Master Thesis

Towards an Efficient Multigrid Algorithm for Solving Pressure-Robust Discontinuous Galerkin Formulations of the Stokes Problem

by
Thomas de Lange

Faculty of Engineering Technology
Program of Mechanical Engineering
Chair of Engineering Fluid Dynamics

Graduation committee

prof.dr.ir. C.H. Venner	(chair)
dr.ir. E.T.A. van der Weide	(supervisor)
prof.dr.ir. H. Askes	(external member)
dr. P. Ströer	(internal member)

Document number: 436

October, 2023

UNIVERSITY OF TWENTE.

Preface

With the conclusion of this Master's thesis, my time at the University of Twente comes to an end. A little over eight years ago I started my journey towards becoming an engineer, of which the last five years - including a gap year - have been all about combining my professional sports career with my academic interests. This setting was made possible by the personal arrangements provided by the University of Twente, such as the "topsportregeling", which I am very grateful for.

When looking for a Master's assignment, I spoke to several people to discuss the options, including dr.ir. E.T.A. van der Weide. He and prof.dr.ir. C.H. Venner were lecturers for the Master's course on computational fluid dynamics (CFD), which I thoroughly enjoyed. In early 2022, I had also done a Capita Selecta assignment under the supervision of dr.ir. E.T.A. van der Weide about solving a given high-order discretization of the incompressible Navier-Stokes equations using Krylov subspace methods. As a follow up to this Capita Selecta assignment, he proposed to research the possibilities of a multigrid algorithm for the high-order discretization of the Stokes problem. As this assignment was a combination of my interests in fluid dynamics, computational science and programming, I chose to embark on this journey.

I would like to thank dr.ir. E.T.A. van der Weide for his supervision during the assignment. I really appreciated the weekly conversations we had, in which he spent many hours teaching me all about numerical methods, solution algorithms and debugging. Even outside of our scheduled meetings, he was always willing to help me. At times where I could not manage to see the bigger picture, his vision guided me in the right direction again. Furthermore, I want to thank him for his interest in my well-being and for guarding my personal boundaries.

I also want to thank my parents, sisters, family and friends for their support during this period of time. Thank you for your believing in me and for putting everything into perspective at times where I felt worried and frustrated.

Lastly, I hope you, the reader, enjoy the reading!

October, 2023,

Thomas de Lange

Summary

For the numerical simulation of incompressible turbulent flows using direct numerical simulation (DNS) or large eddy simulation (LES), solving the Stokes problem efficiently is an important component. In this context, high-order discretization methods such as the discontinuous Galerkin (DG) method receive increasing attention because of their high accuracy, low numerical dissipation and superior convergence properties. However, solving DG formulations of the Stokes problem efficiently is a significant challenge due to the structure and characteristics of the discrete system.

In theory, a multigrid algorithm should be capable of solving the problem in an amount of work that is only proportional to the problem size, making it an attractive candidate as a solver for the DG discretization of the Stokes problem. In this thesis, a multigrid algorithm that can be applied to the DG discretization of the Poisson problem is developed, implemented and analysed. In addition, the DG discretization and a suitable smoother for the Stokes problem are derived, implemented and verified. These three topics form the building blocks for the development of an efficient multigrid algorithm for the DG formulation of the Stokes problem

The implementation of the discretization of the Stokes problem could be verified on a Cartesian grid, while some unexpected results were found on curvilinear grids. Further research should point out if this is due to a implementation error or a fundamental problem in the discretization. Moreover, a multigrid algorithm based on polynomial and geometric coarsening was found to be very effective, it was capable of solving the Poisson problem in an amount cycles that was independent of the problem size. Lastly, a distributive Gauss-Seidel smoother based on the least-squares commutator was found to be capable of smoothing the Stokes system effectively, provided that the order of the discretization was not higher than fourth order. Further research is needed to investigate the cause of this limitation. However, it could also be useful to research if an alternative splitting can be defined which does not have this limitation.

Nomenclature

List of abbreviations

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
CFD	computational fluid dynamics
CG	continuous Galerkin
CS	correction scheme
DG	discontinuous Galerkin
DNS	direct numerical simulation
DOF	degree of freedom
DSS	direct stiffness summation
FAS	full approximation scheme
FE	finite element
FV	finite volume
IMEX	implicit-explicit
LES	large eddy simulation
LGL	Legendre-Gauss-Lobatto
MMS	method of manufactured solutions
PDE	partial differential equation
RANS	Reynolds-averaged Navier-Stokes
RHS	right-hand side
SIMPLE	semi-implicit pressure-linked equations
SIP	symmetric interior penalty
SPD	symmetric positive definite

List of symbols

α	Jacobi polynomial constant	-
β	Jacobi polynomial constant	-
N	Number of elements/nodes/DOFs (depending on subscript)	-
p	Polynomial order	-
$\boldsymbol{\mathcal{A}}$	Coefficient matrix	-
$\boldsymbol{B}$	Iteration matrix	-

$\boldsymbol{b}$	RHS vector	multiple
$\boldsymbol{\mathcal{D}}_r$	Differentiation matrix w.r.t. r	$1/\mathrm{m}$
$\boldsymbol{\mathcal{D}}_s$	Differentiation matrix w.r.t. s	$1/\mathrm{m}$
e	Element | error	- | multiple
$\mathcal{E}$	Consistency correction	multiple
$\mathcal{F}$	Set of faces	-
$\mathbf{f}$	External force per unit mass vector	$\mathrm{m/s^2}$
f	Face	-
$\mathbf{g}_{\mathcal{D}}$	Dirichlet boundary condition vector	$\mathrm{m/s}$
γ	Stabilization parameter	$\mathrm{m^2/s}$
Γ	Boundary of the domain	multiple
h_f	Length scale of a face	m
$\boldsymbol{J}$	Jacobian transformation matrix	m
ℓ	Nodal basis function | Lagrange polynomial	-
L	Lagrange polynomial | Lebesgue function	-
L_2	Error norm	multiple
Λ	Lebesgue constant	-
$\boldsymbol{M}$	Mass matrix	multiple
μ	Amplification factor	-
$\mathbf{n}$	Unit outward normal	-
ν	Kinematic viscosity | number of relaxation sweeps	$\mathrm{m^2/s}$ | -
Ω	Domain	multiple
p	Kinematic pressure	$\mathrm{m^2/s^2}$
P	Jacobi polynomial	-
$\tilde{P}$	Orthonormal Jacobi polynomial	-
ψ	Modal basis function | Legendre polynomial	-
$\mathcal{Q}_h$	Pressure test space	-
q_h	Pressure test function	-
$\mathcal{R}$	Residual	multiple
ρ	Density | spectral radius	$\mathrm{kg/m^3}$ | -
r	Parametric coordinate	-
s	Parametric coordinate	-
σ	Penalty parameter	$\mathrm{m^2/s}$
t	Time	s
$\mathbf{u}$	Velocity vector	$\mathrm{m/s}$
u	Component of the velocity vector in x-direction	$\mathrm{m/s}$
v	Component of the velocity vector in y-direction	$\mathrm{m/s}$
$\boldsymbol{V}$	Vandermonde matrix	-
$\mathcal{V}_h$	(Velocity) test space	-
v_h	(Velocity) test function	-

$\boldsymbol{V}_r$	Derivative of Vandermonde matrix w.r.t. r	1/m
$\boldsymbol{V}_s$	Derivative of Vandermonde matrix w.r.t. s	1/m
w	Integration weight	-
$\boldsymbol{x}$	Spatial coordinate vector	m
x	Spatial coordinate	m
$\boldsymbol{\xi}$	Solution vector	multiple
y	Spatial coordinate	m

List of operators

∇_h	Broken gradient operator
$\nabla_h\cdot$	Broken divergence operator
Δ_h	Broken Laplace operator
$[\![\cdot]\!]$	Jump operator
$\{\!\{\cdot\}\!\}$	Average operator
$\boldsymbol{\mathcal{G}}$	Gradient operator
$\boldsymbol{\mathcal{D}}$	Divergence operator
$\boldsymbol{I}_F^C$	Restriction operator
$\boldsymbol{I}_C^F$	Prolongation operator

List of subscripts and superscripts

∂	Of the boundary
C	Of the coarse grid
e	Of the element
F	of the fine grid
f	Of the face
$grid$	Of the grid
H	Of the coarse grid
h	Numerical | of the fine grid
I	Of the grid standard element
i	One-dimensional index | internal
int	Of the/in the integration nodes
j	One-dimensional index
k	Two-dimensional index | iteration number
n	Of the neighbour

Notation: vectors and matrices are denoted with bold symbols.

Contents

Chapter 1

Introduction

Motivation

Turbulent flows are present everywhere in life, as most flows that occur in nature or flows that are a result of engineering applications are turbulent. All flows become turbulent at high Reynolds numbers, which is the dimensionless number that describes the ratio between the inertial and viscous forces. Depending on the flow, the transition to the turbulent regime happens for Reynolds numbers anywhere between 2000 and 1 million, [Bak08]. Turbulent flows are highly chaotic and involve a wide range of time and length scales. An example is the eruption of a volcano as illustrated in figure 1.1, which clearly shows the difference

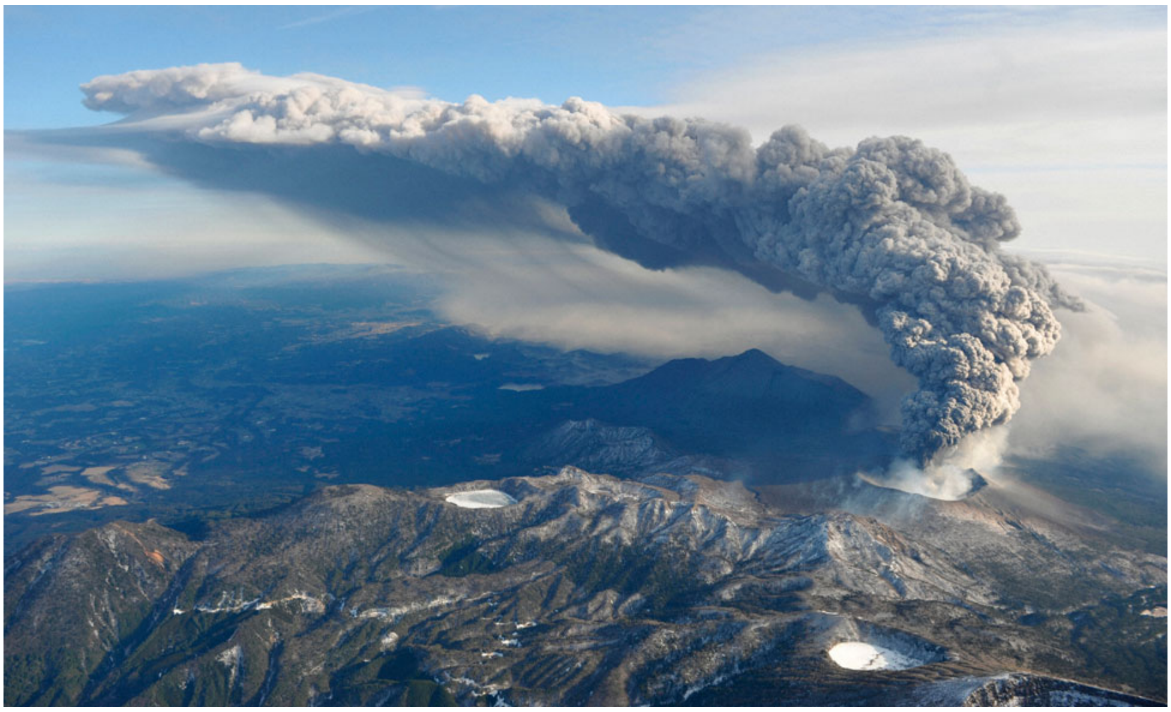


Figure 1.1: A volcano eruption in 2011 on Mount Kirishima, Japan, creating a cloud of smoke and ash up to around 1500 meters into the air, [Dai11; US 11].

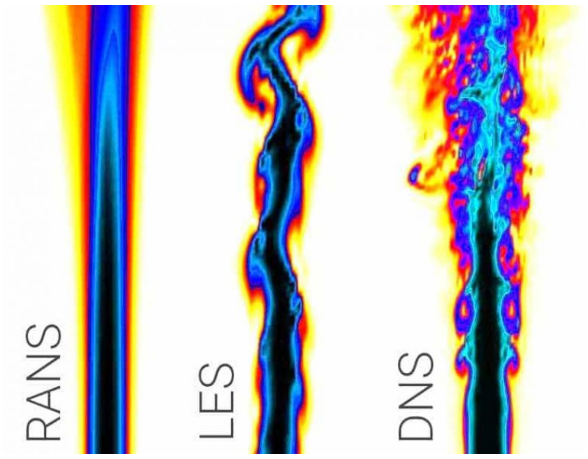


Figure 1.2: A simulation of a turbulent jet using RANS, LES and DNS, [ges19].

in scales. Other examples are all kinds of weather phenomena, oceanic and atmospheric mixed layers, external flows over aircraft, cars, ships and wind turbines, but also blood flow in a partially obstructed artery and many more. Therefore, the numerical simulation of turbulent flows using computational fluid dynamics (CFD) is of great importance for weather forecasting, navigation, energy generation, cardiology and so on.

The motion of a viscous fluid is governed by the Navier-Stokes equations. Hence, numerical simulation of turbulent flows could in theory be done by solving these equations which is known as direct numerical simulation (DNS). However, because even the smallest turbulent scales need to be captured, the Navier-Stokes equations need to be solved on a fine grid using a small time step, leading to a computational cost that is proportional to the Reynolds number cubed in 3D, [Bak08]. Since turbulent flows occur at high Reynolds numbers, the practical use of DNS is very limited. Therefore, for practical applications, turbulence models must be introduced in which the number of scales that are actually solved are reduced. Although a formal mathematical definition and some of the physical features of turbulence itself still remain unknown, flow features can be deduced from observations and experiments. By using these features, certain scales can be modeled and others can be solved, which is the core of turbulence models.

There exist roughly two approaches to model turbulence, large eddy simulations (LES) and Reynolds-averaged Navier-Stokes (RANS) models. In LES, the large-scale motions are solved whereas the scales smaller than the grid size are modelled. In RANS models, only the mean of the turbulent flow is described by modelling all turbulent scales. The difference between RANS, LES and DNS is visualized in figure 1.2. The difference in the amount of scales that are modelled is clearly visible, ranging from all in RANS to none in DNS. Even though RANS is computationally inexpensive, its applicability is limited to attached flows for which the models are tuned. Moreover, the turbulent scales of a flow solution obtained with RANS only have a statistical meaning, since all scales are modelled. Therefore, the turbulent scales can often only be represented accurately using DNS or LES.

A flow is assumed to be incompressible when the Mach number, i.e. the ratio of the flow velocity and the speed of sound, is below 0.3, [Pan13]. As an example, consider water which has a speed of sound of around $1482\,\mathrm{m/s}$, [The04]. The majority of the flows in water have a flow velocity of a couple meters per second, such that the Mach number approaches zero. When the Mach number approaches zero, solvers that are based on solving the compressible Navier-Stokes equations suffer severe deficiencies in both accuracy as well as efficiency, as the time step that needs to be taken for stability is proportional to the speed of sound of the medium, [KBW04]. Therefore, solutions to low Mach number turbulent flows must be obtained by solving the incompressible Navier-Stokes equations.

Consider the set of incompressible Navier-Stokes equations,

$$\left\{\begin{array}{rcll} \partial_t \mathbf{u} - \nu\Delta\,\mathbf{u} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} + \nabla\,\mathrm{p} &=& \mathbf{f}, & \forall \boldsymbol{x} \in \Omega, \qquad (1.1) \\ \nabla\cdot\mathbf{u} &=& 0, & \forall \boldsymbol{x} \in \Omega, \qquad (1.2) \\ \mathbf{u} &=& \mathbf{g}_{\mathcal{D}}, & \forall \boldsymbol{x} \in \Gamma. \qquad (1.3) \end{array}\right.$$

on the time interval $(0, T)$, with an appropriate set of initial conditions and Dirichlet boundary conditions $\mathbf{g}_{\mathcal{D}}$ prescribed on the boundary Γ of the domain Ω. Moreover, $\partial_t\,\mathbf{u}$ denotes the partial derivative of the velocity vector $\mathbf{u} = (\mathrm{u}, \mathrm{v})^T$ with respect to time, ν is the kinematic viscosity, p the kinematic pressure (pressure divided by density) and $\mathbf{f}$ the external force per unit mass. As a result of the velocity-pressure coupling, the divergence-free constraint $\nabla\cdot\mathbf{u} = 0$ is enforced on the velocity leading to a system that is in saddle point form (see chapter 3 for more details) which must be discretized and integrated in time. Performing the time integration using a fully explicit scheme leads to a very small time step that needs to be taken to ensure stability, [Joh16], while integrating it fully implicitly leads to a computationally expensive non-linear system that needs to be solved for every time step, [Chr10]. For this reason, implicit-explicit (IMEX) time integration schemes are used in which the system is split up into a Stokes part that is treated implicitly and a convective part that is treated explicitly. The IMEX schemes have a time step restriction corresponding to the explicit treatment of the convective terms that is comparable to the time scales that need to be solved in the DNS and LES of turbulence. Using this time integration scheme leads to a symmetric Stokes system with a right-hand side corresponding to the convective terms that needs to be solved in every time step. Therefore, solving the Stokes problem efficiently is an important component for the numerical simulation of turbulence of incompressible flows. Due to the saddle point structure of the problem, this poses a significant challenge in the field of scientific computing.

Several methods exist for the spatial discretization of partial differential equations (PDEs). For example, the well-known finite difference (FD) method is based on a Taylor series expansion of the PDEs in differential form. On the other hand, finite element (FE) and finite volume (FV) methods are based on the integral form of the equations and belong to the class of Galerkin methods. This class contains the continuous Galerkin (CG) FE method (which is often just referred to as the FE method) and the discontinuous Galerkin (DG) FE method. The DG method can be interpreted as an higher-order generalization of the FV method. In contrast to the CG element method, the problem definition of the DG method is formulated locally, leading to a both globally and locally conservative method, [Gir20]. Instead of the constant cell value in the FV method, the local solution is approximated by a polynomial expansion in the DG method. The elements are coupled to each other using the same type of numerical fluxes that have been developed for the FV method.

Recently, turbulence simulation using high-order spatial discretization methods such as DG method has increased in popularity. The DG method has several advantages over FE and FV methods which are currently widely used discretization methods for this type of problem. The high-order accuracy of the DG method leads to a significant reduction in numerical dissipation and dispersion errors compared to FE and FV methods, [Cha+14]. This is an important property because these errors can have a significant impact on the solution of DNS and LES simulations, [Gho96; MM98].

The number of turbulent scales that can be solved using numerical simulation has increased in the past years due to both the availability of more computing power and more efficient solution algorithms. This thesis contributes to this increase with the development of an efficient solution algorithm for DG formulations of the Stokes problem. For elliptic problems such as the Stokes and the Poisson problem, multigrid algorithms have proven to be very effective, [VL00]. In theory, a multigrid algorithm is capable of solving the problem in an amount of work that is only proportional to the problem size. In contrast to for example Krylov methods, the multigrid method requires information about the problem to work well. While Krylov methods can also be preconditioned to incorporate some information about the problem into the solver as was done in [BGL05], its performance is inferior compared to the theoretical performance of a multigrid algorithm for elliptic problems. Therefore, the focus of this thesis is on multigrid methods.

For the multigrid algorithm to work well it is important that the Stokes system possesses good error smoothing properties (see chapter 4 for more details). For an elliptic system of equations, it can be proven that it can be smoothed well when when its determinant possesses good smoothing properties. As the determinant of the Stokes system is the Laplace operator squared in 2D and cubed in 3D, the Poisson equation is used as a model problem for the Stokes problem.

The main contribution of this thesis is to develop, implement and analyze a multigrid algorithm that can be applied to the high-order DG formulation of the Poisson problem. In addition, the DG discretization and a suitable smoother for the Stokes problem are derived, implemented and verified. These three topics form the building blocks for the development of an efficient multigrid algorithm for the DG formulation of the Stokes problem.

The implementation will be done in a computer code written in Python. Even though it is not the fastest programming language, it is very user friendly, has a large (scientific) community and has a lot of scientific computing tools available in libraries such as `numpy` and `scipy`. The aim of this thesis is not to program a very efficient code. In fact, this is a scientific field on its own. In this thesis, computer code is merely used as a tool to research the building blocks of an efficient multigrid solver for the Stokes problem.

Outline

The outline of this thesis is as follows. In chapter 2, the main concepts of the DG method are explained in which the differences between continuous and discontinuous Galerkin methods are highlighted. Furthermore, the theory behind nodal and modal basis functions and the relation between them, numerical integration and the mapping to standard elements in 1D and 2D are addressed.

The characteristics of incompressible flows are explored in chapter 3, followed by the construction of the model problems. Thereafter, a pressure-robust DG discretization is derived with a focus on the practical implementation in a computer code.

Chapter 4 addresses several techniques for obtaining solutions to the discrete system that was formed in chapter 3 with a focus on iterative methods. The concept of relaxation is explained which is needed to understand the core principles of the multigrid algorithm. Lastly, the components that are needed for the construction of a multigrid algorithm are discussed.

In chapter 5, the concepts that are presented in the first chapters are combined to construct the methods with which the results are obtained. Then, the results of the multigrid algorithm for the DG formulation of the Poisson problem as well as the pressure-robust DG discretization and smoother for the Stokes problem are presented and discussed in chapter 6. Thereafter, the conclusions and recommendations for further research are given in chapter 7.

Chapter 2

Discontinuous Galerkin method

The aim of this section is to give insight in the fundamentals of the DG method, with a focus on the practical implementation. For a complete overview of DG, refer to [HW08; Gir20]. First, the key concepts of both the continuous and discontinuous Galerkin methods are explained in section 2.1. After that, the focus of this chapter is shifted to the components that are needed for the implementation of the DG method. The construction of the basis functions is discussed in section 2.2, followed by a description of numerical integration in section 2.3 and a mapping to standard elements in section 2.4. These components are extended for 2D problems in section 2.5.

2.1 Key concepts

In order to explain the theory behind the CG and DG methods, consider the 1D scalar conservation law

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} = 0, \qquad x \in \Omega \tag{2.1}$$

where U is the conserved quantity and the flux is defined as $F = F(U)$. The continuous domain Ω is approximated by N_e non-overlapping elements,

$$\Omega \simeq \Omega_h = \bigcup_{e=0}^{N_e-1} \Omega_h^e. \tag{2.2}$$

In Galerkin methods, the solution is expressed as a polynomial expansion,

$$U_h(x,t) = \sum_{i=0}^{N_{DOF}-1} \tilde{U}_i \Psi_i(x), \tag{2.3}$$

$$F_h(x,t) = F(U_h(x,t)), \tag{2.4}$$

where $\tilde{U}_i$ are the expansion coefficients and Ψ_i are basis functions of some sort. The polynomial expansion is generally of nodal or modal form. However, for now it suffices to know that the basis functions Ψ_i exist and either of these formulations can be used. The nodal and modal formulations and their corresponding basis functions are discussed in more detail in section 2.2.

Now, substituting the expressions of equations (2.3) and (2.4) into the original PDE, equation (2.1), yields

$$\frac{\partial U_h}{\partial t} + \frac{\partial F_h}{\partial x} = \mathcal{R}_h. \tag{2.5}$$

Because a finite dimensional polynomial expansion is used to represent the solution, the semi-discrete equation does not satisfy the original continuous problem exactly. Hence, the residual $\mathcal{R}_h$ is nonzero.

The next step is to determine in which sense $\mathcal{R}_h$ should vanish. From linear algebra it is known that, when representing a higher-dimensional space (U) by a lower-dimensional space (U_h), the induced error ($\mathcal{R}_h$) is minimized in the least-squares sense when it is orthogonal to the lower-dimensional space, [TB97; Gir20]. In other words, the residual $\mathcal{R}_h$ must be orthogonal to the approximation space, which is spanned by the basis functions Ψ_j and can be enforced by the inner product

$$\int_Q \mathcal{R}_h \Psi_j \,\mathrm{d}\Omega = 0, \qquad j = 0, \ldots, N_{DOF} - 1, \tag{2.6}$$

where for now, Q is an arbitrary integration domain. In sections 2.1.1 and 2.1.2 specific choices for Q will be made, depending on the type of FE method. Substituting equation (2.5) in equation (2.6), this can be further worked out to

$$\int_Q \frac{\partial U_h}{\partial t} \Psi_j \,\mathrm{d}\Omega + \int_Q \frac{\partial F_h}{\partial x} \Psi_j \,\mathrm{d}\Omega = 0, \qquad j = 0, \ldots, N_{DOF} - 1. \tag{2.7}$$

Using integration by parts on the second term,

$$\int_Q \frac{\partial U_h}{\partial t} \Psi_j \,\mathrm{d}\Omega + \int_Q \frac{\partial}{\partial x} \left(F_h \Psi_j\right) \mathrm{d}\Omega - \int_Q F_h \frac{\partial \Psi_j}{\partial x} \,\mathrm{d}\Omega = 0, \qquad j = 0, \ldots, N_{DOF} - 1 \tag{2.8}$$

and working out the second term using the fundamental theorem of calculus,

$$\int_Q \frac{\partial U_h}{\partial t} \Psi_j \,\mathrm{d}\Omega + \Big[F_h \Psi_j\Big]_{\partial Q} - \int_Q F_h \frac{\partial \Psi_j}{\partial x} \,\mathrm{d}\Omega = 0, \qquad j = 0, \ldots, N_{DOF} - 1, \tag{2.9}$$

where ∂Q denotes the boundary of Q. Equation (2.9) is referred to as the weak form.

A more general definition of the weak form is obtained by multiplying the residual $\mathcal{R}_h$ with a test function belonging to a test vector space, i.e. $v_h \in \mathcal{V}_h$. In that case, $\mathcal{V}_h$ contains all possible test functions v_h belonging to this space. However, since the numerical solution is expressed as a polynomial expansion with a finite number of unknowns, equations (2.3) and (2.4), the problem is overdetermined when this weak form must hold for all $v_h \in \mathcal{V}_h$. To overcome this, it is required to hold for projections of v_h into the function space spanned by the basis functions Ψ_j.

The distinction between CG and DG methods is made in the way in which equations (2.3), (2.4) and (2.9) should hold, either locally (on Ω_h^e) or globally (on Ω_h).

2.1.1 Continuous Galerkin method

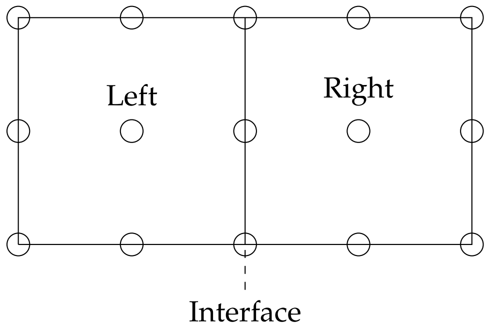


Figure 2.1: The grid points at the interface are shared between the left and right element in the CG method.

The problem statement is global in the CG method. Therefore, the polynomial expansion of the solution reads,

$$U_h(x,t) = \underbrace{\sum_{n=0}^{N_{DOF}-1} \hat{U}_n \psi_n(x)}_{\text{modal}} = \underbrace{\sum_{i=0}^{N_{DOF}-1} U_i \ell_i(x)}_{\text{nodal}}, \tag{2.10}$$

$$F_h(x,t) = F(U_h(x,t)), \tag{2.11}$$

where $\hat{U}_n$ are the expansion coefficients, U_i the physical values in the nodes and ψ_n and ℓ_i are basis functions of some sort. Here, a distinction between nodal and modal formulations is made. More details about these formulations and their basis functions ψ and ℓ will be discussed in section 2.2. While it is possible to use modal formulations in the CG method, nodal formulations are preferred since these enable for continuous functions over element interfaces without the need of additional matching conditions, [Gir20].

Because the CG method is defined globally, the grid points at an element interface are shared between the elements that make up the interface, as depicted in figure 2.1. While it is possible to construct global basis functions, it is more practical to define basis functions with compact support, i.e. they are zero everywhere except inside an element. This enables for direct stiffness summation (DSS) which simply means that the global problem is constructed by summing up the smaller local problems.

Moreover, equation (2.9) is defined globally by taking $Q = \Omega_h$. Combining this with the nodal representation of the solution given in equations (2.10) and (2.11) such that the approximation space is spanned by the basis functions ℓ_j, the weak formulation of the CG method reads

$$\int_{\Omega_h} \frac{\partial U_h}{\partial t} \ell_j \, \mathrm{d}\Omega + \Big[F_h \ell_j \Big]_{\Gamma_h^\partial} - \int_{\Omega_h} F_h \frac{\partial \ell_j}{\partial x} \, \mathrm{d}\Omega = 0, \qquad j = 0, \ldots, N_{DOF} - 1, \tag{2.12}$$

where Γ_h^∂ are the physical boundaries of Ω_h. Since the basis functions are chosen in such a way that they are continuous across element interfaces, the flux (or jump in 1D) vanishes at all internal boundaries. In fact, since boundary conditions are imposed in strong form in the CG method, equation (2.12) reduces to

$$\int_{\Omega_h} \frac{\partial U_h}{\partial t}\psi_j \,\mathrm{d}\Omega - \int_{\Omega_h} F_h \frac{\partial \psi_j}{\partial x}\,\mathrm{d}\Omega = 0, \qquad j = 1, \dots, N_{DOF} - 2. \tag{2.13}$$

when Dirichlet boundary conditions $F(0,t) = F(L,t) = g_{\mathcal{D}}$ are assumed for $F(x,t)$ on the domain $x \in [0, L]$.

2.1.2 Discontinuous Galerkin method

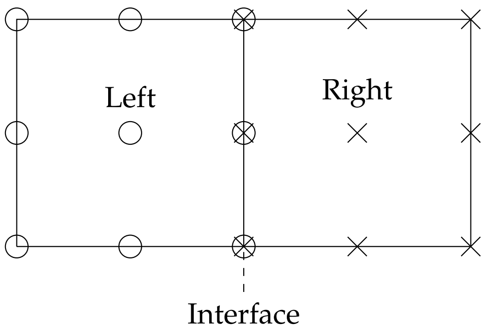


Figure 2.2: The degrees of freedom at the interface are shared between the left and right element in the DG method.

As opposed to the CG method, the DG method is defined locally, leading to a both locally and globally conservative method, [Gir20]. This means that the expansion of the solution is now defined within an element e,

$$U_h^e(x,t) = \underbrace{\sum_{n=0}^{N_{DOF}-1} \hat{U}_n^e \psi_n(x)}_{\text{modal}} = \underbrace{\sum_{i=0}^{N_{DOF}-1} U_i^e \ell_i(x)}_{\text{nodal}}, \tag{2.14}$$

$$F_h^e(x,t) = F(U_h^e(x,t)). \tag{2.15}$$

Requiring equation (2.9) to hold on every element by taking $Q = \Omega_h^e$, using the modal representation given in equations (2.14) and (2.15) such that the approximation space is spanned by the basis functions ψ_j, the weak formulation of the DG method reads

$$\int_{\Omega_h^e} \frac{\partial U_h^e}{\partial t}\psi_j \,\mathrm{d}\Omega + \Big[F_h^e \psi_j\Big]_{\Gamma_h^e} - \int_{\Omega_h^e} F_h^e \frac{\partial \psi_j}{\partial x}\,\mathrm{d}\Omega = 0, \qquad j = 0, \dots, N_{DOF} - 1, \quad \forall e, \tag{2.16}$$

where the boundary Γ_h^e and the domain Ω_h^e are now both defined locally. Equation (2.16) can be further worked out to

$$\int_{\Omega_h^e} \frac{\partial U_h^e}{\partial t}\psi_j \,\mathrm{d}\Omega + \sum_{f \in \mathcal{F}_h^e} \mathbf{n}_f\, F_h^e(x_f)\psi_j(x_f) - \int_{\Omega_h^e} F_h^e \frac{\partial \psi_j}{\partial x}\,\mathrm{d}\Omega = 0, \qquad j = 0, \dots, N_{DOF} - 1, \quad \forall e, \tag{2.17}$$

where f is a face in the set of all faces $\mathcal{F}_h^e$ belonging to the element e. Moreover, $\mathbf{n}$ denotes the unit normal vector at f, pointing outward of e. A big difference with the CG method is that the grid points are now also defined locally instead of globally, as visible in figure 2.2. Because the solution and basis functions are defined within an element, the values at the interface are different when approaching the interface from the left or right element. Therefore, the flux at the (internal) element boundaries does not vanish as was the case for the CG method. This becomes very clear when looking at the flux term in equation (2.17), both F_h^e and ψ_j need to be evaluated at x_f but their values will differ when approaching the interface from the left or right element. While this seems like an unwanted feature, by allowing discontinuous values across element interfaces, knowledge about the problem at hand may be incorporated into the method in the form of an interface flux F_h^*,

$$\int_{\Omega_h^e} \frac{\partial U_h^e}{\partial t} \psi_j \, \mathrm{d}\Omega + \sum_{f \in \mathcal{F}_h^{e,i}} \mathbf{n}_f \, F_h^* \psi_j(x_f) - \int_{\Omega_h^e} F_h^e \frac{\partial \psi_j}{\partial x} \, \mathrm{d}\Omega = 0, \qquad j = 0, \dots, N_{DOF} - 1, \quad \forall e. \tag{2.18}$$

The simplest form of an interface flux would be the mean value of the fluxes at either side of the interface. The construction of numerical fluxes for the discretization of convective terms is very closely related to those used in FV methods. In fact, the DG method can be interpreted as an higher order generalization of the FV method with the obvious difference being that the DG solution is approximated using a polynomial expansion instead of a cell-centered value. This has many advantages, one of which is that many techniques that have been developed for FV methods such as Riemann solvers can be reused in DG methods.

However, since the values at the interface have been allowed to be discontinuous, extra care needs to be taken for the discretization of diffusion terms. This is because in this case only the gradient of the solution is solved for, such that the solution itself is not unique at the interface. This can be resolved by using an interior penalty method, which is further discussed in section 3.3.3.

Thus, the DG method differs in several ways from the CG method, of which the most important are that the problem statement and solution representation are defined locally in the DG method and globally in the CG method. In the next sections, the components that are needed to implement the DG method are discussed.

2.2 Basis functions

Until now, the basis functions were only assumed to exist but no formal definition was given. In the DG method, the solution can be approximated using a nodal or a modal polynomial expansion inside an element e. In its most general form, the local nodal approximation of the function ϕ is defined as the polynomial expansion

$$\phi_h^e(x) = \sum_{i=0}^{p} \phi_i \ell_i(x), \tag{2.19}$$

where the $N_{DOF} = p + 1$ physical values ϕ_i and nodal basis functions ℓ_i are defined at the physical nodes $i = 0, \dots, p$. In practice, cardinal functions are used as a nodal basis, meaning that the function has a value of one in the corresponding node and zero in all other nodes (i.e. Kronecker delta function). An example of a cardinal function is the Lagrange polynomial, which oscillates as an p-th order polynomial between grid nodes, see figure 2.3.

