Utrecht University

Master Thesis

A Chebyshev-Galerkin method for inertial waves

Author:
Anna Kruseman

Supervisors:
Prof. Jason Frank
Prof. Leo Maas

December 18, 2015

Abstract

We consider the incompressible viscous Navier-Stokes equations for a rotating fluid. In a tilted domain inertial waves converge to a wave-attractor and potentially influence the mean flow. This motivates the numerical simulation of the solution to the Navier-Stokes equations with a Chebyshev-Galerkin method. A Stokes time-marching scheme involves two second-order partial differential equations and guarantees continuity.

We develop a Chebyshev-Galerkin method to find a weak solution to a system of separable second-order partial differential equations in a three-dimensional rectangular domain with homogeneous boundary conditions. We consider a spectral method with Chebyshev polynomials as basis functions. Weak solutions of a second-order partial differential equation are obtained by solving a linear system of inner-product matrices, which can be solved in terms of expansion coefficients. Boundary conditions can be satisfied with a superposition of Chebyshev polynomials. This approach is extended to higher dimensions in rectangular domains for linear second-order operators.

Chebyshev polynomials are orthogonal with respect to a weighted inner-product. Thus, elegant expressions exist for the mass, first-derivative and stiffness matrices. We show exponential convergence of accuracy with grid-resolution and design fast schemes with linear complexity for the multiplication of the inner-product matrices. We construct diagonal preconditioners for the Laplace-operator in one, two and three dimensions. The preconditioned Poisson-system has a condition number that increases sublinearly with grid-size. This is shown analytically in one dimension and numerically in up to three dimensions.

Contents

1 Introduction

Inertial waves can occur in fluids that are stratified in angular momentum. As angular momentum is a conserved quantity, perturbations on an equilibrium lead to oscillations around this equilibrium state. These oscillations are the inertial waves. Because the earth rotates, inertial waves may form and they may influence ocean dynamics.

Mander and Maas [17] show that for an inviscid fluids the group velocity of inertial waves has a fixed angle with respect to the rotation axis. The angle depends on the monochromatic wave frequency and the rotation rate. Because of the fixed angle of propagation, the angle of a monochromatic wave beam is conserved with respect to the rotation axis upon reflection at a tilted wall. This is contrary to reflection by Snell's law (for e.g. surface waves, light), in which the angle with respect to the reflecting wall is preserved. In some domains inertial wave beams converge to a limit cycle, referred to as wave-attractor. They do so independently of their starting location. Manders and Maas [15], [16] gave experimental evidence of a wave-attractor in a trapezoidal domain. Maas [13] and Nurijanyan et al. [18] derived analytical stream function solutions in a parallelepiped for inviscid flow. Wave-attractors of inertial waves are similar to attractors of internal-gravity waves. For example, they share the property that in the inviscid case, the energy density on the attractor approaches infinity. In viscous fluids, viscosity dampens the velocity, but the energy density in the vicinity of the attractor is nevertheless significantly increased.

Maas [12] investigated effects of inertial waves on the mean flow in a laboratory experiment. A small rectangular box ($20 \times 10 \times 10$ cm) is filled with homogeneous fluid and two pipes at opposing side walls are connected to a pump via tubes (Figure 3.2). The domain has a rigid-lid, thereby excluding surface wave formation. The pump causes a pressure gradient between the entrance and exit tubes, which in turn drives inflow through one tube and outflow through the other. It is observed that the volumetric flow rate at the exit is increased when the box is tilted with respect to the rotation axis. It is hypothesised that in the titled domain, inertial waves converge to a wave-attractor, which traps the energy. The organised behaviour of the wave-attractor might enhance the mean flow. Numerical simulations of this experiment by Rodda [20] show discrepancies with the observations. The simulations suffered from numerical viscosity and additionally the flow was not exactly divergence free.

The aim of this project is to develop a numerical method to simulate the experiment by Maas [12]. We construct a Chebyshev-Galerkin method to

simulate solutions to the full-nonlinear viscous Navier-Stokes equations and the continuity equation for a rotating domain. We mimic the laboratory set-up, taking into account viscosity and inflow/outflow through pipes.

In this report, we expand the Chebyshev-Galerkin method as introduced in Shen et al. [21] and Shen [22]. This is a spectral method, originally designed for space-discretisation in one dimension. Here the method is extended to higher dimensions and more boundary conditions. We derive elegant expressions for the inner-product matrices (discretisation of the gradient and diffusion terms) and design fast schemes for multiplication of these matrices. With a suitable diagonal preconditioner, the condition number of the matrix representing the Laplace operator (diffusion term) can be reduced. We show that the condition number of the preconditioned discrete Laplace operator grows only sublinearly with resolution. Furthermore, testing the accuracy shows that the numerical approximate solution converges exponentially to a unique solution if such solution exists.

To simulate the time-dependent behaviour of the flow, a time-marching scheme is required that guarantees exact mass conservation. We discuss the pressure-correction method and a generalised Stokes method. For a single generalised Stokes time step, the Chebyshev-Galerkin formulation in terms of inner-product matrices is derived for the three-dimensional Navier-Stokes and continuity equations. The Poisson equation is used as the main example in the derivation of the Chebyshev-Galerkin method, because for simulations of the laboratory set-up, we need to solve Poisson-type equations.

The structure of this thesis is as follows. In the next section, the governing equations are briefly introduced. In part I, the governing equations are derived in more detail and emergent phenomena are discussed. We review the laboratory results by Maas [12] and introduce two time-stepping methods mentioned above. Part II presents the Chebyshev-Galerkin method, including the derivation of the inner-product matrices and the extension to higher-dimensions. The accuracy, complexity and condition number are analysed in part III. A discussion of the numerical method and suggestions for further research can be found in part IV.

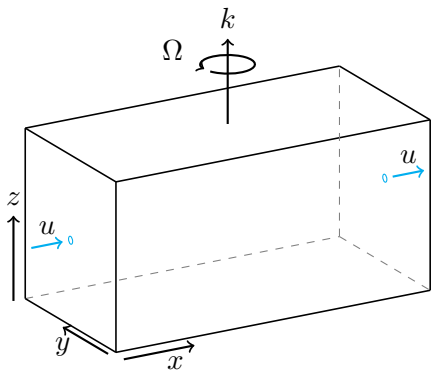

Figure 1.1: Sketch of the set-up. Indicated are the rotation vector k, the rotation rate Ω, and entrance and exit holes (blue) with velocity vector u.

1.1 Governing equations

We aim to approximate the fluid motion in a rotating three-dimensional rectangular domain with inflow and outflow (Figure 1.1). This mimics the set-up used by Maas [12], described in section 3. We align the coordinate-frame with the walls of the box.

The dynamics of a homogeneous incompressible viscous fluid under rotation can be described the Navier-Stokes equations together with a continuity equation. The evolution of the pressure p and the velocity field $\mathbf{u}$ is affected by the prescribed rotation rate Ω and viscosity ν. The governing equations are

$$\underbrace{\frac{\partial \mathbf{u}}{\partial t}}_{\text{evolution}} + \underbrace{(\mathbf{u}\cdot\nabla)\mathbf{u}}_{\text{advection}} = -\underbrace{R_o^{-1}\,\mathbf{k}\times\mathbf{u}}_{\text{rotation}} - \underbrace{\nabla p}_{\substack{\text{pressure}\\ \text{gradient}}} + \underbrace{R_e^{-1}\,\Delta\mathbf{u}}_{\text{diffusion}},$$

$$\underbrace{\nabla\cdot\mathbf{u}}_{\text{divergence}} = 0,$$

where $R_o = U/(2L\Omega)$ is the Rossby number, $R_e = UL/\nu$ is the Reynolds number and $\mathbf{k}$ is the unit rotation vector.

Part I
Physical Background

2 Governing equations

In this section the governing equations for fluids in rotating domains are derived from the momentum and mass balances. Emergent phenomena such as geostrophy and inertial waves are discussed.

2.1 General continuum equations

The mass conservation equation is given by

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

where $\mathbf{u}$ is the velocity vector, and ρ is the density. Additionally, the momentum conservation equation is given by

$$\frac{\partial}{\partial t}(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) = \mathbf{s}, \tag{2.2}$$

where $\mathbf{s}$ is a vector representing the sources and sinks of momentum. We focus on incompressible homogeneous fluids. So, the density ρ is constant. As a result, the mass balance (2.1) reduces to

$$\nabla \cdot \mathbf{u} = 0, \tag{2.3}$$

which is referred to as the continuity equation. So, for incompressibility, the flow must be divergence free. With constant density, the momentum balance (2.2) simplifies to

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = \rho^{-1}\,\mathbf{s}.$$

An ocean does not have a uniform density. Assuming a constant density allows us to focus on the effects of rotation, because in density stratified fluids the appearance of internal-gravity waves complicates the dynamics.

2.1.1 Navier-Stokes equations

Let the sources and sinks of momentum consist of the fictitious Coriolis acceleration due to rotation[1] Ω, the pressure gradient force, gravity and

[1] Traditionally the rotation rate is depicted with the symbol Ω. This same symbol is common notation for a domain. In this report Ω is used for both and its meaning will be clear from context.

the kinematic viscosity. The Coriolis acceleration is explained in Cushman-Roisin and Beckers [3]. Including the sources into the momentum balance (2.2) yields the so-called incompressible Navier-Stokes equations,

$$\underbrace{\frac{\partial \mathbf{u}}{\partial t}}_{\text{variation}} + \underbrace{(\mathbf{u}\cdot\nabla)\mathbf{u}}_{\text{advection}} = -\underbrace{2\boldsymbol{\Omega}\times\mathbf{u}}_{\text{rotation}} - \underbrace{\frac{1}{\rho}\nabla p}_{\substack{\text{pressure}\\ \text{gradient}}} + \underbrace{\nu\Delta\mathbf{u}}_{\text{diffusion}} + \underbrace{\mathbf{g}}_{\text{gravity}}.$$

Here $\mathbf{u}$ is the velocity vector, p is the pressure, $\boldsymbol{\Omega}$ is the rotation vector, ρ is the density, ν is the kinematic viscosity and $\mathbf{g}$ is gravity. Let's define the thermodynamic work per unit mass as $w = p/\rho$. Gravity is exerted by a conservative field ϕ. So, $\mathbf{g} = -\nabla\phi$. Then, we can define reduced work as $p' = w + \phi$. This leads to

$$\partial_t\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -2\boldsymbol{\Omega}\times\mathbf{u} - \nabla p' + \nu\Delta\mathbf{u}. \tag{2.4}$$

The shorthand ∂_t is introduced to denote $\frac{\partial}{\partial t}$, and similar abbreviations can be constructed for spatial partial derivatives. In the rest of this report reduced work is referred to as pressure and its meaning will be clear from context.

2.2 Non-dimensional form

In physics, conservation laws in which the variables have units are called dimensional. To be clear, this has nothing to do with number of spatial dimensions of the domain. Dimensionless variables do not have units and are usually scaled such that the magnitude is order one. With dimensionless variables, simple analysis on the magnitude of different terms is possible. Define

$$\mathbf{x}^* = \frac{\mathbf{x}}{L_0},\ \mathbf{u}^* = \frac{\mathbf{u}}{U_0},\ \nabla^* = L_0\nabla,\ t^* = \frac{U_0}{L_0}t,\ \mathbf{k} = \frac{\boldsymbol{\Omega}}{|\boldsymbol{\Omega}|},\ p^* = \frac{p'}{U_0^2}, \tag{2.5}$$

where L_0 is a characteristic length scale and U_0 is a characteristic velocity. Upon substituting the scaled variables (2.5) into the momentum balance (2.4), one obtains

$$\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{R_o}\mathbf{k}\times\mathbf{u} - \nabla p + \frac{1}{R_e}\Delta\mathbf{u}, \tag{2.6}$$

Here and in the rest of the report the asterisks are omitted as context indicates whether dimensional or dimensionless variables are implied. The Rossby number R_o and the Reynolds number R_e are defined below.

The governing equations are formed by the dimensionless Navier-Stokes equations (2.6) and the continuity condition (2.3), which remains the same after scaling and dropping of the asterisk.

2.2.1 Reynolds number

The Reynolds number R_e is a dimensionless number that characterises different flow regimes. It is defined as the ratio of the inertial forces over the viscous forces

$$R_e = \frac{U_0 L_0}{\nu}.$$

Here U_0 and L_0 are respectively the characteristic velocity and length scale. Furthermore, ν is the kinematic viscosity. Low Reynolds numbers indicate laminar flow and high values occur when the flow is turbulent.

2.2.2 Rossby number

The Rossby number R_o is a dimensionless quantity describing the ratio of the inertial to Coriolis force,

$$R_o = \frac{U_0}{2L_0\Omega},$$

where Ω is the angular frequency of the rotation. A low Rossby number indicates a system in which Coriolis forces dominate and one expects geostrophic balance. When the Rossby number is large, inertial and viscosity forces dominate and turbulence occurs.

2.3 Geostrophic flow

The governing equations (2.6) and (2.3) describe waves and currents. In the next sections we describe phenomena specific to rotating fluids. The laboratory observations [12] were interpreted with these flow characteristics in mind.

Geostrophy or geostrophic flow refers to a flow regime in which the Coriolis and pressure gradient forces dominate. This flow regime is characterised by a low Rossby number R_o. Maintaining only the leading terms, the momentum equations (2.6) then reduce to

$$-R_o^{-1}\,\mathbf{k} \times \mathbf{u} = \nabla p. \tag{2.7}$$

Parallel to the rotation axis, the Coriolis force is zero. Therefore, geostrophy causes an effectively two-dimensional flow. Let $\mathbf{u} = (u, v, w)^T$. Assume rotation is along the z-axis, such that $\mathbf{k} = (0, 0, 1)^T$, then (2.7) becomes

$$R_o^{-1}\, v = \partial_x p, \tag{2.8}$$

$$-R_o^{-1}\, u = \partial_y p, \tag{2.9}$$

$$0 = \partial_z p. \tag{2.10}$$

Observe that the velocity vector is perpendicular to the pressure gradient. For $0 < R_o$, the rotation is counterclockwise. Looking in the direction of the velocity vector, pressure gradient points to the right. In contrast, when $R_o < 0$, the rotation is clockwise and the pressure gradient points to the left.

Substitution of (2.10) into the z-derivatives of (2.8) and (2.9), yields $\partial_z v = 0 = \partial_z u$. So, the geostrophic flow is indeed two-dimensional in the plane perpendicular to the rotation axis $\mathbf{k}$. This result is known as the Taylor-Proudman theorem.

2.4 Inertial waves

Rotating fluids are radially stratified in angular momentum (Figure 3.2). Angular momentum conservation restores perturbations away from the equilibrium stratification. Oscillations around the equilibrium stratification are called inertial waves. These waves occur in the interior of a fluid and propagate with a fixed angle with respect to the rotation axis. When we consider an inviscid fluid with monochromatic waves, e.g. waves with a fixed frequency ω, the angle α of the group velocity with respect to the rotation axis $\mathbf{k}$ is then dictated by the rotation rate Ω and the wave frequency ω as in Manders and Maas [17],

$$\left(\frac{\omega}{2\Omega}\right)^2 = \sin^2 \alpha.$$

As a result, the frequency ω of inertial waves is bounded by $0 \leq \omega \leq 2\Omega$. The angle with respect to the rotation axis is fixed. This means that reflection at a sloping wall (oblique to the rotation axis) can lead to focussing (increase in energy density) of the inertial waves. For certain domain geometries, an inertial wave beam might converge over successive (focusing) reflections to a closed orbit called a wave-attractor. This is similar to the emergence of a wave-attractor for internal-gravity waves (Maas and Lam [14]). An orbit of a wave-attractor is sketched in Figure 2.1. The sketched trajectories converge upon reflection with the sloping walls.

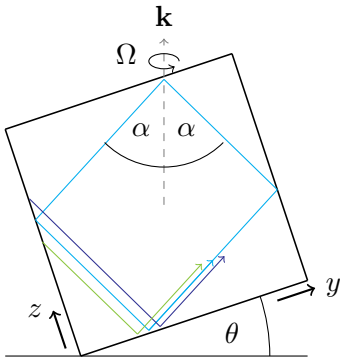

Figure 2.1: Sketch of a wave attractor in a tilted rectangular domain. The positive x-direction points out of the plane. Denoted are the tilt angle θ, the reflection angle α, the rotation axis k, and the rotation rate Ω. The constant angle of propagation results in wave focussing upon reflection.

The existence and shape of an attractor depends on the domain geometry and the direction of the rotation vector $\mathbf{k}$. The depicted rectangular attractor is the simplest closed orbit. Some domains support attractors with more reflections, such as a rectangular figure-eight. When none of the walls is oblique with respect to the rotation vector, the inertial waves do not converge to an attractor.

The synchronised flow of the wave-attractor can store and transport energy. This raises the question of how wave-attractors can influence the mean flow.

2.5 Boundary layers

Viscosity dominates in a small so-called boundary layer, because the velocity must vanish at the boundaries (referred to as the no-slip condition). At the top and bottom walls,t his layer is called the Ekman boundary layer and it has a typical thickness $\delta = \sqrt{\nu/\Omega}$. Under rotation, the velocity changes direction in the boundary layer with respect to the interior in a so-called Ekman-spiral. This may cause vortex stretching and angular momentum transport, aiding the formation of inertial waves.

Near the vertical boundaries (parallel to the rotation axis), viscosity dominates in a so-called Stewartson boundary layer with a thickness proportional to $\sqrt{\delta}$ (Greenspan [7]).

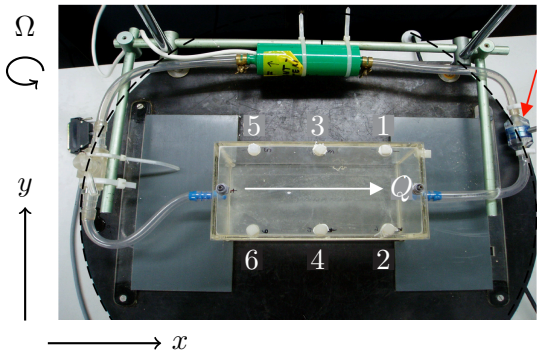

Figure 3.1: Top view of the experimental set-up in the laboratory of the NIOZ. The flow is pumped (pump in green) through the box from left to right. The volume flux Q is measured with a propellor vane (red arrow). The pressure gauges are numbered. The equipment is mounted on a turntable, which rotates counterclockwise. This figure is adapted from Maas [12].

3 Previous laboratory experiments

The aim of this project is to investigate the fluid dynamics in a rotating rectangular domain with inflow and outflow tubes, through which water was pumped, mimicking the set-up at NIOZ used by Maas [12]. In this section, we introduce the laboratory set-up in detail, discuss the expected flow regimes and review the observations.

3.1 Set-up

A top view of the set-up at the NIOZ is shown in Figure 3.1 and a sketch is provided in Figure 3.2. The rectangular box has dimensions $20 \times 10 \times 10$ cm and is located on a rotating table. The box is filled with (homogeneous) degassed tap water. At two sides of the box, a tube with an inner diameter of 8 mm is inserted (blue arrow in Figure 3.2). The tubes are connected to a pump such that water is forced to flow through the box.

The settings for pump rate P in the experiments by Maas [12] are listed in Table 1. Under flat non-rotating conditions, the pump rate can be calibrated linearly to a volumetric flow rate Q, which is also listed in the table. For rotating conditions the pump could be considered as imposing a pressure gradient in the x-direction. Rodda [20] used a prescribed entrance-exit pressure gradient to simulate this set-up with Gerris software. We use prescribed inflow and outflow velocity profiles to drive the flow.

The box is in the centre of a turntable whose motor voltage determines

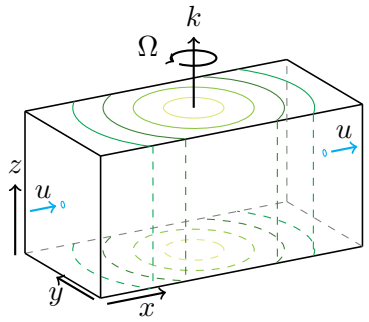

Figure 3.2: Sketch of the set-up. Lines of equal angular momentum are sketched in shades of green. Indicated are the rotation vector k, the rotation rate Ω, and entrance and exit holes (blue) with velocity u in the pump direction (blue arrow).

the rotation rate Ω linearly. The range of rotation rates is listed in Table 1. For each pair of parameter settings, the experiment was run for 200 seconds: 50 seconds for adjustment (spin-up) and 150 seconds for measurements.

The pressure was measured with three pairs of differential pressure gauges that protrude downwards from the lid near the sideway boundaries (Figure 3.1). The volumetric flow rate is measured with a propellor vane behind the exit of the box (red arrow in Figure 3.1).

		min	max	units
pump rate	P	0	7	V
volumetric flow rate	Q	0	5.44	10^{-5} m^3/s
rotation rate	Ω	0	2π	rad/s

Table 1: Settings as in Maas (2007).

3.2 Theoretical expectations

The energy inserted via the pressure gradient is expected to cascade via turbulence to small scale motions and eventually dissipate into heat. The higher the pump rate, the higher the velocity, the more turbulence is generated. We expect turbulence to dominate for high pump rates (high Rossby number). In contrast, geostrophy is expected to dominate for high rotation rates (low Rossby number).

When the box is tilted, inertial waves can converge to a wave-attractor and store wave-energy (section 2.4). This energy is not available for dissipa-

tion. In the geostrophic regime, inertial waves may feed the stored energy back to the mean flow. In a flat domain, inertial waves do not converge to an attractor. It is expected that the outflow is increased in a tilted domain with respect to a flat domain.

3.3 Observations

Figures 3.3 and 3.4 depict the main results of Maas [12]. Figure 3.3 suggests occurrence of different flow regimes. Near-geostrophy is found for low pump rates and high rotation rates. In contrast turbulence dominates at high pump rates and low rotation rates. The division seems to be along the diagonal corresponding to a Rossby number of 1, whose characteristic length scale L_0 is the tube diameter d. The scale d also describes the vertical thickness of the flow from source to sink, because vertical motion is suppressed by rotation. The characteristic velocity U_0 is the sheet velocity Q/dH, where Q is the flow rate $\sim 10^{-5}$ m^3/s, d is the diameter of the tube 8 mm, and H is the width of the box in the y-direction, i.e. 0.1 m.

The outflow for different cases is compared in Figure 3.4, where the flow anomalies are shown for different pump rates P and rotation rates Ω. The results suggest that in the geostrophic regime rotation increases the flow rate (panel A). This unexpected result could be explained by the observation in Van der Lugt [11] that the table does not rotate smoothly but wobbles. Better understood are panels B and C, showing that tilting the box enhances the flow rate, consistent with the conjecture that inertial waves can increase the mean flow.

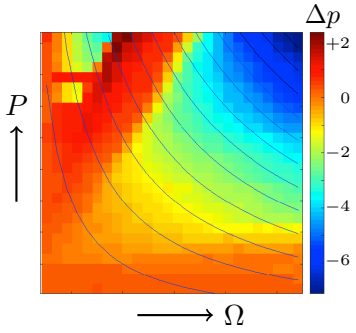

Figure 3.3: Differential pressure $\Delta p = p_5 - p_6$ [N/m^2] between gauges 5 and 6 for pump rates P and rotation rates Ω. The axes are linear between the values in Table 1. The blue hyperbola indicate geostrophic flow, whose velocities are inversely proportional to the rotation rate for a fixed pressure difference. This figure is adapted from Maas [12].

