GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Fakultät für
Physik

GPU-basierte Immersed Boundary Methoden mit Anwendung auf rotierende Strömungen

---

GPU-based Immersed Boundary Methods with Application to Rotating Flows

Masterarbeit

geschrieben am Institut für Geophysik
der Georg-August Universität Göttingen

von

Jonas Ruebsam
aus Paderborn

Abgabedatum: 16. Juni 2016

Die Arbeit wurde im Zeitraum vom 17.Dezember 2015 bis 16. Juni 2016 in der Arbeitsgruppe Fluiddynamik unter der Betreuung von Prof. Dr. Andreas Tilgner angefertigt.

Erstgutachter: Prof. Dr. Andreas Tilgner
Zweitgutachter: Apl. Prof. Dr. Ulrich Parlitz

Contents

NOMENCLATURE

variable	**meaning**
$\vec{r}$	position vector
$\vec{t}$	time
$\vec{v}(\vec{r}, t)$	velocity with components v_x, v_y, v_z
ρ	density
p	static pressure
Φ	an arbitrary scalar variable
$\vec{v}$	kinematic viscosity
$\vec{\Omega}$	angular velocity of the rotating system
V	fluid volume
∂V	surface of a fluid volume
$\vec{n}$	normal of a surface
L	characteristic length scale of a fluid system
V	characteristic velocity scale of a fluid system
p_∞	characteristic reference pressure of a fluid system
Re	Reynolds number
Ek	Ekman number
Ma	Mach number
Pe	Peclet number
Θ	propagation angle of an inertial wave
$\vec{c}_g$	group velocity of an inertial wave
$\vec{c}_p$	phase velocity of an inertial wave
D_P	physical viscosity
D_N	numerical viscosity
$\mathscr{L}$	differential operator containing spatial derivaties
$\mathscr{L}^*$	discretizied differential operator $\mathscr{L}$
$\Delta x, \Delta y, \Delta z$	step size in the numerical grid in x, y and z direction
l_x, l_y, l_z	length of the simulation domain in x, y and z direction
N_x, N_y, N_z	number of grid points in x, y and z direction

variable	**meaning**
i, j, k	indices for a single grid cell
ϵ	numerical error
λ	decay rate of the numerical error
c	articficial speed of sound
δ	articficial compressibility
$H(x, y, z)$	masking function of the IB methods
η	damping coefficient of the VP method
J	non-dimensional Damping coefficient of the VP method
$\mathscr{H}$	Helicity
ω_c	critical libration frequency
$\langle v_z^2 \rangle (t)$	energy in the v_z velocity component
$A\left(\langle v_z^2 \rangle\right)$	amplitude of $\langle v_z^2 \rangle$

Introduction

Fluid flows play an important role in the study of geophysical systems. Subject of current research are rotating and convection driven flows on a large scale, for instance in the ocean, the atmosphere as well as flows in the core of the earth which is a potential mechanism for magnetic field generation. As a result of the Coriolis force, the flow in rotating fluids exhibits characteristic properties with physical solutions in the form of inertial waves. Numerical [Sau+12], [DSL] and experimental studies [Ful59] of inertial wave excitation in rotating cylinders have been performed. Inertial oscillations in spherical shells were investigated for example by [Til99]. The study of dynamos, i.e. the magnetic field of the earth, was performed by additionally considering an electrically conducting fluid. Numerical simulations of precession driven dynamos [Til05] and convection driven dynamos [Til12] were carried out. Furthermore, the DRESDYN dynamo [Ste+15] is an ongoing experimental research project.
In the fluid dynamics research group of the Institute for Geophysics the numerical simulation of such fluid flows using GPU[1] optimized algorithms has been recently developed. The use of GPUs for scientific computing has become increasingly popular in the last years because a vast number of applications can be dramatically speeded up and high oriented frameworks like the NVIDIA CUDA API simplify the creation and maintenance of parallelized algorithms. The current GPU implementation of the algorithm used in our research group is restricted to simulations in cuboid domains. Since an equidistant cartesian grid is used for the discretization of the fluid domain, it is not possible to perform simulations in complex-shaped geometries as the boundaries of the fluid domain do not coincide with the grid points.

Consequently the first goal of this master thesis is to extend the existing GPU algorithm in order to enable the simulation of fluid flows in complex-shaped geometries. The use of unstructured or body conforming grids is difficult to implement, since on a GPU a regular and homogeneous memory access is required to enable a high memory bandwidth [NVI15a]. Hence, the objective is to use a set of methods which retain the speed of the original GPU algorithm and use a cartesian grid.
This requirement is fulfilled by Immersed Boundary methods, for which an overview of is given in [MI05], [Gor13]. The concept behind these methods is the embedding of the curved boundary into the cartesian grid by using additional forcing terms and interpolation methods close to the fluid boundaries. Certain Immersed Boundary methods use a continuous forcing approach where the boundary is approximated by Lagrangian points and the forcing term acts over multiple grid

[1]Graphics Processing Unit

points [MI05]. A similar approach is given by the Volume-Penalization method [Lül11]. Direct forcing approaches exist where the boundary conditions are obtained by setting the velocity values directly at the nearest points to the boundary [Fad+00]. A bilinear interpolated version of the Direct Forcing approach is used in [Gor13], which was furthermore extended to work on a GPU [DeL12]. In this thesis different Immersed Boundary methods will be introduced, implemented and validated for different test cases.

The second goal of this thesis are numerical studies on a rotating fluid system. In particular this system is given by a cone whose rotation rate is modulated which is used as a mechanism for inertial wave excitation. An experimental study by Beardsley [Bea70] of this system shows that the apex of the cone acts as an attractor, where inertial waves travel into the apex and dissipate at the tip of the cone. By inserting a plate into the apex of the cone an inertial wave is reflected at the bottom and inertial modes are observable. The objective of this part of the thesis is to investigate if these results are reproducible with the use of Immersed Boundary methods.

The content of this thesis is separated into different chapters containing different topics which are summarized here

Chapter 1 Introduction to the theoretical concepts. This includes a description of the Navier-Stokes equations and the concept of non-dimensionalization. Moreover, rotating fluid systems are investigated and fundamental properties of inertial waves are discussed.

Chapter 2 Numerical methods used in the present work are introduced. The use of finite difference schemes in combination with Runge-Kutta methods for the temporal integration is discussed. Furthermore a brief overview over numerical stability criteria and the method of artifical incompressibility is given.

Chapter 3 Implementation of a finite difference algorithm on a GPU with the NVIDIA CUDA API. A short description of the CUDA API is given, followed by a discussion of the implementation of a time step integration algorithm on the GPU.

Chapter 4 Introduction to the Immersed Boundary methods, as well as to the the numerical validation with different test cases.

Chapter 5 Theoretical and experimental results of the fluid flow in a librating cone. A first simulation is implemented and validated by using a librating cylinder. Finally, different simulations of a librating cone and a frustum are discussed.

1. Theoretical Principles

For the development of a numerical model it is necessary to give an exact theoretical description of the fluid systems which are investigated in this thesis. Hence, this chapter contains a brief overview of the derivation and the properties of the fundamental equations of motions. For a more detailed description, the interested reader is referred to [FP99], on which Sec. 1.1 is based.

1.1 The Equations of Motion

At any time a viscous, newtonian and incompressible fluid is considered. The equations of motion for such a fluid can be derived by considering the conservation of mass and momentum inside a fixed control volume $V \subset \mathbb{R}^3$. Within the fluid the momentum at the position $\vec{r} = (x, y, z)^T$ is characterized by the velocity $\vec{v}(\vec{r}, t) = (v_x, v_y, v_z)^T \in \mathbb{R}^3$, meanwhile the mass distribution is given by the density distribution $\rho(\vec{r}) \in \mathbb{R}$.

1.1.1 Mass Conservation

Let ∂V be the enclosing surface and $\vec{n}$ the normal vector to the control volume. A control mass with the volume V_M is consider, traversing this control volume. For a fixed point in time it holds that $V_M = V$. For any intensive property ϕ the Reynolds transport theorem states that

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V_M} \mathrm{d}V \rho\phi = \frac{\mathrm{d}}{\mathrm{d}t}\int_{V} \mathrm{d}V \rho\phi + \int_{\partial V} \mathrm{d}S \rho\phi\vec{v}\vec{n}. \tag{1.1}$$

The assumption of a conservation law requires that the left side of equation 1.1 has to be zero. By setting $\phi = 1$ one obtains the integral form of mass conservation

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V} \mathrm{d}V \rho(t) + \int_{\partial V} \mathrm{d}S \rho\vec{v}\vec{n} \overset{\substack{\text{Gauss's}\\ \text{Law}}}{=} \int_{V} \mathrm{d}V \left(\frac{\mathrm{d}}{\mathrm{d}t}\rho(t) + \vec{\nabla}\left(\rho\vec{u}\right)\right). \tag{1.2}$$

The differential form of the equation is obtained by applying Gauss's law and considering an infinitesimal small control volume

$$\frac{\partial \rho}{\partial t} + \nabla\left(\rho \vec{v}\right) = 0. \tag{1.3}$$

As an incompressible fluid is investigated, where $\rho = \text{const.}$, the incompressible continuity equation reduces to

$$\nabla \cdot \vec{v} = 0. \tag{1.4}$$

1.1.2 Momentum Conservation

Using the same approach, but with setting $\phi = \vec{v}$, results in the integral form of the momentum equation,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathrm{d}V \rho \vec{v} + \int_{\partial V} \mathrm{d}S \rho \vec{v}\vec{v} \cdot \vec{n} = \sum \vec{F}_{\text{ext.}} + \sum \vec{F}_{\text{int.}}. \tag{1.5}$$

In addition, to the left hand side the equation is extended by additional internal and external forces which may act on the fluid inside the control volume. The external forces depend on the specific physical system, for example Buoyancy or the Coriolis force. Internal forces are given by the pressure and the normal and shear stresses acting on a fluid element.
For a newtonian fluid the internal forces can be described by the stress tensor $\boldsymbol{T}$

$$\sum \vec{F}_{int.} = \int_{\partial V} \mathrm{d}S \boldsymbol{T}\vec{n} = \int_V \mathrm{d}V \nabla \boldsymbol{T} = \int_V \mathrm{d}V \nabla \left(-\left(p + \frac{2}{3}\mu \nabla \vec{u} \right) + 2\mu \boldsymbol{D} \right) \tag{1.6}$$

with the static pressure p, the dynamic viscosity μ and the deformation tensor $\boldsymbol{D}$. Applying Gauss's law to equation 1.5 and consider an infinitesimal volume, the differential form of the impulse equation, also known as Navier-Stokes equation, is given by [1]

$$\frac{\partial v}{\partial t} + \underbrace{\left(\vec{v}\vec{\nabla}\right)\vec{v}}_{\text{I}} = \underbrace{-\frac{1}{\rho}\nabla p + \nu \Delta \vec{v}}_{\text{II}} + \sum \vec{F}_{\text{ext.}}. \tag{1.7}$$

Where the kinematic viscosity is given by the definition $\nu = \mu/\rho$. For an incompressible fluid the force term generated by $\boldsymbol{T}$ reduces to term II. In summary, the internal force is now represented by the pressure gradient and an diffusive impulse transport proportional to ν. The non-linearity of the fluid originates through term I, which is also denoted as the advection operator. It basically describes

[1] The Navier-Stokes term is generally referred to as the complete set of equations of motion or just the impulse equation here using the latter convention.

the change of the impulse of a fluid element when moving through the velocity field. [2] It should be noted that the set of equations is not yet solvable as the pressure variable is still undetermined.

1.1.3 Initial State and Boundary Conditions

The solution of a partial differential equation is undetermined by a constant of integration. In order to determine the temporal evolution of a fluid system, it is necessary to define its initial state and therefore one specific solution.
For every variable an initial condition must be chosen. With respect to a numerical solution it has to be considered that, for example an instability, is always triggered by some kind of disturbance. Thus, it might be advisable to not choose a trivial solution like a zero velocity field, but instead a solution which quickly evolves into the preferred state of the system. Often it is useful to add pseudo random noise.

Since the fluid domain is spatial restricted it is necessary to define the physical behavior at its boundaries as well. For a fluid domain V with the boundary ∂V the following boundaries are defined in [GDN98; FP99].

No-Slip Boundaries All velocity components are set to zero $\vec{v}|_{\partial V} = 0$. The fluid is at rest on ∂V and no flux through the boundary occurs. In a more general case, this kind of condition is also referred to as Dirichlet-Condition where $\Phi|_{\partial V} = c \in \mathbb{R}$ for any variable Φ.

Free-Slip Boundaries The velocity component in normal direction to the wall has to be zero. Hence, $\vec{n}\nabla\vec{v} = 0$ and $\vec{n}\vec{v} = 0$ is required. No flux through ∂V occurs and no friction is imposed on the fluid.

No-Flux Boundaries For a scalar Φ the flux through the boundary has to be zero. Hence, $\vec{n}\nabla\Phi = 0$. The general case is referred to as Neumann boundary condition, where $\vec{n}\nabla\Phi = c \in \mathbb{R}$. This boundary condition is used for example to obtain adiabatic walls.

Periodic boundaries These boundaries can be applied in all directions of a system. For example if the system is periodic in x-direction with a period of length L, it has to be ensured that all variables match on the boundaries that is $\Phi(x) = \Phi(x + L)$, for any variable Φ.

[2] For example the velocity of a fluid element will change when a velocity field is present and forces are absent.

1.1.4 Nondimensionalization

For many fluid systems nondimensionalization is used to simplify the equations of motion and to reduce the number free parameters. The approach behind this scheme is to define variable substitutions, such that the overall system is free from any physical units.
After the nondimensionalization the system is described by one or more dimensionless quantities, which characterize the overall physical behavior. As a result, it is easier to compare numerical simulations and experimental setups to each other. The following scales for the variables of time, position, velocity and pressure can be chosen [KC02] [3]

$$\text{Length:} \quad \vec{r} = \frac{\vec{r^*}}{L} \quad , \quad \text{Velocity:} \quad \vec{v} = \frac{\vec{v^*}}{V} \tag{1.8}$$

$$\text{Time:} \quad t = t^* \cdot \frac{V}{L} \quad , \quad \text{Pressure:} \quad p = \frac{p^* - p_\infty}{\rho V^2}, \tag{1.9}$$

where L is a length and V a velocity both given by characteristic scales from the fluid system. The pressure scale is set as a difference with respect to a characteristic pressure p_∞. The nondimensionalized Navier-Stokes equation is given by

$$\frac{\partial v}{\partial t} + \vec{v} \cdot \vec{\nabla} \vec{v} = -\nabla p + \frac{1}{Re} \Delta \vec{v} + \vec{F}_{\text{ext.}}. \tag{1.10}$$

The dynamic of the system is reduced to the dimensionless quantity *Re*, also referred to as the Reynolds number [KC02]

$$Re := \frac{VL}{\nu} = \frac{\rho V L}{\mu}. \tag{1.11}$$

Eq. 1.11 shows that the Reynolds number gives the ratio between the inertial forces $\propto VL$ to the viscous forces $\propto \nu$, of the fluid system. This means for small Reynolds numbers the viscosity determines the flow and a laminar flow can be expected. For a large Reynolds numbers an advection driven maybe even turbulent flow dominates.

[3] In this thesis the dimensional variables are denoted by an asterisk when compared to non-dimensional variables.

1.2 Rotational Fluid Dynamics

In this second part of the chapter, the Navier-Stokes equations are extended to govern the physical attributes of rotating systems, which play an important role in the context of geophysical fluid dynamics. Due to the continuous acceleration acting on the fluid, these systems exhibit new characteristic properties. One important case which shall be discussed, is the propagation of inertial waves inside a stratified rotating fluid.

1.2.1 Equations of Motion

The motion of a fluid in a coordinate system (**R**) is considered, rotating with the angular velocity $\vec{\Omega}$ relative to the intertial system (**I**) around the axis $\vec{\Omega}$. The relation of the time derivate between the two frames of motion is given by the operator relation

$$\left.\frac{\partial}{\partial t}\right|_{\boldsymbol{I}} = \left.\frac{\partial}{\partial t}\right|_{\boldsymbol{R}} + \vec{\Omega}\times \tag{1.12}$$

according to [Til07]. Applying this relation to the position vector relative to the coordinate systems **R**, **I** and suppose a constant rotation rate $\partial_t \vec{\Omega} = 0$, yields a coordinate transformation for the acceleration,

$$\left.\frac{\partial \vec{v}}{\partial t}\right|_{\boldsymbol{I}} = \left.\left(\frac{\partial \vec{v}}{\partial t} + \underbrace{2\vec{\Omega}\times\vec{v}|_R}_{\text{I}} - \underbrace{\vec{\Omega}\times(\vec{\Omega}\times\vec{r}|_R)}_{\text{II}}\right)\right|_{\boldsymbol{R}} . \tag{1.13}$$

Hence, the transition into a rotating coordinate system introduces two additional pseudo forces, the coriolis force (I) and the centrifugal force (II).
A substitution of expression 1.13 into the Navier-Stokes equation gives the equations of motion for the rotating system. A further simplification can be obtained by considering that the centrifugal force is independent of the velocity field. Hence, it can be written in terms of a potential Φ

$$\Omega\times(\Omega\times\vec{r}) = -\nabla\left(\frac{1}{2}\Omega^2\vec{r}'^2\right) = -\Phi \tag{1.14}$$

which can be substituted into the pressure gradient by defining $p^* = p - \Phi$ [Tri88].

The following scales are introduced in order to obtain a nondimensional equation

$$\text{Length:} \quad \vec{r} = \frac{\vec{r}^*}{L} \quad , \quad \text{Velocity:} \quad \vec{u} = \frac{\vec{u}^*}{|\vec{\Omega}|L}, \tag{1.15}$$

$$\text{Time:} \quad t = t^* \cdot |\vec{\Omega}| \quad , \quad \text{Pressure:} \quad p = \frac{p^* - p_\infty}{\rho L^2 |\vec{\Omega}|^2}. \tag{1.16}$$

The final nondimensionalized form of the Navier-Stokes equation for the rotating coordinate system is given by

$$\frac{\partial v}{\partial t} + \left(\vec{v}\vec{\nabla}\right)\vec{v} + 2\Omega \times \vec{v} = -\nabla p + Ek\Delta\vec{v} + \vec{F}_{\text{ext.}}, \tag{1.17}$$

with the dimensionless quantity the Ekman number which is defined by

$$Ek := \frac{\nu}{|\vec{\Omega}|L^2} \mathrel{\hat{=}} \frac{\text{viscous force}}{\text{coriolis force}}. \tag{1.18}$$

The Ekman number describes the ratio between viscous force and Coriolis force.

1.2.2 Inertial Waves

Inertial waves are a solution of Eq. 1.17 and will be described here. This section represents a short recapture of some of the fundamental properties of mechanical waves. According to Kundu2012:

> It is perhaps not an overstatement to say that wave motion is the most basic feature of all physical phenomena. Waves are the means by which information is transmitted between two points in space and time, without movement of the medium across the two points. The energy and phase of some disturbance travel during a wave motion, but motion of the matter is generally small. Waves are generated due to the existence of some kind of "restoring force" that tends to bring the system back to its undisturbed state, and of some kind of "inertia" that causes the system to overshoot after the system has returned to the undisturbed state.[KC02, p.194].

The propagation of inertial waves requires a medium which contains a state of equilibrium and a restoring force pointing back to this state in response to a disturbance. In the context of geophysical fluid mechanics an equilibrium state can be given by a stratification.
Considering a fluid with a continuous stratified density. This means, that in the equilibrium, the density of the fluid has to decrease continuously with the height of the system. The displacement of

a fluid element from its resting position results in a restoring force, which is given by the gravitation or the buoyant force. [4] Therefore, it can be seen that a disturbance of the equilibrium state can result in the propagation of so-called gravity waves[5] [Cla11]. [6]
Now a uniform rotating fluid without the presence of external forces is considered. At some point in time a steady state is reached, as a result of the dynamic equilibrium between a radial stratification of the angular momentum and the pressure. The displacement of a fluid element in radial direction results in an imbalance of the preserved angular momentum and the pressure. Due to the restoring force, given by the coriolis force, an oscillation develops. Waves of this type are denoted as inertial waves [Cla11].

1.2.2.1 Plane Inertial Waves

A short introduction of the properties of plane inertial waves shall be given here. All results presented in this section are adopted from [Gre90].The non-linear advection operator and the viscous stress in Eq. 1.17 are neglected, $Ek = 0$. In case of a linear inviscid fluid the equations of motion are fulfilled by plane wave solutions of the form

$$\vec{v} = \vec{V} e^{i(\vec{k}\vec{r} - \omega t)}, \qquad p = P e^{i(\vec{k}\vec{r} - \omega t)} \tag{1.19}$$

where $\vec{k}$ is the wave vector and ω the frequency. The wave is transverse since an insertion into the continuity equation yields $\vec{V}\vec{k} = 0$. From the momentum equation the dispersion relation is given by

$$\omega = \pm \frac{2\vec{\Omega}\vec{k}}{|\vec{k}|} = \pm 2|\vec{\Omega}|\cos(\theta) \tag{1.20}$$

where θ is the polar angle with respect to the rotation axis, such that $\vec{k}\vec{\Omega} = |\vec{k}||\vec{\Omega}|\cos\theta$. This means that an inertial wave can only exist for a wave frequency smaller than twice of the rotation rate $\vec{\Omega}$. The phase velocity $\vec{c}_p$ and the group velocity $\vec{c}_g$ are given by

$$\vec{c}_p = 2\frac{\vec{\Omega}\vec{k}}{|\vec{k}|^3}\vec{k}, \qquad \vec{c}_g = \frac{2\vec{k} \times (\vec{\Omega} \times \vec{k})}{|\vec{k}|^3}. \tag{1.21}$$

[4]This depends on the direction of the displacement.
[5]Not to be confused with gravitational waves.
[6]One import case we know from everyday life are surface waves, which emerge from the discontinuous density stratification between two fluids, i.e. water and air [Cla11].

It can be noted that the group velocity and therefore the energy transport is perpendicular to the phase velocity of the wave. The direction of propagation on an inertial wave is given by the angle θ with respect to the axis of rotation $\vec{\Omega}$, see Fig. 1.1(a).

1.2.2.2 Reflection of Inertial Waves

The reflection of an inertial wave contradict's snells law, since the propagation angle θ is preserved upon a reflection. It can be shown that the following relations holds [Bea70],

$$\vec{\Omega}\cdot\vec{k} = \vec{\Omega}\cdot\vec{k}^{\dagger}, \qquad \hat{n}\times\vec{k} = \hat{n}\times\vec{k}^{\dagger} \tag{1.22}$$

where † denotes wave number upon reflection and $\hat{n}$ corresponds to the normal vector of the wall. As a result, inertial waves exhibit a fundamental different reflection behavior, which is summarized here. Fig. 1.1 shows exemplarily the reflection of an inertial wave with the propagation angle θ, on a wall with the slope α to the rotation axis. Three scenarios exist [Cla11]:

$\theta < \alpha$ See Fig. 1.1(b). A downward propagating wave is reflecting upwards. This is also referred to as subcritical reflection. The distance between two parallel waves is decreased upon a reflection which is denoted as focusing.

$\theta > \alpha$ See Fig. 1.1(c). A downward propagating wave is reflecting downwards. This is also referred to as supercritical reflection. The distance between two upward traveling parallel waves increases upon a reflection which is denoted as defocusing.

$\theta = \alpha$ All waves are reflected parallel to the slope and infinitely focused.

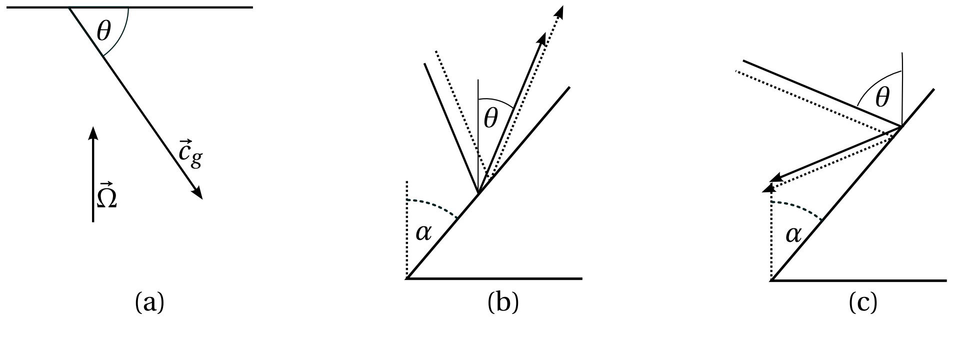


Figure 1.1.: Reflection of inertial waves: (a) Propagation of inertial waves with the propagation angle θ. (b) Subcritical reflection of a downward traveling wave. (c) Supercritical reflection of a downward traveling wave.

1.2.2.3 Inertial Modes in a Cylinder

Within a enclosed geometry the possible number of solutions of Eq. 1.17 can be finite [Cla11]. For the scenario of a rotating cylinder the solutions are given by a discrete spectrum of inertial modes. A analytical solution can be found in [Gre90] and is summarized here.
For the linear inviscid case with $Ek = 0$ and without the advection term, the problem can be reduced to a Poincaré equation for the pressure, which is solved in cylindrical coordinates by

$$p_{nmk}(r,\theta,z,t) = J_{|k|}\left(\frac{\xi_{nmk} r}{a}\right)\cos(n\pi)e^{ik\theta}e^{i\lambda t} \tag{1.23}$$

where $a = r/H$ is the aspect ratio of the cylinder, $n \in \mathbb{N}$ is the vertical and $k \in \mathbb{Z}$ the azimuthal wave number. $m \in \mathbb{N}$ is equal to the number of nodes in radial direction. $J_{|k|}$ denotes a Bessel function of k-th order. The radial wavenumber ξ can be solved for any given k by the transcendental equation

$$\xi\frac{\mathrm{d}}{\mathrm{d}\xi}J_{|k|}(\xi) + k\sqrt{1+\frac{\xi^2}{n^2\pi^2a^2}}\,J_{|k|}(\xi) = 0. \tag{1.24}$$

Finally the eigenvalue of the solution is determined by

$$\lambda_{nmk} = 2\sqrt{1 + \frac{\xi_{nmk}^2}{n^2\pi^2a^2}}\,. \tag{1.25}$$

2. Numerical Methods

This chapter focuses on the methods that are used for the numerical computations in this thesis. In order to compute the temporal evolution of a fluid system from its initial state, it is necessary to discretize the equations of motion by using different numerical schemes.
For this purpose various discretization approaches, for example finite element and finite volume methods, exist. Here, the method of finite differences will be introduced for the spatial discretization and a third order Runge-Kutta method for the temporal discretization in time. Furthermore we will introduce the method of artificial compressibility, which can be used to avoid the numerical expensive solution of a Poisson equation. The choice of these methods in combination with the usage of cartesian grids is in particular time saving when performing computations on the GPU as we will see in chapter 3.

2.1 Finite Difference Schemes

In this section a brief introduction to the method of finite differences will be given. The interested reader is referred to [FP99] for a more general overview, which this section is based on. The partial differential equations which are numerically solved in this thesis are of the generalized form

$$\frac{\partial \Phi}{\partial t} = \left(\sum_{\alpha=x,y,z} \left(A_\alpha \frac{\partial}{\partial \alpha} + B_\alpha \frac{\partial^2}{\partial \alpha^2} \right) + C_\alpha + \vec{v}\vec{\nabla} \right) \Phi =: \mathscr{L}\Phi \tag{2.1}$$

where $\Phi(\vec{r}, t) \in \mathbb{R}$ and $\mathscr{L}$ is a differential operator, containing spatial derivatives up to second order and a non-linear advection term. The numerical integration can be divided into two steps: the calculation of $\mathscr{L}$, which we want to discuss in this section and secondly the integration in time. In this section the numerical approximation of $\mathscr{L}$ with finite differences shall be discussed for the one-dimensional case, the implementation for three dimensions will be discussed in chapter 3.
Let $I = \{x \in \mathbb{R} \mid 0 \leq x \leq L\}$ be the domain on which Eq. 2.1 shall be solved. For the discretization I is divided into N equidistant points $x_i = \sum_i \Delta x_i$, with the index $i \in \{[0, N-1] | i \in \mathbb{N}\}$ and $\Delta x_i = x_{i+1} - x_i = L/(N-1)$. Furthermore it is assumed that Φ is a continuous differentiable function. In the vicinity of a grid point x_i, Φ can than be expressed by a Taylor series

$$\Phi(x) = \sum_{n=0}^{\infty} \frac{\partial^n \Phi(x_i)}{\partial x^n} \frac{(x - x_i)^n}{n!}. \tag{2.2}$$

By evaluating the Taylor expansion at different points, different expressions for the first derivative can be obtained. For example the evaluation at x_{i+1} gives[1]

$$\Phi_{i+1} = \Phi_i + (x_{i+1} - x_i)\left(\frac{\partial \Phi}{\partial x}\right)_i + \frac{(x_{i+1} - x_i)^2}{2}\left(\frac{\partial^2 \Phi}{\partial x^2}\right)_i + \mathcal{O}(\Delta x^3) \tag{2.3}$$

$$\Rightarrow \left(\frac{\partial \Phi}{\partial x}\right)_i \approx \frac{\Phi_{i+1} - \Phi_i}{x_{i+1} - x_i}. \tag{2.4}$$

This approximation is denoted as a forward difference scheme (FDS). The same evaluation at x_{i-1} leads to the backward difference scheme (BDS), see Table 2.1. Furthermore it can be shown that the combined evaluation at the points x_{i+1}, x_{i-1} leads to the expression

$$\left(\frac{\partial \Phi}{\partial x}\right)_i = \frac{\Phi_{i+1} - \Phi_{i-1}}{x_{i+1} - x_{i-1}} - \frac{(x_{i+1} - x_i)^2 - (x_i - x_{i-1})^2}{2(x_{i+1} - x_{i-1})}\left(\frac{\partial^2 \Phi}{\partial x^2}\right)_i + \mathcal{O}(\Delta x^3). \tag{2.5}$$

For a constant grid size, that is $\Delta x := \Delta x_i = \text{const.}$, the second order term in equation 2.5 vanishes. By neglecting all terms of higher order, the approximation for $\partial_x \Phi_i$ is of second order. This is the so-called central-difference (CDS) scheme. A comparison of the FD-schemes is given in Table 2.1 The numerical error which is made by neglecting the higher order terms is in general referred to as the truncation error of a finite difference scheme.

[1] From here on the abbreviation $\Phi_i = \Phi(x_i)$ is used.

Scheme-Name	Stencil	Truncation Error	Evaluation at
Forward (FDS)	$\left(\frac{\partial \Phi}{\partial x}\right)_i = \frac{\Phi_{i+1} - \Phi i}{\Delta x}$	$\mathcal{O}(\Delta x^2)$	x_{i+1}
Backward (BDS)	$\left(\frac{\partial \Phi}{\partial x}\right)_i = \frac{\Phi_i - \Phi_{i-1}}{\Delta x}$	$\mathcal{O}(\Delta x^2)$	x_{i-1}
Central (CDS)	$\left(\frac{\partial \Phi}{\partial x}\right)_i = \frac{\Phi_{i+1} - f\Phi i-1}{2\Delta x}$	$\mathcal{O}(\Delta x^3)$	x_{i+1} & x_{i-1}

Table 2.1.: Different finite difference schemes

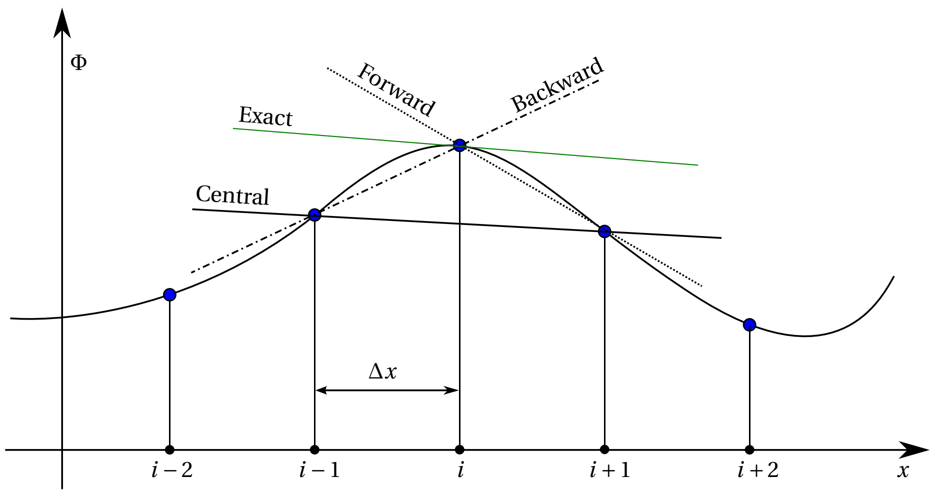


Figure 2.1.: Approximation of the function Φ by different finite difference schemes.

Finally, Fig. 2.1 shows a visual comparison for the three different finite difference schemes. For the computation of the second derivative one approach is to evaluate equation 2.2 halfway between two points at the positions $x_{i\pm\frac{1}{2}}$

$$\left(\frac{\partial^2\Phi}{\partial x^2}\right)\bigg|_{x_i} = \frac{\left(\frac{\partial\Phi}{\partial x}\right)\Big|_{i+\frac{1}{2}} - \left(\frac{\partial\Phi}{\partial x}\right)\Big|_{i-\frac{1}{2}}}{\frac{1}{2}(x_{i+1}-x_{i-1})} + \mathcal{O}(\Delta^2) \approx \frac{\Phi_{i+1}-2\Phi_i+\Phi_{i-1}}{\Delta x^2}. \tag{2.6}$$

For the approximation of the first derivatives in 2.6, it is necessary to use the FDS-scheme at $x_{i+1/2}$ and the BDS-scheme at $x_{i-1/2}$ to obtain a second order accuracy.
So far we introduced methods up to an accuracy of second order. The fourth order methods which are used in this thesis are given by

$$\left(\frac{\partial\Phi}{\partial x}\right)_i \approx \frac{-\Phi_{i+2}+8\Phi_{i+1}-8\Phi_{i-1}+\Phi_{i-2}}{12\Delta x} \tag{2.7}$$

$$\left(\frac{\partial^2\Phi}{\partial x^2}\right)_i \approx \frac{-\Phi_{i+2}+16\Phi_{i+1}-30\Phi_i+16\Phi_{i-1}-\Phi_{i-2}}{12\Delta x^2}. \tag{2.8}$$

The derivation of these equations can be performed in analogy to the second order schemes, but by using more points for the approximation. The interested reader is referred to [For88].

