NTNU
Norwegian University of Science and Technology

Isogeometrisk analyse av Boussinesq ligningene

Abdullah Abdulhaque

Master of Science in Physics and Mathematics

Submission date:	July 2016
Supervisor:	Trond Kvamsdal, MATH
Co-supervisor:	Arne Morten Kvarving, SINTEF IKT
	Timo Matteo van Opstal, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abdullah Abdulhaque

Isogeometric Analysis of the Boussinesq Equations

Master's thesis

Master of Science in Mathematics

Submission date: Trondheim, July 2016

Supervisor: Professor Trond Kvamsdal
Department of Mathematical Sciences, NTNU

Co-supervisors: Research scientist Arne Morten Kvarving
Department of Applied Mathematics, SINTEF ICT
Postdoctoral fellow Timo van Opstal
Department of Mathematical Sciences, NTNU

Department of Mathematical Sciences
Norwegian University of Science and Technology, NTNU

NTNU
Norwegian University of
Science and Technology

Summary

In this master thesis, I have solved the Boussinesq equations with Isogeometric Analysis to study heat transfer coupled with air circulation. The Boussinesq equations consist of the Navier-Stokes' equations and the convection-diffusion equation, and they are coupled together through the Boussinesq-Oberbeck approximation. The equation system is solved on a closed domain with mixed boundary conditions. The Navier-Stokes' equations model the air motion, and the convection-diffusion equation models the temperature distribution.

In order to discretize this system of partial differential equations and to obtain sufficient stability over long time, we combine isogeometric analysis with the mixed and multiscale finite element methods. The global computational complexity and quasilinear structure of the model are managed by using different A-stable time-integrators separately on the sub-equations. Thus, the running time is reduced, and we avoid restrictions on the time steps. We illustrate the advantages which isogeometric analysis has compared with the classical finite element method, and show that it can be used together with already existing algorithms for computational fluid dynamics. An essential description of isogeometric analysis and mesh generation is presented in the beginning.

We perform a systematic analysis of the suitable numerical methods for the Boussinesq equations, with emphasis on h- and p-refinement with a simple tensor mesh on a square domain. We also discuss which time-integrator is best. To verify whether the simulations are correct, we have constructed manufactured reference solutions. Since the domain is simple, we use B-splines as basis functions. We have written complete object-oriented MATLAB codes for the simulations, and GLview Inova is used for visualizing dynamic simulations.

Sammendrag

I denne masteroppgaven har jeg løst Boussinesq ligningene med Isogeometrisk Analyse for å studere varmetransport koblet sammen med luftsirkulasjon. Boussinesq ligningene består av Navier-Stokes ligningene og konveksjon-diffusjonsligningen, og de er koblet sammen ved hjelp av Boussinesq-Oberbeck approksimasjonen. Ligningssystemet er løst på et lukket domene med blandede randbetingelser. Navier-Stokes ligningene modellerer luftstrømningen, og konveksjon-diffusjonsligningen modellerer temperaturfordelingen.

For å diskretisere dette systemet av partielle differensialligninger og oppnå tilstrekkelig stabilitet over lengre tid, kombinerer vi isogeometrisk analyse sammen med blandet og multiskala elementmetode. Den globale beregningskompleksiteten og den kvasilineære strukturen til modellen er håndtert ved å bruke ulike A-stabile tidsintegratorer separat på hver delligning. Da reduseres kjøretiden, og vi unngår restriksjoner på tidskrittene. Vi illustrerer fordelene som isogeometrisk analyse har sammenlignet med klassisk elementmetode, og viser at den kan brukes sammen med de allerede eksisterende algoritmene for numerisk strømningsberegning. En utfyllende beskrivelse av isogeometrisk analyse og grid-generasjon er presentert i begynnelsen.

Vi utfører en systematisk analyse av passende numeriske metoder for å løse Boussinesq ligningene, med fokus på h- og p-forfinelse med et enkelt tensor-grid på et kvadratisk domene. Vi drøfter også om hvilken tidsintegrator som er best. For å verifisere om alle simuleringene er korrekte har vi konstruert egendefinerte referanseløsninger. Siden vårt domene er lett bruker vi B-spliner som basisfunksjoner. Vi har skrevet komplette objektorienterte MATLAB-koder for simuleringene, og GLview Inova er brukt for å visualisere dynamiske simuleringer.

Contents

List of Figures

List of Tables

Preface

This is a Master of Science thesis in numerical mathematics at the Norwegian University of Technology and Science (NTNU). It was written in the spring of 2016. The Master of Science in Applied Physics and Mathematics is carried out over five years and is titled "sivilingeniør" in Norwegian. The first two years is a determined run with common topics for the whole class. During the last three years, I specialized in Industrial Mathematics as my main profile.

I was a summer intern at SIAT (*Centre of Sport Facilities and Technology*, NTNU) for 12 weeks in 2015, where I learned about Isogeometric Analysis (IGA) and Computational Fluid Dynamics (CFD). I had some good knowledge of the finite element method, spline methods, and fluid mechanics. Therefore, I decided that the topic of my master's degree should be how to solve complex problems in CFD with IGA. My supervisor, professor Trond Kvamsdal, proposed that I could apply this new knowledge on how to control the inner climate in ice hockey stadiums. Thus, the topic of the thesis was finally agreed to be isogeometric analysis of the Boussinesq equations.

On the second week of my summer intern, professor Kvamsdal offered me the unique opportunity to participate in *The 3rd International Conference on Isogeometric Analysis*. This gave me a huge insight of new mathematical theory. I began to realize gradually the power of Isogeometric Analysis and got many ideas of what to include in my thesis. I learned from professor Kvamsdal's colleagues at SINTEF about how to implement codes effectively and how to verify whether they work correctly. This knowledge helped me a lot with my research.

The main goal of this thesis is to illustrate how isogeometric analysis can be applied to computational fluid dynamics, with application to control the inner climate of an ice hockey stadium. Isogeometric analysis has received good reputation of being a superset of the classical finite element method, with new superior advantages that were time-consuming to achieve in the past. In my thesis, I will demonstrate how the well-established algorithms that are already being applied to computational fluid dynamics can also be applied to isogeometric analysis, particularly on multiscale modelling and efficient time-integration.

I would like to express my gratitude to professor Trond Kvamsdal for his systematic work approach, valuable guidance and close follow-up through the whole thesis. I was fortunate and blessed with his frequently advising sessions where he showed genuine interest in my work. I am indebted to him for his advice, suggestions, observations, encouragement and support for my thesis. I also thank my co-supervisors Dr. Arne Morten Kvarving and Dr. Timo van Opstal for teaching me how to use the finite element method in computational fluid dynamics and how to verify systematically that complex programming codes work correctly.

I would also like to thank Dr. Eivind Fonn[1] for his excellent tutorials on isogeometric analysis, mesh generation and usage of IFEM-software. He played a major role during the early stages of my work. I also thank Dr. Kjetil André Johannessen[2] and Dr. Knut Morten Okstad[3] for their friendly guidance and teaching of simulations, and the opportunity to learn so much about numerical mathematics.

[1]Research scientist, SINTEF ICT, and Adjoint associate, Department of Mathematical Sciences, NTNU
[2]Postdoctoral Fellow, Department of Mathematical Sciences, NTNU
[3]Research scientist, SINTEF ICT

Chapter 1

Introduction

1.1 Background

Air-conditioning and temperature regulation in an ice hockey stadium is a very complex physical problem. It belongs to the multidisciplinary fields of fluid dynamics and heat transfer. The temperatures at different parts of the stadium should be kept at their own individual levels simultaneously, making good comfort for the audience and preventing the ice from melting. It is also very important with sufficient air circulation to maintain the oxygen levels stable, such that the ice-hockey players as well as the audience will not faint. This has been a long-time research field for many years, but previous attempts for modelling and simulating the situation well have been limited. The main reasons are that we have a three-dimensional model where both temperature and fluid flow are coupled together, and the nonlinearities in the model make it even harder to increase the accuracy of the numerical solution. Nonlinear structure and full 3D character of a physical model typically makes it challenging to solve a partial differential equation numerically with high accuracy.

The time-evolution of air circulation and temperature variation in the ice hockey stadium can be modelled with the *Boussinesq equations*, a system of partial differential equations consisting of the incompressible *Navier-Stokes equations* (air circulation) coupled with the *convection-diffusion equation* (temperature distribution). They are linked together with the *Boussinesq-Oberbeck approximation*, relating fluid density linearly to temperature. The underlying theory behind this system is quite basic and can be explained straightforward. Although our model is incompressible and quasilinear (not fully nonlinear), the complete theory of implementation is very advanced and is far beyond the scope of this master thesis. Hence, we will just discuss the most important parts of the implementation theory as simple as possible, and rather focus on the results from the simulations. They will play a vital role for determining whether the conditions in the ice hockey stadium are acceptable and satisfies the criterions for sustainable comfort.

Figure 1.1. *Leangen ice stadium in Trondheim, Norway*

The complicated structure of the physical model makes it very necessary and desirable to apply a robust and suitable numerical method which can reduce both the running time and increase the accuracy of the solution. We also require that the simulation of time-evolution is stable over long time intervals. *The Finite Element Method* (FEM), the most general numerical method for solving partial differential equations, is appropriate because it can handle complex geometries and boundary conditions easily without big difficulties. Unfortunately, the requirement of high-accuracy is very time-consuming and reduces the effectivity. Traditionally, the *Spectral Element Method* (SEM) has been used extensively for solving numerical problems in fluid mechanics because of its high accuracy, but the major drawbacks of this approach are adaption to very complicated domains and reduced ability to handle material discontinuities.

To avoid these frequent bottlenecks occuring in the classical finite and spectral element methods, we have chosen to use *Isogeometric Analysis* (IGA), a new finite element method under development. It can represent complex geometries exactly, increase the numerical accuracy at high scale, and reduce the computational complexity significantly. Solving the Boussinesq equations with the finite element method is well-known and documented, but solving this system with isogeometric analysis has not been researched much, only separately on the Navier-Stokes and convection-diffusion equations. Results have shown that isogeometric works properly for these sub-equations. Therefore, the main task of the thesis can be formulated as follows:

> *To show that isogeometric analysis can be used for solving the Boussinesq equations by combining it with other existing numerical algorithms, and analyze how the model can be applied to control the inner climate in an ice-hockey stadium.*

	Finite Element Method (FEM)	Spectral Element Method (SEM)	Isogeometric Analysis (IGA)
Convection-diffusion equation	Well-known	Well-known	Well-known
Navier-Stokes equations	Well-known	Well-known	Well-known
Boussinesq equations	Well-known	Well-known	Little research

Figure 1.2. *Current progress in the finite element method for the main equations*

Isogeometric analysis is a big extension and generalization of the classical finite element method. The procedure of creating the variational formulation for a partial differential equation is exactly the same for both methods, and discretization of the domain is not so very different. The main difference is that we are using a new type of shape functions, *splines*, for approximating the solution and meshing the domain more accurately. These new facilities results in a huge number of advantages, and many of them are not fully available in the classical finite element method. Therefore, we find it very convenient to give a concise description of these important properties before going on with the numerical convergence studies with the Boussinesq equations.

It will also be necessary to use sophisticated techniques from *Multiscale Modelling* in the numerical analysis of the Boussinesq equations. This is a relatively new scientific field of research with a vast number of topics including complex multi-physics, decomposition of functions, dimensional analysis, boundary layer theory, perturbation and multiresolution analysis, just to mention a few. Many of the well-known numerical algorithms like multi-grid, adaptive mesh refinement and domain decomposition also belong to this multiscale hierarchy [19]. Some of the main topics in this subject will be of relevant interest in the thesis because they provide better approximation of the unknown solutions.

The *Multiscale Finite Element Method* (MsFEM), developed in the 1980s by leading experts like Babuška, Brezzi and Hughes, provides high stabilization in contrast to the standard finite element approaches. Many of these new techniques have shown success in computational fluid dynamics, so we find it appropriate to combine them together with isogeometric analysis. This will be further discussed in the thesis.

1.2 Outline of the thesis

In chapter 2, we present the derivation of the Navier-Stokes equations and the Boussinesq equations. Then we show how to make them dimensionless by scaling. This part is quite important for the code implementation, and it enables us to apply useful simplifications of the model through perturbation analysis.

We continue with the finite element analysis in chapter 3 to establish the most important properties of the Boussinesq equations. This approach is done with some basic functional analysis and is relevant for the discretization process. The purpose is verifying analytically that the finite element method works properly for our numerical problem.

In chapter 4, we discuss the most important advantages of isogeometric analysis and give a full description of the underlying theory behind B-splines and NURBS (Non-Uniform Rational B-Splines). They constitute the fundamental basis of isogeometric analysis and all other types of splines used for more advanced applications.

The next step is to discretize the Boussinesq equations and determine which numerical methods we can apply to solve the problem. This part is very convenient because the final numerical procedure is a combination of many advanced algorithms. There are also some small remarks on the mesh generation. Chapter 5 treats all these subjects in full detail.

Afterwards, we show that isogeometric analysis works for the Boussinesq equation. The verification procedure is h- and p-refinement in 2D, where the domain is a square. Chapter 6 covers the whole numerical convergence studies, a short description of the refinement techniques, and their compatibility with isogeometric analysis. We will also test a small numerical example with temperature-driven Cavity.

The final conclusion of the thesis is covered in chapter 7.

The appendices at the end covers some of the underlying mathematical theory relevant for this study, and some of the most commonly used algorithms in the simulations.

1.3 Computer facilities

There are many different programming languages, softwares and computers used in the whole thesis. We present them briefly and explain how they have been used.

Programming languages and software

- *MATLAB*: Used for the numerical examples with h- and p-refinement in 2D, and for plotting the figures in chapter 4.
- *Python*: Used for writing the scripts taken as input by GeoModeller.
- *XML*: Used for writing the input files used by IFEM.
- *GeoModeller*: Software developed by SINTEF for creating spline-based meshes on geometric domains, written in Python.
- *IFEM*: Software developed by SINTEF for solving a large class of partial differential equations using isogeometric analysis, written in C++ and FORTRAN.
- *GLview Inova*: Software from Ceetron ASA for visualizing numerical solution of partial differential equations, written in OpenGL.

Computers

- *MacBook Pro*: Personal computer.
- *Markov*: Supercomputer at the Department of Mathematical Sciences, used for all the h- and p-refinements in MATLAB.
- *AFEM*: Supercomputer at SINTEF, used for plotting solutions in GLview.
- *Flop-3*: Supercomputer at SINTEF, used for the simulations with IFEM.

Chapter 2

Heat transfer and fluid flow

The simulation of air circulation inside an ice hockey stadium requires that we develop a consistent model that combines fluid flow and heat transfer as a coupled system of partial differential equations. In our case, we can assume that the air is moving stably with low speed, so it can be regarded as a viscous fluid. The fluid density will not change much during the whole circulation, so the fluid can also be classified as incompressible.

First, we show how to derive the incompressible Navier-Stokes equations by applying the *phenomenological approach*, which is purely based on conservation laws. Then, we show how to derive the time-dependent Convection-Diffusion equation and combine it with the other one by modifying the fluid's energy equation and using the Boussinesq-Oberbeck approximation. This system of partial differential equations describes physically how heat is generated in fluid flow and varies over time.

2.1 The Navier-Stokes equations

The *Navier-Stokes equations* are the fundamental equations of fluid mechanics. We will only consider the incompressible model. The whole derivation is based on the approaches described in [21, 34, 55, 56], and we denote the time interval as $I = (0, \tau]$.

The continuity equation for mass transfer

We start with the universal *continuity equation* for mass conservation. Mass transport is convective because diffusive flux is not present. There is no external mass source either, so the change of mass inside a control volume Ω is defined as

$$\frac{\partial}{\partial t} \iiint\limits_{\Omega} \rho \, dV = - \oint\!\!\!\oint\limits_{\partial\Omega} \rho \mathbf{u} \cdot \hat{\mathbf{n}} \, dS$$

To simplify this integral equation, we use Gauss' theorem on the right-hand side such that we can drop the integral signs. This yields a first order hyperbolic conservation law:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.1}$$

We expand and simplify this conservation law as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = \frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho + \rho \nabla \cdot \mathbf{u} = \frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{u}$$

where $\frac{D}{Dt}$ is the *material derivative operator*. The density change of the fluid is very small and negligible. Hence, we can neglect $\frac{D}{Dt}$ and divide everything by ρ. By doing so, we obtain the linear *incompressibility constraint*:

$$\nabla \cdot \mathbf{u} = 0 \tag{2.2}$$

Physically interpreted, it states that the volume of the fluid is invariant of deformation caused by shear strain acting on it, and the velocity is *solenoidal* (divergence-free).

Navier's equation for linear momentum in fluids

We define $\rho \mathbf{u} \otimes \mathbf{u}$ as the *convective flux tensor* applied to the linear momentum of the fluid. *Newton's second law* states that the sum of the forces acting on the fluid is given by

$$\Sigma \mathbf{F} = \frac{\partial}{\partial t} \iiint\limits_{\Omega} \rho \mathbf{u} \, dV + \oiint\limits_{\partial\Omega} (\rho \mathbf{u} \otimes \mathbf{u}) \cdot \hat{\mathbf{n}} \, dS \tag{2.3}$$

The first integral models the internal change inside the fluid, and the second one represents the outflow on the boundary. By applying Gauss' theorem, we obtain

$$\oiint\limits_{\partial\Omega} (\rho \mathbf{u} \otimes \mathbf{u}) \cdot \hat{\mathbf{n}} \, dS = \iiint\limits_{\Omega} \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) \, dV \tag{2.4}$$

Now, we must decouple the force $\Sigma \mathbf{F}$ on the fluid. We assume that there is an *external body force* acting on the fluid. If $\mathbf{f}$ is the force per mass, then

$$\mathbf{F_f} = \rho \iiint\limits_{\Omega} \mathbf{f} \, dV \tag{2.5}$$

Since our fluid is viscous, both *normal* and *shear stress* must be taken into account. The viscous shear stress arises because of the internal friction force between fluid layers. By applying Gauss' theorem again, we can express the *total surface force* as

$$\mathbf{F}_{\boldsymbol{\sigma}} = \oiint\limits_{\partial\Omega} \boldsymbol{\sigma} \cdot \hat{\mathbf{n}} \, dS = \iiint\limits_{\Omega} \nabla \cdot \boldsymbol{\sigma} \, dV \tag{2.6}$$

where $\boldsymbol{\sigma} = -p\mathbf{I} + 2\mu\boldsymbol{\epsilon}$ is the *Cauchy stress tensor*, and μ is the *dynamic viscosity*. First, we combine equation (2.3), (2.4), (2.5) and (2.6) together and drop the integral operator. It can be shown by expanding the right-hand side of (2.3) and applying $\frac{D\rho}{Dt} = 0$ that the *incompressible Navier equation* is defined as follows:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = \nabla\cdot\boldsymbol{\sigma} + \rho\mathbf{f} \tag{2.7}$$

Incompressible divergence of the stress tensor

To obtain the final model, we must expand the divergence of the Cauchy stress tensor and find the shear stress. This is done in two separate steps. First of all, we see that

$$\nabla\cdot(p\mathbf{I}) = \nabla p$$

The *rate of strain tensor* $\boldsymbol{\epsilon}$, used in the Cauchy stress tensor, is defined as

$$\boldsymbol{\epsilon} = \frac{1}{2}\left(\nabla\otimes\mathbf{u} + (\nabla\otimes\mathbf{u})^T\right)$$

Since our fluid is incompressible, $\nabla\cdot\mathbf{u} = 0$ as shown in (2.2), and this yields

$$2\mu\nabla\cdot\boldsymbol{\epsilon} = 2\mu\cdot\frac{1}{2}\begin{pmatrix}\nabla^2 u_x + \frac{\partial}{\partial x}(\nabla\cdot\mathbf{u})\\ \nabla^2 u_y + \frac{\partial}{\partial y}(\nabla\cdot\mathbf{u})\\ \nabla^2 u_z + \frac{\partial}{\partial z}(\nabla\cdot\mathbf{u})\end{pmatrix} = \mu\nabla^2\mathbf{u}$$

Hence, right-hand side of Navier's equation (2.7), which also corresponds to Newton's second law on integral form, becomes

$$\Sigma\mathbf{F} = \iiint_{\Omega} -\nabla p + \mu\nabla^2\mathbf{u} + \rho\mathbf{f}\,dV \tag{2.8}$$

Now, we can finally remove all the integral signs and combine the net force (2.8) with (2.7). This yields the incompressible Navier-Stokes equations:

$$\boxed{\begin{array}{lll}
\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \rho\mathbf{f}, & \Omega\times I & (2.9a)\\
\nabla\cdot\mathbf{u} = 0, & \Omega\times I & (2.9b)\\
\mathbf{u} = \mathbf{u}_D, & \Gamma_D^u\times I & (2.9c)\\
\frac{\partial\mathbf{u}}{\partial n} - \hat{\mathbf{n}}\cdot p = \mathbf{u}_N, & \Gamma_N^u\times I & (2.9d)\\
\alpha\mathbf{u} + \beta\left(\frac{\partial\mathbf{u}}{\partial n} - \hat{\mathbf{n}}\cdot p\right) = \mathbf{u}_R, & \Gamma_R^u\times I & (2.9e)\\
\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x}) & \overline{\Omega}\times\{t=0\} & (2.9f)
\end{array}}$$

The boundary $\partial\Omega$ is split up in three parts: Dirichlet (Γ_D^u), Neumann (Γ_N^u) and Robin (Γ_R^u) boundary. They are mutually disjoint, and $\partial\Omega = \Gamma_D^u \cup \Gamma_N^u \cup \Gamma_R^u$. The pressure is not subject to any boundary condition at all. The time-dependence requires that we also define an initial condition for the velocity field, but not the pressure. This is a quasilinear and parabolic partial differential equation system of second order. It shows that mass, momentum and energy is conserved in the fluid flow.

2.2 The Boussinesq equations

The derivation of coupled heat transfer and fluid flow using the Boussinesq-Oberbeck approximation is based on the approaches described in [34, 45, 56]. This is the main partial differential equation of interest for our purpose.

First law of thermodynamics

Our derivation starts with the connection between the *first law of thermodynamics* and Navier's equation (2.7). The *total energy* per unit volume is a conserved quantity. It is the sum of *internal* and *kinetic energy* per unit volume, and is defined as

$$e^t = e + \frac{1}{2}(\mathbf{u}\cdot\mathbf{u})$$

Heat transmission and work done by external forces acting on the fluid are the main sources causing variation of e^t. They give rise to the *convective energy flux* and *diffusive heat flux*, which are respectively defined as

$$\mathbf{F}_C = \rho e^t \mathbf{u} \quad , \quad \mathbf{F}_D = -k\nabla T$$

where k is the *thermal conductivity*. The last flux is in accord with *Fourier's law of heat conduction*, for there is no flux related to the motion of the fluid.

The *volume source* is the sum of heat source and work done by volume forces, and the *surface source* is work done by internal shear stress. They are respectively given by

$$Q_V = Q + \rho(\mathbf{f}\cdot\mathbf{u}) \quad , \quad Q_S = \boldsymbol{\sigma}\mathbf{u}$$

Collecting every contribution on integral conservation form yields

$$\frac{\partial}{\partial t}\iiint\limits_{\Omega} \rho e^t \, dV + \oiint\limits_{\partial\Omega} (\rho e^t \mathbf{u})\cdot\hat{\mathbf{n}}\, dS = \oiint\limits_{\partial\Omega} [\boldsymbol{\sigma}\mathbf{u} - (-k\nabla T)]\cdot\hat{\mathbf{n}}\, dS + \iiint\limits_{\Omega} Q + \rho(\mathbf{f}\cdot\mathbf{u})\, dV$$

Using Gauss' theorem and dropping integral signs results in the conservation law

$$\frac{\partial \rho e^t}{\partial t} + \nabla\cdot(\rho e^t \mathbf{u}) = k\nabla^2 T + \nabla\cdot(\boldsymbol{\sigma}\mathbf{u}) + Q + \rho(\mathbf{f}\cdot\mathbf{u}) \tag{2.10}$$

Applying incompressibility as simplification

Taking the dot product of (2.7) with $\mathbf{u}$ yields

$$\rho\frac{D}{Dt}\left(\frac{1}{2}(\mathbf{u}\cdot\mathbf{u})\right) = (\nabla\cdot\boldsymbol{\sigma})\cdot\mathbf{u} + \rho(\mathbf{f}\cdot\mathbf{u}) \tag{2.11}$$

If we apply $\nabla\cdot\mathbf{u} = 0$ and $\frac{D\rho}{Dt} = 0$, then the left-hand side of (2.10) can be rewritten as

$$\begin{aligned}&\frac{\partial}{\partial t}(\rho e) + \nabla\cdot(\rho e\mathbf{u}) + \rho\frac{D}{Dt}\left(\frac{1}{2}(\mathbf{u}\cdot\mathbf{u})\right)\\ =&\rho\frac{De}{Dt} + \rho\frac{D}{Dt}\left(\frac{1}{2}(\mathbf{u}\cdot\mathbf{u})\right)\end{aligned} \tag{2.12}$$

Subtracting (2.11) from (2.10) and applying the simplification (2.12) yields

$$\frac{\partial}{\partial t}(\rho e) + \nabla\cdot(\mathbf{u}\rho e) = -\nabla\cdot\mathbf{q} + Q + \Phi \tag{2.13}$$

Φ is the *dissipation function*, and it can be derived as follows:

$$\begin{aligned}\Phi &= \nabla\cdot(\boldsymbol{\sigma}\mathbf{u}) - (\nabla\cdot\boldsymbol{\sigma})\cdot\mathbf{u}\\ &= \nabla\cdot(-p\mathbf{u} + 2\mu\boldsymbol{\epsilon}\mathbf{u}) - (-\nabla p + 2\mu\nabla\cdot\boldsymbol{\epsilon})\cdot\mathbf{u}\\ &= -\nabla p\cdot\mathbf{u} - p\nabla\cdot\mathbf{u} + \nabla p\cdot\mathbf{u} + 2\mu(\nabla\cdot(\boldsymbol{\epsilon}\mathbf{u}) - (\nabla\cdot\boldsymbol{\epsilon})\mathbf{u})\\ &= 2\mu\sum_{i=1}^{3}\sum_{j=1}^{3}\epsilon_{ij}\frac{\partial u_j}{\partial x_i}\\ &= \mu\sum_{i=1}^{3}\sum_{j=1}^{3}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial u_j}{\partial x_i}\end{aligned}$$

Maxwell's thermodynamic relations

If we apply *Maxwell's thermodynamic relations* and the continuity equation (2.1), we can formulate the internal energy on differential form as follows:

$$de = \left.\frac{\partial e}{\partial T}\right|_{\eta} dT + \left.\frac{\partial e}{\partial \eta}\right|_{T} d\eta = C_v dT + \left(T\left.\frac{\partial p}{\partial T}\right|_{\eta} - p\right) d\eta \tag{2.14}$$

where C_v is the *heat capacity*. By applying the incompressibility of the fluid, we obtain

$$\begin{aligned}\rho\frac{De}{Dt} &= \rho C_v\frac{DT}{Dt} + \left[T\left.\frac{\partial p}{\partial T}\right|_{\eta} - p\right](\nabla\cdot\mathbf{u})\\ \frac{De}{Dt} &= C_v\frac{DT}{Dt}\end{aligned}$$

Boussinesq-Oberbeck approximation

The *Boussinesq-Oberbeck approximation* [2, 18] can be applied when the buoyancy force on the fluid is caused by density change due to very small temperature variation. We can assume that the pressure change will not affect the density significantly because the fluid is incompressible. Hence, the constant β for *isobaric volume expansion* is

$$\beta = -\frac{1}{\rho}\left(\frac{\partial \rho}{\partial T}\right)_p$$

If T_0 is the reference temperature, and the temperature change $\Delta T = T - T_0$ is small, we obtain the desired linear relation

$$\rho = \rho_0[1 - \beta(T - T_0)] \tag{2.15}$$

The dimension of β is K^{-1}, so the temperature should be measured in Kelvin. Since $\mathbf{f} = \mathbf{g}$, the gravity vector, the Boussinesq equations for the buoyancy-driven and non-isothermal flow can be stated as a parabolic system subject to a hyperbolic constraint:

$$\boxed{\begin{array}{lll}
\rho_0\left(\dfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} & & \\
\qquad + \rho_0[1-\beta(T-T_0)]\mathbf{g}, & \Omega\times I & (2.16a)\\
\rho_0 C\left(\dfrac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T\right) = k\nabla^2 T + Q + \Phi, & \Omega\times I & (2.16b)\\
\nabla\cdot\mathbf{u} = 0, & \Omega\times I & (2.16c)\\
 & & \\
\mathbf{u} = \mathbf{u}_D, & \Gamma_D^u\times I & (2.16d)\\
\dfrac{\partial \mathbf{u}}{\partial n} - \hat{\mathbf{n}}\cdot p = \mathbf{u}_N, & \Gamma_N^u\times I & (2.16e)\\
\alpha\mathbf{u} + \beta\left(\dfrac{\partial \mathbf{u}}{\partial n} - \hat{\mathbf{n}}\cdot p\right) = \mathbf{u}_R, & \Gamma_R^u\times I & (2.16f)\\
 & & \\
T = T_D, & \Gamma_D^T\times I & (2.16g)\\
\dfrac{\partial T}{\partial n} = T_N, & \Gamma_N^T\times I & (2.16h)\\
T - \mu\dfrac{\partial T}{\partial n} = T_R, & \Gamma_R^T\times I & (2.16i)\\
 & & \\
\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x}) & \overline{\Omega}\times\{t=0\} & (2.16j)\\
T(\mathbf{x},0) = T_0(\mathbf{x}) & \overline{\Omega}\times\{t=0\} & (2.16k)
\end{array}}$$

2.3 Dimensional analysis and scaling

Hydrodynamic quantities

Making a differential equation of any type *dimensionless* is a useful technique when we are studying complex mathematical problems. This allows us to determine whether the model can be further simplified. It will not depend explicitly on the physical parameters. The computer implementation becomes less complicated. Scaling an equation can be done in several ways. A good approach is ensuring that the highest order derivatives becomes independent of dimension parameters. This enables us to neglect specific quantities in the equations by using *regular perturbation*. In boundary layer theory, it can be necessary with *singular perturbation* on the highest order derivatives, and this is more complicated.

In fluid mechanics and heat transfer, there is a lot of dimensionless parameters for situation-specific applications [16, 21]. Before concerning the scaling of the Boussinesq equations, we will list some of these numbers, which have been used extensively in this thesis.

ν and α are respectively the *kinematic viscosity* and *thermal diffusivity*, defined as

$$\nu = \frac{\mu}{\rho_0} \quad , \quad \alpha = \frac{\kappa}{\rho_0 C}$$

Table 2.1. *List of important dimension parameters in fluid mechanics and heat transfer*

Number	**Formula**	**Description**
Brinkman number	$Br = \frac{\mu U^2}{k\Delta T}$	Compares heat from viscous dissipation with thermal conduction
Courant number	$Cr = \frac{U\Delta t}{L}$	Measure of artificial viscosity
Grashof number	$Gr = \frac{\alpha g \Delta T L^3}{\nu^2}$	Ratio of buoyancy forces to viscosity
Nusselt number	$Nu = \frac{L}{\delta T}\frac{\partial T}{\partial x}$	Measuring horizontal flux over the domain
Péclet number	$Pe = \frac{UL}{\kappa}$	Measure of convection in heat transfer
Prandtl number	$Pr = \frac{\nu}{\kappa}$	Material characteristic
Rayleigh number	$Ra = \frac{\beta g \Delta T L^3}{\nu\kappa}$	Compares destabilizing buoyant terms to stabilizing terms in transition problems
Reynold number	$Re = \frac{UL}{\nu}$	Measure of turbulence in fluid flow
Richardson number	$Ri = \frac{\alpha g \Delta T L}{U^2}$	Compares the buoyant force with the kinetic energy

Scaling the Advection-Diffusion equation with force

If Q is a heat source, and $\mathbf{a}$ is the convection field, we can use the scaling parameters

$$\mathbf{x}^* = \frac{\mathbf{x}}{L} \qquad T^* = \frac{T}{T_0} \qquad Q^* = \frac{TA}{L}Q \qquad \mathbf{a}^* = \frac{1}{A}\mathbf{a} \qquad t^* = \frac{A}{L}t$$

Inserting these expressions, simplifying and dropping all the primes, we obtain

$$\boxed{\frac{\partial T}{\partial t} - \frac{1}{Pe}\nabla^2 T + (\mathbf{a}\cdot\nabla)T = Q, \quad \text{in } \Omega} \tag{2.17}$$

When $Pe \to \infty$, we obtain the linear *advection equation*, and the transport goes in the direction of the vector $\mathbf{a}$. The boundary layers arising will not be so complicated to handle.

Scaling the Navier-Stokes equation with force

This scaling is very relevant for the numerical examples in chapter 6 where we need to verify that the code for solving the Navier-Stokes equation is correct. We assume that the force can be a non-trivial expression different from the gravity vector, and we use the following dimensionless parameters listed below:

$$\mathbf{x}^* = \frac{\mathbf{x}}{L} \qquad \mathbf{u}^* = \frac{\mathbf{u}}{U} \qquad p^* = \frac{p}{\rho U^2} \qquad \mathbf{f}^* = \frac{U^2}{L}\mathbf{f} \qquad t^* = \frac{U}{L}t$$

Inserting these expressions, simplifying and then dropping all the primes, we obtain the standard non-dimensional equations

$$\frac{\partial \mathbf{u}}{\partial t} - \frac{1}{Re}\nabla^2\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f}, \quad \text{in } \Omega \tag{2.18a}$$

$$\nabla\cdot\mathbf{u} = 0, \quad \text{in } \Omega \tag{2.18b}$$

When $Re \to \infty$, we obtain the inviscid *Euler equations*, and there is no diffusion at all. This will also create boundary layers and complex fluctuations in the fluid flow.

Two scalings of the Boussinesq equations

There are two main techniques for scaling the Boussinesq equation. In the first approach, we assume that none of the terms are neglected. The procedure is an extension of the approach in [43], with an additional formula for U as shown in [28]. The dimensionless parameters are listed below:

$$\mathbf{x}^* = \frac{\mathbf{x}}{L} \qquad \mathbf{u}^* = \frac{\mathbf{u}}{U} \qquad t^* = \frac{Ut}{L}$$

$$p^* = \frac{p - \rho_0(\mathbf{g}\cdot\mathbf{x})}{\rho_0 U^2} \qquad U = \sqrt{\beta g L \Delta T}$$

The formula for U is appropriate for Boussinesq approximation because it expresses the velocity by the buoyancy term. Putting all these quantities in (2.16), simplifying everything and dropping the primes yields the system

$$\boxed{\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} &= -\nabla p + \sqrt{\frac{Pr}{Ra}}\nabla^2\mathbf{u} - \frac{\mathbf{g}}{\|\mathbf{g}\|}T, \quad \text{in } \Omega && (2.19a)\\ \frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T &= \frac{1}{\sqrt{RaPr}}\left[\nabla^2 T + \frac{1}{k}(Q+\Phi)\right], \quad \text{in } \Omega && (2.19b)\\ \nabla\cdot\mathbf{u} &= 0, \quad \text{in } \Omega && (2.19c) \end{aligned}}$$

In most applications, the term Q does not appear, so we can discard it. Using the same approach described in [16], we list up the new dimensionless parameters:

$$\mathbf{x}^* = \frac{\mathbf{x}}{L} \qquad \mathbf{u}^* = \frac{\mathbf{u}}{U} \qquad t^* = \frac{Ut}{L}$$

$$p^* = \frac{p - \rho_0(\mathbf{g}\cdot\mathbf{x})}{\rho_0 U^2} \qquad T^* = \frac{T - T_0}{\Delta T}$$

The new difference is that the temperature is scaled instead of the velocity, and the new version of the Boussinesq equations becomes

$$\boxed{\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} &= \frac{1}{Re}\nabla^2\mathbf{u} - \nabla p - Ri\frac{\mathbf{g}}{\|\mathbf{g}\|}T, \quad \text{in } \Omega && (2.20a)\\ \frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T &= \frac{1}{Pe}\left(\nabla^2 T + Br\Phi\right), \quad \text{in } \Omega && (2.20b)\\ \nabla\cdot\mathbf{u} &= 0, \quad \text{in } \Omega && (2.20c) \end{aligned}}$$

Chapter 3

Analysis of the equations

The idea of the finite element method is converting the strong form of a partial differential equation into its equivalent weak form by variational calculus, and then approximate the solution by a linear combination of appropriate basis functions. To really ensure that the approach will work, we can analyze the solution's properties like existence, uniqueness and regularity, if possible. It can be shown that strong and weak solutions are equivalent under special circumstances. This is independent on the type of basis functions. In our case, it is possible to analyze the two parts of the Boussinesq equations separately.

3.1 Analysis of the convection-diffusion equation

Weak formulation

The initial/boundary value problem for the convection-diffusion equation is

$$\boxed{\begin{aligned}
&\frac{\partial T}{\partial t} + (\mathbf{u}\cdot\nabla)T - \nabla^2 T = Q, && \text{in } \Omega\times I && (3.1a)\\
&T = T_D && \text{on } \Gamma_D^T\times I && (3.1b)\\
&\frac{\partial T}{\partial n} = T_N, && \text{on } \Gamma_N^T\times I && (3.1c)\\
&(\hat{\mathbf{n}}\cdot\mathbf{u})T - \frac{\partial T}{\partial n} = T_R, && \text{on } \Gamma_R^T\times I && (3.1d)\\
&T(\mathbf{x},0) = T_0(\mathbf{x}), && \text{in } \overline{\Omega}\times\{t=0\} && (3.1e)
\end{aligned}}$$

We assume that there is no dissipation. Problem (3.1) is on conservative form because it is solenoidal, in accordance with the fluid's velocity [38]:

$$\nabla\cdot\mathbf{u} = 0 \quad\Longrightarrow\quad \nabla\cdot(\mathbf{u}T) = (\mathbf{u}\cdot\nabla)T$$

In order to possess a suitable solution, we must require the conditions $\mathbf{u} \in L^\infty(\Omega)$ and $f \in L^2(\Omega)$ [53]. We define S as a sufficiently smooth test function, and use *Galerkin projection* on the convection-diffusion equation:

$$\begin{aligned}
\frac{\partial T}{\partial t} + (\mathbf{u}\cdot\nabla)T - \nabla^2 T = Q \\
\int_\Omega \frac{\partial T}{\partial t} S + (\mathbf{u}\cdot\nabla T)S - (\nabla^2 T)S \, d\Omega = \int_\Omega QS \, d\Omega \\
\int_\Omega \frac{\partial T}{\partial t} S \, d\Omega + \underbrace{\int_\Omega (\mathbf{u}\cdot\nabla T)S + \nabla T\cdot\nabla S \, d\Omega}_{a(T,S)} = \underbrace{\int_\Omega QS \, d\Omega + \oint_{\partial\Omega} \frac{\partial T}{\partial n} S \, ds}_{F(S)}
\end{aligned}$$

We see that $a(\cdot,\cdot) : H^1(\Omega)\times H^1(\Omega) \to \mathbb{R}$ and $F(\cdot) : L^2(\Omega) \to \mathbb{R}$. Thus, we must find a solution $T \in V_T$ such that it satisfies

$$\int_\Omega \frac{\partial T}{\partial t} S \, d\Omega + a(T,S) = F(S), \quad S \in V_T \tag{3.2}$$

where V_T is the function space defined as

$$V_T = \left\{ S \in L^2\left(I, H^1(\Omega)\right) : \ S \text{ satisfies the boundary conditions (3.1b)-(3.1d)} \right\}$$

Continuity and coercivity

We see that $a(\cdot,\cdot)$ is bilinear and $f(\cdot)$ is linear. Continuity of these functionals follows from Hölder's inequality for the space L^p, and the inequality $\|\cdot\|_{L^2} \le \|\cdot\|_{H^k}$ which follows from the universal inclusion $W^{k,p} \subset L^p$:

$$\begin{aligned}
|a(T,S)| &= \left| \int_\Omega (\mathbf{u}\cdot\nabla T)S + \nabla T\cdot\nabla S \, d\Omega \right| \\
&\le \int_\Omega |(\mathbf{u}\cdot\nabla T)S| \, d\Omega + \int_\Omega |\nabla T\cdot\nabla S| \, d\Omega \\
&\le \|\mathbf{u}\|_{L^\infty} \|S\nabla T\|_{L^1} + \|\nabla T\cdot\nabla S\|_{L^1} \\
&\le \|\mathbf{u}\|_{L^\infty} \|\nabla T\|_{L^2}\|S\|_{L^2} + \|\nabla T\|_{L^2}\|\nabla S\|_{L^2} \\
&\le \|\mathbf{u}\|_{L^\infty} \|T\|_{H^1}\|S\|_{H^1} + \|T\|_{H^1}\|S\|_{H^1} \\
&= C_T \|T\|_{H^1}\|S\|_{H^1}
\end{aligned}$$

$$\begin{aligned}
|F(S)| &= \left| \int_\Omega QS \, d\Omega + \oint_{\partial\Omega} \frac{\partial T}{\partial n} S \, ds \right| \\
&\le \|QS\|_{L^1} + \|gS\|_{L^1} \\
&\le \|Q\|_{L^2}\|S\|_{L^2} + \|g\|_{L^2}\|S\|_{L^2} \\
&\le C_Q \|S\|_{H^1}
\end{aligned}$$

To prove coercivity of $a(\cdot,\cdot)$, we use the incompressibility constraint $\nabla \cdot \mathbf{u} = 0$ and Poincaré's inequality (B.11) on the non-symmetric and symmetric parts, respectively:

$$\begin{aligned}
\int_\Omega T(\mathbf{u} \cdot \nabla T)\, d\Omega &= \frac{1}{2} \int_\Omega \mathbf{u} \cdot \nabla (T^2)\, d\Omega \\
&= \frac{1}{2} \oint_{\partial\Omega} T^2 (\mathbf{u} \cdot \hat{\mathbf{n}})\, ds - \frac{1}{2} \int_\Omega T^2 (\nabla \cdot \mathbf{u})\, d\Omega \\
&= \frac{1}{2} \oint_{\partial\Omega} T^2 (\mathbf{u} \cdot \hat{\mathbf{n}})\, ds
\end{aligned}$$

$$\begin{aligned}
\|T\|_{H^1}^2 &= \|T\|_{L^2}^2 + \|\nabla T\|_{L^2}^2 \\
&\leq C_\Omega^2 \|\nabla T\|_{L^2}^2 + \|\nabla T\|_{L^2}^2 \\
&= (C_\Omega^2 + 1) \|\nabla T\|_{L^2}^2 \\
\alpha \|T\|_{H^1}^2 &\leq \|\nabla T\|_{L^2}^2
\end{aligned}$$

The first derivation shows that coercivity is ensured if $\mathbf{u} \cdot \hat{\mathbf{n}} > 0$ and $\text{meas}(\Gamma_D^T) > 0$ are satisfied. We just combine both inequalities directly to obtain

$$\begin{aligned}
\alpha \|T\|_{H^1}^2 &\leq \|\nabla T\|_{L^2}^2 + \frac{1}{2} \oint_{\partial\Omega} T^2 (\mathbf{u} \cdot \hat{\mathbf{n}})\, ds \\
&= a(T, T)
\end{aligned}$$

Thus, we have shown that $a(\cdot,\cdot)$ and $F(\cdot)$ are always continuous, and $a(\cdot,\cdot)$ is coercive as long as $\mathbf{u} \cdot \hat{\mathbf{n}} > 0$ on the boundary $\partial\Omega$.

Regularity through energy estimate

To prove regularity, we must establish two inequalities first. Since $\|\cdot\|_{L^2} \leq \|\cdot\|_{H^k}$, the coercivity of $a(\cdot,\cdot)$ implies that we have

$$\alpha \|T\|_{L^2}^2 \leq a(T, T) \tag{3.3}$$

If $\epsilon = \frac{1}{2\alpha}$, we can use Young's inequality (B.3) to obtain

$$\begin{aligned}
\left| \int_\Omega fT\, d\Omega \right| &\leq \|f\|_{L^2} \|T\|_{L^2} \\
&\leq \frac{1}{2\alpha} \|f\|_{L^2}^2 + \frac{\alpha}{2} \|T\|_{L^2}^2
\end{aligned} \tag{3.4}$$

Combining (3.3) and (3.4) together yields

$$\begin{aligned}
&\int_\Omega \frac{\partial T}{\partial t} T \, d\Omega + a(T,T) = F(T) \\
&\frac{1}{2}\frac{\partial}{\partial t}\|T\|^2_{L^2} + \alpha\|T\|^2_{L^2} \le \frac{1}{2\alpha}\|f\|^2_{L^2} + \frac{\alpha}{2}\|T\|^2_{L^2} \\
&\frac{\partial}{\partial t}\|T\|^2_{L^2} + \alpha\|T\|^2_{L^2} \le \frac{1}{\alpha}\|f\|^2_{L^2} \\
&\|T\|^2_{L^2} + \alpha\int_0^\tau \|T\|^2_{L^2}\, dt \le \frac{1}{\alpha}\int_0^\tau \|f\|^2_{L^2}\, dt + \|T_0\|^2_{L^2} \qquad (3.5)
\end{aligned}$$

The energy estimate (3.5) shows that any solution of (3.1) is regular and stable.

Uniqueness of the solution

Let us assume that (3.1) has two solutions T_1 and T_2. If $w = T_1 - T_2$, then the equation for w will have homogeneous initial and boundary conditions, and (3.5) takes the form

$$\|w\|^2_{L^2} \le -\alpha \int_0^\tau \|w\|^2_{L^2}\, dt$$

The only way to satisfy this inequality is claiming that $w = 0$, i.e. $T_1 = T_2$. Hence, the solution of the convection-diffusion equation is unique.

Convergence of the solution

The bilinear form in (3.2) time-dependent and non-symmetric, so it does not define an inner product. Since the equation involves $\frac{\partial}{\partial t}$, the time-dependent energy norm becomes

$$\|T\|^2_E = \|T\|^2_{L^2(I,H^1)} + \left\|\frac{\partial T}{\partial t}\right\|^2_{L^2(I,H^1)}$$

A classical a-priori estimate for this norm [25] is

$$\|T\|_E \le C\left(\|f\|_{L^2(I,H^{-1})} + \|T_0\|_{L^2}\right)$$

From this one, it is possible to show by Galerkin orthogonality and the L^2-projection onto V_T that the solution converges [20], due to the following inequality:

$$\|T - T_h\|_E \le Ch\left(\|T\|_{L^2(I,H^2)} + \left\|\frac{\partial T}{\partial t}\right\|_{L^2(I,L^2)}\right)$$

3.2 Analysis of the Navier-Stokes equations

Weak formulation

The initial/boundary value problem for the Navier-Stokes equations is

$$\boxed{\begin{aligned}
&\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} - \nabla^2\mathbf{u} = -\nabla p + \mathbf{f}, && \text{in } \Omega\times I && (3.6a)\\
&\nabla\cdot\mathbf{u} = 0, && \text{in } \Omega\times I && (3.6b)\\
&\mathbf{u} = \mathbf{u}_D, && \text{on } \Gamma_D^u\times I && (3.6c)\\
&\left(\frac{\partial \mathbf{u}}{\partial n} - \hat{\mathbf{n}}\cdot p\right) = \mathbf{u}_N, && \text{on } \Gamma_N^u\times I && (3.6d)\\
&\alpha\mathbf{u} + \beta\left(\frac{\partial \mathbf{u}}{\partial n} - \hat{\mathbf{n}}\cdot p\right) = \mathbf{u}_R, && \text{on } \Gamma_R^u\times I && (3.6e)\\
&\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x}), && \text{in } \overline{\Omega}\times\{t=0\} && (3.6f)
\end{aligned}}$$

where $\mathbf{f}$ is the force on the fluid. When we couple the Navier-Stokes equations with the convection-diffusion equation, we replace $\mathbf{f}$ by $h(T) = \rho_0\left[1-\beta(T-T_0)\right]\mathbf{g}$, the Boussinesq approximation. In any case, the force is independent of $\mathbf{u}$.

This time, we must define a pair of sufficiently smooth test functions $(\mathbf{v}, q)$ on a system of equations. Applying Galerkin projection again, we obtain

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} - \nabla^2\mathbf{u} = -\nabla p + \mathbf{f}$$
$$\int_\Omega \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla\mathbf{u}) - \nabla^2\mathbf{u}\right)\cdot\mathbf{v}\,d\Omega = \int_\Omega (-\nabla p + \mathbf{f})\cdot\mathbf{v}\,d\Omega$$
$$\int_\Omega \frac{\partial \mathbf{u}}{\partial t}\mathbf{v}\,d\Omega + \int_\Omega \nabla\mathbf{u}\cdot\nabla\mathbf{v}\,d\Omega + \int_\Omega (\mathbf{u}\cdot\nabla\mathbf{u})\mathbf{v}\,d\Omega = \int_\Omega p\nabla\cdot\mathbf{v}\,d\Omega + \int_\Omega \mathbf{f}\cdot\mathbf{v}\,d\Omega$$

The incompressibility criterion must also be taken into account. It becomes

$$\nabla\cdot\mathbf{u} = 0$$
$$\int_\Omega (\nabla\cdot\mathbf{u})q\,d\Omega = 0$$

In order to make the pressure unique, we must impose an important compatibility criterion:

$$\int_\Omega p\,d\Omega = \text{const.} \tag{3.7}$$

We require that q cancels over constant values, and this can only be achieved if the average pressure above is zero. This simple integral equality will be of high relevance during the implementation of the numerical method, which involves solving saddle-point systems. It shows that the pressure belongs to L_0^2, which ensures sufficient stability and uniqueness.