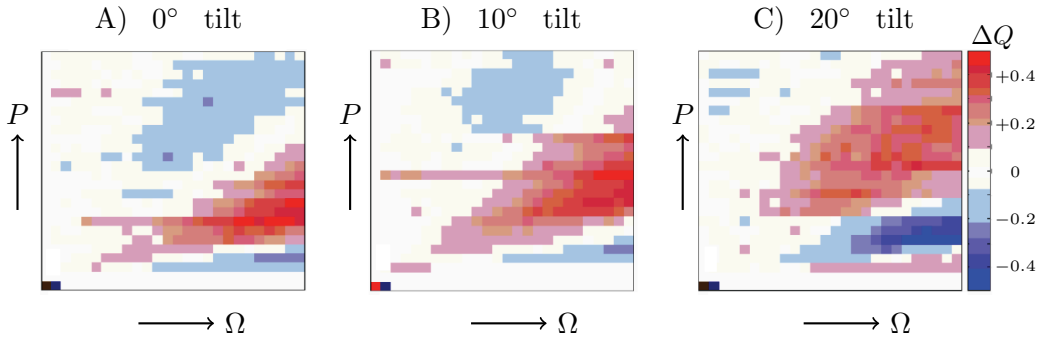

Figure 3.4: Flow rate anomalies ΔQ [L/min] for pump rates P and rotation rates Ω as in Table 1. A) Flow rate relative to stationary (both flat): $\Delta Q = Q_{\text{rot}} - Q_{\text{stat}}$. B) Comparison flat and 10° tilted (both rotating): $\Delta Q = Q_{10} - Q_{\text{flat}}$. C) Flat compared to 20° tilted (both rotating): $\Delta Q = Q_{20} - Q_{\text{flat}}$. This figure is adapted from Maas [12].

4 Numerical Set-up

We intend to examine interior velocity fields with numerical simulations. Numerical simulations for a similar set-up can be found in Jouve and Ogilvie [10] and Rodda [20] The former study is purely two-dimensional. It proves the existence of a wave attractor in tilted rectangles, but excludes mean flow dynamics.

Rodda [20] analysed three-dimensional simulations obtained with Gerris software on settings representing the laboratory experiment. The flow is driven with a pressure gradient between the pipes, mimicking the effect of the pump. The simulations verified increased flow rate in a tilted versus flat domain. Unfortunately, the chosen entrance-exit pressure differences created a higher mean flow than the pump rates in the laboratory, precluding quantitative comparison with laboratory results. Gerris software is based on a finite volume type method. Rodda [20] states that numerical viscosity on top of the kinematic viscosity alters the flow, causing differences between her simulations and the laboratory observations. She additionally observed flow divergence, indicating mass was not exactly conserved.

The time-discretisation of the governing equations forms a system of partial differential equations. We develop a Chebyshev-Galerkin (CG) spectral method to approximate solutions to second-order partial differential equations. A spectral method is not impeded by numerical viscosity. The CG method is applicable to three-dimensional rectangular domains and can account for viscosity.

At the solid walls, we assume no-slip (Dirichlet) boundary conditions on the velocity. In the laboratory the pump causes a pressure difference between the entrance and exit areas. In the CG method, we prescribe a velocity profile at the entrance and exit. To compare with parameter settings in Rodda or Maas, the resulting entrance-exit pressure gradient after simulation could be used as a proxy for the pump-rate.

4.1 Hagen-Poisseuille flow

Assume the velocity profile at the entrance and exit is dominated by the flow in the pipes, which is approximated by incompressible Hagen-Poisseuille flow. The normal (x-direction) velocity u is described in terms of the radial position $r = \sqrt{y^2 + z^2}$ as

$$u(r) = \frac{1}{4\mu} \frac{\Delta p}{\Delta x} (R^2 - r^2),$$

where $\mu = \rho\nu$ is the dynamic viscosity, Δp is the pressure drop in the tube, Δx is the tube length, and R is the tube radius. Integration of the velocity over the entrance gives the volumetric flow rate Q,

$$Q = \frac{\pi}{8\rho\nu} \frac{\Delta p}{\Delta x} R^4.$$

So, one can express the velocity profile as a function of the flow rate Q,

$$u(r) = \frac{2}{\pi} Q\, R^{-4} (R^2 - r^2). \tag{4.1}$$

For prescribed Q the velocity profile at the entrance and exit can be computed with relation (4.1), as displayed in Figure 4.1.

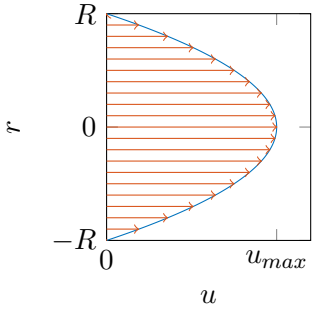

Figure 4.1: Scaled velocity profile at the entrance, where $u_{max} = 2/\pi\, Q\, R^{-2}$.

5 Time-discretisation methods

Simulation of the time-dependent Navier-Stokes equations requires a time-stepping method. We consider a time-discretisation with discrete time-steps of size τ. The local time-derivatives are approximated to map the variables from one time step to the next.

In this section, we introduce a general time-discretisation method for advection-diffusion equations and then discuss two strategies to discretise the Navier-Stokes equations (2.6) constraint by condition (2.3): the pressure-correction method and the generalised Stokes equations.

5.1 IMEX method

The momentum equations (2.6) have advection and diffusion terms. Advection is nonlinear and can only be treated explicitly, while diffusion is stiff and should be treated implicitly to guarantee stability. Therefore, the system is discretised with a so-called implicit-explicit (IMEX) method.

Frank et al. (1996) investigated the stability of several second-order IMEX methods with an A-stable implicit part. Based on their results, we choose the extrapolated backwards differentiation formula (BDF). In Peyret (2002) this method is referred to as Semi-Implicit Adam-Bashforth Backward-Differentiation.

Consider a variable u, whose time-dependent dynamics are governed by a non-stiff operator F and a stiff operator G, as

$$\partial_t u = F(t,u) + G(t,u).$$

Let's consider time steps of size τ and denote $t_n = n\tau$, $u_n = u(t_n)$, $F_n = F(t_n, u_n)$, $G_n = G(t_n, u_n)$. With extrapolated BDF, the evolution can be discretised at time t_{n+1} yielding an equation for u_{n+1}, G_{n+1},

$$\underbrace{\frac{3u_{n+1} - 4u_n + u_{n-1}}{2\tau}}_{\text{BDF}} = \underbrace{2F_n - F_{n-1}}_{\text{extrapolation}} + \underbrace{G_{n+1}}_{\text{operator at } t_{n+1}}$$

The left hand side is derived in Appendix A. This time scheme requires a start-up procedure u_0, u_1.

5.2 Pressure-correction method

The pressure-correction method is a fractional step method, which means that each time step is solved in several steps. A variant of the pressure-correction method is discussed in Guermond et al. [8],[9] and we follow their approach. In short, an intermediate velocity $\tilde{\mathbf{u}}$ is determined from the previous velocity and pressure. Then a pressure-correction is computed to satisfy the continuity conditions. We present a concise version.

Let the time be discretised in steps of size τ and let the superscript t denote the time-dependent variable at the current time step. Similarly superscript $t+1$ indicates the following time step and $t-1$ the previous one. In the first step, we solve the momentum equation with the previous pressure to find the so-called intermediate velocity $\tilde{\mathbf{u}}$

$$\begin{aligned}\frac{1}{2\tau}(3\tilde{\mathbf{u}} - 4\mathbf{u}^t + \mathbf{u}^{t-1}) + R_o^{-1}\,\mathbf{k}\times\tilde{\mathbf{u}} - R_e^{-1}\,\Delta\tilde{\mathbf{u}} \\ = -\nabla p^t - \big(2(\mathbf{u}^t\cdot\nabla)\mathbf{u}^t - (\mathbf{u}^{t-1}\cdot\nabla)\mathbf{u}^{t-1}\big).\end{aligned} \tag{5.1}$$

Let's assume Dirichlet boundary conditions on $\tilde{\mathbf{u}}$. The updated pressure can be composed of the previous pressure and an pressure-correction $\tilde{q}$,

$$p^{t+1} = p^t + \tilde{q}. \tag{5.2}$$

One can approximate the next velocity as the intermediate velocity $\tilde{\mathbf{u}}$ plus the gradient of the pressure-correction $\tilde{q}$ (whose contribution was lacking in (5.1)),

$$\mathbf{u}^{t+1} = \tilde{\mathbf{u}} - (2\tau/3)\,\nabla\tilde{q}. \tag{5.3}$$

The next velocity should be divergence free, $\nabla\cdot\mathbf{u}^{t+1} = 0$. Taking the divergence of (5.3) yields that $\tilde{q}$ must satisfy

$$\Delta\tilde{q} = (3/2\tau)\,\nabla\cdot\tilde{\mathbf{u}}, \tag{5.4}$$

where $\Delta = \nabla^2$ is the Laplace operator. To solve for the pressure correction in relation (5.4) uniquely, one needs boundary conditions on $\tilde{q}$. Since $\mathbf{u}^{t+1}$ and $\tilde{u}$ have Dirichlet boundary conditions, relation (5.3) gives Neumann boundary conditions on $\tilde{q}$. The pressure-correction equation (5.4) with Neumann boundary conditions, satisfies the so-called compatibility condition (6.8), because $\tilde{\mathbf{u}}$ has Dirichlet boundary conditions (section 6.3).

For each time step in the pressure-correction scheme, one first solves (5.1), then (5.4) and lastly updates the velocity and the pressure with (5.2)-(5.3).

5.3 Stokes equations

An alternative time-discretisation of the Navier-Stokes equations leads to a set of (time-independent) Stokes equations at each time step,

$$\begin{aligned} \alpha\,\mathbf{u}^{t+1} + R_o^{-1}\,\mathbf{k}\times\mathbf{u}^{t+1} - R_e^{-1}\,\Delta\mathbf{u}^{t+1} + \nabla p^{t+1} &= \mathbf{f}^t, \\ \nabla\cdot\mathbf{u}^{t+1} &= 0, \end{aligned} \tag{5.5}$$

where $\mathbf{f}^t = \tau^{-1}(2\mathbf{u}^t - 1/2\,\mathbf{u}^{t-1}) - \big(2(\mathbf{u}^t\cdot\nabla)\mathbf{u}^t) - (\mathbf{u}^{t-1}\cdot\nabla)\mathbf{u}^{t-1})\big)$, and $\alpha = (3/2)\,\tau^{-1}$. Define operators A, B and C as

$$\begin{aligned} A\mathbf{u} =& \alpha\,\mathbf{u} + R_o^{-1}\,\mathbf{k}\times\mathbf{u} - R_e^{-1}\,\Delta\mathbf{u}, \\ Bp =& \nabla p, \\ C\mathbf{u} =& \nabla\cdot\mathbf{u}. \end{aligned}$$

To find weak solutions, the operators A, B, C are expanded as a sum of products of mass, stiffness and first-derivative matrices (M, S, K) in chapter 11. The resulting A is invertible. With A, B, C so-defined, the system (5.5) becomes

$$A\mathbf{u}^{t+1} + B\,p^{t+1} = \mathbf{f}^t, \tag{5.6}$$

$$C\mathbf{u}^{t+1} = 0. \tag{5.7}$$

Pre-multiplying the first equation with CA^{-1} and substituting the second, yields

$$CA^{-1}B\,p^{t+1} = CA^{-1}\mathbf{f}^t. \tag{5.8}$$

Solutions to (5.8) yield a unique pressure p^{t+1} after imposing the integral constraint. So p^{t+1} does not require boundary conditions. With the so-obtained p^{t+1}, we can solve (5.6) for $\mathbf{u}^{t+1}$, as

$$A\mathbf{u}^{t+1} = \mathbf{f}^t - B\,p^{t+1}. \tag{5.9}$$

5.4 Lifting of boundary conditions

As discussed in section 6.3, the Chebyshev-Galerkin method requires homogeneous boundary conditions. When an application does not have homogeneous boundary conditions, one can find a corresponding problem that does. The procedure is referred to as lifting of boundary conditions (Appendix B). We use this to remove the inflow and outflow boundary conditions.

Consider the domain $[-2,2] \times [-1,1]^2$ with boundary Γ. Define the entrance, Γ_1, and exit, Γ_2. Then the solid walls are $\Gamma_0 = \Gamma/(\Gamma_1 \cup \Gamma_2)$, with

$$\begin{aligned}\Gamma_1 =&\{(x,y,z) \in \Gamma | x = -2, y^2 + z^2 < R\},\\ \Gamma_2 =&\{(x,y,z) \in \Gamma | x = 2, y^2 + z^2 < R\}.\end{aligned}$$

Let the velocity $\mathbf{u} = (u,v,w)^T$ at the pipes be described by Hagen-Poisseuille flow (4.1) yielding the boundary condition

$$\begin{cases} u = \frac{2}{\pi} Q\, R^{-4}\big(R^2 - (y^2+z^2)\big), & \text{on } \Gamma_1 \text{ and } \Gamma_2,\\ u = 0, & \text{on } \Gamma_0,\\ v = 0 = w, & \text{on } \Gamma. \end{cases} \tag{5.10}$$

These boundary conditions are satisfied by $\acute{\mathbf{u}} = (\acute{u}, \acute{v}, \acute{w})^T$, given by $\acute{v} = 0 = \acute{w}$ and

$$\acute{u} = \begin{cases} \frac{2}{\pi} Q\, R^{-4}\big(R^2 - (y^2+z^2)\big), & \text{for all } y^2 + z^2 \le R^2,\\ 0, & \text{otherwise,} \end{cases}$$

Because $\acute{u}$ is uniform in the x-direction and $\acute{v}$, $\acute{w}$ are zero, the velocity field $\acute{\mathbf{u}}$ is divergence free. We can decompose $\mathbf{u}^{t+1} = \bar{\mathbf{u}} + \acute{\mathbf{u}}$, where $\bar{\mathbf{u}}$ should satisfy Dirichlet boundary conditions. The Stokes equations (5.5) with boundary conditions (5.10) can be rewritten in terms of $\bar{\mathbf{u}}$ as

$$\begin{aligned}\alpha\, \bar{\mathbf{u}} + R_o^{-1}\, \mathbf{k} \times \bar{\mathbf{u}} - R_e^{-1}\, \Delta \bar{\mathbf{u}} + \nabla p^{t+1} &= \bar{\mathbf{f}},\\ \nabla \cdot \bar{\mathbf{u}} &= 0,\end{aligned}$$

where

$$\bar{\mathbf{f}} = \mathbf{f}^t - \left(\alpha\, \acute{\mathbf{u}} + R_o^{-1}\, \mathbf{k} \times \acute{\mathbf{u}} - R_e^{-1}\, \Delta \acute{\mathbf{u}}\right).$$

At the end of each time step, $\acute{\mathbf{u}}$ is added to the obtained $\bar{\mathbf{u}}$ to construct the velocity at the new time step $\mathbf{u}^{t+1}$.

Part II
Chebyshev-Galerkin Method

For the rotating box experiment, we need a method that solves Poisson equation efficiently, because most of the simulation time is used solving Poisson-type equations. In this part we develop a Chebyshev-Galerkin method to solve second-order partial differential equations, such as the Poisson equation.

6 Spectral-Galerkin Method

Consider a second-order linear partial differential equation (PDE) in a connected bounded domain $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$,

$$\mathcal{L}u(x) = f(x), \quad x \in \Omega, \tag{6.1}$$

where $\mathcal{L}$ is a linear operator. We want to solve this for $u(x)$ under forcing $f(x)$. For well-posedness, we require boundary conditions of the form

$$\mathcal{B}u(x) = g(x), \quad x \in \Gamma := \partial\Omega, \tag{6.2}$$

where $\mathcal{B}$ is an at most first-order linear operator and $g(x)$ is the boundary value or flux. A simple example would be the Poisson equation,

$$\Delta u = f. \tag{6.3}$$

where $\Delta = \nabla^2$ is the Laplace operator. The Poisson equation serves as the main example in the next sections.

6.1 Spectral discretisation

Numerical methods need spatial discretisation to compute derivatives. Consider a function $f(x) \in \boldsymbol{X}$, where $x \in \Omega$ and $\boldsymbol{X}$ is an appropriate solution space. In many cases, the solution $f(x)$ can be approximated by a superposition of $N + 1$ basis functions $\phi_k(x)$ as

$$f \approx f_N(x) = \sum_{k=0}^{N} \hat{f}_k \phi_k(x). \tag{6.4}$$

Let $\boldsymbol{X}_N$ be a finite-dimensional approximation of $\boldsymbol{X}$ with $\dim(\boldsymbol{X}_N) = N + 1$. Then we can choose ϕ_n such that $\{\phi_k\}_0^N$ spans $\boldsymbol{X}_N$. A function can thus

be represented by its expansion coefficients $\hat{f}_k$. In the limit $N \to \infty$ the approximation (6.4) becomes an equality.

Numerical methods in which spatial derivatives only consider the values at neighbouring grid points are referred to as local methods. Examples are finite difference, finite volume or finite element methods.

Spectral methods distinguish themselves from local methods by using globally smooth basis functions which have continuous derivatives over the domain. The basis functions are frequently chosen to be orthogonal polynomials. The most common choice of basis functions in periodic or infinite domains are Fourier series. The main advantage is that these global modes provide superior accuracy with resolution. In return, spectral methods allow less domain flexibility.

6.2 Weak formulation

Considering the second-order problem (6.1), the boundary conditions (6.2) can be incorporated into the solution space of u as

$$\boldsymbol{X} = \{u \in \mathcal{C}^2(\Omega) \big| \mathcal{B}u(x) = g(x) \text{ on } \Gamma\}.$$

Solutions to (6.1) can be approximated with a weighted residual method. The residual is defined by

$$\boldsymbol{R}(x) := \mathcal{L}u(x) - f(x), \quad x \in \Omega.$$

For any solution u, the residual should be zero. When the residual is zero multiplying with any function v and taking the integral with respect to some weight function, must also be zero. As a result, u is a weak solution to (6.1) when it holds that

$$(\boldsymbol{R}, v)_\omega = \int_\Omega \mathbf{R}(x)v(x)\omega(x) \ \mathrm{d}x = 0, \quad \text{for all } v \in \boldsymbol{X},$$

where ω is a positive weight function.[2]

Approximating u with (6.4) under boundary conditions (6.2) gives

$$u \approx u_N(x) = \sum_{k=0}^{N} \hat{u}_k \phi_k(x),$$

We choose v from a set of so-called test functions ψ_j also spanning $\boldsymbol{X}_N$. So,

$$(\boldsymbol{R}_N, \psi_j)_\omega = \int_\Omega \mathbf{R}_N \psi_j \omega \ \mathrm{d}x = 0, \quad \text{for all } j \in \{0, \ldots, N\}.$$

[2]The weight function ω is usually chosen such that the basis functions are orthogonal.

When the test functions ψ_j are the same as the basis functions ϕ_j, the method is called a Galerkin method. A Galerkin approach preserves symmetry properties of operators. The advantage of a spectral method is that the equations can be solved in terms of expansion coefficients $\hat{u}_k$.

For weak solutions, relation (6.2) must holds for all j, yielding a linear system in terms of expansion coefficients,

$$L\mathbf{u} = \mathbf{f}, \tag{6.5}$$

where

$$\begin{aligned} \mathbf{u} &= (\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{N_u})^T, & \mathrm{f}_j &= (f, \phi_j)_\omega, \quad \mathbf{f} = (\mathrm{f}_0, \mathrm{f}_1, \ldots, \mathrm{f}_{N-2})^T, \\ l_{jk} &= (\mathcal{L}\phi_k, \phi_j)_\omega, & L &= (l_{jk})_{j,k=0,\ldots,N}. \end{aligned}$$

6.3 Boundary conditions

Second order partial differential equations usually need boundary conditions to obtain a unique solution. One needs a condition of the form

$$a(x)u(x) + b(x)\frac{\partial}{\partial n}u = g(x), \qquad x \in \Gamma, \tag{6.6}$$

where n is the outwards normal vector at the boundary Γ and a, b, g are smooth, continuously differentiable functions. The directional derivative is defined as $\partial/(\partial\, n) = n \cdot \nabla$. A polynomial of order N that must satisfy boundary conditions has only $\dim(N-1)$ degrees of freedom.

When g is zero over all Γ, the boundary conditions are called homogeneous,

$$a(x)u(x) + b(x)\frac{\partial}{\partial n}u = 0, \qquad x \in \Gamma. \tag{6.7}$$

With homogeneous boundary conditions, each of the basis functions ϕ_n must satisfy the boundary conditions (6.7). Therefore, a solution built from these basis functions also satisfies the homogeneous boundary conditions. This does not work for non-homogeneous boundary conditions. As a result, we require homogeneous boundary conditions.

Problems with non-homogeneous boundary conditions can be rewritten as problems with homogeneous boundary conditions. This is explained in Appendix B. From here on boundary conditions are always homogeneous unless specified otherwise.

The special case $b(x) = 0$ is called the Dirichlet boundary condition. The case $a(x) = 0$ is referred to as the Neumann boundary condition. Neumann boundary conditions only fix the solution of the Poisson equation up to an

additive constant. To find a unique solution, one can additionally demand the so-called integral constraint (6.9). Besides, not every forcing f allows solutions u with Neumann boundary conditions. When we integrate the Poisson equation (6.3) over the domain, and apply Gauss's divergence theorem and substitute homogeneous Neumann boundary conditions, we obtain the so-called *compatibility condition*,

$$0 = \int_\Gamma \frac{\partial u}{\partial n} = \int_\Omega \nabla^2 u = \int_\Omega f. \tag{6.8}$$

For solution to exist for a Poisson equation with Neumann boundary conditions, the forcing f needs to satisfy the compatibility condition.

6.3.1 Integral constraint

The Navier-Stokes equations include the gradient of the pressure p. When p is a solution, then so is $p+c_0$ for any constant c_0. Uniqueness of the solution requires an extra condition. We consider the s-called *integral constraint* condition,

$$\int_\Omega p(x) \ \mathrm{d}x = 0, \tag{6.9}$$

As a result of this condition, p has one less degree of freedom.

Interlude on notation

In the next sections, the integral of the product of two functions with respect to a weight function is referred to as the weighted inner product,

$$(f, g)_\omega := \int_\Omega f(x)g(x)\omega(x) \ \mathrm{d}x. \tag{6.10}$$

Its discrete variant is referred to as the discrete weighted inner product,

$$\langle f, g\rangle_\omega := \sum_{j=0}^{N} f(x_j)g(x_j)\omega_j. \tag{6.11}$$

7 Chebyshev

The choice of basis functions and weights determines the complexity, condition number and accuracy of the linear system (6.5). For a confined domain with non-periodic boundary conditions, the Chebyshev-Gauss-Lobatto quadrature has a favourable node (evaluation point) positioning, because is has a high concentration of nodes near the boundaries, thereby minimising of the Gibbs phenomenon.[3]

Additionally, there is a fast discrete transform between the function values at the particular choice of nodes $f(x_j)$ and the Chebyshev coefficients $\hat{f}_k$. This transform takes $O(N \log_2 N)$ computations with the aid of a fast Fourier transform (FFT), see Appendix C.3.

