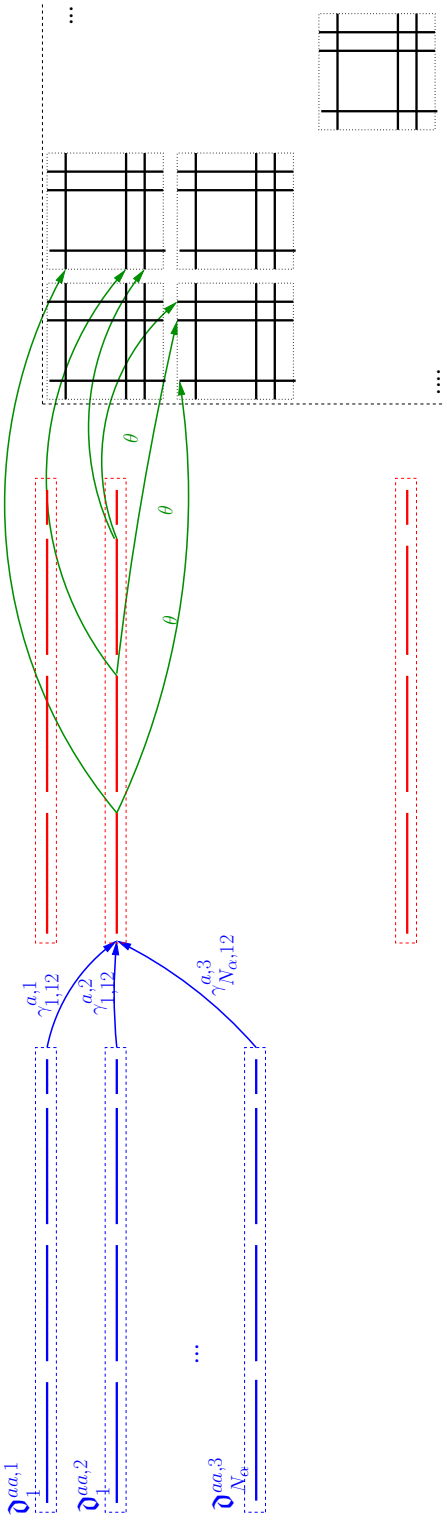


Figure 5.26: Linearisation of interface terms DI and DT. As always the performance can be enhanced by padding the intermediate structure.

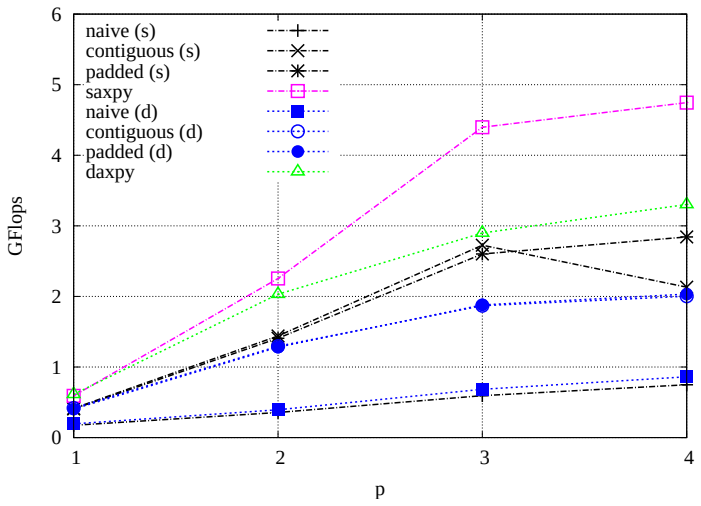


Figure 5.27: Performance of the interface term linearisation counting based on the total count of axpy operations.

Evolution of the global assembly time

The evolution of the assembly time of the Jacobian per element as a function of order is shown in Fig. 5.28 for the naive, the contiguous and the padded implementations.

An additional black curve shows the evolution of the number of operations needed for the quadrature of interface, volume and boundary terms. For recapitulation purposes, we recall the number of operations needed for a correct linearisation in table 5.1.

	operation count
CV	$2\ f_c\ d\ N_\alpha\ {N_v}^2\ {N_\phi}^2$
DV	$2\ f_c\ d^2\ N_\alpha\ {N_v}^2\ {N_\phi}^2$
DI and DT	$8\ d\ N_\beta\ N_\phi\ N_\psi\ f_d N_v^2$
CI	$16\ N_\beta\ {N_v}^2\ {N_\psi}^2$
DP	$16\ N_\beta\ N_v\ {N_\psi}^2$
Mass	$10\ {N_v}^3\ {N_\phi}^3$

Table 5.1: Floating point operation counts for the linearisation of each of the terms as a function of discretisation parameters and dimensions

From the graphs in Fig. 5.28 we see that due to the optimisation the assembly time is reduced by a factor ranging from 3 to 5 both in single as well as in double precision. The padded version thereby consistently provides at least 50% more efficient quadrature in single precision. However the most conspicuous result is that the increased efficiency mitigates the sharp increase of work

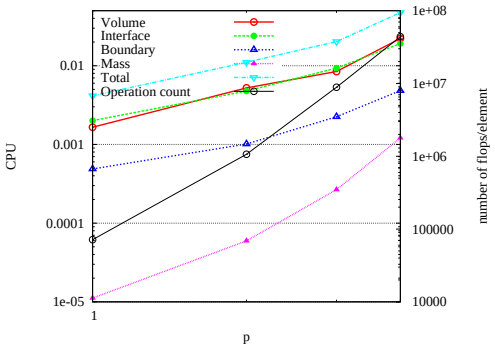
as a function of interpolation order for any version. This will probably only hold for moderate orders, as computational time will again scale with operational complexity as soon as the operations saturate in terms of efficiency.

5.4 Conclusions

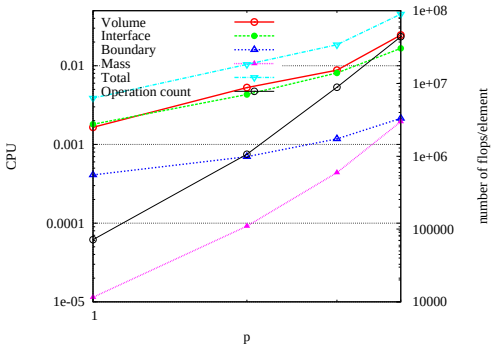
In this chapter the efficient implementation of DGM is discussed. Starting from the efficiency measurements for algebraic primitives, optimised data structures are proposed and assembly is optimised. The following conclusions can be drawn:

- the reference performances of algebraic primitives such as axpy, gemv, gemm and gesv have shown the advantages of using single precision data, and the impact of data size, leading to significant variations of computational efficiency. The latter should be exploited by extending or *padding* data to the optimal size, since the additional effort is more than compensated by the enhanced execution speed;
- preconditioners, which are very memory-consuming but are not critical for the solution quality, can be stored in single precision. This leads not only to decreased memory requirements, but also to increased computational efficiency - twice that of double precision data;
- the data locality of DGM allows for a very efficient yet extremely flexible data structure for blocked matrices and solution vectors, with minimal indexing overhead. The performance of algebraic operations with these data structures is close to optimal;
- the optimisation of assembly routines can amortise the increased cost of higher order up to the point where the algebraic primitives used attain maximal performance. Currently this transition is located around $p = 4$ for 3D computations. This goes to show that the computational efficiency of a discretisation is not only a question of operation count.

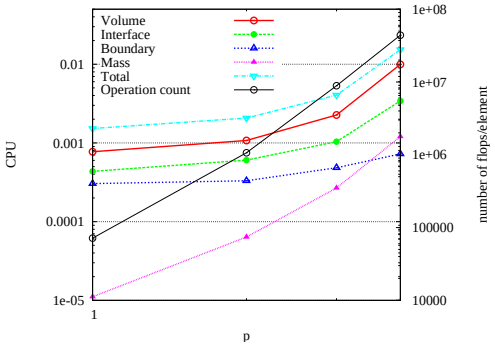
Predicting how the processor core evolution will modify this conclusions is quite difficult. Any further increase in cache size will in any case lead to a larger range of optimal performance for the axpy operation to larger vectors, thereby enhancing the potential for optimisation of Jacobian assembly. On the other hand, the increase in floating point register size will proportionally increase the maximal CPU throughput, but then again make cache effects more important. Ideally both should evolve together. Much will depend also of the memory organisation, in particular the potential use of shared cache and RAM within multicore processors.



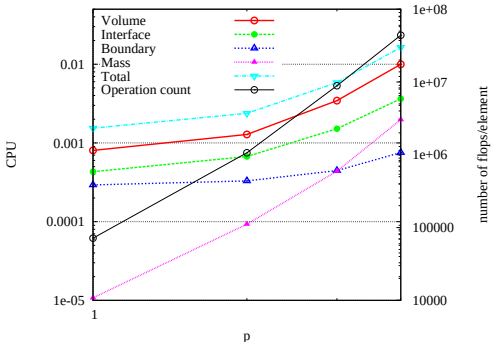
(a) Naive version - single precision



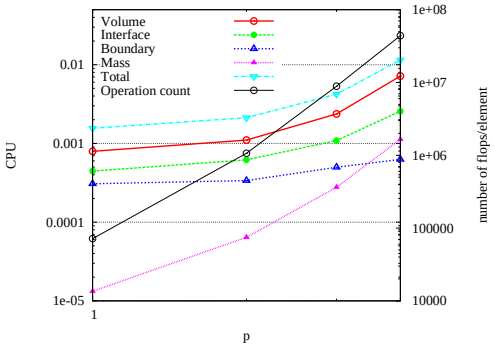
(b) Naive version - double precision



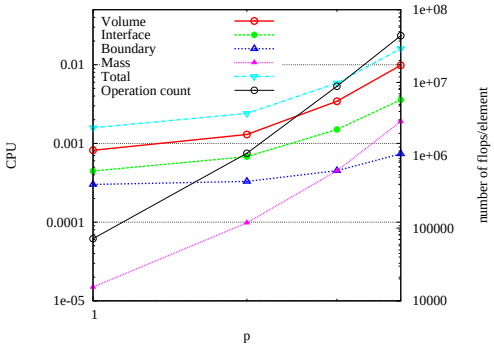
(c) Contiguous version - single precision



(d) Contiguous version - double precision



(e) Padded version - single precision



(f) Padded version - double precision

Figure 5.28: Evolution of total assembly time in comparison to computational complexity (MKL)

CHAPTER 6

noFUDGE: A FIRST INDUSTRIAL APPLICATION

The first industrial application of Argo concerns the flow around a low pressure turbine blade cascade. The computations were conducted on JuGENE, the BlueGene/P machine of the *Forschungszentrum Jülich*, where a grant of

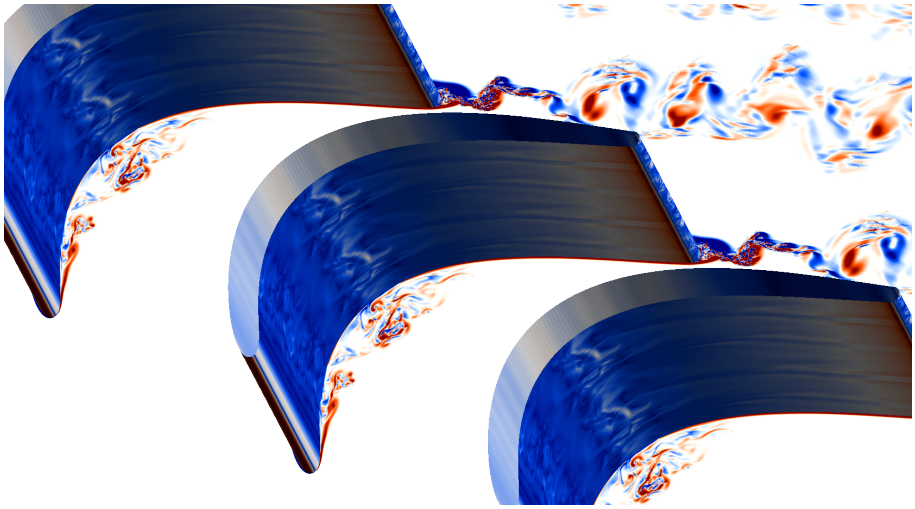


Figure 6.1: 4^{th} order Discontinuous Galerkin / Direct Numerical Simulation of the flow around a low pressure turbine cascade - skin friction on the blade and spanwise vorticity on the periodic boundary

2.4 million core hours was awarded to the project *noFUDGE (Flow Unsteadiness simulated with the Discontinuous Galerkin method)*. noFUDGE is the first industrial pilot project of PRACE (1). The geometry was provided by Snecma, whilst experimental reference data was provided by the von Karman Institute. The results were presented at the ASME Turbo Expo 2012 (29). This is joint work with Corentin Carton de Wiart.

6.1 Description of the flow

The working fluid is air at non-atmospheric conditions. The blade is considered to be adiabatic. At the inlet, total pressure p_{t1} and temperature T_t are imposed as well as the flow direction.The isentropic Reynolds number at the outlet, based on pressure ratio and axial chord, is $Re_{is} = 85000$ whilst the isentropic exit Mach number is $M_{2is} = 0.6$. These are typical cruise conditions for the mid-section of low pressure turbine blades of a small gas turbine engine.

The flow is illustrated in Fig. 6.1. On the suction side, the flow is characterized by laminar separation followed by turbulent reattachment. The thick trailing edge induces furthermore important vortex shedding. On the pressure side, a slow recirculation bubble is caught in the cavity to virtually provide a thick blade, required for a smooth flow acceleration. This recirculation bubble periodically bursts, is subsequently stretched by the high speed flow on the aft section of the pressure side and then interacts with the vortex shedding at trailing edge. The considerable disparity of the time scales in these different phenomena result in very complex turbulent features. The presence of laminar separation and large scale unsteadiness, as well as the complex interactions between the different flow features, make this flow impossible to compute with URANS methods.

6.2 Computational setup

A 4^{th} order accurate discontinuous Galerkin method is used to compute the flow. The results are compared to URANS (Spalart-Allmaras) and LES (WALE) computations performed with the finite volume version of Argo. The latter is run with a central, kinetic energy conserving flux blended with 5% of the standard Roe solver for stability.

The computational domain consists of a prismatic blade with a spanwise periodicity of 30% of the axial chord c. Considering the size of the computed turbulent structures, this distance is sufficiently large to decouple both periodic boundaries. A single blade was taken into account, thereby assuming that the turbulent structures on two consecutive blades do not interact. The meshes were constructed by extrusion from a bi-dimensional mesh. The DGM mesh is composed of 133500 hexahedra and 788500 wedges. The solution is interpolated with third order polynomials leading to a fourth order accurate solution. The mesh elements are curved and described with quadratic polynomials, resulting in a third-order accurate description of the geometry.

The grid specifications have been determined during a number of preliminary two dimensional runs such as to ensure sufficient resolution near critical

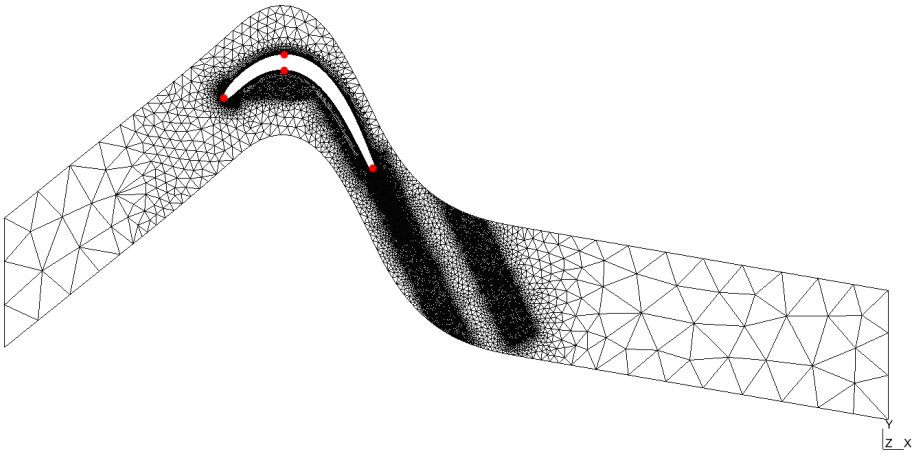


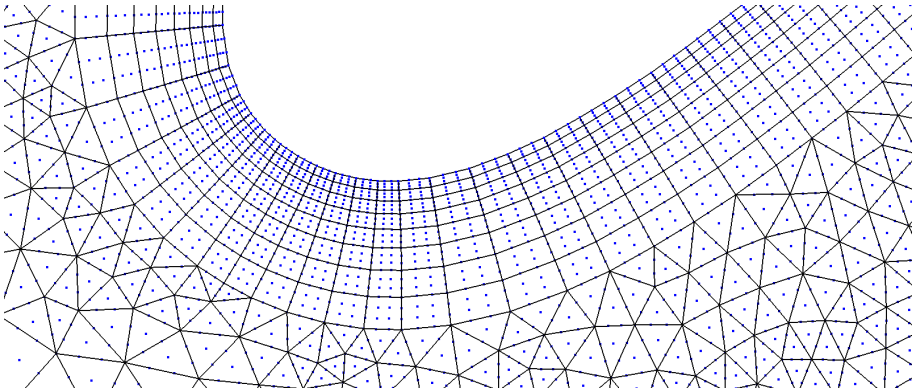
Figure 6.2: Mesh for the dgm computations showing the refinement zones. Black dots indicate the locations used in table 6.1.

regions as shown in Fig. 6.2; the extrusion mesh size is chosen such that mesh isotropy is obtained at midchord and trailing edge of the suction side. The continuity of the flow field and the very near-continuity of the vorticity and skin-friction fields resulting from the three dimensional computations corroborate the adequacy of this resolution.

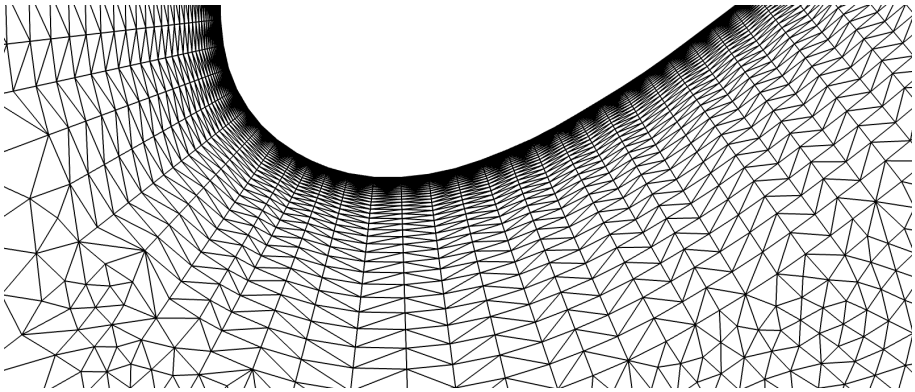
To compare the mesh resolution between the FVM and the DGM computations we should consider that the DGM solver has four control points per cell, such that each cell is can be considered to be further subdivided in three segments in all directions. The mesh used for the FVM computations then has essentially the same specifications. The major difference is the size of the first layer of the boundary layer mesh. The first high-order control point for DGM off the boundary is located at roughly seven times the grid spacing used for the FVM computations, corresponding to a y^+ of about 3 to 6.

The corrected mesh size on the surface of the blade is twice more dense in the DGM case, as a first grid refinement was performed based on discontinuities detected during preliminary computations on a coarser mesh. The total number of degrees of freedom per variable for the FVM mesh is $8.5M$ and $15M$ for the DGM. The table 6.1 summarizes the mesh size at the wall at four locations of the blade (given on Fig. 6.2). It should be noted that the grid size used for the DGM based on the high order point location is only slightly lower than those used for the FVM computation[1]. Yet, sufficient resolution for DNS is obtained whereas it is only sufficient for LES with FVM. The DGM and FVM meshes at the leading and trailing edge are shown in figures 6.3 and 6.4.