2.2 Runge-Kutta Method

With the finite difference approximation $\mathscr{L}^*$ of the differential operator $\mathscr{L}$ it is possible to solve Eq. 2.1 by separation of variables,

$$\Phi(t) = \Phi(t_0) + \int_{t_0}^{t} \mathscr{L}^*(t, \Phi(t))\,\mathrm{d}t. \tag{2.9}$$

For a numerical computation the integration interval $[t_0, t]$ is split into sub-intervals of length Δt and integrated piecewise. In order to approximate the integration on a sub-interval a Runge-Kutta (RK) method is used, a brief introduction based on [SF] shall be given. The idea behind this method is to iteratively evaluate $\mathscr{L}^*$ at s different times τ_i within one sub-interval Δt.

$$k_i = \mathscr{L}^*\left(\tau_i, \Phi(t_n) + \Delta t\left(\sum_{j=1}^{i-1} a_{ij} k_j\right)\right) \tag{2.10}$$

where $\tau_i = c_i \Delta t$ with $c_i = \sum_{j=1}^{s} a_{ij}$ with $a_{ij} \in \mathbb{R}$. Using a weighted average of the k_i, with the coefficients $b_j \in \mathbb{R}$ gives

$$\Phi(t^{n+1}) = \Phi(t^n) + \Delta t\left(\sum_{j=1}^{s} b_j k_j\right). \tag{2.11}$$

The accuracy of an s-stage RK method depends of the right choice of the coefficients. In this thesis a third-order accurate scheme is used. The coefficients are given by the butcher tableau

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b^T} \end{array} \quad = \quad \begin{array}{c|ccc} 0 & & & \\ \frac{1}{3} & \frac{1}{3} & & \\ \frac{3}{4} & \frac{-3}{16} & \frac{15}{16} & \\ \hline & \frac{1}{6} & \frac{3}{10} & \frac{8}{15} \end{array} \tag{2.12}$$

where the components of **c**, **b** and **A** are a_{ij}, c_i and b_j [SF].
For the implementation of this method it has to be concerned that in the simulations a large number of variables is used and the number of grid points scales with a power of three, with an increase in the resolution. Hence, the required amount of storage should be as small as possible. A low-storage

scheme first introduced by [Wil80] is used in this thesis. The RK method 2.12 is translated into an iterative algorithm by keeping informations from previous computation steps. The required amount of storage is reduced to two registers, one for Φ and one for the intermediate step Q [SF]:

$$\begin{aligned} Q_1 = \Delta t \mathscr{L}^*\left(\Phi^n\right) \quad &\Rightarrow \quad \Phi^1 = \Phi^n + \frac{1}{3}Q_1 \\ \Rightarrow Q_2 = \Delta t \mathscr{L}^*\left(\Phi^1\right) - \frac{5}{9}Q_1 \quad &\Rightarrow \quad \Phi^2 = \Phi^1 + \frac{15}{16}Q_2 \\ \Rightarrow Q_3 = \Delta t \mathscr{L}^*\left(\Phi^2\right) - \frac{153}{128}Q_2 \quad &\Rightarrow \quad \Phi^{n+1} = \Phi^2 + \frac{8}{15}Q_3. \end{aligned} \tag{2.13}$$

2.3 Numerical Stability

It is important to consider the stability of a numerical method. In general it is said that a method is numerically stable if the error introduced by truncation and roundoff errors does not grow over time [FP99].

2.3.1 Runge-Kutta Scheme

Suppose a stationary solution for equation 2.1 exist. The system should remain unaffected by further integration, even with the introduction of small numerical errors. In order to obtain a stability criterion it is sufficient enough to perform a linear stability analysis. A common approach is to study the one-dimensional linearized and diagonalized test case

$$\frac{\mathrm{d}\Phi}{\mathrm{d}t} = \lambda\Phi \tag{2.14}$$

with the eigenvalue $\lambda \in \mathbb{C}$ of a linearized operator [Wil80]. It can be shown that the discretization of equation 2.14 using a RK scheme of s^{th}-order results in the mapping [Wil80]

$$\Phi_{i+1} = \left(\sum_{p=1}^{s} \frac{(\Delta t\lambda)^p}{p!} + 1\right)\Phi_i =: 1 + P(\Delta t\lambda)\Phi_i. \tag{2.15}$$

The discretized system contains a stable fixed point when the condition $|P(\Delta t\lambda)| < 1$ is satisfied. Fig. 2.2 shows the different stability regions for Runge-Kutta schemes up to fourth order. The calculation can be found in the Appendix C.1. For a s order scheme the region of stability is independent of

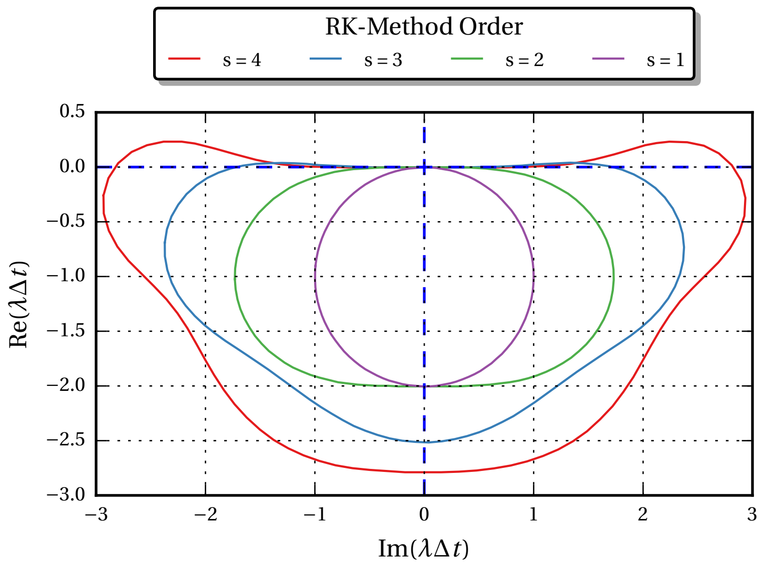


Figure 2.2.: Stability regions Ω_s for different Runge-Kutta methods.

the used implementation [Can+88]. [2] It can be noted that the stability region Ω_s increases with the use of higher order schemes. The substantial detail is that for the methods of third and fourth order, the imaginary axis lies within Ω_s. These methods are often preferred for CFD-simulations, since they tend to stabilize numerical oscillations. [3]

2.3.2 Finite difference schemes

The numerical stability of the finite difference schemes can be estimated with a Von-Neumann stability analysis. In this section the stability criterion will be exemplarily computed for a one-dimensional diffusion equation of the form

$$\frac{\partial \Phi}{\partial t} = \alpha \frac{\partial^2 \Phi}{\partial x^2} \tag{2.16}$$

for a scalar Φ and the diffusion coefficient α. All methods and steps in this example are adapted from [And95]. In order to test if a method is stable it is valid to assume that the numerical error can be expressed in a Fourier mode of the form

$$\epsilon(x,t) = \sum_{m=1}^{N/2} \epsilon_m(x,t) \qquad \text{and} \qquad \epsilon_m(x,t) = e^{at} e^{ik_m x}, \tag{2.17}$$

where a is thhe growth rate of the error, the wavenumber is given by $k_m = {}^{2\pi m}/_{L}$ and L is the size of

[2]For an s-th order RK-scheme different coefficient a_{ij}, b_j and c_i can be used.
[3]Private communication with A.Tilgner

the simulation domain. The index m is restricted by the domain size and the distance Δx between two grid points, i.e. the smallest wave number is resulting from the wavelength $\lambda = L$ and the largest from $\lambda = 2\Delta x$. Due to the linearity of the Laplacian it is sufficient to consider the stability of a single fourier mode ϵ_m. With the use of Eq. 2.6, the discretized version of the diffusion equations reads

$$\frac{\Phi^{n+1}-\Phi^{n}}{\Delta t} = \frac{\alpha}{\Delta x^2}\left(\Phi^n_{i+1} - 2\Phi^n_i + \Phi^n_{i-1}\right). \tag{2.18}$$

In this example the time is discretized by the explicit Euler method, which is equivalent to a RK method of first order. Inserting ϵ_m in Eq. 2.18 for Φ and using trigonometric identities gives

$$e^{a\Delta t} = 1 - \frac{4\alpha\Delta t}{\Delta x^2}\sin^2\left(\frac{k_m \Delta x}{2}\right). \tag{2.19}$$

Furthermore, it holds that

$$\left|\frac{\epsilon^{n+1}_i}{\epsilon^n_i}\right| = \left|\frac{e^{a(t+\Delta t)}e^{ik_m x}}{e^{at}e^{ik_m x}}\right| = \left|e^{a\Delta t}\right|. \tag{2.20}$$

The stability of the finite difference scheme is given when the numerical error does not grow over time. This is fulfilled when the inequality

$$\left|\frac{\epsilon^{n+1}_i}{\epsilon^n_i}\right| = \underbrace{\left|1 - \frac{4\alpha\Delta t}{\Delta x^2}\sin^2\left(\frac{k_m \Delta x}{2}\right)\right|}_{=:G} \le 1 \tag{2.21}$$

holds, where G is denoted as the amplification factor. The stability is fulfilled when $G \le 1$, which holds for

$$\frac{\alpha\Delta t}{\Delta x^2} \le \frac{1}{2}. \tag{2.22}$$

From Eq. 2.22 it can be seen that with an increase in the resolution, the time step has to been scaled by the square of the resolution, in order to maintain stability. The same approach introduced for the diffusion equation can be repeated for a first-order wave equation (see [And95]) and for an advection-diffusion equation (see [FP99]).

As a result two additional stability criteria are given. The first one is (see [And95])

$$\frac{c\Delta t}{\Delta x} \leq 1 \tag{2.23}$$

where c denotes the maximal velocity of the system, for example this could be the speed of sound for a compressible system. The second criterion is given by

$$Pe < 2. \tag{2.24}$$

where $Pe := {}^{u\Delta x}/_{\alpha}$ is defined as the Peclet number [FP99]. The Peclet number can be interpreted as the ratio between the advection and diffusion of a fluid system.

2.3.2.1 Third-Order Upwinding Scheme

For large Peclet numbers it is a common problem that numerical oscillations emerge in the numerical solution. Additional to the central differences the upwinding scheme described in [FP99] is used in this thesis. This method tends to reduce numerical oscillations by interpolating the advection term into the upstream direction of the velocity field. Consider the one-dimensional advection equation

$$\frac{\partial \Phi}{\partial t} + v\frac{\partial \Phi}{\partial x} = 0. \tag{2.25}$$

The discretization for the advection term at the position $x = i$ is given by

$$v\frac{\partial \Phi}{\partial x} \approx v_i \cdot \left(\frac{2\Phi_{i+1} + 3\Phi_i - 6\Phi_{i-1} + \Phi_{i-2}}{6\Delta x} \right) \tag{2.26}$$

for v(x) > 0 and

$$v\frac{\partial \Phi}{\partial x} \approx v_i \cdot \left(\frac{-\Phi_{i+2} + 6\Phi_{i+1} - 3\Phi_{i-1} - 2\Phi_{i-1}}{6\Delta x} \right) \tag{2.27}$$

for v(x) < 0, with the use of third order finite difference approximations from [FP99].

2.4 Numerical Viscosity

In this thesis the finite difference schemes Eq. 2.6 and Eq. 2.8 are used for the approximation of the viscous stress in the Navier-Stokes equation. However, due to the approximation the numerical viscoscity D_N of the approximation may differ from the physical viscosity D_P. An estimation for D_N can be given by considering the diffusion equation

$$\frac{\partial v}{\partial t} = D_P \frac{\partial^2 v}{\partial x^2}. \tag{2.28}$$

A solution is given by $v = V(t)e^{ikx}$. Inserting this ansatz in Eq. 2.28 gives

$$V(t) = V_0 \exp^{-D_P k^2 t}. \tag{2.29}$$

By inserting $V(t)$ into a second order approximation of Eq. 2.28 it follows that

$$\begin{aligned} D_P \Delta v &\approx \frac{D_P}{\Delta x^2}\left(v_{x+\Delta x} - 2v_x + v_{x-\Delta x}\right) && (2.30)\\ \Leftrightarrow -D_P k^2 &\approx \frac{D_P}{\Delta x^2}\left(\exp^{ik\Delta x} - 2 + \exp^{-ik\Delta x}\right) = \frac{D_P}{\Delta x^2}\left(2\cos(k\Delta x) - 2\right) && (2.31)\\ \Rightarrow D_P &\approx -\frac{D_P}{k^2 \Delta x^2}\left(2\cos(k\Delta x) - 2\right). && (2.32) \end{aligned}$$

The numerical viscosity is given by the approximation of D_P from Eq. 2.32,

$$D_N = -\frac{D}{k^2 \Delta x^2}\left(2\cos(k\Delta x) - 2\right). \tag{2.33}$$

The same ansatz for the fourth order scheme gives

$$D_N = -\frac{D}{k^2 \Delta x^2}\left(32\cos(k\Delta) - 2\cos(k\Delta x) - 30\right). \tag{2.34}$$

It can be noted that the numerical viscosity is not constant but depends on the wave number k.

2.5 Method of Artificial Compressibility

For now the fact was ignored that the Navier-Stokes equation cannot directly be solved for the pressure. An equation for p can be obtained by taking the divergence of Eq. 1.7 and apply the continuity equation. The resulting equation is of the form [Lül11]

$$\Delta p = -\nabla \cdot \left((\vec{v}\vec{\nabla})\vec{v} \right). \tag{2.35}$$

In order to determine the pressure it would be necessary to solve this Poisson equation. However, the numerical solution can be very time consuming and difficult to implement on a GPU [4]. Instead here the method of artificial compressibility, first presented in [Cho97], will be introduced. The concept of this method is to introduce the artificial equations of state

$$\frac{\partial \rho}{\partial t} + \nabla \vec{v} = 0 \quad (2.36) \qquad p = \rho/\delta \quad (2.37) \qquad c = \frac{1}{\delta^{1/2}} \quad (2.38)$$

where ρ is the artificial density, δ the artificial compressibility and c the artificial speed of sound. Eq. 2.36 is a modified version of the continuity equation and Eq. 2.37 introduces an artificial equation for the pressure. A substitution of Eq. 2.37 into the Navier-Stokes equation gives

$$\frac{\partial v}{\partial t} + \left(\vec{v}\vec{\nabla}\right)\vec{v} = -c^2 \nabla \rho + Re \Delta \vec{v} + \sum \vec{F}_{\text{ext.}} \tag{2.39}$$

The system now has an artificial compressibility. The motivation behind this approach is to set the artificial speed of sound to a high value such that the Mach number given by

$$Ma = \frac{v}{c} = \frac{1}{c} \max \left(\sum_i |\vec{u}_i|^2 \right)^{1/2} \tag{2.40}$$

is small, a well estimation is $Ma < 0.1$. [4] In this case the fluid can be considered as incompressible. One major drawback of this method is a stiff numerical problem. Due to the stability criterion (see Eq. 2.23) a small time step is needed since $v = c$.

[4] Private Communication with A.Tilgner

3. GPGPU-Based Implementation of a Finite Difference Algorithm

The objective of this chapter is to give an overview of the basic implementation of an algorithm using finite difference schemes. The source code used for the computations is based on an existing and validated version from A. Tilgner. It was furthermore extended by the immersed boundary methods that will be explained in more detail in chapter 4. In this context an introduction to the aspects of GPGPU[1]-Computing with the CUDA [2] architecture will is given. For the computations in this thesis Tesla C1060 and Tesla K20m GPUs by NVIDIA were used. In this chapter all informations related to GPGPU-Computing and CUDA are adapted from the CUDA Programming Guide and the CUDA Best Practices Guide from NVIDIA [NVI15b], [NVI15a]. All details about the implementation of the algorithm were obtained by studying the source code and from private communication with A.Tilgner.

3.1 GPGPU-Computing with CUDA

CUDA is an architecture developed by NVIDIA to enable an easy approach to the implementation of GPGPU-based algorithms. The underlying idea is to hide the complexity of the hardware under a more high oriented and generalized software abstraction layer. This is done by introducing some additional programming language extensions for example in C/C++. Furthermore, it is necessary to use a CUDA suited compiler like NVCC.

3.1.1 Hardware and Memory Architecture

A cuda device contains an array of so called streaming multiprocessors (SM). Each of these SMs contains a group of small execution units, which are called cuda cores. For the exchange of data between different SMs, the cuda cores and the CPU side of the computer, different types of memory are existing, as shown in Fig. 3.1.

[1] General Purpose Computation on Graphics Processing Units
[2] Computing Uniform Device Architecture

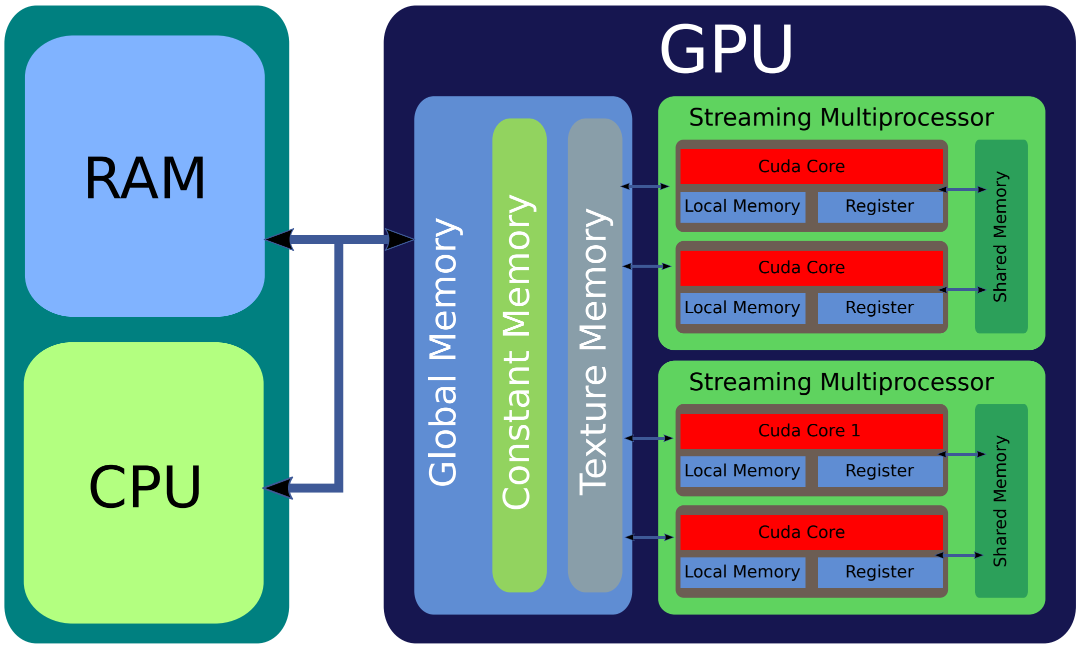


Figure 3.1.: Memory layout of a Nvidia-GPU

Global Memory The global memory is the largest memory on the GPU and can be accessed by all cuda cores. It is used to exchange data with the RAM and distribute it between different SMs. Writing and reading from the global memory is slow. A part of the global memory can be used furthermore as a constant memory. In this case multiple memory accesses are temporary[3] cached. In addition the use of the global memory as a texture memory is possible. The memory access is then spatially [4] cached.

Shared Memory Each SM contains a shared memory, much smaller than the global memory.[5] It is accessible by all cuda cores of the SM. This memory type is much faster (x100) than the global memory. It is useful when multiple operations by different cuda cores of one SM are carried out on the same data.

Register / Local Memory Each cuda core possesses an own set of registers and local memory. The register access is slightly faster than the access on the shared memory. Local memory access is very slow since the storage is created on the global memory when a cuda core runs out of registers.

[3]Memory access is cached when multiple reads are performed on the same memory location.
[4]Memory access is cached when multiple reads are performed simultaneous on adjacent memory locations.
[5]Around 16kb on c1060 and 64kb on k20m.

3.1.2 Thread Management

In order to distribute the workload between different SMs and cuda cores an additional syntax is necessary. Therefore, the CUDA API introduces the notion of a thread and a block.

Thread A thread is the software abstraction of a single cuda core. It can be identified with a maximum of three IDs, **threadIdx.x**, **threadIdx.y** and **threadIdx.z**.

Block A block is defined as a group of threads, i.e. in Fig. 3.2 a block contains 64 threads. There is no direct hardware equivalent to a block. A block is identified by a maximum of two IDs, **blockIdx.x** and **blockIdx.y**

Blocksize The blocksize is defined before running a function on the GPU. It determines how many threads run inside a block. In Fig. 3.2 the blocksize is set to (8, 8) and each thread is defined by a **threadIdx.x** and **threadIdx.y**.

Gridsize The gridsize defines the total number of blocks, i.e. (2, 2) in the example.

The concept of a grid containing blocks of threads might be confusing but these abstractions are necessary to enable a high and parallelized workload on the GPU, since each block can be dynamically assigned to a different SM.

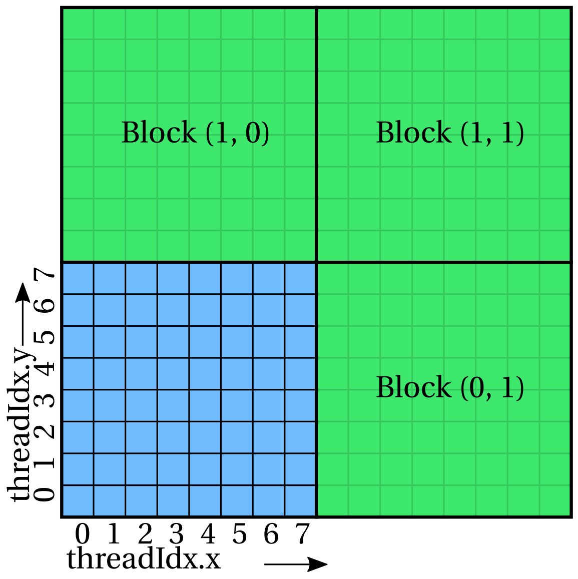


Figure 3.2.: Thread management of the CUDA API, exemplarily shown for a grid of blocks. Each block contains a sub-grid with 8x8 threads.

3.1.3 Performance Considerations

For the programming of a CUDA device a vast number of optimization techniques exist [NVI15a]. A short description of the most import ones is presented below.

Memory Coalescing During the runtime a SM executes a number of 32 threads in parallel, this thread group is denoted as a warp. In order to enable a fast memory access on the global memory, threads which are adjacent in a warp should read from adjacent memory cells. This access pattern is denoted as coalesced memory access. Inhomogeneous access pattern on the global memory can lead to so called bank conflicts. As a result the performance of a cuda program can drastically decrease.

Sparse global Memory Access The global memory should be used for data transfer and distribution with as little as possible global reads and writes. This reason for this approach is the slow global memory access ($\times 100$) in comparison to a fast shared memory access.

Avoidance of Branch Divergence All threads in a single branch should execute the same command. Conditional statements like `if..then..else` create divergent code. The execution is performed serial and the performance decreases.

3.1.4 Simulation Domain and Memory Layout

The three-dimensional simulation domain is given by a cuboid which is discretized by a collocated cartesian grid with a constant step size in each direction. From this section on the following conventions will be used.

l_i The length of the cuboid in x, y or z direction, $i \in x, y, z$

N_i The number of grid points in x, y or z direction, $i \in x, y, z$

$\Delta i(l_i, N_i)$ The step size between to grid points in x, y or z direction, $i \in x, y, z$

A single grid point is determined by the set of indices i, j, k where $0 \leq i < N_x$, $0 \leq j < N_y$ and $0 \leq k < N_z$. The position vector is given by $\vec{r} = (i \cdot \Delta x, j \cdot \Delta y, k \cdot \Delta z)^T$. Each grid point lies in the center of a grid cell of size $\Delta x \cdot \Delta y \cdot \Delta z$. Since a collocated grid is used all variables are located in the cell center. [6] For the computation a variable $\Phi \in \{v_x, v_y, v_z, \rho\}$, is stored into three registers

C_Φ Located on the RAM. Used mainly for data transfer and analysis

R_Φ Located on the global memory on GPU. A variable Φ is stored after the computation of a time step.

[6]For example the velocity components are stored on the cell faces on a staggered grid.

B_Φ Located on the global memory on GPU. This is the buffer register for the Williamson RK scheme, see Sec. 2.2.

The size of a storage register is given by $(N_x+4)(N_y+4)(N_z+4)$. An additional offset +4 is given since two additional layers of ghost points are attached to each side of the simulation domain. This points are used for a computation of the boundary conditions, see Sec. 3.2.1. All registers have a one-dimensional memory layout. Hence, the register position of a grid point is calculated by

$$P(i,j,k) = i(N_y+4)(N_z+4) + j(N_z+4) + k \tag{3.1}$$

which is a row-major access pattern.

3.2 Time Step Algorithm

In this section an overview of the computation of a single time step is presented. For this example the Navier-Stokes equation, given by Eq. 1.7, is used. Before the computation of the time step all initial conditions are saved to the registers C_Φ and R_Φ. The time integration is performed with the Willamson-RK3 scheme, see Sec. 2.2. This schemes consist of three sub time step, in this example only the first step given by

$$Q_1 = \Delta t \mathscr{L}^* \left(\Phi^n\right) \qquad \Rightarrow \qquad \Phi^1 = \Phi^n + \frac{1}{3} Q_1 \tag{3.2}$$

will be considered. The computation of the remaining steps is similar.
The simulation domain is divided into a grid of $N_y/8 \times N_z/8$ blocks. On each of these blocks the integration is performed separately. This segmentation is similar to the example introduced in Sec. 3.1.2 (see Fig. 3.2), except an additional index **threadIdx.z** is used. This index seperates the variables during the integration. For example for ρ it is threadIdx.z = 0, for v_x it is threadIdx.z=1 and so respective. with this convention each block contains a total number of 8x8x4 threads. For a computation of the finite difference schemes a 5-Point stencil is used, as shown in Fig. 3.3. Each cell in this stencil corresponds to one grid cell in the numerical domain. The inner points of the stencil are enumerated with 1, the outer points with 2.

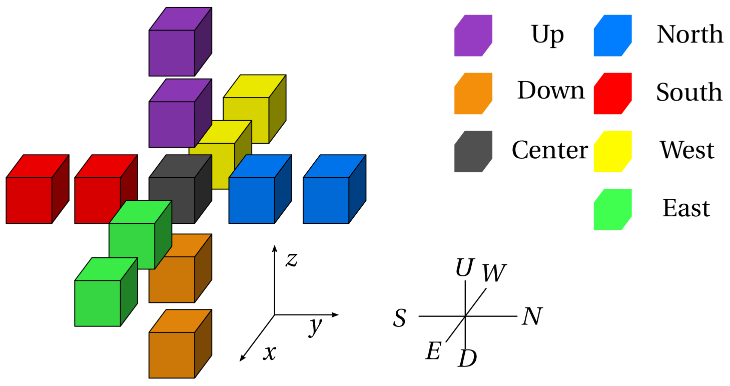


Figure 3.3.: 5-Point stencil used for the discretization, with the directions North, West, South, East, Up, Down and the Center.

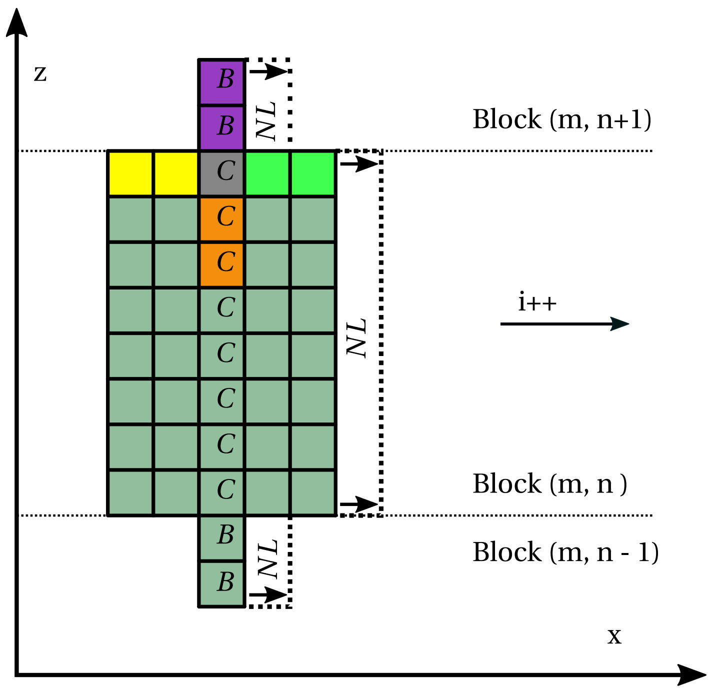


Figure 3.4.: Shared memory block location in the simulation domain during a time step. With the center points (C), additional loaded boundary points (B) and the next layer (NL).

For example in x direction the points are denoted as West2, West, Center, East and East2 in x direction or with the abbreviations W2, W1, C, E1 and E2. For second order finite difference schemes only the inner points are used. It is referred to a 3-Point stencil in this case. The computation of one time step can be separated into an initialization part and a for-loop which iterates by the index i, over the simulation domain. In the initialization part a block from the simulation domain is loaded into the shared memory of the GPU. As a first step each thread is assigned to a grid point (i, j, k).

$$\begin{aligned} i &= 2, \\ j &= \text{blockIdx.x} \cdot 8 + \text{threadIdx.y} + 2, \\ k &= \text{blockIdx.y} \cdot 8 + \text{threadIdx.x} + 2. \end{aligned} \tag{3.3}$$

The position of the Center C is given by $P(i, j, k)$ for each thread. Then each thread has to load the neighbor points W2, W1, E1 and E2. The offset to C is given by a multiple of $L = (N_y + 4)(N_z + 4)$, for example $W2 = C - 2L$. Furthermore, on the sides of the Block additional boundary points (B) are necessary for the discretization in the directions U, D, N and S. With the calculated positions the variables are now loaded into the shared memory of the GPU. How this memory structure is located in the simulation domain for $i > 2$ is shown inFig. 3.4. The 5-Point stencil is exemplarily shown for one center point and the boundary points (B) on the sides of the block are shown. The initialization procedure is now finished.
In the first step of the for-loop the spatial derivatives are computed. Since these terms are quite large this will be shown exemplarily for the laplace operator. A central difference scheme of fourth order, given by Eq. 2.8, is used for the v_x component.

$$\begin{aligned} \Delta v_x \approx & \frac{1}{12\Delta x^2}\left(-v_x(i+2,j,k) + 16v_x(i+1,j,k) - 30v_x(i,j,k) + 16v_x(i-1,j,k) - v_x(i-2,j,k)\right) \\ & + \frac{1}{12\Delta y^2}\left(-v_x(i,j+2,k) + 16v_x(i,j+1,k) - 30v_x(i,j,k) + 16v_x(i,j-1,k) - v_x(i,j-2,k)\right) \\ & + \frac{1}{12\Delta z^2}\left(-v_x(i,j,k+2) + 16v_x(i,j,k+1) - 30v_x(i,j,k) + 16v_x(i,j,k-1) - v_x(i,j,k-2)\right) \end{aligned} \tag{3.4}$$

The intermediate step Q_Φ is than computed with the conditionals

```
**if** (threadIdx.z==0) **then**
    $Q_\rho = -\nabla \vec{v}$
**end if**
**if** (threadIdx.z==1) **then**
    $Q_{v_x} = -(\vec{v}\nabla)v_x - c^2\partial_x\rho + \frac{1}{Re}\partial_x^2 v_x + f_x$
**end if**
```

```
**if** (threadIdx.z==2) **then**
    $Q_{v_y} = -(\vec{v}\nabla)v_y - c^2\partial_y\rho + \frac{1}{Re}\partial_y^2 v_y + f_y$
**end if**
**if** (threadIdx.z==3) **then**
    $Q_{v_z} = -(\vec{v}\nabla)v_z - c^2\partial_z\rho + \frac{1}{Re}\partial_z^2 v_z + f_z$
**end if**
```

where $(f_x, f_y, f_z)^T = \vec{F}_{\text{ext.}}$. The advection term $(\vec{v}\nabla)\vec{v}$ is discretized by the third-order upwinding scheme from Sec. 2.3.2.1 and all remaining derivatives are approximated with the fourth order finite difference schemes from Eq. 2.8. After this step the computation for the center points (C) is finished. The results are stored in the buffer registers R_Φ.

In the next pass of the loop the index i is incremented and the algorithm loads the next layer (NL) as indicated in Fig.3.4. All other layers are shifted by one position. With this approach each layer only gets only loaded once into the shared memory. After the for-loop has iterated over the simulation domain the first part of the time step is finished. The remaining computation is

$$\Phi^1 = \Phi^n + \frac{1}{3}Q_1. \tag{3.5}$$

The algorithm for this computation performs identical except no derivatives need to be calculated and the border points (B) are redundant.

3.2.1 Boundary Conditions

For the implementation of the boundary conditions two additional layers of grid points are added on each side of the simulation domain. In this section the implementation of boundary conditions is presented for the one-dimensional case in x-direction.