We collect the weak formulations as a system of equations:

$$\int_\Omega \frac{\partial u_x}{\partial t} v_x \, d\Omega + \underbrace{\int_\Omega \nabla u_x \cdot \nabla v_x \, d\Omega}_{a(\cdot,\cdot)} - \underbrace{\int_\Omega p \frac{\partial v_x}{\partial x} \, d\Omega}_{b(\cdot,\cdot)} + \underbrace{\int_\Omega (\mathbf{u} \cdot \nabla u_x) v_x \, d\Omega}_{c(\cdot,\cdot,\cdot)} = \underbrace{\int_\Omega f_x v_x \, d\Omega}_{f(\cdot)}$$

$$\int_\Omega \frac{\partial u_y}{\partial t} v_y \, d\Omega + \underbrace{\int_\Omega \nabla u_y \cdot \nabla v_y \, d\Omega}_{a(\cdot,\cdot)} - \underbrace{\int_\Omega p \frac{\partial v_y}{\partial y} \, d\Omega}_{b(\cdot,\cdot)} + \underbrace{\int_\Omega (\mathbf{u} \cdot \nabla u_y) v_y \, d\Omega}_{c(\cdot,\cdot,\cdot)} = \underbrace{\int_\Omega f_y v_y \, d\Omega}_{f(\cdot)}$$

$$\int_\Omega \frac{\partial u_z}{\partial t} v_z \, d\Omega + \underbrace{\int_\Omega \nabla u_z \cdot \nabla v_z \, d\Omega}_{a(\cdot,\cdot)} - \underbrace{\int_\Omega p \frac{\partial v_z}{\partial z} \, d\Omega}_{b(\cdot,\cdot)} + \underbrace{\int_\Omega (\mathbf{u} \cdot \nabla u_z) v_z \, d\Omega}_{c(\cdot,\cdot,\cdot)} = \underbrace{\int_\Omega f_z v_z \, d\Omega}_{f(\cdot)}$$

$$\underbrace{\int_\Omega \frac{\partial u_x}{\partial x} q \, d\Omega}_{b(\cdot,\cdot)} + \underbrace{\int_\Omega \frac{\partial u_y}{\partial y} q \, d\Omega}_{b(\cdot,\cdot)} + \underbrace{\int_\Omega \frac{\partial u_z}{\partial z} q \, d\Omega}_{b(\cdot,\cdot)} = 0$$

We see that the forms and scalar functionals are

$$\begin{aligned} a(\cdot,\cdot): &\quad H^1(\Omega) \times H^1(\Omega) \to \mathbb{R} \\ b(\cdot,\cdot): &\quad H^1(\Omega) \times L_0^2(\Omega) \to \mathbb{R} \\ c(\cdot,\cdot,\cdot): &\quad H^1(\Omega)^3 \times H^1(\Omega) \times H^1(\Omega) \to \mathbb{R} \\ f(\cdot): &\quad L^2(\Omega) \to \mathbb{R} \end{aligned}$$

It can be appropriate to define the functionals and forms as vectorial mappings:

$$\begin{aligned} A(\cdot,\cdot): &\quad H^1(\Omega)^3 \times H^1(\Omega)^3 \to \mathbb{R}^3 \\ B(\cdot,\cdot): &\quad H^1(\Omega)^3 \times L_0^2(\Omega) \to \mathbb{R} \\ C(\cdot,\cdot,\cdot): &\quad H^1(\Omega)^3 \times H^1(\Omega)^3 \times H^1(\Omega)^3 \to \mathbb{R}^3 \\ F(\cdot): &\quad L^2(\Omega)^3 \to \mathbb{R}^3 \end{aligned}$$

Find $(\mathbf{u}, p) \in V_u \times V_p$ for each $t > 0$ such that they satisfy

$$\int_\Omega \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} \, d\Omega + A(\mathbf{u}, \mathbf{v}) - B(\mathbf{v}, p)^T + C(\mathbf{u}, \mathbf{u}, \mathbf{v}) = F(\mathbf{v}) \quad \mathbf{v} \in V \tag{3.8a}$$

$$B(\mathbf{u}, q) = 0 \qquad q \in W \tag{3.8b}$$

where V_u and V_p are the function spaces defined as

$$V_u = \left\{ \mathbf{v} \in L^2(I, H^1(\Omega)^3) : \ S \text{ satisfies the boundary conditions (3.6c)-(3.6e)} \right\}$$

$$V_p = \left\{ q \in L^2(I, L_0^2(\Omega)) : \int_\Omega q \, dA = 0 \right\}$$

Continuity and coercivity

Continuity of $A(\cdot,\cdot)$, $B(\cdot,\cdot)$ and $F(\cdot)$ follows from Hölder's inequality and the inequality $\|\cdot\|_{L^2} \leq \|\cdot\|_{H^k}$. Using the same procedure as previously, we obtain

$$\begin{aligned}
|A(\mathbf{u},\mathbf{v})| &= \left|\int_\Omega \nabla\mathbf{u}\cdot\nabla\mathbf{v}\,d\Omega\right| \\
&\leq \|\nabla\mathbf{u}\cdot\nabla\mathbf{v}\|_{L^1} \\
&\leq \|\nabla\mathbf{u}\|_{L^2}\|\nabla\mathbf{v}\|_{L^2} \\
&= \|\mathbf{u}\|_{H^1}\|\mathbf{v}\|_{H^1}
\end{aligned}$$

$$\begin{aligned}
|B(\mathbf{v},p)| &= \left|\int_\Omega p\nabla\cdot\mathbf{v}\,d\Omega\right| \\
&\leq \|p\nabla\cdot\mathbf{v}\|_{L^1} \\
&\leq \|p\|_{L^2}\|\nabla\cdot\mathbf{v}\|_{L^2}
\end{aligned}$$

$$\begin{aligned}
|F(\mathbf{v})| &\leq \|\mathbf{f}\|_{L^2}\|\mathbf{v}\|_{L^2} \\
&\leq C_F\|\mathbf{v}\|_{H^1}
\end{aligned}$$

For $C(\cdot,\cdot,\cdot)$, we need to decompose the integrand by double sums:

$$\begin{aligned}
|C(\mathbf{u},\mathbf{u},\mathbf{v})| &= \left|\int_\Omega (\mathbf{u}\cdot\nabla\mathbf{u})\mathbf{v}\,d\Omega\right| \\
&= \left|\sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Omega u_{x_j}\frac{\partial u_{x_i}}{\partial x_j}v_{x_i}\,d\Omega\right| \\
&\leq \sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Omega \left|u_{x_j}\frac{\partial u_{x_i}}{\partial x_j}v_{x_i}\right|\,d\Omega \\
&= \sum_{i=1}^{3}\sum_{j=1}^{3}\left\|u_{x_j}\frac{\partial u_{x_i}}{\partial x_j}v_{x_i}\right\|_{L^1}
\end{aligned}$$

Next, we apply the inclusion $L^4(\Omega)\subset L^2(\Omega)$, which is valid because the domain Ω has finite measure. The *Sobolev embedding theorem* provides that $\|\cdot\|_{L^4}\leq\|\cdot\|_{H^1}$ because $H^1(\Omega)\hookrightarrow L^4(\Omega)$ up to three spatial dimensions [6]. Using Hölder's inequality twice, we obtain a component-wise estimate:

$$\begin{aligned}
\left\|u_{x_j}\frac{\partial u_{x_i}}{\partial x_j}v_{x_i}\right\|_{L^1} &\leq \left\|\frac{\partial u_{x_i}}{\partial x_j}\right\|_{L^2}\|u_{x_j}v_{x_i}\|_{L^2} \\
&\leq \left\|\frac{\partial u_{x_i}}{\partial x_j}\right\|_{L^2}\|u_{x_j}\|_{L^4}\|v_{x_i}\|_{L^4} \\
&\leq |\mathbf{u}|_{H^1}\,\|u_{x_j}\|_{H^1}\|v_{x_i}\|_{H^1}
\end{aligned}$$

The Sobolev norms and seminorms will always satisfy the inequality $|\cdot|_{W^{k,p}} \leq \|\cdot\|_{W^{k,p}}$. Since $\mathbf{u}$ is a vector function, its corresponding Sobolev tensor norm becomes

$$\|\mathbf{u}\|^2_{W^{k,p}} = \|u_x\|^2_{W^{k,p}} + \|u_y\|^2_{W^{k,p}} + \|u_z\|^2_{W^{k,p}}$$

By combining these two important results, we obtain

$$|\mathbf{u}|_{H^1} \, \|u_{x_j}\|_{H^1} \|v_{x_i}\|_{H^1} \leq \|\mathbf{u}\|^2_{H^1} \|\mathbf{v}\|_{H^1}$$

Finally, we can insert this inequality in the double sum to end up with the desired estimate:

$$|C(\mathbf{u}, \mathbf{u}, \mathbf{v})| \leq 9\|\mathbf{u}\|^2_{H^1} \|\mathbf{v}\|_{H^1}$$

The nonlinear convection term $C(\cdot, \cdot, \cdot)$ has also the *vanishing property*:

$$\begin{aligned} |C(\mathbf{u}, \mathbf{u}, \mathbf{u})| &= \langle \mathbf{u} \cdot \nabla \mathbf{u}, \mathbf{u} \rangle_{L^2} \\ &= -\frac{1}{2} \langle \nabla \cdot \mathbf{u}, |\mathbf{u}|^2 \rangle_{L^2} \\ &= 0 \end{aligned}$$

Coercivity of $A(\cdot, \cdot)$ follows in almost the same way as for the temperature T:

$$\alpha \|\mathbf{u}\|^2_{H^1} \leq A(\mathbf{u}, \mathbf{u})$$

Hence, the weak formulation is continuous and coercive.

Energy estimate for regularity

The inequality $\|\cdot\|_{L^2} \leq \|\cdot\|_{H^k}$ implies that

$$\alpha \|\mathbf{u}\|^2_{L^2} \leq A(\mathbf{u}, \mathbf{u}) \tag{3.9}$$

From Young's inequality (B.3), we obtain the following result:

$$\begin{aligned} |F(\mathbf{u})| &\leq \|\mathbf{f}\|_{L^2} \|\mathbf{u}\|_{L^2} \\ &\leq \frac{1}{2\alpha} \|\mathbf{f}\|^2_{L^2} + \frac{\alpha}{2} \|\mathbf{u}\|^2_{L^2} \end{aligned}$$

Combining everything yields the following energy estimate ensuring regularity.

$$\begin{aligned} &\int_\Omega \frac{\partial \mathbf{u}}{\partial t} \mathbf{u} \, d\Omega + A(\mathbf{u}, \mathbf{u}) + C(\mathbf{u}, \mathbf{u}, \mathbf{u}) = B(\mathbf{u}, p) + F(\mathbf{u}) \\ &\frac{1}{2} \frac{\partial}{\partial t} \|\mathbf{u}\|^2_{L^2} + \alpha \|\mathbf{u}\|^2_{L^2} \leq \frac{1}{2\alpha} \|\mathbf{f}\|^2_{L^2} + \frac{\alpha}{2} \|\mathbf{u}\|^2_{L^2} \\ &\frac{\partial}{\partial t} \|\mathbf{u}\|^2_{L^2} + \alpha \|\mathbf{u}\|^2 \leq \frac{1}{\alpha} \|\mathbf{f}\|^2_{L^2} \\ &\|\mathbf{u}\|^2_{L^2} + \alpha \int_0^\mathcal{T} \|\mathbf{u}\|^2 \, dt \leq D_u \int_0^\mathcal{T} \|\mathbf{f}\|^2_{L^2} \, dt + \|\mathbf{u}_0\|^2_{L^2} \end{aligned}$$

Hence, the solution of the Navier-Stokes equation is regular for low Reynolds number.

Uniqueness of the solution

There is no general proof for the uniqueness of solutions of the Navier-Stokes equations. In some special cases, it is possible to show uniqueness under strict circumstances. The most famous result until now is *Leray's theorem*. For more details, we refer to [6, 61].

According to Leray's theorem, if the pressure and velocity field satisfies some special functional properties, and the domain Ω is finite, then the solution of the Navier-Stokes' equations are unique in two dimensions. For three dimensions, the solution exists, but the uniqueness is not clear. In both cases, the velocity field will satisfy the energy estimate derived previously, so we have regularity too.

When we start with the numerical convergence studies in chapter 6, we will solve the Navier-Stokes' and Boussinesq equations in two spatial dimensions. The manufactured reference solutions have a characteristic structure which makes the initial/boundary value problem unique, and this makes it easier to verify numerically that the implementation of the finite element code is correct.

Chapter 4

Isogeometric Analysis

This is the part of the thesis that really begins to focus on the new technique for solving our partial differential equations. First, we give a full description of the main ideas of isogeometric analysis and why it is more appropriate to use for our problem rather than the classical finite element method. Then, we focus on the main properties of B-splines and NURBS, which constitute the core of our numerical convergence studies.

4.1 A new paradigm for the finite element method

Main ideas of Isogeometric Analysis

Isogeometric Analysis (IGA) was introduced by Hughes, Bazilevs and Cottrell in their landmark paper from 2005 [36] and formalized in the book [12]. It is a new finite element method under current research and has many efficient advantages that are not available in the classical finite element method. The main idea of isogeometric analysis is using splines as interpolating shape functions. These can be B-splines, NURBS, T-splines, L-splines, M-splines, LR B-splines etc, depending on the specific situation.

An important feature of isogeometric analysis is that the procedure of creating the weak formulation of a partial differential equation, the fundamental cornerstone of finite element modelling, is the same as before, in accordance with standard finite element theory. The basic structure of the discretization matrices are quite similar, so the assembly processes will not be so different. But the real power of isogeometric analysis lies in its ability of discretizing the domain exactly, which is not always possible to achieve with the classical finite element methodology. Several articles have confirmed that this new method works well in structural engineering, fluid mechanics, electromagnetism and other computational mechanics disciplines.

Completeness of the isoparametric approach

The *isoparametric concept*, formally introduced by Zienkiewicz [63], is one of the most important facilities of modern finite element technology. It requires that we use the same shape functions for approximating the unknown solution u_h and creating a suitable mesh the domain Ω. Hence, the shape functions define both the elements' geometric shape and displacement within them. To achieve this, the total number of interpolation points must equal the total number of geometric nodes. The interpolation elements and the geometric elements will therefore coincide with each other at their nodes [46].

In the classical finite element method, the continuity between the interpolation elements is usually C^0. Rising this to higher order continuity is challenging and yields often new interpolation elements that are expensive to use with respect to computational effort and implementation. But in isogeometric analysis, this is not a problem at all. The usage of splines as basis functions yields higher continuity between the elements, and NURBS can represent conic sections exactly. The same spline basis is used both for the geometry and the unknown solution field, so the isogeometric elements are fully isoparametric for any continuity. Because of this characteristic property, it is said that isogeometric analysis really creates a bridge over the existing gap between *Computer Assisted Design* (CAD) and *Finite Element Analysis* (FEA). We obtain an automatic and efficient geometry-to-mesh mapping by using splines. This improves mesh optimization and removes the need for time-consuming conversion between different shape functions used for approximation and visualization of the unknown solution.

In the old finite element methodology, we choose basis functions first to interpolate the numerical solution, and then we use them to mesh the geometry of the domain. But in isogeometric analysis, this procedure is fully reversed. We choose suitable functions for the geometry, and then we use them to interpolate the unknown solution afterwards. Hence, our approximation is geometry independent. If the mesh on the geometry is as exact as possible, then the numerical error is reduced quite much. This new advantage is not fully available in the classical finite element method because the shape functions are straight-edged, so curved edges on the domain are only approximated.

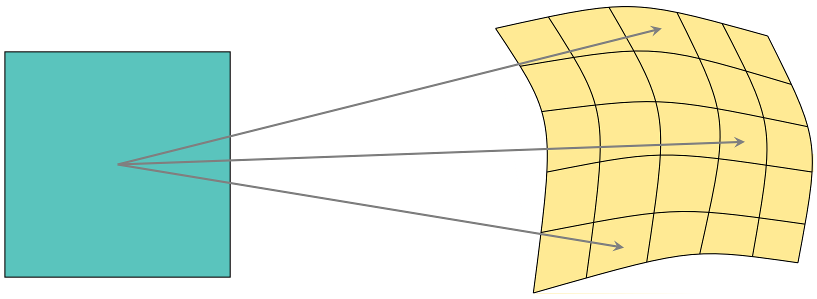
Classical Finite Element Method:	Geometry	$\Leftarrow$	Solution field
Isogeometric Finite Element Method:	Geometry	$\Rightarrow$	Solution field

Splines are appropriate to use because they form a partition of unity, providing stability and better control of the computations. By combining this property with the isoparametric concept, it can be shown that isogeometric analysis is complete. This means that the unknown solution u_h can be written as a linear infinite sum of the basis functions. Another advantage is that we can vary the continuity easily from 0 to $p - 1$, where p is the spline's polynomial degree [12].

Characteristic approximation features

Isogeometric analysis provides higher continuity of the numerical solution such that error oscillation becomes lower than classical finite element functions [12]. The error is reduced significantly for smooth problems. By using proper adaptive refinement, we can achieve the same effect for non-smooth problems [40]. The standard refinement strategies from the finite element method are fully incorporated in isogeometric analysis, and this makes reduction of numerical error easy. Recently, there has also been intensive research on local refinement of splines [27, 40, 24]. Isogeometric analysis provides also k-refinement, which does not exist in the classical finite element method. It has also been shown that splines generate commutative *de Rham diagrams*, and this makes approximation of the differential operators better [10].

In contrast to the classical finite element method, the B-spline parameter space is local to a patch on the domain, not a single individual element. This is time-saving because we do not need to define several different maps for every element, only for specific parts of the global domain of the partial differential equation [12].


(a) *Isoparametric mapping using classical finite element method*

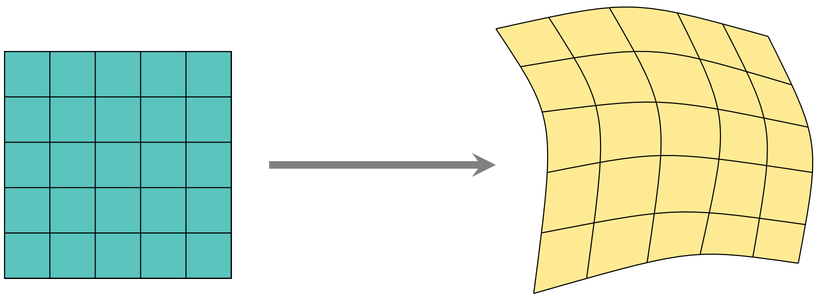

(b) *Isoparametric mapping using isogeometric analysis*

Figure 4.1. *Comparision of isoparametric mappings in the finite element methods*

4.2 B-splines

4.2.1 Univariate B-splines

Definition and properties of B-splines

A *spline function* is a piecewise defined but globally differentiable function on an interval $[a, b]$. We construct it from a *knot vector*, an ordered sequence of nondecreasing numbers (knots), $\Xi = \{\xi_0, \xi_1, \ldots, \xi_m\}$, where $m = n + p$. From this vector, we can define a *B-spline* (basic spline) of polynomial degree p, consisting of $n + 1$ basis functions. The m knots are classified as follows:

1. **End knots**: $\xi_0, \xi_1, \ldots \xi_p = a$ and $\xi_{m-p}, \xi_{m-p+1}, \ldots \xi_m = b$.
2. **Interior knots**: $\xi_{p+1}, \xi_{p+2}, \xi_{p+3}, \ldots, \xi_{m-p-1}$.

The B-splines are defined by the recursive *Cox-de Boor formula* [14]:

$$N_{i,d}(\xi) = \frac{\xi - \xi_i}{\xi_{i+d} - \xi_i} N_{i,d-1}(\xi) + \frac{\xi_{i+d+1} - \xi}{\xi_{i+d+1} - \xi_{i+1}} N_{i+1,d-1}(\xi) \tag{4.2a}$$

$$N_{i,0}(\xi) = \chi_{[\xi_i, \xi_{i+1})} = \begin{cases} 1 & \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{4.2b}$$

We can write the general form of a spline function as

$$s(x) = \sum_{i=0}^{n} c_i N_{i,p}(\xi) \tag{4.3}$$

The derivative of B-splines is given by

$$\frac{d}{d\xi} N_{i,p}(\xi) = \frac{p}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \tag{4.4}$$

This formula can be generalized to any order as follows [52]:

$$\frac{d^k}{d\xi^k} N_{i,p}(\xi) = \frac{p!}{(p-k)!} \sum_{j=0}^{k} a_{k,j} N_{i+j,p-k}(\xi) \tag{4.5a}$$

$$a_{0,0} = 1 \tag{4.5b}$$

$$a_{k,0} = \frac{a_{k-1,0}}{\xi_{i+p-k+1} - \xi_i} \tag{4.5c}$$

$$a_{k,j} = \frac{a_{k-1,j} - a_{k-1,j-1}}{\xi_{i+p+j-k+1} - \xi_{i+j}} \qquad 1 \leq j \leq k-1 \tag{4.5d}$$

$$a_{k,k} = -\frac{a_{k-1,k-1}}{\xi_{i+p+1} - \xi_{i+k}} \tag{4.5e}$$

From equation (4.4), we can define the first derivative of a spline function as

$$s'(x) = \sum_{i=0}^{n} d_i N_{i,p-1}(\xi) \qquad , \qquad d_i = \begin{cases} \frac{c_0 p}{t_p - t_0} & i = 0 \\ p\left(\frac{c_i - c_{i+1}}{t_{i+p} - t_i}\right) & 1 \le i \le n-1 \\ \frac{c_n p}{t_{n+p} - t_n} & i = n \end{cases} \tag{4.6}$$

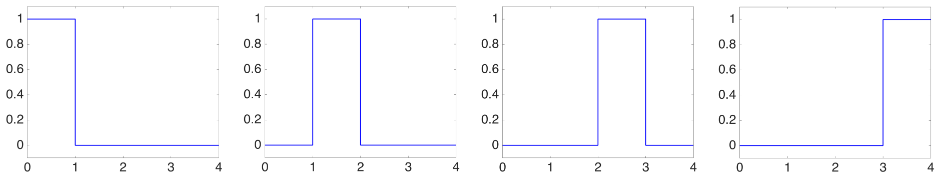
The B-splines have many important properties. We refer to [49, 52] for the proofs.

1. **Uniqueness**: $N_{i,p}$ depends only on the knots $\xi_i, \xi_{i+1}, \ldots \xi_{i+p+1}$.
2. **Positivity**: $N_{i,p} > 0, \quad \xi \in (\xi_i, \xi_{i+p+1})$.
3. **Local support**: supp$(N_{i,p}) = (\xi_i, \xi_{i+p+1})$.
4. **Openness**: $\bar{\xi} = \xi_{i+1} = \cdots = \xi_{i+p} < \xi_{i+p+1} \implies N_{i,p}(\bar{\xi}) = \delta_{ij}$.
5. **Continuity**: $\bar{\xi} \in \Xi$ has multiplicity $m \implies N_{i,p} \in C^{p-m}$ at $\bar{\xi}$. Otherwise, $N_{i,p}$ is a smooth polynomial between the knots.
6. **Stability**: B-splines form a stable and linearly independent basis for all piecewise polynomials on closed intervals.
7. **Partition of unity**: B-splines form a partition of unity:

$$\sum_{i=1}^{n} N_{i,p}(\xi) = 1 \qquad \forall \xi \in \Xi, p \in \mathbb{N}$$

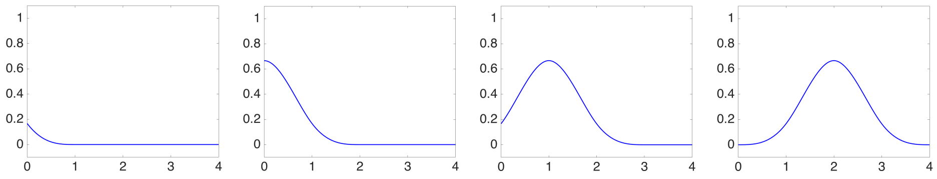
$$\begin{tikzcd}
\mathbf{P}^0_{m-p} \arrow[dd] & \mathbf{P}^0_{m-p+1} \arrow[dd] \arrow[ddl] & \mathbf{P}^0_{m-p+2} \arrow[dd] \arrow[ddl] & \mathbf{P}^0_{m-p+3}\cdots \arrow[ddl] & \mathbf{P}^0_{m} \\
 & & & \vdots & \\
\mathbf{P}^1_{m-p+1} \arrow[dd] & \mathbf{P}^1_{m-p+2} \arrow[dd] \arrow[ddl] & \mathbf{P}^1_{m-p+3}\cdots \arrow[ddl] & \mathbf{P}^1_{m} & \\
 & & \vdots & & \\
\mathbf{P}^2_{m-p+2} & \mathbf{P}^2_{m-p+3}\cdots & \mathbf{P}^2_{m} & & \\
\vdots & \vdots & & & \\
\mathbf{P}^{p-1}_{m-1} \arrow[d] & \mathbf{P}^{p-1}_{m} \arrow[dl] & & & \\
\mathbf{P}^{p}_{m} & & & &
\end{tikzcd}$$

Figure 4.2. *Visualization of the Cox-de Boor algorithm for B-spline evaluation*

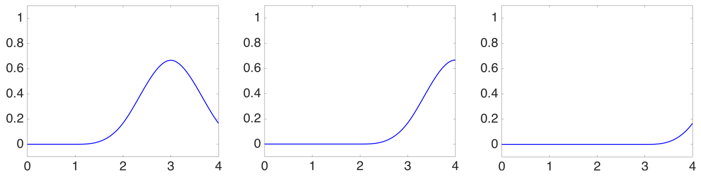


(a) $N_{0,0}$ **(b)** $N_{1,0}$ **(c)** $N_{2,0}$ **(d)** $N_{3,0}$

Figure 4.3. *B-splines of order 0 on* $\Xi = \{0,0,0,0,1,2,3,4,4,4,4\}$



(a) $N_{0,3}$ **(b)** $N_{1,3}$ **(c)** $N_{2,3}$ **(d)** $N_{3,3}$



(e) $N_{4,3}$ **(f)** $N_{5,3}$ **(g)** $N_{6,3}$

Figure 4.4. *B-splines of order 3 on* $\Xi = \{0,0,0,0,1,2,3,4,4,4,4\}$

We see from figure 4.3 and 4.4 that the Cox-de Boor algorithm starts with defining four constant functions on disjoint intervals, and then they are combined recursively into seven new functions of degree 3. In general, this algorithm can be visualized as shown above in figure 4.2. The process is recursive, but we can use dynamic programming and implement it with for-loops to obtain polynomial running time.

Spaces of univariate B-splines

We define a uniform partition on the interval $[a, b]$ as follows:

$$\Delta : \quad a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$$

p is the polynomial degree, and r is the smoothness degree. From these quantities, we can define the *univariate spline space* as follows:

$$\mathbb{S}_r^p(\Delta) = \left\{ s(x) = \sum_{i=1}^{n} c_i N_i(x) \quad : \quad \{c_i\}_{i=1}^n \in \mathbb{R}, \quad s \in C^r([a,b]) \right\} \tag{4.7}$$

where $\{N_i : 1 \le i \le n\}$ is the set of B-splines generated by the knot vector Ξ [59]. The restriction of s on a subinterval $[x_{i-1}, x_i]$ is just a polynomial of degree p:

$$s\Big|_{[x_{i-1},x_i]} \in \mathbb{P}^p(\mathbb{R})$$

If the interior knots occurs only one time, then $r = p - 1$ globally. The dimension is

$$\dim\left(\mathbb{S}^p_r(\Delta)\right) = n(p - r) + p + 1 \tag{4.8}$$

We also have the multiplicative relation

$$f_1 \in \mathbb{S}^{p_1}_{r_1}(\Delta), f_2 \in \mathbb{S}^{p_2}_{r_2}(\Delta) \implies f_1 f_2 \in \mathbb{S}^{p_1+p_2}_{\min(r_1,r_2)}(\Delta) \tag{4.9}$$

Knot insertion

Knot insertion [49] is adding an extra knot $\widehat{\xi}$ into a knot vector Ξ such that we get more basis functions and better shape control of the spline. This can be done without changing the geometric shape or subdividing the spline at the new inserted knot. The method is based on *Böhm's* theorem. We have a B-spline curve $\mathbf{C}(\xi) = \sum_{i=0}^n N_{i,p}\mathbf{P}_i$ defined by $\Xi = \{\xi_0, \dots, \xi_m\}$, and $\widehat{\xi} \in [t_s, t_{s+1})$. Then we can represent $\mathbf{C}(\xi) = \sum_{i=0}^n \widehat{N}_{i,p}\widehat{\mathbf{P}}_i$ by the new knot vector $\widehat{\Xi} = \{\xi_0, \dots, \xi_s, \widehat{\xi}, \xi_{s+1}, \dots, \xi_m\}$ as follows:

$$\widehat{\mathbf{P}}_i = \begin{cases} \mathbf{P}_i, & 0 \le i \le s - p \\ (1 - \alpha_i)\mathbf{P}_{i-1} + \alpha_i \mathbf{P}_i, & s - p + 1 \le i \le s \\ \mathbf{P}_{i-1}, & s + 1 \le i \le n + 1 \end{cases} \tag{4.10}$$

$$\alpha_i = \frac{\widehat{\xi} - \xi_i}{\xi_{i+p} - \xi_i} = \frac{\widehat{\xi} - \widehat{\xi}_i}{\widehat{\xi}_{i+p+1} - \widehat{\xi}_i} \tag{4.11}$$

This process can be generalized such that we can insert multiple knots simultaneously, and a well-known technique for this procedure is the *Oslo algorithm*

Degree elevation

Degree elevation is increasing a spline's polynomial order, making it more differentiable and compatible with the geometric shape [52]. This can be illustrated by a knot vector Ξ on$[a, b]$, where $a = \xi_0 < \xi_1 < \cdots < \xi_s < \xi_{s+1} = b$:

$$\Xi = \{\underbrace{a, \dots, a}_{p+1}, \underbrace{\xi_1, \dots, \xi_1}_{m_1}, \underbrace{\xi_2, \dots, \xi_2}_{m_2}, \dots, \underbrace{\xi_s, \dots, \xi_s}_{m_s}, \underbrace{b, \dots, b}_{p+1}\}$$

$\{m_k, 1 \le k \le s\}$ are the multiplicities of each knot. The initial degree is p, so the multiplicity at the endpoints becomes $p + 1$. To elevate the order to $p + 1$, we define

$$\widehat{\Xi} = \{\underbrace{a, \dots, a}_{p+2}, \underbrace{\xi_1, \dots, \xi_1}_{m_1+1}, \underbrace{\xi_2, \dots, \xi_2}_{m_2+1}, \dots, \underbrace{\xi_s, \dots, \xi_s}_{m_s+1}, \underbrace{b, \dots, b}_{p+2}\}$$

A similar process can be done for reducing the degree to $p-1$:

$$\widehat{\Xi} = \{\underbrace{a,\dots,a}_{p},\underbrace{\xi_1,\dots,\xi_1}_{m_1-1},\underbrace{\xi_2,\dots,\xi_2}_{m_2-1},\dots,\underbrace{\xi_s,\dots,\xi_s}_{m_s-1},\underbrace{b,\dots,b}_{p}\}$$

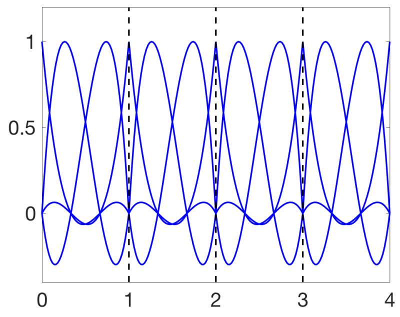
Numerical approximation properties

The advantage of defining B-splines implicitly by knot vectors is saving memory. Explicit symbolic manipulation requires too much memory and computational effort, reducing the algorithm efficiency. When we define the matrices and vectors in the discretization of a partial differential equation, it becomes appropriate to calculate integrals numerically with low error, so it is better to evaluate splines defined by knot vectors quickly at given points. The same can be done with the derivatives of any order, accelerating the algorithmic speed.

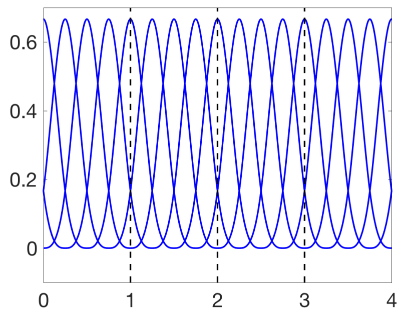
Isogeometric analysis provides *high accuracy*. The old Lagrange interpolation functions illustrate this. Their C^0-continuity is invariant of increasing the polynomial degree p. Doing so can generate high error oscillations in the numerical solution and causes bad approximation. But splines do not share this disadvantage. If the degree is p, then we can easily vary the continuity between 0 and $p-1$. High continuity reduces the error because the solution is more smooth. The computational complexity is also reduced significantly because the discretization matrices are sparse and have lower spectral radius. This may increase the speed of iterative algorithms [15, 23].

In contrast to the other previous finite element approaches, isogeometric analysis tackles discontinuous data much better. This is because splines have the *variation diminishing approximation property*, which smooths out discontinuities and prevents further spurious oscillations in the final solution. This is achieved when the spline continuity is high.

From figure 4.5, the Lagrange functions have C^{-1}-discontinuity at the points 0, 1, 2, 3 and 4, where they are piecewise continuous. This can cause spurious error propagation. But the B-splines have a uniform pattern without discontinuities, making them better.

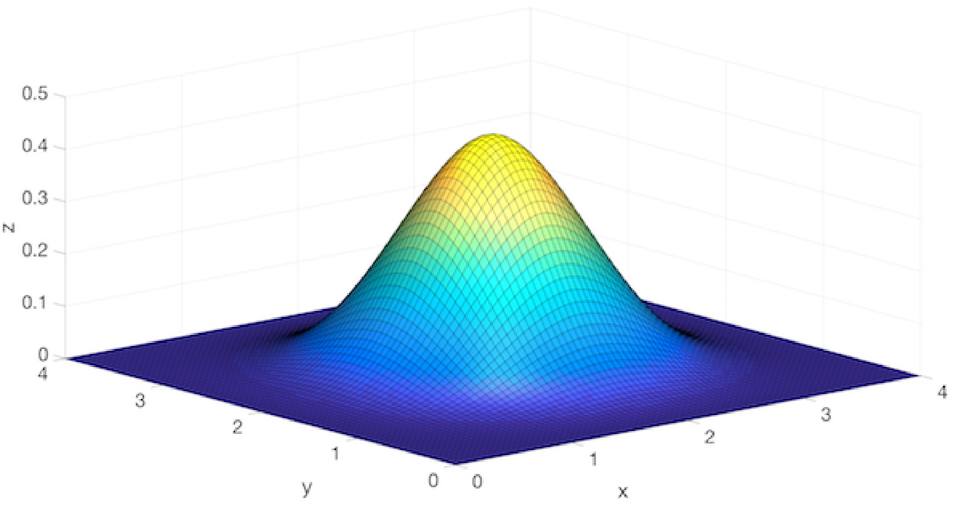


(a) *Cubic Lagrange interpolants*

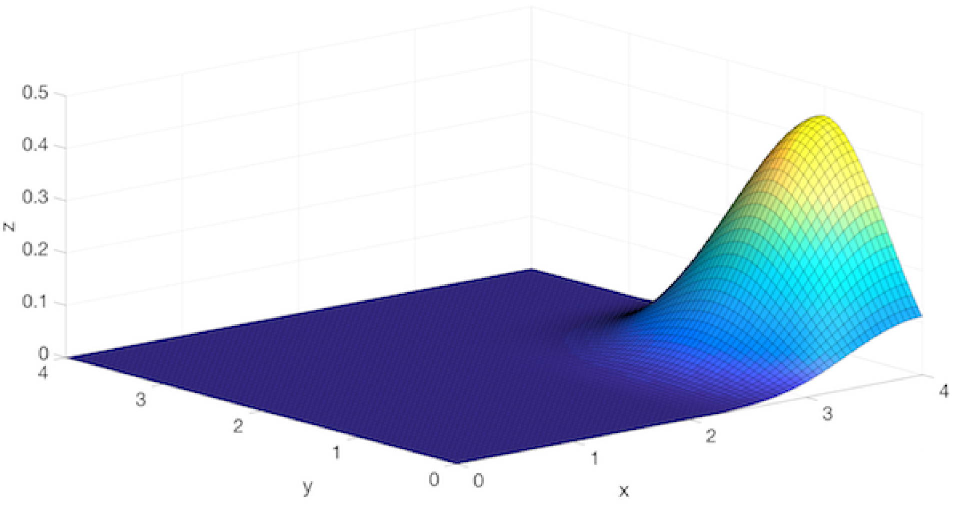


(b) *Cubic B-splines*

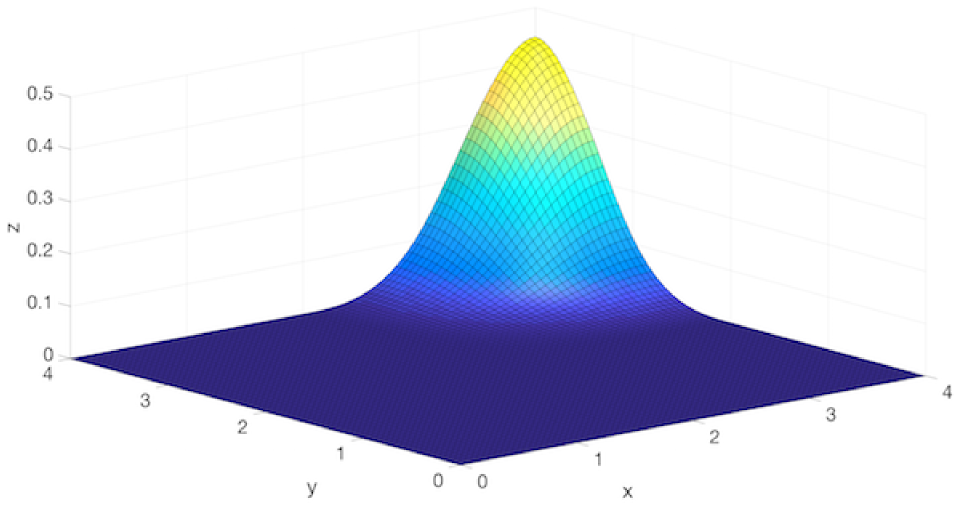
Figure 4.5. *Comparison of Lagrange interpolants and B-splines*



(a) *Bivariate basis function* $N_{3,3,3}$



(b) *Bivariate basis function* $N_{3,5,3}$



(c) *Bivariate basis function* $N_{5,5,3}$

Figure 4.6. *B-spline surfaces of order 3 on* $\Xi, \mathcal{H} = \{0,0,0,0,1,2,3,4,4,4,4\}$

4.2.2 Multivariate splines

Spaces of multivariate B-splines

In several dimensions, we need the partitions Δ_x, Δ_y and Δ_z for each spatial direction. The *bivariate* and *trivariate spline spaces* are defined as

$$\mathbb{S}^{p_x,p_y}_{r_x,r_y}(\Delta_x,\Delta_y) = \left\{ s_2(x,y) : \{c_{ij}\}^{n_x,n_y}_{i=1,j=1} \in \mathbb{R}, f \in I_2 \right\} \tag{4.12}$$

$$\mathbb{S}^{p_x,p_y,p_z}_{r_x,r_y,r_z}(\Delta_x,\Delta_y,\Delta_z) = \left\{ s_3(x,y,z) : \{c_{ijk}\}^{n_x,n_y,n_z}_{i=1,j=1,k=1} \in \mathbb{R}, f \in I_3 \right\} \tag{4.13}$$

where $I_2 = C^{r_x,r_y}([a_x,b_x,] \otimes [a_y,b_y])$ and $I_3 = C^{r_x,r_y,r_z}([a_x,b_x,] \otimes [a_y,b_y] \otimes [a_z,b_z])$. s_2 and s_3 are the *bivariate* and *trivariate tensor-product splines*, respectively.

$$s_2(\xi,\eta) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} c_{ij} N_{i,p}(\xi) M_{j,q}(\eta) \tag{4.14a}$$

$$s_3(\xi,\eta,\zeta) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} c_{ijk} N_{i,p}(\xi) M_{j,q}(\eta) L_{k,r}(\zeta) \tag{4.14b}$$

The restriction property for univariate splines holds in several dimensions. For multivariate B-splines, it is useful with the general decomposition relations [50, 59]:

$$\mathbb{P}^{p_{x_1},\dots,p_{x_d}}(\mathbb{R}^d) = \bigotimes_{i=1}^{d} \mathbb{P}^{p_{x_i}}(\mathbb{R}) \tag{4.15a}$$

$$\mathbb{S}^{p_{x_1},\dots,p_{x_d}}_{r_{x_1},\dots,r_{x_d}}(\Delta_{x_1},\dots,\Delta_{x_d}) = \bigotimes_{i=1}^{d} \mathbb{S}^{p_{x_i}}_{r_{x_i}}(\Delta_{x_i}) \tag{4.15b}$$

$$C^{r_{x_1},\dots,r_{x_d}}\left(\bigotimes_{i=1}^{d}[a_{x_i},b_{x_i}]\right) = \bigotimes_{i=1}^{d} C^{r_{x_i}}([a_{x_i},b_{x_i}]) \tag{4.15c}$$

$$\dim\left(\mathbb{S}^{p_{x_1},\dots,p_{x_d}}_{r_{x_1},\dots,r_{x_d}}(\Delta_{x_1},\dots,\Delta_{x_d})\right) = \prod_{i=1}^{d} \dim\left(\mathbb{S}^{p_{x_i}}_{r_{x_i}}(\Delta_{x_i})\right) \tag{4.15d}$$

Curves, surfaces and volumes

Tensor products are used to create B-spline curves, surfaces and volumes [50]. By using control vectors instead of scalar weights, they become more flexible, and we can easily manipulate the shape. The general tensor product formulas for curves ($\mathbf{C}$), surfaces ($\mathbf{S}$) and volumes ($\mathbf{V}$) are compactly defined as follows:

$$\begin{aligned}
\mathbf{C}(\xi) &= \sum_{i=1}^{n} N_{i,p}(\xi)\mathbf{P}_i && \mathbf{C} \in C^{k_1} \\
\mathbf{S}(\xi,\eta) &= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} N_{i,p}(\xi)M_{j,q}(\eta)\mathbf{P}_{ij} && \mathbf{S} \in C^{k_1} \otimes C^{k_2} \\
\mathbf{V}(\xi,\eta,\zeta) &= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} N_{i,p}(\xi)M_{j,q}(\eta)L_{k,r}(\zeta)\mathbf{P}_{ijk} && \mathbf{V} \in C^{k_1} \otimes C^{k_2} \otimes C^{k_3}
\end{aligned}$$

where $\{N_{i,p}\}_{i\in[1,n_1]}$, $\{M_{j,q}\}_{j\in[1,n_2]}$ and $\{L_{k,r}\}_{k\in[1,n_3]}$ are the sets of B-spline basis functions in ξ-, η- and ζ-directions, respectively, defined by knot vectors Ξ, $\mathcal{H}$ and $\mathcal{Z}$. $\mathbf{P}$ is the control point for the shapes, and the set of these points form a *control polygon* $\mathbf{CP}$. It is defined as a *control net* in 2D and *control lattice* in 3D.

$$\mathbf{CP} = \bigoplus_{i=1}^{n_x}\bigoplus_{j=1}^{n_y} \mathbf{P}_{ij} \qquad \mathbf{P}_{ij} \in \mathbb{R}^{2\times 1} \tag{4.16a}$$

$$\mathbf{CP} = \bigoplus_{i=1}^{n_x}\bigoplus_{j=1}^{n_y}\bigoplus_{k=1}^{n_z} \mathbf{P}_{ijk} \qquad \mathbf{P}_{ijk} \in \mathbb{R}^{3\times 1} \tag{4.16b}$$

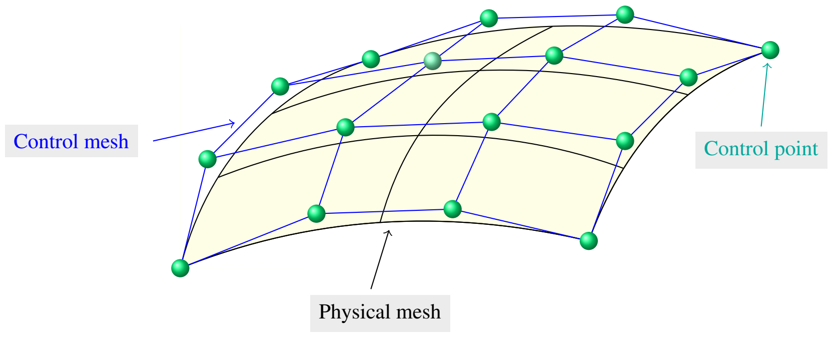
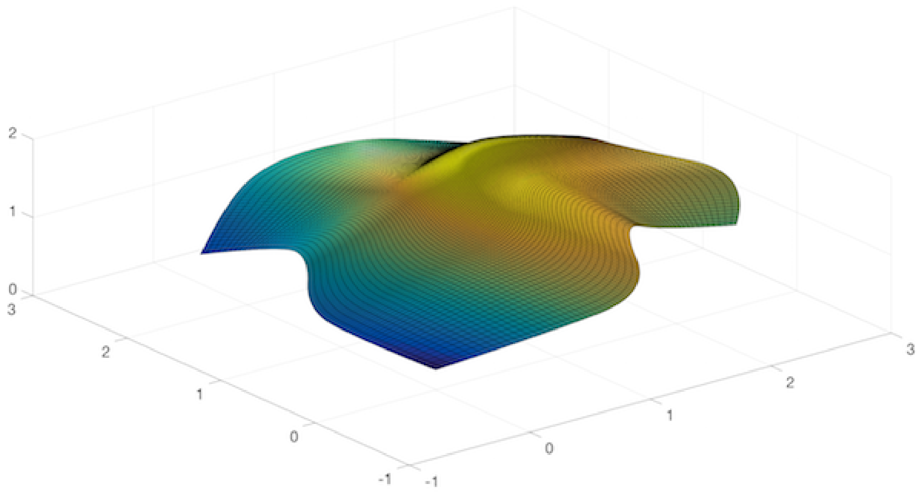
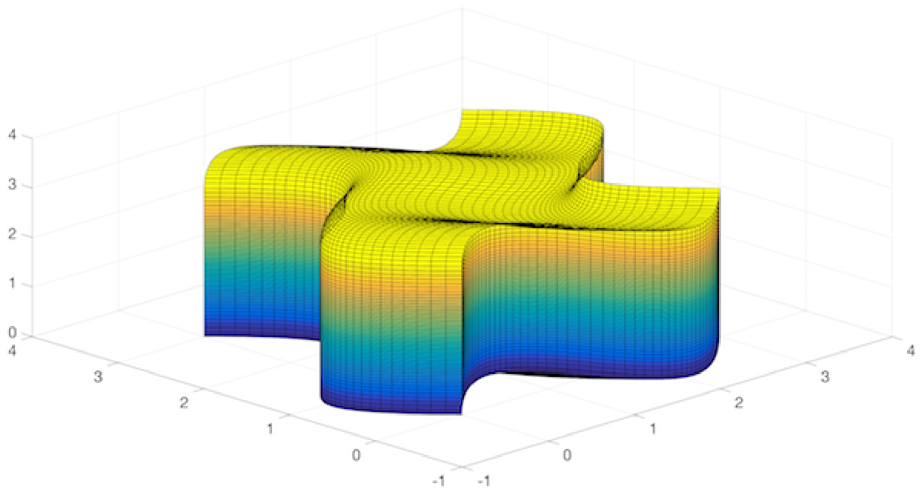


Figure 4.7. *Illustration of physical mesh and control mesh*

In isogeometric analysis, we distinguish between two different meshes because splines are used as interpolating functions. The *physical mesh* is the actual geometry of the domain decomposed into several local patches with their own knot spans making the discretization more flexible, while the *control mesh* is defined by the splines' control points used for adjusting the geometry. The control nets and lattices can be viewed as compositions of quadrilaterals and hexahedrons, respectively. This type of distinction does not exist in the other finite element approaches because their characteristic basis functions cannot change their shape in the same way as splines.



(a) *B-spline surface*



(b) *B-spline solid*

Figure 4.8. *B-spline surface and B-spline solid*

4.2.3 Changing spline basis through least-squares projection

If we have a spline function f, we can change its basis from $\mathcal{S}_1$ to $\mathcal{S}_2$ by least-squares approximation. That is, changing the coefficient vector by L^2-projections in matrix form:

$$f = \sum_{i \in \mathcal{S}_1} f_i \psi_i \approx \sum_{j \in \mathcal{S}_2} \widetilde{f}_j \phi_j \tag{4.17}$$

This is a linear least-squares problem where we reduce the approximation error in the L^2-norm [51]. The new basis functions are defined through the new knot vector. The mathematical statement of this minimization problem is

$$\min_{\widetilde{\mathbf{f}} \in \mathbb{R}^n} \int_\Omega \left| \sum_{i \in \mathcal{S}_1} f_i \psi_i - \sum_{j \in \mathcal{S}_2} \widetilde{f}_j \phi_j \right|^2 d\Omega \quad \equiv \quad \min_{\widetilde{\mathbf{f}} \in \mathbb{R}^n} \|\mathbf{\Psi}^T \mathbf{f} - \mathbf{\Phi}^T \widetilde{\mathbf{f}}\|_{L^2}^2 \tag{4.18}$$

Expanding the expression above by using the L^2 inner product, taking the gradient with respect to the target coefficient vector $\widetilde{\mathbf{f}}$, and defining the matrices $\mathbf{A} = \int_\Omega \mathbf{\Phi}\mathbf{\Phi}^T \, d\Omega$ and $\mathbf{B} = \int_\Omega \mathbf{\Phi}\mathbf{\Psi}^T \, d\Omega$, the linear equation system for $\widetilde{\mathbf{f}}$ becomes

$$\mathbf{A}\widetilde{\mathbf{f}} = \mathbf{B}\mathbf{f} \tag{4.19}$$

This procedure is valid both for B-splines and NURBS, since the target object is the new coefficient vector defining the spline function linearly. In the case of NURBS, the whole procedure becomes more complicated because the shape functions are rational, and we also need to adjust both weights and control points. The resulting system will be nonlinear and requires sophisticated least-squares algorithms for solving it.

