The Cauchy-Lagrange algorithm for axisymmetric incrompressible Euler flow in a wall bounded cylindrical domain : Numerical exploration of regularity loss in the solutions to hydrodynamical equations by high-order semi-Lagrangian methods

Tobias Hertel

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES FONDAMENTALES ET APPLIQUÉES

THÈSE DE DOCTORAT

L'algorithme de Cauchy-Lagrange pour les écoulements axisymétriques et incompressibles d'Euler dans un domaine cylindrique borné

Tobias HERTEL

Laboratoire J.-L. Lagrange (UMR 7293), Observatoire de la Côte d'Azur

Présentée en vue de l'obtention du grade de docteur en Mathématiques
de l'Université Côte d'Azur
Dirigée par : Nicolas Besse
Soutenue le : 4 décembre 2020

Devant le jury composé de :
Rahul Pandit, Professor, Indian Institute of Science
Edriss Titi, Professor, University of Cambridge
Nicolas Besse, Professor, Université Côte d'Azur
Uriel Frisch, Directeur de Recherche CNRS, Observatoire de la Côte d'Azur
Jérémie Bec, Directeur de Recherche CNRS, MINES ParisTech
Bérengère Dubrulle, Directrice de Recherche CNRS, CEA Paris-Saclay
László Szekélyhidi, Professor, Universität Leipzig
Claude Bardos, Emeritus Professor, Université Denis Diderot Paris

L'algorithme de Cauchy-Lagrange pour les écoulements axisymétriques et incompressibles d'Euler dans un domaine cylindrique borné

Exploration numérique de la perte de régularité des solutions des équations de l'hydrodynamique par des méthodes semi-Lagrangiennes d'ordres très élevés

~

The Cauchy-Lagrange algorithm for axisymmetric incrompressible Euler flow in a wall bounded cylindrical domain

Numerical exploration of regularity loss in the solutions to hydrodynamical equations by high-order semi-Lagrangian methods

Jury :

Rapporteurs

Rahul Pandit, Professor, Indian Institute of Science
Edriss Titi, Professor, University of Cambridge

Examinateurs

Uriel Frisch, Directeur de Recherche CNRS, Observatoire de la Côte d'Azur
Bérengère Dubrulle, Directrice de Recherche CNRS, CEA Paris-Saclay
Jérémie Bec, Directeur de Recherche CNRS, MINES ParisTech
László Szekélyhidi, Professor, Universität Leipzig
Claude Bardos, Emeritus Professor, Université Denis Diderot Paris

Directeur de thèse

Nicolas Besse, Professor, Université Côte d'Azur

Résumé et mots clés
Abstract and keywords

Titre : L'algorithme de Cauchy-Lagrange pour les écoulements axisymétriques et incompressibles d'Euler dans un domaine cylindrique borné

Résumé :
À l'aide du nouvel algorithme de Cauchy-Lagrange (CLA) ce travail étudie un écoulement d'Euler incompressible, axisymétrique et périodique, confiné dans un domaine cylindrique, borné par une paroi dans la direction radiale. Des simulations très précises sont effectuées en vue de reconsidérer le résultat récent d'une possible singularité en temps fini (blow-up) pour un flot particulier dans cette géométrie. L'algorithme exploite l'analyticité temporelle des trajectoires des particules du champ de vitesse. Les coefficients de Taylor en temps des trajectoires satisfont des relations de récursion et sont obtenus en résolvant des problèmes de décomposition de Helmholtz-Hodge. Une approche pseudo-spectrale du type Chebyshev-Fourier est appliquée aux problèmes numériques dans un système aux coordonnées cylindriques tout en évitant une différenciation spectrale instable. Des méthodes pour le choix d'un pas de temps, fixe ou adaptatif, sont proposées et étudiées numériquement. Après l'insertion d'un pas de temps, les champs d'écoulement sont connus sur les trajectoires des particules dispersés dans une tranche du cylindre. Plusieurs méthodes d'interpolation sur une grille déformée sont considérées et comparées afin de trouver la meilleur façon de rééchantillonner le flot sur la grille fixe pour la prochaine itération en temps. L'implémentation du CLA présentée se révèle être précise, stable et rapide. L'absence de condition CFL rend le CLA plus performant que les algorithmes traditionnels. En effet, cette propriété permet de prendre des pas de temps plus large, ce qui réduit le nombre total d'itérations. Des tests approfondis sur les écoulements stationnaires et d'autres, non-stationnaires, sans vitesse angulaire sont effectués. Puis, des simulations avec des conditions initiales susceptibles de développer une singularité en temps fini, sont effectuées et analysées. Cette première mise en œuvre du CLA pour un domaine avec une paroi s'avère être un excellent outil pour résoudre les équations d'Euler incompressibles d'une manière rapide et précise dans une géométrie donnée.