3.2.1.1 No-Slip Boundaries

Consider the left side of the simulation domain. The ghost points are given by the points $g_{l2} = (0, j, k)$, $g_{l1} = (1, j, k)$, the boundary is at $b = (2, j, k)$ where $\vec{v} = 0$ and the next inner points are $i_{l1} = (3, j, k)$ and $i_{l2} = (4, j, k)$. For No-Slip boundaries the velocity profile is mirrored and inverted at b.

$$v_x(g_{l2}) = -v_x(i_{l2}) \qquad , \qquad v_x(g_{l1}) = -v_x(i_{l1}). \tag{3.6}$$

The same procedure is performed for v_y and v_z. Due to the symmetry of the velocity profile on the boundary the viscous stress vanishes and $\partial_t \vec{v} = 0$. Hence, $\vec{v}(t) = 0$.

3.2.1.2 No-Flux Boundaries

Consider the left side of the simulation domain. The ghost points are given by the points $g_{l2} = (0, j, k)$, $g_{l1} = (1, j, k)$, the boundary is set to $b = (2, j, k)$ and the next inner points are $i_{l1} = (3, j, k)$ and $i_{l2} = (4, j, k)$. For No-Flux boundaries it is necessary that $\partial_x \Phi = 0$ for any variable Φ. By setting

$$\Phi(g_{l2}) = \Phi(i_{l2}) \qquad , \qquad \Phi(g_{l2}) = \Phi(i_{l1}) \tag{3.7}$$

it follows that

$$\frac{\partial \Phi}{\partial x} = \frac{1}{12\Delta x} - \Phi(g_{l2}) + 8\Phi(g_{l1}) - 8\Phi(i_{l1}) + \Phi(i_{l2}) = 0 \tag{3.8}$$

with the use of a fourth order central difference scheme. This boundary condition is used for ρ to satisfy the mass conservation.

3.2.1.3 Free-Slip Boundaries

The Free-Slip boundaries are implemented by using No-Flux boundaries for the v_y and v_z components and No-Slip boundaries for the v_x component.

3.2.1.4 Periodic Boundaries

Consider the ghost points $g_l 2 = (0, j, k)$, $g_l 1 = (1, j, k)$ and the inner points $i_{l1} = (2, j, k)$ and $i_{l2} = (3, j, k)$ on the left side of the simulation domain. On the right side of the simulation domain the ghost points are $g_{r2} = (N_x + 1, j, k)$, $g_{r1} = (N_x, j, k)$ and the inner points $i_{r1} = (N_x - 1, j, k)$ and $i_{r2} = (N_x - 2, j, k)$. For Periodic boundaries the regions on the left and right side of the simulation domain are simply exchanged by

$$v_x(g_{l2}) = v(i_{r2}) \qquad , \qquad v(g_{l1}) = v(i_{r1}), \tag{3.9}$$

$$v_x(g_{r2}) = v(i_{l2}) \qquad , \qquad v(g_{r1}) = v(i_{l1}). \tag{3.10}$$

The same procedure is performed for v_y and v_z and ρ.

4. Immersed Boundary Methods

4.1 Introduction

For many problems in computational fluid mechanics it is mandatory to solve the equations of motion with respect to complex-shaped geometries. The algorithm introduced in chapter 3 is not yet suitable for such a scenario. For instance the simulation inside a spheric geometry is not possible, since the boundaries do not coincide with the implemented cartesian grid. Nevertheless, different approaches are existing to overcome this problem, which shall be introduced here.
One common approach to extend the algorithm is the use of a body-fitted mesh, different advantages and disadvantages arise with this kind of implementation, see [MI05]. One benefit is a much simpler deployment of the desired boundary condition, due to the overlap of the numerical grid with the domain borders. Furthermore a higher accuracy can be achieved [Gor13]. However, using an such grids generates computational overhead, during and before the execution of a simulation. The grid generation algorithm is complicated in contrast to the use of a cartesian grid. This can be even more difficult when moving boundaries are considered. Moreover, solving the finite difference schemes on a curvilinear coordinate system requires more calculations for a single grid point. The last important aspect is the implementation on the GPU. Like discussed in section 3.1.3 it is more efficient to use homogeneous storage and calculation pattern on a CUDA-device. Using complex data structures that are required for body-fitted meshes makes this very difficult.
A set of alternative methods to resolve the problems described above, are so called Immersed Boundary (IB) methods[1]. The term was first mentioned in [Pes72], where the method was used for the simulation of blood flow through a heart valve. Since then this term has been used for a variety of methods [MI05]. All of them have the idea in common to perform the simulations on a cartesian grid which does not conform to the domain boundary. To satisfy the desired boundary conditions additional terms are introduced into the equations of motion. In general it can be distinguished between continuous forcing methods and direct forcing methods. Continuous forcing methods try to mimic the boundary using a localized forcing term which acts on all points near to the boundary. Since the surface is often described by Lagrangian points, these methods are well suited for moving boundaries [MI05]. Direct forcing approaches try to satisfies the boundary conditions by imposing them directly to points at the fluid boundaries often by using an interpolation procedures. Some of

[1]From here on the term IB method is used as an abbreviation.

the major drawbacks using IB methods is the loss in spatial accuracy at the boundary due to interpolation or approximations. Therefore, it can be necessary to use a higher grid resolution compared to a body-fitted mesh. The benefits of these methods is the use of a cartesian grid, which is more suitable for a GPU-based implementation, see Sec. 3.1.2. One objective of this chapter is to introduce implementation of No-Slip boundaries for cartesian grids, with the use of IB methods. The term IB method is vaguely defined in literature, in this thesis we refer with it to all methods introduced in this chapter.

4.2 Immersed Boundary Methods

For the purpose of discussion the different methods introduced here will be applied to a default geometry. The fluid domain without any immersed boundaries is set to a cube of the size $l_i = 1$ with $i \in \{x, y, z\}$. For the discretization we choose $N_i = 32$ for the number of grid points. As an example of an immersed boundary, the embedding of a cylinder will be discussed. The surface equation is

$$\left(x - \frac{l_x}{2}\right)^2 + \left(y - \frac{l_y}{2}\right)^2 = r^2 \tag{4.1}$$

where $r = 0.4$ is the radius and the center is given by $(l_x/2, l_y/2)$. The simulation domain is than separated into the fluid domain Ω_f and the wall domain Ω_w. The overall goal is to enforce the No-Slip condition $\vec{v} = 0$ on the surface $\partial\Omega_f$, furthermore the conservation of mass should be fulfilled.

4.2.1 Volume Penalization

The concept behind the volume penalization method is to introduce an additional forcing term into the Navier-Stokes equation, which acts on the grid points outside of the fluid domain to ensure the desired boundary conditions. The methods was successfully implemented and tested on pseudo-spectral methods, see for example [Lül11]. For the implementation it is necessary to define a masking function $H(x, y, z)$ which separates the simulation domain into Ω_f and Ω_w. For a cylinder the surface equation can be modified to

$$H(x,y,z) = \begin{cases} 0, & \left(x - \frac{l_x}{2}\right)^2 + \left(y - \frac{l_y}{2}\right)^2 < r^2 \\ 1, & \text{else} \end{cases} . \tag{4.2}$$

With this function an additional forcing term can be added into the Navier-Stokes equation

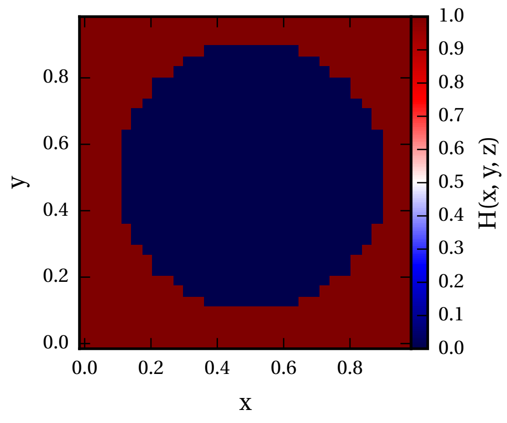


Figure 4.1.: Masking array $H(x, y, z = const.)$ for a cylinder, in the (x,y) plane.

$$\frac{\partial v}{\partial t} + \vec{v} \cdot \vec{\nabla} \vec{v} = -c^2 \nabla \rho + \frac{1}{Re} \Delta \vec{v} + \vec{F}_{ext} + \underbrace{\frac{H(x,y,z)}{\eta} (\vec{v} - \vec{v}_0)}_{\text{Vol. Pen. Forcing}} . \tag{4.3}$$

Where η is a regulation parameter also denoted as damping rate of the forcing and $\vec{v}_0$ is the desired Dirichlet boundary condition. Hence, for No-Slip boundaries it is $\vec{v}_0 = 0$. The additional term in Eq. 4.3 acts as an exponential damping force on a single grid point. The forcing acts inside of Ω_w as $H(x, y, z) = 1$ and vanishes in Ω_f. The strength of the forcing is scaled by the damping rate η. For a small η a stronger forcing is applied to the points in Ω_w. It can be shown that for $\eta \to 0$ the velocity in Ω_w converges against zero and the No-Slip boundaries are fulfilled [Lül11]. However, the additional term in Eq. 4.3 also introduces an additional stability criterion [Lül11]

$$\Delta t < \Delta t_{VP} := \eta. \tag{4.4}$$

The time step is already severely restricted due to the method of artificial compressibility, see Sec. 2.5. Hence, the criterion given by Eq. 4.4 is not of concern. The implementation of the Volume-Penalization method is rather simple. The masking function is defined before runtime. During runtime for each grid point (i, j, k) the value $H(i\Delta x, j\Delta y, k\Delta z)$, is stored into an additional array located on the global memory of the GPU. This value is than loaded during the computation of a time step. For the cylinder this masking array is show in Fig. 4.1. It can be noted that the major-drawback of this method is a pixelated border. The exact boundary can only be satisfied for Δx, Δy and $\Delta z \to 0$.

4.2.2 Direct Forcing

One concern of the Volume-Penalization method is the remaining velocity in the wall domain close to the boundaries. The Direct-Forcing method bypasses this restriction by an implicit calculation of the forcing term. This method was first mentioned by [Moh97] and is discussed in detail in [Fad+00]. With the use of the Euler method the discretization of the Navier-Stokes Equation can be written as

$$\frac{\vec{v}^{n+1}-\vec{v}^{n}}{\Delta t}=\mathscr{L}^{*}+\vec{f} \tag{4.5}$$

where $\mathscr{L}^{*}$ are the discretized advection and viscous terms and the density gradient and $\vec{f}$ is the forcing term. For a boundary point the condition $\vec{u}^{n+1}=\vec{u}_0$ should hold. By substituting this expression into Eq. 4.5 it follows that

$$\frac{\vec{v}_0-\vec{v}^{n}}{\Delta t}=\mathscr{L}+\vec{f}\Rightarrow\vec{f}=\frac{\vec{v}_0-\vec{v}^{n}}{\Delta t\cdot\mathscr{L}}. \tag{4.6}$$

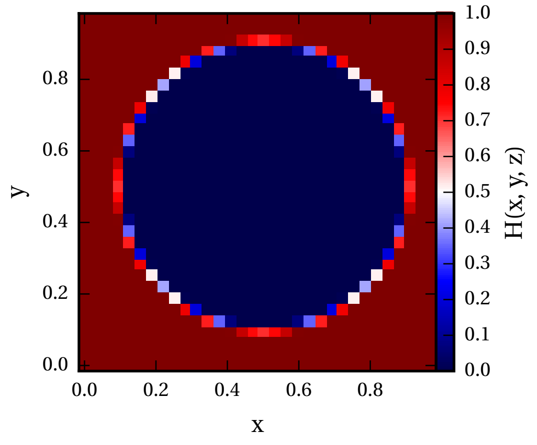
The implementation of this method is similar to the Volume-Penalization method. A masking array is computed before the execution of a simulation. The forcing term is than added into the register Q_Φ (see Sec. 3.2) after the computation of $\mathscr{L}$. It can be noted that this procedure is equivalent to simply setting $\vec{v}=0$ after a time step. However, this is not possible if the boundaries are further improved, for example by the Volume-Fraction method discussed below. Yet the major-drawback of the Direct-Forcing method is a pixelated border. In contrast to the Volume-Penalization method no further stability criteria are introduced.

4.2.3 Volume Fraction Interpolation

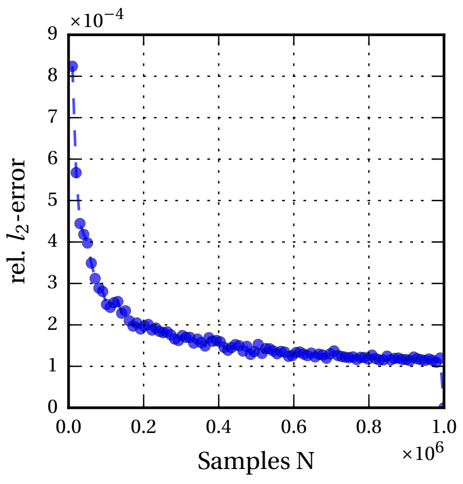
The methods introduced so far lack the ability of an exact implementation of the boundary conditions on Ω_f. Instead the surface $\partial\Omega_f$ is described by the nearest grid points in Ω_w, resulting in a stepwise approximation. To overcome this problem a simple procedure is the use of an volume fraction interpolation scheme, introduced in [Fad+00].

The advantage of this method is a simple implementation into the volume penalization and direct forcing method. Furthermore, the overall computation time of the time step stays constant, in contrast to complex interpolation schemes.

Initially the interpolation begins the determination of all grid cells which are cut by the surface $\partial\Omega_f$. For each of these boundary cells the total volume V_w of the wall domain Ω_w inside each cell is computed. This method can be applied to the Volume-Penalization and the Direct-Forcing method. The force acting on the points inside the boundary cells is than weighted by a scaling factor, $\Phi=V_W/(\Delta x\Delta y\Delta z)$.



(a) Masking array H(x,y,z=const.) for the Volume-Fraction method



(b) Convergence of the relative l_2-error of the Monte Carlo integration.

Figure 4.2.

For the implementation an improved version of the masking array H is used. Like in the previous methods the masking array is precomputed and loaded at runtime. The algorithm is extended to compute the volume fractions of the cells lying next to the surface $\partial\Omega_f$. Since an exact computation is difficult to implement for arbitrary surfaces one possibility is to use a Monte Carlo integration method. For each cell N random samples are generated. Than the volume fraction is simply given by the ratio of samples lying in- and outside outside the fluid domain. This is determined by using the masking function $H(x, y, z)$.

An example for the generated masking array with $N = 2 \cdot 10^5$, is shown in Fig. 4.2a. The symmetric distribution of the boundary values indicates a well approximation. For a better evaluation a convergence study was performed with the number of samples N varied between 100 and 10^6 points. The l_2-error was determined by comparing the results to the computation with the highest number of samples. The results are shown in Fig. 4.2b. For the number of samples in the interval below $N = 2 \cdot 10^5$ a fast decay in the error to the order $2 \cdot 10^{-4}$ can be observed. Above this interval the error is of the order 10^{-4}. It can be concluded that the integration should be accurate enough by using $N = 10^5$ samples.

4.2.4 Interpolation Method

The last IB method introduced in this thesis is a bilinear interpolation method, first presented by [GSB03]. The main difference to the previous methods is an exact representation of the immersed boundary, instead of a pixelated one. The interpolation method is applied to all grid points which posses a nearest neighbor outside of the fluid domain. A schematic representation of the interpolation procedure is shown in Fig. 4.3, for the point b, for which the velocity $\vec{v}$, shall be determined.

The idea of this method is to linearly interpolate b into the direction of the normal $\vec{n}$, of the immersed boundary. a is defined as the closest surface point where the normal of the surface meets point b at a distance $\Delta h = [ab]$.

The point b is projected into the direction of the normal $\vec{n}$, onto the plane defined by the nearest grid points α, β, γ and δ. After the position calculation of the point c, the velocity $\vec{v}(c)$ is obtained by a bilinear interpolation of the velocities at the points α, β, γ and δ, that is [Pre+07]

$$\vec{v}(c) = \frac{1}{\Delta x \Delta y}\big(\vec{v}(\beta)(\Delta x - c_1)(\Delta y - c_2) + \vec{v}(\gamma)(\Delta x - c_1)(c_2) + \vec{v}(\alpha)(c_1)(\Delta y - c_2) + \vec{v}(\delta)(c_1)(c_2)\big). \quad (4.7)$$

(c_1, c_2) is the offset from c to β, or in general the offset to the point of the plane, which is closest to b. With the obtained velocity $v(c)$ and the No-Slip boundary condition $v(a) = 0$, the interpolated velocity

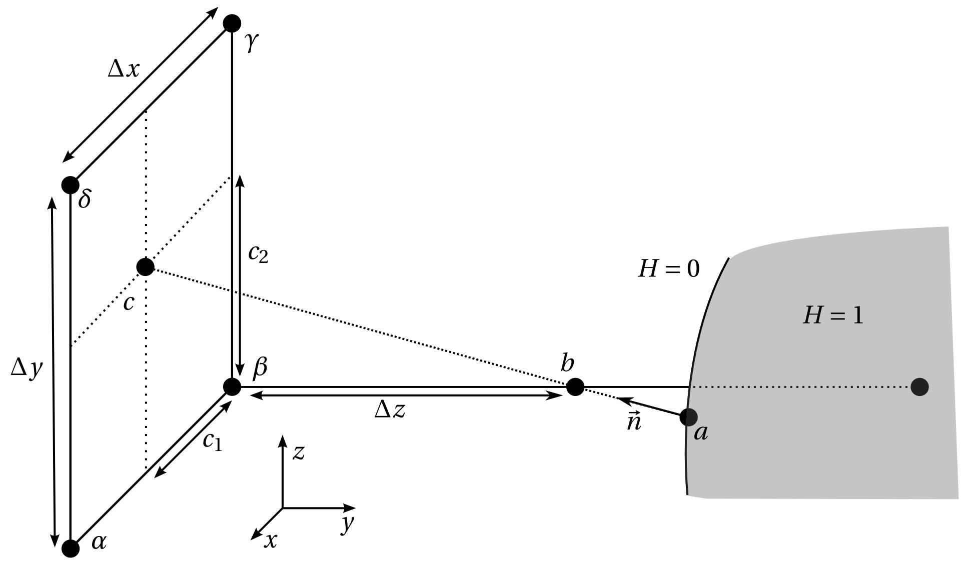


Figure 4.3.: Interpolation of the velocity of the point b. The point c is a projection from the point a into the direction of the normal $\vec{n}$, onto the plane defined by the grid points α, β, γ and δ. The immersed boundary is given by the masking function H(x,y,z). For the bilinear interpolation the coefficient c_1 and c_2 are used.

is given by

$$\vec{v}(b) = \vec{v}(c)\frac{\Delta c}{\Delta h} \tag{4.8}$$

where $\Delta c = [ac]$.

For a simplification of the interpolation procedure for different geometries, a preprocessing routine was implemented. In order to interpolate a certain geometry three functions have to be defined. A masking function $H(x,y,z)$, a function which gives the distance to the boundary $d(x,y,z)$ and a function $\vec{g}(x,y,z)$ which determines the normal of the boundary. For the cylinder the masking function $H(x,y,z)$ is given by Eq. 4.2. The distance is

$$d(x,y,z) = \sqrt{\left(x-\frac{l_x}{2}\right)^2 + \left(y-\frac{l_y}{2}\right)^2 - r^2} \tag{4.9}$$

and the normal direction

$$\vec{g}(x,y,z) = \left(-\left(x-\frac{l_x}{2}\right), -\left(y-\frac{l_y}{2}\right), 0\right)^T. \tag{4.10}$$

A normalization of $\vec{g}(x,y,z)$ is performed in the algorithm. However, it is required that the normals point into the direction of the fluid domain. Before the execution of a simulation the following steps are precomputed: [2]

1. The closest grid point to b, in the example β is determined by the largest component of $\vec{n}$, in this example the z-direction with $n_0 := \max(\vec{n}) = n_z$.
2. The points α, γ, δ are determined by the remaining components n_y, n_z, which are used as directions to compute the offset from β.
3. The offsets $p00$, $p01$, $p10$, $p11$ of the points β, α, γ, δ to the point b are computed.
4. $\vec{n}$ is normalized by $\vec{n}^* = \frac{\vec{n}}{|n_0|}\cdot\Delta z$. [3] The length of $\vec{n}^*$ is $[cb]$.
5. The point c is given by $c = a + \vec{n}^*$ and the coefficient c_1 and c_2 are computed and normalized to the values $c01 := \frac{c_1}{\Delta x}$ and $c10 := \frac{c_1}{\Delta y}$.
6. The ratio $W = \frac{\Delta c}{\Delta h} = \frac{d(x,y,z)}{d(x,y,z)+|\vec{n}|}$ is computed.

The values obtained from this pre-computation are the offsets $p00$, $p01$, $p10$, $p11$, the coefficients $c01$, $c10$ and the ratio W. Each values is stored into an array of shape $N_x \cdot N_y \cdot N_z$, which is loaded

[2] This computation is exemplarily shown for the case where $\vec{n}$ is the largest in z direction, like in Fig.4.3.

[3] For $n_0 = n_z$ this is Δx and for $n_0 = n_y$ it is Δy.

during the simulation. All values in the coefficient arrays which are not close to the immersed boundary are set to zero. During the interpolation procedure the ratio W is checked and if larger than zero an interpolation for the grid point is computed. An example of the implemented interpolation procedure is given by:

```
if ((distance != 0 )){
  //load points of the plane (alpha, beta, gamma, delta)
  p00 = p00_d[global_Point] + global_Point;
  p01 = p01_d[global_Point] + global_Point;
  p10 = p10_d[global_Point] + global_Point;
  p11 = p11_d[global_Point] + global_Point;
  //load coefficients
  c01 = c01_d[global_Point];
  c10 = c10_d[global_Point];

  if (threadIdx.z == 1){
    //load velocities
    f00 = vx_d[p00]; f01 = vx_d[p01]; f10 = vx_d[p10]; f11 = vx_d[p11];
    //bilinear and linear interpolation
    f = W*(f00*(1 - c10)*(1 - c01) + f10*c10*(1 - c01)
                     + f01*c01*(1 - c10) + f11*c01*c10);
    //set the interpolated value
    vx_d[global_Point] = f;
  }
  if (threadIdx.z == 2){
    f00 = vy_d[p00]; f01 = vy_d[p01]; f10 = vy_d[p10]; f11 = vy_d[p11];
    f = W*(f00*(1 - c10)*(1 - c01) + f10*c10*(1 - c01)
                        + f01*c01*(1 - c10) + f11*c01*c10);
    vy_d[global_Point] = f;
  }
  if (threadIdx.z == 3){
    f00 = vz_d[p00]; f01 = vz_d[p01]; f10 = vz_d[p10]; f11 = vz_d[p11];
    f = W*(f00*(1 - c10)*(1 - c01) + f10*c10*(1 - c01)
                     + f01*c01*(1 - c10) + f11*c01*c10);
    vz_d[global_Point] = f;
  }
```

where `global_Point` is the position of the interpolated point.

4.3 Validation

This section presents the numerical validation of the Immersed Boundary methods. The objective of the validation is to determine the numerical accuracy and the numerical stability for each method. Furthermore, it is important to test if conservation laws, in particular the conservation of mass, are fulfilled. For the validation three different test problems were chosen. The flow between two parallel planes which is also known as planar Poiseuille flow, the flow in a Pipe which is referred to as Hagen-Poiseuille flow and the flow between two rotating cylinders which is known as the Taylor-Couette system.

4.3.1 Conventions

In this section the following abbreviations will be used

VP	Volume-Penalization Method	**VF**	Volume-Fraction Method
DF	Direct Forcing Method	**FD2**	Finite Difference Schemes of second order
IP	Interpolation Method	**FD4**	Finite Difference Schemes of fourth order

For example, DF-VF FD2 method, refers to the Direct-Forcing method, extended with the Volume-Fraction method and the use of second order finite difference schemes.

4.3.2 Grid Convergence Studies

For the validation multiple grid convergence studys were performed. The concept of a grid convergence study is to vary the grid resolution of the numerical domain and calculate the error of the simulation with respect to an assumed theoretical solution. The relative error ϵ is computed by the l_2-norm by

$$\epsilon = \frac{\int \mathrm{d}V \left(\Phi^{\text{th}} - \Phi^{\text{num}}\right)^2}{\int \mathrm{d}V \left(\Phi^{\text{th}}\right)^2} = \frac{\sum_{i,j,k}^{N_x,N_y,N_z} \left(\Phi^{\text{th}}_{i,j,k} - \Phi^{\text{num}}_{i,j,k}\right)^2}{\sum_{i,j,k}^{N_x,N_y,N_z} \left(\Phi^{\text{th}}_{i,j,k}\right)^2} \tag{4.11}$$

where Φ^{th} corresponds to the theoretical and Φ^{num} to the numerical solution of any field variable Φ. Often the numerical error is given by a power law of the form $\epsilon = N^{\lambda}$, i.e. for a finite difference scheme of second order this is the truncation error given by the remaining terms of the Taylor expansion. For an evaluation of the error convergence the results of the grid convergence study are, if possible, fitted linearly in the log-log space, to obtain the convergence rate λ. The following testcases are used for a numerical validation.

4.3.3 Laminar Poiseuille Flow

The numerical setup of this test case is presented in Fig. 4.4. It consist of two infinite extending planes at $z = h_1$ and $z = h_2$, which are oriented parallel to the xy-plane, with a distance $\Delta h = h_2 - h_1$. An infinite long channel is numerically realized by using periodic boundaries in x- and y-direction. For the walls immersed boundaries, or in case of the default algorithm No-Slip boundaries, are used. The fluid flow in the channel is a result from a predefined constant pressure gradient $\partial p/\partial x$ in the fluid domain. In the implementation this pressure gradient is added into the Navier-Stokes equation as an additional forcing term. For the steady state the flow is independent of the x and y coordinate. According to [KC02], the equations of motion, here in the non-dimensionalized form, are given by

$$\frac{\partial v_x}{\partial t} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\frac{\partial^2 v_x}{\partial z^2} = 0 \tag{4.12}$$

where $Re = V_m \Delta h/\nu$, with the non-dimensionalization defined as

$$\text{Length:} \quad \vec{r} = \frac{\vec{r}^*}{\Delta h} \quad , \quad \text{Velocity:} \quad \vec{v} = \frac{\vec{v}^*}{V_\mathrm{M}}, \tag{4.13}$$

$$\text{Time:} \quad t = \frac{t^* \cdot V_\mathrm{M}}{\Delta h} \quad , \quad \text{Pressure:} \quad p = \frac{\nabla p^*}{\rho V_\mathrm{M}^2} \tag{4.14}$$

where V_M is defined by the maximal velocity in the channel $\max(v_x(z))$. By the integration of Eq. 4.12

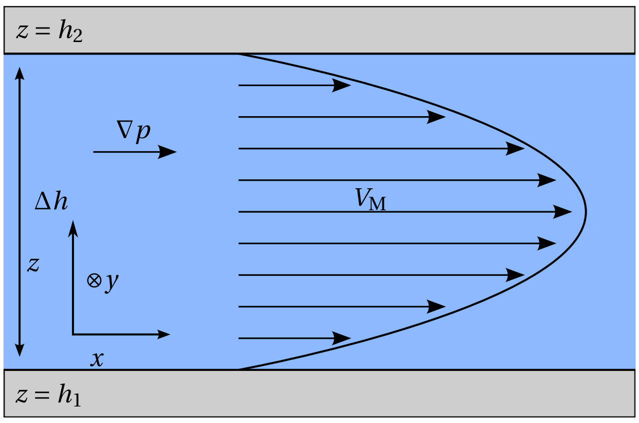


Figure 4.4.: Flow in a channel of height $\Delta h = h_2 - h_1$, with periodic boundaries in x and y direction and a predefined pressure gradient ∇p.

it follows that

$$v_x = \frac{1}{2}\frac{\partial p}{\partial x} Re z^2 + z c_1 + c_2. \tag{4.15}$$

The integration constants are obtained by setting $v_x(h_1) = v_x(h_2) = 0$:

$$c_1 = A\frac{h_1^2 - h_2^2}{h_2 - h_1} = -A(h_1 + h_2) \quad , \quad c_2 = A(h_1(h_1 + h_2) - h_1^2) = A h_1 h_2 \tag{4.16}$$

with the definition $A := \frac{1}{2}\frac{\partial p}{\partial x} Re$. The velocity is then given by the quadratic function

$$v_x = A(z^2 - z(h_1 + h_2) + h_1 h_2). \tag{4.17}$$

The maximum velocity and position are given by

$$z_{\mathrm{M}} = \frac{h_1 + h_2}{2} \wedge v_{\mathrm{M}} = A\left(h_1 h_2 - \frac{(h_1 + h_2)^2}{4}\right). \tag{4.18}$$

Due to the non-dimensionalization it is necessary that $V_{\mathrm{M}} \overset{!}{=} 1$. It follows that

$$\frac{\partial p}{\partial x} = \frac{2}{Re}\frac{1}{\left(h_1 h_2 - \frac{(h_1+h_2)^2}{4}\right)} \tag{4.19}$$

has to be fullfilled as a necessary condition for the pressure gradient. The Poiseuille flow is used in particular as a first validation test case for the VP, DF method and the basic implementation.

4.3.3.1 Simulation: Test of the Basic Algorithm

The purpose of the first simulation is to test the basic implementation of the algorithm which does not use Immersed Boundary methods. For the Poiseuille flow this is still possible since the geometry is non-curved and parallel to the cartesian grid. The heights for this setup are set to $h_1 = 0$ and $h_2 = 1$. A grid convergence test was performed with the main simulation parameters given by [4] [5]

N_z	Δt	Δz	Re	c^2	$l_z, l_x/l_y$	T_{end}
$[8, 256], \Delta N = 8$	10^{-4}	$1/N-1$	500	500	$1, 8\Delta z$	10

.

The simulation was performed for FD2 and FD4 methods.

[4] For reasons of clarity only the important simulation parameters, depending on the setup and the performed simulation, will be presented in this thesis.

[5] The size in l_x/l_y direction is set to the smallest possible value which is 8 grid points due to the GPU algorithm which uses a blocksize of 8x8.

4.3.3.2 Simulation: Test of the Volume-Penalization Method

For the Immersed Boundary methods the upper and lower boundaries of the channel are realized by the masking function

$$H(z) = \begin{cases} 0, & \text{for } h_1 \le z \le h_2 \\ 1, & \text{else.} \end{cases} \tag{4.20}$$

For the VP-method an additional forcing term for Eq. 4.12 of the form

$$\vec{f} = -\frac{H}{J}\vec{v} \tag{4.21}$$

has to be introduced (see Sec. 4.2.1). Where the non-dimensionalized damping coefficient is given by $J = {}^{V_m}/_{\eta}$. A first test is an investigation of the error of the velocity profile, with a variation of the Reynolds number and the damping coefficient J. To ensure that the channel width is equal to $\Delta h = 1$, the total height of the simulation domain was set to $l_z \approx 2.01587$, with $h_1 = 0.5$ and $h_2 = 1.5$. This ensures that the grid points overlap exactly with the masking function at h_1 and h_2. A simulation series was performed for the VP-FD2 method with the simulation parameters

Re	J	N_z	Δt	Δz	c^2	$l_z, l_x/l_y$	T_{end}
$[100, 500], \Delta Re = 25$	$[10^{-5}, 5\cdot10^{-1}]$	64	10^{-4}	${}^{1}/_{N-1}$	500	$\approx 2.015871, 8\Delta z$	10

.

The stepping of J was varied such that each order of magnitude is covered by two values, i.e. 10^{-1} and $5\cdot10^{-1}$. A value of $J = 10^{-5}$ corresponds to a relative strong damping force in comparison to a weak damping at a value of $J = 10^{-1}$. As a second test a grid convergence study was carried out with a constant Reynolds number, where the resolution was varied between $N_z \in [8, 280]$ with $\Delta N_z = 8$. VP-FD2 and VP-FD4 methods were tested with the parameters

N_z	J	Δt	Δz	Re	c^2	$l_z, l_x/l_y$	T_{end}
$[8, 280], \Delta N_z = 4$	$[10^{-4}, 5\cdot10^{-2}]$	10^{-4}	${}^{1}/_{N-1}$	500	500	$\approx 2.015871, 8\Delta z$	10

.

4.3.3.3 Simulation: Test of the Direct-Forcing Method

Since the DF method does not depend on a damping coefficient (see Sec. 4.2.2), it is sufficient enough to carry out a grid convergence study. The simulation parameters are given by

N_z	Δt	Δz	Re	c^2	$l_z, l_x/l_y$	T_{end}
$[8, 280], \Delta N_z = 4$	10^{-4}	${}^{1}/_{N-1}$	500	500	$\approx 2.015871, 8\Delta z$	10

.

The simulation was performed for FD2 and FD4 methods.

4.3.4 Hagen-Poiseuille Flow

In the previous test case the channel walls were aligned parallel to the simulation grid. Hence, no further interpolation procedures are necessary to mimic the exact boundaries. In order to test the accuracy of the VF and the IP method a test case with a curved geometry is necessary. Furthermore it is possible to investigate the discretization error of the non-interpolating VP and DF method, with respect to curved geometries.

A simple adaption of the planar Poiseuille flow, is the laminar flow through a pipe, also referred to as Hagen-Poiseuille flow [Tri88]. The setup of the fluid domain is schematically shown in Fig. 4.5. It consist of a circular pipe with the radius r_0, which extends infinitely parallel to the x axis. This is realized similar to Sec. 4.3.3 by using periodic boundaries in x-direction and Immersed Boundary methods for the channel wall. The center of the pipe is set to the center of the simulation domain $(y_0, z_0) = ({}^{l_y}/2, {}^{l_z}/2)$. In analogy to the Poiseuille flow, the velocity profile is a result from a predefined pressure gradient ${}^{\partial p}/{\partial x}$ in x-direction, which is added into the Navier-Stokes equation as an additional forcing term.

An analytical solution of this problem is based of [KC02]. Once again a steady state flow can be assumed, which implies that $\partial v_x / \partial t = 0$. With the introduction of cylindrical coordinates (r, ϕ, x) and the assumption that the flow is independent of ϕ the equation of motion reduces to

$$0 = -\frac{\partial p}{\partial x} + \frac{1}{Re}\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial v_x}{\partial r}\right) \tag{4.22}$$

where $Re = {}^{V_m r_0}/\nu$. The non-dimensional scales are similar to Sec. 4.3.3 except that the length scale is set to r_0 instead of Δh.