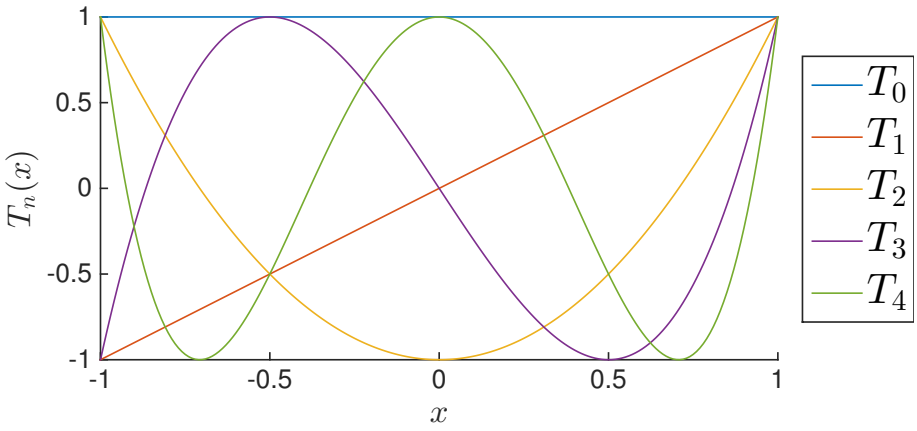

Figure 7.1: The first five Chebyshev polynomials $T_n(x)$

7.1 Properties

The one-dimensional Chebyshev polynomials are given by

$$T_n(x) = \cos(n\theta), \quad \theta = \arccos(x), \quad x \in [-1, 1], \quad 0 \leq n. \tag{7.1}$$

[3]The boundary conditions can be perceived as a discontinuity. The Fourier series of a function with a discontinuity generates large oscillations of the highest frequency near the discontinuity. This is called the Gibbs phenomenon. Chebyshev polynomials do not suffer from the Gibbs phenomenon, because their high node density near the boundaries. The highest frequency of a finite Chebyshev series cannot oscillate on the small scale between the nodes closest to the boundary.

They satisfy the following recursion relation, which shows that T_n is an nth degree polynomial,

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),$$

with $T_0 = 1$ and $T_1 = x$. The Chebyshev polynomials are orthogonal in the weighted inner product $(\ ,\ )_\omega$ with $\omega(x) = (1-x^2)^{-1/2}$,

$$(T_n(x), T_m(x))_\omega = \int_{-1}^{1} T_n(x)T_m(x)\,\omega(x)\ \mathrm{d}x = \frac{c_n\pi}{2}\delta_{nm}, \tag{7.2}$$

where $c_0 = 2$ and $c_n = 1$ for $n \geq 1$. The following properties can be derived (Shen et al. [21]),

$$T_n(\pm 1) = (\pm 1)^n, \tag{7.3}$$

$$T_n'(\pm 1) = (\pm 1)^{n-1}\, n^2, \tag{7.4}$$

where $T_n'(x)$ denotes the spatial derivative $\partial_x T_n(x)$. Additionally, Shen et al. [21] gives

$$T_n'(x) = 2n \sum_{\substack{k=0 \\ k+n \text{ odd}}}^{n-1} \frac{1}{c_k} T_k(x), \tag{7.5}$$

$$T_n''(x) = \sum_{\substack{k=0 \\ k+n \text{ even}}}^{n-2} \frac{1}{c_k} n(n^2-k^2) T_k(x). \tag{7.6}$$

7.2 Quadrature

Consider the Chebyshev-Gauss-Lobatto (CGL) nodes and weights

$$x_j = -\cos\left(\frac{j\pi}{N}\right), \quad \omega_j = \frac{\pi}{\tilde{c}_j N}, \quad \text{for } j = 0, 1, \ldots, N, \tag{7.7}$$

where $\tilde{c}_0 = \tilde{c}_N = 2$ and $\tilde{c}_j = 1$ otherwise. With these nodes x_j and weights ω_j, the quadrature is exact for polynomials p of at most degree $2N-1$, as proved in Shen et al. [21],

$$\int_{-1}^{1} p(x) \frac{1}{\sqrt{1-x^2}} dx = \sum_{j=0}^{N} p(x_j)\omega_j, \quad \forall\, p \in P_{2N-1}. \tag{7.8}$$

Moreover, direct substitution of (7.7) into (7.1) gives,

$$T_n(x_j) = (-1)^n \cos\left(\frac{nj\pi}{N}\right). \tag{7.9}$$

The derivation is straightforward (Appendix C.1). The factor $(-1)^n$ is sometimes (mistakenly) missing in literature. The discrete inner product of T_n with itself is given by integral (7.2) for all $n < N$. For the highest order polynomial ($n = N$), the discrete inner product is

$$\langle T_N, T_N \rangle_\omega = \frac{\pi}{N} \sum_{j=0}^{N} \frac{1}{\tilde{c}_j} \cos^2(j\pi) = \pi.$$

As a result, the discrete inner product (6.11) of T_n and T_m can be given as

$$\langle T_n, T_m \rangle_\omega = \frac{\tilde{c}_n \pi}{2} \delta_{mn}. \tag{7.10}$$

with $\tilde{c}_0 = 2 = \tilde{c}_N$ and else $\tilde{c}_j = 1$.

7.3 Transform

Consider the approximation of u with Chebyshev polynomials as basis functions,

$$u_N(x) = \sum_{n=0}^{N} \hat{u}_n T_n(x). \tag{7.11}$$

The expansion coefficients can be determined via the *forward discrete Chebyshev transform* (see Appendix C.3),

$$\hat{u}_n = \frac{2}{\tilde{c}_n N} \sum_{j=0}^{N} \frac{1}{\tilde{c}_j} u(x_j) T_n(x_j). \tag{7.12}$$

The function values at the quadrature nodes can be retrieved from the coefficients using the *backward discrete Chebyshev transform*,

$$u(x_j) = \sum_{n=0}^{N} \hat{u}_n T_n(x_j) = \sum_{n=0}^{N} \hat{u}_n (-1)^n \cos\left(\frac{nj\pi}{N}\right). \tag{7.13}$$

These transforms can be computed in $O(N \log N)$ computations using FFT as explained in Appendix C.3.

7.4 Basis functions with boundary conditions

The basis functions must span the solution space $\boldsymbol{X}_N$. For homogeneous boundary conditions of the form (6.7), there exists a unique set of $\{a_n, b_n\}$, such that

$$\phi_n(x) = T_n(x) + a_n T_{n+1}(x) + b_n T_{n+2}(x) \tag{7.14}$$

satisfies boundary conditions (6.7) for every $n \geq 0$. The set of basis functions $\{\phi_n\}_0^{N-2}$ spans $\boldsymbol{X}_N$. This is Lemma 4.3 in Shen et al. [21].

7.4.1 Chebyshev-Dirichlet polynomial

Consider the special case of Dirichlet boundary conditions

$$u(x) = 0, \quad x \in \Gamma. \tag{7.15}$$

Define the Chebyshev-Dirichlet (CD) polynomial, as

$$\phi_n(x) = T_n(x) - T_{n+2}(x). \tag{7.16}$$

The basis functions span the solution space. The first five basis functions are depicted in Figure 7.2. It follows from property (7.3) that these polynomials fulfil Dirichlet boundary conditions

$$\phi_n(\pm 1) = 0, \quad \text{for all } n.$$

Lets approximate $u_N(x)$ with Dirichlet boundary conditions,

$$u_N(x) = \sum_{n=0}^{N-2} \tilde{u}_n \phi_n(x), \tag{7.17}$$

where $\tilde{u}_n$ are referred to as the CD coefficients. As described in section 6.3, only $N-1$ basis functions $\{\phi_n\}_0^{N-2}$ are needed to build Nth order polynomials.

The expansion coefficients can also be computed in $O(N \log N)$ computations, using recursion relations with respect to the Chebyshev coefficients $\hat{u}$ (Appendix C.4).

7.4.2 Chebyshev-Neumann polynomial

Similarly, define

$$\phi_n(x) = T_n(x) - \frac{n^2}{(n+2)^2} T_{n+2}(x), \tag{7.18}$$

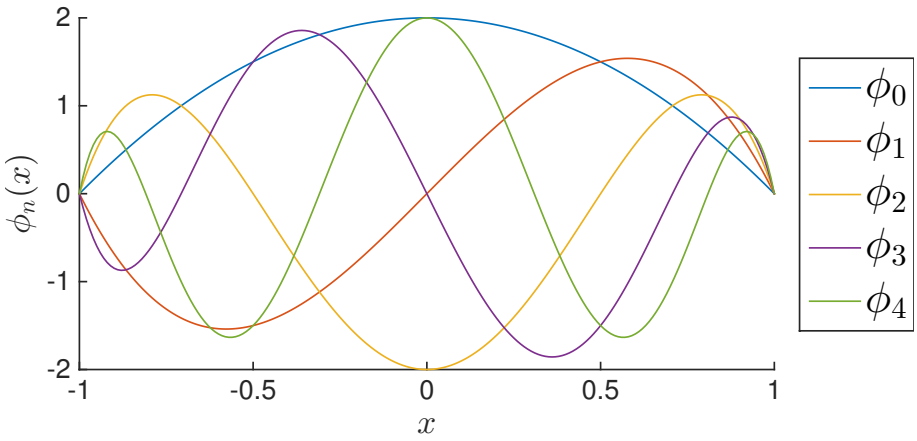

Figure 7.2: The first five Chebyshev-Dirichlet basis functions $\phi_n(x)$as in (7.16).

as the Chebyshev-Neumann (CN) polynomial (see Figure 7.3). It can easily be derived from property (7.4) that the CN polynomial satisfies the Neumann boundary conditions

$$\partial_x \phi_n(\pm 1) = 0, \quad \text{for all } n.$$

Again an Nth order function with Neumann boundary conditions can be represented in $N-1$ basis functions $\{\phi_n\}_0^{N-2}$. Moreover, recursion relations relate the CN coefficients $\tilde{u}_n$ and the Chebyshev coefficients $\hat{u}_n$ (Appendix C.4).

Not every forcing f admits solutions that satisfy Neumann boundary conditions. (section 6.3).

7.4.3 Zero integral polynomial

Lets denote

$$\mu_n = \frac{1}{2}\int_{-1}^{1} T_n(x) = \begin{cases} 1/(1-n^2), & n \text{ is even}, \\ 0, & n \text{ is odd}. \end{cases} \tag{7.19}$$

Then, the basis functions

$$\psi_n = T_n - \frac{\mu_n}{\mu_{n+2}} T_{n+2}, \tag{7.20}$$

satisfy the integral constraint (6.9). An Nth order function p with an integral constraint can be represented with N basis functions $\{\psi_n\}_0^{N-1}$, losing one degree of freedom.

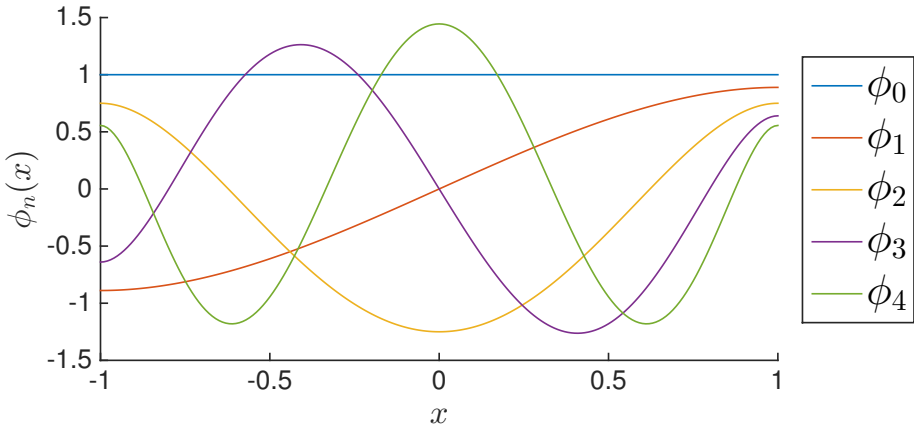

Figure 7.3: The first five Chebyshev-Neumann basis functions $\phi_n(x)$ as in (7.18).

8 Inner products

The linear system (6.5) for the expansion coefficients requires inner product matrices. In this chapter we derive expressions for the elements of the inner product matrices.

8.1 Chebyshev

First we consider Chebyshev polynomials as basis functions. Define the mass matrix M, the first-derivative matrix K, and the stiffness matrix S,

$$M = (m_{mn})_{0\le m,n\le N}, \qquad m_{mn} = (T_n(x), T_m(x))_\omega, \tag{8.1}$$
$$K = (k_{mn})_{0\le m,n\le N}, \qquad k_{mn} = (T_n'(x), T_m(x))_\omega, \tag{8.2}$$
$$S = (s_{mn})_{0\le m,n\le N}, \qquad s_{mn} = (T_n''(x), T_m(x))_\omega. \tag{8.3}$$

The weighted inner products defined in (6.10) can be evaluated with orthogonality (7.2) and the relations for derivatives (7.5) and (7.6). A derivation of the following expressions is detailed in Appendix D.1.

$$m_{mn} = \frac{c_n \pi}{2} \delta_{mn}, \tag{8.4}$$
$$k_{mn} = \begin{cases} \pi n & n = m+1, m+3, m+5, \dots \\ 0 & \text{otherwise,} \end{cases} \tag{8.5}$$
$$s_{mn} = \begin{cases} \frac{\pi}{2} n(n^2 - m^2) & n = m+2, m+4, m+6, \dots \\ 0 & \text{otherwise.} \end{cases} \tag{8.6}$$

8.2 Superpositions

For basis functions ϕ_m (7.14) that satisfy some homogeneous boundary condition (6.7), denote

$$M^* = (m^*_{mn})_{0\le m,n\le N-2}, \qquad m^*_{mn} = (\phi_n(x), \phi_m(x))_\omega, \tag{8.7}$$
$$K^* = (k^*_{mn})_{0\le m,n\le N-2}, \qquad k^*_{mn} = (\phi'_n(x), \phi_m(x))_\omega, \tag{8.8}$$
$$S^* = (s^*_{mn})_{0\le m,n\le N-2}, \qquad s^*_{mn} = (\phi''_n(x), \phi_m(x))_\omega, \tag{8.9}$$
$$R^* = (r^*_{mn})_{0\le m\le N-2,\, 0\le n\le N}, \qquad r^*_{mn} = (T_n(x), \phi_m(x))_\omega, \tag{8.10}$$
$$G^* = (g^*_{mn})_{0\le m,n\le N-2}, \qquad g^*_{mn} = (T'_n(x), \phi_m(x))_\omega. \tag{8.11}$$

Be aware that the asterisk does not indicate complex conjugate. We refer to R as the transformation matrix. The entries can be expressed in terms of the inner products of Chebyshev polynomials m_{mn} (8.1), k_{mn} (8.2) and s_{mn} (8.3). These expressions are given in Appendix D.2.

In the special cases of Dirichlet and Neumann boundary conditions the expressions of S^*, K^* and M^* simplify.

8.2.1 Dirichlet

The derivation of these expression is in Appendix D.3.

$$m^D_{mn} = m^D_{nm} = \begin{cases} \frac{(c_m+1)\pi}{2}, & n = m, \\ -\frac{\pi}{2}, & n = m-2 \text{ or } n = m+2, \\ 0, & \text{otherwise.} \end{cases} \tag{8.12}$$

$$k^D_{mn} = \begin{cases} -(m+1)\pi, & n = m-1, \\ (m+1)\pi, & n = m+1, \\ 0, & \text{otherwise.} \end{cases} \tag{8.13}$$

$$s^D_{mn} = \begin{cases} -2\pi(m+1)(m+2), & n = m, \\ -4\pi(m+1), & n = m+2, m+4, \ldots, \\ 0, & \text{otherwise.} \end{cases} \tag{8.14}$$

$$r^D_{mn} = \begin{cases} \frac{c_m\pi}{2}, & n = m, \\ -\frac{\pi}{2}, & n = m+2, \\ 0, & \text{otherwise.} \end{cases} \tag{8.15}$$

Here the superscript D refers to Dirichlet boundary conditions.

8.2.2 Neumann

For derivations see Appendix D.4. Let $\beta_m = \left(\frac{m}{m+2}\right)^2$,

$$m^N_{mn} = m^N_{nm} = \begin{cases} \frac{\pi}{2}\left(c_m + \beta_m^2\right), & n = m, \\ -\frac{\pi}{2}\beta_m, & n = m+2, \\ -\frac{\pi}{2}\beta_n, & n = m-2, \\ 0, & \text{otherwise.} \end{cases} \tag{8.16}$$

$$k^N_{mn} = \begin{cases} -\frac{\pi n^2}{n+2}, & n = m-1, \\ \frac{\pi n}{n+2}\left(2 - n\beta_m\right), & n = m+1, \\ \frac{2\pi n}{n+2}\left(1 - \beta_m\right), & n = m+3, m+5, \ldots \\ 0, & \text{otherwise.} \end{cases} \tag{8.17}$$

$$s^N_{mn} = \begin{cases} -2\pi m^2 \frac{m+1}{m+2}, & n = m, \\ -4\pi n^2 \frac{m+1}{(m+2)^2}, & n = m+2, m+4, \ldots \\ 0, & \text{otherwise.} \end{cases} \tag{8.18}$$

$$r^N_{mn} = \begin{cases} \frac{c_m \pi}{2}, & n = m, \\ -\beta_m \frac{\pi}{2}, & n = m+2, \\ 0, & \text{otherwise.} \end{cases} \tag{8.19}$$

The Neumann boundary conditions are indicated with the superscript N.

8.3 Integral constraint

For basis functions (7.20) satisfying the integral constraint, no elegant expression for the elements of the inner-product matrices (8.7)-(8.11) have been found. Moreover, in higher dimensions, basis functions satisfying the integral constraint cannot be formed via separation of variables, which is possible for homogeneous boundary conditions (chapter 10).

Therefore, we propose to use Chebyshev polynomials (7.1) as basis functions for a variables with an integral constraint. To yield a unique solution, we add a constant such that the integral is zero. Therefore, the basis functions (7.20) are not considered further.

9 Application to the Poisson equation

Suppose we want to solve the Poisson equation with Neumann boundary conditions. Let's examine the one dimensional case on the domain $\Omega = [-1, 1]$. In short,

$$\Delta u = f, \qquad \frac{\partial}{\partial x} u(\pm 1) = 0. \tag{9.1}$$

In one dimension the Laplace operator Δ is $\partial^2/\partial x^2$. The weak form of (9.1) is

$$(\Delta u, v)_\omega = (f, v)_\omega, \qquad \text{for all } v \in \boldsymbol{X},$$

where $\boldsymbol{X}$ is constrained to functions with Neumann boundary conditions. By approximating u in terms of CN basis functions (7.18),

$$u_N(x) = \sum_{n=0}^{N-2} \tilde{u}_n \phi_n(x),$$

the weak form can be expressed as a linear system. We approximate f in terms of Chebyshev basis functions

$$f_N(x) = \sum_{n=0}^{N} \hat{f}_n T_n(x).$$

Problem (9.1) can then be expressed as

$$S^N \mathbf{u} = R^N \mathbf{f}, \tag{9.2}$$

where S^N and R^N are respectively the stiffness and transformation matrices for Neumann basis functions, as given in (8.18) and (8.19). Additionally, the vectors $\mathbf{u}$ and $\mathbf{f}$ are composed of expansion coefficients,

$$\mathbf{u} = (\tilde{u}_0, \tilde{u}_1, \ldots, \tilde{u}_{N-2})^T, \qquad \mathbf{f} = (\hat{f}_0, \hat{f}_1, \ldots, \hat{f}_N)^T.$$

This situation can be extended to other boundary conditions, with corresponding basis functions (7.14), whose stiffness and transformation matrices can be created as outlined in Appendix D.2. Additionally, the situation can also be scaled to other domain lengths. The Poisson type problem is extended to higher dimensions in section 10.3.

9.1 Solution scheme

Here and in general, the Chebyshev-Galerkin approach to a problem with homogeneous boundary conditions roughly follows these steps:

1. Form a set of basis functions from a superposition of Chebyshev polynomials such that they satisfy the boundary conditions and span the solution space $\boldsymbol{X}_N$. Compute the relevant inner products.

2. Expand forcing f in Chebyshev coefficients $\hat{f}$ and evaluate the right hand side of (9.2).

3. Solve for the coefficients $\tilde{u}$ that minimise the residual of the linear system (9.2).

4. Compute the nodal values $u(x_j)$ from $\tilde{u}$ with a backwards transform.

Step 1 can be costly, but is only performed once. Steps 2 and 4 both involve $O(N \log N)$ operations. Finding the solution in step 3 is most costly.

In step 3, the solution can be approximated with aid of an iterative solver such as BiCGSTAB. All applications in this report use BiCGSTAB, which is described in Van der Vorst [23] and Barrett et al. in [1]. When an iterative method is used, the cost for each iterate must be multiplied by the number of iterations needed. The results in part III demonstrate that the Chebyshev-Galerkin framework allows for cost efficient and accurate solutions to the Poisson equation with Neumann or Dirichlet boundary conditions.

10 Extension to higher dimensions

Rectangular domains can be rescaled to $\Omega = [-1, 1]^D$ for D dimensions. This section extends the Chebyshev-Galerkin framework to higher dimensional second-order partial differential equations. First the two dimensional basis functions and transforms are introduced, followed by the three and higher dimensional case. Next, the linear systems representing the Poisson equation in two and higher dimensions are discussed.

10.1 Basis functions in two dimensions

Take a two dimensional domain $\Omega = [-1, 1]^2$. For $j \in \{0, 1, \ldots, N_x\}$, $k \in \{0, 1, \ldots, N_y\}$, the Chebyshev-Gauss-Lobatto points are (x_j, y_k) with

$$x_j = -\cos\left(\frac{j\pi}{N_x}\right), \quad y_k = -\cos\left(\frac{k\pi}{N_y}\right).$$

Consider Chebyshev polynomials in both the x and y direction

$$\begin{aligned} T_n(x) &= \cos(n\theta), & \theta &= \arccos(x), & x &\in \Omega,\ n \geq 0, \\ T_m(y) &= \cos(m\xi), & \xi &= \arccos(y), & y &\in \Omega,\ m \geq 0. \end{aligned}$$

A finite-dimensional Chebyshev approximation of $u(x, y)$ is

$$u(x, y) \approx \sum_{n=0}^{N_x} \sum_{m=0}^{N_y} \hat{u}_{nm} T_n(x) T_m(y). \tag{10.1}$$

Some literature denotes the two-dimensional basis functions as $T_{nm}(x, y) = T_n(x)T_m(y)$. The coefficients $\hat{u}_{nm}$ form a tensor. The entries of a two-dimensional tensor can be mapped straightforwardly to a matrix.

The coefficients $\hat{u}$ can be obtained from nodal values by applying the forward discrete Chebyshev transform (7.12) twice: once in the x- and once in the y-direction (Appendix E.1). Analogously, the nodal values can be retrieved from the coefficients using the backward discrete Chebyshev transform (7.13) twice.

10.1.1 Dirichlet and Neumann

A function u with boundary conditions can be approximated in terms of the basis functions $\phi_m(x)\phi_n(y)$,

$$u(x, y) \approx \sum_{n=0}^{N_x-2} \sum_{m=0}^{N_y-2} \tilde{u}_{nm} \phi_n(x) \phi_m(y), \tag{10.2}$$

where the notation $\phi_{nm}(x,y) = \phi_n(x)\phi_m(y)$ is often used as a convention. For Dirichlet boundary conditions $\phi_n(x)$ is given by (7.16). Alternatively, the basis functions ϕ_n as defined in (7.18) satisfies Neumann boundary conditions. Similarly to Chebyshev basis functions, transforms between the coefficients $\tilde{u}_{nm}$ and nodal values $u(x_j, y_k)$ are obtained by performing the one-dimensional transforms twice.

10.2 Three and higher dimensions

The applications in this report are at most three-dimensional. However, for completeness, this section examines the general case of D dimensions.

