Aspects of adaptive mesh refinement in the spectral element method

by

Nicolas Offermans

June 2019
Technical Reports
Royal Institute of Technology
Department of Mechanics
SE-100 44 Stockholm, Sweden

Akademisk avhandling som med tillstånd av Kungliga Tekniska Högskolan i Stockholm framlägges till offentlig granskning för avläggande av teknologie doktorsexamen onsdag den 12 Juni 2019 kl 10:15 i sal F3, Kungliga Tekniska Högskolan, Lindstedtsvägen 26, Stockholm.

TRITA-MEK Technical report 2019:28
ISRN KTH/MEK/TR–19/28–SE
ISBN 978-91-7873-223-4

Universitetsservice US–AB, Stockholm 2019

Aspects of adaptive mesh refinement in the spectral element method

Nicolas Offermans
Linné FLOW Centre, KTH Mechanics, Royal Institute of Technology
SE-100 44 Stockholm, Sweden

Abstract

This thesis deals with the improvement of the efficiency of numerical simulations in computational fluid dynamics. Some of the limitations of current algorithms on future exascale supercomputers are addressed with the main goal of using adaptive mesh refinement for the simulation of turbulent and three-dimensional flows. The comparison between two different error estimators is also investigated. The framework considered all throughout this thesis is Nek5000, a highly parallel code based on the spectral element method.

First, the strong parallel scaling of Nek5000 is studied on three petascale machines in order to assess the scalability of the code and identify the current bottlenecks. It is found that the strong scaling limit ranges between $5,000$ and $220,000$ gridpoints per core depending on the machine and the case. The need for synchronized and low latency communication for fast computations is confirmed. It is also shown that, on a single core, Nek5000 is memory limited rather than compute limited.

Then, a new method for the coarse grid part of the preconditioner for the pressure equation is implemented, which represents a significant improvement compared to existing solvers. We use an algebraic multigrid method from the Hypre library to perform the setup and solver parts on the fly and fully in parallel. The setup phase only amounts to a few percents of the wall clock time for a single timestep of the solver and is therefore negligible; this is a requirement for adaptive simulations, where the setup must be performed after each adaptation. In addition, the new solver is shown to be suitable for large and complex simulations in three dimensions.

Finally, the main objective of this work is to perform simulations with adaptive mesh refinement to achieve error control and efficient use of computational resources. We develop adjoint error estimators based on the dual-weighted residuals method and also consider more traditional a posteriori error indicators for comparison. Adaptive simulations are performed on test cases of increasing complexity: the steady flow in a lid-driven cavity in 2D and 3D, the steady flow past a 2D cylinder and the turbulent flow inside a constricted periodic channel in 3D. It is concluded that adjoint error estimators are preferred for flows with a strongly convective nature, while the more straightforward spectral error indicators are sufficient otherwise. Moreover, we perform an adaptive simulation of the turbulent flow past a NACA4412 profile at Reynolds number $Re = 850,000$, using the spectral error indicators. In comparison to cases with

a fixed mesh, the aspect ratio of grid elements in the far field remains low, the convergence of the pressure solver is significantly sped up and the much larger computational domain allows true free-stream boundary conditions and thus makes the results less dependent on the boundary conditions.

Key words: Error estimators, mesh refinement, adaptivity, spectral element method, algebraic multigrid method, NACA4412, periodic hill case.

Aspekter av adaptiv nätförfining i spektralelementmetoden

Nicolas Offermans
Linné FLOW Centre, KTH Mekanik, Kungliga Tekniska Högskolan
SE-100 44 Stockholm, Sverige

Sammanfattning

Den här avhandlingen syftar till att utveckla metoder för att öka effektiviteten av strömningsmekaniska simuleringar. Arbetet visar också vissa begränsningarna hos nuvarande algoritmer om de ska användas på framtida superdatorer i exaskala. Adaptiv nätförfining används i den här avhandlingen för att simulera tredimensionella turbulenta strömningar. Dessutom jämförs två stycken olika metoder för att uppskatta felet av lösningen till Navier–Stokes ekvationer. Hela arbetet baseras på koden Nek5000, som använder den spektrala elementmetoden.

Inledningsvis undersöker vi den starka parallela skalbarheten av Nek5000 på tre superdatorer i petaskala för att bedöma kodens skalbarhet och identifiera befintliga flaskhalsar. Det visas att den starka skalningen per kärna för koden är begränsad till mellan $5,000$ och $220,000$ frihetsgrader, beroende på superdator och fall. Det är även bekräftat att synkroniserad kommunikation med liten fördröjning är nödvändigt för att uppnå effektiva strömningsmekaniska simuleringar. Nek5000 är minnesbegränsat på en enskild kärna.

Sedan implementeras en ny metod som förbättrar förkonditioneringen av tryckequationen på den grova delen av nätet. Vi använder en algebraisk flernätsmetod från Hypre för att automatiskt sätta upp lösningsmatriserna och lösa algoritmen på parallella datasystemer. Tiden som krävs för att initiera ett tidsteg är försumbar jämfört med beräkningstiden; det här är ett krav för adatiptiva simuleringar som behöver initiera efter varje adaption av nätet. Dessutom bekräftas det att den nya lösaren är lämplig för att simulera stora och komplexa tredimensionella strömningsproblem.

Huvudsyftet med arbetet att utföra simuleringar med adaptiva nätförfiningar för att kunna kontrollera felet på lösningen och att effektivt använda beräkningsresurser. Vi utvecklar adjunkta feluppskattningar baserat på dubbelviktade residualmetoden och jämför också med traditionella metoder som uppskattar felet av lösningen i efterhand (eng. spectral error indicators). Adaptiva nätförfiningssimuleringar genomförs på testfall som blir mer och mer komplexa: en stationär strömning i en kavitet med en rörlig rand (*eng.* lid-driven cavity) både två- och tredimensionellt, stationärt flöde kring en tvådimensionell cylinder och turbulent tredimensionell kanalströmningen i en periodisk domän. Slutsatsen är att adjunkta feluppskattningar föredras för flöden där konvektion dominerar, medan traditionella metoder för att uppskatta felet är tillräckligt bra för alla andra fall. Vi genomför även en adaptiv simulering av det turbulenta flödet över en NACA4412 vingprofil vid ett Reynoldstal $Re = 850,000$,

där vi använder traditionella metoder för att uppskatta felet. Resultaten från simuleringen har jämförts med simuleringar på ett nät utan adaptiv förfining, slutsatsen är en snabbare konvergens för att lösa ekvationen för trycket och att resultaten blir mindre beroende av randvillkoren på grund av en större domän, som tillåter korrekta friströmsrandvillkor.

Nyckelord: Feluppskattning, adaptiv nätförfining, spektrala elementmetoden, algebraiska flernätsmetoden, NACA4412, *periodic hill* fall.

Preface

This doctoral thesis deals with the development of adaptive mesh refinement capabilities for computational fluid dynamics simulations using the spectral element method. A large part of the work presented in this thesis represents the contribution made by the author to the ExaFLOW project.

The first part of the thesis describes the state-of-the-art in error estimators and automatic mesh refinement and introduces useful concepts that appear throughout the work. The second part contains 7 papers. Papers 1 and 2 have been published in the proceedings of a conference. Papers 3 and 4 have been accepted for publishing in the proceedings of a conference. Paper 5 has been submitted to a journal. Paper 6 is an technical report. Paper 7 has been accepted as a contribution to a forthcoming conference. Papers 1, 2, 3, 4, 5 and 7 are adjusted to comply with the present thesis format for consistency but their contents have not been altered as compared with their original counterparts.

Paper 1. N. Offermans *et al.*, 2016. *On the Strong Scaling of the Spectral Element Solver Nek5000 on Petascale Systems.* Proceedings EASC '16 .

Paper 2. N. Offermans, A. Peplinski, O. Marin, P. Fischer & P. Schlatter, 2017. *Towards adaptive mesh refinement for the spectral element solver Nek5000.* Proceedings DLES11.

Paper 3. N. Offermans, A. Peplinski, O. Marin, E. Merzari & P. Schlatter, 2018. *Performance of preconditioners for large-scale simulations using Nek5000.* Proceedings ICOSAHOM18.

Paper 4. A. Peplinski, N. Offermans, P. F. Fischer & P. Schlatter, 2018. *Non-conforming elements in Nek5000: Pressure preconditioning and parallel performance.* Proceedings ICOSAHOM18.

Paper 5. N. Offermans, A. Peplinski, O. Marin & P. Schlatter, 2019. *Adaptive mesh refinement for steady flows in Nek5000.* Submitted to Computers & Fluids.

Paper 6. N. Offermans, A. Peplinski & P. Schlatter, 2019. *Unsteady adjoint error estimators and adaptive mesh refinement in Nek5000.* Technical report.

Paper 7. Á. Tanarro *et al.*, 2019. *Using adaptive mesh refinement to simulate turbulent wings at high Reynolds numbers.* Paper contribution to TSFP11.

June 2019, Stockholm
Nicolas Offermans

Division of work between authors

The main advisor for the project is Philipp Schlatter (PS). Adam Peplinski (AP) and Paul Fischer (PF) act as co-advisors. The project was initiated and defined by PS.

Paper 1. The setup of the case has been performed by Nicolas Offermans (NO) with help from Michel Schanen (MS), Oana Marin (OM), AP and PS, with additional input from Jing Gong (JG), Aleks Obabko, Max Hutchinson and Elia Merzari (EM). The simulations have been performed by NO on Beskow, by MS and OM on Mira and by JG on Titan. The paper has been written by NO, MS and OM, with feedback from AP, PF and PS.

Paper 2. The code has been implemented by NO with some help from AP and OM. The paper has been written by NO, with feedback from OM, AP, PS and PF.

Paper 3. The code has been implemented by NO on a suggestion by OM and PS and with help from AP, OM, EM and PS. The test case was provided by EM. NO has run the simulations, produced the data and written the paper with feedback frim AP, OM, EM and PS.

Paper 4. AP has developed the non-conforming pressure preconditioner, with help from NO for debugging. AP has written the paper with feedback from NO, PF and PS.

Paper 5. The steady adjoint error estimators have been developed and implemented by NO, with some help from AP, OM, PS and Niclas Jansson. The tools for mesh refinement have been developed by AP. The simulations have been performed and analyzed by NO. The paper has been mostly written by NO, with contributions from OM and with feedback from AP, OM and PS.

Paper 6. The unsteady adjoint error estimators have been developed and implemented by NO, with some help from AP and PS. The tools for mesh refinement have been developed by AP. The simulations have been performed and analyzed by NO. The paper has been written by NO, with feedback from AP and PS.

Paper 7. The simulations have been performed and analyzed by Álvaro Tanarro (AT), Fermín Mallor (FM) and AP, with help from NO, Ricardo Vinuesa (RV) and PS. The paper has been mostly written by AT with contributions from NO and FM, with feedback from AP, RV and PS.

Conferences

Part of the work in this thesis has been presented at the following international conferences. The presenting author is underlined.

Nicolas Offermans, Oana Marin, Michel Schanen, Jing Gong, Paul Fischer, Adam Peplinski & Philipp Schlatter. *On the Strong Scaling*

of the Spectral Element Solver Nek5000 on Petascale Systems. EASC 2016: Exascale Applications and Software Conference. Stockholm, Sweden, 2016.

Nicolas Offermans, Oana Marin, Adam Peplinski & Philipp Schlatter. *AMG solver for the coarse grid preconditioner.* Nek5000 Users Meeting. Boston, MA, USA, 2016.

Nicolas Offermans, Oana Marin, Adam Peplinski, Paul Fischer & Philipp Schlatter. *Towards adaptive mesh refinement for the spectral element solver Nek5000.* DLES 11: Direct and Large Eddy Simulation. Pisa, Italy, 2017.

Nicolas Offermans, Oana Marin, Adam Peplinski & Philipp Schlatter. *Adjoint error estimators in Nek5000.* Nek5000 Users meeting. Tampa, Florida, USA, 2018.

Nicolas Offermans, Oana Marin, Adam Peplinski & Philipp Schlatter. *Performance of preconditioners for large-scale simulations using Nek5000.* ICOSAHOM 18: International Conference on Spectral and High Order methods. London, UK, 2018.

Nicolas Offermans, Adam Peplinski & Philipp Schlatter. *Error control and grid adaptivity.* ExaFLOW all-hands meetings. Various locations, 2015–2018.

Contents

Part II - Papers

Part I

Overview and summary

Chapter 1

Introduction

In many problems of industrial and scientific interest, the resort to Computational Fluid Dynamics (CFD) has become a common alternative and/or complement to experiments. This has been driven by the dramatic increase in computing power and the development of fast algorithms for the resolution of the Navier–Stokes equations. In the last two decades, for example, the computational speed of the fastest computer available increased by a factor 100,000, going from 1.338 Tflops for the ASCI RED supercomputer to 143,500 Tflops for Summit, the currently fastest supercomputer in the world (Top500, November 1998 *vs.* November 2018). The next major milestone is to reach the exascale, that is 1,000,000 Tflops or 10^{18} flops, which is expected to happen around 2021.

From its inception in the early 1940s, CFD allowed the resolution of more and more complex flow equations over time: from potential flows (Hess & Smith 1967) in the 1960s and 1970s to two- and three-dimensional Euler equations in the 1980s to the Navier–Stokes equations at the end of the 1980s. Yet, the resolution of the full Navier–Stokes equations and the size of problems that can be studied is usually limited by the breakdown to turbulence at high Reynolds numbers and the resolution of all the turbulent scales up to dissipation at the end of the energy cascade (Moin & Kim 1997). Therefore, models are often used and the current methods in CFD can be split in three main categories of increasing complexity and cost.

The first family of models relies on the Reynolds averaged Navier–Stokes (RANS) equations, where the instantaneous velocity is decomposed into a mean component and a fluctuating component. With this approach, only the largest-scale, non-fluctuating structures are resolved (Spalart 2000; Schiestel 2008, Chapter 11). The closure problem can be addressed via models for the eddy viscosity, such as $k-\varepsilon$ (Launder & Spalding 1974), $k-\omega$ (Saffman & Whitham 1970; Wilcox 2008) or Spalart-Allmaras (Spalart & Allmaras 1992), or via a Reynolds stress equation model (Launder *et al.* 1975; Johansson & Hallbäck 1994). RANS methods are cheap and widely used for industrial applications but they cannot well predict transition and and separation.

A second category of models is the large eddy simulation (LES) method, where most scales are resolved and only and the smallest scales are removed

via low-pass filtering and modeled instead (Smagorinsky 1963; Deardorff 1970). LES simulations remain usually too expensive for most industrial applications at high Reynolds number but have become a widespread method in research (Bardina 1983; Piomelli 1999; Schiestel 2008, Chapter 19). LES can also be split into two subcategories: wall-resolved LES, which resolves boundary layers, and wall-modeled LES, which is cheaper but usually less reliable to predict separation. Additionally, some hybrid techniques, lying halfway between RANS and LES, such as detached–eddy simulation (DES) (Spalart 2009) or hybrid RANS-LES modeling (Fröhlich & von Terzi 2008), were also developed in the past few years.

Finally, the direct numerical simulation (DNS) approach resolves all the turbulent scales, without using any model. Given a sufficient mesh resolution, DNS is therefore the exact numerical equivalent of a wind tunnel. However, the computing cost for DNS is prohibitively expensive for virtually all configurations of engineering interest and the method is primarily used as a research tool to validate models or advance the knowledge in the field of turbulence on canonical flow configurations, such as boundary layers or free-stream turbulence (Kim *et al.* 1987; Moin *et al.* 1998). In fact, for these canonical configurations, the cost of numerical simulations is nowadays roughly on par with an experimental campaign in a laboratory.

However, DNS remains out of reach for real-life applications. According to common estimates for the computing cost, DNS roughly requires $N \sim Re^{9/4}$ gridpoints (Rogallo & Moin 1984), where the Reynolds number is based on large eddy characteristic length and velocity scales. For boundary layers, the requirement becomes $N \sim Re_{L_x}^{37/14}$ (Choi & Moin 2012), where L_x is the length of the flat plate in the streamwise direction. This compares to $N \sim Re_{L_x}$ for wall-modeled LES and $N \sim Re_{L_x}^{13/7}$ for wall-resolved LES.

Regardless of the chosen method, the size and complexity of problems to be studied is virtually unbounded and a high competitiveness advantage is to be gained by reaching the desired solution in the shortest time. Consequently, faster and more reliable methods are constantly being developed by the CFD community. For a list of the challenges facing current and future developments, we refer to the vision of CFD in 2030 as illustrated in the NASA technical report by Slotnick *et al.* (2014). In the aforementioned reference, the authors present five main areas where significant progresses are desirable by 2030, with an emphasis on aerodynamic simulations.