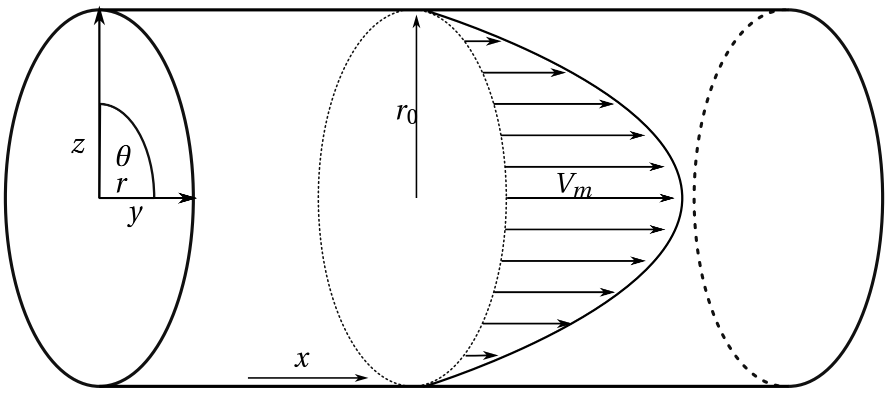


Figure 4.5.: Setup for the Hagen-Poiseuille flow, consisting of a pipe of radius r_0 with the center at $({}^{l_y}/2, {}^{l_z}/2)$ and of infinitesimal length.

The integration of Eq. 4.22 gives

$$v_x(r) = \frac{r^2}{4}\frac{\partial p}{\partial x}Re + A\ln r + B. \tag{4.23}$$

It holds that $v_x(0) = V_\mathrm{M}$ this condition can only be fulfilled if $A = 0$. With the use of the boundary conditions $v(r_0) = 0$ the velocity profile is given by

$$v_x(r) = \frac{r^2 - r_0^2}{4}\frac{\partial p}{\partial x}Re. \tag{4.24}$$

Since $V_M \overset{!}{=} 1$ by definition (see Sec. 4.3.3), the pressure condition for the domain needs to be set to

$$\frac{\partial p}{\partial x} = -\frac{4r_o^2}{Re}. \tag{4.25}$$

4.3.4.1 Simulation: Grid Convergence Study

For an error evaluation a grid convergence study was performed with the Reynolds number set to $Re = 100$. The number of grid points was varied in the interval $N \in [32, 256]$. Furthermore, a simulation with a resolution of $N = 512$ was carried out. Since the maximum velocity of the channel is $V_M = 1$, the sound speed was set to $c^2 = 100$ to fulfill the incompressibility condition $Ma = v/c < 0.1$. The resulting time step for the highest resolution is $\Delta t = 10^{-4}$. The main parameters of the simulation are given by

N	Δt	$\Delta z, \Delta y$	Re	c^2	l_y, l_z, l_x	r_0	T_{end}
$512; [16, 256], \Delta N = 16$	10^{-4}	$1/N$	100	100	$2.5, 2.5, 8\Delta y$	1	20

.

With this setup the VP, DF and IP methods were tested with FD2 and FD4 methods and optional with the VF method. The IP method was additionally tested with the DF method. Thus, the velocity is interpolated on the fluid boundary and set to zero outside of fluid domain. For the VP method the non-dimensionalized damping was set to $J = 10^{-4}$. It would be possible to choose a larger time step for the lower resolution cases, which means that in general one would apply a larger J in a case of an application, due to the stability criterion $J > \Delta t$. As a result the damping force on the boundaries $\propto 1/J$ would be smaller. To remain consistent in the error convergence, here Δt and therefore J are not altered.

4.3.4.2 Simulation: Long-Term

In order to test the numerical stability and conservation of mass, a long-term simulation was performed. A Reynolds number of $Re = 100$ was chosen. The resolution is set to $N = 96$ grid points and the ending time was set to $T_\mathrm{end} = 1600$. The main parameters of the simulation are given by

N	Δt	$\Delta z, \Delta y$	Re	c^2	l_y, l_z, l_x	r_0	T_{end}
96	10^{-4}	$1/N$	100	100	$2.5, 2.5, 8\Delta y$	1	1600

.

With this setup the same methods as in the grid convergence study were tested.

4.3.5 Taylor-Couette Flow

The numerical setup of this test case is shown in Fig. 4.6. It contains two coaxial cylinders which are oriented parallel to the z-axis. The inner cylinder has a radius and angular velocity r_i and Ω_i. In analogy these properties are defined as r_o and Ω_o for the outer cylinder. The fluid domain is given by the gap of width $d = r_o - r_i$ between the cylinders and extends to infinite. For the simulation periodic boundaries are used in z-direction.

In contrast to the previously examined test cases this system provides new characteristics of the flow regime at the fluid boundaries. The flow is not orthogonal to the curvature of the geometry and Furthermore the velocities at the boundaries are not zero but have to fulfill a Dirichlet condition given by $\vec{v}(\vec{r})|_{r_i} = \Omega_i \times \vec{r}$.

In dependency of the defined parameters different flow regimes exist. For small differences in the rotation rates the flow is laminar and azimuthal.

An increase of the angular velocity r_i above a critical value leads to an physical instability. The flow becomes unsteady. In this flow regime toroidal vortices form, which are also denoted as Taylor cells. The flow is not purely azimuthal [Tri88]. In this test case only the laminar and azimuthal flow regime will be considered, when the outer cylinder is not rotating ($\Omega_o = 0$). In literature, this system is usually referred to as Circular Couette flow (CCF) [KC02].

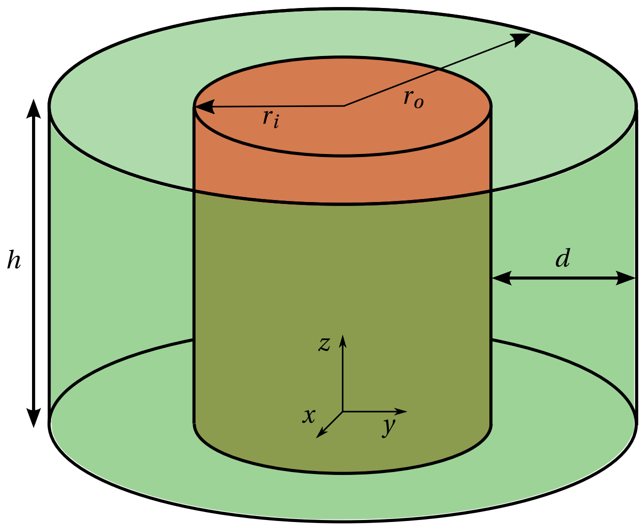


Figure 4.6.: Setup of the Taylor-Couette flow, consisting of two cylinders with radii r_i and r_o with angular velocities Ω_i and Ω_o, that are oriented coaxial and parallel to the z axis.

This problem can be reduced to two dimensions and be described in polar coordinates (r, ϕ). The equations of motion for the steady state are given by [KC02]

$$-\frac{v_\phi^2}{r} = -\frac{\partial p}{\partial r} \quad , \quad 0 = \frac{1}{Re}\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial}{\partial r}(r\, v_\phi)\right). \tag{4.26}$$

For the non-dimensionalization the default convention is chosen according to [CWZ15]

$$\text{Length:} \quad \vec{r} = \frac{\vec{r}^*}{r_o - r_i} \quad , \quad \text{Velocity:} \quad \vec{v} = \frac{\vec{v}^*}{r_i \Omega_i}, \tag{4.27}$$

$$\text{Time:} \quad t = t^* \cdot \frac{r_i \Omega_i}{r_o - r_i} \quad , \quad \text{Pressure:} \quad p = \frac{p^* - p_\infty}{\rho r_i^2 \Omega_i^2}. \tag{4.28}$$

The flow is then characterized by the Reynolds number $Re = {}^{\Omega_i R_i d}/_{\nu}$. With the integration of Eq. 4.26 the solution is given by [KC02]

$$v_\phi = Ar + \frac{B}{r} \tag{4.29}$$

where A and B are defined as

$$A := \frac{-\Omega_i r_i^2}{r_o^2 - r_i^2} \quad , \quad B := \frac{\Omega_i r_i^2 r_o^2}{r_o^2 - r_i^2}. \tag{4.30}$$

4.3.5.1 Simulations

The simulations for this test case were carried out in analogy to the Hagen-Poiseuille flow, see Sec. 4.3.4. Here the resolution is varied on the x and y axis, the radii are set to $r_i = 1$ and $r_o = 2$ and $l_x = l_y = 2.5$. The main parameters of the grid convergence study simulation are given by

N	Δt	$\Delta x, \Delta y$	Re	c^2	l_x, l_y, l_z	r_o, r_i	T_{end}
$512, [16, 256], \Delta N = 16$	10^{-4}	$1/N$	100	100	$2.5, 2.5, 8\Delta x$	2, 1	20

.

The main parameters of the long-term simulations are given by

N	Δt	$\Delta x, \Delta y$	Re	c^2	l_x, l_y, l_z	r_o, r_i	T_{end}
96	10^{-4}	$1/N$	100	100	$2.5, 2.5, 8\Delta x$	2, 1	1600

.

4.4 Results

4.4.1 Poiseuille Flow

4.4.1.1 Test of the Basic Algorithm

The results of the grid convergence study are shown in Fig. 4.7. The double logarithmic axis shows the dependency of the computed relative l_2-error norm with respect to the grid resolution.
On this scale, the FD4 has a linear decreasing error. For the smallest resolution $N = 8$ the error is of order 10^{-3}, in contrast to the highest resolution $N = 256$ with an error of order 10^{-6}. A convergence rate of the error was computed according to $\lambda \approx -2.26$. Hence, it can be said that the accuracy of the method is above secnd order for this test case, which is in contradiction with the theory assuming a 4th order. The error of the second order finite difference scheme is of the order 10^{-8}. The value is approximately constant and does not depend on the grid resolution.

4.4.1.2 Test of the Volume Penalization Method

The first simulation was a parameter study with variable Reynolds number and damping coefficient. A first impression of the influence of the damping coefficient J is given by the velocity profiles of the numerical solution. For varying J and $Re = 500$, this is exemplarily shown in Fig. 4.8. It can be noted that with an decrease of J, the numerical solution converges against the theoretical one.

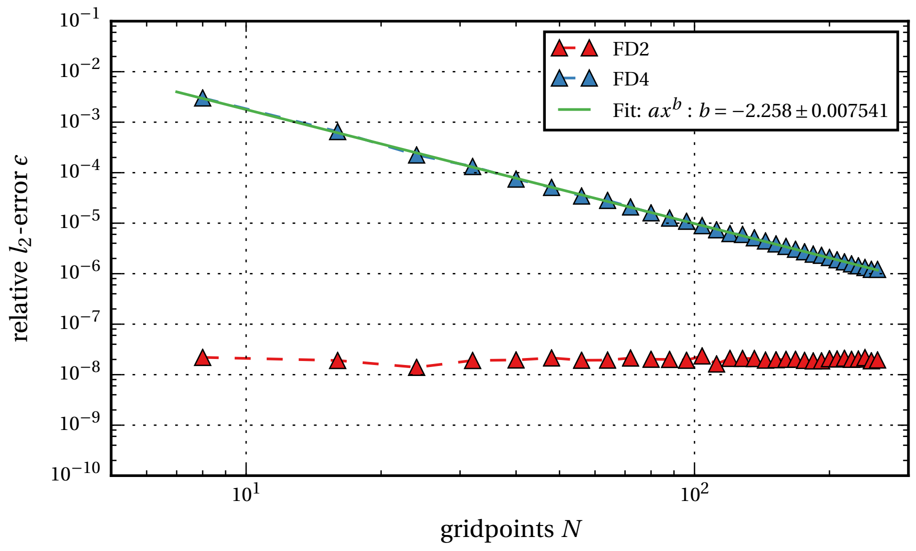


Figure 4.7.: Relative l_2-error for FD2 and FD4 methods with the basic algorithm without the use of an immersed boundary.

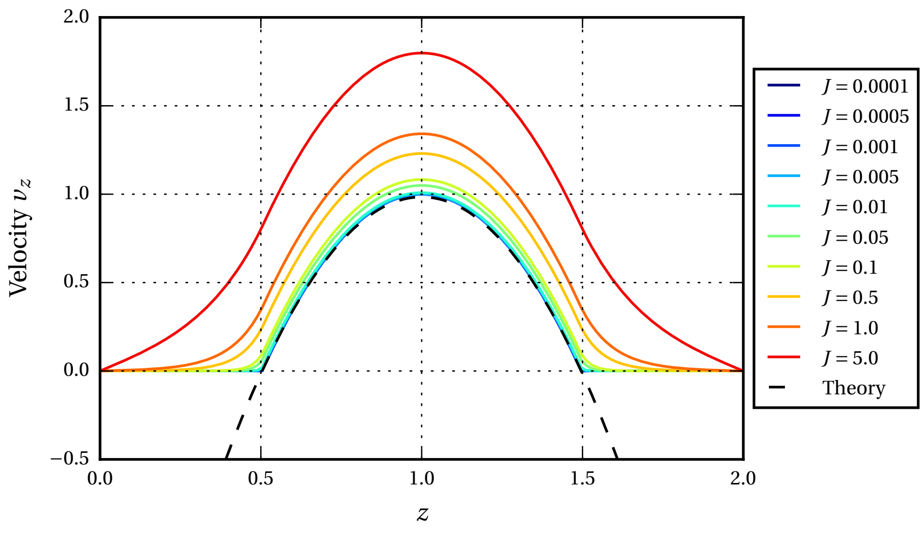


Figure 4.8.: Velocity profile of the numerical solution with variable J and $Re = 500$.

Furthermore, it can be seen that the quadratic part of the velocity profile, inside of the fluid domain ($0.5 \leq z \leq 1.5$), is independent of the damping constant J. Merely a slight offset at the boundaries creates a constant shift upwards of the velocity profile. In the masked area of the volume a decrease in velocity is visible which could eventually be described by an exponential law.
For an error estimation the relative l_2-error was computed, the results are shown in Fig. 4.9. On the left side of the plot the relative error is plotted against the damping coefficient and the Reynolds number is varied over a color scale. On the right side both variables (J and Re) are switched.
A decrease in the damping coefficient J results in a decrease of the error of about four orders, from 10^{-1} to 10^{-5}. With a change of the Reynolds number from 500 to 100 the error decreases about one order. It can be noticed that with a decrease of Re or J the error decreases linearly in the log-log space. For a small damping ($J > 10^{-3}$) a breakdown of the linear relation can be observed. This is an error resulting from the numerical setup. The damping in the masking area is so weak that the flow reaches the boundaries of the numerical domain. This is not the case for $J < 10^{-3}$ since the velocity profile is zero before it reaches the boundaries. A linear fit in the log-log space gives a decay rate of about 1.106 for J and about -1.223 for Re.
The second part of the test was a grid convergence study with a constant Reynolds number, variable J and grid resolution N. The results are shown in Fig. 4.10. It can be noticed that a decrease in J creates a shift in the error profile to overall smaller values. However, it is also noticable that the error of the FD2 method increases together with the resolution.
The error convergence for the FD4 method is even less understandable. With increasing resolution a decrease into a minimum which is followed by an increase can be observed. The position of the

minimum is shifted to the right with an decrease of J. For a constant $J > 10^{-4}$ it is visible that the error for both methods converges against a the same error with an increase in the resolution.

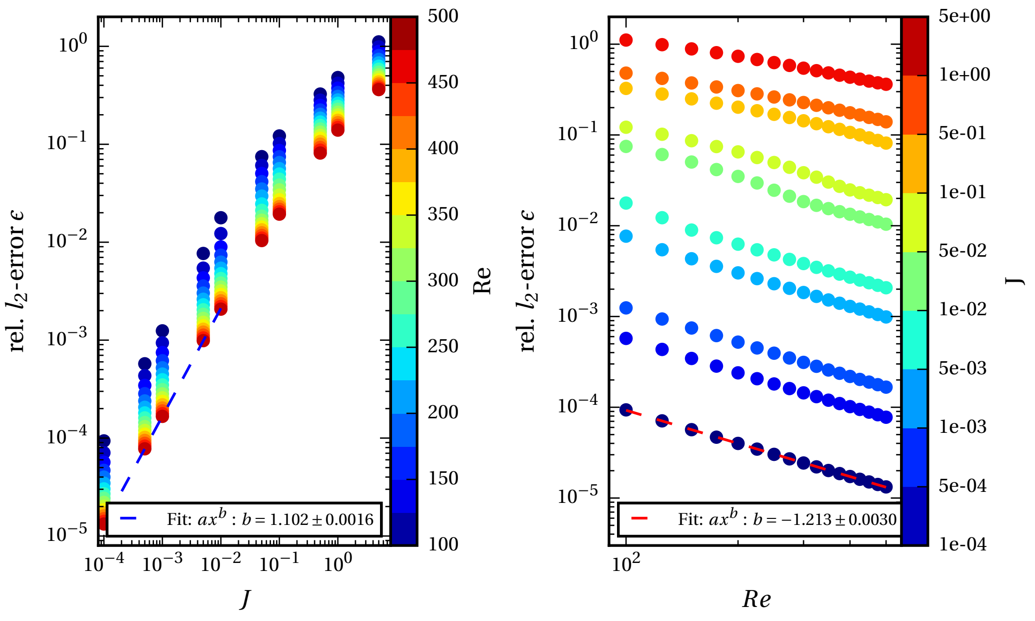


Figure 4.9.: Relative l_2-error for variable damping rate ν and Reynolds number Re.

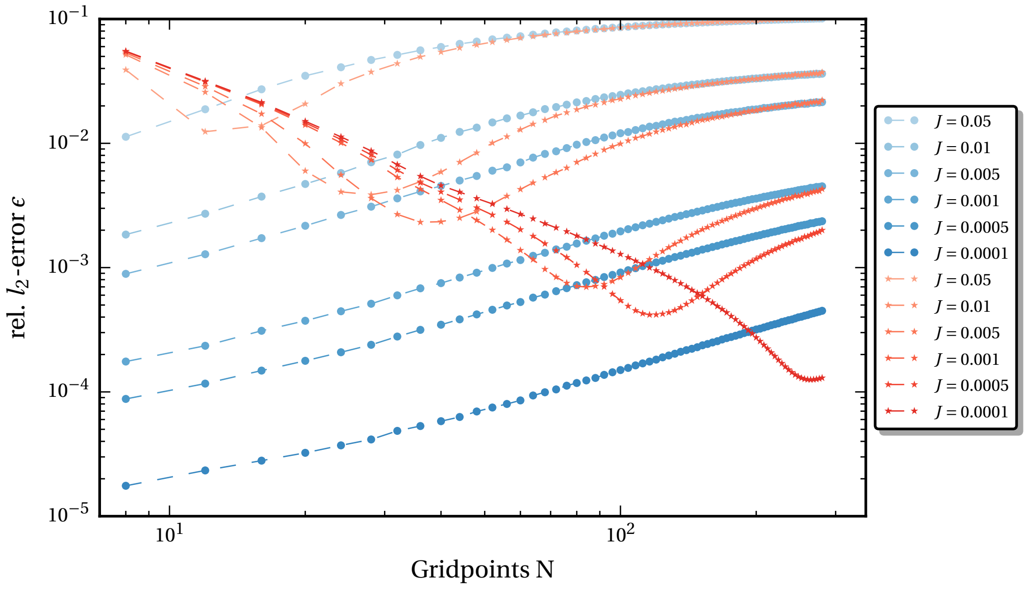


Figure 4.10.: Relative l_2 error for variable damping rate J, for VP-FD2 (blue) and VP-FD4 (red) methods.

4.4.1.3 Test of the Direct Forcing Method

The results of this grid convergence study for the DF method are similar to the results of the basic implementation, except the DF-FD4 method has a larger error. A plot of the relative l_2-error can be found in the Appendix in Fig.B.1. The FD4 method has a linear decreasing error in the log-log space. For the smallest resolution $N = 8$ the error is of order 10^{-2}, in contrast to the highest resolution $N = 256$, with an error of order 10^{-4}. The convergence rate is about $\lambda \approx 1.267$. Hence, it can be said that the accuracy of the method is above first order for this test case. The error of the second order finite difference scheme is of the order 10^{-8}. The value is approximately constant and not depend of the grid resolution.

4.4.2 Hagen-Poiseuille Flow

4.4.2.1 Grid Convergence Study

The results of the grid convergence study are shown from Fig. 4.11 to Fig. 4.14. For a better overview the results are distributed into four plots. For all methods (except IP+DF FD4) an approximately linear decrease in the double logarithmic space can be observed.
In Fig. 4.11 the relative l_2-error is shown for different VP methods. For the default VP-method with FD2 and FD4 the error convergence rate is about $\lambda = 1.17$. The FD4 has a slightly smaller error than the FD2 method. The optional use of the VF-method improves the error convergence is about $\lambda = 1.65$, here the FD2 method has a slightly smaller error. It can be noted that the overall error is smaller ($\approx 10^{-2}$ for $N = 100$) when using the VP-VF-method in comparison to the VP-method ($\approx 2 \cdot 10^{-2}$ for $N = 100$)
In Fig. 4.12 the relative l_2-error is shown for different DF methods. For the default DF-method with FD2 and FD4 the error convergence rate is about $\lambda = 1.2$. Again the FD4 method has a smaller error than the FD2 method. The optional use of the VF-method improves the error convergence to about $\lambda = 1.4$, which is worse in comparison to the VP-method. The FD2 method has a smaller error than the FD4 method when using VF. In comparison to the DF-method ($\approx 2 \cdot 10^{-2}$ for $N = 100$), the overall error is smaller ($\approx 1.5 \cdot 10^{-2}$ for $N = 100$) for the DF-VF-method.
In Fig. 4.13 the relative l_2-error is shown for different IP methods. For the IP FD4 method the decay rate is about $\lambda = 1.4$. For the IP FD2 and IP+DF FD2 method the errors are identical, the decay rate is about $\lambda = 2.4$ The IP+DF FD4 method is numerically not stable and therefore not shown. From the interpolation methods the IP FD2 method gives the smallest error ($\approx 5 \cdot 10^{-5}$ for $N = 100$).
Finally Fig. 4.14 shows the methods with the best convergence rates from the different DF, VP and IP methods in one plot. In summary, it can be said that the overall convergence rate of the IP FD2 method is of one order better than the VP-VF and DF-VF methods. The relative error of the interpolation method ranges between one and two order of magnitudes below all other methods.

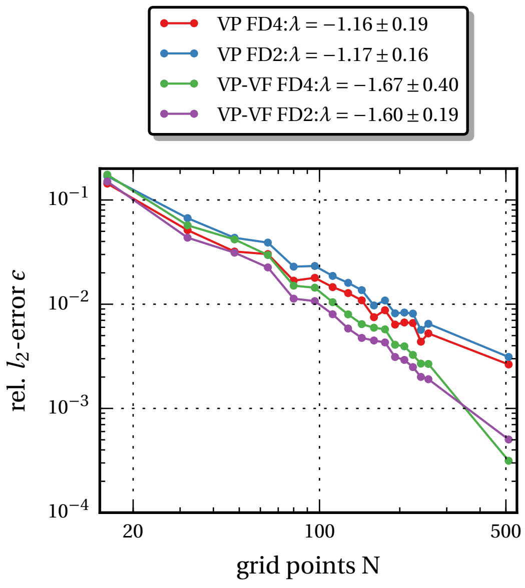


Figure 4.11.: Relative l_2-error for different Volume-Penalization methods.

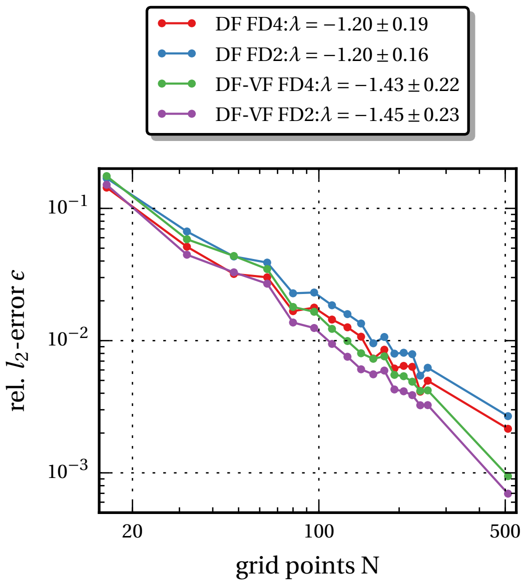


Figure 4.12.: Relative l_2-error for different Direct-Forcing methods.

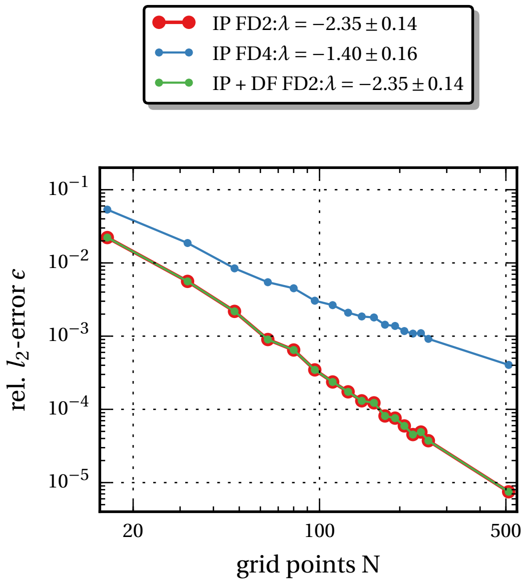


Figure 4.13.: Relative l_2-error for different Interpolation methods.

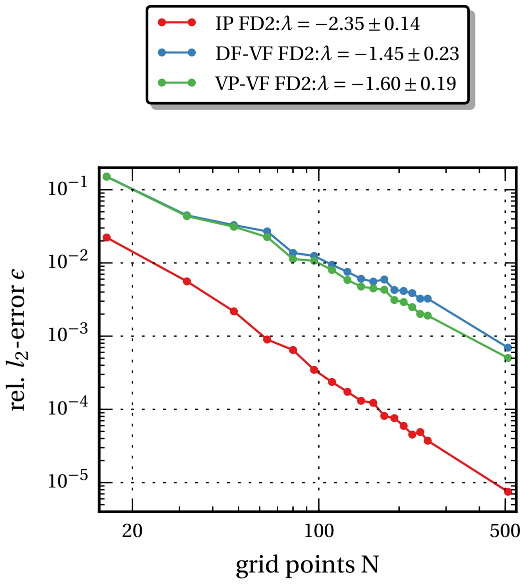


Figure 4.14.: Relative l_2-error for the methods with the smallest error in comparison.

4.4.2.2 Long-Term Simulations

The long-term simulations were performed in order to test the numerical stability and the conservation of mass. For all methods these simulations were numerically stable, except the IP FD4 method. The density was averaged over the fluid domain by

$$\langle \rho(t) \rangle = \frac{\int_V \mathrm{d}V \rho(x,y,z)}{\int_V \mathrm{d}V} = \frac{1}{N}\left(\sum_{i,j,k}^{N_x,N_y,N_z} \rho_{i,j,k} \right). \tag{4.31}$$

For all FD2 methods the averaged density is zero. This indicates that the total mass flux through the fluid domain boundaries is zero. It can be noticed that for all FD4 methods oscillations in the density emerge. For the VP-VF-FD4 method this is exemplarily shown in Fig. 4.15. The profiles of the other FD4 methods are shown in Appendix B.2.
Finally in Fig. 4.16 the averaged density with respect to the simulation time is shown for the FD4 methods. For the VP and DF methods the density increases to above $5 \cdot 10^{-5}$. The VF methods have a decrease in density, remaining within the order 10^{-4}. For all FD4 methods a change in the averaged density is followed by a saturation.

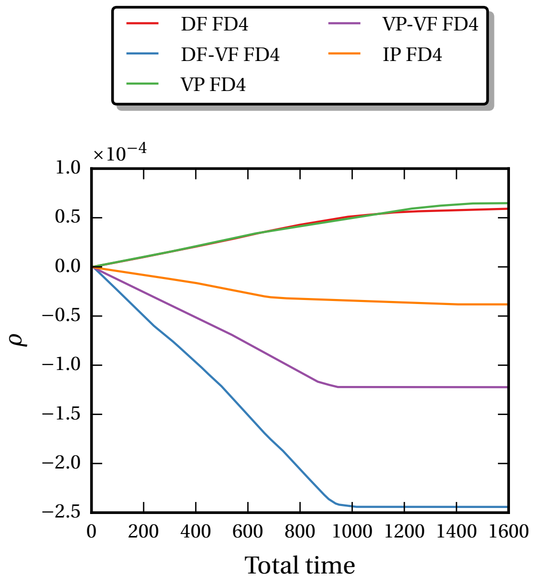


Figure 4.15.: Averaged density for FD4 methods with respect to the simulation time.

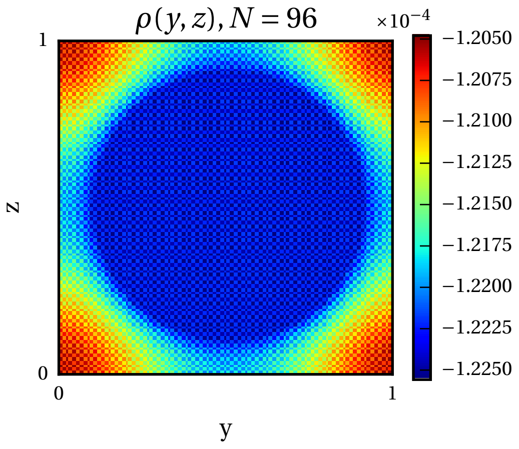


Figure 4.16.: Density in the (y, z) plane, at $T_{\text{end}} = 1600$, for the VP-VF-FD4 method.

4.4.3 Taylor-Couette Flow

4.4.3.1 Grid Convergence Study

The results of the grid convergence study are shown from Fig. 4.17 to Fig. 4.20. For a better overview the results are distributed into four plots.
In Fig. 4.17 the relative l_2-error is shown for different VP methods. For the VP FD2 method the error convergence rate is about $\lambda = 1.07$, for the VP FD4 method it is about $\lambda = 1.16$. However, it can be seen that the convergence rates decrease above $N \approx 100$. In this interval the error of the VP FD4 method remains approximately constant. The VP-VF methods have a larger error (about $5 \cdot 10^{-2}$ at $N = 100$) in comparison the VP methods (about $0.9 \cdot 10^{-3}$ and $2 \cdot 10^{-2}$ at $N = 100$). In contrast a higher convergence can be observed (about $\lambda = 1.3$) for the VP-VF methods.
In Fig. 4.18 the relative l_2-error is shown for different DF methods. For all methods the error convergence rate is about first order, except the DF FD4 method which is about $\lambda = 0.9$. The smallest error is given by the DF FD2 method which is about $2 \cdot 10^{-2}$ for $N = 100$. In comparison the error of the remaining methods is about $5 \cdot 10^{-2}$.
In Fig. 4.19 the relative l_2-error is shown for different IP methods. For the IP FD4 method the convergence rate is about $\lambda = 1.47$. For the IP FD2 and IP+DF FD2 method the errors are nearly identical, the convergence above first order. The IP+DF FD4 method is numerically not stable and not shown. From the interpolation methods the IP-FD4 method gives the smallest error for $N > 100$ which is of order $10 \cdot 10^{-3}$. Again it can be seen that the convergence rate decreases above $N \approx 100$ for all methods.
Finally Fig. 4.20 shows the methods with the best convergence rates from the different DF, VP and IP methods in one plot. It can be noted that in this validation the errors are not decreasing linearly in the log-log space. The best results are given by the interpolation methods with small differences in the overall error and convergence rates.

4.4.3.2 Differences in the Velocity Profiles

Beside the computation of the relative l_2-error, the local difference between the theoretical and numerical velocity profile was computed by subtracting the vector fields from each other in the (x, y) plane for a constant z[6]. The results for the FD2 schemes are presented in Fig. 4.21. The profiles for the FD4 schemes are shown in the Appendix in Fig. B.3. From the profiles it can be noted that for all methods the difference between the numerical and theoretical solution are the largest at the boundary r_i of the inner cylinder. In this region the error is of order 10^{-2} for all methods.

[6]Since $\partial_z \vec{v} = 0$ an arbitrary z value can be used, i.e. $z = 0$

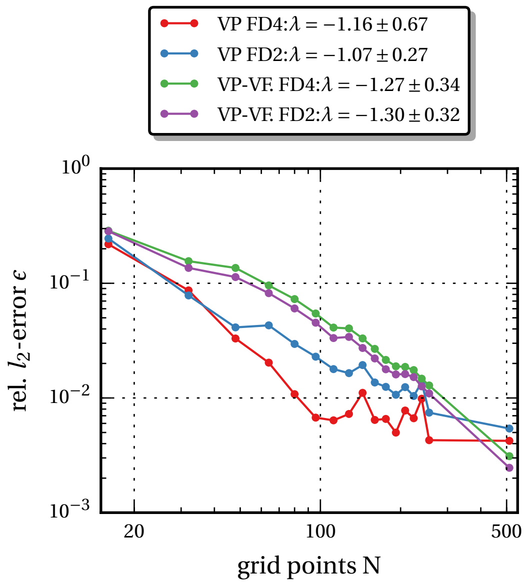


Figure 4.17.: Relative l_2-error for different Volume-Penalization methods.