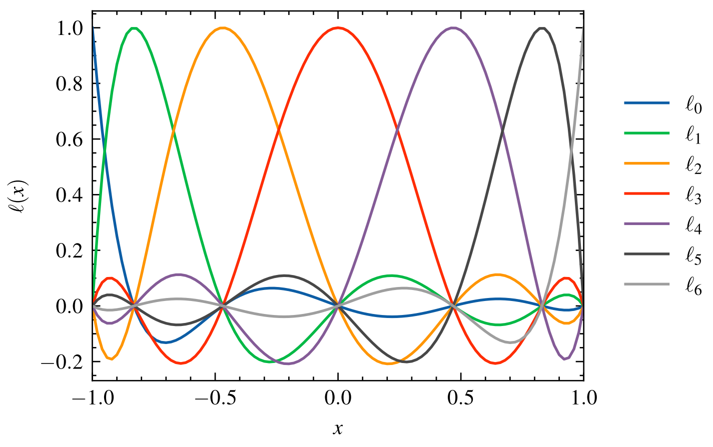


Figure 2.3: An example of nodal basis functions: Lagrange polynomials for $p = 6$.

Similarly, the local modal polynomial expansion is defined as

$$\phi_h^e(x) = \sum_{n=0}^{p} \hat{\phi}_n \psi_n(x), \tag{2.20}$$

where $\hat{\phi}_n$ are the expansion coefficients corresponding to the ψ_n modal basis functions. As opposed to the nodal expansion, $\hat{\phi}_n$ and ψ_n represent the amplitude and frequency of the mode n instead of the physical value and basis function at node i. An example of a modal basis function is the Legendre polynomial, which is illustrated in figure 2.4.

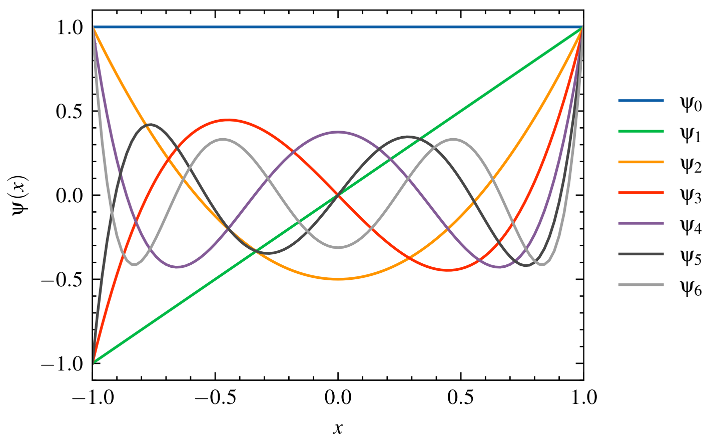


Figure 2.4: An example of modal basis functions: Legendre polynomials (not normalized) until $p = 6$.

Since continuity over element interfaces is not required in DG methods, both nodal and modal formulations can be used. Although modal expansions require the transformation from the nodal grid points to the modal values and back, they are a popular choice when DG methods are pared with multigrid algorithms based on p-coarsening, in which the polynomial order of the solution within the elements is decreased (as is done in this work).

The next question to address is: what makes a good basis function for use in a DG method? A very simple approach to obtain modal basis functions is to construct a basis of monomials. However, this basis is then constructed of non-orthogonal polynomials which means that for higher order polynomials, one basis function cannot be distinguished from another anymore, which leads to an ill-conditioned system of equations. So, it is important to use a basis which ensures discrete linear independence. Therefore, the best way is to find the natural basis for the function space, i.e. use the orthogonal polynomials derived from the eigenvalue problem of the Sturm-Liouville operator.

Orthogonality can be checked by computing the mass matrix which originates from the discretization of the time derivative term in equation (2.18), after substituting the modal formulation in equation (2.14),

$$\begin{aligned} &\int_{\Omega_h^e} \frac{\mathrm{d}U_h^e}{\mathrm{d}t}\psi_j \,\mathrm{d}\Omega = \boldsymbol{M}\frac{\mathrm{d}\hat{U}_n^e}{\mathrm{d}t}, \qquad n,j = 0,\dots,N_{DOF}-1, \\ \Longrightarrow \quad & M_{nj} = \iint_{\Omega_h^e} \psi_n \psi_j \,\mathrm{d}\Omega\,. \end{aligned} \tag{2.21}$$

The basis are orthogonal if the mass matrix is a diagonal matrix with non-zero values on its diagonal.

2.2.1 The Sturm-Liouville operator

While only the result of the Sturm-Liouville operator is used to construct orthogonal basis functions, some background information about the procedure is given in this section.

Most differential operators which are of interest in physical problems (e.g. conservation equations) can be written in the form of the Sturm-Liouville operator defined on the interval (a, b),

$$\frac{\mathrm{d}}{\mathrm{d}x}\left[p(x)\frac{\mathrm{d}\phi(x)}{\mathrm{d}x}\right] + \lambda w(x)\phi(x) = 0, \tag{2.22}$$

where $p(x)$ and $w(x)$ are real-valued functions with $w(x)$ being a weighting function, λ are the eigenvalues and $\phi(x)$ the eigenfunctions. The operator is called singular when $p(x)$ vanishes at at least one of the boundaries.

The eigenfunctions form a complete basis in the space in which the operator is defined. For the singular Sturm-Liouville problem, it can be proven that when the number of eigenfunctions goes to infinity, the error of the solution converges exponentially to zero, [Gir20]. Moreover, when solving a given differential equation analytically, the approximation of the solution can be expanded in terms of the eigenfunctions $\phi(x)$. The solution of the singular Sturm-Liouville operator are orthogonal polynomials which form a natural basis for representing the solution. This highlights the importance of the eigenfunctions derived from the singular Sturm-Liouville problem; when the eigenfunctions are used to represent the numerical solution, semi-analytical methods are incorporated into the numerical solution.

The singular Sturm-Liouville operator can be solved with different boundary conditions and functions $p(x)$ and $w(x)$, resulting in some well-known eigenfunctions. The functions that are defined on finite bounds include the Fourier series (periodic boundary conditions), Legendre and Chebyshev polynomials (non-periodic boundary conditions). Both Legendre and Chebyshev polynomials fall in the class of Jacobi polynomials. The advantage of Legendre polynomials is that their weighting function is 1 which is convenient for the construction of quadrature rules for numerical integration of the integrals. For this reason, the focus of this work will be on Legendre polynomials, which will be elaborated on in the next section. A detailed description about other solutions to the singular Sturm-Liouville operator can be found in [Gir20].

2.2.2 Modal basis

Jacobi polynomials satisfy the solution to the singular Sturm-Liouville problem under the conditions $p(x) = (1 - x^2)w(x)$, $w(x) = (1 - x)^\alpha(1 + x)^\beta$ and $x \in [-1, 1]$, [Sze39],

$$\frac{\mathrm{d}}{\mathrm{d}x}\left[\left(1 - x^2\right) w(x)\frac{\mathrm{d}}{\mathrm{d}x}P_n^{(\alpha,\beta)}(x)\right] + \lambda_n w(x) P_n^{(\alpha,\beta)}(x) = 0, \tag{2.23}$$

where $\lambda_n = n(n + \alpha + \beta + 1)$ are the eigenvalues of the $P_n^{(\alpha,\beta)}(x)$ Jacobi polynomials for the integers α and β. The solution of equation (2.23) is represented by the set of orthogonal Jacobi polynomials with real eigenvalues. It can be proven that the Jacobi polynomials satisfy the following orthogonality condition (with respect to the weighting function $w(x)$) if $\alpha > -1$ and $\beta > -1$, [Koe21],

$$\int_{-1}^{1} w(x)P_n^{(\alpha,\beta)}(x)P_m^{(\alpha,\beta)}(x)\,\mathrm{d}x = \frac{2^{\alpha+\beta+1}\Gamma(n+\alpha+1)\Gamma(n+\beta+1))}{(2n+\alpha+\beta+1)\Gamma(n+\alpha+\beta+1)n!}\delta_{mn}, \tag{2.24}$$

where Γ is the gamma function and δ_{mn} the Kronecker delta. Therefore, the Jacobi polynomials can be made orthonormal by dividing by the norm

$$\tilde{P}_n(x)^{(\alpha,\beta)} = \frac{P_n^{(\alpha,\beta)}(x)}{\sqrt{h_n^{(\alpha,\beta)}}}, \qquad h_n^{(\alpha,\beta)} = \frac{2^{\alpha+\beta+1}\Gamma(n+\alpha+1)\Gamma(n+\beta+1))}{(2n+\alpha+\beta+1)\Gamma(n+\alpha+\beta+1)n!}. \tag{2.25}$$

Another useful identity is, [Sze39; HW08],

$$\frac{\mathrm{d}}{\mathrm{d}x}P_n^{(\alpha,\beta)}(x) = \sqrt{n(n+\alpha+\beta+1)}P_{n-1}^{(\alpha+1,\beta+1)}(x). \tag{2.26}$$

Unfortunately, there does not exist a simple expression to evaluate the Jacobi polynomials. However, expressions for $P_n^{(\alpha,\beta)}$ can still be obtained using recurrence relations such as those given in [Sze39]. These relations are implemented in Python in the `eval_jacobi` function of the `scipy.special` library, [Sci23a].

The simplest form of the Jacobi polynomials, $\alpha = \beta = 0$, results in the Legendre polynomials. In fact, the same set of polynomials is obtained when applying a Gram-Schmidt orthogonalization procedure to a set of monomials. The first three normalized

Legendre polynomials are given by

$$\tilde{P}_0(x) = \frac{1}{\sqrt{2}}, \tag{2.27}$$

$$\tilde{P}_1(x) = \frac{\sqrt{3}}{\sqrt{2}}x, \tag{2.28}$$

$$\tilde{P}_2(x) = \frac{\sqrt{5}}{2\sqrt{2}}\left(3x^2 - 1\right). \tag{2.29}$$

From now on, the notation of the modal basis function ψ_n will be used to denote the Legendre polynomial $\tilde{P}_n$, which is more consistent with the notations used in literature.

2.2.3 Vandermonde matrix

Consider the modal approximation in the $p+1$ grid points x_i inside an element Ω_h^e,

$$\phi(x_i) = \sum_{j=0}^{p} \hat{\phi}_j \psi_j(x_i), \tag{2.30}$$

which can be written in vector notation as

$$\boldsymbol{\phi} = \boldsymbol{V}\hat{\boldsymbol{\phi}}, \tag{2.31}$$

where $\boldsymbol{\phi}$ holds the interpolated (nodal) values of ϕ in the nodes x_i, $\hat{\boldsymbol{\phi}}$ the N_{DOF} (modal) expansion coefficients and V the generalized Vandermonde matrix

$$V_{ij} = \psi_j(x_i). \tag{2.32}$$

For example, the entries of the Vandermonde matrix corresponding to N_{DOF} DOFs and N_i sample points are

$$\boldsymbol{V} = \begin{bmatrix} \psi_0(x_0) & \psi_1(x_0) & \dots & \psi_{N_{DOF}-1}(x_0) \\ \psi_0(x_1) & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \psi_0(x_{N_i-1}) & \dots & \dots & \psi_{N_{DOF}-1}(x_{N_{DOF}-1}) \end{bmatrix}. \tag{2.33}$$

Note that $\boldsymbol{V}$ does not necessarily need to be square when $N_i \neq N_{DOF}$. The Vandermonde matrix plays a critical role in the connection between the nodal values $\boldsymbol{\phi}$ and the modes $\hat{\boldsymbol{\phi}}$

$$\hat{\boldsymbol{\phi}} = \boldsymbol{V}^{-1}\boldsymbol{\phi}. \tag{2.34}$$

Note that in this case it is important that $\boldsymbol{V}$ is square and invertible to be able to retrieve the modal values from the nodal values.

2.2.4 Nodal basis

Since equation (2.30) is an interpolant at the nodes x_i, it suggests that there exists another formulation,

$$\phi(x) = \sum_{i=0}^{p} \phi(x_i) L_i(x), \tag{2.35}$$

which is known as the nodal formulation where L_i is the interpolating Lagrange polynomial with the cardinal property $L_i(x_j) = \delta_{ij}$. The Lagrange polynomials exist and are unique provided that all x_i are distinct, [HW08] and can be computed directly with

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{p} \frac{x - x_j}{x_i - x_j} \tag{2.36}$$

and its derivative

$$\frac{\mathrm{d}L_i}{\mathrm{d}x}(x) = \sum_{\substack{k=0 \\ k \neq i}}^{p} \left(\frac{1}{x_i - x_k} \right) \prod_{\substack{j=0 \\ j \neq i}}^{p} \frac{x - x_j}{x_i - x_j}. \tag{2.37}$$

Equations (2.36) and (2.37) can actually be derived from the modal Lagrange generating polynomials, [Gir20], which stresses the relationship that the Vandermonde matrix establishes between the nodal and modal basis.

In practice, it is easier to construct the Lagrange polynomials from the natural functions of the space, that is, the Legendre polynomials. Using the cardinal property of the Lagrange polynomials the following relation holds

$$\psi_j(x) = \sum_{i=0}^{p} \psi_j(x_i) L_i(x). \tag{2.38}$$

This can be illustrated by considering one of the (nodal) sample points x_k, $k = 0, \dots, p$,

$$L_i(x_k) = \begin{cases} 1 & \text{if} \quad i = k \\ 0 & \text{if} \quad i \neq k \end{cases} \tag{2.39}$$

reducing equation (2.38) to $\psi_j(x_k) = \psi_j(x_k)$. Recall that $V_{ij} = \psi_j(x_i)$, such that equation (2.38) can be written as

$$\psi_j(x) = V_{ij} L_i(x), \tag{2.40}$$

leading to

$$L_j(x) = \sum_{i=0}^{p} V_{i,j}^{-1} \psi_i(x). \tag{2.41}$$

Again, the notation of the nodal basis function ℓ_i will be used to denote the Lagrange basis function L_i to remain consistency with literature.

2.2.5 Lagrange polynomial interpolation

In order to assess the quality of a nodal interpolation using Lagrange polynomials, consider the Lebesgue function, [Ibr16],

$$L_p(x) = \sum_{i=0}^{p} |\ell_i(x)| \tag{2.42}$$

and the Lebesgue constant,

$$\Lambda_p = \max\left(L_p(x)\right) = \max\left(\sum_{i=0}^{p} |\ell_i(x)|\right). \tag{2.43}$$

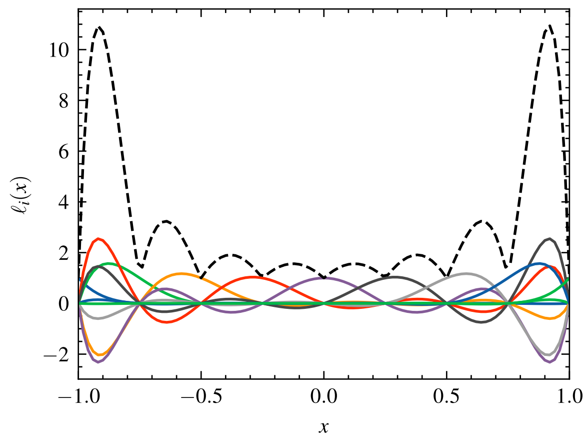
It turns out that the interpolation quality depends on the points that are used as the basis of the interpolation functions (the Lagrange polynomials). A detailed analysis can be found in [Gir20], from which the main findings will be covered in this section.

Consider the Runge function,

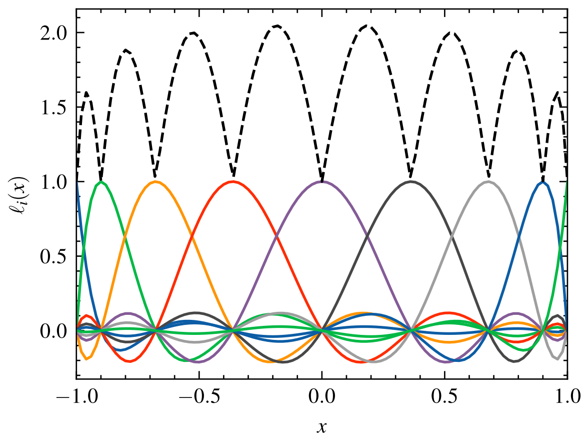
$$f(x) = \frac{1}{1+50x^2}, \qquad x \in [-1,1] \tag{2.44}$$

which will be interpolated on the domain $x \in [-1, 1]$ using various distributions of interpolation points. A set of equidistant and Legendre-Gauss-Lobatto (LGL) quadrature points will be considered, of which the the endpoints ± 1 are included in the latter. The LGL quadrature points can be obtained from the roots of the derivative of the $(p-1)$-th Legendre polynomials using the `roots_jacobi` function of the `scipy.special` library with $n = p - 2$ and $\alpha = \beta = 1$, [Sci23b].

Evaluating the Lagrange polynomials as well as their associated Lebesgue function (figure 2.5, dashed line) for $p = 8$ shows the difference in interpolation quality when



(a) Using equidistant nodes



(b) Using LGL nodes

Figure 2.5: The Lagrange polynomials for $p = 8$ and their corresponding Lebesgue function (dashed black line) on equidistant nodes (a) and LGL nodes (b).

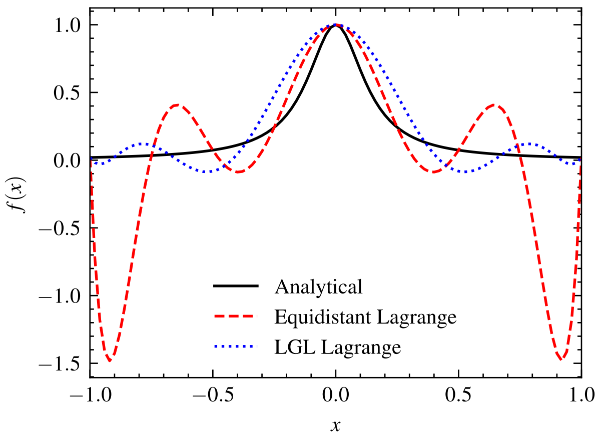


Figure 2.6: The analytical Runge function and its interpolation from Lagrange polynomials for $p = 8$ on equidistant and LGL nodes.

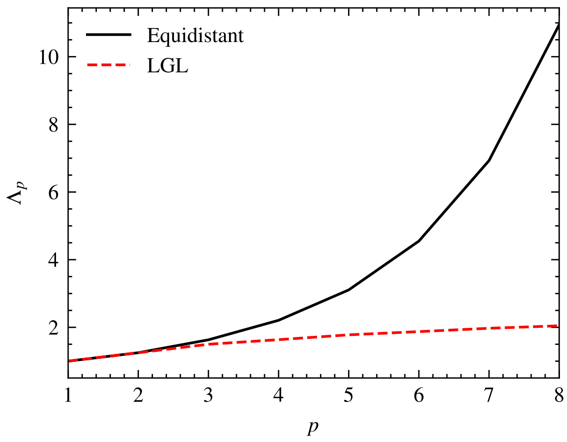


Figure 2.7: Lebesgue constant as a function of the polynomial degree p for equidistant and LGL interpolation points.

using equidistant nodes compared to LGL nodes. The interpolation using equidistant nodes, figure 2.5a, clearly performs worse near the ends of the domain compared to the interpolation using LGL nodes, figure 2.5b. Also, figure 2.6 shows significant oscillations near the boundaries for the interpolation of the Runge function using equidistant nodes, this effect is known as the Runge phenomenon. This especially becomes an issue for higher orders of interpolation, since the Lebesgue constant grows with 2^p for equidistant nodes while it only grows with $\log p$ for LGL nodes, as illustrated by figure 2.7, [HW08]. For practical applications, equidistant nodes might provide sufficient interpolation quality for $p \leq 5$. However, for $p > 5$ the Lebesgue constant becomes so significant that the LGL points provide a much more robust interpolation.

2.3 Numerical integration

The next thing to discuss is the evaluation of the integrals that arise in the DG formulations, for example consider

$$\int_{\Omega_h^e} f(x)\,\mathrm{d}\Omega\,. \tag{2.45}$$

Since the function f is not known in closed form, this integral cannot be computed analytically. However, because the function can be sampled at certain positions, Gaussian quadrature rules can be used where the integral in equation (2.45) is replaced by a sum

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{i=0}^{p_{int}} w_i f(x_i), \tag{2.46}$$

in which x_i and w_i are the integration points and weights, respectively. It can be proven that when $N_{int} = p_{int} + 1$ integration nodes and weights are used, polynomials up to $(2p_{int} + 1)$-th order can be integrated exactly provided that the quadrature rule is based on Legendre polynomials, [Gir20]. Doing this leads to the weights

$$w_i = \frac{2}{(1 - x^2)\left(\psi'_{p_{int}}(x_i)\right)^2} \tag{2.47}$$

where $\psi'_{p_{int}}$ is the derivative of the p_{int}-th order Legendre polynomial with respect to x and the integration nodes x_i are the roots of the Legendre polynomials. Note that the integration interval was assumed to be $[-1, 1]$, this is because the quadrature rule is based on the orthogonal Legendre polynomials, which were derived on $x \in [-1, 1]$. The resulting formulation is unsurprisingly known as the Gauss-Legendre quadrature rule. The N_{int} integration nodes and weights can again be computed using the `roots_jacobi` function of the `scipy.special` library, with $n = N_{int}$ and $\alpha = \beta = 0$.

2.4 Mapping to standard elements

The Jacobi polynomials were derived to be orthogonal in the space $x \in [-1, 1]$ (see section 2.2.2) and consequently the Gauss-Legendre quadrature rule was also defined on the same interval. In practice however, the integrals that need to be evaluated are generally defined on any interval $[a, b]$. Therefore, it is useful to define a mapping from an arbitrary element to a standard element as illustrated by figure 2.8. Here, each individual element is mapped to a standard element with parametric coordinate $r \in [-1, 1]$ on which all calculations can be done very effectively. The main advantage is that all local element matrices can be constructed on the standard element once, and then be scaled to the true size of each element.

An integral on the standard element is realized using coordinate transformation $x \to r$,

$$\int_a^b f(x)\,\mathrm{d}x = \int_{-1}^{1} f(r)\frac{\mathrm{d}x}{\mathrm{d}r}\,\mathrm{d}r\,, \tag{2.48}$$

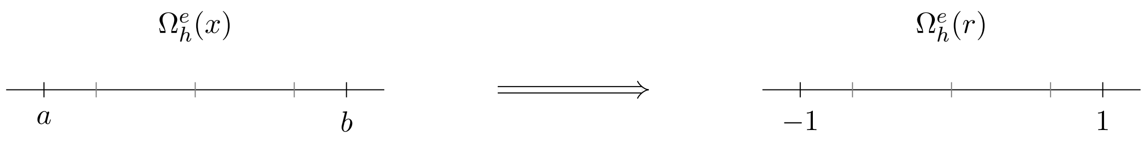


Figure 2.8: Transformation from an arbitrary 1D element $\Omega_h^e(x)$ to a standard element $\Omega_h^e(r)$. The grid points are illustrated by LGL points for $p = 4$.

where in one dimension, $\frac{\mathrm{d}x}{\mathrm{d}r}$ represents the Jacobian of the transformation. Furthermore, for integrands containing derivatives with respect to x, the mapping from x to r introduces the metric term $\frac{\mathrm{d}r}{\mathrm{d}x}$ as a result of the chain rule. Since the integrals will be evaluated using a quadrature rule, the Jacobian of the transformation and the metric terms must be calculated in the integration nodes. By definition, the geometry is defined by a nodal expansion and is independent of the representation of the local solution. Let the coordinate x that describes the geometry be defined by the nodal interpolation of grid order p

$$x(r) = \sum_{i=0}^{p} x_i \ell_i(r), \tag{2.49}$$

where x_i are the LGL nodes of the grid and ℓ_i the corresponding nodal basis functions. Equation (2.49) can be differentiated with respect to the parametric coordinate r, leading to

$$\frac{\mathrm{d}x}{\mathrm{d}r}(r) = \sum_{i=0}^{p} x_i \frac{\mathrm{d}\ell_i}{\mathrm{d}r}(r). \tag{2.50}$$

The Jacobian of the transformation can now be computed in the integration nodes r_i by defining a differentiation matrix $\boldsymbol{\mathcal{D}}_r$ with the entries

$$\mathcal{D}_{r,(ij)} = \frac{\mathrm{d}\ell_j}{\mathrm{d}r}(r_i) \tag{2.51}$$

and applying the matrix-vector product

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}r} = \boldsymbol{\mathcal{D}}_r \boldsymbol{x}, \tag{2.52}$$

where $\boldsymbol{x}$ holds the values of x in the grid nodes. The metric of the mapping in the integration nodes is then obtained from

$$\left(\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}x}\right)_j = \frac{1}{\left(\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}r}\right)_j} = \frac{1}{\left(\boldsymbol{\mathcal{D}}_r \boldsymbol{x}\right)_j}. \tag{2.53}$$

The differentiation matrix can be computed directly using the definition of the Lagrange polynomials, equation (2.36). Analogous to the construction of the Lagrange basis functions in section 2.2.4, $\boldsymbol{\mathcal{D}}_r$ can also be constructed from the Legendre polynomials by

$$\boldsymbol{\mathcal{D}}_r = \boldsymbol{V}_r \boldsymbol{V}^{-1}, \qquad V_{r,(ij)} = \frac{\mathrm{d}\psi_j}{\mathrm{d}r}(r_i), \tag{2.54}$$

where $\boldsymbol{V}_r$ is defined in the integration nodes and $\boldsymbol{V}$ in the grid nodes. For a detailed derivations, see [HW08]. Lastly, computing the unit outward normals is very straightforward in

1D, at the left interface its value is -1 and at the right interface it is 1.

This section concludes the general description of a 1D DG method. The governing equations can now be written in weak form by multiplying the residual with a test function and integrating over the element domain Ω_h^e. Diffusive terms can be treated with for example an interior penalty method while interface fluxes can be used for the discretization of convective terms. Then, the test function can be projected into the space spanned by the basis functions corresponding to the unknown DOFs of the solution and the polynomial expansion of the solution can be substituted. Thereafter, each element can be mapped to a standard element on which the integrals can be evaluated using the metric terms and the Gauss-Legendre quadrature rule. Then, a system of equations can be assembled and solved for the unknown expansion coefficients. If a modal approximation was used, the solution can then be transformed back to the grid nodes using a Vandermonde matrix. In the next section, the DG method will be extended to 2D problems.

2.5 Extension to 2D

In contrast to the 1D elements treated earlier, a choice needs to be made about the shape of the multi-dimensional elements. A popular choice are elements that consist of simplices (e.g. triangles) for which orthonormal basis can be constructed, which is treated in [Dub91; HW08]. While these types of elements provide the most geometric flexibility, quadrilateral elements (e.g. squares) enable for a very straightforward dimension-by-dimension extension from a 1D to a multi-dimensional domain. As a result, the details of the multi-dimensional DG formulations are the easiest for quadrilateral elements. Therefore, since the fundamentals are the same for all element types and this is a conceptual study, this thesis will focus on quadrilateral elements.

The tremendous advantage of a dimension-by-dimension approach is that all theory behind 1D interpolation and numerical integration discussed in sections 2.2 and 2.3 also applies to the 2D quadrilateral element. This is also the case for an extension to 3D hexahedral elements. The 1D basis functions and Gauss-Legendre quadrature were defined on the standard element $r \in [-1, 1]$. Following a dimension-by-dimension approach, the 2D standard elements is defined as $\Omega_I(r, s) = [-1, 1]^2$.

2.5.1 Basis functions

2D basis functions for quadrilateral elements are obtained by applying the tensor-product of the 1D basis functions. For example, the 2D modal basis function $\psi_k(r, s)$ is constructed of the tensor-product of the basis $\psi_i(r)$ and $\psi_j(s)$,

$$\psi_k(r, s) = \psi_i(r) \otimes \psi_j(s), \qquad i, j = 0, \ldots, p, \quad k = 0, \ldots, N_{DOF} - 1. \tag{2.55}$$

Here, the tensor-product base $\psi_k(r, s)$ is written in monolithic form, i.e. a vector with a single index k of length $N_{DOF} = (p + 1)^2$ containing the 2D basis functions. While this is a very compact way to write the basis functions (only a single index is needed), operations such as interpolation and differentiation can be performed more efficiently using tensor-product basis $\psi_{ij}(r, s)$, [Gir20]. For this reason, tensor-product basis are the preferred formulation in computer codes. However, since this significantly adds to the complexity of the code and the scope of this thesis is on solving the resulting discrete system efficiently, only the monolithic formulation will be considered. The tensor-product space is mapped onto the monolithic space using

$$k = j(p+1) + i. \tag{2.56}$$

The nodal 2D basis functions are constructed in a similar fashion from the tensor-product of $\ell_i(r)$ and $\ell_j(s)$,

$$\ell_k(r,s) = \ell_i(r) \otimes \ell_j(s), \qquad i,j = 0,\dots,p, \quad k = 0,\dots,N_{DOF}-1. \tag{2.57}$$

2.5.2 Numerical integration

In two dimensions, the weak DG formulation will contain surface and line integrals (as a result of applying Gauss's divergence theorem) of the form

$$\iint_{\Omega_h^e} \mathrm{f}(x,y)\,\mathrm{d}\Omega \qquad \text{and} \qquad \oint_{\Gamma_h^e} \mathrm{g}(x,y)\,\mathrm{d}\Gamma\,, \tag{2.58}$$

where Γ_h^e is the boundary of the domain Ω_h^e and $\boldsymbol{x} = (x,y)^T$. However, to keep consistency with literature, the aforementioned integrals will be referred to as the (3D) volume and surface integrals, respectively, during the remainder of this text.

As became clear from section 2.3, the Gauss-Legendre quadrature rule is very much tied to the Legendre polynomials. Therefore, the Gauss-Legendre quadrature rule can be extended to a 2D domain in a similar way as the basis functions, thus by applying a tensor product of the 1D rule. On the standard element $\Omega_I \in [-1,1]^2$ this leads to the following 2D quadrature (or cubature) rule

$$\iint_{\Omega_I} \mathrm{f}(x,y)\,\mathrm{d}\Omega = \sum_{k=0}^{N_{int}-1} w_k\,\mathrm{f}(x_k,y_k), \tag{2.59}$$

where w_k is obtained from the tensor-product

$$w_k = w_i \otimes w_j, \qquad i,j = 0,\dots,p_{int}, \quad k = 0,\dots,N_{int}-1. \tag{2.60}$$

2.5.3 Mapping to standard elements

Similar to the 1D mapping discussed in section 2.4, an arbitrary 2D element in physical coordinates $\Omega_h^e(x,y)$ can be mapped to a standard element in parametric coordinates $\Omega_I(r,s)$, which is is depicted in figure 2.9. In order to realize the mapping $(x,y) \to (r,s)$, consider

$$\mathrm{d}x = \frac{\partial x}{\partial r}\,\mathrm{d}r + \frac{\partial x}{\partial s}\,\mathrm{d}s\,, \tag{2.61}$$

$$\mathrm{d}y = \frac{\partial y}{\partial r}\,\mathrm{d}r + \frac{\partial y}{\partial s}\,\mathrm{d}s\,, \tag{2.62}$$

which can be written in matrix-vector form as

$$\begin{bmatrix} \mathrm{d}x \\ \mathrm{d}y \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} \end{bmatrix} \begin{bmatrix} \mathrm{d}r \\ \mathrm{d}s \end{bmatrix}. \tag{2.63}$$

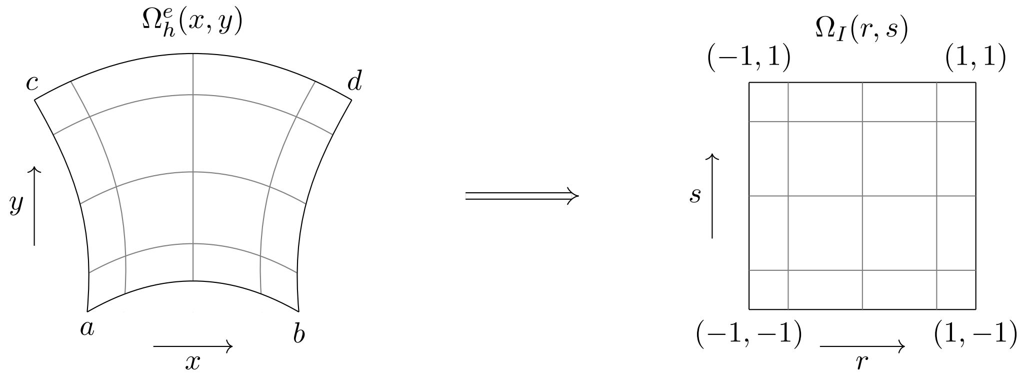


Figure 2.9: Transformation from an arbitrary 2D element $\Omega_I(x,y)$ to a standard element $\Omega_h^e(r,s)$.

The matrix

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} \end{bmatrix} \tag{2.64}$$

is commonly known as the Jacobian of the transformation. Conversely, writing the derivatives of r and s with respect to x and y leads to

$$\begin{bmatrix} \mathrm{d}r \\ \\ \mathrm{d}s \end{bmatrix} = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} \\ \frac{\partial s}{\partial x} & \frac{\partial s}{\partial y} \end{bmatrix} \begin{bmatrix} \mathrm{d}x \\ \\ \mathrm{d}y \end{bmatrix}. \tag{2.65}$$

Comparing equations (2.63) and (2.65), it is clear that they are equal when

$$\begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} \\ \frac{\partial s}{\partial x} & \frac{\partial s}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} \end{bmatrix}^{-1} = \frac{1}{|\boldsymbol{J}|} \begin{bmatrix} \frac{\partial y}{\partial s} & -\frac{\partial x}{\partial s} \\ -\frac{\partial y}{\partial r} & \frac{\partial x}{\partial r} \end{bmatrix}, \tag{2.66}$$

from which the following relations for the metric terms of the mapping are obtained

$$\frac{\partial r}{\partial x} = \frac{1}{|\boldsymbol{J}|}\frac{\partial y}{\partial s}, \tag{2.67}$$

$$\frac{\partial r}{\partial y} = -\frac{1}{|\boldsymbol{J}|}\frac{\partial x}{\partial s}, \tag{2.68}$$

$$\frac{\partial s}{\partial x} = -\frac{1}{|\boldsymbol{J}|}\frac{\partial y}{\partial r}, \tag{2.69}$$

$$\frac{\partial s}{\partial y} = \frac{1}{|\boldsymbol{J}|}\frac{\partial x}{\partial r}, \tag{2.70}$$

where $|\boldsymbol{J}|$ denotes the determinant of the Jacobian of the transformation,

$$|\boldsymbol{J}| = \frac{\partial x}{\partial r}\frac{\partial y}{\partial s} - \frac{\partial y}{\partial r}\frac{\partial x}{\partial s}. \tag{2.71}$$

For the evaluation of the surface integrals, the determinant of the Jacobian of the faces, $|\boldsymbol{J}|_f$, is needed. For an i-face ($r = \pm 1$) this becomes

$$|\boldsymbol{J}|_{f_i} = \sqrt{\left(\frac{\partial x}{\partial s}\right)^2 + \left(\frac{\partial y}{\partial s}\right)^2} \tag{2.72}$$

and similarly for a j-face ($s = \pm 1$),

$$|\boldsymbol{J}|_{f_j} = \sqrt{\left(\frac{\partial x}{\partial r}\right)^2 + \left(\frac{\partial y}{\partial r}\right)^2}. \tag{2.73}$$

Writing the physical coordinates in terms of the parametric coordinates using the nodal interpolation of grid order p with tensor-product basis,

$$x(r,s) = \sum_{i=0}^{p}\sum_{j=0}^{p} x_{ij}\ell_i(r)\ell_j(s), \tag{2.74}$$

$$y(r,s) = \sum_{i=0}^{p}\sum_{j=0}^{p} y_{ij}\ell_i(r)\ell_j(s), \tag{2.75}$$

with its derivatives

$$\frac{\partial x}{\partial r}(r,s) = \sum_{i=0}^{p}\sum_{j=0}^{p} x_{ij}\frac{\partial \ell_i}{\partial r}(r)\ell_j(s), \qquad \frac{\partial y}{\partial r}(r,s) = \sum_{i=0}^{p}\sum_{j=0}^{p} y_{ij}\frac{\partial \ell_i}{\partial r}(r)\ell_j(s), \tag{2.76}$$

$$\frac{\partial x}{\partial s}(r,s) = \sum_{i=0}^{p}\sum_{j=0}^{p} x_{ij}\ell_i(r)\frac{\partial \ell_j}{\partial s}(s), \qquad \frac{\partial y}{\partial s}(r,s) = \sum_{i=0}^{p}\sum_{j=0}^{p} y_{ij}\ell_i(r)\frac{\partial \ell_j}{\partial s}(s). \tag{2.77}$$

Extending the idea of section 2.4, $\frac{\partial x}{\partial r}$, $\frac{\partial x}{\partial s}$, $\frac{\partial y}{\partial r}$ and $\frac{\partial y}{\partial s}$ can be calculated in the integration nodes (r_i, s_i) by using the differentiation matrices in r- and s-direction,

$$\boldsymbol{\mathcal{D}}_r = \boldsymbol{V}_r\boldsymbol{V}^{-1}, \qquad V_{r,(ij)} = \frac{\mathrm{d}\psi_j}{\mathrm{d}r}(r_i, s_i), \tag{2.78}$$

$$\boldsymbol{\mathcal{D}}_s = \boldsymbol{V}_s\boldsymbol{V}^{-1}, \qquad V_{s,(ij)} = \frac{\mathrm{d}\psi_j}{\mathrm{d}s}(r_i, s_i), \tag{2.79}$$

leading to

$$\frac{\partial \boldsymbol{x}}{\partial r} = \boldsymbol{\mathcal{D}}_r\boldsymbol{x}, \tag{2.80}$$

$$\frac{\partial \boldsymbol{x}}{\partial s} = \boldsymbol{\mathcal{D}}_s\boldsymbol{x}, \tag{2.81}$$

$$\frac{\partial \boldsymbol{y}}{\partial r} = \boldsymbol{\mathcal{D}}_r\boldsymbol{y}, \tag{2.82}$$

$$\frac{\partial \boldsymbol{y}}{\partial s} = \boldsymbol{\mathcal{D}}_s\boldsymbol{y}, \tag{2.83}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ contain the values of respectively x and y in the integration nodes. Now, the metric terms $\frac{\partial r}{\partial x}$, $\frac{\partial r}{\partial y}$, $\frac{\partial s}{\partial x}$, $\frac{\partial s}{\partial y}$ can be calculated in the integration nodes using equations (2.67) to (2.71) and (2.80) to (2.83).