[1]the non-dimensional grid sizes specified in table 6.1 are based on the time-averaged friction for each of the computations, and thus depend on the computation.



(a) DGM



(b) FVM

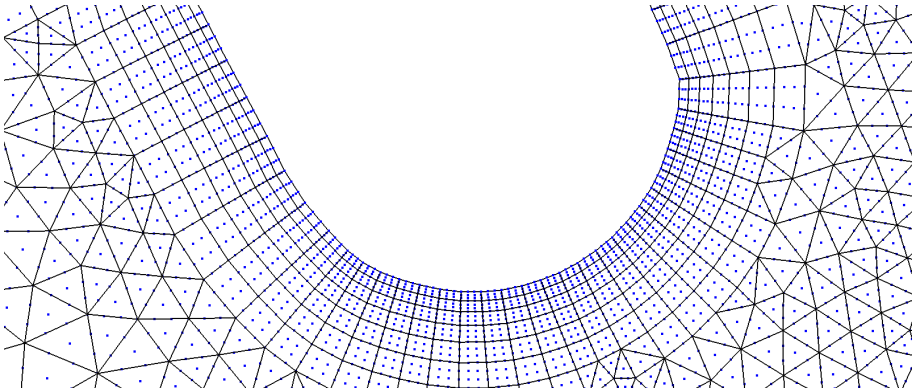
Figure 6.3: Mesh at leading edge. DGM (up) and FVM (down).

	FVM		**DGM**		
	$x^+ = z^+$	y^+	x^+	z^+	y^+
Leading edge	35	0.5	15	30	3
Pressure side	5	0.1	3	3	0.3
Suction side	50	0.3	15	15	1.5
Trailing edge edge	10	0.3	2	10	1

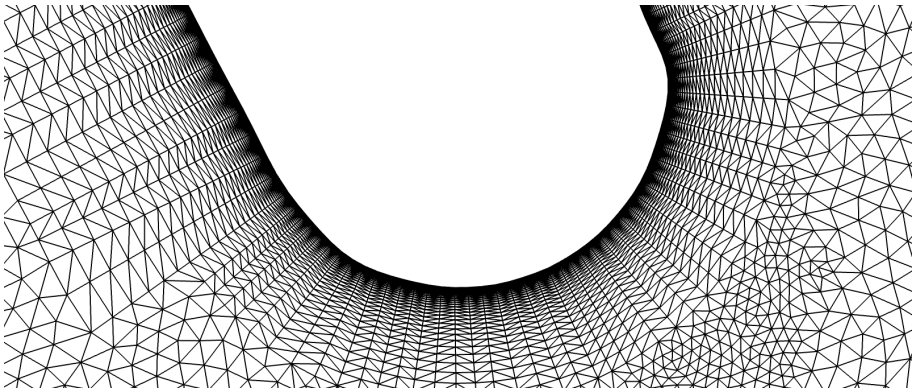
Table 6.1: Comparison between DGM and FVM mesh resolution. Non-dimensionalisation of the grid sizes are based on the time-averaged computed wall friction.

6.3 Comparison of computed flow fields

The computed instantaneous vorticity and entropy fields are shown in Fig. 6.5 and Fig. 6.6. The results are similar but, as expected, the DNS computation



(a) DGM

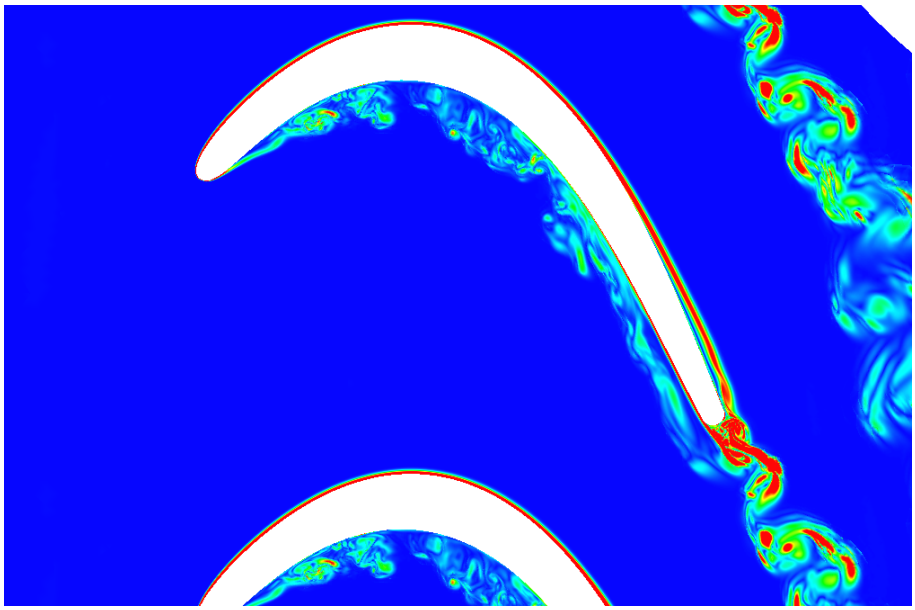


(b) FVM

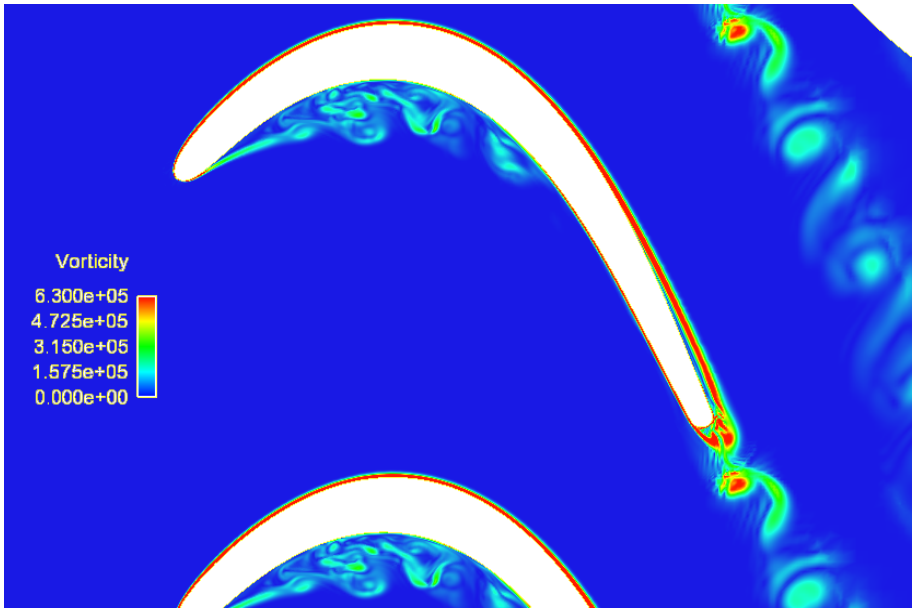
Figure 6.4: mesh at trailing edge. DGM (up) and fvm (down).

captures more scales and thus is closer to the physics. In general the scales captured by the DGM computations are much smaller, and that the vorticity is more intense. Moreover the structures leaving the trailing edge seem to be much more quickly dissipated for the FVM LES computation. This difference in resolution is presumably due to the coarseness of the mesh in the wake combined to an over-dissipation of the LES model. Indeed, the level of dissipation of the LES model is very difficult to calibrate for this kind of industrial application.

Note that absence of turbulent structures at the downstream pressure side in the LES computation does not mean that the physics captured by both methods are different. The instantaneous flow conditions shown for DGM corresponds to the periodic ejection or *bursting* of the pressure side recirculation region, whilst the FVM visualisations shows the bubble building up. In order to reduce the number of figures, it was decided to alternate the conditions between both computations as the difference in resolution remains adequately illustrated.



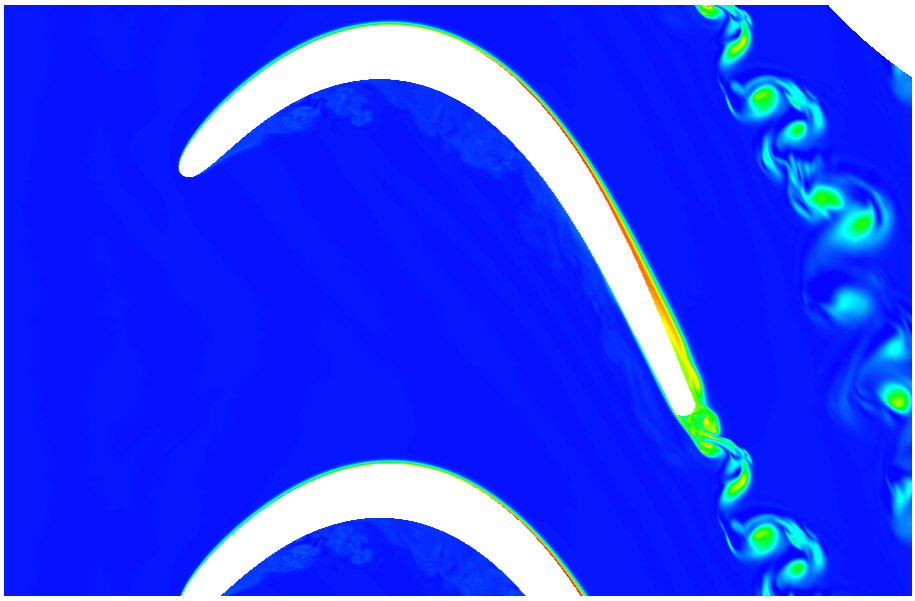
(a) DGM/DNS



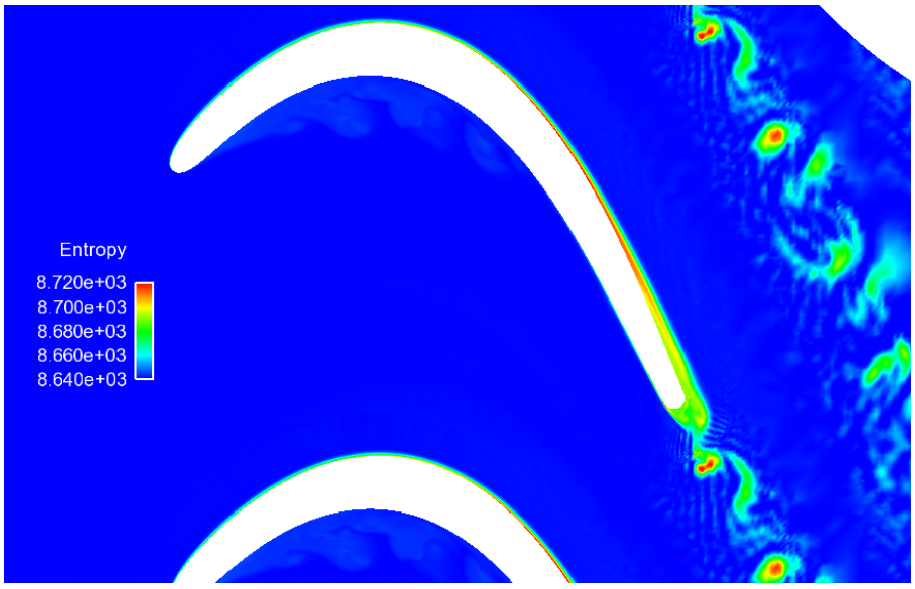
(b) FVM/LES

Figure 6.5: noFUDGE - transitional flow around an LP turbine blade. Computed instantaneous vorticity fields.

The entropy distribution is very similar but some oscillations in the wake can be observed for the LES computation. This is due to the instability of



(a) DGM/DNS

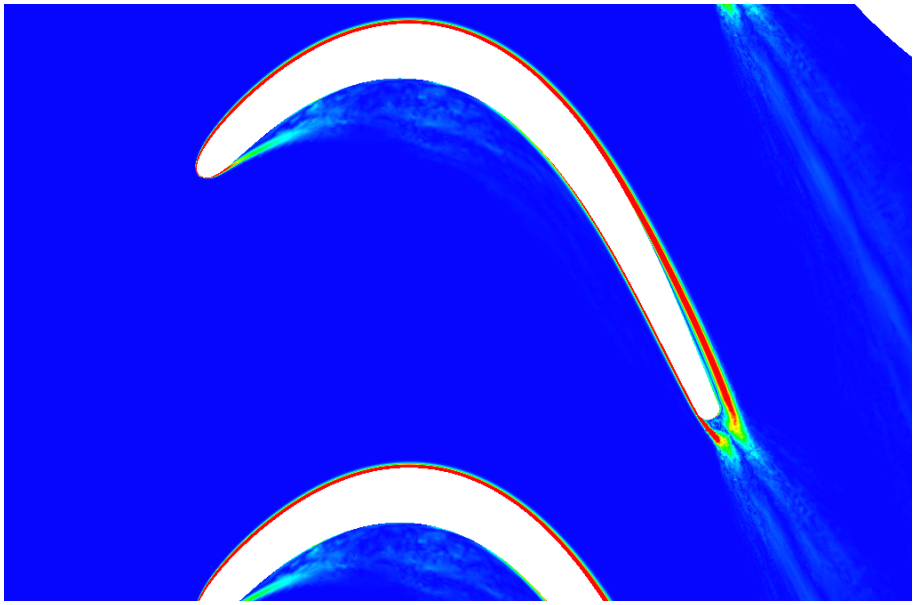


(b) FVM/LES

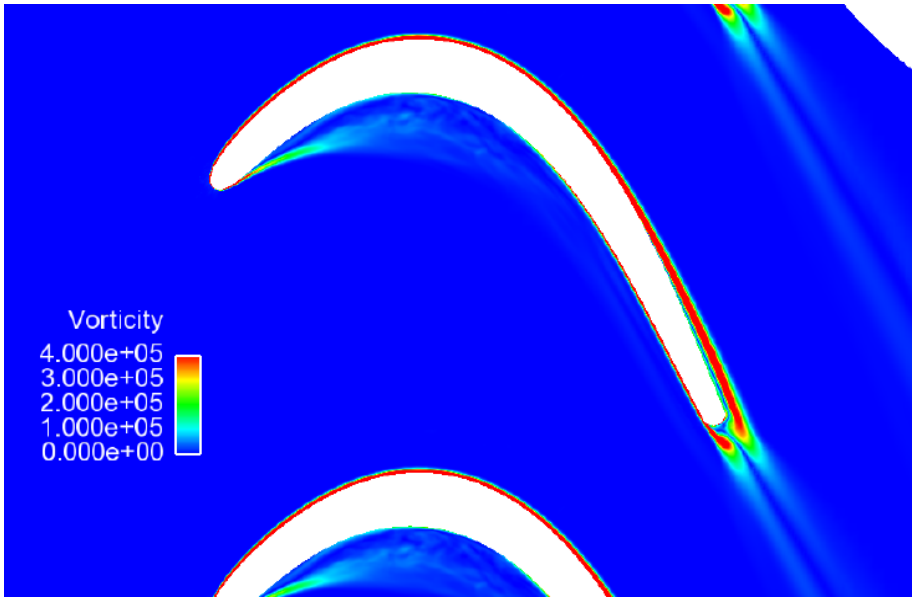
Figure 6.6: noFUDGE - transitional flow around an LP turbine blade. Computed instantaneous entropy fields.

the kinetic-energy preserving discretisation of the finite volume method in the presence of rapid transitions in mesh size or low quality elements. This inherent instability of the method forces the hybridisation with a conventional

upwind solver. In addition the FVM requires much smaller time steps to be used: in this case the timestep about four times smaller than the one used by DGM.

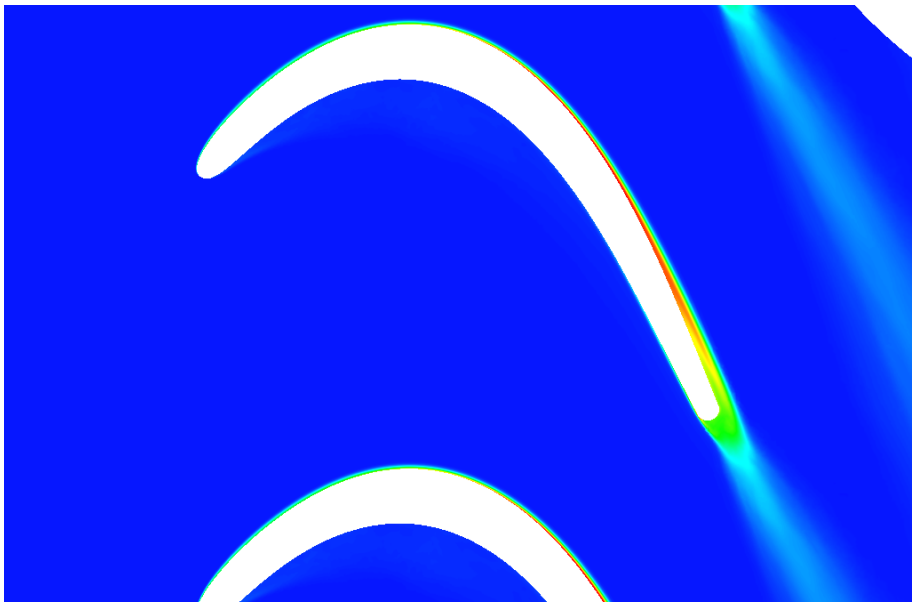


(a) DGM/DNS

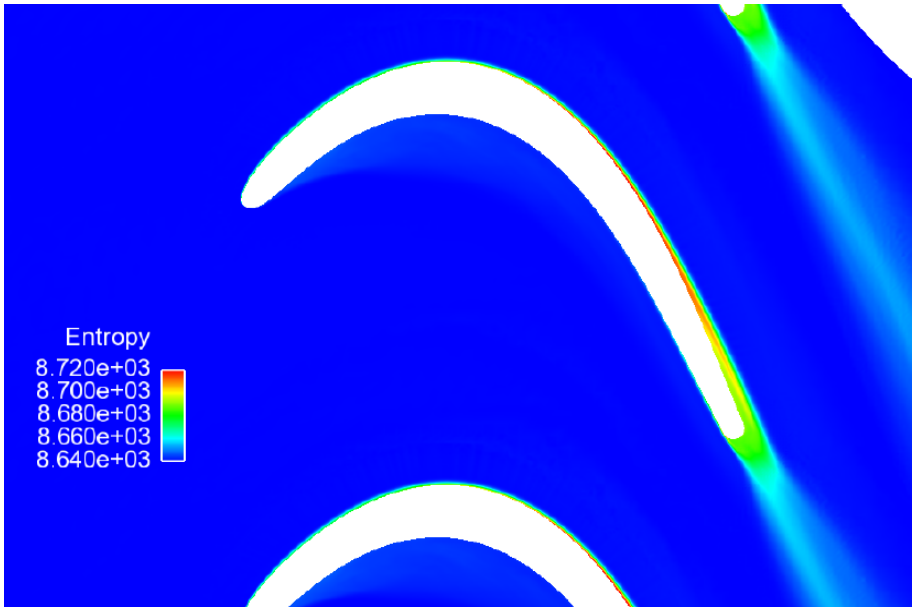


(b) FVM/LES

Figure 6.7: noFUDGE - transitional flow around an LP turbine blade. Time averaged vorticity fields.



(a) DGM/DNS



(b) FVM/LES

Figure 6.8: noFUDGE - transitional flow around an LP turbine blade. Time averaged entropy fields.

The time-averaged flow fields are shown in Fig. 6.7 and Fig. 6.8. Due to the very slow dynamics of the recirculation bubble, it was not possible to fully

converge the statistics. Nevertheless, qualitative analyses can be done. The differences between the mean values obtained by the two methods are less visible than in the instantaneous fields.

From the vorticity plot, one sees that the time-averaged recirculation bubbles at the pressure side and the trailing edge are smaller in the DNS computation. The shear layer detaching from the leading edge at the pressure side seems closer to the blade, leading to this smaller recirculation zone. Small oscillations can be seen near the trailing edge for the DNS computations, which are caused by the small number of samples available to compute the statistics.