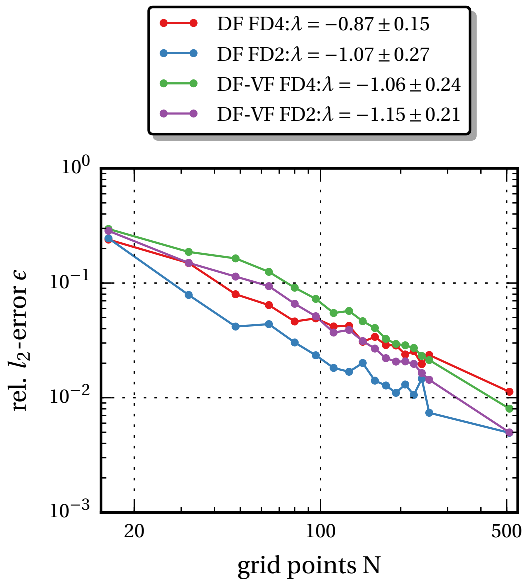


Figure 4.18.: Relative l_2-error for different Direct-Forcing methods.

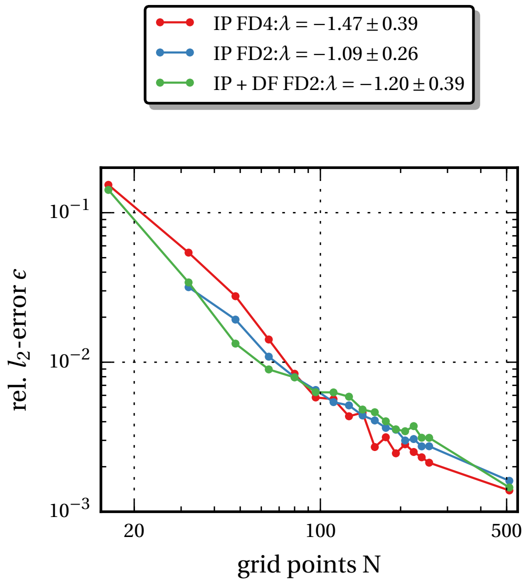


Figure 4.19.: Relative l_2-error for different Interpolation methods.

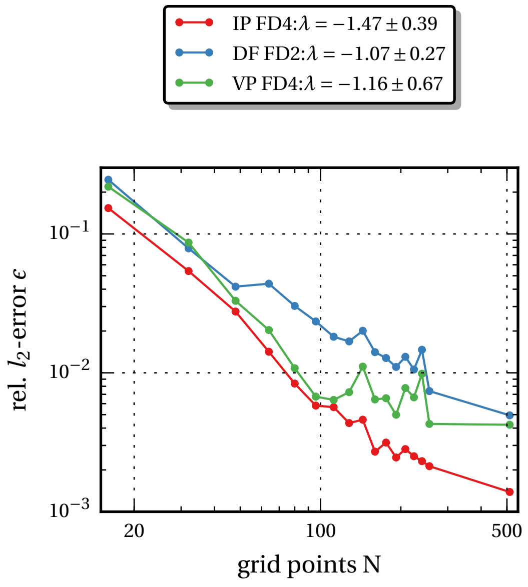


Figure 4.20.: Relative l_2-error for the methods with the smallest error in comparison.

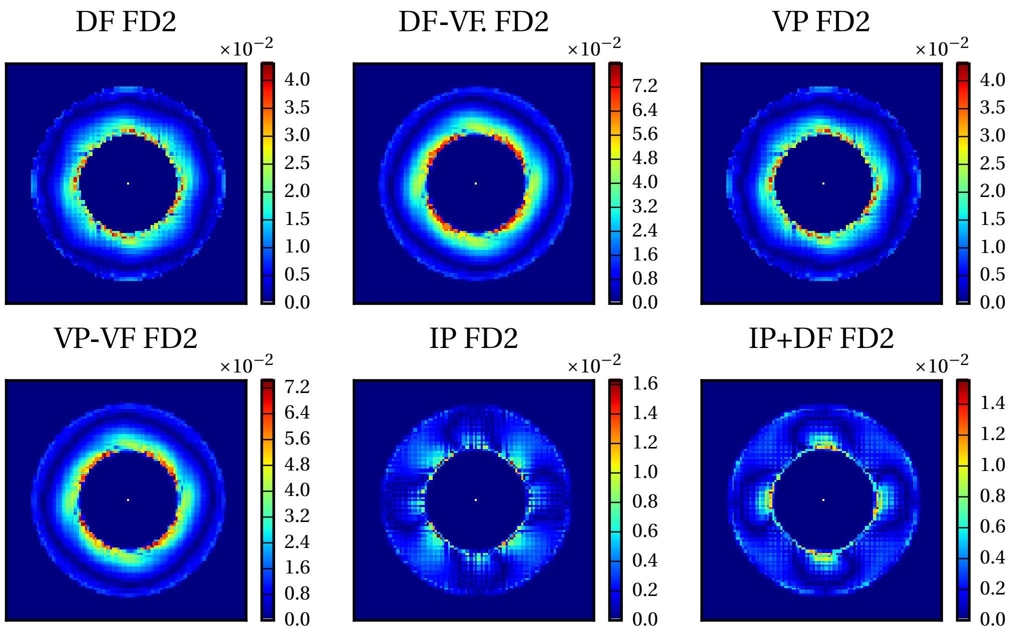


Figure 4.21.: Subtraction of the numerical velocity profile from the theoretical for all FD2 methods.

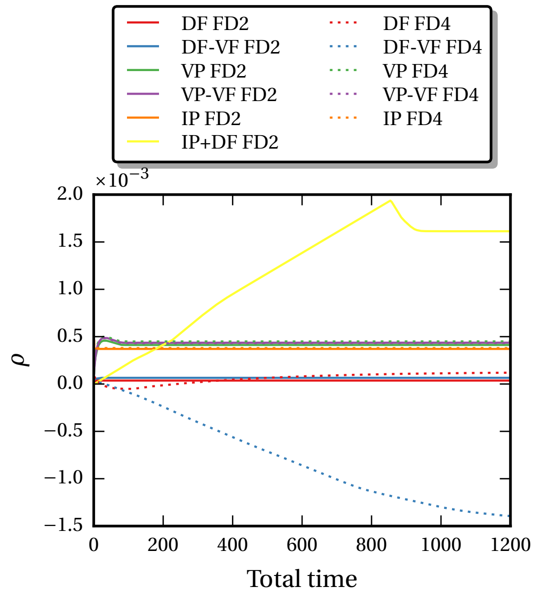


Figure 4.22.: Averaged density for all methods with respect to the simulation time.

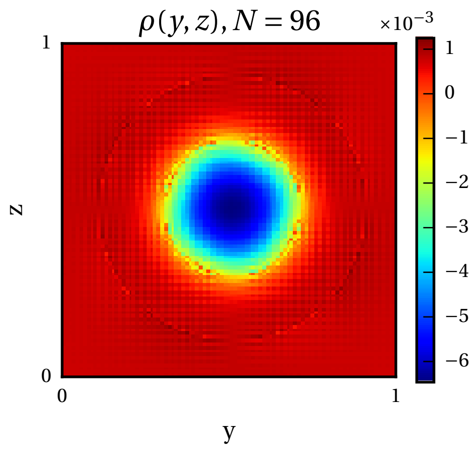


Figure 4.23.: Density oscillations for the VP-VF FD2 method with a decreased density in the inner cylinder.

4.4.3.3 Long-Term Simulations

For all methods the simulations were numerically stable, except the IP FD4 method. The density was averaged over the fluid domain by Eq. 4.31, the results of the computation are shown in Fig. 4.22. For the IP+DF FD2 method the density increases from zero until a convergence is reached at $T \approx 1000$ of about $1.6 \cdot 10^{-3}$. The density of the DF-VF FD4 decreases to $-1.4 \cdot 10^{-3}$ with a convergence at $T \approx 1400$. For all other methods a faster convergence is reached at $T \approx 200$ with an averaged density in the order 10^{-4}.
For all methods numerical oscillations in the density can be observed, which are similar to the Hagen-Poiseuille flow test case. In Fig. 4.23 these oscillations are exemplarily shown for the VP-VF FD2 method. In Appendix B.4 the remaining FD2 methods are presented. Besides these oscillations of order 10^{-4} it can be noted that the density in the inner cylinder where the VP method is applied decreases to about $-6 \cdot 10^{-3}$, in contrast to $\approx 10^{-3}$ in the fluid domain.

4.5 Discussion

4.5.1 Poiseuille Flow

4.5.1.1 Test of the Basic Algorithm

The constant error of the FD2 method can be explained by the lack of complexity of the test case. As the theoretical solution, given by Eq. 4.12 is a polynom of second order, no higher order terms will occur in the numerical solution. Hence, the FD2 method is capable of a perfect approximation, independent of the grid resolution. The remaining error terms, wich are of the order 10^{-8} occur due to the round-off error of the single precision floating-point format.

The convergence of the fourth order scheme, which is above second order reveals that an error exists in the basic implementation of the No-Slip boundaries. An explanation can be given with a comparison to the theoretical solution. The second derivative of Eq. 4.12 is given by

$$\frac{\partial^2 v_x}{\partial x^2} = \frac{1}{Re}\frac{\partial p}{\partial x} := C_f \in \mathbb{R} \tag{4.32}$$

where C_f is the constant curvature of the theoretical solution in the fluid domain. The basic algorithm uses a mirroring method at the boundaries of the fluid domain to fulfill the No-Slip condition (see Sec. 3.2.1). Let C_b be the curvature of the velocity profile in the boundaries, it follows that $C_b = -C_f$, as a result of the mirroring method. Therefore a discontinuity of the second derivative is created at the boundaries. When using FD2 method the 3-Point stencil evaluates to the correct value. The 5-Point stencil of the FD4 method evaluates to a false value, since it uses one point behind the boundary. As a result the discontinuity creates an error in this approximation.

4.5.1.2 Test of the Volume Penalization Method

The offset on the boundaries arises since the VP method cannot fulfill the exact boundary conditions. For the steady state a force equilibrium at the boundaries is given by the pressure gradient, the viscous force and the damping force

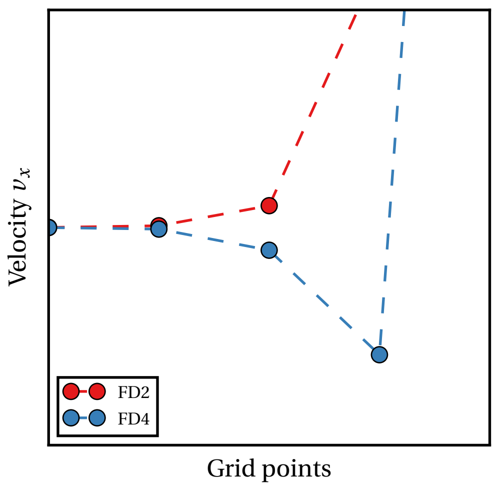
$$\frac{H}{J} v_x = \frac{1}{Re}\frac{\partial^2 v_x}{\partial z^2} - \frac{\partial p}{\partial x}. \tag{4.33}$$

The result is a constant offset $v_x(h_1) = v_x(h_2) =: C \in \mathbb{R}$. The simultaneous decrease of the rel. l_2-error with J is as expected, since a stronger damping force $\propto {}^{1}/_{J}$ creates a smaller error at the boundaries. However, it is counterintuitive that the rel. l_2-error decreases with an increase of the Reynolds number. An explanation is that the viscous force and the pressure gradient are proportional to $1/Re$ (see Eq. 4.19). With an increase of the Reynolds number the left side of Eq. 4.33 becomes smaller, the damping force becomes larger in comparison and the offset $C = v_x(h_1) = v_x(h_2)$ decreases.

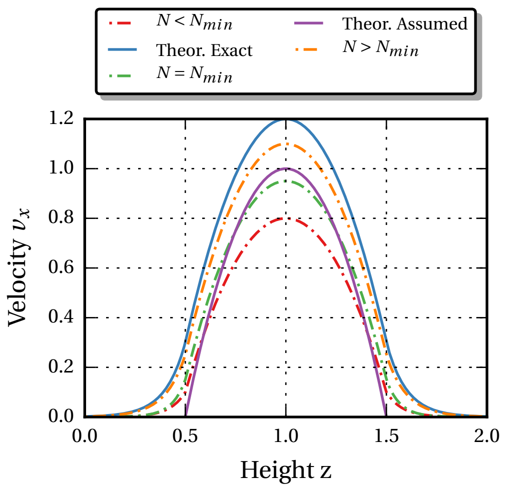
The grid convergence study with a constant Reynolds number and variable damping coefficient, generates an error which increases simultaneously with the grid resolution. An explanation can by given by revising the theoretical solution and the finite difference stencils at the fluid boundaries.

The error for both methods does not converge towards zero. This means there has to be some discrepancy to the theoretical solution given by Eq. 4.17. For a constant J there is an offset C to the theoretical solution, which is given by the equilibrium accord to Eq. 4.33. This further means that the theoretical solution which was assumed in the first place, is wrong for the VP method. An additional offset in the flow which is dependent on the damping coefficient J has to be considered. For a low resolution, the profile of the FD2 method is closer to the assumed theoretical one, which results in a smaller error. With an increase to a higher resolution the error with respect to the real solution decreases.

An explanation for the convergence of the FD4 method can be given by further considering the discretization error at the fluid boundaries. Fig. 4.24a shows exemplarily the velocity profile near to the fluid boundary, for the Volume-Penalization method of FD2 and FD4 order at a resolution of $N = 100$. It can be seen that the velocity profile for the FD4 method is slightly negative. The reason for this is the use of a 5-Point stencil for the discretization. The stencil reaches over the immersed boundary which results in a wrong computation of the Laplacian



(a) Velocity profile at the immersed boundary for the VP-FD2 and the VP-FD4 method



(b) Schematic velocity profiles for different resolutions.

Figure 4.24.

The error convergence is a result from the velocity profiles for different resolutions, as shown schematically in Fig. 4.24b. [7] The velocity profiles for three different resolutions are considered for a constant J, the minimum of the error shall be defined as N_{min}. Due to the negative velocity at the boundary the first profile $N < N_{min}$ lies below the theoretical solution, given by Eq. 4.17. The second profile is at $N = N_{min}$. The error at the boundary is smaller due to the increase of the resolution. The velocity profile overlaps best with the theoretical solution resulting in a minimal error. With a further increase of the resolution $N > N_{min}$ the velocity profile drifts away from the theoretical solution. The error increases. Finally, the method converges towards the same solution as the second order scheme.

4.5.1.3 Test of the Direct Forcing Method

The constant error of the FD2 can again be explained by the lack of complexity of the test case. The FD2 method is capable of a perfect approximation and the remaining error terms of the order 10^{-8} occur due to the round-off error of the single precision floating-point format.
In comparison to the basic algorithm which was above second order, the convergence of the FD4 method is above first order. Once more this error is a result of the used 5-Point stencil. All values behind the fluid boundary are set to zero. The 5-Point stencil uses one grid point behind the boundary which results in a incorrect computation.

4.5.2 Hagen-Poiseuille Flow

The simulation of the Hagen-Poisseuille flow points out multiple numerical problems of the implementations. The first which shall be considered is the error in the IP method. In comparison to the IP-FD2 method, the IP-FD4 method contains an error which is about one to two orders larger. Fig.4.25 shows the subtracted velocity profiles for the IP-FD4 and FD2 method in the (y, z)-plane. It can be noted that the velocity differences inside of the fluid domain are small but large in the wall region. Especially close to the fluid boundary the difference is of order 10^{-2}. One explanation for this result is the use of a different stencil for the FD4 scheme. The 5-Point stencil reaches over the domain boundaries an incorporates errors by a false approximation, similar to the Poiseuille flow. This is not a problem for the FD2 method due to the use of a 3-Point stencil. The motivation of the combination of IP and DF methods was to set the velocities in the wall region to zero. However, the computation is numerically unstable. One assumption is that this instability is created by the density which is interpolated over the domain boundaries. The only case in which the wall and the fluid domain are decoupled are the FD2 methods. Therefore no difference between the IP-FD2 and the IP+DF-FD2 method can be observed. An additional attempt was made to interpolate the density at the border according to [GSB03]. The results are not presented in this thesis, since it turns out that the computation is numerically unstable.

[7] The difference in the computed velocity profiles is barely visible. Therefore, an exaggerated version is used.

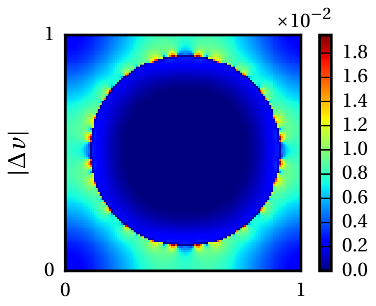


Figure 4.25.: Absolute value of the velocity difference between the IP-FD2 and IP-FD4 in the (y, z)-plane.

The second problem concerning this numerical implementation, are density oscillations which occur for the FD4 methods. One possible explanation for this behaviour might be the odd-even decoupling of the density. The numercial change in the density is proportional to [8]

$$\frac{\rho_i^{j+1}-\rho_i^j}{\Delta t} \propto \frac{v_{i+1}^j - v_{i-1}^j}{\Delta x} + \text{o.t.} \propto \frac{\rho_{i+2}^{j-1} - 2\rho_i^{j-1} + \rho_{i-2}^{j-1}}{\Delta x^2} + \text{o. t.} \tag{4.34}$$

where j is the time step and i a grid point. Here, the one-dimensional case is considered with an FD2 discretization and an explicit Euler step in time. This means that the density field is implicitly coupled over the velocity to itself, but neighbor points are independent of each other.[9] For the three dimensional setup six decoupled pressure fields would be possible. From the artificial continuity equation (see Sec. 2.5) it is known that $\partial_t \rho = \nabla \vec{v} = 0$, which is true for the FD2 case. The oscillation could be a result of the wrong computation due to the use of a 5-Point stencil at the boundaries. Due to the decoupling the computed error evolves differently on the decoupled pressure fields.

Finally a comparison of the error shall be given. In [Fad+00] the DF method was tested with and without the VF method. Furthermore a linear interpolation method, similar to the IP method was used. As a test case the formation of a vortex ring behind a curvilinear noozle was tested. The results show that for the DF method the error converges below first order, but can be improved with the VF method to about first order convergence. The interpolation method is about second order accurate. For all methods the error is of order 10^{-2} at $N = 100$. This result for the IP-method is in argreement with this thesis. However, the DF-method has a better convergence rate of first order and the additional use of the VF method has a smaller influence.

A validation of the IP method can be found in [GSB03; GS05]. In [GSB03] the IP method is tested by the steady flow in a lid-driven cavity containing a sphere, the error convergence of order 1.74. In [GS05] a flow caused by an oscillating sphere in a cavitiy is studied, with an error convergence of order 1.77. In this thesis the IP-FD2 method has a convergence rate about 2.35, above second order,

[8] o.t. = other terms

[9] This example also holds for the FD4 method.

which is slightly better in comparison to these validations. One reason for this could be the lack of complexity in the Hagen-Poiseuille flow as the theoretical solution is a polynom of second order.

4.5.3 Taylor-Couette Flow

In this validation test case the density changes for the FD2 methods as well. The convergence of the averaged density to the order of 10^{-4} is similar to the Hagen-Poiseuille flow with exception of the IP+DF FD2 and DF-VF FD4 method ($\approx 10^{-3}$). The change in density is related to the numerical setup which is more complex in this test case. Due to the boundary conditions the divergence of the velocity field is not zero at the start of a simulation. A change in the density field emerges and grows until a steady state is reached. This is a result of the method of artificial compressibility, see Sec. 2.5. Even with the use of a large speed of sound (here $Ma = {}^{v}/_{c} \approx 0.05$) the fluid is still not completely incompressible. The decrease of the density in the inner cylinder for the VP method can be assumed to be physically correct. The angular velocity Ω_i creates a centrifugal force in the inner cylinder. This results in a density distribution which is large at the outer cylinder walls and small in the center. The negative value is not a concern since the density is undetermined by a constant. Moreover, the overall change in density is small. The spatial oscillations in this test case also occur in the FD2 methods since the density is no longer zero. It can be assumed that just like in the Hagen-Poiseuille flow an odd-even decoupling occurs.

It can be noted that the results of the IP FD2 and the IP+DF FD2 method are not equal. This refutes the previous assumption (see Sec. 4.5.2), that the velocity fields of the fluid domain and the boundaries are decoupled for FD2 methods.

For the DF and VP methods the error and convergence rates are similar, but the error of the IP methods is about one order higher in comparison to the Hagen-Poiseuille flow. The use of the VF method leads to an increase of the overall error, which is contrary to results of the Hagen-Poiseuille flow.

The velocity difference profiles for all methods (Fig. 4.21 and Fig. B.3) indicate an erroneous approximation of the inner boundary. The error is larger in comparison to the outer boundary where $\vec{v} = 0$. Hence, it can be assumed that the approximation of Dirichlet boundaries, with velocities differing from zero at the boundaries, creates an overall larger numerical error.

4.6 Summary

In the first part of the chapter an introduction to different Immersed Boundary methods was given. The volume-penalization method, adapted from [Lül11] uses a forcing term which is proportional to the velocities at the fluid boundaries. In case of the Direct-Forcing method, adapted from [Fad+00] the forcing term is calculated implicitly. Both methods were extended by the Volume-Fraction method, adapted from [Fad+00], which computes the fluid domain volume ratio inside a boundary cell and uses this value as a weighting coefficient for the forcing. Finally, an interpolation method adapted from [GSB03] was introduced. In the first step this method performs a bilinear interpolation on a surface next to the fluid boundary, in the second step this value is linearly interpolated to a grid point close to the boundary.
In the second part of the chapter three different test cases were introduced as a validation for the Immersed Boundary methods. The laminar Poiseuille flow was used as a first validation test case for the DF and VP methods. From the results it was possible to identify an error in the basic algorithm of the simulation. Furthermore it could be shown that the use of a 5-Point stencil leads to discretization errors at the fluid boundaries.
In the second test case a simulation of a Hagen-Poiseuille flow was performed. The results of this simulation pointed out possible numerical issues. One possible problem are numerical oscillations which are assumed to be caused by a decoupling in the density field. Furthermore this test case showed that the use of a 5-Point stencil creates a numerical error, which results in a large error for the IP-DF4 method. It can be noted that using the Volume-Fraction method the error of the DF and VP method can be improved. In summary all errors are approximately below the order 10^{-2} for $N > 100$, The smallest error was obtained with the IP-DF2 method. All convergence rates are at least as good as those from validation test cases from literature [GSB03; GS05; Fad+00].
The last test case was given by a Taylor-Couette system. In comparison to the Hagen-Poiseuille flow the computed numerical error is larger, up to one order for the VF methods, which perform inferior in this test case. In this simulation for both the FD2 and FD4 methods small pressure oscillations can be observed which are again assumed to be caused by a decoupling of the density field. The velocity profiles show that the error is the largest to the inner domain boundary r_i. This indicates that the Immersed Boundary methods in combination with implemented basic GPU algorithm are not a good choice for Dirichlet boundaries with velocities differing from zero.

5. Inertial Waves in a Rotating Cone

5.1 Introduction

Following the introduction and validation of the IB methods, now a rotating fluid system is exemplarily investigated by using these methods. The objective is to reproduce physical properties with respect to theoretical and experimental results. In relation to the research focus of the geophysical fluid mechanics research group, there are a variety of topics of interest.
One area of research lies in the exploration of dynamo effects in geological and stellar systems. In particular this means the generation of magnetic fields by electrically conducting fluids on large scales. In this thesis we will not consider MHD-equations. [1] However, in general it is considered that the helicity of a fluid domain V, given by

$$\mathscr{H} = \int_V \mathrm{d}V\, \vec{v} \cdot (\nabla \times \vec{v}) \tag{5.1}$$

is directly linked to dynamo action [Mof78]. It would be interesting to find a system which exhibits a large helicity as a possible candidate for future researches on dynamos. Furthermore another interest is the propagation of inertial waves in different fluid domains.
The studied system in this chapter is given by a rotating cone, which is compared to a frustum, by inserting a bottom plate into the apex. Inertial waves are excited by a temporal modulation of the rotation frequency, which is denoted as libration. The objective is to examine the ability to generate inertial modes or wave attractors. Furthermore the properties of the helicity of the system are investigated and the decay of inertial oscillations is studied.
In this chapter linearized equations are used. Due to the axial symmetry with the rotating axis, this problem could be reduced to two dimensions. However, the code in tested a three-dimensional setup. One reason for this is a possible future application. The objective would be to study the decay of turbulence in the apex of the cone. A simulation in three dimensions can not be avoided in this case.

[1]MHD - Magnetohydrodynamic

5.2 Conventions & Setup

This section introduces the setup which has been used for the simulations presented in this chapter. Fig. 5.1 shows the geometry of the fluid domain. In the remaining sections of this chapter we want to use the following conventions:

H Total height of the simulation domain

h_c Height of the cone/frustum

h_b Height of the bottom plate (not shown)

R Radius at the top of the cone/frustum

r Radius at the bottom of the frustum, for the cone $r = 0$

O Offset at the top of the cone/frustum

α Slope of the cone/frustum

θ Propagation angle of an inertial wave with respect to the horizontal axis.

Ω Rotation rate

ω Frequency of the libration

ϵ_0 Amplitude of the libration, default is 1

l_x, l_y, l_z Size of the total simulation domain in x, y and z direction

Cone The entire cone with apex, and a possible offset at the top, this setup is obtained by setting $r = 0$

Frustum The cone but with a bottom plate at the bottom with $r > 0$

Critical Frequency ω_c This is the libration frequency where the propagation direction of an inertial wave is parallel to the slope ($\theta = \alpha$)

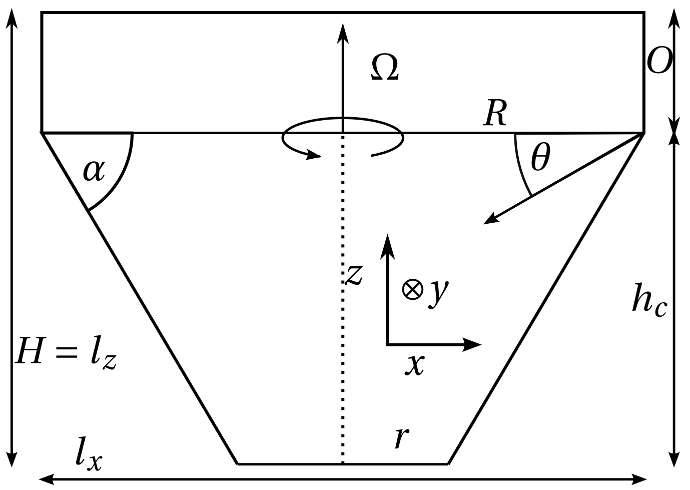


Figure 5.1.: Numerical Setup for the Simulations

5.3 Theoretical Description

This section gives a brief theoretical description of the problem which is adapted from [Gre69; Bea70]. Instead of a cone, a comparison to the two-dimensional analogy of a wedge, is made. This system is based on the idea to introduce a geometric shape containing a singularity.
Considering the propagation of an inertial wave ray, emitted at one of the upper edges of the wedge, as shown in Fig. 5.2. The wave propagates under the angle θ with respect to the horizontal axis. For each reflection the propagation angle with respect to the rotation axis is preserved. In the case of $\theta < \alpha$ a supercritical reflection occurs. Thus, a downward traveling wave ray is reflected downslope. As a result the wave travels towards the lower vertex of the wedge. In respect to the reflection properties, given by Eq. 1.22, the time for a wave ray to travel the path between two reflections is constant [Bea70]. To be more precise,

$$\frac{L}{|\vec{c}_g|} = \frac{L'}{|\vec{c}_g'|} \tag{5.2}$$

where L is the path length between to reflections and the prime denotes the properties after a reflection. This means that the overall propagation time into the apex of the cone becomes infinity, along with the energy density and the wave number. Thus, the apex is acting as an attractor to inertial waves. Since no reflection out of the cone apex can occur it follows that inertial modes can not exist.

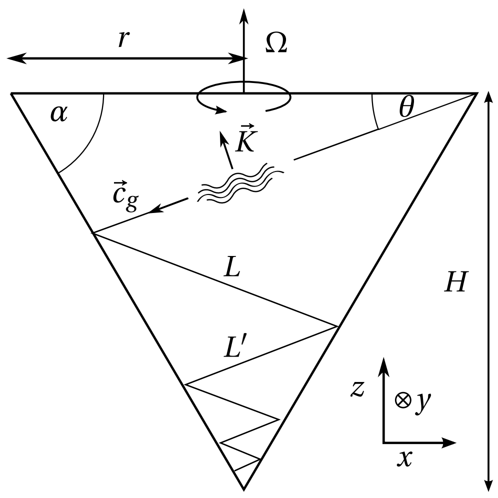


Figure 5.2.: Propagation of an inertial wave emitted from the top edge of a wedge, where θ indicates the direction parallel to the group velocity $\vec{c}_g$. L and L' are the path lengths before and after a reflection.

5.4 Experiment

To test these theoretical assumptions, an experimental study was performed by Beardsley [Bea70]. The schematic setup is shown in Fig. 5.3. In the first part of the experiment a plexiglass cylinder containing a conical shaped cavity, with the height $H = 19.95\,\mathrm{cm}$ and the radius $r = 8.91\,\mathrm{cm}$, was used and filled with water. The apex half angle was set to $\beta = 24°3.7'$ degree. For the rotation rate a frequency of $\Omega_0 = 6.28\,\mathrm{rad\,s^{-1}}$ was chosen. The resulting Ekman number is

$$Ek = \frac{\nu}{\omega r H^2} \approx 7.72 \cdot 10^{-6} \tag{5.3}$$

where $\nu = 1.0\,\mathrm{mPa\,s}$ [TM03]. In order to exite inertial waves the cone is librating. The total rotation rate is proportional to

$$\Omega(t) = \Omega_0 + \epsilon_0 \omega \cos(\omega t) \tag{5.4}$$

where ω is the libration frequency with the amplitude $\epsilon_0 = 0.2\,\mathrm{rad}$. The excitation of the fluid results from the wall friction, induced by the modulation of the constant rotation frequency Ω_0.
For the analysis the dynamic pressure field was measured at different depths along the rotation axis. The ratio of libration to the constant rotation frequency was varied in the range of $0.25 \le \omega/2\Omega_0 \le 1$. The pressure and phase lag spectrum show that no consistent peaks are observable for different measurement heights. Hence, no inertial modes are existing.
In the second part of the experiment, the apex of the cone was replaced by a frustum, with a bottom plate at the position $h_b = z/H = 0.261$. This setup yields the possibility that a wave ray can be reflected on the bottom. The pressure and phase lag spectrum in this setup show that independent

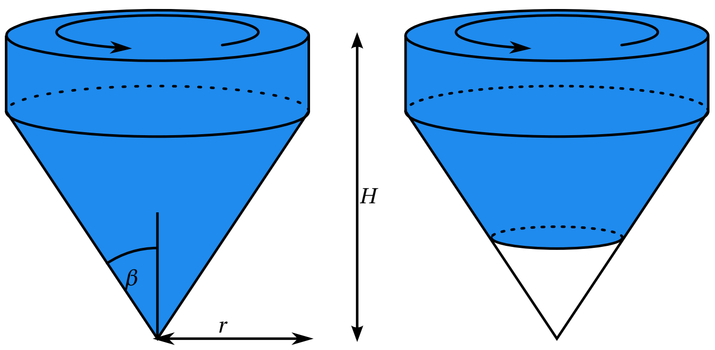


Figure 5.3.: Experimental Setup. In the first experiment a cone is used (left). For the second part the apex of the cone is replaced by a frustum (right).

of measurement depth of the pressure, resonances occur which can be associated with standing waves. Hence, inertial modes exist.

One aim of this thesis is to compare the simulation results to this experiment. Some considerations regarding the experiment are briefly discussed here. For the setup the relation between libration frequency ω and the propagation angle θ is given by Eq. 1.20, which can be simplified to

$$\omega = 2\cos(\theta). \tag{5.5}$$

This means for $\omega = 2\cos(\alpha)$ the propagation angle of an inertial wave is parallel to the slope of the cone. Depending on the libration frequency two different motions can occur in the cone.

$\omega < \omega_c$ It holds that $\theta > \alpha$. A downward traveling wave is reflected upwards on the slope of the cone.

$\omega > \omega_c$ It holds that $\theta < \alpha$. A downward traveling wave is reflected downwards on the slope of the cone.

For the first case the apex of the cone will not act as an attractor, This first scenario is not addressed directly in the experiment. In this chapter it will be investigated in more detail.

