FRIEDRICH-ALEXANDER-UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT • DEPARTMENT INFORMATIK

Lehrstuhl für Informatik 10 (Systemsimulation)

Numerical reconstruction of fundamental solutions of the Stokes system using Finite Elements

Jacob Snoeijer

Master Thesis

Numerical reconstruction of fundamental solutions of the Stokes system using Finite Elements

Jacob Snoeijer

Master Thesis

Aufgabensteller:	Prof. Dr. Ulrich Rüde
Betreuer:	Dr.-Ing. Dominik Bartuschat, M.Sc. Dominik Thönnes M. Sc.
Bearbeitungszeitraum:	1.3.2016 – 31.8.2016

Erklärung:

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Der Universität Erlangen-Nürnberg, vertreten durch den Lehrstuhl für Systemsimulation (Informatik 10), wird für Zwecke der Forschung und Lehre ein einfaches, kostenloses, zeitlich und örtlich unbeschränktes Nutzungsrecht an den Arbeitsergebnissen der Master Thesis einschließlich etwaiger Schutzrechte und Urheberrechte eingeräumt.

Erlangen, den 15. August 2016 .....

Contents

Abstract

Many phenomena in physics can be described by conservation laws such as the *conservation of mass* or the *conservation of momentum*. These laws are represented by partial differential equations, for example the Stokes system which describes viscous flow. In this thesis the fundamental solution of the Stokes system using finite elements in the unit cube in $\mathbb{R}^3$ is numerically reconstructed. Second order convergence is obtained for the fundamental solution of the Poisson problem when we go to finer meshes. Inspired by this result for the Poisson problem, similar results for the fundamental solution of the Stokes problem are shown.

The analytical expression for the Stokeslet as the fundamental solution of the Stokes system is derived. For the finite element method, we derived the weak formulation of the Stokes system and changed the used integration rule for the forcing term by the definition of the integration of a Dirac force.

For Dirac forces located at the vertices of the finite elements, second order convergence for both velocity and pressure is obtained. For Dirac forces located inside an element (tetrahedral elements are used) we were able to recover second order convergence for the velocity but only first order convergence for the pressure.

We observed oscillations in the reconstructed pressure and tried to get rid of the oscillations by using regularization strategies. The regularized Stokeslet was used, where a smooth cutoff function approximates the Dirac measure. In another regularization strategy tent functions are used as spreading of the Dirac measure over an element. Both strategies are able to reduce the magnitude of the oscillations but both are not able to remove the oscillations.

Finally, scalability experiments for the two time consuming parts in the reconstruction of a Stokeslet are shown. First, in a weak scalability experiment where we reconstructed the Stokeslet up to an island of SuperMUC, we found good scalability for the SchurCG Stokes solver. A parallel efficiency of 73% from one to 512 nodes (i.e. 14,336 cores) for the numerical solver is shown. For the integration of the forcing function a second weak scalability experiment is shown. In that experiment we fixed the concentration of the Dirac forces in an increasing domain and found a parallel efficiency of 95% from one to 256 nodes (i.e. 7,168 cores) where up to 1,376,256 Dirac forces are integrated.

Samenvatting

Een heel aantal verschijnsels in de natuurkunde kunnen worden beschreven met behoudswetten, zoals het *behoud van massa* of het *behoud van impuls (Engels: momentum)*. Deze behoudswetten worden gerepresenteerd door partiële differentiaal vergelijkingen zoals bijvoorbeeld de Stokes vergelijkingen voor viskeuze stroming. In deze scriptie is de fundamentele oplossing van de Stokes vergelijkingen met behulp van eindige elementen in de eenheidskubus in $\mathbb{R}^3$ numeriek gereconstrueerd. Tweede orde convergentie is verkregen voor de fundamentele oplossing van het Poisson probleem als we naar fijnere roosters gaan. Geïnspireerd door dit resultaat voor het Poisson probleem worden soortgelijke resultaten voor de fundamentele oplossing van de Stokes vergelijkingen getoond.

De analytische uitdrukking voor de Stokeslet, als de fundamentele oplossing voor de Stokes vergelijkingen, is afgeleid. Voor de eindige elementen methode leiden we de zwakke vorm van de Stokes vergelijkingen af en veranderen we de integratie regel voor de kracht-term door de definitie van de integratie van een Dirac kracht.

Voor Dirac krachten op hoekpunten van de eindige elementen is tweede orde convergentie voor zowel de snelheid als de druk verkregen. Voor Dirac krachten in een element (als eindige elementen worden tetraëders gebruikt) kunnen we ook tweede orde convergentie bereiken voor de snelheid, maar voor de druk krijgen we alleen eerste orde convergentie.

We merken op dat er oscillaties zijn in de gereconstrueerde druk en proberen deze oscillaties op te lossen door regularisatie strategieën. De geregulariseerde Stokeslet is gebruikt, waar de Dirac maat wordt benaderd met een gladde afgesneden functie. In een andere regularisatie strategie worden tent functies gebruikt als een uitgesmeerde Dirac maat. Beide strategieën zijn instaat om de sterkte van de oscillaties te reduceren maar ze kunnen de oscillaties niet helemaal oplossen.

Ten slotte worden schaalbaarheidsonderzoeken naar de twee onderdelen in de reconstructie van de Stokeslet die veel tijd kosten getoond. Als eerste wordt in een weak scalability experiment, waar we de Stokeslet op een eiland van SuperMUC reconstrueren, de goede schaalbaarheid van de SchurCG Stokes solver getoond. Een parallelle efficiëntie van 73% van één tot 512 nodes (d.w.z. 14.336 kernen) voor de numerieke solver is getoond. Voor de integratie van de krachten functie wordt een tweede weak scalability experiment gepresenteerd. In dit experiment zetten we de concentratie van de Dirac krachten vast bij een groter wordend domein en vinden we een parallelle efficiëntie van 95% van één tot 256 nodes (d.w.z. 7.168 kernen) waar we tot 1.376.256 Dirac krachten hebben geïntegreerd.

Chapter 1

Introduction

Many phenomena in physics can be described by conservation laws such as the *conservation of mass* or the *conservation of momentum*. These conservation laws are represented by partial differential equations. For example, the motion of Newtonian fluids is given by the Navier-Stokes equations. Based on Newton's law of conservation of momentum, the Navier-Stokes equations define a balance between momentum, a diffusive viscous term and the pressure. Combined with an incompressibility assumption the Navier-Stokes equations including the continuity equation can be written in dimensionless form as

$$\begin{aligned} \mathrm{Re}\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) - \Delta\mathbf{u} + \nabla p &= \mathbf{f} \\ \nabla\cdot\mathbf{u} &= 0, \end{aligned}$$

where $\mathbf{u}$ is the velocity, p the pressure and $\mathbf{f}$ body forces. Different fluid flows are described by different values for the Reynolds number

$$\mathrm{Re} = \frac{\rho U L}{\mu},$$

where U is the characteristic velocity, L the characteristic length, ρ the density and μ the dynamic viscosity. The inertial forces of a flow can produce instabilities in the flow such as vortices. The Reynolds number can be seen as a ratio between inertial and viscous forces. Therefore, the Reynolds number is a measure of turbulence of a flow.

1.1 Stokes flow

In this thesis we will focus on slow viscous flow such that we can neglect inertial terms, so $\mathrm{Re} = 0$. There are three scenarios where $\mathrm{Re} \to 0$, and these are the cases where the characteristic length L is small, the characteristic velocity U (or characteristic time UL) is small, or the dynamic viscosity μ is high. For example, in sedimentation processes we have small velocities which result in a fluid model where $\mathrm{Re} \to 0$. Our main focus in this thesis will be on numerically solving the Stokes system which describes viscous flows

$$\begin{aligned} -\Delta\mathbf{u} + \nabla p &= \mathbf{f} \\ \nabla\cdot\mathbf{u} &= 0, \end{aligned}$$

where $\mathbf{u}$ is the velocity, p the pressure and $\mathbf{f}$ body forces.

To model sedimentation processes, we need to model interactions between a particle and the fluid. In this thesis we will numerically solve the Stokes system using *finite elements*. The influences of the particles on the fluid can be modeled by Dirac forces on the fluid. An advantage of modeling particles by Dirac forces is that we don't need to resolve the mesh to the same accuracy as the size of the particles.

1.2 Fundamental solutions

By modeling particles as Dirac forces we end up in partial differential equations where the right hand side vector $\mathbf{f}$ with body forces consists of Dirac forces. The solutions of these differential equations are called *fundamental solutions* or *Green's functions*. The fundamental solution can be seen as an impulse response on the partial differential equation. In principle a solution of a partial differential equation can be obtained by convolving the body forces with the fundamental solution.

In this thesis we will numerically reconstruct the fundamental solution of the Stokes equations, called *Stokeslet*, on a finite element mesh. As we will see, we are able to recover the Stokeslet with good accuracy in the far field of the singularity. The Dirac force is only an approximation of a particle and therefore we are especially interested in the quality of the numerical reconstruction in the far field instead of on the complete domain.

1.3 Outline

The remaining chapters in this thesis are structured as follows: in chapter 2 we will take a closer look into fundamental solutions of partial differential equations in general. Because we will compare features of our numerical reconstruction of the Stokeslet with the fundamental solution of the Poisson problem, we will derive for both problems the fundamental solution.

In chapter 3 we will derive the finite element formulation for the Stokes problem. We need to use mixed finite elements to solve for velocity $\mathbf{u}$ and pressure p. We will make a general note about stability of the mixed finite elements for the Stokes system.

The numerical solution will be computed with the Hierarchical Hybrid Grids (HHG) framework. The finite element implementation for fundamental solutions in HHG is covered in chapter 4. Further we discuss stabilization methods and the Schur complement method as numerical solver in HHG to solve Stokes systems.

In chapter 5 we will start the numerical reconstruction of the fundamental solution for the Poisson problem. For this problem there exists a proof of the accuracy that we can get on a quasi-uniform grid, as we will use in the finite element method. In this chapter we will, more or less, recover the proved second order of convergence in the far field of the singularity for the Poisson problem.

After the Poisson problem we continue in chapter 6 with the numerical reconstruction of the Stokeslet itself. For the reconstruction we will use $\mathbb{P}_1$-iso-$\mathbb{P}_2$ elements for the velocity $\mathbf{u}$ and $\mathbb{P}_1$ elements for pressure p. In this chapter we will find that, in most cases, we are able to reach second order convergence in the far field of the singularity for the Stokes problem for both velocity $\mathbf{u}$ and pressure p. We also make an observation that the reconstructed pressure p contains oscillations in certain regions. A short comparison with $\mathbb{P}_2$ finite elements for the velocity $\mathbf{u}$ and a reconstruction of the Stokeslet on a MAC-discretization is included to explore the influence of the discretization on the oscillations.

In chapter 7 we will try to reduce the oscillations by regularizing the Stokes problem by different approaches. We will use the regularized Stokeslet to try to get rid of the oscillations. Further we will use tent functions as a discrete approximation of a Dirac measure.

For the numerical reconstructions we will use the massively parallel framework Hierarchical Hybrid Grids and in chapter 8 we present scalability experiments where we will show good weak scalability for both the Schur complement CG solver as the Dirac integration for the initialization of the load vector.

Finally, in chapter 9 we will summarize the main conclusions and give a discussion and outlook on the topics covered in this thesis.

Chapter 2

Fundamental Solutions

The fundamental solution Φ of a linear differential operator L on a domain Ω is a solution to the differential equation with a Dirac measure on the right hand side of the equation

$$L\Phi(x) = \delta(x).$$

The fundamental solution can be used to construct solutions for general right hand sides. In principle, the solution of a linear differential operator L and a right hand side f can be obtained by convolving the fundamental solution Φ with the right hand side of the differential equation over the complete domain Ω.

Let Φ be the fundamental solution of linear differential operator L, then the solution u of the differential equation

$$Lu = f$$

is given by

$$u(x) = \int_\Omega \Phi(x-y)f(y)dy.$$

We can verify that u is indeed a solution by computing

$$\begin{aligned} Lu(x) &= \int_\Omega L\Phi(x-y)f(y)dy \\ &= \int_\Omega \delta(x-y)f(y)dy \\ &= f(x) \end{aligned}$$

for all $x \in \Omega$. Where we used the linearity of differential operator L and that Φ is a fundamental solution of the equation $L\Phi = \delta$.

2.1 Poisson problem

First, we will consider the fundamental solution of the Laplace operator in the Poisson problem. The fundamental solution $u : \mathbb{R}^d \to \mathbb{R}$ of the Poisson problem vanishes at the boundary of the infinite domain Ω and satisfies

$$\begin{aligned} \Delta u &= \delta_{x_0} \quad \text{in} \quad \Omega \\ u &= 0 \qquad \text{on} \quad \partial\Omega, \end{aligned} \tag{2.1}$$

where $\Omega = \mathbb{R}^d,\ d \in \{2, 3\}$ and δ_{x_0} is the Dirac measure concentrated at the point $x_0 \in \mathbb{R}^d$. Such type of equations arise in many applications, for example in computing potential energy caused by electrical charges.

By linearity of the Laplace operator we can shift the location of the Dirac measure x_0 to the origin and later shift the solution back to simplify equations. So, we consider the following boundary value problem

$$\begin{aligned} \Delta\Phi &= \delta \quad \text{in} \quad \Omega \\ \Phi &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{2.2}$$

where $\Omega = \mathbb{R}^d,\ d \in \{2, 3\}$ and δ_0 is the Dirac measure concentrated at the origin.

Because the Poisson equation is symmetric around the Dirac point we can reduce this problem from a partial differential equation to an ordinary differential equation by considering a radial solution in $r := |x|$, where $r \in \mathbb{R}$.

First, we define a ball $B_r(0) \subset \mathbb{R}^d$ of radius r around the Dirac point at the origin, $x = 0$. By definition of the Dirac measure we have the following equality for volume integrals over ball $B_r(0)$ and surface integrals over boundary of ball $B_r(x)$, denoted by $\partial B_r(0)$

$$\begin{aligned} 1 &= \int_{B_r(0)} \delta(x)\ dV = \int_{B_r(0)} \nabla \cdot \nabla\Phi\ dV \\ &= \int_{\partial B_r(0)} \nabla\Phi \cdot \mathbf{n}\ dS \\ &= \int_{\partial B_r(0)} \frac{\partial\Phi(R)}{\partial r}\ dS, \end{aligned}$$

where we used Gauss divergence theorem to change from a volume integral to a surface integral. Because $\frac{\partial\Phi(R)}{\partial r}$ is constant on all radii $R > 0$ the integral evaluates to

$$1 = \frac{\partial\Phi(R)}{\partial r} \int_{\partial B_r(0)} dS = \frac{\partial\Phi(R)}{\partial r} A_r,$$

where A_r is the 'area' of the surface of the ball $B_r(0) \subset \mathbb{R}^d$, with,

$$A_r = \begin{cases} 2\pi r & \text{for} \quad d = 2, \\ 4\pi r^2 & \text{for} \quad d = 3. \end{cases}$$

Therefore, we obtain the following expression for $\frac{\partial\Phi(r)}{\partial r}$

$$\frac{\partial\Phi(r)}{\partial r} = \begin{cases} \dfrac{1}{2\pi r} & \text{for} \quad d = 2, \\ \dfrac{1}{4\pi r^2} & \text{for} \quad d = 3, \\ \dfrac{1}{A_r} & \text{for} \quad n > 3. \end{cases}$$

Integrating this equation yields an expression for the fundamental solution $\Phi(r)$

$$\Phi(r) = \begin{cases} \dfrac{1}{2\pi} \ln r + c_1 & \text{for} \quad d = 2, \\ -\dfrac{1}{4\pi r} + c_1 & \text{for} \quad d = 3, \end{cases}$$

where c_1 is an integration constant.

For $d = 3$ we can use the boundary condition that $\Phi(r) = 0$ as $r \to \infty$ to identify $c_1 = 0$[1]. Shifting the Dirac point back to x_0, using that r is given by $r = \|x - x_0\|_2$ with $x \in \mathbb{R}^d \backslash \{x_0\}$ and

[1] For $d = 2$ the $\ln(r)$ function will also go to infinity, so if one wants to solve a two dimensional problem one needs to use another kind of boundary condition. Because we intend to solve three dimensional we skip this case here.

changing sign[2] of the potential yields the fundamental solution $\Phi(x)$

$$\Phi(x) = \begin{cases} -\dfrac{1}{2\pi}\ln|x - x_0| & \text{for } d = 2, \\ \dfrac{1}{4\pi|x - x_0|} & \text{for } d = 3. \end{cases} \tag{2.3}$$

2.2 Stokes problem

The fundamental solution of the Stokes system satisfies the following boundary value problem, where the velocity $\mathbf{u} : \mathbb{R}^d \to \mathbb{R}^d$ vanishes at the boundary of $\Omega = \mathbb{R}^d,\ d \in \{2, 3\}$

$$\begin{aligned} -\mu\Delta\mathbf{u} + \nabla p &= \delta_{x_0}\mathbf{f} && \text{in } \Omega \\ \nabla \cdot \mathbf{u} &= \mathbf{0} && \text{in } \Omega \\ \mathbf{u} &= \mathbf{0} && \text{on } \partial\Omega, \end{aligned} \tag{2.4}$$

where $\Omega = \mathbb{R}^d,\ d \in \{2, 3\}$, μ is the viscosity of the fluid and $\delta_{x_0}\mathbf{f}$ is the Dirac force concentrated at the point $x_0 \in \mathbb{R}^d$ in direction $\mathbf{f}$.

The derivation of the exact solution of the Stokeslet follows the vector calculus approach used by Pozrikidis in [26] and Marin in [25]. Other approaches using e.g. Fourier transform can be found in [19] and are summarized in [23].

First we introduce the fundamental solution $\mathbf{G}$ of the Laplace operator as we derived in the previous section. This solution satisfies

$$\Delta\mathbf{G} = \delta_{x_0}. \tag{2.5}$$

Further, we introduce the function $\mathbf{H}$ as

$$\Delta\mathbf{H} = \mathbf{G}. \tag{2.6}$$

Hence, $\mathbf{H}$ is the solution of the biharmonic equation

$$\Delta^2\mathbf{H} = \delta_{x_0}.$$

Computing the divergence of the momentum equation gives

$$\begin{aligned} \nabla \cdot (-\mu\Delta\mathbf{u} + \nabla p) &= \nabla \cdot (\delta_{x_0}\mathbf{f}) \\ -\mu\Delta(\nabla \cdot \mathbf{u}) + \nabla \cdot \nabla p &= \nabla \cdot (\delta_{x_0}\mathbf{f}) \\ \Delta p &= \mathbf{f}\nabla\delta_{x_0}, \end{aligned} \tag{2.7}$$

where we used that the Laplace and the divergence operator commute, the velocity $\mathbf{u}$ satisfies the continuity equation $\nabla \cdot \mathbf{u} = 0$, and the identity

$$\nabla \cdot (\mathbf{a}b) = b\nabla \cdot \mathbf{a} + \mathbf{a}\nabla b$$

to rewrite the right hand side.
Using the definition of $\mathbf{G}$, $\delta_{x_0} = \Delta\mathbf{G}$, in Equation 2.7, gives the following expression for the pressure

$$p = \mathbf{f} \cdot \nabla\mathbf{G}. \tag{2.8}$$

Substitution of this expression for p in the momentum equation of 2.4 gives:

$$\begin{aligned} -\mu\Delta\mathbf{u} + \nabla\left(\mathbf{f} \cdot \nabla\mathbf{G}\right) &= \delta_{x_0}\mathbf{f} \\ \mu\Delta\mathbf{u} &= \mathbf{f}\left(\nabla\nabla\mathbf{G} - \delta_{x_0}\right) \\ \mu\Delta\mathbf{u} &= \mathbf{f}\left(\nabla\nabla - \mathbf{I}\Delta\right)\Delta\mathbf{H}, \end{aligned}$$

[2] Because this problem is related to the potential around an electron it is convenient to see the potential as a negative potential.

where we used that $\delta_{x_0} = \Delta^2\mathbf{H}$ and $\mathbf{G} = \Delta\mathbf{H}$. So, we end up with an expression of $\mathbf{u}$ in terms of $\mathbf{H}$ as

$$\mathbf{u} = \frac{1}{\mu}\mathbf{f}\left(\nabla\nabla - \mathbf{I}\Delta\right)\mathbf{H}. \tag{2.9}$$

To find an explicit expression for $\mathbf{u}$ we need an explicit expression for $\mathbf{H}$. We use the definition of $\mathbf{G}$ as the fundamental solution of the Laplace operator as given in Equation 2.5 and the expression for the fundamental solution for the Laplace operator as given in Equation 2.3 to rewrite the definition of $\mathbf{H}$ from Equation 2.6 to

$$\Delta\mathbf{H} = \mathbf{G} = \begin{cases} -\dfrac{1}{2\pi}\ln|x - x_0| & \text{for } \ d = 2, \\ \dfrac{1}{4\pi|x - x_0|} & \text{for } \ d = 3. \end{cases}$$

Because we want to solve the Stokeslet in three dimensions, we will continue with the case $d = 3$. A derivation for the two dimensional Stokeslet can be found in [26]. To simplify notation, we will define radius $r := |x - x_0|$, where $r \in \mathbb{R}$.

Because the Stokes equations are symmetric around the Dirac force we can reduce this problem from a system of partial differential equations to a system of ordinary differential equations by considering a radial solution in r. Neglecting the angular terms the Laplace operator in polar / spherical coordinates yields the following form in $\mathbb{R}^d,\ d \in \{2, 3\}$

$$\frac{\mathrm{d}^2H}{\mathrm{d}r^2}(r) - \frac{d-1}{r}\frac{\mathrm{d}H}{\mathrm{d}r}(r) = \begin{cases} -\dfrac{1}{2\pi}\ln r & \text{for } \ d = 2, \\ \dfrac{1}{4\pi r} & \text{for } \ d = 3. \end{cases}$$

Restricting this to $d = 3$ we get the following ordinary differential equation

$$H''(r) - \frac{2}{r}H'(r) = \frac{1}{4\pi r},$$

rewriting this equation gives

$$\frac{1}{r^2}\left(r^2H'(r)\right)' = \frac{1}{4\pi r}.$$

Rearranging terms and integrating this expression yields

$$H'(r) = \frac{1}{8\pi} + c_1.$$

Setting $c_1 = 0$, solve for $H(r)$ where we require $H(0) = 0$ gives

$$H(r) = \frac{r}{8\pi}.$$

Using this result for $H(r)$ in Equation 2.9 we get

$$\mathbf{u} = \frac{1}{8\pi\mu}\mathbf{f}\left(\nabla\nabla - \mathbf{I}\Delta\right)r,$$

which give an explicit expression for the velocity $\mathbf{u}$ and the Dirac force at the origin

$$\mathbf{u}(\mathbf{x}) = \frac{1}{8\pi\mu}\mathbf{f}\left(\frac{\delta_{ij}}{\|\mathbf{x}\|} + \frac{x_ix_j}{\|\mathbf{x}\|^3}\right), \tag{2.10}$$

where δ_{ij} is the Kronecker delta. Using linearity to move the Dirac force back to $\mathbf{x}_0$ gives the following fundamental solution for the Stokes system

$$\begin{aligned} \mathbf{u}(\mathbf{x} - \mathbf{x_0}) &= \mathbf{F}(\mathbf{x} - \mathbf{x_0})\mathbf{f} \\ \mathbf{p}(\mathbf{x} - \mathbf{x_0}) &= \mathbf{P}(\mathbf{x} - \mathbf{x_0})\mathbf{f}, \end{aligned} \tag{2.11}$$

where $\mathbf{F}(\mathbf{r})$ is the Stokeslet or Oseen tensor

$$\mathbf{F}(\mathbf{x}-\mathbf{x_0}) = \mathbf{F}(\mathbf{r}) = \frac{1}{8\pi\mu}\left(\frac{\mathbf{I}}{\|\mathbf{r}\|} + \frac{\mathbf{r}\otimes\mathbf{r}}{\|\mathbf{r}\|^3}\right),$$

and $\mathbf{P}(\mathbf{r})$ is given by

$$\mathbf{P}(\mathbf{x}-\mathbf{x_0}) = \mathbf{P}(\mathbf{r}) = \frac{\mathbf{r}}{4\pi\|\mathbf{r}\|^3}.$$

Chapter 3

Finite Elements

In this chapter we will discuss the discretization method that is used to discretize the partial differential equations solved in this thesis. For the discretization we will use the finite element method. The finite element method has some flexibility advantages over other discretization methods like the finite difference method and the finite volume method. The finite difference and finite volume methods can naturally handle *regular* shaped volumes, such as cubes. The finite element method gives a lot more flexibility to handle *irregular* shaped domains Ω. Therefore, it is often used in scientific and engineering applications. Further, the formulation of the finite element method using different function spaces give a rigid basis for functional analysis of this method.

3.1 Strong formulation

To explain the ideas of the finite element method we use the following model problems for the Poisson and the Stokes system.

3.1.1 Poisson problem

For the Poisson problem we use the following boundary value problem on a simulation domain $\Omega \in \mathbb{R}^d$, where $d \in \{2, 3\}$

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \\ u &= g \quad \text{on } \partial\Omega_D \\ \frac{\partial u}{\partial \mathbf{n}} &= h \quad \text{on } \partial\Omega_N. \end{aligned} \tag{3.1}$$

Where $u : \Omega \to \mathbb{R}$ is the potential, and $f : \Omega \to \mathbb{R}$ describes the sources. The boundary of the domain Ω is described by $\partial\Omega$[1] where we have an outward pointing normal vector $\mathbf{n}$. The boundary is separated into two parts $\partial\Omega_D \neq \emptyset$ and optionally $\partial\Omega_N$. On $\partial\Omega_D$ we impose a Dirichlet boundary condition described by $g : \partial\Omega_D \to \mathbb{R}$ and on $\partial\Omega_N$ we impose a Neumann boundary condition described by $h : \partial\Omega_N \to \mathbb{R}$.

The derivation of the weak form for the Poisson problem is similar to that of the Stokes problem. Therefore, we will derive only the weak formulation for the Stokes problem and in the end we will restrict the results also to the Poisson problem.

[1] For the infinitesimal in the integrals we will use the symbol Γ for the boundary of Ω.

3.1.2 Stokes problem

For the Stokes problem we use the following boundary value problem on a simulation domain $\Omega \in \mathbb{R}^d$, where $d \in \{2, 3\}$

$$\begin{aligned} -\Delta \mathbf{u} + \nabla p &= \mathbf{f} \quad \text{in } \Omega \\ \nabla \cdot \mathbf{u} &= 0 \quad \text{in } \Omega \\ \mathbf{u} &= \mathbf{g} \quad \text{on } \partial\Omega_D \\ \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - \mathbf{n}p &= \mathbf{h} \quad \text{on } \partial\Omega_N. \end{aligned} \tag{3.2}$$

Where $\mathbf{u} : \Omega \to \mathbb{R}^d$ is the velocity, $p : \Omega \to \mathbb{R}$ the pressure and $\mathbf{f} : \Omega \to \mathbb{R}^d$ the applied body forces.

Further, the separation of the boundary $\partial\Omega$ into a non-empty Dirichlet boundary part $\partial\Omega_D$ and an optional Neumann boundary part $\partial\Omega_N$ is similar to the setup of the Poisson problem.

3.2 Weak formulation

In order to derive the finite element method [7, 18] we want to relax the requirements on the solution functions for velocity $\mathbf{u}$ and pressure p stated by the strong formulation of Equation 3.2 to the weak formulation. Therefore, we introduce the finite element spaces $\mathcal{U}$ and $\mathcal{V}$ for the velocity

$$\begin{aligned} \mathcal{U} &= \left\{ \mathbf{u} \mid \mathbf{u} \in W^{1,2}(\Omega)^d,\ \mathbf{u}|_{\partial\Omega_D} = \mathbf{g} \right\} \\ \mathcal{V} &= \left\{ \mathbf{v} \mid \mathbf{v} \in W^{1,2}(\Omega)^d,\ \mathbf{v}|_{\partial\Omega_D} = 0 \right\}. \end{aligned} \tag{3.3}$$

In the derivation of the weak formulation we will need the requirement $\mathbf{v} \in \mathcal{V} : \mathbf{v}|_{\partial\Omega_D} = 0$ for the test functions space $\mathcal{V}$. This requirement is related to the Dirichlet boundary condition in the boundary value problem of Equation 3.2.

Further, we define the finite element spaces $\mathcal{P}$ and $\mathcal{Q}$ for the pressure

$$\begin{aligned} \mathcal{P} = \mathcal{Q} &= L^2(\Omega) && \text{for } \ \partial\Omega_N \neq \emptyset \\ \mathcal{P} = \mathcal{Q} &= \left\{ p \mid p \in L^2(\Omega),\ \int_\Omega p \, d\Omega = 0 \right\} && \text{for } \ \partial\Omega_N = \emptyset \end{aligned} \tag{3.4}$$

Note that if we impose only Dirichlet boundary conditions in the boundary value problem of Equation 3.2 the pressure is determined up to a constant [7], to get a unique pressure we impose the additional requirement on functions p:

$$\int_\Omega p \, d\Omega = 0.$$

3.2.1 Momentum equation

First, we consider the momentum equation of the Stokes system

$$-\Delta \mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega.$$

Multiplication of this equation with an arbitrary test function $\mathbf{v} \in \mathcal{V}$ and integration over the domain Ω yields

$$-\int_\Omega \mathbf{v} \cdot \Delta \mathbf{u} \, d\Omega + \int_\Omega \mathbf{v} \cdot \nabla p \, d\Omega = \int_\Omega \mathbf{v} \cdot \mathbf{f} \, d\Omega \qquad \forall \mathbf{v} \in \mathcal{V}. \tag{3.5}$$

Considering the first term, $-\int_\Omega \mathbf{v}\cdot\Delta\mathbf{u}\ d\Omega$, we can use partial integration and apply Gauss's divergence theorem to obtain

$$\begin{aligned}-\int_\Omega \mathbf{v}\cdot\Delta\mathbf{u}\ d\Omega &= -\int_\Omega \nabla\cdot(\nabla\mathbf{u}\cdot\mathbf{v})\ d\Omega + \int_\Omega \nabla\mathbf{u}:\nabla\mathbf{v}\ d\Omega\\ &= -\int_{\partial\Omega}\frac{\partial\mathbf{u}}{\partial\mathbf{n}}\cdot\mathbf{v}\ d\Gamma + \int_\Omega \nabla\mathbf{u}:\nabla\mathbf{v}\ d\Omega\\ &= -\int_{\partial\Omega_N}\frac{\partial\mathbf{u}}{\partial\mathbf{n}}\cdot\mathbf{v}\ d\Gamma + \int_\Omega \nabla\mathbf{u}:\nabla\mathbf{v}\ d\Omega,\end{aligned}$$

where $\nabla\mathbf{u}:\nabla\mathbf{v}=\mathbf{z}$ with components $z_i=\nabla u_i\cdot\nabla v_i,\ i=1,2,\ldots,d$ is the componentwise scalar product. Further, we used for the integral over the Dirichlet boundary the identity

$$\int_{\partial\Omega_D}\frac{\partial\mathbf{u}}{\partial\mathbf{n}}\cdot\mathbf{v}\ d\Gamma = 0,$$

which we imposed by the requirement $\mathbf{v}|_{\partial\Omega_D}=0$ on the test functions $\mathbf{v}\in\mathcal{V}$.

Considering the second term, $\int_\Omega \mathbf{v}\cdot\nabla p\ d\Omega$, we can use partial integration and the property of the test space $\mathcal{V}$ that $\mathbf{v}|_{\partial\Omega_D}=0$ to get

$$\begin{aligned}\int_\Omega \mathbf{v}\cdot\nabla p\ d\Omega &= \int_\Omega \nabla\cdot(p\mathbf{v})\ \ d\Omega - \int_\Omega p\nabla\cdot\mathbf{v}\ d\Omega\\ &= \int_{\partial\Omega_N} p\mathbf{n}\cdot\mathbf{v}\ d\Gamma - \int_\Omega p\nabla\cdot\mathbf{v}\ d\Omega.\end{aligned}$$

Substitution of these two terms in Equation 3.5, results in the first part of the weak formulation for the Stokes system

$$\int_\Omega \nabla\mathbf{u}:\nabla\mathbf{v}\ d\Omega - \int_\Omega p\nabla\cdot\mathbf{v}\ d\Omega = \int_{\partial\Omega_N}\left(\frac{\partial\mathbf{u}}{\partial\mathbf{n}}-p\mathbf{n}\right)\cdot\mathbf{v}\ d\Gamma + \int_\Omega \mathbf{v}\cdot\mathbf{f}\ d\Omega \qquad \forall\mathbf{v}\in\mathcal{V}. \tag{3.6}$$

3.2.2 Continuity equation

Multiplication the continuity equation with an arbitrary test function $q\in\mathcal{Q}$ and integration over Ω yields the second part of the weak formulation

$$\int_\Omega q\nabla\cdot\mathbf{u}\ d\Omega = 0 \qquad \forall q\in\mathcal{Q}. \tag{3.7}$$

3.2.3 Collecting weak formulation

Summarizing, our model problem for the Stokes system in strong form of Equation 3.2 can be written in *weak form* as

$$\begin{aligned}&\text{Find}\ \ (\mathbf{u},p)\in\ \mathcal{U}\times\mathcal{P}\ \ \text{such that}\\ &\qquad \mathbf{a}(\mathbf{u},\mathbf{v}) - b(p,\mathbf{v}) = (\mathbf{f},\mathbf{v}) \qquad \forall\ \mathbf{v}\in\mathcal{V}\\ &\qquad\qquad b(q,\mathbf{u}) = 0 \qquad\qquad \forall\ q\in\mathcal{Q},\end{aligned} \tag{3.8}$$

where the spaces $\mathcal{U},\mathcal{V},\mathcal{P}$ and $\mathcal{Q}$ are defined in Equations 3.3 and 3.4. The bilinear forms $\mathbf{a}(\mathbf{u},\mathbf{v})$, $b(p,\mathbf{v})$ and the linear form $(\mathbf{f},\mathbf{v})$ are given by

$$\begin{aligned}\mathbf{a}(\mathbf{u},\mathbf{v}) &= \int_\Omega \nabla\mathbf{u}:\nabla\mathbf{v}\ d\Omega,\\ b(p,\mathbf{v}) &= \int_\Omega p\nabla\cdot\mathbf{v}\ d\Omega,\\ (\mathbf{f},\mathbf{v}) &= \int_{\partial\Omega_N}\left(\frac{\partial\mathbf{u}}{\partial\mathbf{n}}-p\mathbf{n}\right)\cdot\mathbf{v}\ d\Gamma + \int_\Omega \mathbf{v}\cdot\mathbf{f}\ d\Omega.\end{aligned} \tag{3.9}$$

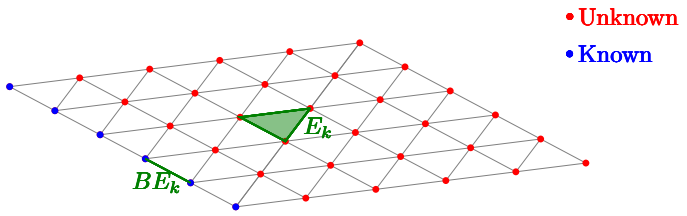


Figure 3.1: Example of a finite element mesh in 2D. On the left boundary we impose Dirichlet boundary conditions, so the value is known at these vertices. On the other boundaries we impose Neumann boundary conditions, so there we have unknowns. Further an example of an element E_k and a boundary element BE_k for a non-homogeneous Dirichlet boundary condition are shown.