First of all, CFD codes need to adapt to the next generation of supercomputers (Gropp & Snir 2013). In particular, CFD codes and algorithms of the future should be robust and scalable, while being energy efficient. In addition, the lack of scalable pre- and post-processing methods needs to be addressed. Access to and usage of high performance computing (HPC) environments need to be facilitated for the expert CFD user to focus mostly on data production and analysis.

Another fundamental improvement for future CFD simulations lies in the development of robust and reliable models for turbulence modeling. Additional research on RANS, LES and hybrid models is required to be able to better predict transition to turbulence and flow separation.

As a third objective, the NASA report mentions that CFD simulations should be made more autonomous and reliable. This goal can be achieved by a combination of three factors: adequate tools for uncertainty quantification and error estimators, mesh adaptivity and robust solvers and numerics. This will ensure that numerical results are more reliable and less dependent on the physics of the problem or on the quality of the mesh.

Then, effective and parallel tools should be implemented to visualize and extract data from large, high-resolution simulation results.

Finally, some frameworks should be built for multiphysics simulations in order to couple various models and phenomena for more accurate and complex simulations.

The present thesis lies in the continuity of the work presented in a previous manuscript (Offermans 2017) and focuses on two of the mentioned challenges: better algorithms and autonomous simulations.

Large numerical problems on highly distributed systems require efficient algorithms. The development of new methods must take into account aspects relating to the physics of the problems, its numerical discretization and the characteristics of the available hardware. For the incompressible Navier–Stokes equations, the pressure solver is most challenging because of its elliptic nature and most efforts are targeted at its fast resolution. A common strategy is to combine a preconditioner and an iterative solver, which has proven efficient. Yet, the constant evolution of hardware and the development of new methods require the adaptation of existing solutions and a constant search for better alternatives.

The idea of autonomous simulations has been around for several decades but has failed to become mainstream in CFD codes. This is partly due to a more complex code development and maintenance. For this reason, most tools for mesh adaptation are available within external libraries rather than being deeply integrated in the code. Another reason is the lack of reliable error estimators, which prevents the refinement to be as effective as it could be.

Thesis structure. Part I of the thesis continues with an introduction on the various topics that are addressed in the second part. In Chapter 2, we present the spectral element method (SEM) and Nek5000, the code we use as our framework. We discus aspects of high performance computing in Chapter 3 and we introduce some useful background regarding parallel efficiency and strong scaling. We also introduce the basic concepts behind the algebraic multigrid method, which is used for preconditioning the pressure equation. In Chapter 4, we perform a literature review on error estimators, mesh refinement and AMR

for the SEM. Finally, conclusions on the current work and outlook on future progress are discussed in Chapter 5.

Part II of the thesis includes seven papers in total, two papers published in the proceedings of a conference, two papers to appear in the proceedings of a conference, one paper submitted to a journal, one technical report and one paper accepted as a contribution to a forthcoming conference. Paper 1 presents the strong scaling of Nek5000. Papers 2 and 3 relate to the improvement of the AMG solver for the pressure preconditioner. Paper 4 assesses the efficiency of the non-conforming solver. Paper 5 and 6 deal with aspects of error estimators and AMR. Paper 7 presents AMR simulation of the turbulent flow around wing profiles.

Chapter 2

Nek5000 and the spectral element method

The developments and applications presented in this thesis are done within the framework of Nek5000 (Fischer *et al.* 2008), an open-source, highly scalable and portable code based on the spectral element method (SEM), which was initially developed by Patera (1984). The SEM can be seen as a high order finite element method (FEM), which benefits from the properties of spectral methods (Henderson 1999; Karniadakis & Sherwin 2005; Kopriva 2009, Chapter 8). The code was originally developed at the beginning of the 1980's by PhD students at the Massachussetts Institute of Technology (M.I.T.). Because of its high-oder nature, the code offers minimal dissipation and dispersion, high accuracy and nearly exponential convergence. Nek5000 is a fast and efficient code, which scales well up to $\mathcal{O}(1,000,000)$ processors. Some of the reasons for its efficiency include

- a matrix–free approach and a tensor representation of the solution, allowing for high cache efficiency and low memory usage, thus reducing computational cost,
- an efficient preconditioner for the pressure equation combining dense local contributions and a coarse global contribution which minimizes communication,
- a projection technique for the solution arrays of the pressure and the velocity components, which speeds up the resolution of the iterative solvers,
- improved numerical stability via filtering, which consists in damping the spectral modes with highest frequencies,
- a communication library that minimizes overhead depending on the properties of the network.

The code is aimed at direct numerical simulations (DNS) but LES–level of resolution can be obtained via filtering. RANS equations have also been implemented but are not the primary focus. Solver options for passive scalars and temperature, including conjugate heat transfer problems, are available as well. Only hexaedral elements are supported in three dimensions (quadrilaterals in two dimensions) but the grid can be unstructured and the elements may be deformed to account for curved boundaries.

Most of the theoretical aspects of the SEM and implementation details behind Nek5000 can be found in the book by Deville *et al.* (2002). In the rest of the section, we briefly summarize the SEM as it is implemented in Nek5000.

2.1. Non-dimensional Navier–Stokes equations

As a starting point, let us consider the incompressible, non-dimensional version of the Navier–Stokes equations for a Newtonian fluid. The velocity and space variables are scaled respectively by U, a reference velocity, and L, a reference length for the problem. The pressure is scaled by ρU^2, where ρ is the density of the fluid, and the forcing terms are scaled by U^2/L. The system of equations to be integrated in time is

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u} = -\nabla p + \frac{1}{Re}\nabla^2\boldsymbol{u} + \boldsymbol{f}, \tag{2.1}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{2.2}$$

for $t \in [0, T]$, with associated boundary and initial conditions. The unknown variables are $\boldsymbol{u}(\boldsymbol{x}, t)$, the velocity, and $p(\boldsymbol{x}, t)$, the pressure. Other parameters are $\boldsymbol{f}(\boldsymbol{x}, t)$ a forcing term and $Re = \frac{UL}{\nu}$ the Reynolds number, a measure of the ratio between convective and diffusive effects, where ν is the kinematic viscosity of the fluid. The solution is defined on $\boldsymbol{x} \in \Omega$, where Ω is the domain being considered.

2.2. Numerical discretization

We introduce the basic concepts behind the discretization and numerical resolution of the Navier–Stokes equations with the SEM. The section is an extension to a similar introduction from a previous thesis (Offermans 2017).

2.2.1. *Weak form*

In dimension d, the problem (2.1)–(2.2) can be recast in weak form as

Find $\boldsymbol{u} \in X,\ p \in Z$ *s.t. for almost every* $t \in [0, T]$

$$\frac{\partial}{\partial t}(\boldsymbol{u}, \boldsymbol{v}) + \mathcal{C}(\boldsymbol{u}; \boldsymbol{u}, \boldsymbol{v}) = \mathcal{B}(\boldsymbol{v}, p) + \frac{1}{Re}\mathcal{A}(\boldsymbol{u}, \boldsymbol{v}) + (\boldsymbol{f}, \boldsymbol{v}) \qquad \forall \boldsymbol{v} \in X_0, \tag{2.3}$$

$$\mathcal{B}(\boldsymbol{u}, q) = 0 \qquad \forall q \in Z, \tag{2.4}$$

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}^0, \tag{2.5}$$

where

$$
\begin{aligned}
(\boldsymbol{a},\boldsymbol{b}) &= \int_\Omega \boldsymbol{a}\left(\boldsymbol{x}\right)\cdot\boldsymbol{b}\left(\boldsymbol{x}\right)\mathrm{d}\boldsymbol{x} \quad \forall \boldsymbol{a},\boldsymbol{b}\in L^2\left(\Omega\right),\\
X &= \left\{\boldsymbol{v} : \boldsymbol{v}_i \in H^1\left(\Omega\right), i=1,\ldots,d, \boldsymbol{v}=\boldsymbol{g}_D \text{ on } \Gamma_D\right\},\\
X_0 &= \left\{\boldsymbol{v} : \boldsymbol{v}_i \in H^1\left(\Omega\right), i=1,\ldots,d, \boldsymbol{v}=0 \text{ on } \Gamma_D\right\},\\
Z &= L^2\left(\Omega\right),\\
\mathcal{A} &: H^1(\Omega)^d \times H^1(\Omega)^d \to \mathbb{R},\ \mathcal{A}\left(\boldsymbol{u},\boldsymbol{v}\right) = \left(\nabla\boldsymbol{u},\nabla\boldsymbol{v}\right),\\
\mathcal{B} &: H^1(\Omega)^d \times L^2(\Omega) \to \mathbb{R},\ \mathcal{B}\left(\boldsymbol{u},p\right) = -\left(\nabla\cdot\boldsymbol{u},p\right),\\
\mathcal{C} &: H^1(\Omega)^d \times H^1(\Omega)^d \times H^1(\Omega)^d \to \mathbb{R},\ \mathcal{C}\left(\boldsymbol{u};\boldsymbol{v},\boldsymbol{w}\right) = \left((\boldsymbol{u}\cdot\nabla)\boldsymbol{v},\boldsymbol{w}\right).
\end{aligned}
$$

L^2 is the Lebesgue space with 2-norm and H^1 is the Sobolev space of functions belonging to L^2, whose first derivative also belongs to L^2. By Γ_D, we denote the part of the boundary of Ω where Dirichlet boundary conditions are applied and $\boldsymbol{u}^0$ is the initial condition at $t=0$. $\mathcal{A}$ and $\mathcal{B}$ are bilinear forms and $\mathcal{C}$ is the convection operator.

2.2.2. *Spectral basis functions*

Similarly to the finite element method, the support for the basis and test functions is obtained by splitting the global domain Ω in E non-overlapping elements Ω^e. On each element Ω^e, the basis functions belong to $\mathbb{P}_N(\Omega^e)$, the set of polynomials of order N. The space X from the weak formulation is then restricted to a finite dimensional subspace

$$X^N = X \cap \mathbb{P}^d_{N,E},$$

where $\mathbb{P}^d_{N,E}$ denotes the tesselation of polynomials of order N in d dimensions, on each of the E elements of the domain. In this section, all computations are done on a reference element $[-1,1]^d$, in dimension d.

One choice of basis can be the Legendre polynomials, which are defined by the partial differential equations

$$\frac{\mathrm{d}}{\mathrm{d}r}\left(\left(1-r^2\right)\frac{\mathrm{d}}{\mathrm{d}r}L_n(r)\right) + n(n+1)L_n(r) = 0, \quad r\in[-1,1].$$

The Legendre polynomials $L_n(r)$ for $n=0,\ldots,4$ are illustrated in Figure 2.1a. This type of basis is known as a modal basis, characterized by a hierarchy of modes from polynomial order 0 to N. The Legendre polynomials also satisfy the orthogonality condition

$$\int_{-1}^{1} L_n(r)L_m(r)\mathrm{d}r = \frac{2}{2n+1}\delta_{nm}$$

Alternatively, a basis can be expressed in terms of a specific type of interpolants associated to a set of nodes. This type of basis is consequently typically called a nodal basis. In Nek5000, the interpolation points are the

Gauss–Lobatto–Legendre (GLL) points, denoted $\xi_i, i = 0, \ldots, N$, satisfying the following equation

$$(1 - \xi^2)L'_N(\xi) = 0, \quad \xi \in [-1, 1],$$

where L_N is the Legendre polynomial of order N and L'_N its derivative. The associated basis functions for X^N are defined on each element as the Lagrangian interpolants of order N on the GLL points. In one dimension, those interpolants, denoted $\pi_i^N(r)$, are defined as

$$\pi_i^N(r) = -\frac{(1 - r^2)L'_N(r)}{N(N + 1)L_N(\xi_i)(r - \xi_i)}, \quad r \in [-1, 1].$$

They are illustrated in Figure 2.1b for $i = 0, \ldots, 4$ and $N = 8$. Let us note that these interpolants are not orthogonal with respect to each other but they satisfy the relation $\pi_i(\xi_j) = \delta_{ij}$, which will lead to some simplifications when building the mass matrix.

In two and three dimensions, the solutions are represented via tensor products of the basis polynomials, which offers a high computational speed and efficiency. The associated Lagrange interpolants in two dimensions are given by

$$\pi_{i,j}^N(r, s) = \pi_i^N(r)\pi_j^N(s), \quad (r, s) \in [-1, 1] \times [-1, 1].$$

A visualization of the GLL points in two dimensions can be seen in Figure 2.2. In particular, Figure 2.2a represents the grid as it is designed by the user, while Figure 2.2 includes the inner GLL grid points for $N = 8$.

The formulation in terms of modal and nodal bases are equivalent and it is possible to go from one to the other via a simple spectral transform. In one dimension, a solution $u(r)$ on a reference element $\hat{\Omega}$ can be expanded spatially using the Lagrange polynomials as

$$u(r)|_{\hat{\Omega}} = \sum_{i=0}^{N} u(\xi_i)\pi_i^N(r), \quad r \in [-1, 1].$$

Alternatively, the solution can be seen as an expansion in terms of Legendre polynomials

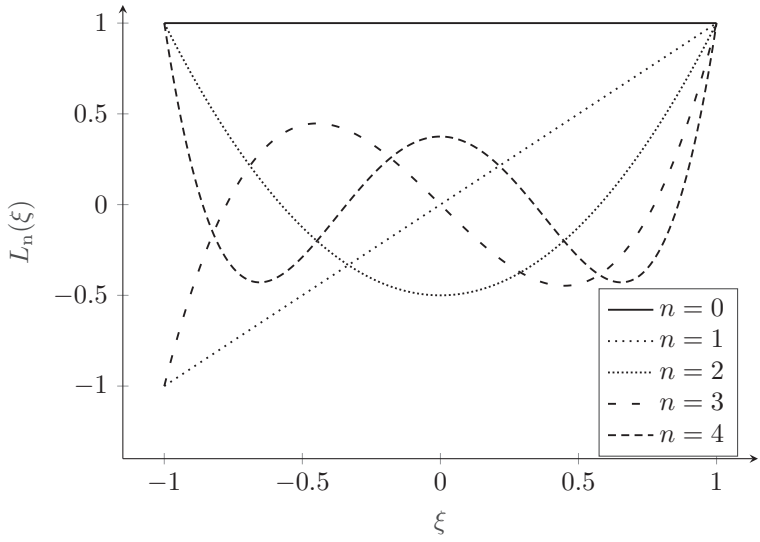
$$u(r)|_{\hat{\Omega}} = \sum_{i=0}^{N} \hat{u}_i L_i(r), \quad r \in [-1, 1],$$

where $\hat{u}_i$ is the modal or spectral coefficient associated to the Legendre polynomial of order i. The relation between the nodal and modal coefficients is given by

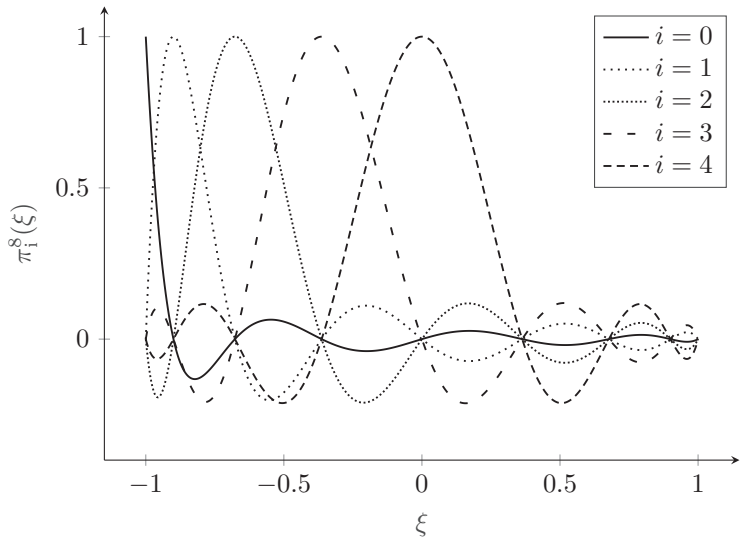
$$\hat{u}_i = \frac{1}{\gamma_i} \int_{-1}^{1} u(r)L_i(r)\mathrm{d}r,$$

where $\gamma_i = \int_{-1}^{1} L_i^2(r)\mathrm{d}r$.

In Nek5000, the expansion of the solution in terms of GLL interpolants is valid for the velocity fields. When it comes to the pressure, the subspace Z^N can be chosen between two options: $Z^N \equiv X^N$ or $Z^N = Z \cap \mathbb{P}^d_{N-2,E}$ the set of $N - 2$ Lagrange interpolants on the Gauss-Legendre (GL) points. The GL

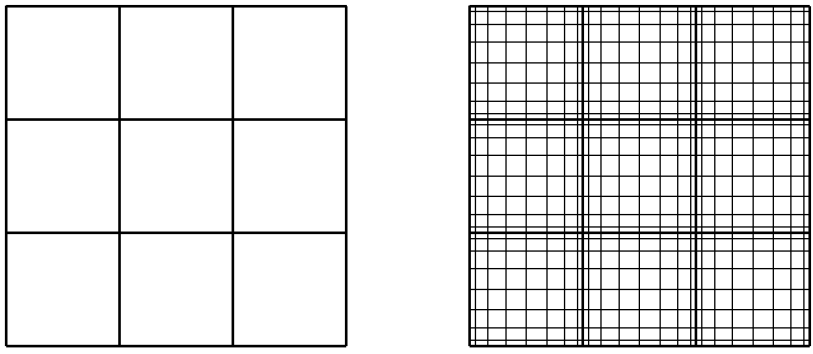


(a) Legendre polynomials of order $n = 0, \ldots, 4$.