Moreover, the unit outward normal of an i-face can be calculated with

$$\mathbf{n} = \pm\frac{\nabla r}{|\nabla r|} = \pm\left(\frac{\partial r}{\partial x}\,\mathbf{i} + \frac{\partial r}{\partial y}\,\mathbf{j}\right) \bigg/ \sqrt{\left(\frac{\partial r}{\partial x}\right)^2 + \left(\frac{\partial r}{\partial y}\right)^2}, \tag{2.84}$$

where $\mathbf{i}$ and $\mathbf{j}$ are the unit vectors in x and y direction and the $\pm$ sign is negative for a left face and positive for a right face. Similarly for j-faces,

$$\mathbf{n} = \pm\frac{\nabla s}{|\nabla s|} = \pm\left(\frac{\partial s}{\partial x}\,\mathbf{i} + \frac{\partial s}{\partial y}\,\mathbf{j}\right) \bigg/ \sqrt{\left(\frac{\partial s}{\partial x}\right)^2 + \left(\frac{\partial s}{\partial y}\right)^2}. \tag{2.85}$$

Now that all metric terms can be calculated in the integration nodes, volume integrals on an arbitrary element Ω_h^e can be evaluated on the standard elements using

$$\iint_{\Omega_h^e} \mathrm{f}(x,y)\,\mathrm{d}\Omega = \sum_{k=0}^{N_{int}^{2D}-1} w_k\ \mathrm{f}(r_k, s_k)|\boldsymbol{J}|. \tag{2.86}$$

In a similar fashion, surface integrals over an element boundary Γ_h^e can be evaluated with

$$\oint_{\Gamma_h^e} \mathrm{g}(x,y)\,\mathrm{d}\Gamma = \sum_{f\in\mathcal{F}_h^e}\left(\sum_{i=0}^{N_{int}^{1D}-1} w_i\ \mathrm{g}_{int}\,|\boldsymbol{J}|_f\right), \tag{2.87}$$

where for i-faces equation (2.72) must be used for $|\boldsymbol{J}|_f$ and $\mathrm{g}_{int} = \mathrm{g}(\pm 1, s_i)$ while for j-faces $|\boldsymbol{J}|_f$ is described by equation (2.73) and $\mathrm{g}_{int} = \mathrm{g}(r_i, \pm 1)$. Remember that the index k runs over the 2D integration nodes and weights while the indices i and j run over the 1D integration nodes and weights.

Chapter 3

Incompressible flow modelling

The motion of a viscous fluid subject to incompressible flow is described by the set of incompressible Navier-Stokes equations,

$$\begin{cases} \partial_t \mathbf{u} - \nu\Delta \mathbf{u} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} + \nabla\, \mathrm{p} &= \mathbf{f}, \qquad \forall \boldsymbol{x} \in \Omega, & (3.1)\\ \nabla\cdot\mathbf{u} &= 0, \qquad \forall \boldsymbol{x} \in \Omega, & (3.2)\\ \mathbf{u} &= \mathbf{g}_{\mathcal{D}}, \qquad \forall \boldsymbol{x} \in \Gamma. & (3.3)\end{cases}$$

on the time interval $(0, T)$, with an appropriate set of initial conditions and Dirichlet boundary conditions $\mathbf{g}_{\mathcal{D}}$ prescribed on the boundary Γ of the domain Ω. Moreover, $\partial_t\,\mathbf{u}$ denotes the partial derivative of the velocity vector $\mathbf{u} = [\mathrm{u}, \mathrm{v}]^T$ with respect to time, ν is the kinematic viscosity, p the kinematic pressure (pressure divided by density) and $\mathbf{f}$ the external force per unit mass. The inertial forces are represented by the non-linear convection term $(\mathbf{u}\cdot\nabla)\,\mathbf{u}$ while the term $\nu\Delta\,\mathbf{u}$ describes the viscous effects. While the incompressibility condition seems to be a nice way to simplify the compressible Navier-Stokes problem, it is actually a constraint for the Stokes problem which will be explained in more detail in this chapter.

Firstly, incompressible flows will be explained in more detail. This section is largely based on [Pan13], in which a much more fundamental description about this subject can be found. Then, the incompressible model problems will be defined and the consequences of the incompressibility constraint for these model problems will become apparent. Lastly, a pressure-robust DG discretization scheme for the model problems is introduced and the characteristics of the discrete systems of equations are revealed.

3.1 Incompressible flow

Incompressible flow is characterized by fluid motion in which density changes of a particle are negligible, which can mathematically be expressed as

$$\frac{1}{\rho}\frac{\mathrm{D}\rho}{\mathrm{D}t} = 0. \tag{3.4}$$

Without going into too much detail, this is the case for flows in which the Mach number, i.e. the flow velocity with respect to the speed of sound, is below 0.3, [Pan13]. Rewriting the (compressible) continuity equation,

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\,\mathbf{u}) = 0, \tag{3.5}$$

using product rule and the material derivative $\frac{\mathrm{D}}{\mathrm{D}t}$ gives

$$\frac{1}{\rho}\frac{\mathrm{D}\rho}{\mathrm{D}t} = -\nabla \cdot \mathbf{u}\,. \tag{3.6}$$

Therefore, the incompressibility condition denoted by equation (3.4) reduces the continuity equation to

$$\nabla \cdot \mathbf{u} = 0. \tag{3.7}$$

which is also referred to as the kinematic divergence-free constraint. This is a constraint because the continuity equation does not have a time derivative anymore while the divergence of the velocity field must be zero. A physical interpretation of $\nabla \cdot \mathbf{u}$ is the rate of change of volume ∂V of a moving particle V,

$$\nabla \cdot \mathbf{u} = \frac{1}{\partial V}\frac{\mathrm{D}(\partial V)}{\mathrm{D}t}, \tag{3.8}$$

So, in incompressible flows, the rate of change of volume of a particle is zero.

A globally constant fluid density is often assumed to be a requirement for incompressible flow, since this also leads to the reduced continuity equation in equation (3.7). However, it is not true that incompressible flow only occurs when the density of the fluid is globally constant. While it is the case that a fluid with a globally constant density is subject to incompressible flow, a fluid with variable density can also experience incompressible flow, as long as the density of each fluid particle remains constant. For example, the density of the air in the atmosphere depends on the spatial location but, under the right flow conditions, each individual fluid particle can have a constant density. Thus, the compressible fluid air can be subject to incompressible flow. Hence, incompressible *flow* is not to be confused with an incompressible *fluid*.

3.2 Model problems

The Stokes problem is considered as the main model problem of this thesis. The Stokes equations are obtained by non-dimensionalization of the incompressible Navier-Stokes equations and requiring the viscous forces to be dominant, i.e. $\mathrm{Re} \ll 1$. In dimensional form, the Stokes equations read

$$\left\{\begin{array}{rcll} -\Delta\,\mathbf{u} + \nabla\,\mathrm{p} & = & \mathbf{f}, & \forall \boldsymbol{x} \in \Omega, \qquad (3.9) \\ \nabla \cdot \mathbf{u} & = & 0, & \forall \boldsymbol{x} \in \Omega, \qquad (3.10) \\ \mathbf{u} & = & \mathbf{g}_{\mathcal{D}}, & \forall \boldsymbol{x} \in \Gamma. \qquad (3.11) \end{array}\right.$$

Note that without loss of generality, the kinematic viscosity is set to $\nu = 1$ since the momentum equation, pressure p and forcing term $\mathbf{f}$ can always be scaled by ν, [BGL05]. Also, in contrast to the non-linear incompressible Navier-Stokes system, the Stokes system is linear. Since only the gradient of the pressure is present in equation (3.9), the pressure is defined up to a constant. A unique pressure solution is obtained by requiring the mean pressure to be zero,

$$\bar{\mathrm{p}} = \frac{\iint_\Omega \mathrm{p}\,\mathrm{d}\Omega}{\iint_\Omega \mathrm{d}\Omega} = 0. \tag{3.12}$$

Writing equations (3.9) and (3.10) as a block linear system of equations of the form $\boldsymbol{\mathcal{A}}\boldsymbol{\xi} = \boldsymbol{b}$,

$$\begin{bmatrix} -\boldsymbol{\Delta} & \boldsymbol{\mathcal{G}} \\ \boldsymbol{\mathcal{D}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathrm{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}, \tag{3.13}$$

where $\boldsymbol{\Delta}$ denotes the (vector) Laplace operator which acts on the velocity $\mathbf{u}$, $\boldsymbol{\mathcal{G}}$ the gradient operator which acts on the pressure and $\boldsymbol{\mathcal{D}}$ the divergence operator which acts on the velocity. As a result of the divergence-free constraint $\nabla \cdot \mathbf{u} = 0$ in the continuity equation, the block coefficient matrix is in saddle point form. Saddle point problems arise in many computational science and engineering applications, often when a certain quantity must be minimized. Due to their indefiniteness and poor spectral properties, saddle point problems pose a serious challenge for (iterative) solvers, [BGL05].

Lastly, for a multigrid method to work well, it is important that the Stokes system possesses good smoothing properties (see chapter 4 for more details). For elliptic systems of equations, it can be proven that when the determinant of a system can be smoothed well, the system itself also possesses good smoothing properties, [TDB03; OL06]. Since the determinant of the Stokes system is associated with the Laplacian, $\det(\boldsymbol{\mathcal{A}}) = \Delta^2$ in 2D and $\det(\boldsymbol{\mathcal{A}}) = \Delta^3$ in 3D, the Poisson problem is used as a model problem for the Stokes system,

$$-\Delta\,\mathrm{u} = \mathrm{f}\,. \tag{3.14}$$

3.3 Discretization scheme

Before the model problems defined in the previous section can be solved using a computer, they first must be discretized. This section mainly focuses on the discretization of the Stokes problem. Since the Poisson problem is a subset of the Stokes problem, the discretization of the Poisson equation is found by taking the discretization of the diffusion and source terms of the Stokes (x-)momentum equation. Literature on the discretization of the Stokes problem is widely available in the field of computational mathematics, such as for example [GLS19]. However, these contributions are often written very compactly which makes the implementation in computer code difficult. Therefore, in this section it is attempted to present the discretization of the Stokes problem in more engineering-like notation which should be closer to the practical implementation.

It is important that the discretization of the Stokes equations is stable, convergent and robust. In recent years, it has been brought to attention that the discretization of the divergence constraint in equation (3.10) plays a critical role in achieving robust spatial discretization schemes of the (Navier-)Stokes equations, [Joh+17; Sch19; GLS19]. It turns out that in most mixed element methods such as the DG method, the divergence constraint is relaxed by only enforcing it discretely, i.e. up to the level of the truncation error of the DG scheme. This means that the discrete velocity solution is not necessarily divergence-free, but only discretely divergence-free. While these methods are still stable and convergent, this introduces a pressure-dependent consistency error which can pollute the velocity solution when large and complicated pressures are present, [Joh+17]. In this case, a contribution from the right-hand side which only influences the pressure in the continuous equations influences both the velocity and pressure in the discrete equations. This is referred to as a lack of pressure-robustness.

A popular method for improving pressure-robustness is grad-div stabilization which is based on penalization of the divergence error. While it does not completely remove the lack of pressure-robustness, its main advantage is that it is very straightforward to apply the technique to existing mixed element discretizations. Another technique that is used to completely remove the lack of pressure robustness is to choose appropriate test functions for some terms in the DG formulation that reestablishes the properties from the continuous problem into the discretization, [Joh+17]. Following this approach, a so-called $\boldsymbol{H}(\mathrm{div})$-conforming scheme is obtained which is proven to be exactly divergence-free, meaning that the divergence constraint is always satisfied. However, the generalization to arbitrary element types is very difficult.

Since grad-div stabilization can be added to existing discretizations, this method is chosen to reduce the lack of pressure-robustness of the existing Stokes discretization. Details on this technique are given in section 3.3.4. However, first an overview of the DG formulations is given in section 3.3.1 followed by the description of the discretization of the continuity equation and momentum equations, sections 3.3.2 and 3.3.3, respectively.

3.3.1 DG formulations

A stable DG formulation for the Stokes equations can be found by solving for the local solution $(\mathrm{p}_h, \mathbf{u}_h)$ belonging to the mixed trial space

$$(\mathrm{p}_h, \mathbf{u}_h) \in \mathcal{Q}_h \times \mathcal{V}_h, \tag{3.15}$$

where the approximation space is spanned by the piece-wise smooth velocity and pressure test functions $q_h \in \mathcal{Q}_h$ and $v_h \in \mathcal{V}_h$, [Hil13]. Odd-even decoupling is avoided by choosing the polynomial order of the pressure space one order lower than the velocity space and by multiplying the weak form of the continuity equation with the pressure test functions and the weak form of the momentum equations with the velocity test functions, [GLS19]. The solution is sought on the discretized 2D domain $\Omega_h \simeq \bigcup_{e=0}^{N_e} \Omega_h^e$.

The local velocity and pressure solutions within an element are defined using a modal expansion,

$$\mathbf{u}_h^e = \sum_{n=0}^{N_{DOF}^{\mathrm{u}}-1} \hat{\mathbf{u}}_n^e \psi_n^{\mathrm{u}}, \qquad \mathrm{p}_h^e = \sum_{n=0}^{N_{DOF}^{\mathrm{p}}-1} \hat{\mathrm{p}}_n^e \psi_n^{\mathrm{p}}, \tag{3.16}$$

where the vectors $\mathbf{u}_h^e$ and $\hat{\mathbf{u}}_n^e$ hold the solution and the expansion coefficients for the velocity components u and v and $N_{DOF}^{\mathrm{u}} = (p_{\mathrm{u}} + 1)^2$ and $N_{DOF}^{\mathrm{p}} = (p_{\mathrm{p}} + 1)^2$ are the velocity and pressure DOFs, respectively. The assumption is made that the same basis functions ψ^{u} can be used for all components of $\mathbf{u}_h^e$. In total, there are N_{DOF}^{u} unknown expansion coefficients $\hat{\mathbf{u}}_n^e$ and N_{DOF}^{p} unknown expansion coefficients $\hat{\mathrm{p}}_n^e$ to be solved per element[1]. The number of

[1]Since $\hat{\mathbf{u}}_n^e$ is a vector, this equals a total of $2N_{DOF}^{\mathrm{u}}$ unknown expansion coefficients, N_{DOF}^{u} expansion coefficients for $\hat{\mathrm{u}}_n^e$ and N_{DOF}^{u} expansion coefficients for $\hat{\mathrm{v}}_n^e$.

DOFs per element depend on the polynomial order of the solution. Although the notation might suggest that the number of DOFs for the velocity and pressure solutions is the same for all elements, this does not have to be the case.

Furthermore, a surface integral over the boundary of an element e can be written as the summation over the faces f, i.e.

$$\int_{\partial\Omega_h^e} (\dots) = \int_{\Gamma_h^e} (\dots) = \sum_{f\in\mathcal{F}_h^e} \int_{\Gamma_{hf}^e} (\dots). \tag{3.17}$$

The faces f of the elements e belong to the set $\mathcal{F}_h^e$ which can be split into interior faces $\mathcal{F}_h^{e,i}$ and boundary faces $\mathcal{F}_h^{e,\partial}$, $\mathcal{F}_h^e = \mathcal{F}_h^{e,i} \cup \mathcal{F}_h^{e,\partial}$. For any piece-wise smooth function ϕ, consider the jump $[\![\cdot]\!]$ and average $\{\!\{\cdot\}\!\}$ operator across the interior element faces,

$$[\![\phi]\!] = \phi^e - \phi^n, \quad \{\!\{\phi\}\!\} = \frac{1}{2}\left(\phi^e + \phi^n\right), \qquad \forall f \in \mathcal{F}_h^{e,i}, \tag{3.18}$$

while for the boundary faces,

$$[\![\phi]\!] = \phi^e, \quad \{\!\{\phi\}\!\} = \phi^e, \qquad \forall f \in \mathcal{F}_h^{e,\partial}, \tag{3.19}$$

where the superscript e and n denote that the value of the function ϕ is approached from the element e and the neighboring element n, respectively, see figure 3.1. The interface unit normal $\mathbf{n}_f$ is defined to point outward of e, into n.

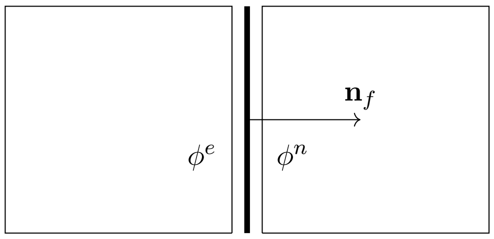


Figure 3.1: The values of ϕ when approaching the interface (thick line) from the element e and neighboring element n. The interface unit normal is defined to be pointing outward of e.

3.3.2 Continuity equation

The locally defined weak formulation of the Stokes continuity equation given in equation (3.10) reads

$$\iint_{\Omega_h^e} \left(\nabla_h \cdot \mathbf{u}_h^e\right) q_h \, \mathrm{d}\Omega = 0, \qquad \forall q_h \in \mathcal{Q}_h, \quad \forall e, \tag{3.20}$$

where $\nabla_h \cdot$ is the broken (i.e. defined within an element) divergence operator. The discretization of the continuity equation follows from the weak formulation in equation (3.20) and additional surface integrals which enable coupling of the elements by describing the normal velocity jump over the (boundary) faces, [GLS19],

$$\begin{aligned} -\iint_{\Omega_h^e} q_h(\nabla_h \cdot \mathbf{u}_h^e)\,\mathrm{d}\Omega + \sum_{f\in\mathcal{F}_h^e} \int_{\Gamma_{hf}^e} \left(\llbracket \mathbf{u}_h \rrbracket \cdot \mathbf{n}_f\right) \{\!\{q_h\}\!\}\,\mathrm{d}\Gamma \\ = \sum_{f\in\mathcal{F}_h^{e,\partial}} \int_{\Gamma_{hf}^e} \left(\llbracket \mathbf{g}_D \rrbracket \cdot \mathbf{n}_f\right) \{\!\{q_h\}\!\}\,\mathrm{d}\Gamma\,, \qquad \forall q_h \in \mathcal{Q}_h, \quad \forall e. \end{aligned} \tag{3.21}$$

By adding these terms, a discrete pressure-velocity coupling is obtained, [GLS19; Sch19]. The surface integral terms can be added because in the limit of the exact solution, the normal velocity jump over the faces goes to zero, such that these integrals vanish. The minus sign in front of the volume integral is there to obtain a symmetric discretization. As will become clear in section 3.3.3, the pressure gradient term is integrated by parts leading to a minus sign in front of the volume integral in equation (3.30). By also putting a minus sign in front of the surface integral in the discretization of the continuity equation, equation (3.21), a symmetric discretization is obtained, i.e. the discretization of the divergence operator $\boldsymbol{\mathcal{D}}$ is equal to the discretization of the gradient operator transposed, $\boldsymbol{\mathcal{G}}^T$.

Working out the jump and average operators using equations (3.18) and (3.19),

$$\begin{aligned} \llbracket \mathbf{u}_h \rrbracket = \mathbf{u}_h^e - \mathbf{u}_h^n, \quad \{\!\{q_h\}\!\} = \frac{1}{2} q_h, \qquad \forall f \in \mathcal{F}_h^{e,i}, \\ \llbracket \mathbf{u}_h \rrbracket = \mathbf{u}_h^e, \quad \llbracket \mathbf{g}_D \rrbracket = \mathbf{g}_{\mathcal{D}}, \quad \{\!\{q_h\}\!\} = q_h, \qquad \forall f \in \mathcal{F}_h^{e,\partial}. \end{aligned} \tag{3.22}$$

After splitting the set $\mathcal{F}_h^e$ in the second term of equation (3.21) into interior and boundary faces, substituting the expressions from equation (3.22) for the jump and average operators and rearranging terms, the discretization of the continuity equation reads

$$\begin{aligned} &-\iint_{\Omega_h^e} q_h\left(\nabla_h \cdot \mathbf{u}_h^e\right)\mathrm{d}\Omega \\ &+ \sum_{f\in\mathcal{F}_h^{e,i}} \int_{\Gamma_{hf}^e} \frac{1}{2} q_h \left(\mathbf{u}_h^e - \mathbf{u}_h^n\right) \cdot \mathbf{n}_f \,\mathrm{d}\Gamma \\ &+ \sum_{f\in\mathcal{F}_h^{e,\partial}} \int_{\Gamma_{hf}^e} q_h \left(\mathbf{u}_h^e - \mathbf{g}_{\mathcal{D}}\right) \cdot \mathbf{n}_f \,\mathrm{d}\Gamma = 0, \qquad \forall q_h \in \mathcal{Q}_h, \quad \forall e. \end{aligned} \tag{3.23}$$

Finally, projecting q_h into the function space spanned by the unknowns, i.e. the basis functions ψ_j^{p}, a set of N_{DOF}^{p} discrete continuity equations corresponding to the discretization of the $\boldsymbol{\mathcal{D}}$-block in equation (3.13) is obtained

$$\begin{aligned} &-\iint_{\Omega_h^e} \psi_j^{\mathrm{p}}\left(\nabla_h \cdot \mathbf{u}_h^e\right)\mathrm{d}\Omega \\ &+ \sum_{f\in\mathcal{F}_h^{e,i}} \int_{\Gamma_{hf}^e} \frac{1}{2} \psi_j^{\mathrm{p}} \left(\mathbf{u}_h^e - \mathbf{u}_h^n\right) \cdot \mathbf{n}_f \,\mathrm{d}\Gamma \\ &+ \sum_{f\in\mathcal{F}_h^{e,\partial}} \int_{\Gamma_{hf}^e} \psi_j^{\mathrm{p}} \left(\mathbf{u}_h^e - \mathbf{g}_{\mathcal{D}}\right) \cdot \mathbf{n}_f \,\mathrm{d}\Gamma = 0, \qquad j = 0, \ldots, N_{DOF}^{\mathrm{p}} - 1, \quad \forall e. \end{aligned} \tag{3.24}$$

3.3.3 Momentum equations

The weak formulation of the momentum equations given in equation (3.9) is given by

$$\underbrace{-\iint_{\Omega_h^e} \Delta_h \, \mathbf{u}_h^e \; v_h \, \mathrm{d}\Omega}_{\text{diffusion term}} \quad \underbrace{+\iint_{\Omega_h^e} \nabla_h \, \mathrm{p}_h^e \; v_h \, \mathrm{d}\Omega}_{\text{pressure gradient term}} \quad \underbrace{-\iint_{\Omega_h^e} \mathbf{f} \, v_h \, \mathrm{d}\Omega}_{\text{source term}} = 0, \qquad \forall \, v_h \in \mathcal{V}_h, \quad \forall e, \tag{3.25}$$

where Δ_h is the broken Laplace operator and ∇_h the broken gradient operator (both defined within an element). The discretization of the momentum equations will be split up into several parts which is treated in the next sections.

Diffusion term

The discretization of the diffusion term follows from applying Green's first identity[2] to the weak formulation of the diffusion term in equation (3.25). For the x-momentum equation, this results in the following discretization, [GLS19]

$$\iint_{\Omega_h^e} \nabla_h \, \mathrm{u}_h^e \cdot \nabla_h v_h \, \mathrm{d}\Omega - \sum_{f \in \mathcal{F}_h^e} \int_{\Gamma_{hf}^e} \{\!\{ \nabla_h \, \mathrm{u}_h \}\!\} \, \mathbf{n}_f \cdot [\![v_h]\!] \, \mathrm{d}\Gamma \,, \qquad \forall \, v_h \in \mathcal{V}_h, \quad \forall e. \tag{3.26}$$

However, if only these terms were to be used for the discretization of the diffusion term, the velocity solution at the interfaces would be multiply defined, as only the gradient of the velocities are solved for. Therefore, the symmetric interior penalty (SIP) method is used to ensure unique solutions at the interfaces. The SIP method consists of adding a penalty term and a symmetrizing term. The penalty term forces the velocities at the interfaces together while the symmetrizing term ensures that the resulting discretization becomes symmetric. For the x-momentum equation, i.e. for a single velocity component u_h of the velocity vector $\mathbf{u}_h$ and scalar component g_x of $\mathbf{g}_\mathcal{D}$, this leads to the following discretization for the diffusion term, [GLS19]

$$\begin{aligned}
&\underbrace{\iint_{\Omega_h^e} \nabla_h \, \mathrm{u}_h^e \cdot \nabla_h v_h \, \mathrm{d}\Omega}_{\text{volume term}} \quad \underbrace{+ \sum_{f \in \mathcal{F}_h^e} \frac{\sigma}{h_f} \int_{\Gamma_{hf}^e} [\![\mathrm{u}_h]\!] \cdot [\![v_h]\!] \, \mathrm{d}\Gamma}_{\text{penalty term}} \quad \underbrace{- \sum_{f \in \mathcal{F}_h^e} \int_{\Gamma_{hf}^e} \{\!\{ \nabla_h \, \mathrm{u}_h \}\!\} \, \mathbf{n}_f \cdot [\![v_h]\!] \, \mathrm{d}\Gamma}_{\text{flux term}} \\
&\underbrace{- \sum_{f \in \mathcal{F}_h^e} \int_{\Gamma_{hf}^e} [\![\mathrm{u}_h]\!] \cdot \{\!\{ \nabla_h v_h \}\!\} \, \mathbf{n}_f \, \mathrm{d}\Gamma}_{\text{symmetrizing term}} \quad \underbrace{- \sum_{f \in \mathcal{F}_h^{e,\partial}} \frac{\sigma}{h_f} \int_{\Gamma_{hf}^e} \mathrm{g}_x \cdot v_h \, \mathrm{d}\Gamma}_{\text{penalty boundary term}} \quad \underbrace{+ \sum_{f \in \mathcal{F}_h^{e,\partial}} \int_{\Gamma_{hf}^e} \mathrm{g}_x \cdot (\nabla_h v_h) \, \mathbf{n}_f \, \mathrm{d}\Gamma}_{\text{symmetrizing boundary term}}, \\
&\forall \, v_h \in \mathcal{V}_h, \quad \forall e,
\end{aligned} \tag{3.27}$$

where σ is the penalty parameter which penalizes the velocity jump over the element faces and h_f is a length scale of the face. The penalty parameter determines how much the

[2]Green's first identity is equivalent to using $\nabla \cdot (ab) = \nabla a \cdot b + a \nabla \cdot b$ followed by applying Gauss' divergence theorem

velocities at the element interfaces are forced to each other and has a critical value, which will be discussed at the end of this section.

The jump and average operators are obtained from equations (3.18) and (3.19),

$$\llbracket \mathrm{u}_h \rrbracket = \mathrm{u}_h^e - \mathrm{u}_h^n,\ \llbracket v_h \rrbracket = v_h,\ \{\!\{\nabla_h \mathrm{u}_h\}\!\} = \frac{1}{2}\left(\nabla_h \mathrm{u}_h^e + \nabla_h \mathrm{u}_h^n\right),\ \{\!\{\nabla_h v_h\}\!\} = \frac{1}{2}\left(\nabla_h v_h\right), \quad \forall f \in \mathcal{F}_h^{e,i},$$
$$\llbracket \mathrm{u}_h \rrbracket = \mathrm{u}_h^e,\ \llbracket v_h \rrbracket = v_h,\ \{\!\{\nabla_h \mathrm{u}_h\}\!\} = \nabla_h \mathrm{u}_h^e,\ \{\!\{\nabla_h v_h\}\!\} = \nabla_h v_h, \quad \forall f \in \mathcal{F}_h^{e,\partial}. \tag{3.28}$$

Analogous to the derivation of the discretization of the continuity equation given in section 3.3.2, the faces $\mathcal{F}_h^e$ are split into interior and boundary faces, the jump and average operators are substituted in equation (3.27), the terms are rearranged and finally the test function v_h is projected into the space spanned by the velocity basis functions ψ_j^{u}, such that for discretization of the diffusion term in the x-momentum equation the N_{DOF}^{u} equations are obtained

$$\begin{aligned}
&\underbrace{\iint_{\Omega_h^e} \nabla_h \mathrm{u}_h^e \cdot \nabla_h \psi_j^{\mathrm{u}} \,\mathrm{d}\Omega}_{\text{volume term}} \\
&\underbrace{+ \sum_{f \in \mathcal{F}_h^{e,i}} \frac{\sigma}{h_f} \int_{\Gamma_{hf}^e} \psi_j^{\mathrm{u}} \left(\mathrm{u}_h^e - \mathrm{u}_h^n\right) \mathrm{d}\Gamma + \sum_{f \in \mathcal{F}_h^{e,\partial}} \frac{\sigma}{h_f} \int_{\Gamma_{hf}^e} \psi_j^{\mathrm{u}} \left(\mathrm{u}_h^e - \mathrm{g}_x\right) \mathrm{d}\Gamma}_{\text{penalty terms}} \\
&\underbrace{- \sum_{f \in \mathcal{F}_h^{e,i}} \int_{\Gamma_{hf}^e} \frac{1}{2} \psi_j^{\mathrm{u}} \left(\nabla_h \mathrm{u}_h^e + \nabla_h \mathrm{u}_h^n\right) \cdot \mathbf{n}_f \,\mathrm{d}\Gamma - \sum_{f \in \mathcal{F}_h^{e,\partial}} \int_{\Gamma_{hf}^e} \psi_j^{\mathrm{u}} \left(\nabla_h \mathrm{u}_h^e\right) \cdot \mathbf{n}_f \,\mathrm{d}\Gamma}_{\text{flux terms}} \\
&\underbrace{- \sum_{f \in \mathcal{F}_h^{e,i}} \int_{\Gamma_{hf}^e} \frac{1}{2} \left(\mathrm{u}_h^e - \mathrm{u}_h^n\right) \left(\nabla_h \psi_j^{\mathrm{u}}\right) \cdot \mathbf{n}_f \,\mathrm{d}\Gamma - \sum_{f \in \mathcal{F}_h^{e,\partial}} \int_{\Gamma_{hf}^e} \left(\mathrm{u}_h^e - \mathrm{g}_x\right) \left(\nabla_h \psi_j^{\mathrm{u}}\right) \cdot \mathbf{n}_f \,\mathrm{d}\Gamma,}_{\text{symmetrizing terms}} \\
&j = 0, \ldots, N_{DOF}^{\mathrm{u}} - 1, \quad \forall e.
\end{aligned} \tag{3.29}$$

The same can be done for the discretization of the y-momentum equation by replacing u by v and g_x by g_y in equation (3.29). The set of discrete x- and y-momentum equations corresponds to the discretization of the $-\boldsymbol{\Delta}$-block of the Stokes system in equation (3.13) and the discrete x-momentum equations to the (minus) Laplacian in the Poisson equation in equation (3.14).

The stability of the discretization of the diffusion term depends on the penalty parameter σ. The critical value of σ depends on the element size and shape. Choosing an arbitrarily high value for σ is generally not a good idea since it will negatively impact the conditioning of the resulting system of equations, making it more difficult to solve. An analytical derivation of the critical value of σ for different element types, including quadrilateral elements, is done in [Hil13]. A critical value of $(p+1)^2$ is found for quadrilateral elements. The author notes that the derivations are done for elements with a constant mapping. For curved elements, the derivation is much more complex and depends on the precise mapping, making it difficult to derive a general critical value. However, provided that the elements are not too distorted, the values should be very similar.

Pressure gradient term

The discretization of the pressure gradient term in equation (3.25) follows from applying integration by parts on it, followed by Gauss' theorem, which in terms of the jump and average operators give [GLS19],

$$-\iint_{\Omega_h^e} \mathrm{p}_h^e \left(\nabla_h v_h\right) \mathrm{d}\Omega + \sum_{f \in \mathcal{F}_h^e} \int_{\Gamma_{hf}^e} \left(\llbracket v_h \rrbracket \cdot \mathbf{n}_f\right) \{\!\{\mathrm{p}_h\}\!\} \,\mathrm{d}\Gamma\,, \qquad \forall\, v_h \in \mathcal{V}_h, \quad \forall e, \tag{3.30}$$

where the jump and average operators are defined as

$$\begin{aligned} \llbracket v_h \rrbracket = v_h, \quad \{\!\{\mathrm{p}_h\}\!\} &= \frac{1}{2}\left(\mathrm{p}_h^e + \mathrm{p}_h^n\right), \qquad \forall f \in \mathcal{F}_h^{e,i}, \\ \llbracket v_h \rrbracket = v_h, \quad \{\!\{\mathrm{p}_h\}\!\} &= \mathrm{p}_h^e, \qquad \forall f \in \mathcal{F}_h^{e,\partial}. \end{aligned} \tag{3.31}$$

Combining equations (3.30) and (3.31), splitting the faces and projecting v_h into the space spanned by the velocity basis functions ψ_j^{u}, the discretization of the pressure gradient term is given by the set of N_{DOF}^{u} equations

$$\begin{gathered} -\iint_{\Omega_h^e} \mathrm{p}_h^e \left(\nabla_h \psi_j^{\mathrm{u}}\right) \mathrm{d}\Omega + \sum_{f \in \mathcal{F}_h^{e,i}} \int_{\Gamma_{hf}^e} \frac{1}{2}\left(\mathrm{p}_h^e + \mathrm{p}_h^n\right) \psi_j^{\mathrm{u}}\, \mathbf{n}_f \,\mathrm{d}\Gamma + \sum_{f \in \mathcal{F}_h^{e,\partial}} \int_{\Gamma_{hf}^e} \mathrm{p}_h^e\, \psi_j^{\mathrm{u}}\, \mathbf{n}_f \,\mathrm{d}\Gamma\,, \\ j = 0, \ldots, N_{DOF}^{\mathrm{u}} - 1, \quad \forall e. \end{gathered} \tag{3.32}$$

Since equation (3.32) is in vector-form, a total of $2N_{DOF}^{\mathrm{u}}$ equations (corresponding to the discretization of the pressure gradient term in the x- and y-momentum equations) are obtained for the discretization of the $\boldsymbol{\mathcal{G}}$-block in equation (3.13).