3.2.3.1 Poisson problem

Using a similar derivation we can get find that the weak formulation of the Poisson problem of Equation 3.1 is given by

$$\begin{aligned} &\text{Find} \quad u \in \mathcal{U} \quad \text{such that} \\ &\qquad a(u,v) = (f,v) \qquad\qquad \forall v \in \mathcal{V}, \end{aligned} \tag{3.10}$$

where the space $\mathcal{U}$ and $\mathcal{V}$ are given by

$$\begin{aligned} \mathcal{U} &= \left\{ u \mid u \in W^{1,2}(\Omega),\ u|_{\partial\Omega_D} = g \right\} \\ \mathcal{V} &= \left\{ v \mid v \in W^{1,2}(\Omega),\ v|_{\partial\Omega_D} = 0 \right\} \end{aligned}$$

and $a(u,v)$ and (f,v) are given by

$$\begin{aligned} a(u,v) &= \int_\Omega \nabla u \cdot \nabla v \; d\Omega \\ (f,v) &= \int_{\partial\Omega_N} v \; g \; d\Gamma + \int_\Omega v \; f \; d\Omega. \end{aligned}$$

3.3 Discrete weak formulation

Using a Galerkin approximation [7] we can discretize the infinite dimensional spaces $\mathcal{U}, \mathcal{V}, \mathcal{P}$ and $\mathcal{Q}$ such that we end up in a discrete weak form.

3.3.1 Discretization of the velocity

First, we will focus on the discretization of the velocity related spaces $\mathcal{U}$ and $\mathcal{V}$. Therefore, we divide the domain $\Omega \cup \partial\Omega_N$ into n_e^u *elements* $E_k, k = 1, 2, \ldots, n_e^u$ with n_v^u vertices. Figure 3.1 shows the idea of domain discretization using triangular elements in two dimensions.

To satisfy the possibly non-homogeneous Dirichlet boundary conditions for the velocity we divide the Dirichlet boundary $\partial\Omega_D$ into n_{be} *boundary elements* $BE_k = E_{n_e^u+k}, k = 1, 2, \ldots n_{be}$ with n_{bv} vertices.

These $n_e^u + n_{be}$ elements are open, polygonal sub regions in the domain Ω in such a way that the elements $E_k, k = 1, 2, \ldots n_e^u + n_{be}$ form a partition $\mathcal{T}_k$ of the domain Ω

$$\begin{aligned} \overline{\Omega} &= \cup_{k=1}^{n_e^u + n_{be}} \overline{E_k} \\ E_i \cap E_j &= \emptyset \quad \text{for } i \neq j. \end{aligned}$$

On the vertices in this finite element mesh we define *basis functions*[2] $\varphi_i : \mathbb{R}^d \to \mathbb{R}^d$, where $d \in \{2, 3\}$

[2] These functions are also called *trial functions* or *shape functions*.

for the discrete finite element spaces:

$$\mathcal{U}^h \subset \mathcal{U} \text{ with } \mathcal{U}^h = \text{span}\{\varphi_i \mid i = 1, 2, \ldots, n_v^u + n_{bv}\}$$
$$\mathcal{V}^h \subset \mathcal{V} \text{ with } \mathcal{V}^h = \text{span}\{\varphi_i \mid i = 1, 2, \ldots, n_v^u\}.$$

We approximate velocity solution $\mathbf{u}(x)$ and test function $\mathbf{v}(x)$ on the vertices with $\mathbf{u}^h(x) \in \mathcal{U}^h$ and $\mathbf{v}^h(x) \in \mathcal{V}^h$, respectively.

On the Dirichlet boundary part $\partial\Omega_D$ the solution for the velocity $\mathbf{u}(x)$ is known. So, we can specify the approximation of the velocity $\mathbf{u}^h(x)$ such that it satisfies the Dirichlet boundary condition. Summarizing, the approximations $\mathbf{u}^h(x)$ and $\mathbf{v}^h(x)$ are given by

$$\begin{aligned} \mathbf{u}^h(x) &= \sum_{i=1}^{n_v^u} u_i^h \varphi_i + \sum_{i=n_v^u+1}^{n_v^u+n_{bv}} \mathbf{g}(x_i)\varphi_i \\ \mathbf{v}^h(x) &= \sum_{j=1}^{n_v^u} v_j^h \varphi_j, \end{aligned} \tag{3.11}$$

where the parameter h is related to the mesh size.

3.3.2 Discretization of the pressure

After discretization of the velocity related spaces we need to discretize the pressure related space $\mathcal{P}$ and $\mathcal{Q}$. Therefore, we divide the domain including the boundary $\Omega \cup \partial\Omega$ into n_e^p *elements* $E_k, k = 1, 2, \ldots, n_e^p$ with n_v^p vertices.

Also these n_e^p elements are open, polygonal subregions in the domain Ω in such a way that the elements $E_k, k = 1, 2, \ldots n_e^p$ form a partition $\mathcal{T}_k$ of the domain Ω

$$\overline{\Omega} = \cup_{k=1}^{n_e^p} \overline{E_k}$$
$$E_i \cap E_j = \emptyset \quad \text{for } i \neq j.$$

On the vertices in this finite element mesh we define basis functions $\psi_i : \mathbb{R}^d \to \mathbb{R}$, where $d \in \{2, 3\}$ for the discrete finite element spaces

$$\mathcal{P}^h \subset \mathcal{P} \text{ with } \mathcal{P}^h = \text{span}\{\psi_i \mid i = 1, 2, \ldots, n_v^p\}$$
$$\mathcal{Q}^h \subset \mathcal{Q} \text{ with } \mathcal{Q}^h = \text{span}\{\psi_i \mid i = 1, 2, \ldots, n_v^p\}.$$

We approximate pressure solution $p(x)$ and test function $q(x)$ on the vertices with $p^h(x) \in \mathcal{P}^h$ and $q^h(x) \in \mathcal{Q}^h$, respectively as

$$\begin{aligned} p^h(x) &= \sum_{i=1}^{n_v^p} p_i^h \psi_i \\ q^h(x) &= \sum_{j=1}^{n_v^p} q_j^h \psi_j, \end{aligned} \tag{3.12}$$

where the parameter h is again related to the mesh size.

3.3.3 Discrete formulation

Now we can define the *discrete weak form* of the continuous weak form of Equation 3.8 as

$$\begin{aligned} &\text{Find } (\mathbf{u}^h, p^h) \in \mathcal{U}^h \times \mathcal{P}^h \text{ such that} \\ &\mathbf{a}(\mathbf{u}^h, \mathbf{v}^h) - b(p^h, \mathbf{v}^h) = (\mathbf{f}^h, \mathbf{v}^h) \quad \forall \mathbf{v}^h \in \mathcal{V}^h \\ &b(q^h, \mathbf{u}^h) = 0 \qquad\qquad \forall q^h \in \mathcal{Q}^h, \end{aligned}$$

with $\mathbf{a}(\mathbf{u},\mathbf{v})$, $b(p,\mathbf{v})$ and $(\mathbf{f},\mathbf{v})$ defined as stated in Equation 3.9. Inserting the representations of $\mathbf{u}^h, \mathbf{v}^h, p^h$ and q^h in integral equations of Equations 3.6 and 3.7 gives the following expression for all $\mathbf{v}^h \in \mathcal{V}^h$

$$\sum_{j=1}^{n_v^u} v_j^h \left\{ \begin{array}{rl} \displaystyle\sum_{i=1}^{n_v^u} u_i^h \int_\Omega \nabla\varphi_i : \nabla\varphi_j d\Omega \; + & \displaystyle\sum_{i=n_v^u+1}^{n_v^u+n_{bv}} \mathbf{g}(x_i) \int_\Omega \nabla\varphi_i : \nabla\varphi_j d\Omega \; - \; \sum_{i=1}^{n_v^p} p_i^h \int_\Omega \psi_i \nabla\cdot\varphi_j d\Omega \\ - \; \displaystyle\sum_{i=1}^{n_v^u} \int_{\partial\Omega_N} \mathbf{h}(x_i)\varphi_j d\Gamma \; - & \displaystyle\sum_{i=1}^{n_v^u} \int_\Omega \mathbf{f}(x_i)\varphi_j d\Omega \end{array} \right\} \; = \; 0$$

$$\sum_{j=1}^{n_v^p} q_j^h \left\{ \sum_{i=1}^{n_v^u} u_i^h \int_\Omega \psi_j \nabla\cdot\varphi_i d\Omega \right\} \; = \; 0.$$

Since this expression needs to be valid for all n_v^u coefficients v_j^h and n_v^p coefficients q_j^h, all terms of the sums over j need to be zero and we can split this expression into $n_v^u + n_v^p$ equations

$$\begin{aligned} \underbrace{\sum_{i=1}^{n_v^u} u_i^h}_{u_i} \underbrace{\int_\Omega \nabla\varphi_i : \nabla\varphi_j d\Omega}_{\mathbf{a}_{ij}} \; - \; \underbrace{\sum_{i=1}^{n_v^p} p_i^h}_{p_i} \underbrace{\int_\Omega \psi_i \nabla\cdot\varphi_j d\Omega}_{-\mathbf{b}_{ij}^\top} \; &= \; f_j \\ f_j \; = \; \int_\Omega f\varphi_j d\Omega + \int_{\partial\Omega_N} h\varphi_j d\Omega - \sum_{i=n_v^u+1}^{n_v^u+n_{bv}} g(x_i) \int_\Omega \nabla\varphi_i : \nabla\varphi_j d\Omega & \qquad (3.13) \\ \underbrace{\sum_{i=1}^{n_v^u} u_i^h}_{u_i} \underbrace{\int_\Omega \psi_k \nabla\cdot\varphi_i d\Omega}_{\mathbf{b}_{ik}} \; &= \; 0 \end{aligned}$$

for $j = 1, 2, \ldots, n_v^u$ and $k = 1, 2, \ldots, n_v^p$.

With the constants a_{ij}, b_{ij}, u_i, p_i and f_j indicated in Equation 3.13, we can build up the matrices A and B, and the vectors $\mathbf{u}^h, \mathbf{p}^h$ and $\mathbf{f}$. Using these matrices and vectors we can write Equation 3.13 in the following algebraic system:

$$\begin{bmatrix} A & B^\top \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^h \\ \mathbf{p}^h \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} \qquad (3.14)$$

3.4 Solvability

At this point we need to make a note about the solvability of the linear system of Equation 3.14. A detailed discussion of the solvability of the linear system can be found in [7]. To discuss the uniqueness of this system we consider the homogeneous system

$$\begin{aligned} A\mathbf{u}^h + B^\top \mathbf{p}^h &= \mathbf{0} \\ B\mathbf{u}^h &= \mathbf{0}. \end{aligned} \qquad (3.15)$$

Premultiplying the momentum equation with ${\mathbf{u}^h}^\top$ and the continuity equation by ${\mathbf{p}^h}^\top$ gives

$$\begin{aligned} {\mathbf{u}^h}^\top A\mathbf{u}^h + {\mathbf{u}^h}^\top B^\top \mathbf{p}^h &= \mathbf{0} \\ (B^\top \mathbf{p}^h)^\top \mathbf{u}^h &= \mathbf{0}. \end{aligned}$$

Substitution of the second equation in the first one gives

$${\mathbf{u}^h}^\top A\mathbf{u}^h = \mathbf{0}.$$

Using that matrix A is the discretization of the Laplacian, hence A is positive definite, it follows that

$$\mathbf{u}^h = \mathbf{0}.$$

So, the discrete Stokes system is uniquely solvable for the velocity vector $\mathbf{u}^h$. Using $\mathbf{u}^h = \mathbf{0}$ in the momentum equation we have

$$B^\top \mathbf{p}^h = \mathbf{0},$$

thus the pressure vector is unique up to the null space of $B^\top$. Using Dirichlet boundary conditions the pressure vector $\mathbf{p}^h$ is defined up to a constant, so the discretization should reflect this as

$$\ker B^\top = \{\mathbf{1}\}.$$

To satisfy the weak continuity equation the discrete pressure $p^h \in \mathcal{P}^h$ has to satisfy the condition

$$\int_\Omega p^h \nabla \cdot \mathbf{v}^h d\Omega = \mathbf{0} \quad \Rightarrow \quad p^h = \text{constant} \qquad \forall\ \mathbf{v}^h \in \mathcal{V}^h,$$

which is formulated as the discrete analogue of the *inf-sup condition* [4, 7]: There exists a positive constant γ independent of h such that

$$\min_{q^h \neq \text{constant}} \max_{\mathbf{v}^h \neq \mathbf{0}} \frac{|(q^h, \nabla \cdot \mathbf{v}^h)|}{\|\mathbf{v}^h\|_{1,\Omega} \|q^h\|_{0,\Omega}} \geq \gamma > 0 \tag{3.16}$$

for all $q^h \in \mathcal{Q}^h$ and $\mathbf{v}^h \in \mathcal{V}^h$. The inf-sup condition itself is given by

$$\inf_{q \neq \text{constant}} \sup_{\mathbf{v} \neq \mathbf{0}} \frac{|(q, \nabla \cdot \mathbf{v})|}{\|\mathbf{v}\|_{1,\Omega} \|q\|_{0,\Omega}} \geq \gamma > 0 \tag{3.17}$$

for all $q \in \mathcal{Q}$ and $\mathbf{v} \in \mathcal{V}$.

The approximation of the both the velocity and pressure spaces by linear elements ($\mathbb{P}_1/\mathbb{P}_1$ elements) is unstable [4, 7]. In general, for stable elements the velocity space should be bigger of higher order than the pressure space. For example, the Taylor-Hood elements ($\mathbb{P}_2/\mathbb{P}_1$ elements) which approximates the velocity $\mathbf{u}$ with quadratic functions and the pressure p with linear functions, is stable [4, 7].

Chapter 4

Hierarchical Hybrid Grids

In this chapter we will introduce *Hierarchical Hybrid Grids* [2, 10] (HHG) as the framework that is used to solve the two model problems we consider in this thesis. HHG is a massively parallel *multigrid solver* using finite elements.

4.1 HHG principles

The solver is based on an unstructured input mesh where macro elements (vertices, edges, faces and volumes) are defined. The unstructured input mesh with macro elements is regularly refined to finer levels to get a hierarchy of meshes on different refinement levels for the multigrid solver. So, the hierarchy of meshes in constructed from coarse to fine instead of a standard multigrid where a fine mesh is coarsened to coarser meshes. The advantage of this strategy is that we have globally an unstructured mesh but inside a macro element we have a structured mesh. This allows for a straight forward implementation of geometric multigrid on the macro elements [2].

Further, HHG employs the *Message Passing Interface* (MPI) parallel software package for distributed memory parallelization [1]. The macro elements are distributed over the processors and the efficient implementation of HHG data structures result in good scalability results [1, 10, 11].

4.1.1 Tetrahedra in unit cube

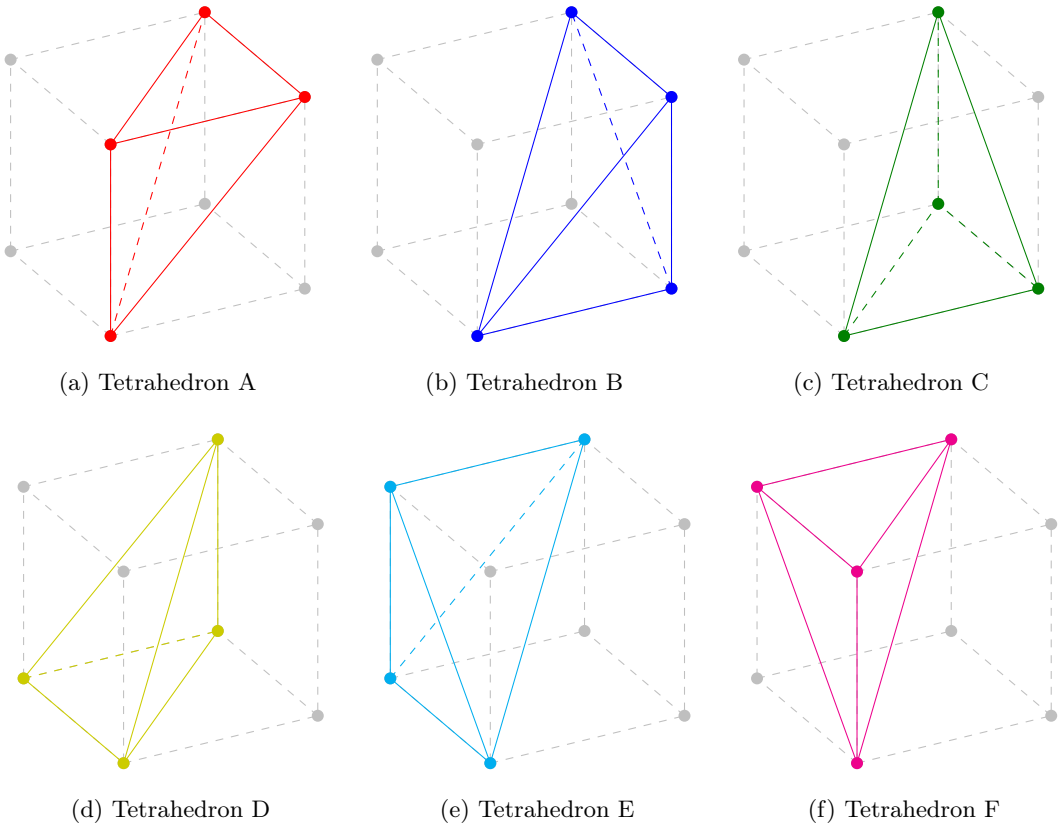
All our experiments introduced in Chapters 5, 6 and 7 will solve the Poisson or Stokes model problem on a unit cube. Also the scalability experiments introduced in Chapter 8 solve a Stokes problem in cubic domain. The HHG framework supports tetrahedral elements, so we need a triangularization from a cube to tetrahedra.

For all experiments we will use the Kuhn triangularization of the unit cube [21]. This triangularization splits the unit cube into six tetrahedra of two different types. The second type of tetrahedra is the inside-out version of the first type of tetrahedra. The six tetrahedra from this triangularization are shown in Figure 4.1. The tetrahedra A, C, E are the same up to rotation, which is also true for the tetrahedra B, D, F.

4.1.2 Tetrahedral elements

For the macro elements HHG supports tetrahedra and uses Bey's refinement method [3] to refine a macro element in a structured mesh. So, for a refinement of a tetrahedron all midpoints of all edges are connected as new edges on the finer mesh. This splits the coarse tetrahedron into eight equally sized new fine tetrahedra, as shown in Figure 4.2.

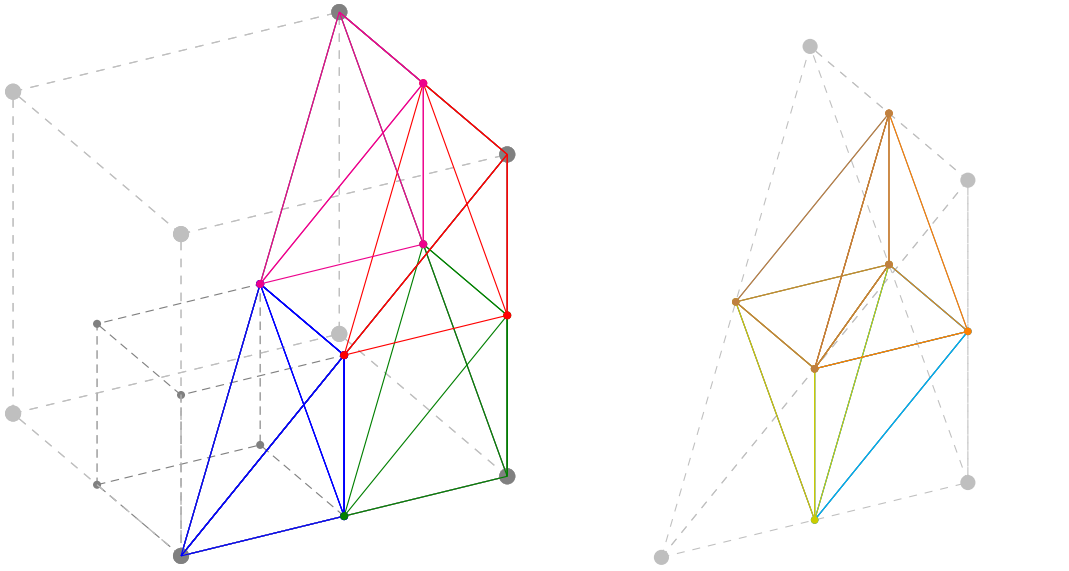
Bey showed that applying Bey's refinement method on the six tetrahedra from the Kuhn triangularization of the unit cube is equivalent to Kuhn triangularization of the eight sub cubes obtained by a regular refinement of the unit cube [3].



(a) Tetrahedron A (b) Tetrahedron B (c) Tetrahedron C

(d) Tetrahedron D (e) Tetrahedron E (f) Tetrahedron F

Figure 4.1: The six tetrahedra that buildup a cube using a Kuhn triangularization [21].



(a) Four rescaled tetrahedra in the corners of a large tetrahedron. The rescaled tetrahedron is identical to the initial tetrahedron in a refined cube.

(b) Double pyramid as left open space between four refined tetrahedra in the vertices of a large tetrahedron. This double pyramid can be split into four known tetrahedra.

Figure 4.2: The refinement of a tetrahedron using Bay's refinement method [3].

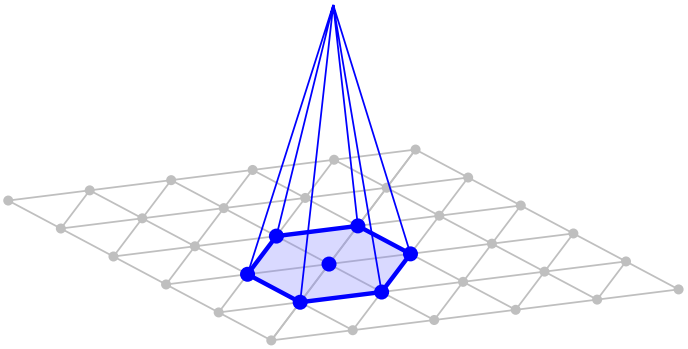


Figure 4.3: Example of a 2D basis function on a triangular 2D mesh.

4.2 Finite Element implementation

To conclude the discussion of the finite element method of Chapter 3 we will include some properties of a tetrahedral mesh as used in HHG and change the integration rule for the force terms with the definition a Dirac measure.

In the implementation of HHG we will use linear basis functions φ_i and ψ_i for the spaces $\mathcal{V}^h$ and $\mathcal{Q}^h$, respectively. For example on a 2D grid a linear basis function φ_i is shown in Figure 4.3.

4.2.1 Integration rule

To obtain the actual values for the matrices A and B and the load vector $\mathbf{f}$ in Equation 3.14 we need to integrate the expressions for the components as given in Equation 3.13. Since in general it is not possible to integrate a function analytically, in most cases a numerical method for approximating an integral is used. A *numerical quadrature rule* replaces the integral over a general function $f(x)$ by a weighted sum of function values at certain quadrature nodes x_i

$$\int_\Omega f(x)d\Omega \approx \sum_{i=1}^{N} \omega_i f(x_i). \tag{4.1}$$

In HHG a Shaidurov quadrature rule is used for tetrahedra [29, 2, 15]. This quadrature rule defines five quadrature nodes and weights as shown in Figure 4.4 and listed in Table 4.1. The barycentric coordinates are expressed in the coordinates β_i which can easily be converted to the tetrahedron reference element. This quadrature rule is exact for polynomials of degree ≤ 2.

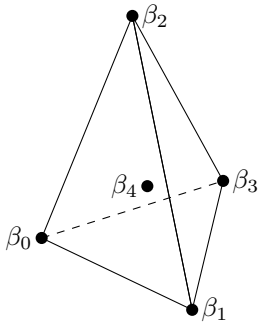


Figure 4.4: Tetrahedral quadrature rule defined by the barycentric coordinates β_i, for $i = 0, 1, 2, 3, 4$.

	barycentric coord.				weight
node	λ_0	λ_1	λ_2	λ_3	ω
β_0	1	0	0	0	$\frac{1}{120}$
β_1	0	1	0	0	$\frac{1}{120}$
β_2	0	0	1	0	$\frac{1}{120}$
β_3	0	0	0	1	$\frac{1}{120}$
β_3	0	0	0	1	$\frac{1}{120}$
β_4	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{2}{15}$

Table 4.1: Barycentric coordinates for quadrature nodes and weights for Shaidurov quadrature rule for tetrahedra.

4.2.2 Integration rule for Dirac right hand side

Because we want to solve the fundamental solution we have a right hand side function f of a special type, i.e. $f(x) = \delta_{x_0}(x)$ for a point $x_0 \in \Omega$. By the definition of a Dirac measure, we have for a general function g the following expression for integration

$$\int_\Omega \delta_{x_0}(x) g(x)\ d\Omega = g(x_0). \tag{4.2}$$

In this section we will only focus on the term in f_j in Equation 3.13 that contains the Dirac measure and for which we changed the integration rule in HHG. So we focus on the integration of the term which we denote by f_j^*

$$f_j^*(x) = \int_\Omega f(x)\varphi_j(x) d\Omega \tag{4.3}$$

where $f(x) = \delta_{x_0}(x)$. Using the observation that E_k for $k = 1, 2, \ldots, n_e$ (where $n_e := n_e^u + n_{be}$) is a partition of domain Ω, a finite element library will split this integral into a sum of integrals over all elements E_k

$$\begin{aligned} f_j^* &= \int_\Omega f(x)\varphi_j(x)\ d\Omega \\ &= \sum_{k=1}^{n_e} \int_{E_k} \delta_{x_0}(x)\varphi_j(x)\ dE_k \\ &= \sum_{k=1}^{n_e} \varphi_j(x_0). \end{aligned} \tag{4.4}$$

For basis functions φ_j and coordinates of vertex v_i we have

$$\varphi_j(v_i) = \delta_{ij},$$

where δ_{ij} denotes the Kronecker delta.

The basis function $\varphi_j(x)$ evaluates to zero on all elements that don't contain vertex v_j in their closure, so a finite element library will change the evaluation of f_j^* in Equation 4.4 to local basis functions

$$f_j^* = \sum_{k=1}^{n_e} \varphi_j(x_0) = \sum_{l=1}^{n_k} \varphi_l^{(k)}(x_0) \quad \text{for } x_0 \in E_k, \tag{4.5}$$

where $\varphi_l^{(k)}$ for $l = 1, 2, \ldots n_k$ are the n_k non zero basis functions over element E_k, and therefore called local basis functions. For example the basis function shown in Figure 4.3 will be in the set of local basis functions $\left\{\varphi_l^{(k)}\right\}_{l=1}^{n_k}$ for the 6 marked elements E_k in the figure.

For the element E_d that contains the Dirac measure we can evaluate f_j^* using the local basis functions of that element. This can be seen as an integration rule where we have at most one quadrature node in an element on a variable location coinciding with the location of the Dirac measure.

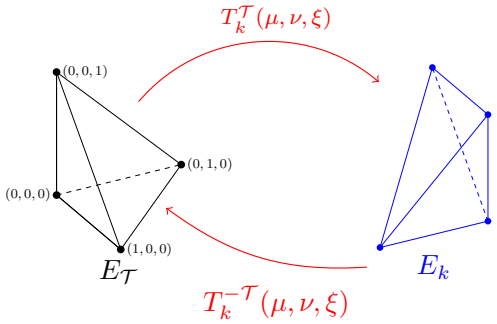


Figure 4.5: Tetrahedral reference element $E_{\mathcal{T}}$ and coordinate mapping $T_k^{\mathcal{T}}$ for a tetrahedron element E_k.

4.2.3 Implementation of Dirac integration

The parallel data structures in HHG support iterators to iterate over the macro elements and the refined elements, but it is not possible to get direct access to a specific element that is related to a certain location in physical space. For the standard computation of the load vector this is a natural approach because we need to evaluate for each fine element E_k the forcing function at the quadrature nodes.

The integration over all elements E_k will be performed on a reference element $E_{\mathcal{T}}$. Therefore, a mapping from a reference element $E_{\mathcal{T}}$ to the fine element E_k is known at time of integration [2, 15].

However, for the Dirac integration the test functions have to be evaluated at one specific location which does not need to coincide with a quadrature node. To get the elements for which we need to update the load vector to implement Equation 4.5, we need to iterate over all macro elements and its refined elements and check if the Dirac measure is located inside that fine element.

To keep track of all Dirac forces in the domain Ω, a new type of HHG's `VolumeFunction` was introduced, the `DiracVolumeFunction`.
Summarizing, the algorithm to integrate a `DiracVolumeFunction` as forcing function is given by

For each *MacroElement* M in domain Ω

For each *FineElement* E_k in *MacroElement* M
Initialize mapping $T_k^{\mathcal{T}} : E_{\mathcal{T}} \to E_k$
Initialize inverse mapping $T_k^{-\mathcal{T}} : E_k \to E_{\mathcal{T}}$

For all 4 *vertices* v_i in *FineElement* E_k
Apply mapping $T_k^{\mathcal{T}}$ on 4 vertices of *ReferenceElement* $E_{\mathcal{T}}$

For each *DiracPoint* D in domain Ω
Check location of *DiracPoint* D w.r.t. E_k in physical space
If $D \in E_k$
Apply mapping $T_k^{-\mathcal{T}}$ on Dirac point D
Evaluate basis functions at location of D in *ReferenceElement* $E_{\mathcal{T}}$.

Where in bold the time consuming changes with respect to an integration using a quadrature rule are marked. The mapping $T_k^{\mathcal{T}}$ is visualized in Figure 4.5.

The time complexity of the integration of a regular forcing function is $\mathcal{O}(m \cdot n)$, where m is the number of macro elements and n the number of fine elements inside a macro element. The time complexity of this algorithm for integration of a forcing function that consists of d Dirac measures grows with the current data structures to $\mathcal{O}(m \cdot n \cdot d)$ in stead of something $\mathcal{O}(d)$ what we would like to have for an almost everywhere zero forcing function.

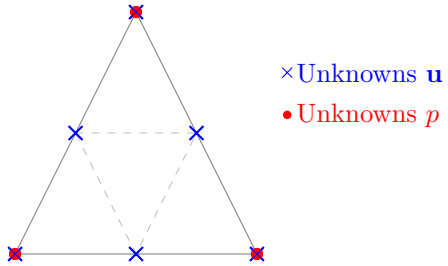


Figure 4.6: Location of **u** and p unknowns of $\mathbb{P}_1$-iso-$\mathbb{P}_2$ elements for velocity and $\mathbb{P}_1$ elements for the pressure.

Using the mapping $T_k^{\mathcal{T}}$ we can map the reference element $E_{\mathcal{T}}$ to the fine element E_k in physical space over which we are integrating, as shown in Figure 4.5. Therefore, the Dirac point and the fine element have coordinates in physical space and we can determine if a Dirac point is inside an element, where we will only support tetrahedral elements.

Checking if a (Dirac) point D is inside a tetrahedral element E_k can be done by constructing the four planes of the boundary of the tetrahedral element E_k based on three vertices. A point D will be inside a tetrahedral element E_k if for all four planes the Dirac force and the fourth coordinate of the tetrahedral element E_k are on the same side of the plane.

When a Dirac force is inside a tetrahedron we can transform its coordinates using the inverse mapping $T_k^{-\mathcal{T}}$ into the reference element $E_{\mathcal{T}}$ and evaluate the basis functions on this coordinate in the reference element.

4.3 Stabilization

As we have seen, HHG supports only linear elements. So, we need to do something to get stable reconstructions for the Stokes system and satisfy the inf-sup condition of Equation 3.16. In this section we will discuss two possible stabilization methods such that we can still use the linear elements that are supported by HHG.

4.3.1 $\mathbb{P}_1$-iso-$\mathbb{P}_2$ stabilization

The Taylor-Hood elements $\mathbb{P}_2/\mathbb{P}_1$ are stable for the Stokes problem [4, 7] and therefore they are a possibility to overcome the instability of the $\mathbb{P}_1$ elements for the velocity **u**. Quadratic functions can be approximated over a line segment by two linear functions where a third unknown is introduced between the two endpoints of the line segment. This idea is shown in two dimensions in Figure 4.6.

Because regular refinements of the domain Ω are employed, this property will also hold between refinement levels in the domain Ω. So, instead of using quadratic elements for the velocity we can approximate the quadratic elements for the velocity by linear elements for the velocity on a refinement level finer than the pressure. This strategy results in $\mathbb{P}_1$-iso-$\mathbb{P}_2$ elements for the velocity and $\mathbb{P}_1$ elements for the pressure.

In all our experiments with the Stokes problem using the HHG framework we will use the $\mathbb{P}_1$-iso-$\mathbb{P}_2$ stabilization, which has the advantage that we don't need to change equations of the Stokes system. A disadvantage of this method is that we reconstruct a solution on which we don't have for all velocity unknowns a reconstructed pressure.