4.3 NURBS

4.3.1 Univariate and multivariate NURBS

Definition and properties of NURBS

NURBS (*Non-Uniform Rational B-Splines*) are the heart of isogeometric analysis. They have the ability of representing conic sections exactly, and this enables us to mesh curved domains better. The generic form of this spline is

$$R(u) = \sum_{i=0}^{n} R_{i,p}(u)\mathbf{P}_i \tag{4.20}$$

NURBS are in many ways a flexible generalization of B-splines which can be used to define curves, surfaces and volumes in the same way as shown previously:

$$\mathbf{C}(\xi) = \frac{\sum_{i=1}^{n} N_{i,p}(\xi)w_i\mathbf{P}_i}{\sum_{i=1}^{n} N_{i,p}(\xi)w_i} \qquad \mathbf{C} \in C^{k_1}$$

$$\mathbf{S}(\xi,\eta) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(\xi)M_{j,q}(\eta)w_{ij}\mathbf{P}_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(\xi)M_{j,q}(\eta)w_{ij}} \qquad \mathbf{S} \in C^{k_1} \otimes C^{k_2}$$

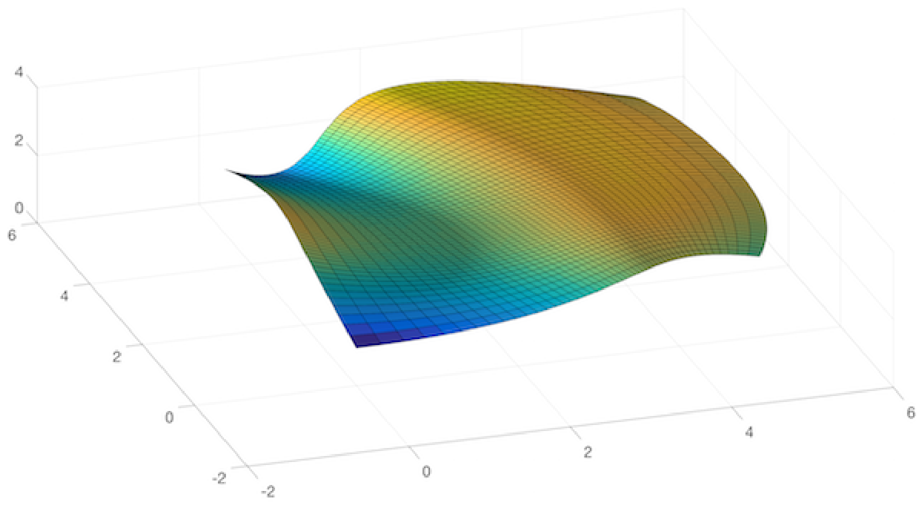
$$\mathbf{V}(\xi,\eta,\zeta) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} N_{i,p}(\xi)M_{j,q}(\eta)L_{k,r}(\zeta)w_{ijk}\mathbf{P}_{ijk}}{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} N_{i,p}(\xi)M_{j,q}(\eta)L_{k,r}(\zeta)w_{ijk}} \qquad \mathbf{V} \in C^{k_1} \otimes C^{k_2} \otimes C^{k_3}$$

w are the weights of the shape, and the denominator W is the *weighting function*. The NURBS have also many useful properties [49, 50, 52]:

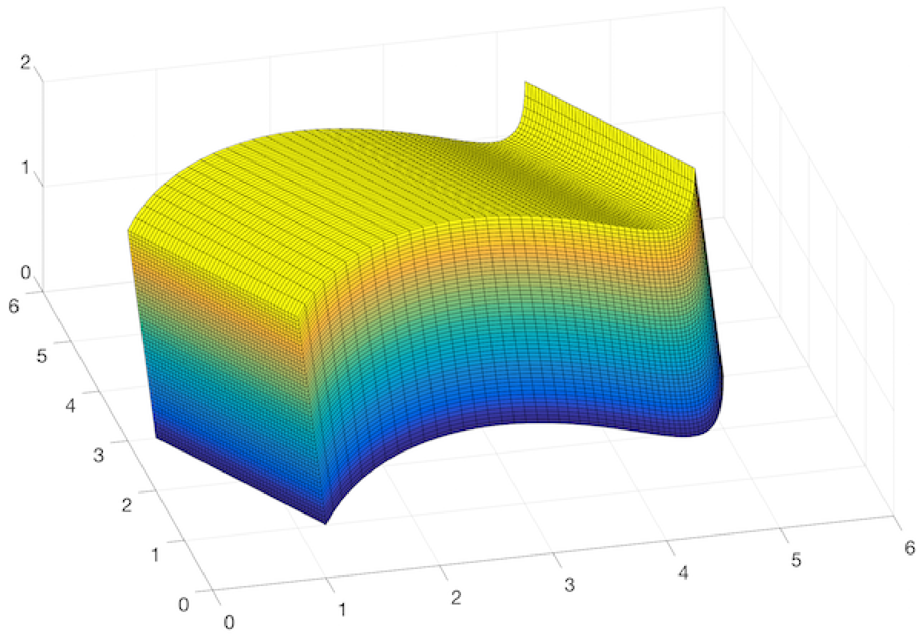
1. **Positivity**: $R_{i,p} \geq 0, \quad \xi \in [\xi_i, \xi_{i+p+1})$.
2. **Uniqueness**: $R_{i,p}$ only depend on the knots $\xi_i, \xi_{i+1}, \ldots \xi_{i+p+1}$.
3. **Positivity**: $R_{i,p} > 0, \quad \xi \in (\xi_i, \xi_{i+p+1})$.
4. **Local support**: supp$(R_{i,p}) = (\xi_i, \xi_{i+p+1})$.
5. **Unique maxima**: If $p > 0$, then $R_{i,p}$ has one unique maximum.
6. **Continuity**: $\bar{\xi} \in \Xi$ has multiplicity $m \implies R_{i,p} \in C^{p-m}$ at $\bar{\xi}$. Otherwise, $R_{i,p}$ is a smooth polynomial.
7. **Nonsingularity**: All derivatives of $R_{i,p}$ exists in the interior of the knot span.
8. **Partition of unity**: NURBS form a partition of unity:

$$\sum_{i=1}^{n} R_{i,p}(\xi) = 1 \qquad \forall \xi \in \Xi, p \in \mathbb{N}$$

9. **Invariance**: NURBS are completely invariant of operations like scaling, rotation, translation, shear and projection.



(a) *NURBS surface*



(b) *NURBS solid*

Figure 4.9. *NURBS surface and NURBS solid*

Evaluation and differentiation

Evaluation of a NURBS-curve $\mathbf{C}$ differs from B-splines because a d-dimensional NURBS curve is the projection of a $(d+1)$-dimensional B-spline curve [26]. Therefore, we must apply a projection $\mathcal{P} : \mathbb{R}^d \mapsto \mathbb{PR}^d$ on the control points as follows:

$$\{\mathbf{P}\}_{i\in[1,n]} = \{(x_i, y_i, z_i)\}_{i\in[1,n]} \longrightarrow \{\mathbf{Q}\}_{i\in[1,n]} = \{w_i(x_i, y_i, z_i, 1)\}_{i\in[1,n]} \tag{4.21}$$

The new curve defined by the original knot vector Ξ and the projected points $\{\mathbf{Q}\}_{i\in[1,n]}$ is a 4-dimensional and non-rational B-spline that can be evaluated directly. The final input is obtained by the inverse projection $\mathcal{P}^{-1} : \mathbb{PR}^d \mapsto \mathbb{R}^d$:

$$\mathbf{Q}_e = (x_0, y_0, z_0, d_0) \longrightarrow \mathbf{P}_e = \frac{1}{d_0}(x_0, y_0, z_0) \tag{4.22}$$

This method described above is also used for surfaces. After projecting every control point, we use the same matrix procedure for non-rational B-spline surfaces to evaluate at a point, and then we project inversely to find the value we seek for.

If $\mathbf{A}(\xi) = W(\xi)\mathbf{C}(\xi)$ [52], then the derivative of order a is defined recursively by

$$\mathbf{C}^{(a)} = \frac{1}{W}\left(\mathbf{A}^{(a)} - \sum_{i=1}^{a}\binom{a}{i}W^{(i)}\mathbf{C}^{(a-i)}\right) \tag{4.23}$$

A similar formula holds for surfaces too, where $\mathbf{A}(\xi,\eta) = W(\xi,\eta)\mathbf{S}(\xi,\eta)$:

$$\mathbf{S}^{(a,b)} = \frac{1}{W}\left(\mathbf{A}^{(a,b)} - \sum_{i=1}^{a}\binom{a}{i}W^{(i,0)}\mathbf{S}^{(a-i,b)} - \sum_{j=1}^{b}\binom{b}{j}W^{(0,j)}\mathbf{S}^{(a,b-j)}\right) \tag{4.24}$$

$$- \frac{1}{W}\sum_{i=1}^{a}\sum_{j=1}^{b}\binom{a}{i}\binom{b}{j}W^{(i,j)}\mathbf{S}^{(a-i,b-j)} \tag{4.25}$$

The real projective space $\mathbb{RP}^n$ ($\mathbb{RC}^n$) consists of all points $x \in \mathbb{R}^{n+1}$ such that x and αx define the same location when $\alpha \neq 0$. The properties of NURBS depend heavily on this n-dimensional manifold with quotient topology [26, 44].

Knot insertion and change of degree

Knot insertion and degree elevation for NURBS must also be done with projection. To insert s new knots in the knot vector Ξ, we project the control points to obtain a non-rational curve, as shown in (4.21). After the insertion, we use inverse projection and obtain a new NURBS, as follows:

$$\{\mathbf{Q}'\}_{i\in[1,n+s]} = \{(x'_i, y'_i, z'_i, w'_i)\}_{i\in[1,n+s]} \longrightarrow \{\mathbf{P}'\}_{i\in[1,n+s]} = \frac{1}{w'_i}\{(x'_i, y'_i, z'_i)\}_{i\in[1,n+s]}$$

4.3.2 Conic sections and quadric surfaces

As an introductory illustration of NURBS, we use the following parameters [49]:

$$\Xi = \left\{0, 0, 0, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{3}{4}, 1, 1, 1\right\}$$
$$w = \left\{1, \frac{1}{2}, \frac{1}{2}, 1, \frac{1}{2}, \frac{1}{2}, 1\right\}$$
$$\mathbf{P} = \left\{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}-1\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}-1\\-1\end{pmatrix}, \begin{pmatrix}1\\-1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\right\}$$

These parameters are used to construct a circle exactly by NURBS. If we scale the x- and y-coordinates of the control points, we get an ellipse, so the procedure is generic.

Surfaces of revolution can also be created by NURBS. We represent the cross section in a proper way and rotate it with respect to an axis. This is done by using the standard transformation matrices. For example, we can create a torus by generating a circle first and then rotate it with respect to the z-axis. A sphere can also be created in a similar way, but that requires a semicircle, and it can be parametrized as follows [52]:

$$(x(t), y(t)) = \left(\frac{1-2t}{1-2t+2t^2}, \frac{2t(1-t)}{1-2t+2t^2}\right)$$

It is important to notice that there is not a unique way for creating a NURBS surface. Indeed, there are many different approaches for representing the same surface exactly. The choice of representation is often depending on the specific situation. For further general details about tensor spaces and their properties, which are used for describing multivariate splines and their respective spaces, we refer to [29].

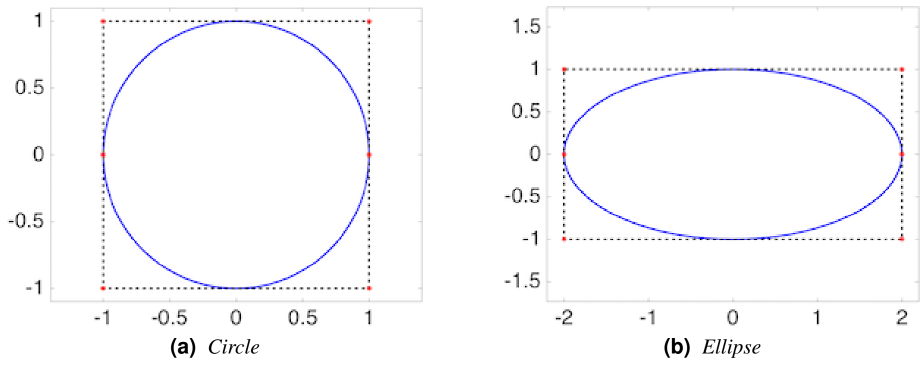
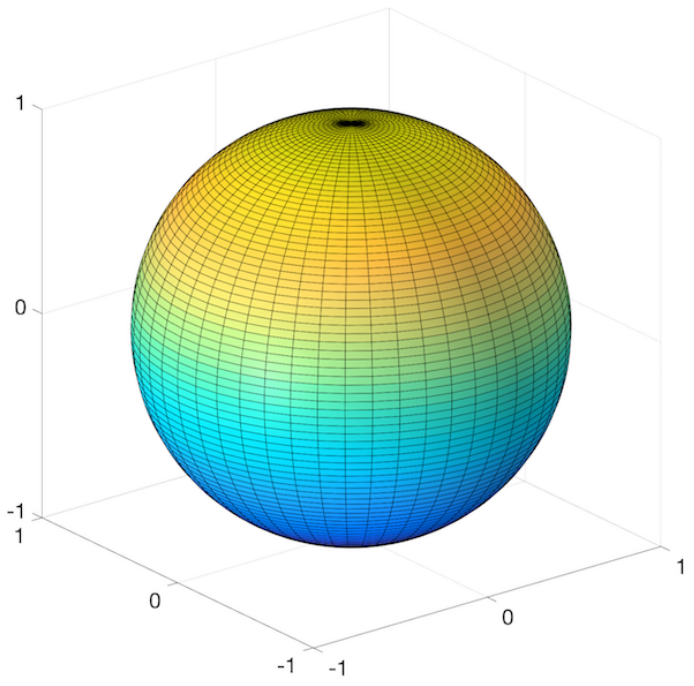
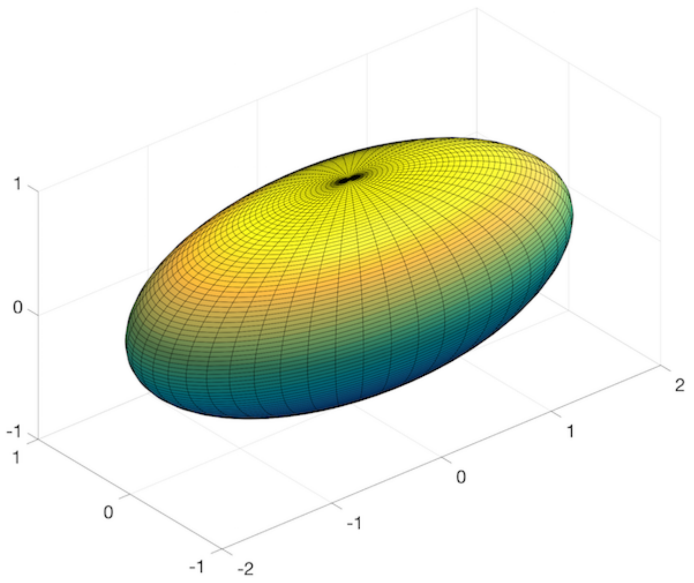


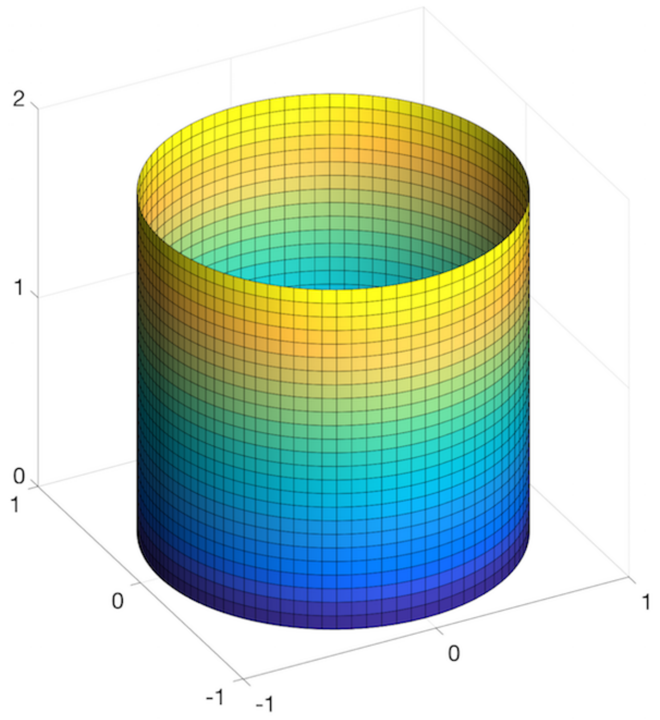
Figure 4.10. *Circle and ellipse constructed exactly by NURBS*



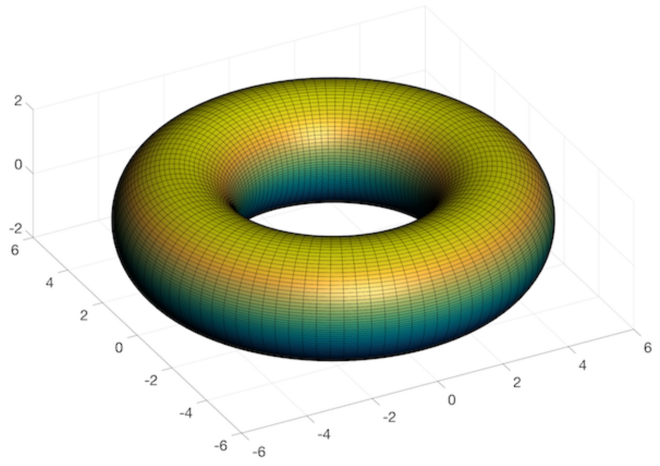
(a) *Sphere*



(b) *Ellipsoid*



(c) *Cylinder*



(d) *Torus*

Figure 4.11. *Torus, cylinder, sphere and ellipsoid constructed exactly by NURBS*

Chapter 5

Computational algorithms

We present the discretization of the Boussinesq equations and discuss the relevant methods for solving the whole system effectively. In order to reduce the computational complexity, we combine several methods and split the equation system in several parts. These parts are solved separately, and their solutions are stepwise coupled together such that they produce the final solution we seek for. The whole numerical procedure is a combination of several algorithms, equipped with efficient techniques from numerical linear algebra.

5.1 Main features of the discretization

General theory of Ritz-Galerkin discretization

The *Galerkin projection* discussed in chapter 2 converts a partial differential equation into a variational equation, and the next step is to make the numerical solution u_h belong to a finite-dimensional function space V_h, a proper subspace of the *trial space* V where u belongs to. At this point, we must invoke the element shape functions, in our case splines, and apply the *Ritz-Galerkin discretization* of the unknown solution:

$$u_h(\mathbf{x}) = \sum_{m \in \mathcal{S}} \psi_m(\mathbf{x}) u_m = \mathbf{\Psi}^T \mathbf{u} \tag{5.1}$$

where $\mathcal{S}$ is the linearly independent basis of shape functions. We define W as the *test space* of the function v used in the Galerkin projection. When the trial and test spaces coincide (the functions are of same type), we have the *Bubnov-Galerkin method*, which is most common. It works well for most partial differential equations. The residual error is forced to be orthogonal to every basis function, and this is optimal for equations with self-adjoint partial differential operators.

In some cases, the *Petrov-Galerkin method* is more appropriate for stabilizing advection. We decompose W in two parts such that $W = W_{\text{coarse}} \oplus W_{\text{fine}}$. This upwind technique is useful in situations where we must avoid spurious oscillation caused by boundary layers, a common phenomenon which occurs in equations with odd-order derivatives [12].

Both approaches described above are special cases of the *Method of Weighted Residual*, the core of finite element discretization. The concept is to minimize the residual error of the discretized solution over the entire domain [46]. This is analogous to minimizing the potential energy of a system, but the fundamental problem is that in many applications, its corresponding functional cannot be derived analytically. If $\mathcal{L}$ is the differential operator, and u_h is the discretized solution, then the two Galerkin methods can be formulated as

- Bubnov-Galerkin (GFEM) $\int \mathcal{L}(u_h) N_i \, d\Omega = 0$.
- Petrov-Galerkin (PGFEM) $\int \mathcal{L}(u_h) \left(N_i + \widetilde{F}_i \right) \, d\Omega = 0$.

It should be noted that implementation of Petrov-Galerkin discretization depends on the equation itself and the procedure for solving the discretized system arising from it.

Mixed finite element discretization

The most appropriate method for solving a system of partial differential equations is the *Mixed Finite Element Method* (MFEM) [5]. This is because the unknown solution consists of several components, and each of them have their own individual features that must be preserved as well as possible. This can be done by using different element shape functions and approximate the functions individually in their respective trial spaces. As a result, the algebraic system arising from the discretization will have a solution and can be solved without applying too complicated stabilization techniques.

The mixed discretization itself must always satisfy the *Ladyzhenskaya-Babuška-Brezzi* (LBB) condition, ensuring the existence of a stable and regular solution. For Stokes' equation, this can be formulated compactly as follows:

$$\inf_{\mathbf{U} \in H^1(\Omega)^2} \sup_{p \in L_0^2(\Omega)} \frac{1}{2} A(\mathbf{U}, \mathbf{U}) + F(\mathbf{U}) - B(\mathbf{U}, p) \tag{5.2}$$

This holds for the Navier-Stokes equation although it is not fully linear. If we discretize it such that the LBB condition is satisfied, the numerical solution exists. Traditionally, the discretization has been performed with the *Taylor-Hood*, *Nédélec* and *Raviart-Thomas* elements. It has been shown recently that these classical elements can be generalized for any polynomial degree and continuity such that the LBB condition (5.2) is satisfied and the mesh becomes conformal. This is very crucial in the isogeometric discretization. These three methods also form a hierarchy of the discretized velocity field and pressure: [9]

Velocity:	$\widehat{V}_h^{RT} \subset \widehat{V}_h^{N} \subset \widehat{V}_h^{TH}$
Pressure:	$\widehat{Q}_h^{RT} \equiv \widehat{Q}_h^{N} \equiv \widehat{Q}_h^{TH}$

To simplify the code implementation, we choose the isogeometric Taylor-Hood method, where $\deg(\mathbf{U}_h) = \deg(p_h) + 1$ due to the fact that $\mathbf{U} \in H^1(\Omega)^3$ and $p \in L^2_0(\Omega)$. If we denote $\mathcal{M} = (\Delta_x, \Delta_y, \Delta_z)$ as the discretization of Ω, then

$$\widehat{V}_h^{TH}: \quad \mathbf{U}_h \in \bigtimes_{i=1}^{3} \mathbb{S}^{p+1,p+1,p+1}_{\alpha,\alpha,\alpha}(\mathcal{M}) \qquad \widehat{Q}_h^{TH}: \quad p_h \in \mathbb{S}^{p,p,p}_{\alpha,\alpha,\alpha}(\mathcal{M})$$

where $0 \le \alpha \le p-1$. In order to obtain high accuracy, the continuity should be maximal, i.e. $\alpha = p-1$ both for the velocity field and the pressure. The velocity field's continuity is reduced in order to fulfill the LBB-condition, see [7]. However, for the temperature, it can be chosen with maximum continuity, i.e. $T_h \in \mathbb{S}^{p+1,p+1,p+1}_{p,p,p}(\mathcal{M})$. Using the same notation as in [56], we define the numerical solutions as

$$u_{d,h}(x,y,z,t) = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\sum_{k=1}^{n_z} \psi_i(x)\psi_j(y)\psi_k(z)u_{d,ijk}(t) = \mathbf{\Psi}^T\mathbf{U}_d \tag{5.3a}$$

$$p_h(x,y,z,t) = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\sum_{k=1}^{n_z} \phi_i(x)\phi_j(y)\phi_k(z)p_{ijk}(t) = \mathbf{\Phi}^T\mathbf{P} \tag{5.3b}$$

$$T_h(x,y,z,t) = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\sum_{k=1}^{n_z} \theta_i(x)\theta_j(y)\theta_k(z)T_{ijk}(t) = \mathbf{\Theta}^T\mathbf{T} \tag{5.3c}$$

where $d \in \{x,y,z\}$. The system of equations arising from the discretization is given by

$$\mathbf{M}_U\dot{\mathbf{U}}_x + \mathbf{C}_U(\mathbf{U})\mathbf{U}_x + a_1\mathbf{K}_U\mathbf{U}_x - \mathbf{D}_x^T\mathbf{P} = \mathbf{0} \tag{5.4a}$$

$$\mathbf{M}_U\dot{\mathbf{U}}_y + \mathbf{C}_U(\mathbf{U})\mathbf{U}_y + a_1\mathbf{K}_U\mathbf{U}_y - \mathbf{D}_y^T\mathbf{P} = \mathbf{0} \tag{5.4b}$$

$$\mathbf{M}_U\dot{\mathbf{U}}_z + \mathbf{C}_U(\mathbf{U})\mathbf{U}_z + a_1\mathbf{K}_U\mathbf{U}_z - \mathbf{D}_z^T\mathbf{P} = -a_2\mathbf{R}_{UT}\mathbf{T} \tag{5.4c}$$

$$\mathbf{D}_x\mathbf{U}_x + \mathbf{D}_y\mathbf{U}_y + \mathbf{D}_z\mathbf{U}_z = 0 \tag{5.4d}$$

$$\mathbf{M}_T\dot{\mathbf{T}} + \mathbf{C}_T(\mathbf{U})\mathbf{T} + a_3\mathbf{K}_T\mathbf{T} = \mathbf{Q} \tag{5.4e}$$

where $a_1 = Re^{-1}$, $a_2 = Ri$ and $a_3 = Pe^{-1}$. The matrices are defined as follows:

$$\begin{aligned}
&\mathbf{M}_U = \int_\Omega \mathbf{\Psi}\mathbf{\Psi}^T\,d\mathbf{x} \quad , \quad \mathbf{K}_U = \int_\Omega (\nabla\mathbf{\Psi})\cdot(\nabla\mathbf{\Psi})^T\,d\mathbf{x} \quad , \quad \mathbf{D}_d = \int_\Omega \mathbf{\Phi}\frac{\partial\mathbf{\Psi}^T}{\partial x_d}\,d\mathbf{x} \\
&\mathbf{M}_T = \int_\Omega \mathbf{\Theta}\mathbf{\Theta}^T\,d\mathbf{x} \quad , \quad \mathbf{K}_T = \int_\Omega (\nabla\mathbf{\Theta})\cdot(\nabla\mathbf{\Theta})^T\,d\mathbf{x} \quad , \quad \mathbf{R}_{UT} = \int_\Omega \mathbf{\Psi}\mathbf{\Theta}^T\,d\mathbf{x} \\
&\mathbf{C}_T(\mathbf{U}) = \int_\Omega \mathbf{\Theta}\mathbf{\Psi}^T\left(\mathbf{U}_x\frac{\partial\mathbf{\Theta}^T}{\partial x} + \mathbf{U}_y\frac{\partial\mathbf{\Theta}^T}{\partial y} + \mathbf{U}_z\frac{\partial\mathbf{\Theta}^T}{\partial z}\right)d\mathbf{x} \quad , \quad \mathbf{Q} = \int_\Omega \mathbf{\Theta}Q\,d\mathbf{x} \\
&\mathbf{C}_U(\mathbf{U}) = \int_\Omega \mathbf{\Psi}\mathbf{\Psi}^T\left(\mathbf{U}_x\frac{\partial\mathbf{\Psi}^T}{\partial x} + \mathbf{U}_y\frac{\partial\mathbf{\Psi}^T}{\partial y} + \mathbf{U}_z\frac{\partial\mathbf{\Psi}^T}{\partial z}\right)d\mathbf{x}
\end{aligned} \tag{5.5}$$

We see that the convection term $\mathbf{u}\cdot\nabla$ can be discretized in two ways, depending on the solution component it is applied to.

Remarks on the semi-discretization

A more compact way of rewriting the semi-discretized Navier-Stokes equations is

$$\begin{aligned} \mathbf{U}_t &= \mathbf{M}^{-1}\left(-\mathbf{C}(\mathbf{U})\mathbf{U} - \mathbf{K}\mathbf{U} + \mathbf{D}^T\mathbf{P}\right) \equiv \mathcal{F}(\mathbf{U},\mathbf{P}) \\ \mathbf{0} &= \mathbf{D}\mathbf{U} \equiv \mathcal{G}(\mathbf{U}) \end{aligned}$$

From this simplified notation, we can deduce an important relation:

$$\begin{aligned} \frac{\partial\mathcal{G}}{\partial\mathbf{U}}\frac{\partial\mathcal{F}}{\partial\mathbf{P}} &= \mathbf{D}\mathbf{M}^{-1}\mathbf{D}^T \\ &= \mathbf{D}_x\mathbf{M}_U^{-1}\mathbf{D}_x^T + \mathbf{D}_y\mathbf{M}_U^{-1}\mathbf{D}_y^T + \mathbf{D}_z\mathbf{M}_U^{-1}\mathbf{D}_z^T \end{aligned}$$

The pressure is stabilized through the LBB-condition (5.2), so $\mathbf{D}$ has full rank, and $\mathbf{M}$ is always invertible. Hence, the *Schur complement* $\mathbf{D}\mathbf{M}^{-1}\mathbf{D}^T$ is nonsingular, and the Navier-Stokes equations can be classified as a *differential algebraic equation of index 2*, commonly abbreviated as *index-2 DAE*. This means that we have a differential equation (system) subject to an algebraic equation, and the solvability depends on the product of two derivatives, one for each equation in the system [33].

The convection-diffusion equation is not subject to any constraint, so it becomes just a linear dynamical system of differential equations when we semi-discretize it.

5.2 A hybrid multistage algorithm

The procedure of choosing an efficient time-integrator for any dynamical system is always depending on the equation itself and the conditions we want to examine. When we solve complex systems like the Boussinesq equations, it is very appropriate to segregate it. This means that for each time step, we find a solution of the Navier-Stokes and convection-diffusion equations separately. Then we repeat the same process repeatedly. Doing so is more effective than solving both equations simultaneously for every time-step because smaller equations are easier to handle quickly. We can even use different time-integrators for each equation, and this yields higher flexibility of the complete numerical procedure.

5.2.1 Time-integration of the Navier-Stokes equations

Choice of implicit time-integrator

The first approach tested for solving the Navier-Stokes equations in this thesis was the *fractional-step method*. The goal was to separate the linear and nonlinear parts of the equation system in such a way that we could avoid Newton-iteration. Unfortunately, this method did not give sufficiently accurate solutions although it was correctly implemented.

Hence, the conclusion was that the saddle-point structure of the Navier-Stokes equations, discretized with isogeometric analysis, caused instabilities. If we were solving a single partial differential equation or an equation system which was not classified as a saddle-point system, then the splitting method would probably have worked correctly and given the desired results. Hence, the conclusion was to discard the fractional-step method and use a fully implicit time-integrator using Newton-iteration in each time step.

The natural choice was therefore to use the *Backward Differentiation Formula* (BDF), one of the most common integrators used for solving large stiff systems arising from partial differential equations. Its main characteristics are A-stability and low usage of memory. We do not need to store the nonlinear convection from the previous time steps, only the one for the next step, which will be used in the Newton iteration. As long as the BDF-integrator does not have more than 2 steps, it will indeed be A-stable as a consequence of the universal *Dahlquist barrier* [32, 33].

Implementation of Newton-iteration

By using compact tensor notation and dropping the scaling parameters, we can assume for simplicity that the Navier-Stokes equations with force is given by

$$\mathbf{M}\mathbf{U}_t + \mathbf{C}(\mathbf{U})\mathbf{U} + \mathbf{K}\mathbf{U} - \mathbf{D}^T\mathbf{P} = \mathbf{F} \tag{5.6a}$$

$$\mathbf{D}\mathbf{U} = \mathbf{0} \tag{5.6b}$$

It is important to notice that the pressure is used for adjusting the velocity field such that it remains solenoidal for each time step, so we do not need to discretize the pressure with respect to time. The generic form of any BDF-integrator, applied on the model equation $y = f(y,t)$, can be formulated as follows:

$$\gamma_0 y^{n+1} + \sum_{i=1}^{k} \alpha_i y^{n+1-i} = h f^{n+1}$$

Applying this implicit integrator on the Navier-Stokes equations, which has been earlier classified as a differential algebraic equation of index 2, we obtain

$$\left(\frac{\gamma_0}{h}\mathbf{M} + \mathbf{K} + \mathbf{C}(\mathbf{U}^{n+1})\right)\mathbf{U}^{n+1} - \mathbf{D}^T\mathbf{P}^{n+1} = \mathbf{F}^{n+1} - \frac{1}{h}\mathbf{M}\left[\sum_{i=1}^{k}\alpha_i \mathbf{U}^{n+1-i}\right]$$

$$\mathbf{D}\mathbf{U}^{n+1} = \mathbf{0}$$

This can be more compactly stated as the nonlinear equation system

$$\begin{bmatrix} \frac{\gamma_0}{h}\mathbf{M} + \mathbf{K} & -\mathbf{D}^T \\ -\mathbf{D} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \mathbf{P} \end{bmatrix}^{n+1} + \begin{bmatrix} \mathbf{C}(\mathbf{U}^{n+1})\mathbf{U}^{n+1} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{G}^{n+1} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{5.7}$$

For each time step, this nonlinear system must be solved by Newton-iteration. Thus, the Jacobian of the left-hand side of (5.7) is defined as

$$\mathbf{J} = \begin{bmatrix} \frac{\gamma_0}{h}\mathbf{M} + \mathbf{K} & -\mathbf{D}^T \\ -\mathbf{D} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{R}(\mathbf{U}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{5.8}$$

As we see, the first matrix is a constant saddle-point matrix which never changes it value during the whole time-integration, but the second one is nonlinear and has to be updated for each iteration. In order to define the nonlinear part of the Jacobian, we need some special differentiation rules. Let us assume that $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are vectors with n entries, and I_n is the identity matrix. Then we can state the following relations:

$$\frac{\partial}{\partial \mathbf{a}} \left(\mathbf{c}\mathbf{b}^T\mathbf{a}\right) = \mathbf{c}\mathbf{b}^T \tag{5.9a}$$

$$\frac{\partial}{\partial \mathbf{a}} \left(\mathbf{a}\mathbf{b}^T\mathbf{c}\right) = \left(\mathbf{b}^T\mathbf{c}\right) I_n \tag{5.9b}$$

$$\frac{\partial}{\partial \mathbf{a}} \left(\mathbf{a}\mathbf{b}^T\mathbf{a}\right) = \mathbf{a}\mathbf{b}^T + \left(\mathbf{b}^T\mathbf{a}\right) I_n \tag{5.9c}$$

Applying this on the nonlinear vector $\mathbf{C}(\mathbf{U})\mathbf{U}$, we obtain the components of $\mathbf{R}$:

$$\mathbf{R}_{ij}(\mathbf{U}) = \delta_{ij}\mathbf{C}(\mathbf{U}) + \int_\Omega \left(\frac{\partial \mathbf{\Psi}^T}{\partial x_j}\mathbf{U}_{x_i}\right) \mathbf{\Psi}\mathbf{\Psi}^T \, d\mathbf{x} \tag{5.10}$$

where δ_{ij} is Kronecker's delta and $1 \leq i, j \leq 3$. The convergence of Newton's method is quadratic, so we do not need many iterations before reaching the desired tolerance level.

Algorithm 1 Newton's method for systems of equations

```
1: procedure NEWTON_ITERATION($\mathbf{H}$, $\mathbf{J}$)
2:     Define the tolerance level $\lambda$
3:     Define the maximal iteration number $N$
4:     Define the initial guess $\mathbf{Z}_0$ as a zero vector
5:     Initialize the number of iterations, $k = 1$
6:     Initialize the error of the iteration, $err = \inf$
7:     while $k \leq N$ and $err \leq \lambda$ do
8:         Update the equation's left-hand side, $\mathbf{H}(\mathbf{Z}^{k-1})$
9:         Update the Jacobian, $\mathbf{J}(\mathbf{Z}^{k-1})$
10:        Calculate the increment $\mathbf{Y}^{k-1} = \mathbf{J}(\mathbf{Z}^{k-1})^{-1}\mathbf{H}(\mathbf{Z}^{k-1})$
11:        Find the new iterate $\mathbf{Z}^k = \mathbf{Z}^{k-1} - \mathbf{Y}^{k-1}$
12:        Calculate the new error, $err = \|\mathbf{Y}^{k-1}\|_\infty$
13:        Increment the number of iterates, $k = k + 1$
14:     return $\mathbf{Z}$
```

Solvability by multipoint constraining

Since the pressure belongs to $L_0^2(\Omega)$, it must satisfy the compatibility condition

$$\int_\Omega p(x,y,z,t)\,d\mathbf{x} = 0$$

In terms of the element shape functions, the left-hand side can be expressed as

$$\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\sum_{k=1}^{n_z}\int_\Omega \phi_i(x)\phi_j(y)\phi_k(z)p_{ijk}(t)\,d\mathbf{x} = \sum_{m\in\mathcal{S}}\left(\int_\Omega \phi_m(x,y,z)\right)p_m(t)\,d\mathbf{x}$$

If there are N_p basis functions for the pressure, then we can express the first coefficient as

$$p_1(t) = -\frac{1}{\alpha_1}\sum_{J=2}^{N_p}\alpha_J p_J(t)$$

We apply the multipoint constraint above on equation system in (5.6). If $\mathbf{D}^T = [d_{IJ}^T]$ and g_I^{n+1} comes from the composite right-hand side vector, then

$$\sum_{J=2}^{N_p}\left(d_{IJ}^T - \frac{\alpha_J}{\alpha_1}d_{I,1}^T\right)p_J^{n+1} = g_I^{n+1} \qquad , \qquad 2 \le I \le N_u \tag{5.11}$$

5.2.2 Solving the linear convection-diffusion equation

Choice of implicit-explicit time-integrator

The convection-diffusion equation is parabolic, so explicit time-integrators do not work. If the advection field was given by a known function, a fully implicit integrator would work very well. But in the Boussinesq system, the advection is given by the unknown velocity field. If we use an implicit integrator, we need information about $\mathbf{U}^{n+1}$ when we are at step t_n, but this quantity belonging to the next step t_{n+1} is unknown. Similarly for the Navier-Stokes equations, the use of the BDF integrator requires the value of $\mathbf{F}^{n+1}$ for each time step t_n, and this is straightforward because the force on the right-hand side of the equation is known. In the Boussinesq system, anyway, we also require the value of $\mathbf{T}^{n+1}$ as shown in (5.4), and this quantity is also unknown.

Fortunately, this bottleneck preventing the Boussinesq system from being segregated can be efficiently removed if we also use an IMEX integrator on the convection-diffusion equation. This means that the convection and diffusion terms are integrated explicitly and implicitly, respectively. By doing so, we only require $\mathbf{U}^n$ when we are at step t_n. Since the equation is linear, A-stability of the integrator's implicit part is sufficient. Thus, the Navier-Stokes and convection-diffusion equations are fully separated from each other. The computational complexity required for solving the Boussinesq equations is reduced quite significantly.

The first attempt was using IMEX Runge-Kutta methods because they have large stability regions. This approach did not give sufficiently accurate results during the simulation. A probable explanation of this failure might be that we need to evaluate the convection matrix several times in each time step, but we only did it a few times because we are using the BDF integrator on the Navier-Stokes equation. Hence, we had to use IMEX multistep methods instead, and the natural choice was therefore the *Semi-implicit Backward Differentiation Formula* (SBDF), since it shares the same advantages of BDF like low storage and less computational effort. By using BDF and SBDF simultaneously on the Boussinesq system, the segregation was achieved, and it did indeed give accurate results. This will be shown in chapter 7, where the MATLAB-codes are verified numerically. Further details on IMEX integrators and their construction can be found in [4, 3].

Coupling the equations together

We assume that the convection-diffusion equations is defined as

$$\mathbf{M}_T\dot{\mathbf{T}} + \mathbf{C}_T(\mathbf{U})\mathbf{T} + \mathbf{K}_T\mathbf{T} = \mathbf{Q} \tag{5.12}$$

By applying the SBDF integrator, we just need to solve the linear equation

$$\left(\frac{\gamma_0}{h}\mathbf{M}_T + \mathbf{K}_T\right)\mathbf{U}^{n+1} = \mathbf{Q}^{n+1} - \frac{1}{h}\sum_{i=1}^{k}\left[\alpha_i\mathbf{M}_T\mathbf{U}^{n+1-i} + \beta_i(\mathbf{C}_T(\mathbf{U})\mathbf{T})^{n-i}\right] \tag{5.13}$$

This equation has the same structure as the Helmholtz equation. When the temperature $\mathbf{T}^{n+1}$ is found, we multiply it with the matrix $\mathbf{R}_{UT}$ from (5.4). This can actually be regarded as a least-squares projection $\mathbb{S}_p^{p+1} \to \mathbb{S}_{p-1}^{p+1}$, ensuring that the temperature is transferred to the right-hand side of the Navier-Stokes equations.

Special facilities for reducing computational effort

In the whole finite element projection, the matrices are not inverted more than necessary, and the inversions are performed such that matrix sparsities are preserved. It is also a good idea to define the convection as vectors, $\mathbf{C}_T(\mathbf{U})\mathbf{T}$ and $\mathbf{C}_U(\mathbf{U})\mathbf{U}$. The assembly will take less time, and we do not need to multiply matrices with vectors afterwards. Thus, the global running time of the algorithm is minimized in the first stage.

The next stage is solving the equation systems effectively. The symmetry of equation (5.13) makes it appropriate to use *algebraic multigrid*. It is completely independent of the discretization method and is an optimal multiresolution algorithm for solving linear partial differential equations. The saddle-point system in (5.7) might be decomposed with special algorithms resembling the classical *Uzawa decomposition*. Because of its nonlinear and nonsymmetric structure, the process can be a little bit complicated.

5.3 Stabilizing boundary layers

In fluid mechanics, the solution of partial differential equations might contain boundary layers. The reason is that the highest order derivative depends on a very small parameter. When it tends to zero, we lose one of the specified boundary conditions. This yields the *outer solution* of the equation, in the context of singular perturbation. When we apply the Bubnov-Galerkin method to a convection-dominated problem, the odd-order derivatives make the problem nonsymmetric, and spurious oscillations ruin the approximation. But the Petrov-Galerkin method adds a small perturbation function to the test function in the weak formulation, resulting in high stabilization of the numerical solution. There is a big hierarchy of similar algorithms, and a complete description is given in [57].

5.3.1 The Streamline-Upwind/Petrov-Galerkin method

The SUPG (*Streamline-Upwind/Petrov-Galerkin*) method is a classical stabilization in the multiscale finite element method (MsFEM). It was originally designed by Zienkiewicz for the advection-diffusion equation [62], and then generalized by Brooks and Hughes for the Navier-Stokes equations [8]. This method is compatible with isogeometric analysis, mostly with respect to the ease of implementation.

In the finite difference method, *upwind discretization* precludes the spurious oscillations caused by boundary layers. Instability will always imply inaccuracy, but stability does not imply high accuracy automatically. Although the upwind method is stable, consistent and removes the oscillatory modes, the accuracy is just $\mathcal{O}(h)$. Transferring this directly to multidimensional and unsteady problems, as in the *artificial diffusion method*, causes a potential risk of adding too much artificial diffusion, which is $\mathcal{O}(h)$ in all directions. Unexpected crosswind diffusion perpendicular to the advection field can also occur. The stabilizing term is not compatible with the polynomial approximation's optimal order no matter the polynomial degree of the finite element interpolant [54]. The error estimate of this upwind discretization is just

$$\|u - u_h^*\| \leq C \left(\inf_{v_h \in V} \|u - v_h\|_{H^1} + h\|u\|_{H^1} \right) \tag{5.14}$$

The advantage of SUPG is adding some artificial diffusion in the streamline direction, an appropriate limitation. Hughes et al. showed in [36] that if we combine SUPG together with isogeometric analysis, the stabilization is strengthened because of high continuity in the spline interpolants. Thus, we combine two stabilizing factors together such that any spurious oscillation is easily removed away. Asymmetric weight functions ensure that nodal upstream is weighted more heavily than the nodal downstream, for each element. The mesh does not need to be too dense in order to avoid spurious oscillation, and the solution will be efficient to find, but mesh-dependency must be taken into account in order to determine the stabilizing parameters.

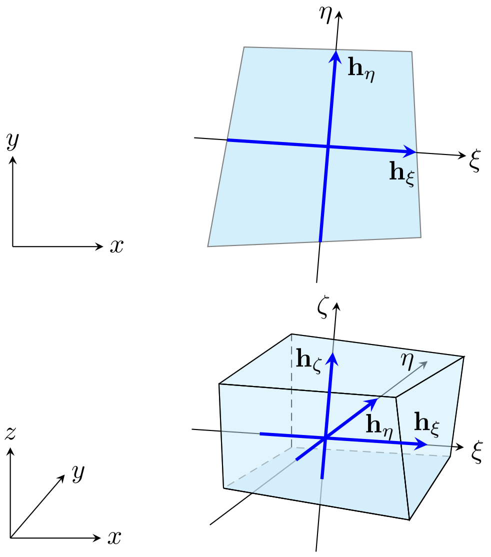


Figure 5.1. *Quadrilateral and hexahedral elements with characteristic element length*

The *characteristic element length* is a frequent quantity used in the SUPG-formulation of partial differential equations with the same structure as the advection-diffusion equation. It simplifies much of the calculations because it is easy to compute. For any finite element, the characteristic element length is defined as

$$h^e = \begin{cases} |h_1| + |h_2| & \text{2D} \\ |h_1| + |h_2| + |h_3| & \text{3D} \end{cases} \tag{5.15}$$

We define $\mathbf{u}^e$ as the element convection and obtain

$$h_1 = \frac{1}{\|\mathbf{u}^e\|}(\mathbf{u}^e \cdot \mathbf{h}_\xi) \qquad h_2 = \frac{1}{\|\mathbf{u}^e\|}(\mathbf{u}^e \cdot \mathbf{h}_\eta) \qquad h_3 = \frac{1}{\|\mathbf{u}^e\|}(\mathbf{u}^e \cdot \mathbf{h}_\zeta) \tag{5.16}$$

If the convection is given by a known function, or the mesh is a rectangular grid, then the formulas above can be very simplified [17].

5.3.2 Outline of the different formulations

SUPG on convection-diffusion equation

We assume first that the convection-diffusion equation is on the general form

$$\frac{\partial T}{\partial t} - k\nabla^2 T + (\mathbf{u} \cdot \nabla)T = Q$$

We use the same approach described in [17]. If the current element is denoted by e, we can define the element-wise collection of perturbed test functions as

$$W_I^e(\mathbf{x}) = N_I^e(\mathbf{x}) + \frac{h^e}{2\|\mathbf{u}^e\|}\left(\alpha + \frac{\beta \Delta t}{2}\frac{\partial}{\partial t}\right)(\mathbf{u}^e \cdot \nabla N_I^e(\mathbf{x}))$$

$\alpha \in [0,1]$ is a free upwind parameter for artificial diffusion, and β is the dependent upwind parameter for temporal discretization. They are defined as

$$\alpha = \coth(Pe) - \frac{1}{Pe} \qquad \beta = \frac{Cr}{3} - \frac{\alpha}{PeCr}$$
$$Pe = \frac{\|\mathbf{u}^e\| h^e}{2k} \qquad Cr = \frac{\|\mathbf{u}^e\|\Delta t}{h^e}$$

Pe and Cr are the local *Péclet* and *Courant numbers*, respectively. h^e is the characteristic element length. The adjustable tuning parameters α and β are optimally constructed for any dimension, and the local definition on each element makes the method more flexible. The velocity field is not a constant vector field as in the case of linear advection, and it is given as a sum of interpolating shape functions. An easy and flexible procedure is computing the L^2-norm on the current element:

$$\|\mathbf{u}\|^2_{L^2(\Omega^e)} = \|u_x\|^2_{L^2(\Omega^e)} + \|u_y\|^2_{L^2(\Omega^e)} + \|u_z\|^2_{L^2(\Omega^e)} \tag{5.17}$$

Applying this on the convection-diffusion equation yields

$$\widetilde{\mathbf{A}}_T\dot{\mathbf{T}} + \widetilde{\mathbf{B}}_T\mathbf{T} = \widetilde{\mathbf{Q}}$$

The SUPG-matrices for the convection-diffusion equation are defined by

$$\widetilde{\mathbf{A}}_T = \int_\Omega (\mathbf{\Theta} + (\mathbf{e}\cdot\nabla)\mathbf{\Theta})\,\mathbf{\Theta}^T\,d\mathbf{x}$$
$$\widetilde{\mathbf{Q}}_T = \int_\Omega (\mathbf{\Theta} + (\mathbf{e}\cdot\nabla)\mathbf{\Theta})\,Q\,d\mathbf{x}$$
$$\widetilde{\mathbf{B}}_T = \int_\Omega \sum_{i=1}^{d}\left[k\frac{\partial\mathbf{\Theta}}{\partial x_i} + (\mathbf{\Theta} + (\mathbf{e}\cdot\nabla)\mathbf{\Theta})\,\mathbf{\Psi}^T\mathbf{u}_{x_i}\right]\frac{\partial\mathbf{\Theta}^T}{\partial x_i} + \dots$$
$$k\sum_{i=1}^{d}\left[e_i\frac{\partial^2\mathbf{\Theta}}{\partial x_i^2}\frac{\partial\mathbf{\Theta}^T}{\partial x_i} + \sum_{j>i}\frac{\partial^2\mathbf{\Theta}}{\partial x_i x_j}\left(e_i\frac{\partial\mathbf{\Theta}}{\partial x_j} + e_j\frac{\partial\mathbf{\Theta}}{\partial x_i}\right)^T\right]d\mathbf{x}$$

where $\mathbf{\Theta} + (\mathbf{e}\cdot\nabla)\mathbf{\Theta}$ is the perturbed test function. It should be noted that the components multiplied with $\mathbf{e}$ will change for each time step because the velocity field is varying over time, which again changes the value of $\mathbf{e}$. The integrals of $\mathbf{\Theta}$ and $\mathbf{\Psi}$ are invariant of time evolution because they depend on the spatial derivatives. Only $\mathbf{e}$ and $\mathbf{u}$ are not constant and change the SUPG-matrices. Hence, it can be appropriate to store the integrals efficiently with complex linked lists and use them as function handles for the matrices. In this way, we do not need to calculate the integrals over and over again for each time step, for it is only the coefficient vectors for the velocity field that changes during the whole simulation.

SUPG on Navier-Stokes equations

There are several ways of creating an SUPG-formulation of the Navier-Stokes equations. They are chosen after special conditions like turbulence and compressibility in order to make them more flexible. For the incompressible model, the SUPG method formulated for the convection-diffusion equation is exactly the same for the Navier-Stokes equation. The main difference is that we replace P_j with R_j, the *local Reynolds number* [8]:

$$\beta_j = \coth(R_j) - \frac{1}{R_j} \qquad\qquad R_j = \frac{\rho_0 u_j h_j}{2\mu} \tag{5.18}$$

Several numerical experiments have shown that this approach above works properly for the Navier-Stokes equations. We can also determine the tuning parameter β with the L^2-norm as in equation (5.17). Unfortunately, there was not enough time to implement SUPG on Navier-Stokes, so we will restrict ourselves to a problem which is relatively easy to solve and also diffusion-dominated.

5.4 General remarks on the global assembly process

5.4.1 Brief description of the local assembly

All the integrals defined in (5.5) must be evaluated on each individual element before they are inserted into the global matrices and vectors used in the discrete equation system. The only difference is that the domain changes from Ω (global) to Ω_e (local), and we must detect which spline functions are defined on this local element. But the integrand formula is still the same. To do so, we need an efficient enumeration of each element, and this will be used to determine how the local matrices and vectors will be inserted into the global ones. There are many ways of doing this. If the domain is a square or rectangle, as in our introductory case, it is easiest to use standard *lexicographical* enumeration of the elements, as depicted in figure 5.2. This procedure is quite straightforward.

In any case, the procedure of inserting the local components depends on the polynomial degree and continuity of the spline interpolants. High continuity implies that the global matrices and vectors become smaller and smaller when rising the polynomial degree, and the spectral radii decreases. This is one of the superior advantages isogeometric analysis has over the classical finite element methodology as pointed out in chapter 4.