6.4 Validation

The computations are validated with respect to the total pressure losses downstream of the passage, as measured at the von Karman institute. The comparison, shown in Fig. 6.9, indicates a much better capture of the location of the losses by the DGM computation than the LES computations, whilst the URANS computation is fully off.

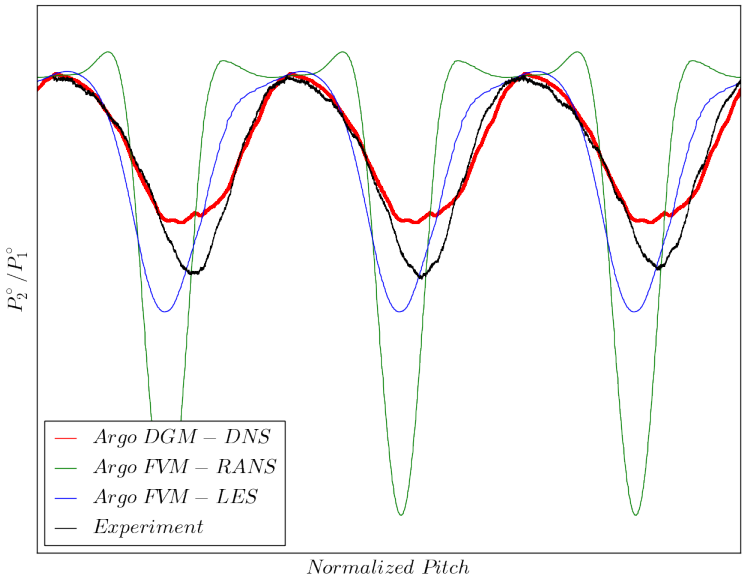


Figure 6.9: noFUDGE - transitional flow around an LP turbine blade. Comparison of the computed total pressure profile with measurements.

There is however still a discrepancy concerning the total loss in the center of the wake. Putting aside measurement errors (*e.g.* due to the finite size of the probes), three causes have been identified, which will be discussed in order of perceived plausibility:

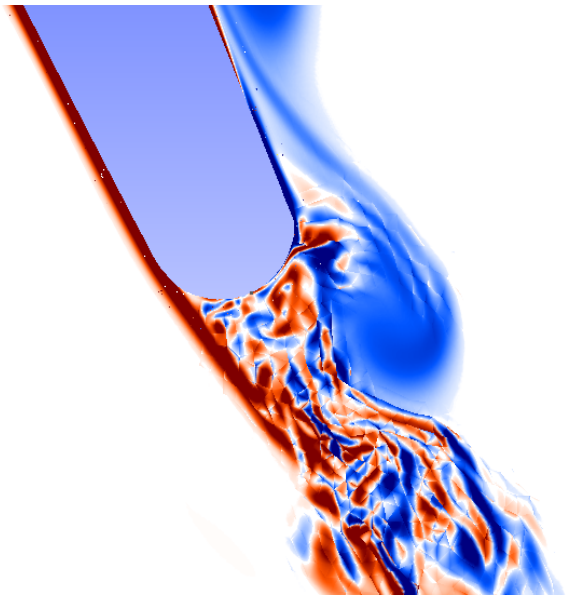


Figure 6.10: noFUDGE - transitional flow around an LP turbine blade. Instantaneous vorticity field at the trailing edge, computed with DNS/DGM

- the computed time-frame is not long enough to adequately average or even to guarantee reaching the fully developed regime, due to the low-frequency bursting of the pressure side recirculation bubble and in particular its irregular behaviour. This is the most difficult to check since it will probably require huge computational resources. This issue might by specific to the case at hand, since the pressure side recirculation is not a very common feature of LP turbine blades.

- the wind tunnel generates a certain amount of turbulence upstream of the cascade, which is currently not taken into account. This turbulence may trigger bypass transition, whereas now natural transition is computed. To model this effect, synthetic turbulence will have to be injected at the inlet. It is however difficult to judge whether bypass transition is really activated in this case. One should bear in mind that this approach will also increase computational cost, since the syntheticized turbulent structures need to be transported from the inlet to the blade.

- the resolution in the wake is currently inadequate to resolve all of the features. This is illustrated in Fig. 6.10, where small jumps in vorticity can be seen. However, given the experience with the behaviour of DGM in case of underresolved flow features, in particular the application to *implicit LES* (28), and the very slight underresolution, this is considered unlikely.

6.5 Comparison of computational cost

It is difficult to accurately estimate the relative cost of FVM and DGM for similar resolution on the turbine case as a reference is not available, and further grid convergence studies are too expensive, at least for the FVM solver. It is clear that a significant increase in resolution will be required to reach the resolution levels of the DGM computation.

The FVM computation takes approximately $11k$ cpu hours to compute one convective time and uses 700MB of memory on 256 Intel cores. The DGM computation performs one convective time in the equivalent of $1.12M$ cpu hours and takes 125MB on each of the 4096 BG/P cores. Table 6.2 summarizes the cost of the methods. Considering that a BG/P core is about 4 times slower than the Intel core, DGM is currently still 40% more expensive in terms of computational effort and about 80% in terms of memory for the same number of degrees of freedom. However DGM is much more accurate, less dissipative and much more stable than FVM with central scheme.

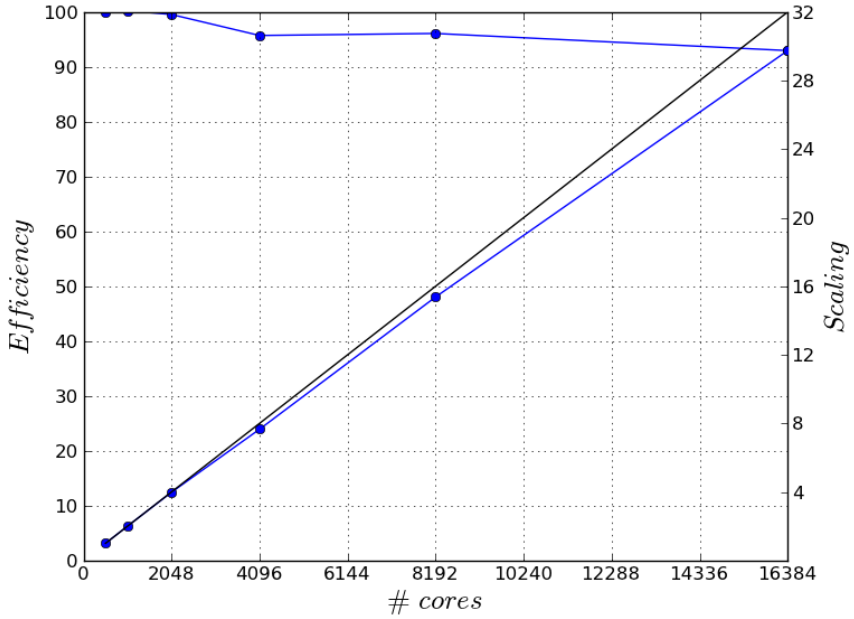
Table 6.2: Computational effort summary

	FVM	**DGM**
order of accuracy	2	4
mesh (m nodes)	8.5	15
cpu time for one t_c (kcpuh)	11	112
memory per core (mb)	700	500
number of cpu	256	4096
cpu time / mesh size	0.0013	0.0019
memory / mesh size	0.02	0.036

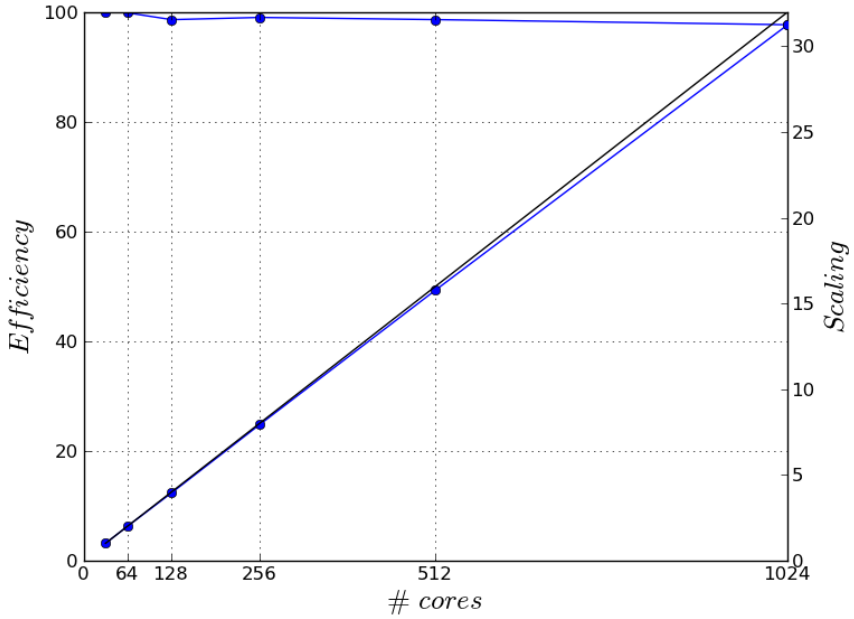
6.6 Scaling tests

A very important issue for this type of computations, is the efficient use of large computational resources. Furthermore, the PRACE consortium explicitly requires the demonstration of good scalability when requesting for a computational grant.

During the campaign weak scaling tests from 512 up to 16384 computational cores on the BlueGene/P machine have been undertaken. Thereby the standard setup for the implicit time-integration, namely Jacobian-free Newton-GMRES with a block-Jacobi preconditioner was used. A parallel efficiency of about 94% was obtained thereby demonstrating the excellent scalability of the method. For reference, similar tests from 16 up to 1024 processors were undertaken on the *zenobe* cluster of Cenaero.



(a) JuGene - BlueGene/P



(b) Zenobe - Intel/infiniband

Figure 6.11: Weak scaling obtained by the 4th order accurate version of Argo, using the implicit Newton-GMRES-Jacobi iterative scheme.

6.7 Conclusions

noFUDGE has shown that the developed DGM code methodology can be used for an industrial thematic, and thereby significantly increases the fidelity of the simulation up to direct numerical simulation. A very important issue that - in contrast to the FVM code - insufficient resolution can be detected easily. Indeed, only for the DGM computation it was clear where to further refine the mesh, or even possible to know if such a refinement would be useful, without having access to measurement data. Combined with the enhanced solution efficiency mentioned above, this is a very powerful feature.

The resolution used for this computation is not entirely sufficient to capture all of the very small-scale flow features at the trailing edge. In the near future resolution will be enhanced by increasing the interpolation order to allow for true DNS resolution. This type of adaptation should converge much faster than mesh refinement for regular flow features. The ease of grid and order adaptation, combined with simple criteria, namely continuity of the solution and vorticity, pave the way for significant further gains in computational resource usage.

CHAPTER 7

CURRENT STATUS AND PROSPECTS

7.1 Conclusions

The development of the discontinuous Galerkin code Argo is the result of a long and fruitful collaboration between Cenaero and Prof. Remacle; this thesis should be seen as one of its results. During the course of this research, the following steps towards an industrially viable high-resolution CFD code have been taken:

- The discretisation of the viscous terms by the interior penalty method on hybrid, high aspect ratio meshes was undertaken. On such meshes the stability of the interior penalty method was theoretically investigated, leading to proven optimal penalty coefficients.
- On the basis of the measured performance of algebraic primitives, flexible yet performant data structures were defined. This entailed first of all a generalisation of the data structure developed by Chevaugeon for the storage of data and residual vectors to hybrid meshes. The main contribution is the development of a flexible block-structured matrix, which provides near optimal performance for decomposition and and matrix-vector products.
- Implicit and parallel iterative strategies were developed
  - The most used class of solvers, is the matrix-free Newton-GMRES strategy with ILU and Jacobi preconditioners. Several measures were taken to enhance iterative efficiency, including the use of single precision preconditioners and frozen preconditioners;

  - The work on multigrid is more prospective. Up to now, a generic optimal framework for hp-multigrid was developed, which allows for the definition of high-quality transfer operators. The method is successfully applied for inviscid problems, but work remains on the definition of efficient smoothers for convection-diffusion problems.

- It was demonstrated that by reorganisation of the Jacobian assembly operations, the rapid increase of computational complexity, in other words the number of operations, can be offset by increased efficiency. The definition of intermediate contiguous and padded intermediate storage structures were shown to be of paramount importance;
- the applicability of DGM to frequential acoustics, including PML, was demonstrated;
- a post-processing technique was proposed to easily integrate non-conformal methods, in particular for the incompressible Navier-Stokes equations at low to moderate Reynolds numbers.

7.2 Current status of the Argo group

The Argo group has also obtained recognition in the high-order community, illustrated by the participation to the European research projects Adigma (FP6) and IDIHOM (FP7), and the organisation of test cases in the workshop on high-order methods for CFD (2; 3).

During the PRACE industrial pilot *noFUDGe*, the industrial viability was demonstrated by a proof-of-concept computation with industrial relevance. It concerned the direct numerical simulation of the transitional flow in a low pressure turbine cascade, representative for a small jet engine at cruise conditions.

The parallel scaling obtained by the code on BlueGene/P and Beowulf clusters has allowed Argo to successfully apply for HPC grants:

- *Computational Backbone for Unsteady LES and DNS (CoBaULD)*. This DEISA Extreme Computing Initiative project, obtained in 2009, applied Argo to the prediction of transitional flow on a low Reynolds airfoil;
- In 2011, the Argo team was invited as the first industrial user of PRACE resources during the project *Flow Unsteadiness predicted by DG (noFUDGe)*, which was dedicated to the simulation of the low pressure turbine;
- *P-Adaptive Discretisations for LES (PAdDLES)*. This two year project (2013-2014) was recently obtained during the 5th regular call of PRACE and will tackle LES simulations of the channel flow and full passage large eddy simulations.

The acceptance of Argo as an HPC-capable code hightens the chances to obtain further computational grants, which will be indispensible for the further development towards large scale applications.

7.3 Prospects

Two application areas have crystallised into strategic research axes for the Argo group at Cenaero, namely scale-resolving simulations of industrial aerodynamic turbulent flows on the one hand, and complex multiphase flows, includes production processes involving molten metal, glass or polymers, on the other. A very important transverse thematic, which will be picked up soon, is the development of hp-adaptation strategies. Finally, in the longer term, the coupling to vortex particle methods would be a very interesting development, greatly enhancing the capabilities of both methods.

The following sections briefly outline the current status and research prospects for the near future for the research directions mentioned above. Finally, the last sections will detail more prospective research projects done within the framework of the Argo platform.

Resolved turbulence for turbomachinery flow

Scale resolving simulations. The work within this thematic is the central topic of the doctoral research of Corentin Carton de Wiart, and has benefited greatly from the collaboration with Prof. Winckelmans and coworkers at UCL, and Prof. Bricteux at UMons.

Tests on the Taylor-Green vortex and low-Reynolds airfoils have demonstrated the interest and viability of the high order discontinuous Galerkin method for the *direct numerical simulation (DNS)* of transitional flows. A comprehensive comparison of different methods on this test case in the first international workshop on higher order methods (2; 108) shows that DGM and unstructured high-order methods in general can be competitive with high-order finite difference codes, provided the precision requirements are sufficiently high. The application to the LP turbine blade in noFUDGe has further demonstrated the industrial relevance and viability of the method. However, at least for some time to come, DNS will be limited to a small niche of applications.

The more generalised application of the method requires a further reduction of the computational cost through the use of large eddy simulation methods, and - further down the line - hybrid approaches based on wall models or a combination with RANS models near the wall.

Currently the research in the Argo group is mainly focused on the calibration of LES models (30; 28). The first results seem to indicate that the discontinuous Galerkin method is well-suited for the use of an *implicit LES (ILES)* modeling strategy, where the impact of the unresolved scales, and in particular the destruction of the kinetic energy, is taken care of by the numerical dissipation of the method (28). This is a very promising approach, because a priori it does not need calibration and can hence be applied to very complex flows, hosting a range of flow regimes. However, further work is needed to corroborate these results on higher Reynolds numbers; this effort is currently underway within the PRACE project *PAdDLES*. Further developments will include the introduction of hybrid modeling approaches, in particular wall models and / or hybrid RANS-LES.

Shock capturing A second thematic is the implementation of shock capturing strategies, and in particular the investigation of their interaction with turbulence modeling. This is persued in the doctoral research of Guillaume Verheylewegen, in collaboration with Prof. Remacle (UCL).

Complex multiphase flows

Surface capturing. In collaboration with Prof. Marchandise (UCL) surface capturing capabilities have been integrated within Argo by François Pochet. The method is a full high-order implementation of the level-set approach developed by Prof. Marchandise in her doctoral thesis (78). In this earlier work, the discontinuous Galerkin method was used for the resolution of the level-set equation only.

The availability of the DGM discretisation of both the level-set and the Navier-Stokes equations in Argo, combined with the observation that the flow fields require the same kind of resolution near the interface, led to the idea of solving both sets of equations in a strongly coupled fashion. The current status is described in the paper (86), which is currently in review.

The surface capturing technique is currently further extended in the thesis of Pierre Schrooyen (UCL), who will apply it to the simulation of the interaction of transition with the ablation of thermal protection shields, under the supervision of Prof. Chatelain (UCL) and Prof. Magin (VKI).

Complex rheology The numerical technology for the simulation of industrial processes, in particular those involving the flow of metals, is currently not very well developed. This is partly due to the complexity of the phenomena, as they combine flow interfaces to complex equations of state. The latter include phase transition, visco-elasto-plasticity, and physical phenomena such as the Marangoni stresses. To our knowledge the development of a model which combines all of these effects with a very flexible way of describing interfaces, would already be a huge technological advance.

Moreover these highly non-linear physics result in extremely complex and unpredictable flow configurations, where high order accuracy is probably required to avoid a too large dependence on the initial mesh. Also here adaptive strategies will be an important ingredient to truely efficient computations.

Adaptive computations

Order and mesh adaptation strategies are very powerful tools to reduce computational cost and dependence with respect to the mesh. A very good example is the tracking of wakes moving through a fixed mesh.

In theory DGM is particularly suited for this, since on the one hand, it provides the machinery for detecting and quantifying under resolution, and on the other it supports low quality meshes that may result from adaptation as well as variable order. Currently, a variable order prototype of Argo is being tested, in preparation of adaptive unsteady computations.

As the order of the method in some of the elements will be higher than currently used, high-order interpolants need to be developed. The use of Bézier splines seems a very simple and elegant approach, which provides additional possibilities for the verification and correction of the solution.

Coupling to vortex particle methods.

In the longer run, the turbulence resolving capabilities of Argo could be enhanced by coupling to vortex-particle methods, as developed at UCL by the TFL unit in iMMc. This would allow the combination of the high resolution near the wall typical of the high-order DGM code element methods to the capacity of the particle methods to convect turbulent structures over very long distances. This combination could offer significant advantages for the simulation of jet and wake noise, interaction of wind turbines in wind farms, the interaction of blade wakes in vertical axis wind turbines, ...

APPENDIX A

ELEMENTS OF FUNCTIONAL ANALYSIS

This chapter serves as to introduce the most basic concepts of functional analysis, needed to study the stability of the interior penalty method. It is very far from being exhaustive, and the reader is referred to (90) for a thorough general introduction to functional analysis, and (22) and (34) for a more finite element specific treatment.