4.3.2 PSPG stabilization

Another possible stabilization strategy is using the Petrov-Galerkin pressure stabilization (PSPG-stabilization) [5, 10, 17] where the continuity equation is modified. The Stokes system is given by

$$\begin{aligned} -\mu\Delta\mathbf{u} + \nabla p &= \mathbf{f} \\ \nabla \cdot \mathbf{u} &= \mathbf{0}. \end{aligned} \tag{4.6}$$

To stabilize this system, we can consider the divergence of the momentum equation

$$\begin{aligned} \nabla \cdot \mathbf{f} &= \nabla \cdot (-\mu\Delta\mathbf{u} + \nabla p) \\ &= -\mu\Delta (\nabla \cdot \mathbf{u}) + \Delta p \\ &= \Delta p, \end{aligned} \tag{4.7}$$

where we used the continuity equation $\nabla \cdot \mathbf{u}$ to end up with an expression only in terms of the pressure p. Adding a multiple τh^2 of Equation 4.7 to the continuity equation we get the modified Stokes system

$$\begin{aligned} -\mu\Delta\mathbf{u} + \nabla p &= \mathbf{f} \\ \nabla \cdot \mathbf{u} + \tau h^2 \Delta p &= \tau h^2 \nabla \cdot \mathbf{f}, \end{aligned} \tag{4.8}$$

where τ is a stabilization parameter.

The advantage of the PSPG stabilization will be that we are able to reconstruct the pressure on all nodes where we have also a velocity unknown. Disadvantage is that we need an additional integration of the divergence of a Dirac measure for the modified continuity equation.

4.4 Multigrid

To solve the linear systems obtained from the finite element discretization a multigrid solver is used in HHG [2, 10] because of its optimal complexity of $\mathcal{O}(n)$, where n is the number of unknowns in the problem [31].

Because of the structured refinement of the macro elements in HHG, we can use geometric multigrid methods. For the experiments in this thesis we will restrict to V-cycles where we do a couple of pre- and post-smoothing steps before and after a restriction or prolongation step. Details about the multigrid solver in HHG are described by Ben Bergen [2] and Björn Gmeiner [10].

4.5 Schur Complement CG solver

In Chapter 3 we derived the algebraic system

$$K \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} A & B^\top \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} \tag{4.9}$$

for the Stokes equations. In this section we will take a closer look at the linear system 4.9 and at the Schur complement method to solve this Stokes system. Using an LDU-decomposition we can split matrix K of Equation 4.9 into three matrices L, D and U [32]

$$K = \begin{bmatrix} A & B^\top \\ B & 0 \end{bmatrix} = LDU = \begin{bmatrix} I & 0 \\ BA^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & -S \end{bmatrix} \begin{bmatrix} I & A^{-1}B^\top \\ 0 & I \end{bmatrix} \tag{4.10}$$

where $S = BA^{-1}B^\top$ is the Schur complement matrix. Multiplying out DU gives

$$\begin{bmatrix} I & 0 \\ BA^{-1} & I \end{bmatrix} \begin{bmatrix} A & B^\top \\ 0 & -S \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix},$$

which results in the following system for $\mathbf{u}$ and $\mathbf{p}$

$$\begin{bmatrix} A & B^\top \\ 0 & S \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ BA^{-1}\mathbf{f} \end{bmatrix}. \tag{4.11}$$

4.5.1 Schur method

We have obtained a decoupled system for the velocity $\mathbf{u}$ and pressure $\mathbf{p}$ and can solve for both variables [32, 33].
Given a pressure $\tilde{\mathbf{p}}$, we define $\mathbf{r}$ as

$$\mathbf{r} = B\mathbf{u}^* \tag{4.12}$$

where $\mathbf{u}^*$ is the unique solution of

$$A\mathbf{u}^* = \mathbf{f} - B^\top \tilde{\mathbf{p}}. \tag{4.13}$$

Further, let $\mathbf{u}_\delta$ and $\mathbf{p}$ be the unique solutions of

$$\begin{aligned} A\mathbf{u}_\delta &= B^\top \mathbf{p} - B^\top \mathbf{p}^k = B^\top \mathbf{p}_\delta \\ S\mathbf{p} &= B\mathbf{u}_\delta, \end{aligned} \tag{4.14}$$

where $\mathbf{p}_\delta = \mathbf{p} - \mathbf{p}^k$.

This setup for $\mathbf{u}^*$ and $\mathbf{u}_\delta$ enables us to set up the following theorem [33] and show the equivalence with the Stokes system:

Theorem 4.5.1. The pair $(\mathbf{u}, \mathbf{p})$ is a solution of Equation 4.9 *if and only if*

$$S\mathbf{p} = \mathbf{r}, \tag{4.15}$$

where $\mathbf{r}$ is defined as Equation 4.12 and

$$A\mathbf{u} = \mathbf{f} - B^\top \mathbf{p}. \tag{4.16}$$

Proof. Verfurth [33, Lemma 3.1] proofs this theorem for a linear symmetric operator L and a different splitting $\mathbf{u}_f$ and $\mathbf{u}_p$. Here we will show the proof for the specific linear symmetric operator $S = BA^{-1}B^\top$ with our splitting $\mathbf{u}^*$ and $\mathbf{u}_\delta$.

Let $(\mathbf{u}, \mathbf{p})$ be given by Equations 4.15 and 4.16. Summarizing Equations 4.13, 4.14 and 4.16 we have

$$\begin{aligned} A\mathbf{u}^* &= \mathbf{f} - B^\top \tilde{\mathbf{p}} \\ A\mathbf{u}_\delta &= B^\top \mathbf{p} - B^\top \tilde{\mathbf{p}} \\ A\mathbf{u} &= \mathbf{f} - B^\top \mathbf{p}. \end{aligned} \tag{4.17}$$

Hence, the velocity $\mathbf{u}$ is given by

$$\mathbf{u} = \mathbf{u}^* - \mathbf{u}_\delta.$$

Combining this with the divergence expressions in Equations 4.12, 4.14 and requirement 4.15 gives

$$\begin{aligned} \mathbf{r} &= B\mathbf{u}^* \\ S\mathbf{p} &= B\mathbf{u}_\delta \\ S\mathbf{p} &= \mathbf{r}, \end{aligned} \tag{4.18}$$

which yields

$$B\mathbf{u} = 0.$$

Hence, $(\mathbf{u}, \mathbf{p})$ is a solution of 4.9. Uniqueness of the solution of problem 4.9 completes this proof. □

Now we can use the Schur method both as a direct method or as a iterative method [32]. The evaluation of $S\mathbf{p}$ is expensive, because $S = BA^{-1}B^\top$ involves an inversion of matrix A. Therefore we will approximate S and use the Schur method as an iterative solver [27, 32, 33] where we will use a couple of preconditioned Conjugate Gradient iterations[1] to solve

$$S\mathbf{p} = \mathbf{r}. \tag{4.19}$$

[1] A short introduction to (preconditioned) Conjugate Gradient methods can be found in Appendix A.

The pressure mass matrix M is componentwise given by

$$m_{ij} = \int_\Omega \psi_i \psi_j d\Omega \quad i,j = 1,2,\ldots,n_v^p$$

where n_v^p is the number of vertices in the discretization for the pressure and ψ_i is the pressure basis function for pressure vertex i.

Elman [7] shows the spectral equivalence between the Schur complement matrix $S = BA^{-1}B^\top$ and the pressure mass matrix M in the following theorem:

Theorem 4.5.2. For any flow problem with $\partial\Omega = \partial\Omega_D$, discretized using uniformly stable mixed approximation on a shape regular quasi-uniform subdivision of $\mathbb{R}^3$ the pressure Schur complement matrix $S = BA^{-1}B^\top$ is spectrally equivalent to the pressure mass matrix M:

$$\gamma^2 \leq \frac{(S\mathbf{q},\mathbf{q})}{(M\mathbf{q},\mathbf{q})} \leq Ch^2 \quad \forall \mathbf{q} \in \mathbb{R}^{n_v^p}. \tag{4.20}$$

The inf-sup constant γ is bounded away from zero and independent of h. The "effective" condition number satisfies

$$\kappa(S) \leq \frac{C}{c\gamma^2}$$

where c and C are the constants given by the mass matrix bounds as

$$ch^3 \leq \frac{(M\mathbf{q},\mathbf{q})}{(\mathbf{q},\mathbf{q})} \leq Ch^3.$$

Proof. The proof is given in [7, Theorem 5.22] for $\mathbb{R}^2$ and using [7, Remark 5.23] it can be extended to stable mixed approximations using tetrahedra in $\mathbb{R}^3$. □

For the preconditioned CG solver we will use the mass matrix M as diagonal preconditioner [9].

4.5.2 Schur CG iteration

To solve a general linear system $A\mathbf{x} = \mathbf{b}$ using a preconditioned CG method we need to compute $\|\mathbf{s}\|_A$ where $\mathbf{s}$ is the (preconditioned) search direction and A the system matrix. For Equation 4.19 we must compute the norm for the search direction $\mathbf{s}$ as

$$\|\mathbf{s}\|_{BA^{-1}B^\top} = \mathbf{s}^\top BA^{-1}B^\top \mathbf{s}.$$

Introducing an auxiliary vector $\mathbf{v} = A^{-1}B^\top\mathbf{s}$ simplifies the evaluation of this norm to

$$\|\mathbf{s}\|_{BA^{-1}B^\top} = \mathbf{s}^\top B\mathbf{v}.$$

This auxiliary vector $\mathbf{v}$ satisfies the linear system

$$A\mathbf{v} = B^\top\mathbf{s}, \tag{4.21}$$

which is a vector Poisson problem and can efficiently be solved using a multigrid method [33].

With the search direction $\mathbf{s}$ and distance α we can update the pressure $\mathbf{p}$ as

$$\mathbf{p}_{i+1} = \mathbf{p}_i + \alpha_i \mathbf{s}_i.$$

In terms of Equation 4.14 we have $\mathbf{p}_\delta = \alpha_i \mathbf{s}_i$, which satisfies

$$A\mathbf{u}_\delta = -B^\top \mathbf{p}_\delta.$$

Using the definition of $\mathbf{v}_i$ we can update the velocity by $\mathbf{u}_\delta$ which is given by

$$\begin{aligned} \mathbf{u}_\delta &= -A^{-1}B^\top \mathbf{p}_\delta \\ &= -\alpha_i \mathbf{v}_i, \end{aligned} \tag{4.22}$$

which yields

$$\mathbf{u}_{i+1} = \mathbf{u}_i - \alpha_i \mathbf{v}.$$

4.5.3 Final Schur CG solver

The Schur CG solver is a nested iterative solver with n_v outer iterations that solves for the velocity using a multigrid method. For each velocity $\mathbf{u}_i$ as a solution of the outer iteration a preconditioned CG solver runs n_p inner iterations to solve for the pressure $\mathbf{p}_i$. We will use double subscript indices for velocity and pressure denoting the iteration number of the outer and inner iteration, so $\mathbf{u}_{i,j}$ is the reconstructed velocity for outer iteration i and inner iteration (i.e, CG iteration) j.

We use $\mathbb{P}_1$-iso-$\mathbb{P}_2$ elements for the velocity. Therefore, we reconstruct the velocity $\mathbf{u}$ on refinement level l and pressure p on refinement level $l-1$. Using the prolongation and restriction operators $\mathcal{P}$ and $\mathcal{R}$ we can shift variables $\mathbf{u}$ and p to the correct level. In the listing of the algorithm we will drop the index that refers to the refinement level for the used variables.
Using initial guess $\mathbf{p}_0 = \mathbf{0}$, the Schur complement CG algorithm [27, 32, 33] is given by

For $i = 1, 2, \ldots, n_v$

1. Solve momentum equation
$$A\tilde{\mathbf{u}}_i = \mathbf{f} - B^\top \mathcal{P}\mathbf{p}_{i-1} \tag{4.23}$$
for $\tilde{\mathbf{u}}_i$ on level l using a Multigrid method.

2. Solve Schur complement equation
$$S\mathbf{p}_i = \mathbf{r}_0 = \mathcal{R}B\mathbf{u}_i \tag{4.24}$$
for $\mathbf{p}_i$ on level $l-1$ using the Preconditioned CG method:

$\mathbf{u}_{i,0} = \tilde{\mathbf{u}}_i$
$\mathbf{p}_{i-1,0} = \mathbf{p}_{i-1}$
$\mathbf{z}_0 = M^{-1}\mathbf{r}_0$
$\mathbf{s}_0 = \mathbf{z}_0$
`for` $j = 0, 1, 2, \ldots, n_p - 1$
  `if` $j > 0$
    $\mathbf{z}_j = M^{-1}\mathbf{r}_j$
    $\beta_j = \frac{\mathbf{r}_j^\top \mathbf{z}_j}{\mathbf{r}_{j-1}^\top \mathbf{z}_{j-1}}$
    $\mathbf{s}_j = \mathbf{z}_j + \beta_j \mathbf{s}_{j-1}$
  `end`
  Solve $A\mathbf{v}_j = B^\top \mathcal{P}\mathbf{s}_j$ for $\mathbf{v}_j$ on level l using a multigrid method
  $\alpha_j = \frac{\mathbf{r}_j^\top \mathbf{z}_j}{\mathbf{s}_j^\top \mathcal{R}B\mathbf{v}_j}$
  $\mathbf{p}_{i,j+1} = \mathbf{p}_{i,j} + \alpha_j \mathbf{s}_j$
  $\mathbf{u}_{i,j+1} = \mathbf{u}_{i,j} - \alpha_j \mathbf{v}_j$
  $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathcal{R}B\mathbf{v}_j$
  `if` $\mathbf{r}_{j+1}$ small
    DONE; algorithm converged
    $\mathbf{p}_{i,n_p} = \mathbf{p}_{i,j+1}$
    $\mathbf{u}_{i,n_p} = \mathbf{u}_{i,j+1}$
  `end`
`end`
$\mathbf{u}_i = \mathbf{u}_{i,n_p}$
$\mathbf{p}_i^{h-1} = \mathbf{p}_{i,n_p}$

In this algorithm we have the following parameters:

- n_v: number of iterations to reconstruct a new velocity $\mathbf{u}_i$.
- n_p: number of preconditioned CG iterations to apply pressure correction to make $\mathbf{u}_i$ divergence free.
- Multigrid parameters such as cycle type, number of cycles, smoothers, number of pre- and post-smoothing steps for the two systems that will be solved using a multigrid method:
  - $A\tilde{\mathbf{u}}_i = \mathbf{f} - B^\top \mathcal{P}\mathbf{p}_{i-1}$
  - $A\mathbf{v}_j = B^\top \mathcal{P}\mathbf{s}_j$

Remark 4.5.3. Because the system $S\mathbf{p}_i = \mathbf{r}$ with the lumped mass preconditioner M is well conditioned [7] we need only a couple of preconditioned CG iterations to get a good approximation for the pressure $\mathbf{p}_i$.

Chapter 5

Poisson problem

In this chapter we will numerically reconstruct the fundamental solution to the Poisson problem

$$\begin{aligned} -\Delta u &= \delta_{x_0} \quad \text{in } \Omega \\ u &= g \qquad \text{on } \partial\Omega, \end{aligned} \tag{5.1}$$

where $\Omega \subset \mathbb{R}^d,\ d \in \{2, 3\}$ is a bounded, open, convex and polyhedral domain. Further, we will state an a priori error estimate by Köppl and Wohlmuth [20] and verify the results with our numerical reconstruction.

5.1 A priori error estimate

As we have seen in Chapter 3 we can write the Poisson problem in weak form as

$$\begin{aligned} &\text{Find } \ u \in \ \mathcal{U} \ \text{ such that:} \\ &\quad (\nabla u, \nabla v) = v(x_0) \qquad \forall v \in \mathcal{V}, \end{aligned} \tag{5.2}$$

where

$$\begin{aligned} \mathcal{U} &= \left\{ u \mid u \in W^{1,2}(\Omega),\ u|_{\partial\Omega} = g \right\} \\ \mathcal{V} &= \left\{ v \mid v \in W^{1,2}(\Omega),\ v|_{\partial\Omega} = 0 \right\}. \end{aligned}$$

If we want to numerically reconstruct the fundamental solution $\Phi(x)$ on Ω as defined in Equation 2.3 we encounter suboptimal convergence for the finite element method. This because the solution is not in $H^1(\Omega)$. Köppl and Wohlmuth show in [20] that for the lowest order finite elements quasi-optimal and for higher order finite elements optimal order a priori estimates on quasi-uniform meshes in an L^2-seminorm exist.

They define a semi norm as an L^2 norm on a fixed sub domain where the locations of the delta source functions are excluded. Thus, there is no need for local mesh refinement around the locations of the delta source functions when an other norm is used to measure the error. In [20] the following theorem is proved.

Theorem 5.1.1. Let $u \in W_0^{1,p}(\Omega) = \mathcal{U}$ be the weak solution of Equation 5.2 and let $u^h \in \mathcal{U}^h$ be the finite element approximation using polynomials on the triangularization of Ω of at most k. Then we have for the L^2-error $\left\|u - u^h\right\|_{L^2(\Omega\setminus B)}$ the following upper bound,

$$\left\|u - u^h\right\|_{L^2(\Omega\setminus B)} \lesssim \begin{cases} h^2 |\ln h| & \text{if } \ k = 1, \\ h^{k+1} & \text{if } \ k > 1. \end{cases} \tag{5.3}$$

Proof. This theorem is proved in [20, Theorem 2.1].
This result is shown by induction over the approximation order k. In the proof a sequence of

nested concentric balls $B_{r_l}(x_0)$ $l = 1, 2, \ldots, k$ are introduced. In the notation a ball $B_r(x_0)$ is a ball centered at x_0 with a radius $r > 0$. On each ball suitable bounds for L^2-errors $\left\|u - u^h\right\|_{L^2(\Omega \setminus B_{r_l})}$ are derived. □

Note, the $\lesssim$ is used to omit a generic constant on the right hand side independent of h, but possibly depending on $\operatorname{dist}(x_0, B)$, $\operatorname{dist}(B, \partial\Omega)$, the solution u and the order k of the finite element approximation.

Further, the relative position of x_0 with respect to vertices does not play a role and the theorem holds for all $x_0 \in B$ with $\operatorname{dist}(x_0, \partial B) > 0$.

5.2 Numerical experiments

In this section we will verify the theoretical result from Theorem 5.1.1 and show some influences of specific locations for the Dirac points.

5.2.1 Problem setup

In this experiment we will reconstruct the fundamental solution on a cube using the HHG framework in three dimensions. We will restrict the domain to the unit cube $\Omega = (0, 1)^3$ and apply the Dirac distribution at $x_0 \in \Omega$. We will impose Dirichlet boundary conditions using the exact solution for the fundamental solution on $\partial\Omega$. The exact solution is given by

$$\Phi(x) = \frac{1}{4\pi|x - x_0|},$$

where $x_0 \in \Omega$. For the reconstruction we will use linear finite elements, i.e. $k = 1$.

To solve the Poisson equation we will use a multigrid solver where we use V-cycles with three pre- and post- smoothing steps. As stopping criteria we will use the criteria that the relative residual should be smaller than a tolerance $\epsilon = 10^{-12}$.

As input mesh we will use the unit cube consisting of six tetrahedra using Kuhn's triangularization of the cube [21], as discussed in section 4.1.1. To get different levels of accuracy in the mesh we will use the number of refinement levels l as a parameter that is related to the mesh size: $h \sim \frac{1}{2^l}$.

5.2.1.1 Euclidean norm

In Theorem 5.1.1 Köppl and Wohlmuth use an L^2 norm on a domain Ω excluding the singularity at the location where the Dirac measure is located. For the analysis we will use the l^2 Euclidean norm on our linear elements. This can be done because we use linear elements and all tetrahedra in the domain will have the same volume. Thus, integration over an linear element is equivalent to an average weighting of the function values at the vertices of the tetrahedron.

The theorem states that we need to exclude a ball $B_r(x_0)$ of a certain radius r around x_0, the location of the Dirac measure, to get quasi optimal convergence rates. Changing to the l^2 Euclidean norm has the advantage that we can make a clear distinction between vertices that are inside or outside the ball $B_r(x_0)$.

On a cubic-like domain such as we use in our experiments we are not able to correctly represent a ball. Therefore, we need to approximate a ball using the tetrahedra as we have in a cube. This approximation of a ball becomes better when the the mesh is refined. To get a fair comparison of the norm between different refinement levels we will replace the ball $B_r(x_0)$ by a cube $Q_r(x_0)$ to get a refinement level invariant norm[1]. This cube $Q_r(x_0) \subset \mathbb{R}^d,\ d \in \{2, 3\}$ centered in x_0 with 'radius' (i.e. half edge length) r is defined as

$$Q_r(\tilde{\mathbf{x}}) = \left\{\mathbf{x} \in \mathbb{R}^d \ :\ |\tilde{x}_i - x_i| < r,\ \forall i = 1, 2, \ldots, d\right\} \tag{5.4}$$

[1] Some not presented experiments hit that the replacement from a ball $B_r(x_0)$ to a cube $Q_r(x_0)$ is strictly not needed to recover second order convergence.

and the resulting Euclidean norm for errors $\mathbf{e} \in \mathbb{R}^n$ is given by

$$\|\mathbf{e}\|_{l^2(\Omega \setminus Q_r(x_0))} = \sqrt{\frac{\mathbf{e}^T \chi \chi^T \mathbf{e}}{n}}, \tag{5.5}$$

where χ is a vector indicating if the corresponding vertex $v_i \notin Q_r(x_0)$

$$\chi_i = \begin{cases} 0 & \text{if } v_i \in Q_r(x_0) \\ 1 & \text{if } v_i \notin Q_r(x_0). \end{cases}$$

5.2.2 Single Dirac distribution on a vertex

In this experiment we put the Dirac distribution on $x_0 = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, which will be on a vertex from refinement level $l = 1$ on. The masked Euclidean error for refinement levels $l \in \{2, 3, \ldots, 8\}$ and radii $r \in \left\{\frac{1}{2^2}, \frac{1}{2^3}, \ldots, \frac{1}{2^6}\right\}$ of the cube $Q_r(x_0)$ is plotted in Figure 5.1a. The corresponding estimated order of convergence is printed in Table 5.1 and shown in Figure 5.1b.

Refinement level	$r = \frac{1}{2^2}$	$r = \frac{1}{2^3}$	$r = \frac{1}{2^4}$	$r = \frac{1}{2^5}$	$r = \frac{1}{2^6}$
$2 \to 3$	1.69	0.53	0.53	0.53	0.53
$3 \to 4$	2.71	1.73	0.63	0.63	0.63
$4 \to 5$	2.31	2.58	1.66	0.57	0.57
$5 \to 6$	2.14	2.23	2.53	1.63	0.53
$6 \to 7$	**2.07**	2.10	2.21	2.51	1.61
$7 \to 8$	**2.03**	**2.05**	**2.09**	2.20	2.51

Table 5.1: Estimated order of convergence for Dirac measure at a vertex, $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$.

As we can see from the estimated order of convergence we are indeed able to recover second oder convergence with linear finite elements. The order of convergence is clearest for large radii, but also when we reduce the radii of the mask we recover second order convergence but then we need to go to finer meshes.

In general, we can bound the error e by a constant C and a power p of the mesh width h as

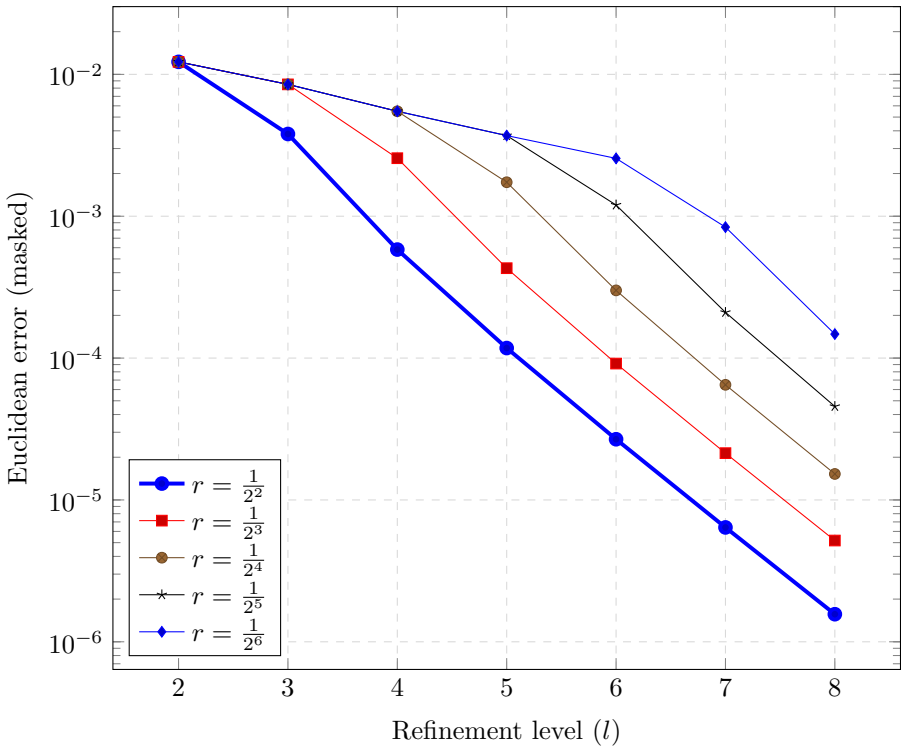
$$e = Ch^{-p},$$

where p is the order of convergence and C a constant. As noted by Theorem 5.1.1 the constant C may depend on the distance between the location of the Dirac measure x_0 and the boundary of the masked region $\partial Q_r(x_0)$.

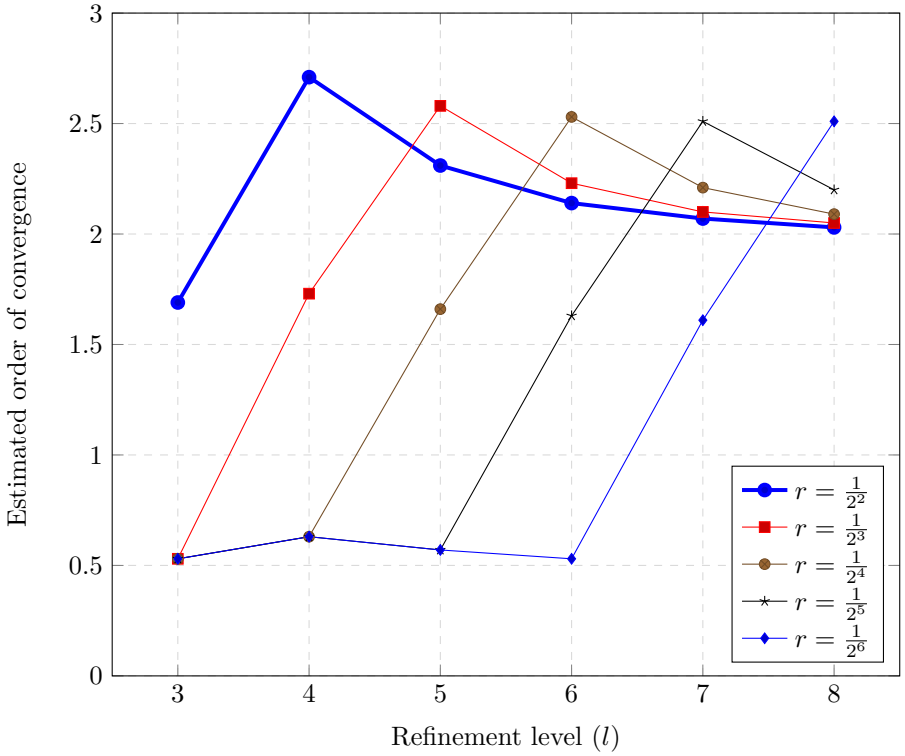
Because we use a strict inequality in Equation 5.4 for the 'distance' we see the fast decrease of the error starting from $r > h \sim \frac{1}{2^l}$. This because for $r \leq h \sim \frac{1}{2^l}$ we exclude at most one vertex. Because the singularity is on the vertex, we will exclude the singularity for all $r > 0$.

By taking a closer look at the error for this Poisson problem we see a well-shaped error plot in Figure 5.2a. In the figure we show the error in the numerical reconstruction of the Poisson in a cube and show the plane with the normal vector $\mathbf{n} = (0, 1, 1)$ through the Dirac measure.

As reflected in the estimated order of convergence the error will decrease when we go to finer meshes. In Figure 5.2b we zoom into the region around the Dirac measure and can clearly see that the line $x = 0.5$ might be interesting because we see negative (or very small) errors in positive error regions. This suggest that we can see oscillations which could cause problems for the convergence rates on this line. As we can see in Figure 5.3 the estimated order of convergence drops close to the Dirac measure but when we increase the mesh accuracy we see also a clear second order convergence over this line.

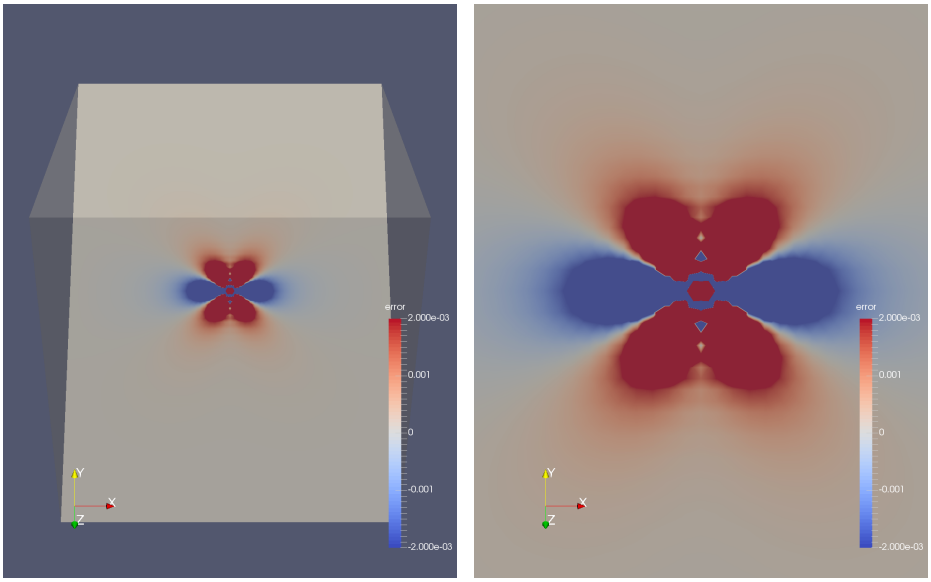


(a) Euclidean error (masked) for Dirac measure at a vertex, $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$.



(b) Estimated order of convergence for Dirac measure at a vertex, $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$.

Figure 5.1: Masked Euclidean error and estimated order of convergence for the Poisson problem with a Dirac measure at a vertex, $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$.



(a) Plot of the error in the numerical reconstruction of the Poisson problem on level 6 with a Dirac measure on a vertex.

(b) Detail plot of the error in the numerical reconstruction of the Poisson problem on level 6 with a Dirac measure on a vertex.

Figure 5.2: Color plots of Euclidean error in the numerical reconstruction with a Dirac measure at a vertex, $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$. The error is plotted over the plane through the Dirac measure with normal $n = (0, 1, 1)$.

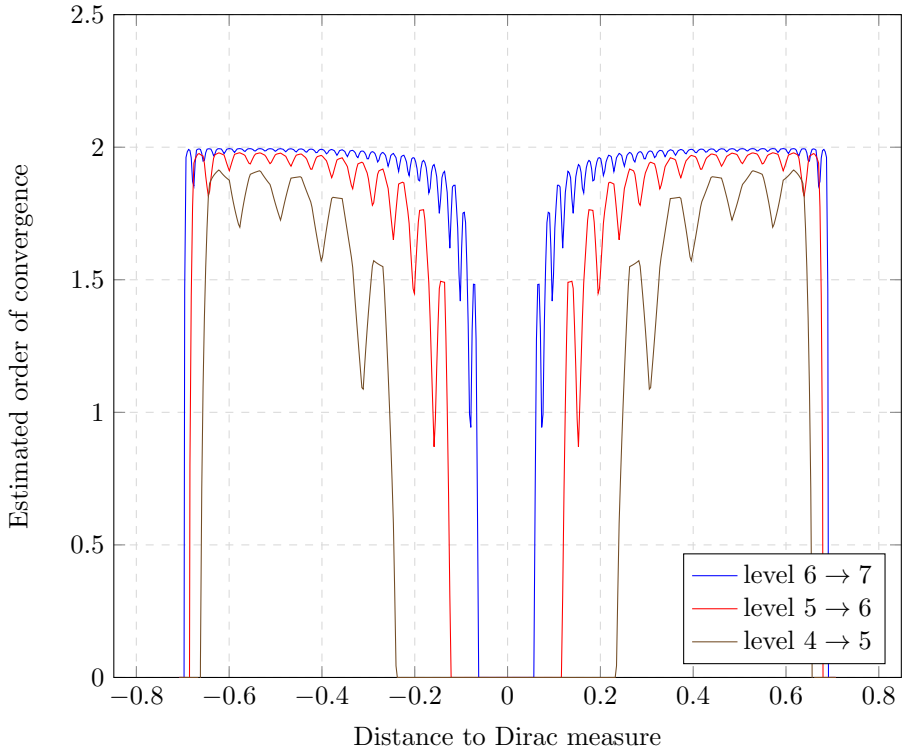
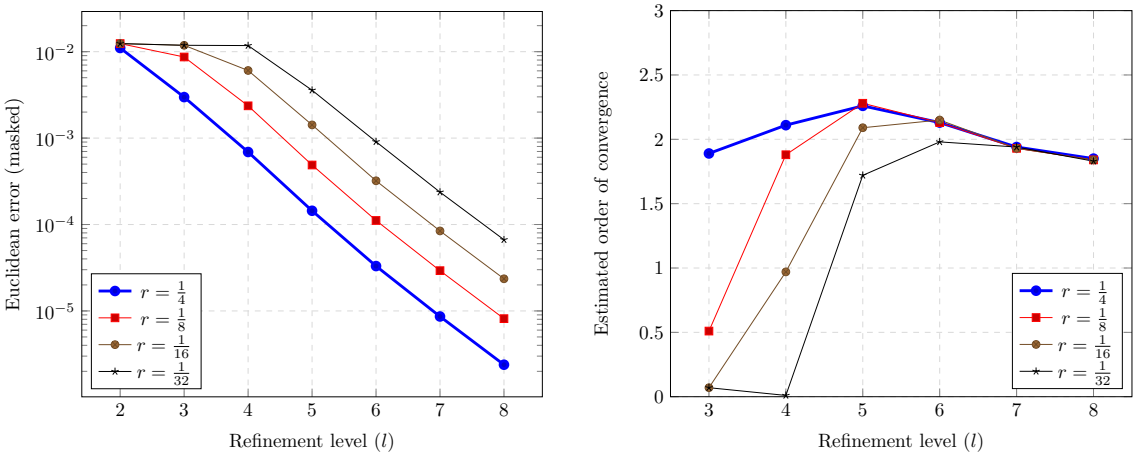


Figure 5.3: Estimated order of convergence over line $l : (\frac{1}{2}, 0, 1) \to (\frac{1}{2}, 1, 0)$ through Dirac measure on vertex $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$. Large oscillations close to the Dirac measure are removed and estimated order of convergence is set to zero.