(b) Lagrangian interpolants of order $N = 8$ on the Gauss–Lobatto–Legendre points (interpolants shown for $i = 0, \ldots, 4$).

Figure 2.1: Comparison of two spectral bases: (a) Legendre polynomials forming a modal basis; (b) Lagrangian interpolants on the GLL points forming a nodal basis.



(a) Spectral elements.

(b) GLL grid points (located at the intersection of the straight lines).

Figure 2.2: Visualization of the Gauss–Lobatto–Legendre quadrature points for polynomial order $N = 8$.

points are the zeros of $L_{N-1}(\xi)$. In the first method, which is referred to as $\mathbb{P}_N - \mathbb{P}_N$, the pressure and velocity points are collocated and spurious modes are avoided thanks to a method developed by Tomboulides *et al.* (1997). In the second method, referred to as $\mathbb{P}_N\mathbb{P}_{N-2}$, the spurious pressure modes are suppressed by considering a space of smaller polynomial order for the pressure.

2.2.3. *Gauss quadrature*

In the spectral element method, the inner products appearing in equations (2.3)–(2.4) are computed using Gauss quadrature, not analytically. On a one-dimensional reference domain, the general expression for Gaussian quadrature is

$$\int_{-1}^{1} u(r)\,\mathrm{d}r \approx \sum_{k=0}^{N} \rho_k u(\xi_k),$$

where ξ_k are the quadrature points and ρ_k are the associated quadrature weights. A convenient choice for the quadrature points is the GLL points (or possibly the GL points for the pressure), for which the error behaves as $\varepsilon_N \approx \mathcal{O}(u^{-(2N-1)}(\zeta))$, for some point $\zeta \in [-1, 1]$. As will become evident soon, the choice of a single set of points for quadrature and interpolation has some advantages for the computation of the discrete operators. A more detailed discussion on Gauss quadrature using the more general family of Jacobi polynomials is presented by Deville *et al.* (2002).

2.2.4. *Local spectral-element matrices*

Assuming that the global domain Ω is made of the tesselation of E elements, denoted Ω^e $(e = 1, \ldots, E)$, a function on Ω^e can be mapped to the reference

element $\hat{\Omega} = [-1, 1]^d$ by transform of coordinates. In one dimension, a solution $u(x)$ on an element Ω^e can be expanded spatially using the Lagrange polynomials as

$$u(x^e(r))|_{\Omega^e} = \sum_{i=0}^{N} u(x^e(\xi_i))\pi_i^N(r), \quad r \in [-1, 1], \tag{2.6}$$

and we assume that the coordinates transform between x and r satisfies the following affine mapping

$$x^e(r) = x^{e-1} + \frac{L^e}{2}(r + 1),$$

where $L^e = x^e - x^{e-1}$ is the length of element e. We introduce an expansion of the form (2.6) in the weak form (2.3)–(2.5) and use Gauss quadrature on the GLL points for the inner products. On each element e, the local mass matrix is given by

$$\boldsymbol{M}_{ij}^e = \frac{L^e}{2}\int_{\hat{\Omega}} \pi_i(r)\pi_j(r)\mathrm{d}r \approx \frac{L^e}{2}\sum_{k=0}^{N} \rho_k \pi_i(\xi_k)\pi_j(\xi_k) = \frac{L^e}{2}\rho_i\delta_{ij}, \tag{2.7}$$

while the local stiffness matrix is

$$\boldsymbol{K}_{ij}^e = \frac{2}{L^e}\int_{\hat{\Omega}} \frac{\mathrm{d}\pi_i(r)}{\mathrm{d}r}\frac{\mathrm{d}\pi_j(r)}{\mathrm{d}r}\mathrm{d}r = \frac{2}{L^e}\sum_{k=0}^{N} \rho_k \frac{\mathrm{d}\pi_i(\xi_k)}{\mathrm{d}r}\frac{\mathrm{d}\pi_j(\xi_k)}{\mathrm{d}r}.$$

These operators can be rewritten in terms of the derivative, mass and stiffness matrices on the reference element, denoted respectively by $\hat{\boldsymbol{D}}$, $\hat{\boldsymbol{M}}$ and $\hat{\boldsymbol{K}}$, defined as $\hat{\boldsymbol{D}}_{ij} = \frac{\mathrm{d}\pi_j(\xi_i)}{\mathrm{d}r}$, $\hat{\boldsymbol{M}} = diag(\rho_k)$ and $\hat{\boldsymbol{K}} = \hat{\boldsymbol{D}}^T\hat{\boldsymbol{M}}\hat{\boldsymbol{D}}$. Therefore, we get $\boldsymbol{D}^e = \frac{2}{L^e}\hat{\boldsymbol{D}}$, $\boldsymbol{M}^e = \frac{L^e}{2}\hat{\boldsymbol{M}}$ and $\boldsymbol{K}^e = \frac{2}{L^e}\hat{\boldsymbol{K}}$. Thanks to the choice of the polynomial basis and the GLL points, we notice that the resulting mass matrix is diagonal. However, this is only an approximation because the integral in Equation (2.7) is exact up to order $2N - 1$ only (and the integrand is of order $2N$).

In three dimensions, considering an element e with dimensions $L_x^e \times L_y^e \times L_z^e$, the derivative and mass matrices are obtained by tensor product of the one–dimensional operators as

$$\boldsymbol{D}_1^e = \frac{2}{L_x^e}(\hat{\boldsymbol{D}} \otimes \boldsymbol{I} \otimes I),\ \boldsymbol{D}_2^e = \frac{2}{L_y^e}(\boldsymbol{I} \otimes \hat{\boldsymbol{D}} \otimes I),\ \boldsymbol{D}_3^e = \frac{2}{L_z^e}(\boldsymbol{I} \otimes \boldsymbol{I} \otimes \hat{\boldsymbol{D}})$$

and

$$\boldsymbol{M}^e = \frac{L_x^e L_y^e L_z^e}{8}\left(\hat{\boldsymbol{M}} \otimes \hat{\boldsymbol{M}} \otimes \hat{\boldsymbol{M}}\right).$$

In practice, element deformation in 2D and 3D is not limited to affine transformation. A general mapping from an arbitrary element $\Omega(\boldsymbol{x})$ to the reference one $\hat{\Omega}(\boldsymbol{r})$, as illustrated in Figure 2.3 is obtained by introducing the

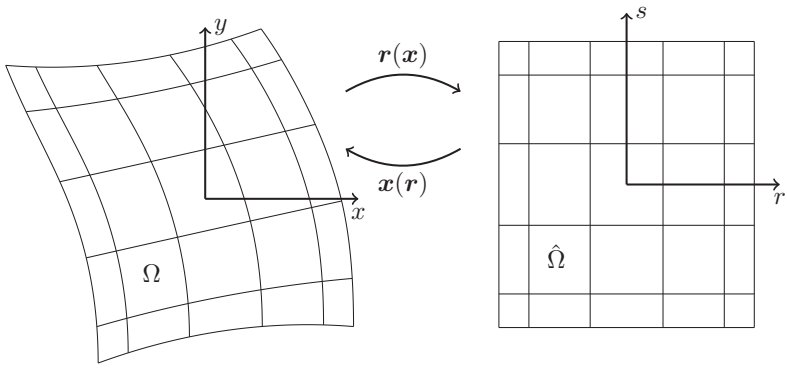


Figure 2.3: Mapping from/to a reference element.

Jacobian

$$J(\boldsymbol{r}) = \det \begin{pmatrix} \frac{\partial x_1}{\partial r_1} & \cdots & \frac{\partial x_1}{\partial r_d} \\ \vdots & & \vdots \\ \frac{\partial x_d}{\partial r_1} & \cdots & \frac{\partial x_d}{\partial r_d} \end{pmatrix}.$$

In three dimensions, the inner product and bilinear form $\mathcal{A}$ appearing in the weak form (2.3)–(2.5) then become

$$\begin{aligned} (\boldsymbol{u}, \boldsymbol{v}) &= \sum_{i=1}^{d} \int_{\Omega^e} u_i v_i \, \mathrm{d}\boldsymbol{x} \\ &= \sum_{i=1}^{d} \int_{\hat{\Omega}} u_i v_i J(\boldsymbol{r}) \, \mathrm{d}\boldsymbol{r}, \\ \mathcal{A}(u, v) &= \sum_{j=1}^{d} \int_{\Omega^e} \frac{\partial u}{\partial x_j} \frac{\partial v}{\partial x_j} \, \mathrm{d}\boldsymbol{x} \\ &= \sum_{k=1}^{d} \sum_{l=1}^{d} \int_{\hat{\Omega}} \frac{\partial u}{\partial r_k} \left(\sum_{j=1}^{d} \frac{\partial r_k}{\partial x_j} \frac{\partial r_l}{\partial x_j} J(\boldsymbol{r}) \right) \frac{\partial v}{\partial r_l} \, \mathrm{d}\boldsymbol{r}. \end{aligned}$$

The derivation to the corresponding discrete operators can be found in the book by Deville *et al.* (2002).

2.2.5. *Continuity enforcement*

The global, unassembled spectral operators are then formed by gathering the local operators as

$$\boldsymbol{M}_L = \begin{bmatrix} \boldsymbol{M}^1 & & & \\ & \boldsymbol{M}^2 & & \\ & & \ddots & \\ & & & \boldsymbol{M}^E \end{bmatrix},$$

$$\boldsymbol{K}_L = \begin{bmatrix} \boldsymbol{K}^1 & & & \\ & \boldsymbol{K}^2 & & \\ & & \ddots & \\ & & & \boldsymbol{K}^E \end{bmatrix}$$

and

$$\boldsymbol{D}_L = \begin{bmatrix} \boldsymbol{D}^1 & & & \\ & \boldsymbol{D}^2 & & \\ & & \ddots & \\ & & & \boldsymbol{D}^E \end{bmatrix}.$$

These operators are called unassembled because they act locally and independently on each element. Gridpoints at the interface between two elements are computed locally as two separate degrees of freedom, while they globally represent a single degree of freedom. Therefore, after application of the unassembled operators, the local values of a same grid point might differ among the elements. Since it is required that the space for the basis function belongs to $H^1(\Omega)$, C^0 continuity needs to be enforced at the interface. If polynomial order N is considered, let $\bar{\mathcal{N}}$ ($\bar{\mathcal{N}} \leq (N+1)^d E$) denote the number of distinct nodes in Ω and $\underline{\boldsymbol{u}} \in \mathbb{R}^{\bar{\mathcal{N}}}$ denote the associated vector of nodal values. Here, the notation with an underscore $\underline{\boldsymbol{u}}$ denotes the discrete version of $\boldsymbol{u}$. Then, let $\underline{\boldsymbol{u}}^e \in \mathbb{R}^{(N+1)^d}$ be the vector of local basis coefficient associated with Ω^e. The collection of all the vectors $\underline{\boldsymbol{u}}^e$ is denoted $\underline{\boldsymbol{u}}_L$. For example, if we refer to Figure 2.4, then the vectors $\underline{\boldsymbol{u}}$ and $\underline{\boldsymbol{u}}_L$ are given by

$$\begin{aligned} \underline{\boldsymbol{u}} &= (\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, \ldots, \boldsymbol{u}_{15})^T, \\ \underline{\boldsymbol{u}}_L &= (\boldsymbol{u}^1_{1,1}, \boldsymbol{u}^1_{1,2}, \ldots, \boldsymbol{u}^1_{3,3}, \boldsymbol{u}^2_{1,1}, \boldsymbol{u}^2_{1,2}, \ldots, \boldsymbol{u}^2_{3,3})^T. \end{aligned}$$

In order to enforce continuity across the elements, we introduce a Boolean connectivity matrix, denoted $\boldsymbol{Q}$ that maps $\underline{\boldsymbol{u}}$ to $\underline{\boldsymbol{u}}_L$. This operator is defined such that

$$\underline{\boldsymbol{u}}_L = \boldsymbol{Q}\underline{\boldsymbol{u}}.$$

This operation is called a scatter from the global to the local vector and its application copies the global values into the local vectors. On the other hand, the reverse operation

$$\underline{\boldsymbol{v}} = \boldsymbol{Q}^T \underline{\boldsymbol{u}}_L$$

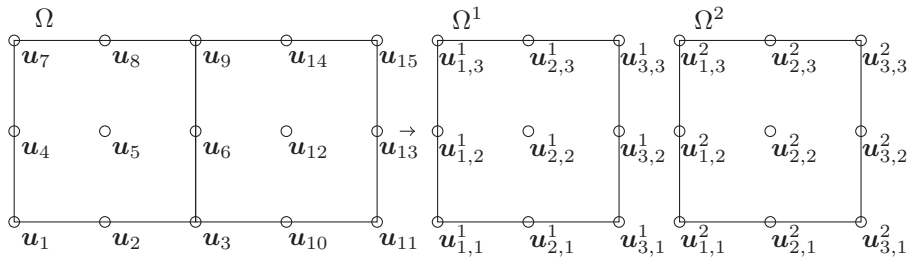


Figure 2.4: Example of a mapping between local and global numberings for a domain made of two spectral elements.

is called a gather. The action of $\boldsymbol{Q}^T$ is to sum entries of the local nodes that correspond to the same global one. Very often a gather is immediately followed by a scatter, resulting in a gather–scatter operation $\boldsymbol{Q}\boldsymbol{Q}^T$. In practice the $\boldsymbol{Q}\boldsymbol{Q}^T$ operator is never explicitly built but only its application is computed. In order to obtain $\underline{\boldsymbol{u}}$ an additional averaging step needs do be performed on $\underline{\boldsymbol{v}}$ since it contains the sum of the entries of $\underline{\boldsymbol{u}}_L$.

2.2.6. *Semi-discrete Navier–Stokes equations*

Denoting by $\underline{\boldsymbol{u}}$ and $\underline{p}$ the discrete counterparts of $\boldsymbol{u}$ and p, and using the operators previously defined, the following semi-discrete problem is obtained:

$$\begin{aligned} \boldsymbol{M}\frac{\mathrm{d}\underline{\boldsymbol{u}}_i}{\mathrm{d}t} + \boldsymbol{C}\underline{\boldsymbol{u}}_i &= \boldsymbol{D}_i^T\underline{p} - \frac{1}{Re}\boldsymbol{K}\underline{\boldsymbol{u}}_i + \boldsymbol{M}\underline{\boldsymbol{f}}_i \quad i = 1,\dots,d \\ \boldsymbol{D}_i\underline{\boldsymbol{u}}_i &= 0, \end{aligned}$$

where $\boldsymbol{M} = \boldsymbol{Q}^T\boldsymbol{M}_L\boldsymbol{Q}$ is the global mass matrix, $\boldsymbol{C}$ is the convection operator, $\boldsymbol{D}_i = \boldsymbol{D}_{L,i}\boldsymbol{Q}$ is the global first derivative operator in the direction i ($i = 1, 2, 3$) and $\boldsymbol{K} = \boldsymbol{Q}^T\boldsymbol{K}_L\boldsymbol{Q}$ is the global stiffness matrix. The convection operator can be expressed in terms of the mass and derivative matrices using either the convective, conservative or skew-symmetric forms. We also note that the derivative matrix $\boldsymbol{D}_i$ is square if the $\mathbb{P}_N - \mathbb{P}_N$ method is used for representing the pressure, while it is rectangular with dimensions $(N-1)\times(N+1)$ if $\mathbb{P}_N - \mathbb{P}_{N-2}$ is used instead.

2.2.7. *Time integration*

The time derivative is discretized via implicit backward differentiation (BDF). The treatment of the advective term is done explicitly using the convective form and an extrapolation (EXT) formula; it is also overintegrated (dealiasing) to ensure that skew-symmetry is retained. If the BDF coefficients are denoted b_j and the EXT coefficients are denoted a_j, then temporal discretization of order

k	b_0	b_1	b_2	b_3	a_1	a_2	a_3
1	1	-1	0	0	1	0	0
2	$\frac{3}{2}$	-2	$\frac{1}{2}$	0	2	-1	0
3	$\frac{11}{6}$	-3	$\frac{3}{2}$	$-\frac{1}{3}$	3	-3	1

Table 2.1: Coefficients for the time integration as a function of the time order k.

k leads to the system

$$\sum_{j=0}^{k} \frac{b_j}{\Delta t} \boldsymbol{M} \underline{\boldsymbol{u}}_i^{n-j} + \sum_{j=1}^{k} a_j \boldsymbol{C} \underline{\boldsymbol{u}}_i^{n-j} = \boldsymbol{D}_i^T \underline{p}^n - \frac{1}{Re} \boldsymbol{K} \underline{\boldsymbol{u}}_i^n + \boldsymbol{M} \underline{\boldsymbol{f}}_i^n, \quad (2.8)$$

$$\boldsymbol{D}_i \underline{\boldsymbol{u}}_i^n = 0 \quad i = 1, \ldots, d. \quad (2.9)$$

The value for coefficients a_i and b_i for time order 1 to 3 is given in Table 2.1 for a constant timestep Δt (see Fischer *et al.* 2017).