Source term

The discretization of the source term in equation (3.25) is as straightforward as projecting v_h into the space spanned by the basis functions ψ_j^{u}, leading to the set of N_{DOF}^{u} equations

$$-\iint_{\Omega_h^e} \mathbf{f}\, \psi_j^{\mathrm{u}} \,\mathrm{d}\Omega\,, \qquad j = 0, \ldots, N_{DOF}^{\mathrm{u}} - 1, \quad \forall e, \tag{3.33}$$

Again, equation (3.33) is in vector-form leading to a total of $2N_{DOF}^{\mathrm{u}}$ equations for the discretization of the Stokes source $\mathbf{f}$ in equation (3.13) and N_{DOF}^{u} equations (by only taking x-momentum into account) for the discretization of the Poisson source f in equation (3.14), which can be added to the right-hand side by multiplying them by -1.

3.3.4 Divergence-free constraint stabilization

A property that is tied to the divergence-free condition is the continuity of the normal velocity components over an element face. In fact, continuity of the normal velocities is a necessary requirement for an exactly (so not only in a discrete sense) divergence-free method, [Sch19]. Pressure-robustness can be improved by using the stabilization given by given by [Sch19],

$$\begin{aligned}\iint_{\Omega_h^e} \gamma \left(\nabla_h \cdot \mathbf{u}_h^e\right) \left(\nabla_h v_h\right) \mathrm{d}\Omega + \sum_{f \in \mathcal{F}_h^e} \frac{\gamma}{h_f} \int_{\Gamma_{hf}^e} \left(\llbracket \mathbf{u}_h \rrbracket \cdot \mathbf{n}_f\right) \left(\llbracket v_h \rrbracket \, \mathbf{n}_f\right) \mathrm{d}\Gamma \\ - \sum_{f \in \mathcal{F}_h^{e,\partial}} \frac{\gamma}{h_f} \int_{\Gamma_{hf}^e} \left(\mathbf{g}_{\mathcal{D}} \cdot \mathbf{n}_f\right) \left(v_h \, \mathbf{n}_f\right) \mathrm{d}\Gamma \, \forall \, v_h \in \mathcal{V}_h, \quad \forall e,\end{aligned} \tag{3.34}$$

which must be added to the discretization of the momentum equations discussed before. This stabilization includes a grad-div stabilization in the first term which arises from adding $\mathbf{0} = -\gamma\nabla(\nabla \cdot \mathbf{u})$ to the continuous momentum equations and a normal velocity penalization. As a result, the element-wise divergence-free constraint and normal velocity continuity property are fulfilled more accurately. The stabilization parameter γ determines how much the normal velocities at the interface of an element are forced to each other.

The jump and average operators in this case correspond to the vector equivalent of those written in equation (3.28). Working out these operators in equation (3.34), splitting the faces and projecting v_h into the space spanned by the basis functions ψ_j^{u},

$$\begin{aligned}\iint_{\Omega_h^e} \gamma \left(\nabla_h \cdot \mathbf{u}_h^e\right) \left(\nabla_h \psi_j^{\mathrm{u}}\right) \mathrm{d}\Omega + \sum_{f \in \mathcal{F}_h^{e,i}} \frac{\gamma}{h_f} \int_{\Gamma_{hf}^e} \left(\mathbf{u}_h^e - \mathbf{u}_h^n\right) \cdot \mathbf{n}_f \, \psi_j^{\mathrm{u}} \, \mathbf{n}_f \, \mathrm{d}\Gamma \\ + \sum_{f \in \mathcal{F}_h^{e,\partial}} \frac{\gamma}{h_f} \int_{\Gamma_{hf}^e} \left(\mathbf{u}_h^e - \mathbf{g}_{\mathcal{D}}\right) \cdot \mathbf{n}_f \, \psi_j^{\mathrm{u}} \, \mathbf{n}_f \, \mathrm{d}\Gamma \, , \\ j = 0, \ldots, N_{DOF}^{\mathrm{u}} - 1, \quad \forall e\end{aligned} \tag{3.35}$$

Looking at the dimensions of equation (3.35), the stabilization parameter γ must have a dimension of m^2/s, i.e. a kinematic viscosity. Therefore, a logical choice would be to base the value of γ on ν. In this specific case however, the dimensions of γ do not really matter as the viscosity has been set to 1 (see section 3.2), such that the Stokes problem is essentially dimensionless. Ideally, γ is chosen as high as possible, since a large value leads to an (almost) exactly divergence-free solution, or in other words, pressure-robustness. However, choosing an arbitrarily large γ also increases the condition number of the system of equations, making it harder to solve, [Joh+17]. A study of optimal values of γ with respect to different error norms is done in [Jen+14]. It turns out that the optimal value depends on multiple factors and is difficult to estimate a priori. However, [Jen+14] has observed that for certain cases taking $\gamma = \mathcal{O}(1)$ leads to an enormous improvement of pressure-robustness.

Lastly, it could also be argued that choosing a high value of the penalty parameter σ has the same effect on pressure-robustness as this forces the velocities over the faces to be closer to each other, thus also forcing continuity of the normal velocity over the faces.

3.3.5 Characteristics of the discrete systems

Combining equations (3.24), (3.29), (3.33) and (3.35), substituting the modal polynomial expansion of the local solutions using equation (2.20) and evaluating the integrals by mapping each element to the standard element and applying an appropriate quadrature rule, a set discrete linear system of algebraic Stokes equations of the form $\boldsymbol{\mathcal{A}}\boldsymbol{\xi} = \boldsymbol{b}$ is obtained where $\boldsymbol{\mathcal{A}}$ is the coefficient matrix, $\boldsymbol{\xi}$ are the unknown modal expansion coefficients and $\boldsymbol{b}$ is the right-hand side (RHS) vector. Analogously, a similar system of equations for the Poisson problem is obtained by combining equations (3.29) and (3.33) and evaluating the integrals. The aim is to solve the modal expansion coefficients from these systems of equations.

The coefficient matrix holds the contributions from the elements and the connection to their neighboring elements. All known quantities such as for example prescribed values on the boundary (Dirichlet boundary conditions) or the contribution of external forces are added to the RHS vector $\boldsymbol{b}$. The coefficient matrix $\boldsymbol{\mathcal{A}}$ has size of $M \times M$ and the RHS vector $\boldsymbol{b}$ has a length of M, where $M = N_e N_{DOF}$. In two dimensions, $N_{DOF} = (p+1)^2$ for the Poisson problem and $N_{DOF} = (p_\mathrm{u}+1)^2 \cdot 2 + (p_\mathrm{p}+1)^2$ for the Stokes problem.

In order to illustrate how the system of equations is assembled, consider the Cartesian 4×4 grid depicted in figure 3.2 with numbering $k = j(N_{e_i}+1)+i$, $k = 0,\dots,K-1$ consisting of 16 2D elements, where N_{e_i} is the number of elements in i-direction. Adding

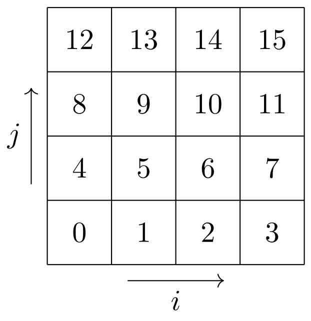


Figure 3.2: Illustration of a two-dimensional Cartesian grid with numbering $k = j(N_{e_i} + 1) + i$.

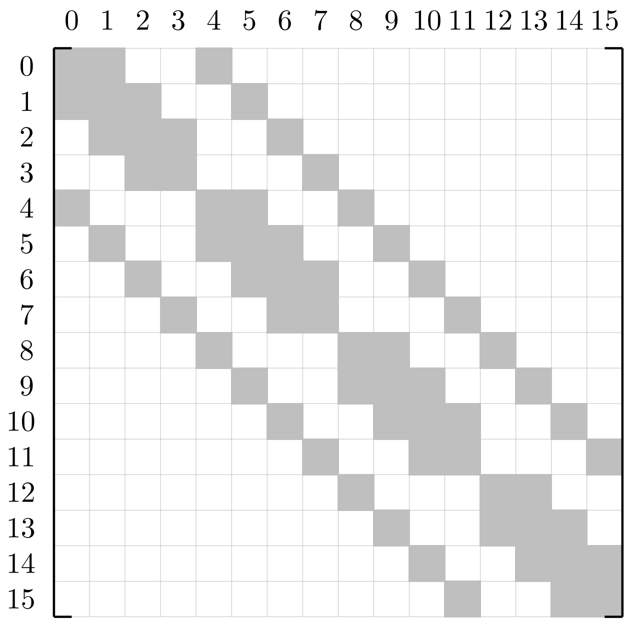


Figure 3.3: Illustration of the coefficient matrix corresponding to the discretization on a two-dimensional grid with order p elements.

the contributions of the elements and its neighbors results in the coefficient block matrix illustrated in figure 3.3. Each block row corresponds to an element and each block column to a contribution to it from itself and its neighbors. For example, element 0 has a contribution from element 0 (itself), 1 and 4. Each block stores the coefficients corresponding to the DOFs of the element. For $p_\mathrm{u} = p_\mathrm{v} = 2$ and $p_\mathrm{p} = 1$ in 2D, a block contains $(2+1)^2 \cdot 2 + (1+1)^2 = 22$ DOFs and $22^2 = 484$ coefficients. The RHS vector is partitioned very similar to the coefficient block matrix, i.e.

$$\boldsymbol{b} = [\bar{\boldsymbol{b}}_0^\mathrm{u},\, \bar{\boldsymbol{b}}_0^\mathrm{v},\, \bar{\boldsymbol{b}}_0^\mathrm{p}, \quad \bar{\boldsymbol{b}}_1^\mathrm{u},\, \bar{\boldsymbol{b}}_1^\mathrm{v},\, \bar{\boldsymbol{b}}_1^\mathrm{p} \quad \dots \quad \bar{\boldsymbol{b}}_{14}^\mathrm{u},\, \bar{\boldsymbol{b}}_{14}^\mathrm{v},\, \bar{\boldsymbol{b}}_{14}^\mathrm{p}, \quad \bar{\boldsymbol{b}}_{15}^\mathrm{u},\, \bar{\boldsymbol{b}}_{15}^\mathrm{v},\, \bar{\boldsymbol{b}}_{15}^\mathrm{p}]^T \tag{3.36}$$

where $\bar{\boldsymbol{b}}_k^\mathrm{u}$, $\bar{\boldsymbol{b}}_k^\mathrm{v}$ and $\bar{\boldsymbol{b}}_k^\mathrm{p}$ hold the RHS entries corresponding to respectively the u, v and p DOFS, for the element k.

Storing the system of equations in this "local" order ($\boldsymbol{\xi} = [\bar{\mathbf{u}}_0, \bar{\mathbf{v}}_0, \bar{\mathbf{p}}_0, \dots \bar{\mathbf{u}}_{N_e-1}, \bar{\mathbf{v}}_{N_e-1}, \bar{\mathbf{p}}_{N_e-1}]^T$ with $\bar{\mathbf{u}} = [\mathrm{u}_0, \dots, \mathrm{u}_{N_{DOF}^\mathrm{u}-1}]^T$) is straightforward and simple. However, another possibility is to first store all entries for u, followed by all entries for v and lastly all entries for p. This leads to a "global" ordering ($\boldsymbol{\xi} = [\bar{\mathbf{u}}_0, \dots \bar{\mathbf{u}}_{N_e-1}, \bar{\mathbf{v}}_0 \dots \bar{\mathbf{v}}_{N_e-1}, \bar{\mathbf{p}}_0 \dots \bar{\mathbf{p}}_{N_e-1}]^T$) in which the discrete equivalent of the continuous saddle point problem in equation (3.13) is formed,

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{G} \\ \boldsymbol{D} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \mathbf{0} \end{bmatrix}. \tag{3.37}$$

Here, $\boldsymbol{A}$ encompasses the discretization of the Laplace operator $-\boldsymbol{\Delta}$, $\boldsymbol{G}$ that of the gradient operator $\boldsymbol{\mathcal{G}}$ and $\boldsymbol{D}$ that of the divergence operator $\boldsymbol{\mathcal{D}}$. Since $\boldsymbol{A}$ is the result of a mimetic discretization of the Laplace operator, it must be symmetric positive definite (SPD), meaning that it must be symmetric and all its eigenvalues must be positive. This is because the Laplacian has a maximum principle whose discrete equivalent is a symmetric positive definite system. This property is also important because it dictates the convergence of relaxation methods, which will become clear in section 4.2. These properties also apply to the coefficient matrix of the Poisson equation, since that is also a discretization of the Laplace operator. Since the SIP method was used for the discretization of the Laplace operator, $\boldsymbol{A}$ should be SPD. However, for the reasons mentioned earlier it is an important property to verify. Moreover, as a result of the design of the discretization of the continuity equation and the pressure gradient term in the momentum equations, $\boldsymbol{D} = \boldsymbol{G}^T$, such that also the block-coefficient matrix in equation (3.37) is symmetric.

Since the number of neighboring elements is generally relatively small compared to the total number of elements, the coefficient matrix is sparse which is illustrated in figure 3.4. This is a property that is specific to a coefficient matrix arising from the discretization of any PDE, [Maz16]. Storing the coefficient matrix in full form is generally not a good idea. Consider for example the coefficient matrix resulting from the discretization of the Stokes problem on a 3D grid with 16 elements in each direction and a polynomial order of 2 for the velocity solutions and 1 for the pressure solution, i.e. 22 DOFs per element. This matrix will have a size of $(16 \times 16 \times 16 \times 22)^2 = 8.1 \cdot 10^9$, for which $8.1 \cdot 10^9 \times 8 = 6.5 \cdot 10^{10}$ bytes or 65 GB needs to be allocated to store it with double precision. This shows that for a rather small and low order case, already a lot of memory is required to store this matrix which is simply not practical. A much better alternative is to store it in block compressed sparse row (CSR) format. This way, only three arrays need to be stored, one containing the values

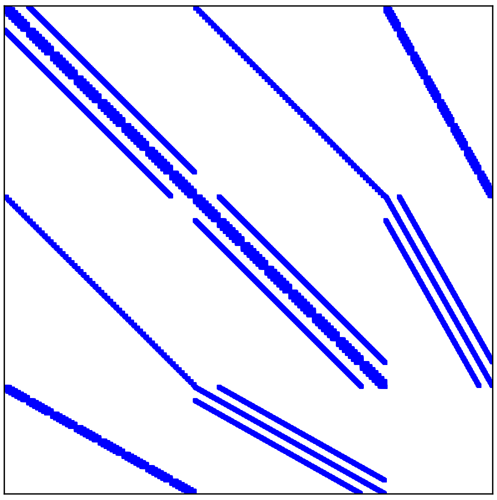


Figure 3.4: The sparsity pattern of the globally ordered coefficient matrix of a Stokes discretization.

of the non-zero blocks, one containing the column indices of the non-zero blocks and one containing pointers to the beginning of each row in the first two arrays. For more details about the block CSR format and other storage schemes for sparse matrices, see [Saa03].

The structure and sparsity of the coefficient matrices resulting from the discretization of the Stokes and Poisson problems makes it challenging to efficiently compute solutions to these systems. This will be addressed in the next chapter.

Chapter 4

Solving the discrete system

Performing the discretization as explained in the previous chapter results in a system of linear algebraic equations of the form $\boldsymbol{\mathcal{A}}\boldsymbol{\xi} = \boldsymbol{b}$, where $\boldsymbol{\mathcal{A}}$ is the coefficient matrix, $\boldsymbol{\xi}$ is the solution vector and $\boldsymbol{b}$ is the right-hand side vector. Methods for solving such a system can generally be divided into two categories, direct and iterative methods.

Direct methods are designed to obtain exact solutions to linear systems of equations. The solutions are exact in the sense that the system of equations resulting from the discretization are solved exactly, which however does not necessarily mean that the governing PDE is solved exactly. An example of a direct method is by simply left-multiplying the right-hand side vector with the inverse of the coefficient matrix, $\boldsymbol{\xi} = \boldsymbol{\mathcal{A}}^{-1}\boldsymbol{b}$. Another option is to perform Gaussian (or Gauss-Jordan) elimination in which the equations are successively substituted into each other, until only the unknowns are left. A variant of Gaussian elimination is the LU factorization method which has the advantage that when the LU factorization is performed, the linear system can be solved for different values of the right-hand side vector $\boldsymbol{b}$, without performing additional elimination, [MMD16]. These methods are straightforward, often easy to implement and require no assumption about the nature of the coefficient matrix.

Key properties of the coefficient matrices that arise from the FE discretization of PDEs are that they are large and sparse, which was illustrated for the DG discretization in section 3.3.5. Alternative direct methods exist that exploit the sparsity and structure of the coefficient matrix such as the banded linear system solvers given in [Maz16; MMD16]. While these solvers are already more efficient than direct solvers that use the full coefficient matrix, the memory and computational requirements for solving systems arising in CFD applications can seriously challenge the most efficient direct solvers, [Saa03]. Therefore, iterative methods are generally better suited for solving this kind of systems efficiently. Moreover, the implementation of parallel computing can be done more easily in iterative methods than in direct methods, [Saa03]. Hence, the focus of this text will be on iterative solvers.

First a general description of an iterative method is given in section 4.1. Thereafter, Jacobi, Gauss-Seidel and distributive Gauss-Seidel relaxation methods are introduced in section 4.2. While relaxation methods possess excellent error smoothing capabilities, they are almost never used as standalone iterative solvers because of their slow convergence. Multigrid methods aim to exploit these smoothing properties which is explained in section 4.3.

4.1 Iterative methods

In an iterative method, the solution to the system of equations is obtained by computing a sequence of approximate solutions, starting from an initial guess $\boldsymbol{\xi}^{(0)}$, where the new approximation $\boldsymbol{\xi}^{(k+1)}$ is calculated from the previous approximation $\boldsymbol{\xi}^{(k)}$. Depending on the linear system and the method used, an iterative solver will converge to the exact solution. However, in practical applications it often suffices to solve until a certain accuracy is reached, for example measured by a L_p-norm of the residual $\boldsymbol{\mathcal{R}} = \boldsymbol{b} - \boldsymbol{\mathcal{A}}\boldsymbol{\xi}$.

Decomposing the coefficient matrix as $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{N}}$, the system of equations $\boldsymbol{\mathcal{A}}\boldsymbol{\xi} = \boldsymbol{b}$ can be rewritten as

$$(\boldsymbol{\mathcal{M}} - \boldsymbol{\mathcal{N}})\boldsymbol{\xi} = \boldsymbol{b}. \tag{4.1}$$

Next, applying a fixed point iteration solution procedure, equation (4.1) yields

$$\boldsymbol{\mathcal{M}}\boldsymbol{\xi}^{(k+1)} = \boldsymbol{\mathcal{N}}\boldsymbol{\xi}^{(k)} + \boldsymbol{b}, \tag{4.2}$$

which can be further rewritten to

$$\boldsymbol{\xi}^{(k+1)} = \boldsymbol{\mathcal{M}}^{-1}\boldsymbol{\mathcal{N}}\boldsymbol{\xi}^{(k)} + \boldsymbol{\mathcal{M}}^{-1}\boldsymbol{b}. \tag{4.3}$$

For the iterative method to be convergent, the spectral radius of $\boldsymbol{B} = \boldsymbol{\mathcal{M}}^{-1}\boldsymbol{\mathcal{N}}$ must be smaller than 1, i.e. $\rho(\boldsymbol{B}) = \max\left(|\lambda_i(\boldsymbol{B})|\right) < 1$, where λ_i are the eigenvalues of $\boldsymbol{B}$. The asymptotic speed of convergence of an iterative method is also determined by the spectral radius. If $\rho\left(\boldsymbol{B}\right)$ is bounded away from 1, only a few iterations are needed to obtain sufficient error reduction leading to fast convergence. On the contrary, if $\rho\left(\boldsymbol{B}\right)$ is close to unity, the error is only reduced by a very small amount leading to slow convergence.

Different choices for $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{N}}$ will lead to different iterative methods which will be treated in the next section.

4.2 Relaxation methods

Relaxation methods are iterative methods where, given a current approximation to the solution in each grid point, a new approximation is computed by changing the value in each grid point such that the local equation in that grid point is satisfied. In other words, the local residual in each grid point should vanish. One relaxation - or sweep - is completed when a new approximation to the solution is computed in all grid points.

Well-known relaxation methods are Jacobi and Gauss-Seidel relaxation, which are very similar but differ in which points are used to compute the new approximation. Jacobi relaxation always uses the old values in the surrounding grid points to compute the new approximation whereas Gauss-Seidel relaxation takes already relaxed points into account. Therefore, the order in which the grid points are relaxed does not matter for Jacobi relaxation, but it does make a difference for Gauss-Seidel relaxation. In Gauss-Seidel relaxation the grid points are often relaxed using lexicographic ordering, i.e. in order of increasing grid indices, [VL00].

For the sake of illustration, define the splitting of the coefficient matrix $\boldsymbol{\mathcal{A}} = \boldsymbol{D} - \boldsymbol{E} - \boldsymbol{F}$, where $\boldsymbol{D}$ is the diagonal part of $\boldsymbol{A}$, $-\boldsymbol{E}$ its strict lower part and $-\boldsymbol{F}$ its strict upper part, as depicted in figure 4.1.

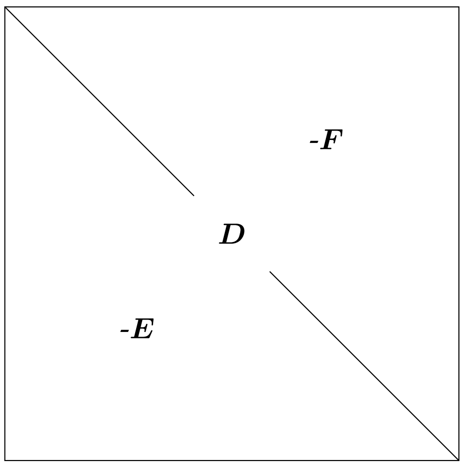


Figure 4.1: The splitting of the matrix $\boldsymbol{\mathcal{A}}$ into $\boldsymbol{D}$, $-\boldsymbol{E}$ and $-\boldsymbol{F}$

4.2.1 Jacobi

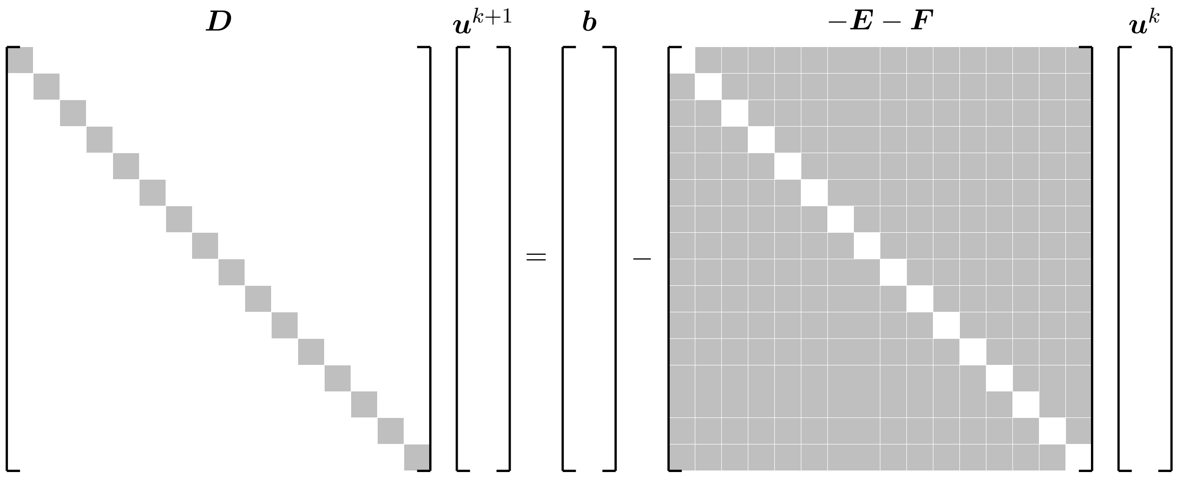


Figure 4.2: Illustration of the Jacobi relaxation method.

Provided that the diagonal elements of the coefficient matrix are nonzero, each i-th equation can be used to solve for the new approximation of the solution $\xi_i^{(k+1)}$, which is illustrated in figure 4.2.In Jacobi relaxation, the old values $\xi_i^{(k)}$ (the off-diagonal elements) are used to approximate the new solution $\xi_i^{(k+1)}$. This procedure leads to the following equation for $\xi_i^{(k+1)}$

$$\xi_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{M} a_{ij} \xi_j^{(k)} \right), \qquad i = 1, \ldots, M, \tag{4.4}$$

where $M = N_e N_{DOF}$ is the total number unknowns. Combining equations (4.3) and (4.4) gives the Jacobi relaxation in vector-form

$$\boldsymbol{\xi}^{(k+1)} = \boldsymbol{D}^{-1}(\boldsymbol{E}+\boldsymbol{F})\boldsymbol{\xi}^{(k)} + \boldsymbol{D}^{-1}\boldsymbol{b}, \tag{4.5}$$

with

$$\boldsymbol{B}_J = \boldsymbol{D}^{-1}(\boldsymbol{E}+\boldsymbol{F}). \tag{4.6}$$

Consequently, Jacobi relaxation converges when $\rho(\boldsymbol{B}_J) < 1$. It can be proven that this condition is satisfied when both $\boldsymbol{\mathcal{A}}$ and $2\boldsymbol{D} - \boldsymbol{\mathcal{A}}$ are symmetric and positive definite matrices (SPD), [QSS07]. In an attempt to influence convergence rates, the Jacobi method may be adjusted by taking a relaxation factor ω into account, where $\omega < 1$ is referred to under-relaxation or damped Jacobi,

$$\boldsymbol{\xi}^{(k+1)} = \omega(\boldsymbol{D}^{-1}(\boldsymbol{E}+\boldsymbol{F})\boldsymbol{\xi}^{(k)} + \boldsymbol{D}^{-1}\boldsymbol{b}) + (1-\omega)\boldsymbol{\xi}^{(k)}. \tag{4.7}$$

As new approximations are computed solely from the old approximations in the Jacobi method, two solution vectors need to be stored in memory, one for the old approximations $\boldsymbol{\xi}^{(k)}$ and one for the new approximations $\boldsymbol{\xi}^{(k+1)}$.

4.2.2 Gauss-Seidel

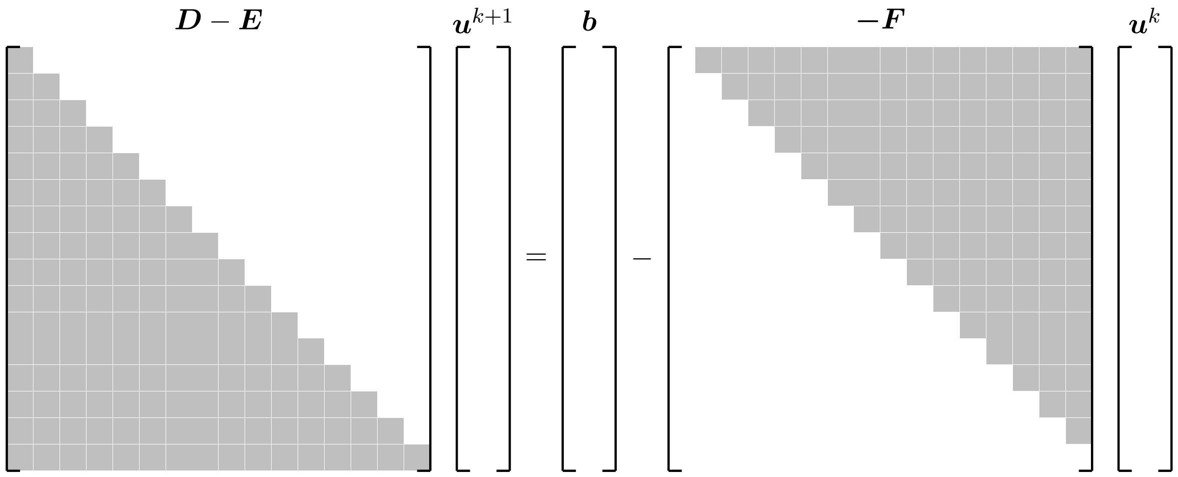


Figure 4.3: Illustration of the Gauss-Seidel relaxation method.

Similar to the Jacobi method, the Gauss-Seidel method approximates the new solution $\boldsymbol{\xi}^{(k+1)}$ using the the previous approximation $\boldsymbol{\xi}^{(k)}$. However, previously relaxed points are now also taken into account (the below-diagonal elements), as illustrated in figure 4.3. When the equations are relaxed in lexicographic order, i.e. with increasing $i = 1, \dots, M$, the following equation for $\xi_i^{(k+1)}$ is obtained

$$\xi_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}\xi_j^{(k+1)} - \sum_{j=i+1}^{M} a_{ij}\xi_j^{(k)}\right), \qquad i = 1, \dots, M, \tag{4.8}$$

which is referred to as *forward* Gauss-Seidel relaxation. Similar to the Jacobi method, the Gauss-Seidel method can be written in vector-form by combining equations (4.3) and (4.8) yielding

$$\boldsymbol{\xi}^{(k+1)} = (\boldsymbol{D} - \boldsymbol{E})^{-1}\boldsymbol{F}\boldsymbol{\xi}^{(k)} + (\boldsymbol{D} - \boldsymbol{E})^{-1}\boldsymbol{b}, \tag{4.9}$$

with

$$\boldsymbol{B}_{GS} = (\boldsymbol{D} - \boldsymbol{E})^{-1}\boldsymbol{F}, \tag{4.10}$$

which shows that the method is convergent when $\rho\left(\boldsymbol{B}_{GS}\right) < 1$.

Moreover, A *backward* Gauss-Seidel relaxation scheme is obtained when the equations are relaxed with decreasing i, i.e. $i = M, \ldots, 1$, leading to

$$\boldsymbol{\xi}^{(k+1)} = (\boldsymbol{D} - \boldsymbol{F})^{-1}\boldsymbol{E}\boldsymbol{\xi}^{(k)} + (\boldsymbol{D} - \boldsymbol{F})^{-1}\boldsymbol{b}, \tag{4.11}$$

which is convergent when $\rho\left((\boldsymbol{D} - \boldsymbol{F})^{-1}\boldsymbol{E}\right) < 1$. A *symmetric* Gauss-Seidel scheme is obtained by performing one forward followed by one backward Gauss-Seidel sweep.

The Gauss-Seidel method can be proven to be convergent When $\boldsymbol{\mathcal{A}}$ is SPD, [QSS07]. Gauss-Seidel relaxation generally convergences faster than Jacobi relaxation, [VL00; QSS07; MMD16]. Moreover, since the most recent values of the approximate new solution are used, only a single solution vector is needed to perform Gauss-Seidel relaxation. Therefore less memory needs to be allocated to perform the Gauss-Seidel method compared to the Jacobi method. Moreover, the discussed so-called point relaxation methods of Jacobi and Gauss-Seidel can be extended to block relaxation schemes by using the block decomposition of $\boldsymbol{\mathcal{A}} = \boldsymbol{D} - \boldsymbol{E} - \boldsymbol{F}$ for non-singular invertible block diagonal matrices $\boldsymbol{D}$.

When Jacobi and Gauss-Seidel relaxation are applied to discretizations of the Poisson equation, it can be derived that the largest eigenvalue of their iteration matrices, $\rho(\boldsymbol{B}_J)$ and $\rho(\boldsymbol{B}_{GS})$ increases with the mesh size h with order $1 - \mathcal{O}\left(h^2\right)$, [VL00]. So, for increasing grid sizes, the spectral radius of the iteration matrices goes to 1 leading to slow convergence.

Due to the local nature of the process, the relaxation schemes are very effective in reducing high-frequency error components, while smooth error components are hardly effected. This is the case for many iterative solvers applied to systems of equations resulting from the discretization of elliptic partial differential equations, [VL00]. Therefore, relaxation methods are almost never used as standalone iterative solvers. However, their error smoothing properties can be exploited to design solvers of optimal complexity, where the problem is solved in an amount of operations comparable to the problem size. The algorithm behind this is explained in section 4.3.

4.2.3 Distributive Gauss-Seidel

Applying Gauss-Seidel relaxation to the Stokes problem requires special attention. The smoother should smooth the error for all unknowns in the equations, which is not the case for the saddle point system of the Stokes equations, [OL06]. A solution to this is to decouple the original system and perform Gauss-Seidel relaxation on the decoupled system on which relaxation is known to be effective, which is known as distributive Gauss-Seidel relaxation.

To this end, consider the discrete Stokes system

$$\mathcal{L}\begin{bmatrix}\mathbf{u}\\ \mathbf{p}\end{bmatrix}=\begin{bmatrix}\mathbf{f}\\ \mathbf{0}\end{bmatrix}, \qquad \mathcal{L}=\begin{bmatrix}\boldsymbol{A} & \boldsymbol{G}\\ \boldsymbol{D} & \mathbf{0}\end{bmatrix} \tag{4.12}$$

and the updates

$$\mathbf{u}^{k+1}=\mathbf{u}^{k}+\Delta\,\mathbf{u}, \tag{4.13}$$

$$\mathbf{p}^{k+1}=\mathbf{p}^{k}+\Delta\,\mathbf{p}, \tag{4.14}$$

to write

$$\begin{bmatrix}\boldsymbol{A} & \boldsymbol{G}\\ \boldsymbol{D} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\Delta\,\mathbf{u}\\ \Delta\,\mathbf{p}\end{bmatrix}=\begin{bmatrix}\mathbf{f}-\boldsymbol{A}\,\mathbf{u}^{k}-\boldsymbol{G}\,\mathbf{p}^{k}\\ -\boldsymbol{D}\,\mathbf{u}^{k}\end{bmatrix}. \tag{4.15}$$

Now, define the transformation

$$\begin{bmatrix}\Delta\,\mathbf{u}\\ \Delta\,\mathbf{p}\end{bmatrix}=\mathcal{M}\begin{bmatrix}\Delta\,\mathbf{u}^{*}\\ \Delta\,\mathbf{p}^{*}\end{bmatrix}, \tag{4.16}$$

where $\mathcal{M}$ now denotes the transformation matrix which must be chosen such that the transformed system

$$\mathcal{L}\mathcal{M}\begin{bmatrix}\Delta\,\mathbf{u}^{*}\\ \Delta\,\mathbf{p}^{*}\end{bmatrix}=\begin{bmatrix}\mathbf{f}-\boldsymbol{A}\,\mathbf{u}^{k}-\boldsymbol{G}\,\mathbf{p}^{k}\\ -\boldsymbol{D}\,\mathbf{u}^{k}\end{bmatrix} \tag{4.17}$$

can be solved effectively using Gauss-Seidel relaxation.