5.5 Numerical Implementation of Libration

For the numerical implementation of the experiment a modified set of the equations introduced in Sec. 1.2.1, is presented here. It has to be concerned that the system has now a time-dependent rotation rate. For the non-dimensional system we set

$$\vec{\Omega}(t) = 1 + \epsilon_0 \sin(\omega t)\vec{e}_z. \tag{5.6}$$

There are two options for an implementation. First of all, a rotating coordinate system with a constant velocity Ω_0 can be chosen. This enables the direct use of the Eqs. 1.17. Since the overall rotation rate of the system is modulated, it is necessary to introduce the boundary conditions

$$\vec{v}|_{Border} = \Omega \times \vec{r} = \begin{pmatrix} -y(1+\epsilon_0 \sin(\omega t)) \\ x(1+\epsilon_0 \sin(\omega t)) \\ 0 \end{pmatrix}. \tag{5.7}$$

However, obtaining a linearized version of the boundary methods is not as easy. In addition it can be assumed that it is safer to use No-Slip boundaries with zero velocity, as the Taylor-Couette system showed that velocities other than zero can create a larger numerical error. The alternative option is the introduction of an accelerated frame of reference. In this case the boundary conditions do not need to be modified, but the coriolis forcing term $\vec{f}$ is given by [Til07].

$$\vec{f} = 2\vec{\Omega} \times \vec{v} + \frac{\partial}{\partial t}\left(\vec{\Omega} \times \vec{v}\right) = \begin{pmatrix} -y\omega\epsilon_0 \cos(\omega t) \\ -x\omega\epsilon_0 \cos(\omega t) \\ 0 \end{pmatrix} + 2 \begin{pmatrix} -(1+\epsilon_0 \sin(\omega t)\, v_x \\ (1+\epsilon_0 \sin(\omega t)\, v_y \\ 0 \end{pmatrix} = \begin{pmatrix} -y\epsilon_0 \cos(\omega t) \\ -x\epsilon_0 \cos(\omega t) \\ 0 \end{pmatrix} \tag{5.8}$$

In the last step the linearization was introduced. The non-linear advection term is removed from the equations of motion, which are then given by

$$\frac{\partial v}{\partial t} = -c^2 \nabla \rho + Ek \Delta \vec{v} + \vec{f} \qquad , \qquad \nabla \vec{v} = 0 \tag{5.9}$$

Finally, one additional stability criterion shall be introduced. It is important that the time for propagating information from on side of the domain to the other side is much smaller than the overall rotation rate. [2] This can be estimated by

$$t = \sqrt{\frac{2}{c^2}} \ll 2\pi \tag{5.10}$$

where c is the artificial speed of sound. Thus, the propagation time and therefore this criterion, depends on the artificial sound speed c. A good estimation is given by setting $c^2 = 500$. It could be observed that otherwise not only physical wrong results are obtained but also numerical instabilities can occur. [3]

[2]Private communication with A. Tilgner
[3]Private communication with O. Goepfert

5.6 Data Analysis

The main interest in the analysis of the simulation data is given by the identification of possible inertial modes. An appropriate choice is to consider the energy in the z-component of the velocity, which is given by

$$\langle v_z^2 \rangle (t) = \int \mathrm{d}V \, v_z(t)^2 \approx \sum_{i,j,k}^{N_x,N_y,N_z} \Delta x \Delta y \Delta z \; v_z(t)^2 \big|_{i,j,k} . \tag{5.11}$$

This has multiple benefits

- The v_z-velocity component is independent of the coordinate system which is used. It is not necessary to compute a transformation to an inertial frame.
- The energy induced by the libration is perpendicular to this component. Thus, all components in the v_z-component are created by inertial oscillations.

In Fig. 5.4 this energy is shown exemplarily for two libration frequencies for a cylinder. [4] For a resonance of the system ($\omega = 1.2$), it takes further time until a steady state with a constant oscillation is reached.

[4]The results are taken from Sec. 5.7.1

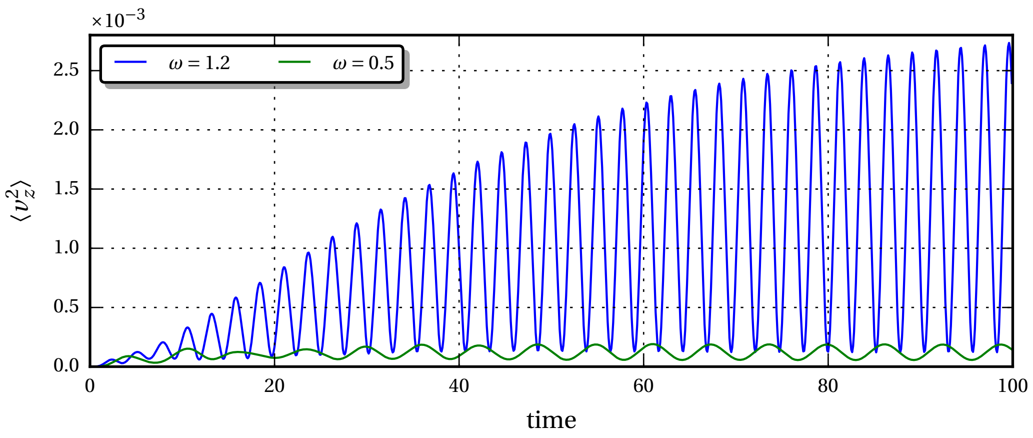


Figure 5.4.: Kinetic energy in z-direction for the Direct-Forcing method of second order, at different libration frequencies.

In order to obtain a spectrum, the energy amplitude is taken from the last two extrema of $< v_z^2 >$. The computation is given by[5]

$$A\left(\langle v_z^2\rangle\right) = \frac{\max(\mathrm{argmax}(\langle v_z^2\rangle_v)) - \max(\mathrm{argmin}(\langle v_z^2\rangle_v))}{2}. \tag{5.12}$$

Since the error of this calculation depends on the total simulation time, it was ensured that all simulations were carried out long enough. For the same simulation times, the error of one resonance is estimated in the range $\sigma_A \approx 5\% - 10\%$. The error is estimated to be of order $\sigma_A \approx 2.5\% - 5\%$, when no resonances are present. Another part of the analysis is the computation of the relative helicity of the system

$$\mathscr{H}(t) = \frac{\int_V \mathrm{d}V\,\vec{v}(\nabla\times\vec{v})}{\sqrt{\int_V \mathrm{d}V\,\vec{v}^2 \int_V \mathrm{d}V(\nabla\times\vec{v})^2}} = \frac{\sum_{i,j,k=0}^{N_x,N_y,N_z} \vec{v}_{i,j,k}(\nabla\times\vec{v}_{i,j,k})}{\sqrt{\left(\sum_{i,j,k=0}^{N_x,N_y,N_z} \vec{v}_{i,j,k}^2\right)\left(\sum_{i,j,k=0}^{N_x,N_y,N_z} (\nabla\times\vec{v}_{i,j,k})^2\right)}}. \tag{5.13}$$

For the discretization of the rotation operator, a central difference scheme of second order is used.

[5]The argmax function gives the largest argument from all maxima of a fucntion, the argmin function in analogy for all minima.

5.7 Simulation of a Librating Cylinder

5.7.1 Introduction

As a first step towards the implementation of the librating cone, simulations of fluid flow in a librating cylinder were carried out. Despite the theoretical and experimental results which are discussed so far, it is difficult to find results in literature which would be suitable for an appropriate validation for the cone. For the cylinder otherwise, theoretical and numerical results are available [Gre90; Sau+12]. The theoretical solution for the inviscid case was introduced in Sec. 1.2.2.3. Due to the excitation by libration the possible observable modes have to be axially symmetric, meaning the azimuthal wavenumber is zero. Therefore, all possible modes are defined by the indices (n, m), corresponding to the radial and axial wavenumber.
The simulations were performed with a librating cylinder, of aspect ratio $\Gamma = r/H = 0.5/1.1$ [6], and a varying libration frequency. The main simulation parameters are given by

ω	Γ	Δt	Δx	c^2	Ek	l_x, l_y, l_z	T_{end}
$[0.2,\ 2], \Delta\omega = 1/10$	$0.5/1.1$	10^{-4}	$1/128$	500	10^{-4}	(1.1, 1, 1)	100

.

The simulations were performed for all IB methods used in this thesis, with second and fourth order finite difference schemes. The results are presented and discussed separately in comparison to the results of the librating cone in the next section, for reasons of clarity.

[6]Due to a misunderstanding the aspect ratio of the cylinder was set to $\Gamma = r/H = 0.5/1.1$ instead of $\frac{1}{2}$.

5.7.2 Results

5.7.2.1 Frequency Spectrum

For all simulations the kinetic energy of the z-component of the velocity was computed. The results for the computation of the amplitude $A\left(\langle v_z^3\rangle\right)$, in dependency of the libration frequency are shown in Fig. 5.5. All IB methods show a similar profile. The largest amplitude deviations can be spotted at a frequency of $\omega = 1.2$ and are about $5 \cdot 10^{-3}$.
It can be noted that at this position the largest resonance MI occurs which is about $3.5 \cdot 10^{-2}$. Two other resonances can be observed at $\omega = 0.75$ (MII) and $\omega \approx 1.7$ (MII) which are about $\approx 1.3 \cdot 10^{-2}$. To make the deviations between the methods more visible two inset plots, which are set to the location $\omega = 0.8$ and $\omega = 1.0$, are shown. For both positions, the largest amplitude is given by the Direct-Forcing method of second order, followed by the Volume-Penalization method of second and fourth order. The amplitude difference between the first two methods is just slightly visible when looking at the right inset plot. It seems like these three methods yield nearly the same results for all frequencies. For all remaining methods no strict order or grouping can be recognized.
The results for the interpolation method are not presented here, since the simulations became numerically unstable.

5.7.2.2 Eigenvalues for the inviscid equations

For a comparison to the theoretical solution, the eigenvalues for the (n, m)-inertial modes with $n \in [1,6]$ and $m \in [1,2]$ were computed. The results are shown in Table 5.1. There are four possible candidates, which could correspond to the peaks in the spectrum. The largest peak at MI is close to the eigenvalues of the (2, 1) and (4, 2) mode. The left peak MII could be related to the (2, 2) mode, whereas the peak on the right side MIII is close to the eigenvalue of the (4, 1) mode. The eigenvalues for uneven n are not considered. Due to the excitation by libration only azimuthal velocity fields are possible, which are even with respect to the plane $z = H/2$ [Sau+12]. [7] Furthermore we are not considering modes with $m \geq 2$. According to [Sau+12], these modes are not visible at an Ekman number of $5 \cdot 10^{-5}$ as a result of the viscous damping at large wave numbers.
A further comparison was done by computing the eigenmodes for the possible (n, m) values of MI, MII and MIII. The results are shown in Fig. 5.6. The assumed mode (4, 2) is not shown since the profiles do not match. The theoretical solution yields the pressure field of the inertial mode. Since this quantity is not directly accessible from the simulation results the absolute velocity in z-direction integrated over the simulation time is computed. This expression should be proportional to the theoretical pressure gradient in z-direction, with exception of the sign, which is eliminated by taking the absolute value of v_z.

[7] An uneven pressure gradient from the theoretical solution results in an even velocity field.

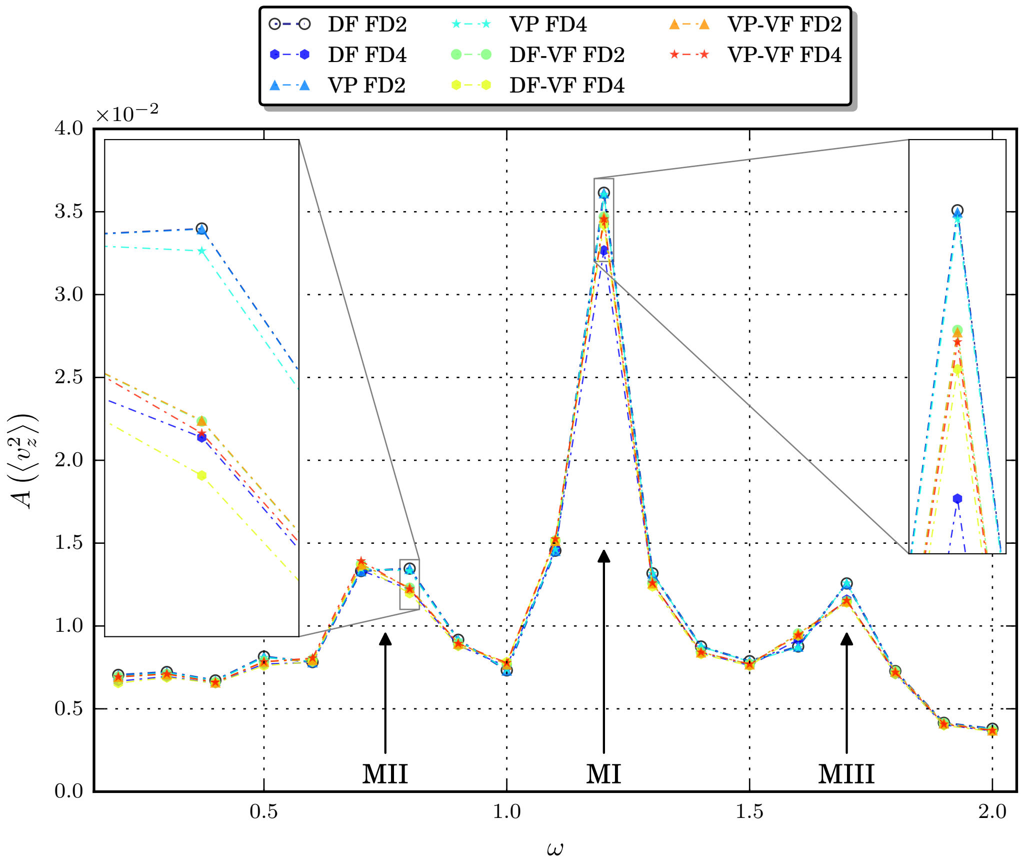


Figure 5.5.: Amplitude of the kinetic energy in v_z, as function of the liberating frequency ω. For the Direct-Forcing (DF), Volume-Penalization (VP) method and the Volume-Fraction (VF) method second (FD2) and fourth (FD4) order finite differences.

	m = 1	m = 2
n=1	0.698433	0.398913
n=2	1.195232	0.754094
n=3	1.490715	1.042313
n=4	1.660912	1.262759
n=5	1.762270	1.426587
n=6	1.825743	1.547435

Table 5.1.: Eigenvalues of the (n, m)-inertial modes in a cylinder with aspect ratio 5/11. Possible modes are highlighted.

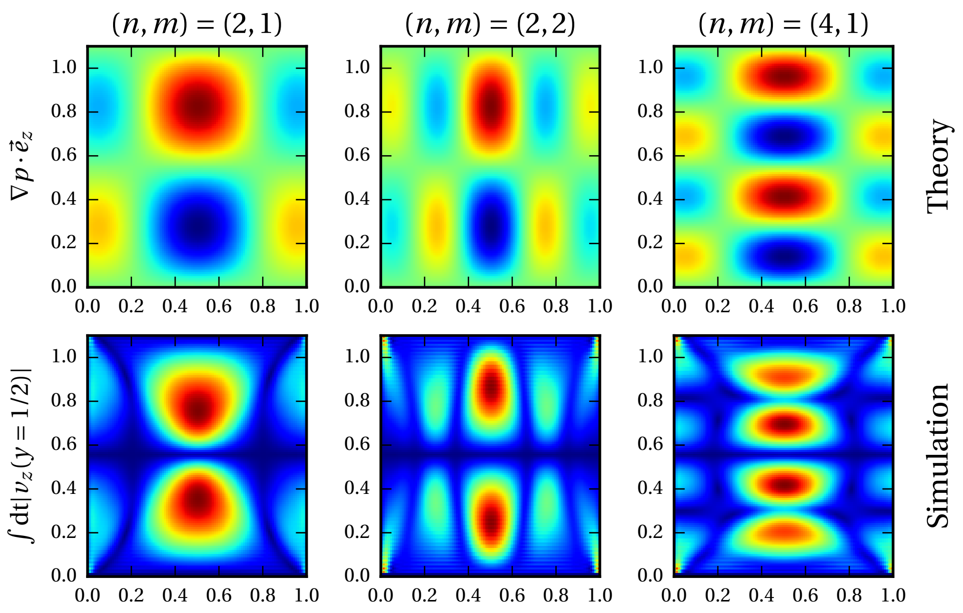


Figure 5.6.: Theoretical inertial modes compared to the absolute of the velocity component v_z of the simulation. For the peaks I, II and III.

Indeed, the theoretical profiles for the inviscid problem look very similar to the simulated profiles. The simulation profiles are slightly distorted on the walls of the fluid domain in comparison to the theoretical profiles. For all modes the number of wave nodes in z and x direction is identical when comparing the theoretical and numerical profiles.

5.7.2.3 Helicity

The computation of the helicity was split into an integral over the upper and lower half of the cylinder. A result for $\omega = 1.5$ is exemplarily shown in Fig. 5.7. In the upper half of the cylinder a negative helicity can be observed with the maximum of the amplitude is of order 10^{-1}. In the lower half of the cylinder a positive helicity can be observed with the maximum of the same order. The helicities are symmetric with respect to zero, thus $H_{\text{up.}} = -H_{\text{low.}}$. As a result the total helicity of the cylinder is zero for all times.

5.7.2.4 Numerical Oscillations

For all IB methods numerical oscillations are observable. This is shown in Fig. 5.8 for the Direct-forcing method of second and fourth order and $\omega = 1.5$. The profile is taken along the z-axis at $(x, y) = (1/2, 1/2)$. It can be noted that the error in the regions near to the wall is larger.

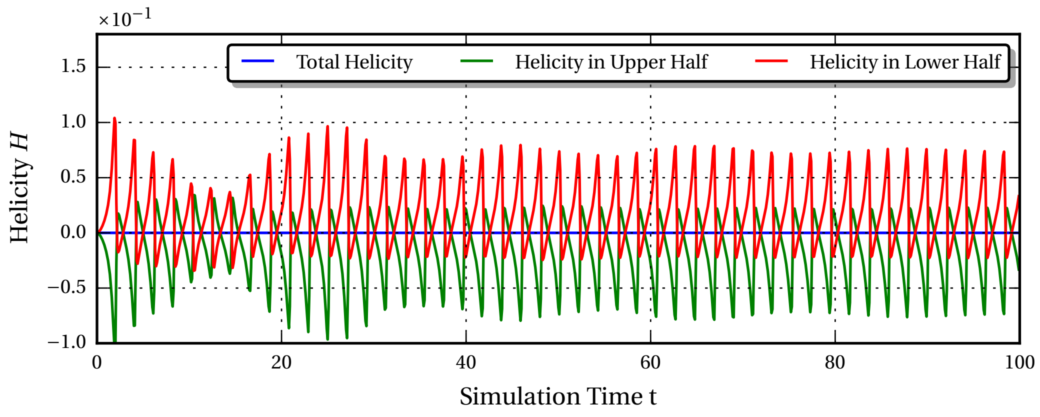


Figure 5.7.: Time-dependent Helicity $H(t)$ for $\omega = 1.5$. The Direct-Forcing method of second order was used.

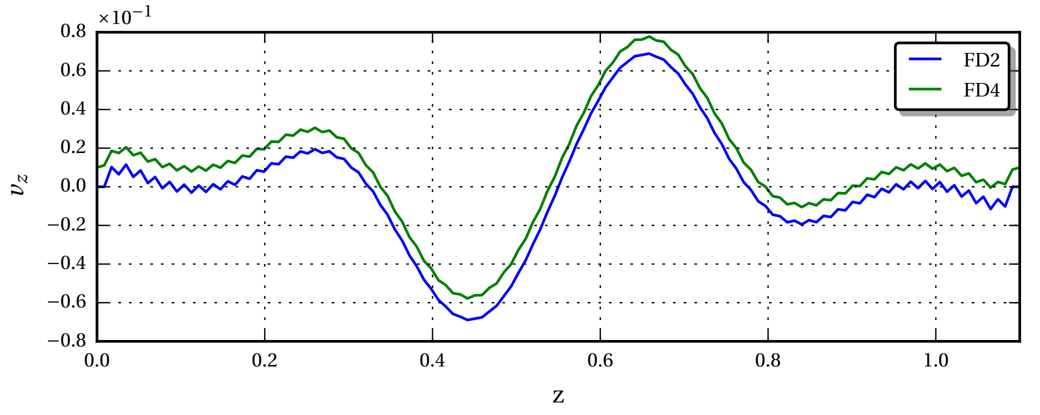


Figure 5.8.: Numerical oscillations along the z axis trough $(x, y) = (0.5, 0.5)$. For the Direct-Forcing method of second (FD2) and fourth (FD4) order, at $\omega = 1.5$.

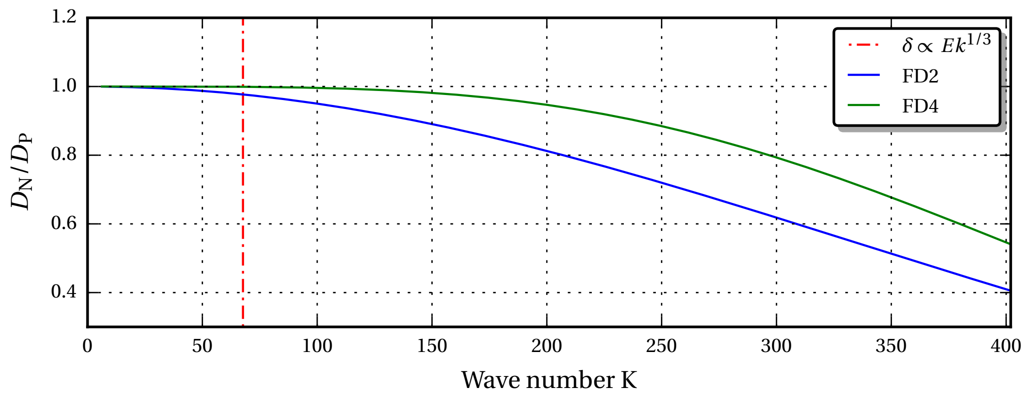


Figure 5.9.: Ratio D_N/D_P of numerical viscosity D_N and physical viscosity computed for $Ek = 10^{-4}$. For second (FD2) and fourth (FD4) order central differencing schemes. The red line indicates the wave number of an inertial wave with width $Ek^{1/3}$.

One detail is that the oscillations are only observable in z-direction and not in th (x, y) plane. In contrast to the FTCS-Scheme of the second order method, the fourth order method uses the upwind scheme. It is visible that the oscillations are slightly diminished for the upwind method.

5.7.2.5 Numerical Viscosity

The ratio between the numerical and physical viscosity was computed with Eq. 2.33 and Eq. 2.34. The results are shown in Fig. 5.9. The red bar denotes the wave number for the width of an inertial wave, which is $\propto Ek^{1/3.}$ [Til00]. For this value the ratio is ≈ 0.977. For both methods a decrease in the numerical viscosity can be observed. At larger wave numbers, the fourth order method has a better approximation than the second order method. The largest wave number of the system is possible for $\lambda = 2\Delta x$, it follows that $K_{\max} = {}^{2\pi}/_{\lambda} \approx 402.1$.

5.7.3 Discussion

5.7.3.1 Frequency Spectrum

The peaks in the frequency spectrum can be identified as inertial modes. A first good indicator are the computed eigenvalues which are close to the peak positions in the spectrum. However, the assumption that MI could be the (4, 2) mode is wrong. The representation of the averaged v_z-component in Fig. 5.6, in comparison to the pressure gradient, clarifies that the peak MI corresponds to the (2, 1) mode. MII corresponds to the (2, 2) mode and MIII to the (4, 1) mode.
The distortion in the averaged velocity profiles can be explained by the viscous friction from domain boundaries. Therefore, the results of the simulations are in a high agreement with the theoretical description of the inviscid case.
A numerical study of the non-linear system was performed by [Sau+12]. The inertial modes which were given in this study confirm the present results. One difference is given by a slight shift ≈ 0.06 of the eigenvalues which is due to the mistakenly used aspect-ratio. Furthermore, three additional modes were found in this study. They do not occur in the frequency spectrum in this thesis. One explanation for this is the use of larger Ekman number in this simulations $\propto 10^{-4}$. The width of a resonance peak is proportional to the damping rate of the system, in analogy to an harmonic oscillator. [8] Due to the stronger damping in our simulations the spectrum shows broader resonance peaks than in the study of [Sau+12].
All methods are able to reproduce these results with exception of the interpolation method. Unfortunately, it was not possible to figure out the reason for the numerical instability. The almost identical errors of the Volume-Penalization and Direct-Forcing method of second order indicate that the time step could be to small. The velocities local to the domain boundaries are so small, that the

[8] Private communication with A.Tilgner

damping force (Volume-Penalization method) is almost identical to setting the velocity direct to zero (Direct-Forcing method).

5.7.3.2 Numerical Oscillations

The velocities in the system are above the order 10^{-1}. The resulting Peclet number is given by

$$Pe = \frac{0.1 \cdot 1/128}{Ek} = 15.625 \tag{5.14}$$

This value gives a possible explanation for the occurrence of oscillations, since it violates the stability criterion $Pe < 2$. The assumption is supported by the improvement from the upwind scheme. It still remains unclear why the oscillations only occur in the z-direction.

5.7.3.3 Numerical Viscosity

The numerical viscosity was computed in order to determine its influence on the system. For the physical properties of our system the estimated wavelengths are in the range of $1 - Ek^{1/3}$. The error of the numerical viscosity in this range can be neglected.

5.7.3.4 Helicity

The computation shows that the total helicity of the cylinder is zero. This result is not unexpected. Due to the axial symmetry in z-direction the helicities in the upper and lower part of the cylinder cancel each other.

5.8 Simulation of a Librating Cone

This section introduces the simulations which have been performed with a librating cone. The first simulation is a setup inspired by the experiment of [Bea70]. In the second setup the geometrical transition from a cylinder to a librating cone is investigated. A study of the influence of different offsets on top of the cone is performed in the last simulation. All simulations use the conventions and setup from Sec. 5.2. As an IB method the Direct-Forcing method of with second order finite difference schemes is used.

5.8.1 Simulation of the Experimental Setup

The setup for this simulation is oriented at the experiment with slightly changed parameters. The Ekman number is set to $Ek = 10^{-4}$. A simulation of higher Ekman numbers closer to the experiment is difficult to realize, due to the computational effort. In the first part of the simulation a cone is simulated with the geometric parameters $\alpha = 60°$, $H = 1$ and $r = 0$. This means that the critical libration frequency is $\omega_c = 1$, where an inertial wave traverses parallel to the slope of the cone. The resulting offset on top is given by

$$O = H - h_c = 1 - {}^{l_x}\!/\!{}_2 \tan\alpha \approx 0.134. \tag{5.15}$$

In the second simulation the cone is replaced by a frustum, by setting a bottom plate to the height 0.25. The total height is reduced to $H = 0.75$ and the radius of the plate is $r \approx 0.144$. A series of simulations for the two setups were performed. The main simulation parameters are given by

ω	Δt	Δx	c^2	Ek	l_x, l_y, l_z	T_{end}
$[0.2, 2], \Delta\omega = 1/10$	10^{-5}	$1/128$	500	10^{-4}	$(1, 1, \{1, 0.75\})$	100

.

5.8.2 Results

The results of the simulations are shown in Fig. 5.10. For both cases, the frustum and the cone as well, an increase in $A\left(\langle v_z^2 \rangle\right)$ can be observed. From about $2 \cdot 10^{-4}$ at $\omega = 0.2$, to $\approx 3.4 \cdot 10^{-4}$ for the cone and $\approx 4 \cdot 10^{-4}$ for the frustum, at $\omega = 1$. From here on, the amplitude decreases to $\approx 5 \cdot 10^{-5}$ at $\omega = 2$. A difference between the two setups can be observed in the surrounding area of $\omega = 1$.
For the frustum the increase of the amplitude is interrupted at $\omega = 0.7$, a minimum can be observed at $\omega = 0.8$, followed by a peak at $\omega = 1$. For the cone a minimum is not directly visible. However, it can be noted that the amplitude does not increase as much at $\omega = 0.7$ as for lower frequencies.

The maxima of the frustum ($\omega = 1$) and cone ($\omega = 0.9$) do not occur at the same frequencies and have a difference of $0.5 \cdot 10^{-4}$ in amplitude.

It has to be considered, that due do the step width of $\Delta\omega = 0.1$ the exact position of the maxima is not apparent. It appears that the expansion of the frustum to a cone leads to a frequency shift to the left. Furthermore, it seems that the spectrum can be divided into two domains $0 \le \omega \le 1$ and $1 \le \omega \le 2$. This result is not unexpected, since the slope α of the cone was chosen such that $\omega_c = 1$.

5.8.3 Discussion

For $\omega < \omega_c$ the results for both setups are similar. A possible explanation is that the cone tip does not act as an attractor in this domain. An inertial wave propagates to the top after a reflection on the side of the cone. Hence, for both setups a similar spectrum exists.

The differences occur when ω reaches the critical frequency $\omega = \omega_c$. In this scenario, an inertial wave emitted from the top edge traverses directly into the apex of the cone or is reflected slightly at the bottom plate of the frustum. As a result the slight increase in the amplitude for the frustum can be observed.

For $\omega > \omega_c$ further resonances in the spectrum of the frustum should be observable, since the bottom plate enables the reflection of inertial waves. The similar decay of the amplitude for both geometries refutes this assumption. Neither the experiment [Bea70] nor the theoretical description [Gre90] are in accordance with the results of the simulation.

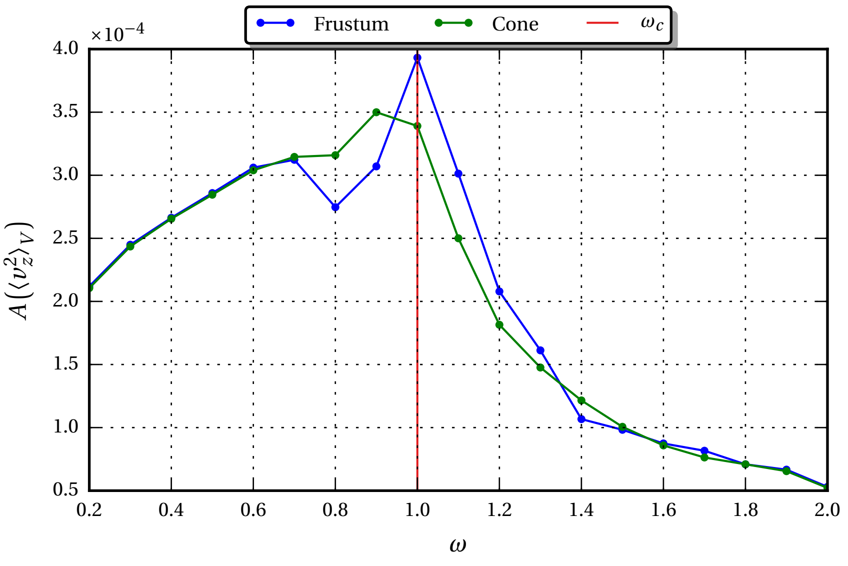


Figure 5.10.: Amplitude $A(\langle v_z^2 \rangle_V)$ as a function of the libration frequency ω, for the cone and the frustum.

An explanation could be given by the fact that a different Ekman number ($Ek = 10^{-4}$) was used for the simulations. The width of the envelope of an inertial wave ray can be estimated to $Ek^{1/3} \approx 0.046$, according to [Til00]. The experiment used an Ekman number of $Ek = 10^{-5}$, resulting in a width of $Ek^{1/3} \approx 0.022$. As a consequence, the width of an inertial wave ray is around twice the size as in the experiment. It can be assumed that a wave reflecting on the bottom of the frustum is stronger damped as in the experiment. The bottom plate of the frustum is too narrow to support an efficient reflection of inertial waves, due to frictional losses.

5.8.4 Simulation of the Transition from a Cylinder to a Cone

The objective of this simulation is to test the influence of different gap radii r at the bottom of the frustum. Like before it is $H = 1$ and $O \approx 0.134$ but α will now be dependent of r. The first setup contains the largest possible bottom gap with $r = 0.5$. In the next steps the size of the gap is decreased by $\Delta r = 0.125$ until $r = 0$ is reached. For $r = 0.5$ the domain is given by a cylindric geometry, since $r = l_x/2 = l_y/2$. With an decrease of r, the geometry transforms into a frustum and finally into a cone with an apex for $r = 0$. The main simulation parameters are given by

r	ω	Δt	Δx	c^2	Ek	l_x, l_y, l_z	T_{end}
$[0, 0.5], \Delta r = 0.125$	$[0.2, 2], \Delta\omega = 1/10$	10^{-5}	$1/128$	500	10^{-4}	(1, 1, 1)	100

.

5.8.5 Results

The results for the simulations are shown in Fig. 5.11. On the left side of the plot the frequency spectrum is shown, the right side illustrates the geometry used. The radius r decreases from the top to the bottom. For $r = 0.5$ the inertial modes of a librating cylinder can be observed, in accordance to the results of the librating cylinder, see Sec. 5.7.1. The result for $r = 0$ is identical to the simulation of the cone from the previous section. With a decrease of the radius a change of the position and the amplitude of the inertial modes is visible. This pattern shall be exemplarily discussed for the (2, 1) mode at $\omega = 1.3$, where it is best visible. For all other modes the transition results in a similar behavior.