2.2.8. *Fractional step method*

The system of Equations (2.8)–(2.9) is commonly solved by decoupling the viscous and pressure terms via a time splitting operation (Fischer 1997; Brynjell-Rahkola 2017). To keep notations simple, we introduce the Helmholtz operator $\boldsymbol{H} = \frac{b_0}{\Delta t}\boldsymbol{M} + \frac{1}{Re}\boldsymbol{K}$ and we define $\underline{\boldsymbol{r}}_i^n = -\sum_{j=1}^{k} \frac{b_j}{\Delta t} \boldsymbol{M} \underline{\boldsymbol{u}}_i^{n-j} - \sum_{j=1}^{k} a_j \boldsymbol{C} \underline{\boldsymbol{u}}_i^{n-j} + \boldsymbol{M} \underline{\boldsymbol{f}}_i^n$.

If $\mathbb{P}_N - \mathbb{P}_{N-2}$ is considered, a block LU-decomposition as proposed by Perot (1993) is performed. Following ideas by Couzy (1995), the system to be solved becomes

$$\begin{aligned} \boldsymbol{H} \underline{\boldsymbol{u}}_i^* &= \boldsymbol{D}_i^T \underline{p}^{n-1} + \underline{\boldsymbol{r}}_i^n, \\ -\frac{b_0}{\Delta t} \boldsymbol{D}_i \boldsymbol{M}^{-1} \boldsymbol{D}_i^T (\underline{p}^n - \underline{p}^{n-1}) &= \boldsymbol{D}_i \underline{\boldsymbol{u}}_i^*, \\ \underline{\boldsymbol{u}}_i^n &= \underline{\boldsymbol{u}}_i^* + \frac{\Delta t}{b_0} \boldsymbol{M}^{-1} \boldsymbol{D}_i^T (\underline{p}^n - \underline{p}^{n-1}), \end{aligned}$$

where $\underline{\boldsymbol{u}}_i^*$ is an intermediate velocity field that is not divergence-free.

The splitting technique for the resolution of the $\mathbb{P}_N - \mathbb{P}_N$ case is described in two papers (Tomboulides *et al.* 1997; Tomboulides & Orszag 1998). While the method was originally developed for a compressible solver, it can be applied to the incompressible case. Skipping numerical details, it is shown that the

system can be solved via the following splitting

$$\boldsymbol{K}\underline{p}^n = \boldsymbol{D}_i \left(\underline{\boldsymbol{r}}_i^n - \frac{1}{Re}\boldsymbol{M} \left(\nabla \times \left(\nabla \times \sum_{j=1}^{k} a_j \underline{\boldsymbol{u}}_i^{n-j} \right) \right) \right),$$
$$\boldsymbol{H}\underline{\boldsymbol{u}}_i^n = \boldsymbol{D}_i^T \underline{p}^n + \underline{\boldsymbol{r}}_i^n.$$

Chapter 3

High performance computing aspects

In this section, we explore aspects related to high performance computing. We start by discussing the various international projects preparing the transition to exascale computing and the possible architectures for the forthcoming machines. Then, we introduce the concept of strong scaling, which allows to assess the efficiency of the combination between hardware and software on large systems and we present hardware properties of current supercomputers. Finally, the core concepts for the efficient resolution of the pressure equation for massively parallel computations in Nek5000 are presented.

3.1. Exascale computing

Current supercomputers operate in the $\mathcal{O}(100)$ Pflops regime and a tremendous effort is produced by the international community to reach the exascale (*i.e.* $\mathcal{O}(1000)$ Pflops) as soon as possible.

The exact technology for exascale computers is still uncertain and will probably be machine dependent. The general trend seems to indicate that such machines will comprise an heterogeneous architecture, characterized by the interaction between a large number ($\mathcal{O}(1,000,000)$) of central processing units (CPUs) and accelerators, such as graphical processing units (GPUs). Both reduced instruction set computer (RISC) and complex instruction set computer (CISC) are being considered. Additional technologies such as hierarchical memory network might as well be included. Other predictions on the future of computers depend on the availability of some HPC technologies currently being developed, such as quantum and molecular computing or superconducting logic for instance (the feasibility of some technologies is discussed by Shalf & Leland (2015)). Irrespectively of the final technology, the many technical challenges include energy efficiency, node failure and resilience, new programming paradigms, management of large data, etc. (Shalf & Leland 2015; Costa *et al.* 2015; Gropp & Snir 2013). Given the size and cost of the challenges ahead, the development of such a computer is led by only a small number of large national and international projects.

China is working on three prototypes of exascale computers, based on different CPU architectures and networks, and hopes to reach the milestone by the end of 2020 or in the beginning of 2021 (Feldman 2018).

In Japan, the Post-K supercomputer is expected to begin its operations around 2021, based on the ARM instruction set architecture. [1]

The first American exascale machine will be the so-called Aurora 21 or A21 computer.[2] It is expected to be built around a still undisclosed architecture in 2021. In addition, a large part of the research on exascale technologies has been gathered under the Exascale Computing Project (ECP) and funded by the US Department of Energy (DOE).[3] ECP is a joint collaboration, which covers various sub-projects in hardware and integration, application development and software technology.

In Europe, the construction of two pre-exascale computers by 2020 and two exascale systems in 2022-2023 is the objective of the European High-Perfomance Computing Joint Undertaking.[4] Furthermore, the European Union funded several projects oriented towards exascale computing under its Horizon 2020 framework programme for research and innovation. One of those projects is ExaFLOW, which brings together academical and industrial partners in order to address the algorithmic challenges toward exascale computations in the field of CFD.[5] The objectives of the ExaFLOW project align well with the challenges presented by NASA in its CFD Vision 2030 Study, as it focuses mostly on the following aspects:

- error control and automatic mesh refinement,
- optimization of the communication strategy on exascale systems,
- solvers efficiency and optimized preconditioners,
- managing fault tolerance and resilience,
- efficient parallel input/output,
- heterogeneous modeling allowing the use of different models depending on the region of the domain,
- reducing energy consumption via energy-efficient algorithms.

The present work is a contribution to the ExaFLOW project, which ran from October 2015 to November 2018, on the topics of error control and automatic mesh refinement. While the ExaFLOW project does not have a direct continuation, most of our contributions will be continued within the Excellerat Centre for Excellence (2019 onwards). [6]

3.2. Strong scaling and hardware characterization

A common question with a parallel code is that of its scalability, *i.e.* how efficiently this code will operate on large, distributed systems. This concern

[1] Fujitsu press release: `http://www.fujitsu.com/global/about/resources/news/press-releases/2018/0621-01.html`
[2] ALCF call for proposals, Argonne National Laboratory: `https://www.alcf.anl.gov/alcf-aurora-2021-early-science-program-data-and-learning-call-proposals`
[3] ECP website: `https://www.exascaleproject.org/`
[4] EuroHPC JU website: `https://eurohpc-ju.europa.eu/index.html`
[5] ExaFLOW website: `http://exaflow-project.eu/`
[6] Excellerat Centre website: `https://www.excellerat.eu/wp/`

can be traced back to the introduction of interconnected machines and the formulation of the Amdahl's law (Amdahl 1967). The relation gives a theoretical limit to the achievable speedup for a given problem solved on an increasing number of computing cores (a configuration known as strong scaling). The law has been extended by Gustafson (1988), with the different assumption that the total amount of work also increases with the number of available processors (a configuration known as weak scaling).

The performance on current supercomputers, composed of multiple nodes, each of them made of several processors and linked via some network, can be assessed by measuring the time to perform some mathematical operations, as well as the time required to send data between two processors on the machine. The framework to study the scalability of a computation on a given machine follows the classical concepts of memory latency, bandwidth, parallel efficiency, *etc.*, as for instance described by Fischer *et al.* (2015*b*).

Assuming a simple linear model for interprocessor communication, the cost for one communication operation t_c can be expressed as a function of the message length m as

$$t_c(m) = \alpha^* + \beta^* m,$$

where α^* is the time required to initiate the communication, also known as latency, and β^* denotes the inverse bandwidth. The bandwidth is the rate at which data is being transferred and is expressed in byte per second ($\mathrm{B\,s^{-1}}$). We compute the values for α^* and β^* on a given machine via a procedure, called "ping-pong test", where the time required to send messages of increasing sizes between processors is measured. Alternatively, these parameters can be expressed in non-dimensional form as

$$\alpha = \alpha^*/t_a \quad \text{and} \quad \beta = \beta^*/t_a,$$

where t_a is the inverse of the observed flop rate. The knowledge of α and β gives some valuable information to understand the parallel efficiency of some algorithms on various computers. We report the various values in Table 3.1 for several machines: Mira (IBM BG/Q), Titan (Cray XK7), Beskow (Cray XC40), Hazel Hen (Cray XC40), Kebnekaise (cluster) and Tetralith (cluster). In all cases, the reported data correspond to the best case of inter-node communication. The results are obtained on 512 cores for Mira, Titan, Beskow and Tetralith, on 560 cores for Kebnekaise and on 528 cores for Hazel Hen. These values have are the results of a single test only and are only indicative.

A common way to assess the efficiency of a computational code on a given supercomputer is by performing a scaling study, which consists in running a given numerical simulation on an increasingly large number of cores. For a strong scaling study, the parallel efficiency, denoted η, can formally be defined as

$$\frac{S_P}{P} = \eta \frac{S_{\mathrm{ref}}}{P_{\mathrm{ref}}},$$

	Year built	$\alpha^*(\mu s)$	$\beta^*(\mu s/wd)$	$t_a(\mu s)$	α	β
Mira	2012	4	$5 \cdot 10^{-3}$	$1.1 \cdot 10^{-3}$	3600	5
Titan	2012	2.25	$1.42 \cdot 10^{-3}$	$6.5 \cdot 10^{-4}$	3500	2.2
Beskow	2014	2.55	$8.25 \cdot 10^{-4}$	$1.5 \cdot 10^{-4}$	17000	5.5
Hazel Hen	2015	1.16	$6.43 \cdot 10^{-4}$	$1.08 \cdot 10^{-4}$	10700	5.94
Kebnekaise	2016	1.65	$8.06 \cdot 10^{-4}$	$1.25 \cdot 10^{-4}$	13200	6.45
Tetralith	2019	1.34	$8.06 \cdot 10^{-4}$	$1.23 \cdot 10^{-4}$	10900	6.55

Table 3.1: Overview of the latencies and bandwidths for various supercomputers. A word (wd) is 64 bits long.

where S_P (in Mflops) is the speed on P processors and S_{ref} is a reference speed using the minimum possible number of processors P_{ref} (ideally one, in practice the lowest achievable number).

We illustrate those concepts with a test case taken from a previous thesis (Offermans 2017). The parallel efficiency of Nek5000 for the case of a turbulent straight pipe at a friction Reynolds number $Re_\tau = 180$ on Beskow is shown in Figure 3.1. This plot shows the typical behavior of the parallel efficiency of Nek5000 in a strong scaling test. At first, the efficiency increases with the number of processors, in this case between 256 and $2,048$ cores. Then, the efficiency drops significantly and becomes roughly half of its value for the reference case on the largest core counts. The existence of a peak in efficiency is due to an optimal balance between two conflicting effects. On a low number of cores, the number of grid points per core is large and the computations are slowed down because of an increase in cache misses. On a high number of cores, on the other hand, the parallel efficiency decreases because of the low amount of local computations and the overhead induced by global communication.

These observations are dependent on the specific test case and computer architecture but they give a good qualitative overview of the different bottlenecks appearing in numerical computations. We also mention that there is not a unique definition of the strong scaling limit. A relevant rule of thumb is to define it as the number of degrees of freedom at which computational time is equal to communication time. This measure identifies the configuration in which more time is spent moving data around than doing actual computations during a simulation. Furthermore, the strong scaling limit does not tell the whole story: total time to solution is another important measure for example (a code which is poorly optimized and slow might still scale well). However, it provides valuable information in a concise way and it helps identify the optimal configuration on a given hardware.

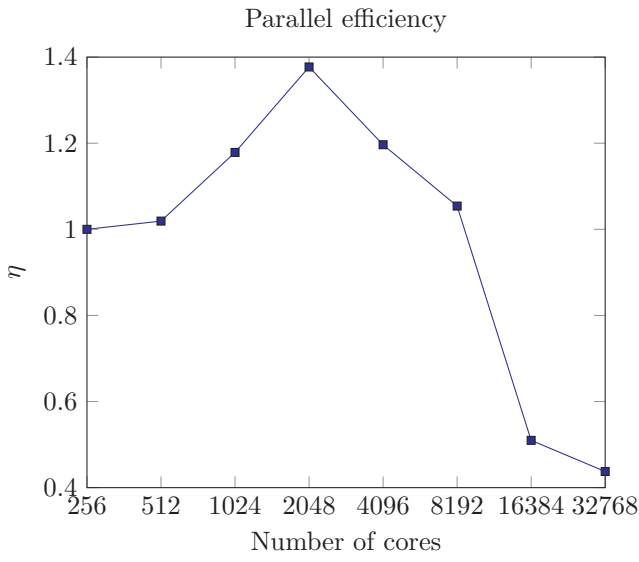


Figure 3.1: Parallel efficiency for the pipe at $Re_\tau = 180$.

3.3. Efficient preconditioning for the pressure equation

Because of its elliptic nature, the pressure equation is the major source of stiffness when solving the incompressible Navier–Stokes equations, making it the most computationally time-consuming part of the solver. Therefore, preconditioners are essential to ensure a rapid resolution of the problem. In this section, we present the two strategies employed by Nek5000 to precondition the pressure equation.

3.3.1. *Preconditioner in Nek5000*

The method chosen for Nek5000 is called additive Schwarz (Fischer 1997) and the preconditioner can be expressed as

$$\boldsymbol{M}^{-1} = \boldsymbol{R}_0^T \boldsymbol{A}_0^{-1} \boldsymbol{R}_0 + \sum_{e=1}^{E} \boldsymbol{R}_e^T \boldsymbol{A}_e^{-1} \boldsymbol{R}_e,$$

where E is the number of spectral elements and $\boldsymbol{R}_e$ and $\boldsymbol{R}_0$ are restrictions operators. This preconditioner can be seen as the sum of the global coarse grid operator (subscript 0) and local subdomain operators (subscript e). We pay a particular attention to the solution of the coarse grid operator, $\boldsymbol{A}_0$, which is constructed as the finite element Laplacian derived from linear elements whose vertices are coincident with the subdomain vertices. Therefore, the coarse grid operator and the coarse grid solver do not depend on the order of the polynomial expansion for the inner points within each element. Two methods are available

in Nek5000 to solve this problem. The first one is a sparse basis projection method, called XX^T developed by Tufo & Fischer (2001). The second method uses an algebraic multigrid method (AMG), which is more efficient for massively parallel (number of processors $P > 10^4$) large simulations ($E > 10^5$). For the sake of completeness, we present the basic concepts behind both the XX^T and the AMG solvers below.

3.3.2. XX^T

The XX^T algorithm is presented because we regularly refer to it and we use it as a reference against the AMG option. However, we do not modify or optimize this algorithm in the present work and we refer to Tufo & Fischer (2001) for more details regarding complexity and implementation. XX^T is based on the Cholesky factorization of the matrix $\boldsymbol{A}_0^{-1}$ into $\boldsymbol{X}\boldsymbol{X}^T$ with a convenient refactoring of the underlying matrix to maximize the sparsity pattern of $\boldsymbol{X}^T$. The underlying idea behind XX^T is to build a sparse basis $\boldsymbol{X}$ such that $\boldsymbol{X}\boldsymbol{X}^T \approx \boldsymbol{A}_0^{-1}$. The first step in building the basis is to find a set of k_1 unit vector that are $\boldsymbol{A}_0$-conjugate. A vector $\boldsymbol{x}_k$ is $\boldsymbol{A}_0$-conjugate if it satisfies

$$\boldsymbol{x}_k^T \boldsymbol{A}_0 \boldsymbol{x}_k = \delta_{ij},$$

where δ_{ij} is the Kronecker operator and $\boldsymbol{A}_0$ has size $n \times n$. Then, if we denote by $\boldsymbol{X}_{k-1} = (\boldsymbol{x}_1 \boldsymbol{x}_2 ... \boldsymbol{x}_{k-1})$ the $n \times (k-1)$ matrix at iteration k and by $\boldsymbol{V} = (\boldsymbol{v}_1 \boldsymbol{v}_2 ... \boldsymbol{v}_n)$ an appropriate column permutation of the identity matrix, the procedure to compute $\boldsymbol{x}_k$ is given by

$$\begin{aligned}
&do\ k = 1, ..., n: \\
&\quad \boldsymbol{w} := \boldsymbol{v}_k - \boldsymbol{X}_{k-1} \boldsymbol{X}_{k-1}^T \boldsymbol{A}_0 \boldsymbol{v}_k \\
&\quad \boldsymbol{x}_k := \boldsymbol{w} / ||\boldsymbol{w}||_{\boldsymbol{A}_0} \\
&\quad \boldsymbol{X}_k := (\boldsymbol{X}_{k-1} \boldsymbol{x}_k) \\
&enddo.
\end{aligned}$$

The algorithm is optimized in order to find an ordering for $\boldsymbol{V}$ that minimizes the fill for $\boldsymbol{X}$. The operator $\boldsymbol{X}$ is computed once at setup time and is then stored in memory for the rest of the simulation.