Consider the domain $\Omega = [-1,1]^D$ in D-dimensional space. Choose the number of grid points N_d in each direction $d \leq D$ and group them together as $\boldsymbol{N} = (N_1, N_2, \ldots, N_D)$. The vector $\boldsymbol{x} = (x^{(1)}, x^{(2)}, \ldots, x^{(D)})$ denotes the position vector. Let the grid nodes be labeled with a D-dimensional index $\boldsymbol{n} = (n_1, n_2, \ldots, n_D)$, where $0 \leq n_d \leq N_d$. The positions of the Chebyshev-Gauss-Lobatto nodes $\boldsymbol{x_n}$ are

$$\begin{aligned} \boldsymbol{x_n} =& (x^{(1)}_{n_1}, \ldots, x^{(d)}_{n_d}, \ldots, x^{(D)}_{n_D}), & x^{(d)}_{n_d} =& -\cos\left(\frac{n_d \pi}{N_d}\right), \\ 0 \leq& n_d \leq N_d, & 1 \leq& d \leq D. \end{aligned}$$

We want to approximate u in terms of a tensor $\tilde{u}$ containing the expansion coefficients. A D-dimensional version of (10.1) is

$$u(\boldsymbol{x_n}) \approx \sum_{\boldsymbol{k}=0}^{\boldsymbol{N}} \hat{u}_{\boldsymbol{k}} \prod_{d=1}^{D} T_{k_d}(x^{(d)}_{n_d}), \tag{10.3}$$

where $\boldsymbol{k} = (k_1, k_2, \ldots, k_D)$ labels the expansion coefficients with $0 \leq k_d \leq N_d$. Above, the sum means summing over each k_d in $\boldsymbol{k}$ independently (Appendix E). The Chebyshev coefficients $\hat{u}_{\boldsymbol{k}}$ of (10.3) can be determined by applying the forward discrete Chebyshev transform (7.12) D times. Likewise, one executes the backwards discrete Chebyshev transform (7.13) D times to retrieve the nodal values (Appendix E.1).

The forward and backward transforms cost order $N_{tot} \log(N_{tot})$ computations, where $N_{tot} = \prod_{d=1}^{D} N_d$.

10.2.1 Dirichlet and Neumann

Consider a three-dimensional domain $\Omega = [-1, 1]^3$. Let's approximate u in terms of the basis functions ϕ and expansion coefficients $\tilde{u}$:

$$u(x, y, z) \approx \sum_{n=0}^{N_1-2} \sum_{m=0}^{N_2-2} \sum_{k=0}^{N_3-2} \tilde{u}_{nmk} \phi_n(x) \phi_m(y) \phi_k(z). \tag{10.4}$$

Recall that the basis functions are

$$\phi_n = T_n - \beta_n T_{n+2}$$

where $\beta_n = 1$ and $\beta_n = \frac{n^2}{(n+2)^2}$ for respectively Dirichlet and Neumann boundary conditions. Moreover, the transforms between nodal values and coefficients are performed by computing the one-dimensional fast transform three times.

10.3 Poisson in 2D

The Poisson equation serves as a good example of a second-order partial differential equation in the following sections. In two dimensions, the Poisson equation is

$$\mathcal{L}u := \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u(x, y) = f(x, y).$$

Assume Neumann boundary conditions and let the solution in exist in the space of Nth-order polynomials. Then, the solution space $\boldsymbol{X}_N$ is spanned by the Neumann basis functions $\{\phi_i(x)\phi_j(y)\}_{i,j=0}^{N-2}$, each defined in (7.18). In the weak form

$$(\mathcal{L}u_N, \phi_i(x)\phi_j(y))_\omega = (f, \phi_i(x)\phi_j(y))_\omega, \quad \text{for all } i, j. \tag{10.5}$$

It holds that

$$\Big(\phi_m(x)\phi_n(y), \phi_k(x)\phi_l(y)\Big)_\omega = \Big(\phi_m(x), \phi_k(x)\Big)_\omega \Big(\phi_n(y), \phi_l(y)\Big)_\omega. \tag{10.6}$$

Substitution of approximation (10.2) into relation (10.5) yields a linear system (Appendix E.2),

$$SUM^T + MUS^T = RFR^T, \tag{10.7}$$

where U and F are matrices composed of coefficients $\tilde{u}_{nm}$ and $\hat{f}_{kl}$ and the elements of matrices S, M and R are given by s_{mn}^N (8.18), m_{mn}^N (8.16) and r_{mn}^N (8.19).

The linear system for U, F is equivalent to the linear system for vectors $\mathbf{u}, \mathbf{f}$ employing Kronecker products

$$(M \otimes S + S \otimes M)\,\mathbf{u} = (R \otimes R)\mathbf{f}, \tag{10.8}$$

where $\mathbf{u} = \mathrm{vec}(\tilde{u}_{nm})$ and $\mathbf{f} = \mathrm{vec}(\hat{f}_{kl})$. Appendix E.3 gives the definition of a Kronecker product $\otimes$ and of $\mathrm{vec}(X)$. Additionally, the equivalence between (10.7) and (10.8) is shown in Appendix E.3.

10.4 Operators in higher dimensions

Let's consider the general problem (6.1) on a D-dimensional domain

$$\mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x}).$$

Assume the operator is linear. Then we can split the operator as a sum of operators, each of which is the product of a partial operator in each direction

$$\sum_{k=1}^{3^D} \left(\prod_{d=1}^{D} \mathcal{L}^{(d)} \right) u(\boldsymbol{x}) = f(\boldsymbol{x}).$$

For example, the three-dimensional Laplace operator is the sum of three operators, each of which takes the second-derivative in one direction and is the identity operator in the other directions.

The functions $u(\boldsymbol{x})$ can be approximated in coefficients $\tilde{u}_{\boldsymbol{n}}$, forming a D-dimensional tensor. Let the indices be indicated by $\boldsymbol{n} = (n_1, n_2, \ldots, n_D)$, where each component $0 \le n_d \le N_d - 2$. The chosen number of grid points in each direction can be grouped as $\boldsymbol{N} = (N_1, N_2, \ldots, N_D)$. Expansion in terms of $\tilde{u}$ and $\hat{f}$ gives the following system for weak solutions,

$$\sum_{\boldsymbol{n}=0}^{\boldsymbol{N}-2} \left(\prod_{d=1}^{D} l^{(d)}_{m_d,n_d} \right) \tilde{u}_{\boldsymbol{n}} = \sum_{\boldsymbol{k}=0}^{\boldsymbol{K}} \left(\prod_{d=1}^{D} r^{(r)}_{m_d,k_d} \right) \hat{f}_{\boldsymbol{k}}, \quad \text{for } 0 \le m_d \le N_d - 2, \tag{10.9}$$

where

$$l^{(d)}_{m_d,n_d} = \left(\mathcal{L}^{(d)} \phi_{n_d}, \phi_{m_d} \right)_\omega \text{ and } r^{(d)}_{m_d,k_d} = \left(T_{k_d}, \phi_{m_d} \right)_\omega .$$

Analogous to the situation in two dimensions, this problem can be rewritten in terms of Kronecker products

$$\left(L^{(1)} \otimes L^{(2)} \otimes \cdots \otimes L^{(D)} \right) \mathbf{u} = \left(R^{(1)} \otimes R^{(2)} \otimes \cdots \otimes R^{(D)} \right) \mathbf{f}, \tag{10.10}$$

where

$$L^{(d)} = (l^{(d)}_{m_d,n_d})_{0\le m,n\le N_d-2}, \qquad \mathbf{u} = \mathrm{vec}(\tilde{u}_{\boldsymbol{n}}),$$
$$R^{(d)} = (r^{(d)}_{m_d,k_d})_{0\le m\le N_d-2,\, 0\le k\le N_d}, \qquad \mathbf{f} = \mathrm{vec}(\hat{f}_{\boldsymbol{k}}).$$

Upon designing a computation scheme, it is desired that the multiplications can be computed efficiently. This is often measured in terms of computation cost, which counts the number of required floating point operations.

Computations with the Kronecker formulation (10.10) can be very costly, because one needs a matrix of size N_{tot}^2, where $N_{tot} = \prod_{d=1}^{d} N_d$. Instead, like in two dimensions, one can perform the multiplications in (10.10) as a sequence of D matrix-matrix multiplications. As an advantage, the Kronecker product need not be formed explicitly and the multiplication cost is reduced to $O(\sum_{d=1}^{D} N_d N_{tot})$. In chapter 12 efficient matrix multiplication is performed with help of recursion relations, reducing costs further.

The Kronecker product multiplication can be computed indirectly with D successive matrix-matrix multiplications. For each direction d, we map the tensor entries $\tilde{u}_{\mathbf{n}}$ to an $N_d \times (N_{tot}/N_d)$ matrix such that the dth tensor index is the row index of the matrix. Then, we multiply $L^{(d)}$ with this matrix and map the result back to a tensor (to be used for the next direction). Repeat this for each $1 \le d \le D$.

The two-dimensional case was given in section 10.3, where equation (10.7) is the two-dimensional Poisson case of (10.9), and (10.8) corresponds to (10.10).

10.5 Poisson in 3D

The Poisson equation uses the Laplace operator Δ in three dimensions

$$\Delta u = \frac{\partial^2}{\partial x^2}u + \frac{\partial^2}{\partial y^2}u + \frac{\partial^2}{\partial z^2}u = f.$$

So, the Laplace operator can be split into a sum of three operators: each taking the second derivative in one direction, and being the identity operator in the other directions. In the weak case, approximating u in terms of basis functions, for every $0 \le i \le N_1 - 2,\ 0 \le j \le N_2 - 2,\ 0 \le l \le N_3 - 2$ it must

hold that

$$\sum_{n=0}^{N_1-2}\sum_{m=0}^{N_2-2}\sum_{k=0}^{N_3-2} s_{in}m_{jm}m_{lk}\,\tilde{u}_{nmk} + \sum_{n=0}^{N_1-2}\sum_{m=0}^{N_2-2}\sum_{k=0}^{N_3-2} m_{in}s_{jm}m_{lk}\,\tilde{u}_{nmk} + \sum_{n=0}^{N_1-2}\sum_{m=0}^{N_2-2}\sum_{k=0}^{N_3-2} m_{in}m_{jm}s_{lk}\,\tilde{u}_{nmk} = \sum_{r=0}^{N_1}\sum_{s=0}^{N_2}\sum_{t=0}^{N_3} r_{ir}r_{js}r_{lt}\,\hat{f}_{rst}. \quad (10.11)$$

Here S, M and R are the inner product matrices for a set of appropriate basis functions. Equivalently one can write (10.11) in Kronecker product notation

$$(S \otimes M \otimes M + M \otimes S \otimes M + M \otimes M \otimes S)\,\mathbf{u} = (R \otimes R \otimes R)\,\mathbf{f}. \quad (10.12)$$

In chapter 12 it is derived that the left hand side can be computed using cost efficient recursive schemes in $O(N_1N_2N_3)$ computations.[4] As a result, solving this linear system with an iterative method is attractive.

[4] As explained above, the multiplication with the Kronecker product can be obtained as matrix-matrix multiplications in each x, y, z-direction. The multiplications can be computed with a fast recursion scheme in $O(\text{size}(\tilde{u}))$ flops. For example, in three dimensions $\text{size}(\tilde{u}) = N_1N_2N_3$.

11 Application to the Stokes equations

In this section we derive the inner product system equivalent to the Stokes equations (5.6)-(5.7) with homogeneous Dirichlet boundary conditions. In three dimensions, define $\mathbf{u} = (u, v, w)^T$ and $\mathbf{v} = (a, b, c)^T$. In the weak form of (5.5), it must hold for all $\mathbf{v}$ and q that

$$\begin{aligned} \alpha\left(\mathbf{u}, \mathbf{v}\right)_\omega + R_o^{-1}\left(\mathbf{k} \times \mathbf{u}, \mathbf{v}\right)_\omega - R_e^{-1}\left(\Delta \mathbf{u}, \mathbf{v}\right)_\omega + \left(\nabla p, \mathbf{v}\right)_\omega &= \left(\mathbf{f}, \mathbf{v}\right)_\omega, \\ \left(\nabla \cdot \mathbf{u}, q\right)_\omega &= 0. \end{aligned} \tag{11.1}$$

Consider a finite-dimensional expansion of u, v, w in terms of CD basis functions $\{\phi_{\boldsymbol{n}}\}_0^{\boldsymbol{N}-2}$ described by (7.16) with coefficients $\tilde{u}, \tilde{v}, \tilde{w}$. Analogously, one expands p, f in terms of Chebyshev polynomials $\{T_n\}_0^{N-2}$ with coefficients $\hat{p}, \hat{f}$. As the weak form must hold for all a, b, c, q, we can choose them from the set $\{\phi_{\boldsymbol{j}}\}_0^{\mathbf{N}-2}$.

It was motivated in Shen et al. [21] that pressure p and test function q should be in the space of $N-2$ degree polynomials to yield a well-posed problem. This can be understood intuitively because the second-order derivatives of an order N polynomial are order $N-2$. Therefore, it seems logical to expand u with second-order derivatives in two-order-higher polynomials N, which (with boundary conditions) can be spanned by basis functions $\{\phi\}_0^{N-2}$. As a result, for an $N \times N \times N$ grid, all the matrices are $N-2 \times N-2$.

Following the example in section 10.5, it is straightforward to derive the next Kronecker product formulations. The one-dimensional inner-product matrices are given in section 8. Define $M^{(3)} = M^D \otimes M^D \otimes M^D$. The first term in (11.1) is discretised as,

$$\left(\mathbf{u}, \mathbf{v}\right)_\omega \mapsto \begin{pmatrix} M^{(3)} & 0 & 0 \\ 0 & M^{(3)} & 0 \\ 0 & 0 & M^{(3)} \end{pmatrix} \begin{pmatrix} \mathrm{vec}(\tilde{u}) \\ \mathrm{vec}(\tilde{v}) \\ \mathrm{vec}(\tilde{w}) \end{pmatrix}.$$

with vec( ) as defined in Appendix E.3. Similarly, with $\mathbf{k} = (k_1, k_2, k_3)^T$

$$\left(\mathbf{k} \times \mathbf{u}, \mathbf{v}\right)_\omega \mapsto \begin{pmatrix} 0 & -k_3 M^{(3)} & k_2 M^{(3)} \\ k_3 M^{(3)} & 0 & -k_1 M^{(3)} \\ -k_2 M^{(3)} & k_1 M^{(3)} & 0 \end{pmatrix} \begin{pmatrix} \mathrm{vec}(\tilde{u}) \\ \mathrm{vec}(\tilde{v}) \\ \mathrm{vec}(\tilde{w}) \end{pmatrix}.$$

Recall $L^{(3)} = S^D \otimes M^D \otimes M^D + M^D \otimes S^D \otimes M^D + M^D \otimes M^D \otimes S^D$. Here $L^{(3)}$ represents the discrete Laplace operator with Dirichlet boundary conditions in contrast to its definition with Neumann boundary conditions in section

13.3. So,

$$(\Delta\mathbf{u}, \mathbf{v})_\omega \mapsto \begin{pmatrix} L^{(3)} & 0 & 0 \\ 0 & L^{(3)} & 0 \\ 0 & 0 & L^{(3)} \end{pmatrix} \begin{pmatrix} \text{vec}(\tilde{u}) \\ \text{vec}(\tilde{v}) \\ \text{vec}(\tilde{w}) \end{pmatrix}.$$

With $G^x = G^D \otimes R^D \otimes R^D$, $G^y = R^D \otimes G^D \otimes R^D$, and $G^z = R^D \otimes R^D \otimes G^D$,

$$(\nabla p, \mathbf{v})_\omega \mapsto \begin{pmatrix} G^x & G^y & G^z \end{pmatrix} \begin{pmatrix} \text{vec}(\hat{p}) \end{pmatrix}.$$

We consider $\mathbf{f} = (f_1, f_2, f_3)^T$ and define $R^{(3)} = R^D \otimes R^D \otimes R^D$, such that

$$(\mathbf{f}, \mathbf{v})_\omega \mapsto \begin{pmatrix} R^{(3)} & 0 & 0 \\ 0 & R^{(3)} & 0 \\ 0 & 0 & R^{(3)} \end{pmatrix} \begin{pmatrix} \text{vec}(\hat{f}_1) \\ \text{vec}(\hat{f}_2) \\ \text{vec}(\hat{f}_3) \end{pmatrix}.$$

Lastly, $K^x = K^D \otimes M^D \otimes M^D$, $K^y = M^D \otimes K^D \otimes M^D$, and $K^z = M^D \otimes M^D \otimes K^D$, such that

$$(\nabla \cdot \mathbf{u}, q)_\omega \mapsto \begin{pmatrix} K^x \\ K^y \\ K^z \end{pmatrix} \begin{pmatrix} \text{vec}(\tilde{u}) \\ \text{vec}(\tilde{v}) \\ \text{vec}(\tilde{w}) \end{pmatrix}.$$

Gathering the terms together in $A^{(3)}, B^{(3)}$ and $C^{(3)}$ yields

$$A^{(3)} = \alpha \begin{pmatrix} M^{(3)} & 0 & 0 \\ 0 & M^{(3)} & 0 \\ 0 & 0 & M^{(3)} \end{pmatrix} + R_0^{-1} \begin{pmatrix} 0 & -k_3 M^{(3)} & k_2 M^{(3)} \\ k_3 M^{(3)} & 0 & -k_1 M^{(3)} \\ -k_2 M^{(3)} & k_1 M^{(3)} & 0 \end{pmatrix}$$
$$-R_e^{-1} \begin{pmatrix} L^{(3)} & 0 & 0 \\ 0 & L^{(3)} & 0 \\ 0 & 0 & L^{(3)} \end{pmatrix},$$

$$B^{(3)} = \begin{pmatrix} G^x & G^y & G^z \end{pmatrix},$$

$$C^{(3)} = \begin{pmatrix} K^x \\ K^y \\ K^z \end{pmatrix}.$$

Let us define $\tilde{\mathbf{u}} = (\text{vec}(\tilde{u}), \text{vec}(\tilde{v}), \text{vec}(\tilde{w}))^T$, $\hat{\mathrm{p}} = \text{vec}(\hat{p})$, and

$$\hat{\mathbf{f}} = \begin{pmatrix} R^{(3)} & 0 & 0 \\ 0 & R^{(3)} & 0 \\ 0 & 0 & R^{(3)} \end{pmatrix} \begin{pmatrix} \text{vec}(\hat{f}_1) \\ \text{vec}(\hat{f}_2) \\ \text{vec}(\hat{f}_3) \end{pmatrix}.$$

As a result, one can write a single time-step of the Navier-Stokes equations as

$$A^{(3)}\tilde{\mathbf{u}} + B^{(3)}\hat{\mathrm{p}} = \hat{\mathbf{f}}, \tag{11.2}$$

$$C^{(3)}\tilde{\mathbf{u}} = 0. \tag{11.3}$$

This can be separated into an equation for $\hat{\mathrm{p}}$ and for $\tilde{\mathbf{u}}$ like in (5.8)-(5.9).

For time-dependent simulations, linear system (11.2)-(11.3) must be solved at each time step. Because of the low complexity of these matrices, we propose to use an iterative solver such as BiCGSTAB. Equivalent to the simple Poisson case in section 13.3, one could investigate preconditioners formed by the diagonals of the matrices themselves.

11.1 Implementation

In this project, a pilot toolbox for the Chebyshev-Galerkin method was developed for Matlab. It includes transforms between the grid point values and several expansion coefficients, the inner-product matrices, and the fast multiplication schemes (chapter 12). Additionally, functions for grid creation, numerical weighted integration, boundary condition lifting for pipe flow (section 5.4), and integral constraint adjustment are present. All functions can handle one, two or three dimensional domains. The implemented functions can be combined to solve the Stokes time-marching scheme (11.2)-(11.3).

A complete time-marching for the pressure-correction method was implemented in Matlab. It was observed that Poisson equation (5.4) was not tolerably solved for randomly generated velocity $\tilde{\mathbf{u}}$ (with Dirichlet boundary conditions) and the simulations were unstable.

Part III
Mathematical Results

Linear second-order partial differential equations on rectangular domains can be reformulated in the finite-dimensional weak form as a linear system. This chapter explores the properties of these linear systems. The complexity, convergence and condition number of the matrices that form the linear system are presented. To the best of our knowledge, the subsequent results are new to the literature. Note that in this part, an $N \times N$ matrix has indices $0 \leq i, j \leq N - 1$.

12 Complexity

In the literature the term *complexity* is most often used to express the difficulty of a problem. Different fields have different definitions. In computational sciences complexity often measures the amount of either communication, storage or processors. Within this report we take a machine-independent definition: the complexity (of an algorithm) is the computational cost in terms of floating-point operations (flops).

For example, the general matrix-vector multiplication of a matrix in $\mathbb{R}^{N\times N}$ with column vector in $\mathbb{R}^N$ costs $2N^2$ operations, thus the complexity is $O(N^2)$.

Sparse matrices can be computed in fewer operations: approximately twice the number of nonzero entries. The simplest example is a diagonal matrix having nonzero entries on the main diagonal and zeros elsewhere. Multiplication with a diagonal matrix is equivalent to N element-wise multiplications.

Moreover, consider a matrix with nonzero entries along one or more (diagonal) bands distance d from the main diagonal: $a_{ij} \neq 0$ when $j = i + d$, for $d \in \mathbb{Z}$. A matrix is *banded* when its nonzeros are restricted to a bounded region around the main diagonal. The number of bands k with nonzero entries is matrix-size independent and multiplication of such a matrix with a vector costs $2kN$ flops.

A checkerboard matrix consists of alternating ones and zeros. Formally, $a_{ij} = 1$ if $i + j$ is even. Upon multiplying a vector with a checkerboard matrix, the resulting column vector $g \in \mathbb{R}^N$ has two alternating values: $g_j = g_0$ for all j even, and $g_j = g_1$ for all j odd. Computation of g_0 and g_1 each cost N flops, totalling $2N$ operations for the entire matrix-vector

multiplication.

We are interested in the complexity of the stiffness, first-derivative and mass matrixes, because they form the linear system representing our application. As described in section 9.1, an iterative method involves many matrix multiplications. Hence we want to compute these as efficiently as possible. Our application is composed of Poisson-type equations, whose corresponding linear system involves the stiffness matrix, which has an upper-triangular checkerboard structure.

An upper-triangular checkerboard matrix has a checkerboard structure in the upper right part and is zero below the main diagonal: $a_{ij} = 1$ if $i \leq j$ and $i + j$ even. Fast matrix-vector multiplication starts at the bottom row and recursively extends upwards. For the upper-triangular checkerboard matrix A, consider $g = Au$. The fast algorithm is

$$\begin{aligned} g_{N-1} &= u_{N-1}, \\ g_{N-2} &= u_{N-2}, \\ g_n &= g_{n+2} + u_n, \quad \text{for } n = N-3 \text{ to } 0, \end{aligned} \tag{12.1}$$

which takes N operations.

Moreover, a strictly-upper-triangular checkerboard matrix has zeros along the main diagonal: $a_{ij} = 1$ for $i < j$ and $i + j$ even. With a slight variation of algorithm (12.1)

$$\begin{aligned} g_{N-1} &= 0, \\ g_{N-2} &= 0, \\ g_n &= g_{n+2} + u_{n+2}, \quad \text{for } n = N-3 \text{ to } 0, \end{aligned} \tag{12.2}$$

the costs are again N flops.

The matrices introduced in chapter 8 can be decomposed into products and sums of diagonal, banded and (strictly) upper-triangular checkerboard matrices. As a result, the multiplication costs are $O(N)$. The fast algorithms and their complexity are in Appendix F.

12.1 Extension to more dimensions

Sections 10.3 and 10.4 show that linear second-order partial differential equations in more dimensions can be solved as a linear system involving products of the mass, first-derivative and stiffness matrices. In two dimensions these systems require operations of the form $G = AUB^T$ with A and B being $N \times N$ matrices. Decompose $G = AUB^T$ as

$$G = AV^T, \quad V = BU^T.$$

If Av and Bu can be computed in $O(N)$ operations via a fast scheme, then the products AV and BU for any $U, V \in \mathbb{R}^{N\times N}$ entail $O(N^2)$ complexity. So when the matrix-vector product of A, B entails $O(N)$ complexity, the product $A(BU^T)^T$ cost $O(N^2)$ flops, which is the size of the system. Thus the complexity is linear.