51	52	53	54	55	56	57	58	59	60
41	42	43	44	45	46	47	48	49	50
31	32	33	34	35	36	37	38	39	40
21	22	23	24	25	26	27	28	29	30
11	12	13	14	15	16	17	18	19	20
1	2	3	4	5	6	7	8	9	10

Figure 5.2. *Lexicographical enumeration on a rectangle*

The *physical coordinates*, (x, y, z), are those ones describing the actual geometry of the real physical domain Ω^e. They are mapped first to the *parameter coordinates* $(\widehat{\xi}, \widehat{\eta}, \widehat{\zeta})$ of the parameter space $\widehat{\Omega}^e$, used for the interpolating shape functions. Lastly, they are mapped to the *parent coordinates* $(\widetilde{\xi}, \widetilde{\eta}, \widetilde{\zeta})$ of the integration domain $\widetilde{\Omega}^e$, as shown in figure 5.3. This is vital for the numerical integration on each element, which requires a continuous and isoparametric mapping to the parent elements. The whole process, expressed through the Jacobian of the coordinate transform, can be stated as

$$\begin{aligned} \iiint\limits_{\Omega_e} f(x, y, z)\, dx\, dy\, dz &= \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} g(\widetilde{\xi}, \widetilde{\eta}, \widetilde{\zeta}) \big|\det(\mathbf{J})\big|\, d\widetilde{\xi}\, d\widetilde{\eta}\, d\widetilde{\zeta} \\ &\approx \sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\sum_{k=1}^{n_z} w_i w_j w_k\, g(\widetilde{\xi}_i, \widetilde{\eta}_j, \widetilde{\zeta}_k) \big|\det(\mathbf{J})\big| \end{aligned} \tag{5.19}$$

The Jacobian of the isoparametric coordinate mapping is defined as

$$\mathbf{J}(\widetilde{\xi}, \widetilde{\eta}, \widetilde{\zeta}) = \begin{bmatrix} \frac{\partial x}{\partial \widetilde{\xi}} & \frac{\partial y}{\partial \widetilde{\xi}} & \frac{\partial z}{\partial \widetilde{\xi}} \\ \frac{\partial x}{\partial \widetilde{\eta}} & \frac{\partial y}{\partial \widetilde{\eta}} & \frac{\partial z}{\partial \widetilde{\eta}} \\ \frac{\partial x}{\partial \widetilde{\zeta}} & \frac{\partial y}{\partial \widetilde{\zeta}} & \frac{\partial z}{\partial \widetilde{\zeta}} \end{bmatrix} \tag{5.20}$$

When the domain is a square or rectangle, the Jacobian will always have a constant value depending on the length and width of each element, and it does not change indeed. But such domains are quite simple, and in the general case, the coordinate mapping shown in (5.19) must be invoked. Using the same notation as Hughes [35], we can describe the assembly process of the global matrices and vectors as

$$\mathbf{M} = \mathop{\mathsf{A}}_{e=1}^{n_e}(\mathbf{M}_e) \quad , \quad \mathbf{f} = \mathop{\mathsf{A}}_{e=1}^{n_e}(\mathbf{f}_e) \tag{5.21}$$

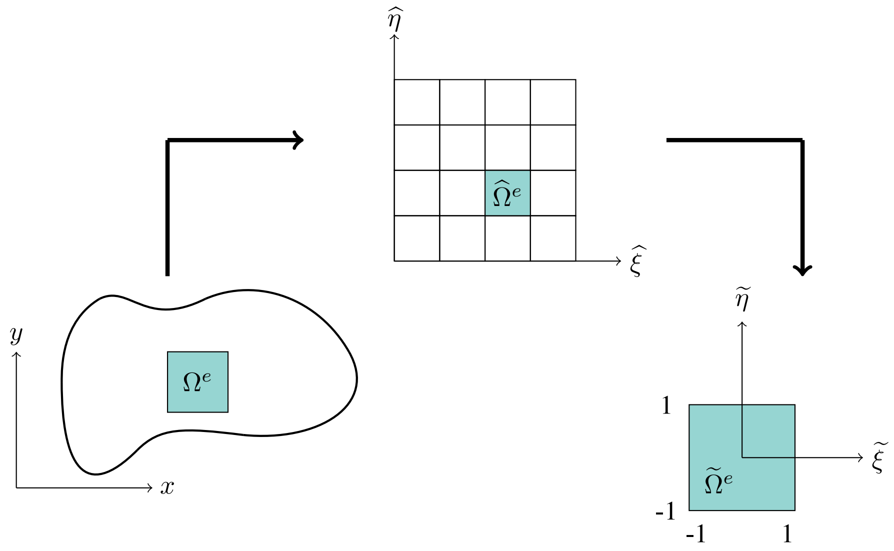


Figure 5.3. *Mapping between different spaces in the quadrature process*

5.4.2 Implementation of boundary conditions

The finite element method is very flexible because it can handle any boundary condition by incorporating them directly in the weak formulation, instead of approximating them as in the finite difference method. Although we are using splines as the new basis functions in isogeometric analysis, the procedure is exactly the same as before.

Neumann conditions are often called *natural* because they are implemented as vectors in addition to the load vector on the right-hand side of the equation. We loop over the element edges, calculate line integrals, and then we assemble them in a vector.

Dirichlet conditions are *essential* because they are enforced in the system of equations. If $u = g$ on $\partial\Omega$, we can express this as $\mathbf{Bu} = \mathbf{g}$, where $\mathbf{B}$ is a rectangular constraint matrix. In terms of Lagrange multipliers, the system of equations becomes

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}$$

The easiest procedure is removing the entries in $\mathbf{A}$ and $\mathbf{f}$ corresponding to the boundary, solve the modified system $\widetilde{\mathbf{A}}\widetilde{\mathbf{u}} = \widetilde{\mathbf{f}}$, and then insert $\mathbf{g}$ correctly into $\widetilde{\mathbf{u}}$ to obtain $\mathbf{u}$.

For Robin conditions (*radiation*), we must combine both procedures mentioned above simultaneously, since displacement and flux are coupled together at once.

5.4.3 Optimal quadrature

In the finite element method, *Gaussian quadrature* is used in the assembly, and it works because the continuity between most of the elements is C^{-1}. The splines form a smooth subspace of C^{-1}, so Gaussian quadrature works, and it integrates polynomials exactly if we use enough quadrature points. But in isogeometric analysis, high continuity of the splines is required for increasing accuracy, and this causes a problem pointed out in [37].

Gaussian quadrature neglects high continuity between the elements, so it integrates the splines more than necessary, and the assembly speed is slowed down. Hence, it becomes desirable to create weights and nodes to an individual knot vector. The optimal quadrature of splines should work for any polynomial degree and continuity. If the spline space has dimension n, we do not need more than $\frac{n+1}{2}$ quadrature points. If the total running time of Gaussian quadrature is $\mathcal{O}(N^d)$, optimal quadrature can reduce it up to $\mathcal{O}((N/2)^d)$, where d is the Euclidean dimension. A full description of this algorithm is given in [39].

This approach was tested in the early assembly implementation, but it only worked for $p \leq 3$. The reason was too low continuity causing divergence in the Newton iteration. This had already been pointed out because the nonlinear equation system for nodes and weights is very ill-conditioned. Hence, Gaussian quadrature had to be used. In the future, it can be actual to use a more robust quadrature method that converges for any continuity.

5.5 Post-processing of the solution

When we solve a partial differential equation by the finite difference method, we obtain a solution vector containing evaluations of the unknown solution at discrete points, and this can be plotted directly without complications. The more discretization points or higher order of the scheme, the more will the discrete solution converge to the exact solution. In terms of functional analysis, this is an example of *strong convergence*.

In the finite element method, anyway, the solution vector contains coefficients of the shape functions that are used to interpolate the unknown solution, so the plotting becomes quite different. We must loop over every element, plot the local solutions, and then combine all these local plots together such that they form the final graph or surface of the unknown solution. Since the coefficients are used in a functional describing the unknown solution, the convergence is classified as *weak*. The number of element shape functions increases covariantly with the polynomial degree and the total number of elements.

The static visualizations are done in MATLAB, and this goes relatively fast because the solution is stationary. But the time-dependent problems requires too much computational effort to visualize, so we have chosen to use GLview Inova. To do so, we create a .vtf-file for the numerical data, and then we use it directly as input for visualization. The final calculation of relative error in the L^2- and H^1-norms are always done in MATLAB, for we only require the final coefficient vector for doing this task.

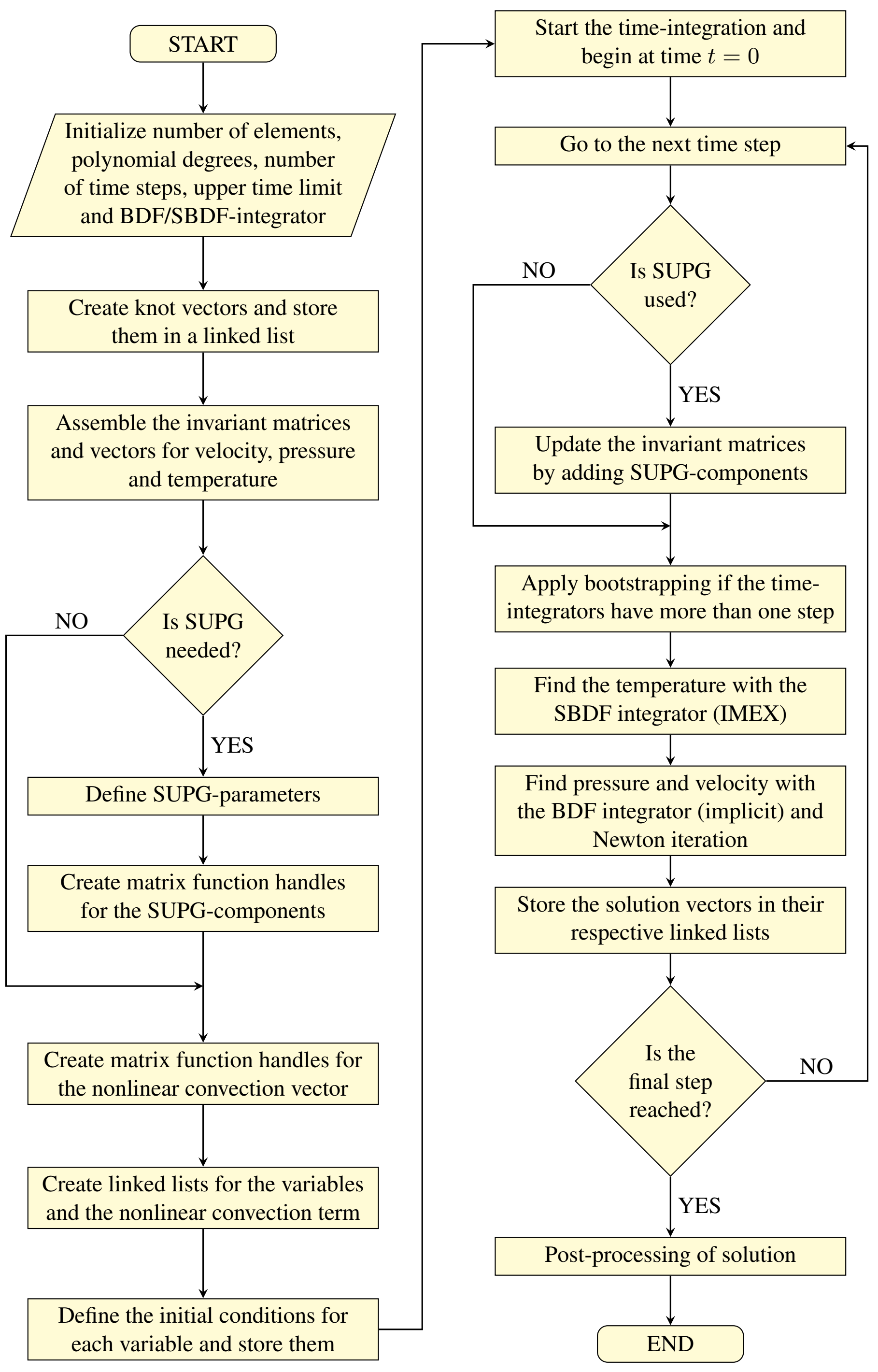

START
Initialize number of elements, polynomial degrees, number of time steps, upper time limit and BDF/SBDF-integrator
Create knot vectors and store them in a linked list
Assemble the invariant matrices and vectors for velocity, pressure and temperature
Is SUPG needed?
NO
YES
Define SUPG-parameters
Create matrix function handles for the SUPG-components
Create matrix function handles for the nonlinear convection vector
Create linked lists for the variables and the nonlinear convection term
Define the initial conditions for each variable and store them
Start the time-integration and begin at time $t = 0$
Go to the next time step
Is SUPG used?
NO
YES
Update the invariant matrices by adding SUPG-components
Apply bootstrapping if the time-integrators have more than one step
Find the temperature with the SBDF integrator (IMEX)
Find pressure and velocity with the BDF integrator (implicit) and Newton iteration
Store the solution vectors in their respective linked lists
Is the final step reached?
NO
YES
Post-processing of solution
END

Figure 5.4. *Complete flowchart for solving the Boussinesq equations*

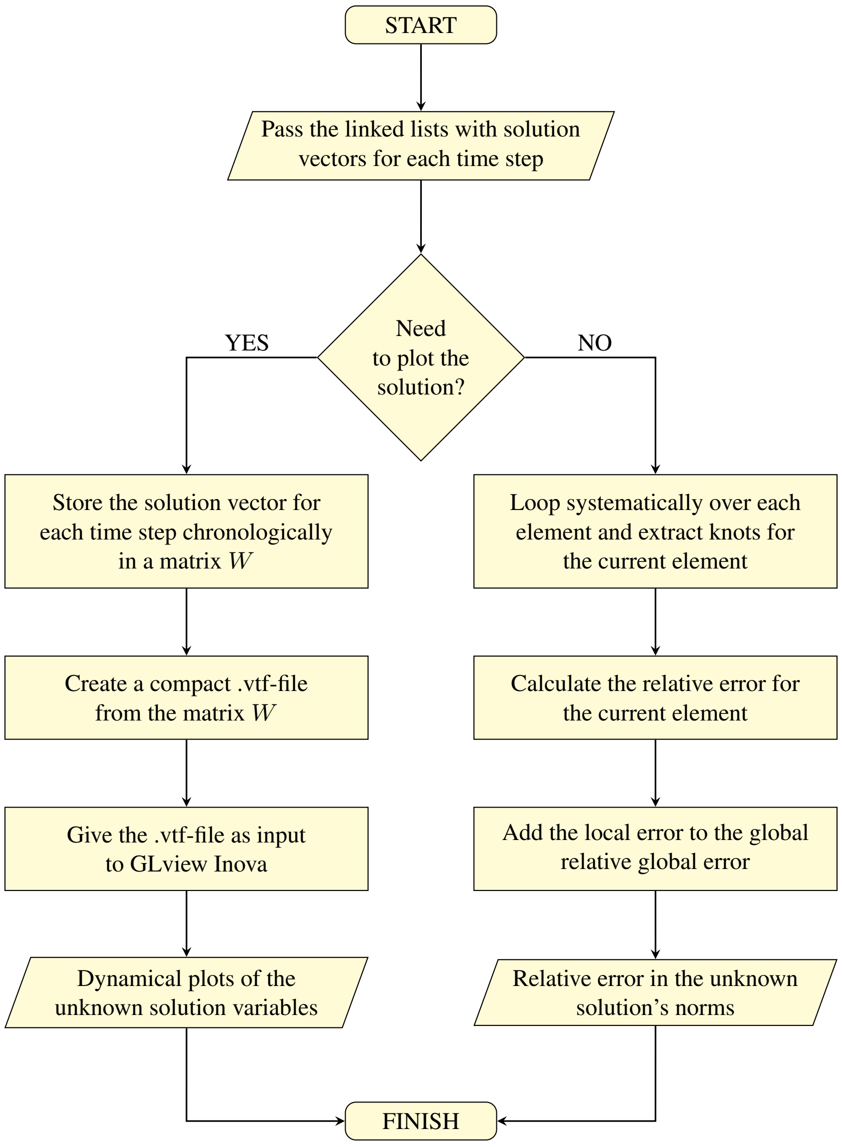

START
Pass the linked lists with solution vectors for each time step
Need to plot the solution?
YES
NO
Store the solution vector for each time step chronologically in a matrix W
Create a compact .vtf-file from the matrix W
Give the .vtf-file as input to GLview Inova
Dynamical plots of the unknown solution variables
Loop systematically over each element and extract knots for the current element
Calculate the relative error for the current element
Add the local error to the global relative global error
Relative error in the unknown solution's norms
FINISH

Figure 5.5. *Flowchart for post-processing the solution*

1

[1]The MATLAB-file used for creating .vtf-files was originally written by Kjetil André Johannessen in 2012 for the classical finite element method, and later extended for isogeometric analysis with B-splines.

Chapter 6

Numerical examples

6.1 General theory of refinement

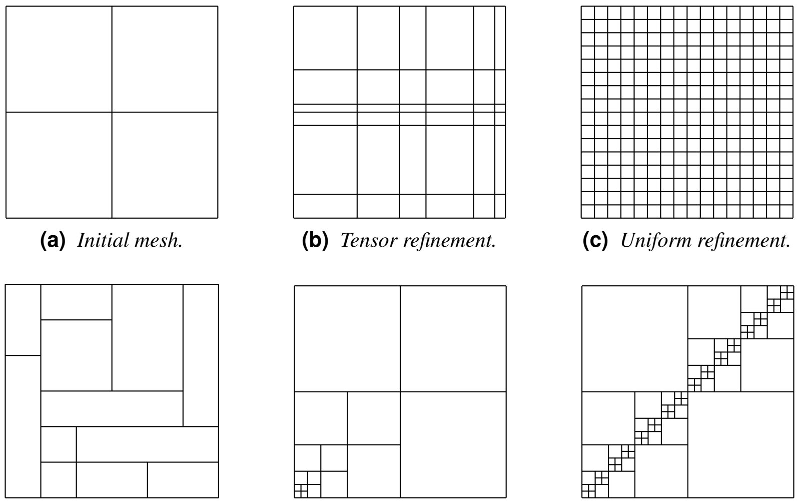
6.1.1 Incorporation in isogeometric analysis

Refinement is an important feature of the finite element method. It allows us to verify that the code is correct through convergence analysis, and can be used to make the numerical simulation more accurate in cases where geometry of the domain is complicated. All the classical refinement techniques are directly incorporated in isogeometric analysis and can be implemented such that the computational complexity is significantly reduced. Using splines allows us to apply refinement methods that preserves the geometric structure and its parametrization [12].

In h-refinement, the polynomial degree p of the shape functions is constant, and the mesh size h is reduced. This can be done in several ways [63]:

- *Uniform Mesh Refinement* (UMR): The global mesh is preserved, and every element is divided repeatedly into smaller elements over and over again.
- *Element Subdivision*: Only some selected elements are locally refined individually, and by proper handling of "hanging nodes", we achieve a conformal mesh.
- *Remeshing*: The whole global mesh is discarded and replaced by a new one.

In p-refinement, we maintain the global mesh while increasing p. This process can be done globally (degree elevation on every shape function) or locally (only some functions are elevated). If h- and p-refinement are combined, we can obtain hp-convergence.



(a) *Initial mesh.* **(b)** *Tensor refinement.* **(c)** *Uniform refinement.*

(d) *T-mesh refinement.* **(e)** *Local refinement.* **(f)** *Diagonal refinement.*

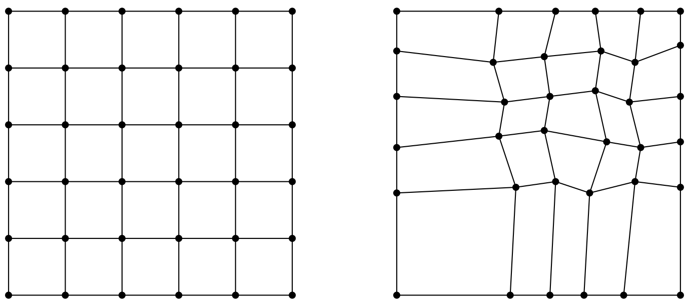
Figure 6.1. *Examples of refinement using different types of splines.*

These refinement techniques exist in isogeometric analysis with additional advantages. Reducing the mesh size is equivalent to knot insertion, and this can be done such that the continuity of a spline is preserved on its interval of support. The increase of polynomial degree and continuity of a spline are covariant, and the numerical accuracy will be higher. We do not need to start from scratch with a C^0-continuous mesh, and computational effort is lower that the classical finite element approach.

B-splines and NURBS provide *tensor refinement*. If we refine a single element, we must refine the other ones such that the global mesh remains conformal. If we use LR B-splines or T-splines, we can apply local refinement over the elements in many interesting ways [40, 41, 27, 24]. However, both these methods are quite complicated to implement from scratch and beyond the scope of this thesis, so we will not consider them.

Isogeometric analysis provides k-refinement. It is applicable due to the homogeneous structure of the patches, and the growth of control variables is limited. When we want to increase the polynomial order from p to $p+1$, we increase the continuity similarly from q to $q+1$. This yields a C^q-continuous spline, so we can raise the degree and continuity simultaneously. This is not possible in the classical finite element method. The operators for h- and p-refinement are not commutative, so if we use knot insertion first, then the continuity is the same on the new knots after degree elevation [12].

Another strategy is r-refinement [63]. The number of nodes on the mesh is constant, but their position is adjusted such that the error is reduced. This special refinement was complicated to achieve in the past, but now it is easier because of efficient mappings and exact geometry representations provided by isogeometric analysis [12].



(a) *Uniformly distributed nodes.* **(b)** *Adjusted nodes*

Figure 6.2. *Illustration of node adjustment in r-refinement.*

Using the same notation as in [42], we define $\mathcal{M}$ as the initial mesh and $\overline{\mathcal{M}}$ is the new mesh obtained by halving h. If $\mathcal{S}_h^{p,k}(\mathcal{M})$ is the isogeometric finite element subspace of $\mathcal{M}$, then the three refinement strategies can be defined mathematically as

$$\mathcal{S}_h^{p,k}(\mathcal{M}) \xrightarrow{h\text{-refinement}} \mathcal{S}_{h/2}^{p,k}(\overline{\mathcal{M}}) \tag{6.1a}$$

$$\mathcal{S}_h^{p,k}(\mathcal{M}) \xrightarrow{p\text{-refinement}} \mathcal{S}_h^{p+1,k}(\mathcal{M}) \tag{6.1b}$$

$$\mathcal{S}_h^{p,k}(\mathcal{M}) \xrightarrow{k\text{-refinement}} \mathcal{S}_h^{p+1,k+1}(\mathcal{M}) \tag{6.1c}$$

It can also be shown that these subspaces satisfy

$$\mathcal{S}_h^{p,k}(\mathcal{M}) \subseteq \mathcal{S}_{h/2}^{p,k}(\overline{\mathcal{M}}) \tag{6.2a}$$

$$\mathcal{S}_h^{p,k}(\mathcal{M}) \subseteq \mathcal{S}_h^{p+1,k}(\mathcal{M}) \tag{6.2b}$$

$$\mathcal{S}_h^{p,k}(\mathcal{M}) \nsubseteq \mathcal{S}_h^{p+1,k+1}(\mathcal{M}) \tag{6.2c}$$

$$\mathcal{S}_h^{p,k}(\mathcal{M}) \nsupseteq \mathcal{S}_h^{p+1,k+1}(\mathcal{M}) \tag{6.2d}$$

6.1.2 Remarks on error estimates

When we analyze h- and p-refinement, it is preferable to use the *number of degrees of freedom* (ndof), the total number of unknowns in the discrete equation system. This is because the mesh size can vary freely, and the elements can even be locally refined. If the domain is $\Omega \subset \mathbb{R}^d$, and the splines' polynomial degree is p, then we can express the convergence rate in the H^k-norm asymptotically as

$$\mathcal{O}\left(\text{ndof}^{-\frac{p+1-k}{2^{d-1}}}\right) \tag{6.3}$$

The general convergence estimates for refinement are

$$\text{Smooth:} \qquad \|u - u_h\|_E \leq Ch^p \|u\|_{H^{p+1}} \tag{6.4a}$$

$$\text{Non-smooth:} \qquad \|u - u_h\|_E \leq Ch^\alpha \|u\|_{H^{\alpha+1}} \tag{6.4b}$$

where $\alpha = \min\{p, \lambda\}$, and λ is a real number characterizing the singularity strength [40]. The last estimate holds for insufficiently smooth functions. In both cases, the convergence is algebraic for h-refinement and exponential for p-refinement.

For time-dependent problems, the situation becomes somewhat different, for in this case, there are two sources of inaccuracy in the simulations: *spatial* and *temporal errors*. The relative error in the energy norm can be expressed as

$$\frac{\|u - u_h\|_E}{\|u\|_E} \leq C_1 h^p + C_2 (\Delta t)^s \tag{6.5}$$

where s is the order of the time-integrator, and Δt is the time step. We define t-refinement as the process where h and p are constant, while Δt is halved repeatedly. Hence, we expect that the convergence rate in t-refinement is the same for the relative L^2-norm and H^1-seminorm.

6.2 Hierachy of the models

It was pointed out in the early stages in this thesis that implementing the whole Boussinesq system at once from scratch would be very risky because it is quite difficult to debug such a complex system. First of all, this is a system of partial differential equations with quasi-linear structure, and there are many algorithms that must be combined simultaneously in order to solve it. Therefore, we decided here to build up the whole code by analyzing the different parts separately and making sure that all of them worked properly as they should. At the end, all these parts were combined together into a single unity which could be used for solving the Boussinesq equations.

Because of this systematic work methodology, were the code is builded block by block, we will present all the convergence studies which have been performed during the work on the different parts of the Boussinesq system. In the specialization project during the autumn 2015, the Poisson equation and Stokes' equation were solved using isogeometric analysis, and those algorithms were used as a basis for the present study. The Poisson equation was extended to the heat equation by adding the time-dependent term, and then we added a linear convective term to obtain a solver for the unsteady advection-diffusion equation. The Navier-Stokes' equations were created similarly by extending the Stokes solver, but this was more challenging because Newton-iteration had to be applied, and there was a need for effective routines updating the nonlinear convection for each iteration.

When the unsteady advection-diffusion and Navier-Stokes solvers were implemented and tested, the last stage was to combine them together and obtain a code for the Boussinesq system. At this point, we found it convenient to segregate the system properly to reduce the computational complexity, and this approach did work very well. The whole procedure for building up the Boussinesq solver is shown in figure 6.3.

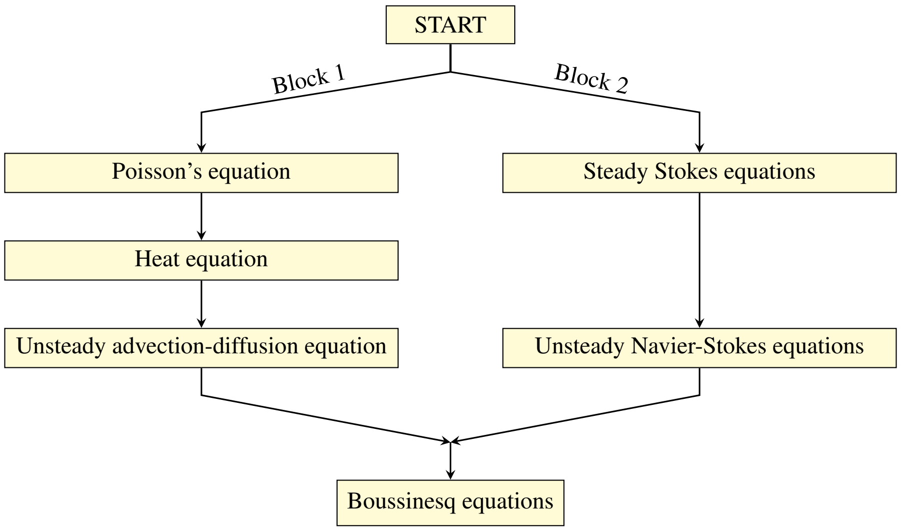


Figure 6.3. *Procedure for building up the Boussinesq solver blockwise.*

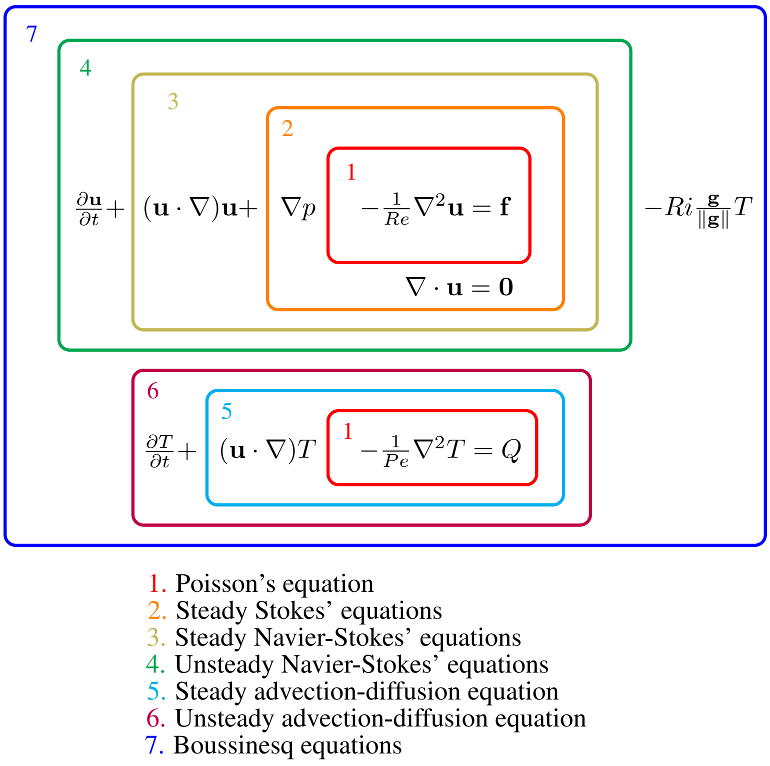

7
4
3
2
1
$\frac{\partial \mathbf{u}}{\partial t}+$
$(\mathbf{u}\cdot\nabla)\mathbf{u}+$
∇p
$-\frac{1}{Re}\nabla^2\mathbf{u}=\mathbf{f}$
$\nabla\cdot\mathbf{u}=\mathbf{0}$
$-Ri\frac{\mathbf{g}}{\|\mathbf{g}\|}T$
6
5
1
$\frac{\partial T}{\partial t}+$
$(\mathbf{u}\cdot\nabla)T$
$-\frac{1}{Pe}\nabla^2 T=Q$
1. Poisson's equation
2. Steady Stokes' equations
3. Steady Navier-Stokes' equations
4. Unsteady Navier-Stokes' equations
5. Steady advection-diffusion equation
6. Unsteady advection-diffusion equation
7. Boussinesq equations

Figure 6.4. *Illustration of the composition of the Boussinesq system.*

6.3 Heat equation

6.3.1 Manufactured reference solution

If u is a continuous temperature distribution, and κ is the thermal diffusivity, then the homogeneous heat equation is defined as

$$\frac{\partial u}{\partial t} = \kappa \nabla^2 u \tag{6.6}$$

The domain is a rectangle, $\Omega = (0, L_x) \otimes (0, L_y)$. We assume homogeneous Dirichlet conditions along the whole boundary $\partial\Omega$. We apply separation of variables by setting $u(x, y, t) = X(x)Y(y)T(t)$ and defining γ as a positive constant. This splitting yields

$$\begin{aligned} X(x)Y(y)T'(t) &= \kappa \left[X''(x)Y(y) + X(x)Y''(y)\right] T(t) \\ \frac{T'(t)}{\kappa T(t)} &= \frac{X''(x)}{X(x)} + \frac{Y''(y)}{Y(y)} = -\gamma^2 \end{aligned}$$

We assume from Pythagoras' theorem that $\gamma^2 = \alpha^2 + \beta^2$ and obtain

$$\begin{aligned} X''(x) + \alpha^2 X(x) &= 0 \\ Y''(y) + \beta^2 Y(y) &= 0 \\ T'(t) + \kappa\gamma^2 T(t) &= 0 \end{aligned}$$

Since the Dirichlet conditions are homogeneous, we get the following eigenfunctions:

$$\begin{aligned} X(x) &= \sin\left(\frac{n\pi x}{L_x}\right) \\ Y(y) &= \sin\left(\frac{m\pi y}{L_y}\right) \end{aligned}$$

$X(x)Y(y)$ is the *amplitude function*, and the *time function* becomes

$$T(t) = C(0)e^{-\left[\left(\frac{n\pi}{L_x}\right)^2 + \left(\frac{m\pi}{L_y}\right)^2\right]\kappa t}$$

If we denote the initial condition as $u_0(x, y)$, then $C(0)$ is the double Fourier coefficient d_{nm}. We obtain the final solution

$$u(x, y, t) = \sum_{n=1}^{\infty}\sum_{m=1}^{\infty} d_{nm} \sin\left(\frac{n\pi x}{L_x}\right) \sin\left(\frac{m\pi y}{L_y}\right) e^{-\left[\left(\frac{n\pi}{L_x}\right)^2 + \left(\frac{m\pi}{L_y}\right)^2\right]\kappa t}$$

We eliminate the whole series by assuming that u_0 is a sinusoidal function in the x- and y-directions, and this yields the manufactured reference solution

$$\boxed{u(x, y, t) = \sin\left(\frac{n\pi x}{L_x}\right) \sin\left(\frac{m\pi y}{L_y}\right) e^{-\left[\left(\frac{n\pi}{L_x}\right)^2 + \left(\frac{m\pi}{L_y}\right)^2\right]\kappa t}} \tag{6.8}$$

6.3.2 Discussion on the results

In our analysis, we choose the parameters such that the analytical solution becomes

$$u(x, y, t) = \sin(2\pi x)\sin(4\pi y)\, e^{-t} \tag{6.9}$$

This is a product of trigonometric and exponential functions, so we cannot express it as a finite linear combination of polynomial splines. This is valid both for the amplitude part and time part of the function. When we do convergence studies, we expect the numerical discretization errors to decrease whenever we reduce the mesh size h, increase the degree p, and reduce the time increment Δt. Thus, when we do t-refinement, p and h should be as large and small as possible, respectively. Otherwise, they will generate disturbing noise (large spatial error) making it harder to analyze the individual behavior of the temporal error. Likewise, Δt should be very small in the h- and p-refinements.

In the h- and p-refinements, we set $\Delta t = 0.001$ on the time interval $[0, 1]$ and used the Runge-Kutta Gauss-Legendre (RKGL) method of order 3. This symplectic and symmetric integrator is both A- and B-stable. The linear systems of equations are relatively small, so RKGL works well for our purpose. The graphs decreased quite well in figure 6.6 and 6.7. There were some small irregularities when the error reached 10^{-14}, but this was probably noise in the error computation. The number of Gauss points was chosen such that most quantities were integrated exactly, so the lack of sufficiently enough Gauss quadrature points cannot have influenced the numerical accuracy very much.

In the t-refinement, we set $p = 9$ and $h = 1/40$ for the multistep integrators Adams-Moulton Formula (AMF) and Backward Differentiation Formula (BDF), and $h = 1/50$ for the RKGL-integrators. In total, eight integrators were tested. We see from figure 6.8 and 6.9 that the numerical and analytical convergence graphs coincide with each other, which indicates strongly that the simulation is reliable. The graphs of the three last RKGL-integrators reached machine precision quite fast instead of decaying gradually as the two first ones. This behaviour was expected due to their high order.

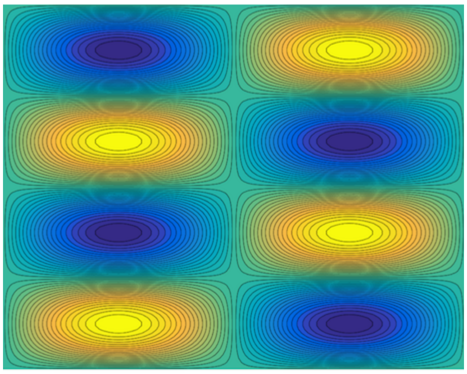
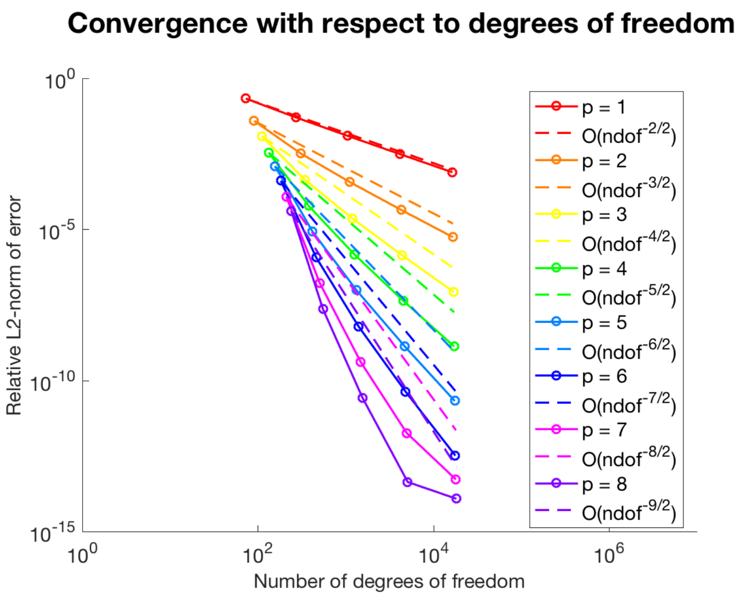
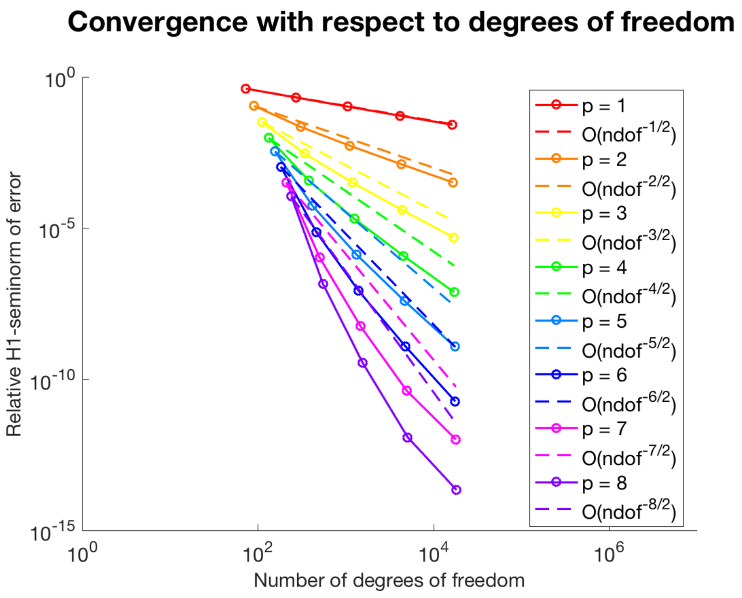


Figure 6.5. Heat equation: *Contour plot of manufactured reference solution.*

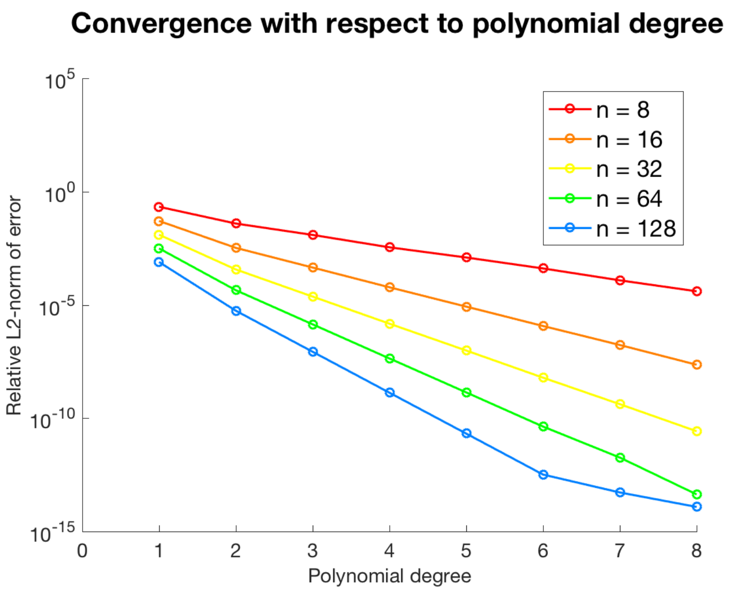


(a) *h-refinement: Relative error (L^2-norm).*

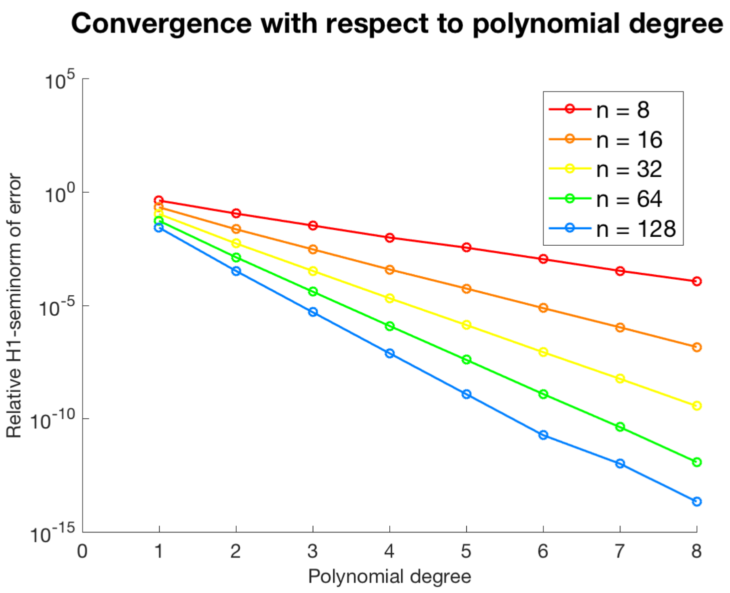


(b) *h-refinement: Relative error (H^1-seminorm).*

Figure 6.6. ***Heat equation:*** *h-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*



(a) *p-refinement: Relative error (L^2-norm).*



(b) *p-refinement: Relative error (H^1-seminorm).*

Figure 6.7. ***Heat equation:*** *p-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*

Table 6.1. Heat equation: *UMR with* $p = 1$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
49	1/8	2.1409e-01	-	4.1005e-01	-
225	1/16	5.0641e-02	2.0798	2.0788e-01	0.98002
961	1/32	1.2479e-02	2.0208	1.0437e-01	0.99408
3969	1/64	3.1096e-03	2.0047	5.2239e-02	0.99849
16129	1/128	7.7683e-04	2.0011	2.6126e-02	0.99962

Table 6.2. Heat equation: *UMR with* $p = 2$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
64	1/8	3.8953e-02	-	1.0974e-01	-
256	1/16	3.2898e-03	3.5656	2.2298e-02	2.2991
1024	1/32	3.6593e-04	3.1684	5.2774e-03	2.079
4096	1/64	4.4338e-05	3.0449	1.3011e-03	2.02
16384	1/128	5.4984e-06	3.0115	3.2415e-04	2.005

Table 6.3. Heat equation: *UMR with* $p = 3$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
81	1/8	1.2254e-02	-	3.2483e-02	-
289	1/16	4.4503e-04	4.7832	2.8939e-03	3.4886
1089	1/32	2.3185e-05	4.2626	3.2463e-04	3.1561
4225	1/64	1.3778e-06	4.0728	3.9406e-05	3.0423
16641	1/128	8.5000e-08	4.0187	4.8888e-06	3.0109

Table 6.4. Heat equation: *UMR with* $p = 4$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
100	1/8	3.4773e-03	-	9.5811e-03	-
324	1/16	5.9762e-05	5.8626	3.7550e-04	4.6733
1156	1/32	1.4779e-06	5.3376	2.0164e-05	4.2189
4356	1/64	4.3113e-08	5.0993	1.2146e-06	4.0533
16900	1/128	1.3242e-09	5.0249	7.5423e-08	4.0093

Table 6.5. Heat equation: *UMR with* $p = 5$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
121	1/8	1.2355e-03	-	3.4368e-03	-
361	1/16	8.3811e-06	7.2038	5.4083e-05	5.9898
1225	1/32	9.6396e-08	6.442	1.3310e-06	5.3445
4489	1/64	1.3661e-09	6.1409	3.8712e-08	5.1036
17161	1/128	2.0778e-11	6.0388	1.1866e-09	5.0279

Table 6.6. Heat equation: *UMR with* $p = 6$.

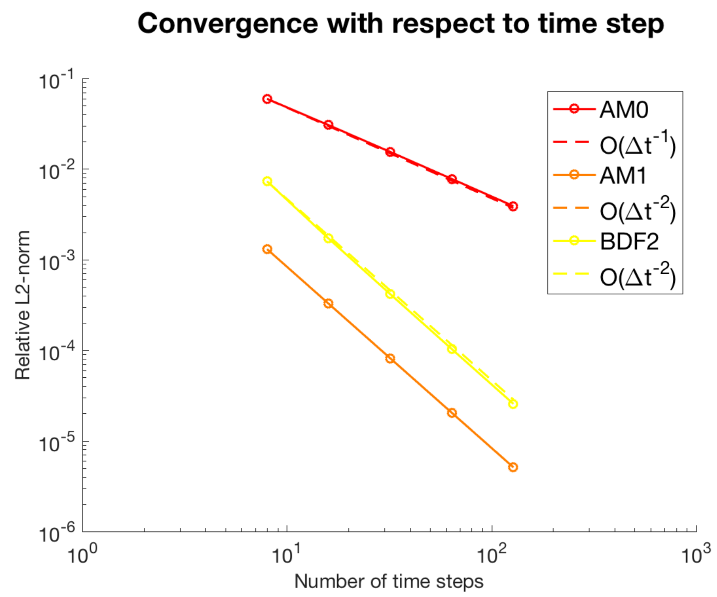
ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
144	1/8	4.0661e-04	-	1.0671e-03	-
400	1/16	1.1701e-06	8.4409	7.4258e-06	7.1669
1296	1/32	6.1146e-09	7.5801	8.4266e-08	6.4614
4624	1/64	4.2391e-11	7.1724	1.2030e-09	6.1303
17424	1/128	3.2254e-13	7.0381	1.8428e-11	6.0285

Table 6.7. Heat equation: *UMR with* $p = 7$.