3.3.3. *Multigrid methods*

The convergence rate of iterative solvers usually stalls after a certain number of iterations because of the slow decay of low frequency errors. Multigrid solvers tackle this issue by building an equivalent, yet coarser problem, via an operation called restriction. Then the relaxation or smoothing operation is performed, where an iterative solver is applied on the coarse level, which efficiently damps errors with lower frequencies. This process is applied recursively until the problem is sufficiently small to be solved directly and features with lowest frequency have been taken into account. Finally, the prolongation operation transfers back to coarsest solution back to the original grid, with possible additional relaxation on the intermediate problems. This combination of a downward and an upward

legs is called a V-cycle. More complex strategies combining more than two, possible partial, legs are also common. The underlying idea behind multigrid methods was first developed by (Brandt 1973), who introduced the concept of multi-level adaptive technique for boundary value problems. Broadly, multigrid methods can be split in two categories: the geometric multigrid and the algebraic multigrid. Both of them are used extensively in CFD codes and have enabled the development of fast and highly scalable solvers. The first method builds a geometrically explicit coarser grid, while the latter builds a coarser algebraic operator. The differences between both approaches are presented for example by Wesseling & Oosterlee (2001) and by K. Stüben (Trottenberg *et al.* 2001, Appendix A) and illustrated in Figure 3.2. Algebraic multigrid methods are generally more flexible and robust than their geometric counterparts, especially for unstructured grids, because coarsening is fully automatic and the smoother is usually a simple relaxation scheme. As a consequence, smoothing is done only in directions that actually reduce the error, which is more efficient. However, they are harder to parallelize, are more memory intensive and require a setup phase, which can be complex and lengthy.

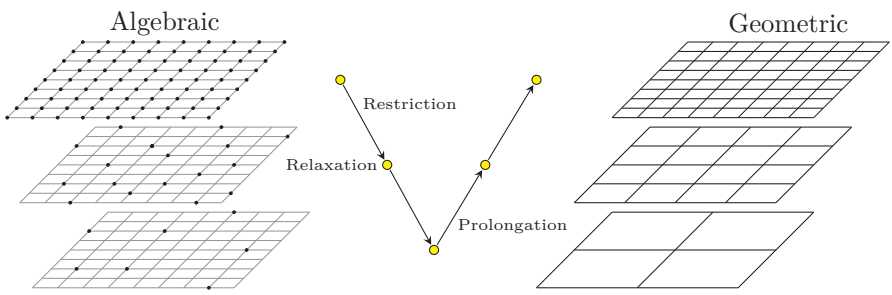


Figure 3.2: Comparison between algebraic and geometric multigrid methods. In the algebraic version, an operator is built at each level on the nodes identified by a black dot; in the geometric approach, an actual grid is considered at each level.

Before solving any problem with the AMG method, a setup phase needs to be done once per grid. The purpose of the setup is to perform three main operations: coarsening, where the nodes for the next coarser level are selected, interpolation, where the relaxation and prolongation operators between two consecutive levels are computed, and computing the relaxation operator, or smoother, on each level.

The coarsening is done based on the concept of the strength of connection between two vertices (Xu & Zikatanov 2017). Considering the problem $\boldsymbol{A}_0\boldsymbol{x} = \boldsymbol{b}$ in a vector space V and assuming that $\boldsymbol{A}_0$ is such that its entries satisfy

$|a_{ij}| = -a_{ij}$, we say that vertex j is θ-strongly connection to vertex i if

$$-a_{ij} \geq \theta \max_{j \neq i} -a_{ik}.$$

As is usually preferable, a symmetric strength measure can also be defined as, for example,

$$s(i,j) = \frac{-a_{ij}}{\min\left(\min_{k \neq i}\left(-a_{ik}\right), \min_{k \neq j}\left(-a_{jk}\right)\right)},$$

and it is said that the vertices are strongly connected if $s(i,j) \geq \theta$. For more measure functions, see Xu & Zikatanov (2017). At each level, the coarsening operation seeks a list of "C-variables", which will form the next coarse grid and "F-variables", which are the remaining variables. This distinction is based on the previously defined strength function $s(i,j)$. Common algorithms for the coarsening include Cleary-Luby-Jones-Plassman (Jones & Plassmann 1993; Cleary *et al.* 1998), Ruge-Stüben (Ruge & Stüben 1987), parallel modified independent set (PMIS) and hybrid modified independent set (HMIS) (De Sterck *et al.* 2006).

Once the coarse nodes have been identified, the interpolation operation computes the restriction and prolongation operators that allow to transfer the solution between levels (see for example Lottes 2017). Broadly speaking, the computation of the interpolation operator can be divided into two main parts:

- computation of the interpolation weights,
- computation of the interpolation support.

The interpolation produces at each level an operator $\boldsymbol{W}$ such that the interpolation matrix

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{W} \\ \boldsymbol{I} \end{bmatrix}$$

allows to restrict an array $\boldsymbol{x}$ to its coarse and remaining subparts $\boldsymbol{x}_f = \boldsymbol{x}(F)$ and $\boldsymbol{x}_c = \boldsymbol{x}(C)$ as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_f \\ \boldsymbol{x}_c \end{bmatrix} \approx \boldsymbol{P}\boldsymbol{x}_c = \begin{bmatrix} \boldsymbol{W}\boldsymbol{x}_c \\ \boldsymbol{I}\boldsymbol{x}_c \end{bmatrix}.$$

For a description of common interpolation algorithms, see Ruge & Stüben (1987); Stüben (2001); Yang (2010); Xu & Zikatanov (2017).

Finally, the smoother can be one of many iterative solvers, such as Jacobi, Gauss-Seidel, Chebyshev etc. The problem on the coarsest grid is ideally solved exactly via Gaussian elimination for example.

The parallel implementation of an AMG setup and solver is a complex task and is an active area of research (Baker *et al.* 2011, 2012). An efficient solver should keep a low computational complexity and memory footprint, while ensuring good convergence.

Chapter 4

Adaptive mesh refinement

When solving partial differential equations in numerical analysis, adaptive mesh refinement (AMR) is a technique for automatically adapting the resolution of the discrete grid in the course of the simulation. The objective of AMR is to increase the accuracy of the solution at a lower computational cost than what is achievable with a fixed mesh. The method is particularly well suited for cases when little a priori knowledge on the solution is available and offers extra flexibility compared to a static mesh. Despite being first introduced more than three decades ago, AMR did not become common practice in major CFD codes because of the additional difficulty regarding software development and maintenance. With the present work, we wish to overcome these difficulties and to deliver AMR tools fully integrated in the CFD solver Nek5000.

Adaptive local refinement was first introduced by Berger & Oliger (1984) and then applied to 2D (Berger & Colella 1989) and 3D computations (Bell *et al.* 1994). The astrophysics community adopted AMR early because such a capability is a requirement when dealing with the large scale variations in cosmological problems (*e.g.* see Bryan *et al.* 2014; Fryxell *et al.* 2000; Mignone *et al.* 2012). Several libraries and software packages have also been developed, providing AMR tools for multiphysics simulations (see Mandli *et al.* 2016; Adams *et al.* 2013; Anderson *et al.* 2013). These packages however are usually restricted to block–structured refinement. Another library is deal.II, which supports *h*-, *p*- and *hp*-refinement for the finite element method (Alzetta *et al.* 2018). Similarly, the libMesh library offers a framework for adaptive simulations for a variety of finite elements (Kirk *et al.* 2006). Wriggers (2005) used AMR to solve contact problems in solid mechanics. In CFD, Hoffman developed Unicorn, a framework for adaptive finite element computation of turbulent flow and fluid–structure interaction within the FEniCS project (Hoffman *et al.* 2013; Alnæs *et al.* 2015) and applied adjoint error estimators on a number of complex 3D geometries (Hoffman 2009; Hoffman *et al.* 2015). Hartmann *et al.* (2010) performed goal-oriented mesh refinement applied to three-dimensional turbulent aerodynamic test cases. In the field of the shallow water equations and tsunami modeling, AMR is commonly used as well because of the strongly convective nature of the problem (Pranowo *et al.* 2008; Blaise *et al.* 2013). Shock modeling

is another obvious application, where localized and possibly moving high mesh resolution is needed (Löhner 1987; Devloo *et al.* 1988).

In this chapter, we introduce the two components required for AMR: a method for error estimation and tools for mesh adaptivity. We start with a review of some common a posteriori error estimators for the spectral element method. Then we introduce the concept of adjoint error estimators, along with a presentation of the adjoint equation. We also briefly mention another common application to the adjoint equation in CFD, that is shape, or topology, optimization. Finally, we address aspects of automatic mesh adaptation and we introduce the three main strategies for mesh refinement, namely h-, p- and r-refinement.

4.1. Local a posteriori error indicators

The section is an extension to a similar introduction from a previous thesis (Offermans 2017). We present local error estimators specifically for spectral element or high order methods, which depend only on the local properties of the solution. In other words, they are a local measure of the conservation of mass and momentum. For an extensive study of error estimators in the finite element method, we refer to Ainsworth & Craig (1991); Ainsworth & Oden (1997); Grätsch & Bathe (2005); Roy (2010); Verfürth (2013).

4.1.1. *Spectral error indicators*

Errors resulting from the discretization and resolution of a system of partial differential equations using the spectral element method arise from four different sources. Modeling error occurs when the mathematical model for the equations does not match reality. This kind of error cannot be handled by the code and we assume that the model is consistent with the actual physics that each user wants to simulate. Then, roundoff error is due to the finite accuracy of computers. This kind of error is also ignored as we once again assume that numerical parameters have been chosen properly (by using double precision floating point arithmetic for example). This means that we are left with truncation error, which arises because the solution is approximated by a finite spectral expansion, and quadrature error, due to the discrete integration. Several methods to compute an estimate to both terms are presented.

4.1.1.1. *Truncation error*

The local error on a spectral element can be estimated by extrapolating the decay of the spectral coefficients as shown by Mavriplis (1990, 1989). If we consider $u(x)$, the exact solution of a 1D partial differential equation, then its spectral transform is

$$u(x) = \sum_{k=0}^{\infty} \hat{u}_k \, p_k(x),$$

where $\hat{u}_k$ are the spectral coefficients and p_k a family of orthogonal polynomials (k denotes the polynomial order). The spectral coefficients are given by

$$\hat{u}_k = \frac{1}{\gamma_k} \int_{-1}^{1} w(x) u(x) p_k(x) \mathrm{d}x, \tag{4.1}$$

where w is a weight associated to the family of polynomials and $\gamma_k = \|p_k\|^2_{L^2_w}$.

The corresponding discrete expansion is truncated to order N by the truncation operator P_N as

$$P_N u(x) = \sum_{k=0}^{N} \hat{u}_k \, p_k(x).$$

Consequently, the truncation error can be written as the L^2_w-norm of $u - P_N u$. If we assume Legendre polynomials, then $w = 1$ and the estimate for the truncation error ϵ_t becomes

$$\begin{aligned}
\epsilon_t = \|u - P_N u\|_{L^2_w} &= \left(\int_{-1}^{1} w(x) \left[\sum_{k=0}^{\infty} \hat{u}_k \, p_k(x) - \sum_{k=0}^{N} \hat{u}_k \, p_k(x) \right]^2 \mathrm{d}x \right)^{\frac{1}{2}} \\
&= \left(\int_{-1}^{1} \sum_{k=N+1}^{\infty} \hat{u}_k^2 \, p_k^2(x) \right)^{\frac{1}{2}} \\
&= \left(\sum_{k=N+1}^{\infty} \frac{\hat{u}_k^2}{\frac{2k+1}{2}} \right)^{\frac{1}{2}}.
\end{aligned} \tag{4.2}$$

However, the coefficients $\hat{u}_k$ are unknown for $k > N$ and need to be approximated. This is achieved by interpolating a linear least-squares approximation of $\log(\hat{u}_k)$ with respect to k for $k \leq N$ and then extrapolating the coefficients for $k > N$. In practice, this is done by computing two parameters c and σ (in a least–square best fit sense) such that

$$\hat{u}_k \approx c \exp(\sigma k).$$

This interpolation gives valuable information about the decay rate σ of the coefficients, which is a good indication for convergence and can be used to decide which refinement method to choose. If the solution is smooth and its decay is monotonic, this estimator performs well. In addition, Willyard (2011) suggests to shift the linear least–squares interpolation upwards so that no spectral coefficient lies above it.

4.1.1.2. *Quadrature error*

Mavriplis (1990) also suggests to estimate the quadrature error, besides the truncation error. We denote by $\bar{u}_k$ the discrete version of the continuous

coefficients from Equation (4.1). They are evaluated as

$$\bar{u}_k = \frac{1}{\gamma_k} \sum_{i=0}^{N} w_i u(\xi_i) p_k(\xi_i),$$

where ξ_i are the Gaussian quadrature points and w_i are the associated quadrature weights. We let $I_N u$ be the Lagrange polynomial interpolation of u

$$I_N u(x) = \sum_{k=0}^{N} \bar{u}_k \, p_k(x).$$

Therefore, the quadrature error ϵ_q is is given by

$$\epsilon_q = \|I_N u - P_N u\|_{L^2_w} = \left(\sum_{k=0}^{N} \frac{(\bar{u}_k - \hat{u}_k)^2}{\frac{2k+1}{2}} \right)^{\frac{1}{2}}.$$

Arguing that the spectral coefficients are obtained exactly for $k \leq N-1$, the quadrature error can be reduced to

$$\epsilon_q = \left(\frac{(\bar{u}_N - \hat{u}_N)^2}{\frac{2N+1}{2}} \right)^{\frac{1}{2}}.$$

In practice, the following upper bound is computed instead

$$\epsilon_q \approx \left(\frac{\bar{u}_N^2}{\frac{2N+1}{2}} \right)^{\frac{1}{2}},$$

which is a safe over-estimate of the actual error.

4.1.1.3. *Alternative error estimate*

It is suggested by Willyard (2011) to use a simpler error estimate. If exponential decay is strong enough, the truncation error ϵ_t can be estimated by

$$\epsilon_t \approx \bar{u}_N.$$

This solution is cheaper to compute than Equation (4.2) but is also supposedly less accurate.

4.1.2. *Constrained Legendre coefficients error estimator*

Another way of estimating the error developed by Willyard (2011) is by comparing the solution $I_N u$ with polynomial order N and another estimated with fewer degrees of freedom $I_{N-M} u$. The solution $I_{N-M} u$ is not computed by solving the problem a second time but is obtained by truncating the spectral series of $I_N u$. We denote the spectral coefficients of $I_{N-M} u$ by $\tilde{u}_k$ such that its spectral transform is

$$I_{N-M} u = \sum_{k=0}^{N-M} \tilde{u}_k \, p_k(x).$$

For $k = 0, ..., N - M - 2$, we take the same coefficients as for $I_N u$ but the last two Legendre coefficients are chosen such that continuity is enforced at the boundary of the reference element. This leads to the system

$$\tilde{u}_k = \bar{u}_k, \qquad k = 0, ..., N - M - 2$$
$$I_{N-M}u(-1) = \sum_{k=0}^{N-M} \tilde{u}_k \, p_k(-1) = \sum_{k=0}^{N-M} \tilde{u}_k \, (-1)^k = I_N u(-1)$$
$$I_{N-M}u(1) = \sum_{k=0}^{N-M} \tilde{u}_k \, p_k(1) = \sum_{k=0}^{N-M} \tilde{u}_k \, 1^k = I_N u(1).$$

This constitutes a second estimate of the local truncation error on a coarser mesh with polynomial order $N - M$

$$\epsilon_t = \|I_N u - I_{N-M} u\|_{L^2_w} = \left(\sum_{k=N-M-1}^{N-M} \frac{(\bar{u}_k - \tilde{u}_k)^2}{\frac{2k+1}{2}} + \sum_{k=N-M+1}^{N} \frac{\bar{u}_k^2}{\frac{2k+1}{2}} \right)^{\frac{1}{2}}.$$

4.1.3. *Sensitivity to refinement with respect to the total error*

Willyard (2011) presents an adjoint based error estimate to find the contribution of the local error at every time step to the total error at the final time, for the case of a bilinear operator. The concepts behind the method are based on those presented by Eriksson *et al.* (1995), where the authors devised an adaptive method, also for nonlinear systems of partial differential equations, in the framework of the finite element method.