We extend this approach to D-dimensional rectangular domains. As explained in section 10.4, we can compute several matrix-matrix products instead of a kronecker-product matrix-vector multiplication. Each matrix-matrix multiplication has linear complexity using fast schemes. With $N_{tot} = \prod_{d=1}^{D} N_d$, computation with fast schemes has $O(N_{tot})$ complexity. The cost of vector multiplication with a $N_{tot}\times N_{tot}$ matrix resulting from the kronecker product (10.10) is $O(N_{tot}^2)$. With fast schemes the complexity is the square root of the cost with the kronecker-product matrix.

Iterative solution methods repeatedly evaluate the residual $r^t = g - Au^t$ for improved estimates of u. For systems that are not symmetric, BiCGSTAB is a fast-converging iterative method. More details on BiCGSTAB can be found in Van der Vorst [23] and in Barrett et al. [1]. One advantage of iterative methods is that they avoid constructing the inverse of an $N^D \times N^D$ matrix, which is generally full even when the matrix itself is sparse and which requires $O(N_{tot}^3)$ flops. An iterative solver employing fast schemes saves both computation and memory costs.

13 Condition number

In this section the condition number of the discrete Laplace operator Δ is analysed. The one-dimensional Laplace operator is just the stiffness matrix. This allows for both numerical computations and analytical bounds on the growth of the condition number with matrix size N. Furthermore, we examine the dependence on N of numerically computed condition numbers in higher dimensions. In both the one and more dimensional case preconditioners are proposed to improve the condition number.

The condition number κ of a matrix A is

$$\kappa(A) = \left|\left|A^{-1}\right|\right| \, ||A|| \,, \tag{13.1}$$

where one can use any induced norm or the Frobenius norm (Golub and Van Loan [6]). The standard choice is the 2-norm.

Let's consider $Au = f$. The condition number κ indicates the sensitivity of the output f on the input u. A small condition number suggests that a slight change in the input will not lead to a large change in the output. This property is beneficial when solving a linear system. Matrices with low condition numbers take less iterations in iterative methods.

Preconditioning is a procedure to lower the condition number of a matrix A by multiplying with the inverse of a so-called preconditioner M. Let M be a matrix whose inverse is cheap and such that $M^{-1}A$ has a lower condition number than A. Then

$$M^{-1}Au = M^{-1}f$$

is equivalent to solving the original $Au = f$ system and takes less iterations in an iterative method. This is referred to as the preconditioned system and $\hat{A} = M^{-1}A$ is referred to as the preconditioned operator.

Similarly one can precondition from the right as

$$AM^{-1}v = f, \quad u = M^{-1}v.$$

13.1 Preconditioning of Dirichlet

The stiffness matrix S^D with Dirichlet boundary conditions defined in (8.14) is given by

$$s^D_{mn} = \begin{cases} -2\pi(m+1)(m+2), & n = m, \\ -4\pi(m+1), & n = m+2, m+4, \ldots, \\ 0, & n < m \,\text{or}\, n+m \,\text{odd}, \end{cases}$$

With a diagonal preconditioner P^D, whose entries are given by the diagonal of S^N,

$$p^D_{mm} = -2\pi(m+1)(m+2),$$

we can precondition S^D as

$$\hat{S} = (P^D)^{-1} S^D,$$

Substitution of P^D and S^D yields that $\hat{S}$ is an upper-triangular matrix given by

$$\hat{s}_{mn} = \begin{cases} 1, & m = n, \\ 2/(m+2), & n = m+2, m+4, \ldots, \\ 0, & \text{otherwise.} \end{cases} \tag{13.2}$$

13.1.1 Analytical bounds

To find a bound on the condition number, the norms of the matrix and its inverse must be bounded. Let's denote the inverse of S^D by $Z = (S^D)^{-1}$, whose elements are derived explicitly in Appendix G.1.1,

$$z_{kj} = \begin{cases} -\big(2\pi\,(n+1)(n+2)\big)^{-1}, & n = m, \\ \big(4\pi\, n(n+1)(n+2)\big)^{-1}, & n = m+2, m+4, \ldots, \\ 0, & \text{otherwise.} \end{cases} \tag{13.3}$$

Let N denote the size of S^D. Using norm inequalities, it is straightforward to compute (Appendices G.1.2 and G.1.3)

$$\left|\left|S^D\right|\right|_2 \leq O(N^2), \text{ and } ||Z||_2 \leq O(1).$$

Combine this with definition (13.1) to conclude

$$\kappa_2(S^D) \leq O(N^2). \tag{13.4}$$

Preconditioned

The preconditioned version of S^D was denoted $\hat{S}$ and its entries are given in (13.2). The inverse of $\hat{S}$ is denoted by $\hat{Z}$, whose entries can easily be derived (Appendix G.1.1) as

$$\hat{z}^{-1}_{mn} = \begin{cases} 1, & n = m, \\ -\frac{2}{n}, & n = m+2, m+4, \ldots \\ 0, & \text{otherwise.} \end{cases} \tag{13.5}$$

From the explicit expressions for $\hat{S}$ and $\hat{Z}$, one can derive (Appendices G.1.4-G.1.5) that the condition number of $\hat{S}$ is bounded by

$$\begin{aligned}\kappa_1(\hat{S}) \leq & O(N^{1/2}),\\ \kappa_2(\hat{S}) \leq & O\left(\sqrt{N\ln(N/2)}\right).\end{aligned} \tag{13.6}$$

This upper bound grows sublinearly with N.

13.2 Preconditioning of Neumann

For Neumann boundary conditions the stiffness matrix S^N defined in (8.18) is given by

$$s^N_{mn} = \begin{cases} -2\pi m^2(m+1)/(m+2), & n = m,\\ -4\pi n^2(m+1)/(m+2)^2, & n = m+2, m+4, m+6...,\\ 0, & n < m \text{ or } n+m \text{ odd}.\end{cases}$$

We decompose S^N as

$$S^N = P^N \hat{S} B^N, \tag{13.7}$$

where $\hat{S}$ is defined in (13.2), and P^N, B^N are diagonal matrices given by

$$\begin{aligned}P^N_{mm} &= -2\pi(m+1)/(m+2),\\ B^N_{mm} &= m^2.\end{aligned} \tag{13.8}$$

Because P^N has amplitude order 1, we combine P^N and $\hat{S}$. Define $\hat{L}^{(1)}$ as

$$\hat{L}^{(1)} = P^N \hat{S}, \tag{13.9}$$

such that

$$S = \hat{L}^{(1)} B^N.$$

The superscript of $\hat{L}^{(1)}$ refers the one-dimensional domain on which the operator is relevant. Consider $S^N u = g$, which can be rewritten as

$$\hat{L}^{(1)} v = g, \quad u = (B^N)^{-1} v. \tag{13.10}$$

The inverse of B^N is performed element-wise along the diagonal, which can be executed in N entry-wise divisions.

13.2.1 Analytical bound on the condition number

In section 13.1, the condition number of $\hat{S}$ has been analytically bounded (13.6),

$$\kappa_2(\hat{S}) \leq O\left(\sqrt{N \ln(N/2)}\right).$$

It is straightforward to extend the derivation in Appendix G.1.4 with a factor p_{mm}^N. Because a bound on the amplitude of P^N is N-independent ($|p_{mm}^N| \leq 2\pi$ and $|p_{mm}^{-1}| \leq 1/\pi$), the condition number of $\hat{L}^{(1)}$ is the same order as the condition number of $\hat{S}$,

$$\kappa_2(\hat{L}^{(1)}) \leq O\left(\sqrt{N \ln(N/2)}\right). \tag{13.11}$$

13.2.2 Numerical computation of the condition number

The condition number in the 1-norm, κ_1, and 2-norm, κ_2, of matrices S^N and $\hat{L}^{(1)}$ are computed explicitly for various matrix sizes N. Figure 13.1 displays the condition numbers as a function of N. The graphs also show the average number of iterations the BiCGSTAB algorithm needs to obtain the required precision. The settings for these tests are in Appendix G.3. Note the different axis scaling in panels A and B.

The results in Figure 13.1.A indicate that the condition number of the stiffness matrix S^N grows with order N^2 and that the number of iterations grows with $O(N)$. The preconditioned operator $\hat{L}^{(1)}$ in Figure 13.1.B has a condition number that grows with less than order $N^{1/2}$. As expected this is below the upper bound predicted by inequality (13.11) . On top of that, the growth rate of the number of iterations is much lower than $N^{1/2}$.

As contemplated in section 13.2.1, the numerically obtained condition numbers for $\hat{S}$ and $\hat{L}^{(1)}$ show similar growth with N. Comparison of the condition numbers and iteration counts is shown in Appendix G.2.

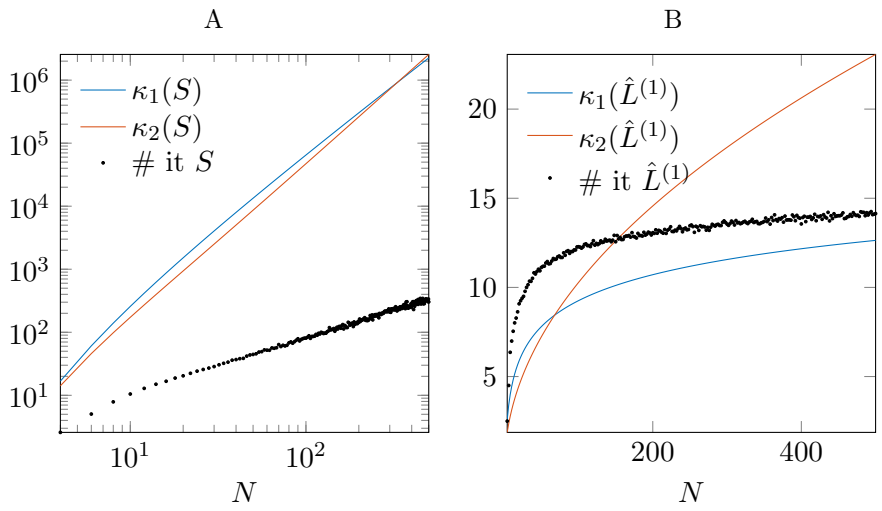

Figure 13.1: The condition numbers in 1-norm (blue) and 2-norm (red) of the Neumann stiffness matrix S^N (panel A) and the preconditioned stiffness matrix $\hat{L}^{(1)}$ (panel B) are shown on the left and right respectively, where N denotes the number of CGL points. Additionally, the dots denote the average number of iterations the BiCGSTAB algorithm needs to obtain the required precision.

13.3 Preconditioning in Multiple Dimensions

In two dimensions the Poisson equation gives linear system (10.8),

$$(M \otimes S + S \otimes M)\,\mathbf{u} = (R \otimes R)\,\mathbf{f}.$$

Let's denote by $L^{(2)} = (M \otimes S + S \otimes M)$ the discrete Laplacian operator in two dimensions. One can precondition $L^{(2)}$ as $\hat{L}^{(2)} = L^{(2)}(B^{(2)})^{-1}$ with

$$B^{(2)} = (I \otimes B^N + B^N \otimes I),$$

where B^N is defined in (13.8). Similarly, in three dimensions, denote

$$L^{(3)} = (M \otimes M \otimes S + M \otimes S \otimes M + S \otimes M \otimes M). \tag{13.12}$$

With this notation, the three-dimensional Poisson equation (10.12) becomes

$$L^{(3)}\,\mathbf{u} = (R \otimes R \otimes R)\,\mathbf{f}$$

One can precondition $L^{(3)}$ as $\hat{L}^{(3)} = L^{(3)}(B^{(3)})^{-1}$ with

$$B^{(3)} = (I \otimes I \otimes B^N + I \otimes B^N \otimes I + B^N \otimes I \otimes I).$$

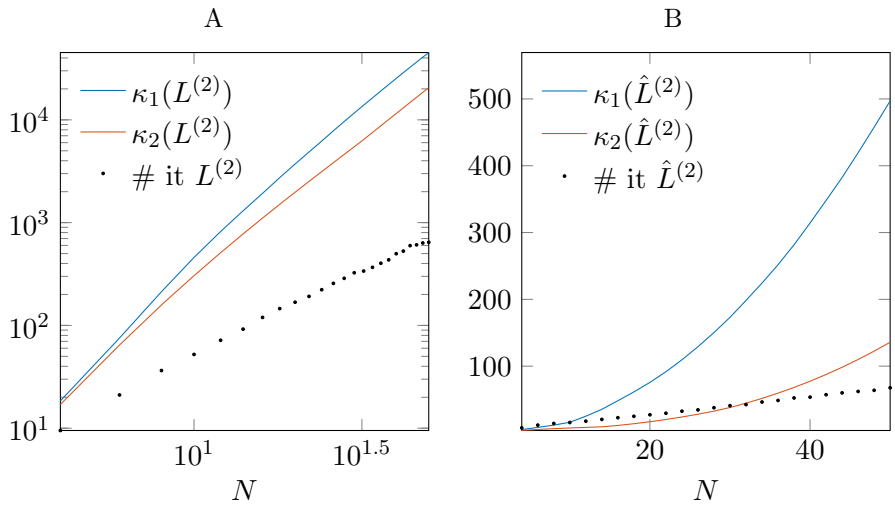

Figure 13.2: In two dimensions, the condition numbers in 1-norm (blue) and 2-norm (red) of the Laplacian operator $L^{(2)}$ (panel A) and the preconditioned operator $\hat{L}^{(2)}$ (panel B) with respect to N, denoting the number of CGL points in each direction. Additionally, the dots denote the average number of iterations the BiCGSTAB algorithm needs to obtain the required precision.

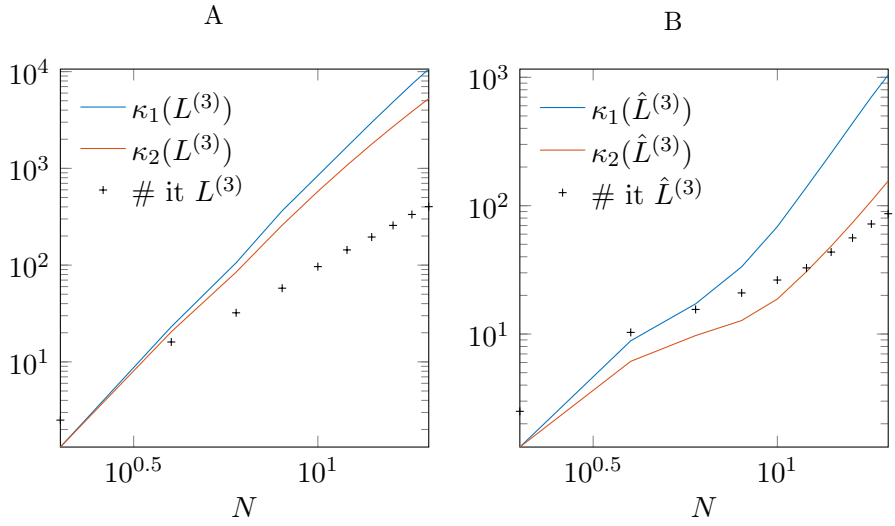

Figure 13.3: For the three-dimensional Laplacian, the condition numbers in 1-norm (blue) and 2-norm (red) of $L^{(3)}$ (A) and preconditioned $\hat{L}^{(3)}$ (B) are shown on the left and right respectively. The values are plotted against N denoting the number of CGL points in each direction. The crosses denote the average number of iterations the BiCGSTAB algorithm needs to obtain the required precision.

13.3.1 Condition Number

For notational convenience, the resolution is set to N in all directions. Thus, the column vectors $\mathbf{u}, \mathbf{f}$ have length N^2 or N^3 for two or three dimensions respectively.

Figure 13.2 displays that the condition number of the Laplacian operator $L^{(2)}$ grows with N^3 and that the average number of BiCGSTAB iterations is order N^2. In contrast, for the preconditioned matrix $\hat{L}^{(2)}$, the condition number growth rate reduces to order N^2 and the number of iterations is order N. Note that the matrix is $N^2 \times N^2$, thus with N iterations, the scheme is sublinear. Observe the different scaling of the axes in Figure 13.2.

In three dimensions, the condition number of the Laplacian operator $L^{(3)}$ grows with $O(N^4)$ and the iteration count with order N^2, shown in Figure 13.3. For the preconditioned Laplace operator $\hat{L}^{(3)}$ the growth rate of the condition number and the iteration count are $O(N^3)$ and $O(N^{3/2})$ respectively. The latter growth rate is sublinear for a problem with an unknown vector of length N^3.

When the tolerance value of the BiCGSTAB is increased or decreased, the average number of iterations changes accordingly. However, for any fixed tolerance, the iterations seems to grow with $O(N^{D/2})$ for the preconditioned D-dimensional discrete Laplace operator.

14 Convergence

Spectral methods have global basis functions. Therefore, the error between a solution and its spectral approximation is not concentrated on part of the domain, but spread out evenly. Spectral methods generally display exponential convergence, also referred to as *spectral convergence*. This means that, as a function of resolution N, the error decreases faster than $O(N^{-p})$ for any p.

The convergence test is conducted on a Poisson equation $\Delta u = f$, with Neumann boundary conditions. Consider the following forcing f and corresponding solutions u, in one, two and three dimensions respectively:

$$\begin{aligned} u(x) &= \cos(x\pi), & f(x) &= -\pi^2 \cos(x\pi), \\ u(x,y) &= \cos(x\pi)\cos(y\pi), & f(x,y) &= -2\pi^2 \cos(x\pi)\cos(y\pi), \\ u(x,y,z) &= \cos(x\pi)\cos(y\pi)\cos(z\pi), & f(x,y,z) &= -3\pi^2 \cos(x\pi)\cos(y\pi)\cos(z\pi). \end{aligned} \tag{14.1}$$

We observe that the forcings f satisfy the compatibility condition (6.8).

In general, the error can be defined as the norm of the difference between the approximate solution u_N and analytical solution u as in (14.1),

$$\text{error} = \left(\sqrt{N}\right)^{-1} \left|\left| u_N - u \right|\right|_2 \tag{14.2}$$

Figure 14.1 suggests that the convergence of the approximate solution is exponential in one, two and three dimensions until it hits machine precision around 10^{-15}.

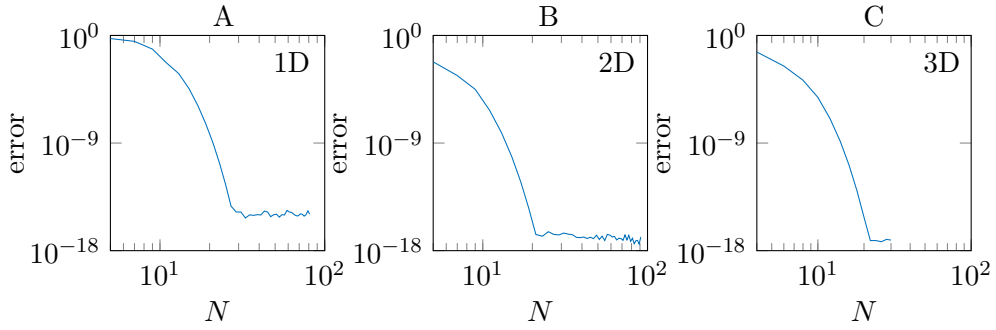

Figure 14.1: Convergence u_N to u in 1D (A), 2D (B) and 3D (C). The resolution N indicates the number of Chebyshev-Gauss-Lobatto points in one direction. So, in 2D u represents an $N \times N$ tensor and in 3D u represents an $N \times N \times N$ tensor. The error is defined in (14.2). The settings for these results can be found in Appendix G.4.

Part IV
Discussion and conclusions

15 Discussion

In this section we discuss the Chebyshev-Galerkin (CG) method, in particular its suitability for the inertial wave application.

15.1 Comparison with the laboratory set-up

Let's highlight some of the advantages and disadvantages of using a Chebyshev-Galerkin method to simulate the rotating-box experiment. The method is developed for rectangular domains and is thus applicable to the right-angled tank. A Chebyshev-Gauss-Lobatto quadrature places more grid points near the boundaries. Therefore, the boundary layer, which plays an important role in viscous fluids, is better resolved.

In contrast to the laboratory, where an adjustable voltage determines the pump rate, and to the Gerris simulations, in which a pressure gradient drives the flow, the Chebyshev-Galerkin method needs specified velocity inflow and outflow. As the solid walls have no-slip conditions, the boundary conditions at the entrance and exit cannot be Neumann type, but must be a prescribed velocity (for continuity of the boundary condition (6.7) at the edges of the entrance/exit). The in- and outflow is approximated as Hagen-Poisseuille pipe flow. To compare CG simulations with laboratory parameters, one could calibrate the resulting entrance-exit pressure gradient to a pump-voltage, using the flow rate in a stationary flat experiment as the calibration control.

Other boundary conditions might be possible when implementing domain decomposition, in which the domain is subdivided into multiple domains. One can pose the boundary conditions at the outer boundaries for each subdomain independently, allowing for a prescribed pressure-difference between the entrance and exit tubes. The solutions in the subdomains must match at the division surfaces, which adds additional constraints.

15.2 Numerical method

Because the derivatives in the Chebyshev-Galerkin framework are computed analytically, even low resolutions yield high accuracy. The convergence test in section 14 show exponential converge to the analytical solution with in-

creasing resolution. For a simple Poisson equation the accuracy reaches machine precision for a grid resolution of $N = 30$ points.

In the current method, the continuity equation is automatically satisfied. This is a huge advantage with respect to the Gerris simulations which could not guaranty continuity. Rodda [20] reported a flux variation of around 10%, indicating significant divergence of the flow.

15.3 Time-marching methods

We implemented the pressure-correction time-stepping method in Matlab, including the inner-product formulation of the pressure-correction step, equation (5.4). Upon spatially discretising (5.4) we find the discrete Laplace operator. In one dimension, this is the stiffness matrix (8.18) for Neumann boundary conditions, which is rank deficient. As a result, not every forcing f has solutions $\tilde{q}$. The forcing should satisfy the compatibility constraint (6.8). When $f = \nabla \cdot \tilde{u}$ and when $\tilde{u}$ has Dirichlet boundary conditions, the compatibility constraint is satisfied and solutions should exist. In one dimension we find accurate solutions of the discrete Poisson equation for every such forcing.

For the higher dimensional discrete Laplace operator, we do not seem to be able to find accurate solutions for all $f = \nabla \cdot \tilde{u}$ with $\tilde{u}$ satisfying Dirichlet boundary conditions. For simple examples, such as the forcing f in (14.1), whose analytical solutions we can derive, accurate solutions are found. However, for randomly generated $\{u_n\}$ with Dirichlet boundary conditions, not all approximations of q returned by the iterative solver actually solve equation (5.4).

This issue is specific to Neumann boundary conditions. The Stokes time-stepping method does not employ Neumann boundary conditions. Therefore, we think that with this scheme simulations are stable. Additionally, one may solve the discrete Stokes equations (5.6)-(5.7) with a fractional step method (as a Lagrange Multiplier method). This is based on first discretising in space and then in time, thereby avoiding Neumann boundary conditions.

15.4 Inviscid limit

The first priority was to simulate viscous Navier-Stokes equations, because viscosity is significant in the small laboratory box. For the inviscid limit let the inverse of the Reynolds number R_e^{-1} go to 0. The full Navier-Stokes equations cannot exploit the structure of the inviscid limit. For inviscid

Navier-Stokes equations, analytical reduction is possible and could be preferred above just putting the inverse Reynolds number to zero in the full equations.