Refinement level	$r = \frac{1}{2^2}$	$r = \frac{1}{2^3}$	$r = \frac{1}{2^4}$	$r = \frac{1}{2^5}$
$2 \to 3$	1.89	0.51	0.07	0.07
$3 \to 4$	2.11	1.88	0.97	0.01
$4 \to 5$	2.26	2.28	2.09	1.72
$5 \to 6$	2.13	2.13	2.15	1.98
$6 \to 7$	1.94	1.93	1.93	1.94
$7 \to 8$	1.85	1.84	1.84	1.83

Table 5.2: Estimated order of convergence for Dirac measure inside a tetrahedron, $x_0 = (0.51, 0.52, 0.53)$.



(a) Euclidean error (masked) for Dirac measure inside a tetrahedron, $x_0 = (0.51, 0.52, 0.53)$.

(b) Estimated order of convergence for Dirac measure inside a tetrahedron, $x_0 = (0.51, 0.52, 0.53)$.

Figure 5.4: Masked Euclidean error and estimated order of convergence for the Poisson problem with a Dirac measure inside a tetrahedron, $x_0 = (0.51, 0.52, 0.53)$.

5.2.3 Single Dirac distribution inside a tetrahedron

In this experiment we put the Dirac distribution on $x_0 = (0.51, 0.52, 0.53)$ which will be inside a tetrahedron. The masked Euclidean error for refinement levels $l \in \{2, 3, \dots, 8\}$ and radii $r \in \left\{\frac{1}{2^2}, \frac{1}{2^3}, \dots, \frac{1}{2^5}\right\}$ of the cube $Q_r(x_0)$ is plotted in Figure 5.4a. The corresponding estimated order of convergence is printed in Table 5.2 and shown in Figure 5.4b.

For a Dirac measure inside an element on a fine level we are not completely able to recover the second order convergence but we obtain convergence in the order of 1.8. Compared to the case where the Dirac measure is on a vertex this scenario is also less optimal because the 'load' of the Dirac measure is spread over all vertices of the element that contains the Dirac measure. Therefore, the Dirac measure is no longer concentrated at one vertex but at a set of vertices which results in a spreading of the Dirac measure over a couple of vertices.

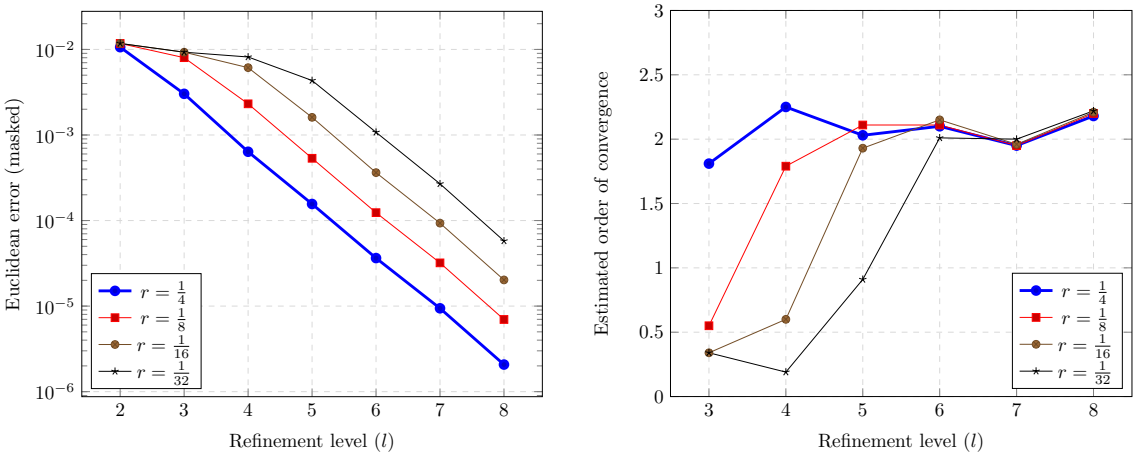
Because we didn't fix the relative location of the Dirac measure inside the finest tetrahedron, the weights for the spreading of the Dirac measure vary over the refinement levels and could influence the estimated order of convergence.

5.2.4 Single Dirac distribution at a face

In this experiment we put the Dirac distribution on $x_0 = (0.53, 0.52, 0.51)$ which will at a face between two tetrahedra. The masked Euclidean error for refinement levels $l \in \{2, 3, \dots, 8\}$ and radii $r \in \left\{\frac{1}{2^2}, \frac{1}{2^3}, \dots, \frac{1}{2^5}\right\}$ of the cube $Q_r(x_0)$ is plotted in Figure 5.5a. The corresponding estimated order of convergence is printed in Table 5.3 and shown in Figure 5.5b.

Refinement level	$r = \frac{1}{2^2}$	$r = \frac{1}{2^3}$	$r = \frac{1}{2^4}$	$r = \frac{1}{2^5}$
$2 \to 3$	1.81	0.55	0.34	0.34
$3 \to 4$	2.25	1.79	0.60	0.19
$4 \to 5$	2.03	2.11	1.93	0.91
$5 \to 6$	2.10	2.11	2.15	2.01
$6 \to 7$	1.95	1.95	1.96	2.00
$7 \to 8$	2.18	2.20	2.21	2.22

Table 5.3: Estimated order of convergence for Dirac measure at a face, $x_0 = (0.53, 0.52, 0.51)$.



(a) Euclidean error (masked) for Dirac measure at a face, $x_0 = (0.53, 0.52, 0.51)$.

(b) Estimated order of convergence for Dirac measure at a face, $x_0 = (0.53, 0.52, 0.51)$.

Figure 5.5: Masked Euclidean error and estimated order of convergence for the Poisson problem with a Dirac measure at a face, $x_0 = (0.53, 0.52, 0.51)$.

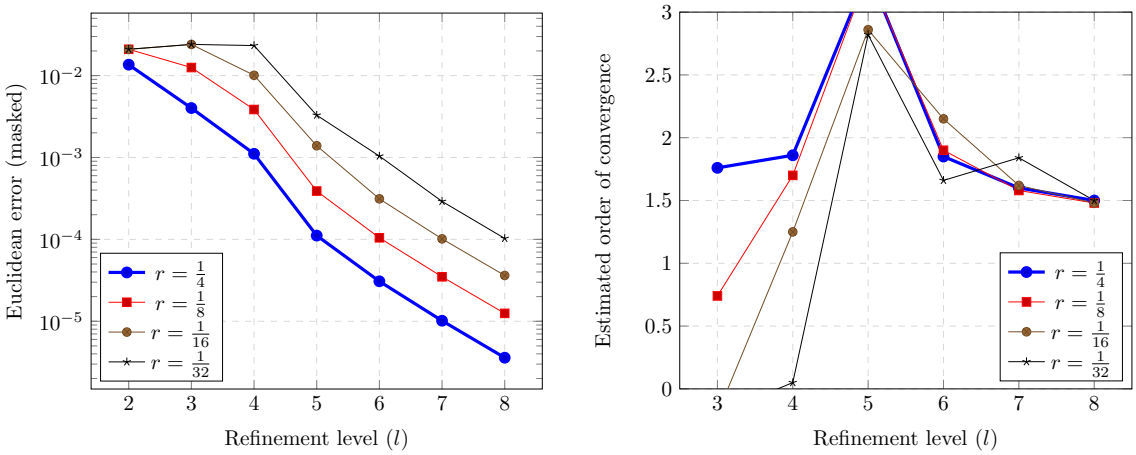
In this experiment where the Dirac measure is at a face between two elements we see the estimated order of convergence oscillates around 2.0. Again, this result is most visible for larger radii but on finer meshes we see also that the estimated order of convergence for the smaller radii tends to 2.0.

5.2.5 Single Dirac distribution at an edge

In this experiment we put the Dirac distribution on $x_0 = (0.53, 0.53, 0.53)$ which will at the main diagonal through the cube, which is an edge between the six macro elements. The masked Euclidean error for refinement levels $l \in \{2, 3, \ldots, 8\}$ and radii $r \in \left\{\frac{1}{2^2}, \frac{1}{2^3}, \ldots, \frac{1}{2^5}\right\}$ of the cube $Q_r(x_0)$ is plotted in Figure 5.6a. The corresponding estimated order of convergence is printed in Table 5.4 and shown in Figure 5.6b.

Refinement level	$r = \frac{1}{2^2}$	$r = \frac{1}{2^3}$	$r = \frac{1}{2^4}$	$r = \frac{1}{2^5}$
$2 \to 3$	1.76	0.74	-0.20	-0.20
$3 \to 4$	1.86	1.70	1.25	0.05
$4 \to 5$	3.32	3.31	2.86	2.82
$5 \to 6$	1.85	1.90	2.15	1.66
$6 \to 7$	1.60	1.58	1.62	1.84
$7 \to 8$	1.50	1.48	1.48	1.50

Table 5.4: Estimated order of convergence for Dirac measure at an edge, $x_0 = (0.53, 0.53, 0.53)$.



(a) Euclidean error (masked) for Dirac measure at an edge, $x_0 = (0.53, 0.53, 0.53)$.

(b) Estimated order of convergence for Dirac measure at an edge, $x_0 = (0.53, 0.53, 0.53)$.

Figure 5.6: Masked Euclidean error and estimated order of convergence for the Poisson problem with a Dirac measure at an edge, $x_0 = (0.53, 0.53, 0.53)$.

When a Dirac measure is located at an edge we get the lowest estimated order of convergence, around 1.5. Here, a similar problem may influence the convergence rate as we have seen in case of the Dirac measure inside a tetrahedron. For each refinement level we fixed the location of the Dirac measure which will lead to a nonzero contribution from this Dirac measure to the two vertices at the ends of this edge. On a mesh with another refinement level the weighting of the contributions will differ. For better results we could fix the relative position of the Dirac measure on this line and change the exact location of the Dirac measure on different refinement levels.

The high peak drop of the error on refinement level 5 can be explained by the observation that on refinement level 5 we have a vertex at $(\frac{17}{32}, \frac{17}{32}, \frac{17}{32})$ which is relatively close to the singularity. Note for the Dirac measure at $x_0 = (0.53, 0.53, 0.53)$ with $l = 5, r = \frac{1}{4}$ we have an Euclidean error of $e = 1.11 \cdot 10^{-4}$, which is in the same order as the Euclidean error for $l = 5, r = \frac{1}{4}$ with x_0 on a vertex: $e = 1.18 \cdot 10^{-4}$. While all other errors in the case of a Dirac measure on an edge are higher than the error with a Dirac measure on a vertex.

5.3 Conclusion and optimizations

As we have seen in these experiments, depending on the location of the Dirac measure relative to the finite element mesh, we are more or less able to recover second order convergence. The results are clear for the case when we put the Dirac measure on a vertex. In the other cases the results are not completely second order. Especially for the case where we have a Dirac measure on an edge the estimated order of convergence drops.

However, the numerical reconstruction of the potential close to the singularity has small oscillations. But these small oscillations don't influence the convergence rate as we have seen in the experiment where we put the Dirac measure on a vertex.

As pointed out in the treatment of the results for the Dirac measure inside the tetrahedron one artifact that can spoil down the results is that we didn't fix the relative location of a Dirac measure to the mesh on the finest level. When we put the Dirac measure not on a vertex this will introduce a spreading of the Dirac measure over all neighboring vertices. The spreading of the Dirac measure over all neighboring vertices is weighted by the distance between the vertex and the location of the Dirac measure. So, when we fix the location relative to the mesh on the finest level the spreading of the Dirac measure on the runs over different refinement levels will be constant. The experiment with the Dirac measure on an edge and refinement level 5 shows that the varying

spreading has indeed a large impact on the error and the estimated order of convergence.

Because our main focus will be on the Stokes problem we will fix the relative position of a Dirac measure to the finest refinement level instead of fixing a specific location for the Stokes problem and leave the experiments for the Poisson problem unchanged.

As shown in [20] the Poisson problem with Dirac measures not on a vertex should also result in second order convergence.

Chapter 6

Stokes problem

The aim of this chapter is to verify if we can get similar convergence rates for the reconstruction of the fundamental solution of the Stokes system on a finite element mesh, as we found for the Poisson problem. Therefore, we will numerically reconstruct the fundamental solution to the Stokes system on a simulation domain $\Omega \subset \mathbb{R}^d$, with $d \in \{2, 3\}$ where Ω is a bounded, open, convex and polyhedral domain.

$$\begin{aligned} -\Delta \mathbf{u} + \nabla p &= \mathbf{f} \quad \text{in } \Omega \\ \nabla \cdot \mathbf{u} &= 0 \quad \text{in } \Omega \\ \mathbf{u} &= \mathbf{g} \quad \text{on } \partial\Omega, \end{aligned} \tag{6.1}$$

where $\mathbf{u} : \Omega \to \mathbb{R}^d$ is the velocity, $p : \Omega \to \mathbb{R}$ the pressure and $\mathbf{f} : \Omega \to \mathbb{R}^d$ the applied body forces. The boundary of the domain Ω is described by $\partial\Omega$ where we impose Dirichlet boundary conditions described by $\mathbf{g} : \partial\Omega \to \mathbb{R}^d$.

6.1 Numerical experiments $\mathbb{P}_1$-iso-$\mathbb{P}_2/\mathbb{P}_1$ finite elements

In this section we will examine whether an a priori error estimate, similar to Theorem 5.1.1, also exists for the fundamental solution of the Stokes system for velocity $\mathbf{u}$ and pressure p.

6.1.1 Problem setup

In this experiment we will reconstruct the fundamental solution on a cube using the HHG framework. We will restrict the domain to the unit cube $\Omega = (0, 1)^3$ and apply the Dirac force at $\mathbf{x}_0 \in \Omega$ in the x-direction. We will impose Dirichlet boundary conditions using the exact solution for the fundamental solution on $\partial\Omega$. The exact solution $(\mathbf{u}, p)$ with Dirac force at $\mathbf{x}_0 \in \Omega$ in direction $\mathbf{f} = [1, 0, 0]^T$ is given by (see also Equation 2.11)

$$\begin{aligned} \mathbf{u}(\mathbf{x} - \mathbf{x_0}) &= \mathbf{F}(\mathbf{x} - \mathbf{x_0})\mathbf{f} \\ p(\mathbf{x} - \mathbf{x_0}) &= \frac{\mathbf{r}}{4\pi \|\mathbf{r}\|^3}\mathbf{f} \end{aligned} \tag{6.2}$$

where $\mathbf{F}(\mathbf{r})$ is the Stokeslet or Oseen tensor:

$$\mathbf{F}(\mathbf{x} - \mathbf{x_0}) = \mathbf{F}(\mathbf{r}) = \frac{1}{8\pi\mu}\left(\frac{\mathbf{I}}{\|\mathbf{r}\|} + \frac{\mathbf{r} \otimes \mathbf{r}}{\|\mathbf{r}\|^3}\right)$$

For the reconstruction we will use $\mathbb{P}_1$-iso-$\mathbb{P}_2$ elements for the velocity $\mathbf{u}$ and linear elements for the pressure p. To solve the Stokes system we will use the Schur complement CG solver where we use $n_p = 4$ CG iterations for reconstructing the pressure p for each reconstructed velocity $\mathbf{u}$.

For the reconstruction of the velocity $\mathbf{u}$ and auxiliary variable $\mathbf{v}$ we will use a multigrid solver with three V-cycles with 5 pre- and post- smoothing steps. As stopping criteria we will require the relative residual to be smaller as tolerance $\epsilon = 10^{-8}$.

As input mesh we will use the unit cube consisting of six tetrahedra using Kuhn's triangularization of the cube [21], as discussed in Section 4.1.1. To get different levels of accuracy in the mesh we will use the number of refinements for the velocity $\mathbf{u}$, which we will call l, as a parameter that is related to the mesh size: $h \sim \frac{1}{2^l}$. The pressure p is solved on one level coarser, thus on refinement level $l-1$. The refinement level for the pressure is often called p-level.

Similar to the Poisson problem, we will use the masked Euclidean l^2 norm of Equation 5.5 for the pressure and the velocity components. But because imposing Dirichlet boundary conditions on all boundaries may cause errors in the discretization for the Stokes system we will exclude a boundary layer with a thickness of $t = \frac{1}{16}$ around the boundary of domain Ω. The analysis of this error is out of the scope of this project but it seems that the error is bounded to the boundary elements itself.

6.1.2 Single Dirac force on a vertex

In this experiment we put the Dirac force in the x-direction on $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$. Because we use regular refinements for the cube, this location will be on a vertex after one refinement level, as visualized in Figure 4.2a. The masked Euclidean error for velocity $\mathbf{u}$ and pressure p with refinement levels $l \in \{3, 4, \ldots, 10\}$ and radii $r \in \left\{\frac{1}{2^2}, \frac{1}{2^3}, \ldots, \frac{1}{2^6}\right\}$ of the cube $Q_r(x_0)$ are plotted in Figure 6.1a and Figure 6.2a, respectively. The corresponding estimated order of convergence for velocity $\mathbf{u}$ and pressure p are printed in Table 6.1 and Table 6.2 and shown in Figure 6.1b and Figure 6.2b.

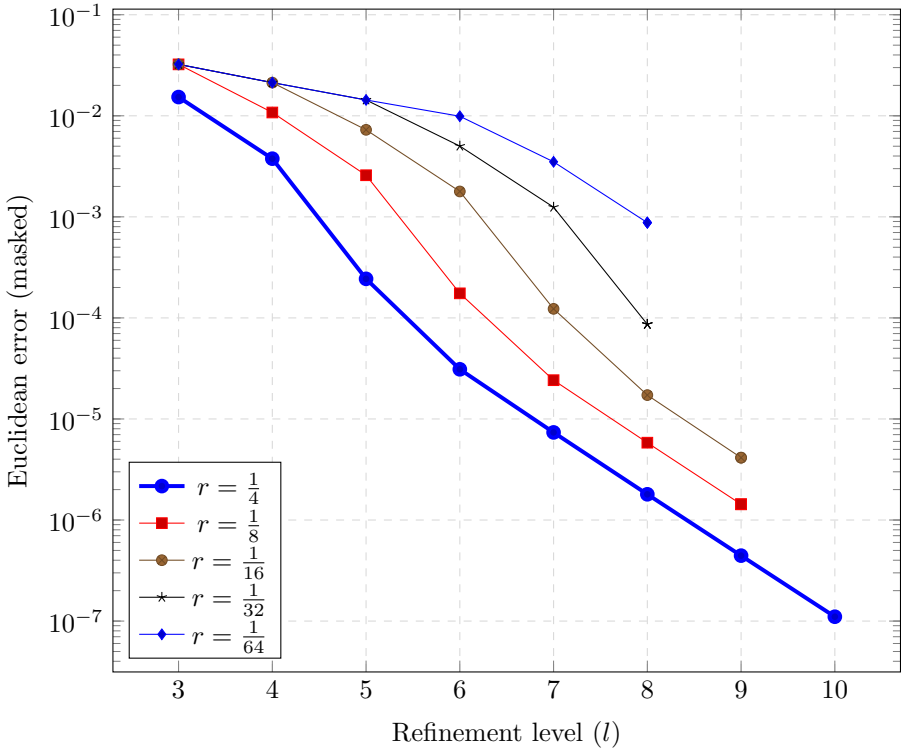
Refinement level	$r = \frac{1}{2^2}$	$r = \frac{1}{2^3}$	$r = \frac{1}{2^4}$	$r = \frac{1}{2^5}$	$r = \frac{1}{2^6}$
$3 \to 4$	2.03	1.58	0.60	0.60	0.60
$4 \to 5$	3.95	2.06	1.55	0.57	0.57
$5 \to 6$	2.97	3.88	2.03	1.52	0.53
$6 \to 7$	2.07	2.86	3.86	2.01	1.50
$7 \to 8$	**2.03**	2.05	2.84	3.85	2.00
$8 \to 9$	**2.02**	**2.02**	2.06		
$9 \to 10$	**2.01**				

Table 6.1: Estimated order of convergence for the velocity $\mathbf{u}$ with Dirac force in x-direction at a vertex, $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$.

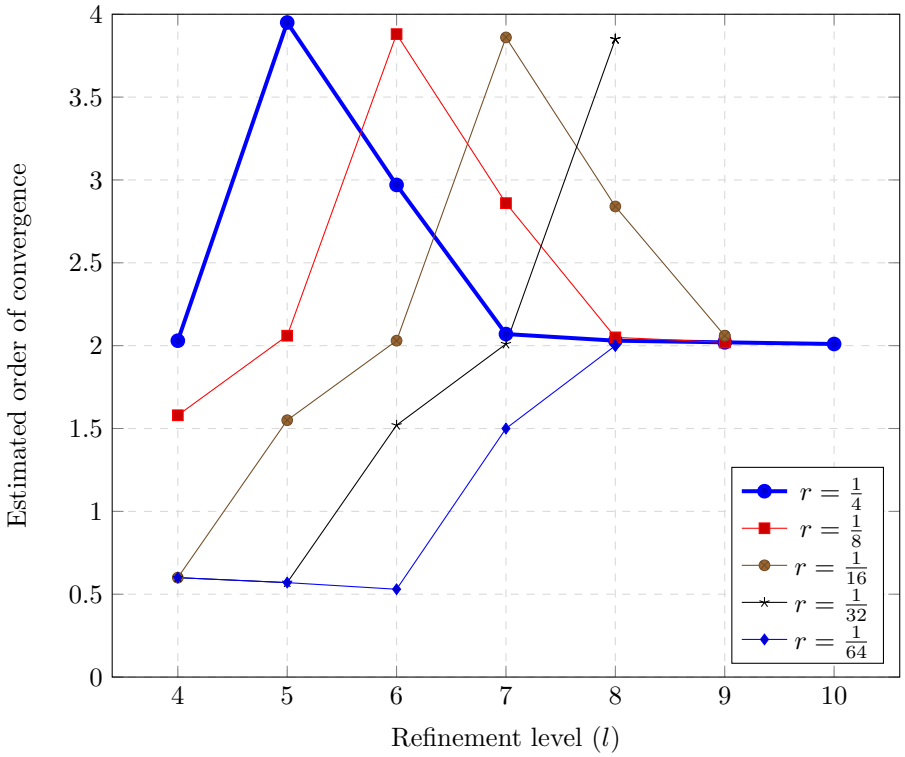
Refinement p-level	$r = \frac{1}{2^2}$	$r = \frac{1}{2^3}$	$r = \frac{1}{2^4}$	$r = \frac{1}{2^5}$	$r = \frac{1}{2^6}$
$2 \to 3$	1.47	-0.07	-0.07	-0.07	-0.07
$3 \to 4$	3.10	1.20	-0.35	-0.35	-0.35
$4 \to 5$	4.54	3.01	1.12	-0.43	-0.43
$5 \to 6$	2.28	4.46	2.97	1.09	-0.47
$6 \to 7$	2.08	2.24	4.44	2.95	1.07
$7 \to 8$	**2.04**	2.06	2.23		
$8 \to 9$	**2.02**				

Table 6.2: Estimated order of convergence for the pressure p with Dirac force in x-direction at a vertex, $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$.

As we can see also for the Stokes problem with a Dirac force at a vertex we are able to recover second order convergence in the far field for both velocity $\mathbf{u}$ and pressure p. The results are clear for large radii and if we decrease the radius of the mask we see similar progress in the estimated order of convergence on a finer mesh.

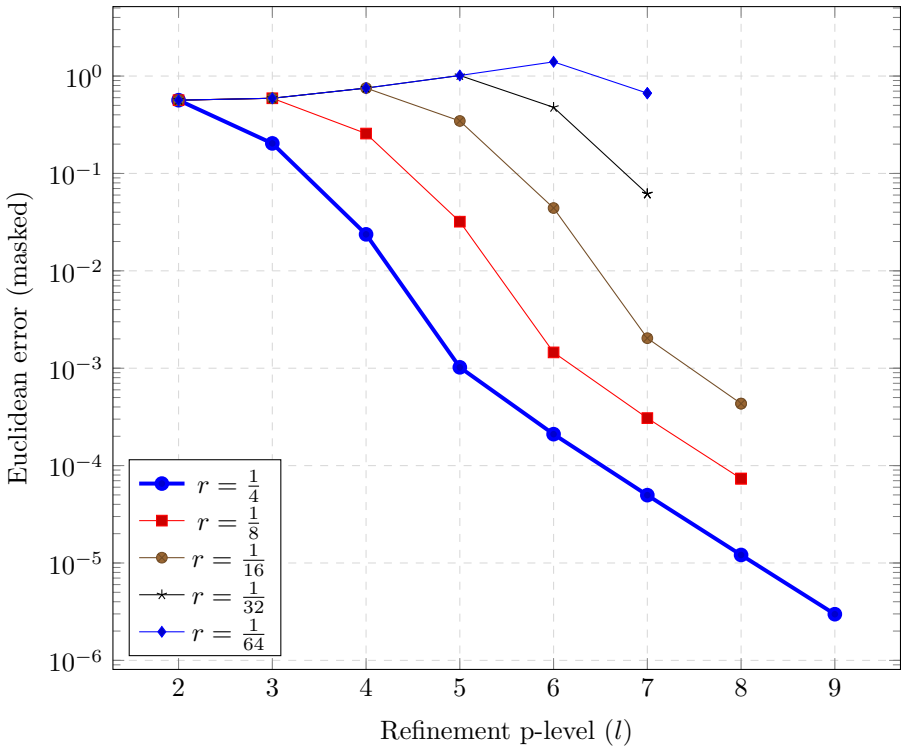


(a) Euclidean error (masked) for the velocity $\mathbf{u}$ (Dirac force at a vertex).

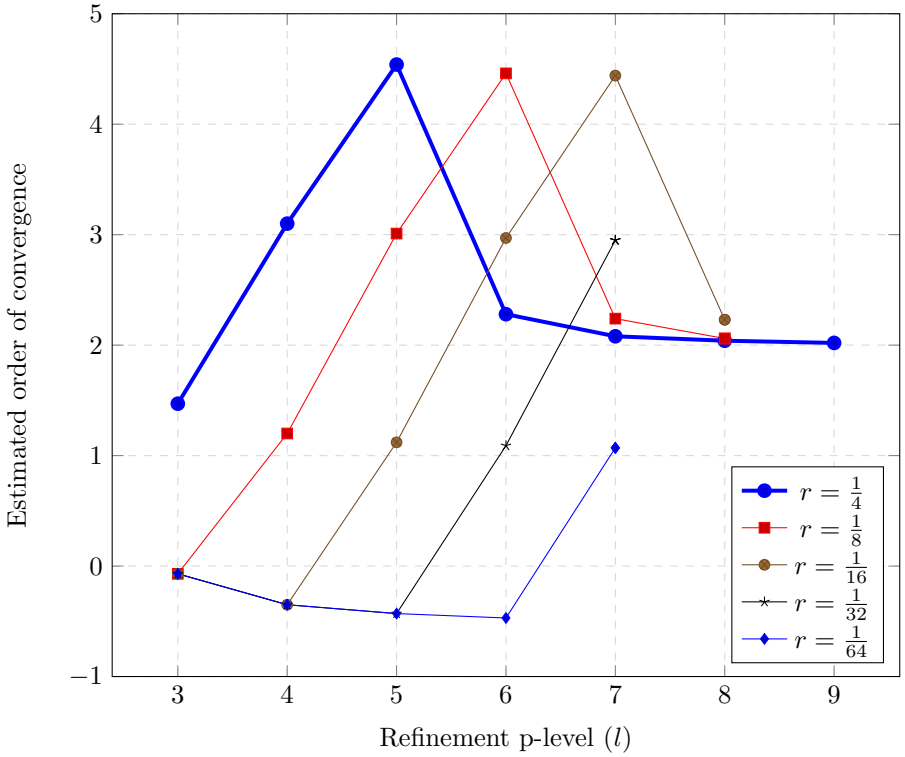


(b) Estimated order of convergence for the velocity $\mathbf{u}$ (Dirac force at a vertex).

Figure 6.1: Masked Euclidean error and estimated order of convergence for the velocity $\mathbf{u}$ of the Stokes problem with a Dirac force in x-direction at a vertex, $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$.



(a) Euclidean error (masked) for the pressure p (Dirac force at a vertex).



(b) Estimated order of convergence for the pressure p (Dirac force at a vertex).

Figure 6.2: Masked Euclidean error and estimated order of convergence for the pressure p of the Stokes problem with a Dirac force in x-direction at a vertex, $x_0 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$.

If we compare the progress of the estimated order of convergence for the velocity $\mathbf{u}$ of the Stokes problem with the solution of the Poisson problem with a Dirac force on a vertex, then we see similar results. First we need a certain amount of refinement levels to get an accurate approximation of the solution and after a peak in the estimated order of convergence we see a decrease to second order convergence. For the Stokes problem we need approximately one refinement level more to get an estimated order of convergence close to two. This can be explained because we reconstruct the pressure p on one level coarser. So, to get also the pressure in a regime where we have enough unknowns to get good reconstructions we need one additional refinement level.

Next we will take a closer look into the reconstructed pressure to get a better feeling about what happens with the pressure around a singularity on a finite element mesh. The error in the reconstructed pressure on refinement level 7 (which means, the pressure is reconstructed on p-level 6) is shown in Figure 6.3. Here we show four slices that rotate through the cube around the x-axis. This leads to figures where the Dirac force is parallel to the rotation axis of these slices.

As we can see from the figures the error seems reasonable, but we see also oscillation patterns in these figures. Especially Figure 6.3c shows oscillations in an elliptical shaped region.

Because the oscillations on this line will dominate the error for lower refinement levels we plot the reconstructed pressure on different refinement levels over this line which is given by $(0, \frac{1}{6}, \frac{1}{6}) \to (0, \frac{5}{6}, \frac{5}{6})$. The reconstructed pressure on different refinement levels for the pressure are shown in Figure 6.4. Figures 6.4b and 6.4c show clearly the oscillations in the reconstructed pressure. From the plots on the finer levels of Figures 6.4e and 6.4f we can conclude that the domain where this oscillations exists will shrink.

To measure how fast the oscillations will concentrate around the Dirac force we compute the estimated order of convergence over the line $(0, \frac{1}{6}, \frac{1}{6}) \to (1, \frac{5}{6}, \frac{5}{6})$ through Dirac force at the vertex $x_0 = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. As we can see in Figure 6.5 it is hard to recover the second order convergence on refinement levels up to level 6.

From level 7 on we get indeed a clear second order convergence over the line far away from the Dirac force[1], as shown in Figure 6.5. To get a clean image around the Dirac force we masked out the oscillations in the estimated order of convergence close to the Dirac force.

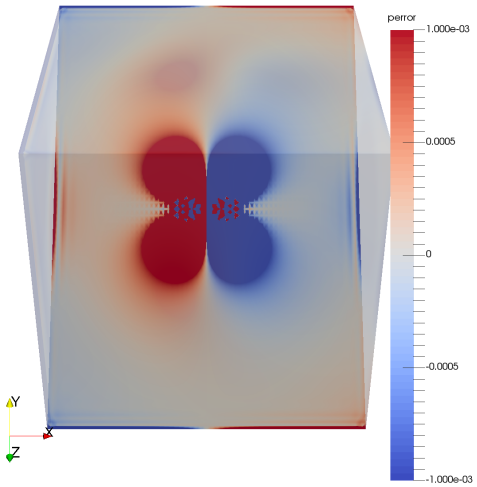
From the estimated order of convergence over a line we can compute the radius of the mask that we will need to hide the oscillations around the Dirac force. We will define the radius such that it hides all oscillations with an amplitude > 0.05 around the Dirac force. For p-level $5 \to 6$ this radius is undefined but for the two other estimated order of convergence lines we find the radii as shown in Table 6.3.

Level difference	**radius**
p-level $5 \to 6$	NaN
p-level $6 \to 7$	0.24725
p-level $7 \to 8$	0.12383

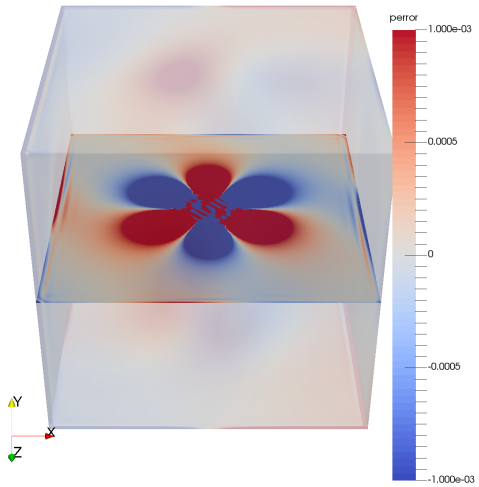
Table 6.3: Minimal needed radius to mask out oscillations in estimated order of convergence around Dirac force on line $(0, \frac{1}{6}, \frac{1}{6}) \to (1, \frac{5}{6}, \frac{5}{6})$ as shown in Figure 6.5.

The number of meaning full data points in Table 6.3 is minimal, but we see a straight linear relation between incrementing the refinement level and reducing the mask-radius. So, we can half the radius for each new refinement level without having trouble with the oscillations in the estimated order of convergence. Similar results are also visible in Figure 6.1 and 6.2, where we see also that halving the radius results in a shift of the error and estimated order of convergence plots to one additional refinement level.