Mots clés : algorithme de Cauchy-Lagrange – Schéma semi-Lagrangien – écoulement axisymétrique – équations d'Euler incompressible – cylindre périodique – blow-up – singularité en temps fini – problèmes de Poisson cylindriques

Title: The Cauchy-Lagrange algorithm for axisymmetric incrompressible Euler flow in a wall bounded cylindrical domain

Abstract:
This work investigates wall bounded axisymmetric flow via the novel Cauchy-Lagrange algorithm (CLA) for the incompressible Euler equations. Accurate simulations of periodic flow confined in a cylinder with boundaries are pursued in sight of a recently reported finite time singularity (blow-up) in this geometry. The algorithm exploits the fact that the particle trajectories of an incompressible Euler flow are analytic in the temporal variable. The time-Taylor coefficients of those trajectories fulfill recursion relations and are obtained by solving Helmholtz-Hodge decomposition problems. A Chebyshev-Fourier pseudo-spectral approach to the appearing numerical problems in the underlying cylindrical coordinate system is studied while avoiding unstable spectral differentiation. The appearing Poisson problems are directly solved for the second derivatives of their solutions. For this, a sophisticated Poisson solver, based on modified Bessel functions, is developed and employed. Time-stepping techniques, fixed and adaptive, are proposed and numerically investigated. Furthermore, several methods for the scattered interpolations of the flow fields, known on the particle trajectories after a time step insertion, back onto the fixed grid are reviewed and compared. The presented implementation of the CLA aims for accuracy and is found to be stable and fast also due to the absence of a CFL condition. The latter allows larger time steps and thus reduces tremendously the total number of iterations. Thorough tests on stationary and swirl-free flows are executed and discussed before a number of simulations of the blow-up candidate are inspected. The implementation of the CLA that is presented in this thesis proves to be an excellent tool for fast and accurate incompressible Euler computations in the given geometry.

Keywords: Cauchy-Lagrange algorithm – semi-Lagrangian method – axisymmetric flow – incompressible Euler equations – periodic cylinder – blow-up – finite time singularity – cylindrical Poisson problems

Acknowledgments

I would like to thank Nicolas Besse for supervising this thesis and for providing valuable feedback on the thesis manuscript.

To Uriel Frisch I would like to express my profound gratitude for his guidance as a quasi co-supervisor. I learned a lot from him, not only about turbulence.

I am grateful to Rahul Pandit and Sai Swetha Kolluru Venkata for many helpful conversations. Unfortunately, it was not possible that we, Sai Swetha and I, meet in person due to the pandemic situation in the past months, but I am very thankful for our numerous video conferences in which we discussed a wide spectrum of Euler-flow related topics.

I would also like to thank László Szekélyhidi for having me introduced to Lagrangian fluid dynamics in the first place and for enabling me to start this Ph.D. project in Nice. During my time under his supervision in Leipzig, I acquired mathematical key competences and learned to always keep the big picture in mind, which helped to realize the results presented in this thesis.

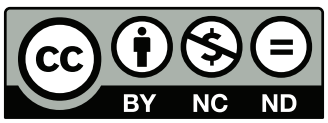
I thank Jeremie Bec, Giorgio Krstulovic, Holger Homann and Yannick Ponty for having supplied me with information on meetings, summer schools and conferences on fluid dynamics, and also for having had an open ear during challenging times.

I dearly thank my girlfriend Perla Sperandei for her incredible support during the writing of this manuscript. She really kept the boat afloat by creating a stable and loving atmosphere.

I thank Gabriele Pichierri for his help to increase the readability of this manuscript.

And last, but not least, I would like to profoundly thank my family and friends for their continuous support along the way.