A.1 Hilbert spaces

Definition 1 (Vector space) *A vector space $\mathcal{V}$ is a set of which the elements satisfy the following properties*

- $u, v \in \mathcal{V} \Rightarrow u + v \in \mathcal{V}$
- $u \in \mathcal{V}, \alpha \in \mathbb{R} \Rightarrow \alpha u \in \mathcal{V}$

Definition 2 (Cauchy sequence) *A Cauchy sequence on the space $\mathcal{V}$ is a sequence of which the elements come arbitrarily close to each other as the sequence progresses*

$$a_n : \forall \epsilon, \exists n : |a_k - a_{k+1}| < \epsilon \ \forall k > n \tag{A.1}$$

Definition 3 (Complete vector space) *A vector space $\mathcal{V}$ is complete if any Cauchy sequence of its elements converges to an element of $\mathcal{V}$.*

Definition 4 (Norm) *A norm $||.||$ on $\mathcal{V}$ is an operator $\mathcal{V} \to \mathbb{R}$ that satisfies*

- $||\alpha u|| = |\alpha| \, ||u|| \, , \ \forall u \in \mathcal{V}, \alpha \in \mathbb{R}$
- $||u + v|| \leq ||u|| + ||v|| \, , \ \forall u, v \in \mathcal{V}$ *(triangle inequality)*

- $u + v = u \Leftrightarrow ||v|| = 0\ ,\ \forall u, v \in \mathcal{V}$

Several norms can be defined on the same space. It is important to remark that for finite-dimensional spaces all norms are equivalent, *i.e.* any norm $||.||_a$ can be bounded above and below by any another norm $||.||_b$

$$\exists \alpha, \beta \in \mathbb{R}, \alpha, \beta > 0 : \alpha||u||_a \leq ||u||_b \leq \beta||u||_a \ \ \forall u \in \mathcal{V}$$

Definition 5 (Inner product) *An inner product $(.,.)$ is an operator $\mathcal{V} \times \mathcal{V} \to \mathbb{R}$ that satisfies*

- *symmetry:* $\forall u, v \in \mathcal{V} : (u, v) = (v, u)$
- *linearity in both arguments*

$$\forall u, v, w \in \mathcal{V} : (u + v, w) = (u, w) + (v, w)$$
$$(u, v + w) = (u, v) + (u, w)$$

- *positivity:* $\forall u \in \mathcal{V} : (u, u) \geq 0$

Several inner products can be defined on the same space. Any inner product furthermore defines a corresponding norm $||u|| = (u, u)$ on the same space.

Definition 6 (Hilbert space) *A Hilbert space is a complete vector space on which an inner product is defined.*

A.2 Solvability of variational problems

The following theorem governs the solvability of variational problems defined on two Hilbert spaces $\mathcal{U}$ and $\mathcal{V}$ (see Braess (22)) [1]

Theorem 1 (inf-sup condition) *Let $\mathcal{U}$ and $\mathcal{V}$ be Hilbert spaces and $a(.,.) : \mathcal{U} \times \mathcal{V} \to \mathbb{R}$ a a bilinear form*

$$a(u + v, w) = a(u, w) + a(v, w)$$
$$a(u, v + w) = a(u, v) + a(u, w)$$

that is both continuous

$$\exists C_1 \in \mathbb{R} : |a(u, v)| \leq C_1||u||||v|| \ \ \forall (u, v) \in \mathcal{U} \times \mathcal{V}$$

and satisfies the following inf-sup condition

$$\exists C_2 > 0, \forall v \in \mathcal{V} : \sup_{u \in \mathcal{U}} \frac{a(u, v)}{||u||} > C_2||v|| \tag{A.2}$$

Then the linear mapping

$$w \in \mathcal{V} \to u \in \mathcal{U} : a(u, v) = (w, v) \ \ \forall v \in \mathcal{V} \tag{A.3}$$

is an isomorphism (ie. a one to one mapping) between $\mathcal{U}$ and $\mathcal{V}$

[1] This is an adaption of the classical inf-sup theorem using the Riesz representation theorem, to avoid the introduction of dual space $\mathcal{V}'$, and to make the identification with the solution of variational problems more intuitive for non-mathematicians such as the author.

Remark that equation A.3 defines a variational problem, with which for instance a finite element problem can be identified. This is the most general setting for solvability, and holds both the famous Lax-Milgram theorem concerning the solvability of conformal problems and the Ladyzhenskaya-Babuška-Brezzi theorem for saddle-point problems (*e.g.* the Stokes equations) as special cases.

A.3 The Lax-Milgram theorem

The Lax-Milgram theorem provides sufficient conditions for the solvability of a variational problem defined on a single Hilbert space:

Theorem 2 (Lax-Milgram) *If $\mathcal{V}$ is a Hilbert space, $a(.,.) : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ a continuous bilinear form on $\mathcal{V} \times \mathcal{V}$ which is furthermore coercive*

$$\exists C_2 \in \mathbb{R}, C_2 > 0 : a(u,u) \geq C_2 ||u||^2 \ \ \forall u \in \mathcal{V}$$

and $f(.) : \mathcal{V} \to \mathbb{R}$ a continuous linear form

$$\exists C_3 \in \mathbb{R} : |f(v)| \leq C_3 ||v|| \ \ \forall v \in \mathcal{V}$$

then the variational problem

$$a(u,v) = f(v) \ , \ \forall v \in \mathcal{V} \tag{A.4}$$

has a unique solution $u \in \mathcal{V}$

The solvability of such a problem is in fact guaranteed by a specific variant of theorem 1, where $\mathcal{U} = \mathcal{V}$. However, the conditions of the inf-sup theorem are much more difficult to check than those of the Lax-Milgram theorem.

A.4 The most simple example

The space of n-dimensional vectors $\mathbb{R}^n$ equipped with the standard internal product

$$(\mathbf{u}, \mathbf{v}) = \mathbf{v}^T \mathbf{u} = \sum_i \mathbf{u}_i \mathbf{v}_i \tag{A.5}$$

is obviously a Hilbert space.

A simple bilinear form $a(.,.)$ on the spaces $\mathbb{R}^m$ and $\mathbb{R}^n$ is then defined with the help of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, whilst a linear form $f(.)$ on $\mathbb{R}^m$ is defined using a vector $\mathbf{f} \in \mathbb{R}^m$

$$\begin{aligned} a(.,.) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} : a(\mathbf{u}, \mathbf{v}) = \mathbf{v}^T \mathbf{A} \mathbf{u} \\ f(.) : \mathbb{R}^m \to \mathbb{R} : f(\mathbf{v}) = \mathbf{v}^T \mathbf{f} \end{aligned} \tag{A.6}$$

This example, harmless as it may seem, can be used as the basis of understanding solvability of finite element discretisations. In the end, the discretised equations will solve for the expansion weights which are effectively vectors in $\mathbb{R}^n$, with n the number of degrees of freedom. The matrix $\mathbf{A}$ and vector $\mathbf{f}$ then include the discretised variational form.

Continuity of the bilinear and linear operators are trivially satisfied by requiring all entries of $\mathbf{A}$ and $\mathbf{f}$ to be finite. The inf-sup condition to find a unique solution $\mathbf{u} \in \mathbb{R}^n$ to the variational problem

$$\mathbf{v}^T \mathbf{A}\mathbf{u} = \mathbf{v}^T \mathbf{b} \ , \ \forall \mathbf{v} \in \mathbb{R}^m \tag{A.7}$$

then translates as

$$\exists C_2 > 0 \ \ \forall \mathbf{v} \in \mathbb{R}^m : \ \sup_{\mathbf{u} \in \mathbb{R}^n} \frac{\mathbf{v}^T \mathbf{A}\mathbf{u}}{||\mathbf{u}||} > C_2 ||\mathbf{v}|| \tag{A.8}$$

Otherwise stated, the rank of $\mathbf{A}$ should be m; this directly implies of $n \geq m$.

The Lax-Milgram theorem on $\mathbb{R}^n$ can then be illustrated for a symmetric bilinear form and associated matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$. Coercivity of the bilinear form implies first of all that $\mathbf{A}$ has full rank. It is obvious that this latter condition is already guarantees a unique solution for the variational problem

$$\mathbf{v}^T \mathbf{A}\mathbf{u} = \mathbf{v}^T \mathbf{f} \ , \ \forall \mathbf{v} \in \mathbb{R}^n \tag{A.9}$$

or equivalently the linear system

$$\mathbf{A} \cdot \mathbf{u} = \mathbf{f} \tag{A.10}$$

since $\mathbf{A}$ is invertible. However coercivity is more restrictive than that.

The eigenvalue decomposition of $\mathbf{A}$ is given by

$$\mathbf{A} = \mathbf{R}\mathbf{\Lambda}\mathbf{R}^T \tag{A.11}$$

where

$$\mathbf{R} = \left[\mathbf{r}^1 \dots \mathbf{r}^n\right] \ , \ \mathbf{\Lambda} = \text{diag}\left(\lambda^1 \dots \lambda^n\right) \ , \ \mathbf{A} \cdot \mathbf{r}^i = \lambda^i \mathbf{r}^i \tag{A.12}$$

The set of eigenvectors can always be chosen such that they form an orthonormal basis for $\mathbb{R}^n$

$$\left(\mathbf{r}^i\right)^T \mathbf{r}^j = \delta_{ij} \tag{A.13}$$

Writing the expansion of an arbitrary vector $\mathbf{u}$ in the basis formed by $\mathbf{r}^i$ as

$$\mathbf{u} = \sum_{i=1}^{n} \alpha_i \mathbf{r}^i \tag{A.14}$$

one finds that

$$a\left(\mathbf{u}, \mathbf{u}\right) = \mathbf{u}^T \mathbf{A}\mathbf{u} = \sum_{i=1}^{n} \alpha_i^2 \lambda_i \tag{A.15}$$

As $||\mathbf{u}|| = \sqrt{\sum_{i=1}^{n} \alpha_i^2}$ this implies that all $\lambda_i > 0$ and hence that $\mathbf{A}$ is positive definite. Clearly this is more than is really required. The inf-sup theorem only requires that the matrix $\mathbf{A}$ has full rank, as one would expect.

APPENDIX B

FUNCTION SPACES, REFERENCE ELEMENTS AND QUADRATURE

The shape functions ϕ_i in Argo are defined as the Lagrange interpolants, which are members of the function $\mathcal{V}$ and pass through a set of well-chosen interpolation points $M = \{\mu_k\}$. The construction of these interpolants is elaborated in section B.1. In section B.2 it is shown that for a correct choice of interpolatory space and interpolation points, the set of shape functions will be such that on each of the element boundaries only a number of functions are active. In section B.3 we will elaborate reference element, functional space and interpolation points for different types of elements. Finally B.4 deals with the quadrature rules.

B.1 Construction of Lagrange interpolants

Consider the functional space $\mathcal{V}^e$, defined on element e, to be the span of the basis functions ψ_k

$$\mathcal{V} = \text{span}\,(\psi_k\ ,\ k = 1, \ldots, N_\phi) \tag{B.1}$$

Consider a set of interpolation points $\Xi = \{\mu_k \in e\ ,\ k = 1 \ldots N\}$ on the element e where the number of interpolation points corresponds to the number of shape functions N_ϕ. The set of Lagrange interpolants Λ, based on the function space $\mathcal{V}$ and a given set of interpolation point $\Xi = \{\mu_i\}$ is then defined as

$$\Lambda^e(\mathcal{V}, M) = \{\lambda_i \in \mathcal{V} : \lambda_i(\mu_j) = \delta_{ij}\} \tag{B.2}$$

If we consider an arbitrary set Ψ of basis functions ψ_k, we can express

$$\lambda_i = \sum \beta_{ij}\psi_j \tag{B.3}$$

Using the Lagrange interpolation property $\lambda_i(\mu_j) = \delta_{ij}$ we find

$$\begin{aligned} \delta_{ij} &= V(\mathcal{V}, \Xi)_{ik}\beta_{kj} \\ V(\Psi, \Xi)_{ik} &= \psi_i(\mu_j) \end{aligned} \tag{B.4}$$

Hence we can find a unique set of Lagrange interpolants, only if $V(\mathcal{V}, \Xi)$ is invertible.

B.2 Interpolation on the boundary

We call $\Lambda^e(\mathcal{V}, M)$ closed if the set of restricted Lagrange functions $\Lambda^f(\mathcal{V}^f, \Xi^f)$ is uniquely defined. Hereby we define Ξ^f as the set of interpolation points located on f

$$\Xi^f = \mu_i : \mu_i \in \Xi \cap f \tag{B.5}$$

and $\mathcal{V}^f$ the function space formed by the restriction of $\mathcal{V}$ to f.

First of all we observe that the restriction λ_i^f of any of the Lagrange interpolant λ_i associated to one of the boundary points $\mu_i \in \Xi^f$ to the face fis again a Lagrange interpolant based on $\mathcal{V}^f$ and Ξ^f. This implies linear independence of these restricted functions. Hence we can conclude that the set of functions λ_i^f forms a basis for the restricted function space $\mathcal{V}^f$ if the number of points corresponds to the dimension of $\mathcal{V}^f$.

Now let us consider a closed Lagrange basis. For any interpolation point ξ_k the restriction of its associated Lagrange interpolant λ_k to f can be expressed as

$$\lambda_k\Big|_f(\xi) = \sum_{j \in \Xi^f} \beta_{kj}\lambda_j\Big|_f(\xi) \tag{B.6}$$

Using the Lagrange interpolation property, and applying it to any λ_k associated to any point $\mu_k \in e \setminus f$ we find that for any point $\mu_i \in \Xi^f$:

$$\begin{aligned} \lambda_k\Big|_f(\mu_i) &= 0 \\ &= \sum_{j \in \Xi^f} \beta_{kj}\lambda_j\Big|_f(\mu_i) = \beta_{ki} \end{aligned} \tag{B.7}$$

showing that any interpolant λ_k associated to a point μ_k that is not located on the boundary f is exactly zero everywhere on that boundary.

Closed bases are useful mainly for two purposes:

- for continuous interpolation, the sharing of interpolation points on the common boundary between two elements ensures continuity. This is not only needed for C_0 continuity of the solution in classical FEM, but closed bases are equally indispensible for the mapping of curved elements, since they guarantees watertight meshes;
- the independence of the interpolated solution on the boundary from all points that are not on this boundary results in considerable savings for the interface terms in the discontinuous Galerkin method.

B.3 Specific elements

The canonical element of dimension d is defined by the bounds for the parametric coordinates ξ_i, whilst the function space is usually, but not always, defined by a set of monomials

$$\mu(\xi_1, \ldots, \xi_d) = \prod_{i=1}^{d} \xi_i^{p_i} \tag{B.8}$$

of which the exponents p_i are restricted. The description of the interpolation points μ_k finalises the description of the canonical finite element; these points are defined by an equidistant distribution on edges, faces, ... of the element.

The categories of simplices and tensor product elements are of generic dimension. The first contains triangles and simplices, whilst the second quadrilaterals and hexahedra. The last two element types are wedges with triangular basis and pyramids.

Obviously we can use the Vandermonde matrix to find the expressions for the Lagrange base functions for any type of element; yet for some elements, the interpolants can be constructed more economically on the basis of tensor products between function spaces of lower dimension, thus allowing easier implementations.

Lines

The canonical line element is defined by the following bound on the parametric coordinate ξ:

$$-1 \leq \xi \leq 1 \tag{B.9}$$

The functional space is given by

$$\mathcal{V}^{\mathcal{L}} = \text{span}\left(\xi^i \;\middle|\; 0 \leq i \leq p\right) \tag{B.10}$$

An obvious way to define the Lagrange interpolants is given by the classical formula

$$\lambda_i^*(\xi) = \prod_{j \neq i} \frac{\xi - \mu_j}{\mu_i - \mu_j} \tag{B.11}$$

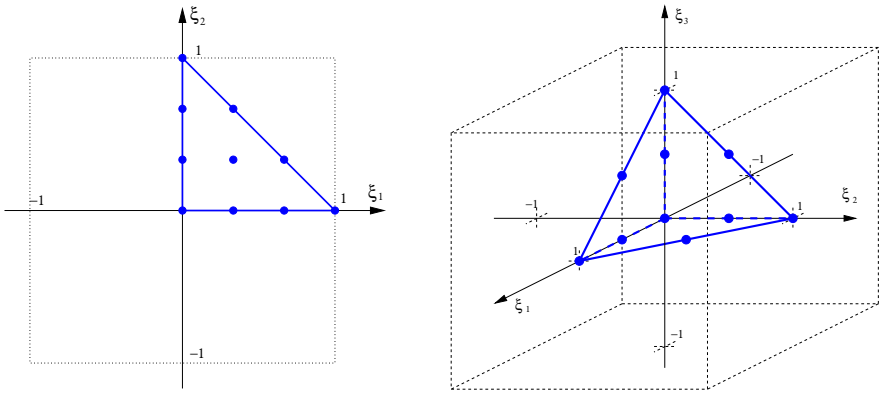
Simplices

In the simplex, next to the bounds on each of the coordinates, the sum of the parametric coordinates is bound as well

$$\begin{aligned} &0 \leq \xi_i \leq 1\ ,\ i = 1, \ldots, d \\ &0 \leq \sum_{i=1}^{d} \xi_i \leq 1 \end{aligned} \tag{B.12}$$

as is the sum of the exponents of the monomials

$$\mathcal{V}^{S_d} = \text{span}\left(\prod_{i=1}^{d} \xi_i^{p_i} \;\middle|\; 0 \leq \sum_{i=1}^{d} p_i \leq p\right) \tag{B.13}$$

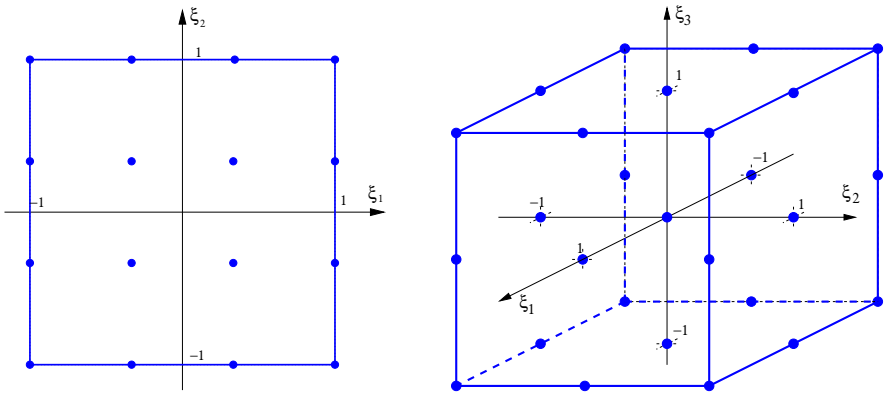


(a) 3^{rd} order triangle (b) 2^{nd} order tetrahedron

Figure B.1: Canonical simplices

This corresponds to the Pascal triangle or tetrahedron. The position of the interpolation points is illustrated in Fig. B.1.

Tensor product elements



(a) 3^{rd} order quadrilateral (b) 2^{nd} order hexahedron

Figure B.2: Canonical tensor product elements

The coordinates of the tensor product elements are bound between -1 and 1

$$-1 \leq \xi_i \leq 1 \ , \ i = 1, \ldots, d \tag{B.14}$$

whilst the function space $\mathcal{V}^{\mathcal{T}}$ is defined as

$$\mathcal{V}^{\mathcal{T}} = \text{span}\left(\prod_{i=1}^{d} \xi_i^{p_i} \ \middle| \ \forall i : 0 \leq p_i \leq p \right) \tag{B.15}$$

The position of the points is illustrated in Fig. B.2 for a 3^{rd} order quadrilateral and a 2^{nd} order hexahedron. The Lagrange interpolants are not constructed explicity, but by the tensor product of 1D Lagrange interpolants in each of the three directions.