4.1.4. *Data-driven error estimators*

Based on recent successes of artificial intelligence and machine learning to model turbulence and transition, Yu *et al.* developed a data-driven artificial viscosity model for the resolution of the compressible Navier–Stokes equations using the discontinuous Galerkin method (Yu *et al.* 2018). Their approach uses artificial neural network to estimate the decay of the modal coefficients as opposed to the linear regression from the method previously presented.

4.2. Goal-oriented adjoint error estimators

We investigate a second type of error estimators, which aim at optimally computing the value of a global output in the form of a functional of physical interest (typically stresses, mean fluxes, drag or lift coefficients, etc.). This is done via the resolution of an adjoint problem, which provides some sensitivity of the output quantity with respect to the residuals of the direct solution. This is therefore a more expensive procedure than traditional a posteriori error indicators but the extra work induced by mesh refinement is much more focused towards a specific goal. In this section, we introduce the concept of adjoint equation and we present the basic ideas behind two approaches for error estimators: adjoint sensitivity to grid refinement and dual-weighted error estimators.

4.2.1. *The adjoint equation*

The concept of duality is well-known in optimization, where the dual problem provides an alternative formulation to the primal one and offers additional insight. With the numerical resolution of partial differential equations, the dual formulation is called adjoint and it provides useful information about the sensitivity of the primal problem to some parameters, variables or boundary conditions. One important aspect of adjoint equations is that there exists two versions, known as the discrete and continuous formulations (Nadarajah & Jameson 2000; Kast 2017). This distinction is mostly a design choice and both options should provide the same information about sensitivity. However, the chosen approach will have an impact on the accuracy of the solution.

In the discrete version, the adjoint equations are derived from the discrete operators appearing in the numerical resolution of the Navier–Stokes equations. With this approach, the adjoint solver is based on the discretization of the direct problem and is therefore strongly tied to it. Considering the residuals

$$\underline{r}(\underline{u}) \equiv \boldsymbol{A}\underline{u} - \underline{b},$$

of the discrete system $\boldsymbol{A}\underline{u} = \underline{b}$ (underline is used to denote the discrete quantities), the computation of the adjoint problem seeks to find the sensitivity of some quantity of interest $J(\underline{u})$, depending on the discrete solution $\underline{u}$, with respect to the residuals $\underline{r}$, that is $\frac{\partial J}{\partial \underline{r}}$. The discrete adjoint variable, denoted $\underline{\varphi}$, can therefore be expressed and expanded by the chain rule as

$$\underline{\varphi} = \left(\frac{\partial J}{\partial \underline{r}}\right)^T = \left(\frac{\partial \underline{u}}{\partial \underline{r}}\right)^T \left(\frac{\partial J}{\partial \underline{u}}\right)^T.$$

Rewriting this equation slightly, we obtain a first intuitive expression for the discrete adjoint formulation, given by

$$\left(\frac{\partial \underline{r}}{\partial \underline{u}}\right)^T \underline{\varphi} = \left(\frac{\partial J}{\partial \underline{u}}\right)^T. \tag{4.3}$$

In this expression, the computation of $\frac{\partial \underline{r}}{\partial \underline{u}}$ and $\frac{\partial J}{\partial \underline{u}}$ can be done by finite difference, algorithmic differentiation or analytic differentiation (Kast 2017).

The continuous adjoint approach, on the other hand, is based on an analytical derivation. The direct equations are multiplied by so-called adjoint variables and integrated by parts to transfer the differential operators from the direct to the adjoint variables, a procedure very similar to the derivation of the weak form. In this way, the process is independent on the direct discretization and the method is more flexible. Considering a linear differential L and an equation of the form

$$Lu = b, \qquad u \in V,$$

where V is a suitable function space, the adjoint operator L^* and the adjoint variable $\varphi \in V$ are given by the following relation

$$(Lu, \varphi) = (u, L^*\varphi),$$

where the notation $(\cdot, \cdot)$ denotes an inner product.

Shape optimization

While on the topic of the adjoint equation, we mention another of its applications besides the computation of error estimators, namely shape optimization. Many of the concepts and notations are shared between both approaches but the final objectives differ. While adjoint error estimators aim at optimally computing a physical quantity of engineering interest for a given configuration, shape optimization seeks to maximize the value of this functional by optimal tweaking of some design parameters.

A first example is aerodynamic shape optimization, where optimal design of wings, airfoils, fuselage, blades, etc. is sought. The use of adjoint equations in CFD was first introduced by Pironneau (Pironneau 1982) for the Euler equations. Jameson extended the method to more complex configurations, up to the full Navier–Stokes equations, using the continuous adjoint approach (Jameson 1988, 1995; Jameson *et al.* 1998). For the discrete adjoint, we mention the work of Giles (Giles & Pierce 2000; Giles 2003). We also mention other works by Anderson (Anderson & Venkatakrishnan 1999) on unstructured grids, Martins (Martins *et al.* 2005) on coupled aero-structural design, Li and Hartmann (Li & Hartmann 2015) on the combination of shape optimization with mesh refinement and we refer to Skinner & Zare-Behtash (2018) for an overview of state-of-the-art in shape optimization.

A second and quite recent application for shape optimization is problems with conjugate heat transfer. Kontoleontos performed topology optimization of various flow cases using the continuous adjoint formulation for both the turbulence model and heat equations and applied their method to various test cases (Kontoleontos *et al.* 2013). Saglietti studied topology optimization of a heat sink in a differentially heated cavity in the case of a natural convection-driven flow (Saglietti *et al.* 2017, 2018). Gkaragkounis used the continuous adjoint method and the Spalart-Almaras turbulence model for shape optimization of a 2D turbine blade and a 3D piston head (Gkaragkounis *et al.* 2018). Van Oevelen also investigated topology optimization of finned heat sinks (Van Oevelen & Baelmans 2014).

4.2.2. *Adjoint error estimators*

In the context of mesh adaptivity, the knowledge of a dual solution can be used to compute useful error estimators. We focus on two main methods, where the adjoint solution is used to minimize the error of a given objective function.

4.2.2.1. *Adjoint sensitivity to grid refinement*

An estimate of the error can be obtained by considering the sensitivity of the output functional between the actual grid and an hypothetical refined one. This approach was first explored by Giles and Pierce (Pierce & Giles 2000). Using notations introduced in the section on discrete adjoint, assume that we want

to optimally compute the value of an integral quantity quantity $J(\underline{u})$, where $\underline{u}$ is the discrete solution of the system of partial differential equations $\underline{r}(\underline{u}) = 0$. Consider a coarse mesh Ω_H and a fine mesh Ω_h, where H and $h < H$ represent typical dimensions of the elements and assume that only the coarse mesh Ω_H actually exists. Now, we can to estimate the value of $J_h(\underline{u}_h)$ on the hypothetical fine mesh Ω_h while knowing only the coarse solution $\underline{u}_H$ as

$$J_h(\underline{u}_h) \approx J_h(\boldsymbol{I}_h^H \underline{u}_H) - (\boldsymbol{I}_h^H \underline{\varphi}_H)^T \underline{r}_h(\boldsymbol{I}_h^H \underline{u}_H),$$

where $\boldsymbol{I}_h^H$ is a prolongation operator which interpolates the solution from the coarse to the fine mesh and $\underline{\varphi}_H$ represents the adjoint solution on the coarse mesh. The adjoint solution is given by the previously defined adjoint problem from Equation (4.3).

This formulation is typically well-suited for the discrete adjoint formulation but this is not a requirement. Such estimators have been implemented and applied on the 1D viscous Burger equation (Ou & Jameson 2011) and for 2D inviscid incompressible flows (Venditti & Darmofal 2002; Balasubramanian & Newman 2009). Luchini *et al.* (2017) used a similar method but considered the global error on the solution as an objective function and managed to reduce the accuracy of the solution by two orders of magnitude by redistributing the initial gridpoints.

The method was also used in combination with a continuous adjoint solver on the incompressible Navier–Stokes equations for the spectral difference method (Li 2013) and the high-order spectral/*hp*-element framework `Nektar++` (Ekelschot *et al.* 2017). In the latter work, the functional of interest is the sum of the drag and the lift for various incompressible flows past a NACA0012 wing section and for the flow past a three-dimensional ellipsoid; the *p*-refinement method was used to refine the mesh.

4.2.2.2. *Dual-weighted error estimators*

With dual-weighted error estimators, the knowledge of some local measure of the error, such as gradients or strong residuals, is weighted by a measure of the dual solution. In essence, this technique is very similar to the previous one but the use of a prolongation operator is avoided. The method is based on the early work by Johnson & Hansbo (1992), who applied dual error estimators to linear elasticity and elasto-plasticity problems.

The foundation to this method has been established by Bangerth & Rannacher (2002), who developed the *Dual Weighted Residual* method. Assuming a linear problem and its discrete approximation of the form

$$\boldsymbol{A}\underline{u} = \underline{b} \quad \text{and} \quad \boldsymbol{A}_h \underline{u}_h = \underline{b}_h,$$

with matrices $\boldsymbol{A}$, $\boldsymbol{A}_h \in \mathbb{R}^{n \times n}$, right hand sides $\underline{b}$, $\underline{b}_h \in \mathbb{R}^n$ and unknown vectors $\underline{x}$, $\underline{x}_h \in \mathbb{R}^n$, it is possible to define the approximation error $\underline{e} \triangleq \underline{u} - \underline{u}_h \in \mathbb{R}^n$ and the residual $\underline{r} \triangleq \underline{b} - \boldsymbol{A}\underline{u}_h \in \mathbb{R}^n$, where h denotes the quality of the approximation (*i.e.* $\boldsymbol{A}_h \to \boldsymbol{A}$ and $\underline{b}_h \to \underline{b}$ as $h \to 0$). We consider a linear functional $J(\underline{u})$,

which we can rewrite as the inner product $(\underline{u}, \underline{j})$, for some $\underline{j} \in \mathbb{R}^n$. Then, the error on the functional associated with the discrete evaluation of the solution is given by

$$J(\underline{u} - \underline{u}_h) = J(\underline{e}) = (\underline{e}, \underline{j}).$$

The corresponding adjoint problem is expressed as

$$\boldsymbol{A}^* \underline{\varphi} = \underline{j},$$

for $\underline{\varphi} \in \mathbb{R}^n$, which leads to the following relation for the error

$$J(\underline{e}) = (\underline{e}, \underline{j}) = (\underline{e}, \boldsymbol{A}^* \underline{\varphi}) = (\boldsymbol{A}\underline{e}, \underline{\varphi}) = (\underline{r}, \underline{\varphi}).$$

Finally, the norm on the error is bounded by

$$|J(\underline{e})| \leq \sum_{k=1}^{K} |\underline{r}_k| \, |\underline{\varphi}_k|,$$

where the strong residuals are weighted by the solution of the adjoint problem. These simple developments lay the basic idea behind the dual weighted residual method. They can be extended to non-linear cases and applied to a wide range of numerical methods and functionals.

Dual-weighted residual estimators were used by Hoffman in the framework of the finite element method and applied to the simulation of turbulent flows at high Reynolds number (Hoffman *et al.* 2015; Hoffman & Johnson 2007, Chapter 30). Richter (2010) expanded the method to include information about anisotropy in the error. Richter & Wick (2015) developed an approach which does not require the computation of the strong residuals but uses filtering of the adjoint solution and a partition of unity.

4.3. Mesh refinement strategies

Following the computation of an error estimator on a given mesh, we wish to adapt this grid, either via refinement or coarsening, in order to control the error. There exists three main strategies for mesh adaptation in the finite and spectral element methods, which we present in this section: moving the existing gridpoints (r-refinement), locally increasing the number of gridpoints (h-refinement) or locally increasing the polynomial order (p-refinement).

4.3.1. *r-refinement*

With the r-refinement technique, the gridpoints are moved and clustered around the regions corresponding to higher error estimates, without changing the topology of the mesh. While seemingly simple, the implementation of an efficient algorithm for r-refinement is actually quite complex and offers less flexibility compared to the two other methods.

Such algorithms can for example be based on a classical steepest-descent method (Xu *et al.* 2011) or make use of a numerical center of mass based on an estimated truncation error (McRae 2000). A common implementation is by using the spring analogy and the Hooke's law, where a force acts on each node

of the mesh, with an intensity depending on the measure of the error, and the edges of the mesh linking the nodes act as springs. The adaption of the mesh follows the computation of the equilibrium length for each spring, *i.e.* the new length of each edge.

4.3.2. *h-refinement*

The h-refinement method consists in refining the mesh locally by dividing some of the elements into smaller ones, with the apparition of so-called hanging nodes at the interface between fine and coarse elements. In the most general case, any element can be refined independently from the others in an anisotropic way, *i.e.* refined in a preferred direction only. However, many implementations impose additional constraints to the refinement pattern for ease of implementation and more efficient computations. In practice, it is common for the refinement to be isotropic, meaning that one element is refined similarly in each direction. Additionally, some codes use a block-structured refinement strategy, where the mesh is seen as a set of non-overlapping structured blocks (or tiles), which need to be refined as a whole. Another method for mesh refinement is patched grids: the mesh is split into patches having coincident interfaces but the grid points at the interface do not need to match.

The treatment of the hanging nodes at the interface between elements can be dealt with by so-called mortar elements. The idea behind this method is that the facets of each element do not communicate directly but via an intermediate element. This "ghost" element is called a mortar element and it is used to compute the flux between the elements, which is then mapped back to the respective facets. This technique is suitable for both the spectral differences (Li 2013) and the spectral element method (see Maday *et al.* 1988, for an extension of the direct stiffness procedure to non-conforming elements).

Another way to perform h-refinement is by direct interpolation of the solution between the facets (Kruse 1997). This is also done by extending the direct stiffness procedure by adding an interpolation operator between the GLL points at the interface.

4.3.3. *p-refinement*

With high-order methods, p-refinement is also possible, where the polynomial order for the solution expansion is locally increased.

This has been done for the spectral difference method (Li 2013) and a high–order spectral/hp formulation of the discontinuous Galerkin method (Ekelschot *et al.* 2017).

Chapter 5

Conclusions and outlook

The present thesis is devoted to improving the efficiency of numerical solvers in the field of computational fluid dynamics. So far, we have introduced various aspects related to the spectral element method, high performance computing and adaptive mesh refinement. In the rest of the thesis, we investigate these topics further. In particular, we look at strong scaling results, we study the use of an AMG solver to precondition the pressure equation and we perform adaptive mesh refinement on selected test cases. For all applications, we discuss details regarding implementation and we provide quantitative results. The particular framework we consider is Nek5000, an open-source code based on the spectral element method, but the results and conclusions discussed in this work remain general to high-order methods.

The strong scaling study of Nek5000 (Paper 1) provides relevant information about the behavior of the code on current supercomputers and gives valuable reference data to assess the efficiency of future improvements. Knowing the time spent in communication and computation on a range of core counts and problem sizes, we can identify the strong scaling limit and the reason for the bottleneck in various regimes. We also observe the known fact that synchronized and low latency communication are paramount for efficient computational fluid dynamics simulations.

The use of an efficient preconditioner for the pressure equation is necessary to maintain a fast and scalable solver. The existing preconditioning strategies in Nek5000 do not extend well to non-conforming meshes and h-refinement, where the preconditioner is possibly reset many times during a simulation. Of the two solvers for the coarse grid part of the preconditioner, the first option (a direct solver called XX^T) is not suitable for large cases while the second option (an in-house AMG solver) requires a slow, external setup phase. This forces us to look for alternatives and we settle on the *BoomerAMG*solver from the Hypre library for linear algebra (Falgout *et al.* 2006). The new implementation is split in two steps: first we only use the setup part from *BoomerAMG*, while leaving the original AMG solver untouched (Paper 2); then, we fully replace the original AMG solver by *BoomerAMG*(Paper 3). In the latter case, it is shown that the setup time is reduced by two orders of magnitude, while the efficiency of the original AMG is maintained.

Since Nek5000 does not support any of the three refinement techniques we mentioned before, at least one of them needs to be implemented before performing AMR. The r-refinement technique is too constraining and not well suited for the large scale applications we have in mind. Regarding the p-refinement method, its implementation requires significant modifications in the code and it is not investigated. Therefore, it is decided to go with the h-refinement approach for its flexibility and relative ease of implementation. The extension to non-conforming meshes began during a previous European project, called Cresta, and is partly based on the work by Kruse (1997). The changes brought to the code include a new implementation of interpolation operators, the modification of the preconditioner and new tools for the management and partitioning of the mesh in parallel (Paper 4).