The decrease of the radius leads to a shift to lower frequencies of the (2, 1) mode, from $r = 0.5$ to $r = 0.375$ it is of the order $\Delta\omega = 0.1$. It can be noticed that during this shift, a damping of the mode occurs. For $r = 0.5$ the amplitude is about $9.2 \cdot 10^{-3}$, for $r = 0.375$ it is about $8.7 \cdot 10^{-3}$ and for $r = 0.25$ about $3.8 \cdot 10^{-3}$. Whereas the first decrease of the radius does not significantly affect the amplitude, the second decrease leads to a strong damping to less than half of the original value. With a further decrease in r, the (2, 2) mode is annihilated. For the (4, 1) mode, a slight increase of the amplitude at

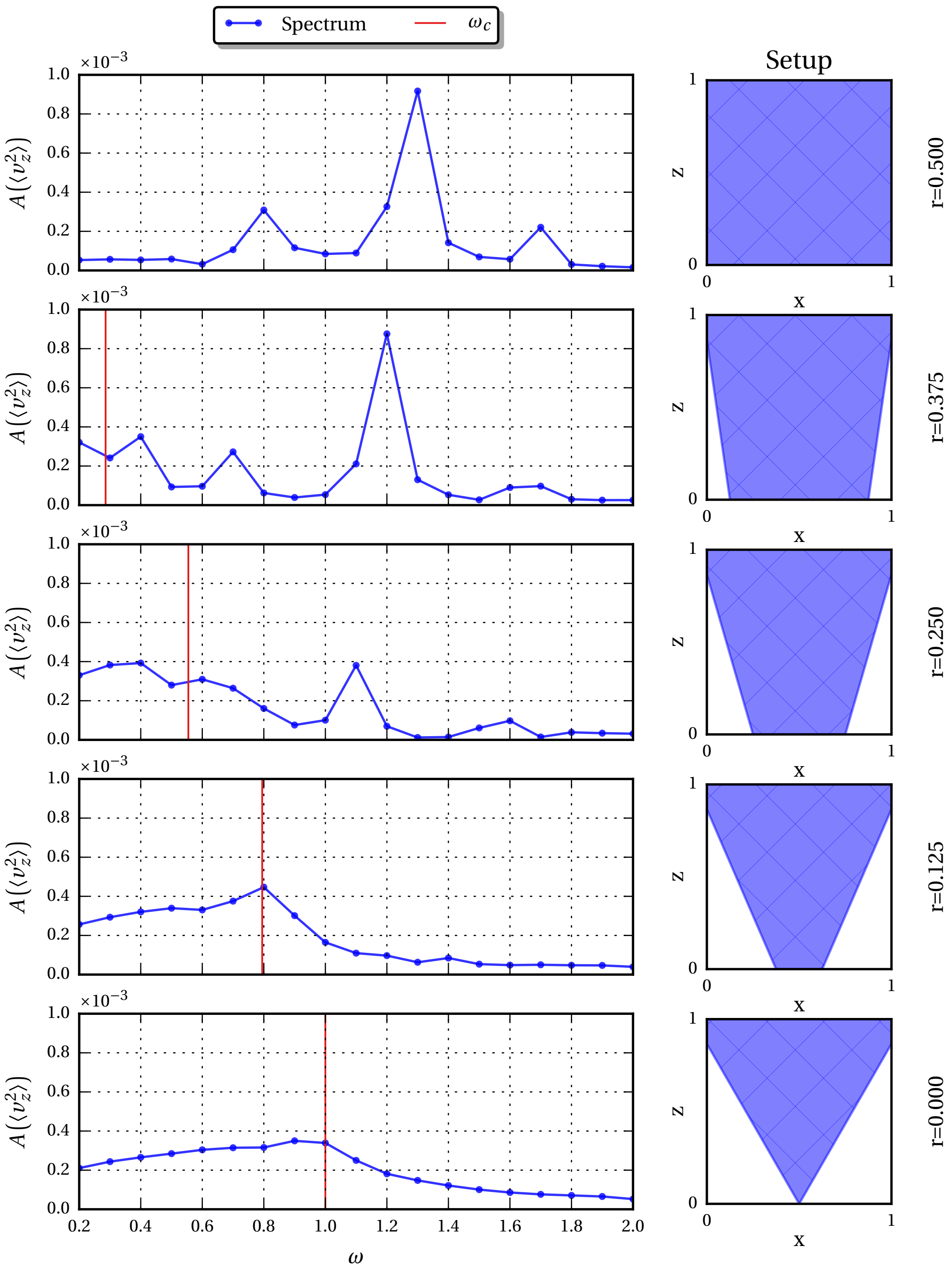


Figure 5.11.: Amplitude $A(\langle v_z^2 \rangle_V)$ as a function of the libration frequency ω, for the transition from a cylinder to a cone.

$r = 0.125$ is still noticeable. In addition this mode is stronger damped for $r = 0.375$, in comparison to the (2, 1) mode. The vertical lines in Fig. 5.11 are set to the position $\omega = \omega_c$. The area $\omega < \omega_c$ can again be associated with the frequency domain, where the wave propagation angle θ is larger than the slope of the cone α, so that downward traveling waves propagate to the top after a reflection. In this area the amplitudes increase, in accordance with the results from the simulation of the experiment.

5.8.6 Discussion

The simulation shows that the transition from the cylinder to the frustum results in a shift of the modes. Furthermore, it can be noticed that all modes are damped with a decrease of the bottom gap width. An explanation for the shift can be given by a look at the inertial mode structure for different radii, as shown in Fig. 5.12, for the (2, 1) mode.

For $r = 0.5$ an inertial mode is visible which is symmetric to the plane $z = H/2$. In this plane the waves excited from the bottom and top of the cylinder annihilate each other to a wave node. A decrease of the radius breaks the vertical symmetry of the inertial mode. As a result only a distorted version of the mode can exist, where the center of the wave node is shifted downwards.

The attempt was made to obtain the new position of the center by computing the intersection of the diagonals from the top to the bottom edges of the frustum. The resulting center height is given by

$$c = r \frac{H}{r + {}^{l_x}/_2}. \tag{5.16}$$

The resulting points are in a good agreement, as the marked position in Fig. 5.12 shows. For the same inertial mode the propagation angle θ has to be increased for it to exist in the narrower shape.
This is equivalent to a shift to a lower libration frequency. For $r \to 0$ the center position converges against zero. Hence, the mode cannot exist in this state.

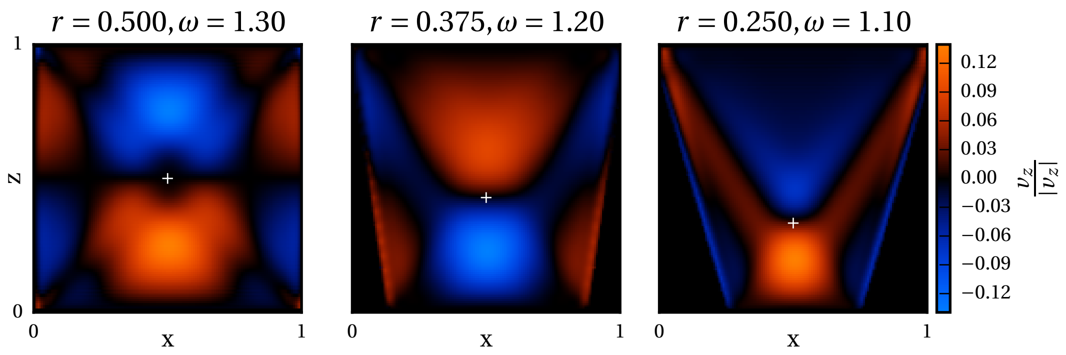


Figure 5.12.: Inertial mode structure for the (2,1) mode from the cylinder at different radii on the bottom of the frustum. The white points mark the center of the wave node, computed with Eq. 5.16.

5.8.7 Simulation of a Librating Cone and Frustum with Differing Offsets at the Top

For the simulations which have been carried out in this thesis yet, the influence of the offset on top of the cone has been left out from any discussion. As already explained, this offset has an influence on inertial wave propagation. In particular for the interval $\omega < \omega_c$ where downward traveling waves are reflected upwards. In the case where the offset is set to zero the corners of the cone act as an attractor. For small gap sizes it could be possible that an inertial wave will be damped. This would be in accordance with the previous results and motivates a simulation series with different offsets. The simulation of the transition from a cylinder to a cone showed that a damping acts on an inertial wave when reflecting at the bottom. This damping depends probably on the radius of the bottom gap and motivates the use of a larger radius at the bottom in comparison to the experiment.

For the last simulation series, two setups with different offsets on top of the cone are used. The first setup is a cone with height $H = 1$, $r = 0$ and $\alpha = 60$. The second setup is the frustum with a larger gap radius $r \approx 0.22$ in comparison to the experiment.

The position of the bottom plate is than $h_b = 0.375$ and the resulting frustum height is $H = 0.625$. An additional offset is added to the default one, varying in the range of $h_+ = [0, 0.5]$ with a stepsize of $\Delta h_+ = 1/8$. The complete offset at the top is $O \approx 0.134 + h_+$. The main simulation parameters for both setups are given by

h_+	ω	Δt	Δx	c^2	Ek	l_x, l_y, l_z	T^*_{end}
$[0,\ 0.5], \Delta h_+ = 0.125$	$[0.2,\ 2], \Delta\omega = 1/20$	10^{-5}	$1/128$	500	10^{-4}	$(1, 1, 1 + h_+)$	$\gtrsim 100$

.

Compared to the previous simulations a higher resolution was used for the frequency. The total simulation time $T^*_{end.}$ is marked as $\gtrsim 100$. Here, the modification

$$T^*_{\text{end.}} = \underbrace{\left\lfloor \left(\mathrm{T}_{\text{end}} \frac{\omega}{2\pi} \right) \right\rfloor}_{\substack{\text{Number of Periods} \\ \text{below } T_{\text{end}}}} \cdot \frac{2\pi}{\omega} + \frac{2\pi}{\omega} \tag{5.17}$$

is introduced. As a consequence the ending time of all simulations is set to the point where the libration force is crossing zero. The scenario which will be studied from there on is the decay of inertial waves.

To achieve this, a simulation with the previously computed end state is continued, with the libration amplitude ϵ_0 set to zero. By using a modified end time, all intertial waves are assumed to be approximately in the same phase with respect to their libration frequency. The end time for this simulations is set to $T_{end} = 150$. Finally a simulation series will be repeated, by using the same settings as before, but with a higher resolution of $\Delta x = 1/256$.

5.8.8 Results

5.8.8.1 Spectrum

The results for this simulation are shown in Fig. 5.13. On the left side of the plot the frequency spectrum is shown, the right side illustrates the geometry used. The height h_+ is in descending order, from top to bottom. Each plot contains two spectra, one for the cone (green) and the other one for the frustum (blue). We begin with a qualitative description of the curves for the frustum.
At all heights h_+ several resonance modes can be observed. For the height $h_+ = 0.25$ the largest peaks occur in the spectrum, which are numerated here in decreasing order of the amplitude. The largest peak MI can be observed at a frequency $\omega = 1.25$ and an amplitude of $5 \cdot 10^{-4}$. The next resonances are: MII at $\omega = 0.9$, $A = 3.7 \cdot 10^{-4}$, MIII at $\omega = 0.5$, $A = 2.8 \cdot 10^{-4}$. On the right side of the spectrum there are two possible small resonances at $\omega = 1.7$ and $\omega = 1.5$ (MIV and MV). For the case $h_+ = 0.5$ small additional resonances MVI at $\omega = 1.8$ and MVII at $\omega = 0.55$ can be noticed, the latter is embedded in the slope of the MIII peak.
Now the results of the cone are considered. Above the critical slope $\omega \geq \omega_c$ nearly all resonances are eliminated. There are two exceptions, the MVI and **O** peak for $h_+ \geq 0.375$. The **O** peak can be associated with an MI mode for the cone. For further discussions the minimum ω_{min} to the left of MVI is shown.
On the left side of the critical frequency ω_c, the MIII-peak can be identified at the same frequency and with the same amplitude as for the frustum. To the left side of this point, the spectrum for the cone and the frustum are similar. Furthermore, at the position of peak MII of the cone, a slightly diminished peak MVIII can be noticed. The maximum has the same position for $h_+ = 0.5$, but diverges slightly with a decrease of the height h_+.
For all peaks it can be noted, that a lowering of the offset height h_+ results into a shift towards higher frequencies. For a better overview, the changes in the amplitudes and shift in the frequencies are shown in Fig. 5.14. The amplitudes of the two peaks MI and MII have a maximum at $h_+ = 0.25$. For the peak MIII and MVIII the amplitude decreases almost linearly. The second plot shows that with a decreasing height h_+, all peaks shift approximately linearly to lower frequencies.
In order to give a better insight on the modal structures the velocity profiles for the peaks MI, MII and MVIII are presented in Fig. 5.15 for different geometries . For each setup the velocity profile is shown for two different time steps. The offset Δt^* varies for each geometry in order to emphasize the important flow properties. For the peak MI and MII in the frustum, the velocity profile looks like a standing wave, in contrast to the cone where a downward traveling wave can be observed for the peak MI and MVIII.

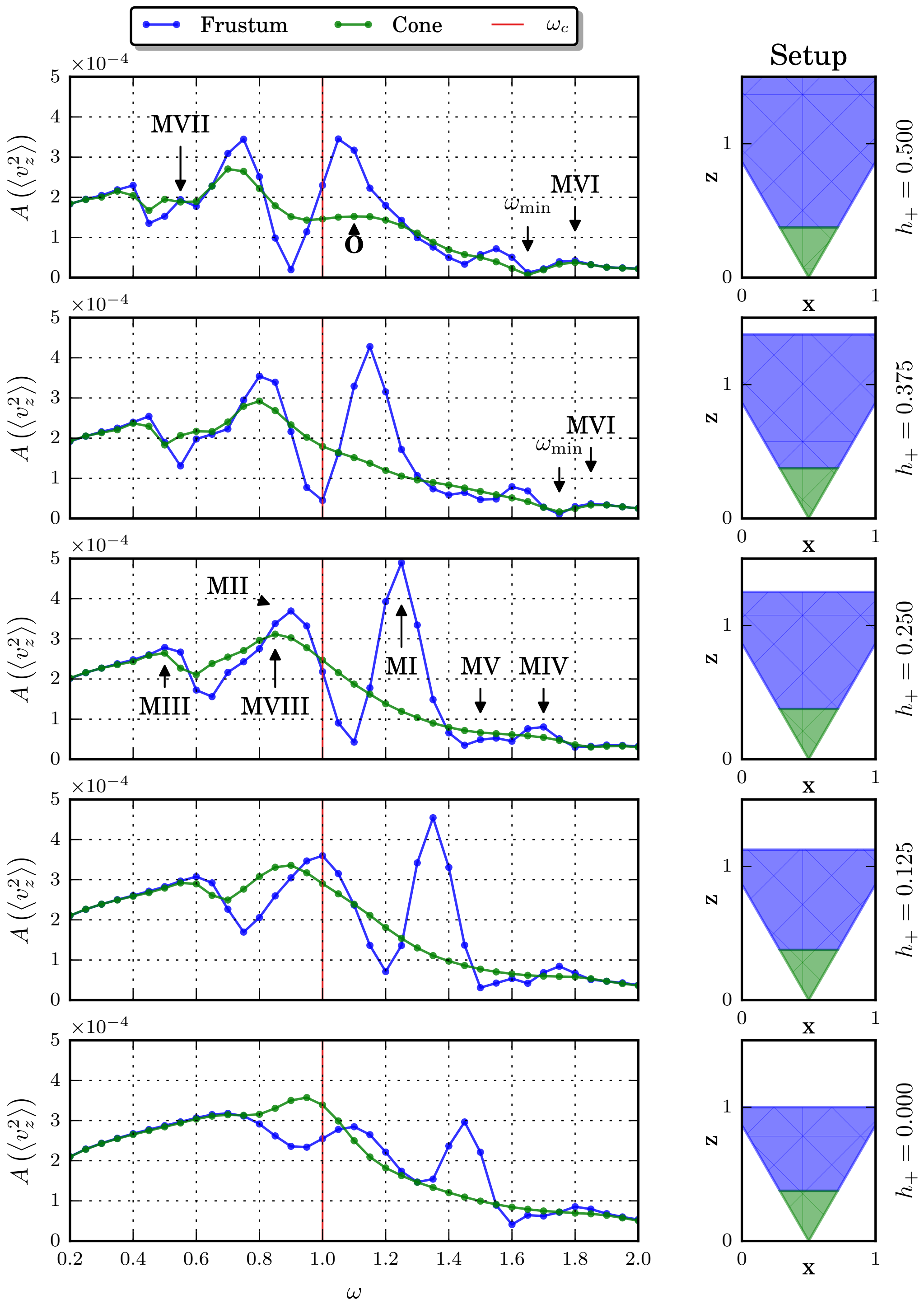


Figure 5.13.: Amplitude $A\left(\langle v_z^2 \rangle_V\right)$ as a function of the libration frequency ω, for the cone and the frustum for different offsets h_+.

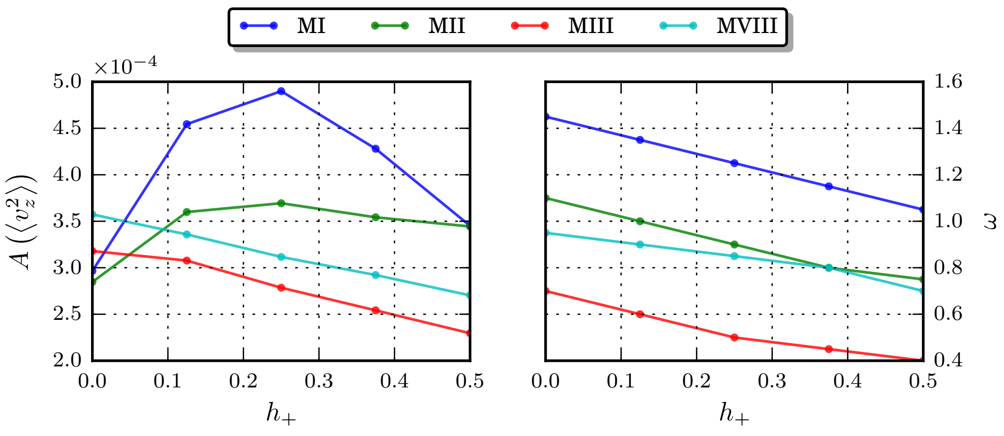


Figure 5.14.: Amplitude and frequency position of the resonances MI, MII, MIII and MVIII for different offsets h_+.

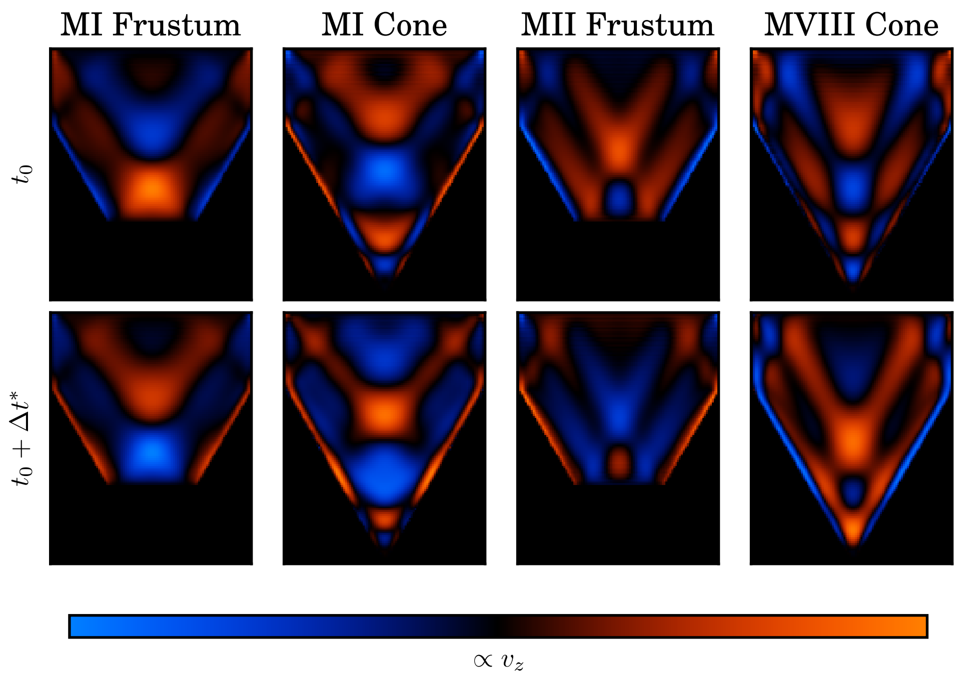


Figure 5.15.: Velocity component v_z in the xz plane at y=1/2. For the peaks MI, MII and MVIII in the cone or frustum at $h_+ = 0.25$.

5.8.8.2 Helicity

The computed relative helicity is exemplarily discussed for the case $h_+ = 0.25$. For all other heights, frequency shifts and small changes in the amplitude can be observed, that are not of importance for further discussions. The results are shown in Fig. 5.16. For the computation an inertial frame of reference was used. Therefore an additional offset in the velocity has to be considered, that is

$$\vec{v} = \vec{v}|_{\text{rot.}} + \Omega(t) \times \vec{r} = \vec{v}|_{\text{rot.}} + \begin{pmatrix} -y(1+\epsilon_0 \sin(\omega t)) \\ x(1+\epsilon_0 \sin(\omega t)) \\ 0 \end{pmatrix}. \tag{5.18}$$

Furthermore, the relative helicity was averaged over the time domain. For both cases, the frustum and the cone, a positive helicity can be observed in the region $\omega < \omega_c$, which is approximately $5 \cdot 10^{-2}$. At the critical slope $\omega = \omega_c$ the helicity decreases below zero and is of the order 10^{-1}. For $\omega > 1.8$ the helicity is increasing slightly above zero to the order 10^{-3}. There are marginal differences between the cone and the frustum. Moreover, it can be noted that the positions of the peaks MI, MII and MVIII, correspond to an increasing helicity. The computed relative helicity of the system is small in comparison to the maximum value of 1.

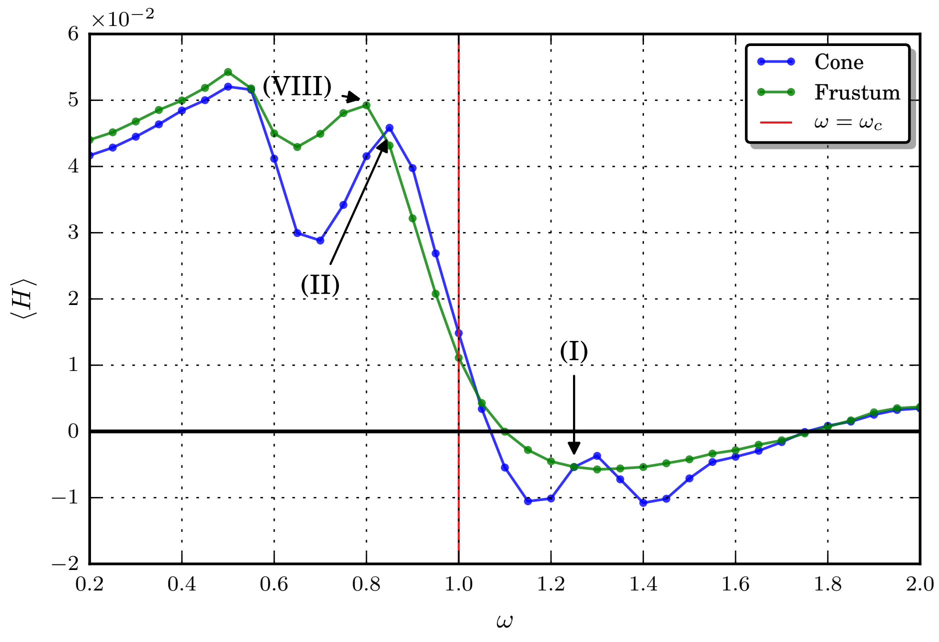


Figure 5.16.: Relative helicity for $h_+ = 0.25$ of the cone and the frustum.

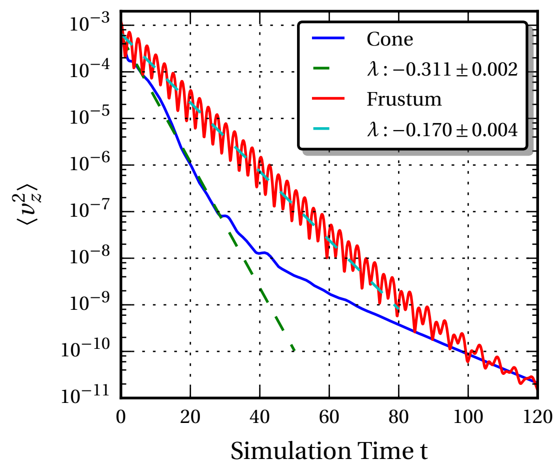


Figure 5.17.: Decay of the MI peak in the frustum in comparison to the cone.

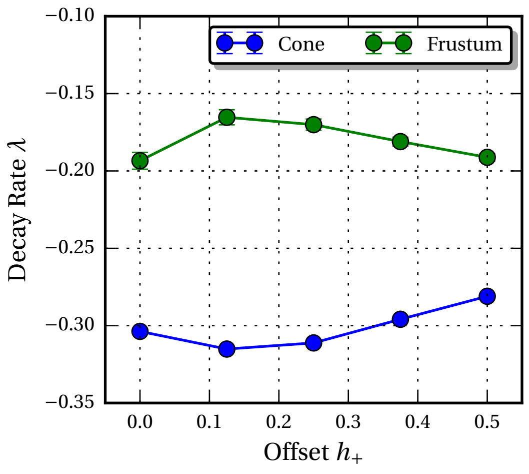


Figure 5.18.: Interpolation of the decay rate λ for different heights h_+.

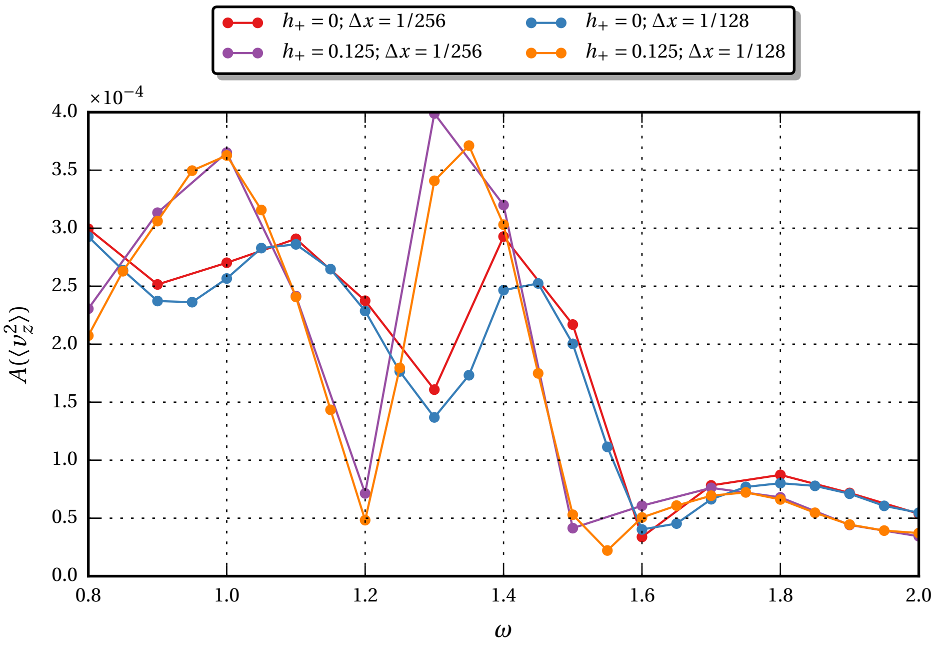


Figure 5.19.: Comparison of the results for the frustum and the same simulation with a higher resolution.

5.8.8.3 Decay Rates

The simulation of the decay of inertial waves was performed for all heights h_+ of the peak MI in the frustum. In comparison the same frequencies were used for a simulation of the decay in the cone. This choice is motivated by the position of MI which is above ω_c for all heights. Hence, the expectation is a fast decay into the apex for the cone and a slower decay for the frustum. Fig. 5.17 shows the decay exemplarily, for $\omega = 1.25$ and $h_+ = 0.25$ on a semi-log plot.
For both geometries two different decay rates are observable. The first decay for the frustum takes place in the time interval $t \leq 100$, followed by a slow decay for $t > 100$. The cone shows a faster decay in the interval $t \leq 30$, followed by a slow decay for $t > 30$. The energy in the cone decays faster until $t \approx 100$. After this time both decay rates are equal. In contrast to the cone, the decay of the frustum is superimposed by an oscillation. This oscillation is the resonance of the MI mode which is still observable in the v_z velocity component.
Fig. 5.18 shows the interpolated values of the decay rate for different heights h_+. The time interval for the interpolation was set to $0 \leq t \leq 25$. The decay rate of the cone is about $1.5 - 2$ times as large than the decay rate for the frustum.
Finally, Fig. 5.20 shows the velocity profile for different times, during the decay in the cone. During time passes, the wave is traveling into the apex of the cone. A profile for the frustum is not shown here, but it should be mentioned that in this case reflections to the top occur and the decay is homogeneous over the fluid domain.

5.8.8.4 Comparison with a Higher Resolution

A part of the simulations was repeated with a spatial discretization of $\Delta x = 1/256$. Since the computation time at this resolution is expensive, the frequency was limited to the interval $0.8 \leq \omega \leq 2$ with a step size of $\Delta\omega = 0.1$ and the heights $h_+ \in \{0.0, 0.125\}$. The results in comparison to the lower resolution are shown in Fig. 5.19. All peaks in the spectrum for the high resolution are in accordance

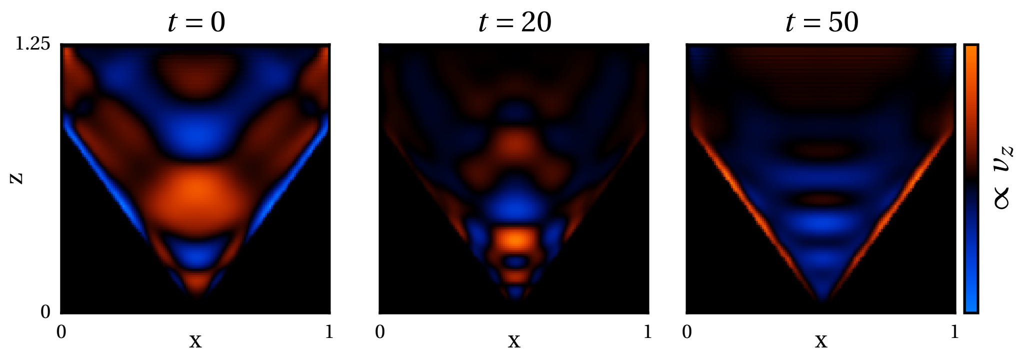


Figure 5.20.: Velocity profile for the decay in the cone for different times, for $h_+ = 0.25$ and $\omega = 1.24$.

to the previous computations. It can be noted that the higher resolution spectra are slightly shifted to the top. Furthermore it shall be mentioned that numerical oscillations, like for the cylinder (see Fig. 5.8) are still present. The amplitude of these oscillations is about half the size in comparison to the oscillations for a low resolution with $\Delta x = 1/128$.

5.8.9 Discussion

5.8.9.1 Spectrum

For both scenarios, the cone and the frustum, resonances are visible. The observation of the velocity profile for these resonances indicates that the peaks MI and MII are inertial modes in the frustum. One possible assumption was that the resonance MVIII is related to the MII mode. However, the velocity profile of MVIII indicates a downward traveling wave. Since the peak is located below the critical slope $\omega < \omega_c$, the cone apex should not act as an attractor, which is in contrast to this observation. The propagation angle for MVIII is about 65°. For this value a wave ray originating from the upper edges of the cone travels directly into the cone apex. It is possible that such a wave is not reflected due to damping effects in the apex of the cone. As a result a downward traveling wave is observed. However, with this approach the resonance of the MVIII peak in the spectrum can't be explained.
The shift of the resonance peaks is a result from the change in the geometry. In analogy to the discussion in Sec.5.8.6 we assume that with a change of the offset on top a possible inertial mode can only exist in a distorted state. Therefore, the propagation angle θ has to change, which is equivalent to a shift in the frequency domain.
We were not able to find a reasonable explanation for the change in the amplitudes. The assumption was that for $h_+ = 0$ the damping due to the small corners at the top would diminish the amplitudes the strongest. For the inertial modes MI and MII an increase of the amplitude is observed until $h_+ = 0.25$, which is in accordance with the assumption. However, the decrease for $h_+ > 0.25$ is difficult to explain. Even more confusing are the peaks MIII and MVIII where a decrease in the amplitude can be observed.
For the cone geometry the MVI resonance is still apparent. The assumption, that this resonance is an inertial mode, violates the argument that the apex of the cone acts as an attractor. A possible explanation is given in Fig. 5.21. The plot shows the wave path from an upper edge of the cone for the frequency ω_{min}. With an increase in the frequency the propagation angle decreases. For $\omega < \omega_{\text{min}}$ the wave is reflected on the slope of the cone. For $\omega > \omega_{\text{min}}$ the reflection is above the slope on the walls resulting from the offset O. The assumption is that the reflection on the slope of the cone creates more dissipation. This is due to the Direct-Forcing method, which creates a more pixelized border on the slope, in comparison to the side walls on top of the cone. As a result the energy and therefore the amplitude increases when the wave ray is passing the corner to the top. It is still not possible to

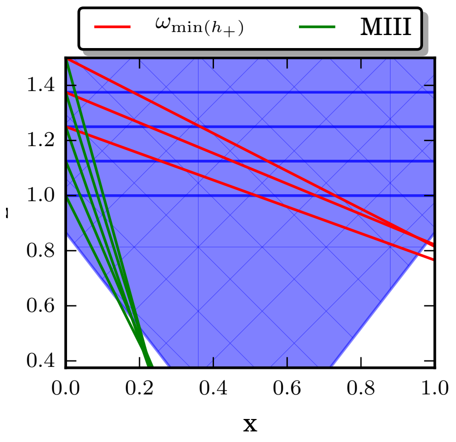


Figure 5.21.: Wave paths from the upper edges of the cone, for a wave ray with a frequency at the position $\omega_{\min}$ (red) and for a wave ray with the frequency at the position of peak MIII (green), for different heights h_+.

explain the slight increase **O** by using this argumentation.
A similar idea can be applied to the left side of the peak III (see Fig. 5.21). For $\omega < \omega$(MIII) the wave is reflected on the slope of the cone. for $\omega > \omega$(MIII) the reflection moves to the bottom plate of the frustum, or remains on the slope in case of the cone. This could explain why for $\omega < \omega$(MIII) the spectra look identical, since the wave ray does not see the difference between the cone and the frustum. This argumentation is furthermore consistent with the results from the first simulation (see discussion Sec. 5.8.3). For both assumptions the computed wave rays shown in Fig. 5.21 do not match exactly and are slightly shifted from the corners of the cone. This is probably a result of the step size $\Delta\omega = 0.05$. Hence, the exact position of $\omega_{\min}$ and MIII is not given. The shift could be resulting from the width of a wave ray $\propto Ek^{1/3}$.
As last point of the discussion of the spectrum, the MI inertial mode is discussed. The results of the simulations show that an inertial mode exists for the frustum. By taking away the bottom plate the inertial mode is not apparent instead a downward traveling wave into the apex can be observed. This results is in excellent agreement with the theoretical ideas and the experiment we introduced in Sec. 5.3 and Sec. 5.4 from [Gre69; Bea70].

5.8.9.2 Helicity

The objective behind the computation of the helicity was to see if this system could be used for further simulations, concerning MHD-equations and the generation of a dynamo. The results show that the total helicity of the system is not zero in contrast to the cylinder. An explanation can be given by the asymmetry of the system in z-direction. The frequency spectrum of the relative helicity of the system can be separated into two different regions. For $\omega < \omega_c$ the largest helicity of order $5 \cdot 10^{-2}$ is observable. This value is too small to be of interest for further researches. [9]

[9]Private Communication with A.Tilgner

5.8.9.3 Decay Rates

The computed decay rates show that an inertial mode in the frustum has a slower decay than the equivalent case in the cone. Moreover, the velocity profiles show that in the frustum the inertial mode decays homogeneously over the whole domain. In case of the cone an inertial wave travels down into the cone, where it decays. Those results demonstrate that the apex of the cone is acting as an attractor. In respect to that, it can be said that the system could be interesting for further researches, for example the study of the decay of turbulences in the apex of the cone.