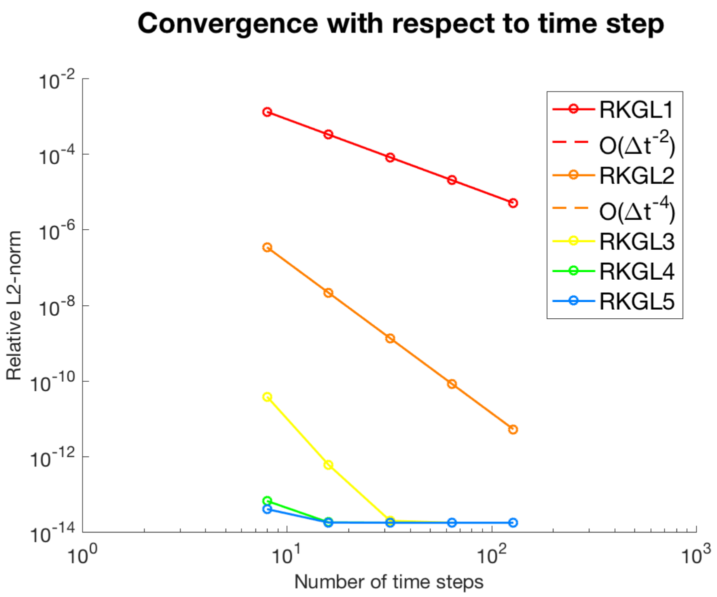
ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
169	1/8	1.2128e-04	-	3.2601e-04	-
441	1/16	1.6942e-07	9.4835	1.0536e-06	8.2734
1369	1/32	4.1321e-10	8.6795	5.6255e-09	7.5491
4761	1/64	1.8171e-12	7.8291	4.1748e-11	7.0741
17689	1/128	5.2241e-14	5.1203	1.0137e-12	5.3641

Table 6.8. Heat equation: *UMR with* $p = 8$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
196	1/8	3.9845e-05	-	1.1209e-04	-
484	1/16	2.2766e-08	10.7733	1.4227e-07	9.6218
1444	1/32	2.6390e-11	9.7527	3.5424e-10	8.6497
4900	1/64	4.3017e-14	9.2609	1.1960e-12	8.2104
17956	1/128	1.2030e-14	1.8383	2.1691e-14	5.7849

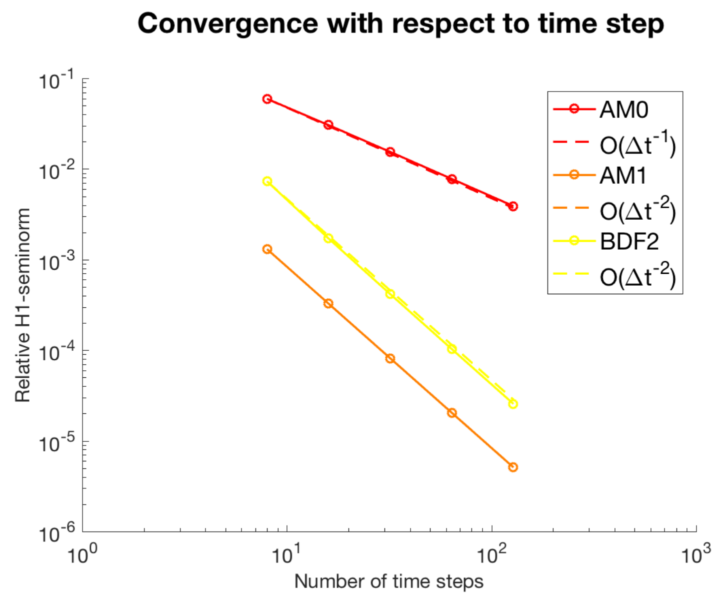


(a) *t-refinement with multistep integrators: Relative error (L^2-norm).*

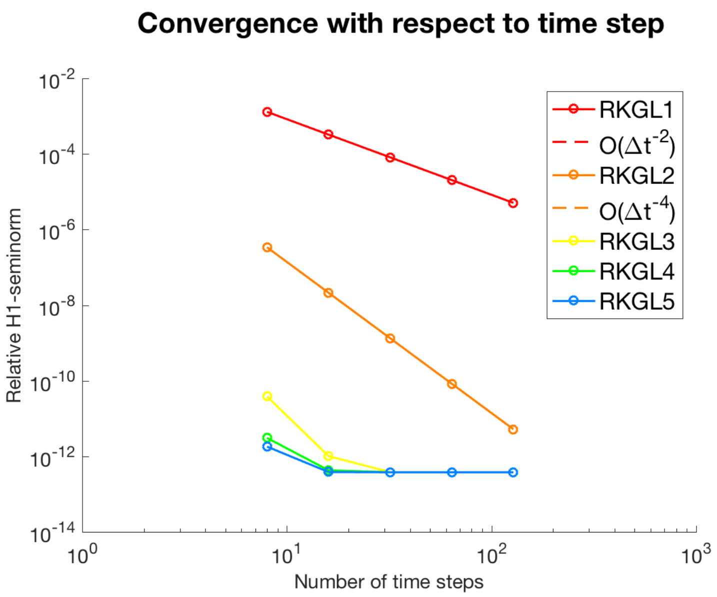


(b) *t-refinement with Runge-Kutta integrators: Relative error (L^2-norm).*

Figure 6.8. ***Heat equation:*** *t-refinement, relative error plots (%) in L^2-norm.*



(a) *t-refinement with multistep integrators: Relative error (H^1-seminorm).*



(b) *t-refinement with Runge-Kutta integrators: Relative error (H^1-seminorm).*

Figure 6.9. ***Heat equation:*** *t-refinement, relative error plots (%) in H^1-seminorm.*

Table 6.9. Heat equation: *t-refinement with AM0.*

Δt	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	5.9435e-02	-	5.9435e-02	-
1/16	3.0461e-02	0.96436	3.0461e-02	0.96436
1/32	1.5425e-02	0.98171	1.5425e-02	0.98171
1/64	7.7620e-03	0.99073	7.7620e-03	0.99073
1/128	3.8936e-03	0.99534	3.8936e-03	0.99534

Table 6.10. Heat equation: *t-refinement with AM1.*

Δt	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	1.3043e-03	-	1.3043e-03	-
1/16	3.2566e-04	2.0018	3.2566e-04	2.0018
1/32	8.1389e-05	2.0005	8.1389e-05	2.0005
1/64	2.0346e-05	2.0001	2.0346e-05	2.0001
1/128	5.0863e-06	2	5.0863e-06	2

Table 6.11. Heat equation: *t-refinement with BDF2.*

Δt	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	7.3022e-03	-	7.3022e-03	-
1/16	1.7061e-03	2.0977	1.7061e-03	2.0977
1/32	4.1557e-04	2.0375	4.1557e-04	2.0375
1/64	1.0274e-04	2.0161	1.0274e-04	2.0161
1/128	2.5554e-05	2.0074	2.5554e-05	2.0074

Table 6.12. Heat equation: *t-refinement with RKGL1.*

Δt	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	1.3043e-03	-	1.3043e-03	-
1/16	3.2566e-04	2.0018	3.2566e-04	2.0018
1/32	8.1389e-05	2.0005	8.1389e-05	2.0005
1/64	2.0346e-05	2.0001	2.0346e-05	2.0001
1/128	5.0863e-06	2	5.0863e-06	2

Table 6.13. Heat equation: *t-refinement with RKGL2.*

Δt	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	3.3940e-07	-	3.3940e-07	-
1/16	2.1198e-08	4.001	2.1198e-08	4.001
1/32	1.3246e-09	4.0003	1.3246e-09	4.0003
1/64	8.2786e-11	4.0001	8.2787e-11	4
1/128	5.1748e-12	3.9998	5.1886e-12	3.996

Table 6.14. Heat equation: *t-refinement with RKGL3.*

Δt	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	3.7867e-11	-	3.8163e-11	-
1/16	5.9108e-13	6.0014	1.0072e-12	5.2437
1/32	1.9513e-14	4.9208	3.7941e-13	1.4086
1/64	1.7429e-14	0.16295	3.7729e-13	0.0081038
1/128	1.7350e-14	0.0065589	3.7648e-13	0.0030852

Table 6.15. Heat equation: *t-refinement with RKGL4.*

Δt	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	6.5519e-14	-	3.0532e-12	-
1/16	1.7803e-14	1.8798	4.2437e-13	2.8469
1/32	1.7453e-14	0.028669	3.7744e-13	0.16907
1/64	1.7371e-14	0.0068064	3.7646e-13	0.0037761
1/128	1.7330e-14	0.003384	3.7609e-13	0.0014096

Table 6.16. Heat equation: *t-refinement with RKGL5.*

Δt	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	3.9911e-14	-	1.7967e-12	-
1/16	1.7563e-14	1.1843	3.8202e-13	2.2337
1/32	1.7406e-14	0.012944	3.7689e-13	0.019482
1/64	1.7352e-14	0.0044182	3.7616e-13	0.0028001
1/128	1.7346e-14	0.0005068	3.7627e-13	-0.00040924

6.4 Unsteady advection-diffusion equation

6.4.1 Manufactured reference solution

The unsteady and homogeneous advection-diffusion equation is given by

$$\frac{\partial u}{\partial t} + (\mathbf{c}\cdot\nabla)u = \kappa\nabla^2 u \tag{6.10}$$

where $\mathbf{c} = (c_1, c_2)$ is a constant advection field and κ is the thermal diffusivity. The unknown function u can be a distribution for temperature or chemical concentration. The boundary value problem is exactly the same as the heat equation. Before using separation of variables, we introduce $u(x, y, t) = v(x, y, t)e^{ax+by}$ to get rid of the advection term. With this substitution, it can be shown that choosing $a = c_1/2\kappa$ and $b = c_2/2\kappa$ will make the first partial derivatives vanish. This yields the unsteady diffusion-reaction equation:

$$\frac{\partial v}{\partial t} - \kappa\nabla^2 v + \frac{c_1^2 + c_2^2}{4\kappa}v = 0$$

Thus, we can find v by separating the variables as before and assuming that the initial condition v_0 is sinusoidal in both directions. After finding v, we obtain the final analytic solution of the unsteady advection-diffusion equation:

$$\boxed{\begin{aligned} & u(x,y,t) = \sin\left(\frac{n\pi x}{L_x}\right)\sin\left(\frac{m\pi y}{L_y}\right)e^{ax+by-Rt} && (6.11a) \\ & a = \frac{c_1}{2\kappa} \qquad b = \frac{c_2}{2\kappa} \qquad R = \kappa\left[\left(\frac{n\pi}{L_x}\right)^2 + \left(\frac{m\pi}{L_y}\right)^2 + \frac{c_1^2+c_2^2}{4\kappa^2}\right] && (6.11b) \end{aligned}}$$

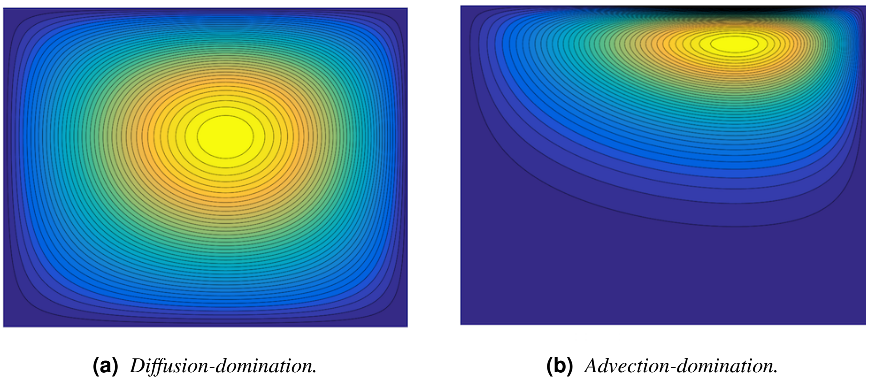
When we analyzed the heat equation, we only had one manufactured solution because the physical problem represented pure diffusion. This time, advection will also be taken into account, so we will use the following two test functions instead:

$$u_1(x, y) = \sin(\pi x)\sin(\pi y)e^{\pi^2(0.05x+0.1y)-(2+0.0125\pi^2)t} \tag{6.12a}$$

$$u_2(x, y) = 0.01\sin(\pi x)\sin(\pi y)e^{\pi^2(0.2x+0.8y)-(1+0.34\pi^2)t} \tag{6.12b}$$

The second function is advection-dominated and will therefore be solved with SUPG-formulation, but not the first one since it is diffusion-dominated. It is not always possible to be completely sure whether we should use SUPG-formulation or not. The best idea is to solve the equation with the standard Bubnov-Galerkin method first, and then use SUPG afterwards if the solution is too inaccurate.

As we see from figure 6.10, the peak of the first solution is almost at the middle of the domain, but the second one is more centered to the upper part and indicates a boundary layer with sharply decreasing gradient.



(a) *Diffusion-domination.* **(b)** *Advection-domination.*

Figure 6.10. Advection-diffusion equation: *Contour plot of manufactured reference solutions.*

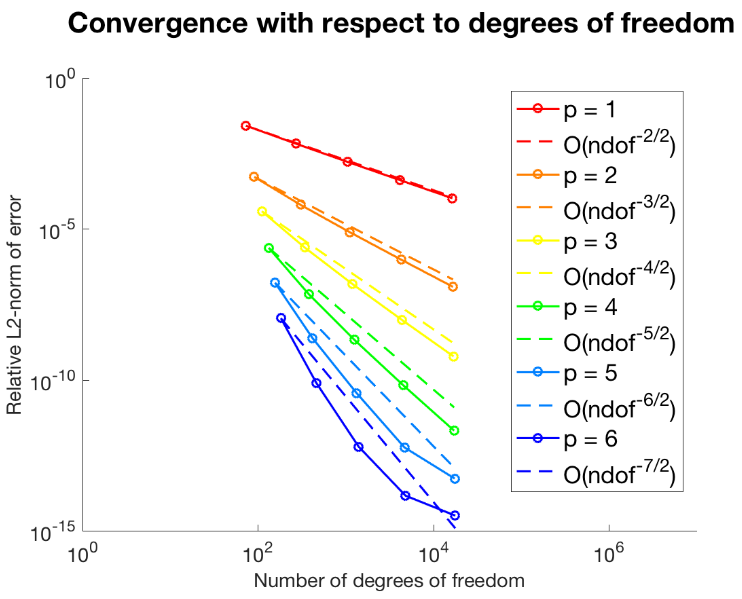
6.4.2 Discussion on the results

As we see from figure 6.11, 6.12, 6.13 and 6.14, the graphs converged very well as they should. There were some small irregularities when the relative error reached 10^{-40}, but this is very close to machine precision. Since $\Delta t = 0.001$ as previously, the temporal error's influence has been quite limited. It also seems that the advection was not so very significant, and the SUPG method worked for this elementary problem.

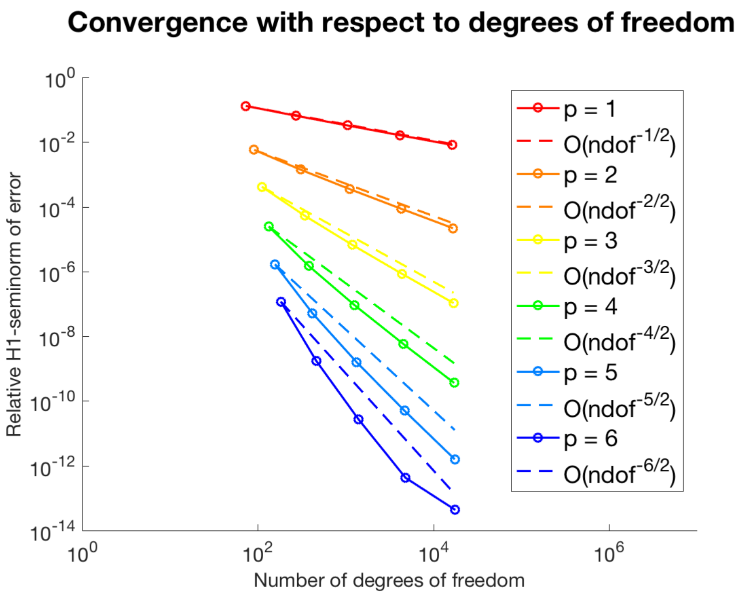
From figure 6.15 and 6.16, the numerical convergence graphs almost coincided with the exact convergence graphs, both for the multistep and Runge-Kutta integrators. Hence, we can conclude that the spatial error has been small enough ($h = 1/50$ and $p = 9$) and did not cause too much noise in the total error.

All the simulations until now shows that the isogeometric discretization of the heat and advection-diffusion equations has worked well. Hence, the latter one is ready for being incorporated in the Navier-Stokes' equations. As a final note, the RKGL integrators could be applied because the systems of equations were not so large, and the total computational effort required for solving them were sufficiently limited. But if the systems of equations are very large, and we still want a high-order method that does not take too much time, it can be appropriate to use diagonally implicit Runge-Kutta methods (DIRK). The structure of these integrators enables us to make a sophisticated type of LU-factorization which can reduce the total amount of memory needed. They are also unconditionally A-stable. A full description of these integrators and their characteristic properties can be found in [33]. All the coefficients of the RKGL integrators up to order 10 are listed in [11]. For more details on Runge-Kutta integrators and their stability properties, we refer to [32, 33, 31].

For the nonlinear simulations in the next sections, it will not be actual to use Runge-Kutta methods because they will be too expensive, and will just restrict ourselves to the standard BDF-integrators because they are not so time-consuming.

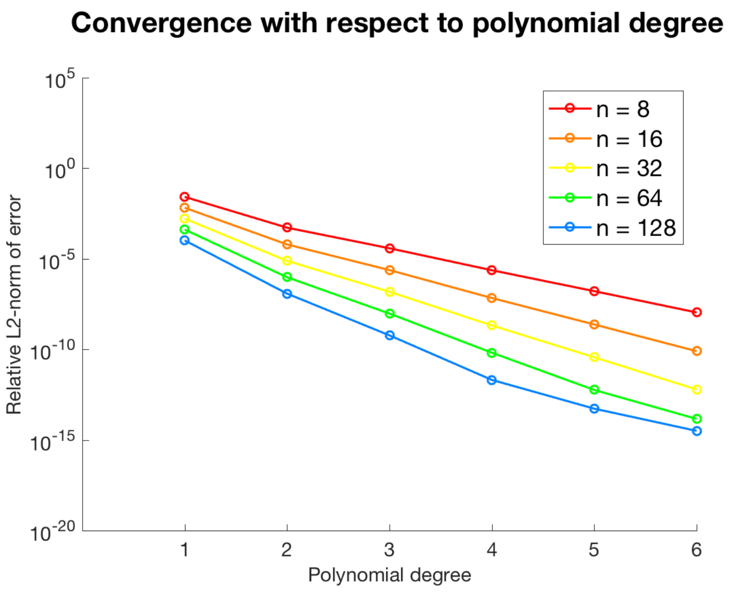


(a) *h-refinement: Relative error (L^2-norm).*

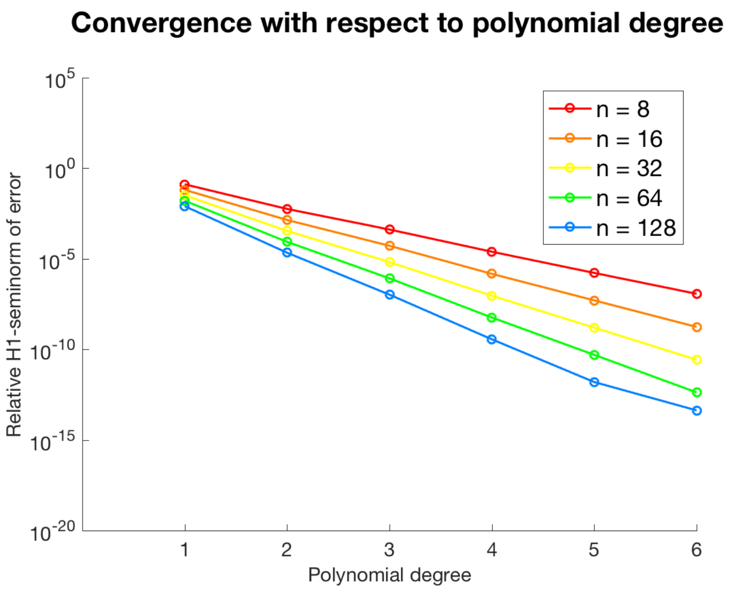


(b) *h-refinement: Relative error (H^1-seminorm).*

Figure 6.11. Advection-diffusion equation, Case 1: *h-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*

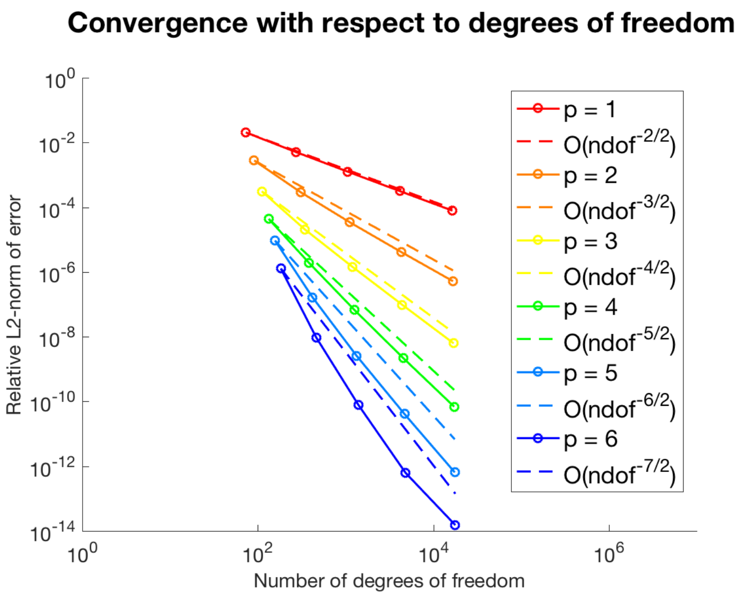


(a) *h-refinement: Relative error (L^2-norm).*

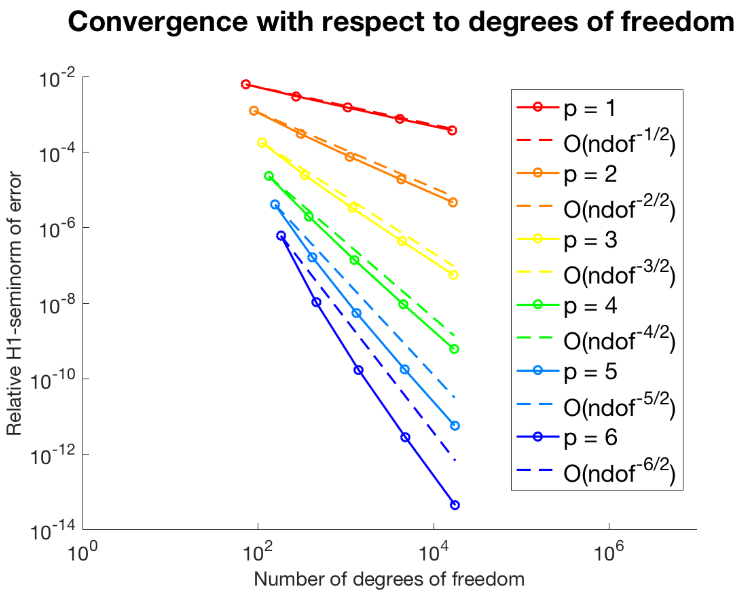


(b) *h-refinement: Relative error (H^1-seminorm).*

Figure 6.12. Advection-diffusion equation, Case 1: *p-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*

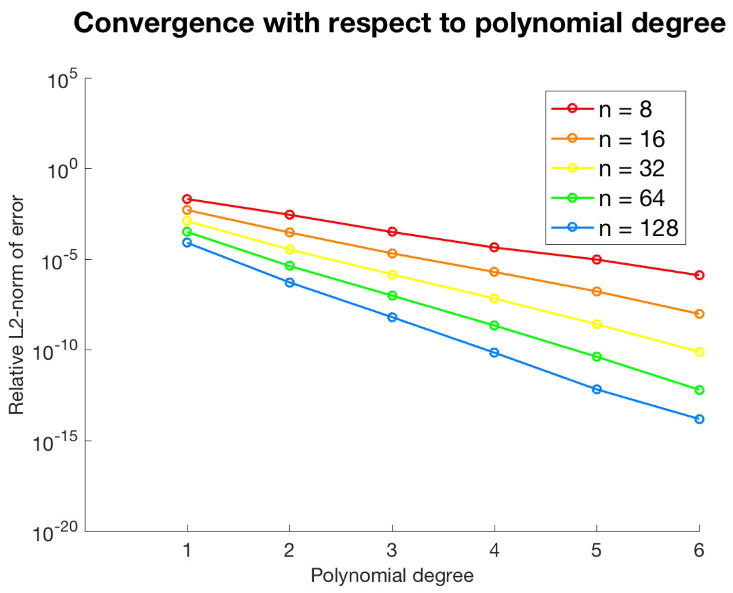


(a) *h-refinement: Relative error (L^2-norm).*

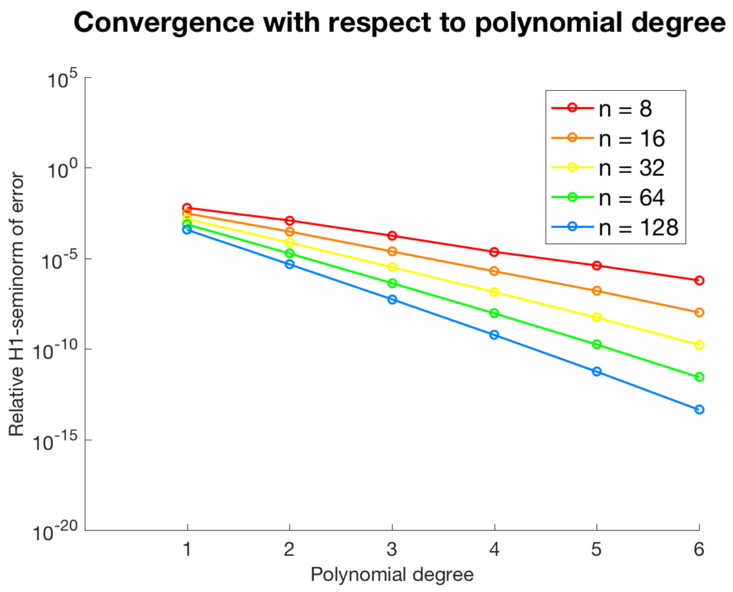


(b) *h-refinement: Relative error (H^1-seminorm).*

Figure 6.13. Advection-diffusion equation, Case 2: *h-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*



(a) *h-refinement: Relative error (L^2-norm).*



(b) *h-refinement: Relative error (H^1-seminorm).*

Figure 6.14. Advection-diffusion equation, Case 2: *p-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*

Table 6.17. **Advection-diffusion equation, Case 1:** *UMR with* $p = 1$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
49	1/8	2.6427e-02	-	1.3073e-01	-
225	1/16	6.6370e-03	1.9934	6.5090e-02	1.0061
961	1/32	1.6617e-03	1.9979	3.2509e-02	1.0016
3969	1/64	4.1558e-04	1.9994	1.6250e-02	1.0004
16129	1/128	1.0391e-04	1.9998	8.1246e-03	1.0001

Table 6.18. **Advection-diffusion equation, Case 1:** *UMR with* $p = 2$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
64	1/8	5.2866e-04	-	5.7289e-03	-
256	1/16	6.2763e-05	3.0744	1.4052e-03	2.0274
1024	1/32	7.7379e-06	3.0199	3.4951e-04	2.0074
4096	1/64	9.6382e-07	3.0051	8.7260e-05	2.0019
16384	1/128	1.2037e-07	3.0013	2.1808e-05	2.0005

Table 6.19. **Advection-diffusion equation, Case 1:** *UMR with* $p = 3$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
81	1/8	3.8446e-05	-	4.1688e-04	-
289	1/16	2.3997e-06	4.0019	5.2803e-05	2.9809
1089	1/32	1.5114e-07	3.9889	6.6760e-06	2.9836
4225	1/64	9.5047e-09	3.9911	8.4034e-07	2.9899
16641	1/128	5.9624e-10	3.9947	1.0545e-07	2.9945

Table 6.20. **Advection-diffusion equation, Case 1:** *UMR with* $p = 4$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
100	1/8	2.3527e-06	-	2.4745e-05	-
324	1/16	6.8638e-08	5.0991	1.4852e-06	4.0584
1156	1/32	2.1073e-09	5.0256	9.2106e-08	4.0112
4356	1/64	6.5583e-11	5.0059	5.7543e-09	4.0006
16900	1/128	2.0636e-12	4.9901	3.5990e-10	3.999

Table 6.21. **Advection-diffusion equation, Case 1:** *UMR with* $p = 5$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
121	1/8	1.6460e-07	-	1.6692e-06	-
361	1/16	2.3812e-09	6.1111	5.0804e-08	5.0381
1225	1/32	3.6626e-11	6.0227	1.5910e-09	4.9969
4489	1/64	5.7394e-13	5.9958	4.9988e-11	4.9922
17161	1/128	5.3252e-14	3.43	1.5705e-12	4.9922

Table 6.22. **Advection-diffusion equation, Case 1:** *UMR with* $p = 6$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
144	1/8	1.0955e-08	-	1.1744e-07	-
400	1/16	7.9642e-11	7.1039	1.7398e-09	6.0768
1296	1/32	6.1195e-13	7.024	2.6865e-11	6.0171
4624	1/64	1.4464e-14	5.4029	4.2084e-13	5.9963
17424	1/128	3.1604e-15	2.1942	4.2703e-14	3.3009

Table 6.23. **Advection-diffusion equation, Case 2:** *UMR with* $p = 1$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
49	1/8	2.0525e-02	-	6.1663e-03	-
225	1/16	5.0753e-03	2.0158	3.0152e-03	1.0321
961	1/32	1.2645e-03	2.005	1.4985e-03	1.0088
3969	1/64	3.1583e-04	2.0013	7.4809e-04	1.0022
16129	1/128	7.8937e-05	2.0003	3.7390e-04	1.0006

Table 6.24. **Advection-diffusion equation, Case 2:** *UMR with* $p = 2$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
64	1/8	2.7945e-03	-	1.2370e-03	-
256	1/16	2.9407e-04	3.2484	3.0209e-04	2.0338
1024	1/32	3.4161e-05	3.1057	7.4600e-05	2.0177
4096	1/64	4.1770e-06	3.0318	1.8571e-05	2.0062
16384	1/128	5.1909e-07	3.0084	4.6369e-06	2.0018

Table 6.25. Advection-diffusion equation, Case 2: *UMR with* $p = 3$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
81	1/8	3.1290e-04	-	1.7810e-04	-
289	1/16	2.0777e-05	3.9126	2.4063e-05	2.8878
1089	1/32	1.4630e-06	3.828	3.2592e-06	2.8842
4225	1/64	9.7314e-08	3.9102	4.2752e-07	2.9305
16641	1/128	6.2655e-09	3.9571	5.4817e-08	2.9633

Table 6.26. Advection-diffusion equation, Case 2: *UMR with* $p = 4$.

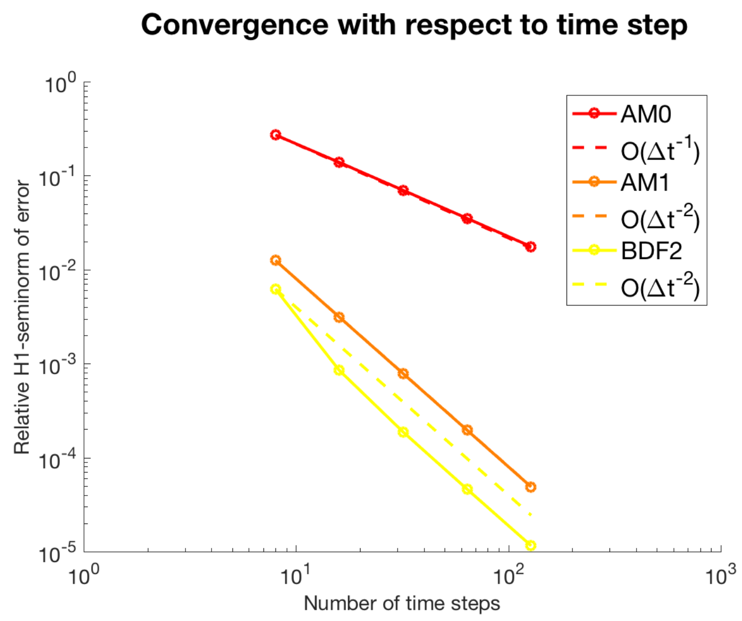
ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
100	1/8	4.3728e-05	-	2.2937e-05	-
324	1/16	1.9557e-06	4.4828	1.9370e-06	3.5657
1156	1/32	6.7421e-08	4.8584	1.3828e-07	3.8082
4356	1/64	2.1743e-09	4.9546	9.1596e-09	3.9162
16900	1/128	6.8755e-11	4.9829	5.8799e-10	3.9614

Table 6.27. Advection-diffusion equation, Case 2: *UMR with* $p = 5$.

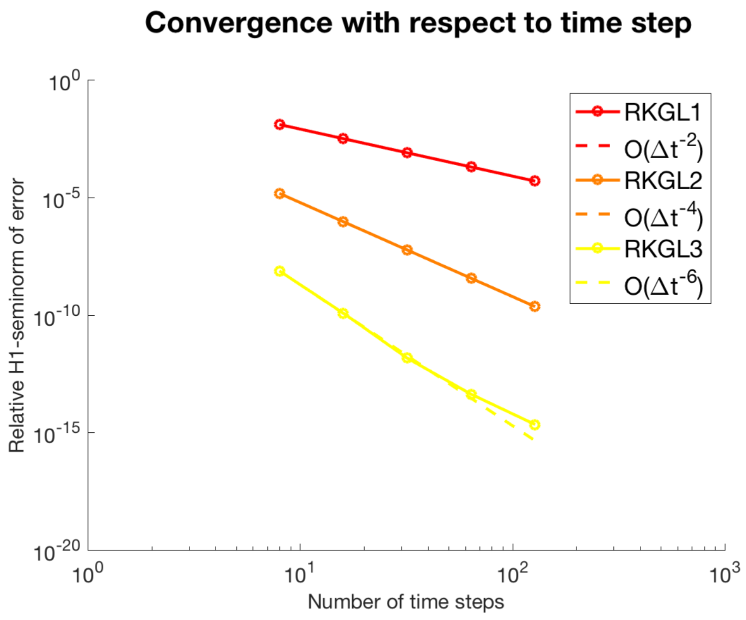
ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
121	1/8	9.4351e-06	-	4.0392e-06	-
361	1/16	1.6428e-07	5.8438	1.5980e-07	4.6597
1225	1/32	2.5962e-09	5.9836	5.3994e-09	4.8873
4489	1/64	4.0900e-11	5.9882	1.7445e-10	4.9519
17161	1/128	6.4308e-13	5.991	5.5388e-12	4.9771

Table 6.28. Advection-diffusion equation, Case 2: *UMR with* $p = 6$.

ndof	h	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(h)}}{u_{L^2(h/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(h)}}{u_{E(h/2)}}\right)$
144	1/8	1.2824e-06	-	6.0792e-07	-
400	1/16	9.4988e-09	7.0769	1.0284e-08	5.8854
1296	1/32	7.6718e-11	6.952	1.6830e-10	5.9332
4624	1/64	6.1974e-13	6.9517	2.7082e-12	5.9575
17424	1/128	1.5041e-14	5.3647	4.3047e-14	5.9753

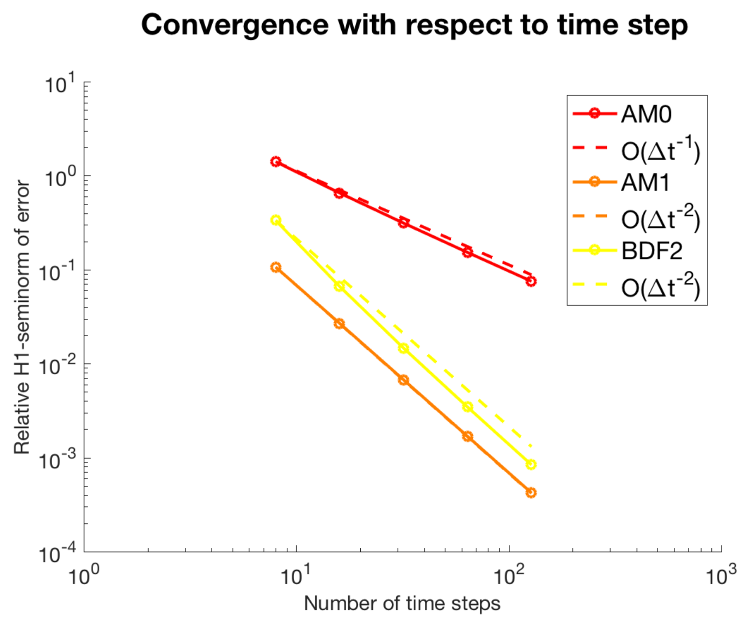


(a) *t-refinement with multistep integrators: Relative error (H^1-seminorm).*

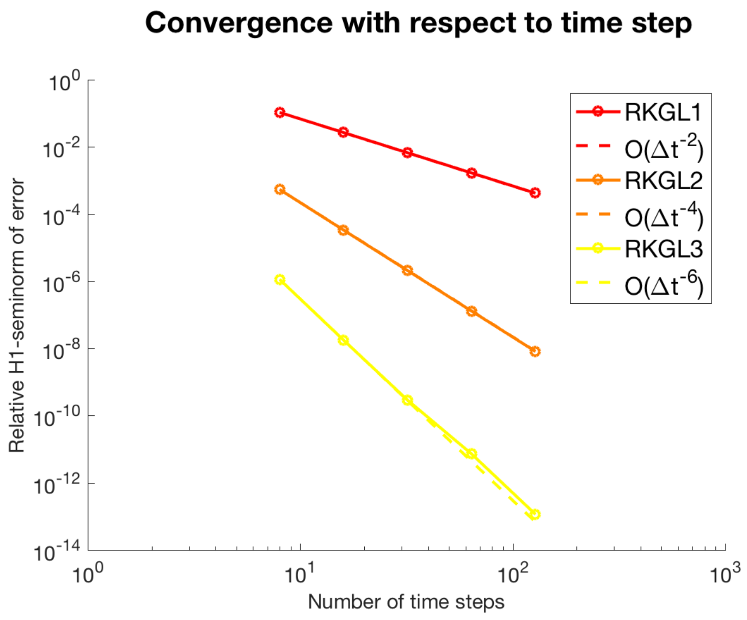


(b) *t-refinement with Runge-Kutta integrators: Relative error (H^1-seminorm).*

Figure 6.15. Advection-diffusion equation, Case 1: *t-refinement, relative error plots (%) in H^1-seminorm.*



(a) *t-refinement with multistep integrators: Relative error (H^1-seminorm).*



(b) *t-refinement with Runge-Kutta integrators: Relative error (H^1-seminorm).*

Figure 6.16. Advection-diffusion equation, Case 2: *t-refinement, relative error plots (%) in H^1-seminorm.*

Table 6.29. Advection-diffusion equation, Case 1: *t-refinement with AM0.*

Δt	$\frac{\Vert u-u_h \Vert_{L^2}}{\Vert u \Vert_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\Vert u-u_h \Vert_E}{\Vert u \Vert_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	2.7141e-01	-	2.7141e-01	-
1/16	1.3832e-01	0.97249	1.3832e-01	0.97249
1/32	6.9808e-02	0.98654	6.9808e-02	0.98654
1/64	3.5065e-02	0.99337	3.5065e-02	0.99337
1/128	1.7572e-02	0.99672	1.7572e-02	0.99672

Table 6.30. Advection-diffusion equation, Case 1: *t-refinement with AM1.*

Δt	$\frac{\Vert u-u_h \Vert_{L^2}}{\Vert u \Vert_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\Vert u-u_h \Vert_E}{\Vert u \Vert_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	1.2520e-02	-	1.2520e-02	-
1/16	3.1198e-03	2.0047	3.1198e-03	2.0047
1/32	7.7932e-04	2.0012	7.7932e-04	2.0012
1/64	1.9479e-04	2.0003	1.9479e-04	2.0003
1/128	4.8695e-05	2.0001	4.8695e-05	2.0001

Table 6.31. Advection-diffusion equation, Case 1: *t-refinement with BDF2.*

Δt	$\frac{\Vert u-u_h \Vert_{L^2}}{\Vert u \Vert_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\Vert u-u_h \Vert_E}{\Vert u \Vert_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	6.2155e-03	-	6.2155e-03	-
1/16	8.4523e-04	2.8785	8.4523e-04	2.8785
1/32	1.8645e-04	2.1805	1.8645e-04	2.1805
1/64	4.5908e-05	2.022	4.5908e-05	2.022
1/128	1.1510e-05	1.9959	1.1510e-05	1.9959

Table 6.32. Advection-diffusion equation, Case 1: *t-refinement with RKGL1.*

Δt	$\frac{\Vert u-u_h \Vert_{L^2}}{\Vert u \Vert_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\Vert u-u_h \Vert_E}{\Vert u \Vert_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	1.2520e-02	-	1.2520e-02	-
1/16	3.1198e-03	2.0047	3.1198e-03	2.0047
1/32	7.7932e-04	2.0012	7.7932e-04	2.0012
1/64	1.9479e-04	2.0003	1.9479e-04	2.0003
1/128	4.8695e-05	2.0001	4.8695e-05	2.0001

Table 6.33. Advection-diffusion equation, Case 1: *t-refinement with RKGL2.*

Δt	$\frac{\lVert u-u_h\rVert_{L^2}}{\lVert u\rVert_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\lVert u-u_h\rVert_E}{\lVert u\rVert_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	1.4698e-05	-	1.4698e-05	-
1/16	9.1574e-07	4.0045	9.1574e-07	4.0045
1/32	5.7189e-08	4.0011	5.7189e-08	4.0011
1/64	3.5736e-09	4.0003	3.5736e-09	4.0003
1/128	2.2334e-10	4.0001	2.2334e-10	4.0001

Table 6.34. Advection-diffusion equation, Case 1: *t-refinement with RKGL3.*

Δt	$\frac{\lVert u-u_h\rVert_{L^2}}{\lVert u\rVert_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\lVert u-u_h\rVert_E}{\lVert u\rVert_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	7.3853e-09	-	7.3855e-09	-
1/16	1.1516e-10	6.0029	1.1793e-10	5.9687
1/32	1.8035e-12	5.9967	1.4357e-12	6.3601
1/64	3.4604e-14	5.7037	4.1820e-14	5.1014
1/128	1.0489e-15	5.044	2.1594e-15	4.2755

Table 6.35. Advection-diffusion equation, Case 2: *t-refinement with AM0.*

Δt	$\frac{\lVert u-u_h\rVert_{L^2}}{\lVert u\rVert_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\lVert u-u_h\rVert_E}{\lVert u\rVert_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	1.4068e+00	-	1.4068e+00	-
1/16	6.5423e-01	1.1045	6.5423e-01	1.1045
1/32	3.1260e-01	1.0655	3.1260e-01	1.0655
1/64	1.5237e-01	1.0368	1.5237e-01	1.0368
1/128	7.5160e-02	1.0195	7.5160e-02	1.0195

Table 6.36. Advection-diffusion equation, Case 2: *t-refinement with AM1.*

Δt	$\frac{\lVert u-u_h\rVert_{L^2}}{\lVert u\rVert_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\lVert u-u_h\rVert_E}{\lVert u\rVert_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	1.0654e-01	-	1.0654e-01	-
1/16	2.6836e-02	1.9891	2.6836e-02	1.9891
1/32	6.7209e-03	1.9974	6.7209e-03	1.9974
1/64	1.6810e-03	1.9994	1.6810e-03	1.9994
1/128	4.2029e-04	1.9998	4.2029e-04	1.9998

Table 6.37. Advection-diffusion equation, Case 2: *t-refinement with BDF2.*

Δt	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	3.3431e-01	-	3.3431e-01	-
1/16	6.6298e-02	2.3341	6.6298e-02	2.3341
1/32	1.4616e-02	2.1814	1.4616e-02	2.1814
1/64	3.4439e-03	2.0855	3.4439e-03	2.0855
1/128	8.3639e-04	2.0418	8.3639e-04	2.0418

Table 6.38. Advection-diffusion equation, Case 2: *t-refinement with RKGL1.*

Δt	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	1.0654e-01	-	1.0654e-01	-
1/16	2.6836e-02	1.9891	2.6836e-02	1.9891
1/32	6.7209e-03	1.9974	6.7209e-03	1.9974
1/64	1.6810e-03	1.9994	1.6810e-03	1.9994
1/128	4.2029e-04	1.9998	4.2029e-04	1.9998

Table 6.39. Advection-diffusion equation, Case 2: *t-refinement with RKGL2.*

Δt	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	5.4112e-04	-	5.4112e-04	-
1/16	3.3372e-05	4.0192	3.3372e-05	4.0192
1/32	2.0788e-06	4.0048	2.0788e-06	4.0048
1/64	1.2982e-07	4.0012	1.2982e-07	4.0012
1/128	8.1120e-09	4.0003	8.1120e-09	4.0003

Table 6.40. Advection-diffusion equation, Case 2: *t-refinement with RKGL3.*

Δt	$\frac{\|u-u_h\|_{L^2}}{\|u\|_{L^2}}$	$\log_2\left(\frac{u_{L^2(\Delta t)}}{u_{L^2(\Delta t/2)}}\right)$	$\frac{\|u-u_h\|_E}{\|u\|_E}$	$\log_2\left(\frac{u_{E(\Delta t)}}{u_{E(\Delta t/2)}}\right)$
1/8	1.1386e-06	-	1.1386e-06	-
1/16	1.7638e-08	6.0124	1.7639e-08	6.0124
1/32	2.7501e-10	6.0031	2.8855e-10	5.9338
1/64	4.2980e-12	5.9996	7.2261e-12	5.3194
1/128	6.9388e-14	5.9528	1.1451e-13	5.9796

6.5 Unsteady Navier-Stokes equations

6.5.1 Manufactured reference solution

Throughout this section, we will consider the scaled and unsteady Navier-Stokes equations with force. The domain Ω is a unit square, and we still assume homogeneous Dirichlet conditions on the velocity field.

$$\frac{\partial \mathbf{u}}{\partial t} - \frac{1}{Re}\nabla^2 \mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f} \tag{6.13a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{6.13b}$$

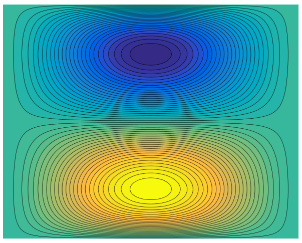
This time, we define f and g as polynomials of the same degree, and they are constructed such that the velocity field becomes a priori solenoidal. The velocity field and pressure are multiplied with the same amplitude function h, and these constructions result in some tedious expressions for the force. We have the following class of reference solutions:

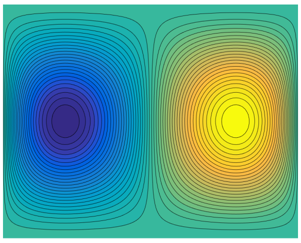
$$\boxed{\begin{aligned}
u_x(x,y) &= f(x)g'(y)h(t) && (6.14a)\\
u_y(x,y) &= -f'(x)g(y)h(t) && (6.14b)\\
p(x,y) &= \frac{1}{Re}f'(x)g'(y)h(t) && (6.14c)\\
f_x(x,y) &= f(x)g'(y)h'(t) - \frac{1}{Re}f(x)g'''(y)h(t) && (6.14d)\\
&\quad + \left[g'(y)^2 - g(y)g''(y)\right] f(x)f'(x)h(t)^2 \\
f_y(x,y) &= -f'(x)g(y)h'(t) + \frac{1}{Re}\left[2f'(x)g''(y) + f'''(x)g(y)\right]h(t) && (6.14e)\\
&\quad + \left[f'(x)^2 - f(x)f''(x)\right] g(y)g'(y)h(t)^2
\end{aligned}}$$

This manufactured solution allows us to construct four test functions from

Case 1.1:	$f(x) = 4(x-x^2)^2$	$g(y) = 4(y-y^2)^2$	$h(t) = t$
Case 1.2:	$f(x) = 8(x-x^2)^3$	$g(y) = 8(y-y^2)^3$	$h(t) = t$
Case 2.1:	$f(x) = 4(x-x^2)^2$	$g(y) = 4(y-y^2)^2$	$h(t) = 1+t^3$
Case 2.2:	$f(x) = 8(x-x^2)^3$	$g(y) = 8(y-y^2)^3$	$h(t) = 1+t^3$

The main advantage of this construction is that we can reach machine precision very fast, and we can also define p and Δt in such a way that temporal error is totally excluded from the h- and p-refinements. Similarly, we can choose p such that spatial error is not influent on the temporal error, and thus, we do not need so many elements on the domain. The h- and p-refinements are performed with BDF1, and $Re = 1$ such that the problem is always diffusion-dominated and becomes easy to solve.


(a) *Velocity, x-component.*


(b) *Velocity, y-component.*

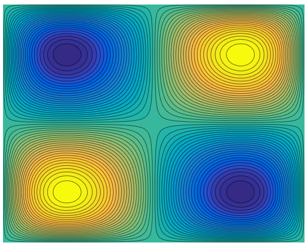
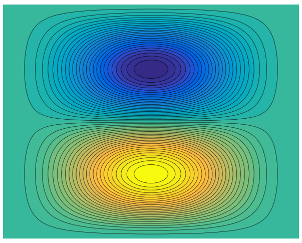
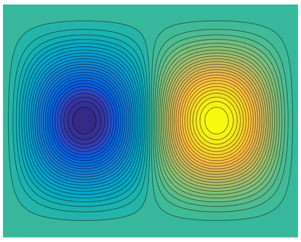

(c) *Pressure.*

Figure 6.17. Navier-Stokes equations, Case 1.1: *Contour plot of the manufactured reference solutions' initial value.*


(a) *Velocity, x-component.*


(b) *Velocity, y-component.*

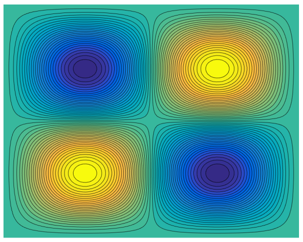

(c) *Pressure.*

Figure 6.18. Navier-Stokes equations, Case 1.2: *Contour plot of the manufactured reference solutions' initial value.*

6.5.2 Discussion on the results

In the simulations with the Navier-Stokes equations, the time interval was still $[0, 1]$ as before, but we only used $\Delta t = 0.05$ in the h- and p-refinements. This was because we had $h(t) = t$ and used the BDF1 integrator in these refinements. The time function was integrated exactly and the temporal error vanished, so we did not need to make Δt very small. As we see from figure 6.19 and 6.20, the numerical convergence graphs decayed gradually in accordance with the theoretical assumptions. When $p_p = 3$ and $p_u = 4$, the graphs dropped down immediately to machine precision. This is because the analytical solution was represented exactly as a finite linear sum of B-splines with the same degree. This expected behaviour was also observed in figure 6.21 and 6.22, and this shows that the h- and p-refinements are correct.

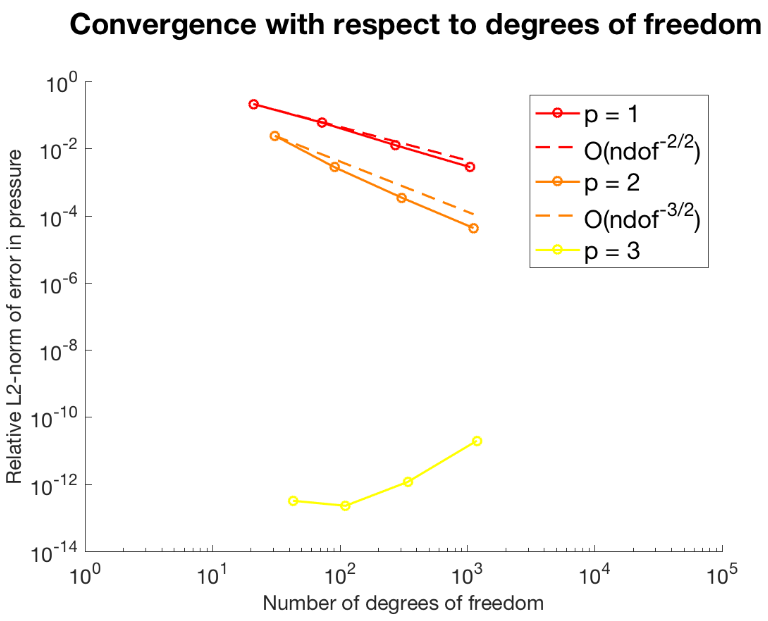
In the t-refinements, we excluded the spatial error by letting the polynomial degree be the same as the analytical solution. These simulations went quick because we let $h = 1/16$ all the time. As we see from figure 6.23 and 6.24, all the numerical convergence graphs coincided with the analytical convergence graphs, so the simulation has worked.

Since the Navier-Stokes equations are nonlinear, Newton-iteration had to be invoked. The error in the approximation from this technique must have been very small because the iteration was subject to two predefined restrictions:

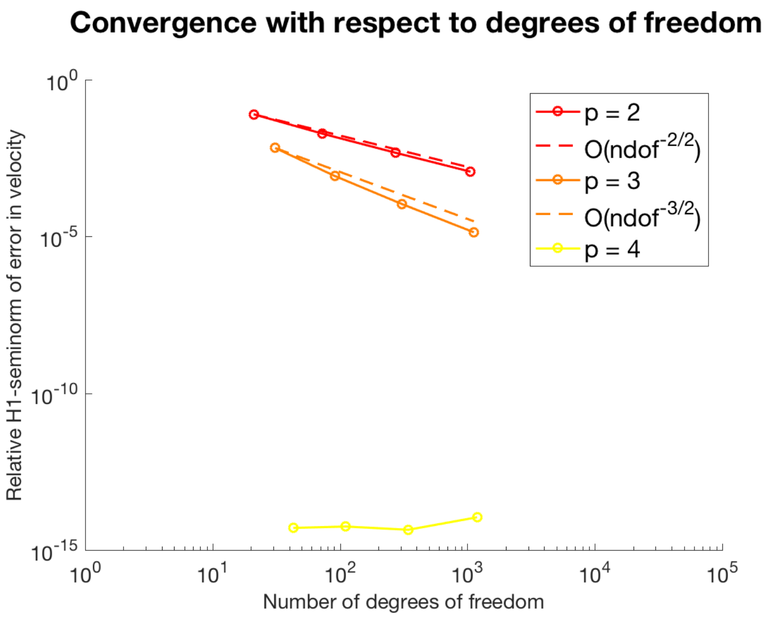
$$\begin{aligned} &\text{tolerance} \leq 10^{-12} \\ &\text{\# iterations} \leq 6 \end{aligned}$$

Empirical observations showed that the tolerance level was reached after 3 or 4 iterations, and since this value is so small, it is not probable that error from the Newton iteration has influenced the refinements much. It was also expected that this iteration would go very fast because of its quadratic convergence.