Once a framework for mesh refinement is available, a critical step is the choice of suitable error estimators. Local spectral error indicators, based on the modal properties of the solutions, are readily available and cheap to compute (see Mavriplis 1990). Then, we also implement adjoint error estimators based on the dual-weighted residual method, developed by Bangerth & Rannacher (2002). Given the high cost induced with the resolution of an adjoint problem, our aim is to compute only a limited number of dual solutions. After a few rounds of refinement, the final mesh is fixed and the simulation proceeds. The hope being that this better mesh compensates for the cost of computing adjoint error estimators. Refinement results and convergence study along with a discussion about cost and efficiency for both choices are obtained on steady (Paper 5) and unsteady (Paper 6) cases. The largest and most complex case, which we run with AMR, is that of the turbulent flow around a NACA airfoil at $Re_c = 200,000$, using spectral error indicators (Paper 7). Overall, many advantages of AMR are observed. It is either possible to reduce the computational cost at a given accuracy or, inversely, to increase the accuracy at a given cost. In addition, we achieve better error control, as shown by a more uniform error over the mesh. Finally, the design of the mesh is made easier and the general quality of the final mesh, in terms of aspect ratio, dependence on boundary conditions, etc. is improved.

At the end of this work, we have reached a point where we can run 3D turbulent simulations with AMR and we have observed very promising results. Yet, some questions remain and more testing will be required to exploit these features to the maximum of their potential.

First, it is not fully clear what choice of error estimators is best for a given configuration. From our first observations, confined geometries do not typically profit from the dual solution but a clear criterion should be established.

Then, the impact of the accuracy of the adjoint solution on the quality of the estimators is not fully understood. We are currently solving the adjoint in the most accurate, and also the most costly, way possible (using the revolve algorithm). Some significant speedup could possibly be obtained with a limited

effect on the error estimators if we are less strict on the accuracy of the dual problem.

An interesting application, which has not been investigated yet, is that of a flow with a strongly convected feature, such as a tsunami or when tracking optimal perturbations. In this case, our strategy of refining the mesh until a final, optimal version is reached, does not work and it is necessary to adapt the mesh frequently to track the feature in time. This different approach requires a different workflow and might lead to different conclusions regarding efficiency and usage.

Another concern relates to the viability and maintenance of the code. The tools relating to AMR add a fair bit of complexity to the original code and along with it, a certain workload to maintain them. This extra effort is usually the reason why AMR has not become widespread in many CFD codes and time will tell if this applies to our attempt as well.

Further improvement in simulation performance can be achieved by a tight integration between software and hardware. The future of computing is probably made of very many cores coupled with accelerators, such as GPUs. Being able to harvest this power in an efficient way is an active area of research and a challenging task. Software improvement might possibly come from the field of artificial intelligence and machine learning. Such methods have already been applied to turbulence modeling and to simple error estimators. They also have the potential to accelerate the computation of the adjoint problem, improve the accuracy of estimators, etc.

Acknowledgements

First of all, I want to thank my main supervisor, Philipp Schlatter, for giving me the opportunity to work under his guidance on this research project. I am deeply grateful for your trust, your constant support and your ability to explain even the most complicated concepts in a clear way on the wide range of topics related to computational fluid dynamics. My sincere gratitude also goes to my co-supervisor, Adam Peplinski, for his invaluable help and his patience in answering all of my questions. The present thesis would not have been possible without your remarkable work during these past four years. I am very indebted to Paul Fischer – my other co-supervisor – for sharing his expertise on the spectral element method and Nek5000.

I also thank all the researchers that I have met during my three stays at the Argonne National Laboratory for their warm welcome. The dearest thanks goes to Oana Marin, my supervisor and mentor at Argonne, for making those stays possible and for teaching me about the workings of the lab. You were the kindest host and a great tutor, and getting to know you enriched my PhD experience considerably. Elia Merzari and Aleks Obabko are also thanked for letting me join their research group and for their assistance during my visits.

Thanks to Niclas Jansson for his help with the formulation and the implementation of the adjoint error estimators.

I wish to extend my gratitude to all of my colleagues and friends from the department of Mechanics for their kindness and for making the department such a pleasant workplace.

Finally, a very special thanks goes to my family and to Rebeka for their understanding and their unfailing support.

Financial support for this work was provided by the European Commission Horizon 2020 project grant entitled "ExaFLOW: Enabling Exascale Fluid Dynamics Simulations" (grant reference 671571) and by the Knut and Alice Wallenberg (KAW) Foundation via the Wallenberg Academy Fellow (WAF) program. Additional support from the Swedish e-Science Research Centre (SeRC) and the SeRC Exascale Simulation Software Initiative (SESSI) is gratefully acknowledged. This research also used resources provided by the Swedish National Infrastructure for Computing (SNIC) at the PDC Centre for High Performance Computing (PDC-HPC), at the High Performance Computing Center North (HPC2N) and at the National Supercomputer Centre (NSC).

Bibliography

Adams, M., Colella, P., Graves, D. T., Johnson, J., Keen, N., Ligocki, T. J., Martin, D. F., McCorquodale, P., Modiano, D., Schwartz, P., Sternberg, T. & Van Straalen, B. 2013 Chombo software package for AMR applications - design document. *Tech. Rep.* LBNL-6616E. Lawrence Berkeley National Laboratory.

Ainsworth, M. & Craig, A. 1991 A posteriori error estimators in the finite element method. *Numerische Mathematik* **60** (1), 429–463.

Ainsworth, M. & Oden, J. 1997 A posteriori error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering* **142** (1), 1–88.

Alnæs, M. S., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M. E. & Wells, G. N. 2015 The fenics project version 1.5. *Archive of Numerical Software* **3** (100).

Alzetta, G., Arndt, D., Bangerth, W., Boddu, V., Brands, B., Davydov, D., Gassmoeller, R., Heister, T., Heltai, L., Kormann, K., Kronbichler, M., Maier, M., Pelteret, J.-P., Turcksin, B. & Wells, D. 2018 The `deal.II` library, version 9.0. *Journal of Numerical Mathematics* **26** (4), 173–183.

Amdahl, G. M. 1967 Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, *AFIPS '67 (Spring)* , pp. 483–485. New York, NY, USA: ACM.

Anderson, R., Arrighi, W., Elliott, N., Gunney, B. & Hornun, R. 2013 SAMRAI concepts and software design. *Tech. Rep.* LLNL-SM-617092-DRAFT. Lawrence Berkeley National Laboratory.

Anderson, W. & Venkatakrishnan, V. 1999 Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers & Fluids* **28** (4), 443–480.

Baker, A. H., Falgout, R. D., Kolev, T. V. & Yang, U. M. 2011 Multigrid smoothers for ultraparallel computing. *SIAM J. Sci. Comput.* **33** (5), 2864–2887.

Baker, A. H., Falgout, R. D., Kolev, T. V. & Yang, U. M. 2012 Scaling hypre's multigrid solvers to 100,000 cores. In *High-Performance Scientific Computing: Algorithms and Applications*, pp. 261–279. London: Springer London.

Balasubramanian, R. & Newman, J. 2009 Adjoint-based error estimation and grid adaptation for functional outputs: Application to two-dimensional, inviscid, incompressible flows. *Computers & Fluids* **38** (2), 320–332.

Bangerth, W. & Rannacher, R. 2002 *Adaptive Finite Element Methods for Differential Equations*. Birkhäuser, Basel.

Bardina, J. 1983 Improved turbulence models based on large eddy simulation of homogeneous, incompressible turbulent flows. PhD thesis, Stanford University, CA., USA.

Bell, J., Berger, M., Saltzman, J. & Welcome, M. 1994 Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM Journal on Scientific Computing* **15** (1), 127–138.

Berger, M. & Colella, P. 1989 Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* **82** (1), 64–84.

Berger, M. J. & Oliger, J. 1984 Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics* **53** (3), 484–512.

Blaise, S., St-Cyr, A., Mavriplis, D. & Lockwood, B. 2013 Discontinuous Galerkin unsteady discrete adjoint method for real-time efficient tsunami simulations. *Journal of Computational Physics* **232** (1), 416–430.

Brandt, A. 1973 Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. In *Lecture Notes in Physics, Berlin Springer Verlag*, *Lecture Notes in Physics, Berlin Springer Verlag*, vol. 18, pp. 82–89.

Bryan, G. L., Norman, M. L., O'Shea, B. W., Abel, T., Wise, J. H., Turk, M. J., Reynolds, D. R., Collins, D. C., Wang, P., Skillman, S. W., Smith, B., Harkness, R. P., Bordner, J., Kim, J.-h., Kuhlen, M., Xu, H., Goldbaum, N., Hummels, C., Kritsuk, A. G., Tasker, E., Skory, S., Simpson, C. M., Hahn, O., Oishi, J. S., So, G. C., Zhao, F., Cen, R., Li, Y. & The Enzo Collaboration 2014 ENZO: An Adaptive Mesh Refinement Code for Astrophysics. *The Astrophysical Journal Supplement Series* **211**, 19.

Brynjell-Rahkola, M. 2017 Studies on instability and optimal forcing of incompressible flows. PhD thesis, KTH Royal Institue of Technology, Sweden.

Choi, H. & Moin, P. 2012 Grid-point requirements for large eddy simulation: Chapman's estimates revisited. *Physics of Fluids* **24** (1), 011702.

Cleary, A. J., Falgout, R. D., Henson, V. E. & Jones, J. E. 1998 Coarse-grid selection for parallel algebraic multigrid. In *Solving Irregularly Structured Problems in Parallel* (ed. A. Ferreira, J. Rolim, H. Simon & S.-H. Teng), pp. 104–115. Berlin, Heidelberg: Springer Berlin Heidelberg.

Costa, G. D., Fahringer, T., Gallego, J. A. R., Grasso, I., Hristov, A., Karatza, H., Lastovetsky, A., Marozzo, F., Petcu, D., Stavrinides, G., Talia, D., Trunfio, P. & Astsatryan, H. 2015 Exascale machines require new programming paradigms and runtimes. *Supercomputing Frontiers and Innovations* **2** (2).

Couzy, W. 1995 Spectral element discretization of the unsteady Navier-Stokes equations and its iterative solution on parallel computers. PhD thesis, École Polytechnique Fédérale de Lausanne, Switzerland.

De Sterck, H., Yang, U. & Heys, J. 2006 Reducing complexity in parallel algebraic multigrid preconditioners. *SIAM Journal on Matrix Analysis and Applications* **27** (4), 1019–1039.

Deardorff, J. W. 1970 A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers. *Journal of Fluid Mechanics* **41** (2), 453–480.

Deville, M. O., Fischer, P. F. & Mund, E. H. 2002 *High-Order Methods for Incompressible Fluid Flow*. Cambridge University Press.

Devloo, P., Oden, J. T. & Pattani, P. 1988 An h-p adaptive finite element method for the numerical simulation of compressible flow. *Computer Methods in Applied Mechanics and Engineering* **70** (2), 203–235.

Ekelschot, D., Moxey, D., Sherwin, S. & Peir, J. 2017 A p-adaptation method for compressible flow problems using a goal-based error indicator. *Comput. Struct.* **181** (C), 55–69.

Eriksson, K., Estep, D., Hansbo, P. & Johnson, C. 1995 Introduction to adaptive method for diffential equations. *Acta Numerica* pp. 1–54.

Falgout, R. D., Jones, J. E. & Yang, U. M. 2006 The design and implementation of hypre, a library of parallel high performance preconditioners. In *Numerical Solution of Partial Differential Equations on Parallel Computers* (ed. A. M. Bruaset & A. Tveito), pp. 267–294. Berlin, Heideleberg: Springer Berlin Heidelberg.

Feldman, M. 2018 China spills details on exascale prototypes. `https://www.top500.org/news/china-spills-details-on-exascale-prototypes/`.

Fischer, P., Schmitt, M. & Tomboulides, A. 2017 Recent developments in spectral element simulations of moving-domain problems. In *Recent Progress and Modern Challenges in Applied Mathematics, Modeling and Computational Science*, pp. 213–244. New York, NY: Springer New York.

Fischer, P. F. 1997 An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations. *Journal of Computational Physics* **133** (1), 84–101.

Fischer, P. F., Heisey, K. & Min, M. 2015 Scaling Limits for PDE-Based Simulation (Invited). In *Proceedings of the 22nd AIAA Computational Fluid Dynamics Conference*, *AIAA Aviation* . American Institute of Aeronautics and Astronautics.

Fischer, P. F., Lottes, J. W. & Kerkemeier, S. G. 2008 Nek5000 Web page. Http://nek5000.mcs.anl.gov.

Fröhlich, J. & von Terzi, D. 2008 Hybrid LES/RANS methods for the simulation of turbulent flows. *Progress in Aerospace Sciences* **44** (5), 349–377.

Fryxell, B., Olson, K., Ricker, P., Timmes, F. X., Zingale, M., Lamb, D. Q., MacNeice, P., Rosner, R., Truran, J. W. & Tufo, H. 2000 FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *The Astrophysical Journal Supplement Series* **131** (1), 273–334.

Giles, M. B. 2003 Discrete adjoint approximations with shocks. In *Hyperbolic Problems: Theory, Numerics, Applications* (ed. T. Y. Hou & E. Tadmor), pp. 185–194. Berlin, Heidelberg: Springer Berlin Heidelberg.

Giles, M. B. & Pierce, N. A. 2000 An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion* .

Gkaragkounis, K., Papoutsis-Kiachagias, E. & Giannakoglou, K. 2018 The continuous adjoint method for shape optimization in conjugate heat transfer problems with turbulent incompressible flows. *Applied Thermal Engineering* **140**, 351–362.

Gropp, W. & Snir, M. 2013 Programming for exascale computers. *Computing in Science Engineering* **15** (6), 27–35.

Grätsch, T. & Bathe, K.-J. 2005 A posteriori error estimation techniques in practical finite element analysis. *Computers & Structures* **83** (4), 235–265.

GUSTAFSON, J. L. 1988 Reevaluating amdahl's law. *Commun. ACM* **31** (5), 532–533.

HARTMANN, R., HELD, J., LEICHT, T. & PRILL, F. 2010 Error estimation and adaptive mesh refinement for aerodynamic flows. In *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* (ed. N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. van der Ven & K. Sørensen), pp. 339–353. Berlin, Heidelberg: Springer Berlin Heidelberg.

HENDERSON, R. D. 1999 Adaptive spectral element methods for turbulence and transition. In *High-Order Methods for Computational Physics* (ed. T. J. Barth & H. Deconinck), pp. 225–324. Berlin, Heidelberg: Springer Berlin Heidelberg.

HESS, J. & SMITH, A. 1967 Calculation of potential flow about arbitrary bodies. *Progress in Aerospace Sciences* **8**, 1–138.

HOFFMAN, J. 2009 Efficient computation of mean drag for the subcritical flow past a circular cylinder using general Galerkin G2. *International Journal for Numerical Methods in Fluids* **59** (11), 1241–1258.

HOFFMAN, J., JANSSON, J., DE ABREU, R. V., DEGIRMENCI, N. C., JANSSON, N., MÜLLER, K., NAZAROV, M. & SPÜHLER, J. H. 2013 Unicorn: Parallel adaptive finite element simulation of turbulent flow and fluid–structure interaction for deforming domains and complex geometry. *Computers & Fluids* **80** (Supplement C), 310–319.

HOFFMAN, J., JANSSON, J., JANSSON, N. & ABREU, R. V. D. 2015 Towards a parameter-free method for high Reynolds number turbulent flow simulation based on adaptive finite element approximation. *Computer Methods in Applied Mechanics and Engineering* **288**, 60–74.

HOFFMAN, J. & JOHNSON, C. 2007 *Computational Turbulent Incompressible Flow: Applied Mathematics: Body and Soul 4*. Berlin, Heidelberg: Springer Berlin Heidelberg.

JAMESON, A. 1988 Aerodynamic design via control theory. *Journal of Scientific Computing* **3** (3), 233–260.

JAMESON, A. 1995 Optimum aerodynamic design using CFD and control theory. *CFD Review* **3**.

JAMESON, A., MARTINELLI, L. & PIERCE, N. 1998 Optimum aerodynamic design using the Navier–Stokes equations. *Theoretical and Computational Fluid Dynamics* **10** (1), 213–237.

JOHANSSON, A. V. & HALLBÄCK, M. 1994 Modelling of rapid pressure–strain in Reynolds–stress closures. *Journal of Fluid Mechanics* **269**, 143–168.

JOHNSON, C. & HANSBO, P. 1992 Adaptive finite element methods in computational mechanics. *Computer Methods in Applied Mechanics and Engineering* **101** (1), 143–181.

JONES, M. T. & PLASSMANN, P. E. 1993 A parallel graph coloring heuristic. *SIAM Journal on Scientific Computing* **14** (3), 654–669.