Classical splitting

A well-known classical solver for the Navier-Stokes equations is the SIMPLE (semi-implicit method for pressure-linked equations) algorithm, [PS83]. This splitting can also be used to decouple the Stokes system and is often used as a preconditioner for Krylov subspace methods, [BGL05]. The classical splitting corresponds to

$$\mathcal{M}=\begin{bmatrix}\boldsymbol{I} & -\boldsymbol{A}^{-1}\boldsymbol{G}\\ \mathbf{0} & \boldsymbol{I}\end{bmatrix}, \tag{4.18}$$

leading to

$$\mathcal{L}\mathcal{M}=\begin{bmatrix}\boldsymbol{A} & \mathbf{0}\\ \boldsymbol{D} & -\boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{G}\end{bmatrix}, \tag{4.19}$$

where $\boldsymbol{S}=-\boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{G}$ is known as the Schur complement. Often, $\mathcal{L}\mathcal{M}$ is approximated by

$$\mathcal{S}=\begin{bmatrix}\hat{\boldsymbol{A}} & \mathbf{0}\\ \boldsymbol{D} & \hat{\boldsymbol{S}}\end{bmatrix}, \tag{4.20}$$

where $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{S}}$ are approximations of the $\boldsymbol{A}$-block matrix and the Schur complement $\boldsymbol{S}$. In the original SIMPLE algorithm, $\boldsymbol{A}$ is approximated by its diagonal $\boldsymbol{A}_D$ and $\hat{\boldsymbol{S}}=-\boldsymbol{D}\boldsymbol{A}_D^{-1}\boldsymbol{G}$.

For DG formulations, the block diagonal is approximated. Combining equations (4.17) and (4.20) gives two decoupled systems on which Gauss-Seidel relaxation can be applied,

$$\hat{\boldsymbol{A}}\Delta\,\mathbf{u}^* = \mathbf{f} - \boldsymbol{A}\,\mathbf{u}^k - \boldsymbol{G}\,\mathbf{p}^k, \tag{4.21}$$

$$\hat{\boldsymbol{S}}\Delta\,\mathbf{p}^* = -\boldsymbol{D}(\mathbf{u}^k + \Delta\,\mathbf{u}^*), \tag{4.22}$$

which can be transformed back to the original system using equations (4.16) and (4.18),

$$\boldsymbol{A}\Delta\,\mathbf{u} = \boldsymbol{A}\Delta\,\mathbf{u}^* - \boldsymbol{G}\Delta\,\mathbf{p}, \tag{4.23}$$

$$\Delta\,\mathbf{p} = \Delta\,\mathbf{p}^*, \tag{4.24}$$

by applying a few Gauss-Seidel relaxation sweeps on equation (4.23) after which the solution can be updated with equations (4.13) and (4.14).

Least-squares splitting

The classical splitting requires the existence of good approximations to the $\boldsymbol{A}$-block and the Schur complement $\boldsymbol{S}$, which is the case when $\boldsymbol{A}$ is strongly diagonally dominant, [BGL05]. A possibly better splitting, that does not require approximations to $\boldsymbol{A}$ and $\boldsymbol{S}$ and the computation of $\boldsymbol{A}^{-1}$, is based on the least-squares commutator as proposed by [WC13],

$$\boldsymbol{\mathcal{M}} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{G} \\ \mathbf{0} & -(\boldsymbol{D}\boldsymbol{G})^{-1}\boldsymbol{D}\boldsymbol{A}\boldsymbol{G} \end{bmatrix}, \tag{4.25}$$

leading to

$$\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{M}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{P}\boldsymbol{A}\boldsymbol{G} \\ \boldsymbol{D} & \boldsymbol{D}\boldsymbol{G} \end{bmatrix} \tag{4.26}$$

where the commutator $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{G}(\boldsymbol{D}\boldsymbol{G})^{-1}\boldsymbol{D}$ is minimum in the Frobenius-norm. If $\boldsymbol{D}$ and $\boldsymbol{G}$ were square matrices, $\boldsymbol{P}$ would be a zero matrix since $\boldsymbol{G} = \boldsymbol{D}^T$. An efficient scheme is obtained by approximating $\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{M}}$ by

$$\mathcal{S} = \begin{bmatrix} \hat{\boldsymbol{A}} & \mathbf{0} \\ \boldsymbol{B} & \hat{\boldsymbol{A}}_p \end{bmatrix}, \tag{4.27}$$

leading to

$$\hat{\boldsymbol{A}}\Delta\,\mathbf{u}^* = \mathbf{f} - \boldsymbol{A}\,\mathbf{u}^k - \boldsymbol{G}\,\mathbf{p}^k, \tag{4.28}$$

$$\hat{\boldsymbol{A}}_p\Delta\,\mathbf{p}^* = -\boldsymbol{D}(\mathbf{u}^k + \Delta\,\mathbf{u}^*), \tag{4.29}$$

where the approximate solutions can be found by performing a few Gauss-Seidel relaxation sweeps on the systems

$$\boldsymbol{A}\Delta\,\mathbf{u}^* = \mathbf{f} - \boldsymbol{A}\,\mathbf{u}^k - \boldsymbol{G}\,\mathbf{p}^k, \tag{4.30}$$

$$(\boldsymbol{D}\boldsymbol{G})\Delta\,\mathbf{p}^* = -\boldsymbol{D}(\mathbf{u}^k + \Delta\,\mathbf{u}^*). \tag{4.31}$$

The transformation back to the original system is realized using equations (4.16) and (4.25),

$$\Delta\,\mathbf{u} = \Delta\,\mathbf{u}^* + \boldsymbol{G}\Delta\,\mathbf{p}^* \tag{4.32}$$

$$(\boldsymbol{D}\boldsymbol{G})\Delta\,\mathbf{p} = -\boldsymbol{D}\boldsymbol{A}\boldsymbol{G}\Delta\,\mathbf{p}^*, \tag{4.33}$$

by performing a few Gauss-Seidel sweeps on equation (4.33) and using the updates, equations (4.13) and (4.14).

4.3 Multigrid methods

In section 4.2 it was mentioned that the error smoothing properties of relaxation methods can be used to design an algorithm of optimal complexity. Generally, error components with wavelengths comparable to the grid size are reduced very effectively. When the high frequency components of the error are reduced sufficiently, i.e. the error is smooth on the scale of the grid size, a fine grid is not needed to accurately represent it. Hence, with little loss of accuracy it can be represented on a coarser grid where it can be solved much cheaper. This two-level concept forms the basis of multigrid methods, which aim to exploit the error smoothing properties of the relaxation methods by solving the error on a coarser grid and using it as a correction for the fine grid problem.

4.3.1 Correction scheme

The problem on the fine grid h is given by

$$\boldsymbol{\mathcal{A}}_h \boldsymbol{\xi}_h = \boldsymbol{b}_h. \tag{4.34}$$

After performing ν_1 relaxation sweeps on the fine grid problem, the high frequency components are removed from the error and an approximation $\tilde{\boldsymbol{\xi}}_h$ to $\boldsymbol{\xi}_h$ is obtained. The residual on the fine grid after ν_1 sweeps reads

$$\boldsymbol{\mathcal{R}}_h = \boldsymbol{b}_h - \boldsymbol{\mathcal{A}}_h \tilde{\boldsymbol{\xi}}_h. \tag{4.35}$$

Since $\boldsymbol{\mathcal{A}}_h$ is linear, equation (4.35) can be combined with the definition of the exact solution $\boldsymbol{b}_h = \boldsymbol{\mathcal{A}}_h \boldsymbol{\xi}_h$ to obtain

$$\boldsymbol{\mathcal{R}}_h = \boldsymbol{\mathcal{A}}_h \boldsymbol{e}_h, \qquad \boldsymbol{e}_h = \boldsymbol{\xi}_h - \tilde{\boldsymbol{\xi}}_h. \tag{4.36}$$

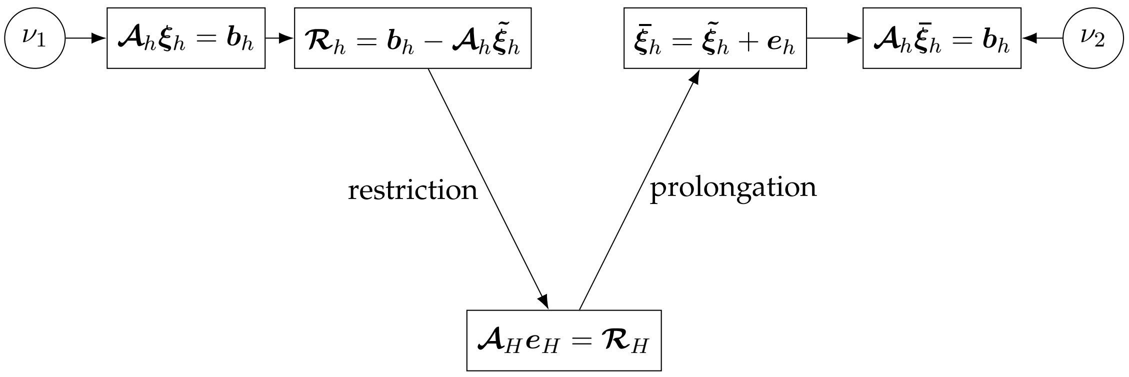
So, an equation for the error in the approximation $\tilde{\boldsymbol{\xi}}_h$ is obtained which is the same as the original fine grid problem but with a different right-hand side vector. Equation (4.36) can be solved by using the residual vector given in equation (4.35). As noted earlier, after a few relaxation sweeps on the fine grid, the error $\boldsymbol{e}_h$ is smooth and can be represented on a coarse grid. Therefore the coarse grid problem is formulated as

$$\boldsymbol{\mathcal{A}}_H \boldsymbol{e}_H = \boldsymbol{\mathcal{R}}_H, \qquad \boldsymbol{\mathcal{R}}_H = \boldsymbol{I}_h^H \boldsymbol{\mathcal{R}}_h, \tag{4.37}$$

where H denotes the coarse grid. The restriction operator $\boldsymbol{I}_h^H$ transforms the residual from the fine grid to the coarse grid. After solving the coarse grid problem of equation (4.37) to obtain a solution to $\boldsymbol{e}_H$, it can be used to correct the approximation $\tilde{\boldsymbol{\xi}}_h$ on the fine grid using

$$\bar{\boldsymbol{\xi}}_h = \tilde{\boldsymbol{\xi}}_h + \boldsymbol{e}_h, \qquad \boldsymbol{e}_h = \boldsymbol{I}_H^h \boldsymbol{e}_H, \tag{4.38}$$

followed by ν_2 relaxation sweeps to remove high-frequency components introduced by the interpolation of the error. The operator $\boldsymbol{I}_H^h$ is the prolongation (or interpolation) operator which transforms the error from the coarse grid to the fine grid. The procedure outlined above is called a correction scheme (CS). The simplest form of a correction scheme is a two-level coarse grid correction cycle, which is depicted in figure 4.4. The coarse grid problem is assumed to be solved exactly. This cycle pattern is carried out until after a number of cycles the approximate solution is within a certain predefined accuracy.



ν_1
$\boldsymbol{\mathcal{A}}_h \boldsymbol{\xi}_h = \boldsymbol{b}_h$
$\boldsymbol{\mathcal{R}}_h = \boldsymbol{b}_h - \boldsymbol{\mathcal{A}}_h \tilde{\boldsymbol{\xi}}_h$
$\bar{\boldsymbol{\xi}}_h = \tilde{\boldsymbol{\xi}}_h + \boldsymbol{e}_h$
$\boldsymbol{\mathcal{A}}_h \bar{\boldsymbol{\xi}}_h = \boldsymbol{b}_h$
ν_2
restriction
prolongation
$\boldsymbol{\mathcal{A}}_H \boldsymbol{e}_H = \boldsymbol{\mathcal{R}}_H$

Figure 4.4: Overview of a two-level coarse grid correction cycle.

The correction scheme is applicable to linear systems of equations. For non-linear systems, an extension to a full approximation scheme (FAS) can be made. However, since the Poisson and Stokes problems are linear systems, this will not be considered in this thesis, as a FAS will reduce to a CS for linear problems.

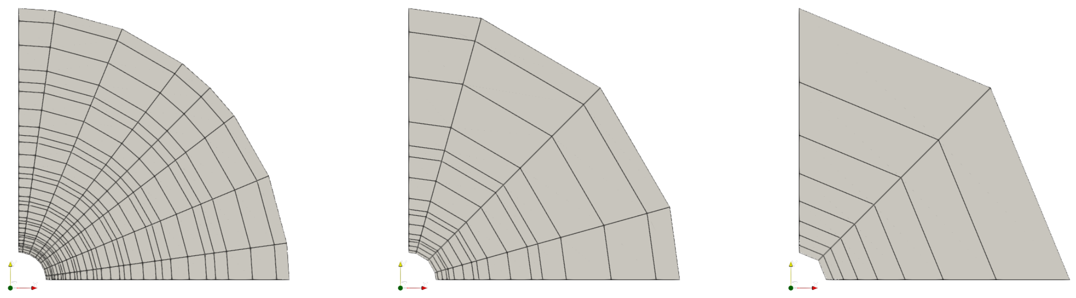
The multigrid algorithm as explained here is based on the reduction of unknowns by geometric coarsening, which is also known as geometric multigrid, or h-multigrid. In a DG method, the number of unknowns can also be reduced by decreasing the polynomial degree of the solution, which is known as polynomial multigrid, or p-multigrid. As mentioned in section 3.3, the penalty parameter σ and stabilization parameter γ that are present in the Stokes and Poisson discretizations negatively impact the conditioning of those systems. Therefore, although it is not a reduction in unknowns but an improvement in conditioning, coarsening based on decreasing penalty parameters can also be beneficial. This will be referred to as penalty-multigrid.

A combination of penalty- p- and h-multigrid is also possible. The multigrid algorithms are typically combined by first performing penalty-multigrid until minimum (or critical) values of σ and γ are obtained. Then, p-multigrid is carried out until the order of the solution is decreased to $p = 1$. Thereafter, the $p = 1$ problem is solved using h-multigrid. More details about the motivation behind a combined multigrid algorithm and its implementation are given in section 4.3.4.

Thus far, the transfer of the residual and correction between the fine and coarse grid and the formulation of the coarse grid problem were treated as a given. These concepts will be elaborated on in more detail in the upcoming sections.

4.3.2 Coarse grid problem

In order to formulate the coarse grid problems for geometric, polynomial and penalty multigrid methods, some choices need to be made. Recall that the representation of the grid is also a local nodal expansion of a given polynomial order p_{grid}. If the coarsened grids (obtained by either geometric or polynomial coarsening) were to be used to construct the coarse grid problems, a lot of information about the geometry would be lost. This is illustrated in figure 4.5, where a part of a circular $8X8$ grid with $p_{grid} = 4$ is illustrated in figure 4.5a, a geometric coarsened $4X4$ $p_{grid} = 4$ grid in figure 4.5b and a polynomial



(a) $8X8, p_{grid} = 4$ (b) $4X4, p_{grid} = 4$ (c) $8X8, p_{grid} = 1$

Figure 4.5: Representation (nodal) of a part of a circular geometry for different number of elements and polynomial orders.

coarsened $8X8$ $p_{grid} = 1$ grid in figure 4.5c. The loss of accuracy in the representation of the geometry is clearly visible on the coarsened grids (figures 4.5b and 4.5c). To prevent this, the geometry of the fine grid can be used to formulate the coarse grid problems.

For penalty parameter coarsening, only the penalty parameters are decreased when defining coarser grids. Hence, the information about the fine grid geometry is automatically preserved. For polynomial coarsening, the coarse grids can be formulated by decreasing the polynomial order of the expansion of the local solution, while keeping the polynomial order of the representation of the geometry the same. Lastly, for geometrical coarsening, the geometry of the fine grid can be retained by grouping fine grid elements into coarse elements, which is also referred to as agglomeration-based geometric multigrid. One advantage of this approach is that it can easily be extended to unstructured grids. An example of geometric multigrid based on agglomeration is illustrated in figure 4.6.

4.3.3 Intergrid transfer

The intergrid transfer operators are obtained by finding the optimal representations of the coarse solution on the fine grid and vice versa. A useful property of the transfer operators is that they are each others transpose multiplied by a constant, [VL00]. Therefore only the restriction or the prolongation operator needs to be found, from which the other operator immediately follows. For a scalar component ξ of $\boldsymbol{\xi}$, the optimal representations can be found by minimizing the L_2 norm of the difference between the fine and coarse representations of the solution,

$$L_2^2 = \int_{\Omega_e} \left(\xi_e^F - \xi_e^C\right)^2 \mathrm{d}\Omega \,, \tag{4.39}$$

with respect to the coarse solution ξ_e^C or fine solution ξ_e^F and where the subscript h has been omitted for clarity. Since the expansions of the coarse and fine solutions can be worked using the symbolic expressions, the L_2-minimization can be performed using the `sympy` library in Python, [Sym23].

It is important to mention that in all derivations of intergrid operators the influence of the shape of the elements has been neglected. The motivation for this is that the coarse levels only serve as a correction for the fine grid problem. So, if the elements are not too distorted, neglecting the mapping of the curved elements should not influence the correction too much while it takes away the need to store these operators for every element. The multigrid results should verify if this assumption is indeed justified.

Penalty parameter coarsening

In case of penalty parameter coarsening, the optimal representations of the solutions can very easily be determined. Since only the penalty parameters are decreased when coarsening whereas the grid definition as well as the local expansion stays the same, the restriction and prolongation operators reduce to the identity matrix.

Polynomial coarsening

Consider the coarsening from a $p = 2$ (fine) to a $p = 1$ 1D solution, given by the expansions

$$\xi_e^F = \hat{\xi}_0^F \psi_0 + \hat{\xi}_1^F \psi_1 + \hat{\xi}_2^F \psi_2, \tag{4.40}$$

$$\xi_e^C = \hat{\xi}_0^C \psi_0 + \hat{\xi}_1^C \psi_1, \tag{4.41}$$

with

$$\psi_0 = \frac{1}{\sqrt{2}}, \tag{4.42}$$

$$\psi_1 = \sqrt{\frac{3}{2}} r, \tag{4.43}$$

$$\psi_2 = \sqrt{\frac{5}{2\sqrt{3}}} r. \tag{4.44}$$

The optimal representation of the fine solution on the coarse grid is found by combining equations (4.39) to (4.44) and minimizing L_2^2 with respect to the expansion coefficients of the coarse solution,

$$\frac{\partial L_2^2}{\partial \hat{\xi}_0^C} = \frac{\partial L_2^2}{\partial \hat{\xi}_1^C} = 0. \tag{4.45}$$

After implementation in Python using `sympy`, it turns out that the best coarse grid approximation is obtained by neglecting the contribution of the highest polynomial degree, i.e. by neglecting the term $\hat{\xi}_2^F \psi_2$ in this example. This underlines the huge advantage that the modal formulation has for p-multigrid. This yields the following restriction operator

$$\boldsymbol{I}_F^C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{4.46}$$

The prolongation operator is found by $\boldsymbol{I}_C^F = (\boldsymbol{I}_F^C)^T$.

The extension to the polynomial coarsening of 2D tensor-product elements requires some extra attention. For example, consider the coarsening from a $p = 2$ (fine) to a $p = 1$

(coarse) 2D solution polynomial for which the expansions of the fine and coarse solutions, respectively, read

$$\xi_e^F = \hat{\xi}_0^F\psi_0^F + \hat{\xi}_1^F\psi_1^F + \hat{\xi}_2^F\psi_2^F + \hat{\xi}_3^F\psi_3^F + \hat{\xi}_4^F\psi_4^F + \hat{\xi}_5^F\psi_5^F + \hat{\xi}_6^F\psi_6^F + \hat{\xi}_7^F\psi_7^F + \hat{\xi}_8^F\psi_8^F, \tag{4.47}$$

$$\xi_e^C = \hat{\xi}_0^C\psi_0^C + \hat{\xi}_1^C\psi_1^C + \hat{\xi}_2^C\psi_2^C + \hat{\xi}_3^C\psi_3^C, \tag{4.48}$$

with corresponding basis functions

$$\psi_0^F = \tfrac{1}{2}, \qquad \psi_0^C = \tfrac{1}{2}, \tag{4.49}$$

$$\psi_1^F = \tfrac{\sqrt{3}}{2}r, \qquad \psi_1^C = \tfrac{\sqrt{3}}{2}r, \tag{4.50}$$

$$\psi_2^F = \tfrac{\sqrt{5}}{4}\left(3r^2-1\right), \qquad \psi_2^C = \tfrac{\sqrt{3}}{2}s, \tag{4.51}$$

$$\psi_3^F = \tfrac{\sqrt{3}}{2}s, \qquad \psi_3^C = \tfrac{3}{2}rs, \tag{4.52}$$

$$\psi_4^F = \tfrac{3}{2}rs, \tag{4.53}$$

$$\psi_5^F = \tfrac{\sqrt{15}}{4}\left(3r^2s-s\right), \tag{4.54}$$

$$\psi_6^F = \tfrac{\sqrt{5}}{4}\left(3s^2-1\right), \tag{4.55}$$

$$\psi_7^F = \tfrac{\sqrt{15}}{4}\left(3rs^2-r\right), \tag{4.56}$$

$$\psi_8^F = \tfrac{5}{8}\left(9rs-3r-3s+1\right), \tag{4.57}$$

Which shows that ψ_0^C corresponds to ψ_0^F, ψ_1^C to ψ_1^F, ψ_2^C to ψ_3^F and ψ_3^C to ψ_4^F. Therefore, neglecting the highest polynomial degree in this case yields the restriction operator

$$\boldsymbol{I}_F^C = \begin{bmatrix} 1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0 \end{bmatrix}. \tag{4.58}$$

Again, the prolongation operator is found by $\boldsymbol{I}_C^F = (\boldsymbol{I}_F^C)^T$.

Geometric coarsening

Consider the grouping of the fine grid elements 0, 1, 2 and 3 together to form the coarse element as depicted in figure 4.6. Geometric multigrid is typically performed after the polynomial degree has been coarsened to $p=1$. For $p=1$, the coarse and fine local solutions are obtained via the expansion,

$$\xi_{e,j}^F = \hat{\xi}_{0,j}^F\psi_{0,j} + \hat{\xi}_{1,j}^F\psi_{1,j} + \hat{\xi}_{2,j}^F\psi_{2,j} + \hat{\xi}_{3,j}^F\psi_{3,j}, \qquad j = 0,1,2,3, \tag{4.59}$$

$$\xi_e^C = \hat{\xi}_0^C\psi_0 + \hat{\xi}_1^C\psi_1 + \hat{\xi}_2^C\psi_2 + \hat{\xi}_3^C\psi_3, \tag{4.60}$$

with the basis functions

$$\psi_0 = \tfrac{1}{2}, \tag{4.61}$$

$$\psi_1 = \tfrac{\sqrt{3}}{2}r, \tag{4.62}$$

$$\psi_2 = \tfrac{\sqrt{3}}{2}s, \tag{4.63}$$

$$\psi_3 = \tfrac{3}{2}rs. \tag{4.64}$$

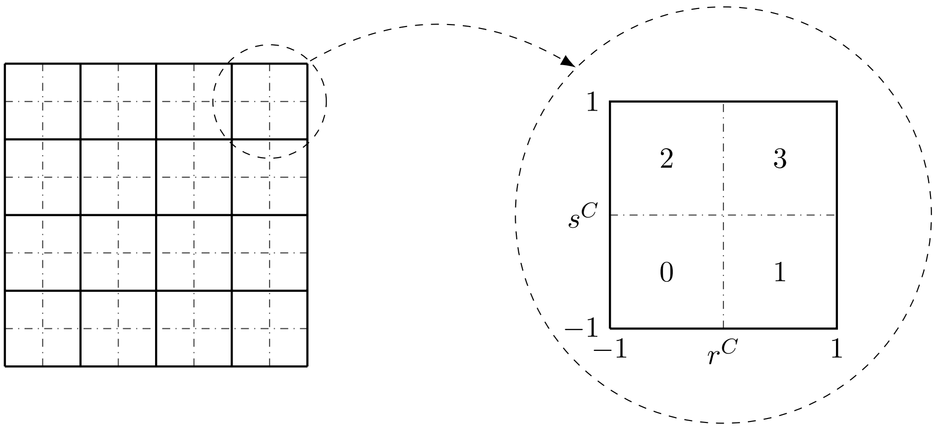


Figure 4.6: Geometric coarsening based on agglomeration of fine grid elements (dashed lines) into coarse grid elements (solid lines).

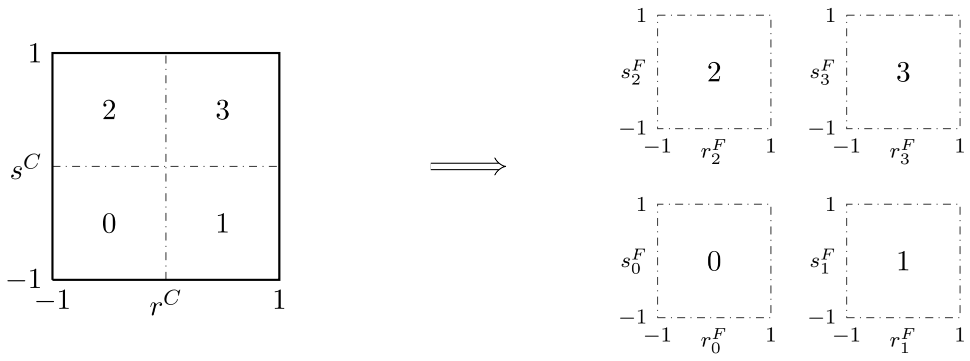


Figure 4.7: An illustration of the mapping of the coarse grid coordinates r^C and s^C to the fine grid coordinates r_j^F and s_j^F.

The basis functions $\psi_{i,j}$ are defined in the fine grid coordinates by mapping the coarse grid r^C and s^C to the fine grid r_j^F and s_j^F, see figure 4.7, using

$$r_0^F = 2r^C + 1, \qquad s_0^F = 2s^C + 1, \tag{4.65}$$

$$r_1^F = 2r^C - 1, \qquad s_1^F = 2s^C + 1, \tag{4.66}$$

$$r_2^F = 2r^C + 1, \qquad s_2^F = 2s^C - 1, \tag{4.67}$$

$$r_3^F = 2r^C - 1, \qquad s_3^F = 2s^C - 1. \tag{4.68}$$

Combining equations (4.39), (4.59), (4.60) and (4.65) to (4.68) and minimizing L_2^2 with respect to the expansion coefficients of the coarse solution using `sympy`, i.e.

$$\frac{\partial L_2^2}{\partial \hat{\xi}_0^C} = \frac{\partial L_2^2}{\partial \hat{\xi}_1^C} = \frac{\partial L_2^2}{\partial \hat{\xi}_2^C} = \frac{\partial L_2^2}{\partial \hat{\xi}_3^C} = 0, \tag{4.69}$$

leads to the restriction operator

$$\boldsymbol{I}_F^C = \begin{bmatrix} \frac{1}{4} & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 \\ -\frac{\sqrt{3}}{8} & \frac{1}{8} & 0 & 0 & \frac{\sqrt{3}}{8} & \frac{1}{8} & 0 & 0 & -\frac{\sqrt{3}}{8} & \frac{1}{8} & 0 & 0 & \frac{\sqrt{3}}{8} & \frac{1}{8} & 0 & 0 \\ -\frac{\sqrt{3}}{8} & 0 & \frac{1}{8} & 0 & -\frac{\sqrt{3}}{8} & 0 & \frac{1}{8} & 0 & \frac{\sqrt{3}}{8} & 0 & \frac{1}{8} & 0 & \frac{\sqrt{3}}{8} & 0 & \frac{1}{8} & 0 \\ \frac{3}{16} & -\frac{\sqrt{3}}{16} & -\frac{\sqrt{3}}{16} & \frac{1}{16} & -\frac{3}{16} & -\frac{\sqrt{3}}{16} & \frac{\sqrt{3}}{16} & \frac{1}{16} & -\frac{3}{16} & \frac{\sqrt{3}}{16} & -\frac{\sqrt{3}}{16} & \frac{1}{16} & \frac{3}{16} & \frac{\sqrt{3}}{16} & \frac{\sqrt{3}}{16} & \frac{1}{16} \end{bmatrix}, \tag{4.70}$$

which can be applied to the (fine) residual vector of each element ordered as

$$\boldsymbol{\mathcal{R}}_e^F = \left[\mathcal{R}_{0,0}^F, \mathcal{R}_{1,0}^F, \mathcal{R}_{2,0}^F, \mathcal{R}_{3,0}^F, \mathcal{R}_{0,1}^F, \mathcal{R}_{1,1}^F, \mathcal{R}_{2,1}^F, \mathcal{R}_{3,1}^F, \mathcal{R}_{0,2}^F, \mathcal{R}_{1,2}^F, \mathcal{R}_{2,2}^F, \mathcal{R}_{3,2}^F, \mathcal{R}_{0,3}^F, \mathcal{R}_{1,3}^F, \mathcal{R}_{2,3}^F, \mathcal{R}_{3,3}^F\right]^T, \tag{4.71}$$

where the first index in the subscript denotes the corresponding DOF and the second the corresponding fine grid element as illustrated in figure 4.6.

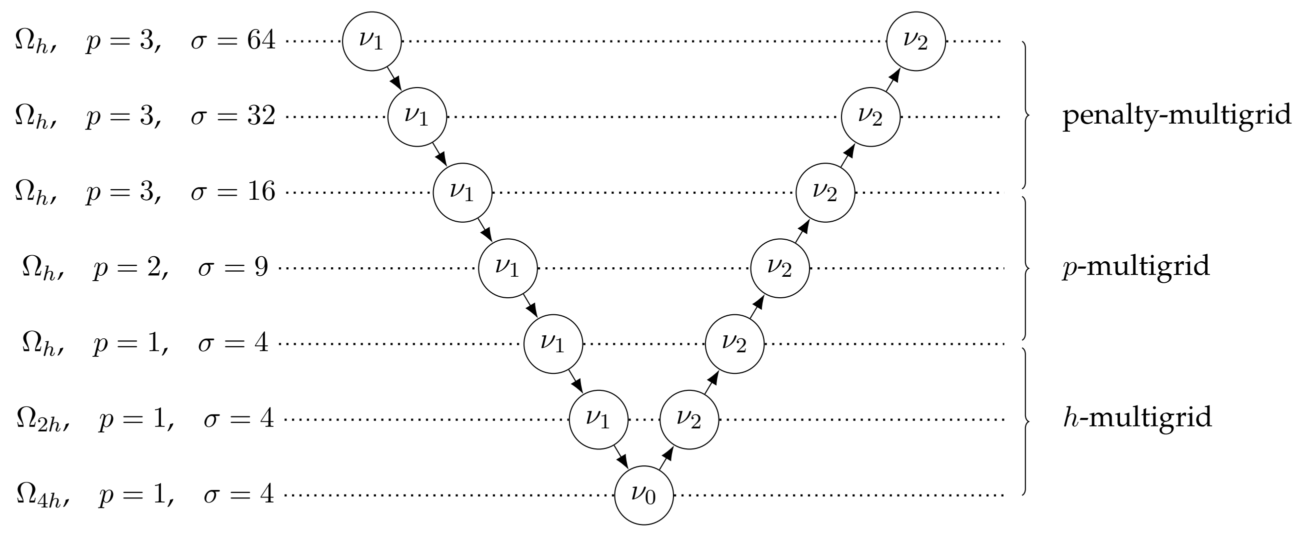
As a result of the agglomeration of four fine elements into one coarse element, the restriction operator in equation (4.70) can only be applied to the residual if the residual is multiplied by the inverse of the mass matrix of the fine elements, defined in terms of the coarse grid coordinates r^C and s^C. This can be seen by writing equation (4.34) as a transient problem with the mass matrix of the fine elements $\boldsymbol{M}_h$,

$$\boldsymbol{M}_h \frac{\mathrm{d}\boldsymbol{\xi}_h}{\mathrm{d}t} + \boldsymbol{\mathcal{A}}_h \boldsymbol{\xi}_h = \boldsymbol{b}_h \qquad \Longrightarrow \qquad \frac{\mathrm{d}\boldsymbol{\xi}}{\mathrm{d}t} = (\boldsymbol{M}_h)^{-1} \boldsymbol{\mathcal{R}}_h, \tag{4.72}$$

where $\boldsymbol{\mathcal{R}}_h = \boldsymbol{b}_h - \boldsymbol{\mathcal{A}}_h \boldsymbol{\xi}_h$. Using equation (2.21), the mass matrix of the fine elements can be calculated to be $\frac{1}{4}\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix (note that the effect of mapping has been neglected). This means that if the residual is not multiplied by the inverse of the mass matrix, equation (4.70) must be multiplied by a factor 4. Moreover, the prolongation operator of the error can hence be found by $\boldsymbol{I}_C^F = 4(\boldsymbol{I}_F^C)^T$.

4.3.4 Final multigrid algorithm

The two-level coarse grid correction cycle explained in section 4.3.1 may still be quite inefficient. This is because the coarse grid problem is solved exactly which can be very costly when the coarse grid contains many unknowns (even though it is reduced with respect to the fine grid). It is important to note that after a few relaxation sweeps on the coarse grid, the error components with wavelengths comparable to that grid are smooth which means that it can be solved using the coarse grid correction cycle itself. A multi-level coarse grid correction cycle can be designed by applying the coarse grid correction cycle recursively until the coarse grid problem can be solved inexpensively using relaxation. Taking into account penalty-p- and h-multigrid, the scheme depicted in figure 4.8 is obtained. Relaxation on each level is indicated by a ν_i, restriction by a downwards arrow and prolongation by an upwards arrow. As an example, the fine grid has a mesh size of Ω_h, solution polynomial degree of $p = 3$ and penalty parameter of $\sigma = 64$. Note that the penalty parameter σ is coarsened here, the same procedure applies for the stabilization parameter γ or a combination of both. This example can be extended to arbitrary problems by taking a variety of coarse levels into account. Because of its recursive pattern, the scheme presented here is known as a V-cycle. For specific problems, other patterns such as a W-cycle can be used.



$\Omega_h, \quad p=3, \quad \sigma=64$
$\Omega_h, \quad p=3, \quad \sigma=32$
$\Omega_h, \quad p=3, \quad \sigma=16$
$\Omega_h, \quad p=2, \quad \sigma=9$
$\Omega_h, \quad p=1, \quad \sigma=4$
$\Omega_{2h}, \quad p=1, \quad \sigma=4$
$\Omega_{4h}, \quad p=1, \quad \sigma=4$
ν_1
ν_2
ν_0
penalty-multigrid
p-multigrid
h-multigrid

Figure 4.8: An example of a coarse grid correction cycle for penalty- p- and h-multigrid.

However, since the the V-cycle is known to work well with the Poisson problem, [VL00], the focus of this work is on the V-cycle.

The question remains how accurate the problem must be solved on the coarse grid to provide a good enough prediction of the correction of the error. It is unnecessary to solve the coarse grid problem exactly. Since the solution of the coarse grid problem is used as a correction, it is sufficient to solve it with at least the same accuracy as the problems on the other levels. On a sufficiently coarse grid, $\nu_0 = \mathcal{O}(10)$ is generally good enough, [VL00].

Chapter 5

Methodology

This chapter describes how the concepts that were discussed in chapters 2 to 4 are combined to research the building blocks of a multigrid algorithm that can be applied to a DG formulation of the Stokes problem. The chapter starts in section 5.1 with a description of the mesh types that will be used in this study. Thereafter, the specifics of the DG discretization are discussed in section 5.2, followed by a description of a method to verify the implementation of the DG discretizations of the Poisson and Stokes problems in section 5.3. The design of the multigrid algorithm for the Poisson problem and a smoother for the Stokes problem are addressed in sections 5.4 and 5.5, respectively.

5.1 Mesh types

Since the focus of this thesis is on solving the discretized systems of equations, it should not matter whether the results are obtained on structured or unstructured grids, as long as the algorithms that are used are applicable to both grids. This is the case for penalty-multigrid as well as p- and agglomeration based h-multigrid. Although the discretization scheme becomes more complex, the results obtained on structured grids should be extendable to unstructured grids. Therefore, for simplicity, only structured grids are considered.