5.8.9.4 Higher Resolution Simulation

The resulting frequency spectra for the higher resolutions show, that the default resolution of $\Delta x = 1/128$ yields qualitative comparable results, since the overall appearance of the resonance spectrum is equal. For a better comparison it could be interesting to repeat the simulations with a higher resolution in the frequency stepping and a smaller Ekman number. Furthermore, the resulting velocity profiles support the assumption that the numerical oscillations are based on a violation of the Peclet criterion. Since the oscillations are small with respect to the physical solution and not growing over time, the error is acceptable for the moment. However, a simulation of smaller Ekman numbers could be problematic.

5.9 Summary

5.9.1 Librating Cylinder

In the first part of the chapter we reviewed the simulation of a librating cylinder. The simulation was performed as a validation test case for the cone, since for this system more literature is available for comparison. The obtained frequency spectra revealed three inertial modes, which are in well agreement with the theoretical results from [Gre90] and the numerical results from [Sau+12]. Furthermore, it could be shown that all IB methods were able to reproduce these modes. The only exception is the interpolation method, which is numerically unstable. This is a little bit of a disappointing result, since this methods tends to contain the smallest numerical error. Finally, the results show an error regarding the numerical implementation. For all test cases numerical oscillations could be observed which are related to the large Peclet number of the system. These oscillations are not directly an issue, since a growth of the error amplitude over time could not be observed. Beside that, it has to be considered that the error is small in comparison to the physical solution of the system.

5.9.2 Librating Cone and Frustum

In the second part of the chapter different simulations of a librating cone and a librating frustum were reviewed. The first simulation was oriented towards the experimental setup described in [Bea70]. However, the results did not agree with the experiment. The assumption was made that the difference is based on the stronger damping of inertial waves, due to the use of a larger Ekman number.
In the second simulation the influence of the bottom radius of the frustum was varied to test the influence on the ability of inertial wave reflection. It could be shown that a smaller radius leads to a damping of an inertial mode and that the frequency shift is related to a distortion of the inertial wave geometry.
In the last simulation series the offset on top of the cone was varied in order to test the influence on inertial wave propagation. The gap radius was set to a large value in contrast to the experiment, to enable a better reflection in accordance with results from previous simulations. The results show that the offset on top affects the position and the amplitude of the resonances found in the frequency spectrum. Furthermore, the simulation showed that inertial modes can exist in the frustum. More important is the result that the replacement of the frustum with a cone annihilates possible inertial modes for $\omega > \omega_c$. Waves are reflected downwards into the apex which is in well agreement with the theoretical discussion of [Gre69] and the experiment by [Bea70].

Conclusions and Outlook

In the first part of this thesis different Immersed Boundary methods were implemented and successfully tested with a GPU based algorithm. The validation provided good results for No-Slip boundaries, however the implementation of velocities differing from zero at the boundaries tended to be problematic. One concern of this validation is that only steady-state flows were used as test cases. This approach is justified since the local error created through approximations of the boundaries has a larger influence on the global error. However, for further validations unsteady flows could be of interest. The validation of the interpolation method resulted in the smallest numerical error by far. Unfortunately this method became numerically unstable for the simulations of the librating cylinder and cone. For possible future applications it would be important to find and eliminate the origin of this instability.

The Immersed Boundary methods introduced in this thesis were used to satisfy No-Slip boundaries, however in many scenarios it is necessary to use No-Flux and Free-Slip boundaries. One example is the Rayleigh-Bénard system. In this system a fluid in a cubic or cylindric container is heated from below and cooled from above. The resulting heat transport is purely diffusive for small temperature gradients and becomes convective for large temperature gradients above a critical instability. For experiments and numerical simulations of this system adiabatic side walls are used [Lül11].

A simple extension of the Volume-Penalization method was performed [Lül11] and to enable No-Flux walls for the temperature, a inhomogeneous thermal diffusivity $\kappa(x, y, z)$ was introduced. By setting $\kappa(x, y, z) \ll 1$ in the boundary domain, a decent approximation of No-Flux walls was achieved [Lül11]. A further improved version of the Volume-Penalization method for No-Flux walls can be found in [BKV14]. Here the No-Flux boundaries were obtained by introducing a forcing term of the form

$$\vec{f} = -\frac{H}{\nu_c}\left(n_k \frac{\partial v}{\partial x_k} - q(\vec{r}, t)\right) \tag{5.19}$$

where ν_c is a regulation parameter, $\vec{n}$ the normal and $\vec{q}$ the desired flux through the boundary. With this implementation the temperature flux is forced to zero in direction of $\vec{n}$. Both of these methods were implemented and a first simulation of a test case verified the numerically stability. However, a proper validation needs to be carried out in the future. With a validated method it would particular interesting to simulate the Rayleigh-Bénard system in a cylindric domain.

The use of Free-Slip boundaries could reduce the numerical oscillations in the simulations of the librating cone. These oscillations are the result of the large Peclet number of the system and are generated close to the boundaries where the velocity gradients are the largest. In simulations of precession driven dynamos in a cube with small Ekman numbers ($\propto 10^{-5}$), no oscillations occur when using Free-Slip boundaries [10]. A first implementation of Free-Slip boundaries was performed by inserting diagonal walls into the fluid domain which overlap with the grid points. On these walls the Free-Slip condition can simply be applied just like in the basic GPU algorithm, using the mirroring method introduced in Sec. 3.2.1. However, a first test of this implementation became numerically unstable which is yet to be understood. It would be desirable to obtain a stable Free-Slip implementation for future applications.
There are some proposals regarding the basic implementation of the algorithm on the GPU. Numerical oscillations in the density could be a concern for future simulations. This problem seems to be quiet common for numerical computations involving first order derivatives in CFD problems. A common approach to avoid this is the use of a staggered grid, where the velocities are stored on the cell faces and the scalar fields in the cell center. Alternatively, different methods exist for pressure-based solvers where the pressure is interpolated to the cell faces. One popular method in particular is the Rhie-Chow interpolation [RC83]. Different improvements of this method that could be used for a future implementation can be found in literature.
It could be of further interest to implement an unstructured cartesian grid into the GPU algorithm. This is a very difficult task which would bring some drastic improvements in return. The error of the Immersed Boundary methods could be decreased by using a higher resolved grid in the vicinity to the boundary. Furthermore, this would improve the resolution of boundary layers and reduce the oscillations resulting from the use of high Peclet numbers.
In the second part of this thesis the Immersed Boundary methods were applied to different setups of a librating cylinder, a cone and a frustum. The results of these simulations are in agreement with theoretical predictions [Gre69] and experimental results [Bea70]. An experimental study of these systems is in discussion as a possible future research project. A possible experimental setup is already in development. It contains a rotating table, which can be controlled over a serial interface to enable different angular velocities and acceleration rates. A camera module is integrated into the rotating frame and could be used for the computation of the velocity profiles with a PIV[11] method. With this setup it would be possible to validate the numerical results of this thesis. Furthermore different excitation mechanism of inertial waves could be studied. The study of the decay of turbulent flows in the apex of the cone is of particular interest.

[10]Private communication with O.Goepfert
[11]Particle Image Velocimetry

Appendices

A. Python Simulation API

A.1 Introduction

In this chapter a brief overview will be given to the high object oriented simulation API which was written in the context of this thesis, with the python programming language. The purpose of the library in general is to simplify the execution of simulations but furthermore includes the possibility to analyse the results and use visualization techniques to study the overall behavior of the simulations during runtime. At first some of the different API classes and objects will be introduced, followed by a small application in form of a grid convergence study.

A.2 Parameter File

The Parameter file contains every flag and parameter which is used during the runtime of the simulation. During compile time all parameters are converted into a specific set of C-macros which are then compiled into the CUDA code. As a file format the open-standard format **Json** [1] is used. An example of a Parameter file is shown in Listing A.1, it contains of two sections:

Conditions Flags which are set to a constant value during runtime mostly zero or one. For example the boundary conditions and interpolation methods. Not all flags have to be set for a simulation.

Parameters This set defines all parameters which are used for a simulation. In contrast to the conditions each parameter has to be defined in order to enable a proper execution.

[1] Javascript Object Notation

Listing A.1: 'Example of a "parameter.json" file.

```
{
    "conditions": {
        "all_periodic" :1  //periodioc boundaries in all directions
    },
    "parameters": {
        "BLOCKSIZE": 8,                 //Defaultl GPU blocksize
        "STEPMAX": 200000,              //Number of timesteps
        "IC_NAME": "\"c2_1000\"",       // Initial condition files (not in use anymore)
        "RAYLEIGH":0,                   // Rayleigh number, not used in this thesis
        "DELTA_T": 0.0001,              // Time step
        "SOUND_SPEED_SQUARED": 400,     // Speed of sound squared
        "PRANDTL": 0.01,                // Prandtl number, not used in this thesis
        "GPU_ID": 2,                    // GPU id, there are up to 8 gpus in one system
        "EKMAN": 0.0001,                // Ekman number
        "NX": 64,                       // Resolution in x-direction
        "NY": 64,                       // Resolution in y-direction
        "NZ": 64,                       // Resolution in z-direction
        "LX": 1.0,                      // Length in x-direction
        "LX": 1.0,                      // Length in y-direction
        "LZ": 1.0,                      // Length in z-direction
        "LZ": 1.0,
        "RUN_NAME": "\"c2_1000\"",      //Name of .ekin output files
        "NUM_GPU": 1,                   //Number of GPUs used for the computation
        "SAMPLING_RATE": 5000,          //Number of samplings for data analysis
        "NX_D": "NX/NUM_GPU",           //NX for one GPU with threadings
        "NU": 0.0001,                   //Damping force
        "PM": 1,                        //Magnetic Prandtl number (not used)
        "KAPPA": 0.0001                 //Thermal diffusivity (not used)
    }
}
```

A.3 Generator Class

The generator class is responsible for the generation of all initial data. This means the initial conditions for all variables i.e. velocity, temperature and the computation of interpolation and domain masks which are necessary for the different IB methods.
During the execution of a simulation all precomputed arrays are stored within a HDF5-File format, which is optimized for the storage and structuring of large amounts of data. Furthermore, the format simplifies the data exchange between the python API and the CUDA program.
For the generation of data a generator object has to be initialized with a generator function, there are currently implemented two different types of these functions.

Initial Conditions These functions simply generate the initial conditions for a certain flow problem, for example the Taylor-Couette flow or a simple cylindric domain. The definition of the functions can be found in **pycurb.ic**.

Testcase The test case functions extend the initial conditions the possibility to add certain forcing parameters into the time step for example a pressure gradient is necessary for the Poiseuille flow test case described in Sec. 4.3.3 All defined functions can be found in **pycurb.testcase**.

Listing A.2: 'Generator class usage'

```
import pycurb as pc #import the simulation API
import pycurb.testcase as tf #import test cases
import pycurb.ic as as ic    #import initial conditions

#create a generator object for hagen poiseuille flow in z direction
generator = pc.Generator.from_testcase(tf.cylinder_flow_z)
#set the default velocity profile, pressure gradient and radius
generator.add_option('SETV', True)
generator.add_option('PMAX', pmax)
generator.add_option('r', 1.)

#create a generator object for taylor couette flow
generator = pc.Generator.from_ic(ic.taylor_couette)
#set velocity, inner and outer radius
generator.add_option('SETV', True)
generator.add_option('ri', ri)
generator.add_option('ro', ro)
```

Additionally it is possible to create a generator which takes the data from an old simulation. In listing A.2 the creation of a generator object is exemplary shown. In a first step the generator is created by using a generator function. In the next step it is possible to define certain attributes for example the radius of a cylindric fluid domain.

A.4 Simulation Class

An instance of the simulation class is the main object of the API and necessary to execute a simulation. For the creation of a simulation object the initial arguments are the file path, where all simulation data will be stored and the json-file for the parameter setting. Following the creation of the simulation object it is possible to alter all parameters and conditions which were previously stored in the json-file. This gives the possibility to create simulations with different parameter settings on the fly, i.e. changing the resolution of the numerical grid to perform a grid convergence study. Before the execution of the CUDA code, the simulation object has to be bind to a generator object with the **generate_files** class function, in return the generator object will begin with the data creation. The last step is the execution of the CUDA code which can be done with the **start_simulation** class function. A minimal use case of the complete procedure is shown in listing A.3.

Listing A.3: 'Simulation class usage'

```
import pycurb as pc
import pycurb.ic as ic

#create a generator
generator = pc.Generator.from_ic(ic.taylor_couette)
#create a simulation object
sim = pc.Simulation('data', 'parameter.json')
sim.parameter.set('NX', 128)
sim.parameter.set('NY', 128)
sim.parameter.set_condition('o2', order)
sim.generate_files(generator)
#start the simulation
sim.start_simulation()
```

A.5 Usage Example

Finally a usage example for a grid convergence study using the Simulation API is given. The source code is shown in listing A.4. In the example a Hagen-Poiseuille flow is simulated (see Sec. 4.3.4).
As first step all modules from the API are imported and all relevant constants are defined, for example the Reynolds number is set to $Re = 100$. In the next step a for-loop iterates over the array which contains all resolutions for which the simulation should be executed.
Inside the for-loop the first step is to set a unique simulation path for each resolution. Next, a generator with the pipe-flow settings from the **pycurb.testcase** module is created and the options for the pressure gradient, the initial velocity field and the pipe radius are set.
Finally the simulation object is created. All parameters like resolution and domain size, the order of the finite difference scheme and the direct forcing method are set. The last step is the execution of a simulation.

Listing A.4: 'Grid Convergence Study Example'

```python
import pycurb as pc
import pycurb.testcase as tf
import numpy as np
import os

def main():
    #define simulation parameters
    re = 100.
    pmax, pr = 4./re,  1./re
    lx, ly, order = 2.5, 2.5, 1

    #vary N from 16 to 256 with dN=16
    res = np.linspace(16, 256., 256./16)

    for rs in res:#iterate through resolution array
        #create filepath for each simulation
        var_path = os.path.join(method,'res_%i' % rs)
        sim_path = os.path.join(os.path.dirname(__file__), "data", var_path)

        #create generator with simulation settings
        generator = pc.Generator.from_testcase(tf.cylinder_flow_z)
        generator.add_option('SETV', True)
        generator.add_option('PMAX', pmax)
        generator.add_option('r', 1.)

        #create simulation object and set parameters
        sim = pc.Simulation(sim_path, "parameter.json")
        sim.parameter.set("PRANDTL", pr)
        lz = dx*(lx/rs)
        sim.parameter.set("NX", rs)
        sim.parameter.set("NY", rs)
        sim.parameter.set("LX", lx)
        sim.parameter.set("LY", ly)
        sim.parameter.set("NZ", nz)
        sim.parameter.set("LZ", lz)
        sim.parameter.set_condition('o2', order)
        sim.parameter.set_condition('SET_ZERO', 1)
        sim.generate_files(generator)

        #execute simulation
        sim.start_simulation()

if __name__=='__main__':
    main()
```

B. Additional Figures

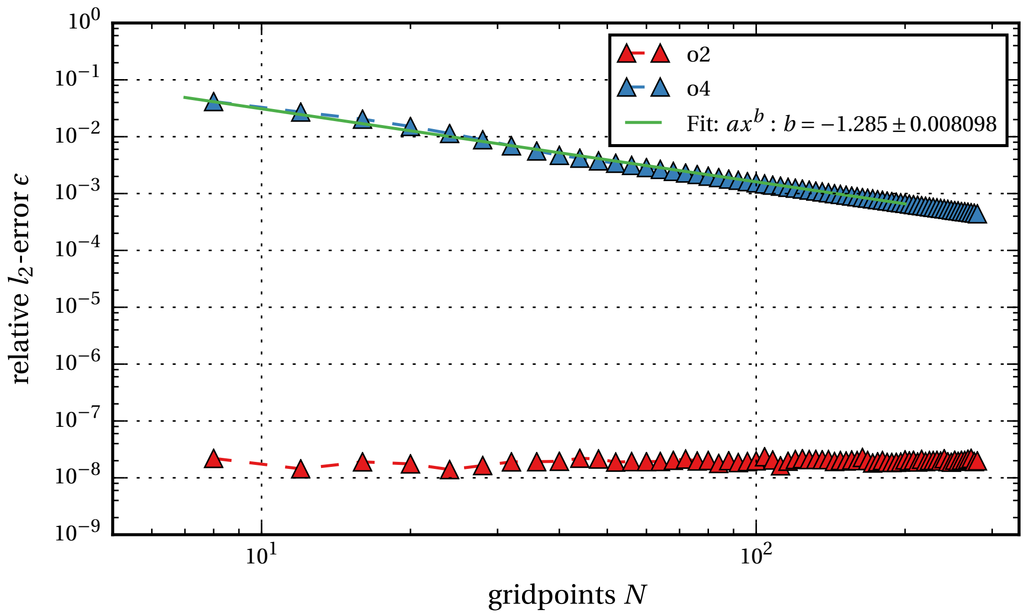


Figure B.1.: Relative l_2-error for the DF-FD2 and DF-FD4 method.

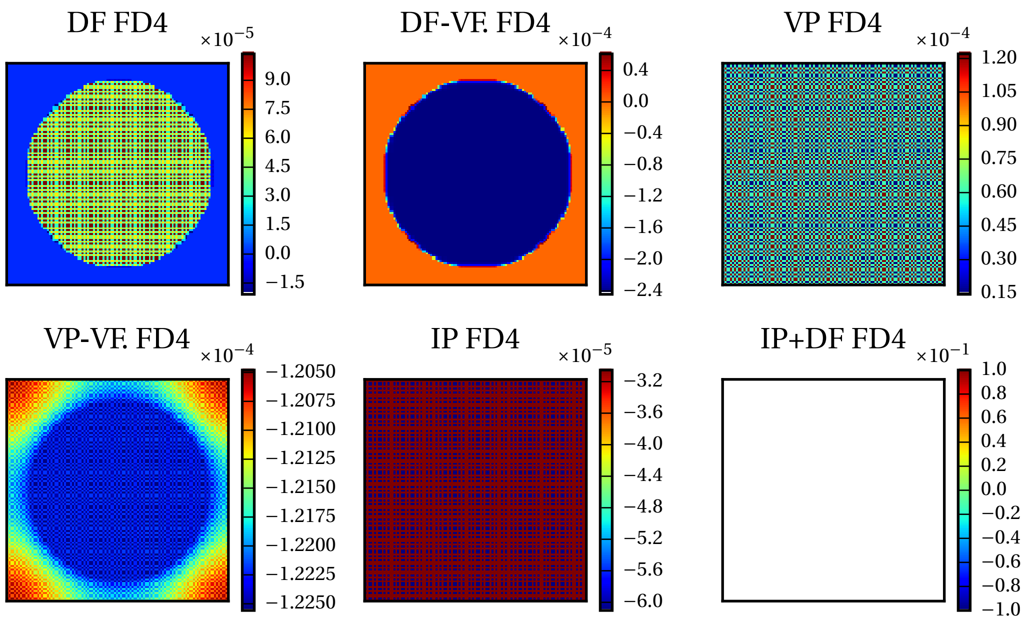


Figure B.2.: Densitiy oscillations for different Immersed Boundary methods with the use of FD4. The IP+DF FD4 method is not stable.

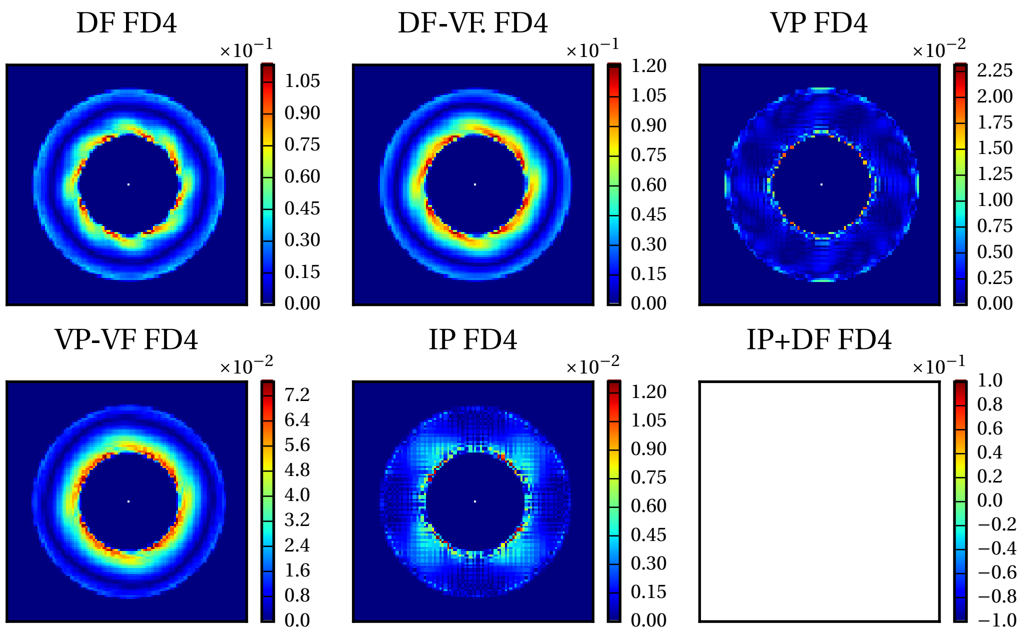


Figure B.3.: Substraction of the numerial velocity profile from the theoretical for all FD4 methods.

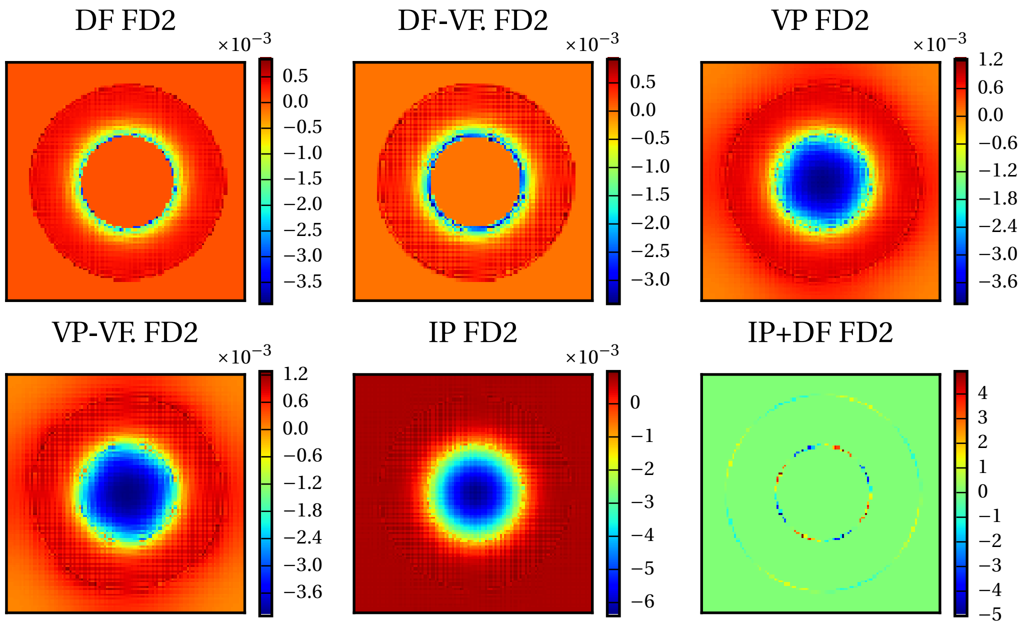


Figure B.4.: Density oscillations for different Immersed Boundary methods of the FD2 methods.

C. Source Code

C.1 Runge-Kutta stability computation

```python
import matplotlib.pyplot as plt
import numpy as np
from math import factorial as fac
from scipy import interpolate

def radial_sort(x, y):
    """Sort line by angle from center (-1, 0)"""
    angle = np.arctan2(y, x + 1.)
    idx = angle.argsort()
    x, y = x[idx], y[idx]
    # Split at opening in line
    dx = np.diff(np.append(x, x[-1]))
    dy = np.diff(np.append(y, y[-1]))
    max_gap = np.abs(np.hypot(dx, dy)).argmax() + 1
    x = np.append(x[max_gap:], x[:max_gap])
    y = np.append(y[max_gap:], y[:max_gap])
    return x, y

def main():
    f, ax = style.newfig(0.8)
    x = np.linspace(-4, 4, 1500)
    X, Y = np.meshgrid(x, x)
    C = X + 1j*Y

    for i in range(1,6)[::-1]:
        b = np.zeros_like(C)
        for j in range(0,i):
            b += C**j/fac(j)
        out = np.where(np.diff((np.abs(b)<=1).astype('float')) != 0)
        pts = np.column_stack((X[out], Y[out]))
        x, y = pts[:, 0], pts[:, 1]
        try:
            x, y = radial_sort_line(x,y)
            x = np.append(x, x[0])
            y = np.append(y, y[0])

            tck, u = interpolate.splprep([x, y], s=0)
            unew = np.linspace(0, 1.0, 100)
            out = interpolate.splev(unew, tck)
            plt.plot(out[1], out[0])
        except:
            print 'Error for %i' % i
    plt.show()

if __name__=='__main__':
    main()
```

Bibliography

[And95] John Anderson. *Computational Fluid Dynamics*. 1st ed. McGraw-Hill Education, Feb. 1995. ISBN: 0070016852.

[Bea70] R. C. Beardsley. “An Experimental Study of Inertial Waves in a Closed Cone”. In: *Studies in Applied* 49.2 (June 1970), pp. 187–196.

[BKV14] Eric Brown-Dymkoski, Nurlybek Kasimov, and Oleg V. Vasilyev. “A characteristic based volume penalization method for general evolution problems applied to compressible viscous flows”. In: *Journal of Computational Physics* 262 (2014), pp. 344–357. ISSN: 00219991.

[Can+88] Claudio Canuto et al. *Spectral methods in fluid dynamics*. New York etc.: Springer-Verlag, 1988, pp. xiv + 557. ISBN: 3-540-17371-4.

[Cho97] Alexandre Joel Chorin. “A Numerical Method for Solving Incompressible Viscous Flow Problems”. In: *Journal of Computational Physics* 135.2 (1997), pp. 118–125. ISSN: 00219991.

[Cla11] Niels Clausen. “Inertialwellen im differentiel rotierenden Kugelspalt”. PhD thesis. Georg-August-Universität Göttingen, 2011, p. 146.

[CWZ15] Cheng Chen, Zhen-Hua Wan, and Wei-Guo Zhang. “Transient growth in Taylor-Couette flow of a Bingham fluid.” In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 91.4 (Apr. 2015), p. 043202. ISSN: 1550-2376.

[DeL12] Anthony DeLeon. “GPU-accelerated Modeling of Microscale Atmospheric Flows over Complex Terrain”. PhD thesis. 2012.

[DSL] Yohann Duguet, Julian F Scott, and Lionel Le Penven. “Oscillatory jets and instabilities in a rotating cylinder”. In: ().

[Fad+00] E.A. Fadlun et al. “Combined Immersed-Boundary Finite-Difference Methods for Three-Dimensional Complex Flow Simulations”. In: *Journal of Computational Physics* 161.1 (June 2000), pp. 35–60. ISSN: 00219991.

[For88] Bengt Fornberg. “Generation of finite difference formulas on arbitrarily spaced grids”. In: *Mathematics of Computation* 51.184 (1988), pp. 699–699. ISSN: 0025-5718.

[FP99] J H Ferziger and M Peric. *Computational Methods for Fluid Dynamics*. Berlin: Springer, 1999.

[Ful59] D. Fultz. “A Note on Overstability and the Elastoid-Inertia Oscillations of Kelvin, Solberg,and Bjerknes”. In: *Journal of Meteorology* 16.2 (Apr. 1959), pp. 199–208. ISSN: 0095-9634.

[GDN98] Michael Griebel, Thomas Dornseifer, and Tilman Neunhoeffer. “Numerical simulation in fluid dynamics: a practical introduction”. In: (Jan. 1998).

[Gor13] T. Gornak. *A goal oriented survey on immersed boundary methods.* Kaiserslautern, 2013.

[Gre69] H. P. Greenspan. “On the lnviscid Theory of Rotating Fluids”. In: *Studies in Applied Mathematics* 48.1 (Mar. 1969), pp. 19–28. ISSN: 00222526.

[Gre90] Harvey P. Greenspan. *The Theory of Rotating Fluids.* 1990. ISBN: 0962699802.

[GS05] Anvar Gilmanov and Fotis Sotiropoulos. “A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies”. In: *Journal of Computational Physics* 207.2 (Aug. 2005), pp. 457–492. ISSN: 00219991.

[GSB03] A. Gilmanov, F. Sotiropoulos, and E. Balaras. “A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids”. In: *Journal of Computational Physics* 191.2 (Nov. 2003), pp. 660–669. ISSN: 00219991.

[KC02] Pijush K. Kundu and Ira M. Cohen. *Fluid mechanics.* Academic Press, 2002, p. 730.

[Lül11] Johannes Lülff. “Statistische Eigenschaften turbulenter Rayleigh-Bénard-Konvektion”. PhD thesis. 2011, p. 142.

[MI05] Rajat Mittal and Gianluca Iaccarino. “Immersed Boundary Methods”. In: *Annu. Rev. Fluid Mech* 37 (2005), pp. 239–61.

[Mof78] Moffatt. *Magnetic Field Generation in Electrically Conducting Fluids.* 1978. ISBN: 0521216400.

[Moh97] J Mohd-Yusof. *Combined Immersed-Boundary/B-spline methods for simulations of flow in complex geometries.* Annual Research Briefs. Center for Turbulence Research, 1997, pp. 317–327.

[NVI15a] NVIDIA Corporation. *Cuda C Best Practices Guide.* NVIDIA Corporation, 2015.

[NVI15b] NVIDIA Corporation. *Cuda C Programming Guide.* NVIDIA Corporation, 2015.

[Pes72] Charles S Peskin. “Flow patterns around heart valves: A numerical method”. In: *Journal of Computational Physics* 10.2 (Oct. 1972), pp. 252–271. ISSN: 00219991.

[Pre+07] William H Press et al. *Numerical Recipes 3rd Edition: The Art of Scientific Computing.* 3rd ed. New York, NY, USA: Cambridge University Press, 2007. ISBN: 0521880688, 9780521880688.

[RC83] C. M. Rhie and W. L. Chow. “Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation”. In: 21.11 (1983).

[Sau+12] Alban Sauret et al. “Fluid flows in a librating cylinder”. In: *Physics of Fluids* 24.2 (Feb. 2012), p. 026603. ISSN: 10706631.

[SF] Christoph Schär and Oliver Fuhrer. *Numerische Methoden in der Umweltphysik*.

[Ste+15] F Stefani et al. “Towards a precession driven dynamo experiment”. In: *Magnetohydrodynamics* 51.2 (2015), pp. 275–284.

[Til00] A Tilgner. “Oscillatory shear layers in source driven flows in an unbounded rotating fluid”. In: *Physics of Fluids* 12.5 (2000).

[Til05] A. Tilgner. “Precession driven dynamos”. In: *Physics of Fluids* 17.3 (2005), p. 034104. ISSN: 10706631.

[Til07] A. Tilgner. “8.07 – Rotational Dynamics of the Core”. In: *Treatise on Geophysics*. 2007, pp. 207–243. ISBN: 9780444527486.

[Til12] a. Tilgner. “Transitions in Rapidly Rotating Convection Driven Dynamos”. In: *Physical Review Letters* 109.24 (Dec. 2012), p. 248501. ISSN: 0031-9007.

[Til99] A. Tilgner. “Driven inertial oscillations in spherical shells”. In: *Physical Review E* 59.2 (Feb. 1999), pp. 1789–1794. ISSN: 1063-651X.

[TM03] Paul A Tipler and Gene Mosca. *Physics for Scientists and Engineers: Extended Version*. Fifth Edit. W. H. Freeman, Aug. 2003. ISBN: 0716743892.

[Tri88] D J Tritton. *Physical Fluid Dynamics (Oxford Science Publications)*. 2nd ed. Oxford University Press, USA, Nov. 1988. ISBN: 0198544936.

[Wil80] J.H Williamson. “Low-storage Runge-Kutta schemes”. In: *Journal of Computational Physics* 35.1 (Mar. 1980), pp. 48–56. ISSN: 00219991.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während meiner Masterarbeit und auch während meines Studiums motiviert und unterstützt haben.

Zuerst gebührt mein Dank Prof. Dr. Andreas Tilgner, für die Bereitstellung des interessanten Themas und die Betreuung während der Masterarbeit. Er war jederzeit eine große Unterstützung und hat mit seinen Anregungen geholfen die vorliegenden Fragenstellungen in dieser Masterarbeit zu lösen.

Ebenfalls möchte ich mich bei Apl. Prof. Dr. Ulrich Parlitz für sein Interesse an meiner Masterarbeit und die Übernahme des Korreferat bedanken.

Weiterhin möchte ich mich bei allen bedanken die mir bei der Korrektur der Arbeit zur Seite standen, insbesondere bei Max Koch und Luzie J. Almenräder. Bedanken möchte ich mich auch bei allen anderen Mitgliedern des Instituts für Geophysik.

Abschließend möchte ich mich bei meiner Familie bedanken, inbesondere bei meinen Eltern die mir dieses Studium ermöglicht und mich immer voll unterstützt haben.

Erklärung

nach §17(9) der Prüfungsordnung für den Bachelor-Studiengang Physik und den Master-Studiengang Physik an der Universität Göttingen:

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe.
Darüberhinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, im Rahmen einer nichtbestandenen Prüfung an dieser oder einer anderen Hochschule eingereicht wurde.

Göttingen, den 16. Juni 2016

(Jonas Ruebsam)