Wedges

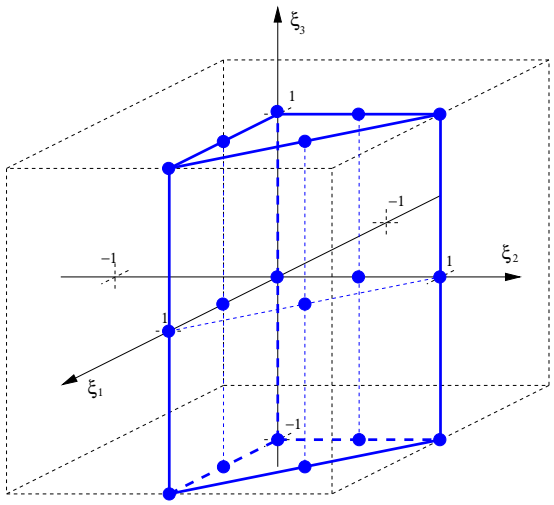


Figure B.3: 2^{nd} order canonical wedge

As the faces normal to ξ_3 are triangles, the sum of the two first coordinates is bound, next to the bounds per coordinate:

$$\begin{aligned} 0 \leq \xi_i \leq 1 \ , \ i = 1, 2 \\ -1 \leq \xi_3 \leq 1 \\ 0 \leq \sum_{i=1}^{2} \xi_i \leq 1 \end{aligned} \tag{B.16}$$

This element should provide a quadrilateral function space on each of the bounds along ξ_3, and a triangular function space on the faces normal to ξ_3. The functional space is defined by the tensor product of a triangular and a line function space

$$\mathcal{V}^{\mathcal{W}} = \mathcal{V}^{\mathcal{S}^2}(\xi_1, \xi_2) \otimes \mathcal{V}^{\mathcal{L}} \tag{B.17}$$

Using equidistant points as indicated in Fig. B.3 provides $p(p+1)(p+2)/2$ points, as many as the monomials.

Pyramids

The canonical pyramid is composed of a quadrilateral for the face normal to ξ_3 and triangles at all of the other. This means that next to the bound per

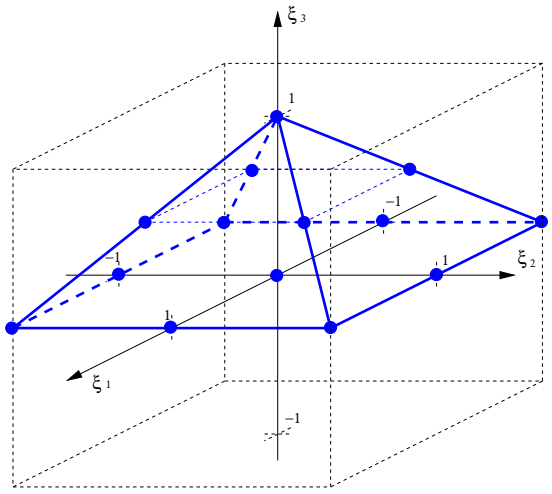


Figure B.4: 2^{nd} order canonical pyramid

coordinate, two partial sums - each involving ξ_3 - are bounded as well:

$$\begin{aligned} -1 \leq \xi_i \leq 1 \ , \ i = 1, 2 \\ 0 \leq \xi_3 \leq 1 \\ 0 \leq \xi_i + \xi_3 \leq 1 \ , \ i = 1, 2 \end{aligned} \tag{B.18}$$

The following set of functions $\psi^{\mathfrak{P}}_{ijk}$, proposed by (16), are orthogonal on the canonical pyramid $\mathfrak{P}$

$$\begin{aligned} \psi^{\mathfrak{P}}_{ijk} &= P_i\left(\frac{\xi_1}{1-\xi_3}\right) \ P_j\left(\frac{\xi_2}{1-\xi_3}\right) \ (1-\xi_3)^{\mu_{ij}} \ P_k^{(2\mu_{ij}+2,0)}\left(2\xi_3 - 1\right) \\ \mu_{ij} &= \max\left(i, j\right) \end{aligned} \tag{B.19}$$

here $P_p^{(m,n)}$ denotes the Jacobi polynomial of order p corresponding to the pair of exponents (m, n) (4). Bergot *et al.* furthermore show that the functional space $\mathcal{L}^{\mathfrak{P}}_p$

$$\mathcal{L}^{\mathfrak{P}}_p = \texttt{span}\left\{\psi^{\mathfrak{P}}_{ijk} \ : \ 0 \leq i, j \leq p \ , \ 0 \leq k \leq p - \mu_{ij}\right\} \tag{B.20}$$

is compatible with the standard Lagrange interpolants of the quadrilateral and the triangle on its faces (see (16)), thus allowing for conformal Lagrange interpolation. Furthermore it was also shown that $\mathcal{S}_p \subset \mathcal{L}^{\mathfrak{P}}_p$.

B.4 Quadrature rules

Generic quadrature rules are defined with respect to a given weight function g.

$$\int_{\mathfrak{e}} g f d\xi = \sum_{q=1}^{N_\alpha} w_q f(\alpha_q) \ , \ \forall f \in \mathcal{P}^o_{\mathfrak{e}} \tag{B.21}$$

Here w_q and α_q are the quadrature weight and position, whilst o is the order of accuracy of the rule.

The quadrature rules for the line element are well known and documented in most reference works on numerical analysis *e.g.* (4). In the general case the Gauss-Legendre rule for a set of functions up to order 2p is constructed using the zeros of the p-th order polynomial within the hierarchy of orthogonal polynomials - of course with respect to the weight function - defined on the same set.

The rules for other elements can be found in specific literature on high-order FEM, such as the book by (96). Table B.1 recapitulates the main characteristics of the Gauss-Legendre quadrature rules used for the tetrahedron and its boundary triangles up to an interpolation order $p = 6$.

o	4	5	6	7	8	9	11	13
triangle	6	7	12	13	16	19	25	27
tetrahedron	11	14	24	31	43	53	126	210

Table B.1: Number of quadrature points for order o on simplices.

The same reference contains also quadrature rules for other element types. However, one should bear in mind that these correspond to the Pascal triangular / tetrahedral functional space. Optimal quadrature rules for the standard Lagrange functional spaces as defined in for the quadrilateral, hexahedron and wedge are found by appropriate tensor products of the quadratures for the line and triangle. The Lagrange pyramid element is a special case, since the functional space is rational. The construction of the functional space and the appropriate quadrature rules are defined in (16); these quadrature rules result from the deformation of the hexahedral element, leading to the same computational cost.

APPENDIX C

SHARP VALUES FOR THE TRACE INVERSE INEQUALITY

During the coercivity analysis of the interior penalty method the trace inverse inequality

$$\int_f u^2 dS \leq C(p) \cdot \frac{\mathcal{A}(f)}{\mathcal{V}(e)} \int_e u^2 dV \ , \ \forall u \in \mathcal{P}(e) \tag{C.1}$$

is used to determine minimal values for the penalty coefficient σ. This inequality bounds the energy of any given function u of $\mathcal{P}$ integrated on a face f with respect to that integrated on the element e. Here $\mathcal{A}(f)$ is the boundary area/length of the face/edge f, and $\mathcal{V}(e)$ the volume/area of the element e.

As the conditioning of the discretised system depends heavily on σ, one would like to have as small values for $C(p)$ as possible, and this for all of the functional spaces and elements used. Such sharp values are available for the case of simplicial elements and have been developed by (109).

In the remainder of this chapter sharp values are developed for $C(p)$ for all standard Lagrange interpolation spaces $\mathcal{L}$, as well as the Pascal triangle or tetrahedron $\mathcal{S}$ on all "missing" types of elements used in hybrid unstructured meshes. In addition to simplices these include *tensor product elements* (lines, quadrilaterals and hexahedra), *wedges* (prisms with triangular basis) and *pyramids*. In order to keep the algebra tractable the analysis is restricted to linear mappings, ie. with a constant Jacobian. For practical application, it is further assumed that when the element is not too much deformed, the proposed values will be close to the actual ones. These developments are the subject of a paper submitted to SINUM (60).

C.1 Simplices

In the case of the Pascal triangle / tetrahedron functional space, the trace inequality reads (109)

$$\int_f u^2 dS \leq \frac{(p+1)(p+d)}{d} \cdot \frac{\mathcal{A}(f)}{\mathcal{V}(e)} \int_e u^2 dV \tag{C.2}$$

The sharpness of the bound was proven by providing a function for which equation C.2 reverts to an equality. The outline of the method is given in the next section, as it is at the basis of the computation of the newly proposed values.

C.2 Outline of Warburton's method

First of all an orthonormal basis $\{\psi_i\}$ is chosen for the functional space $\mathcal{P}$ - which is defined in parametric coordinates - on the element. This leads obviously to a simple expression for the volume integral:

$$\int_e u^2 dV = \alpha_{\mathcal{V}} \mathcal{V}(e) \sum_{i=1}^{n} \mathbf{u}_i^2 \tag{C.3}$$

In this expression $\mathbf{u}_i$ are the expansion weights of u with respect to ψ_i, whilst α_V is a factor depending on the mapping between the real and the canonical element. Due to the loss of complete orthogonality, the surface integral of the energy then takes the following form

$$\int_f u^2 dS = \alpha_{\mathcal{A}} \mathcal{A}(f) \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{u}_i \mathbf{F}_{ij} \mathbf{u}_j \tag{C.4}$$

Given C.3, one can then bound C.4 with respect to C.3 by the 2-norm of $\mathbf{F}$

$$\int_f u^2 dS \leq \frac{\mathcal{A}(f)}{\mathcal{V}(e)} \frac{\alpha_{\mathcal{A}}}{\alpha_{\mathcal{V}}} ||\mathbf{F}||_2 \int_e u^2 dV \tag{C.5}$$

Due to the remaining partial orthogonality of the functions ψ_i on f, the functions ψ_i can be ordered such that $\mathbf{F}$ is block-diagonal, *ie.* composed of K symmetric positive semi-definite sub-blocks $\mathbf{F}^k$ on the diagonal, with (potentially different) sizes n_k

$$\begin{aligned} \int_f u^2 dS &= \alpha_{\mathcal{A}} \mathcal{A}(f) \sum_{k=1}^{K} \sum_{i=1}^{n_k} \sum_{j=1}^{n_k} \mathbf{u}_{(i+N_k)} \mathbf{F}_{ij}^k \mathbf{u}_{(j+N_k)} \\ N_k &= \sum_{k'<k} n_k \end{aligned} \tag{C.6}$$

The 2-norm of $\mathbf{F}$ is then obtained by considering the maximum of the 2-norms of the sub-blocks $\mathbf{F}^k$

$$\begin{aligned}\int_f u^2 dS &= \alpha_{\mathcal{A}}\mathcal{A}(f)\sum_{k=1}^{K}\left(\sum_{i=1}^{n_k}\sum_{j=1}^{n_k}\mathbf{u}_{(i+N_k)}\mathbf{F}^k_{ij}\mathbf{u}_{(j+N_k)}\right)\\ &\leq \alpha_{\mathcal{A}}\mathcal{A}(f)\sum_{k=1}^{K}||\mathbf{F}^k||_2\sum_{i=1}^{n_k}\mathbf{u}^2_{(i+N_k)}\\ &\leq \alpha_{\mathcal{A}}\mathcal{A}(f)\max_k||\mathbf{F}^k||_2\sum_{k=1}^{K}\sum_{i=1}^{n_k}\mathbf{u}^2_{(i+N_k)}\end{aligned} \tag{C.7}$$

Since $\mathbf{F}$ and all of its sub-blocks $\mathbf{F}^k$ are symmetric positive semi-definite, the 2-norm of each $\mathbf{F}^k$ is equal to its spectral radius $\rho(\mathbf{F}^k)$. Consequently equation C.7 reverts to an equality whenever $\mathbf{u}$ aligns with the eigenvector corresponding to the largest eigenvalue of the corresponding block(s). This immediately proves the sharpness of the bounds obtained in the process.

A useful result often used in the following sections is that for any matrix the trace sum is equal to the sum of its eigenvalues. This property is particularly easy to exploit for rank-1 matrices, since there is only one non-zero eigenvalue. Therefore

$$\rho(\mathbf{A}) = \left|\sum_{i=1}^{n}\mathbf{A}_{ii}\right| \tag{C.8}$$

Obviously an external (dyadic) product of two vectors results in a rank-1 matrix. Therefore, to justify the use of C.8, the expression for $\mathbf{F}^k$ will be cast as such a product.

C.3 Tensor product elements

The canonical tensor product element $\mathfrak{Q}^d$ is defined as

$$\mathfrak{Q}^d = \{(\xi_1,\ldots,\xi_d)\ :\ -1\leq\xi_i\leq 1\ ,\ i=1\ldots\ d\} \tag{C.9}$$

The functions $\psi^{\mathfrak{Q}}_{(p_1\ldots\ p_d)}$ defined as

$$\psi^{\mathfrak{Q}}_{(p_1\ldots\ p_d)} = {P_{p_1}}^*(\xi_1)\ldots\ {P_{p_d}}^*(\xi_d) \tag{C.10}$$

form an orthonormal basis for both $\mathcal{L}^{\mathfrak{Q}}_p$ and $\mathcal{S}_p$ on the element $\mathfrak{Q}$

$$\begin{aligned}\mathcal{L}^{\mathfrak{Q}}_p &= \mathtt{span}\left\{\psi^{\mathfrak{Q}}_{p_1\ldots p_d}\ :\ 0\leq p_i\leq p\ ,\ i=0\ldots d\right\}\\ \mathcal{S}_p &= \mathtt{span}\left\{\psi^{\mathfrak{Q}}_{p_1\ldots p_d}\ :\ 0\leq\sum_{i=1}^{d}p_i\leq p\right\}\end{aligned} \tag{C.11}$$

Here ${P_n}^*$ is the normalised Legendre polynomial of order n, *i.e.*

$${P_n}^*(\xi) = P_n(\xi)\cdot\sqrt{\frac{2n+1}{2}} \tag{C.12}$$

with P_n the standard Legendre polynomial as defined in (4). The values of ${P_n}^*$ on the boundaries of the interval $[-1, 1]$ are given by

$$ {P_n}^*(\pm 1) = P_n(\pm 1)\sqrt{\frac{2n+1}{2}} = (\pm 1)^n \sqrt{\frac{2n+1}{2}} \tag{C.13} $$

A function $u \in \mathcal{P}_p$ is then expanded as

$$ u = \sum_{(p)} \mathbf{u}_{(p)} \cdot \psi_{(p)} \tag{C.14} $$

where (p) is shorthand for the composite index $(p_1 \dots\ p_d)$. Due to the orthonormality of the functions $\psi_{(p)}$ the element integral can be written as

$$ \int_{\mathfrak{e}} u^2 \, d\xi_1 \dots \, d\xi_d = \sum_{(p)} \mathbf{u}^2_{(p)} \tag{C.15} $$

The bound for the boundary integral will be computed on the face $\xi_1 = -1$

$$ \int_{\mathfrak{f}} u^2 \, d\xi_2 \dots \, d\xi_d = \sum_{(p)} \sum_{(q)} \mathbf{u}_{(p)} \mathbf{F}_{(p)(q)} \mathbf{u}_{(q)} \tag{C.16} $$

where one finds

$$ \mathbf{F}_{(p)(q)} = \frac{\sqrt{(2p_1+1)(2q_1+1)}}{2} (-1)^{p_1+q_1} \cdot \delta_{p_2 q_2} \dots \, \delta_{p_d q_d} \tag{C.17} $$

with δ the Kronecker delta.

For the Lagrange space $\mathcal{L}_p^{\Omega}$ the matrix $\mathbf{F}$ is block diagonal, composed of $(p+1)^{d-1}$ blocks, each of size $(p+1)$. Each of these blocks is the same rank-one symmetric positive semi-definite matrix. The only non-trivial eigenvalue of a rank one matrix is equal to its trace, resulting in the same eigenvalue for all sub-blocks:

$$ \frac{1}{2} \sum_{i=1}^{p} (2i+1) = \frac{1}{2}(p+1)^2 \tag{C.18} $$

Hence

$$ \sum_{(p)} \sum_{(q)} \mathbf{u}_{(p)} F^{\xi}_{(p)(q)} \mathbf{u}_{(q)} \leq \frac{(p+1)^2}{2} \sum_{p} \mathbf{u}_p^2 \tag{C.19} $$

For an element with a constant mapping between the parametric coordinates ξ_i and physical coordinates this inequality finally reads

$$ \int_f u^2 dS \leq (p+1)^2 \frac{\mathcal{A}(f)}{\mathcal{V}(e)} \int_e u^2 dV \tag{C.20} $$

For the Pascal functional space $\mathcal{S}_p$ the matrix $\mathbf{F}$ is composed of $\frac{\prod_{i=1}^{d-1} p+i}{(d-1)!}$ blocks of size $(p - \sum_{i=2}^{d} p_i)$. Each of these blocks is a rank-one symmetric positive semi-definite matrix. The largest block, *i.e.* corresponding to $p_i = 0\ ,\ i = 2 \ldots d$ again has spectral radius $(p+1)^2$, such that the same bound (Eq. (C.20)) applies.

C.4 Wedges

The canonical wedge is defined as

$$\mathfrak{W} = \{(\xi_1, \xi_2, \xi_3)\ :\ 0 \leq \xi_1 + \xi_2 \leq 1\ ,\ -1 \leq \xi_3 \leq 1\} \tag{C.21}$$

The tensor product of any set of orthonormal functions $\kappa^*_{ij}(\xi, \eta)$ on the triangle[1] and the scaled Legendre polynomial $\psi^*_k(\zeta)$ of the previous section provide an orthonormal basis for both the standard Lagrange $\mathcal{L}^{\mathfrak{W}}_p$ as for the Pascal $\mathcal{S}_p$ space on the wedge $\mathfrak{W}$

$$\begin{aligned} \mathcal{L}^{\mathfrak{W}}_p &= \mathtt{span}\left\{\psi^{\mathfrak{W}}_{ijk}\ :\ 0 \leq i+j \leq p\ ,\ 0 \leq k \leq p\right\} \\ \mathcal{S}_p &= \mathtt{span}\left\{\psi^{\mathfrak{W}}_{ijk}\ :\ 0 \leq i+j+k \leq p\right\} \end{aligned} \tag{C.22}$$

Triangular faces

At $\xi_3 = -1$ one finds for both the Pascal $\mathcal{S}_p$ and standard Lagrange space $\mathcal{L}^{\mathfrak{W}}_p$

$$\begin{aligned} \int_{\mathfrak{f}} u^2 d\xi_1 d\xi_2 &= \sum_{ijk}\sum_{lmn}\left(\int_{\mathfrak{f}} \kappa^*_{(ij)}(\xi_1,\xi_2)\kappa^*_{(lm)}(\xi_1,\xi_2) d\xi_1 d\xi_2\right) {P_m}^*(-1){P_n}^*(-1)\mathbf{u}_{ijk}\mathbf{u}_{lmn} \\ &= \sum_{ijk}\sum_{lmn}\delta_{(ij)(lm)}(-1)^{k+n}\frac{\sqrt{(2k+1)(2n+1)}}{2}\mathbf{u}_{ijk}\mathbf{u}_{lmn} \end{aligned} \tag{C.23}$$

For $\mathcal{L}^{\mathfrak{W}}_p$ we find

$$\begin{aligned} \int_{\mathfrak{f}} u^2 d\xi_1 d\xi_2 &= \sum_{ij}\sum_{k=1}^{p}\sum_{n=1}^{p}\underbrace{(-1)^{k+n}\frac{\sqrt{(2k+1)(2n+1)}}{2}}_{\mathbf{F}^{ij}_{kn}}\mathbf{u}_{ijk}\mathbf{u}_{ijn} \\ &\leq \sum_{ij}\left(\sum_{k'=1}^{p}\frac{2k'+1}{2}\right)\sum_{k=1}^{p}\mathbf{u}^2_{ijk} \\ &\leq \frac{(p+1)^2}{2}\sum_{k=1}^{p}\mathbf{u}^2_{ijk} = \frac{(p+1)^2}{2}\int_{\mathfrak{e}} u^2 d\xi_1 d\xi_2 d\xi_3 \end{aligned} \tag{C.24}$$

[1] The particular form does not matter; obvious choices are the Priorol-Koornwinder-Dubiner basis, or the functions obtained by Gramm-Schmidt orthogonalisation starting from the functions in the Pascal triangle.