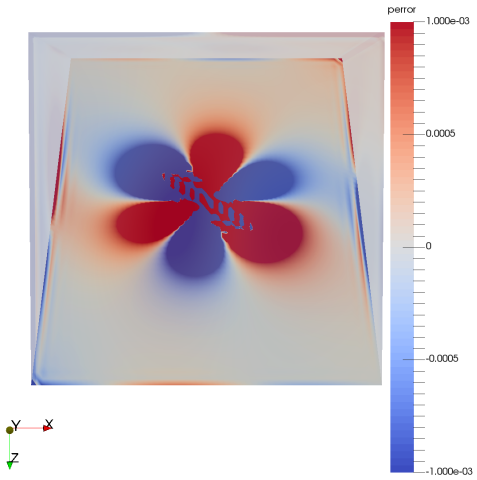
[1] Also on the boundary we see some problems but because we masked out a boundary layer these artifacts will not influence the results of our norm.



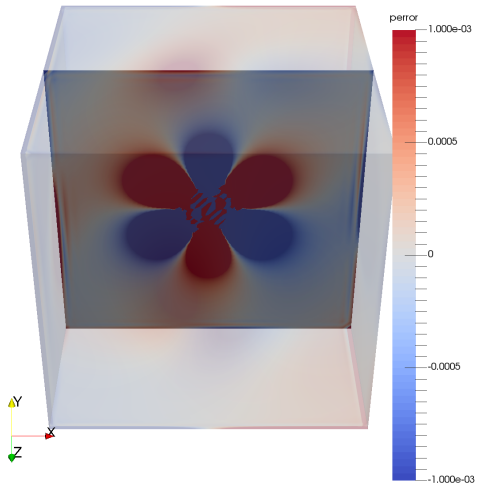
(a) Error in reconstructed pressure on slice $\mathbf{n} = (0, 1, 1)$.



(b) Error in reconstructed pressure on slice $\mathbf{n} = (0, 1, 0)$.

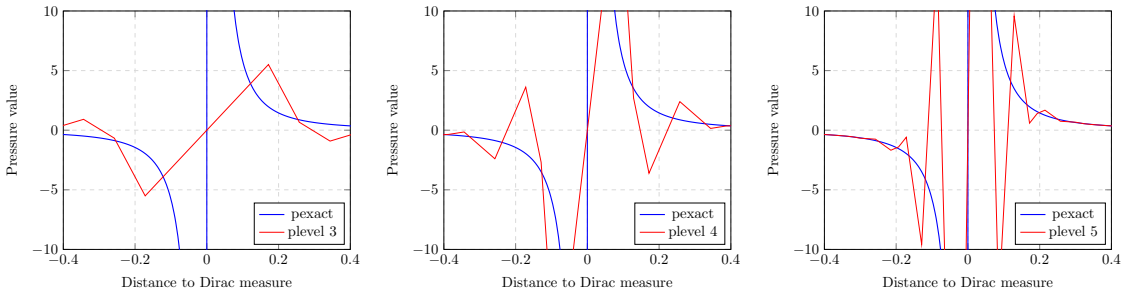


(c) Error in reconstructed pressure on slice $\mathbf{n} = (0, 1, -1)$. Note: cube is rotated w.r.t. the other views such that we have a clear view on this slice.



(d) Error in reconstructed pressure on slice $\mathbf{n} = (0, 0, 1)$.

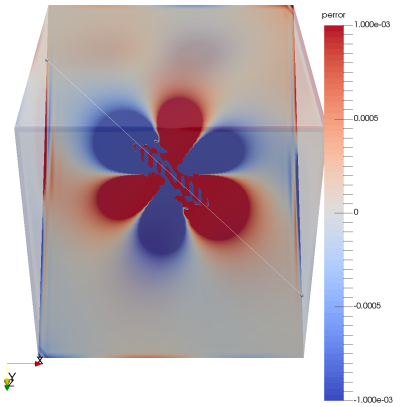
Figure 6.3: Color plots of the error in reconstructed pressure of some interesting slices through the domain on plevel 6. The slices go through the Dirac force in the center of the domain $x_0 = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ and the normals are given per plot. Color scale range for all figures: $[-10^{-3}, 10^{-3}]$.



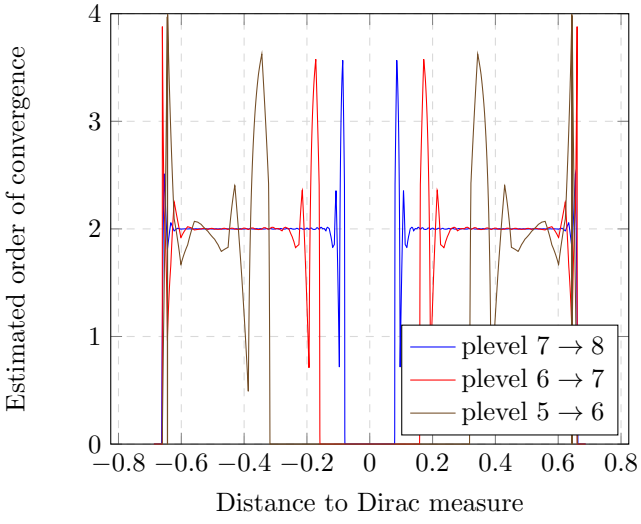
(a) Pressure on p-level 3. (b) Pressure on p-level 4. (c) Pressure on p-level 5.

(d) Pressure on p-level 6. (e) Pressure on p-level 7. (f) Pressure on p-level 8.

Figure 6.4: Exact pressure and the reconstructed pressure for the Stokes system with a Dirac force on a vertex over line $l : (0, \frac{1}{6}, \frac{1}{6}) \to (1, \frac{5}{6}, \frac{5}{6})$ through Dirac measure on a vertex $x_0 = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. Reconstructed pressures are shown on different refinement levels for the pressure.



(a) The line $l : (0, \frac{1}{6}, \frac{1}{6}) \to (1, \frac{5}{6}, \frac{5}{6})$ through error in reconstructed pressure on p-level 6.



(b) Estimated order of convergence over line l.

Figure 6.5: Estimated order of convergence over line $l : (0, \frac{1}{6}, \frac{1}{6}) \to (1, \frac{5}{6}, \frac{5}{6})$ through Dirac measure on a vertex $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. Large oscillations around the Dirac force are masked out and estimated order of convergence is set to zero.

Refinement level	$r = \frac{1}{2^2}$
$3 \to 4$	1.96
$4 \to 5$	3.47
$5 \to 6$	2.99
$6 \to 7$	2.08
$7 \to 8$	**2.04**
$8 \to 9$	**2.02**
$9 \to 10$	**2.01**

Table 6.4: Estimated order of convergence for the velocity $\mathbf{u}$ with Dirac force in x-direction on a vertex outside the center.

Refinement p-level	$r = \frac{1}{2^2}$
$2 \to 3$	1.63
$3 \to 4$	1.64
$4 \to 5$	4.55
$5 \to 6$	2.28
$6 \to 7$	2.08
$7 \to 8$	**2.04**
$8 \to 9$	**2.02**

Table 6.5: Estimated order of convergence for the pressure p with Dirac force in x-direction on a vertex outside the center.

6.1.3 Single Dirac force on a vertex outside center

In this experiment we put the Dirac force in the x-direction on

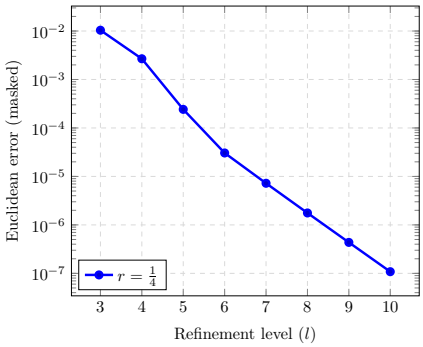
$$x_0 = \left(\frac{1}{2} + \frac{1}{2^4}, \frac{1}{2} - \frac{1}{2^4}, \frac{1}{2} + \frac{1}{2^4}\right),$$

which will be on a vertex from refinement level $l = 4$. The masked Euclidean error for velocity $\mathbf{u}$ and pressure p with refinement levels $l \in \{3, 4, \ldots, 10\}$ and radius $r = \frac{1}{2^2}$ of the cube $Q_r(x_0)$ are plotted in Figure 6.6a and Figure 6.6b, respectively. The corresponding estimated order of convergence for velocity $\mathbf{u}$ and pressure p are printed in Table 6.4 and Table 6.5 and shown in Figure 6.7a and Figure 6.7b.

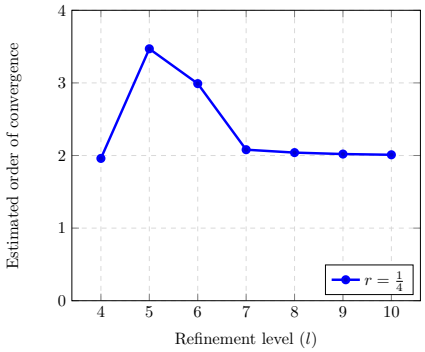
Because this test case is just a shift of the Dirac force to another vertex we expect similar errors as the case where the Dirac force is on the center vertex. In Figure 6.8 we plot the error for the velocity $\mathbf{u}$ and pressure p for the case where the Dirac force is on the center vertex and the case where the Dirac force is on a vertex of refinement level 4.

As we can see from the error plots, the differences are only in the cases where x_0 is not on a vertex. When the accuracy of the mesh is such that we have indeed a vertex at the location of the Dirac force the differences are minimal.

As we can see also for the Stokes problem with a Dirac force at a vertex outside the center we are able to recover second order convergence in the far field for both velocity $\mathbf{u}$ and pressure p. The results are clear for large radius $r = \frac{1}{4}$ and its similarity to the case where the Dirac is on the center vertex leads to the fact that we can decrease the radius r and get also second order convergence in the far field on finer refinement levels.

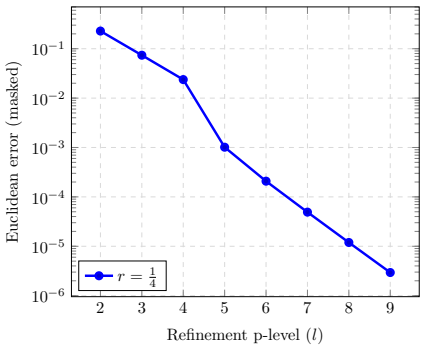


(a) Euclidean error (masked) for the velocity $\mathbf{u}$ (Dirac force at a vertex outside the center).

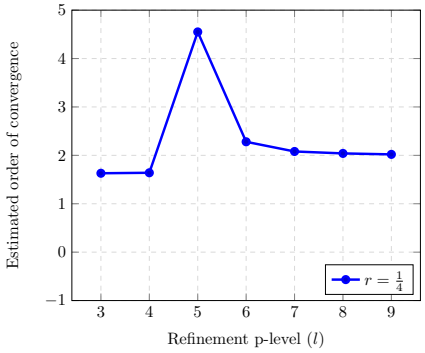


(b) Estimated order of convergence for the velocity $\mathbf{u}$ (Dirac force at a vertex outside the center).

Figure 6.6: Masked Euclidean error and estimated order of convergence for the velocity $\mathbf{u}$ of the Stokes problem with a Dirac force in x-direction on a vertex outside the center.

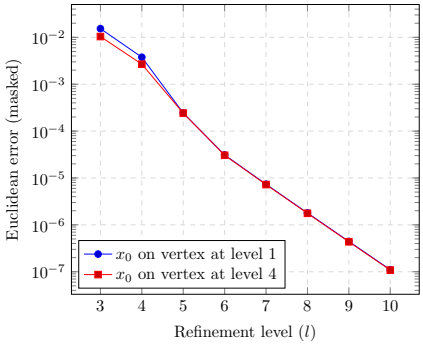


(a) Euclidean error (masked) for the pressure p (Dirac force at a vertex outside the center).

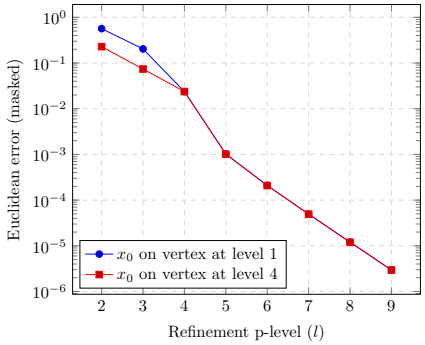


(b) Estimated order of convergence for the pressure p (Dirac force at a vertex outside the center).

Figure 6.7: Masked Euclidean error and estimated order of convergence for the pressure p of the Stokes problem with a Dirac force in x-direction a vertex outside the center.



(a) Euclidean error (masked) for the velocity $\mathbf{u}$ (Dirac force at different vertices).



(b) Euclidean error (masked) for the pressure p (Dirac force at different vertices).

Figure 6.8: Comparison of masked Euclidean error for velocity $\mathbf{u}$ and pressure p of the Stokes problem with a Dirac force in x-direction on different vertices.

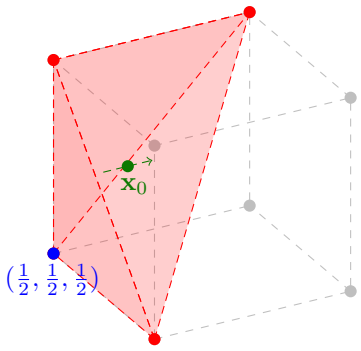


Figure 6.9: The Dirac force inside the tetrahedron to the north-east-front side of the center vertex. In the name convention of Figure 4.1 this will be inside tetrahedron E.

6.1.4 Single Dirac force inside a tetrahedron

In this experiment we put the Dirac force in the x-direction inside a tetrahedron. Because the uniform refinement will change the relative position of a Dirac force to the nodes of the refined mesh we will fix the relative position of the Dirac force and change its location on each refinement level.

We will fix the Dirac force inside the tetrahedron to the north-east-front side of the center vertex, as shown in Figure 6.9. For each refinement level l the location $\mathbf{x}_0$ for the Dirac force is given by

$$\mathbf{x}_0 = \left(\frac{1}{2} + \frac{1}{2} \cdot \left(\frac{1}{2}\right)^l; \quad \frac{1}{2} + \frac{1}{4} \cdot \left(\frac{1}{2}\right)^l; \quad \frac{1}{2} - \frac{1}{4} \cdot \left(\frac{1}{2}\right)^l\right). \tag{6.3}$$

The masked Euclidean error for velocity $\mathbf{u}$ and pressure p with refinement levels $l \in \{3, 4, \ldots, 10\}$ and radii $r \in \left\{\frac{1}{2^2}, \frac{1}{2^3}, \frac{1}{2^4}\right\}$ of the cube $Q_r(x_0)$ are plotted in Figure 6.10a and Figure 6.11a, respectively. The corresponding estimated order of convergence for velocity $\mathbf{u}$ and pressure p are printed in Table 6.6 and Table 6.7 and shown in Figure 6.10b and Figure 6.11b.

Refinement level	$r = \frac{1}{2^2}$	$r = \frac{1}{2^3}$	$r = \frac{1}{2^4}$
$3 \to 4$	2.25	1.35	1.59
$4 \to 5$	3.19	2.39	1.39
$5 \to 6$	2.22	3.03	2.35
$6 \to 7$	**2.02**	2.15	2.99
$7 \to 8$	**2.01**	**2.01**	2.13
$8 \to 9$	**2.00**	**2.00**	**2.00**
$9 \to 10$	**2.00**	**2.00**	

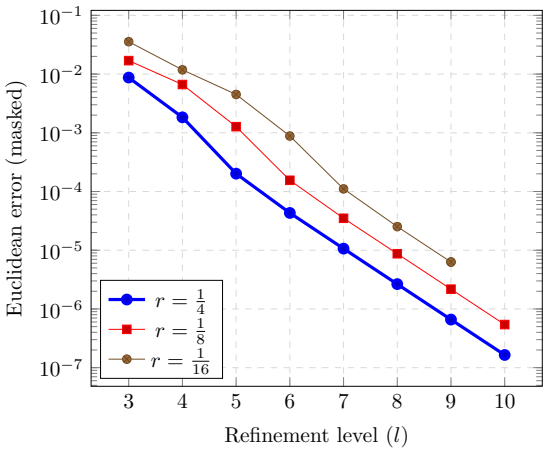
Table 6.6: Estimated order of convergence for the velocity $\mathbf{u}$ with Dirac force in x-direction inside a tetrahedron.

Also for a Dirac force inside a tetrahedron we are able to recover second order convergence in the far field for the velocity $\mathbf{u}$. The measured errors for the pressure p result only in first order convergence in the far field.

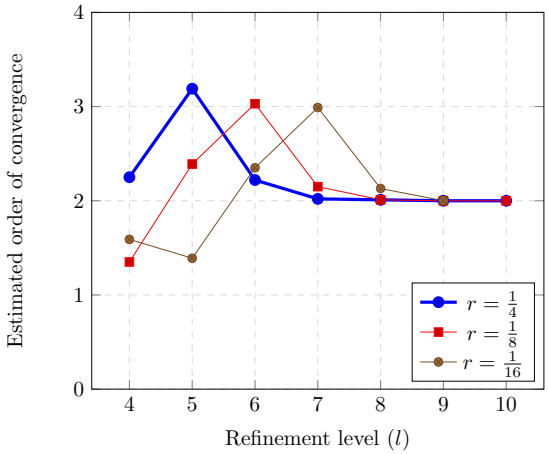
To compare these results with the results for the Dirac force on a vertex we will slice also through the pressure error on similar planes as the planes used in Figure 6.3. For the Dirac measure inside a tetrahedron we will set the origin of the plane at the location of the Dirac force and keep the normals unchanged. The results are shown in Figure 6.13. Note that the color scale for this series of images is on a range $\pm 10^{-2}$ instead of the $\pm 10^{-3}$ for the case where the Dirac force is on a vertex.

Refinement p-level	$r = \frac{1}{2^2}$	$r = \frac{1}{2^3}$	$r = \frac{1}{2^4}$
$2 \to 3$	1.49	0.95	1.77
$3 \to 4$	2.87	1.55	0.64
$4 \to 5$	2.23	2.86	1.46
$5 \to 6$	1.07	3.11	2.84
$6 \to 7$	1.02	1.27	3.60
$7 \to 8$	1.01	1.08	1.74
$8 \to 9$	1.00	1.02	

Table 6.7: Estimated order of convergence for the pressure p with Dirac force in x-direction inside a tetrahedron.

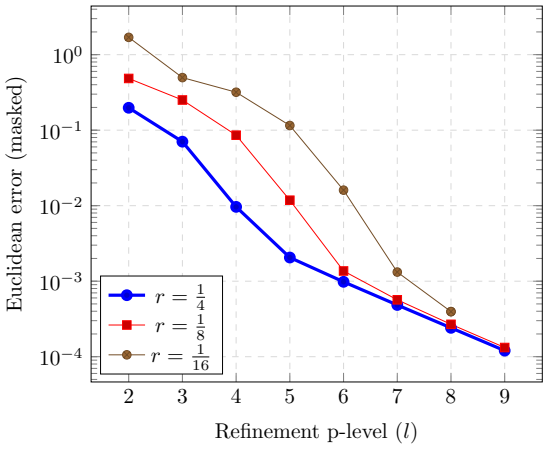


(a) Euclidean error (masked) for the velocity **u** (Dirac force inside a tetrahedron).

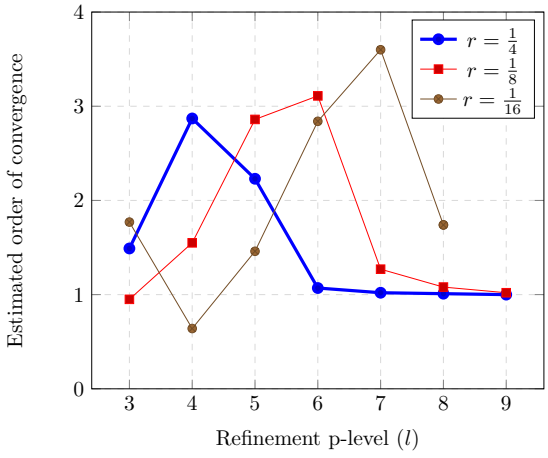


(b) Estimated order of convergence for the velocity **u** (Dirac force inside a tetrahedron).

Figure 6.10: Masked Euclidean error and estimated order of convergence for the velocity **u** of the Stokes problem with a Dirac force in x-direction inside a tetrahedron.

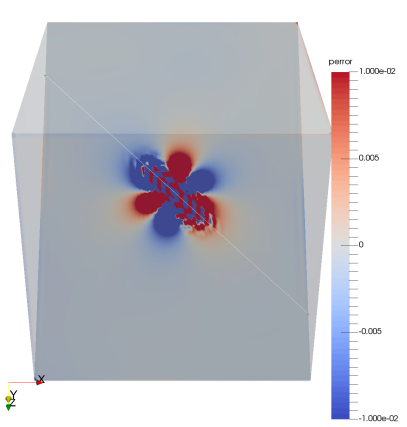


(a) Euclidean error (masked) for the pressure p (Dirac force inside a tetrahedron).

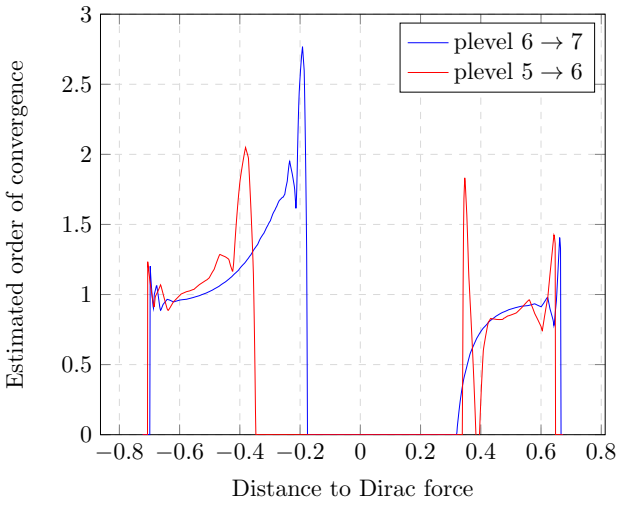


(b) Estimated order of convergence for the pressure p (Dirac force inside a tetrahedron).

Figure 6.11: Masked Euclidean error and estimated order of convergence for the pressure p of the Stokes problem with a Dirac force in x-direction inside a tetrahedron.



(a) The equivalent diagonal line to 6.5a through the Dirac force inside a tetrahedron. The line is plotted over the pressure error on plevel 6.



(b) Estimated order of convergence over line, oscillations around the Dirac force are masked out.

Figure 6.12: Estimated order of convergence for an inside tetrahedron equivalent line as through Dirac measure as specified in Equation 6.3.

Again, we see the oscillation patterns in these figures, especially in Figure 6.13c where the oscillations are inside an elliptical shaped region. The estimated order of convergence over the line in this plane through the oscillations are shown in Figure 6.12. As expected also in this line we don't recover the second order convergence in the far field.

If we take a closer look to the color plots of the pressure error, we see a slight bias to negative error values. This does suggest that we may violate our constraint to make the reconstructed pressure unique and we need to shift the reconstructed pressure in some sense.

Because the Stokes system with pure Dirichlet boundary conditions is singular the pressure is determined up to a constant. To make the pressure unique we included the constraint

$$\int_\Omega p \, d\Omega = 0, \tag{6.4}$$

on the pressure.

But because the singularity in this experiment is not on a vertex but inside an element we make an error in reflecting the exact solution and its peak on a discrete grid. This will result in a exact pressure on a discrete mesh and a numerical reconstruction on a discrete mesh that will not satisfy

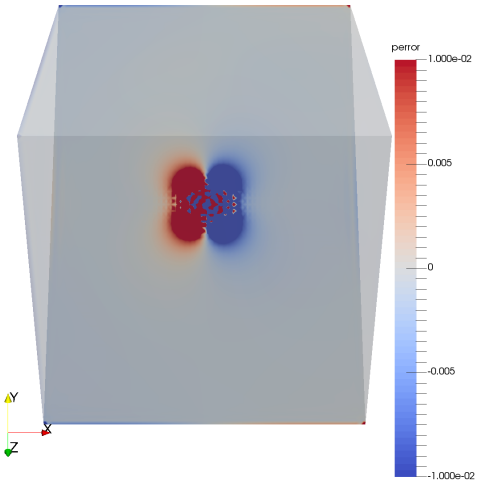
$$\sum_{i=1}^{n_e^p} \tilde{p}_i^h = 0, \quad \text{and} \quad \sum_{i=1}^{n_e^p} p_i^h = 0,$$

where $\tilde{p}^h$ is the projection of the exact solution p on a mesh with mesh size h.

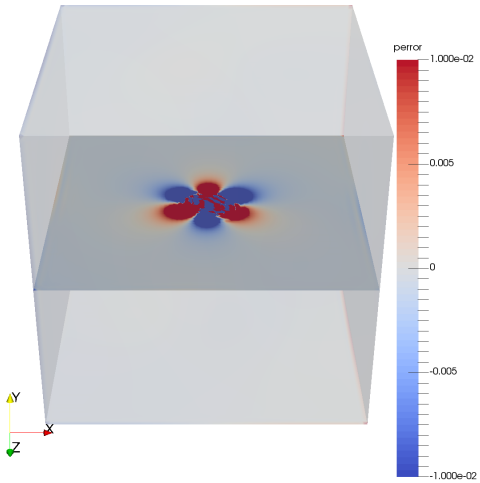
If we compute the difference

$$\Delta p = \sum_{i=1}^{n_e^p} \left(p_i^h - \tilde{p}_i^h \right),$$

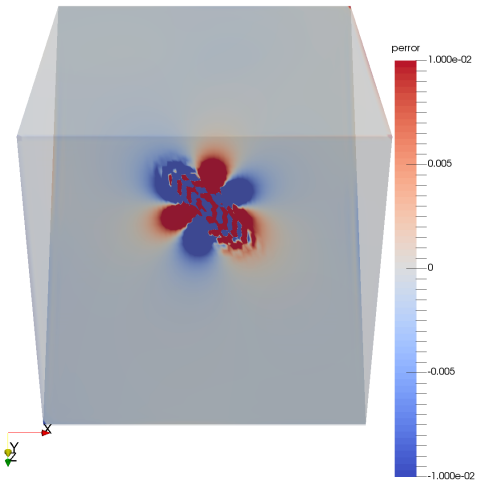
on the different reconstructed pressure levels between the "integrated" exact solution and numerical reconstruction we get results as shown in Figure 6.14 and Table 6.8. As we can see from the table, Δp decays only with $\mathcal{O}(h)$.



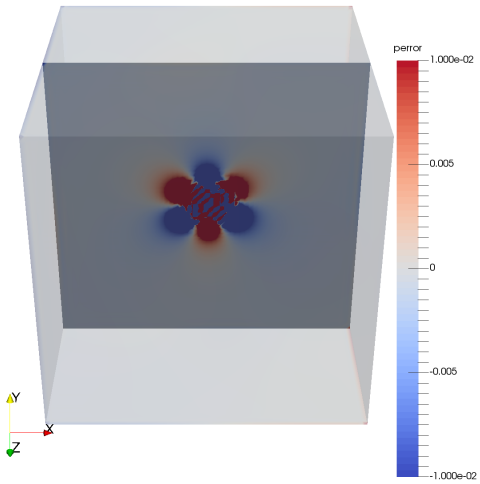
(a) Error in reconstructed pressure on slice $\mathbf{n} = (0, 1, 1)$.



(b) Error in reconstructed pressure on slice $\mathbf{n} = (0, 1, 0)$.



(c) Error in reconstructed pressure on slice $\mathbf{n} = (0, 1, -1)$. Note: cube is rotated w.r.t. the other views such that we have a clear view on this slice.



(d) Error in reconstructed pressure on slice $\mathbf{n} = (0, 0, 1)$.

Figure 6.13: Color plots of the error in reconstructed pressure of some interesting slices through the domain on plevel 6. The slices go through the Dirac force inside the tetrahedron on level 7 as specified by Equation 6.3. The normals are given per plot. Color scale range for all figures: $[-10^{-2}, 10^{-2}]$.

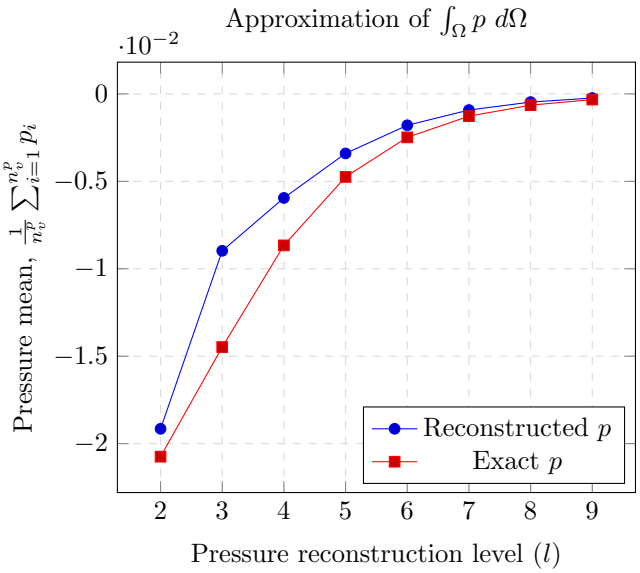


Figure 6.14: "Integrated" exact pressure and numerical reconstructed pressure for different pressure refinement levels.

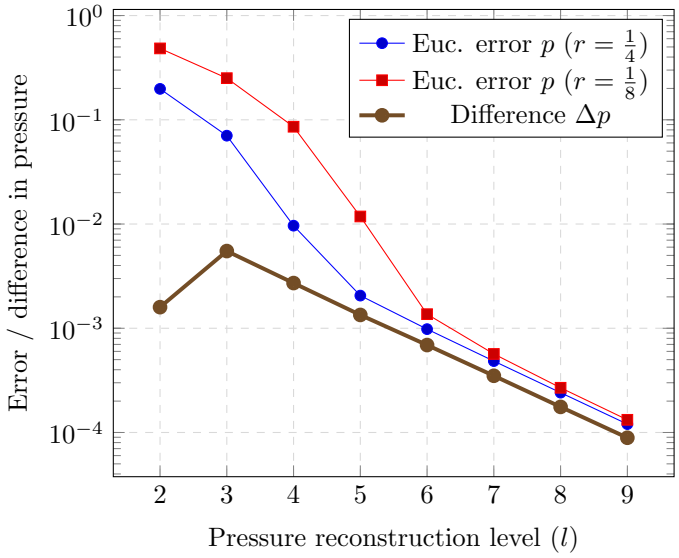
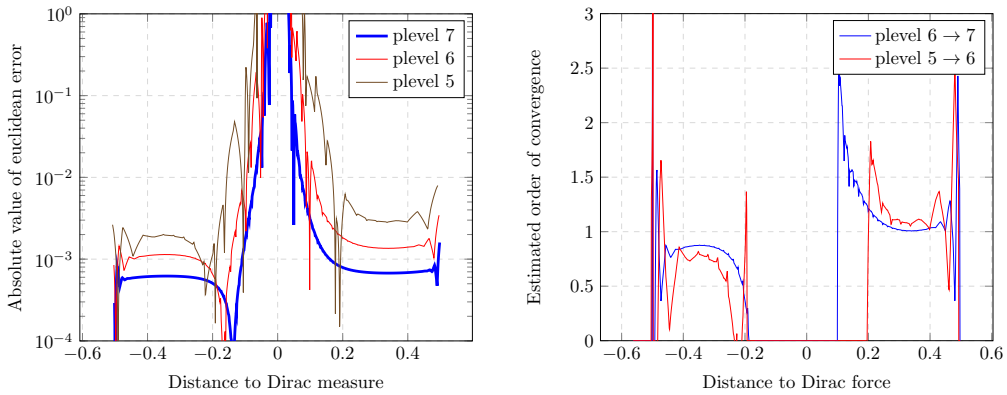


Figure 6.15: Euclidean error for pressure seems to be bounded by the difference Δp.

Pressure level	p **mean**	$\tilde{p}$ **mean**	Δp	**eoc**
2	$-1.96 \cdot 10^{-2}$	$-2.07 \cdot 10^{-2}$	$1.594 \cdot 10^{-3}$	
3	$-8.97 \cdot 10^{-3}$	$-1.45 \cdot 10^{-2}$	$5.503 \cdot 10^{-3}$	-1.79
4	$-5.95 \cdot 10^{-3}$	$-8.67 \cdot 10^{-3}$	$2.714 \cdot 10^{-3}$	1.02
5	$-3.40 \cdot 10^{-3}$	$-4.75 \cdot 10^{-3}$	$1.342 \cdot 10^{-3}$	1.02
6	$-1.80 \cdot 10^{-3}$	$-2.49 \cdot 10^{-3}$	$6.90 \cdot 10^{-4}$	0.96
7	$-9.22 \cdot 10^{-4}$	$-1.27 \cdot 10^{-3}$	$3.50 \cdot 10^{-4}$	0.98
8	$-4.67 \cdot 10^{-4}$	$-6.43 \cdot 10^{-4}$	$1.76 \cdot 10^{-4}$	0.99
9	$-2.35 \cdot 10^{-4}$	$-3.24 \cdot 10^{-4}$	$8.9 \cdot 10^{-5}$	0.98

Table 6.8: Mean of reconstructed pressure p and exact pressure $\tilde{p}$. Also difference Δp and the estimated order of convergence (eoc) of Δp are printed.



(a) Absolute value of Euclidean error for different pressure levels.

(b) Estimated order of convergence for different pressure levels.

Figure 6.16: Plots of error over a line and the estimated order of convergence for a line parallel to the Dirac force (i.e. from $x = 0$ to $x = 1$) and through the Dirac force on the different refinement levels.

If we plot the difference Δp against the Euclidean error for the pressure from Figure 6.11a it seems indeed that the pressure is bounded by an error that decays linearly with h, as shown in Figure 6.15.

One idea to circumvent this issue is to shift $\tilde{p}$ and p as the exact and numerical solution such that both have an average of zero which reflects the constraint of Equation 6.4. But this strategy will spoil down the convergence results even further because we have oscillations in the reconstructed pressure. And because the Dirac force is not on a vertex the oscillations are not symmetric anymore which means that the oscillations will contribute to the average value of the pressure.

Another possibility is that the assumption that errors on the boundary in the Stokes system with purely Dirichlet boundary conditions don't propagate into the domain is false when a Dirac force is not at the center of the domain.

For example, if we plot the error over a line parallel to the Dirac force (i.e. from $x = 0$ to $x = 1$) through the Dirac force we get error plots over these lines as shown in Figure 6.16a. Also there we see an estimated order of convergence that goes to 1, but before it starts to oscillate the estimated order of convergence increases to some higher value.