For example in two dimensions the system can be written in terms of a stream function eliminating the continuity equation. Three dimensional inviscid flow in a parallelepiped is described in Maas [13]. Without viscosity the two-dimensional governing equations can be rewritten as a Poincare equation with Robin (mixed) boundary conditions. One can construct basis functions for these boundary conditions and find expressions for the inner-product matrices. So, the Poincare system can be investigated with a Chebyshev-Galerkin method, which might be more efficient than the current eigenvalue problem approach [13].

15.5 Trapezoidal domains

The Chebyshev-Galerkin method has domain restrictions. The current approach only works for rectangular domains. It seems natural that with an appropriate mapping one could simulate trapezoidal domains as well. In that case, the equations in transformed coordinates can have cross partial derivatives (such as $\partial^2/(\partial x\,\partial y)$), which negates the mainly block diagonal structure of the matrix $A^{(3)}$. Trapezoidal domains will involve more bookkeeping and additional inner-product matrices.

For separability it seems crucial that the domain has corners to be mapped to the corners of a rectangular domain, because the singularity at the corners is exploited. Circular domains lack corners and applying the presented CG approach seems impossible. Alternatively, spherical or cylindrical domains in polar- or cylindrical coordinates could be represented by Chebyshev-type basis functions in the bounded directions and Fourier basis functions in the periodic directions.

16 Conclusions

Rotating systems can have complex dynamics. Laboratory experiments have shown that the flow through a rotating tank is enhanced when the domain has tilted walls. It is expected that wave-attractors of inertial waves inhibit turbulence and thus aid the mean flow. To study the interaction between inertial waves and currents in more detail, we propose a numerical model for the three-dimensional viscous Navier-Stokes equations under rotation.

In this project we developed a Chebyshev-Galerkin method to find weak solutions for second-order partial differential equations numerically. Starting from Chebyshev polynomials, basis functions satisfying homogeneous boundary conditions were constructed and their inner product matrices were derived analytically. Because of the upper-triangular and/or banded structure, these mass, first-derivative and stiffness matrices possess linear complexity. For the Poisson equation under Neumann and under Dirichlet boundary conditions, we proposed a diagonal pre-conditioner that reduces the growth rate of the condition number to sublinear order. Finally, the accuracy was shown to be exponential. To conclude, the Chebyshev-Galerkin method is near-optimal for a Poisson equation.

We consider the Navier-Stokes equations as an application for the CG method. To simulate the time-dependent behaviour of the flow, an implicit-explicit Stokes method can be employed for time-marching. Inflow and outflow at the boundary could be approximated as Hagen-Poisseuille flow. The non-homogeneous boundary conditions at the entrance and exit can be lifted and result in an additional forcing.

Several advantages of the Chebyshev-Galerkin method include guaranteed incompressibility, naturally more grid points in the boundary layers and fast transforms between point evaluations and expansion coefficients. Some disadvantages are the limited domain flexibility, and the Hagen-Poisseuille pipe flow approximation. In short, the Chebyshev-Galerkin framework is an elegant method. Further development could extend the method for more applications and geometries.

Part V
Appendix

A Backward differentiation formula

Consider the time-derivative $w'(t) = \partial_t w(t)$. The two-step backward differentiation (BDF) is an implicit time-derivative scheme. Its derivation uses linear extrapolation and the midpoint method,

$$\begin{aligned}
w'\left(t_{n+1}\right) \approx & \frac{1}{2}\left(w'\left(t_{n+1}+\tau/2\right)+w'\left(t_{n+1}-\tau/2\right)\right), \\
\approx & \frac{1}{2}\left(w'\left(t_{n+1}+\tau/2\right)+\left(\frac{w_{n+1}-w_n}{\tau}\right)\right), \\
\approx & \frac{1}{2}\left(\left(2w'(t_n+\tau/2)-w'(t_{n-1}+\tau/2)\right)+\left(\frac{w_{n+1}-w_n}{\tau}\right)\right), \\
\approx & \frac{1}{2}\left(2\left(\frac{w_{n+1}-w_n}{\tau}\right)-\left(\frac{w_n-w_{n-1}}{\tau}\right)+\left(\frac{w_{n+1}-w_n}{\tau}\right)\right), \\
\approx & \frac{3w_{n+1}-4w_n+w_{n-1}}{2\tau}.
\end{aligned}$$

□

B Boundary conditions

Consider problem (6.1) with non-homogeneous boundary conditions (6.6). Take some $v(x)$ such that the boundary conditions are satisfied,

$$a_\pm v + b_\pm \frac{\partial}{\partial n} v = g.$$

Note that v does not need to satisfy the problem (6.1). Assume there exists a solution u that solves the problem and satisfies the boundary conditions. Then split $u = \bar{u} + v$. Because v already satisfies (6.6), $\bar{u}$ only needs to satisfy the homogeneous boundary conditions (6.7), given by

$$a_\pm \bar{u} + b_\pm \frac{\partial}{\partial n} \bar{u} = 0 \tag{B.1}$$

Substitute $u = \bar{u} + v$ into the problem (6.1), so

$$\mathcal{L}\bar{u}(x) = f(x) - \mathcal{L}v(x), \quad x \in \Omega. \tag{B.2}$$

Together (B.2) and (B.1) form a problem for $\bar{u}$ with homogeneous boundary conditions.

C Chebyshev

C.1 Quadrature

Consider the Chebyshev polynomials (7.1) at the quadrature points (7.7),

$$\begin{aligned} x_j &= -\cos\left(\frac{\pi j}{N}\right) = \cos\left(\frac{\pi(N-j)}{N}\right), \\ \theta_j &= \arccos(x_j) = \frac{\pi(N-j)}{N}, \\ T_n(x_j) &= \cos(n\theta_j) = (-1)^n \cos\left(\frac{\pi n j}{N}\right). \end{aligned}$$

This proves (7.9). □

C.2 Discrete Chebyshev Transform

Substitution of basis function (7.9) into the Chebyshev approximation (7.11) gives the backwards Chebyshev transform (7.13),

$$u(x_j) = \sum_{n=0}^{N} \hat{u}_n T_n(x_j) = \sum_{n=0}^{N} \hat{u}_n (-1)^n \cos\left(\frac{\pi n j}{N}\right).$$

One can derive the forward transform (7.12) from the backwards transform (7.13), using the discrete inner product (7.10),

$$\begin{aligned} u(x_j) &= \sum_{n=0}^{N} \hat{u}_n T_n(x_j), \\ \sum_{j=0}^{N} \frac{1}{\tilde{c}_j} u(x_j) T_m(x_j) &= \sum_{n=0}^{N} \hat{u}_n \sum_{j=0}^{N} \frac{1}{\tilde{c}_j} T_m(x_j) T_n(x_j), \\ \sum_{j=0}^{N} \frac{1}{\tilde{c}_j} u(x_j) T_m(x_j) &= \sum_{n=0}^{N} \hat{u}_n \frac{\tilde{c}_m N}{2} \delta_{mn}, \\ \sum_{j=0}^{N} \frac{1}{\tilde{c}_j} u(x_j) T_m(x_j) &= \hat{u}_m \frac{\tilde{c}_m N}{2}, \\ \frac{2}{\tilde{c}_m N} \sum_{j=0}^{N} \frac{1}{\tilde{c}_j} u(x_j) T_m(x_j) &= \hat{u}_m. \end{aligned}$$

□

C.3 Fast Chebyshev Transform

The discrete Chebyshev transform can be computed in $O(N \log N)$ operations because of its close relation to the discrete cosine transform and the fast Fourier transform.

C.3.1 Fourier transform

The discrete Fourier transform is given by,

$$u_j = \sum_{k=0}^{N-1} \tilde{u}(k) e^{-i2\pi jk/N}, \tag{C.1}$$

with inverse,

$$\tilde{u}(k) = \frac{1}{N} \sum_{j=0}^{N-1} u_j \, e^{i2\pi jk/N}. \tag{C.2}$$

Substitute the trigonometric identity $e^{i\theta} = \cos(\theta) + i \sin(\theta)$ into (C.2) and take the real part. So,

$$\Re(\tilde{u}(k)) = \frac{1}{N} \sum_{j=0}^{N-1} \Re(u_j) \cos\left(\frac{2\pi jk}{N}\right) + \Im(u_j) \sin\left(\frac{2\pi jk}{N}\right).$$

For purely real u_j, this simplifies to

$$\Re(\tilde{u}(k)) = \frac{1}{N} \sum_{j=0}^{N-1} u_j \cos\left(\frac{2\pi jk}{N}\right).$$

Denote $C_n(x_j) := \cos(2\pi nj/N)$. It holds that

$$C_n(x_j) = C_{N-n}(x_j), \quad \text{for all } n \in \{0, 1, \ldots, N-1\} \tag{C.3}$$

Consider taking the inverse transform of a function that is twice as long ($M = 2N$) and even. First using (C.3), and next using $u_j = u_{M-j}$, because

the extension is even, gives

$$\begin{aligned}
\Re(\tilde{u}(k)) =& \frac{1}{M}\sum_{j=0}^{M-1} u_j \cos\left(\frac{2\pi jk}{M}\right), \\
\Re(\tilde{u}(k)) =& \frac{1}{M}\left[u_0 + \sum_{j=1}^{N-1}(u_j + u_{M-j})\cos\left(\frac{2\pi jk}{M}\right) + u_N\cos(\pi k)\right], \\
\Re(\tilde{u}(k)) =& \frac{1}{2N}\left[u_0 + \sum_{j=1}^{N-1} 2u_j \cos\left(\frac{2\pi jk}{2N}\right) + u_N\cos(\pi k)\right], \\
\Re(\tilde{u}(k)) =& \frac{1}{N}\sum_{j=0}^{N}\frac{1}{\tilde{c}_j} u_j \cos\left(\frac{\pi jk}{N}\right),
\end{aligned}$$

where $\tilde{c}_0 = \tilde{c}_N = 2$ and $\tilde{c}_{1\le j\le N-1} = 1$. It follows that

$$(-1)^k \Re(\tilde{u}(k)) = \frac{1}{N}\sum_{j=0}^{N}\frac{1}{\tilde{c}_j} u_j (-1)^k \cos\left(\frac{\pi jk}{N}\right).$$

C.3.2 Left and right extensions

Let us consider a function f on $[a, b]$ and its even extension to the left, f_{ext} on $[2a - b, b]$ given by:

$$f_{\text{left-ext}}(x) := \begin{cases} f(x), & a \le x \le b, \\ f(2a - x), & 2a - b \le x \le a. \end{cases}$$

Note that the left extension is different from the right extension of the previous section. The left extension is related to the right extension via $\tilde{f}_{\text{right-ext}}(k) = (-1)^k \tilde{f}_{\text{left-ext}}(k)$, because for all odd k the $\cos\left(\frac{\pi jk}{N}\right)$ are odd on the interval $0 \le j \le N$. In short, the left extension is the original function on the domain $N \le j \le 2N$, where the values of $\cos\left(\frac{\pi jk}{N}\right)$ are opposite. As a result,

$$\begin{aligned}
\Re(\tilde{f}_{\text{left-ext}}(k)) =& (-1)^k \Re(\tilde{f}_{\text{right-ext}}(k)), \\
=& \frac{1}{N}\sum_{j=0}^{N}\frac{1}{\tilde{c}_j} f_{\text{right-ext},j}(-1)^k \cos\left(\frac{\pi jk}{N}\right). \qquad \text{(C.4)}
\end{aligned}$$

C.3.3 Inverse Fourier and forward Chebyshev transforms

The Chebyshev transform (7.12),

$$\hat{u}_n = \frac{2}{c_n N} \sum_{j=0}^{N} \frac{1}{\tilde{c}_j} u(x_j)(-1)^n \cos\left(\frac{nj\pi}{N}\right),$$

is related to the Fourier expansion expression (C.4), as

$$\hat{u}_n = \frac{2}{c_n} \Re(\tilde{u}_{\text{left-ext}}(n)),$$

where the coefficients $\tilde{u}_{\text{left-ext}}(n)$ can be determined from $u_{\text{left-ext}}(x_j)$ with the inverse fast Fourier transform (iFFT).

C.3.4 Backward Chebyshev Transform

Analogously, the backward Chebyshev transform (7.13) can be related to the forward Fourier transform. Denote with $\bar{u}$ a left extension of the expansions coefficients,

$$\bar{u}_n = \begin{cases} (1/2)\, c_n\, (-1)^n\, \hat{u}_n, & 0 \le n \le N, \\ (1/2)\, c_n\, (-1)^n\, \hat{u}_{2N-n}, & N+1 \le n \le 2N-1. \end{cases}$$

Substitution into backward transform (7.13), and taking $M = 2N$ gives

$$u(x_j) = \sum_{n=0}^{N} \frac{2}{c_n} \bar{u}_n \cos\left(\frac{\pi n j}{N}\right) = \sum_{n=0}^{M-1} \bar{u}_n \cos\left(\frac{2\pi n j}{M}\right).$$

This expression is identical to the discrete Fourier transform (C.1) of $\bar{u}$. The backward Chebyshev transform can be performed in $O(N \log N)$ operations, through computation of $\bar{u}$, executing a fast Fourier transform and then taking the real part of the right half.

C.4 Transform to superposition

The CD coefficients $\tilde{u}_n$ of (7.17) are related to the Chebyshev coefficients[1] $\hat{u}_n$, through the recursion relations

$$\begin{aligned} \hat{u}_0 &= \tilde{u}_0, \\ \hat{u}_1 &= \tilde{u}_1, \\ \hat{u}_n &= \tilde{u}_n - \tilde{u}_{n-2}, \qquad \forall n \in [2, N-2], \\ \hat{u}_{N-1} &= -\tilde{u}_{N-3}, \\ \hat{u}_N &= -\tilde{u}_{N-2}. \end{aligned}$$

These transformations cost $O(N)$ operations.

In addition, the recursion relations between the Chebyshev coefficients $\hat{u}_n$ of a functions with Neumann boundary conditions and its CN coefficients $\tilde{u}_n$ are

$$\begin{aligned}
\hat{u}_0 &= \tilde{u}_0, \\
\hat{u}_1 &= \tilde{u}_1, \\
\hat{u}_n &= \tilde{u}_n - \frac{(n-2)^2}{n^2}\tilde{u}_{n-2}, \qquad \forall n \in [2, N-2], \\
\hat{u}_{N-1} &= -\frac{(N-3)^2}{(N-1)^2}\tilde{u}_{N-3}, \\
\hat{u}_N &= -\frac{(N-2)^2}{N^2}\tilde{u}_{N-2}.
\end{aligned}$$

The above recursion relations can also be performed in $O(N)$ computations.

[1] Of course, this only holds for Chebyshev coefficients representing a function that actually has Dirichlet boundary conditions.

D Inner product matrices

D.1 Chebyshev

The mass matrix (8.4) for Chebyshev basis functions is just the weighted inner product (7.2)

$$m_{mn} = (T_n(x), T_m(x))_\omega = \int_{-1}^{1} T_n(x) T_m(x) \frac{1}{\sqrt{1-x^2}} \, \mathrm{d}x = \frac{c_n \pi}{2} \delta_{mn}.$$

The first derivatives are expressed in property (7.5) as a sum of lower order polynomials:

$$T_n'(x) = 2n \sum_{\substack{k=0 \\ k+n \, odd}}^{n-1} \frac{1}{c_k} T_k(x).$$

Substitution of the property into the definition of K (8.2) yields

$$\begin{aligned} k_{mn} =& (T_n'(x), T_m(x))_\omega, \\ =& 2n \sum_{\substack{k=0 \\ k+n \text{ odd}}}^{n-1} \frac{1}{c_k} (T_k(x), T_m(x))_\omega, \\ =& \pi n \sum_{\substack{k=0 \\ k+n \text{ odd}}}^{n-1} \delta_{km}, \\ k_{mn} =& \begin{cases} \pi n, & n = m+1, m+3, m+5, \ldots, \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

The last result proves expression (8.5). □

For the stiffness matrix, we use property (7.6), given by

$$T_n''(x) = \sum_{\substack{k=0 \\ k+n \, even}}^{n-2} \frac{1}{c_k} n(n^2 - k^2) T_k(x).$$

Substitution of the property into definition (8.3) gives

$$\begin{aligned}
s_{mn} =& (T_n''(x), T_m(x))_\omega, \\
=& \sum_{\substack{k=0 \\ k+n\ even}}^{n-2} \frac{1}{c_k} n(n^2-k^2)(T_k(x), T_m(x))_\omega, \\
=& \frac{\pi}{2} \sum_{\substack{k=0 \\ k+n\ even}}^{n-2} n(n^2-k^2)\delta_{km}, \\
s_{mn} =& \begin{cases} \frac{\pi}{2} n(n^2-m^2), & n = m+2, m+4, m+6, \ldots, \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}$$

This proves (8.6). □

D.2 General boundary conditions

Assume any mixed boundary condition of type (6.7). The choice fixes the coefficients $\{a_n, b_n\}$ of superposition (7.14),

$$\phi_n(x) = T_n(x) + a_n T_{n+1}(x) + b_n T_{n+2}(x).$$

D.2.1 Mass matrix

The elements of the mass matrices are

$$\begin{aligned}
m^\phi_{mn} =& (\phi_n(x), \phi_m(x))_\omega, \\
=& m_{mn} + a_n m_{m,n+1} + b_n m_{m,n+2} \\
&+ a_m m_{m+1,n} + a_m a_n m_{m+1,n+1} + a_m b_n m_{m+1,n+2} \\
&+ b_m m_{m+2,n} + b_m a_n m_{m+2,n+1} + b_m b_n m_{m+2,m+2}, \\
m^\phi_{mn} =& m^\phi_{nm} = \begin{cases} \frac{\pi}{2}\left(c_m + a_m^2 + b_m^2\right) & n = m, \\ \frac{\pi}{2}(a_m + a_{m+1} b_m), & n = m+1, \\ \frac{\pi}{2}(a_n + a_{n+1} b_n), & n = m-1, \\ \frac{\pi}{2} b_m, & n = m+2, \\ \frac{\pi}{2} b_n, & n = m-2, \\ 0, & \text{Otherwise,} \end{cases}
\end{aligned}$$

with m_{mn} according to (8.4), a_n, b_n as in (7.14) and still $c_0 = 2$ and $c_n = 0$ for all $n \geq 1$.

D.2.2 First derivative matrix

The first derivative matrix has elements

$$\begin{aligned} k^{\phi}_{mn} =&(\phi'_n(x), \phi_m(x))_\omega, \\ =&k_{mn} + a_n k_{m,n+1} + b_n k_{m,n+2} \\ &+ a_m k_{m+1,n} + a_m a_n k_{m+1,n+1} + a_m b_n k_{m+1,n+2} \\ &+ b_m k_{m+2,n} + b_m a_n k_{m+2,n+1} + b_m b_n k_{m+2,m+2}, \end{aligned}$$

with k_{mn} as in (8.5).

D.2.3 Stiffness matrix

Analogously,

$$\begin{aligned} s^{\phi}_{mn} =&(\phi''_n(x), \phi_m(x))_\omega, \\ =&s_{mn} + a_n s_{m,n+1} + b_n s_{m,n+2} \\ &+ a_m s_{m+1,n} + a_m a_n s_{m+1,n+1} + a_m b_n s_{m+1,n+2} \\ &+ b_m s_{m+2,n} + b_m a_n s_{m+2,n+1} + b_m b_n s_{m+2,m+2}. \end{aligned}$$

D.2.4 Transformation matrix

The inner product with Chebyshev polynomials is

$$\begin{aligned} r^{\phi}_{mn} =&(T_n(x), \phi_m(x))_\omega, \\ =&m_{mn} + a_m m_{m+1,n} + b_m m_{m+2,n}, \\ r^{\phi}_{mn} =& \begin{cases} \frac{\pi}{2} c_m, & n = m, \\ a_m \frac{\pi}{2}, & n = m+1, \\ b_m \frac{\pi}{2}, & n = m+2, \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

Next, we consider the special cases of Dirichlet and Neumann boundary conditions.

D.3 Dirichlet

Suppose, we have Dirichlet boundary conditions and employ CD basis functions (7.16) given by

$$\phi_n(x) = T_n(x) - T_{n-2}(x).$$

D.3.1 Mass matrix

For Dirichlet boundary conditions we obtain,

$$\begin{aligned} m^D_{mn} =&(\phi_n(x), \phi_m(x))_\omega, \\ =&m_{mn} - m_{m,n+2} - m_{m+2,n} + m_{m+2,m+2}, \\ m^D_{mn} =& \begin{cases} \frac{(c_m+1)\pi}{2}, & n = m, \\ -\frac{\pi}{2}, & n = m-2 \text{ and } n = m+2, \\ 0, & \text{otherwise}, \end{cases} \end{aligned}$$

where $c_0 = 2$ and other $c_n = 1$.

D.3.2 First derivative matrix

Property (7.5), given by

$$T'_n(x) = 2n \sum_{\substack{k=0 \\ k+n\ odd}}^{n-1} \frac{1}{c_k} T_k(x),$$

can be adapted for Chebyshev-Dirichlet basis functions (7.16),

$$\begin{aligned} \phi'_n =&2n \sum_{\substack{k=0 \\ k+n\ odd}}^{n-1} \frac{1}{c_k} T_k(x) - 2(n+2) \sum_{\substack{k=0 \\ k+n\ odd}}^{n+1} \frac{1}{c_k} T_k(x), \\ =&-4 \sum_{\substack{k=0 \\ k+n\ odd}}^{n-1} \frac{1}{c_k} T_k(x) - 2(n+2)\frac{1}{c_{n+1}} T_{n+1}(x), \\ =& \sum_{\substack{k=0 \\ k+n\ odd}}^{n+1} \gamma_{n,k} T_k(x), \end{aligned}$$

where

$$\gamma_{n,k} = \begin{cases} -2(n+2)\frac{1}{c_k}, & k = n+1, \\ -\frac{4}{c_k}, & k = n-1, n-3, \ldots. \end{cases}$$

Thus,

$$(\phi'_n, \phi_m)_\omega = \sum_{\substack{k=0 \\ k+n\ odd}}^{n+1} \gamma_{n,k}(T_k(x), T_m(x))_\omega - \sum_{\substack{k=0 \\ k+n\ odd}}^{n+1} \gamma_{n,k}(T_k(x), T_{m+2}(x))_\omega,$$

$$k^d_{mn} = \begin{cases} -(m+1)\pi, & n = m-1, \\ (m+1)\pi, & n = m+1, \\ 0, & \text{otherwise}. \end{cases}$$

The first-derivative matrix is thus a double banded matrix.

D.3.3 Stiffness matrix

In addition, we require an expression for the stiffness matrix. Denoting $\alpha_{n,k} = \frac{1}{c_k} n(n^2 - k^2)$, reduces property (7.6) to

$$T_n''(x) = \sum_{\substack{k=0 \\ k+n\ even}}^{n-2} \alpha_{n,k} T_k(x).$$

So,

$$\begin{aligned} T_n''(x) - T_{n+2}''(x) &= \sum_{\substack{k=0 \\ k+n\ even}}^{n-2} \alpha_{n,k} T_k(x) - \sum_{\substack{k=0 \\ k+n\ even}}^{n} \alpha_{n+2,k} T_k(x), \\ &= -\alpha_{n+2,n} T_n(x) + \sum_{\substack{k=0 \\ k+n\ even}}^{n-2} (\alpha_{n,k} - \alpha_{n+2,k}) T_k(x), \\ &= \sum_{\substack{k=0 \\ k+n\ even}}^{n} d_{n,k} T_k(x), \end{aligned} \tag{D.1}$$

where

$$d_{n,k} = \begin{cases} -\frac{1}{c_n} 4(n+1)(n+2), & k = n, \\ \frac{1}{c_k}(n^3 - (n+2)^3 + 2k^2), & k < n. \end{cases}$$

Substitute relation (D.1) in the stiffness matrix

$$\begin{aligned} s_{mn}^D =& (\phi_n''(x), T_m(x))_\omega - (\phi_n''(x), T_{m+2}(x))_\omega, \\ =& \sum_{\substack{k=0 \\ k+n\ even}}^{n} d_{n,k} (T_k(x), T_m(x))_\omega - \sum_{\substack{k=0 \\ k+n\ even}}^{n} d_{n,k} (T_k(x), T_{m+2}(x))_\omega \end{aligned} \tag{D.2}$$

With property (7.2), only certain elements of S^D are nonzero. When $m > n$ or when $n + m$ is odd, $(\phi_n''(x), \phi_m(x))_\omega = 0$.