Two different types of structured meshes are considered, namely Cartesian and curvilinear which is illustrated in figure 5.1. The difference between the two is that the mapping from the Cartesian elements to the standard element is constant while this is not the case for the curvilinear elements. By defining these two grid types, the influence of the mapping on the results can be identified. A square grid with dimensions $[-1, 1]^2$ is used for the Cartesian grid and a so-called circular O-grid with an inner radius of 0.1 and an outer radius of 1 is used for the curvilinear grid. The grids are generated based on the number of elements and the degree of the Lagrange polynomials. Note that by definition, the representation of the geometry is in nodal formulation, see equations (2.74) and (2.75). For the reasons explained in section 2.2.5, LGL nodes are used for the interpolation points.

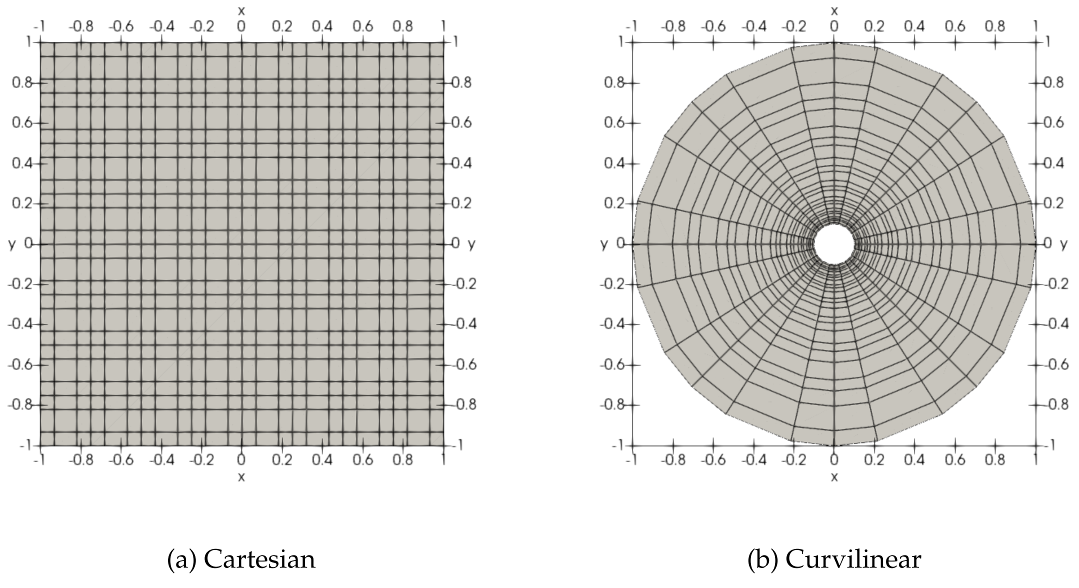

(a) Cartesian (b) Curvilinear

Figure 5.1: Illustrations of the structured grid geometries, $8X8$, $p = 3$.

5.2 DG formulations

While the geometry is described using a nodal formulation, the DG solution is approximated by a modal expansion for the use of p-adaption. To minimize the influence of the geometry representation, the polynomial order of the solution will be kept equal to that of the geometry. For example, a $p = 5$ grid will be used for a $p = 5$ DG solution approximation. The Poisson and Stokes problems will be discretized using the discretization given in section 3.3, where for the discretization of the Poisson problem only the scalar equations for diffusion and source terms are taken into account. The integrals will be evaluated using a Gauss-Legendre quadrature that is able to integrate polynomials up to order $3p$ exactly.

This should lead to exact integration of all the integrals on the Cartesian grids. As a result of the non-constant Jacobian of the mapping on curvilinear grids, it might not be possible to evaluate these integrals exactly. However, an integration rule of order $3p$ accuracy should be able to evaluate the integrals with sufficient accuracy. If it turns out that this is insufficient, the accuracy of the quadrature rule can further be increased.

5.2.1 Parameter choice

In the discretization scheme given in section 3.3, there are some parameter choices to be made. First of all, the stability of the discretization of the diffusion term in equation (3.29) depends on the penalty parameter σ. As discussed in section 3.3.3, the critical value for stability derived by [Hil13] for quadrilateral elements is $(p+1)^2$. While this critical value is derived for elements with a constant mapping, the penalty parameter will initially be set to this value, regardless of the shape of the physical elements. If it turns out that this is insufficient for the construction of a stable discretization of the diffusion term, it can always be increased. However, before increasing σ and weakening the conditioning of the system, there is something else that could be worth investigating.

In the construction of the DG method, the basis functions were defined to be orthonormal on the standard element, so in the parametric space spanned by r and s. However, this does not guarantee that the basis functions are orthonormal - or at least orthogonal - on physically curved elements because the Jacobian of the mapping is not constant. In fact, according to [Bot12], the mapping from curved elements to the standard element can spoil the convergence properties of the discrete approximation space of the standard element (which is spanned by the basis functions on the standard element). To investigate if the stabilization of the discretization of the diffusion term on curved elements can be fixed by orthonormalizing the basis functions in the physical space, a Gram-Schmidt orthogonalization procedure can be done on all curved elements. If this is the case, there is no need to increase σ and consequently the worsen the conditioning of the system. More information regarding the Gram-Schmidt procedure can be found in appendix B.1.

Another parameter that needs to be chosen in the discretization of the diffusion term is the length scale h_f. Since it is not always easy to calculate this value, especially for curved elements, it will be approximated by the square root of the area of the element, $h_f = \sqrt{A}$. The area can easily be computed using the Gauss-Legendre quadrature.

Lastly, the stabilization parameter γ that influences the stabilization of the divergence-free constraint must be chosen. Ideally it is chosen as large as possible to create an exactly divergence-free solution. However, since this weakens the conditioning of the system and because the an optimal value depends on a lot of factors. The value of $\gamma = 1$ proposed by [Jen+14] is used.

5.2.2 Error measurements

A measure for the error in the solution u is obtained with the L_2-norm

$$L_2(\mathrm{u}) = \sqrt{\left(\frac{1}{N_{DOF}} \sum_{n=0}^{N_{DOF}-1} |\mathrm{u}_n - \mathrm{u}_n^{\mathrm{ex}}|\right)}, \tag{5.1}$$

where u_n and $\mathrm{u}_n^{\mathrm{ex}}$ are the numerical and the exact solution in the n-th DOF, respectively. Similarly, the L_2-norms of the error in v and p are obtained using $L_2(\mathrm{v})$ and $L_2(\mathrm{p})$, respectively.

5.3 Verification

Before diving deeper into the development of a multigrid algorithm that can be applied to the DG formulation of the Stokes problem, it is important that the implementation of the discretization schemes, but also the schemes themselves work as expected. This can be done using the method of manufactured solutions (MMS), which is a very straightforward and robust way of verifying codes that approximate solutions of partial differential equations, [BS19]. The method works by prescribing a solution and setting a source term accordingly, followed by a grid refinement study. Since the convergence rate of the discretization schemes are known, order h^{p+1} for the Stokes velocity and Poisson solution and h^p for the Stokes pressure solution, the code implementation as well as the discretization itself can be tested. As an example, suppose that the implementation of the one-dimensional Laplace equation is to be tested with the exact solution $\mathrm{u} = \sin(\pi x)$, the source term

$$\mathrm{f}_{MMS} = \frac{\mathrm{d}^2}{\mathrm{d}x^2}\left(\sin(\pi x)\right) = -\pi^2 \sin(\pi x) \tag{5.2}$$

must be included in the code. The manufactured solutions (the solutions that are prescribed) do not necessarily have to have any physical meaning, the MMS is a purely mathematical procedure. What is important though, is that the solution must include all ordered derivatives in the error expansion, e.g. cross-derivative terms. To avoid bugs in the Python code, `sympy` is used to generate symbolic expressions of the source terms for different manufactured solutions, which then can be included in the code by converting the `sympy` objects to `numpy` objects.

In case of the Stokes problem, since the stabilization of the divergence-free constraint is added to the discretization (see section 3.3.4), it is convenient to prescribe a solution that is also divergence-free ($\nabla \cdot \mathbf{u} = 0$). While it is possible to perform the MMS procedure for arbitrary (non divergence-free) exact solutions, this leads to more complex formulations of the divergence-free constraint stabilization. Taking this into account, the manufactured solutions proposed by [BB15] will be used to verify the Stokes implementation,

$$\mathrm{u} = -2\sin(\pi x)^2 \sin(\pi y)\cos(\pi y), \tag{5.3}$$

$$\mathrm{v} = 2\sin(\pi x)\cos(\pi x)\sin(\pi y)^2, \tag{5.4}$$

$$\mathrm{p} = \sin(\pi x)\sin(\pi y) - \bar{\mathrm{p}}, \tag{5.5}$$

where $\bar{\mathrm{p}}$ is the mean pressure given in equation (3.12). The manufactured solutions are infinitely differentiable and divergence-free. For the Poisson problem, only equation (5.3) will be used.

In order to verify that the code implementation and discretizations are correct, a direct solver will be used. As mentioned in chapter 4, direct solvers are only applicable to smaller problems, because they require an impractical amount of computing resources to solve larger systems. Therefore, the grid convergence study will be carried out using Cartesian and curvilinear grids with $4X4$ to $64X64$ (Poisson) and $32X32$ (Stokes) elements and solution polynomial degrees ranging from $p = 1$ (Poisson) and $p = 2$ (Stokes, pressure is $p = 1$) to $p = 5$.

Care needs to be taken when solving the Stokes system with a direct solver because physically, the pressure is determined up to a constant leading to a singular system. The problem can be made regular by fixing the first pressure DOF of one element to 0, which can be done by putting a 1 in the zero-block of the coefficient matrix at the location that corresponds to that pressure DOF. This is done for the first DOF of the first element. This means that the numerical pressure solution must be shifted afterwards by subtracting 2 times (because the expansion coefficient of the first DOF is $\frac{1}{2}$) the numerical mean pressure from all elements. Moreover, for solutions that cannot be integrated exactly, the right-hand side vector of the continuity equation must be corrected to ensure that the system is consistent, i.e. it does not matter for the final solution in which element the first pressure

DOF has been fixed to 0. This can be done by integrating the continuity equation as follows

$$\iint_{\Omega_h^e} \nabla \cdot \mathbf{u} \, \mathrm{d}\Omega = \iint_{\Omega_h^e} \mathrm{MMS}_{\text{continuity}} \, \mathrm{d}\Omega \tag{5.6}$$

$$\implies \oint_{\partial\Omega_h^e} \mathbf{u} \cdot \mathbf{n} \, \mathrm{d}\Omega = \iint_{\Omega_h^e} \mathrm{MMS}_{\text{continuity}} \, \mathrm{d}\Omega \tag{5.7}$$

where Gauss' divergence theorem has been used. When these terms are not integrated exactly, they are not equal such that

$$\mathcal{E}\Omega = \iint_{\Omega_h^e} \mathrm{MMS}_{\text{continuity}} \, \mathrm{d}\Omega - \oint_{\partial\Omega_h^e} \mathbf{u} \cdot \mathbf{n} \, \mathrm{d}\Omega \,, \tag{5.8}$$

where $\mathcal{E}$ is the correction and Ω the area of the element. Of course, $\mathrm{MMS}_{\text{continuity}}$ was set to 0 which means that

$$\mathcal{E} = -\frac{\oint_{\partial\Omega_h^e} \mathbf{u} \cdot \mathbf{n} \, \mathrm{d}\Omega}{\Omega}, \tag{5.9}$$

which must be added as the continuity source. Although the stabilization of the divergence-free constraint requires that the continuity source is 0, the correction $\mathcal{E}$ is assumed to be small. The effect, if any, will become even smaller with grid refinement.

5.4 Multigrid algorithm for the Stokes problem

One of the building blocks of this thesis is to develop a multigrid algorithm to solve the systems of equations arising from the DG discretization of the Poisson problem. Since multigrid is not a plug-and-play solution method, some choices need to be made.

Firstly, since the penalty parameter σ has been set to the minimum (critical) value, penalty-multigrid has been left out of the scope of this research. However, if the penalty parameter is taken larger than the critical value, the solution algorithm must certainly be expanded with penalty multigrid to reduce the dependency on the conditioning of the systems.

Moreover, developing a multigrid algorithm based on polynomial coarsening only will not be sufficient since the coarse grid problem (the original grid with $p = 1$) will still be of significant size, such that the low frequency components of the error cannot be solved efficiently. Therefore, after to $p = 1$, geometric multigrid is added to solve the coarse grid problem arising from the p-coarsening. The geometric multigrid is carried out by the agglomeration of elements described in section 4.3.3 until a grid of $4X4$ elements is obtained. A V-cycle pattern will be used.

Because the discrete system resulting from a DG discretization is in block-form, also a block-type smoother will be used. Because the convergence properties of the Gauss-Seidel method are generally superior to the Jacobi method, as discussed in section 4.2, the block Gauss-Seidel method will be used as the smoother for both polynomial and geometric multigrid. The number of pre-relaxation and post-relaxation sweeps is set to respectively $\nu_1 = 2$ and $\nu_2 = 1$. The coarse grid problem is solved by applying $\nu_0 = 10$ sweeps of the same smoother. Both the coefficient matrix as well as the right-hand side vector will be multiplied with the inverse of the mass matrix, such that the intergrid transfer operators derived in section 4.3.3 do not need to be manipulated.

When assessing a multigrid algorithm, it is important to be able to monitor the reduction in the residual. Since it is difficult to judge when the problem is solved up to the level of the discretization error of the DG scheme, the algorithm will be monitored based on the number of V-cycles that are needed to reduce the L_2-norm of the residual by at least 6 orders of magnitude. The L_2-norm of the residual is defined using equation (5.1) as

$$L_2(\mathcal{R}) = \sqrt{\left(\frac{1}{N_{DOF}} \sum_{n=0}^{N_{DOF}-1} |\mathcal{R}_n|\right)}, \tag{5.10}$$

where the n-th DOF of the residual is defined as $\mathcal{R}_n = (\boldsymbol{b} - \boldsymbol{\mathcal{A}\xi})_n$. All residuals have been normalized by the value of the initial residual.

5.4.1 Smoother performance

As a way of monitoring the smoothing properties of a given relaxation method on a certain discrete system, an amplification factor can be computed from the response of the smoother on a Fourier component. Recall that at any point in the approximation, the error can be decomposed in its Fourier components,

$$\tilde{e}^h = \sum_{\theta_x, \theta_y} \tilde{A}(\theta_x, \theta_y) e^{i(\theta_x k + \theta_y l)}, \qquad -\pi \leq \theta_x, \theta_y \leq \pi, \tag{5.11}$$

where A is the amplitude of the component with angular frequencies θ_x and θ_y in x- and y-direction, respectively. Moreover, i denotes the imaginary unit with the well-known property $i^2 = -1$ and the indices k and l are related to the LGL nodes of the grid in x- and y-direction, respectively. The numbering of k and l takes the discontinuous nature of the elements into account, see figure 5.2, as the nodes at the interfaces are not shared in a DG method.

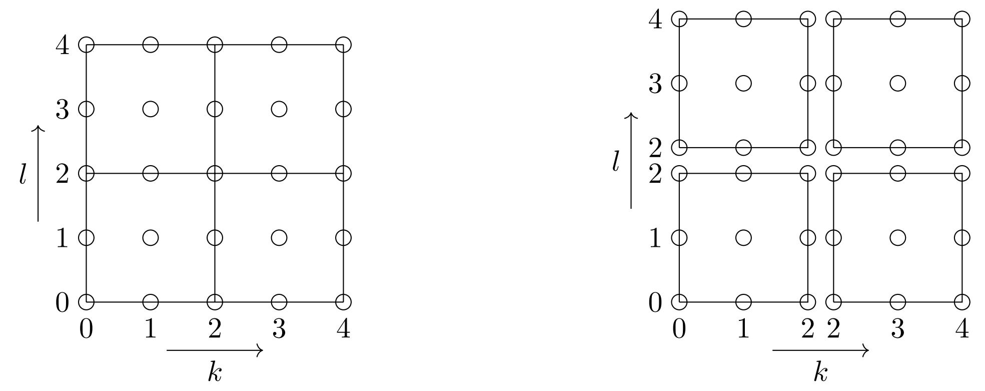


Figure 5.2: Numbering of the k and l indices related to the LGL nodes of the grid, illustrated on a $2X2$ $p = 2$ grid.

When the right-hand side of the linear system is set to zero, the current approximation

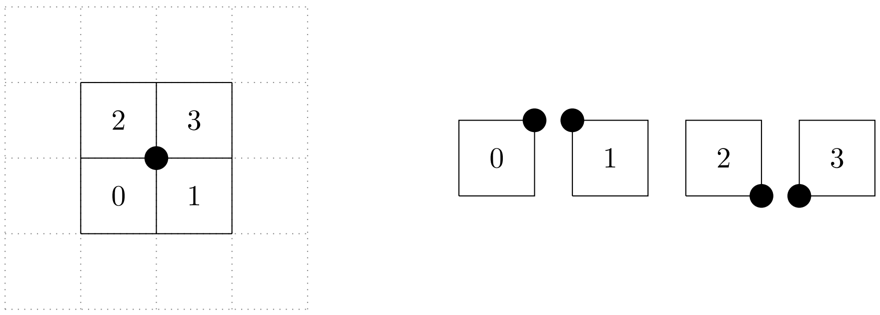


Figure 5.3: Illustration of the four elements that make up the midpoint in a DG method.

equals the error $\tilde{e}^h$. The new approximation then reads

$$\bar{e}^h = \sum_{\theta_x,\theta_y} \bar{A}(\theta_x,\theta_y)e^{i(\theta_x k+\theta_y l)}, \qquad -\pi \leq \theta_x,\theta_y \leq \pi, \tag{5.12}$$

such that the amplitude amplification factor μ can be written as

$$\mu(\theta_x,\theta_y) = \left|\frac{\bar{A}(\theta_x,\theta_y)}{\tilde{A}(\theta_x,\theta_y)}\right|, \qquad -\pi \leq \theta_x,\theta_y \leq \pi. \tag{5.13}$$

Thus, when setting the initial solution to a Fourier component with a grid related frequency, the performance of the relaxation method can be monitored when performing a single relaxation sweep. It is convenient to set the initial solution to a Fourier component with an amplitude of 1, such that $\mu = \left|\bar{A}(\theta_x,\theta_y)\right|$. By ranging the frequencies between $-\pi$ and π, the amplification factor as a function of the frequency components of the error can be visualized. To reduce the interference of boundaries, it is useful to do this for a point that is in the middle of the domain of a sufficiently fine grid.

Since the initial solution is set to a frequency that is related to the (nodal) grid, this must first be transformed to a modal formulation before it can be used to monitor the smoothing performance of a DG discretization. Thereafter, it must be transformed back to the nodal formulation before the amplification factor of equation (5.13) can be calculated. Both transformations can be done using a Vandermonde interpolation matrix. Moreover, since the grid points are not shared in a DG method, the response needs to be monitored in the four elements that make up the midpoint as illustrated in figure 5.3.

5.5 Smoother for the Stokes problem

The Distributive Gauss-Seidel smoother based on both the classical and least-squares splittings described in section 4.2.3 will be applied to the Stokes problem. The inner sweeps are done by a single symmetric block Gauss-Seidel sweep on the systems in equations (4.21) to (4.23) (classical splitting) and equations (4.30), (4.31) and (4.33) (least-squares splitting). To determine if the relaxation schemes converge, the L_2-norm of the residual, equation (5.10), will be monitored.

Chapter 6

Results and Discussion

The results are presented and discussed in this chapter. First, the verification of the discretizations of both the Poisson as well as the Stokes problem is addressed in section 6.1. Thereafter, the results of the multigrid algorithm for the Poisson problem as well as the results of the distributive Gauss-Seidel relaxation for the Stokes problem are discussed in sections 6.2 and 6.3, respectively.

6.1 Verification

6.1.1 Poisson

First of all, it is important that the discrete system of equations resembles the continuous problem and is stable. As explained in sections 3.3.5 and 5.2.1, this means that the coefficient matrix of the discretized Poisson equation must be symmetric positive definite (SPD) and the penalty parameter σ must be sufficiently high. Moreover, it was shown in section 4.2 that the convergence of the relaxation methods depend on the SPD property as well. This property can be verified by looking at the eigenvalues of the coefficient matrix $\boldsymbol{\mathcal{A}}$ and the spectral radius (i.e. the largest absolute eigenvalue) of the Gauss-Seidel iteration matrix.

In table 6.1, the minimum and maximum eigenvalues of the coefficient matrix and the spectral radius of the Gauss-Seidel iteration matrix have been computed for different grid sizes and solution polynomial orders for both a Cartesian as well as a curvilinear grid. These results show that on a Cartesian grid, the eigenvalues of $\boldsymbol{\mathcal{A}}$ are positive and the spectral radius of $\boldsymbol{B}_{GS}$ is smaller than 1, meaning that the coefficient matrix is positive definite and Gauss-Seidel relaxation converges. However, there are curvilinear grids (see figure 5.1b) for which the eigenvalues of $\boldsymbol{\mathcal{A}}$ have a different sign and consequently $\rho\left(\boldsymbol{B}_{GS}\right) > 1$ (indicated with red), meaning that the discretization is unstable and Gauss-Seidel relaxation will not converge. Additional checks show that all discrete systems are symmetric ($\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{A}}^T$) and not diagonally dominant ($|\mathcal{A}_{ii}| \ngeq \sum_{j \neq i} |\mathcal{A}_{ij}|$). This means the coefficient matrices on the Cartesian grids are not only positive definite, but in fact symmetric positive definite.

As was discussed in section 5.2.1, the critical value of the penalty parameter derived by [Hil13], $\sigma = (p+1)^2$, does not apply for curved elements. It is however very similar when the elements are not too much distorted, which can be observed on the $32X32$ $p = 3$ grid. Before increasing σ and consequently weakening the conditioning of the coefficient matrix,

Table 6.1: Eigenvalues of the Poisson coefficient matrix and the spectral radius of the Gauss-Seidel iteration matrix on the Cartesian and curvilinear grids (as illustrated in figure 5.1).

Grid size	Order	Cartesian grid			Curvilinear grid		
		$\min(\lambda_{\mathcal{A}})$	$\max(\lambda_{\mathcal{A}})$	$\rho(\boldsymbol{B}_{GS})$	$\min(\lambda_{\mathcal{A}})$	$\max(\lambda_{\mathcal{A}})$	$\rho(\boldsymbol{B}_{GS})$
$4X4$	1	$3.23 \cdot 10^{-1}$	$1.69 \cdot 10^{1}$	$8.590 \cdot 10^{-1}$	$5.53 \cdot 10^{-1}$	$1.95 \cdot 10^{1}$	$7.937 \cdot 10^{-1}$
$8X8$	1	$7.81 \cdot 10^{-2}$	$1.77 \cdot 10^{1}$	$9.623 \cdot 10^{-1}$	$1.14 \cdot 10^{-1}$	$1.80 \cdot 10^{1}$	$9.515 \cdot 10^{-1}$
$16X16$	1	$1.93 \cdot 10^{-2}$	$1.79 \cdot 10^{1}$	$9.904 \cdot 10^{-1}$	$2.70 \cdot 10^{-2}$	$1.79 \cdot 10^{1}$	$9.882 \cdot 10^{-1}$
$32X32$	1	$4.82 \cdot 10^{-3}$	$1.80 \cdot 10^{1}$	$9.976 \cdot 10^{-1}$	$6.63 \cdot 10^{-3}$	$1.78 \cdot 10^{1}$	$9.971 \cdot 10^{-1}$
$4X4$	3	$3.08 \cdot 10^{-1}$	$2.03 \cdot 10^{2}$	$9.638 \cdot 10^{-1}$	$-1.27 \cdot 10^{1}$	$2.01 \cdot 10^{2}$	$2.088 \cdot 10^{0}$
$8X8$	3	$7.71 \cdot 10^{-2}$	$2.08 \cdot 10^{2}$	$9.905 \cdot 10^{-1}$	$-3.86 \cdot 10^{0}$	$2.00 \cdot 10^{2}$	$2.251 \cdot 10^{0}$
$16X16$	3	$1.93 \cdot 10^{-2}$	$2.10 \cdot 10^{2}$	$9.976 \cdot 10^{-1}$	$-7.22 \cdot 10^{-1}$	$1.98 \cdot 10^{2}$	$1.243 \cdot 10^{0}$
$32X32$	3	$4.82 \cdot 10^{-3}$	$2.10 \cdot 10^{2}$	$9.994 \cdot 10^{-1}$	$6.60 \cdot 10^{-3}$	$1.98 \cdot 10^{2}$	$9.993 \cdot 10^{-1}$
$4X4$	5	$3.08 \cdot 10^{-1}$	$9.60 \cdot 10^{2}$	$9.838 \cdot 10^{-1}$	$-1.51 \cdot 10^{2}$	$9.17 \cdot 10^{2}$	$2.109 \cdot 10^{3}$
$8X8$	5	$7.71 \cdot 10^{-2}$	$9.77 \cdot 10^{2}$	$9.958 \cdot 10^{-1}$	$-1.02 \cdot 10^{2}$	$9.11 \cdot 10^{2}$	$8.021 \cdot 10^{0}$
$16X16$	5	$1.93 \cdot 10^{-2}$	$9.81 \cdot 10^{2}$	$9.989 \cdot 10^{-1}$	$-8.18 \cdot 10^{1}$	$9.12 \cdot 10^{2}$	$3.133 \cdot 10^{0}$
$32X32$	5	$4.82 \cdot 10^{-3}$	$9.82 \cdot 10^{2}$	$9.997 \cdot 10^{-1}$	$-7.22 \cdot 10^{1}$	$9.15 \cdot 10^{2}$	$2.296 \cdot 10^{0}$

Table 6.2: Eigenvalues of the Poisson coefficient matrix and the spectral radius of the Gauss-Seidel iteration matrix on curvilinear grids (as illustrated in figure 5.1b).

Grid size	Order	Orthonormalized basis			$\sigma = 2 \cdot (p+1)^2$		
		$\min(\lambda_{\mathcal{A}})$	$\max(\lambda_{\mathcal{A}})$	$\rho(\boldsymbol{B}_{GS})$	$\min(\lambda_{\mathcal{A}})$	$\max(\lambda_{\mathcal{A}})$	$\rho(\boldsymbol{B}_{GS})$
$4X4$	5	$-6.76 \cdot 10^{4}$	$2.45 \cdot 10^{5}$	$2.109 \cdot 10^{3}$	$4.46 \cdot 10^{-1}$	$2.27 \cdot 10^{3}$	$9.888 \cdot 10^{-1}$
$8X8$	5	$-1.80 \cdot 10^{5}$	$1.15 \cdot 10^{6}$	$8.021 \cdot 10^{0}$	$1.07 \cdot 10^{-1}$	$2.52 \cdot 10^{3}$	$9.973 \cdot 10^{-1}$
$16X16$	5	$-5.77 \cdot 10^{5}$	$5.33 \cdot 10^{6}$	$3.133 \cdot 10^{0}$	$2.65 \cdot 10^{-2}$	$2.57 \cdot 10^{3}$	$9.993 \cdot 10^{-1}$
$32X32$	5	$-2.04 \cdot 10^{6}$	$2.33 \cdot 10^{7}$	$2.296 \cdot 10^{0}$	$6.60 \cdot 10^{-3}$	$2.59 \cdot 10^{3}$	$9.998 \cdot 10^{-1}$

it is attempted to stabilize the discretization by orthonormalizing the basis functions in the physical space using a Gram-Schmidt procedure, see appendix B.1 for more details.

The resulting eigenvalues for the most problematic curvilinear grids (5-th order solution polynomial) are shown in table 6.2. Clearly, orthonormalizing the basis functions in the physical space does not fix the issue. The eigenvalues of $\mathcal{A}$ still have a different sign and $\rho(\boldsymbol{B}_{GS})$ is still larger than 1. In fact, the spectral radii of the Gauss-Seidel iteration matrix remain unchanged compared to the non-orthonormalized curvilinear grids in table 6.1. Increasing σ by a factor 2 ($\sigma = 2 \cdot (p+1)^2$) gives the desired results. Apparently, the discretization is only stable when σ is sufficiently high and is not impacted by the properties of the basis functions in the physical space. Finding the critical σ for stability on curved elements is not of interest at this moment since the derivation is complex and depends on the precise mapping. Therefore the penalty parameter on all curvilinear grids will be increased by a factor 2, which is sufficient for all cases considered in this work. However, if it turns out that that a factor 2 leads to unstable results for specific cases, the value for σ can be further increased until stability is found.

Looking at the Cartesian grids and the curvilinear grids with increased σ, $\rho(\boldsymbol{B}_{GS})$ increases with increasing grid sizes. In section 4.2 it was mentioned that this increases with $1-\mathcal{O}(h^2)$ for discretizations of the Poisson problem. Displaying $1-\rho(\boldsymbol{B}_{GS})$ against the grid size on a log-log scale, see figure 6.1, shows that this is also the case for the DG discretization of the Poisson problem. Since $\rho(\boldsymbol{B}_{GS})$ is very close to unity on fine grids, the error is only reduced by a very small amount per iteration, leading to slow convergence. This motivates why relaxation methods are not suitable as standalone solvers. However, their smoothing properties can still be exploited for the design of a multigrid algorithm.

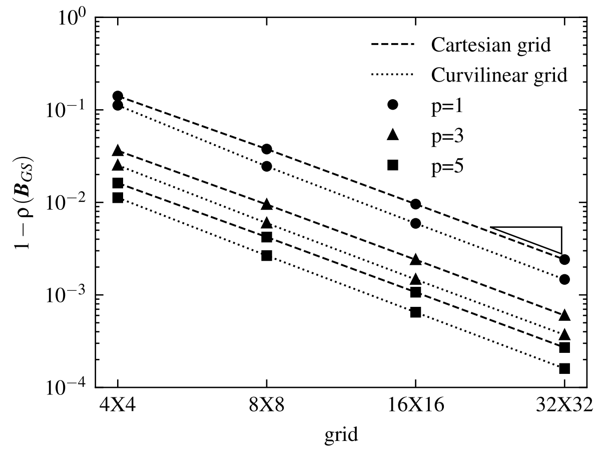


Figure 6.1: One minus the spectral radius of the Gauss-Seidel iteration matrix. The triangle indicates quadratic behavior.

The next thing to consider is whether the solution of the discretized system of equations converges to the exact solution and if so, with which order. The DG solutions on the Cartesian and curvilinear coarse and fine grids are given in figures 6.2 and 6.3. The solutions on the coarse grids elegantly illustrate the discontinuous nature of the DG method, the solution is free to jump between elements. On the fine grids the solution visually resembles the features of the prescribed exact solution very well. An overview of the exact solutions to the model problems can be found in appendix A. Figure 6.4 shows the decrease in the L_2-norm of the solution with grid refinement. From this it can be confirmed that the solutions on the fine grids in figures 6.2b and 6.3b are already very close to the exact solutions. Even though it takes some more grid refinement on the curvilinear grids before the asymptotic convergence rate is obtained, the theoretical asymptotic convergence rate of h^{p+1} is often more than achieved on both grids. The convergence rate on the coarsest curvilinear grids is probably not yet in asymptotic range because the influence of the mapping is quite significant for coarse curved elements.

Figure 6.4 also shows how accurate the higher-order solutions actually are. For example, the L_2-norm of the error of the $p=1$ solution on the $16X16$ Cartesian grid is approximately $1.6\cdot10^{-2}$, while it is already $6.5\cdot10^{-8}$ for the $p=5$ solution on the same grid. Also, following the $p=1$ convergence rate of h^2, if the grid of the $p=1$ problem were to be refined until the error in the $p=1$ solution is comparable to that of the $16X16$ $p=5$ solution, a grid size of at least $4096X4096$ elements would be needed. This corresponds to $4096\times4096\times4\approx67$

million DOFs, opposed to $16 \times 16 \times 36 = 9216$ DOFs for the $16X16$ $p = 5$ solution. While this is not an entirely fair comparison because the convergence rates are not perfectly in asymptotic range on the $16X16$ grids, it still emphasizes the huge advantage of higher-order solutions in both accuracy and problem size.

So, since the discrete systems resemble the continuous problem and the solution converges with the correct rate to the exact solution on both Cartesian and curvilinear grids, the results of the Poisson problem are verified.

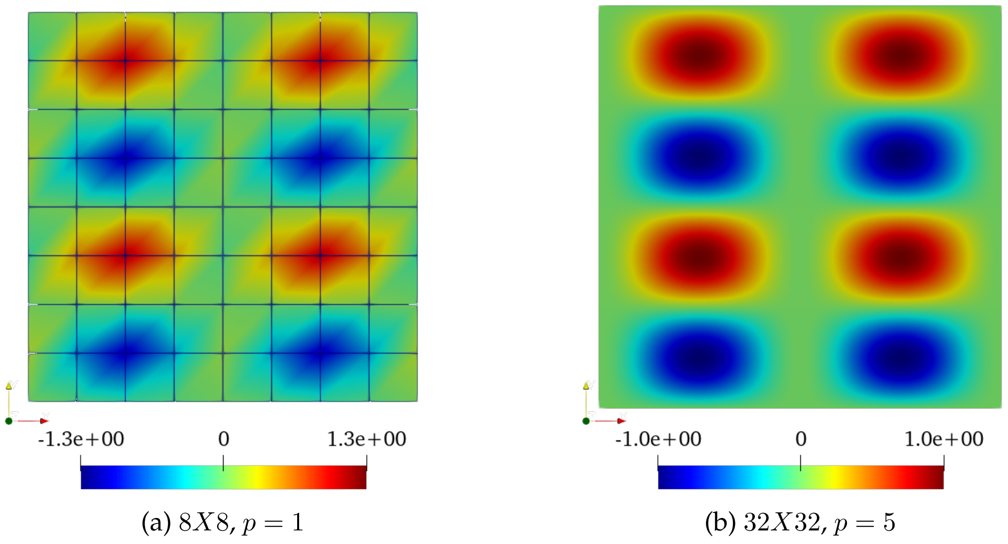


Figure 6.2: DG solutions of the Poisson problem on the Cartesian coarse (left) and fine (right) grids. For clarity, the grid lines on the fine grid (right) are not shown.