Further analysis of the impact of violation of conservation of mass on the boundary in the Stokes system on a cube could give insight whether this is an artifact from boundary errors or due to something else.

6.1.5 Double Diracs forces along a line

By linearity of the Stokes problem we can add multiple Dirac forces in the domain and should be able to reconstruct a linear combination of Stokeslets. In this experiment we will put two Dirac forces in opposite direction on a line and move the Dirac forces towards each other. So by reducing the distance between the two forces we approximate a Stresslet[2].

The Dirac forces move in the x-direction towards the center $x = \frac{1}{2}$ over the line $y = z = \frac{1}{2}$.

For this experiment we will refine the unit cube 7 and 8 times and the error of the reconstructed velocity $\mathbf{u}$ and pressure p are shown in Figure 6.17a. From the errors on this two refinement levels we can compute an estimated order of convergence, which is shown in Figure 6.17b.

To measure the error we will use the linear combination of the two exact solutions of the Stokeslet as exact solution for the Stresslet. As we can see from the error in the velocity $\mathbf{u}$ this assumption is indeed true independent of the quality of the approximation of the Stresslet by

[2]The Stresslet can for example be used to model micro swimmers such as bacteria.

a double Stokeslet. The exact pressures will cancel out each other and we need higher order approximations for the exact pressure if the distance between the two forces gets smaller than the mesh size.

The region where the distance hits the mesh size $h \approx \frac{1}{2^l}$, where l is the refinement level are marked with dashed lines. When the distance between the two Dirac forces is larger than the mesh size on a given level both Dirac forces will have an independent support on that level. As soon as the distance hits the refinement level on which the pressure is solved the Dirac forces don't have independent support anymore and because the forces are in opposite direction they start to cancel out each other on the level where the pressure is solved.

On the velocity level we can use the interpolation as a spreading of the Dirac force over all the vertices of the element that contains the Dirac force. Therefore, we always have a non overlapping part of the support of a single Dirac force as long as the two Dirac force don't coincide in one point. This is also visible in the decay of the error in the reconstructed velocity even on shorter distances as the mesh size.

Based on the results for a single Dirac force we could expect that the estimated order of convergence between refinement level 7 and 8 should be close to the order of convergence that we should recover. From these two refinement levels we can compute the estimated order of convergence. As we can see in Figure 6.17b for the velocity we are indeed able to get second order convergence almost independent from the relation between mesh side and distance of the Dirac forces.

Because the velocity is recovered with a high quality we could expect that the pressure is also recovered with a reasonable quality. But because of the assumption, that the exact pressure for a Stresslet is the linear combination of the two pressures for a Stokeslet is invalid, we cannot quantify the quality of the reconstructed pressure.

6.1.5.1 Optimizations

As we have seen in Figure 6.17a the error in the reconstructed pressure increases if the distance between two Dirac forces become smaller than the mesh side.

One possibility to also get a good reconstruction for the pressure could be to change to a PSPG stabilization where we reconstruct the pressure on the same refinement level as the velocity. Reconstructing the pressure on the same level as the velocity (and the Dirac force) should not lead to the loss of interpolation by the restriction operator.

Another possibility is changing to a mesh independent implementation of a Dirac force such that we can circumvent the restriction operator and recompute the restricted Dirac force on a coarser mesh, which allows us to get a correctly interpolated Dirac force over the vertices of the element that contains the Dirac force.

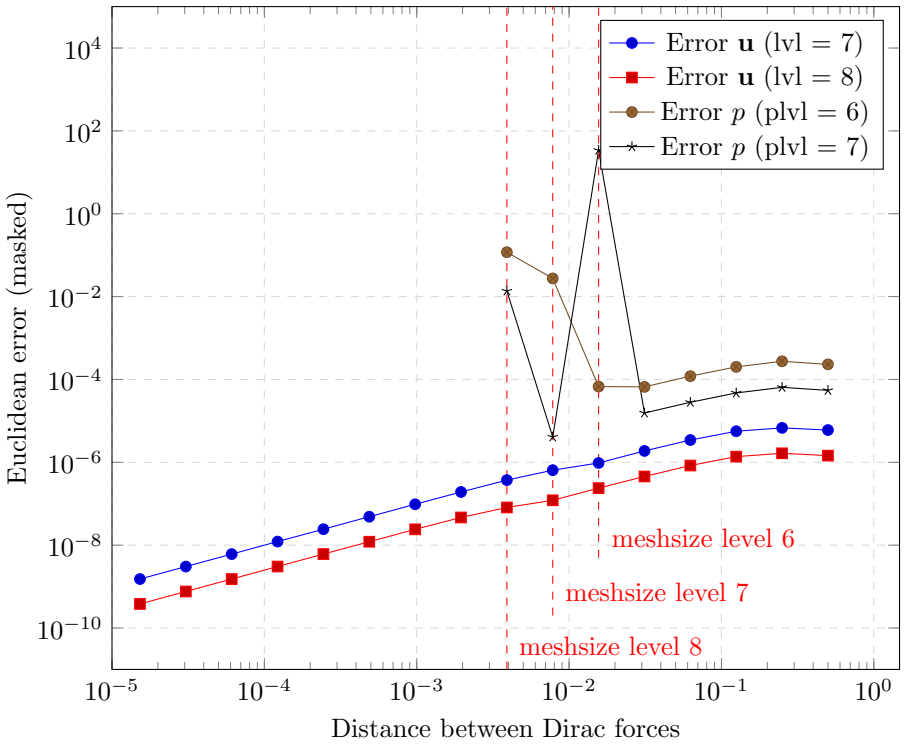
6.2 Numerical experiments $\mathbb{P}_2/\mathbb{P}_1$ finite elements

To solve the Stokes system with HHG we used $\mathbb{P}_1$-iso-$\mathbb{P}_2$ elements for the velocity $\mathbf{u}$ and $\mathbb{P}_1$ elements for the pressure p. We did this because with the HHG framework we are only able to solve using linear elements on a given refinement level. In this section we will solve the Stokes system on a finite element mesh with $\mathbb{P}_2$ elements for the velocity $\mathbf{u}$ and $\mathbb{P}_1$ elements for pressure p.

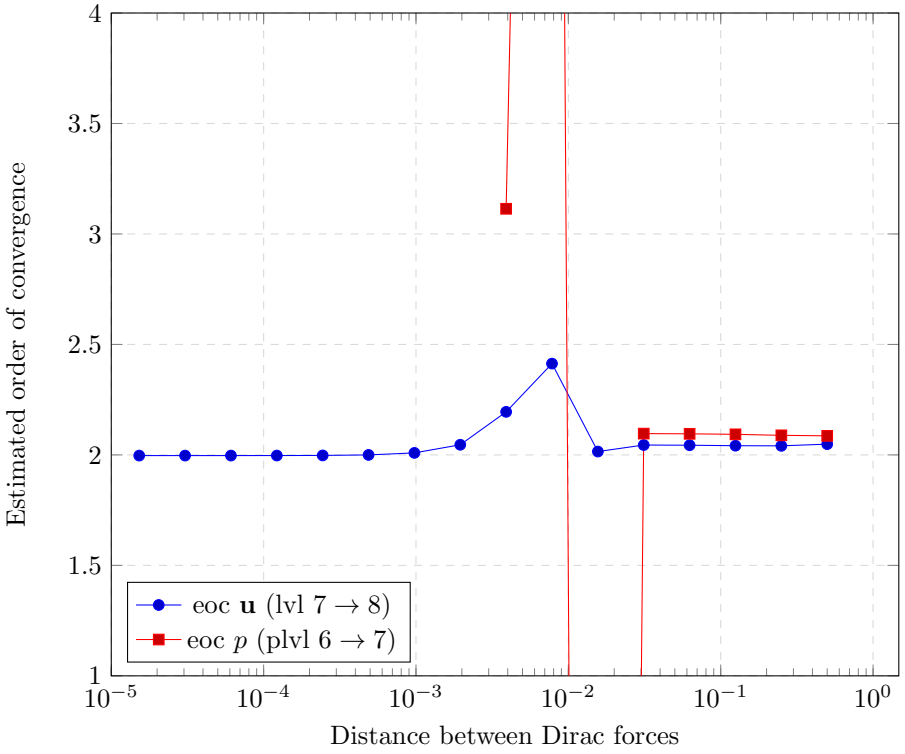
For the $\mathbb{P}_2/\mathbb{P}_1$ discretization we will use a FEniCS [24] experiment on the unit cube. The unit cube is uniformly refined in 16 subcubes in each direction and all subcubes consist of 6 tetrahedra. We will use a direct solver to solve the Stokes system.

The Dirac force is located at the center of the domain and pointing into the x-direction. The reconstructed pressure over the line $\left(0, \frac{1}{6}, \frac{1}{6}\right) \to \left(0, \frac{5}{6}, \frac{5}{6}\right)$ is shown in Figure 6.18.

As we can see from the plot we still have oscillations in the pressure but the magnitude of the oscillations is a bit smaller.



(a) Euclidean error (masked) for two Dirac forces in the two x-directions with decreasing distance on the line $y = z = \frac{1}{2}$.



(b) Estimated order of convergence for two Dirac forces in the two x-directions with decreasing distance on the line $y = z = \frac{1}{2}$.

Figure 6.17: Euclidean error and estimated order of convergence for the velocity **u** and pressure p for two Dirac forces in opposite direction moving towards each other around the center of the unit cube.

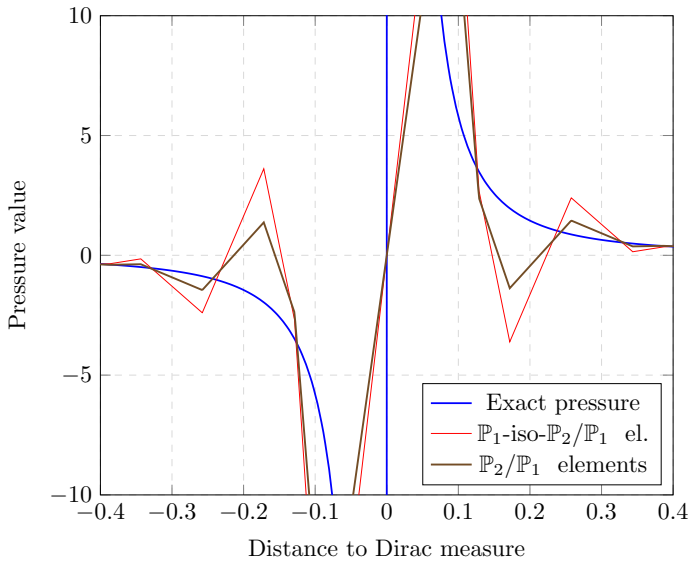


Figure 6.18: Comparison of reconstructed pressure over line $\left(0, \frac{1}{6}, \frac{1}{6}\right) \rightarrow \left(0, \frac{5}{6}, \frac{5}{6}\right)$ through the Dirac force for $\mathbb{P}_1$-iso-$\mathbb{P}_2/\mathbb{P}_1$ elements from HHG and $\mathbb{P}_2/\mathbb{P}_1$ elements from FEniCS.

6.3 Numerical experiments MAC discretization

In stead of using a finite element discretization of the Stokes problem we can also use a finite volume or finite difference discretization. To get a stable pressure in the Stokes problem we will use the Marker-and-Cell (MAC) discretization [16] where we put the unknowns for the velocity $\mathbf{u}$ in the x-, y-, z-direction and the pressure p on different locations of the grid, as shown in Figure 6.19.

We will discretize the unit cube using a MAC discretization with a uniform mesh of mesh size $h = \frac{1}{16}$. The Dirac force is located in the x-direction close to the center but coinciding with a location where we have a velocity unknown in the x-direction. For this experiment we will use a GMRES [28] with solver to solve the Stokes system. An impression of the reconstructed pressure over the plane $z \approx \frac{1}{2}$ which goes through the Dirac force is shown in Figure 6.20.

Also for the MAC discretization we can plot the pressure over a line through the Dirac force. We use a line through the Dirac force equivalent to the line $l : \left(0, \frac{1}{6}, \frac{1}{6}\right) \rightarrow \left(0, \frac{5}{6}, \frac{5}{6}\right)$ in the case that the Dirac force would be in the center of the domain. The plot of the pressure for mesh size $h = \frac{1}{16}$ is shown in Figure 6.21.

For comparison we added also the two finite element reconstructions for the pressure. As already

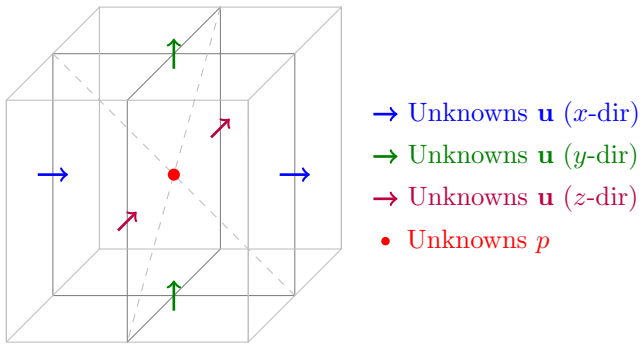
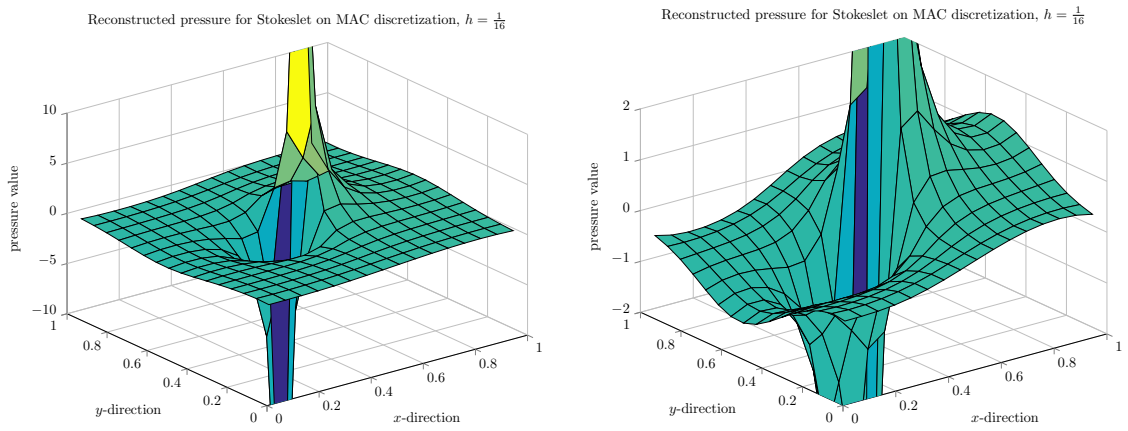


Figure 6.19: Location of $\mathbf{u}$ and p unknowns of MAC discretization.



(a) Overall view of pressure over plane

(b) Detail view of pressure over plane

Figure 6.20: Plots of the reconstructed pressure for the Stokeslet on a MAC-discretization over the plane $z \approx 0.5$ which goes through the Dirac force on mesh size $h = \frac{1}{16}$.

expected from the pressure plots over the plane we don't see oscillations in the reconstructed pressure on a MAC discretization. So it seems that the oscillations in the finite element solutions has something to do with the irregularity of tetrahedral elements.

6.3.1 Side remark

Currently we looked only into the reconstructed pressure of the MAC discretization. Because the unknowns for the velocity are spread over the boundaries of a volume we need to interpolate the velocity if we want to get a velocity in all directions at a certain location. In the setup of the Stokeslet we have a strong force at a certain location where at most one velocity component can be specified. In our experiment we place the Dirac point at a location of a velocity unknown in the x-direction.

In Figure 6.22 we plot the reconstructed and exact solution for the velocity on the two planes $z \approx \frac{1}{2}$ and $x = \frac{1}{2}$ through the Dirac force. The artifact from the spreading of the velocity components on three different locations is clearly visible in a 'clockwise rotation' of the velocity in the plane perpendicular to the Dirac force as shown in Figure 6.22c. Note that we imposed homogeneous Diriclet boundary conditions for the velocity on this MAC discretization experiment which explains the circulation that are visible in Figure 6.22a.

6.4 Conclusion and optimizations

For the Stokes problem we have both unknowns for the velocity $\mathbf{u}$ and for the pressure p. In all experiments on a finite element mesh with $\mathbb{P}_1$-iso-$\mathbb{P}_2/\mathbb{P}_1$ elements we were able to recover a clear second order convergence for the velocity $\mathbf{u}$.

When we put the Dirac force on a vertex then the estimated order of convergence for the pressure will also go clearly to two. But when we fix the location of the Dirac force to the center of a tetrahedron on the finest level we are only able to recover first order convergence for the pressure. Currently it is not clear why we don't see second order convergence. Maybe one needs to take a closer look into the influences for the error on the boundary due to a violation of conservation of mass in the Stokes system itself.

By linearity we would expect to get similar results when we add a second Dirac force. For the velocity $\mathbf{u}$ we are indeed able to move two Dirac forces close to each other and still recover good reconstructions and second order convergence. For the pressure we are only able to get good reconstructions if the distance between the Dirac forces is larger than the mesh size. One possibility

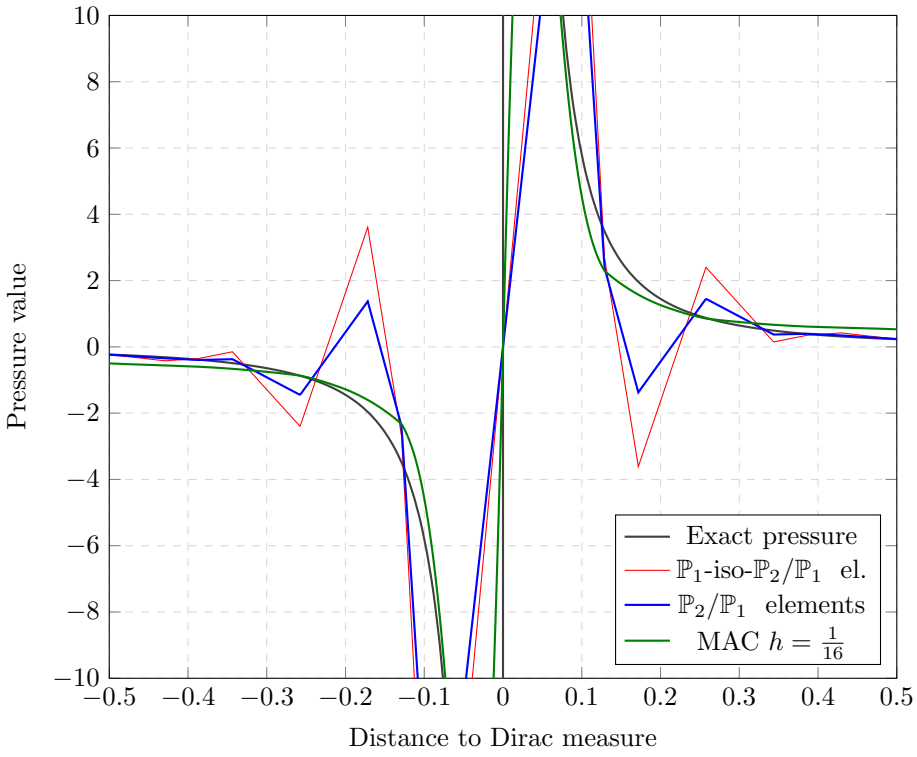
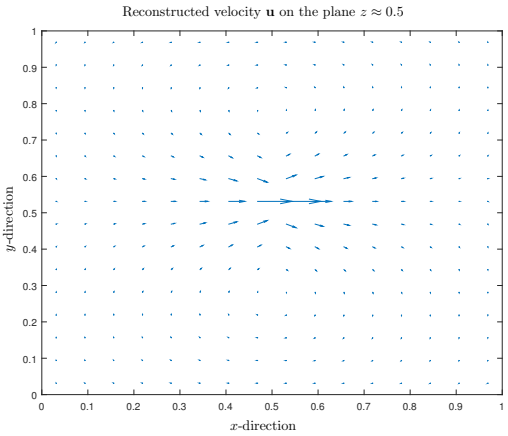


Figure 6.21: Comparison of reconstructed pressure over an equivalent line $l : \left(0, \frac{1}{6}, \frac{1}{6}\right) \to \left(0, \frac{5}{6}, \frac{5}{6}\right)$ through the Dirac force for $\mathbb{P}_1$-iso-$\mathbb{P}_2/\mathbb{P}_1$ and $\mathbb{P}_2/\mathbb{P}_1$ finite elements and also a MAC discretization with mesh side $h = \frac{1}{16}$.

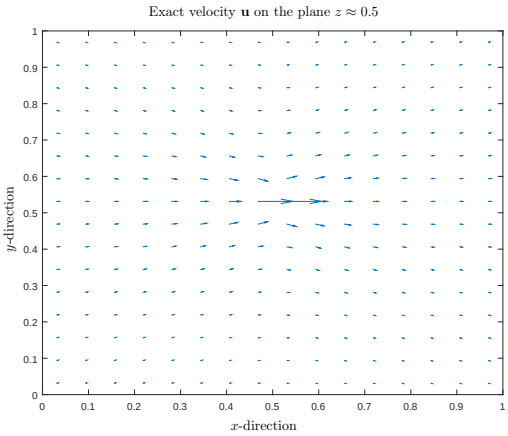
for this problem is that we use a restriction operator where interpolation information of opposite pointing Dirac forces will be lost. As pointed out in Section 6.1.5.1 we could try to optimize this with a PSPG stabilization or a Dirac force implementation where we recompute the interpolation of the Dirac force on a coarser level instead of using a restriction operator.

Similar to what we have seen in the reconstruction of the Poisson equation, also for the Stokes problem we see oscillations in the reconstructed pressure. Currently the oscillations don't influence the convergence rate for the velocity or the pressure.

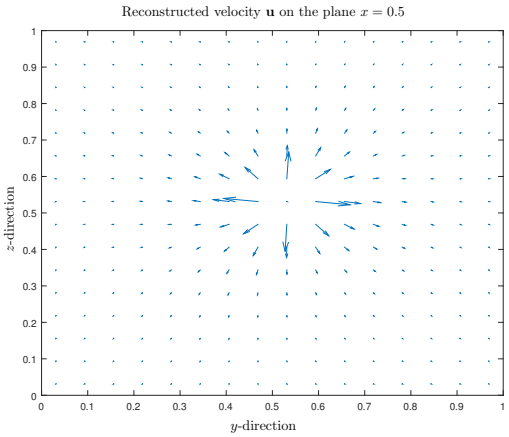
A comparison with $\mathbb{P}_2/\mathbb{P}_1$ finite elements shows that we have also oscillations in the pressure but the magnitude of the oscillations are smaller. A comparison with the Stokes problem on a MAC discretization shows that the oscillations are not directly related to the singularity in the Stokes system but that this oscillations will be an artifact of the used finite elements.



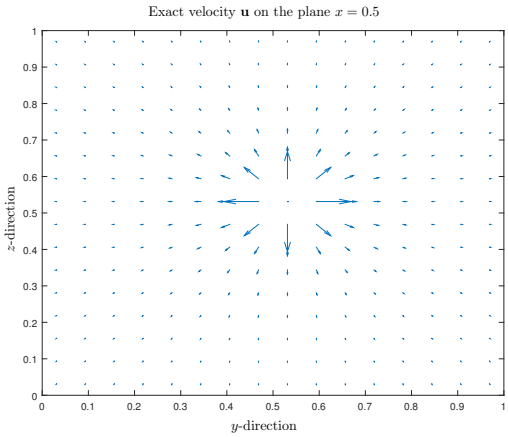
(a) Reconstructed velocity on the plane $z \approx \frac{1}{2}$.



(b) Exact velocity on the plane $z \approx \frac{1}{2}$.



(c) Reconstructed velocity on the plane $x = \frac{1}{2}$.



(d) Exact velocity on the plane $x = \frac{1}{2}$.

Figure 6.22: Plots of the reconstructed velocity $\mathbf{u}$ of the Stokeslet on a MAC discretization and the exact solution of the Stokeslet on the same planes.

Chapter 7

Regularizing Stokeslet

As we have seen in Chapter 6 we are able to recover the Stokeslet with second order convergence in the far field. In the case the Dirac force is located on a vertex also for the pressure we obtained second order convergence. But we observed oscillations in the reconstructed pressure (and also in the velocity). In the numerical reconstruction we used $\mathbb{P}_1$-iso-$\mathbb{P}_2$ elements for the velocity and linear elements for the pressure as approximation for the $\mathbb{P}_2$-$\mathbb{P}_1$ finite elements, which should be stable [4, 7]. Also a numerical experiment with FEniCS using $\mathbb{P}_2$-$\mathbb{P}_1$ finite elements and a direct solver results in solutions with oscillations in the pressure.

We saw that if we use a Marker-and-Cell (MAC) discretization [16] for the Stokes system we get a smooth solution. So, the oscillations seems to be a feature of the finite element discretization of the domain. Therefore, in this chapter we will try to regularize the Stokeslet such that we can get rid of the oscillations as we have seen in chapter 6.

7.1 Regularized Stokeslet

We can regularize the Stokeslet to get rid of the singularities in the system by using the method of regularized Stokeslets [6]. Cortez presents the regularized Stokeslet where he replaced the Dirac force by a cutoff function $\varphi_\epsilon(\mathbf{x})$ with the property $\int_\Omega \varphi_\epsilon(\mathbf{x}) d\Omega = 1$. We can think of the $\varphi_\epsilon(\mathbf{x})$ as a radial symmetric, smooth approximation of the Dirac distribution concentrated at $\mathbf{x} = 0$. The spreading is controlled by a parameter ϵ.

As cutoff function we will use

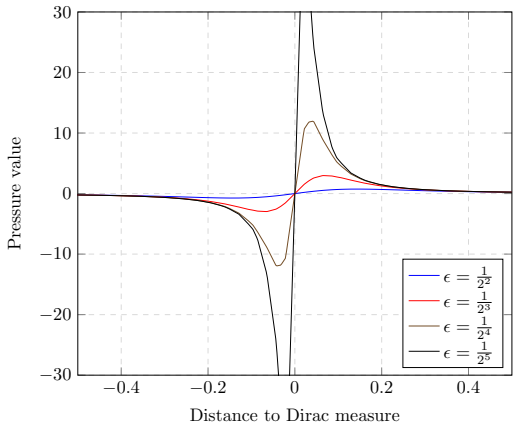
$$\varphi_\epsilon(\mathbf{x} - \mathbf{x}_0) = \frac{15\epsilon^4}{8\pi(r^2 + \epsilon^2)^{\frac{7}{2}}}, \tag{7.1}$$

where $r := \|\mathbf{x} - \mathbf{x}_0\|$. Using this cutoff function, the Stokeslet problem will be regularized to the regularized Stokeslet problem:

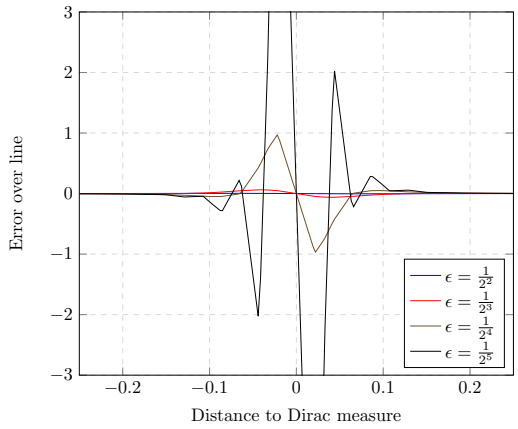
$$\begin{aligned} -\mu\Delta\mathbf{u} + \nabla p &= \varphi_\epsilon(\mathbf{x} - \mathbf{x}_0)\mathbf{f} \\ \nabla \cdot \mathbf{u} &= 0. \end{aligned} \tag{7.2}$$

And the exact solution for this system is given by

$$\begin{aligned} \mathbf{u}(\mathbf{x} - \mathbf{x_0}) &= \mathbf{F}_\epsilon(\mathbf{x} - \mathbf{x_0})\mathbf{f} \\ \mathbf{p}(\mathbf{x} - \mathbf{x_0}) &= \mathbf{P}_\epsilon(\mathbf{x} - \mathbf{x_0})\mathbf{f}, \end{aligned} \tag{7.3}$$



(a) Reconstructed pressure for the regularized Stokeslet over the line $l : \left(0, \frac{1}{6}, \frac{1}{6}\right) \to \left(1, \frac{5}{6}, \frac{5}{6}\right)$.



(b) Error of the reconstructed pressure for the Regularized Stokeslet over the line $l : \left(0, \frac{1}{6}, \frac{1}{6}\right) \to \left(1, \frac{5}{6}, \frac{5}{6}\right)$.

Figure 7.1: Reconstructed pressure and error of the Regularized Stokeslet over the line $l : \left(0, \frac{1}{6}, \frac{1}{6}\right) \to \left(1, \frac{5}{6}, \frac{5}{6}\right)$ with different regularization parameter values ϵ. Only the minimal oscillatory pressures are shown.

where $\mathbf{F}_\epsilon(\mathbf{r})$ and $\mathbf{P}_\epsilon(\mathbf{r})$ are given by:

$$\mathbf{F}_\epsilon(\mathbf{x} - \mathbf{x_0}) = \mathbf{F}_\epsilon(\mathbf{r}) = \frac{1}{8\pi\mu}\left(\frac{\left(\|\mathbf{r}\|^2 + 2\epsilon^2\right)\mathbf{I}}{\left(\|\mathbf{r}\|^2 + \epsilon^2\right)^{\frac{3}{2}}} + \frac{\mathbf{r}\otimes\mathbf{r}}{\left(\|\mathbf{r}\|^2 + \epsilon^2\right)^{\frac{3}{2}}}\right)$$

$$\mathbf{P}_\epsilon(\mathbf{x} - \mathbf{x_0}) = \mathbf{P}_\epsilon(\mathbf{r}) = \frac{\mathbf{r}}{8\pi}\left(\frac{2\|\mathbf{r}\|^2 + 5\epsilon^2}{\left(\|\mathbf{r}\|^2 + \epsilon^2\right)^{\frac{5}{2}}}\right).$$

The analysis of the accuracy and performance of the regularized Stokeslet is given by Cortez [6].

7.1.1 Numerical results regularized Stokeslet

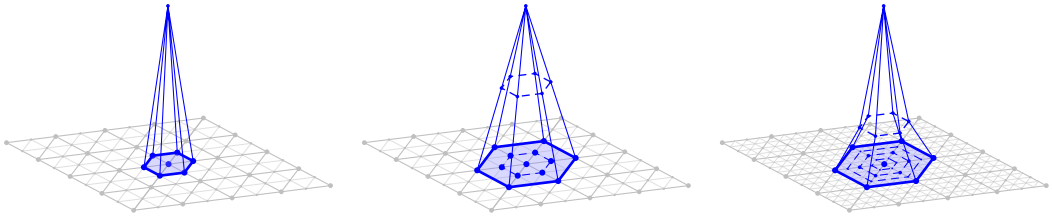
In this experiment we will setup the regularized Stokeslet problem to solve its equivalent problem as discussed in section 6.1.2 for different ϵ. So we will use $\mathbb{P}_1$-iso-$\mathbb{P}_2$ elements for the velocity $\mathbf{u}$ and $\mathbb{P}_1$ elements for the pressure p. Further we will use the Schur complement CG solver for the Stokes system with the stopping criteria that the relative residual should be smaller than the tolerance 10^{-8}.

For the simulation, we will concentrate the cutoff function on the center vertex $\mathbf{x}_0 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$. Further, we will fix the refinement level to $l = 7$ (so we will solve the pressure on level 6), such that we can get visually also a feeling about the influences of the regularization on the oscillations. We will vary the regularization parameter $\epsilon \in \left\{\frac{1}{2^2}, \frac{1}{2^3}, \ldots, \frac{1}{2^7}\right\}$. The reconstructed pressure and the error over the line $l : \left(0, \frac{1}{6}, \frac{1}{6}\right) \to \left(1, \frac{5}{6}, \frac{5}{6}\right)$ are shown in Figure 7.1.

As we can see we start to have oscillations from $\epsilon = \frac{1}{2^5}$ in the reconstructed pressure on a mesh with a mesh size $h \sim \frac{1}{2^6}$. So, if we spread the Dirac pulse to a region with a bigger support it seems that we can get rid of the oscillations on this line.

7.2 Regularizing using tent approximations

Putting a Dirac distribution on a vertex will result in a tent function on the velocity refinement level, as shown in Figure 7.2a. Because we solve the pressure on level coarser it could be that the



(a) Tent function defined on the refinement level for the velocity.

(b) Tent function defined on the refinement level for the pressure.

(c) 'Quadratic' tent function.

Figure 7.2: Different tent functions in 2D on coarse and fine triangular meshes. Note that for representing a Dirac measure the tent need to have a volume equal to one. So, for representing a Dirac with these tent functions one needs to scale the tent by its volume to get a unit-volumed tent.

representation of the Dirac measure on the pressure level is not accurate enough. Therefore, we will approximate the Dirac measure with tent functions and a first ansatz will be to define the Dirac measure on the pressure level and prolongate the approximation to the velocity level. This will result in a tent approximation for the Dirac distribution with a 'radius' $2h$. The shape of the approximated Dirac on the velocity level is shown in Figure 7.2b.

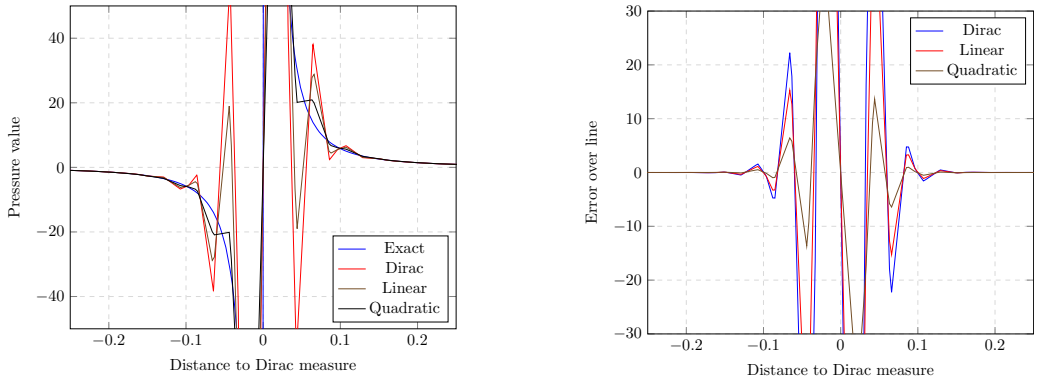
To prolongate the property

$$\int_\Omega \mathbf{f}\ d\Omega = 1$$

from level l to level $l+1$ we need to scale the tent function by a factor $\frac{1}{2^d}$ where $d \in \{1,2,3\}$ is the dimension of the problem.