since any of the sub-blocks of $\mathbf{F}$ (indices k and n) is clearly a rank-one matrix. For $\mathcal{S}_p$ we find the same value since

$$\begin{aligned}\int_{\mathfrak{f}} u^2 d\xi_1 d\xi_2 &= \sum_{ij} \sum_{k=1}^{p-(i+j)} \sum_{n=1}^{p-(i+j)} (-1)^{k+n} \frac{\sqrt{(2k+1)(2n+1)}}{2} \mathbf{u}_{ijk}\mathbf{u}_{ijn} \\ &\leq \sum_{ij} \left(\sum_{k'=0}^{p-(i+j)} \frac{2k'+1}{2} \right) \sum_{k=1}^{p-(i+j)} \mathbf{u}_{ijk}^2 \\ &\leq \sum_{ij} \frac{(p+1-i-j)^2}{2} \sum_{k=1}^{p-(i+j)} \mathbf{u}_{ijk}^2 \\ &\leq \frac{(p+1)^2}{2} \sum_{ijk} \mathbf{u}_{ijk}^2 = \frac{(p+1)^2}{2} \int_{\mathfrak{e}} u^2 d\xi_1 d\xi_2 d\xi_3\end{aligned} \tag{C.25}$$

Finally when factoring in the mapping from $\mathfrak{e}$ to e resp. $\mathfrak{f}$ to f, thereby assuming a linear transformation for simplicity, one finds

$$\int_f u^2 dS \leq (p+1)^2 \frac{\mathcal{A}(f)}{\mathcal{V}(e)} \int_e u^2 dV \tag{C.26}$$

Quadrangular faces

On the face $\xi_1 = 0$ the surface integral for both $\mathcal{S}_p$ and $\mathcal{L}_p^{\mathfrak{W}}$ is found to be

$$\begin{aligned}&\int_{\mathfrak{f}} u^2 d\xi \\ &\quad = \sum_{ijk} \sum_{lmn} \mathbf{u}_{ijk}\mathbf{u}_{lmn} \int_{-1}^{1} {P_k}^*(\xi_3) {P_n}^*(\xi_3) d\xi_3 \left(\int_0^1 \kappa^*_{(ij)}(0,\xi_2)\kappa^*_{(lm)}(0,\xi_2) d\xi_2 \right)\end{aligned} \tag{C.27}$$

Using Warburtons results, and taking into account the difference in the definition of the canonical triangle, the following inequality is found

$$\int_0^1 \kappa^*_{(ij)}(0,\xi_2)\kappa^*_{(lm)}(0,\xi_2) d\xi_2 \leq (p+1)(p+2) \int_0^1 d\xi_1 \int_0^{1-\xi_1} \kappa^*_{(ij)}(\xi_1,\xi_2)\kappa^*_{(lm)}(\xi_1,\xi_2) d\xi_2 \tag{C.28}$$

and hence

$$\int_{\mathfrak{f}} u^2 d\xi \leq (p+1)(p+2) \int_{\mathfrak{e}} u^2 d\xi \tag{C.29}$$

When factoring in the mapping between parametric and physical element, again assuming a linear transformation, one finds

$$\int_f u^2 dS \leq \frac{(p+1)(p+2)}{2} \frac{\mathcal{A}(f)}{\mathcal{V}(e)} \int_e u^2 dV \tag{C.30}$$

C.5 Lagrange interpolation on pyramids

The canonical pyramid $\mathfrak{P}$ is defined as

$$\mathfrak{P} = \{(\xi_1, \xi_2, \xi_3) \ : \ -1 + \xi_3 \leq \xi_1, \xi_2 \leq 1 - \xi_3 \ , \ 0 \leq \xi_3 \leq 1\} \tag{C.31}$$

The functions $\psi_{ijk}^{\mathfrak{P}}$ are first normalised. To compute the autocorrelations between the functions a Duffy transformation

$$(\hat{\xi}_1, \hat{\xi}_2, \hat{\xi}_3) \to (\xi_1, \xi_2, \xi_3) : \xi_1 = \frac{1}{2}(1 - \hat{\xi}_3)\hat{\xi}_1 \ , \ \xi_2 = \frac{1}{2}(1 - \hat{\xi}_3)\hat{\xi}_2 \ , \ \xi_3 = \frac{1}{2}(1 + \hat{\xi}_3) \tag{C.32}$$

is used to map the canonical cube onto the canonical pyramid, resulting in

$$\begin{aligned}
&\int_{\mathcal{P}} \psi_{ijk}^{\mathfrak{P}} \psi_{lmn}^{\mathfrak{P}} d\xi \\
&= \frac{1}{2^{\mu_{ij}+\mu_{lm}+3}} \int_{-1}^{1} P_i(\hat{\xi}_1) P_l(\hat{\xi}_1) d\hat{\xi}_1 \int_{-1}^{1} P_j(\hat{\xi}_2) P_m(\hat{\xi}_2) d\hat{\xi}_2 \dots \\
&\quad \dots \int_{-1}^{1} (1 - \hat{\xi}_3)^{\mu_{ij}+\mu_{jm}+2} P_k^{(2\mu_{ij}+2,0)}(\hat{\xi}_3) P_n^{(2\mu_{lm}+2,0)}(\hat{\xi}_3) d\hat{\xi}_3 \\
&= \delta_{il}\delta_{jm} \frac{1}{2^{2\mu_{ij}+3}} \frac{2}{2i+1} \frac{2}{2j+1} \int_{-1}^{1} (1 - \hat{\xi}_3)^{2\mu_{ij}+2} P_k^{(2\mu_{ij}+2,0)}(\hat{\xi}_3) P_n^{(2\mu_{ij}+2,0)}(\hat{\xi}_3) d\hat{\xi}_3 \\
&= \delta_{il}\delta_{jm}\delta_{kn} \frac{1}{2^{2\mu_{ij}+3}} \frac{2}{2i+1} \frac{2}{2j+1} \frac{2^{2\mu_{ij}+3}}{2k+2\mu_{ij}+3} \frac{\Gamma(k+2\mu_{ij}+3)\Gamma(k+1)}{k!\Gamma(k+2\mu_{ij}+3)} \\
&\quad = \delta_{il}\delta_{jm}\delta_{kn} \frac{2}{2i+1} \ \frac{2}{2j+1} \ \frac{1}{2k+2\mu_{ij}+3}
\end{aligned} \tag{C.33}$$

so $\psi_{ijk}^{\mathfrak{P}}$ is normalised as

$$\psi_{ijk}^{\mathfrak{P}*} = \frac{1}{2}\sqrt{(2i+1)(2j+1)(2k+2\mu_{ij}+3)}\ \psi_{ijk}^{\mathfrak{P}} = C_{ijk}\psi_{ijk}^{\mathfrak{P}} \tag{C.34}$$

Quadrilateral face

The quadrilateral base is given by $\xi_3 = 0$ so one finds

$$\begin{aligned}
\int_{\mathfrak{f}} \psi_{ijk}^{\mathfrak{P}*} \psi_{lmn}^{\mathfrak{P}*} d\xi_1 d\xi_2 &= C_{ijk} C_{lmn} (-1)^{k+n} \int_{-1}^{1} P_i(\xi_1) P_l(\xi_1) d\xi_1 \int_{-1}^{1} P_j(\xi_2) P_m(\xi_2) d\xi_2 \\
&= C_{ijk} C_{ijn} (-1)^{k+n} \delta_{il}\delta_{jm} \frac{2}{2i+1} \frac{2}{2j+1} \\
&= \delta_{il}\delta_{jm} (-1)^{k+n} \sqrt{(2k+2\mu_{ij}+3)(2n+2\mu_{ij}+3)}
\end{aligned} \tag{C.35}$$

leading to

$$\begin{aligned}
&\int_{\mathfrak{f}} u^2 d\xi_1 \\
&\quad = \sum_{i=0}^{p}\sum_{j=0}^{p}\left(\sum_{k=0}^{p-\mu_{ij}}\sum_{n=0}^{p-\mu_{ij}} \mathbf{u}_{(ijk)}(-1)^{k+n}\sqrt{(2k+2\mu_{ij}+3)(2n+2\mu_{ij}+3)}\mathbf{u}_{(ijn)}\right) \\
&\quad = \sum_{i=0}^{p}\sum_{j=0}^{p}\left(\sum_{k=0}^{p-\mu_{ij}}\sum_{n=0}^{p-\mu_{ij}} \mathbf{u}_{(ijk)}\mathbf{F}^{(ij)}_{kn}\mathbf{u}_{(ijn)}\right)
\end{aligned} \tag{C.36}$$

The matrix $\mathbf{F}$ is composed of $(p+1)^2$ blocks $\mathbf{F}^{(ij)}$ on the diagonal - one for each of the combinations (i,j). All of these blocks are symmetric positive rank one matrices, so we can bound the surface integral as

$$\int_{\mathfrak{f}} u^2 d\xi_1 \leq \sum_{i=0}^{p}\sum_{j=0}^{p} ||\mathbf{F}^{(ij)}||_2 \sum_{k=0}^{p-\mu_{ij}} \mathbf{u}_{ijk}^2 \tag{C.37}$$

with

$$\begin{aligned}
||\mathbf{F}^{(ij)}||_2 &= \sum_{k=0}^{p-\mu_{ij}} (2k+2\mu_{ij}+3) \\
&= (p-\mu_{ij})^2 + (p-\mu_{ij}) + (p-\mu_{ij}+1)(2\mu_{ij}+3) \\
&= (p+1)(p+3) - \mu_{ij}(\mu_{ij}+2)
\end{aligned} \tag{C.38}$$

The 2-norm of the blocks $\mathbf{F}^{(ij)}$ depends on the index pair (i,j); hence the norm of the global matrix $\mathbf{F}$ is given by the largest value, leading to

$$\int_{\mathfrak{f}} u^2 d\xi_1 d\xi_2 \leq (p+1)(p+3)\int_{\mathcal{P}} u^2 d\xi_1 d\xi_2 d\xi_3 \tag{C.39}$$

When factoring in the mapping to the real element the following inequality results

$$\int_f u^2 dS \leq \frac{(p+1)(p+3)}{3}\frac{\mathcal{A}(f)}{\mathcal{V}(e)}\int_e u^2 dV \tag{C.40}$$

Triangular faces

One can take for instance the face defined by $1-\xi_2-\xi_3=0$. Restricted to this face, the functions $\psi^{\mathfrak{P}*}_{ijk}$ reduce to

$$\begin{aligned}
\psi^{\mathfrak{P}*}_{ijk}\Big|_f &= C_{ijk}\, P_i\left(\frac{\xi_1}{1-\xi_3}\right)\, P_j(1)\, \left(1-\xi_3\right)^{\mu_{ij}}\, P_k^{(2\mu_{ij}+2,0)}(2\xi_3-1) \\
&= C_{ijk} P_i\left(\frac{\xi_1}{1-\xi_3}\right)\, \left(1-\xi_3\right)^{\mu_{ij}}\, P_k^{(2\mu_{ij}+2,0)}(2\xi_3-1) \\
&= \frac{C_{ijk}}{2^{\mu_{ij}}} P_i(\hat{\xi}_1)(1-\hat{\xi}_3)^{\mu_{ij}} P_k^{(2\mu_{ij}+2,0)}(\hat{\xi}_3)
\end{aligned} \tag{C.41}$$

Using again the Duffy transformation (equation (C.32)) the surface integral is given by

$$\int_f u^2 d\xi_1 d\xi_3 = \sum_{i=0}^{p} \sum_{j=0}^{p} \sum_{k=0}^{p-\mu_{ij}} \sum_{l=0}^{p} \sum_{m=0}^{p} \sum_{n=0}^{p-\mu_{lm}} \mathbf{u}_{(ijk)} \mathbf{F}_{(ijk)(lmn)} \mathbf{u}_{(lmn)} \tag{C.42}$$

where $\mathbf{F}$ is given by

$$\begin{aligned}
\mathbf{F}_{(ijk)(lmn)} =& \frac{C_{ijk} C_{lmn}}{2^{\mu_{ij}+\mu_{lm}+2}} \int_{-1}^{1} P_i(\hat{\xi}_1) P_l(\hat{\xi}_1) d\hat{\xi}_1 \dots \\
& \dots \int_{-1}^{1} (1-\hat{\xi}_3)^{\mu_{ij}+\mu_{lm}+1} P_k^{(2\mu_{ij}+2,0)}(\hat{\xi}_3) P_n^{(2\mu_{lm}+2,0)}(\hat{\xi}_3) d\hat{\xi}_3 \\
=& \frac{C_{ijk} C_{lmn}}{2^{\mu_{ij}+\mu_{im}+2}} \frac{2\delta_{il}}{2i+1} \dots \\
& \dots \int_{-1}^{1} (1-\hat{\xi}_3)^{\mu_{ij}+\mu_{im}+1} P_k^{(2\mu_{ij}+2,0)}(\hat{\xi}_3) P_n^{(2\mu_{im}+2,0)}(\hat{\xi}_3) d\hat{\xi}_3 \\
=& \delta_{il} \mathbf{F}^i_{(jk)(mn)}
\end{aligned} \tag{C.43}$$

and

$$\int_f u^2 d\xi_1 d\xi_3 = \sum_{i=0}^{p} \left(\sum_{j=0}^{p} \sum_{k=0}^{p-\mu_{ij}} \sum_{m=0}^{p} \sum_{n=0}^{p-\mu_{im}} \mathbf{u}_{(ijk)} \mathbf{F}^i_{(jk)(mn)} \mathbf{u}_{(imn)} \right) \tag{C.44}$$

$\mathbf{F}$ is again block-diagonal. The spectral radii of the blocks $\mathbf{F}^i$ are difficult to compute in closed form, due to the complex integrals but mostly since these blocks are not rank 1 (except for the case $i = p$).

An engineering approach was used to circumvent this problem. Thereto each of the blocks was computed by numerical quadrature using the extension *Jacobi (v0.9.2)* (65) of *GNU Scientific Library (v1.14)* (47). Subsequently their spectrum was computed to determine $\rho(\mathbf{F}^i)$, The minimal and maximal values of the spectral radii, together with quadratic regressions are shown in Fig. C.1.

During the computations it was found that the rank of a block $\mathbf{F}^i$ is given by $p+1-i$. Hence the global rank of $\mathbf{F}$ is $\sum_{k=0}^{p+1} k = (p+1)(p+2)/2$. This rank corresponds to the dimension of $\mathcal{L}_p^{\mathfrak{P}}$ as it is restricted to the triangular face ƒ. Furthermore the spectral radius diminishes with i, leading to the smallest value for $i = p$.

The minimal value $\rho(\mathbf{F}^p)$ seems to fit the expression $(p+1)(2p+3)/4$ exactly. Upon close inspection one can confirm this result easily. Indeed, if $i = p$, then both μ_{ij} and μ_{im} are equal to p, while k and n only can have the value 0.

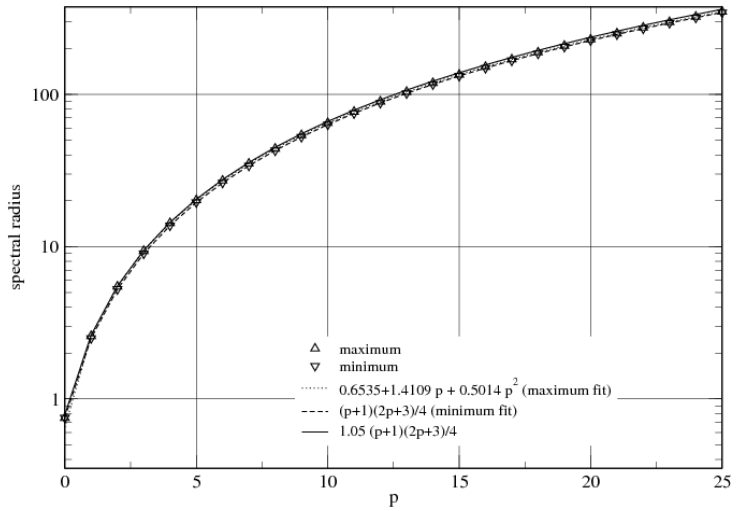


Figure C.1: Computed values of the minimal and maximal spectral radius of the blocks $\mathbf{F}^k$ for triangular faces embedded in a pyramid together with the quadratic curve fit.

Hence $\mathbf{F}^p$ has dimension $p+1$ and rank 1 since

$$\begin{aligned}\mathbf{F}^p_{(j0)(m0)} &= \frac{(2p+3)\sqrt{(2j+1)(2k+1)}}{2^{p+3}} \int_{-1}^{1} (1-\hat{\xi}_3)^{2p+1} P_0^{(2p+2,0)} P_0^{(2p+2,0)} d\hat{\xi}_3 \\ &= \frac{(2p+3)\sqrt{(2j+1)(2k+1)}}{2^{p+3}} \int_{-1}^{1} (1-\hat{\xi}_3)^{2p+1} d\hat{\xi}_3 \\ &= \frac{(2p+3)\sqrt{(2j+1)(2k+1)}}{4(p+1)}\end{aligned} \tag{C.45}$$

The spectral radius of $\mathbf{F}^p$ is then given by its trace

$$\rho(\mathbf{F}^p) = \sum_{j=0}^{p} \mathbf{F}^p_{(j0)(j0)} = \frac{(2p+3)}{4(p+1)} \sum_{j=0}^{p} (2j+1) = \frac{(p+1)(2p+3)}{4} \tag{C.46}$$

confirming the results obtained by numerical means.

Such an exact quadratic fit cannot be found for the maximal value. Yet in Fig. C.1 we notice that the difference between minimum and maximum spectral radii remains very small - less than 5 percent - for all of the orders ($p \leq 25$) considered. In fact, this difference seems so small that numerical error cannot be excluded.

As one really needs an expression that bounds, not approximates, the spectral radius, the following approximation is proposed

$$\int_f u^2 dS \leq 1.05 \frac{(p+1)(2p+3)}{3} \frac{\mathcal{A}(f)}{\mathcal{V}(e)} \int_e u^2 dV \tag{C.47}$$

C.6 The Pascal space on the pyramid

An orthonormal base for the modal space $\mathcal{L}_p$ on the pyramid is given by

$$\begin{aligned}
\psi_{ijk}^{\mathfrak{P}\prime} &= C'_{ijk} P_i\left(\frac{\xi_1}{1-\xi_3}\right)\ P_j\left(\frac{\xi_2}{1-\xi_3}\right)\ (1-\xi_3)^{i+j}\, P_k^{(2i+2j+2,0)}(2\xi_3-1) \\
C'_{ijk} &= \frac{1}{2}\sqrt{(2i+1)(2j+1)(2i+2j+2k+3)} \\
0 &\leq i+j+k \leq p
\end{aligned} \tag{C.48}$$

Quadrilateral face

On the quadrilateral face, the correlations between the functions $\psi_{ijk}^{\mathfrak{P}\prime}$ are given by

$$\begin{aligned}
\int_f \psi_{ijk}^{\mathfrak{P}\prime}\psi_{lmn}^{\mathfrak{P}\prime} d\xi &= (-1)^{k+n} C'_{ijk} C'_{lmn} \int_{-1}^{1} P_i(\xi_1)P_l(\xi_1)d\xi_1 \int_{-1}^{1} P_j(\xi_2)P_m(\xi_2)d\xi_2 \\
&= \delta_{il}\delta_{jm}(-1)^{k+n}\frac{2}{2i+1}\frac{2}{2j+1} C'_{ijk} C'_{lmn} \\
&= \delta_{il}\delta_{jm}(-1)^{k+n}\sqrt{2i+2j+2k+3}\sqrt{2i+2j+2n+3}
\end{aligned} \tag{C.49}$$

Again the blocks $\mathbf{F}^{ij}$ are found to be rank-one matrices, such that we can easily compute the inequality as

$$\begin{aligned}
\int_f u^2 d\xi &= \sum_i \sum_j \delta_{il}\delta_{jm} \sum_k \sum_n (-1)^{k+n}\sqrt{2i+2j+2k+3}\sqrt{2i+2j+2n+3} \\
&\leq \sum_i \sum_j \left(\sum_{k'=0}^{p-(i+j)} (2i+2j+3) + 2k' \right) \sum_k \mathbf{u}_{ijk}^2 \\
&= \sum_i \sum_j \left((p-(i+j)+1)(p+(i+j)+3) \right) \sum_k \mathbf{u}_{ijk}^2 \\
&\leq (p+1)(p+3) \sum_{ijk} \mathbf{u}_{ijk}^2
\end{aligned} \tag{C.50}$$

since the expression $(p-x+1)(p+x+3)$ attains its maximum for $x=0$, corresponding to $i=j=0$. Hence we find the same bound as for $\mathcal{L}_p^{\mathfrak{P}}$.