The described strategy of reducing the global computational effort will also be used when we are finally going to solve the Boussinesq equations, which are more complex than the Navier-Stokes equations.

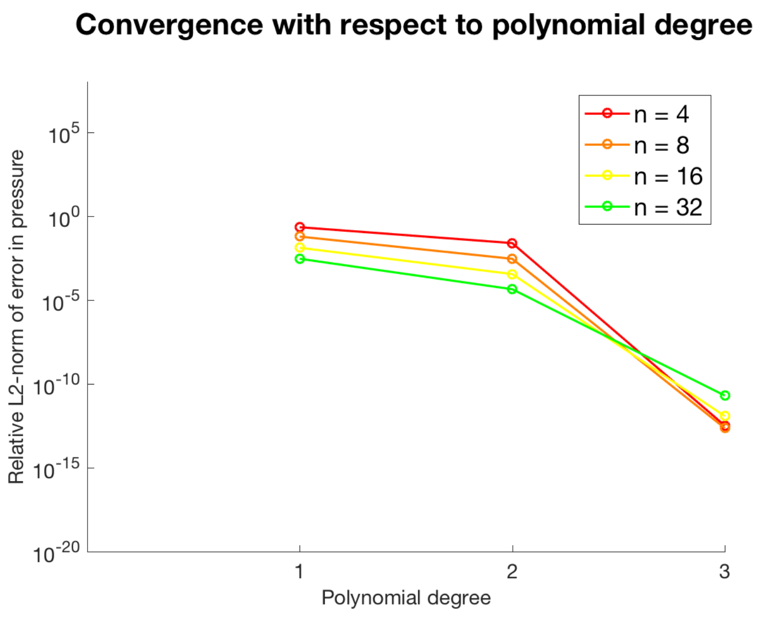


(a) *h-refinement: Relative error in the pressure (L^2-norm).*

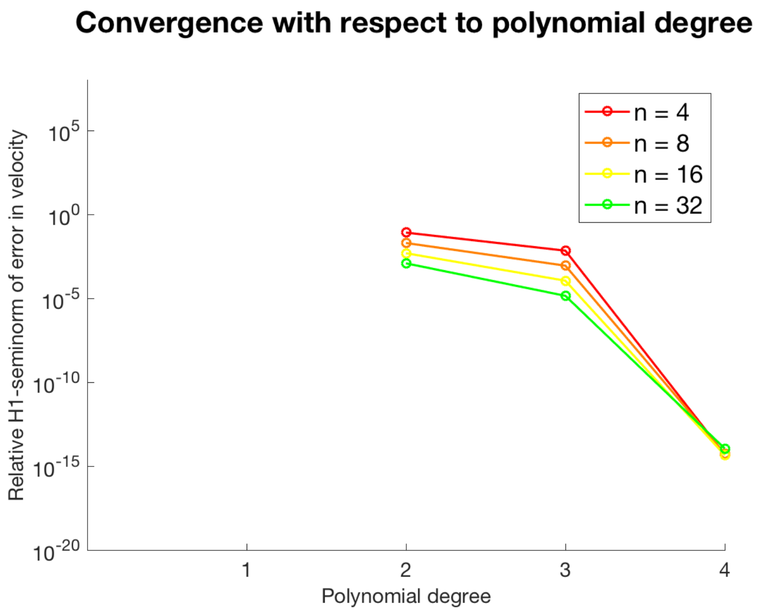


(b) *h-refinement: Relative error in the velocity field (H^1-seminorm).*

Figure 6.19. Navier-Stokes equations, Case 1.1: *h-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*

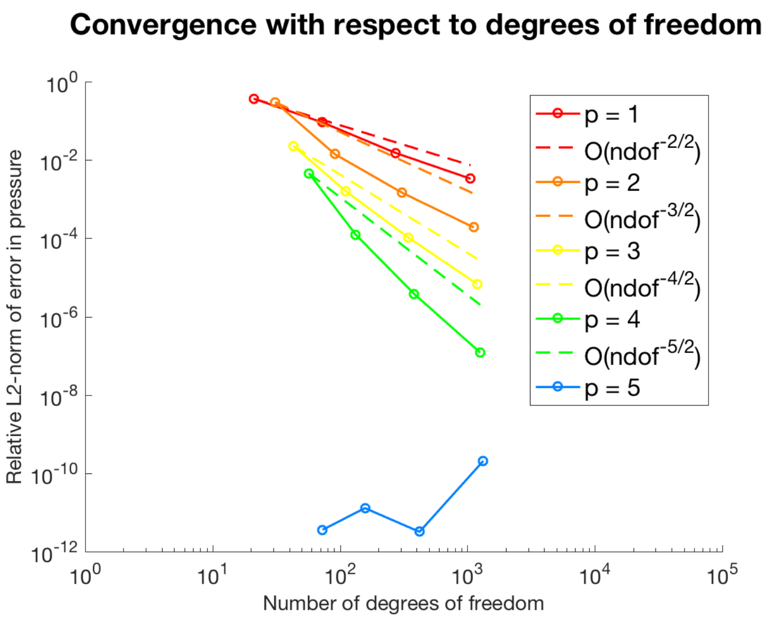


(a) *p-refinement: Relative error in the pressure (L^2-norm).*

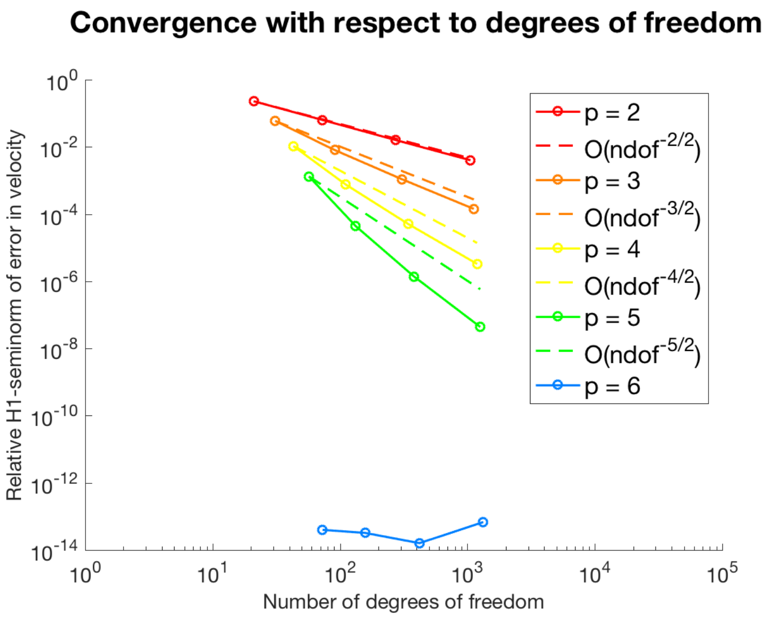


(b) *p-refinement: Relative error in the velocity field (H^1-seminorm).*

Figure 6.20. Navier-Stokes equations, Case 1.1: *p-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*

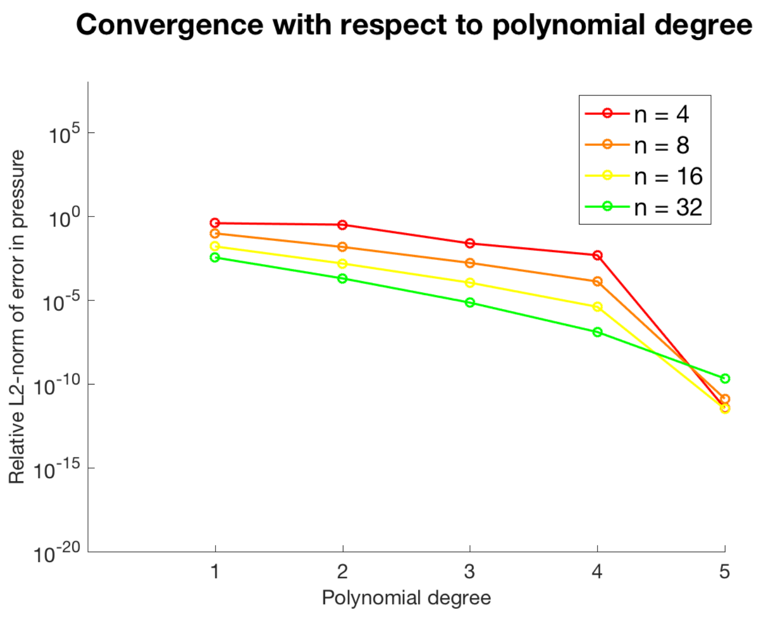


(a) *h-refinement: Relative error in the pressure (L^2-norm).*

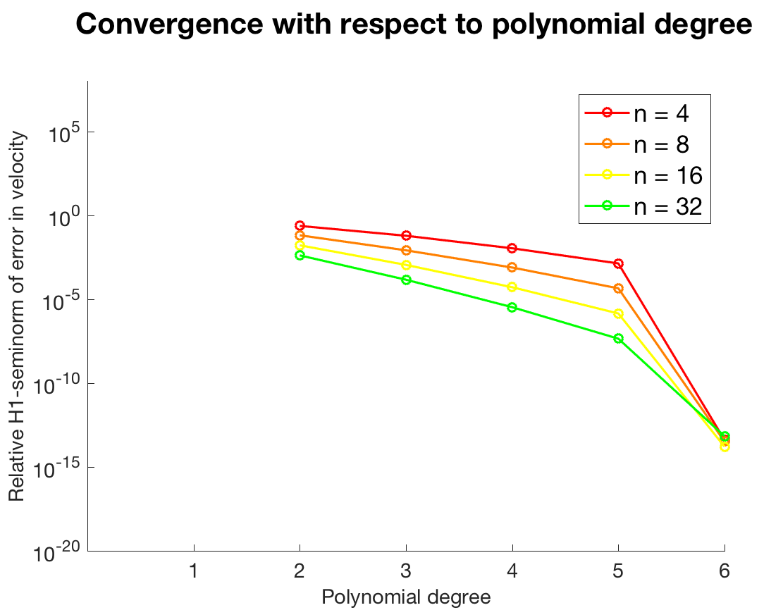


(b) *h-refinement: Relative error in the velocity field (H^1-seminorm).*

Figure 6.21. Navier-Stokes equations, Case 1.2: *h-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*



(a) *p-refinement: Relative error in the pressure (L^2-norm).*



(b) *p-refinement: Relative error in the velocity field (H^1-seminorm).*

Figure 6.22. Navier-Stokes equations, Case 1.2: *p-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*

Table 6.41. Navier-Stokes equations, Case 1.1: *UMR with $p_u = 2$ and $p_p = 1$.*

ndof	h	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_{L^2}}{\|\mathbf{u}\|_{L^2}}$	$\log_2\left(\frac{\mathbf{u}_{L^2(h)}}{\mathbf{u}_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
170	1/4	2.1227e-01	-	1.0966e-01	-	7.9135e-02	-
626	1/8	5.8763e-02	1.8529	1.4820e-02	2.8874	1.9028e-02	2.0562
2402	1/16	1.2614e-02	2.2198	1.9356e-03	2.9367	4.6934e-03	2.0194
9410	1/32	2.7914e-03	2.176	2.4760e-04	2.9667	1.1688e-03	2.0056

Table 6.42. Navier-Stokes equations, Case 1.1: *UMR with $p_u = 3$ and $p_p = 2$.*

ndof	h	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_{L^2}}{\|\mathbf{u}\|_{L^2}}$	$\log_2\left(\frac{\mathbf{u}_{L^2(h)}}{\mathbf{u}_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
215	1/4	2.4073e-02	-	1.0205e-02	-	6.6533e-03	-
711	1/8	2.7836e-03	3.1124	6.9124e-04	3.884	8.5179e-04	2.9655
2567	1/16	3.4313e-04	3.0201	4.4472e-05	3.9582	1.0804e-04	2.9789
9735	1/32	4.2856e-05	3.0012	2.8177e-06	3.9803	1.3633e-05	2.9864

Table 6.43. Navier-Stokes equations, Case 1.1: *UMR with $p_u = 4$ and $p_p = 3$.*

ndof	h	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_{L^2}}{\|\mathbf{u}\|_{L^2}}$	$\log_2\left(\frac{\mathbf{u}_{L^2(h)}}{\mathbf{u}_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
266	1/4	3.1919e-13	-	6.7961e-15	-	5.0355e-15	-
802	1/8	2.2632e-13	0.49604	3.6291e-15	0.90511	5.5837e-15	-0.14907
2738	1/16	1.1709e-12	-2.3711	4.6608e-15	-0.36098	4.3986e-15	0.34416
10066	1/32	1.9604e-11	-4.0655	1.5059e-14	-1.692	1.0986e-14	-1.3206

Table 6.44. Navier-Stokes equations, Case 1.2: *UMR with $p_u = 2$ and $p_p = 1$.*

ndof	h	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_{L^2}}{\|\mathbf{u}\|_{L^2}}$	$\log_2\left(\frac{\mathbf{u}_{L^2(h)}}{\mathbf{u}_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
170	1/4	3.6914e-01	-	3.9285e-01	-	2.2976e-01	-
626	1/8	9.1049e-02	2.0195	6.2189e-02	2.6592	6.3422e-02	1.8571
2402	1/16	1.5171e-02	2.5853	8.8246e-03	2.8171	1.6058e-02	1.9817
9410	1/32	3.3267e-03	2.1891	1.1805e-03	2.9022	4.0197e-03	1.9981

Table 6.45. Navier-Stokes equations, Case 1.2: *UMR with* $p_u = 3$ *and* $p_p = 2$.

ndof	h	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_{L^2}}{\|\mathbf{u}\|_{L^2}}$	$\log_2\left(\frac{\mathbf{u}_{L^2(h)}}{\mathbf{u}_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
215	1/4	3.0135e-01	-	9.8461e-02	-	5.9489e-02	-
711	1/8	1.4155e-02	4.4121	7.2767e-03	3.7582	8.0594e-03	2.8839
2567	1/16	1.4609e-03	3.2764	5.2996e-04	3.7793	1.0895e-03	2.887
9735	1/32	1.8949e-04	2.9466	3.5957e-05	3.8815	1.4272e-04	2.9324

Table 6.46. Navier-Stokes equations, Case 1.2: *UMR with* $p_u = 4$ *and* $p_p = 3$.

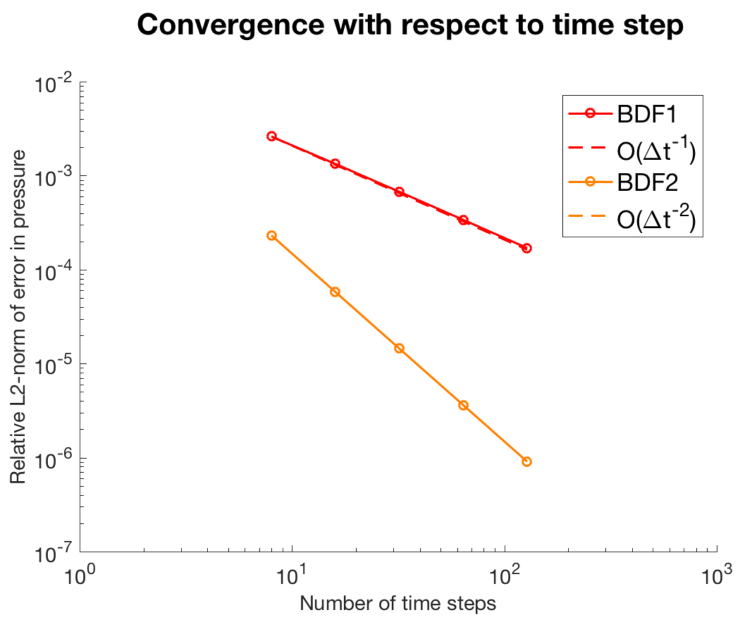
ndof	h	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_{L^2}}{\|\mathbf{u}\|_{L^2}}$	$\log_2\left(\frac{\mathbf{u}_{L^2(h)}}{\mathbf{u}_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
266	1/4	2.3033e-02	-	1.9355e-02	-	1.0651e-02	-
802	1/8	1.5922e-03	3.8546	6.9366e-04	4.8023	7.5920e-04	3.8103
2738	1/16	1.0518e-04	3.9201	2.3713e-05	4.8705	5.0797e-05	3.9017
10066	1/32	6.8078e-06	3.9495	7.7967e-07	4.9267	3.2857e-06	3.9505

Table 6.47. Navier-Stokes equations, Case 1.2: *UMR with* $p_u = 5$ *and* $p_p = 4$.

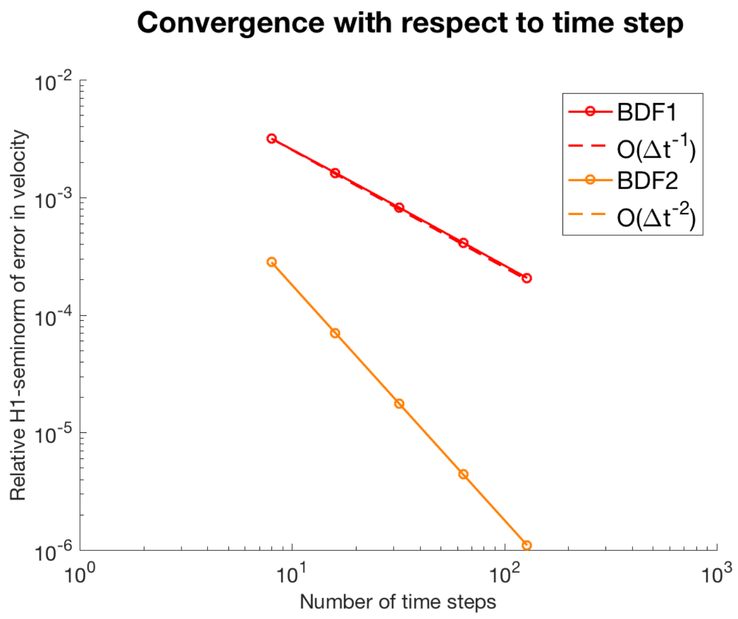
ndof	h	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_{L^2}}{\|\mathbf{u}\|_{L^2}}$	$\log_2\left(\frac{\mathbf{u}_{L^2(h)}}{\mathbf{u}_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
323	1/4	4.5875e-03	-	2.1766e-03	-	1.3323e-03	-
899	1/8	1.2397e-04	5.2097	3.8233e-05	5.8311	4.3789e-05	4.9272
2915	1/16	3.8282e-06	5.0172	6.2465e-07	5.9356	1.3859e-06	4.9817
10403	1/32	1.2049e-07	4.9897	9.9562e-09	5.9713	4.3624e-08	4.9895

Table 6.48. Navier-Stokes equations, Case 1.2: *UMR with* $p_u = 6$ *and* $p_p = 5$.

ndof	h	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_{L^2}}{\|\mathbf{u}\|_{L^2}}$	$\log_2\left(\frac{\mathbf{u}_{L^2(h)}}{\mathbf{u}_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
386	1/4	3.5898e-12	-	4.4562e-14	-	3.9615e-14	-
1002	1/8	1.2813e-11	-1.8357	2.4020e-14	0.89158	3.2152e-14	0.30111
3098	1/16	3.1793e-12	2.0108	1.7998e-14	0.41642	1.5923e-14	1.0138
10746	1/32	2.0584e-10	-6.0167	9.7242e-15	0.88816	6.7446e-14	-2.0826

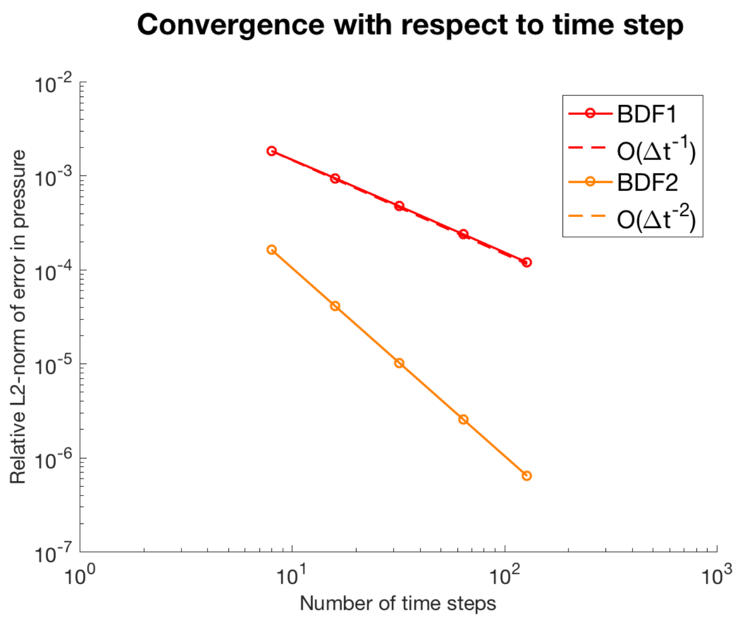


(a) *t-refinement: Relative error in the pressure (L^2-norm).*

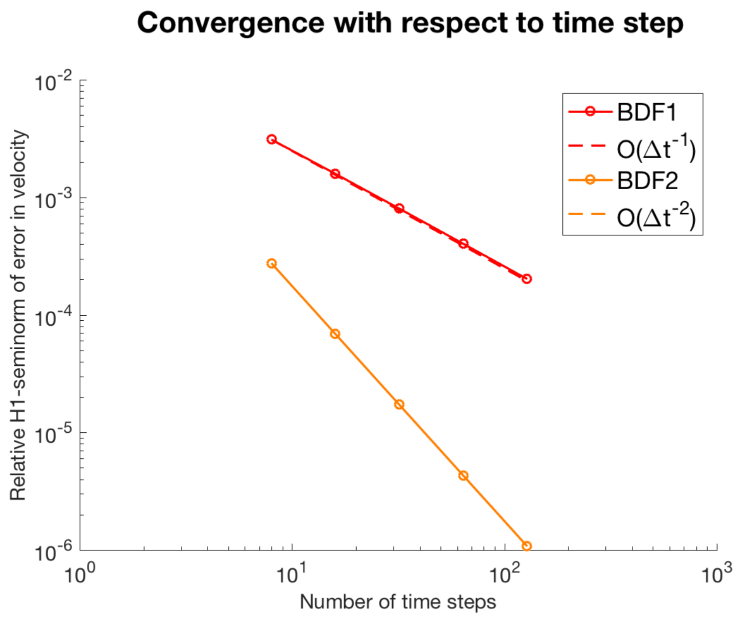


(b) *t-refinement: Relative error in the velocity field (H^1-seminorm).*

Figure 6.23. Navier-Stokes equations, Case 2.1: *t-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*



(a) *t-refinement: Relative error in the pressure (L^2-norm).*



(b) *t-refinement: Relative error in the velocity field (H^1-seminorm).*

Figure 6.24. Navier-Stokes equations, Case 2.2: *t-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*

Table 6.49. **Navier-Stokes equations, Case 2.1:** *t-refinement with BDF1.*

Δt	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(\Delta t)}}{p_{L^2(\Delta t/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(\Delta t)}}{\mathbf{u}_{E(\Delta t/2)}}\right)$
1/8	2.6095e-03	-	3.1617e-03	-
1/16	1.3336e-03	0.96844	1.6159e-03	0.96835
1/32	6.7402e-04	0.98448	8.1672e-04	0.98443
1/64	3.3881e-04	0.9923	4.1055e-04	0.99228
1/128	1.6986e-04	0.99617	2.0582e-04	0.99615

Table 6.50. **Navier-Stokes equations, Case 2.1:** *t-refinement with BDF2.*

Δt	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(\Delta t)}}{p_{L^2(\Delta t/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(\Delta t)}}{\mathbf{u}_{E(\Delta t/2)}}\right)$
1/8	2.3143e-04	-	2.8051e-04	-
1/16	5.7856e-05	2	7.0127e-05	2
1/32	1.4464e-05	2	1.7532e-05	2
1/64	3.6160e-06	2	4.3829e-06	2
1/128	9.0400e-07	2	1.0957e-06	2

Table 6.51. **Navier-Stokes equations, Case 2.2:** *t-refinement with BDF1.*

Δt	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(\Delta t)}}{p_{L^2(\Delta t/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(\Delta t)}}{\mathbf{u}_{E(\Delta t/2)}}\right)$
1/8	1.8327e-03	-	3.1010e-03	-
1/16	9.3670e-04	0.96828	1.5849e-03	0.96836
1/32	4.7344e-04	0.9844	8.0105e-04	0.98443
1/64	2.3799e-04	0.99226	4.0267e-04	0.99228
1/128	1.1932e-04	0.99615	2.0187e-04	0.99616

Table 6.52. **Navier-Stokes equations, Case 2.2:** *t-refinement with BDF2.*

Δt	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(\Delta t)}}{p_{L^2(\Delta t/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(\Delta t)}}{\mathbf{u}_{E(\Delta t/2)}}\right)$
1/8	1.6299e-04	-	2.7509e-04	-
1/16	4.0748e-05	2	6.8772e-05	2
1/32	1.0187e-05	2	1.7193e-05	2
1/64	2.5468e-06	2	4.2982e-06	2
1/128	6.3669e-07	2	1.0746e-06	2

6.6 Boussinesq equations

6.6.1 Manufactured reference solution

Finally, we can focus on the scaled Boussinesq equations with force. The domain Ω is a unit square, and we also assume homogeneous Dirichlet conditions on the temperature.

$$\frac{\partial \mathbf{u}}{\partial t} - \frac{1}{Re}\nabla^2\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f} - Ri\frac{\mathbf{g}}{\|\mathbf{g}\|}T \tag{6.15a}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{6.15b}$$

$$\frac{\partial T}{\partial t} - \frac{1}{Pe}\nabla^2 T + (\mathbf{u}\cdot\nabla)T = Q \tag{6.15c}$$

We will use the exactly same procedure for creating manufactured reference solutions as for the Navier-Stokes equations. The new trick is to define $f(x)g(y)$ as the temperature's amplitude function. By doing so, the linear convection in the source term vanishes, and source itself gets a simple structure. The force component f_y must be extended because of the extra temperature term. Hence, we obtain the following class of reference solutions:

$$u_x(x,y) = f(x)g'(y)h(t) \tag{6.16a}$$

$$u_y(x,y) = -f'(x)g(y)h(t) \tag{6.16b}$$

$$p(x,y) = \frac{1}{Re}f'(x)g'(y)h(t) \tag{6.16c}$$

$$T(x,y) = f(x)g(y)h(t) \tag{6.16d}$$

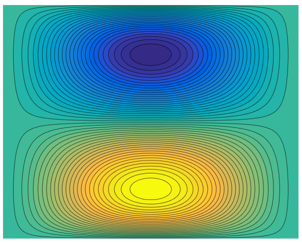
$$\begin{aligned} f_x(x,y) = {} & f(x)g'(y)h'(t) - \frac{1}{Re}f(x)g'''(y)h(t) \\ & + \left[g'(y)^2 - g(y)g''(y)\right]f(x)f'(x)h(t)^2 \end{aligned} \tag{6.16e}$$

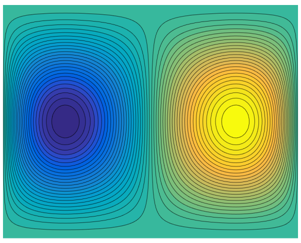
$$\begin{aligned} f_y(x,y) = {} & -f'(x)g(y)h'(t) + \frac{1}{Re}\left[2f'(x)g''(y) + f'''(x)g(y)\right]h(t) \\ & + \left[f'(x)^2 - f(x)f''(x)\right]g(y)g'(y)h(t)^2 + Ri f(x)g(y)h(t) \end{aligned} \tag{6.16f}$$

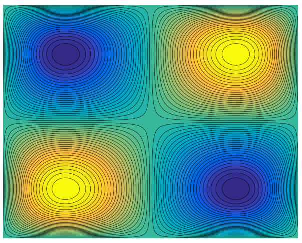
$$Q(x,y) = f(x)g(y)h'(t) - \frac{1}{Pe}(f''(x)g(y) + f(x)g''(y))h(t) \tag{6.16g}$$

As before, we can still construct four test functions from

Case 1.1:	$f(x) = 4(x-x^2)^2$	$g(y) = 4(y-y^2)^2$	$h(t) = t$
Case 1.2:	$f(x) = 8(x-x^2)^3$	$g(y) = 8(y-y^2)^3$	$h(t) = t$
Case 2.1:	$f(x) = 4(x-x^2)^2$	$g(y) = 4(y-y^2)^2$	$h(t) = 1+t^3$
Case 2.2:	$f(x) = 8(x-x^2)^3$	$g(y) = 8(y-y^2)^3$	$h(t) = 1+t^3$


(a) *Velocity, x-component.*


(b) *Velocity, y-component.*


(c) *Pressure.*

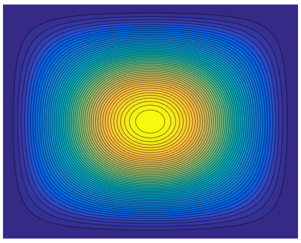
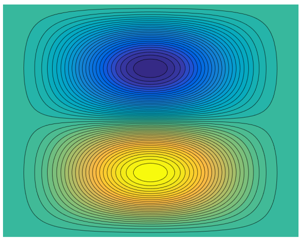
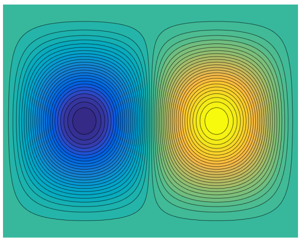
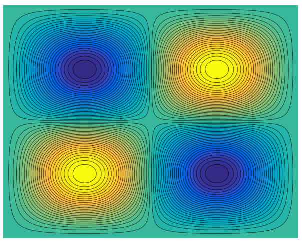

(d) *Temperature.*

Figure 6.25. Boussinesq equations, Case 1.1: *Contour plot of the manufactured reference solutions' initial value.*


(a) *Velocity, x-component.*


(b) *Velocity, y-component.*


(c) *Pressure.*

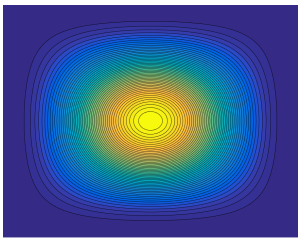

(d) *Temperature.*

Figure 6.26. Boussinesq equationa, Case 1.2: *Contour plot of the manufactured reference solutions' initial value.*

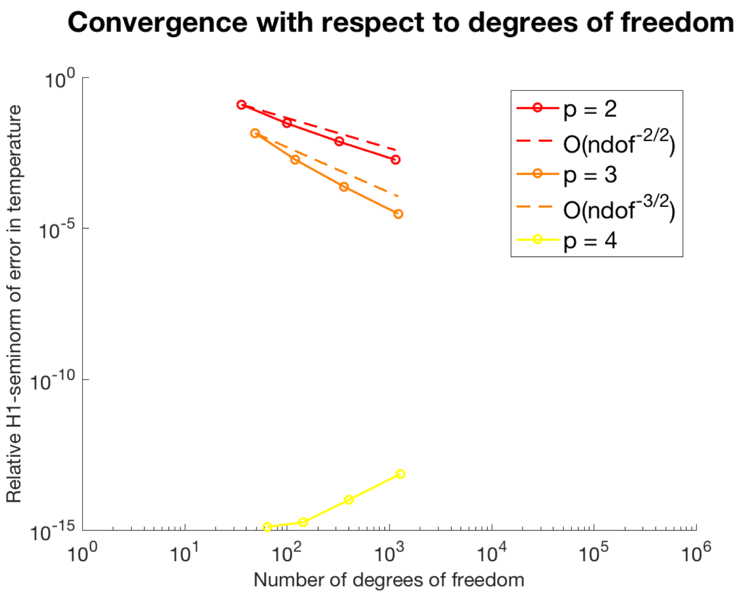
6.6.2 Discussion on the results

In the simulations with the Boussinesq equations, we have used the same procedure for choosing p, h and Δt as for the Navier-Stokes equations. To be as simple as possible, we assume that $Re, Ri, Pe = 1$ such there are no boundary layers in the solution. This time, we are using two different integrators to segregate the whole system, and the second one has an explicit part. It seems that $\Delta t = 0.05$ was a suitable time restriction for the SBDF integrator used in the h- and p-refinements, for the numerical convergence graphs have behaved in exactly the same way as for the Navier-Stokes equations.

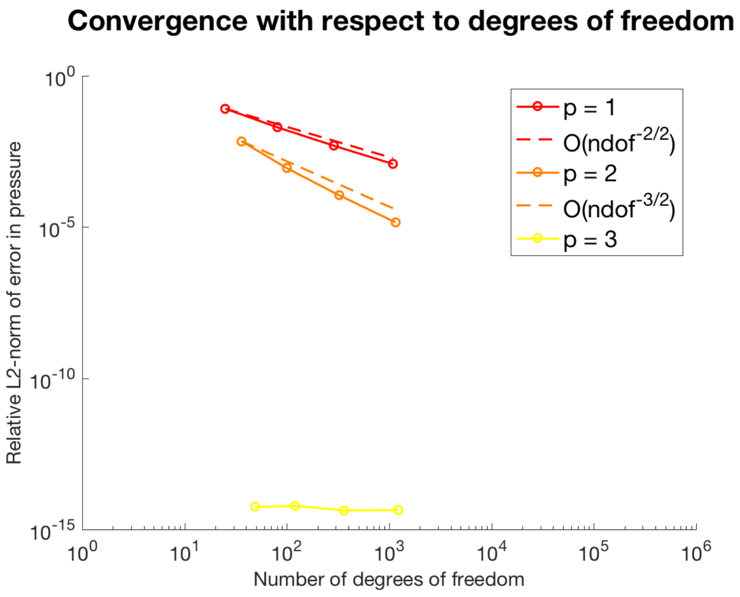
We see from figure 6.27, 6.28, 6.29 and 6.30 that all the numerical convergence graphs decay as expected from the theoretical assumptions. This behaviour is clearly observed in the temperature, pressure and velocity field. Hence, we can conclude that the h- and p-refinements have worked as they should.

In the t-refinement, the numerical convergence graphs for the pressure and velocity field coincided with the analytical graphs, but the temperature converged faster than expected. This behaviour has been observed both in figure 6.31 and 6.32. The reason for this is not clear, and it might happen that this was just incidental for our particular problem.

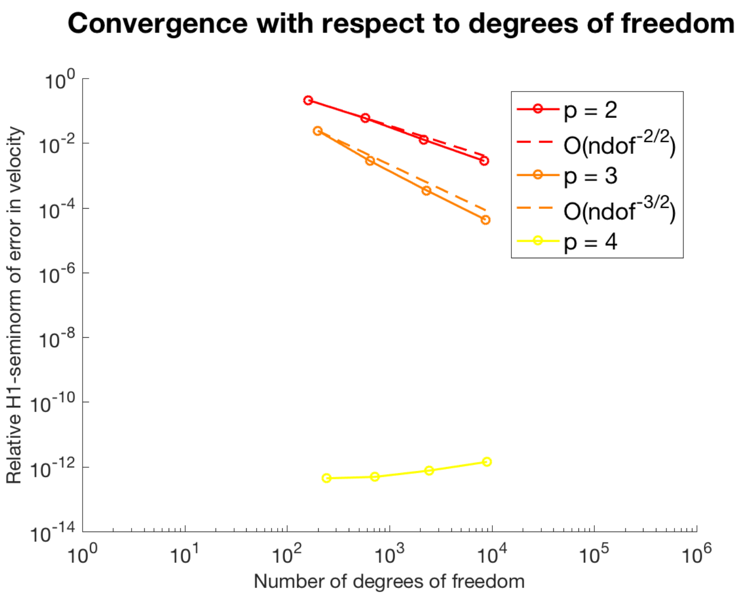
Anyway, the method of segregating the Boussinesq system by using two time-integrators instead of just one has given suitable solutions and ensured that the total running time is sufficiently limited. Using Newton iteration on the whole coupled system would have taken much more effort to solve. Therefore, we can finally conclude that our approach has worked as it should.



(a) *h-refinement: Relative error in the temperature (H^1-seminorm).*

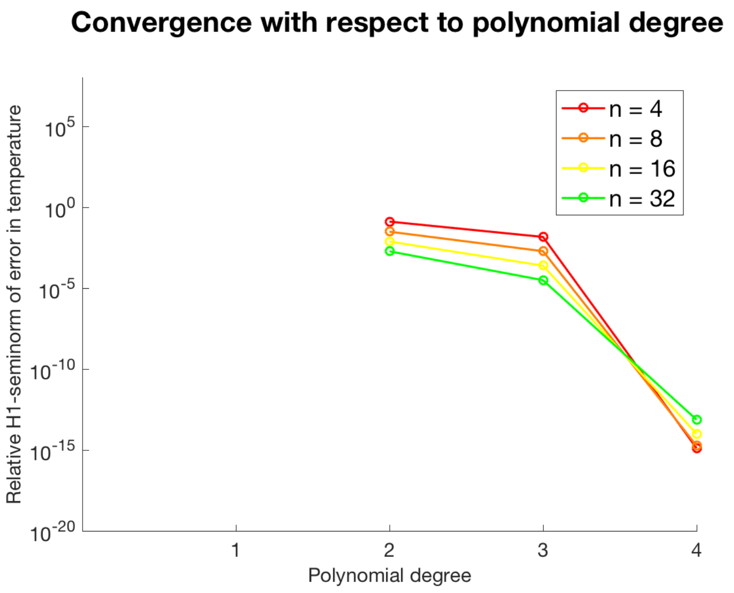


(b) *h-refinement: Relative error in the pressure (L^2-norm).*

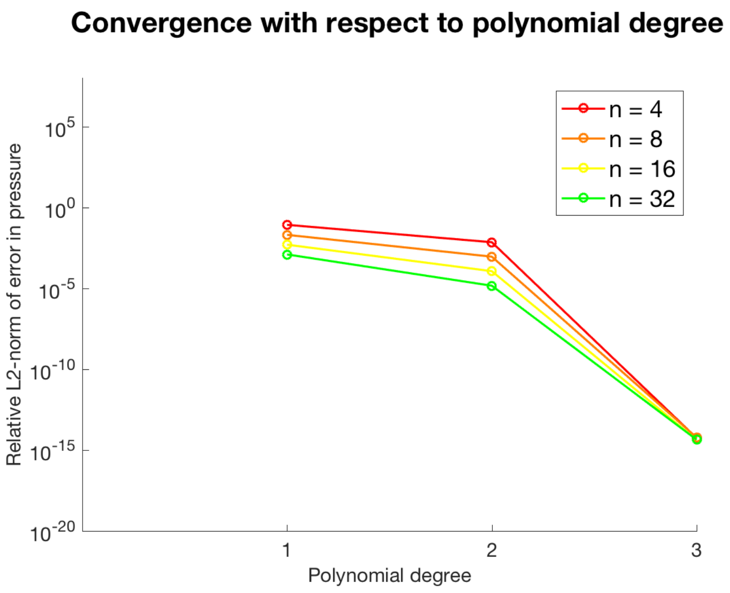


(c) *h-refinement: Relative error in the velocity field (H^1-seminorm).*

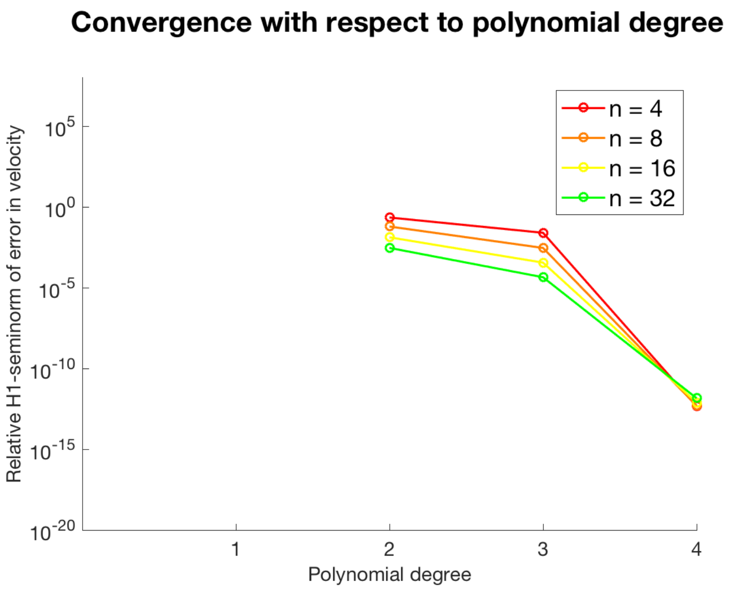
Figure 6.27. **Boussinesq equations, Case 1.1:** *h-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*



(a) *p-refinement: Relative error in the temperature (H^1-seminorm).*

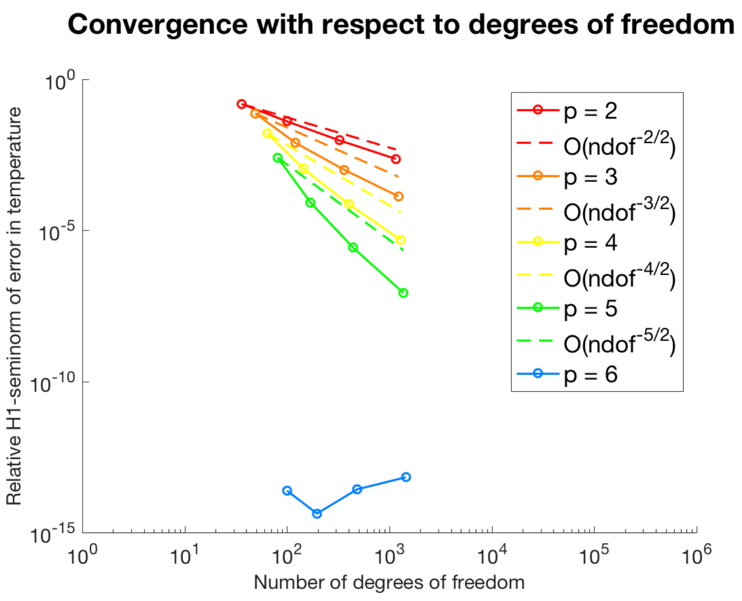


(b) *p-refinement: Relative error in the pressure (L^2-norm).*

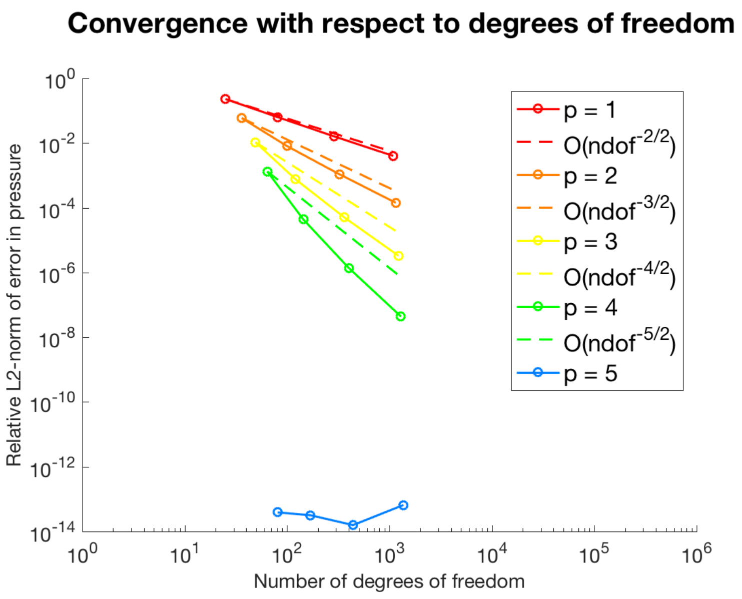


(c) *p-refinement: Relative error in the velocity field (H^1-seminorm).*

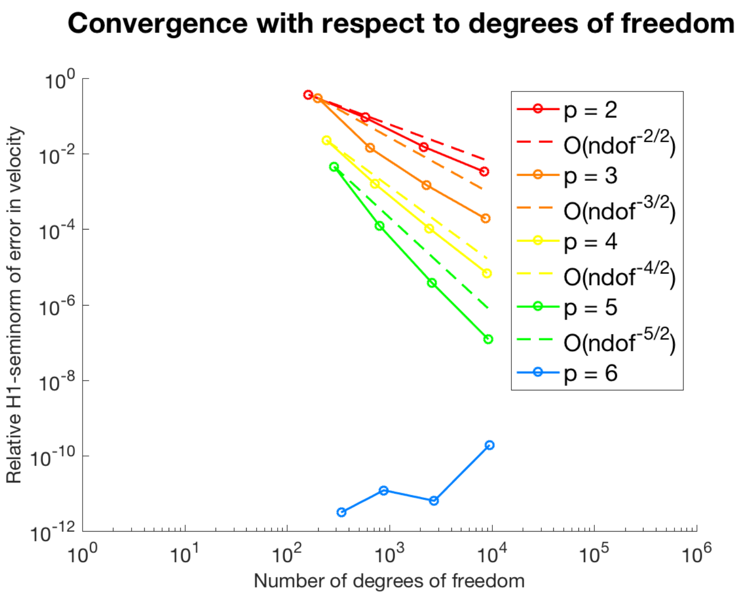
Figure 6.28. Boussinesq equations, Case 1.1: *p-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*



(a) *h-refinement: Relative error in the temperature (H^1-seminorm).*

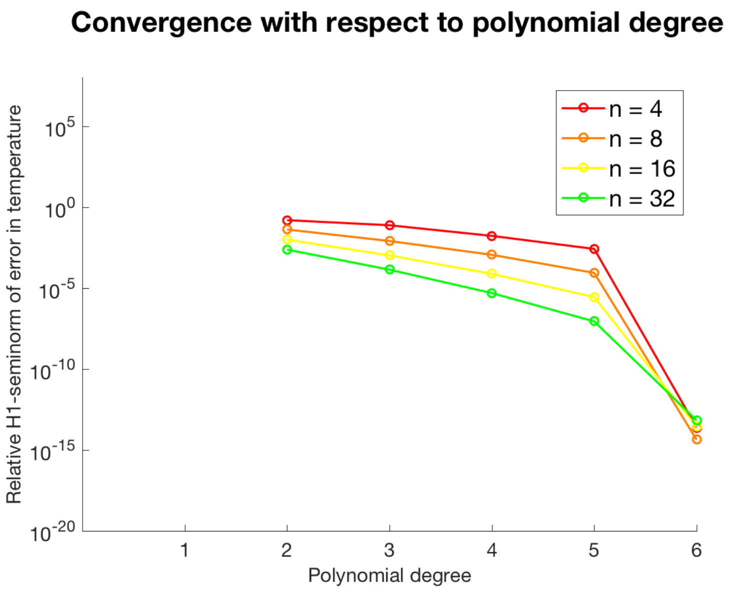


(b) *h-refinement: Relative error in the pressure (L^2-norm).*

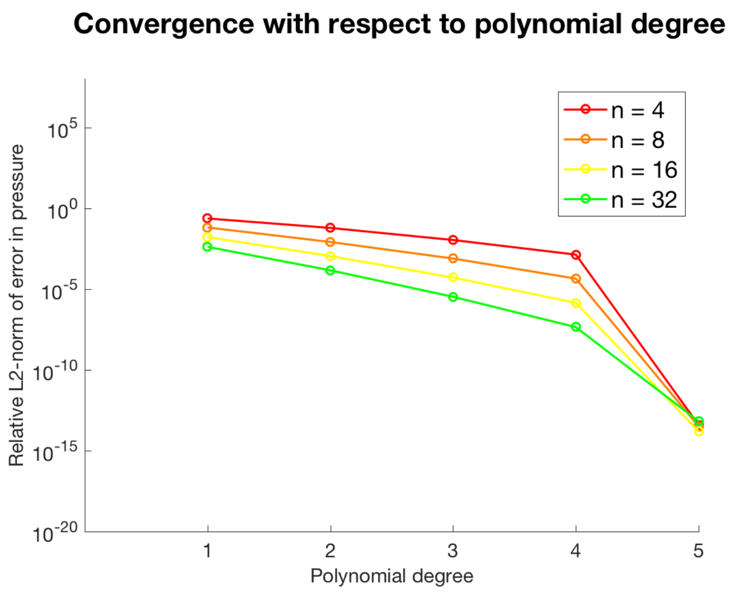


(c) *h-refinement: Relative error in the velocity field (H^1-seminorm).*

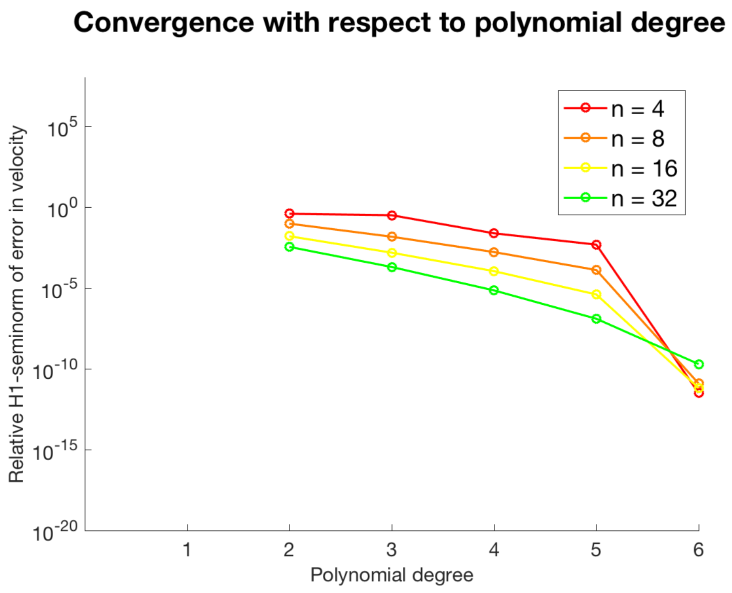
Figure 6.29. Boussinesq equations, Case 1.2: *h-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*



(a) *p-refinement: Relative error in the temperature (H^1-seminorm).*



(b) *p-refinement: Relative error in the pressure (L^2-norm).*



(c) *p-refinement: Relative error in the velocity field (H^1-seminorm).*

Figure 6.30. Boussinesq equations, Case 1.2: *p-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*

Table 6.53. **Boussinesq equations, Case 1.1:** *UMR with* $p_t = 2$, $p_p = 1$ *and* $p_u = 2$.

ndof	h	$\frac{\|T-T_h\|_E}{\|T\|_E}$	$\log_2\left(\frac{T_{E(h)}}{T_{E(h/2)}}\right)$	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
138	1/4	1.2473e-01	-	8.2577e-02	-	8.2577e-02	-
594	1/8	2.9768e-02	2.067	1.9855e-02	2.0562	1.9855e-02	1.8533
2466	1/16	7.3431e-03	2.0193	4.8975e-03	2.0194	4.8975e-03	2.22
10050	1/32	1.8292e-03	2.0051	1.2197e-03	2.0056	1.2197e-03	2.1761

Table 6.54. **Boussinesq equations, Case 1.1:** *UMR with* $p_t = 3$, $p_p = 2$ *and* $p_u = 3$.

ndof	h	$\frac{\|T-T_h\|_E}{\|T\|_E}$	$\log_2\left(\frac{T_{E(h)}}{T_{E(h/2)}}\right)$	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
188	1/4	1.4101e-02	-	6.9427e-03	-	6.9427e-03	-
692	1/8	1.8599e-03	2.9226	8.8884e-04	2.9655	8.8884e-04	3.1125
2660	1/16	2.3827e-04	2.9646	1.1274e-04	2.9789	1.1274e-04	3.0202
10436	1/32	3.0148e-05	2.9824	1.4226e-05	2.9864	1.4226e-05	3.0012

Table 6.55. **Boussinesq equations, Case 1.1:** *UMR with* $p_t = 4$, $p_p = 3$ *and* $p_u = 4$.

ndof	h	$\frac{\|T-T_h\|_E}{\|T\|_E}$	$\log_2\left(\frac{T_{E(h)}}{T_{E(h/2)}}\right)$	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
246	1/4	1.2694e-15	-	5.5246e-15	-	5.5246e-15	-
798	1/8	1.7858e-15	-0.49241	5.9368e-15	-0.10381	5.9368e-15	-0.14567
2862	1/16	9.7651e-15	-2.4511	4.2300e-15	0.48903	4.2300e-15	-0.64602
10830	1/32	7.2781e-14	-2.8979	4.3433e-15	-0.038148	4.3433e-15	-0.89591

Table 6.56. **Boussinesq equations, Case 1.2:** *UMR with* $p_t = 2$, $p_p = 1$ *and* $p_u = 2$.

ndof	h	$\frac{\|T-T_h\|_E}{\|T\|_E}$	$\log_2\left(\frac{T_{E(h)}}{T_{E(h/2)}}\right)$	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
138	1/4	1.5325e-01	-	2.2976e-01	-	2.2976e-01	-
594	1/8	4.0561e-02	1.9177	6.3422e-02	1.8571	6.3422e-02	2.0198
2466	1/16	9.5631e-03	2.0845	1.6058e-02	1.9817	1.6058e-02	2.5854
10050	1/32	2.3209e-03	2.0428	4.0197e-03	1.9981	4.0197e-03	2.1892

Table 6.57. **Boussinesq equations, Case 1.2:** *UMR with* $p_t = 3$, $p_p = 2$ *and* $p_u = 3$.

ndof	h	$\frac{\|T-T_h\|_E}{\|T\|_E}$	$\log_2\left(\frac{T_{E(h)}}{T_{E(h/2)}}\right)$	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
188	1/4	7.4405e-02	-	5.9489e-02	-	5.9489e-02	-
692	1/8	7.8907e-03	3.2372	8.0594e-03	2.8839	8.0594e-03	4.4123
2660	1/16	1.0191e-03	2.9528	1.0895e-03	2.887	1.0895e-03	3.2764
10436	1/32	1.3392e-04	2.9279	1.4272e-04	2.9324	1.4272e-04	2.9466

Table 6.58. **Boussinesq equations, Case 1.2:** *UMR with* $p_t = 4$, $p_p = 3$ *and* $p_u = 4$.