Using this procedure, we can define also higher order tent functions. For the 'quadratic' tent function we define a tent function on level $l-2$ and level $l-1$. Next, we can prolongate both tent functions to velocity level l and use a linear combination of these two tent functions to construct a higher order tent function. Therefore, the 'quadratic' tent will have a 'radius' $4h$. An example of the used 'quadratic' tent function is shown in Figure 7.2c.

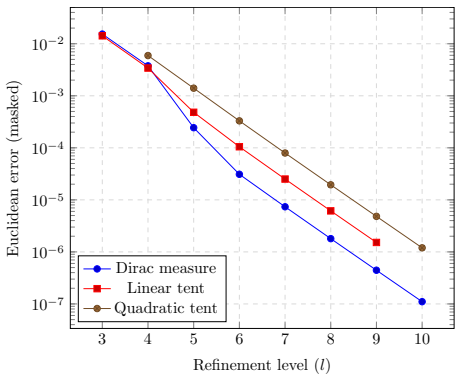
Because we introduced the tent functions to get rid of the oscillations on the line $\left(0, \frac{1}{6}, \frac{1}{6}\right) \to \left(0, \frac{5}{6}, \frac{5}{6}\right)$ we consider first the results for the Stokes system with the tent functions over this line. The reconstructed pressure and the error are shown in Figure 7.3.



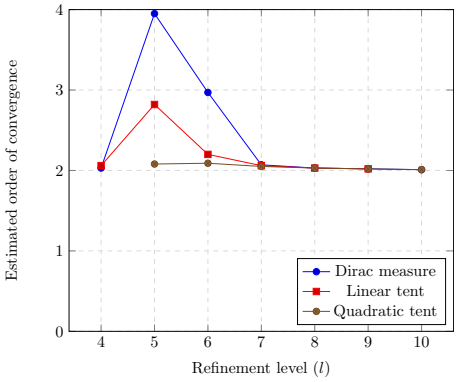
(a) Reconstructed pressure for the tent approximation for the Stokeslet over the line $l : \left(0, \frac{1}{6}, \frac{1}{6}\right) \to \left(1, \frac{5}{6}, \frac{5}{6}\right)$.

(b) Error of the reconstructed pressure for the tent approximation for the Stokeslet over the line $l : \left(0, \frac{1}{6}, \frac{1}{6}\right) \to \left(1, \frac{5}{6}, \frac{5}{6}\right)$.

Figure 7.3: Reconstructed pressure and error of the Stokeslet with tent approximations over the line $l : \left(0, \frac{1}{6}, \frac{1}{6}\right) \to \left(1, \frac{5}{6}, \frac{5}{6}\right)$ with linear and quadratic tent functions.

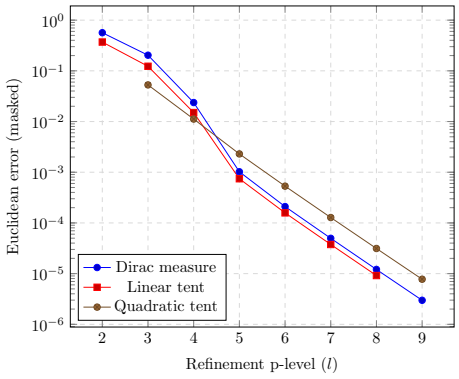


(a) Euclidean error (masked) for Dirac force in x-direction at the center vertex using tent approximations.

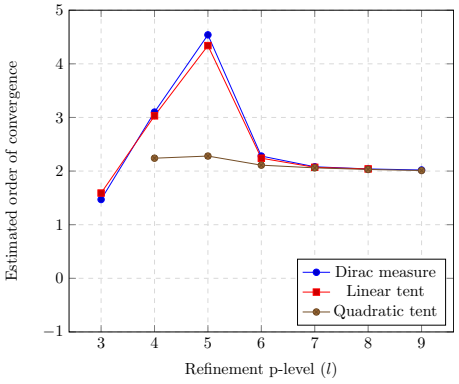


(b) Estimated order of convergence for Dirac force in x-direction at the center vertex using tent approximations.

Figure 7.4: Euclidean error and estimated order of convergence for the velocity $\mathbf{u}$ of the Stokes problem with a Dirac force in x-direction at the center vertex using tent approximations.



(a) Euclidean error (masked) for Dirac force in x-direction at the center vertex using tent approximations.



(b) Estimated order of convergence for Dirac force in x-direction at the center vertex using tent approximations.

Figure 7.5: Euclidean error and estimated order of convergence for the pressure p of the Stokes problem with a Dirac force in x-direction at the center vertex using tent approximations.

As we can see from these plots we are indeed able to reduce the amplitude of the oscillations. Especially the 'quadratic' tent is able to suppress the oscillations in the pressure on the region where the oscillations start. If we consider the error then we see also there that the amplitude of the error becomes lower, but we are not able to remove the oscillations.

If we compute the error and the estimated order of convergence for the linear and quadratic tent approximation with $r = \frac{1}{4}$ we get results for the velocity $\mathbf{u}$ as shown in Figures 7.4a and 7.4b. The results for the pressure p are shown in Figures 7.5a and 7.5b. For reference purposes also the results as found in Section 6.1.2 with radius $r = \frac{1}{4}$ are plotted.

As we can see, also with the tent approximations we are able to recover the Stokeslet with second order convergence. The linear tent approximation gives for the pressure a slightly better solution as the results with the Dirac measure itself. The additional spreading in the quadratic tent will reduce the accuracy of the reconstructed pressure slightly.

The results for the reconstructed velocity are a bit worse with the tent approximations. There we will lose up to one digit in the accuracy on a specific refinement level.

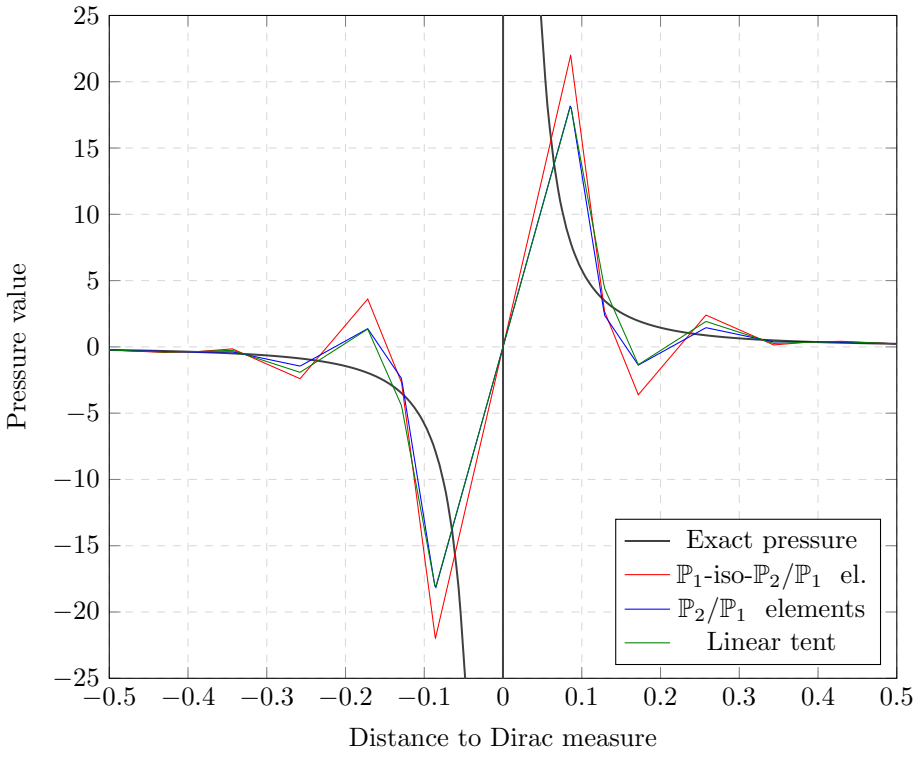


Figure 7.6: Comparison of reconstructed pressure using $\mathbb{P}_1$-iso-$\mathbb{P}_2/\mathbb{P}_1$ elements with Dirac force, a linear tent approximation and a pressure reconstruction using $\mathbb{P}_2/\mathbb{P}_1$ elements and a Dirac force.

7.2.1 Relation linear tent with $\mathbb{P}_2/\mathbb{P}_1$ finite elements

If we solve the Stokeslet using $\mathbb{P}_2/\mathbb{P}_1$ finite elements we actually solve a system where the Dirac force is located on the same level as the pressure and using the quadratic elements we have an higher degree of freedom for the velocity. So, the reconstructed pressure of the Stokeslet using a linear tent right hand side should be a better approximation for the pressure solution from $\mathbb{P}_2/\mathbb{P}_1$ finite elements.

If we compare the reconstructed pressure using a linear tent approximation with the reconstructed pressure of the $\mathbb{P}_2/\mathbb{P}_1$ finite elements of our FEniCS experiment, we see indeed that both reconstructions are very similar. In Figure 7.6 we show the reconstructed pressure over the line $\left(0, \frac{1}{6}, \frac{1}{6}\right) \to \left(0, \frac{5}{6}, \frac{5}{6}\right)$.

Chapter 8

Scalability experiments

The second order convergence for a Stokes solver on a finite element mesh for the Stokeslet as shown in Chapter 6 is one part to efficiently compute a numerical solution for a problem. To handle large scale problems, we need also a *scalable* algorithm which will allow us to use a lot of cores with a good parallel efficiency. Therefore, in this chapter we will show that we can efficiently solve the Stokeslet on large scale machines. Further, we will show, that we are able to scale the number of Dirac forces where we keep the concentration (i.e. the number of Dirac forces per volume) of the Dirac forces constant.

For the large scale experiments we will use SuperMUC Phase 2[1], a Petascale System located at the Leibniz-Rechenzentrum (Leibniz Supercomputing Centre) in Munich (Germany). The nodes contain two Intel Haswell processors with 14 cores per processor[2], so in total one node of SuperMUC has 28 cores.

8.1 Strong scaling inside a single node

To verify that we can effectively use all cores in a node we will start with a strong scalability experiment from one to 28 cores inside a single node. For the strong scalability experiment we will fix the problem setup and increase the number of used cores. From the obtained timings we can compute the speedup for a number of cores n as

$$S(n) = \frac{T_1}{T_n},$$

where T_1 is the runtime of the serial program and T_n the runtime of the program using n cores.

As domain we will choose a rectangular box with 7 cubes in the x-direction, 2 cubes in the y-direction and 2 cubes in the z-direction. All cubes are buildup out of 6 tetrahedra. This geometry allows us to equally distribute the $28 \times 6 = 168$ tetrahedra over 28 cores. The Dirac force is pointed in the x-direction. Further, we put the Dirac force in the center of the domain. The domain is 7 times uniformly refined and we will use 40 SchurCG iterations (i.e. we will use in total 40 Preconditioned CG iterations to solve the continuity equation and per 4 CG iterations we recompute the system matrix using the momentum equation and a multigrid algorithm).

The 168 tetrahedra can be equally distributed over $n \in \{1, 2, 3, 4, 6, 7, 8, 12, 14, 21, 24, 28\}$ cores, and the runtime of this strong scalability experiment in a single node is shown in Table 8.1 and Figure 8.1.

From these runtimes on multiple cores we can see that both the integration of the load vector and the SchurCG solver reduce the runtime on increasing number of cores. From the speedup, as shown in Figure 8.2a, and the parallel efficiency, as shown in Figure 8.2b we may conclude that we

[1] Details about the system can be found in Section B.1.

[2] Actually, these cores support also Simultaneous Multithreading but because solving a linear system will be mainly memory bounded multiple threads on one core will not increase the performance of a Stokes solver.

#Cores	Setup load (sec)			SchurCG solver (sec)
	x-dir	y-dir	z-dir	
1	946.3 sec	876.3 sec	876.4 sec	2084.0 sec
2	461.9 sec	442.7 sec	426.3 sec	1027.5 sec
3	338.6 sec	305.6 sec	305.1 sec	695.7 sec
4	236.4 sec	239.3 sec	220.4 sec	506.2 sec
6	187.9 sec	161.8 sec	161.7 sec	346.0 sec
7	148.6 sec	128.0 sec	125.5 sec	305.7 sec
8	133.4 sec	129.1 sec	129.0 sec	261.6 sec
12	86.8 sec	81.7 sec	80.7 sec	185.0 sec
14	82.1 sec	77.5 sec	77.4 sec	158.7 sec
21	50.2 sec	47.0 sec	47.8 sec	121.8 sec
24	48.6 sec	45.0 sec	45.0 sec	105.9 sec
28	35.4 sec	32.8 sec	35.6 sec	90.6 sec

Table 8.1: Runtime in seconds of strong scalability experiment reconstructing the Stokeslet inside one node of SuperMUC. Runtime is dominated by the 4 components: Setup load in three components and the time needed for the SchurCG Stokes solver.

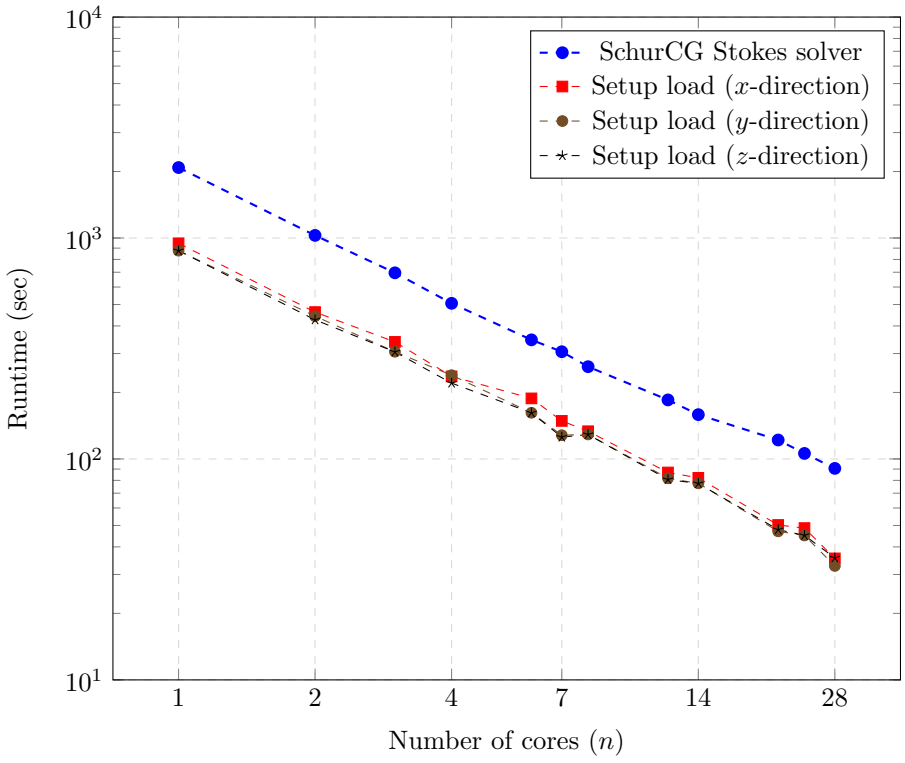


Figure 8.1: Runtime in seconds of strong scalability experiment reconstructing the Stokeslet inside one node of SuperMUC.

don't have completely linear speedup but still a parallel efficiency for the SchurCG Stokes solver of 82%.

If we take a closer look to the parallel efficiency of the SchurCG solver we see three regions where the parallel efficiency is approximately constant and we have two drops in parallel efficiency. From one to 7 cores the parallel efficiency stays at approximately 100% and from 7 to 11 cores we see a drop in parallel efficiency to 94%. This can be explained by the fact that the 14 cores of the used processor are clustered into two groups of 7 cores, called *cluster on die*. So, when we exceed the 7 cores communication between the two clusters is needed instead of only communication between cores on the same bus. This will have a slightly impact on the performance as we can see in the drop in the parallel efficiency.

At 14 cores we have one full socket and the bigger drop in performance when we go to two sockets can be explained by the communication that is needed between the cores on different sockets. The communication will be more expensive due to the fact that both sockets have non uniform memory access, i.e. form two NUMA domains. For the setup load we have an computational unbalance between the cores because only one core needs to integrate the single Dirac force. Further, the oscillations in the parallel efficiency for the setup load can be explained by how well the communication is balanced over the different clusters on the processors.

8.2 Weak Scalability of Stokes Solver

Instead of fixing the problem and increasing the number of cores we can also fix the problem size per core. In a weak scalability experiment we increase both the problem size (i.e. for the scalability of the Stokes solver we will increase the geometric domain) and the number of used cores. We can compute the parallel efficiency of a weak scalability experiment from n to m cores as

$$E_n(m) = \frac{T_n}{T_m},$$

where T_n is the runtime on n cores which we use as reference and T_m is the runtime on m cores for which we compute the parallel efficiency.

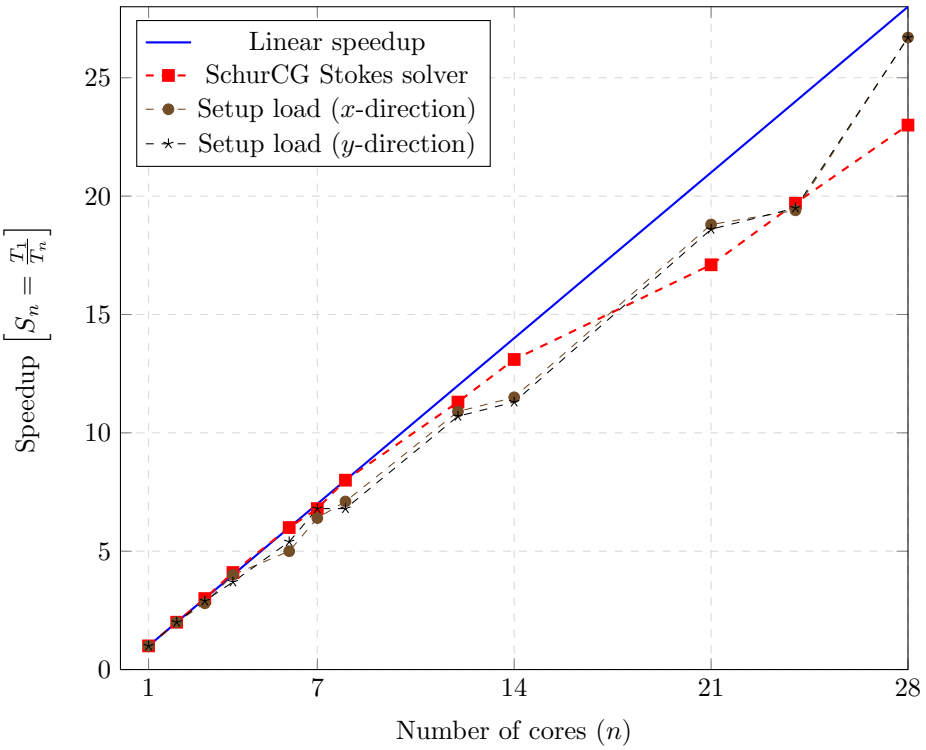
As we have seen in the strong scalability we can effectively use all cores in a node. So, for the weak scalability experiment we start the experiment on one node (i.e. 28 cores) with a geometric domain Ω that consists of $7 \times 8 \times 8$ cubes. This will give an equal distribution of macro elements over the cores, where each core will have 16 cubes or 96 macro elements.

To construct the finest level we will uniformly refine the mesh 5 times. By doubling the number of nodes we will alternately double the domain size in the x-, y- and z-direction. The Dirac force is pointed in the x-direction. Further, we put the Dirac force in the center of the domain and fix the number of SchurCG iterations to 40. The runtime for the weak scalability experiment is shown in Table 8.2 and Figure 8.3. From the runtime we can compute the parallel efficiency as shown in Figure 8.4.

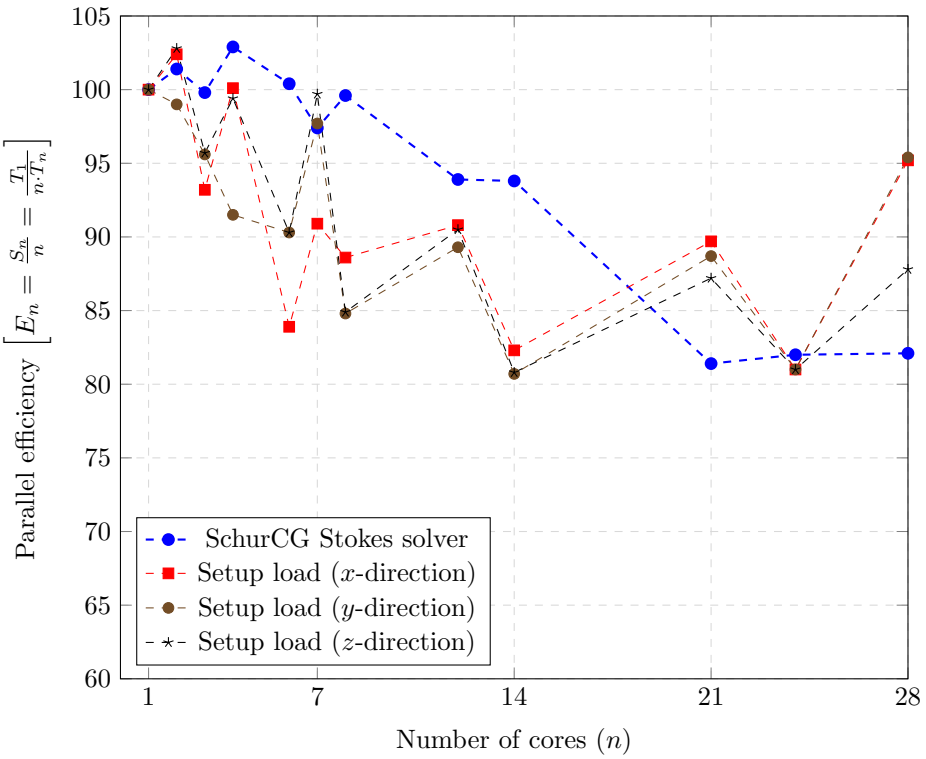
Because the main focus of this thesis is to solve fundamental solutions instead of parallel solving Stokes systems we skip the analysis and performance model for the SchurCG Stokes solver. An detailed analysis and performance model for the Stokes solvers as implemented in HHG can be found in [11, 12, 13]. As Figure 8.4 shows we are able to solve the Stokes system on an island of SuperMUC (i.e. 512 nodes with a total of 14,336 cores) with a parallel efficiency for the SchurCG solver of 73% compared to the case with one node of 28 cores.

8.2.1 Quality of reconstructed solution

One side remark which is not further explored in this thesis is how to fix the quality of a reconstructed stokeslet on an increasing domain. In the SchurCG solver we use a multigrid algorithm to solve the momentum equation. On the coarsest level of the multigrid solver we use a CG solver with a fixed number of iterations. As mentioned in [11] the CG solver on the coarsest level of a multigrid solver for a Poisson problem needs more iterations when the domain size increases.



(a) Speedup per component of the strong scalability experiment reconstructing the Stokeslet.



(b) Parallel efficiency per component of the strong scalability experiment reconstructing the Stokeslet.

Figure 8.2: Speedup and parallel efficiency of strong scalability experiment reconstructing the Stokeslet inside on node of SuperMUC.

#Nodes	#Cores	#Unknowns	Setup load (sec)			SchurCG (sec)
			x-dir	y-dir	z-dir	
1	28	$1.450 \cdot 10^7$	11.2 sec	10.0 sec	10.0 sec	49.2 sec
2	56	$2.907 \cdot 10^7$	11.0 sec	9.8 sec	9.9 sec	53.2 sec
4	112	$5.825 \cdot 10^7$	10.6 sec	9.4 sec	9.3 sec	56.4 sec
8	224	$1.167 \cdot 10^8$	10.6 sec	9.3 sec	9.3 sec	57.6 sec
16	448	$2.337 \cdot 10^8$	9.8 sec	8.7 sec	8.7 sec	61.6 sec
32	896	$4.679 \cdot 10^8$	10.4 sec	9.3 sec	9.3 sec	61.9 sec
64	1792	$9.366 \cdot 10^8$	10.9 sec	8.6 sec	8.7 sec	62.9 sec
128	3584	$1.874 \cdot 10^9$	9.7 sec	8.7 sec	9.0 sec	65.6 sec
256	7168	$3.750 \cdot 10^9$	9.9 sec	8.6 sec	8.8 sec	67.9 sec
512	14336	$7.505 \cdot 10^9$	10.2 sec	8.6 sec	8.9 sec	66.9 sec

Table 8.2: Runtime in seconds of weak scalability experiment reconstructing the Stokeslet inside one island of SuperMUC. Runtime is dominated by the 4 components: Setup load in three components and the time needed for the SchurCG Stokes solver itself.

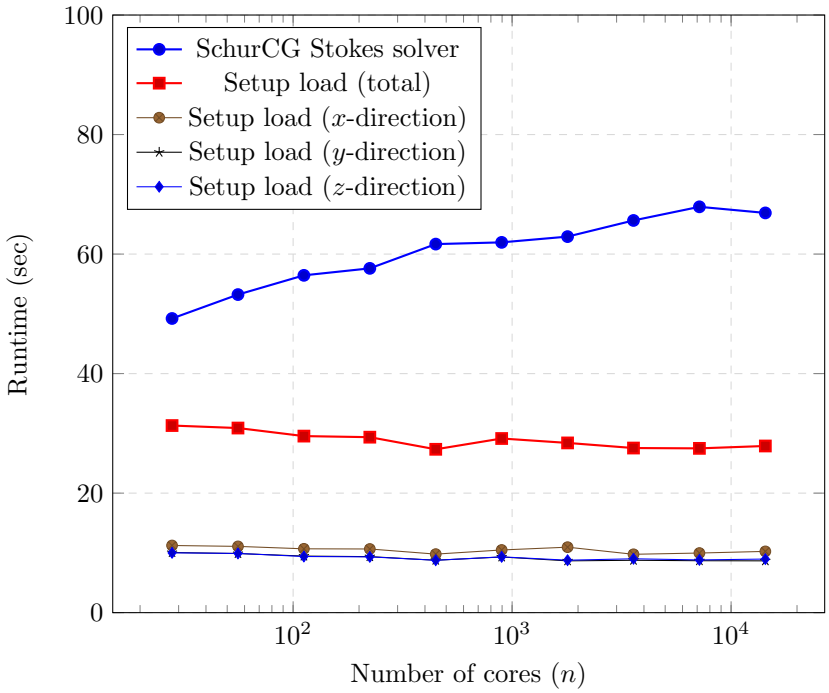


Figure 8.3: Runtime in seconds of weak scalability experiment reconstructing the Stokeslet inside one island of SuperMUC. Domain starting from 2688 tetrahedra (i.e. $7 \times 8 \times 8$ cubes of 6 tetrahedra) with 5 uniform refinement levels.

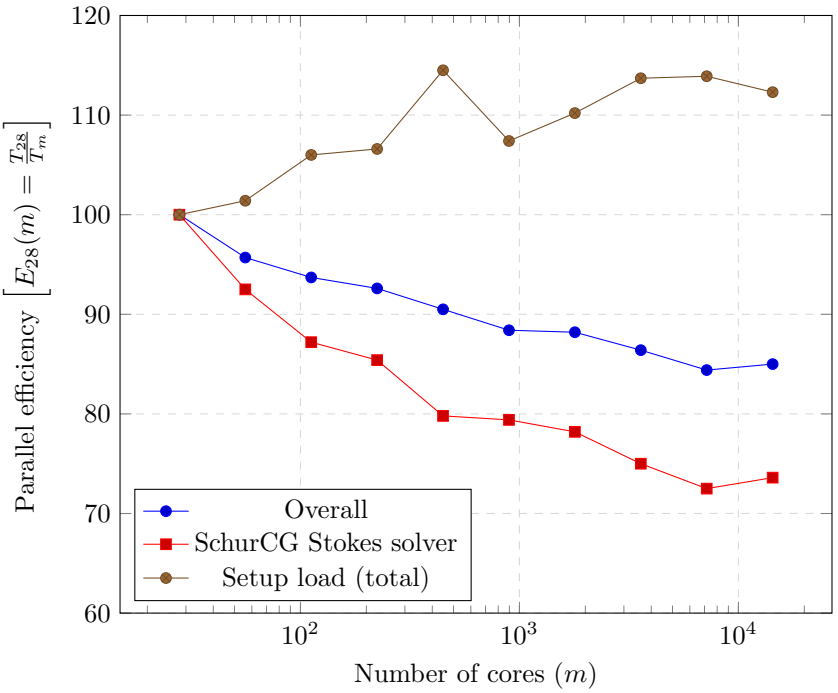


Figure 8.4: Efficiency of weak scalability experiment reconstructing the Stokeslet inside an island of SuperMUC.

Björn Gmeiner et al. found that the number of CG iterations on the coarsest level scales with the diameter of the domain [11].

For the SchurCG Stokes solver we combine the multigrid solver with another Preconditioned CG solver which could influence the prediction of Gmeiner et al. On the other hand, the SchurCG Stokes solver gives additional parameters to control the quality of the reconstructed solution. If we plot the error of the reconstructed velocity $\mathbf{u}$ and pressure p over the problem size, we get Figure 8.5. As we can see in the figure we are not able to keep the high quality of the numerical solution. The error is computed on the domain where we masked out a cube of radius $r = 0.25$ on all problem sizes and a small layer on the boundary.

8.3 Weak Scalability of Dirac integration

As we have seen in the runtime of the previous Stokeslet experiments the setup load function for only one Dirac force takes a significant amount of time. As we have discussed in Section 4.2.3 this is due to the lack of global indexing of the fine elements. By the specification of the Dirac force we know it's location but using the HHG data structures it is impossible to get the finite element that contains this Dirac point. Therefore, we store the Dirac points and when we integrate an element we check if there exists Dirac points inside that element. So, this strategy has complexity of $\mathcal{O}(m \cdot n \cdot d)$ where m is the number of macro elements, n is the number of elements inside a macro element and d the number of Dirac points.

8.3.1 Optimizing naive integration

The HHG framework is parallelized by distributing the macro elements over the processors, so parallel integration of the load vector has a complexity of $\mathcal{O}(\frac{m}{p} \cdot n \cdot d)$, where p is the number of cores that are used.

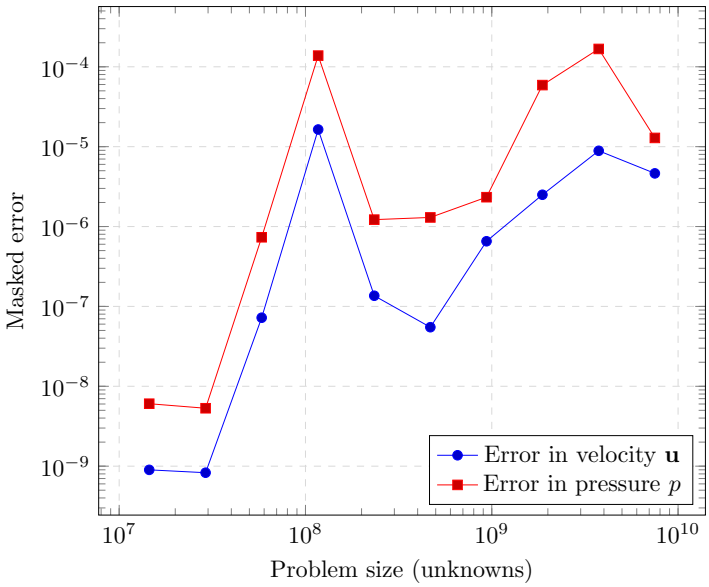


Figure 8.5: Quality of reconstructed Stokeslet on increasing domain sizes.

But, there is no need that a processor keeps track of Dirac points that are outside the closure of its domain, because the integration of all these Dirac points will end up to zero. So, instead of having p copies of a Dirac load vector containing all Dirac points of the global domain we optimize the Dirac integration by using distributed load vectors where each processor has a local load vector that contains the Dirac points inside the closure of its own domain. Assuming a homogeneous distribution of the Dirac forces this will improve the parallel complexity $\mathcal{O}(\frac{m}{p} \cdot n \cdot \frac{d}{p})$ for the integration of the load vector.

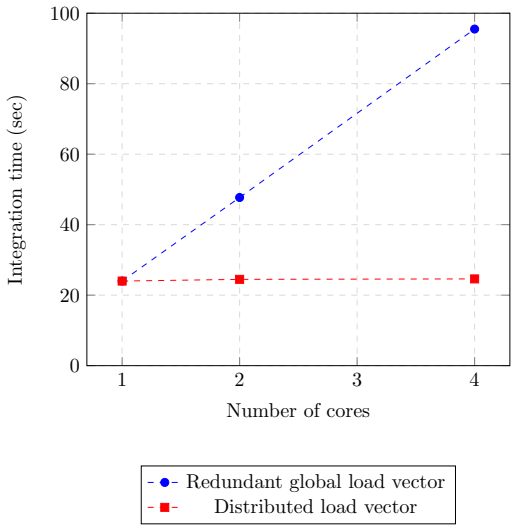
To show the effects of this optimization we run a small weak scaling experiment with both integration strategies on a workstation[3]. In this weak scaling experiment, we will increase both domain size and the number of Dirac points. So, by increasing the number of cores we increase the domain size and keep the *concentration* of Dirac forces constant. The Dirac forces are randomly distributed over the domain. Concerning the domain size, we start with the unit cube of 6 tetrahedra, which we will 5 times uniformly refine. Further we will start with 384 Dirac forces. The effects of this optimization are clearly visible in the integration time as shown in Figure 8.6a. With the distributed load vector, we are able to get an almost constant integration time for a constant amount of Dirac forces per core.