$\sim$



Contents

Chapter 1

Introduction

The particle trajectories of an incompressible Euler flow are in possession of one of the finest properties in calculus, they are analytic in time [106]. In other words, those particle trajectories, also called characteristics and denoted by X, are the sum of a locally absolutely converging Taylor series with respect to the temporal argument t,

$$X(a,t;t_0) = \sum_{s=0}^{\infty} X^{(s)}(a)\,(t-t_0)^s, \tag{1.1}$$

where a is the spatial argument and t_0 a time in the existence interval of the Euler solution. Remarkably, as underlined by Serfati, the time-analyticity of the characteristics is largely independent of the spatial regularity of the underlying flow. In the classical sense, a minimal regularity slightly smoother than Lipschitz of the initial velocity field v_0 is sufficient to entail the result. As a matter of fact, the last assumption is also a natural solvability condition for the incompressible Euler equations in the classical sense [82, 61]. See for instance [5] for the non-solvability of the classical equations, if v_0 fails to be Lipschitz. Bluntly put, one can say that the series (1.1) has a positive radius of convergence, whenever the incompressible Euler equations are solvable in some time span. This even holds under the most general geometrical assumptions and lowest regularity requirements on the flow [111, 11].

Background. In the early 1990s, Chemin [23] proved a C^∞ result regarding the temporal regularity of the characteristics. His proof relied on continuous reapplications of the material derivative $\partial_t + v \cdot \nabla$ to the pressure gradient written as an operator on the velocity field v. Later, Gamblin [54] refined this technique and showed time-analyticity of the Lagrangian particle paths using estimates in a purely Eulerian viewpoint. However, before Gamblin, Serfati [105, 104, 106] gave a first proof of time-analyticity of the particle trajectories in a purely Lagrangian viewpoint for classical Hölderian solutions in the full d-dimensional space. His proof is based on a reformulation of the incompressible Euler equations as a self-contained Lagrangian acceleration equation, which is an ordinary differential equation (ODE) of second order in time with an analytic right-hand side (RHS). A holomorphic extension of this ODE with values in well-suited complex Banach spaces is then solved via the Picard iteration to infer time-analyticity. This first proof of temporal analyticity of the characteristics was thoroughly analyzed by the author of this thesis in [64]. After the first proofs in the full space, it took more than a decade before this subject regained attention. Shnirelman [111] revived the problem on generalized grounds in the language of differential geometry, using Arnold's formulation of the Euler equations, and proved time-analyticity on the three-dimensional torus. A domain with boundary was first considered by Kato in [70], who proved temporal

C^∞-regularity of the characteristics in a domain $\Omega \subset \mathbb{R}^d$ with smooth boundaries. The incompressible Euler equations in this case read

$$\begin{aligned} v_t + v \cdot \nabla v &= -\nabla p, \\ \nabla \cdot v &= 0, \qquad \text{in } \Omega, \\ \nu \cdot v &= 0, \qquad \text{on } \partial\Omega, \\ v\,|_{t=0} &= v_0, \end{aligned} \tag{1.2}$$

with an initial velocity field v_0 that is smother than Lipschitz. In this setting, the particle trajectories are given by the unique solution of the characteristic equation

$$\partial_t X(a,t) = v(X(a,t),t), \quad X(a,0) = a, \quad a \in \Omega. \tag{1.3}$$

Time-analyticity in the case of a smoothly bounded domain was proved by Glass et al [59], who studied the movement of a rigid body immersed in an ideal incompressible fluid. They showed that the temporal regularity of the trajectories is only limited by the regularity of the boundary defining map. Thus, analytic boundaries of the domain imply time-analytic characteristics. Little later, Constantin et al [30] shed more light on the subject of Lagrangian time-analyticity in full space. They declared that the latter is common to several hydrodynamic models whose Eulerian velocities are slightly smoother than Lipschitz. Their proof mainly exploits a so called uniform arc-chord property of the particle trajectories. They further employ singular integral calculus to obtain estimates on the time derivatives of X, which allows them to conclude by induction. Singular integrals appear naturally in the study of fluid dynamical equations, the most prominent example being the Biot-Savart law (e.g. [84, p. 72]), and their analysis is essential in most of the above references.