KARNIADAKIS, G. & SHERWIN, S. 2005 *Spectral/hp element methods for computational fluid dynamics*, 2nd edn. Oxford University Press.

KAST, S. M. 2017 An introduction to adjoints and output error estimation in computational fluid dynamics. *arXiv* Preprint:1712.00693.

Kim, J., Moin, P. & Moser, R. 1987 Turbulence statistics in fully developed channel flow at low Reynolds number. *Journal of Fluid Mechanics* **177**, 133–166.

Kirk, B. S., Peterson, J. W., Stogner, R. H. & Carey, G. F. 2006 `libMesh`: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. *Engineering with Computers* **22** (3–4), 237–254.

Kontoleontos, E. A., Papoutsis-Kiachagias, E. M., Zymaris, A. S., Papadimitriou, D. I. & Giannakoglou, K. C. 2013 Adjoint-based constrained topology optimization for viscous flows, including heat transfer. *Engineering Optimization* **45** (8), 941–961.

Kopriva, D. A. 2009 Spectral element methods. In *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers*, pp. 293–354. Dordrecht: Springer Netherlands.

Kruse, G. W. 1997 Parallel nonconforming spectral element solution of the incompressible Navier–Stokes equations in three dimensions. PhD thesis, Brown University, RI., USA.

Launder, B. & Spalding, D. 1974 The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering* **3** (2), 269–289.

Launder, B. E., Reece, G. J. & Rodi, W. 1975 Progress in the development of a Reynolds-stress turbulence closure. *Journal of Fluid Mechanics* **68** (3), 537–566.

Li, D. & Hartmann, R. 2015 Adjoint-based airfoil optimization with discretization error control. *International Journal for Numerical Methods in Fluids* **77** (1), 1–17.

Li, Y. 2013 Automatic mesh adaptation using the continuous adjoint approach and the spectral difference method. PhD thesis, Stanford University, CA., USA.

Lottes, J. W. 2017 *Towards Robust Algebraic Multigrid Methods for Nonsymmetric Problems*. *Springer Theses* 1. Springer International Publishing.

Luchini, P., Giannetti, F. & Citro, V. 2017 Error sensitivity to refinement: a criterion for optimal grid adaptation. *Theoretical and Computational Fluid Dynamics* **31** (5), 595–605.

Löhner, R. 1987 An adaptive finite element scheme for transient problems in CFD. *Computer Methods in Applied Mechanics and Engineering* **61** (3), 323–338.

Maday, Y., Mavriplis, C. & Patera, A. 1988 Nonconforming mortar element methods: Application to spectral discretizations. In *Proceedings of the 2nd International Conference on Domain Decomposition Methods*, pp. 392–418.

Mandli, K. T., Ahmadia, A. J., Berger, M., Calhoun, D., George, D. L., Hadjimichael, Y., Ketcheson, D. I., Lemoine, G. I. & LeVeque, R. J. 2016 Clawpack: building an open source ecosystem for solving hyperbolic PDEs. *PeerJ Computer Science* **2**, e68.

Martins, J. R., Alonso, J. J. & Reuther, J. J. 2005 A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. *Optimization and Engineering* **6** (1), 33–62.

Mavriplis, C. 1989 Nonconforming discretizations and a posteriori error estimators for adaptive spectral element techniques. PhD thesis, Massachusetts Institute of Technology, MA., USA.

Mavriplis, C. 1990 A posteriori error estimators for adaptive spectral element techniques. In *Notes on Numerical Fluid Mechanics* (ed. P. Wesseling), pp. 333–342.

McRae, D. 2000 r-refinement grid adaptation algorithms and issues. *Computer Methods in Applied Mechanics and Engineering* **189** (4), 1161–1182.

Mignone, A., Zanni, C., Tzeferacos, P., Van Straalen, B., Colella, P. & Bodo, G. 2012 The PLUTO code for adaptive mesh computations in astrophysical fluid dynamics. *Astrophysical Journal, Supplement Series* **198** (1).

Moin, P., & Mahesh, K. 1998 Direct numerical simulation: A tool in turbulence research. *Annual Review of Fluid Mechanics* **30** (1), 539–578.

Moin, P. & Kim, J. 1997 Tackling Turbulence with Supercomputers. *Scientific American* **276**, 62–68.

Nadarajah, S. K. & Jameson, A. 2000 A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. AIAA 38th Aerospace Sciences Meeting and Exhibit, Reno, NV, U.S.A.

Offermans, N. 2017 Towards adaptive mesh refinement in Nek5000. Licentiate thesis, KTH Royal Institute of Technology, Sweden.

Ou, K. & Jameson, A. 2011 Unsteady adjoint method for the optimal control of advection and burgers equations using high order spectral difference method. In *Proceedings of the 49th AIAA Aerospace Sciences Meeting*.

Patera, A. T. 1984 A spectral element method for fluid dynamics: Laminar flow in a channel expansion. *Journal of Computational Physics* **54** (3), 468–488.

Perot, J. 1993 An analysis of the fractional step method. *Journal of Computational Physics* **108** (1), 51–58.

Pierce, N. A. & Giles, M. B. 2000 Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM REVIEW* **42** (2), 247–264.

Piomelli, U. 1999 Large-eddy simulation: achievements and challenges. *Progress in Aerospace Sciences* **35** (4), 335–362.

Pironneau, O. 1982 Optimal shape design for elliptic systems. In *System Modeling and Optimization* (ed. R. F. Drenick & F. Kozin), pp. 42–66. Berlin, Heidelberg: Springer Berlin Heidelberg.

Pranowo, W. S., Behrens, J., Schlicht, J. & Ziemer, C. 2008 Adaptive mesh refinement applied to tsunami modeling: Tsunaflash. In *Proceedings of the International Conference on Tsunami Warning (ICTW)*.

Richter, T. 2010 A posteriori error estimation and anisotropy detection with the dual-weighted residual method. *International Journal for Numerical Methods in Fluids* **62** (1), 90–118.

Richter, T. & Wick, T. 2015 Variational localizations of the dual weighted residual estimator. *Journal of Computational and Applied Mathematics* **279**, 192–208.

Rogallo, R. S. & Moin, P. 1984 Numerical simulation of turbulent flows. *Annual Review of Fluid Mechanics* **16** (1), 99–137.

Roy, C. 2010 Review of discretization error estimators in scientific computing. In *Proceedings of the 48th AIAA Aerospace Sciences Meeting*.

Ruge, J. W. & Stüben, K. 1987 Algebraic multigrid. In *Multigrid Methods*, pp. 73–130.

Saffman, P. G. & Whitham, G. B. 1970 A model for inhomogeneous turbulent flow. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* **317** (1530), 417–433.

Saglietti, C., Schlatter, P., Monokrousos, A. & Henningson, D. S. 2017

Adjoint optimization of natural convection problems: differentially heated cavity. *Theoretical and Computational Fluid Dynamics* **31** (5), 537–553.

Saglietti, C., Schlatter, P., Wadbro, E., Berggren, M. & Henningson, D. S. 2018 Topology optimization of heat sinks in a square differentially heated cavity. *International Journal of Heat and Fluid Flow* **74**, 36–52.

Schiestel, R. 2008 *Modeling and Simulation of Turbulent Flows*. ISTE and Wiley.

Shalf, J. M. & Leland, R. 2015 Computing beyond Moore's law. *Computer* **48** (12), 14–23.

Skinner, S. & Zare-Behtash, H. 2018 State-of-the-art in aerodynamic shape optimisation methods. *Applied Soft Computing* **62**, 933–962.

Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E. & Mavriplis, D. 2014 CFD Vision 2030 Study: A path to revolutionary computational aeroscience. *Tech. Rep.* NASA/CR-2014-218178. National Aeronautics and Space Administration.

Smagorinsky, J. 1963 General circulation experiments with the primitive equations. *Monthly Weather Review* **91** (3), 99–164.

Spalart, P. 2000 Strategies for turbulence modelling and simulations. *International Journal of Heat and Fluid Flow* **21** (3), 252–263.

Spalart, P. & Allmaras, S. 1992 A one-equation turbulence model for aerodynamic flows. In *30th Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics.

Spalart, P. R. 2009 Detached-eddy simulation. *Annual Review of Fluid Mechanics* **41** (1), 181–202.

Stüben, K. 2001 A review of algebraic multigrid. *Journal of Computational and Applied Mathematics* **128** (1), 281 – 309, numerical Analysis 2000. Vol. VII: Partial Differential Equations.

Tomboulides, A. G., Lee, J. C. Y. & Orszag, S. A. 1997 Numerical simulation of low Mach number reactive flows. *Journal of Scientific Computing* **12** (2), 139–167.

Tomboulides, A. G. & Orszag, S. A. 1998 A quasi-two-dimensional benchmark problem for low mach number compressible codes. *Journal of Computational Physics* **146** (2), 691–706.

Trottenberg, U., Oosterlee, C. & Schuller, A. 2001 *Multigrid*. Elsevier Academic Press.

Tufo, H. & Fischer, P. 2001 Fast parallel direct solvers for coarse grid problems. *Journal of Parallel and Distributed Computing* **61** (2), 151–177.

Van Oevelen, T. & Baelmans, M. 2014 Application of topology optimization in a conjugate heat transfer problem. In *Proceedings of the 1st International Conference on Engineering and Applied Sciences Optimization*, pp. 562–577.

Venditti, D. A. & Darmofal, D. L. 2002 Grid adaptation for functional outputs: Application to two-dimensional inviscid flows. *Journal of Computational Physics* **176** (1), 40–69.

Verfürth, R. 2013 *A Posteriori Error Estimation Techniques for Finite Element Methods*. Oxford University Press.

Wesseling, P. & Oosterlee, C. 2001 Geometric multigrid with applications to computational fluid dynamics. *Journal of Computational and Applied Mathematics* **128** (1), 311–334.

Wilcox, D. C. 2008 Formulation of the k-w turbulence model revisited. *AIAA Journal* **46** (11), 2823–2838.

Willyard, M. 2011 Adaptive spectral element methods to price american options. PhD thesis, The Florida State University, FL., USA.

Wriggers, P. 2005 Adaptive methods for contact problems. In *Adaptive Finite Elements in Linear and Nonlinear Solid and Structural Mechanics* (ed. E. Stein), chap. 6, pp. 321–363. Vienna: Springer Vienna.

Xu, G., Mourrain, B., Duvigneau, R. & Galligo, A. 2011 Parameterization of computational domain in isogeometric analysis: Methods and comparison. *Computer Methods in Applied Mechanics and Engineering* **200** (23), 2021–2031.

Xu, J. & Zikatanov, L. 2017 Algebraic multigrid methods. *Acta Numerica* **26**, 591–721.

Yang, U. M. 2010 On long-range interpolation operators for aggressive coarsening. *Numerical Linear Algebra with Applications* **17** (2-3), 453–472.

Yu, J., Hesthaven, J. S. & Yan, C. 2018 A data-driven shock capturing approach for discontinuous Galerkin methods. *Tech. Rep.*. École Polytechnique Fédérale de Lausanne.

Part II

Papers

Summary of the papers

Paper 1

On the Strong Scaling of the Spectral Element Solver Nek5000 on Petascale Systems

The strong scaling of Nek5000 is studied on three modern supercomputers. The turbulent flow in a straight pipe is considered at four different friction Reynolds number for the simulations. A short introduction to Nek5000 is presented, as well as the modifications that were done on the code for profiling. The characteristics of each supercomputer are also studied in terms of latency and bandwidth. A strong scaling limit, defined as the minimal number of grid points per process such that the computation time is equal to the communication time, is identified for each case on each machine. Additionally, two methods for the preconditioning of the pressure equation are compared. Others aspects are also briefly discussed, such as the observed super-linearity in some cases, load balancing and weak scaling.

The version of Nek5000 used for the scaling tests, which includes the Cray timers, can be found at `https://github.com/nicooff/Nek5000_deprecated/tree/timers_cray` with commit number `c05663f7f7cdcdde3d61207142e3fce90dee1501`.

Paper 2

Towards adaptive mesh refinement for the spectral element solver Nek5000

Part of the preconditioner for the pressure equation in Nek5000 requires the resolution of a coarse grid solver, which is solved most efficiently by an algebraic multigrid method for large cases. The setup step, required for any multigrid solver, was initially done by slow Matlab code, which can be slow for large cases. A new setup using the Hypre library for linear algebra is implemented and tested on three test casse, namely a jet in crossflow, a straight pipe and a NACA airfoil. It is shown that the new setup is more than one order of magnitude higher than the original code, while the solver time is not significantly affected.

The AMG setup presented in the paper has become the standard option in Nek5000 V17.0 and can be found in the code's main repository at `https:`

//github.com/Nek5000/Nek5000/tree/v17.0/tools/amg_hypre, with initial commit number 5fb22f41f9f591159f028997a7ccfec145c6a20a.

Paper 3

Performance of preconditioners for large-scale simulations using Nek5000

We implement a new method to solve the coarse grid part of the preconditioner for the pressure equation in Nek5000, which is critical for good strong scaling and solver efficiency. We make use of *BoomerAMG*, the algebraic multigrid (AMG) solver from the Hypre library for linear algebra, for both the setup and solver parts, which are both performed on the fly and in parallel. The best parameters for the AMG setup are determined from a parameter study based on the flow past a NACA4412 airfoil. Then, we perform a strong scaling study of the solver on two different supercomputers, namely Mira and Hazel Hen. The test case considered is the turbulent flow around wire-tapped pin bundles, on up to $131,072$ compute cores. We show that the efficiency of the *BoomerAMG* solver is on par with an existing AMG solver, which was designed specifically for the problem at hand, while the overhead induced by a slow and external setup is removed. On Mira, *BoomerAMG* turns out to be slightly slower, while it is a bit faster on Hazel Hen. Yet, these differences are small and we conclude that the use of *BoomerAMG* is a suitable alternative for the resolution of the coarse grid problem.

The AMG solver presented in this paper has been included as an experimental feature in Nek5000 V19.0-rc1 and the related code can be found in Nek5000's main repository at https://github.com/Nek5000/Nek5000/blob/v19.0-rc1/core/experimental/crs_hypre.c with initial commit number 16bfa9e25b60c745b904ad09c1d5ab9cf1bd3d0c.

Paper 4

Non-conforming elements in Nek5000: Pressure preconditioning and parallel performance

We present aspects of the implementation of h-refinement capabilities to the spectral element code Nek5000, for adaptive simulations. These include the description of the interpolation operators at the interface between refined and coarse elements and the modification of the preconditioner for the pressure equation. We rely on external libraries for the mesh management in parallel (p4est) and for the grid partitioning (ParMetis). We use a two-level partitioning strategy, consisting of inter-node step, followed by an intra-node one. We validate our implementation via the strong scaling analysis of the turbulent flow around a NACA4412 airfoil at $Re_c = 200,000$. We show that the parallel efficiency of the solver is retained, as compared to the conforming case.

Paper 5

Adaptive mesh refinement for steady flows in Nek5000

We perform automatic mesh refinement in Nek5000, a code based on the spectral element method, in order to optimise the use of the computational resources. We first present recent modifications that have been made in the code to enable nonconforming h-refinement of the mesh. Then, we introduce spectral error indicators, which are based on the spectral properties of the method. We also develop and implement adjoint error estimators for the spectral element method. Finally, we combine the nonconforming refinement tools and the error estimators to perform adaptive mesh refinement on simple 2D steady test cases: a lid-driven cavity and the flow past a cylinder. The number of degrees of freedom required to reach a given accuracy on the solution decreases significantly compared to a conforming mesh. With the adjoint error estimators, refinement occurs more upstream compared to the spectral error indicators.

Paper 6

Unsteady adjoint error estimators and adaptive mesh refinement in Nek5000

We present the extension from the steady to the unsteady adjoint error estimators and we apply those for the optimal mesh design in the case of the flow around a periodic hill. The functional of interest in this case is the pressure integral on the flat, bottom region between two consecutive hills. With this particular choice, we want to predict the location of the reattachment point at minimal numerical cost. We consider four Reynolds number (Re = 700, 1400, 2800 and 5600) and we use spectral error indicators as a comparison. We study the impact of refinement on the instantaneous resolution and on the the statistics of turbulence by looking at mean velocity profiles. We also look at the convergence of the skin friction coefficient along the bottom wall and especially at the location of the detachment and reattachment points.

Paper 7

Using adaptive mesh refinement to simulate turbulent wings at high Reynolds numbers

Adpative mesh refinement simulations are perfomed for large-eddy simulations of the flow around a NACA4412 airfoil at a Reynolds bnumber of $1,600,000$. We use the spectral element code Nek5000, for which we have implemented h-refinement capabilities. We study the effect of the adverse pressure gradient on the development of the boundary layer. We also validate the use of adaptive simulations at large Reynolds numbers and we show that solver efficiency and accuracy are retained at a lower computational cost.