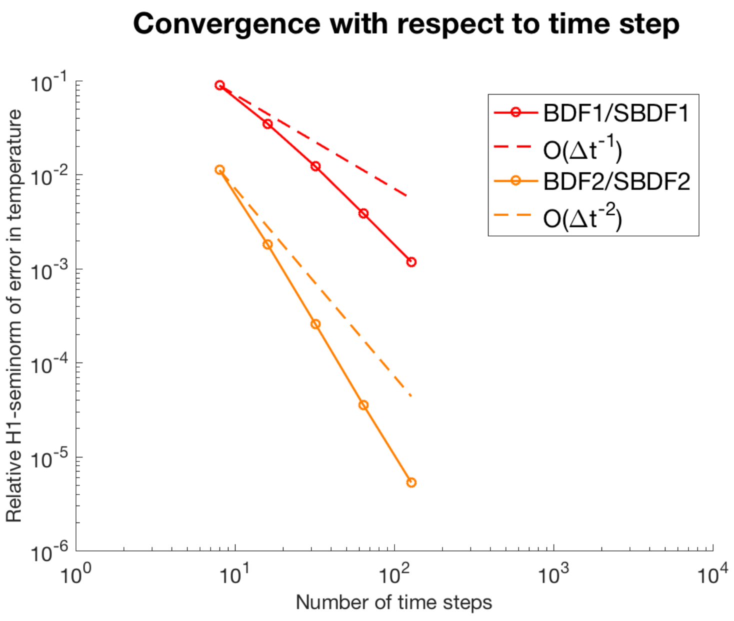
ndof	h	$\frac{\|T-T_h\|_E}{\|T\|_E}$	$\log_2\left(\frac{T_{E(h)}}{T_{E(h/2)}}\right)$	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
246	1/4	1.6391e-02	-	1.0651e-02	-	1.0651e-02	-
798	1/8	1.1105e-03	3.8836	7.5920e-04	3.8103	7.5920e-04	3.8546
2862	1/16	7.4223e-05	3.9032	5.0797e-05	3.9017	5.0797e-05	3.9201
10830	1/32	4.8132e-06	3.9468	3.2857e-06	3.9505	3.2857e-06	3.9495

Table 6.59. **Boussinesq equations, Case 1.2:** *UMR with* $p_t = 5$, $p_p = 4$ *and* $p_u = 5$.

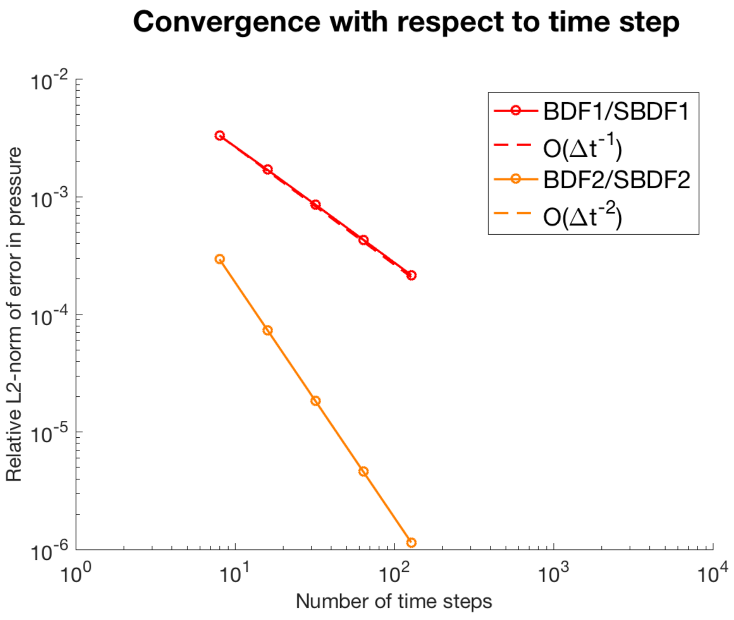
ndof	h	$\frac{\|T-T_h\|_E}{\|T\|_E}$	$\log_2\left(\frac{T_{E(h)}}{T_{E(h/2)}}\right)$	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
312	1/4	2.5682e-03	-	1.3323e-03	-	1.3323e-03	-
912	1/8	8.4729e-05	4.9218	4.3789e-05	4.9272	4.3789e-05	5.2097
3072	1/16	2.6999e-06	4.9719	1.3859e-06	4.9817	1.3859e-06	5.0172
11232	1/32	8.5190e-08	4.9861	4.3624e-08	4.9895	4.3624e-08	4.9897

Table 6.60. **Boussinesq equations, Case 1.2:** *UMR with* $p_t = 6$, $p_p = 5$ *and* $p_u = 6$.

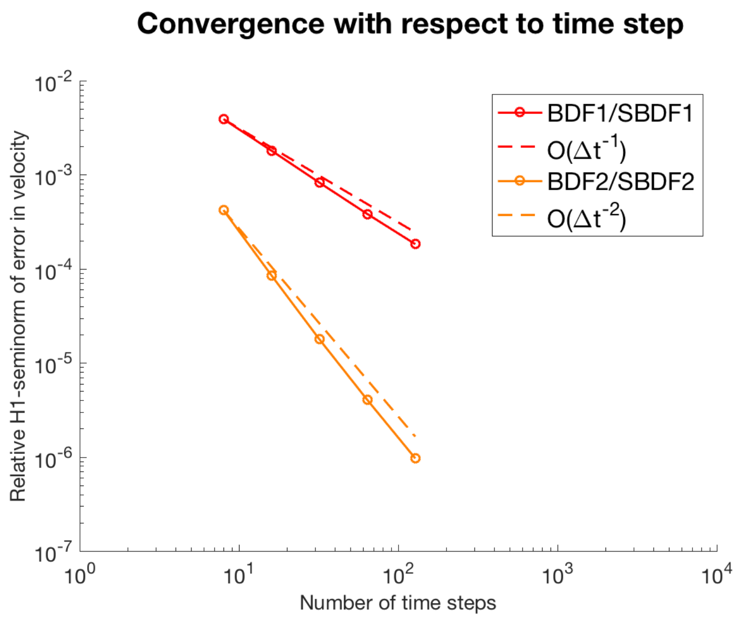
ndof	h	$\frac{\|T-T_h\|_E}{\|T\|_E}$	$\log_2\left(\frac{T_{E(h)}}{T_{E(h/2)}}\right)$	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(h)}}{p_{L^2(h/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(h)}}{\mathbf{u}_{E(h/2)}}\right)$
386	1/4	2.3901e-14	-	3.8543e-14	-	3.8543e-14	-
1034	1/8	4.2023e-15	2.5078	3.1583e-14	0.28732	3.1583e-14	-1.9467
3290	1/16	2.6659e-14	-2.6654	1.5655e-14	1.0125	1.5655e-14	0.92971
11642	1/32	6.6789e-14	-1.325	6.5052e-14	-2.0549	6.5052e-14	-4.9212



(a) *t-refinement: Relative error in the temperature (H^1-seminorm).*

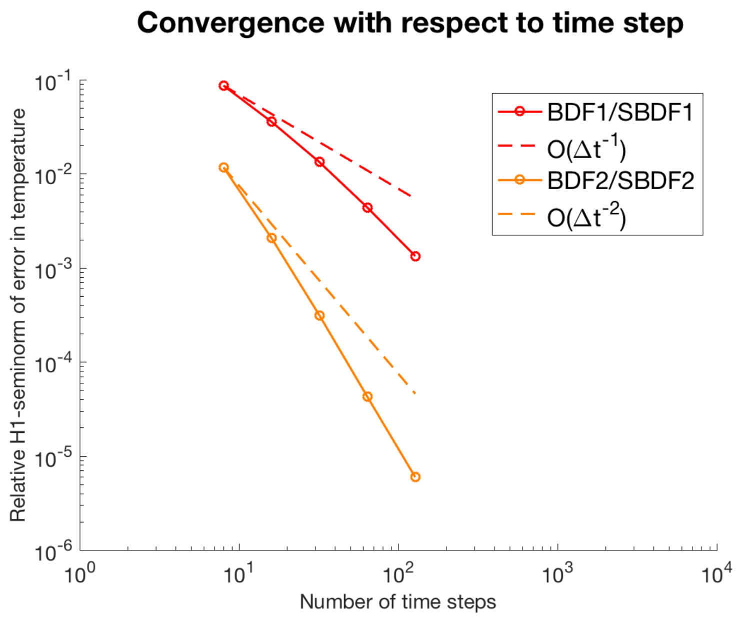


(b) *t-refinement: Relative error in the pressure (L^2-norm).*

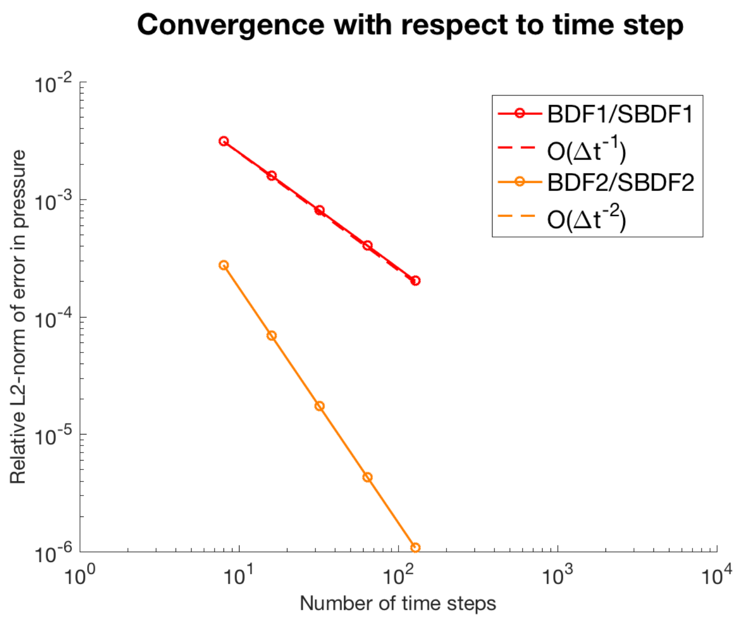


(c) *t-refinement: Relative error in the velocity field (H^1-seminorm).*

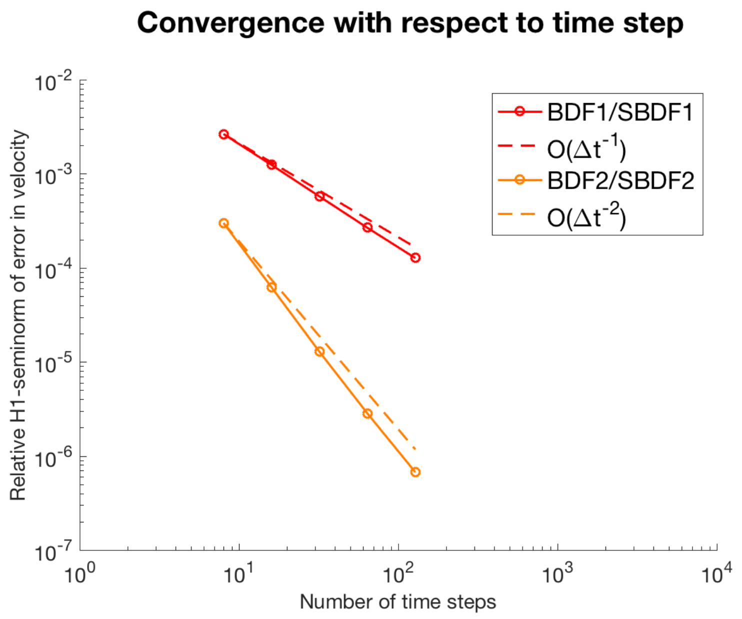
Figure 6.31. ***Boussinesq equations, Case 2.1***: *t-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*



(a) *t-refinement: Relative error in the temperature (H^1-seminorm).*



(b) *t-refinement: Relative error in the pressure (L^2-norm).*



(c) *t-refinement: Relative error in the velocity field (H^1-seminorm).*

Figure 6.32. ***Boussinesq equations, Case 2.2**: t-refinement, relative error plots (%) in L^2-norm and H^1-seminorm.*

Table 6.61. **Boussinesq equations, Case 2.1:** *t-refinement with BDF1/SBDF1.*

Δt	$\frac{\|T-T_h\|_E}{\|T\|_E}$	$\log_2\left(\frac{T_{E(\Delta t)}}{T_{E(\Delta t/2)}}\right)$	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(\Delta t)}}{p_{L^2(\Delta t/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(\Delta t)}}{\mathbf{u}_{E(\Delta t/2)}}\right)$
1/8	8.9024e-02	-	3.2992e-03	-	3.9061e-03	-
1/16	3.4955e-02	1.3487	1.6862e-03	0.96835	1.8068e-03	1.1123
1/32	1.2237e-02	1.5142	8.5225e-04	0.98443	8.2581e-04	1.1296
1/64	3.8591e-03	1.6649	4.2841e-04	0.99228	3.8314e-04	1.1079
1/128	1.1849e-03	1.7035	2.1478e-04	0.99615	1.8305e-04	1.0656

Table 6.62. **Boussinesq equations, Case 2.1:** *t-refinement with BDF2/SBDF2.*

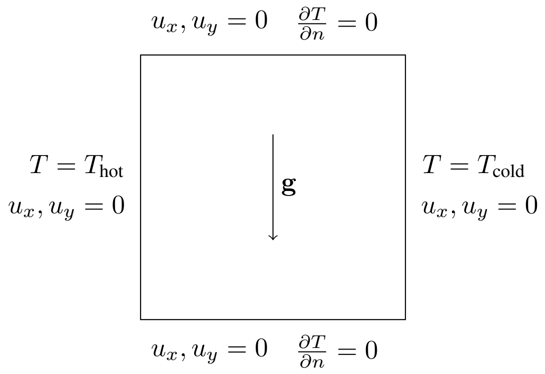
Δt	$\frac{\|T-T_h\|_E}{\|T\|_E}$	$\log_2\left(\frac{T_{E(\Delta t)}}{T_{E(\Delta t/2)}}\right)$	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(\Delta t)}}{p_{L^2(\Delta t/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(\Delta t)}}{\mathbf{u}_{E(\Delta t/2)}}\right)$
1/8	1.1193e-02	-	2.9271e-04	-	4.2091e-04	-
1/16	1.8232e-03	2.6181	7.3178e-05	2	8.5340e-05	2.3022
1/32	2.5783e-04	2.822	1.8294e-05	2	1.7820e-05	2.2597
1/64	3.5419e-05	2.8638	4.5736e-06	2	4.0330e-06	2.1436
1/128	5.2984e-06	2.7409	1.1434e-06	2	9.6460e-07	2.0639

Table 6.63. **Boussinesq equations, Case 2.2:** *t-refinement with BDF1/SBDF1.*

Δt	$\frac{\|T-T_h\|_E}{\|T\|_E}$	$\log_2\left(\frac{T_{E(\Delta t)}}{T_{E(\Delta t/2)}}\right)$	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(\Delta t)}}{p_{L^2(\Delta t/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(\Delta t)}}{\mathbf{u}_{E(\Delta t/2)}}\right)$
1/8	8.6767e-02	-	3.1010e-03	-	2.6482e-03	-
1/16	3.5813e-02	1.2767	1.5849e-03	0.96835	1.2448e-03	1.0891
1/32	1.3325e-02	1.4264	8.0105e-04	0.98443	5.7510e-04	1.114
1/64	4.3834e-03	1.604	4.0267e-04	0.99228	2.6688e-04	1.1076
1/128	1.3328e-03	1.7176	2.0187e-04	0.99616	1.2708e-04	1.0705

Table 6.64. **Boussinesq equations, Case 2.2:** *t-refinement with BDF2/SBDF2.*

Δt	$\frac{\|T-T_h\|_E}{\|T\|_E}$	$\log_2\left(\frac{T_{E(\Delta t)}}{T_{E(\Delta t/2)}}\right)$	$\frac{\|p-p_h\|_{L^2}}{\|p\|_{L^2}}$	$\log_2\left(\frac{p_{L^2(\Delta t)}}{p_{L^2(\Delta t/2)}}\right)$	$\frac{\|\mathbf{u}-\mathbf{u}_h\|_E}{\|\mathbf{u}\|_E}$	$\log_2\left(\frac{\mathbf{u}_{E(\Delta t)}}{\mathbf{u}_{E(\Delta t/2)}}\right)$
1/8	1.1717e-02	-	2.7509e-04	-	3.0036e-04	-
1/16	2.0795e-03	2.4943	6.8772e-05	2	6.2296e-05	2.2695
1/32	3.1251e-04	2.7343	1.7193e-05	2	1.2834e-05	2.2792
1/64	4.2540e-05	2.877	4.2982e-06	2	2.8362e-06	2.1779
1/128	5.9335e-06	2.8419	1.0746e-06	2	6.7186e-07	2.0778



$u_x, u_y = 0$ $\frac{\partial T}{\partial n} = 0$
$T = T_{\text{hot}}$
$u_x, u_y = 0$
$\mathbf{g}$
$T = T_{\text{cold}}$
$u_x, u_y = 0$
$u_x, u_y = 0$ $\frac{\partial T}{\partial n} = 0$

Figure 6.33. **Temperature-driven cavity:** *Mixed boundary conditions.*

6.6.3 Temperature-driven cavity

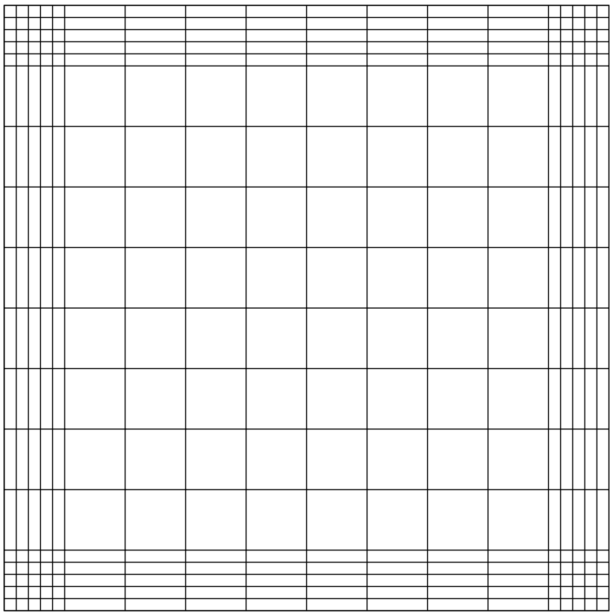
Boundary conditions

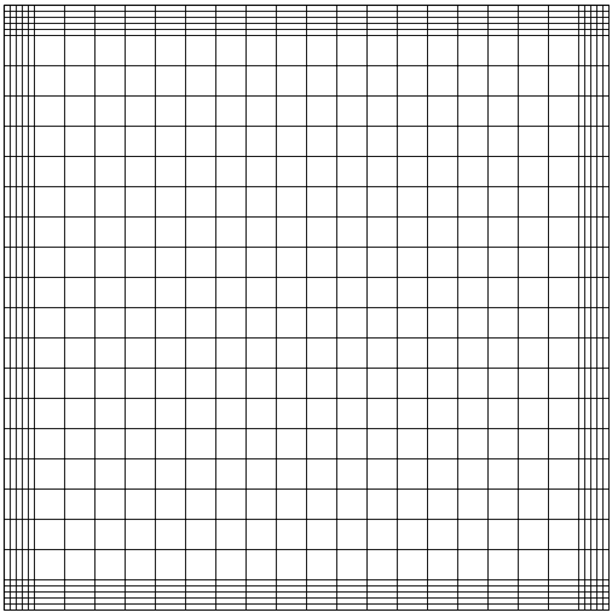
To test the proposed numerical algorithms, we will investigate a well-known benchmark example denoted *temperature-driven cavity*. The velocity field's Dirichlet conditions are still homogeneous as before (*no-slip boundary*), but the temperature is subject to mixed boundary conditions (figure 6.33). The momentum and temperature equations do not have any force term $\mathbf{f}$ and Q, respectively. The driving force is the non-homogeneous Dirichlet condition on the temperature field, which couples to the momentum equation through the buoyancy term on the right-hand side (2.16).

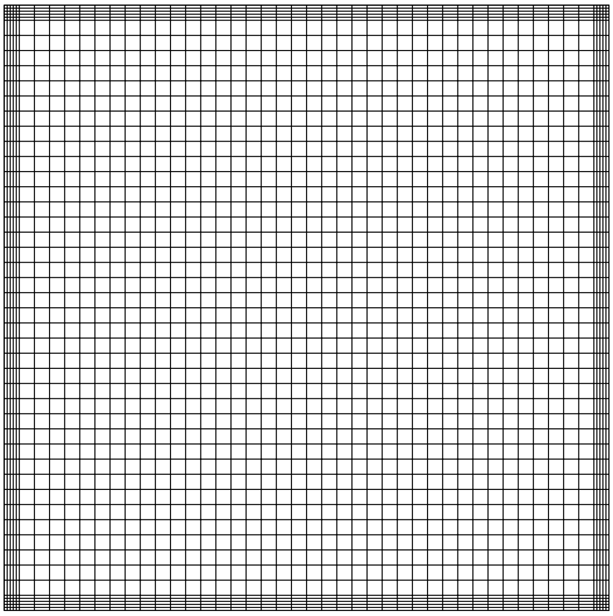
Before starting with the simulations, we will make some remarks on the discretization. The mixed temperature boundary conditions are continuous because the inhomogeneous Dirichlet conditions are constant, and this is indeed compatible with the homogeneous Neumann conditions. We use the decomposition $H^1 = H^1 \oplus H^1_0$ and write the solution as $T = T_I + T_H$, where T_H is homogenous along the boundary. The constant Dirichlet conditions implies, in the sense of isogeometric analysis, that the shape functions on Γ^T_D have the constant coefficients T_{hot} and T_{cold} (partition of unity).

To create T_I, we need a zero matrix of size $n_T \times n_T$. Then we insert T_{hot} and T_{cold} in the entries corresponding to Γ^T_D, and transform the matrix into a tensor vector for T_I. By inserting $T = T_I + T_H$ in the temperature equation, we get a "source term" Q and solve the equation with respect to T_H. The term Q does not vary with time because $\frac{\partial T_i}{\partial t} = 0$.

Since we are using a tensor mesh, it is appropriate to make the grid lines very dense near the boundary, as shown in figure 6.34. This makes the approximation of the boundary layers more accurate. Since the solution does not have sharp gradients in the centre of the domain, the grid resolution may be less there.


(a) *Temperature-driven cavity: Graded tensor mesh $\mathcal{M}_0$, ndof = 3352.*


(b) *Temperature-driven cavity: Graded tensor mesh $\mathcal{M}_1$, ndof = 8012.*



(c) *Temperature-driven cavity: Graded tensor mesh $\mathcal{M}_2$, ndof = 23332.*

Figure 6.34. Temperature-driven cavity: *Adaptive tensor meshes.*

We also need compatible initial conditions. Since the velocity field has homogeneous Dirichlet conditions, it is natural to choose $\mathbf{u}_0(x,y) = \mathbf{0}$. For the temperature, we choose an interpolating *blending function* defined as

$$T_0(x,y) = T_{\text{hot}} + (T_{\text{cold}} - T_{\text{hot}})x$$

On the Neumann boundary Γ_N^T, we require $\frac{\partial T}{\partial y} = 0$, and since the initial condition is independent of y, the Neumann conditions are satisfied at the corners.

The Nusselt number

When the solution has been found, we will calculate the *Nusselt number* [34]. This is a dimensionless coefficient used for estimating the wall heat transfer by measuring the horizontal heat flux over the domain. It can be expressed as follows:

$$Nu = T\mathbf{u} - \nabla T \tag{6.17}$$

Since we need to measure the heat flux on the wall, we will use the formula

$$Nu_\Gamma = \int_\Gamma (T\mathbf{u} - \nabla T) \cdot \widehat{\mathbf{n}}\, d\Gamma \tag{6.18}$$

It is common to analyze how the Nusselt number varies as a function of the Rayleigh number. In our MATLAB-code, the Boussinesq system is scaled in terms of the Reynold, Péclet and Richardson numbers. From table 2.3, we can deduce a new relation:

$$Ra = RePeRi$$

6.6.4 Simulation of temperature-driven cavity

Procedure of the simulation

In the next simulations, we let $Re = 1$ and $Pe = 1$, and Ri varies between 10^1, 10^2, 10^3 and 10^4. We expect from the theory that higher Ra will cause highly varying fluctuations and patterns in the contour lines of the numerical solution components.

In the initial mesh $\mathcal{M}_0$, there are 10 elements in each spatial direction. The boundary elements are divided into 5 smaller parts such that we can manage the boundary layers better, since we have not implemented SUPG for our Boussinesq solver. The polynomial degrees are $p_t = 3$, $p_p = 2$ and $p_u = 3$. We run the simulation with 200 time steps on the interval $[0, 0.5]$, such that the time step restriction of the IMEX-integrator is satisfied.

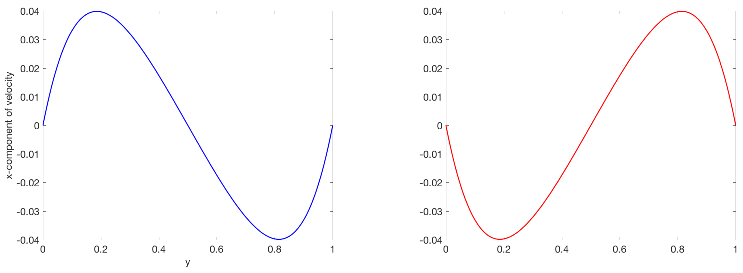
The contour lines of the four solution components are plotted with GLview, while the velocity components and average Nusselt numbers along $x = 1/2$ and $y = 1/2$ are plotted with MATLAB. The Nusselt number is plotted as a function of time, using formula (6.18) along the mentioned streamlines.

Discussion on the results

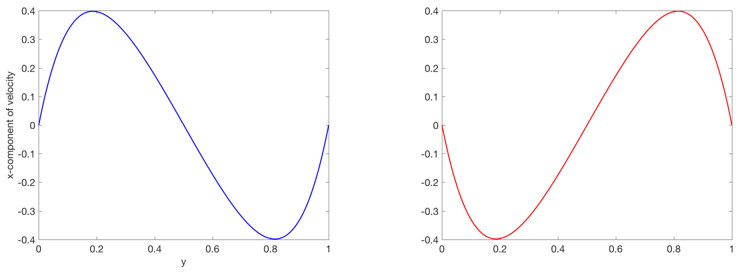
We see from figure 6.35 that the graphs of the velocity components along the streamlines $x = 1/2$ and $y = 1/2$ have a sinusoidal shape. As Ra increases, the graph is scaled by a factor of 10, but the shape is the same. But we see that the y-component along $y = 1/2$ becomes a little bit irregular when $Ra = 10^4$. This is expected because high Rayleigh number causes dynamical chaos. The same was observed in figure 6.38 and 6.41.

As seen in figure 6.36, 6.39 and 6.42, the graph of the Nusselt number as a function of time has some interesting patterns. For the Nusselt number along $x = 1/2$, the three first graphs have the same shape but is scaled with a factor of 10 as the Rayleigh number increases, but the last graph has a high peak before it flattens out. Along the streamline $y = 1/2$, the first plot is quite jagged, the two next ones are smooth and resemble each other, and the last has a negative peak. This is almost like in the streamline $x = 1/2$.

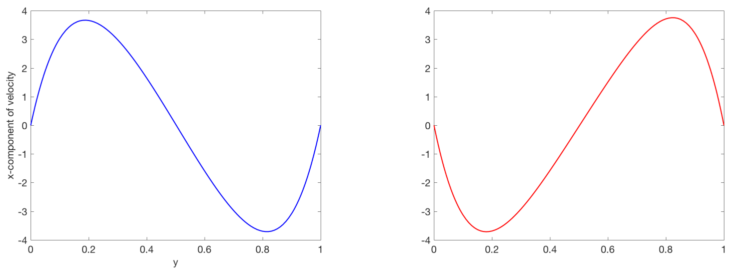
In figure 6.37, 6.40 and 6.43, we have contour plots of the velocity (x- and y-components), pressure and temperature. The velocity components do not change much in the beginning, but when $Ra = 10^4$, the contour lines become skewer. The contour plots of the pressure and temperature, anyway, changes much already at $Ra = 10^2$. At the end, the patterns become extremely irregular, and this indicates the highly varying fluctuations as expected from the theory.



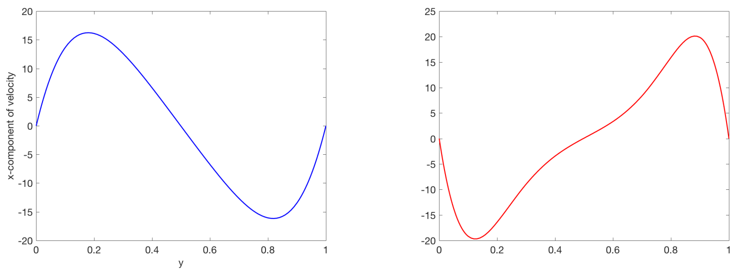
(a) $Re = 1$, $Ri = 10^1$, $Pe = 1$, $Ra = 10^1$.



(b) $Re = 1$, $Ri = 10^2$, $Pe = 1$, $Ra = 10^2$.

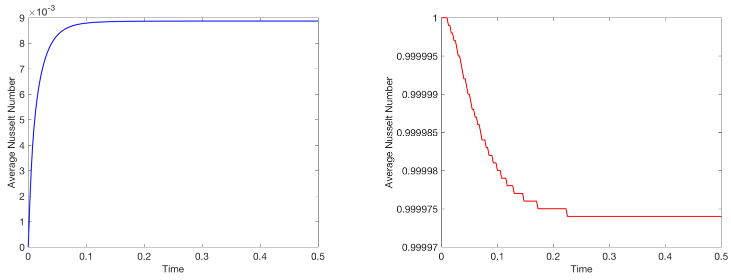


(c) $Re = 1$, $Ri = 10^3$, $Pe = 1$, $Ra = 10^3$.

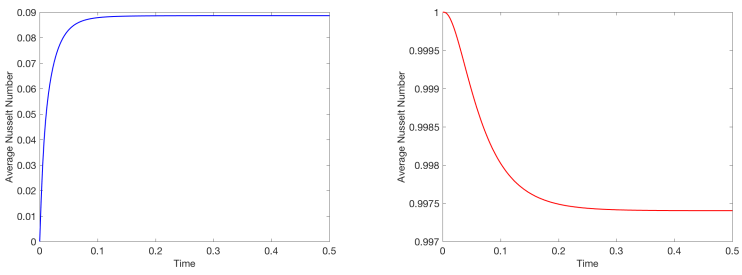


(d) $Re = 1$, $Ri = 10^4$, $Pe = 1$, $Ra = 10^4$.

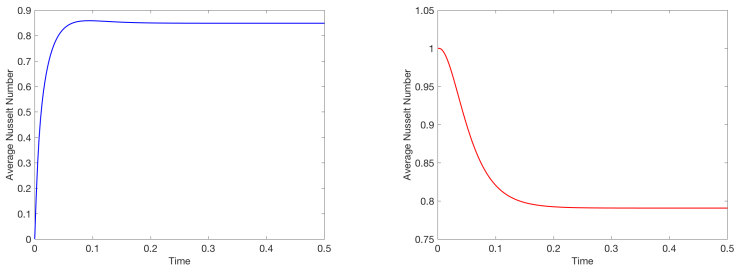
Figure 6.35. Temperature-driven cavity: *Cross-section plot of the velocity components along $x = 1/2$ and $y = 1/2$, for mesh $\mathcal{M}_0$.*



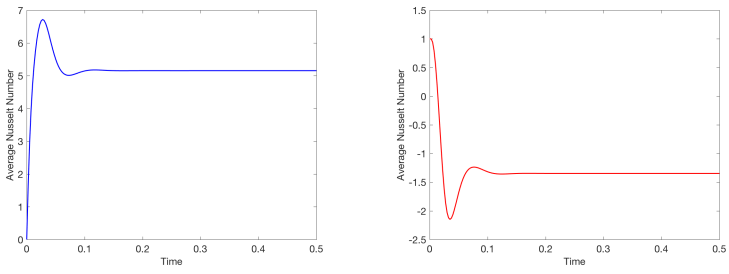
(a) $Re = 1,\ Ri = 10^1,\ Pe = 1,\ Ra = 10^1.$



(b) $Re = 1,\ Ri = 10^2,\ Pe = 1,\ Ra = 10^2.$

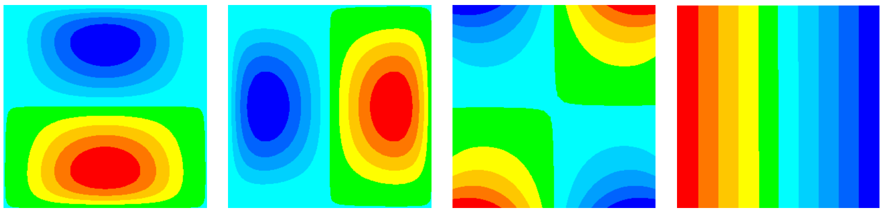


(c) $Re = 1,\ Ri = 10^3,\ Pe = 1,\ Ra = 10^3.$

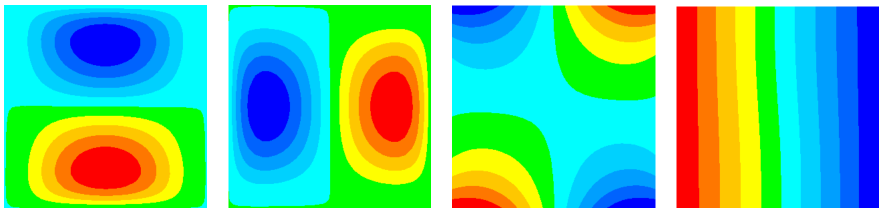


(d) $Re = 1,\ Ri = 10^4,\ Pe = 1,\ Ra = 10^4.$

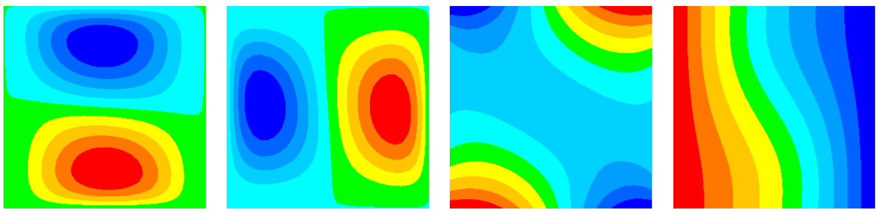
Figure 6.36. Temperature-driven cavity: *Plot of average Nusselt number, for mesh* $\mathcal{M}_0$.



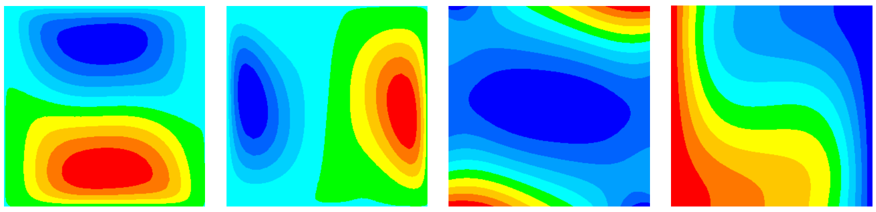
(a) $Re = 1$, $Ri = 10^1$, $Pe = 1$, $Ra = 10^1$



(b) $Re = 1$, $Ri = 10^2$, $Pe = 1$, $Ra = 10^2$

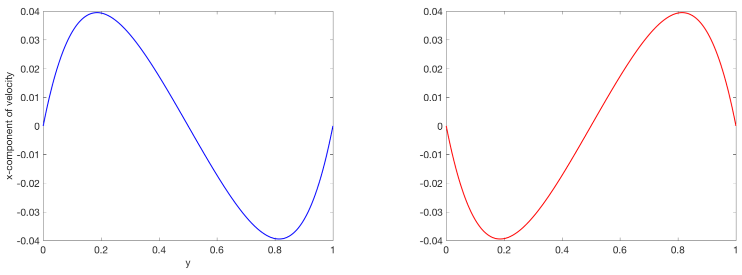


(c) $Re = 1$, $Ri = 10^3$, $Pe = 1$, $Ra = 10^3$

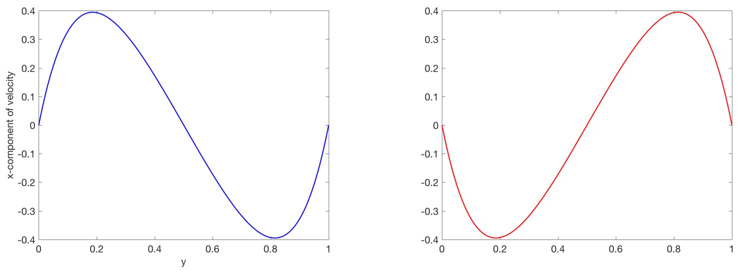


(d) $Re = 1$, $Ri = 10^4$, $Pe = 1$, $Ra = 10^4$

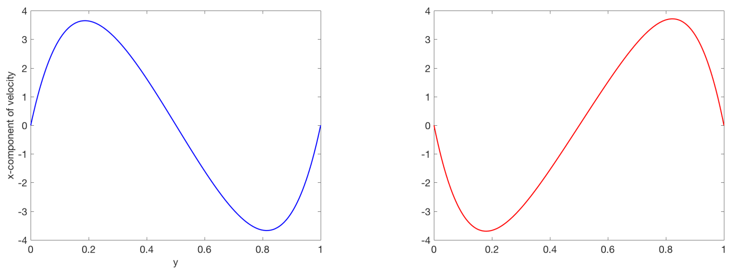
Figure 6.37. Temperature-driven cavity: *Contour plots of (from left) velocity components (u_x and u_y), pressure (p) and temperature (T), for mesh $\mathcal{M}_0$.*



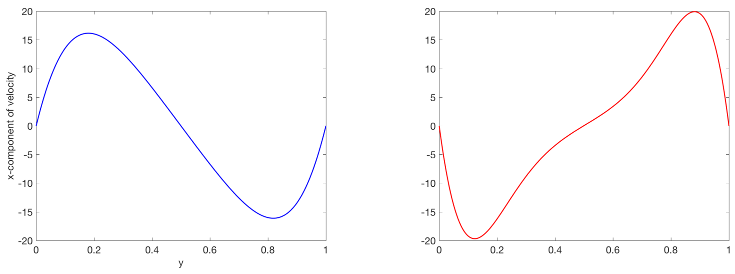
(a) $Re = 1$, $Ri = 10^1$, $Pe = 1$, $Ra = 10^1$.



(b) $Re = 1$, $Ri = 10^2$, $Pe = 1$, $Ra = 10^2$.

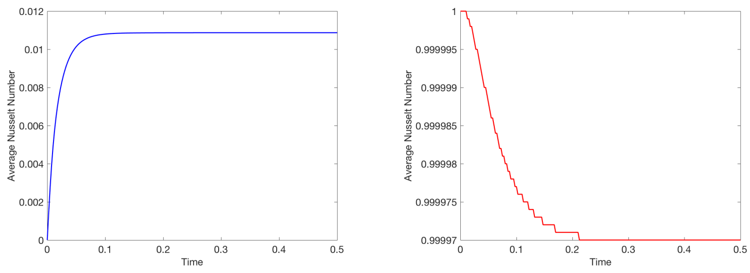


(c) $Re = 1$, $Ri = 10^3$, $Pe = 1$, $Ra = 10^3$.

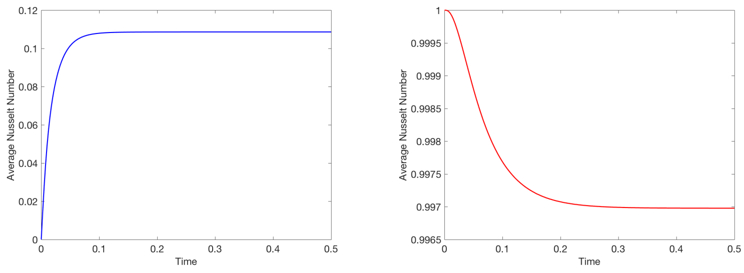


(d) $Re = 1$, $Ri = 10^4$, $Pe = 1$, $Ra = 10^4$.

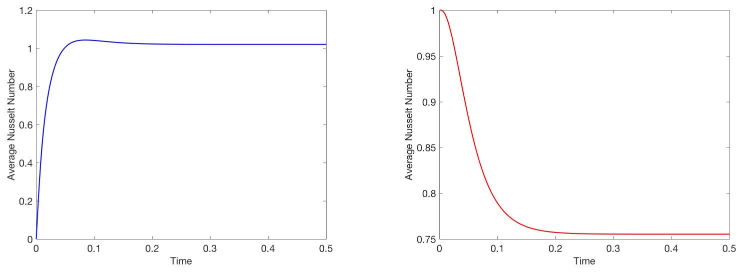
Figure 6.38. Temperature-driven cavity: *Cross-section plot of the velocity components along $x = 1/2$ and $y = 1/2$, for mesh $\mathcal{M}_1$.*



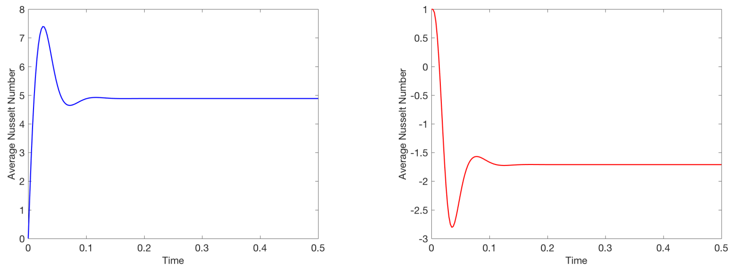
(a) $Re = 1$, $Ri = 10^1$, $Pe = 1$, $Ra = 10^1$.



(b) $Re = 1$, $Ri = 10^2$, $Pe = 1$, $Ra = 10^2$.

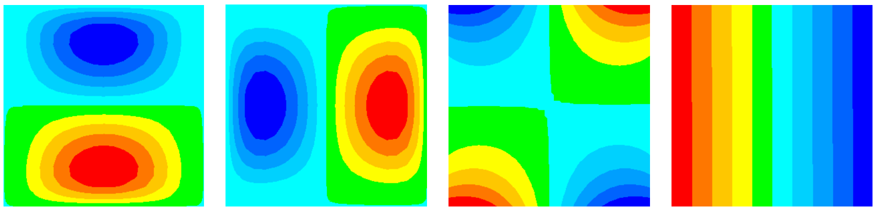


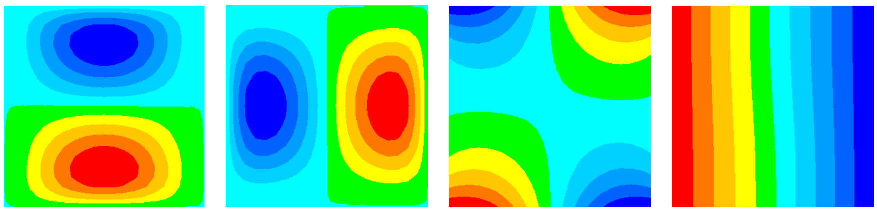
(c) $Re = 1$, $Ri = 10^3$, $Pe = 1$, $Ra = 10^3$.

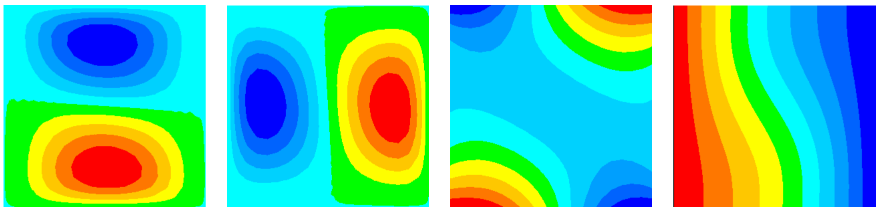


(d) $Re = 1$, $Ri = 10^4$, $Pe = 1$, $Ra = 10^4$.

Figure 6.39. Temperature-driven cavity: *Plot of average Nusselt number, for mesh* $\mathcal{M}_1$.


(a) $Re = 1$, $Ri = 10^1$, $Pe = 1$, $Ra = 10^1$


(b) $Re = 1$, $Ri = 10^2$, $Pe = 1$, $Ra = 10^2$


(c) $Re = 1$, $Ri = 10^3$, $Pe = 1$, $Ra = 10^3$

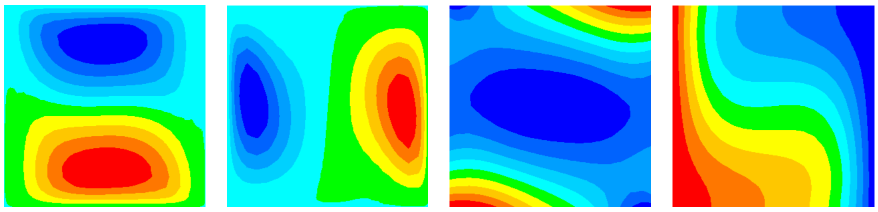
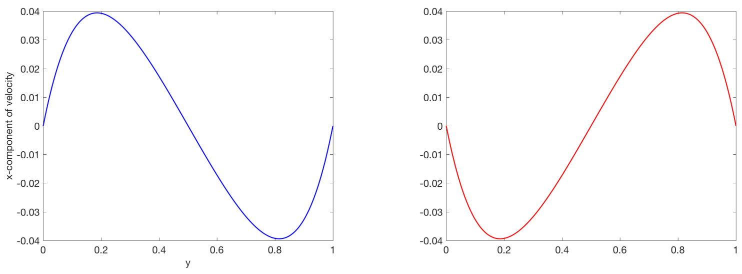
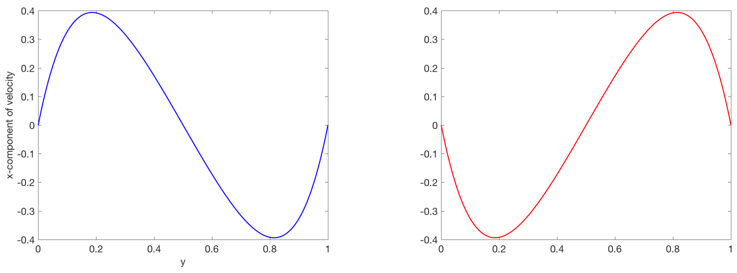

(d) $Re = 1$, $Ri = 10^4$, $Pe = 1$, $Ra = 10^4$

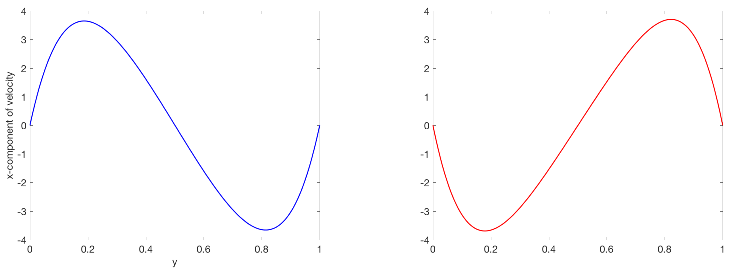
Figure 6.40. Temperature-driven cavity: *Contour plots of (from left) velocity components (u_x and u_y), pressure (p) and temperature (T), for mesh $\mathcal{M}_1$.*



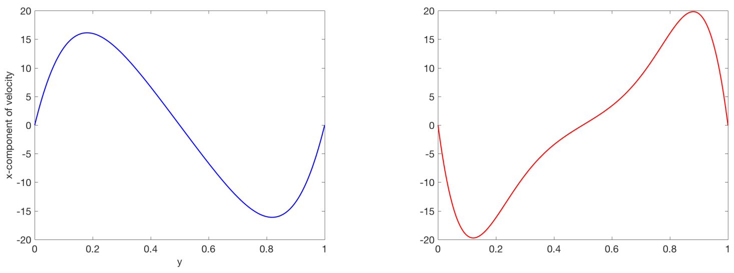
(a) $Re = 1$, $Ri = 10^1$, $Pe = 1$, $Ra = 10^1$.