Triangular faces

Again we can take the face defined by $1-\xi_2-\xi_3=0$. On this face we find, using the Duffy transformation

$$\psi_{ijk}^{\mathfrak{P}\prime}\Big|_f = \frac{C'_{ijk}}{2^{i+j}} P_i(\hat{\xi}_1)(1-\hat{\xi}_3)^{i+j} P_k^{(2(i+j)+2,0)}(\hat{\xi}_3) \tag{C.51}$$

Now $\mathbf{F}$ is given by

$$\begin{aligned}\mathbf{F}_{(ijk)(lmn)} =& \frac{C'_{ijk}C'_{lmn}}{2^{i+j+l+m+2}} \int_{-1}^{1} P_i(\hat{\xi}_1)P_l(\hat{\xi}_1)d\hat{\xi}_1 \dots \\ & \dots \int_{-1}^{1} (1-\hat{\xi}_3)^{i+j+l+m+1} P_k^{(2(i+j)+2,0)}(\hat{\xi}_3)P_n^{(2(l+m)+2,0)}(\hat{\xi}_3)d\hat{\xi}_3 \\ =& \frac{C'_{ijk}C'_{lmn}}{2^{2i+j+m+2}} \frac{2\delta_{il}}{2i+1} \dots \\ & \dots \int_{-1}^{1} (1-\hat{\xi}_3)^{2i+j+m+1} P_k^{(2(i+j)+2,0)}(\hat{\xi}_3)P_n^{(2(i+m)+2,0)}(\hat{\xi}_3)d\hat{\xi}_3 \\ =& \delta_{il}\mathbf{F}^i_{(jk)(mn)}\end{aligned} \tag{C.52}$$

As for the Lagrange interpolation, a numerical computation of the spectral radii has been performed. This computation shows that the rank of the blocks $\mathbf{F}^i$ is given by $p+1-i$, and the spectral radius decreases - however much more drastically - with order. As the first block $\mathbf{F}^O$ is exactly the same as for the Lagrange interpolation, we find exactly the same value for $c_{\mathfrak{e},\mathfrak{f}}$.

APPENDIX D

NONLINEAR INSTABILITY OF QUADRATURE-FREE METHODS

D.1 Original formulation

Quadrature free integration was proposed by Atkins *et al.* (8) in the context of the DGM discretisation on simplex meshes of the *Linearized Euler Equations in stagnant flow*. In general, quadrature free integration can be applied without approximation if

- the system consists of *linear* convection equations with *constant coefficients*, *i.e.* the flux vector $\vec{f}$ is defined as

$$f_m^k = \sum_{n=1}^{N_v} \mathbf{F}_{mn}^k \tilde{u}_n \tag{D.1}$$

  whereby the flux Jacobians $\mathbf{F}^k$ are independent of the solution and position.

- the mesh consists of *simplices* (ie. nodes, lines, triangles and tetrahedra). Simplicial elements have linear coordinate mapping and hence the associated Jacobian is constant per element.

These assumptions allow us to perform quadrature as a preprocessing step, hence greatly reducing the number of operations involved in assembly. A very important remark is that *no approximation will be used, and the quadrature will be performed in preprocessing up to an arbitrary level of accuracy*.

Volume term

We can rewrite the convective volume contribution CV_{im} to the residual associated to shape function ϕ_i and variable m:

$$\begin{aligned} CV_{im} &= \sum_{q=1}^{N_\alpha} w_q \left(|\mathbf{J}_e| \sum_{k=1}^{d} \sum_{u=1}^{d} \frac{\partial \phi_i}{\partial \xi^u} \frac{\partial \xi^u}{\partial x^k} f_m^k \right)_{\alpha_q} \\ &= \sum_{u=1}^{d} \sum_{j=1}^{N_\phi} \left(\sum_{q=1}^{N_\alpha} w_q \left(\frac{\partial \phi_i}{\partial \xi^u} \phi_j \right)_{\alpha_q} \right) \cdot \left(|\mathbf{J}_e| \sum_{k=1}^{d} \frac{\partial \xi^u}{\partial x^k} \mathbf{F}_{mn}^k \mathbf{u}_{jn} \right) \\ &= \sum_{u=1}^{d} \sum_{j=1}^{N_\phi} \mathfrak{R}^u{}_{ij} \cdot \mathfrak{f}^u{}_{jm} \end{aligned} \tag{D.2}$$

Hence the quadrature free assembly of CV is a two-step procedure:

1. computation of the matrices $\mathfrak{f}^u$, storing the parametric fluxes evaluated *in the interpolation points*;
2. redistribution with the parametric convection matrices $\mathfrak{R}^u$.

Compared to the full quadrature version, the quadrature free version leads to a the following gains in terms of operations:

- solution collocation is avoided;
- the flux evaluation only has to be done for N_ϕ values instead of N_α;
- the redistribution is slightly cheaper, since $\mathfrak{R}^u \in \mathrm{R}^{N_\phi \times N_\phi}$ while $\mathfrak{R}^{*u} \in \mathrm{R}^{N_\phi \times N_\alpha}$

The contribution of the volume term CV to the residual Jacobian is then assembled as:

$$\frac{\partial CV_{im}}{\partial \mathbf{u}_{jn}} = \sum_{u=1}^{d} \sum_{j=1}^{N_\phi} \mathfrak{R}^u{}_{ij} \cdot \left(|\mathbf{J}_e| \sum_{k=1}^{d} \frac{\partial \xi^u}{\partial x^k} \mathbf{F}_{mn}^k \right) \tag{D.3}$$

In terms of number of operations, this linearisation is N_α times cheaper as its full quadrature counterpart, since per parametric coordinate ξ^u and variable combination (m, n) only one influence matrix $\mathfrak{R}^u$ needs to be added to the Jacobian, compared to one matrix $\mathfrak{R}^{*u}{}_q$ *per quadrature point* in the case of classical quadrature.

Interface convective term

The interface convective flux contribution CI for a set of linear equations through a simplex face (implying a constant normal $\vec{n}$) may be rewritten as follows:

$$\begin{aligned} CI^a_{im} &= \sum_{q=1}^{N_\beta} v_q \left(\phi^a_i \; \mathcal{H}_m \left(u^a, u^b, \vec{n} \right) \; |\mathbf{J}_f| \right)_{\beta_q} \\ &= \sum_{q=1}^{N_\beta} \left(v_q \phi^a_i \left(\mathbf{H}^+_{mn} \cdot \left(\sum_{j=1}^{N_\psi} \phi^a_j \mathbf{u}^a_{jn} \right) + \mathbf{H}^-_{mn} \cdot \left(\sum_{j=1}^{N_\psi} \phi^b_j \mathbf{u}^b_{jn} \right) \right) \right)_{\beta_q} \end{aligned} \tag{D.4}$$

$\mathbf{H}^+$ and $\mathbf{H}^-$ are the Jacobians of the Riemann flux $\mathcal{H}$ with respect to resp. the in- and outward state vector. The the projected flux Jacobian

$$\mathbf{F}^n = \sum_{k=1}^{d} \mathbf{F}^k n^k \tag{D.5}$$

is constant on the face, and we can define the upwind interface flux Jacobians $\mathbf{H}^+$ and $\mathbf{H}^-$ on the whole face based on its eigenvalue decomposition

$$\mathbf{F}^n = \mathbf{R}^n \cdot \mathbf{\Lambda}^n \cdot \mathbf{L}^n \tag{D.6}$$

where we define

$$\begin{aligned} \mathbf{\Lambda}^n &= diag(\lambda^n_i) \\ \mathbf{R}^n_{*i} &= \mathbf{r}^n_i \\ \mathbf{L}^n_{j*} &= \mathbf{l}^n_i \end{aligned} \tag{D.7}$$

with λ_i, $\mathbf{r}_i$ and $\mathbf{l}_i$ are resp. the i-th eigenvalue and the corresponding right and left eigenvector. We find

$$\begin{aligned} \mathbf{H}^{n+} &= \mathbf{R}^n \; \mathbf{\Lambda}^{n+} \; \mathbf{L}^n \\ \mathbf{H}^{n-} &= \mathbf{R}^n \; \mathbf{\Lambda}^{n+} \; \mathbf{L}^n \\ \mathbf{\Lambda}^{n+} &= diag\left(max(0, \lambda^n_i)\right) \\ \mathbf{\Lambda}^{n-} &= diag\left(min(0, \lambda^n_i)\right) \end{aligned} \tag{D.8}$$

Reformulating equation D.4 in the frame of reference of the face, where the restrictions of pairs of shape functions ϕ^a_j and ϕ^b_j coincide in ψ_j (note that the index j does not necessarily refer to the same shape functions in the elements a and b) we finally come up with

$$\begin{aligned} CI^a_{im} &= \sum_{j=1}^{N_\psi} \left(\sum_{q=1}^{N_\beta} \left(v_q \psi_i \psi_j \right)_{\beta_q} \right) \cdot \left(\mathbf{H}^{n+}_{mn} \cdot \mathbf{u}^a_{jn} + \mathbf{H}^{n-}_{mn} \cdot \mathbf{u}^b_{jn} \right) \\ &= \sum_{j=1}^{N_\psi} \mathfrak{R}^b{}_{ij} \cdot \mathbf{h}^{ab}{}_{jm} \end{aligned} \tag{D.9}$$

Again the assembly of CI is a two-step procedure, starting with the computation of the matrices of parametric fluxes $\mathbf{h}^{ab}$ followed by the redistribution.

The contribution of the interface term CI can be assembled as

$$\begin{aligned}\frac{\partial CI^a_{im}}{\partial \mathbf{u}^a_{jn}} &= \mathfrak{R}^b{}_{ij} \cdot \mathbf{H}^+_{mn} \\ \frac{\partial CI^a_{im}}{\partial \mathbf{u}^b_{jn}} &= \mathfrak{R}^b{}_{ij} \cdot \mathbf{H}^-_{mn}\end{aligned} \tag{D.10}$$

The quadrature free version is now N_β times cheaper than the full-quadrature version.

D.2 Extension to non-linear equations of state

Both formulations D.2 and D.4 suggest an extension to the non-linear case where the matrices of parametric fluxes $\mathfrak{f}^u$ and $\mathbf{h}$ are computed using the non-linear flux functions applied to the values in one interpolation point j only.

$$\begin{aligned}\mathfrak{f}^u{}_{jm} &= |\mathbf{J}_e| \sum_{k=1}^{d} \frac{\partial \xi^u}{\partial x^k} f^k_m(\mathbf{u}_j) \\ \mathbf{h}^{ab}{}_{jm} &= |\mathbf{J}_f| \, \mathcal{H}(\mathbf{u}^a_j, \mathbf{u}^b_j, \vec{n})\end{aligned} \tag{D.11}$$

This amounts to interpolating the fluxes in the same functional space $\mathcal{V}$ as the solution, where the expansion weights are defined by injection. The alternative approach, *i.e.* Galerkin projection, would result in an evaluation cost similar to classical quadrature.

The contribution of the quadrature free volume term to the Jacobian of the residual are then given by

$$\frac{\partial CV_{im}}{\partial \mathbf{u}_{jn}} = \sum_{u=1}^{d} \mathfrak{R}^u{}_{ij} \cdot \sum_{k=1}^{d} |\mathbf{J}_e| \frac{\partial \xi^u}{\partial x^k} \left(\frac{\partial f^k_m}{\partial u_n} \right)_{\mathbf{u}_j} \tag{D.12}$$

and of the interface terms by

$$\begin{aligned}\frac{\partial CI^a_{im}}{\partial \mathbf{u}^a_{jn}} &= \mathfrak{R}^b{}_{ij} \cdot \left(\frac{\partial \mathcal{H}_m}{\partial u^+_n} \right)_{\mathbf{u}_j} \\ \frac{\partial CI^a_{im}}{\partial \mathbf{u}^b_{jn}} &= \mathfrak{R}^b{}_{ij} \cdot \left(\frac{\partial \mathcal{H}_m}{\partial u^-_n} \right)_{\mathbf{u}_j}\end{aligned} \tag{D.13}$$

D.3 Spurious modes

This chapter refines the explanation of the quadrature-free instability mechanism as elaborated in (57).

Decoupling mechanism

In the quadrature free approach for non-linear convective equations (see D.2) any of the state vector expansion weights $\mathbf{u}_j$ is used for the evaluation of only one set of fluxes f^k only. In case the flux Jacobian $\mathbf{F}^k$ is singular for a given $\mathbf{u}_j$, then the corresponding flux f^k does not depend on (part of) $\mathbf{u}_j$. Since $\mathbf{u}_j$ is not used for the evaluation of any other terms, part of $\mathbf{u}_j$ has no impact on the residual and decouples itself from the rest of the unknowns (this phenomenon is also referred to as a *spurious mode* in finite volume/finite difference terminology).

To understand the decoupling a brief recapitulation of characteristic theory is needed. Consider again the eigenvalue decomposition of $\mathbf{F}^k$

$$\begin{aligned} \mathbf{F}^k &= \mathbf{R}^k \mathbf{\Lambda}^k \mathbf{L}^k \\ \mathbf{\Lambda}^n &= diag(\lambda_i^k) \\ \mathbf{R}^n_{*i} &= \mathbf{r}_i^k \\ \mathbf{L}^n_{j*} &= \mathbf{l}_i^k \end{aligned} \tag{D.14}$$

Using this decomposition, any variation of the solution may then be decomposed in the direction k as as a sum of waves with amplitude α_i^k

$$\delta\tilde{u} = \sum_{i=1}^{N_v} \alpha_i \mathbf{r}_i^k = \sum_{i=1}^{N_v} \left(\mathbf{l}_i^k \cdot \delta\tilde{u}\right) \mathbf{r}_i^k \tag{D.15}$$

where each of the waves is convected with speed λ_i^k along direction k. In terminology of hyperbolic PDE λ_i^k is referred to as a convection speed, and $\mathbf{r}_i^k$ as a characteristic direction.

Now if $\mathbf{F}^k$ is singular, some of the eigenvalues λ_i^k are zero. For the corresponding right eigen vector $\mathbf{r}_i^k$ one finds

$$\mathbf{F}^k \cdot \mathbf{r}_i^k = 0 \tag{D.16}$$

Obviously the right eigenvectors corresponding to zero eigenvalues span the kernel of $\mathbf{F}^k$, *i.e.* the space of vectors which map to 0 after multiplication with $\mathbf{F}^k$. Returning to the fluxes, one finds

$$\mathbf{f}^k(\mathbf{u} + \alpha\mathbf{r}^i) \approx \mathbf{f}^k(\mathbf{u}) + \alpha\mathbf{F}^k\mathbf{r}^i = \mathbf{f}^k(\mathbf{u}) \tag{D.17}$$

this implies that the solution may vary along the corresponding right eigenvectors $\mathbf{r}^i$ *without modifying the flux* $\mathbf{f}^k$. Even in this case the singularity of (some of) the $\mathbf{F}^k$ does not lead to decoupling, *unless the kernels of the different* $\mathbf{F}^k$ *have a non-trivial intersection*.

Application to the Euler equations

Now consider the Euler equations. Although the system is composed of 5 equations, $\mathbf{F}^k$ only has 3 distinct eigenvalues

$$\begin{aligned} \lambda_u^k &= u^k \\ \lambda_{\pm a}^k &= u^k \pm a \end{aligned} \tag{D.18}$$

where u^k is the component of the velocity in Cartesian direction k and a is the speed of sound. The convective speed λ_u is a triple eigenvalue, corresponding to two waves δc_l conveying perpendicular velocity components, and a third wave δc_s conveying entropy. The eigenvalues $\lambda_\pm^k$ correspond to acoustic waves $\delta c_\pm$. Expressed in the variations of the variables $[\rho\ u^1\ u^2\ u^3\ p]$ the following expressions for the characteristic wave strengths are found (see *a.o.* (61)):

$$\begin{aligned} \delta c_s &= \delta\rho - \frac{\delta p}{a^2} \\ \delta c_l &= \delta u^l\ ,\ l \neq k \\ \delta c_{\pm a} &= \delta u^k \mp \frac{\delta p}{\rho a} \end{aligned} \tag{D.19}$$

The system of the Euler equations has two types of singular points.

- in *sonic points*, one of the acoustic speeds is zero, however, only in a particular direction. This means that this type of singular point does not result in decoupling since only one $\mathbf{F}^k$ is singular;
- in *stagnation points*, all of the convection speeds u^k are zero, and hence all of the $\mathbf{F}^k$ are singular. Moreover, one of the corresponding characteristics, namely the entropy characteristic δc_s, is common to all $\mathbf{F}^k$. Hence the latter may vary without affecting the residual and the corresponding variation of the solution is decoupled.

Implications for quadrature techniques

Following the above discussion one can conclude that the quadrature free approach for the DGM discretisation of the Euler equations will *always be unstable in stagnation points*, as the residual allows for spurious modes, and the associated residual Jacobian is singular.

For explicit methods, a cure can be found in the application of a limiter that reverts the computed solution to reasonable values; however this approach is far from ideal, and difficult to integrate in an implicit iterative strategy. Since this approach is implicitly present in RKDG methods, the instability problem has not been acknowledged until recently. Given the singularity of the residual Jacobian, no such obvious cure exists for implicit iterative strategies.

Obviously the same problem occurs for *collocation spectral element methods*, where interpolation points coincide with quadrature points. Classical collocation quadrature avoids this problem, since any $\mathbf{u}_j$ is used for a flux evaluation in many, if not all quadrature points. This greatly reduces the risk of $\mathbf{u}_j$ becoming decoupled.

BIBLIOGRAPHY

[1] Advanced tools to design future jet engines. http://www.prace-project.eu/Advanced-tools-to-design-future-jet-engines.

[2] 1st international workshop on High-Order CFD Methods. http://zjw.public.iastate.edu/hiocfd.html, 7-8 January 2012. Organised at the occasion of the 50^{th} AIAA ASM, Nashville, Tenessee.

[3] 2nd international workshop on High-Order CFD Methods. http://www.dlr.de/as/desktopdefault.aspx/tabid-8170/13999_read-35550/, 28-29 May 2013. Köln, Germany.

[4] M. Abramowitz and I.A. Stegun, editors. *Handbook of Mathematical Functions*. Dover Publications, New York, 10th edition, 1972.

[5] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[6] M. Arioli and I.S. Duff. Using FGMRES to obtain backward stability in mixed precision. *Electronic Transactions on Numerical Analysis*, 33:31–44, 2009.

[7] D.N. Arnold, F. Brezzi, B. Cockburn, and L.D. Marini. Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems. *SIAM Journal of Numerical Analysis*, 39(5):1749–1779, 2002.

[8] H.L. Atkins and C.-W. Shu. Quadrature-free implementation of the Discontinuous Galerkin method for Hyperbolic Equations. Technical Report 96-51, ICASE, 1996.

[9] M. Baboulin, A. Buttari, J. Dongarra, J. Kurzak, J. Langou, J. Langou, P. Luszczek, and S. Tomov. Accelerating scientific computations with mixed precision algorithms. *Computer Physics Communications*, 180(12):2526–2533, 2009.

[10] F. Bassi, L. Botti, A. Colombo, and S. Rebay. Agglomeration based discontinuous Galerkin discretization of the Euler and Navier-Stokes equations. *Computers and Fluids*, 61:77–85, 2012.

[11] F. Bassi, N. Franchina, A. Ghidoni, and S. Rebay. A numerical investigation of a spectral-type nodal collocation discontinuous Galerkin approximation of the Euler and Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 2012. published online (doi 10.1002/fld.3713).