In Figure 8.6b we show the runtime for integration with and without Dirac forces and the runtime of the SchurCG Stokes solver for an average number of Dirac forces per core. As we would expect the runtime of the SchurCG Stokes solver is independent of the concentration of the Dirac forces. Taking into account that we plot Figure 8.6b with the average number of Dirac forces and runtime on a log-log scale we can see a linear dependence between the number of Dirac forces per core and the time needed to integrate these Dirac forces.

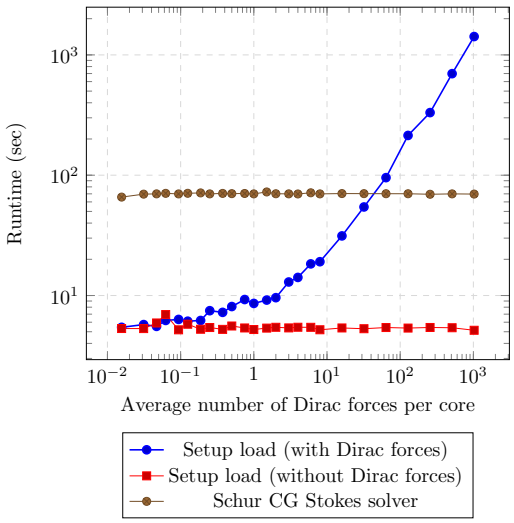
8.3.2 Model of FLOP computations

The integration of the Dirac function is compute intensive because there is no direct mapping from a location in the domain to a certain finite element that contains that point. So for each pair of element and Dirac force on a processor we need to compute if the Dirac force will be inside that element, i.e. apply coordinate transformations, cross products and dot products for a set of vertices.

[3]The workstation consists of an Intel Core i5 3570 processor with an IvyBridge architecture and 4 cores. For details: `http://ark.intel.com/products/65520/Intel-Core-i5-3570K-Processor-6M-Cache-up-to-3_80-GHz`



(a) Integration time in seconds for a weak scalability for a naive and a distributed implementation of the load vector.



(b) Runtime of Dirac integration and Stokes Solver for multiple Dirac points.

Figure 8.6: Scalability of the distributed load vector and runtime for different Dirac force concentrations.

Setup	**Runtime (sec)**
Setup load x-direction with normalization	152 sec
Setup load x-direction without normalization	98 sec

Table 8.3: Integration time in seconds needed for experiment with 6 refinements of the 8 cubes and 512 Dirac forces on a workstation with fixed frequency of 3.4 GHz. So, we have 12 macro elements per core, 128 Dirac forces per core and 3,145,728 elements per core. Integration time for the load vector with Dirac points is shown for both with and without normalization of normal vectors.

We can check if a point is inside a tetrahedron by constructing the normals of the faces (which are described by three vertices of tetrahedron N) and projecting the fourth vertex of tetrahedron N and Dirac point D on that normal. If we want to normalize the normal vector we need to compute a square root and three divisions per face.

As illustrated in Table 8.3 don't normalizing the normal vector saves approximately 36% of the integration time. Because we are only interested if the two points are on the same side of the face described by the normal vector it should not matter that the normal vector is not normalized. So for estimating the number of needed FLOPs we don't normalize the normal vectors which will have the additional advantage that checking if a Dirac point is inside a tetrahedron can be done without divisions and square roots.

As we have seen the computational complexity of the integration of the Dirac measures is $\mathcal{O}(m \cdot n \cdot d)$ where m is the number of macro elements, n is the number of fine elements inside a macro element and d the number of Dirac points.

To evaluate the constants in this expression we will use LIKWID [4] [30] to read the performance counters and get the amount of FLOPs that are processed during the SetupLoad function. These experiments are done on the Lima cluster[5] located at RRZE (Regionales RechenZentrum Erlangen).

Because the integration needs to be done over fine elements we can assume that we don't spend FLOPs when $m \cdot n = 0$. So, we will vary the number of Dirac points and do the experiments on a single core with 6 tetrahedra in the unit cube and 5 refinement levels. The results are shown in

[4] `https://github.com/RRZE-HPC/likwid`
[5] Details about the system can be found in Section B.2.

#Forces	Estimated FLOPs				Measured FLOPs	Converage
	x-dir	y-dir	z-dir	total		
0	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$5.951 \cdot 10^8$	$5.819 \cdot 10^8$	97.8 %
1	$3.179 \cdot 10^8$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$7.147 \cdot 10^8$	$7.016 \cdot 10^8$	98.2 %
2	$4.375 \cdot 10^8$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$8.342 \cdot 10^8$	$8.220 \cdot 10^8$	98.5 %
4	$6.765 \cdot 10^8$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$1.073 \cdot 10^9$	$1.066 \cdot 10^9$	99.3 %
8	$1.155 \cdot 10^9$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$1.551 \cdot 10^9$	$1.547 \cdot 10^9$	99.7 %
16	$2.111 \cdot 10^9$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$2.508 \cdot 10^9$	$2.505 \cdot 10^9$	99.9 %
32	$4.024 \cdot 10^9$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$4.420 \cdot 10^9$	$4.434 \cdot 10^9$	100.3 %
64	$7.849 \cdot 10^9$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$8.246 \cdot 10^9$	$8.265 \cdot 10^9$	100.2 %
128	$1.550 \cdot 10^{10}$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$1.590 \cdot 10^{10}$	$1.592 \cdot 10^{10}$	100.2 %
192	$2.315 \cdot 10^{10}$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$2.355 \cdot 10^{10}$	$2.360 \cdot 10^{10}$	100.2 %
256	$3.080 \cdot 10^{10}$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$3.120 \cdot 10^{10}$	$3.125 \cdot 10^{10}$	100.2 %
384	$4.610 \cdot 10^{10}$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$4.650 \cdot 10^{10}$	$4.659 \cdot 10^{10}$	100.2 %
512	$6.140 \cdot 10^{10}$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$6.180 \cdot 10^{10}$	$6.188 \cdot 10^{10}$	100.1 %
768	$9.200 \cdot 10^{10}$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$9.240 \cdot 10^{10}$	$9.251 \cdot 10^{10}$	100.1 %
1024	$1.226 \cdot 10^{11}$	$1.984 \cdot 10^8$	$1.984 \cdot 10^8$	$1.230 \cdot 10^{11}$	$1.232 \cdot 10^{11}$	100.1 %

Table 8.4: Measured and estimated FLOPs for integration of Load vector for a different number of Dirac forces. The estimated FLOPs are computed using Equation 8.1 with $m = 1, n = 8^5$ and d is given by #Forces.

Table 8.4 and Figure 8.7.

Scaling the number of measured FLOPs by the number of fine elements we get a number of FLOPs spend per Dirac force. The number of Dirac forces is neglectable compared to the number of fine elements, so the FLOPs spend on evaluating the linear basis functions on the locations of the Dirac forces will not have a big influence on this scaling by the number of fine elements. Using a linear fitting of the data we recover the following relation for the data:

$$E(m, n, d) = m \cdot n \cdot (1009 + 608d)\,, \tag{8.1}$$

where m is the number of macro elements, n is the number of fine elements inside a macro element and d the number of Dirac points. As we can see from the coverage in Table 8.4, which is the ratio between measured and estimated FLOPs, this linear fitting matches very well the measured data.

8.3.3 Weak scalability integration SuperMUC

Using this setup we can do a weak scalability experiment for the Dirac integration on SuperMUC where we start the experiment on one node (i.e. 28 cores) with a domain that consists of $7 \times 8 \times 4$ cubes. For the finest level we will uniformly refine the mesh 5 times. By doubling the number of nodes we well alternately double the domain size in the z-, x- and y-direction. Further we start with 5376 Dirac forces and double the number of Dirac forces when we double the number of nodes. Because we are only interested in the scalability of the integration the solving of the constructed system is omitted, but Figure 8.4 show the weak scalability of the SchurCG solver and Figure 8.6b shows its independence of the concentration of Dirac forces (i.e. the actual value of the right hand side vector).

The integration time for both the x-direction with the Dirac forces and the y-direction without Dirac forces is shown in Table 8.5 and Figure 8.8a. The parallel efficiency is shown in Figure 8.8b. Note that we didn't normalize the normal vectors describing the faces of the tetrahedra.

As we can see from Figure 8.8b for the integration of the load vector we have a good parallel efficiency of 95%. For 32 nodes (or 172,032 Dirac points) we see an outlier in the parallel efficiency which is visible in both the x-direction as the y-,z-direction. So, maybe this is related to a slightly variation in the clock frequency.

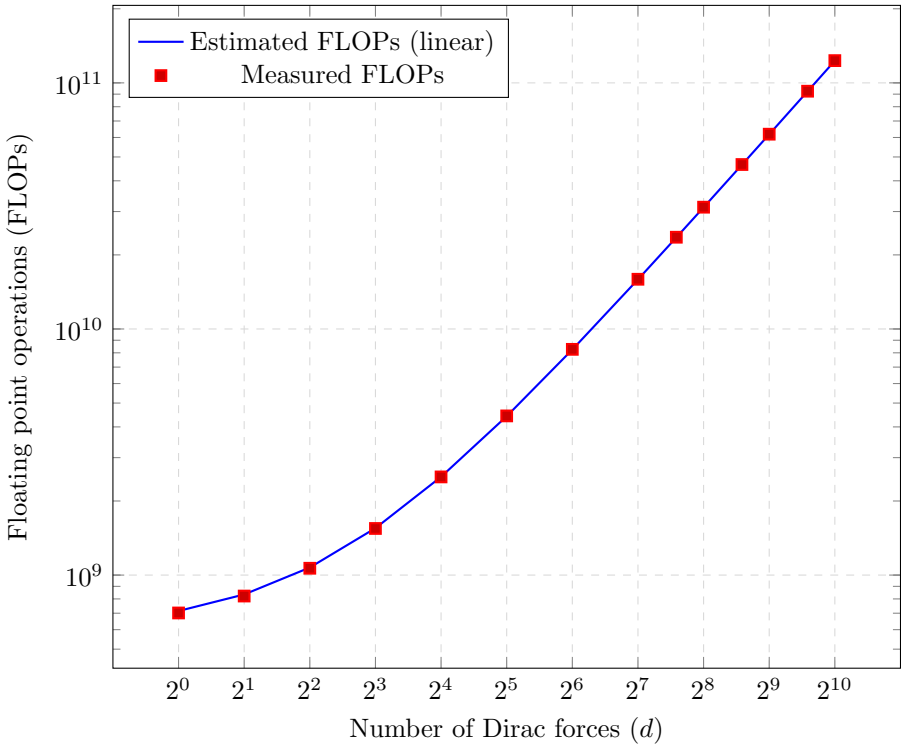
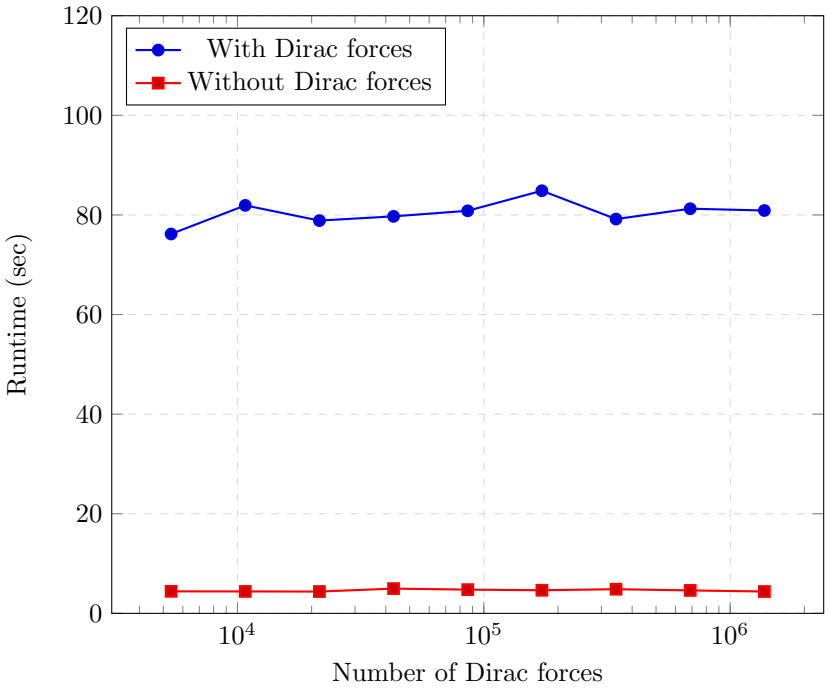


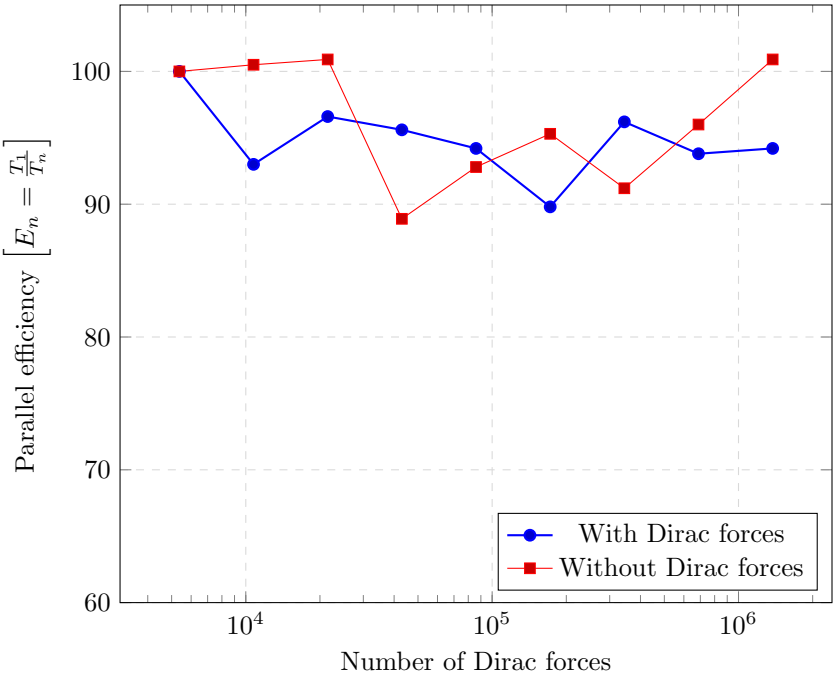
Figure 8.7: Measured and estimated FLOPs using a linear fitting of the measured data. The linear fitting is given by Equation 8.1.

#Nodes	#Cores	#Diracs	#Macro elements	Setup Load (sec)	
				x-dir	y-, z-dir
1	28	5376	1344	76.1 sec	4.4 sec
2	56	10752	2688	81.9 sec	4.4 sec
4	112	21504	5376	78.8 sec	4.3 sec
8	224	43008	10752	79.7 sec	4.9 sec
16	448	86016	21504	80.8 sec	4.7 sec
32	896	172032	43008	84.8 sec	4.6 sec
64	1792	344064	86016	79.1 sec	4.8 sec
128	3584	688128	172032	81.2 sec	4.6 sec
256	7168	1376256	344064	80.8 sec	4.3 sec

Table 8.5: Integration time in seconds of the weak scalability experiment reconstructing multiple Dirac forces in x-direction in a rectangular box-shaped domain on SuperMUC. Setup load in x-direction shows the integration time of the load vector with Dirac forces and y-, z-direction show the integration time without Dirac forces. In this experiment we have 48 macro elements per core and on average 192 Dirac forces per core. The timings are without normalization of the normals of the faces of the fine elements.



(a) Runtime in seconds for setup load in weak scalability experiment on SuperMUC with 192 Dirac forces per core.



(b) Parallel efficiency for setup load in weak scalability experiment on SuperMUC with 192 Dirac forces per core.

Figure 8.8: Runtime and parallel efficiency for setup load in weak scalability experiment on SuperMUC with increasing number of Dirac forces and constant density.

8.4 Conclusion and optimizations

In this scalability chapter we saw that the SchurCG Stokes solver is strong and weakly scalable for solving the Stokes problem. For the strong scalability we saw a parallel efficiency of 82% inside one node. The SchurCG Stokes solver is at least weakly scalable up to one island of SuperMUC with a parallel efficiency of 73%.

Further, we saw even better weakly scalable results for integration of the Dirac forces where we increase the domain size and keep the concentration of the Dirac forces constant. We could setup the load vector for the Stokes problem with 1,376,256 Dirac forces with a parallel efficiency around 95%. So, in the current implementation the weak scalability of a multiple Dirac force simulation will be bounded by the scalability of the Stokes solver. This will allow us already to use HHG for running particle fluid interaction simulations on large systems where we model the particles by Dirac forces.

Finally, if HHG would support direct access to elements that are related to a point in physical space then we could completely get rid of the computationally intensive setup load for Dirac forces and get a compute load that matches the thinness of the Dirac forces in the domain. As we have seen checking if a point is inside a fine tetrahedron will cost approximately 608 FLOPs per check.

Because the macro elements in HHG are regularly refined the geometric coordinates (or offsets with respect to a local origin) of the fine element that contains the Dirac force can be computed $\mathcal{O}(1)$, if it is known that the Dirac force is inside the macro element (which can be verified in just $\mathcal{O}(m)$, where m is the number of macro elements). Using these offsets to update a specific entry in the load vector could give at least a refinement level independent integration of the load vector. In that way we can get rid of the factor n as the number of fine elements inside a macro element, which is non-parallelizable and exponential in the refinement level.

Chapter 9

Conclusion and Outlook

Through this thesis, the main goal was to verify if we can numerically reconstruct the fundamental solution of the Stokes system, the Stokeslet, on a finite element mesh with second order convergence in the far field of the singularity. This goal was inspired by the result that was found for the Poisson problem, where we can get indeed second order convergence in the far field excluding a region around the singularity.

9.1 Conclusion

In this thesis we showed that second order convergence in the far field is, for a couple of test cases, also true for the Stokes system where we used $\mathbb{P}_1$-iso-$\mathbb{P}_2$ elements for the velocity $\mathbf{u}$ and $\mathbb{P}_1$ elements for the pressure p. We found that we can get second order convergence for the velocity $\mathbf{u}$ in all test cases and for the pressure we were able to get second order convergence for the cases where the Dirac force is on a vertex.

We observed that the reconstructed pressure is not smooth and that $\mathbb{P}_2$ elements for the velocity $\mathbf{u}$ gives the same oscillations in the same regions. We tried to come up with some ideas to get rid of these oscillations using the regularized Stokeslet and tent approximations as a spreading of a Dirac measure. Both approaches are (up to a certain level) able to reduce the amplitude of the oscillations but in both cases we cannot get rid of them. If we take a closer look to the error in the reconstructed potential u of the Poisson problem, then also there we see oscillations. A comparison with the MAC-discretization showed that we can reconstruct smooth solution of the Stokeslet using a marker-and-cell discretization. So, the oscillations seem to be related to the singularity on a irregular finite element mesh.

As second part we showed that we are able to scale the reconstruction of the Stokeslet problem on large systems like SuperMUC. Further we are able to scale the number of Dirac points where we fix the concentration of the Dirac forces. In a weak scalability experiment where we reconstruct the Stokeslet up to one island (i.e. 512 nodes or 14,236 cores) of SuperMUC we saw a parallel efficiency for the Schur complement CG solver of 73%. The results for the integration of multiple Dirac forces is even better, where we found a parallel efficiency of 95% for 1,376,256 Dirac forces.

9.2 Discussion

As we have pointed out in Chapter 6 there are some issues with the conservation of mass in a general Stokes system. We resolved that issue by masking out the error on a boundary layer of a certain thickness. In most cases this seems to work well. But especially in the case were we are not able to recover second order convergence for the pressure we found a line where the estimated order of convergence increases when the distance to the boundary increases.

Also the issue with possible errors in the pressure alignment in case of a Dirac force inside an element need to be considered in more detail. We observed only first order decrease of the

difference between the average of the exact pressure and the reconstructed pressure, maybe due to the oscillations in the reconstructed pressure. This could be another reason why we are not able to recover second order convergence for the pressure in that case.

All errors in this thesis are measured in the l^2 or Euclidean norm, but it is not the natural norm on the finite element spaces. Because HHG uses only linear elements and we used only tetrahedra with the same volumes the results for the l^2 and the L^2-norm should be equivalent, but for a correct treatment one should verify the results with a correct implementation of the L^2-norm.

9.3 Outlook

One other possibility to reconstruct the Stokeslet would be that we can integrate the momentum equation to get rid of the Dirac measure on the right hand side of the equation. By integrating the Dirac measure we get a 'step function'. Setting the integration constant to $-\frac{1}{2}$ and integrate again we will get the 'absolute value function'. Because differentiation and integration commutes we can solve this momentum equation for auxiliary velocity $\tilde{\mathbf{u}}$. Excluding the location of the Dirac measure this system should be solvable. The velocity $\mathbf{u}$ of the Stokes system can be obtained by getting the second derivative of the auxiliary velocity $\tilde{\mathbf{u}}$. Generalization of this idea to multiple dimensions needs to be considered but should be existing.

The driving problem for reconstructing the Stokeslet was to model sedimentation processes. The advantage of using Dirac forces on a finite element mesh is that we don't need to get the finite element mesh in the same accuracy as the particles that we want to model. In this thesis we showed that we can get accurate results for the impact of a particle on the Stokes flow around it. By linearity of the Stokes system we can add this impact to the Stokes flow to compute the interaction from a particle to the Stokes flow.

The Stokes system is time independent but using velocities and force balances we can recompute the locations of the particles and move them accordingly to get some kind of 'time integration' to model particle Stokes flow interactions. This will allow us, for example, to model sedimentation processes on a finite element mesh. Combining this with massively parallel framework like Hierarchical Hybrid Grids could overcome the cost of the huge amount of unknowns that we introduced by using the finite element method to solve these sedimentation processes.

Appendix A

Conjugate Gradient method

The Conjugate Gradient method is an iterative method that solves the linear system

$$A\mathbf{x} = \mathbf{b} \tag{A.1}$$

for $\mathbf{x}$. The CG method constructs a vector $\mathbf{x}_i \in K^i(A, \mathbf{r}_0)$[1] such that $\|\mathbf{x}^* - \mathbf{x}_i\|$ is minimal.
We can write the first iterate $\mathbf{x}_1 = \alpha_0 \mathbf{r}_0$, where α_0 need to be chosen such that $\|\mathbf{x}^* - \mathbf{x}_1\|$ is minimal. Thus we need to minimize

$$\begin{aligned} f_2(\alpha_0) &= \|\mathbf{x}^* - \mathbf{x}_1\|_2^2 \\ &= (\mathbf{u}^* - \alpha_0 \mathbf{r}_0)^\top (\mathbf{u}^* - \alpha_0 \mathbf{r}_0) \\ &= {\mathbf{u}^*}^\top \mathbf{u}^* - 2\alpha_0 \mathbf{r}_0^\top \mathbf{u}^* + 2\alpha_0^2 \mathbf{r}_0^\top \mathbf{r_0} \end{aligned}$$

which is minimized when we choose α_0 as

$$\alpha_0 = \frac{\mathbf{r}_0^\top \mathbf{u}^*}{\mathbf{r}_0^\top \mathbf{r_0}}$$

Because $\mathbf{u}^*$ is unknown we cannot minimize $\|\mathbf{x}^* - \mathbf{x}_i\|_2$, but because A is an SPD-matrix, hence $\|\cdot\|_A$ defines norm, we can minimize $\|\mathbf{x}^* - \mathbf{x}_i\|_A$:

$$\begin{aligned} f_A(\alpha_0) &= \|\mathbf{x}^* - \mathbf{x}_1\|_A^2 \\ &= {\mathbf{u}^*}^\top A\mathbf{u}^* - 2\alpha_0 \mathbf{r}_0^\top A\mathbf{u}^* + 2\alpha_0^2 \mathbf{r}_0^\top A\mathbf{r_0} \\ &= {\mathbf{u}^*}^\top A\mathbf{u}^* - 2\alpha_0 \mathbf{r}_0^\top \mathbf{b} + 2\alpha_0^2 \mathbf{r}_0^\top A\mathbf{r_0} \end{aligned}$$

which gives

$$\alpha_0 = \frac{\mathbf{r}_0^\top \mathbf{b}}{\mathbf{r}_0^\top A\mathbf{r_0}}$$

In the CG method we introduce search direction vectors $\mathbf{p}_k$ which forms an orthogonal basis for the Krylov subspace $K^k(A, \mathbf{r}_0)$ and are A-conjugate to all $\mathbf{p}_i$ for $i < k$ [14].

The approximations $\mathbf{x}_k$ obtained by the CG method satisfy the following inequality [34]:

$$\|\mathbf{x}^* - \mathbf{x}_k\|_A \leq 2\left(\frac{\kappa_2(A) - 1}{\kappa_2(A) + 1}\right)^k \|\mathbf{x}^* - \mathbf{x}_0\|_A$$

So the distribution of the eigenvalues of A determine the rate of convergence of the CG method.

[1] The Krylov subspace $K^i(A, \mathbf{r}_0)$ is defined as $K^i(A, \mathbf{r}_0) = \text{span}\left\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots, A^{i-1}\mathbf{r}_0\right\}$

A.1 Preconditioned CG Solver

In stead of solving the system of Equation A.1 we can apply a preconditioner to the system. Applying a preconditioner M to the system, we will solve the linear system

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}.$$

Because rate of convergence of the CG method is determined by the distribution of the eigenvalues of the system matrix we can influence the rate of convergence by choosing a preconditioner of which the distribution of the eigenvalues of $M^{-1}A$ is better as the distribution of the eigenvalues of A.

The algorithm for the preconditioned CG solver with preconditioner M is given by:

$\mathbf{x}_0 = \mathbf{0}$
$\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$
$\mathbf{z}_0 = M^{-1}\mathbf{r}_0$
$\mathbf{p}_0 = \mathbf{z}_0$
`for` $k = 0, 1, 2, \ldots$
$\quad \alpha_k = \frac{\mathbf{r}_k^\top \mathbf{z}_k}{\mathbf{p}_k^\top A\mathbf{p}_k}$
$\quad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
$\quad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{p}_k$
$\quad$ `if` $\mathbf{r}_{k+1}$ small
$\quad\quad$ DONE; algorithm converged
$\quad \mathbf{z}_{k+1} = M^{-1}\mathbf{r}_{k+1}$
$\quad \beta_k = \frac{\mathbf{z}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{z}_k^\top \mathbf{r}_k}$
$\quad \mathbf{p}_{k+1} = \mathbf{z}_{k+1} + \beta_k \mathbf{p}_k$
`end`

Appendix B

System specifications

B.1 SuperMUC Phase 2

SuperMUC Phase 2 is a Petascale System at the Leibniz-Rechenzentrum [22] (Leibniz Supercomputing Centre) in Munich (Germany) in June 2016 at position 28 in the TOP500[1] ranking.

General	
Processor type	Intel Haswell Xeon Processor E5-2697 v3 [2]
Nominal Frequency	2.6 GHz (SuperMUC Default: 2.3 GHz)
Performance per Core	41.6 DP FLOPs/sec
Total Number of Islands	6
Total Number of Nodes	3072
Total Number of Cores	86016
Total Peak Performance	3.58 PFLOPs/sec
Total size of memory	194 Tbyte
Typical Power Consumption	1.1 MW

Components	
Nodes per Island	512
Processors per Node	2
Cores per Processor	14
Cores per Node	28

Memory and Caches	
Memory per Core	2.3 Gbyte (typically available 2.1 Gbyte)
Size of shared Memory per node	64 Gbyte
Bandwidth to Memory per node	137 Gbyte/s

Interconnect	
Interconnect Technology	Infiniband FDR14
Intra-Island Topology	non-blocking Tree
Inter-Island Topology	Pruned Tree 4:1

Table B.1: Specifications of SuperMUC Phase 2

[1] `https://www.top500.org/`
[2] `http://ark.intel.com/products/81059/Intel-Xeon-Processor-E5-2697-v3-35M-Cache-2_60-GHz`

B.2 Lima Cluster

Lima Cluster is a high-performance compute resource at Regionales RechenZentrum Erlangen [8] (Germany).

General	
Processor type	Intel Westmere Xeon Processor 5650 [3]
Nominal Frequency	2.66 GHz
Total Number of compute nodes	500
Total Number of Cores	1000
Total size of memory	194 Tbyte
Typical Power Consumption	1.1 MW

Components	
Processors per Node	2
Cores per Processor	12
Cores per Node	24

Memory and Caches	
Memory per Core	1.0 Gbyte
Size of shared Memory per node	24 Gbyte

Table B.2: Specifications of Lima cluster

[3] ark.intel.com/products/47922/Intel-Xeon-Processor-X5650-12M-Cache-2_66-GHz-6_40-GTs-Intel-QPI

Bibliography

[1] Benjamin Bergen, Tobias Gradl, Frank Hulsemann, and Ulrich Rude. A massively parallel multigrid method for finite elements. *Computing in Science and Engg.*, 8(6):56–62, November 2006.

[2] B.K. Bergen. *Hierarchical Hybrid Grids: Data Structures and Core Algorithms for Efficient Finite Element Simulations on Supercomputers*. Advances in simulation. SCS Publishing House, 2005.

[3] J. Bey. Tetrahedral grid refinement. *Computing*, 55(4):355–378, 1995.

[4] F Brezzi, D Boffi, L Demkowicz, RG Durán, RS Falk, and M Fortin. *Mixed finite elements, compatibility conditions, and applications*. Springer, 2008.

[5] Franco Brezzi and Jim Douglas Jr. Stabilized mixed methods for the stokes problem. *Numerische Mathematik*, 53(1-2):225–235, 1988.

[6] Ricardo Cortez, Lisa Fauci, and Alexei Medovikov. The method of regularized stokeslets in three dimensions: analysis, validation, and application to helical swimming. *Physics of Fluids (1994-present)*, 17(3):031504, 2005.

[7] H.C. Elman, D.J. Silvester, and A.J. Wathen. *Finite Elements and Fast Iterative Solvers : with Applications in Incompressible Fluid Dynamics: with Applications in Incompressible Fluid Dynamics*. Numerical Mathematics and Scientific Computation. OUP Oxford, 2005.

[8] Regionales RechenZentrum Erlangen. Lima cluster. `https://www.rrze.fau.de/dienste/arbeiten-rechnen/hpc/systeme/lima-cluster.shtml`. Accessed: 2016-08-15.

[9] B. Gmeiner, M. Huber, L. John, U. Rüde, and B. Wohlmuth. A quantitative performance analysis for Stokes solvers at the extreme scale. *ArXiv e-prints*, November 2015.

[10] Björn Gmeiner. *Design and Analysis of Hierarchical Hybrid Multigrid methods for peta-scale systems and beyond*. 2013.

[11] Björn Gmeiner, Harald Köstler, Markus Stürmer, and Ulrich Rüde. Parallel multigrid on hierarchical hybrid grids: a performance study on current high performance computing clusters. *Concurrency and Computation: Practice and Experience*, 26(1):217–240, 2014.

[12] Björn Gmeiner, Ulrich Rüde, Holger Stengel, Christian Waluga, and Barbara Wohlmuth. Performance and scalability of hierarchical hybrid multigrid solvers for stokes systems. *SIAM Journal on Scientific Computing*, 37(2):C143–C168, 2015.

[13] Björn Gmeiner, Ulrich Rüde, Holger Stengel, Christian Waluga, and Barbara Wohlmuth. Towards textbook efficiency for parallel multigrid. *Numerical Mathematics: Theory, Methods and Applications*, 8(01):22–46, 2015.

[14] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.

[15] Tobias Gradl. *Adaptive Refinement of Hierarchical Hybrid Grids*. 2015.

[16] Francis H Harlow, J Eddie Welch, et al. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface.

[17] Markus Huber, Lorenz John, Petra Pustejovska, Ulrich Rüde, Christian Waluga, and Barbara Wohlmuth. Solution techniques for the stokes system: A priori and a posteriori modifications, resilient algorithms. *arXiv preprint arXiv:1511.05759*, 2015.

[18] Manfred Kaltenbacher. *Numerical Simulation of Mechatronic Sensors and Actuators*, volume 3. Springer, 2015.

[19] S. Kim and S.J. Karrila. *Microhydrodynamics: Principles and Selected Applications*. Butterworth - Heinemann series in chemical engineering. Dover Publications, 2005.

[20] T Köppl and B Wohlmuth. Optimal a priori error estimates for an elliptic problem with dirac right-hand side. *SIAM Journal on Numerical Analysis*, 52(4):1753–1769, 2014.

[21] H. W. Kuhn. Some combinatorial lemmas in topology. *IBM J. Res. Dev.*, 4(5):518–524, November 1960.

[22] Leibniz-Rechenzentrum. Supermuc petascale system. `https://www.lrz.de/services/compute/supermuc/systemdescription`. Accessed: 2016-08-15.

[23] Maciej Lisicki. Four approaches to hydrodynamic green's functions–the oseen tensors. *arXiv preprint arXiv:1312.6231*, 2013.

[24] Anders Logg, Kent-Andre Mardal, and Garth Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*, volume 84. Springer Science & Business Media, 2012.

[25] Oana Marin. Boundary integral methods for stokes flow: Quadrature techniques and fast ewald methods. 2012.

[26] C. Pozrikidis. *Boundary Integral and Singularity Methods for Linearized Viscous Flow*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 1992.

[27] Alison Ramage and Andrew J. Wathen. Iterative solution techniques for the stokes and navier-stokes equations. *International Journal for Numerical Methods in Fluids*, 19(1):67–83, 1994.

[28] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[29] V.V. Shaidurov. *Multigrid Methods for Finite Elements*. Mathematics and Its Applications. Kluwer Academic Publishers, 1995.

[30] Jan Treibig, Georg Hager, and Gerhard Wellein. Likwid: A lightweight performance-oriented tool suite for x86 multicore environments. In *2010 39th International Conference on Parallel Processing Workshops*, pages 207–216. IEEE, 2010.

[31] Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. *Multigrid*. Academic press, 2000.

[32] M. ur Rehman, T. Geenen, C. Vuik, G. Segal, and S. P. MacLachlan. On iterative methods for the incompressible stokes problem. *International Journal for Numerical Methods in Fluids*, 65(10):1180–1200, 2011.

[33] Rudiger Verfurth. A combined conjugate gradient-multi-grid algorithm for the numerical solution of the stokes problem. *IMA Journal of Numerical Analysis*, 4(4):441–455, 1984.

[34] C. Vuik and D.J.P. Lahaye. *Scientific Computing (wi4201)*. Delft Institute of Applied Mathematics, 2015.