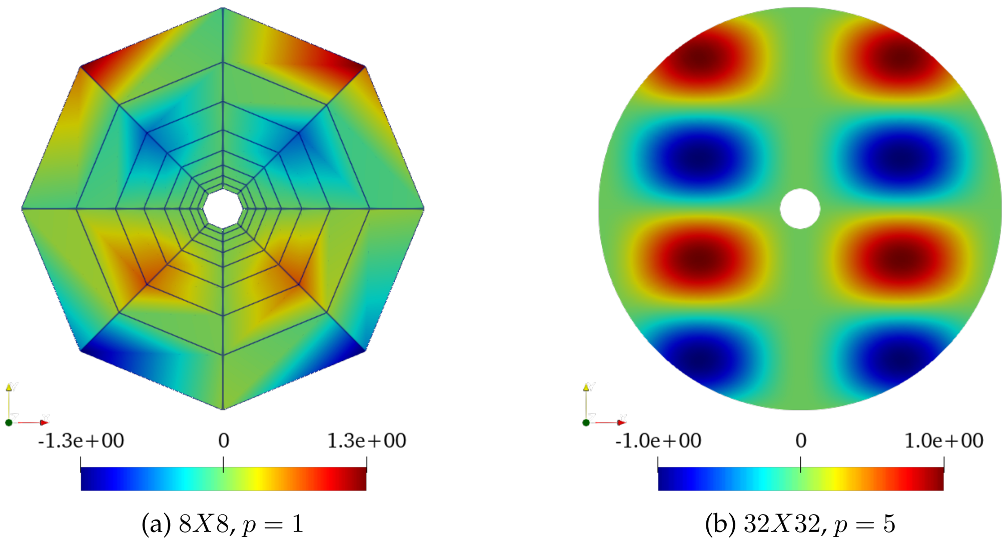
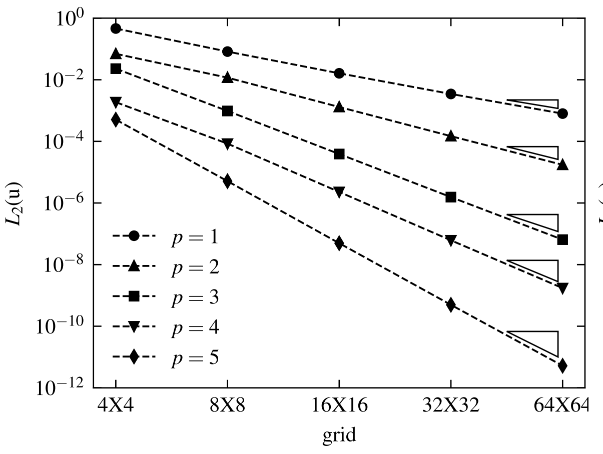


Figure 6.3: DG solutions of the Poisson problem on the curvilinear coarse (left) and fine (right) grids. For clarity, the grid lines on the fine grid (right) are not shown.



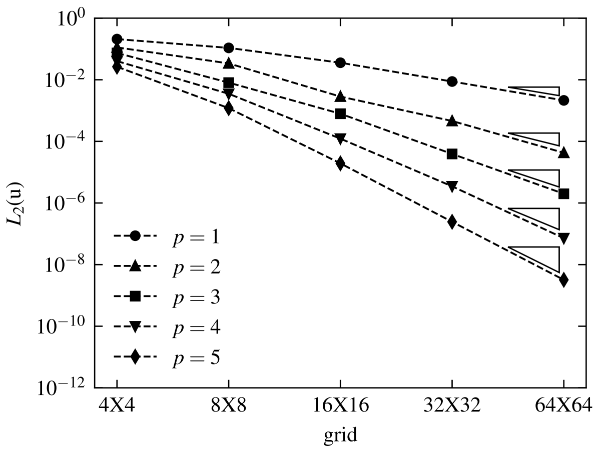


Figure 6.4: The L_2-norm of the error in u as a function of the grid size for the Poisson problem on the Cartesian grid (left) and the curvilinear grid (right). The triangles indicate the expected convergence rate h^{p+1}.

6.1.2 Stokes

Looking at the convergence rates for the Stokes solutions in figure 6.5, it shows that the solutions on the Cartesian grid show the expected behavior of h^{p+1} for the velocity and h^p for the pressure. However, on the curvilinear grid this is not the case, especially for the pressure solutions. Moreover, looking at the pressure solutions and the absolute errors $|\mathrm{p}_n - \mathrm{p}_n^{\mathrm{ex}}|$, figure 6.6, shows pressure spikes at the boundaries. Even though this effect is reduced with grid refinement from a $8X8$ $p = 5$ grid to $16X16$ $p = 5$, it is still present. This is an unexpected result, even more so because a $16X16$ $p = 5$ grid is already quite fine.

These pressure spikes could be the result of two things, an error in the implementation or a fundamental problem in the discretization. Clearly, something in the mapping has an effect on either of these causes because the solutions are correct on the Cartesian grid. The mapping from circular elements to the standard element is not exact. However, for grids where the mapping was exact and non-constant, as is the case for a square O-grid, the same results are observed, see figure 6.7. More details about the geometry of a square O-grid and the exact Stokes solution on this grid are given in section A.2.1.

Also, with increasing stabilization parameter γ, the results changed marginally and the pressure spikes were still present. Therefore, further research should identify if this effect is due to a implementation error or due to a fundamental issue in the DG formulation of the Stokes problem.

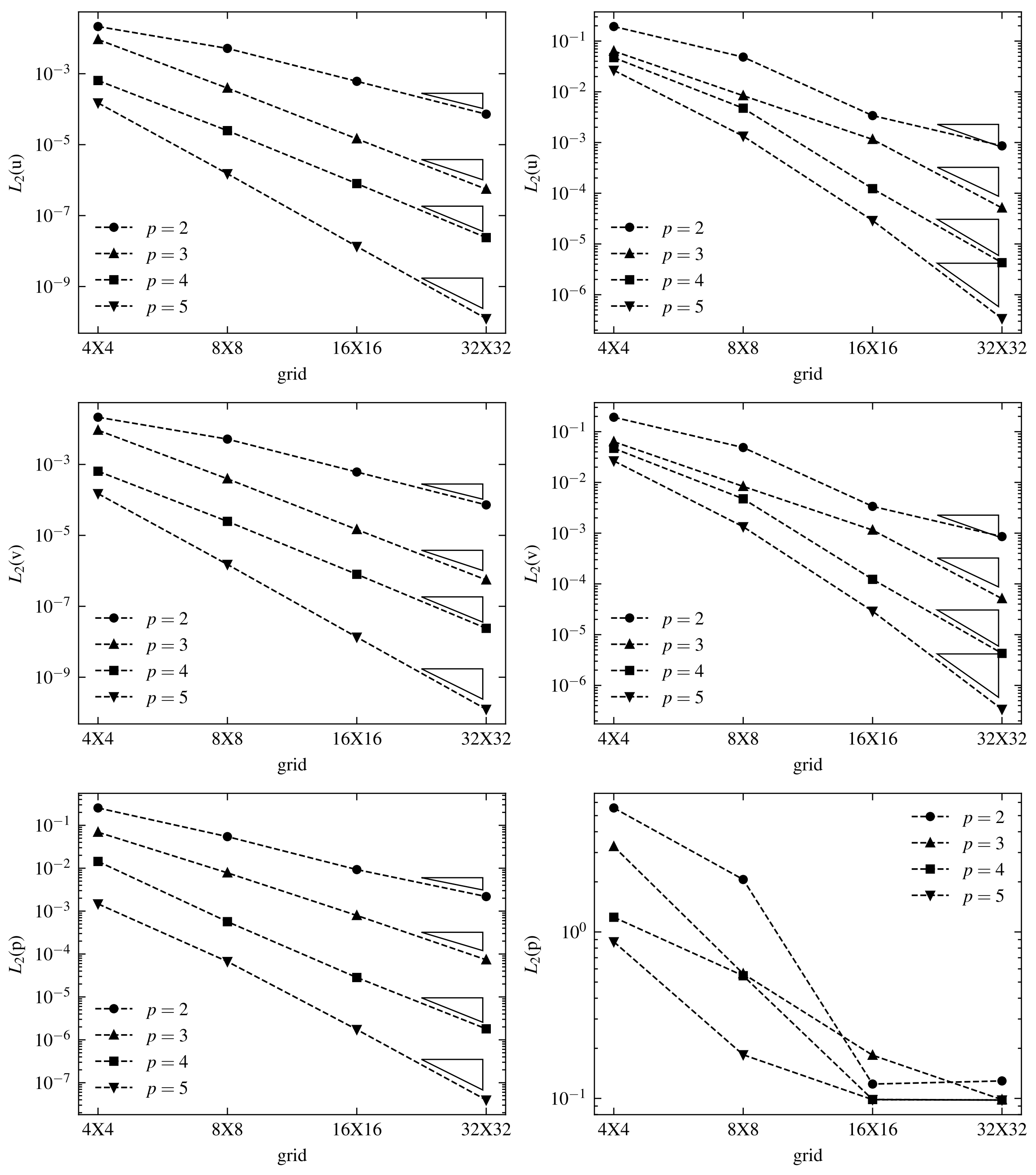
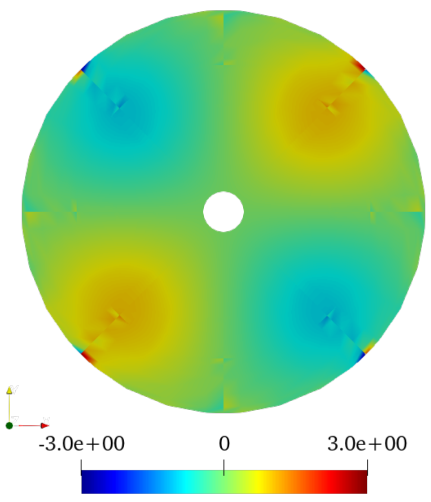
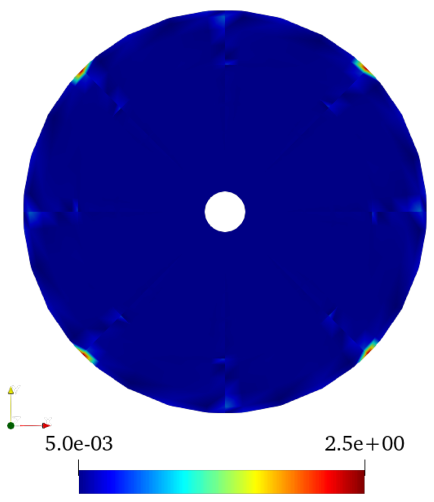


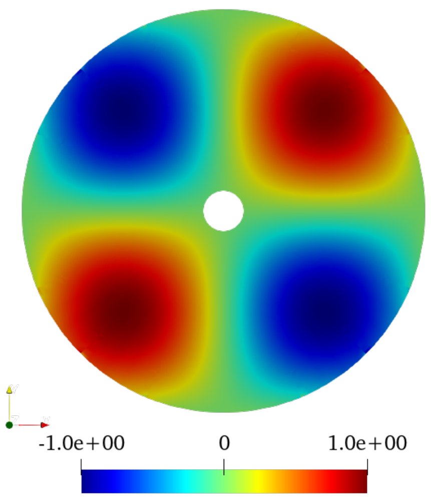
Figure 6.5: The L_2-norm of the error in u, v and p as a function of the grid size for the Stokes problem on the Cartesian grid (left) and the curvilinear grid (right). The triangles indicate the expected convergence rate h^{p+1} for the velocities and h^p for the pressure.



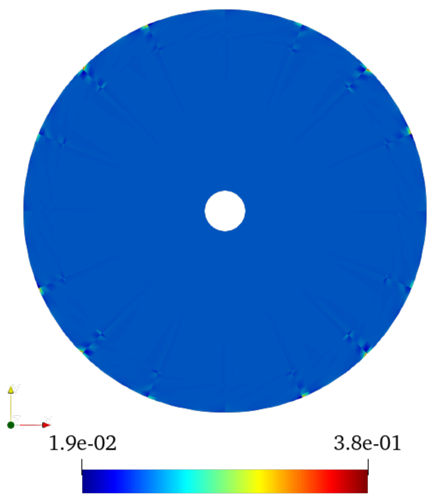
(a) $8X8$, $p = 5$



(b) $8X8$, $p = 5$

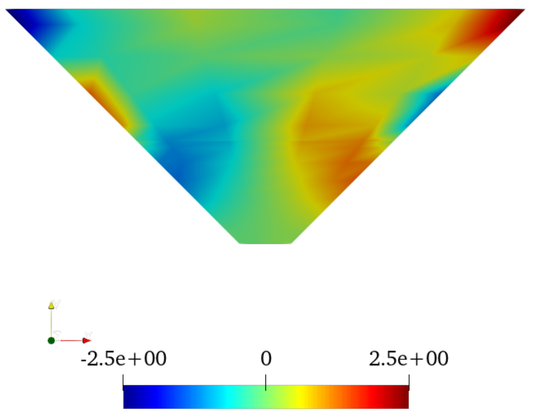


(c) $16X16$, $p = 5$

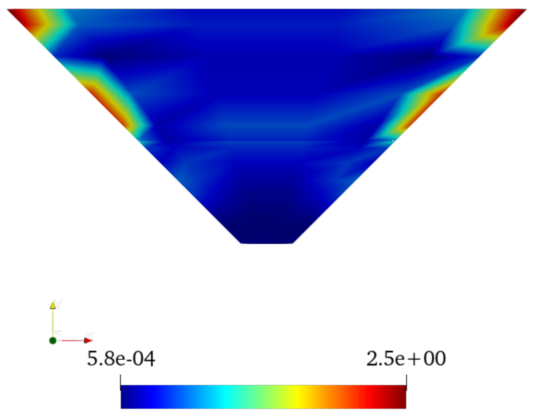


(d) $16X16$, $p = 5$

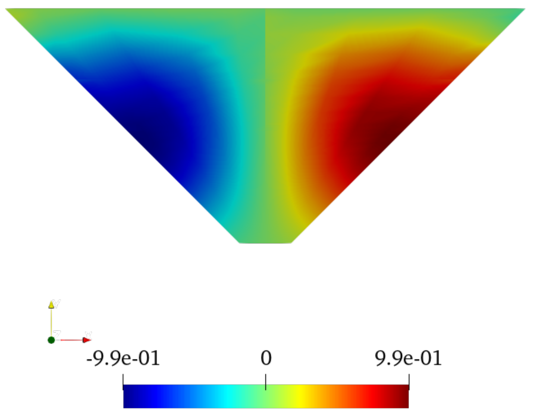
Figure 6.6: DG solutions of the pressure (left) and the absolute error in the pressure (right) of the Stokes problem on the curvilinear grid.



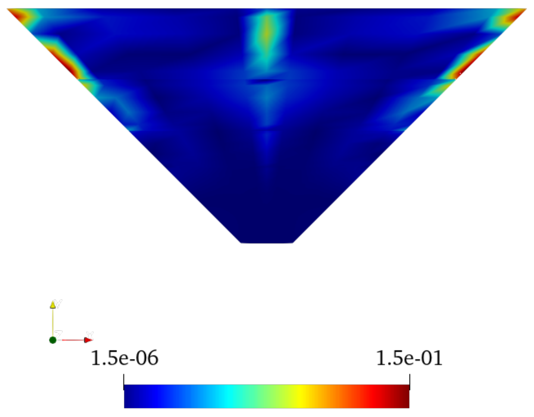
(a) $4X4$, $p = 5$



(b) $4X4$, $p = 5$



(c) $8X8$, $p = 5$



(d) $8X8$, $p = 5$

Figure 6.7: DG solutions of the pressure (left) and the absolute error in the pressure (right) of the Stokes problem on a quarter of a square O-grid.

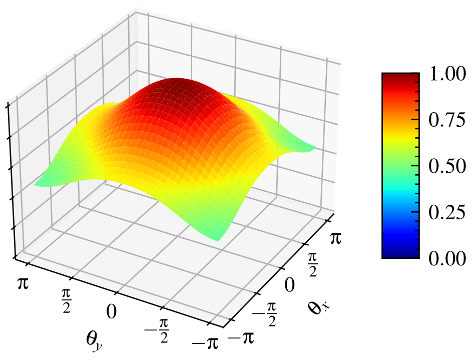
6.2 Multigrid algorithm for the Poisson problem

6.2.1 Smoother performance

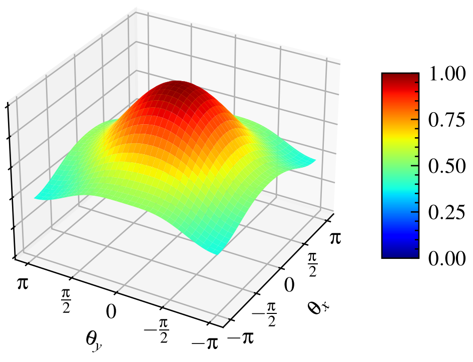
The amplitude amplification factors μ as a function of the frequencies θ_x and θ_y for a DG discretization are given in figure 6.8 for the elements 0, 1, 2 and 3 which share the midpoint of the grid, see figure 5.3. It immediately stands out that the response to the Fourier component is different in all 4 elements. This is because the Fourier component is a function that cannot be represented exactly. Hence, the solution after relaxation may be discontinuous over element boundaries, which is exactly what is observed here. This effect is smaller if the polynomial degree of the solution is increased, because this gives a more accurate solution. However, this does not mean that the smoothing behavior is the same at that polynomial order, which can be observed from figure 6.9 where μ is plotted for a $p = 6$ DG discretization.

The smoothing of the high frequency components (on which the principle of multigrid algorithms is based) is not optimal for a DG discretization compared to a FV discretization, which is plotted in figure 6.10. For a $p = 1$ DG discretization the minimum value of μ in the high frequency domain for the elements $0, 1, 2$ and 3 is still around respectively $0.47, 0.38, 0.38$ and 0.23, whereas the minimum goes to zero for a FV discretization. Moreover, convergence - and hence the smoothing performance - will be dictated by the worst component, which in this case is element 0. Also, for increasing polynomial orders the DG smoothing properties are even worse.

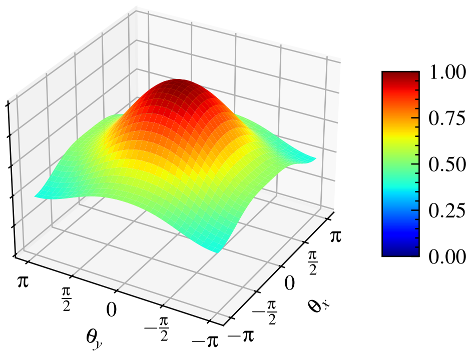
This means that the error reduction per relaxation sweep is much less for a DG discretization compared to a FV discretization and as a consequence, more sweeps will be needed to smooth the high frequency components of the error. This also motivates the use of p-multigrid before moving to geometric multigrid, because the smoothing performance is much better for a $p = 1$ solution than for example the $p = 5$ solution shown in figure 6.9. However, because even for a $p = 1$ DG discretization the smoothing performance is not optimal, symmetric block Gauss-Seidel will be used as the smoother for the multigrid algorithm. By doing a forward sweep followed by a backward sweep, essentially double the amount of relaxation is done. The number of pre- and post-relaxation sweeps will still be kept as $\nu_1 = 2$ and $\nu_2 = 1$.



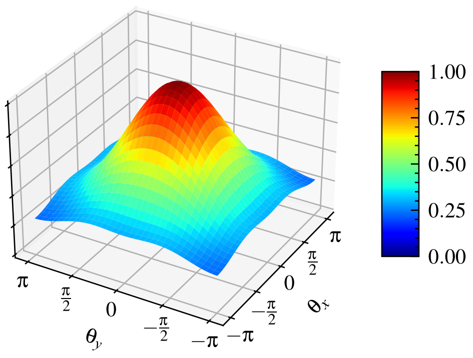
(a) Element 0



(b) Element 1

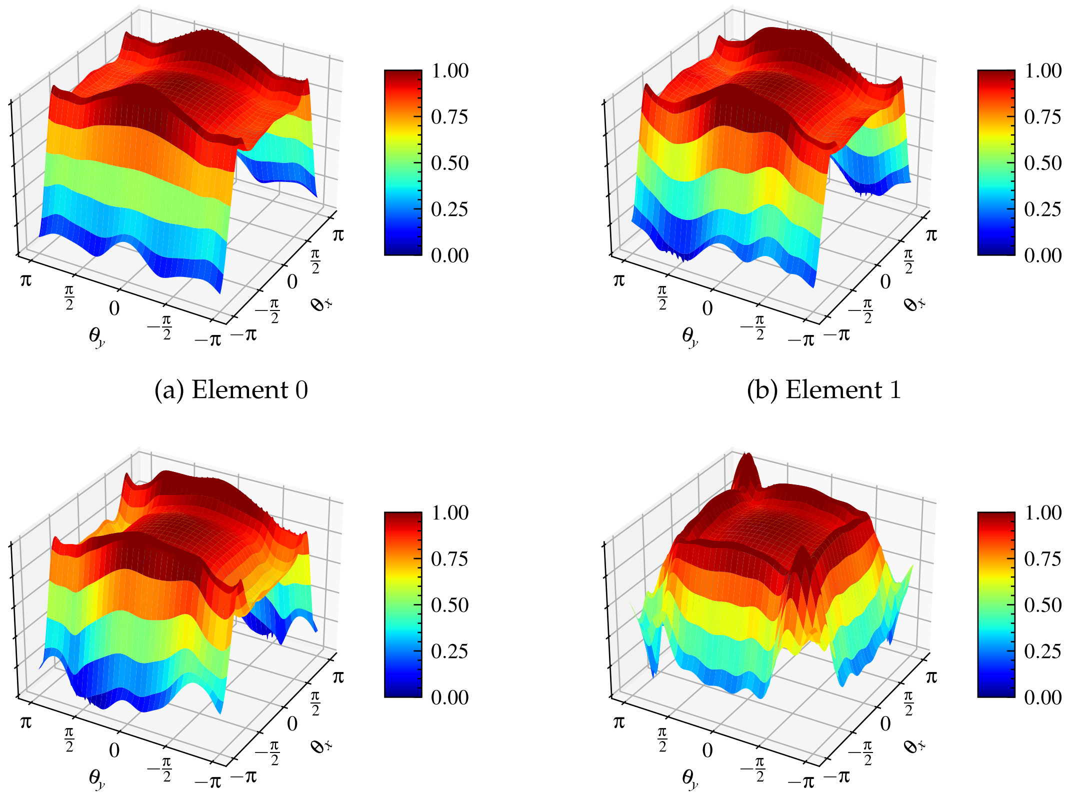


(c) Element 2



(d) Element 3

Figure 6.8: The amplitude amplification factor μ as a function of the grid frequencies θ_x and θ_y using a $p = 1$ DG discretization on a $64X64$ grid.



(a) Element 0

(b) Element 1

(c) Element 2

(d) Element 3

Figure 6.9: The amplitude amplification factor μ as a function of the grid frequencies θ_x and θ_y using a $p = 6$ DG discretization on a $64X64$ grid.

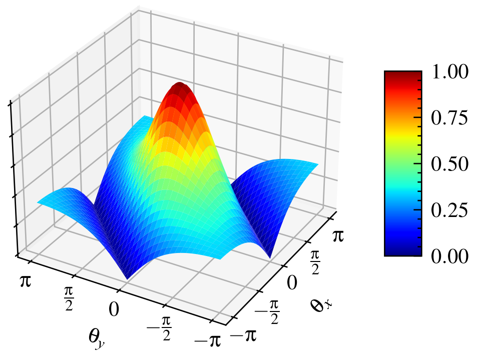


Figure 6.10: The amplitude amplification factor μ as a function of the grid frequencies θ_x and θ_y using a cell-centered FV discretization on a $64X64$ grid.

6.2.2 Multigrid algorithm performance

In theory, a multigrid algorithm should be of optimal complexity, i.e. the total workload should scale with the number of unknowns, leading to a complexity of $\mathcal{O}(N_{DOF})$. This implies that the number of multigrid cycles needed to solve the problem should remain constant with grid refinement. In table 6.3 the number of V-cycles that the hp-multigrid algorithm needs to reduce the L_2-norm of the residual by at least 6 orders of magnitudes is shown for both the Cartesian and curvilinear problems. The polynomial multigrid part is carried out in two ways, by decreasing the polynomial order of the solution with one order at a time, referred to as the $p-1$ strategy and by halving the polynomial order in each coarsening step, referred to as the $p/2$ strategy[1].

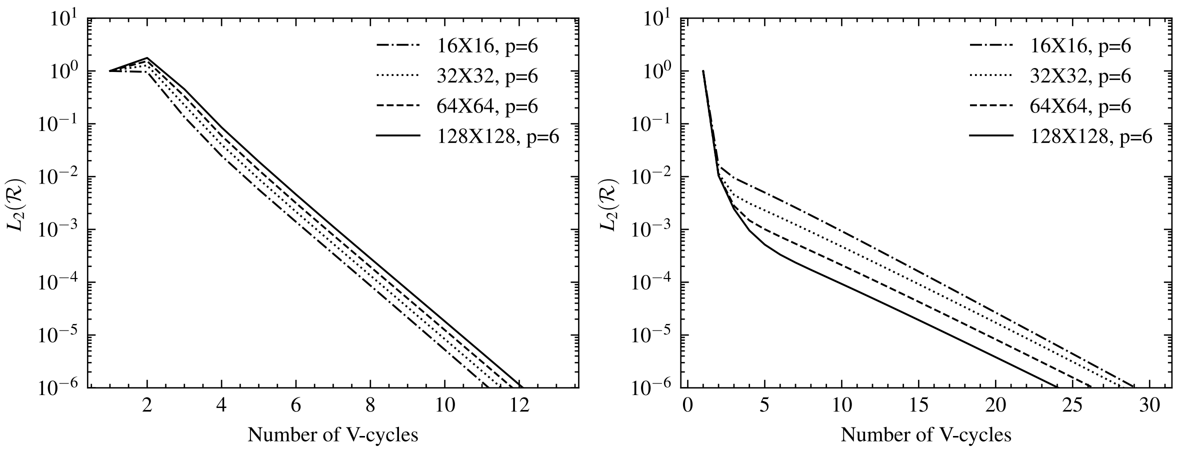
Table 6.3: Number of V-cycles of the hp-multigrid algorithm needed to reduce the L_2-norm of the residual by at least 6 orders of magnitude.

Grid size	p	Cartesian grid		Curvilinear grid	
		$p-1$	$p/2$	$p-1$	$p/2$
$16X16$	4	9	9	23	25
$32X32$	4	10	10	26	28
$64X64$	4	11	11	26	27
$128X128$	4	11	11	24	25
$16X16$	6	9	12	21	30
$32X32$	6	10	12	24	29
$64X64$	6	11	12	24	27
$128X128$	6	11	13	22	25

Firstly, the number of cycles remains relatively constant in all cases. This means that using these forms of the multigrid algorithm, each problem can be solved in roughly $\mathcal{O}(N_{DOF})$ operations. It can also be observed that a $p-1$ polynomial coarsening strategy only offers a small improvement compared to a $p/2$ strategy, which is a similar result as reported by [HA08]. It is more computationally expensive to coarsen each level by a single polynomial order and more memory needs to be allocated to store more levels. Therefore, a $p-1$ coarsening strategy provides no added benefit over a $p/2$ strategy.

Secondly, quite a lot more cycles are needed on the curvilinear grids compared to the Cartesian grids. The difference in convergence rates between these grid types is also clearly visible in figure 6.11, in which the L_2-norm of the residuals are plotted as a function of the number of V-cycles on the $p=6$ Cartesian and curvilinear grids. The lower convergence rates on curvilinear grids can partly be attributed to the increased penalty parameter σ, which must be increased on curvilinear grids to retain stability. As a result, the conditioning of the linear systems on all multigrid levels are weakened, leading to decreased smoother performance. This can be confirmed by the fact that the number of cycles increases from 9 to 17 when σ is increased with a factor 2 on the $16X16$ $p=4$ Cartesian grid, following a $p-1$ coarsening strategy. Moreover, in the derivation of the restriction and prolongation operators the influence of the mapping was neglected. While this removes the need to calculate

[1]The $p/2$ coarsening is implemented as an integer division. So for $p=6$ the p-levels are $6, 3, 1$.



(a) Cartesian grid

(b) Curvilinear grid

Figure 6.11: The L_2-norm of the residual as a function of the number of V-cycles of the hp-multigrid algorithm using a $p/2$ polynomial coarsening.

and store these operators for every element, this also has an effect on the accuracy of the restriction of the residual and prolongation of the correction between the multigrid levels. This is underlined by the observation that the number of cycles actually decreases when the fine grid elements become less distorted, which occurs when increasing the number of elements.

Even though the multigrid algorithm already scales with the number of unknowns, the smoothing properties of the DG discretization probably hamper the overall efficiency of the algorithm. The difference in smoothing performance of a DG and FV discretization was already touched upon in section 6.2.1. Since the geometric multigrid part only serves as a correction for the fine grid problem, it is not strictly necessary to use a DG discretization for these levels. Therefore, to exploit the smoothing properties of the FV method, it is attempted to use a FV discretization for the geometric multigrid levels. This is done by applying the well-known cell-centered FV discretization to the Poisson problem and using the same agglomeration based geometric coarsening as before. Since the focus of this work is on DG formulations, the reader is referred to literature for more details about a cell-centered FV discretization, for example consider [Maz16; MMD16]. First, the DG problem is coarsened to $p = 1$ before applying one last polynomial coarsening step to the $p = 0$ FV problem. Then, the geometric multigrid using the FV discretization is carried out using respectively 2 and 1 pre- and post-relaxation forward (instead of symmetric) Gauss-Seidel sweeps, since the FV method is known to smooth very well. The intergrid transfer operators for the geometric multigrid levels using the FV discretization are very similar to those derived for the DG discretization in section 4.3.3 and are worked out in more detail for the FV discretization in appendix B.2. The results for a $p = 4$ problem are shown in figure 6.12.

These results seem unexpected, why are so many cycles needed and why does the algorithm not scale with the number of unknowns anymore? As it turns out, the transition from the discontinuous $p = 1$ space to the continuous $p = 0$ space spoils the convergence of the algorithm. This was also observed by [HA08] and they claim this is due to the fact that the long wavelength eigenfunctions of the $p = 1$ discontinuous system are not represented

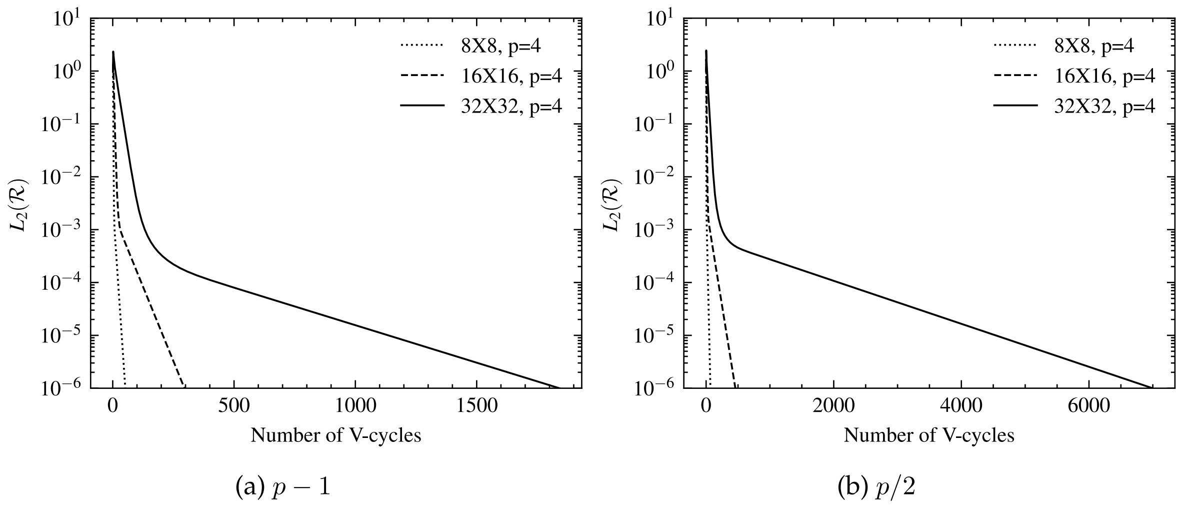


Figure 6.12: The L_2-norm of the residual as a function of the number of V-cycles of the hp-multigrid algorithm on a Cartesian grid using a FV discretization for the h-levels.

well in the $p = 0$ space. This can indeed be observed by the fact that in this case there is a significant difference between the coarsening strategies $p - 1$ and $p/2$, whereas there was no difference for the fully DG-based multigrid algorithm on a $p = 4$ Cartesian grid. This certainly hints in the direction of a poor representation of the continuous FV-space, since this was not observed for the DG-based algorithm. To overcome this, [HA08] propose a method to couple p multigrid to geometric multigrid by transitioning from the $p = 1$ discontinuous space to a $p = 1$ continuous space, i.e. a vertex-based FV discretization. Geometric multigrid is well defined for a $p = 1$ FV discretization, therefore they are able to obtain an efficient algorithm.

The question to be asked is if all this complexity is worth the gain. As seen in table 6.3 and figure 6.11, the hp-multigrid algorithm based on the geometric coarsening of a $p = 1$ DG discretization already performs independently of the problem size. Moreover, only the smoothing is slightly improved by the FV discretization. The argument that the construction of the FV discretization is cheaper than a DG discretization does not hold in this context since it is constructed for the coarse grids only, on which the work is assumed to be negligible compared to the work on the fine grid anyway. Therefore, using the existing hp-multigrid algorithm where the $p = 1$ DG based geometric multigrid algorithm is coupled to the p-multigrid algorithm could still be a viable option.

6.3 Smoother for the Stokes problem

To determine if the relaxation schemes converge, the L_2-norm of the residuals are given as a function of the number of single grid distributive Gauss-Seidel relaxation iterations in figure 6.13 for the classical splitting and in figure 6.14 for the least-squares splitting. Since only the capability of convergence is of interest at this moment, the results are plotted on a log-log scale such that convergence can be monitored for different problem sizes in one picture. The results are obtained on the Cartesian grid since the results for the Stokes problem on the curvilinear grids could not be verified in section 6.1.2. Moreover, to determine

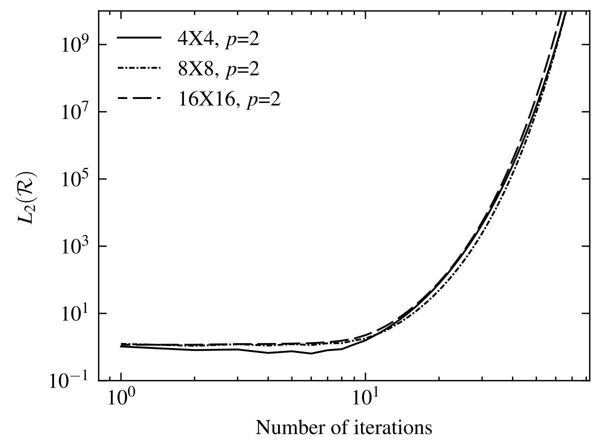


Figure 6.13: The L_2-norm of the residual as a function of the number of distributive Gauss-Seidel sweeps using the classical splitting on Cartesian grids.

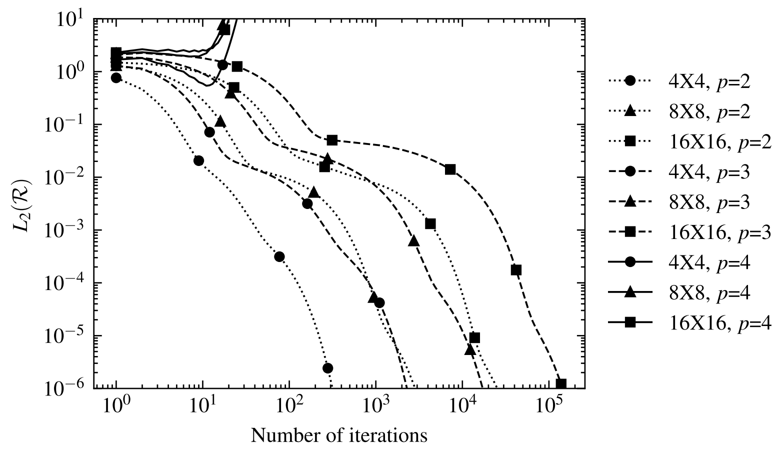


Figure 6.14: The L_2-norm of the residual as a function of the number of distributive Gauss-Seidel sweeps using the least-squares splitting on Cartesian grids.

if a distributive Gauss-Seidel relaxation scheme is applicable to the DG discretization of the Stokes problem, the characteristics of the coupled system of only the Cartesian grid are representative as well.