Additionally, when $n = m$, only the first sum in (D.2) is nonzero:

$$\begin{aligned} (\phi_n''(x), \phi_n(x))_\omega =& d_{n,n} (T_n(x), T_n(x))_\omega, \\ =& -\frac{1}{c_n} 4(n+1)(n+2) \frac{c_n \pi}{2}, \\ =& -2\pi(n+1)(n+2). \end{aligned}$$

For $n = m+2, m+4, m+6, ...$, relation (D.2) yields

$$\begin{aligned}(\phi_n''(x), \phi_m(x))_\omega = &\sum_{\substack{k=0 \\ k+n\ even}}^{n} d_{n,k}(T_k(x), T_m(x))_\omega - \sum_{\substack{k=0 \\ k+n\ even}}^{n} d_{n,k}(T_k(x), T_{m+2}(x))_\omega, \\ &= d_{n,m}(T_m(x), T_m(x))_\omega - d_{n,m+2}(T_{m+2}(x), T_{m+2}(x))_\omega, \\ &= \frac{1}{c_m}(n^3 - (n+2)^3 + 2m^2)\frac{c_m\pi}{2} - \frac{1}{c_{m+2}}(n^3 - (n+2)^3 + 2(m+2)^2)\frac{c_{m+2}\pi}{2}, \\ &= (2m^2 - 2(m+2)^2)\frac{\pi}{2}, \\ &= -4\pi(m+1).\end{aligned}$$

As a result, the stiffness matrix is given by

$$s_{mn}^D = \begin{cases} -2\pi(m+1)(m+2), & n = m, \\ -4\pi(m+1), & n = m+2, m+4, \ldots, \\ 0, & n < m \text{ or } n+m \text{ odd}, \end{cases}$$

which is presented in expression (8.14). This matrix is checkerboard upper triangular.

D.3.4 Transformation matrix

Substituting the Chebyshev-Dirichlet basis functions (7.16) into the definition of the transformation matrix (8.10) gives

$$\begin{aligned} r_{mn}^D =& (T_n(x), \phi_m(x))_\omega, \\ =& \begin{cases} \frac{c_n\pi}{2}, & n = m, \\ -\frac{\pi}{2}, & n = m+2, \\ 0, & \text{otherwise}. \end{cases} \end{aligned}$$

D.4 Neumann

Assuming Neumann boundary conditions, the basis functions (7.18) are

$$\phi_n(x) = T_n(x) - \frac{n^2}{(n+2)^2}T_{n+2}(x).$$

Define $\beta_n := \frac{n^2}{(n+2)^2}$.

D.4.1 Mass matrix

We build the mass matrix as

$$\begin{aligned}(\phi_n(x),\phi_m(x))_\omega =&(T_n(x),T_m(x))_\omega - \beta_n(T_{n+2}(x),T_m(x))_\omega \\ &- \beta_m(T_n(x),T_{m+2}(x))_\omega + \beta_n\beta_m(T_{n+2}(x),T_{m+2}(x))_\omega, \\ =&\frac{c_n\pi}{2}\delta_{n,m} + \beta_m\beta_n\frac{\pi}{2}\delta_{n+2,m+2} - \beta_n\frac{\pi}{2}\delta_{n+2,m} - \beta_m\frac{\pi}{2}\delta_{n,m+2},\end{aligned}$$

where still $c_0 = 2$ and $c_n = 1$ otherwise. As a result,

$$m^N_{mn} = m^N_{nm} = \begin{cases} \frac{\pi}{2}\left(c_m + \frac{m^4}{(m+2)^4}\right), & n = m, \\ -\frac{\pi}{2}\frac{m^2}{(m+2)^2}, & n = m+2, \\ -\frac{\pi}{2}\frac{n^2}{(n+2)^2}, & n = m-2, \\ 0, & \text{otherwise,} \end{cases}$$

where $c_0 = 2$ and $c_m = 1$ for all $m \geq 1$.

D.4.2 First derivative matrix

Also, for Neumann boundary conditions, the first derivative matrix K^N is aided by property (7.5). Denote $\beta_n = \left(\frac{n}{n+2}\right)^2$, then

$$\begin{aligned}\phi_n' =&2n \sum_{\substack{k=0 \\ k+n\ odd}}^{n-1} \frac{1}{c_k}T_k(x) - \beta_n 2(n+2) \sum_{\substack{k=0 \\ k+n\ odd}}^{n+1} \frac{1}{c_k}T_k(x), \\ =& \sum_{\substack{k=0 \\ k+n\ odd}}^{n+1} \gamma_{n,k}T_k(x).\end{aligned}$$

with

$$\gamma_{n,k} = \begin{cases} -2\frac{n^2}{n+2}\frac{1}{c_k}, & k = n+1, \\ \frac{4n}{n+2}\frac{1}{c_k}, & k = n-1, n-3, \ldots, \\ 0, & \text{otherwise.} \end{cases}$$

Hence,

$$(\phi_n', \phi_m)_\omega = \sum_{\substack{k=0 \\ k+n\ odd}}^{n+1} \gamma_{n,k}(T_k(x), T_m(x))_\omega - \sum_{\substack{k=0 \\ k+n\ odd}}^{n+1} \gamma_{n,k}(T_k(x), T_{m+2}(x))_\omega,$$

$$= \begin{cases} -\frac{\pi n^2}{n+2}, & n = m-1, \\ \frac{\pi n}{n+2}\left(2 - n\beta_m\right), & n = m+1, \\ \frac{2\pi n}{n+2}\left(1 - \beta_m\right), & n = m+3, m+5, m+7, \ldots, \\ 0, & \text{otherwise.} \end{cases}$$

D.4.3 Stiffness matrix

For the stiffness matrix, consult property (7.6) with $\alpha_{n,k} = \frac{1}{c_k} n(n^2 - k^2)$,

$$T_n''(x) = \sum_{\substack{k=0 \\ k+n\ even}}^{n-2} \alpha_{n,k} T_k(x).$$

Then

$$T_n''(x) - \frac{n^2}{(n+2)^2} T_{n+2}''(x) = \sum_{\substack{k=0 \\ k+n\ even}}^{n-2} \alpha_{n,k} T_k(x) - \frac{n^2}{(n+2)^2} \sum_{\substack{k=0 \\ k+n\ even}}^{n} \alpha_{n+2,k} T_k(x),$$

$$= -\frac{n^2}{(n+2)^2} \alpha_{n+2,n} T_n(x) + \sum_{\substack{k=0 \\ k+n\ even}}^{n-2} (\alpha_{n,k} - \frac{n^2}{(n+2)^2} \alpha_{n+2,k}) T_k(x),$$

$$= \sum_{\substack{k=0 \\ k+n\ even}}^{n} d_{n,k} T_k(x), \qquad \text{(D.3)}$$

with

$$d_{n,k} = \begin{cases} -\frac{4n^2}{c_n} \frac{n+1}{n+2}, & k = n, \\ -\frac{1}{c_k} \frac{1}{n+2} n(2n^2 + 4n - 2k^2), & k < n. \end{cases}$$

Combine relation (D.3) with the stiffness matrix (7.18),

$$(\phi_n''(x), \phi_m(x))_\omega = (\phi_n''(x), T_m(x))_\omega - \frac{m^2}{(m+2)^2} (\phi_n''(x), T_{m+2}(x))_\omega,$$

$$= \sum_{\substack{k=0 \\ k+n\ even}}^{n} d_{n,k} (T_k(x), T_m(x))_\omega - \frac{m^2}{(m+2)^2} \sum_{\substack{k=0 \\ k+n\ even}}^{n} d_{n,k} (T_k(x), T_{m+2}(x))_\omega.$$

Because Chebyshev polynomials are orthogonal with respect to the weight function, many elements are zero. For example, when $m > n$ or when $n+m$ odd, $(\phi_n''(x), \phi_m(x))_\omega = 0$. Suppose $n = m$, then

$$\begin{aligned}(\phi_n''(x), \phi_n(x))_\omega =& d_{n,n}(T_n(x), T_n(x))_\omega, \\ =& -\frac{4n^2}{c_n}\frac{n+1}{n+2}\frac{c_n\pi}{2}, \\ =& -2\pi n^2\frac{n+1}{n+2}.\end{aligned}$$

Moreover, when $n = m+2, m+4, m+6, ...$, then

$$\begin{aligned}(\phi_n''(x), \phi_m(x))_\omega =& \sum_{\substack{k=0 \\ k+n\ even}}^{n} d_{n,k}(T_k(x), T_m)_\omega - \frac{m^2}{(m+2)^2}\sum_{\substack{k=0 \\ k+n\ even}}^{n} d_{n,k}(T_k(x), T_{m+2}(x))_\omega, \\ =& d_{n,m}(T_m(x), T_m(x))_\omega - \frac{m^2}{(m+2)^2}d_{n,m+2}(T_{m+2}(x), T_{m+2}(x))_\omega, \\ =& -\frac{1}{n+2}(2n^3+4n^2-2m^2n)\frac{\pi}{2} + \frac{m^2}{(m+2)^2}\frac{1}{n+2}(2n^3+4n^2-2(m+2)^2n)\frac{\pi}{2}, \\ =& -\frac{\pi}{2}\frac{1}{n+2}\frac{1}{m+2}\Big(\big((m+2)^2-m^2\big)(2n^3-4n^2)\Big), \\ =& -4\pi n^2\frac{m+1}{(m+2)^2}.\end{aligned}$$

As a result,

$$s_{mn}^N = \begin{cases} -2\pi m^2\frac{m+1}{m+2}, & n = m, \\ -4\pi n^2\frac{m+1}{(m+2)^2}, & n = m+2, m+4, m+6..., \\ 0, & n < m \text{ or } n+m \text{ odd}. \end{cases}$$

D.4.4 Transformation matrix

For Neumann boundary conditions, the transformation matrix can be expressed with $\beta_m = (\frac{m}{m+2})^2$ as

$$\begin{aligned}r_{mn}^N =& (T_n(x), \phi_m(x))_\omega, \\ =& \begin{cases} \frac{c_n\pi}{2}, & n = m, \\ -\beta_m\frac{\pi}{2}, & n = m+2, \\ 0, & \text{otherwise}. \end{cases}\end{aligned}$$

E Extension to more dimensions

Introduce the short hand

$$\sum_{\boldsymbol{k}=0}^{\boldsymbol{N}} := \sum_{k_1=0}^{N_1} \sum_{k_2=0}^{N_2} \cdots \sum_{k_D=0}^{N_D},$$

where $\boldsymbol{N} = (N_1, N_2, \ldots, N_D)$ and $\boldsymbol{k} = (k_1, k_2, \ldots, k_D)$ for some $D \geq 1$.

E.1 Three dimensional transform

Higher dimensional transforms are obtained by performing the one-dimensional transform in each direction. As an example, this section discusses the three dimensional case. Denote N_1, N_2, N_3 the chosen number of grid points in the x, y, z directions respectively. The N^{th} order approximation[5] $u_N(x, y, z)$ is expanded in terms of basis functions

$$u_N(x, y, z) = \sum_{n=0}^{N_1} \sum_{m=0}^{N_2} \sum_{k=0}^{N_3} \hat{u}_{nmk} T_n(x) T_m(y) T_k(z).$$

The coefficients $\hat{u}_{nmk}$ are obtained from the nodal values $u(x_i, y_j, z_l)$ upon performing the forward discrete Chebyshev transform (7.12) three times. First transform the z direction

$$u_k^{(1)}(x, y) = \frac{2}{\tilde{c}_k N_3} \sum_{j=0}^{N_3} \frac{1}{\tilde{c}_j} u(x, y, z_j) T_k(z_j),$$

then transform in the y direction

$$u_{m,k}^{(2)}(x) = \frac{2}{\tilde{c}_m N_2} \sum_{j=0}^{N_2} \frac{1}{\tilde{c}_j} u_k^{(1)}(x, y_j) T_m(y_j),$$

and in the x direction

$$\hat{u}_{n,m,k} = \frac{2}{\tilde{c}_n N_1} \sum_{j=0}^{N_1} \frac{1}{\tilde{c}_j} u_{m,k}^{(2)}(x_j) T_n(x_j),$$

where $\tilde{c}_0 = \tilde{c}_N = 2$ and $\tilde{c}_n = 1$ otherwise. The forward transforms can each be performed using the fast Chebyshev transform described in section C.3.

[5] This is a shorthand to denote that u_N is at most order N_1 in the x-direction, N_2 in the y-direction and N_3 in the Z-direction.

Likewise, the nodal values can be retrieved from the coefficients using the backward discrete Chebyshev transform (7.13) three times, whose fast implementation is outlined in section C.3.

$$u^{(2)}_{m,k}(x_j) = \sum_{n=0}^{N_1} \hat{u}_{n,m,k} T_n(x_j)$$

$$u^{(1)}_{k}(x, y_j) = \sum_{m=0}^{N_2} u^{(2)}_{m,k}(x) T_m(y_j)$$

$$u(x, y, z_j) = \sum_{k=0}^{N_3} u^{(1)}_{k}(x, y) T_k(z_j)$$

The computation cost is order $N_1\, N_2\, N_3 \log(N_1\, N_2\, N_3)$. If $N_d = N$ for $d = 1, 2, 3$, then the cost simplifies to $O(N^3 \log N)$.

Analogously, one can transform with other basis functions in each direction successively.

E.2 Operator to matrix notation

The derivation of (10.7) from (10.5) is based on seperation of variables. It uses $\mathcal{L}u = u_{xx} + u_{yy}$, approximation (10.2) and observation (10.6). For simplicity assume equal order polynomials in both directions $N_1 = N_2 = N$.

$$\begin{aligned}
&(\mathcal{L}u_N, \phi_i(x)\phi_j(y))_\omega \\
&\quad = \left(\frac{\partial^2}{\partial x^2} u_N, \phi_i(x)\phi_j(y)\right)_\omega + \left(\frac{\partial^2}{\partial y^2} u_N, \phi_i(x)\phi_j(y)\right)_\omega, \\
&\quad = \sum_{m,n=0}^{N} \Big(\phi''_m(x)\phi_n(y), \phi_i(x)\phi_j(y)\Big)_\omega \hat{u}_{mn} + \sum_{m,n=0}^{N} \Big(\phi_m(x)\phi''_n(y), \phi_i(x)\phi_j(y)\Big)_\omega \hat{u}_{mn}, \\
&\quad = \sum_{m,n=0}^{N} \Big(\phi''_m(x), \phi_i(x)\Big)_\omega \hat{u}_{mn} \Big(\phi_n(y), \phi_j(y)\Big)_\omega + \sum_{m,n=0}^{N} \Big(\phi_m(x), \phi_i(x)\Big)_\omega \hat{u}_{mn} \Big(\phi''_n(y), \phi_j(y)\Big)_\omega, \\
&\quad = (SUM^T + MUS^T)_{ij},
\end{aligned}$$

where the elements of U are given by the coefficients $\tilde{u}_{nm}$ and the elements of S and M are given by (8.9) and (8.7). Each element $(SUM^T + MUS^T)_{ij}$ represents the left hand side of (10.5) for one pair of test functions $\phi_{ij}(x, y)$.

Similarly,

$$\begin{aligned}
&(f,\phi_i(x)\phi_j(y))_\omega \\
&\quad = \sum_{m,n=0}^{N} \Big(T_k(x)T_l(y), \phi_i(x)\phi_j(y)\Big)_\omega \hat{f}_{mn}, \\
&\quad = \sum_{m,n=0}^{N} \Big(T_k(x), \phi_i(x)\Big)_\omega \hat{f}_{mn} \Big(T_l(y), \phi_j(y)\Big)_\omega, \\
&\quad = (RFR^T)_{ij},
\end{aligned}$$

where the elements of F are given by the coefficients $\hat{f}_{nm}$ and the elements of R are given by (8.10).

E.3 Kronecker product

The Kronecker product of A an $m \times n$ matrix and B a $p \times q$ matrix, is given by

$$(A \otimes B) = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}$$

Suppose we want to compute X in $AXB = C$, then

$$\big(B^T \otimes A\big)\ \mathrm{vec}(X) = \mathrm{vec}(AXB) = \mathrm{vec}(C)$$

where $\mathrm{vec}(X)$ is called the vectorization of X. The vectorization is obtained by stacking the column of X into a single column vector:

$$\mathbf{x} = \mathrm{vec}(X) = \begin{pmatrix} X_{00} \\ \vdots \\ X_{n0} \\ X_{01} \\ \vdots \\ X_{n1} \\ \vdots \\ X_{np} \end{pmatrix}$$

F Fast schemes

Consider matrix vector multiplications of the type $Au = g$ with column vectors $u = (u_0, u_1, \ldots, u_{N-1})^T$ and $g = (g_0, g_1, \ldots, g_{N-1})^T$. Intermediate steps use column vectors $f^{(i)} = (f_0^{(i)}, f_1^{(i)}, \ldots, f_{N-1}^{(i)})^T$ and $h^{(i)} = (h_0^{(i)}, h_1^{(i)}, \ldots, h_{N-1}^{(i)})^T$.

F.1 Fast schemes for Chebyshev

Mass matrix

The mass matrix M with entries (8.4) is diagonal. So $g = Mu$ is simply

$$g_n = \begin{cases} \pi u_n, & n = 0, \\ \frac{\pi}{2} u_n, & 1 \le n \le N-1. \end{cases}$$

which costs $O(N)$ flops.

First-derivative matrix

To perform $g = Ku$ with K as expressed in (8.5), we first create f as

$$f_n = \pi n \, u_n, \qquad 0 \le n \le N-1,$$

and then use f in the following recursion relations to obtain g:

$$\begin{aligned} g_{N-1} &= 0, \\ g_{N-2} &= f_{N-1}, \\ g_n &= g_{n+2} + f_{n+1}, \qquad \text{for } n = N-2, \ldots, 0. \end{aligned}$$

The order of execution is important. The cost is again $O(N)$.

Stiffness matrix

To compute $g = Su$ as (8.6), start with

$$\begin{aligned} f_n^{(1)} &= \frac{\pi}{2} n \, u_n, && 0 \le n \le N-1, \\ f_n^{(3)} &= n^2 f_n^{(1)} \, u_n, && 0 \le n \le N-1. \end{aligned}$$

We again use backward recursion relations for both $i = 1$ and $i = 3$

$$\begin{aligned} h_{N-1}^{(i)} &= 0 = h_{N-2}^{(i)} \\ h_n^{(i)} &= h_{n+2}^{(i)} + f_{n+2}^{(i)}, \qquad \text{for } n = N-3, \ldots, 0. \end{aligned}$$

Finally, we combine

$$g_n = h_n^{(3)} - n^2 h_n^{(1)}, \quad 0 \le n \le N-1.$$

Even this more complicated scheme is completed in $O(N)$ computations.

F.2 Fast schemes for Dirichlet boundary conditions

Mass matrix

Consider the computation $g = M^D u$. The elements of the banded matrix M^D are given by (8.12). The product can be computed efficiently as

$$g_n = \begin{cases} \pi\left(\frac{3}{2}u_n - \frac{1}{2}u_{n+2}\right), & n = 0, \\ \pi\left(u_n - \frac{1}{2}u_{n+2}\right), & n = 1, \\ \pi\left(u_n - \frac{1}{2}(u_{n-2} + u_{n+2})\right), & 2 \le n \le N-3, \\ \pi\left(u_n - \frac{1}{2}u_{n-2}\right), & n = N-2, N-1. \end{cases}$$

This costs $O(N)$ computations.

First-derivative matrix

Furthermore, we examine $g = K^D u$ with entries of K^D as in (8.13).

$$g_n = \begin{cases} \pi\, u_{n+1}, & n = 0, \\ \pi(n+1)\,(u_{n+1} - u_{n-1}), & 1 \le n \le N-2, \\ \pi(n+1)\,u_{n-1}, & n = N-1, \end{cases}$$

The above scheme is computed in $O(N)$ computations.

Stiffness matrix

Let us explore recursions for $g = S^D u$, where S^D is given by (8.14). We split S^D into

$$S^D = D^{(1)}(I + D^{(0)} C^S)$$

where I is the identity matrix, $D^{(1)}$ and $D^{(0)}$ are diagonal matrices with entries below and C^S is a strictly upper-triangular checkerboard matrix whose elements are given by:

$$\begin{aligned} d_m^{(1)} &= -2\pi(m+1)(m+2), \\ d_m^{(0)} &= 2/(m+2), \\ c_{mn}^S &= \begin{cases} 1 & n = m+2, m+4, m+6 \ldots \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{F.1}$$

These multiplications can be computed efficiently using the following backward recursion relations. We start with $f = C^S u$,

$$\begin{aligned} f_{N-1} &= f_{N-2} = 0, \\ f_n &= f_{n+2} + u_{n+2}, \quad \text{for } n = N-3, \ldots, 0. \end{aligned} \tag{F.2}$$

Compute $g = D^{(1)}(I + D^{(0)}C^S)u$ with aid of (F.2) as

$$g_n = d_n^{(1)}(u_n + d_n^{(0)} f_n), \quad 0 \le n \le N-1.$$

This scheme of recursion relations takes $O(N)$ computations.

Transformation matrix

Take $g = R^D u$ with R^D as in (8.15):

$$g_n = \begin{cases} \pi \left(u_n - \frac{1}{2}u_{n+2}\right), & n = 0, \\ \frac{\pi}{2}\left(u_n - u_{n+2}\right), & 1 \le n \le N-3, \\ \frac{\pi}{2}u_n, & n = N-2, N-1. \end{cases}$$

F.3 Fast schemes for Neumann boundary conditions

Mass matrix

Analogously, we take $g = M^N u$ for basis functions with Neumann boundary conditions, with M^N as in (8.16). Recall $\beta_n = \left(\frac{n}{n+2}\right)^2$, such that

$$g_n = \begin{cases} \pi\, u_n, & n = 0, \\ \frac{\pi}{2}\left(\beta_n^2 u_n - \beta_n u_{n+2}\right), & n = 1, \\ \frac{\pi}{2}\left(\beta_n^2 u_n - \beta_n u_{n+2} - \beta_n u_{n-2}\right), & 2 \le n \le N-3, \\ \frac{\pi}{2}\left(\beta_n^2 u_n - \beta_n u_{n+2}\right), & n = N-2, N-1, \end{cases}$$

which can be done in $O(N)$ computations.