(b) $Re = 1$, $Ri = 10^2$, $Pe = 1$, $Ra = 10^2$.

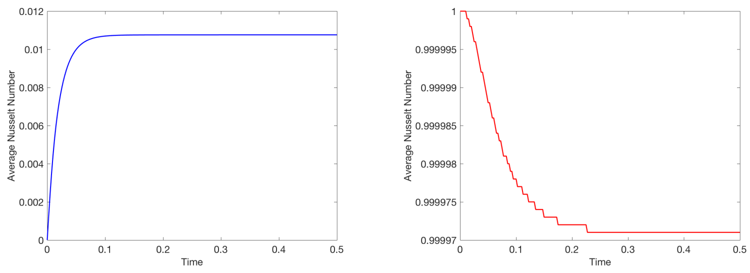


(c) $Re = 1$, $Ri = 10^3$, $Pe = 1$, $Ra = 10^3$.

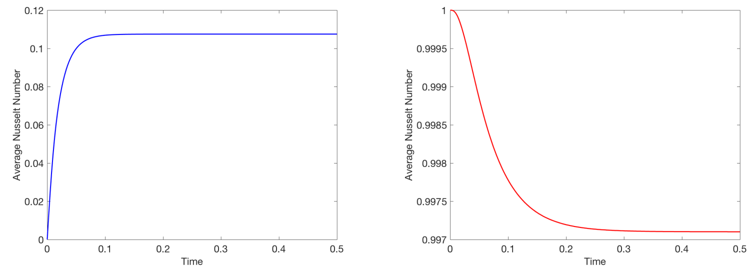


(d) $Re = 1$, $Ri = 10^4$, $Pe = 1$, $Ra = 10^4$.

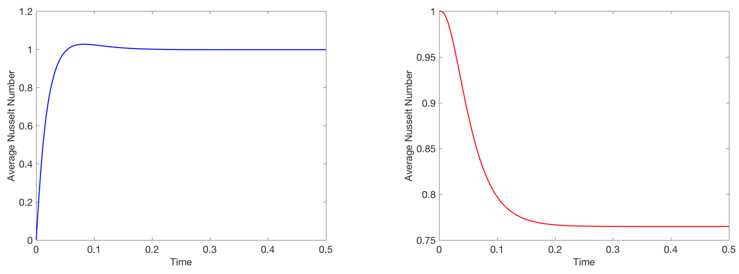
Figure 6.41. Temperature-driven cavity: *Cross-section plot of the velocity components along $x = 1/2$ and $y = 1/2$, for mesh $\mathcal{M}_2$.*



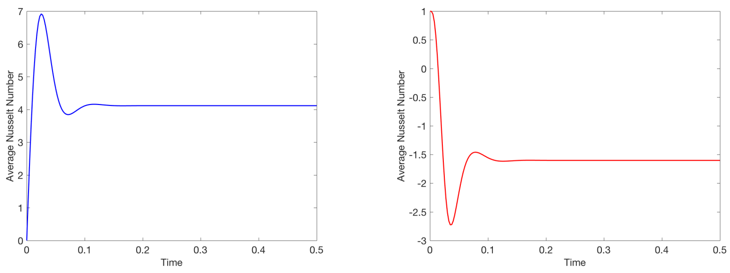
(a) $Re = 1,\ Ri = 10^1,\ Pe = 1,\ Ra = 10^1.$



(b) $Re = 1,\ Ri = 10^2,\ Pe = 1,\ Ra = 10^2.$

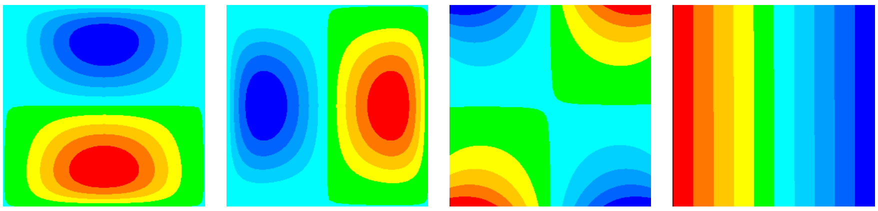


(c) $Re = 1,\ Ri = 10^3,\ Pe = 1,\ Ra = 10^3.$

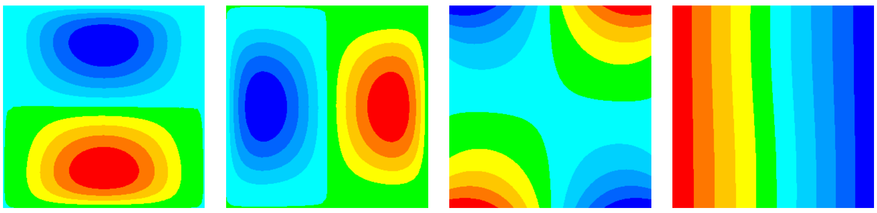


(d) $Re = 1,\ Ri = 10^4,\ Pe = 1,\ Ra = 10^4.$

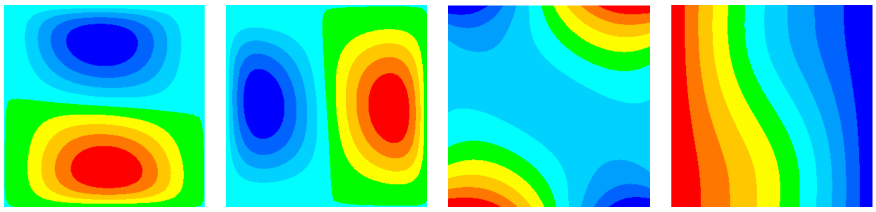
Figure 6.42. Temperature-driven cavity: *Plot of average Nusselt number, for mesh* $\mathcal{M}_2$.



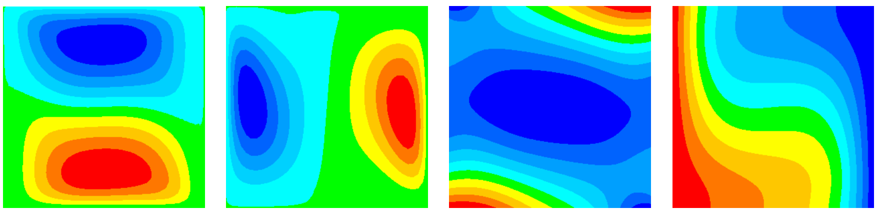
(a) $Re = 1$, $Ri = 10^1$, $Pe = 1$, $Ra = 10^1$



(b) $Re = 1$, $Ri = 10^2$, $Pe = 1$, $Ra = 10^2$



(c) $Re = 1$, $Ri = 10^3$, $Pe = 1$, $Ra = 10^3$



(d) $Re = 1$, $Ri = 10^4$, $Pe = 1$, $Ra = 10^4$

Figure 6.43. Temperature-driven cavity: *Contour plots of (from left) velocity components (u_x and u_y), pressure (p) and temperature (T), for mesh $\mathcal{M}_2$.*

Table 6.65. Temperature-driven cavity: *Maximal value of velocity component u_x along $x = 0.5$ and the associated y-coordinate.*

Ra	10^1	10^2	10^3	10^4
Ref [13]	—	—	3.649	16.178
	—	—	(0.813)	(0.823)
Ref [47]	—	—	3.68	16.1
	—	—	(0.817)	(0.817)
Ref [22] (FEM)	—	—	3.489	16.122
	—	—	(0.813)	(0.815)
Ref [22] (DSC)	—	—	3.6434	15.967
	—	—	(0.8167)	(0.8167)
Ref [30] $strat_{25\%}$	—	—	3.6574	16.1732
	—	—	(0.8000)	(0.8000)
Ref [30] $strat_{\eta_T^e}$	—	—	3.652	16.094
	—	—	(0.8)	(0.8)
Ref [30] $strat_{\max(\eta_{NS}^e,\eta_T^e)}$	—	—	3.649	16.202
	—	—	(0.813)	(0.825)
Mesh $\mathcal{M}_0$	0.040	0.398	3.711	16.171
	(0.814)	(0.814)	(0.815)	(0.818)
Mesh $\mathcal{M}_1$	0.040	0.395	3.672	16.126
	(0.814)	(0.814)	(0.814)	(0.819)
Mesh $\mathcal{M}_2$	0.039	0.394	3.659	16.119
	(0.814)	(0.814)	(0.814)	(0.819)

There is no analytical solution to the temperature-driven cavity problem. Hence, we must compare our numerical results with other previous results to determine whether our own simulations are good enough. In the PhD-thesis of Hægland (2006), we found a detailed comparison of various results on temperature-driven cavity using the finite element method with adaptive meshing. Thus, we found it convenient to list down some of these results directly in this thesis and compare them with our own.

In table 6.65, we have listed the maximal value of the velocity's x-component along $x = 1/2$, and the corresponding y-coordinate (in paranthesis). The same is done in table 6.66 with the y-component of the velocity along $y = 1/2$. None of the other authors did not consider the cases where $Ra = 10$ and $Ra = 100$, so parts of the corresponding columns are therefore blank. We have taken all the three meshes $\mathcal{M}_0$, $\mathcal{M}_1$ and $\mathcal{M}_2$ into account in the comparison of the results.

Table 6.66. Temperature-driven cavity: *Maximal value of velocity component u_y along $y = 0.5$ and the associated x-coordinate.*

Ra	10^1	10^2	10^3	10^4
Ref [13]	—	—	3.697	19.617
	—	—	(0.178)	(0.119)
Ref [47]	—	—	3.73	19.9
	—	—	(0.1827)	(0.1246)
Ref [22] (FEM)	—	—	3.686	19.79
	—	—	(0.188)	(0.12)
Ref [22] (DSC)	—	—	3.686	19.98
	—	—	(0.183)	(0.117)
Ref [30] $strat_{25\%}$	—	—	3.6224	19.4871
	—	—	(0.2000)	(0.1250)
Ref [30] $strat_{\eta_T^e}$	—	—	3.699	19.43
	—	—	(0.2)	(0.125)
Ref [30] $strat_{\max(\eta_{NS}^e, \eta_T^e)}$	—	—	3.697	19.61
	—	—	(0.175)	0.125
Mesh $\mathcal{M}_0$	0.040	0.398	3.714	19.708
	(0.186)	(0.186)	(0.181)	(0.125)
Mesh $\mathcal{M}_1$	0.040	0.395	3.695	19.696
	(0.186)	(0.186)	(0.180)	(0.122)
Mesh $\mathcal{M}_2$	0.039	0.394	3.693	19.722
	(0.186)	(0.186)	(0.179)	(0.122)

As we see from table 6.65 and 6.66, the obtained results from temperature-driven cavity do not deviate so much from the other authors, and this indicates that the simulation went well. It might happen that if we had used SUPG-formulation on the Boussinesq equations, then the results would have been more accurate, in addition to using adaptive mesh refinement. The latter one would make approximation of the boundary layers better.

Davis [13] has estimated the error in the benchmark solutions that he reports to be 0.1 % and 0.2 % for $Ra = 10^3$ and $Ra = 10^4$, respectively. Notice that his benchmark solutions are obtained by using Richardson extrapolation of the computed numerical results. Our results obtained with the finest mesh $\mathcal{M}_2$ are within 0.5 % difference to the reported benchmark solutions of Davis [13], which we consider to be very convincing results.

Chapter 7

Concluding remarks

7.1 Conclusion

The numerical examples have demonstrated that isogeometric analysis works very well for solving many of the partial differential equations arising in heat transfer and fluid flow. All the numerical solutions converged as expected from the theoretical assumptions. There were sometimes a few small disturbances in the convergence graphs, but this was mostly caused by noise at the machine precision level, so this can be neglected.

The use of a conformal mesh with elementary tensor refinement worked well because all the analytical solutions belonged to C^{∞}. It seems that combining the Bubnov-Galerkin and Petrov-Galerkin methods with splines as basis functions is optimal for reducing the effect of boundary layers, especially the latter one.

When we solved the Boussinesq equations, we observed that segregating the system of equations by applying two different but A-stable integrators worked properly. The global computational complexity was quite reduced, and the final results were suitable. From the theoretical assumptions, we presumed that the convergence of the numerical solution in the h- and p-refinements could be expressed asymptotically by the general estimate

$$\mathcal{O}\left(\text{ndof}^{-\frac{p+1-k}{2^{d-1}}}\right) \tag{7.1}$$

as described in chapter 6. This pattern was discovered in all the numerical studies. There were some irregularities in the heat and advection-diffusion equations, for the analytical solutions were not polynomials. But for the Navier-Stokes and Boussinesq equations, we could construct polynomial solutions such that either the spatial or temporal error was totally excluded. Hence, there were no disturbing noise polluting the error convergence.

In the t-refinement, we assumed that the convergence in the different norms would be the same because h and p were constant all the time, so the order of the time-integrator alone would determine the convergence rate. This behaviour has been observed, and this indicates that the time integration was correct.

Our final benchmark case, Temperature-driven cavity, has been thoroughly investigated by many authors and in particular Davis [13]. He has estimated the error in the benchmark solutions (obtained with Richardson extrapolation) that he reports to be $0.1\ \%$ and $0.2\ \%$ for $Ra = 10^3$ and $Ra = 10^4$, respectively. Our results obtained with the finest mesh $\mathcal{M}_2$ are within $0.5\ \%$ difference to the reported benchmark solutions of Davis [13], which we consider to be very good.

7.2 Future work

In all the simulations, we used a very simple tensor mesh with B-splines, and it worked properly. In the future, it will be very appropriate to use LR B-splines such that we can apply local refinement and adaptive mesh generation. This makes it easier to approximate parts of the domain where the numerical solution behaves irregularly, like boundary layers.

Applying algebraic multigrid and special saddle-point algorithms can be very important because the equation system is large and nonlinear, so we need some robust methods to solve it quickly. Implementing the solver in C++ could also have been an alternative way for increasing the algorithmic speed. Using a supercomputer with higher memory and faster computations can also be relevant for future work where the boundary and initial value problems are more complicated to handle.

Anyway, the next major task is to examine how we can control the inner climate of ice hockey stadiums, by solving the Boussinesq equations inside the stadium and determine how the air circulation and temperature distribution can be regulated effectively such that all the criterions for good inner climate are fully satisfied. This is too complicated to be implemented from scratch, so using IFEM for this purpose will be of high importance.

Appendix A

Multivariate calculus

The derivation of the Navier-Stokes equation and Boussinesq equations relies on many vector identities and multiple integral theorems. Therefore, we present all these topics detailed but short. For further details, we refer to [25, 48, 58].

A.1 Multiple integration

Proposition 1 (Line and surface integrals)**.** *Let f be a continuous function, and C is a smooth curve parametrized by $\mathbf{r}(t) = a(t)\hat{\mathbf{i}} + b(t)\hat{\mathbf{j}} + c(t)\hat{\mathbf{k}}$, and $t_1 \leq t \leq t_2$. Then the line integral of f over C is*

$$\int_C f(x,y,z)\,ds = \int_{t_1}^{t_2} f(a(t),b(t),c(t))|\mathbf{r}'(t)|\,dt$$

Let a smooth surface S be parametrized by $\mathbf{r}(u,v) = a(u,v)\hat{\mathbf{i}} + b(u,v)\hat{\mathbf{j}} + c(u,v)\hat{\mathbf{k}}$, $u_1 \leq u \leq u_2$ and $v_1 \leq v \leq v_2$, such that $\mathbf{r} \in C^1$ and $\frac{\partial \mathbf{r}}{\partial u} \times \frac{\partial \mathbf{r}}{\partial v} \neq 0$ on $\operatorname{int}(S)$. Then the surface integral of f over S is

$$\iint_S f(x,y,z)\,dS = \int_{v_1}^{v_2}\int_{u_1}^{u_2} f(a(u,v),b(u,v),c(u,v))\left|\frac{\partial \mathbf{r}}{\partial u} \times \frac{\partial \mathbf{r}}{\partial v}\right|\,du\,dv$$

Definition 1 (Jacobian transform)**.** *Define (x, y, z) and (u, v, w) as two different sets of coordinates. The Jacobian of the bijective coordinate transformation $x = g(u, v, w)$, $y = h(u, v, w)$, $z = k(u, v, w)$ is defined as the determinant*

$$J(u, v, w) = \begin{vmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} & \frac{\partial x}{\partial w} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} & \frac{\partial y}{\partial w} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} & \frac{\partial z}{\partial w} \end{vmatrix} \tag{A.1}$$

The determinant of the inverse transformation $(u, v, w) \mapsto (x, y, z)$ is given by

$$J(x, y, z) = \frac{1}{J(u, v, w)} \tag{A.2}$$

Proposition 2 (Variable change in multiple integrals)**.** *Let G be a region in the uvw-plane transformed into the region R in the xyz-plane, such that the coordinate transform is a diffeomorphism (bijective and smooth at every point). Then R is the image of G, and G is the preimage of R. If J is the Jacobian of the transform, then we have the following change of variables in a multiple integral:*

$$\begin{aligned} &\iiint\limits_R f(x, y, z)\, dx\, dy\, dz \\ = &\iiint\limits_G f(g(u, v, w), h(u, v, w), k(u, v, w)) \big| J(u, v, w) \big|\, du\, dv\, dw \end{aligned} \tag{A.3}$$

A.2 Vector calculus

Definition 2 (Common operators)**.** *Let $f : \mathbb{R}^3 \to \mathbb{R}$ be a scalar field, and $\mathbf{F} : \mathbb{R}^3 \to \mathbb{R}^3$ is a vector field. The standard differential vector operators in Cartesian coordinates are*

Gradient: $$\nabla f = \frac{\partial f}{\partial x}\hat{\mathbf{i}} + \frac{\partial f}{\partial y}\hat{\mathbf{j}} + \frac{\partial f}{\partial z}\hat{\mathbf{k}}$$

Divergence: $$\nabla \cdot \mathbf{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}$$

Curl: $$\nabla \times \mathbf{F} = \begin{vmatrix} \hat{\mathbf{i}} & \hat{\mathbf{j}} & \hat{\mathbf{k}} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ F_x & F_y & F_z \end{vmatrix}$$

Laplacian: $$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

Proposition 3 (Vector identities)**.**

$$\nabla \times (\nabla f) = \mathbf{0} \tag{A.5a}$$
$$\nabla \cdot (\nabla \times \mathbf{F}) = 0 \tag{A.5b}$$
$$\nabla \cdot (\nabla f) = \nabla^2 f \tag{A.5c}$$
$$\nabla(\mathbf{F} \cdot \mathbf{G}) = \mathbf{F} \times (\nabla \times \mathbf{G}) + \mathbf{G} \times (\nabla \times \mathbf{F}) + (\mathbf{F} \cdot \nabla)\mathbf{G} + (\mathbf{G} \cdot \nabla)\mathbf{F} \tag{A.5d}$$
$$\nabla \cdot (\mathbf{F} \times \mathbf{G}) = \mathbf{G} \cdot (\nabla \times \mathbf{F}) - \mathbf{F} \cdot (\nabla \times \mathbf{G}) \tag{A.5e}$$
$$\nabla \times (\mathbf{F} \times \mathbf{G}) = (\mathbf{G} \cdot \nabla)\mathbf{F} - (\mathbf{F} \cdot \nabla)\mathbf{G} + (\nabla \mathbf{G})\mathbf{F} - (\nabla \mathbf{F})\mathbf{G} \tag{A.5f}$$
$$\nabla \times (\nabla \times \mathbf{F}) = \nabla(\nabla \cdot \mathbf{F}) - \nabla^2 \mathbf{F} \tag{A.5g}$$
$$\nabla \cdot (\nabla f \times \nabla g) = 0 \tag{A.5h}$$
$$\nabla^2 (fg) = f\nabla^2 g + g\nabla^2 f + 2(\nabla f \cdot \nabla g) \tag{A.5i}$$
$$(\mathbf{F} \cdot \nabla)\mathbf{F} = (\nabla \times \mathbf{F}) \times \mathbf{F} + (1/2)\nabla |\mathbf{F}|^2 \tag{A.5j}$$

Theorem 1 (The theorems of Green, Stokes and Gauss)**.** *Let A be a closed area with the piecewise smooth boundary C, and V is a closed volume with the piecewise smooth surface S. Let $\hat{\mathbf{k}}$ be the unit vector in z-direction, and $\hat{\mathbf{n}}$ is the unit normal vector on the surface. Then we have the following integral theorems:*

$$\textit{Green's theorem 1:} \quad \oint_C \mathbf{F} \cdot \, d\mathbf{r} = \iint_R (\nabla \times \mathbf{F}) \cdot \hat{\mathbf{k}} \, dA \tag{A.6a}$$
$$\textit{Green's theorem 2:} \quad \oint_C \mathbf{F} \cdot \hat{\mathbf{n}} \, ds = \iint_R \nabla \cdot \mathbf{F} \, dA \tag{A.6b}$$
$$\textit{Stokes' theorem:} \quad \oint_C \mathbf{F} \cdot \, d\mathbf{r} = \iint_S (\nabla \times \mathbf{F}) \cdot \hat{\mathbf{n}} \, dS \tag{A.6c}$$
$$\textit{Gauss' theorem:} \quad \oiint_S \mathbf{F} \cdot \hat{\mathbf{n}} \, dS = \iiint_V \nabla \cdot \mathbf{F} \, dV \tag{A.6d}$$

Corollary 1 (Green's identities)**.** *From Gauss' theorem, Green's identities holds:*

$$\iiint_V g\nabla^2 f \, dV = \oiint_S g\frac{\partial f}{\partial n} \, dS - \iiint_V \nabla f \cdot \nabla g \, dV \tag{A.7a}$$
$$\iiint_V g\nabla^2 f - f\nabla^2 g \, dV = \oiint_S g\frac{\partial f}{\partial n} - f\frac{\partial g}{\partial n} \, dS \tag{A.7b}$$

Corollary 2 (Special integral identities)**.** *From Stokes' and Gauss' theorems, we have the following integral identities:*

$$\iiint_V \nabla f \, dV = \oiint_S f \hat{\mathbf{n}} \, dS \tag{A.8a}$$

$$\iiint_V \nabla^2 f \, dV = \oiint_S \frac{\partial f}{\partial n} \, dS \tag{A.8b}$$

$$\iiint_V f(\nabla \cdot \mathbf{F}) \, dV = \oiint_S f(\mathbf{F} \cdot \hat{\mathbf{n}}) \, dS - \iiint_V \nabla f \cdot \mathbf{F} \, dV \tag{A.8c}$$

$$\iiint_V \nabla \times \mathbf{F} \, dV = -\oiint_S \mathbf{F} \times \hat{\mathbf{n}} \, dS \tag{A.8d}$$

$$\iiint_V \mathbf{F} \cdot (\nabla \times \mathbf{G}) \, dV = \iiint_V \mathbf{G} \cdot (\nabla \times \mathbf{F}) \, dV - \oiint_S (\mathbf{F} \times \mathbf{G}) \cdot \hat{\mathbf{n}} \, dS \tag{A.8e}$$

Appendix B

Real and functional analysis

The proofs in Chapter 3 relies on many theorems and inequalities from real and functional analysis, so we give a brief but systematic description of the relevant theory.

B.1 Lebesgue space

Definition 3 (Lebesgue space L^p [1]). *If $\Omega \subset \mathbb{R}^n$ has nonzero measure and $p \geq 1$, then the Lebesgue space $L^p(\Omega)$ is a Banach space consisting of Lebesgue measurable functions on Ω such that they possess the following finite norms:*

$$\|u\|_{L^p} = \left(\int_\Omega |u(x)|^p \, dx \right)^{\frac{1}{p}} \qquad 1 \leq p < \infty \tag{B.1a}$$

$$\|u\|_{L^\infty} = \sup_{x \in \Omega} |f(x)| \qquad p = \infty \tag{B.1b}$$

The space $L^2(\Omega)$ is a Hilbert space with the inner product

$$\langle u, v \rangle_{L^2} = \int_\Omega |u(x)v(x)| \, dx \tag{B.2}$$

Theorem 2 (Young's inequalities [1]). *Let p and q be conjugate exponents ($1 \leq p, q \leq \infty$, $1/p + 1/q = 1$), and $a, b, \epsilon > 0$. Then, Young's inequalities are defined as follows:*

$$\textit{Young's inequality 1:} \qquad ab \leq \frac{a^p}{p} + \frac{b^q}{q} \tag{B.3a}$$

$$\textit{Young's inequality 2:} \qquad ab \leq \epsilon a^2 + \frac{b^2}{4\epsilon} \tag{B.3b}$$

Theorem 3 (L^p-inequalities [1]). *In L^p, we have the following inequalities:*

$$\text{Hölder's inequality:} \qquad \|uv\|_{L^1} \leq \|u\|_{L^p}\|v\|_{L^q} \tag{B.4a}$$

$$\text{Minkowski's inequality:} \qquad \|u+v\|_{L^p} \leq \|u\|_{L^p} + \|v\|_{L^p} \tag{B.4b}$$

Theorem 4 (Completeness, inclusion and reflexivity properties of L^p [1]). *The space L^p is the completion of C^0 in the L^p-norm:*

$$L^p \equiv \overline{C^0}^{\|\cdot\|_{L^p}}$$

If Ω is a domain with finite measure, and $1 < p < q < \infty$, then we have the inclusion

$$L^q(\Omega) \subset L^p(\Omega) \subset L^1(\Omega)$$

L^p is reflexive iff p and q are conjugate exponents satisfying $1 < p < q < \infty$:

$$(L^q)^* = L^p$$

Definition 4 (Quotient Lebesgue space [6]). *The quotient Lebesgue space is a Banach space given by*

$$L_0^p(\Omega) = \left\{ f \in L^p(\Omega) : \int_\Omega f\,dx = 0 \right\} \tag{B.5}$$

$$\|u\|_{L_0^p} = \inf_{\alpha\in\mathbb{R}} \|u+\alpha\|_{L^p} \tag{B.6}$$

Definition 5 (Time-dependent Lebesgue space [25]). *$L^p([0,T],X)$ is a Banach space consists of time-dependent L^p-functions $u : [0,T] \to X$ such that*

$$\|u\|_{L^p([0,T],X)} = \left(\int_0^T \|u(t)\|_X^p \,dt \right)^{\frac{1}{p}} \qquad 1 \leq p < \infty \tag{B.7a}$$

$$\|u\|_{L^\infty([0,T],X)} = \sup_{0\leq t\leq T} \|u(t)\|_X \qquad p = \infty \tag{B.7b}$$

B.2 Sobolev space

Definition 6 (Sobolev space $W^{k,p}$ [1]). *Let $p \in [1,\infty)$ be the Lebesgue index, and k is the derivative order. The Sobolev space $W^{k,p}(\Omega)$ consists of all functions $u \in L^p(\Omega)$ such that the $k-1$ first partial derivatives are absolutely continuous, and the k-th derivative is Lebesgue measurable. When $1 \leq p < \infty$, the seminorm and norm are defined as*

$$|u|_{W^{k,p}} = \left(\sum_{|\alpha|=k} \|D^\alpha u\|_{L^p}^p \right)^{\frac{1}{p}} \tag{B.8a}$$

$$\|u\|_{W^{k,p}} = \left(\sum_{|\alpha|=0}^{k} \|D^\alpha u\|_{L^p}^p \right)^{\frac{1}{p}} = \left(\sum_{|\alpha|=0}^{k} |u|_{W^{k,p}}^p \right)^{\frac{1}{p}} \tag{B.8b}$$

For $p = \infty$, the seminorm and norm become

$$|u|_{W^{k,\infty}} = \max_{|\alpha|=k} \|D^\alpha v\|_{L^\infty} \tag{B.9a}$$

$$\|u\|_{W^{k,\infty}} = \max_{|\alpha|\le k} \|D^\alpha v\|_{L^\infty} \tag{B.9b}$$

$W^{k,p}(\Omega)$ is a Banach space. If $p = 2$, we have a Hilbert space H^k with the inner product

$$\langle u, v\rangle_{H^k} = \sum_{|\alpha|=0}^{k} \langle D^\alpha u, D^\alpha v\rangle_{L^2} \tag{B.10}$$

Theorem 5 (Inequalities for $W^{k,p}$ [1])**.** *If Ω has finite measure and $u \in W_0^{1,p}$, then we have the following inequalities in the Sobolev space:*

Poincaré's inequality: $\quad \|u\|_{L^p(\Omega)} \le C_\Omega |u|_{W^{1,p}(\Omega)} \quad$ (B.11a)

Gagliardo-Nirenberg inequality: $\quad \|u\|_{L^q(\mathbb{R}^n)} \le C\|u\|_{L^p(\mathbb{R}^n)} \quad$ (B.11b)

General Sobolev inequality: $\quad \|u\|_{L^q(\Omega)} \le C\|u\|_{W^{k,p}(\Omega)} \quad$ (B.11c)

Theorem 6 (Completeness, inclusion and reflexivity properties of $W^{k,p}$ [1])**.** *The space $W^{k,p}$ is the completion of C^k in the $W^{k,p}$-norm:*

$$W^{k,p} \equiv \overline{C^k}^{\|\cdot\|_{W^{k,p}}}$$

If Ω is a domain of any measure, then for $1 \le p < \infty$, we have the inclusion

$$\cdots \subset W^{3,p}(\Omega) \subset W^{2,p}(\Omega) \subset W^{1,p}(\Omega) \subset L^p(\Omega)$$

If k is fixed, Ω has finite measure, and $1 < p < q < \infty$, then the following inclusion holds:

$$W^{k,q}(\Omega) \subset W^{k,p}(\Omega) \subset W^{k,1}(\Omega)$$

$W^{k,p}$ is reflexive iff p and q are conjugate exponents satisfying $1 < p < q < \infty$:

$$(W^{k,q})^* = W^{k,p}$$

Definition 7 (Sobolev space $W_0^{k,p}$ [1])**.** *The space $W_0^{k,p}(\Omega)$ consists of functions $u \in W^{k,p}(\Omega)$ such that $\{f^i : 1 \leq i \leq k\}$ equal zero on the boundary $\partial\Omega$:*

$$W_0^{k,p} \equiv \overline{C^\infty}^{\|\cdot\|_{W^{k,p}}}$$

If $1/p + 1/q = 1$, then the dual space is $W^{-k,q}(\Omega)$ equipped with the supremum norm

$$\|v\|_{W^{-k,q}} = \sup_{u \in W_0^{m,p}, \|u\|_{W^{k,p}} \leq 1} |\langle u, v\rangle| \tag{B.12}$$

These special Sobolev spaces satisfy a very special inclusion:

$$W_0^{k,p}(\Omega) \subset L^p(\Omega) \subset W^{-k,q}(\Omega) \tag{B.13}$$

Definition 8 (The time-dependent Sobolev space [25])**.** *$W^{k,p}([0,T], X)$ consists of time-dependent $W^{k,p}$-functions $u : [0,T] \to X$ such that*

$$\|u\|_{W^{k,p}([0,T],X)} = \left(\int_0^T \sum_{i=0}^{k} \|u^{(i)}(t)\|_X^p \, dt \right)^{\frac{1}{p}} \qquad 1 \leq p < \infty \tag{B.14a}$$

$$\|u\|_{W^{k,\infty}([0,T],X)} = \max_{i \leq k} \|u^{(i)}(t)\|_{L^\infty} \qquad p = \infty \tag{B.14b}$$

Definition 9 (Convergence types)**.** *Let $\{u_n\}_n$ be a sequence in a Banach space X, and $\{f_n\}_n$ is a sequence in the dual space X'. Then we can define the following types of convergence for these elements.*

Strong convergence:	$u_n \longrightarrow u$
Weak convergence:	$f(u_n) \longrightarrow f(u)$
Weak- convergence:*	$f_n(u) \longrightarrow f(u)$

Appendix C

Finite element analysis

Definition 10 (Functionals and forms [53]). *A functional is an operator $F(\cdot) : V \to \mathbb{R}$, assigning a real number to each element in V. We call it* [1]

Linear:	$F(\lambda u + \mu v) = \lambda F(u) + \mu F(v)$
Bounded:	$\|F(u)\| \leq M\|u\|_V$

A form is an operator $a(\cdot,\cdot) : V \times V \to \mathbb{R}$, assigning a real number to a pair of elements in V. For any $u, v \in V$ and $\lambda, \mu \in \mathbb{R}$, we call it

Bilinear:	$a(u, \lambda v + \mu w) = \lambda a(u, v) + \mu a(u, w)$
	$a(\lambda u + \mu v, w) = \lambda a(u, w) + \mu a(v, w)$
Continuous:	$\|a(u, v)\| \leq M\|u\|_V\|v\|_V$
Coercive:	$\alpha\|u\|_V^2 \leq a(u, u)$
Symmetric:	$a(u, v) = a(v, u)$
Positive:	$a(u, u) > 0$

Theorem 7 (Lax-Milgram theorem [25]). *Let a finite element problem be given by*

$$\text{find } u \in V : a(u, v) = F(v), \qquad \forall v \in V.$$

If V is a Hilbert space, $a(\cdot,\cdot)$ is a continuous and coercive bilinear form, and $F(\cdot)$ is a linear and continuous functional, then the problem above has a unique solution.

Definition 11 (Regular bilinear form [60]). *Let $C(\cdot,\cdot) : A \times B \to \mathbb{R}$ such that $A, B \subset X$ have dimension n. A and B are the test and trial spaces, respectively. If*

$$\begin{aligned} &\forall u \in A, u \neq 0, \quad \exists v \in B : C(u,v) \neq 0 \\ &\forall v \in B, v \neq 0, \quad \exists u \in A : C(u,v) \neq 0 \end{aligned}$$

then $C(\cdot,\cdot)$ is regular, and the discretization matrix arising from C is nonsingular.

Definition 12 (Boundary conditions [25]). *Let $\Omega \subset \mathbb{R}^n$ be a well-defined domain with a regular boundary $\partial\Omega$. For any 2nd order partial differential equation, we have the following general boundary conditions*

$$\begin{aligned} &\textit{Dirichlet conditions:} && u = g_D \quad , \quad u \in \Gamma_D \\ &\textit{Neumann conditions:} && \frac{\partial u}{\partial n} = g_N \quad , \quad u \in \Gamma_N \\ &\textit{Robin conditions:} && \alpha \frac{\partial u}{\partial n} + \beta u = g_R \quad , \quad u \in \Gamma_R \end{aligned}$$

We assume that $\partial\Omega = \Gamma_D \cup \Gamma_N \cup \Gamma_R$, and all the boundary segments are mutually disjoint.

Appendix D

Spline algorithms

In the whole implementation, several algorithms have been used for increasing the speed of the assembly process and post-processing. Some of them are slight generalizations of already existing methods for evaluation of splines. Hence, we will therefore present the most important ones in a general way that can be implemented in any language.

The first algorithm is creating a knot vector for representing a spline of degree p with continuity C^α on a closed interval I. This method is constructed in such a way that all the interior knots in the final vector are sorted in a nondecreasing way, and the end knots have multiplicity $p+1$. The continuity will automatically define the multiplicity of the interior knots, and the resulting knot vector is uniform.

Algorithm 2 Generating knot vectors

```
 1: procedure GENERATE_KNOT_VECTOR($p$, $\alpha$)
 2:     $I \leftarrow$ Sorted, increasing array with $s$ unique knots of multiplicity 1
 3:     $k = p - \alpha$
 4:     $e = 2(p+1) + (s-2)k$
 5:     $\Xi \leftarrow$ Array of $e$ zeros
 6:     for $k = 1$ to $p$ do
 7:         $\Xi_i = I_1$
 8:         $\Xi_{e-p-1+i} = I_s$
 9:     for $i = 1$ to $s - 2$ do
10:         for $j = 1$ to $k$ do
11:             $\Xi_{p+1+(i-1)k+j} = I_{i+1}$
12:     return $\Xi$
```

Algorithm 3 Evaluation of a B-spline

```
 1: procedure EVALUATE($\Xi$, $p$, $t$)
 2:     $R$ ← Array of $2p + 1$ zeros
 3:     for $k = 1$ to $p$ do
 4:         $NV$ ← Array of $2p + 1 - k$ zeros
 5:         for $i = 1$ to $2p + 1 - k$ do
 6:             $LBF = R_i$
 7:             $a = \Xi_{i+k} - \Xi_i$
 8:             if $a > 0$ then
 9:                 $LRU = (t - \Xi_i)/a$
10:             else
11:                 $LRU = 0$
12:             $RBF = R_{i+1}$
13:             $a = \Xi_{i+k+1} - \Xi_{i+1}$
14:             if $a > 0$ then
15:                 $RRD = (\Xi_{i+k+1} - t)/a$
16:             else
17:                 $RRD = 0$
18:             $NV_i = LBF * LRU + RBF * RRD$
19:         $R = NV$
20:     return $R$
```

Algorithm 4 First derivative of a B-spline

```
 1: procedure DERIVATIVE_1($\Xi$, $p$, $t$)
 2:     $C$ ← Array of $p$ zeros
 3:     for $i = 1$ to $p$ do
 4:         $r = \Xi_{i+p+1} - \Xi_{i+1}$
 5:         if $r = 0$ then
 6:             $C_i = 0$
 7:         else
 8:             $C_i = p/r$
 9:     $E =$ EVALUATE($\Xi, p, t$)
10:     $R$ ← Array of $p + 1$ zeros
11:     $R_1 = -C_1/E_1$
12:     for $i = 1$ to $p - 1$ do
13:         $R_{i+1} = C_i E_i - C_{i+1} E_{i+1}$
14:     $R_{p+1} = C_{p+1} E_{p+1}$
15: return $R$
```

Algorithm 5 Second derivative of a B-spline

```
1:  procedure DERIVATIVE_2($\Xi$, $p$, $t$)
2:      $C \leftarrow$ Array of $p$ zeros
3:      $D \leftarrow$ Array of $p+1$ zeros
4:      for $i = 1$ to $p+1$ do
5:          if $i \leq p$ then
6:              $r = \Xi_{i+p+1} - \Xi_{i+1}$
7:              if $r \neq 0$ then
8:                  $C_i = 1/r$
9:          $s = \Xi_{i+p} - \Xi_{i+1}$
10:         if $s \neq 0$ then
11:             $D_i = 1/s$
12:     $e = p + 1$
13:     $R \leftarrow$ Array of $e$ zeros
14:     if $p = 2$ then
15:         $R_1 = C_1 D_2$
16:         $R_2 = -(C_1 + C_2) D_2$
17:         $R_3 = C_2 D_2$
18:     else if $p = 3$ then
19:         $E$ = EVALUATE($\Xi_{3:e-2}, 1, t$)
20:         $R_1 = C_1 D_2 E_1$
21:         $R_2 = -(C_1 + C_2) D_2 E_1 + C_2 D_3 E_2$
22:         $R_3 = C_2 D_2 E_1 - (C_2 + C_3) D_3 E_2$
23:         $R_4 = C_3 D_3 E_1$
24:     else if $p \geq 4$ then
25:         $E$ = EVALUATE($\Xi_{3:e-2}, p-2, t$)
26:         $R_1 = C_1 D_2 E_1$
27:         $R_2 = -(C_1 + C_2) D_2 E_1 + C_2 D_3 E_2$
28:         for $i = 2$ to $p-2$ do
29:             $R_{i+1} = C_i D_i E_{i-1} - (C_i + C_{i+1}) D_{i+1} E_i + C_{i+1} D_{i+2} E_{i+1}$
30:         $R_{e-1} = C_{e-1} D_{e-2} E_{e-1} - (C_{e-1} + C_e) D_{e-1} E_e$
31:         $R_e = C_{e-1} D_{e-1} E_e$
32:     $R = p(p-1)R$
33: return $R$
```

Algorithm 3, 4 and 5 are optimal because they evaluate all the B-splines on an interval $I_0 \subset I$ such that $t \in I_0$. This is crucial in the assembly process because we loop over several elements, and we need to evaluate *all* the B-splines that belong to this specific element. When the element matrix or vector has been found by taking tensor product, they are inserted into the global matrices and vectors.

Algorithm 6 Evaluation on a surface

```
1: procedure EVALUATE_SURFACE($\Xi$, $p_1$, $\overline{\xi}$, $\mathcal{H}$, $p_2$, $\overline{\eta}$)
2:     Find the knot span such that $\overline{\xi} \in [\xi_i, \xi_{i+1}]$.
3:     Evaluate the basis functions $N_{i-p,p}(\overline{\xi}), \dots, N_{i,p}(\overline{\xi})$.
4:     Find the knot span such that $\overline{\eta} \in [\eta_j, \eta_{j+1}]$.
5:     Evaluate the basis functions $M_{j-q,q}(\overline{\eta}), \dots, M_{j,q}(\overline{\eta})$.
6:     Multiply the function values with the corresponding control points.
```

$$\mathbf{S}(\overline{\xi}, \overline{\eta}) = \left[N_{a,p}(\overline{\xi})\right]^T [\mathbf{P}_{ab}] \left[M_{b,q}(\overline{\eta})\right] \qquad \begin{cases} i-p \leq a \leq i \\ j-q \leq b \leq j \end{cases} \tag{D.1}$$

The same method is applied to the partial derivative:

$$\frac{\partial^{s_1+s_2}}{\partial \xi^{s_1} \partial \eta^{s_2}} \mathbf{S}(\xi, \eta) = \left[N_{a,p}^{(s_1)}(\overline{\xi})\right]^T [\mathbf{P}_{ab}] \left[M_{b,q}^{(s_2)}(\overline{\eta})\right] \qquad \begin{cases} 0 \leq s_1 + s_2 \leq d \\ i-p \leq a \leq i \\ j-q \leq b \leq j \end{cases} \tag{D.2}$$

Bibliography

[1] Robert A. Adams and John J. F. Fournier. *Sobolev Spaces*. Elsevier, Amsterdam, 2003.

[2] John Argyris, Gunter Faust, Maria Haase, and Rudolf Friedrich. *An Exploration of Dynamical Systems and Chaos*. Springer-Verlag, Berlin, 2015.

[3] Uri M. Ascher, Steven J. Ruuth, and Raymond J. Spiteri. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Appl. Numer. Math.*, 25:151–167, 1997.

[4] Uri M. Ascher, Steven J. Ruuth, and Brian T. R. Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM J. Numer. Anal.*, 32:797–823, 1995.

[5] Daniele Boffi, Franco Brezzi, and Michel Fortin. *Mixed Finite Element Methods and Applications*. Springer-Verlag, Berlin, 2013.

[6] Franck Boyer and Pierre Fabrie. *Mathematical Tools for the Study of the Incompressible Navier-Stokes Equations and Related Models*. Springer-Verlag, New York, 2013.

[7] A. Bressan and G. Sangalli. Isogeometric discretizations of the Stokes problem: stability analysis by the macroelement technique. *IMA Journal of Numerical Analysis*, 33:629–651, 2013.

[8] Alexander N. Brooks and Thomas J. R. Hughes. Streamline-Upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 32:199–259, 1982.

[9] A. Buffa, C. de Falco, and G. Sangalli. Isogeometric analysis: Stable elements for the 2D Stokes equation. *Int. J. Numer. Meth. Fluids*, 65:1407–1422, 2010.

[10] A. Buffa, J. Rivas, G. Sangalli, and R. Vazquez. Isogeometric discrete differential forms in three dimensions. *SIAM J. Numer. Anal.*, 49:818–844, 2011.

[11] John Butcher. Implicit Runge-Kutta processes. *Math. Comput.*, 18:50–64, 1964.

[12] J. Austin Cottrell, Thomas J. R. Hughes, and Yuri Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley, UK, 2009.

[13] G. D. V. Davis. Natural convection of air in a square cavity: A bench mark solution. *Int. J. Numer. Meth. Fluids*, 3:249–264, 1983.

[14] Carl de Boor. *A Practical Guide to Splines*. Springer-Verlag, New York, 2001.

[15] Luca Dedé and Alfio Quarteroni. Isogeometric analysis for second order partial differential equations on surfaces. *Comput. Methods Appl. Mech. Engrg.*, 284:807–834, 2015.

[16] M. O. Deville, E. F. Fischer, and E. H. Mund. *High-Order Methods for Incompressible Fluid Flow*. Cambridge University Press, UK, 2005.

[17] Hans-Jörg G. Diersch. *FEFLOW: Finite Element Modeling of Flow, Mass and Heat Transport in Porous and Fractured Media*. Springer-Verlag, Berlin, 2014.

[18] P. G. Drazin and W. H. Reid. *Hydrodynamic Stability*. Cambridge University Press, UK, 2004.

[19] Weinan E. *Principles of Multiscale Modeling*. Cambridge University Press, UK, 2011.

[20] Herbert Egger. Energy norm error estimates for finite element discretization of parabolic problems. Technical report, Department of Mathematics, TU Darmstadt, Germany, 2015.

[21] Howard C. Elman, David J. Silvester, and Andrew J. Wathen. *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*. Oxford University press, UK, 2014.

[22] D. C. Wan et al. A new benchmark quality solution for the buoyancy-driven cavity by discrete singular convolution. *Numerical Heat Transfer*, 40:199–228, 2001.

[23] Marco Donatelli et al. Robust and optimal multi-iterative techniques for IGA Galerkin linear systems. *Comput. Methods Appl. Mech. Engrg.*, 284:230–264, 2015.

[24] T. W. Sederberg et al. T-splines and T-NURCCs. *ACM Trans. Graph.*, 22:477–484, 2003.

[25] Lawrence Evans. *Partial differential equations*. American Mathematical Society, Providence, USA, 2010.

[26] Gerald Farin. *NURBS from Projective Geometry to Practical Use*. A K Peters Ltd, Massachusetts, 1999.

[27] D.R. Forsey and R.H. Bartels. Hierarchical B-spline refinement. *Comput. Graph.*, 22:205–212, 1988.

[28] D. Gray and A. Giorgini. The validity of the Boussinesq approximation for liquid and gases. *International Journal of Heat and Mass Transfer*, 19:545–551, 1976.

[29] Wolfgang Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Springer-Verlag, Berlin, 2012.

[30] Bjarte Hægland. *Computational methods for handling incompressible fluid flows involving internal density interfaces and boundary layers*. Phd-thesis, Norwegian University of Science and Technology, Trondheim, Norway, 6 2006.

[31] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration*. Springer-Verlag, Berlin, 2006.

[32] Ernst Hairer, Syvert Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I*. Springer-Verlag, Berlin, 1993.

[33] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II*. Springer-Verlag, Berlin, 1996.

[34] Charles Hirsch. *Numerical computation of internal and external flows: introduction to the fundamentals of computational fluid dynamics*. Elsevier, UK, 2007.

[35] Thomas J. R. Hughes. *The finite element method*. Prentice-Hall, USA, 1987.

[36] Thomas J. R. Hughes, Yuri Bazilevs, and J. A. Cottrell. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Methods Appl. Mech. Engrg.*, 194:4135–4195, 2005.

[37] Thomas J. R. Hughes, A. Reali, and G. Sangalli. Efficient quadrature for NURBS-based isogeometric analysis. *Comput. Methods Appl. Mech. Engrg.*, 199:301–313, 2008.

[38] Willem Hundsdorfer and Jan Verwer. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*. Springer-Verlag, Berlin, 2003.

[39] Kjetil André Johannessen. Optimal quadrature rules for smooth spline spaces on arbitrary knot vectors. *Comput. Methods Appl. Mech. Engrg.*, To appear, 2016.

[40] Kjetil André Johannessen, Trond Kvamsdal, and Tor Dokken. Isogeometric analysis using LR B-splines. *Comput. Methods Appl. Mech. Engrg.*, 269:471–514, 2014.

[41] Kjetil André Johannessen, Filippo Remonato, and Trond Kvamsdal. On the similarities and differences between classical hierarchical, truncated hierarchical and LR B-splines. *Comput. Methods Appl. Mech. Engrg.*, 291:64–101, 2015.

[42] Mukesh Kumar, Trond Kvamsdal, and Kjetil André Johannessen. Simple a posteriori error estimators in adaptive isogeometric analysis. *Computers and Mathematics with Applications*, 70:1555–1582, 2015.

[43] William E. Langlois and Michel O. Deville. *Slow Viscous Flow*. Springer-Verlag, Switzerland, 2014.

[44] John M. Lee. *Introduction to Smooth Manifolds*. Springer-Verlag, New York, 2013.

[45] Roland W. Lewis, Perumal Nithiarasu, and Kankanhalli N. Seetharamu. *Fundamentals of the Finite Element Method for Heat and Fluid Flow*. Wiley, UK, 2004.

[46] Daryl Logan. *A First Course in the finite Element Method*. Thomson Canada Limited, Toronto, 2007.

[47] M. Manzari. An explicit finite element algorithm for convection heat transfer problems. *International Journal of Numerical Methods for Heat & Fluid Flow*, 9(8):860–877, 1999.

[48] Jerrold Marsden and Anthony Tromba. *Vector Calculus*. W. H. Freeman and company, New York, 2012.

[49] Duncan Marsh. *Applied Geometry for Computer Graphics and CAD*. Springer-Verlag, London, 2005.

[50] Gheorghe Micula and Sanda Micula. *Handbook of Splines*. Springer-Verlag, Netherlands, 1999.

[51] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer-Verlag, New York, 2006.

[52] Les Piegl and Wayne Tiller. *The NURBS Book*. Springer-Verlag, Berlin, 1997.

[53] Alfio Quarteroni. *Numerical Models for Differential Problems*. Springer-Verlag, Milan, 2014.

[54] Alfio Quarteroni and Alberto Valli. *Numerical Approximation of Differential Differential Equations*. Springer-Verlag, Berlin, 2008.

[55] J. N. Reddy. *An Introduction to Continuum Mechanics*. Cambridge University Press, UK, 2008.

[56] J. N. Reddy and D. K. Gartling. *The Finite Element Method in Heat Transfer and Fluid Dynamics*. CRC Press, Boca Raton, Florida, 2010.

[57] Hans-Görg Roos, Martin Stynes, and Lutz Tobiska. *Robust Numerical Methods for Singularly Perturbed Differential Equations*. Springer-Verlag, Berlin, 2008.

[58] Sandro Salsa. *Partial Differential Equations in Action*. Springer-Verlag, Milan, 2015.

[59] Larry Schumaker. *Spline Functions, Computational Methods*. Society for Industrial and Applied Mathematics, Philadelphia, 2015.

[60] Christoph Schwab. *p- and hp-Finite Elements Methods*. Oxford University press, UK, 1998.

[61] Hermann Sohr. *The Navier-Stokes Equations*. Springer-Verlag, Switzerland, 2001.

[62] O. C. Zienkiewicz, R. L. Taylor, and P. Nithiarasu. *The Finite Element Method for Fluid Dynamics*. Elsevier, Massachusetts, 2013.

[63] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method: Its basis and fundamentals*. Elsevier, Massachusetts, 2013.