The results show that the relaxation using the classical splitting does not converge. As mentioned in section 4.2.3, good approximations for $\boldsymbol{A}$ and $\boldsymbol{S}$ only exist when $\boldsymbol{A}$ is diagonally dominant. In the verification of the Poisson problem in section 6.1.1, it was observed that the discrete system of the Poisson problem, which corresponds to the discretization of $\boldsymbol{A}$ in the Stokes problem, was not diagonally dominant. Hence, the approximations of $\boldsymbol{A}$ and $\boldsymbol{S}$ in the classical splitting are poor, leading to divergence.

The relaxation based on the least-squares splitting shows to converge, provided that the polynomial order of the velocity solution is not higher than fourth order (and consequently that of the pressure solution not higher than third order). In the splitting the assumption is made that the commutator $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{G}(\boldsymbol{DG})^{-1}\boldsymbol{D}$ is minimum in the least-squares sense and therefore it is neglected. While the commutator could still be minimum, it might be too significant to neglect for higher order solutions such that the relaxation does not converge anymore. Moreover, the influence of the number of sweeps on the decoupled systems could play a role in this. Currently a single symmetric block Gauss-Seidel sweep is used as the inner smoother but this could be insufficient for higher order problems.

Regardless of the fact that relaxation based on the classical splitting does not even converge in this case, it would be quite computationally expensive because the Schur complement $\boldsymbol{S}$ requires the inverse of $\boldsymbol{A}$ to be computed. Although it is tried to alleviate this by approximating $\boldsymbol{A}$ with its block diagonal, it will still be more expensive compared to the least-squares based splitting because of the multiplication $-\boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{G}$. Therefore, it would be beneficial to research the possibilities of a distributive Gauss-Seidel relaxation scheme based on the least-squares splitting that is also applicable to higher order discretizations of the DG method, or a distributive scheme that is based on a different splitting that does not have these higher-order limitations.

Chapter 7

Conclusions and Recommendations

Multiple building blocks for the construction of an efficient algorithm for DG formulations of the Stokes problem were considered in this thesis. A multigrid algorithm for the DG discretization of the Poisson problem and the DG discretization as well as a smoother for the Stokes problem were developed, implemented and tested.

Firstly, the hp-multigrid algorithm based on the polynomial and geometric coarsening of the problem using DG formulations was found to be capable of solving the problem in an amount of cycles that was independent of the problem size. Moreover, a p-coarsening strategy of $p-1$ led to only a minor improvement compared to a $p/2$ strategy. Since it is more computationally expensive to coarsen the polynomial order with one order at a time, it is a better choice to use a coarsening strategy of $p/2$. Based on the observation that the smoothing of the DG system was inferior compared to a FV discretization, it was attempted to couple a geometric multigrid algorithm based on a cell-centered FV discretization to the polynomial multigrid algorithm. It turned out that this lead to poor convergence rates due to the fact that the long wavelength eigenfunctions of the $p=1$ discontinuous DG scheme are not represented well in the $p=0$ space. For future research, the performance of a multigrid algorithm for the DG formulation of the Poisson problem where the $p=1$ discontinuous space is coupled to a $p=1$ continuous space using a vertex-based FV discretization as proposed by [HA08] could be investigated.

Secondly, the DG formulation of the Stokes problem was derived and verified. For the Cartesian grids the results show the correct convergence behavior of h^{p+1} for the velocities and h^p for the pressure. However, incorrect results in the form of pressure spikes near the boundaries were observed on the curvilinear grids. This could be due to an implementation error or a fundamental problem in the discretization scheme. Further research should identify the underlying cause of these pressure spikes on curvilinear grids. Moreover, to avoid odd-even decoupling, the polynomial order of the pressure space was taken one order lower than the velocity space, leading to a decreased accuracy for the pressure solution compared to the velocity solution. Further research could be done to investigate if it is possible to stabilize the discretization when the polynomial order of the pressure and velocity spaces are equal, such that there is no accuracy difference between the solutions.

Lastly, a distributive Gauss-Seidel smoother based on the classical SIMPLE splitting was found incapable of solving the system. Because the $\boldsymbol{A}$-block, although positive definite, is not diagonally dominant, the approximations to $\boldsymbol{A}$ and $\boldsymbol{S}$ are poor, which leads to divergence.

Convergence was observed for the smoother based on the least-squares commutator for $p < 4$. For higher order problems the assumption that the commutator-block could be neglected is probably not correct anymore. Since the least-squares based splitting does not require the computation of the inverse of the $\boldsymbol{A}$-block (or an approximate of it), it will be computationally less expensive. Therefore, it could be useful to conduct further research on the distributive Gauss-Seidel smoother based on the least-squares splitting that can also be applied to higher order, $p \geq 4$, DG discretizations. Also, the effect of the number and type of inner sweeps on the decoupled systems could be of importance in this context. Moreover, it may be possible to define an alternative splitting that does not have these limitations for higher order DG discretizations.

Bibliography

[Bak08] Andre Bakker. “Lectures on Applied Computational Fluid Dynamics”. en. In: (2008).

[BS19] Claus Beisbart and Nicole J. Saam, eds. *Computer Simulation Validation: Fundamental Concepts, Methodological Frameworks, and Philosophical Perspectives*. en. Simulation Foundations, Methods and Applications. Cham: Springer International Publishing, 2019. ISBN: 978-3-319-70765-5 978-3-319-70766-2. DOI: 10.1007/978-3-319-70766-2. URL: https://link.springer.com/10.1007/978-3-319-70766-2 (visited on 10/02/2023).

[BGL05] Michele Benzi, Gene H. Golub, and Jörg Liesen. “Numerical solution of saddle point problems”. en. In: *Acta Numerica* 14 (May 2005), pp. 1–137. ISSN: 0962-4929, 1474-0508. DOI: 10.1017/S0962492904000212. URL: https://www.cambridge.org/core/product/identifier/S0962492904000212/type/journal_article (visited on 03/28/2023).

[BB15] Bruno Blais and François Bertrand. “On the use of the method of manufactured solutions for the verification of CFD codes for the volume-averaged Navier–Stokes equations”. en. In: *Computers & Fluids* 114 (July 2015), pp. 121–129. ISSN: 00457930. DOI: 10.1016/j.compfluid.2015.03.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793015000675 (visited on 08/01/2023).

[Bot12] Lorenzo Botti. “Influence of Reference-to-Physical Frame Mappings on Approximation Properties of Discontinuous Piecewise Polynomial Spaces”. en. In: *Journal of Scientific Computing* 52.3 (Sept. 2012), pp. 675–703. ISSN: 0885-7474, 1573-7691. DOI: 10.1007/s10915-011-9566-3. URL: http://link.springer.com/10.1007/s10915-011-9566-3 (visited on 04/24/2023).

[Cha+14] J.-B. Chapelier et al. “Evaluation of a high-order discontinuous Galerkin method for the DNS of turbulent flows”. en. In: *Computers & Fluids* 95 (May 2014), pp. 210–226. ISSN: 00457930. DOI: 10.1016/j.compfluid.2014.02.015. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793014000784 (visited on 09/26/2023).

[Chr10] Lehrenfeld Christoph. “Hybrid Discontinuous Galerkin Methods for Solving Incompressible Flow Problems”. PhD thesis. Rheinisch-Westfälischen Technischen Hochschule Aachen, May 2010. URL: https:%20//www.igpm.rwth-aachen.de/Download/reports/lehrenfeld/DA_HDG4NSE_1_0.pdf.

[Dai11] Daily. "Lightning and fire: Japan on alert after volcano's biggest eruption in 50 years". In: *Mail Online* (Jan. 2011). Section: News. URL: https://www.dailymail.co.uk/news/article-1351064/Japan-raises-alert-following-volcanos-biggest-eruption-50-years.html (visited on 10/01/2023).

[Dub91] Moshe Dubiner. "Spectral methods on triangles and other domains". en. In: *Journal of Scientific Computing* 6.4 (Dec. 1991), pp. 345–390. ISSN: 0885-7474, 1573-7691. DOI: 10.1007/BF01060030. URL: http://link.springer.com/10.1007/BF01060030 (visited on 08/07/2023).

[GLS19] Nicolas R. Gauger, Alexander Linke, and Philipp W. Schroeder. "On high-order pressure-robust space discretisations, their advantages for incompressible high Reynolds number generalised Beltrami flows and beyond". en. In: *The SMAI journal of computational mathematics* 5 (Sept. 2019), pp. 89–129. ISSN: 2426-8399. DOI: 10.5802/smai-jcm.44. URL: https://smai-jcm.centre-mersenne.org/item/SMAI-JCM_2019__5__89_0 (visited on 03/28/2023).

[ges19] gestione. *Turbulence models in CFD - RANS, DES, LES and DNS*. en-GB. Aug. 2019. URL: https://www.idealsimulations.com/resources/turbulence-models-in-cfd/ (visited on 10/01/2023).

[Gho96] Sandip Ghosal. "An Analysis of Numerical Errors in Large-Eddy Simulations of Turbulence". en. In: *Journal of Computational Physics* 125.1 (Apr. 1996), pp. 187–206. ISSN: 00219991. DOI: 10.1006/jcph.1996.0088. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999196900881 (visited on 10/01/2023).

[Gir20] Francis X. Giraldo. *An Introduction to Element-Based Galerkin Methods on Tensor-Product Bases: Analysis, Algorithms, and Applications*. en. Vol. 24. Texts in Computational Science and Engineering. Cham: Springer International Publishing, 2020. ISBN: 978-3-030-55068-4 978-3-030-55069-1. DOI: 10.1007/978-3-030-55069-1. URL: http://link.springer.com/10.1007/978-3-030-55069-1 (visited on 03/28/2023).

[HA08] Brian T. Helenbrook and H. L. Atkins. "Solving Discontinuous Galerkin Formulations of Poisson's Equation using Geometric and p Multigrid". en. In: *AIAA Journal* 46.4 (Apr. 2008), pp. 894–902. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/1.31163. URL: https://arc.aiaa.org/doi/10.2514/1.31163 (visited on 09/25/2023).

[HW08] Jan S. Hesthaven and Tim Warburton. *Nodal Discontinuous Galerkin Methods*. en. Ed. by J. E. Marsden, L. Sirovich, and S. S. Antman. Vol. 54. Texts in Applied Mathematics. New York, NY: Springer New York, 2008. ISBN: 978-0-387-72065-4 978-0-387-72067-8. DOI: 10.1007/978-0-387-72067-8. URL: http://link.springer.com/10.1007/978-0-387-72067-8 (visited on 03/28/2023).

[Hil13] Koen Hillewaert. *Development of the discontinuous Galerkin method for high-resolution, large scale CFD and acoustics in industrial geometries*. en. Presses univ. de Louvain, 2013.

[Ibr16] Bayram Ali Ibrahimoglu. “Lebesgue functions and Lebesgue constants in polynomial interpolation”. en. In: *Journal of Inequalities and Applications* 2016.1 (Dec. 2016), p. 93. ISSN: 1029-242X. DOI: 10.1186/s13660-016-1030-3. URL: http://www.journalofinequalitiesandapplications.com/content/2016/1/93 (visited on 07/31/2023).

[Jen+14] Eleanor W. Jenkins et al. “On the parameter choice in grad-div stabilization for the Stokes equations”. en. In: *Advances in Computational Mathematics* 40.2 (Apr. 2014), pp. 491–516. ISSN: 1019-7168, 1572-9044. DOI: 10.1007/s10444-013-9316-1. URL: http://link.springer.com/10.1007/s10444-013-9316-1 (visited on 09/12/2023).

[Joh16] Volker John. *Finite Element Methods for Incompressible Flow Problems*. en. Vol. 51. Springer Series in Computational Mathematics. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-45749-9 978-3-319-45750-5. DOI: 10.1007/978-3-319-45750-5. URL: http://link.springer.com/10.1007/978-3-319-45750-5 (visited on 10/03/2023).

[Joh+17] Volker John et al. “On the Divergence Constraint in Mixed Finite Element Methods for Incompressible Flows”. en. In: *SIAM Review* 59.3 (Jan. 2017), pp. 492–544. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/15M1047696. URL: https://epubs.siam.org/doi/10.1137/15M1047696 (visited on 05/30/2023).

[KBW04] I J Keshtiban, F Belblidia, and M F Webster. “Compressible flow solvers for low Mach number flows”. en. In: (2004).

[Koe21] Roelof Koekoek. *Special Functions – Orthogonal polynomials – Jacobi polynomials*. 2021. URL: https://homepage.tudelft.nl/11r49/teaching/specfunc/orthopoly/jacobi.html (visited on 07/28/2023).

[Maz16] Sandip Mazumder. *Numerical methods for partial differential equations: finite difference and finite volume methods*. en. OCLC: ocn913556966. Amsterdam: Academic Press, 2016. ISBN: 978-0-12-849894-1.

[MM98] Parviz Moin and Krishnan Mahesh. “DIRECT NUMERICAL SIMULATION: A Tool in Turbulence Research”. en. In: *Annual Review of Fluid Mechanics* 30.1 (Jan. 1998), pp. 539–578. ISSN: 0066-4189, 1545-4479. DOI: 10.1146/annurev.fluid.30.1.539. URL: https://www.annualreviews.org/doi/10.1146/annurev.fluid.30.1.539 (visited on 10/01/2023).

[MMD16] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*. en. Vol. 113. Fluid Mechanics and Its Applications. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-16873-9 978-3-319-16874-6. DOI: 10.1007/978-3-319-16874-6. URL: https://link.springer.com/10.1007/978-3-319-16874-6 (visited on 09/05/2023).

[OL06] C.W. Oosterlee and F.J.G. Lorenz. “Multigrid Methods for the Stokes System”. en. In: *Computing in Science & Engineering* 8.6 (Nov. 2006), pp. 34–43. ISSN: 1521-9615. DOI: 10.1109/MCSE.2006.115. URL: http://ieeexplore.ieee.org/document/1717313/ (visited on 06/26/2023).

[Pan13] Ronald Lee Panton. *Incompressible flow*. eng. 4. ed. Hoboken, NJ: Wiley, 2013. ISBN: 978-1-118-01343-4.

[PS83] S.V. Patankar and D.B. Spalding. "A CALCULATION PROCEDURE FOR HEAT, MASS AND MOMENTUM TRANSFER IN THREE-DIMENSIONAL PARABOLIC FLOWS". en. In: *Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion*. Elsevier, 1983, pp. 54–73. ISBN: 978-0-08-030937-8. DOI: 10.1016/B978-0-08-030937-8.50013-1. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780080309378500131 (visited on 10/02/2023).

[QSS07] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. "Iterative Methods for Solving Linear Systems". In: *Numerical Mathematics*. Vol. 37. Series Title: Texts in Applied Mathematics. New York, NY: Springer New York, 2007, pp. 123–181. DOI: 10.1007/978-0-387-22750-4_4. URL: http://link.springer.com/10.1007/978-0-387-22750-4_4 (visited on 09/10/2023).

[Saa03] Y. Saad. *Iterative methods for sparse linear systems*. 2nd ed. Philadelphia: SIAM, 2003. ISBN: 978-0-89871-534-7.

[Sch19] Philipp W. Schroeder. "Robustness of High-Order Divergence-Free Finite Element Methods for Incompressible Computational Fluid Dynamics". en. PhD thesis. Georg-August-University Göttingen, 2019. DOI: 10.53846/goediss-7330. URL: https://ediss.uni-goettingen.de/handle/11858/00-1735-0000-002E-E5BC-8 (visited on 05/30/2023).

[Sci23a] SciPy. *scipy.special.eval_jacobi — SciPy v1.11.1 Manual*. 2023. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.eval_jacobi.html#scipy.special.eval_jacobi (visited on 07/28/2023).

[Sci23b] SciPy. *scipy.special.roots_jacobi — SciPy v1.11.1 Manual*. 2023. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.roots_jacobi.html#scipy.special.roots_jacobi (visited on 07/28/2023).

[Sym23] SymPy. *SymPy 1.12 documentation*. 2023. URL: https://docs.sympy.org/latest/index.html (visited on 10/14/2023).

[Sze39] Gábor Szegő. *Orthogonal polynomials*. 4th ed. Colloquium publications - American Mathematical Society v. 23. Providence: American Mathematical Society, 1939. ISBN: 978-0-8218-1023-1.

[The04] The Engineering Toolbox. *Liquids - Speed of Sound*. 2004. URL: https://www.engineeringtoolbox.com/sound-speed-liquids-d_715.html (visited on 10/11/2023).

[TDB03] James L. Thomas, Boris Diskin, and Achi Brandt. "Textbook Multigrid Efficiency For Fluid Simulations". en. In: *Annual Review of Fluid Mechanics* 35.1 (Jan. 2003), pp. 317–340. ISSN: 0066-4189, 1545-4479. DOI: 10.1146/annurev.fluid.35.101101.161209. URL: https://www.annualreviews.org/doi/10.1146/annurev.fluid.35.101101.161209 (visited on 10/12/2023).

[TB97] Lloyd N. Trefethen and David Bau. *Numerical linear algebra*. en. Philadelphia: Society for Industrial and Applied Mathematics, 1997. ISBN: 978-0-89871-361-9.

[US 11] U.S. Geological Survey. *2011 Eruption of Shinmoedake, Japan. Note the eruption plume being ... | U.S. Geological Survey*. 2011. URL: https://www.usgs.gov/media/images/2011-eruption-shinmoedake-japan-note-eruption-plume-being (visited on 10/04/2023).

[VL00] C. H. Venner and A. A. Lubrecht. *Multilevel methods in lubrication*. en. 1st ed. Tribology series 37. Amsterdam ; New York: Elsevier, 2000. ISBN: 978-0-444-50503-3.

[WC13] Ming Wang and Long Chen. "Multigrid Methods for the Stokes Equations using Distributive Gauss–Seidel Relaxations based on the Least Squares Commutator". en. In: *Journal of Scientific Computing* 56.2 (Aug. 2013), pp. 409–431. ISSN: 0885-7474, 1573-7691. DOI: 10.1007/s10915-013-9684-1. URL: http://link.springer.com/10.1007/s10915-013-9684-1 (visited on 06/26/2023).

Appendices

Appendix A

Exact solutions

This section contains the exact solutions to the Poisson and Stokes problems. These solutions correspond to the manufactured solutions explained in section 5.3

A.1 Poisson

The manufactured solution that is used for the Poisson problem is defined by equation (5.3) and is repeated:

$$\mathrm{u} = -2\sin(\pi x)^2 \sin(\pi y)\cos(\pi y). \tag{A.1}$$

The exact solution u is presented in figure A.1.

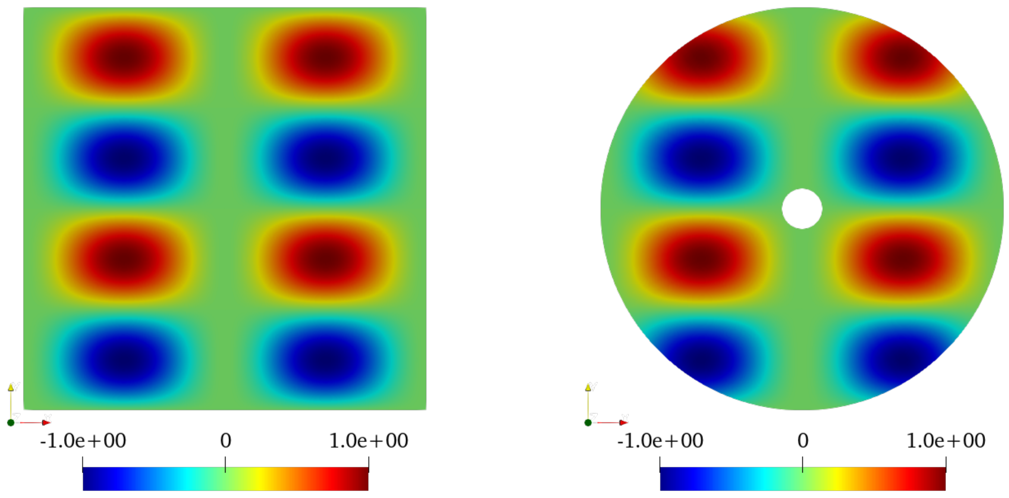


Figure A.1: Exact solution u of the Poisson problem on a $32X32$ $p = 5$ Cartesian grid (left) and curvilinear grid (right).

A.2 Stokes

The manufactured solutions that are used for the Stokes problem are defined by equations (5.3) to (5.5) and are repeated:

$$\mathrm{u} = -2\sin(\pi x)^2 \sin(\pi y)\cos(\pi y), \tag{A.2}$$

$$\mathrm{v} = 2\sin(\pi x)\cos(\pi x)\sin(\pi y)^2, \tag{A.3}$$

$$\mathrm{p} = \sin(\pi x)\sin(\pi y) - \bar{\mathrm{p}}, \tag{A.4}$$

The exact solutions u, v and p are presented in figures A.2 to A.4, respectively.

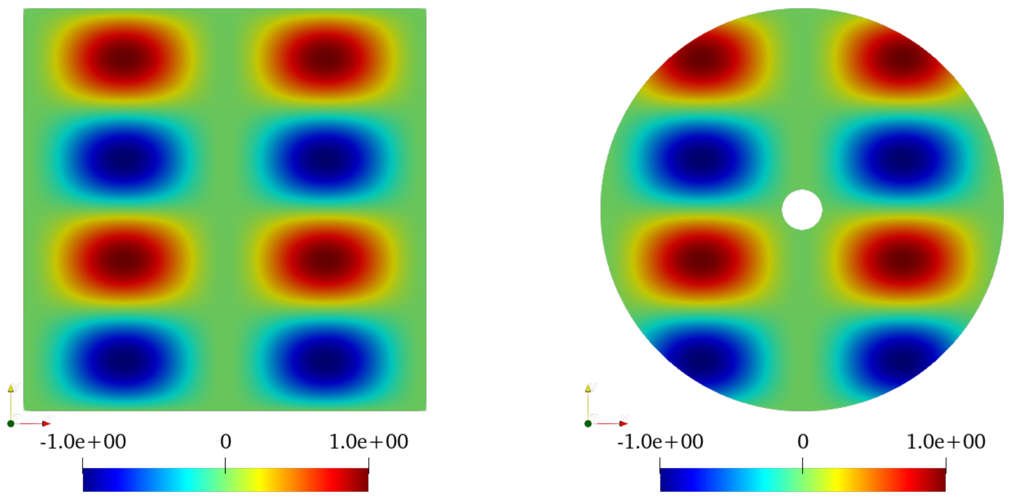


Figure A.2: Exact solution u of the Stokes problem on a $32X32$ $p = 5$ Cartesian grid (left) and curvilinear grid (right).

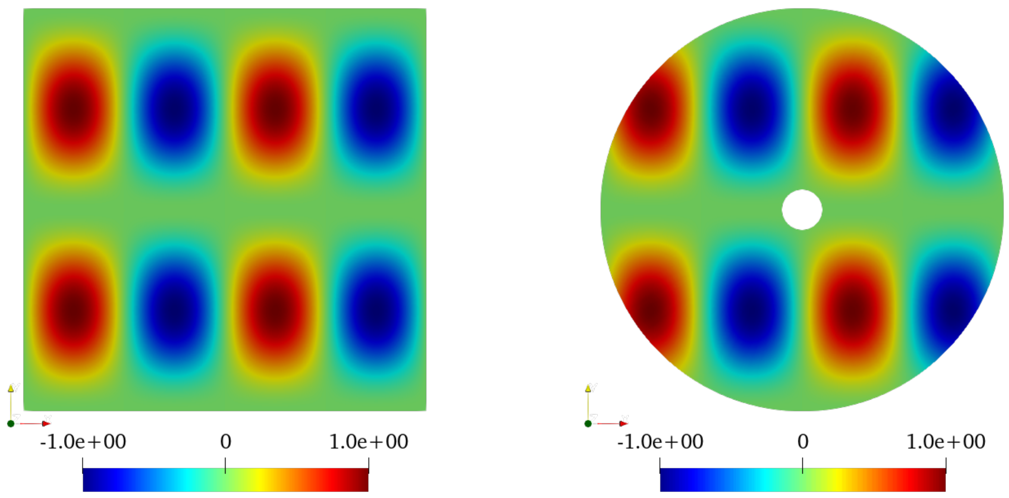


Figure A.3: Exact solution v of the Stokes problem on a $32X32$ $p = 5$ Cartesian grid (left) and curvilinear grid (right).

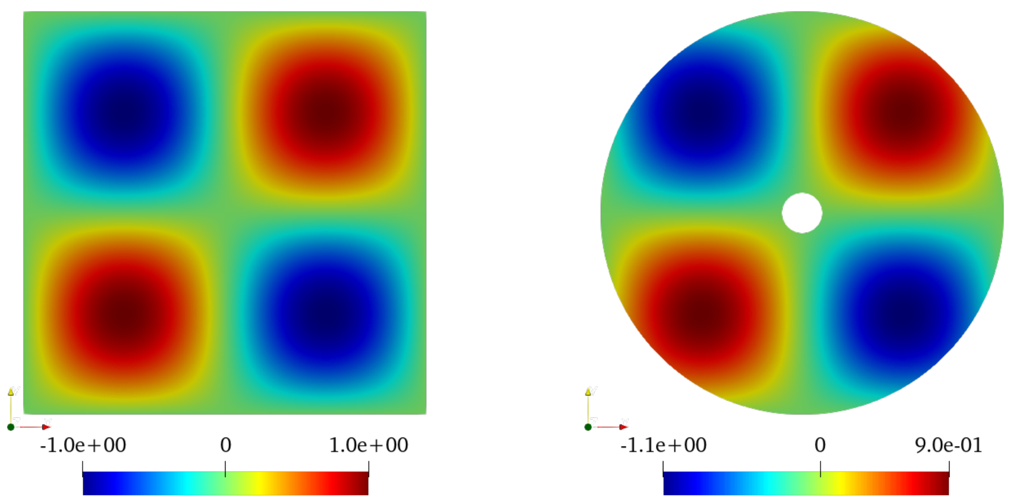
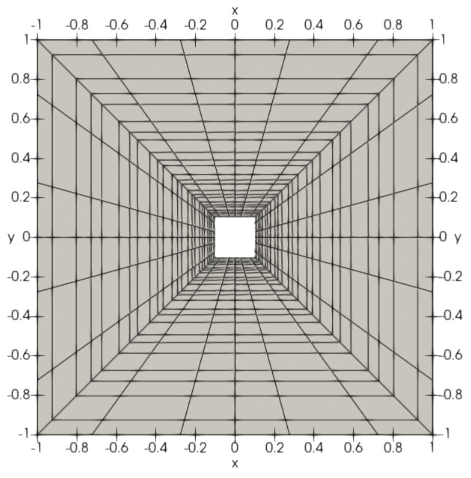


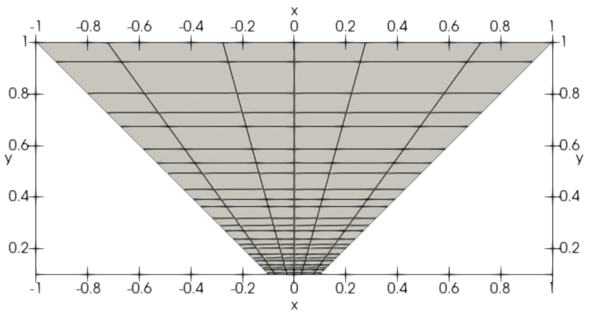
Figure A.4: Exact solution p of the Stokes problem on a $32X32$ $p = 5$ Cartesian grid (left) and curvilinear grid (right).

A.2.1 Square O-grid

The square O-grid depicted in figure A.5a is very similar to the circular O-grid illustrated in figure 5.1b. The difference is that the mapping from the elements of the square O-grid to the standard elements is exact (which is not the case for the standard curvilinear grid in figure 5.1b). The dimensions of the square O-grid are similar to the curvilinear grid in figure 5.1b. The square O-grid is further simplified by taking a quarter of it, resulting in the grid depicted in figure A.5b. The exact solution given in equation (A.4) is represented on a quarter of the square O-grid in figure A.6.



(a) Square O-grid



(b) Trapezoidal grid

Figure A.5: Square O-grid (left) and a quarter of the square O-grid (right).

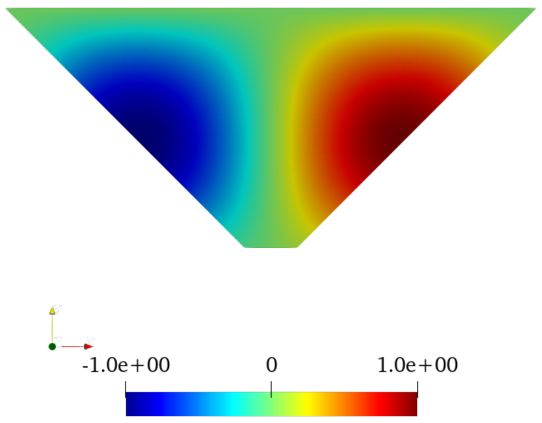


Figure A.6: Exact solution p of the Stokes problem on a quarter of a $32X32$ $p = 5$ square O-grid.

Appendix B

Mathematical concepts and derivations

B.1 Gram-Schmidt orthogonalization

The Gram-Schmidt process can be used to orthonormalize a set of basis functions $\boldsymbol{\psi}$. An orthogonal set of basis functions $\hat{\boldsymbol{\psi}}$ is obtained by subtracting the projections of the basis functions ψ_j on $\hat{\psi}_j$ from it, where $j = 0, \ldots, i-1$, leading to

$$\hat{\psi}_i = \psi_i - \sum_{j=0}^{j=i-1} \operatorname{proj}_{\hat{\psi}_j} \psi_j = \psi_i - \sum_{j=0}^{j=i-1} \frac{\left\langle \psi_i,\, \hat{\psi}_j \right\rangle}{\left\langle \hat{\psi}_j,\, \hat{\psi}_j \right\rangle} \hat{\psi}_j, \tag{B.1}$$

where $\langle a,\, b\rangle$ denotes the inner product $\displaystyle\iint_{\Omega_h^e} ab \,\mathrm{d}\Omega$. As an example, the first 3 orthogonal basis functions read

$$\hat{\psi}_0 = \psi_0, \tag{B.2}$$

$$\hat{\psi}_1 = \psi_1 - \frac{\left\langle \psi_1,\, \hat{\psi}_0 \right\rangle}{\left\langle \hat{\psi}_0,\, \hat{\psi}_0 \right\rangle} \hat{\psi}_0, \tag{B.3}$$

$$\hat{\psi}_2 = \psi_2 - \frac{\left\langle \psi_2,\, \hat{\psi}_0 \right\rangle}{\left\langle \hat{\psi}_0,\, \hat{\psi}_0 \right\rangle} \hat{\psi}_0 - \frac{\left\langle \psi_2,\, \hat{\psi}_1 \right\rangle}{\left\langle \hat{\psi}_1,\, \hat{\psi}_1 \right\rangle} \hat{\psi}_1. \tag{B.4}$$

Furthermore, the set of orthonormal basis functions $\tilde{\boldsymbol{\psi}}$ is obtained by scaling the orthogonal basis functions using

$$\tilde{\psi}_i = \frac{\hat{\psi}_i}{\sqrt{\left\langle \hat{\psi}_i,\, \hat{\psi}_i \right\rangle}} \tag{B.5}$$

Due to the finite precision of a computer, the "classical" Gram-Schmidt method as described here might not yield accurate results for higher polynomial orders. In this case, the modified Gram-Schmidt method or even an algorithm based on Householder transformations can be used.

B.2 FV multigrid operators

Before the geometric FV multigrid algorithm can be performed, the $p = 1$ DG solution must be represented on a cell-centered $p = 0$ grid which can be used for the FV discretization. This is done by one polynomial coarsening step, leading to the restriction operator

$$\boldsymbol{I}_F^C = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \tag{B.6}$$

and corresponding prolongation operator $\boldsymbol{I}_C^F = \left(\boldsymbol{I}_F^C\right)^T$. The restriction operator must be multiplied by a factor 2 and the prolongation operator by a factor $\frac{1}{2}$ to account for the fact that the first modal DG basis function is $\frac{1}{2}$.

Similar to the geometric coarsening using the DG formulation, the geometric coarsening of the FV representation is done the agglomeration of fine elements. In this case, 16 fine elements are merged together to form 4 coarse elements, which is depicted in figure B.1. As a result of the cell-centered FV discretization, the solutions at the cell-centers are represented by the nodal formulation, which for the coarse grid cells read

$$\xi_e^C = \xi_0\ell_0 + \xi_1\ell_1 + \xi_2\ell_2 + \xi_3\ell_3, \tag{B.7}$$

with corresponding basis

$$\ell_0 = \frac{(1-r)(1-s)}{4}, \tag{B.8}$$
$$\ell_1 = \frac{(1+r)(1-s)}{4}, \tag{B.9}$$
$$\ell_2 = \frac{(1-r)(1+s)}{4}, \tag{B.10}$$
$$\ell_3 = \frac{(1+r)(1+s)}{4}. \tag{B.11}$$

The prolongation of the coarse grid solution to a fine grid cell can now be obtained by evaluating equation (B.7) in the (r, s)-location of the fine grid cell centers. For example, the fine grid cell 5 with $(r, s) = (-\frac{1}{2}, -\frac{1}{2})$ has contributions from the coarse grid cells 0, 1, 2 and 3, leading to

$$\xi_5^F = \xi_0\ell_0(-\tfrac{1}{2}, -\tfrac{1}{2}) + \xi_1\ell_1(-\tfrac{1}{2}, -\tfrac{1}{2}) + \xi_2\ell_2(-\tfrac{1}{2}, -\tfrac{1}{2}) + \xi_3\ell_3(-\tfrac{1}{2}, -\tfrac{1}{2}) \tag{B.12}$$
$$= \tfrac{9}{16}\xi_0 + \tfrac{3}{16}\xi_1 + \tfrac{3}{16}\xi_2 + \tfrac{1}{16}\xi_3. \tag{B.13}$$

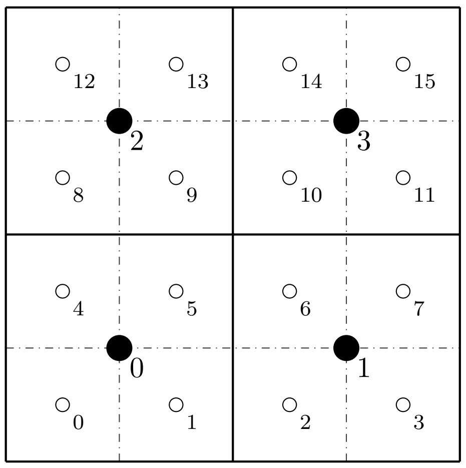


Figure B.1: Geometric coarsening based on agglomeration of fine grid elements (dashed lines) into coarse grid elements (solid lines). The cell centers of the fine elements are illustrated by an open circle and those of the coarse elements by a filled circle.

Doing this for all fine grid cells leads to the prolongation operator

$$\boldsymbol{I}_C^F = \begin{bmatrix} 9 & 0 & 0 & 0 \\ 9 & 3 & 0 & 0 \\ 3 & 9 & 0 & 0 \\ 0 & 9 & 0 & 0 \\ 9 & 0 & 9 & 0 \\ 9 & 3 & 3 & 1 \\ 3 & 9 & 1 & 3 \\ 0 & 9 & 0 & 3 \\ 3 & 0 & 9 & 0 \\ 3 & 1 & 9 & 3 \\ 1 & 3 & 3 & 9 \\ 0 & 3 & 0 & 9 \\ 0 & 0 & 9 & 0 \\ 0 & 0 & 9 & 3 \\ 0 & 0 & 3 & 9 \\ 0 & 0 & 0 & 9 \end{bmatrix} /16. \tag{B.14}$$

In this case, the contributions of the boundaries are neglected but these could be added by using a layer of ghost cells. The restriction operator follows from $\boldsymbol{I}_F^C = 4\left(\boldsymbol{I}_C^F\right)^T$.