First-derivative matrix

Additionally, we explore $g = K^N u$ with K^N from (8.17). Define $\eta_n = \frac{n}{n+2}$. We take

$$\begin{aligned} f_n^{(1)} &= \eta_n u_n, \\ f_n^{(2)} &= n f_n^{(1)}. \end{aligned}$$

We perform a recursion scheme for h as

$$\begin{aligned} h_{N-1} =& 0 = h_{N-2} = h_{N-3}, \\ h_n =& h_{n+2} + f^{(1)}_{n+3}, \qquad\qquad \text{for } n = N-4, \ldots, 0. \end{aligned}$$

We combine $f^{(1)}$, $f^{(2)}$ and h to get g

$$g_n = \begin{cases} \pi\Big(2f^{(1)}_{n+1} - \beta_n f^{(2)}_{n+1} + 2(1-\beta_n)h_n\Big), & n = 0, \\ \pi\Big(2f^{(1)}_{n+1} - \beta_n f^{(2)}_{n+1} + 2(1-\beta_n)h_n - f^{(2)}_{n-1}\Big), & 1 \le n \le N-1. \end{cases}$$

The costs of the above algorithm is order N as well.

Stiffness matrix

Similarly to the Dirichlet case in section F.2, one can split S^N as given in (8.18) as

$$S^N = D^{(1)}(I + D^{(2)}C^S D^{(3)}),$$

where again I is the identity matrix, C has entries (F.1) and $D^{(1)}, D^{(2)}, D^{(3)}$ are diagonal matrices with entries:

$$\begin{aligned} d^{(1)}_m &= -2\pi m^2(m+1)/(m+2), \\ d^{(2)}_m &= \frac{2}{m^2(m+2)}, \\ d^{(3)}_m &= m^2. \end{aligned}$$

Efficient recursion relations for $g = S^N u$ are

$$\begin{aligned} h_{N-1} &= h_{N-2} = 0, \\ h_n &= h_{n+2} + d^{(3)}_{n+2} u_{n+2}, \qquad\qquad \text{for } n = N-3, \ldots, 0. \end{aligned}$$

We incorperate h_n into

$$g_n = d^{(1)}_n(u_n + d^{(2)}_n h_n), \quad 0 \le n \le N-1.$$

This recursion scheme can be done in $O(N)$ computations.

Transformation matrix

Lastly, we investigate $g = R^N u$ with (8.19) and $\beta_n = \left(\frac{n}{n+2}\right)^2$,

$$g_n = \begin{cases} \pi\left(u_n - \frac{\beta_n}{2}u_{n+2}\right), & n = 0, \\ \frac{\pi}{2}\left(u_n - \beta_n u_{n+2}\right), & 1 \le n \le N-3, \\ \frac{\pi}{2}u_n, & n = N-2, N-1. \end{cases}$$

G Condition number

Suppose $A \in \mathbb{R}^{N\times N}$ with elements a_{ij} for $0 \le i,j \le N-1$. We introduce several norms (Golub and Van Loan [6]),

$$||A||_1 = \max_{0\le j\le N-1} \sum_{i=0}^{N-1} |a_{ij}|\,, \tag{G.1}$$

$$||A||_\infty = \max_{0\le i\le N-1} \sum_{i=0}^{N-1} |a_{ij}|\,, \tag{G.2}$$

$$||A||_F = \left(\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} |a_{ij}|^2 \right)^{1/2}. \tag{G.3}$$

The last of the above is referred to as the Frobenius norm. Definitions (G.1) and (G.2) are special cases of the general p-norm

$$||A||_p = \sup_{x\neq 0} ||Ax||_p\,,$$

where $x \in \mathbb{R}^N$ such that $||x||_p = 1$. The following inequalities [6] relate the norms,

$$||A||_2 \le \sqrt{N}\,||A||_1\,,$$
$$||A||_2 \le \sqrt{N}\,||A||_\infty\,,$$
$$||A||_2 \le ||A||_F\,, \tag{G.4}$$
$$||A||_2 \le (||A||_1\,||A||_\infty)^{1/2}\,. \tag{G.5}$$

G.1 Condition number of the stiffness matrix

G.1.1 Derivation of the inverse of stiffness matrix

First we define $d_m = (m+1)(m+2)$ and $a_m = 2(m+1)$. Consider the stiffness matrix S^D with Dirichlet boundary conditions (8.14) in terms of d and a

$$s^D_{mn} = \begin{cases} -2\pi d_m, & n = m, \\ -2\pi a_m, & n = m+2, m+4, \ldots, \\ 0, & \text{otherwise.} \end{cases}$$

Let us denote the inverse of S^D with $Z = (S^D)^{-1}$ such that $S^D Z = Z S^D = I$, where I is the identity matrix. The diagonal elements must satisfy

$$1 = I_{mm} = (S^D Z)_{mm} = -2\pi z_{mm} d_m, \quad \text{for all } m.$$

As a result,

$$z_{mm} = (-2\pi d_m)^{-1} = \Big(-2\pi(m+1)(m+2)\Big)^{-1}.$$

Define $\delta_m = -2\pi z_m m$ such that

$$\delta_m = \big((m+1)(m+2)\big)^{-1}. \tag{G.6}$$

For $n = m+2$, the elements of $S^D Z = I$ are

$$0 = I_{m,m+2} = (S^D Z)_{m,m+2} = a_m \delta_m - 2\pi d_{m+2} z_{m,m+2}.$$

This gives

$$-2\pi\, z_{m,m+2} = -\frac{a_m}{d_m d_{m+2}} = -\frac{2}{(m+2)(m+3)(m+4)}.$$

Define $\alpha_n = -2\pi\, z_{n-2,n}$. Thus,

$$\alpha_n = -2\big(n(n+1)(n+2)\big)^{-1}. \tag{G.7}$$

Some bookkeeping verifies that for all m

$$a_m \delta_m + a_{m+2}\alpha_{m+2} = \frac{2}{m+4} = a_{m+2}\delta_{m+2}. \tag{G.8}$$

We construct an expression for the fourth off-diagonal $(S^D Z)_{m,m+4}$ and substitute equality (G.8)

$$\begin{aligned} I_{m,m+4} =& (S^D Z)_{m,m+4}, \\ 0 =& a_m\delta_m + a_{m+2}\alpha_{m+2} - 2\pi d_{m+4} z_{m,m+4}, \\ =& a_{m+2}\delta_{m+2} - 2\pi d_{m+4} z_{m,m+4}. \end{aligned}$$

Comparison to

$$\begin{aligned} I_{m+2,m+4} =& (S^D Z)_{m+2,m+4}, \\ 0 =& a_{m+2}\delta_{m+2} + d_{m+4}\alpha_{m+4}, \end{aligned}$$

gives that α_{m+4} must be equal to $-2\pi z_{m,m+4}$. Likewise, for all $k = 6, 8, 10, \ldots,$ the expression for$(S^D Z)_{m,m+k}$ can be compared with $(S^D Z)_{m,m+k-2}$ resulting in $-2\pi z_{m,m+k} = \alpha_{m+k}$.

As a result, the inverse Z of the Dirichlet stiffness matrix S^D also has an upper-triangular checkerboard structure

$$z_{mn} = \begin{cases} -(2\pi)^{-1}\delta_n, & n = m, \\ -(2\pi)^{-1}\alpha_n, & n = m+2, m+4, \ldots, \\ 0, & \text{otherwise}, \end{cases}$$

with δ_n as in (G.6) and α_n as in (G.7).

G.1.2 Norm of S^D

Computation of (G.1), (G.2) and (G.3) for S^D gives

$$\begin{aligned}
\left|\left|S^D\right|\right|_1 \leq & 2\pi \left(d_N + \sum_{\substack{k=0 \\ k+m \text{ odd}}}^{N-2} a_{k,N} \right), \\
\leq & 2\pi \left(d_N + \left\lfloor \frac{N}{2} \right\rfloor a_{\lfloor \frac{N-1}{2} \rfloor} \right), \\
\leq & 2\pi \left((N+1)(N+2) + \frac{N}{2}(N+1) \right), \\
\leq & O(N^2).
\end{aligned}$$

$$\begin{aligned}
\left|\left|S^D\right|\right|_\infty = & 2\pi d_{N-1} = 2\pi(N+1), \\
= & O(N^2).
\end{aligned}$$

$$\begin{aligned}
\left|\left|S^D\right|\right|_F = & 2\pi \sqrt{\sum_{k=0}^{N-1} d_k^2 + \sum_{k=0}^{N-3} \left\lfloor \frac{N-k}{2} \right\rfloor a_k^2}, \\
\leq & 2\pi \sqrt{\sum_{k=0}^{N-1} (k+1)^2(k+2)^2 + \sum_{k=0}^{N-3} \frac{N-i}{2} 4(k+1)^2}, \\
\leq & 2\pi \sqrt{\frac{1}{15}(N+1)(N+2)(N+3)\left(3N^2+12N+10\right)}, \\
& \qquad \overline{+\frac{1}{6}(N-1)N(N^2+5N-2)}, \\
\leq & O(N^{5/2}).
\end{aligned}$$

In the last derivation above Maple software was used for the analytic expression of the sum. Inequality (G.5) yields

$$\begin{aligned}
\left|\left|S^D\right|\right|_2 & \leq \sqrt{\left|\left|S^D\right|\right|_1 \left|\left|S^D\right|\right|_\infty}, \\
& \leq \sqrt{2\pi \left((N+1)(N+2) + \frac{N}{2}(N+1) \right) 2\pi(N+1)(N+2)}, \\
& \leq O(N^2).
\end{aligned}$$

G.1.3 Norm of Z

The norms (G.1)-(G.3) of the inverse stiffness matrix Z (13.3) are

$$\begin{aligned} ||Z||_1 =& \frac{1}{2\pi}\delta_0 = \frac{1}{4\pi}, \\ \leq& O(1). \end{aligned}$$

$$\begin{aligned} ||Z||_\infty =& \frac{1}{2\pi}\left(\delta_0 + \sum_{k=1}^{\lfloor N/2 \rfloor} |\alpha_{2k}|\right), \\ =& \frac{1}{2\pi}\left(\frac{1}{2} + \sum_{k=1}^{\lfloor N/2 \rfloor} \frac{1}{k(2k+1)(2k+2)}\right), \\ \leq& \frac{1}{2\pi}\left(\frac{1}{2} + \frac{\pi^2}{6}\right), \\ \leq& O(1). \end{aligned}$$

$$\begin{aligned} ||Z||_F =& \frac{1}{2\pi}\sqrt{\sum_{k=0}^{N-1}\delta_k^2 + \sum_{k=2}^{N-1}\left\lfloor \frac{k}{2} \right\rfloor \alpha_k^2}, \\ \leq& \frac{1}{2\pi}\sqrt{\sum_{k=0}^{N-1}(k+1)^{-2}(k+2)^{-2} + \sum_{k=2}^{N-1}\frac{k}{2}\left(\frac{2}{k(k+1)(k+2)}\right)^2}, \\ \leq& \frac{1}{2\pi}\sqrt{\frac{\pi^2}{6} + \frac{\pi^2}{6}}, \\ \leq& O(1). \end{aligned}$$

Using inequality (G.4) an upper bound on the norm of Z is

$$\begin{aligned} ||Z||_2 \leq& ||Z||_F, \\ \leq& O(1). \end{aligned}$$

G.1.4 Norm of $\hat{S}$

The relation

$$\sum_{j=1}^{N}\frac{1}{j} \leq 1 + \ln(N) \leq 1 + 2\sqrt{N}, \tag{G.9}$$

is derived in Appendix section G.1.6. For $\hat{S}$ as in (13.2), the norms (G.1) is

$$\begin{aligned}\left|\left|\hat{S}\right|\right|_1 =& 1 + \sum_{\substack{k=0 \\ k \text{ even}}}^{N-3} \frac{2}{i+2}, \\ \leq& 1 + \sum_{k=1}^{\lfloor N/2 \rfloor} \frac{1}{k}, \\ \leq& 2 + \ln\left(\frac{N}{2}\right), \\ \leq& 2 + 2\sqrt{N}, \\ \leq& O(N^{1/2}).\end{aligned}$$

where (G.9) was used. Moreover, norms (G.2)-(G.3) are

$$\begin{aligned}\left|\left|\hat{S}\right|\right|_\infty =& \sum_{\substack{k=0 \\ k \text{ even}}}^{N-1} 1, \\ =& 1 + \lfloor N/2 \rfloor, \\ \leq& O(N).\end{aligned}$$

$$\begin{aligned}\left|\left|\hat{S}\right|\right|_F =& \sqrt{\sum_{k=0}^{N-1} 1 + \sum_{k=0}^{N-3} \left\lfloor \frac{N-k}{2} \right\rfloor \left(\frac{2}{k+2}\right)^2}, \\ \leq& \sqrt{N + 1 + 2N\frac{\pi^2}{6}}, \\ \leq& O(N^{1/2}).\end{aligned}$$

Due to inequality (G.4), an upper bound of the norm of $\hat{S}$ can be found as

$$\left|\left|\hat{S}\right|\right|_2 \leq O(N^{1/2}).$$

G.1.5 Norm of $\hat{Z}$

With aid of (G.9)we evaluate the (G.1)-(G.3) norms for $\hat{Z}$ given by (13.5) produces

$$\left|\left|\hat{Z}\right|\right|_1 = 2.$$

$$\begin{aligned}\left|\left|\hat{Z}\right|\right|_\infty =& 1+\sum_{k=1}^{\lfloor N/2\rfloor}\frac{1}{k},\\ \leq& 2+\ln\left(\frac{N}{2}\right),\\ \leq& 2+2\sqrt{N},\\ \leq& O(N^{1/2}).\end{aligned}$$

$$\begin{aligned}\left|\left|\hat{Z}\right|\right|_F =& \sqrt{\sum_{k=0}^{N-1}1+\sum_{k=2}^{N-1}\left\lfloor\frac{k}{2}\right\rfloor\left(\frac{2}{k}\right)^2},\\ \leq& \sqrt{N+1+\sum_{k=2}^{N-1}\frac{2}{k}},\\ \leq& \sqrt{N+1+N-1},\\ \leq& O(N^{1/2}).\end{aligned}$$

An upper bound for $\hat{Z}$ can be found with inequality (G.4),

$$\left|\left|\hat{Z}\right|\right|_2 \leq O(N^{1/2}).$$

G.1.6 Derivation of (G.9)

For $k \geq 2$,

$$\frac{1}{k} \leq \frac{1}{x}, \quad \text{for all } k-1 \leq x \leq k.$$

Thus

$$\frac{1}{k} \leq \int_{k-1}^{k}\frac{1}{x}dx, \quad \text{for all } k \geq 2. \tag{G.10}$$

Let's consider

$$\sum_{j=1}^{N}\frac{1}{j} = 1+\sum_{j=2}^{N}\frac{1}{j}.$$

Substitute (G.10) for each element of the sum and join the integrals

$$\begin{aligned}\sum_{j=1}^{N}\frac{1}{j} \leq & 1+\int_{1}^{N}\frac{1}{x}dx,\\ \leq & 1+\ln(N).\end{aligned} \tag{G.11}$$

Logarithmic rules state

$$\ln(x^{\alpha}) = \alpha\ln(x).$$

If $x \geq 1$, then $\ln(x) \leq x$, because $\frac{d}{dx}\ln(x) = \frac{1}{x} \leq 1 = \frac{d}{dx}x$. As a result,

$$\ln(x^{\alpha}) \leq \alpha x, \quad \text{for all } x \leq 1.$$

We take $x = (\sqrt{N})^2$ such that

$$\ln(N) \leq 2\sqrt{N}.$$

Combining this with inequality (G.11) shows relation (G.9):

$$\sum_{j=1}^{N}\frac{1}{j} \leq 1+\ln(N) \leq 1+2\sqrt{N}.$$

G.2 Condition number of $\hat{S}$ versus $\hat{L}^{(1)}$

Figure G.1 shows that the condition number of $\hat{L}^{(1)}$ is (as expected) the same order as the condition number of $\hat{S}$.

G.3 Settings in numerical tests on condition number

The condition numbers in section 13.2.2 have been computed in Matlab and the iteration numbers are obtained by solving the system $Au = g$ for u with the Matlab built-in BiCGSTAB routine. The iteration number per N was averaged over 10 trials with g comprising normally distributed random numbers.

For the one-dimensional case displayed in Figures 13.1 and G.1, the range is $N = 4, 6, 8, \ldots, 500$. The BiCGSTAB algorithm was executed with tolerance 10^{-14}, which is well-above the Matlab machine precision of 10^{-16}.

The two-dimensional case in Figure 13.2 has range $N = 4, 6, 8, \ldots, 50$, and the three dimensional situation in Figure 13.3 employed $N = 2, 4, 6, \ldots, 20$. The tolerance was set to 10^{-14} and 10^{-8} for respectively, the two and three dimensional case.

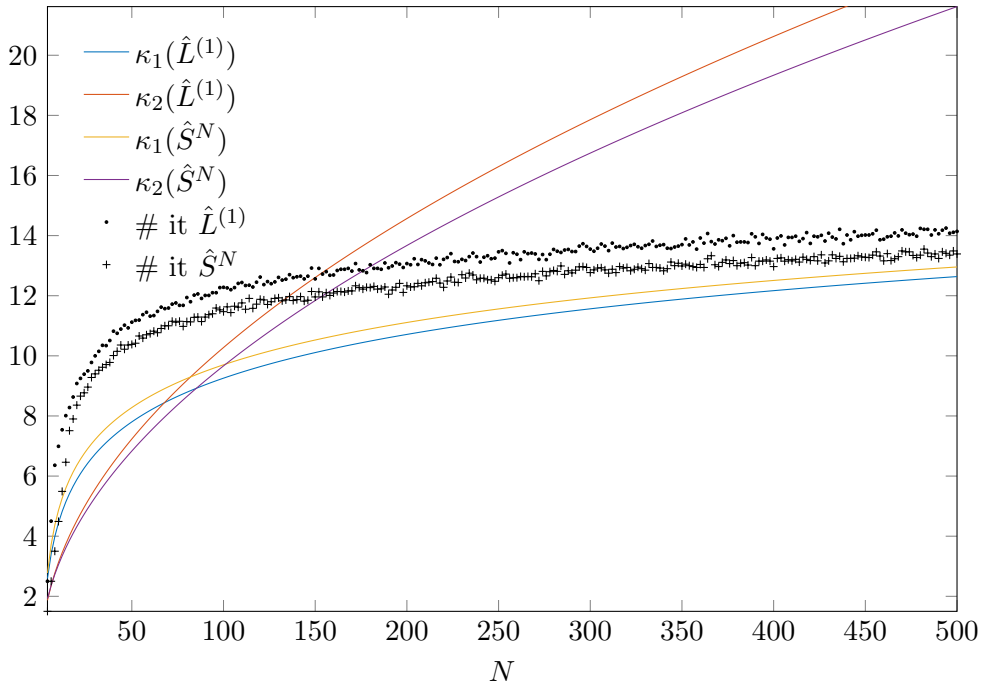

Figure G.1: The condition numbers in 1-norm (blue,yellow) and 2-norm (in red,purple) of preconditioned Neumann stiffness matrices $\hat{S}$ (13.7) (in yellow,purple) and $\hat{L}^{(1)}$ (13.9) (in blue,red), where N denotes the number of CGL points. Additionally, the average iteration counts in BiCGSTAB are shown as dots for $\hat{L}^{(1)}$ and crosses for $\hat{S}^N$.

G.4 Setting in tests on convergence

The tests on the condition numbers where performed using Matlab with test problem (14.1). Solutions were obtained from the linear systems solved with BiCGSTAB with a tolerance of 10^{-15}. The error was measured as (14.2).

In one, two and three dimensions, the resolution (in each direction) was set to respectively $N = 5, 7, 9, \ldots, 81$, $N = 3, 5, 7, \ldots, 91$ and $N = 4, 6, 8, \ldots, 30$.

Acknowledgements

Firstly, I would like to thank my supervisors prof. Leo Maas and prof. Jason Frank for their valuable input. I enjoyed the physical discussions with Leo about the theoretical expectations and I treasure the relentless optimism with which Jason helped me find new methods.

My thanks go to Marlies for giving me a tour of the laboratory set-up at the NIOZ and to Costanza and Andrea for providing me with details about their simulations with Gerris software. Furthermore, I am grateful to the IMAU institute for the facility and interesting colloquia. Last but not least, many thanks to my fellow students in the student room for all their support.

References

[1] Barrett, R., M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. van der Vorst, 1994. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM.

[2] Canuto, C., M.Y. Hussaini, A. Quarteroni, and T.A. Zang, 2006. *Spectral Methods: Fundamentals in Single Domains.* Springer-Verlag.

[3] Cushman-Roisin, B. and J.M. Beckers, 2011. *Introduction to Geophysical Fluid Dynamics.* Elsevier.

[4] Dijkstra, H. A., 2008. *Dynamical Oceanography* Springer-Verlag, 2008.

[5] Frank, J., W. Hundsdorfer and J,G, Verwer, 1997. *On the stability of Implicit-Explicit Linear Multistep Methods.* Applied Numerical Mathematics, 25, 193-205.

[6] Golub, G.H., and C.F. Van Loan, 2013. *Matrix computations.* The John Hopkins University Press, 4th ed.

[7] Greenspan, H.P., 1968. *The Theory of Rotating Fluids.* Cambridge University Press.

[8] Guermond, J.L., P. Minev, and J. Shen, 2006. *An overview of project methods for incompressible flows.* Computer Methods Applied Mechanics and Engineering., 195, 6011-6045.

[9] Guermond, J.L. and J. Shen, 2004. *On the error estimates for the rotational pressure-correction projection methods.* Mathematics of Computation, 73:2478, 1719-1737.

[10] Jouve, L. and G.I. Ogilvie, 2014. *Direct numerical simulations of an inertial wave attractor in linear and nonlinear regimes.* Journal of Fluid Mechanics, 745, 233-250.

[11] van der Lugt, M., 2014. *Ocean flow captured in a box: experiments on geostrophy.* Unpublished bachelor thesis at NIOZ.

[12] Maas, L.R.M., 2007. *Experiments on rotating flows: impact of rotation on flow through tilted rectangular ducts.* St. Petersburg conference 'Fluxes and Structures in Fluids', Ed. Y. D. Chashechkin.

[13] Maas, L.R.M. 2003. *On the ampidromic structure of inertial waves in a rectangular parallelepiped.* Fluid Dynamics Research, 33, 373-401.

[14] Maas, L.R.M., Lam, F.P. A., 1995. *Geometric focusing of internal waves.* J. Fluid Mech., 300, 1-41.

[15] Manders, A.M.M. and L.R.M. Maas, 2004. *On the three-dimensional structure of the inertial wave field in a rectangular basin with one sloping boundary.* Fluid Dynamics Research, 35, 1-21.

[16] Manders, A.M.M. and L.R.M. Maas, 2003. *Observations of inertial waves in a rectangular basin with one sloping boundary.* Journal of Fluid Mechanics, 493, 59-88.

[17] Manders, A.M.M. and L.R.M. Maas, 2003. *Inertial wave propagation, focusing and mean flow generation.* Towards a Balanced Methodology in European Hydraulic Research.

[18] Nurijanyan, S., O. Bokhoven and L.R.M. Maas, 2013. *A new semi-analytical solution for inertial waves in a rectangular parallelepiped.* Journal of Fluid Mechanics, 729, 445-470.

[19] Peyret, R., 2002. *Spectral Methods for Incompressible Viscous Flow.* Springer-Verlag.

[20] Rodda, C., 2015. *The ocean in a tank: numerical simulations on the wave-mean flow interactions in a rotating box.* Unpublished master thesis at NIOZ.

[21] Shen, J., T. Tang and L.L. Wang, 2011. *Spectral Methods: Algorithms, Analysis and Applications.* Springer Series in Computational Mathematics.

[22] Shen, J., 1995. *Efficient Spectral-Galerkin Method II. Direct Solvers of Second and Fourth Order Equations by Using Chebyshev Polynomials.* SIAM J. Sci. Comput. 16:1, 74-87.

[23] Van der Vorst, H. A., (1992). *Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems.* SIAM J. Sci. and Stat. Comput. 13:2, 631644.