[12] F. Bassi and S. Rebay. High-Order Accurate Discontinuous Galerkin Finite Element Solution of the 2D Euler Equations. *Journal of Computational Physics*, 138(2):251–285, 1997.

[13] F. Bassi and S. Rebay. *Discontinuous Galerkin methods: theory, computation and applications*, volume 11 of *Lecture Notes in Computational Science and Engineering*, chapter GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations, pages 197–208. Springer Verlag, 2000.

[14] P. Bastian and V. Reichenberger. Multigrid for higher order discontinuous Galerkin finite elements applied to groundwater flow. Technical Report 2000-37, SFB 359, 2000.

[15] J.-P. Bérenger. A Perfectly Matched Layer for the Absorption of Electromagnetic Waves. *Journal of Computational Physics*, 114(2):185–200, 1994.

[16] M. Bergot, G. Cohen, and M. Duruflé. High-Order Finite Elements for Hybrid Meshes Using New Nodal Pyramidal Elements. *Journal of Scientific Computing*, 42(3):345:381, 2010.

[17] A. Bermudez, L. Hervella-Nieto, A. Prieto, and R. Rodriguez. An optimal perfectly matched layer with unbounded absorbing function for time-harmonic acoustic scattering problems. *Journal of Computational Physics*, 223(2):469–488, 2007.

[18] R.T. Beyer. *Nonlinear Acoustics*. Acoustical Society of America, 2nd edition, 1997.

[19] P. Birken, G. Gassner, M. Haas, and C.-D. Munz. A new class of preconditioners for discontinuous Galerkin methods for unsteady 3D Navier-Stokes equations: ROBO-SGS. *Journal of Computational Physics (preprint, submitted)*, 2012.

[20] S. L. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and C.R. Whaley. An updated set of Basic Linear Algebra Subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28(2):135–151, 2002.

[21] R. Boucheron, H. Bailliet, and J.-C. Valiere. Analytical solution of multimodal acoustic propagation in circular ducts with laminar mean flow profile. *Journal of Sound and Vibration*, 292(3-5):504–518, 2006.

[22] D. Braess. *Finite Elements: Theory, fast solvers and applications in solid mechanics*. Cambridge University Press, 1997.

[23] A. Brandt. Multigrid guide. http://www.wisdom.weizmann.ac.il/~achi/, 2011. Revised from 1984 version.

[24] W. L. Briggs, V.E. Henson, and S. F. McCormick. *A multigrid tutorial (2nd ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.

[25] H. Van Brummelen. Private communication.

[26] E. Burman and A. Ern. Continuous interior penalty hp-finite element methods for advection and advection-diffusion equations. *Mathematics of Computation*, 76(259):1119–1140, 2007.

[27] C.D. Cantwell, S.J. Sherwin, R.M. Kirby, and P.H.J. Kelly. From h to p efficiently: Strategy selection for operator evaluation on hexahedral and tetrahedral elements. *Computers and Fluids*, 43(1):23 – 28, 2011. Symposium on High Accuracy Flow Simulations. Special Issue Dedicated to Prof. Michel Deville, Symposium on High Accuracy Flow Simulations.

[28] C. Carton de Wiart and K. Hillewaert. DNS and ILES of transitional flows around a SD7003 airfoil using a high order Discontinuous Galerkin Method. In *Seventh International Conference on Computational Fluid Dynamics (ICCFD7)*, Big Island, Hawaii, July 9-13 2012.

[29] C. Carton de Wiart, K. Hillewaert, and P. Geuzaine. DNS of a Low Pressure Turbine Blade computed with the Discontinuous Galerkin Method (asme-gt2012-68900). In *ASME Turbo Expo 2012 Turbine Technical Conference and Exposition*, Copenhagen, Denmark, June 2012. ASME.

[30] C. Carton de Wiart, K. Hillewaert, P. Geuzaine, R. Luccioni, L. Bricteux, G. Coussement, and G. Winckelmans. Assessment of les modeling within a high order discontinuous galerkin solver. In *9th International ERCOFTAC Symposium on Engineering Turbulence Modelling and Measurements*, Thessaloniki, Greece, 6-8 june 2012. ERCOFTAC.

[31] Q. Chen and I. Babuška. Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle. *Computer Methods in Applied Mechanics and Engineering*, 128(1):405–417, 1995.

[32] Q. Chen and I. Babuška. The Optimal Symmetrical Points for Polynomial Interpolation of Real Functions in the Tetrahedron. Technical Report 1186, Maryland University, College Park Institute For Physical Science And Technology, 1995.

[33] N. Chevaugeon, K. Hillewaert, X. Gallez, P. Ploumhans, and J.-F. Remacle. Optimal numerical parameterization of discontinuous Galerkin method applied to wave propagation problems. *Journal of Computational Physics*, 223(1):188–207, 2007.

[34] P.G. Ciarlet. *The Finite-Element Method for Elliptic Problems*. North-Holland, The Netherlands, 1978.

[35] B. Cockburn. *Higher-Order Methods for Computational Physics*, volume 9 of *Lecture Notes in Computational Science and Engineering*, chapter Discontinuous Galerkin Methods for Convection-Dominated Problems, pages 69–224. Springer, 1999.

[36] A. Crivellini and F. Bassi. An implicit matrix-free Discontinuous Galerkin solver for viscous and turbulent aerodynamic simulations. *Computers and Fluids*, 50(1):81–93, 2011.

[37] Laszlo Tibor Diosady. A Linear Multigrid Preconditioner for the solution of the Navier-Stokes Equations using a Discontinuous Galerkin Discretization. Master's thesis, MIT, 2007.

[38] L.T. Diosady and D.L. Darmofal. Discontinuous Galerkin solutions of the Navier-Stokes Equations using Linear Multigrid Preconditioning. In *18th AIAA Computational Fluid Dynamics Conference*, Miami, 2007. AIAA.

[39] L.T. Diosady and D.L. Darmofal. Preconditioning methods for discontinuous Galerkin discretizations of the Navier-Stokes equations. *Journal of Computational Physics*, 228(11):3917–3935, 2009.

[40] M. Drosson, K. Hillewaert, and J.-E. Essers. Stability and boundary resolution analysis of the discontinuous galerkin method applied to the reynolds-averaged navier-stokes equations using the Spalart-Allmaras model. *In review for the SIAM Journal on Scientific Computing*, 2011.

[41] M. Drosson, K. Hillewaert, and J.-E. Essers. On the stability of the symmetric interior penalty method - application to 3D compressible RANS simulations of a high lift cascade flow. *In review for the Journal of Computational and Applied Mathematics*, 2012.

[42] Y. Epshteyn and B. Rivière. Estimation of penalty parameters for symmetric interior penalty Galerkin methods. *Journal of Computational and Applied Mathematics*, 206(2):843–872, 2007.

[43] G.D. Farin. *Handbook of Grid Generation*, chapter 28: Computer-Aided Geometric Design. CRC Press, 1999.

[44] D.R. Ferguson. *Handbook of Grid Generation*, chapter 27: Spline Geometry : A Numerical Analysis View. CRC Press, 1999.

[45] K.J. Fidkowski. High-Order Discontinuous Galerkin Multigrid Solver for Aerodynamic Applications. Master's thesis, MIT, 2004.

[46] K.J. Fidkowski, T.A. Oliver, J. Lu, and D.L. Darmofal. p-Multigrid solution of high-order discontinuous Galerkin discretizations of the Navier-Stokes equations. *Journal of Computational Physics*, 207(1):92–113, 2005.

[47] M. Galassi et al. GSL - the GNU Scientific Library. http://www.gnu.org/software/gsl.

[48] V. Girault, B. Rivière, and M. Wheeler. A Discontinuous Galerkin Method with Non-Overlapping Domain Decomposition for the Stokes and Navier-Stokes Problems. *Mathematics of Computation*, 74:53–84, 2004.

[49] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[50] K. Harriman, D. J. Gavaghan, and E. Suli. The importance of adjoint consistency in the approximation of linear functionals using the discontinuous galerkin finite element method. Technical Report 04/18, Oxford Numerical Analysis Group, 2004.

[51] R. Hartmann and P. Houston. Symmetric interior penalty DG method for the compressible Navier-Stokes equations I : formulation. *International Journal of Numerical Analysis and Modeling*, 3(1):1–20, 2006.

[52] Ralf Hartmann. *Adaptive Finite Element Methods for the Compressible Euler Equations*. PhD thesis, Ruprechts-Karls-Universität Heidelberg, 2002.

[53] B.T. Helenbrook, D.J. Mavriplis, and H.L. Atkins. Analysis of "p"-Multigrid for Continuous and Discontinuous Finite Element Discretizations. In *The 16th AIAA Computational Fluid Dynamics Conference*, Orlando, Florida, 2003. AIAA. AIAA-2003-3989.

[54] J.S. Hesthaven. From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex. *SIAM Journal of Numerical Analysis*, 35(2):655–676, 1998.

[55] J.S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods; Algorithms, Analysis and Applications*. Text in Applied Mathematics. Springer Verlag, 2008.

[56] K. Hillewaert. *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*, volume 113 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, chapter 2: Exploiting data locality in the DGM discretisation for optimal efficiency, pages 11–24. Springer, 2010.

[57] K. Hillewaert, N. Chevaugeon, P. Geuzaine, and J.-F. Remacle. Hierarchic multigrid iteration strategy for the discontinuous Galerkin solution of the steady Euler equations. *International Journal for Numerical Methods in Fluids*, 51(9-10):1157 – 1176, 2006.

[58] K. Hillewaert, J.-F. Remacle, N. Chevaugeon, and P. Geuzaine. Discontinuous Galerkin Methods. Implementation Issues. In *VKI 35th CFD/ADIGMA course on very high order discretization methods*, St-Genesius-Rode, 2008.

[59] K. Hillewaert, J.-F. Remacle, N. Cheveaugeon, P.-E. Bernard, and P. Geuzaine. Analysis of a hybrid p-multigrid method for the Discontinuous Galerkin Discretisation of the Euler Equations. In P. Wesseling, E. Oñate, and J. Périaux, editors, *ECCOMAS CFD Conference*, 2006.

[60] K. Hillewaert, J.-F. Remacle, and M. Drosson. Sharp constants in the hp-finite element trace inverse inequality for standard functional spaces on all element types in hybrid meshes. *In review fo the SIAM Journal on Numerical Analysis*, 2011.

[61] C. Hirsch. *Numerical Computation of Internal and External Flows, Volume 2: Computational Methods for Inviscid and Viscous Flows*. Numerical Methods in Engineering. Wiley, 1988.

[62] F.Q. Hu. On absorbing boundary conditions for Linearized Euler Equations by a Perfectly Matched Layer. *Journal of Computational Physics*, 129(1):201–219, 1996.

[63] F.Q. Hu. A perfectly matched layer absorbing boundary condition for linearized euler equations with a non-uniform mean flow. *Journal of Computational Physics*, 208(2):469–492, 2005.

[64] H.T. Huynh. A flux reconstruction approach to high-order schemes including discontinuous galerkin methods. In *AIAA Computational Fluid Dynamics Meeting*, 2007.

[65] P. Jabardo. Jacobi, a library to compute Jacobi polynomials and Gauss-Jacobi quadrature related stuff. http://www.network-theory.co.uk/download/gslextras/Jacobi/jacobi-0.9.2.tar.gz.

[66] G. Jiang and C.-W. Shu. On a cell entropy inequality for discontinuous Galerkin methods. *Mathematics of Computation*, 62(206):531–538, 1994.

[67] G. Kanschat. Preconditioning methods for local discontinuous Galerkin discretizations. *SIAM Journal of Scientific Computing*, 25(3):815–831, 2003.

[68] G. Kanschat. Multi-level methods for discontinuous Galerkin FEM on locally refined meshes. *Computers & Structures*, 84(28):2437–2445, 2004.

[69] G. Karypis and V. Kumar. METIS: A software package for partitioning unstructured graphs, partitioning meshes and computing fill-reducing orderings of sparse matrices, version 4.0. Technical report, University of Minnesota, Dept. of Computer Science and Engineering, 1998.

[70] C.M. Klaij, M.H. van Raalte, H. van der Ven, and J.J.W van der Vegt. h-multigrid for space-time discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *Journal of Computational Physics*, 227(2):1024–1045, 2007.

[71] D. Kopriva and G. Gassner. On the Quadrature and Weak Form Choices in Collocation Type Discontinuous Galerkin Spectral Element Methods. *Journal of Scientific Computing*, 44(2):136–155, 2010.

[72] L.I.G. Kovasznay. Laminar flow behind a two-dimensional grid. *Mathematical Proceedings of the Cambridge Philosophical Society*, 44:58–62, 1948. doi:10.1017/S0305004100023999.

[73] Jonathan Lambrechts. *Finite Element Methods for Coastal Flows: Application to the Great Barrier Reef*. PhD thesis, Université catholique de Louvain, 2011.

[74] R.J. Leveque. *Numerical Methods for Conservation Laws*. Lectures in Mathematics, ETH Zürich. Birkhäuser Verlag, 1992.

[75] Y. Liu, M. Vinokur, and Z.-J. Wang. Spectral difference method for unstructured grids I: basic formulation. *Journal of Computational Physics*, 216(2):780801, 2006.

[76] Y. Liu, M. Vinokur, and Z.-J. Wang. Spectral (finite) volume method for conservation laws on unstructured grids V: extension to three-dimensional systems. *Journal of Computational Physics*, 212(2):454472, 2006.

[77] H. Luo and C. Pozrikidis. A Lobatto interpolation grid in the tetrahedron. *IMA Journal of Applied Mathematics*, 71(2):298–313, 2006. doi:10.1093/imamat/hxh111.

[78] Emilie Marchandise. *Simulation of three-dimensional two-phase flows: coupling of a stabilized finite element method with a discontinuous level set approach*. PhD thesis, Université catholique de Louvain, 2006.

[79] G. May and A. Jameson. A spectral difference method for the euler and navier-stokes equations on unstructured meshes (aiaa 2006-304). In *44th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 9 - 12 January 2006.

[80] C.R. Nastase and D.J. Mavriplis. High-order discontinuous Galerkin methods using an hp-multigrid approach. *Journal of Computational Physics*, 213(1):330–357, 2006.

[81] J.A. Nitsche. Uber ein variationsprinzip zur Lösung dirichlet-problemen bei verwendung von teilraumen, die keinen randbedingungen unterworfen sind. *Abh. Math. Sem. Univ. Hamburg*, 36:9–15, 1971.

[82] T.A. Oliver. Multigrid solution for High-Order Discontinuous Galerkin Discretization of the Compressible Navier-Stokes Equations. Master's thesis, MIT, 2004.

[83] P.-O. Persson. Scalable Parallel Newton-Krylov Solvers for Discontinuous Galerkin Discretizations. AIAA-2009-606. In *47th AIAA Aerospace Sciences Meeting and Exhibit*, January 2009.

[84] P.-O. Persson and J. Peraire. An efficient low memory implicit DG algorithm for time dependent problems (AIAA-2006-0113). In *44th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, 2006. AIAA.

[85] P.-O. Persson and J. Peraire. Newton-GMRES Preconditioning for the Discontinuous Galerkin Discretizations of the Navier-Stokes Equations. *SIAM Journal of Scientific Computing*, 30(6):2709–2733, 2008.

[86] F. Pochet, K. Hillewaert, P. Geuzaine, J.-F. Remacle, and E. Marchandise. A 3D strongly coupled implicit discontinuous Galerkin level set-based method for modeling two-phase flows. *In review for Computers and Fluids*, 2012.

[87] N. Poletz, A. François, and K. Hillewaert. Multiphysics welding simulation model. *International Journal of Material Forming*, 1(1):1047–1050, 2008.

[88] A.N. Rahmouni. An algebraic method to develop well-posed PML models. *Journal of Computational Physics*, 197(1):99–115, 2004.

[89] P. Rasetarinera and M.Y. Hussaini. An Efficient Implicit Discontinuous Spectral Galerkin Method. *Journal of Computational Physics*, 172(2):718–738, 2001.

[90] B.D. Reddy. *Introductory functional analysis*. Number 27 in Texts in Applied Mathematics. Springer, 1997.

[91] B. Rivière. *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations - Theory and Implementation*. Society for Industrial and Applied Mathematics, 2008.

[92] Y. Saad. *Iterative methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2nd edition, 2003.

[93] K. Shahbazi. An explicit expression for the penalty parameter of the interior penalty method (Short note). *Journal of Computational Physics*, 205(2):401–407, 2005.

[94] K. Shahbazi, D.J. Mavriplis, and N. K. Burgess. Multigrid algorithms for high-order discontinuous galerkin discretizations of the compressible navier-stokes equations. *Journal of Computational Physics*, 228(21):7917–7940, 2009.

[95] S.J. Sherwin and G.E. Karniadakis. Tetrahedral hp Finite Elements: Algorithms and Flow Simulations. *Journal of Computational Physics*, 124(1):14 – 45, 1996.

[96] P. Solin, K. Segeth, and I. Dolezel. *Higher Order Finite Element Methods*. Studies in Advanced Mathematics. Chapman and Hall/CRC, 2004.

[97] Y. Sun, Z.-J. Wang, and Y. Liu. Spectral (finite) volume method for conservation laws on unstructured grids VI: Extension to viscous flow. *Journal of Computational Physics*, 215(1):4158, 2006.

[98] M.A. Taylor, B.A. Wingate, and R.E. Vincent. An algorithm for computing Fekete points in the triangle. *SIAM Journal for Numerical Analysis*, 38(5):1707–1720, 2000.

[99] Pietro Tesini. *An h-Multigrid Approach for High-Order Discontinuous Galerkin Methods*. PhD thesis, Universita degli studi di Bergamo - Dipartimento di Ingegneria Industriale, 2008.

[100] P. Tucker. Computation of unsteady turbomachinery flows: Part 1 - progress and challenges. *Progress in Aerospace Sciences*, 47(7):522–545, 2011.

[101] P. Tucker. Computation of unsteady turbomachinery flows: Part 2 - LES and hybrids. *Progress in Aerospace Sciences*, 47(7):546–569, 2011.

[102] J.J.W van der Vegt and S. Rhebergen. hp-multigrid as smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows. Part I. Multilevel analysis. *Journal of Computational Physics*, 231(22):7537–7563, 2012.

[103] J.J.W van der Vegt and S. Rhebergen. hp-multigrid as smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows. Part II. Optimization of the Runge-Kutta smoother. *Journal of Computational Physics*, 231(22):7564–7583, 2012.

[104] J.J.W. van der Vegt and H. van der Ven. Space Time Discontinuous Galerkin Finite Element Method with Dynamic Grid Motion for Inviscid Compressible Flows: I. General Formulation. *Journal of Computational Physics*, 182(2):546–585, 2002.

[105] P.E. Vincent, P. Castonguay, and A. Jameson. A New Class of High-Order Energy Stable Flux Reconstruction Schemes. *Journal of Scientific Computing*, 47(1):50–72, 2011.

[106] P.E. Vincent and A. Jameson. Facilitating the adoption of unstructured high order methods amongst a wider community of fluid dynamicists. *Mathematical modeling of Natural Phenomena*, 6(3):97–140, 2011.

[107] Z.-J. Wang. Spectral (finite) volume method for conservation laws on unstructured grids: I. basic formulation. *Journal of Computational Physics*, 178(2):665–697, 2002.

[108] Z.-J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H.T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal. High-order CFD methods: Current status and perspectives. *in review for the International Journal for Numerical Methods in Fluids*, 2012.

[109] T. Warburton and J.S. Hesthaven. On the constants in hp-finite element trace inverse inequalities. *Computer Methods in Applied and Mechanical Engineering*, 192(25):2765–2773, 2003.

[110] P. Wesseling. *An Introduction to Multigrid Methods*. Pure and Applied Mathematics. Wiley, 1992.

[111] M.F. Wheeler. An elliptic collocation-finite element method with interior penalties. *SIAM Journal of Numerical Analysis*, 15:152–161, 1978.