A constructive approach to time-analyticity. There is yet another approach to temporal analyticity which does not involve any singular integrals. Frisch and Zheligovsky [52] presented shortly after Shnirelman a constructive and elementary proof of the time-analyticity, using a revived Lagrangian formulation of incompressible flow. The latter is attributed to Cauchy and now well known as the Cauchy-invariants formulation (CIF) of the incompressible Euler equations [22, 51], it will be recalled at the beginning of Section 2.3. By inserting the particle displacement $X - \mathrm{Id}$, where Id denotes the identity map, into the CIF, the authors of [52] showed the existence of recursion relations between the time-Taylor coefficients of different orders. More precisely, those relations express the divergence and curl of the coefficient to some order $X^{(s)}$ by the gradients of lower order coefficients. In the reference, the coefficient $X^{(s)}$ is then obtained by solving a Helmholtz-Hodge decomposition problem with periodic boundary conditions. A first numerical implementation of this recurrence mechanism was carried out by Podvigina et al [95] for a periodic two-dimensional flow. On that occasion, the Cauchy-Lagrange Algorithm (CLA) was born and proved to be effective when compared to the traditional Eulerian implementations of incompressible flow. The efficiency of the CLA lies in the fact that it is a Semi-Lagrangian scheme and thus entirely independent of a Courant-Friedrichs-Lewy (CFL) condition, which limits the time step in traditional schemes. Indeed, the time step in this novel algorithm is only bound by the radius of convergence, which behaves typically as $\mathrm{cst.}/\left\|\omega_0\right\|_\alpha$, i.e. as the inverse of the Hölder norm of the initial vorticity (at t_0). Contrary to CFL bound time steps, the time step in the CLA only depends on the convergence radius regardless of the underlying spatial discretization.

Because the proof of time-analyticity in [52] assumes spatial periodicity, the work in [95] only treats two-dimensional periodic flow. The particular case of a bounded domain in three-

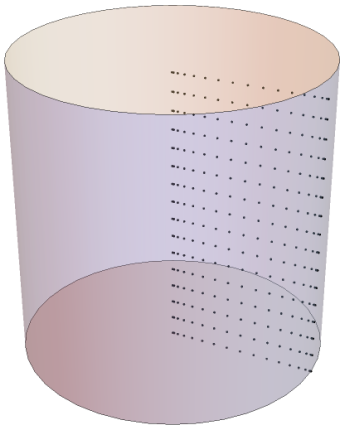


Figure 1.1: The modeled region $D(1, 2)$ with the Chebyshev-Fourier grid on a slice. The flow inside the cylinder is periodically continued in vertical direction.

dimensional space was treated by Besse and Frisch [12] in extending the arguments from [52] to suit the geometrical setting that involves smooth (or ultra-differentiable) boundaries. As in [95], the constructive nature of the proof in [12] gives rise to an efficient algorithm to compute wall-bounded incompressible Euler flow. The investigation of this novel algorithm and its implementation for a simple geometry is the subject of this thesis. It describes the CLA for wall-bounded flow in a domain $\Omega \subset \mathbb{R}^3$, and provides many details for the particular case when Ω is an infinite cylinder along the vertical axis of $\mathbb{R}^3$. The flow in the cylinder is further assumed to be axisymmetric with respect to the vertical axis and periodic along the vertical direction with periodicity length L. Note that the treated geometry is a domain with boundary, but not bounded. Without going into detail, it can be readily verified that the results from [12] also hold for this cylindrical domain. Because of the periodicity, the modeled region in the implementation is finite and denoted by $D(R, L)$, where R is the radius of the cylinder and L as before. The corresponding cylindrical coordinate system and notation can be found in Section 2.2. One reason for the choice of this geometry is of a practical nature. The assumed axisymmetry reduces the underlying problems to two dimensions and cuts down the overall numerical complexity of the algorithm. A second reason for the chosen geometry and flow properties is the potential finite time singularity that was reported in such a setting by Lou and Hou in [83]. Let us briefly recall this blow-up problem for the incompressible Euler equations and discuss how our numerical investigations can contribute to this long-standing and still unresolved question.

Euler singularities. As for the Navier-Stokes equations, the question for global well-posedness of the incompressible Euler equations is still open. The blow-up problem asks if an initially smooth velocity field (with finite energy) can give rise to a solution of system (1.2) whose gradient develops a finite-time singularity, such that the underlying equations of motion are not fulfilled anymore due to insufficient regularity in the classical sense. This problem arises for many other hydrodynamic models too and, given the importance of the problem, there are numerous analytical and computational investigations on this subject. For brevity, we list only the important works which directly address the blow-up problem for the incompressible Euler equations. A thorough inspection of the concerned literature can be found for instance in [56]. A celebrated result of Beale, Kato, and Majda [9] for the three-dimensional space connects the blow-up event to the time-integrated vorticity modulus.

More accurately, if a blow-up occurs at a finite time T^*, then the time-integrated L^∞-norm of the vorticity ω until T^* becomes infinite and in particular

$$\limsup_{t \to T^*} \|\omega\|_{L^\infty} = \infty.$$

This necessary condition for a blow-up also holds in a bounded domain as shown by Ferrari in [45]. Several refinements of the above criteria as well as other blow-up criteria were brought to light in the subsequent years, most notably that of Constantin et al in [29]. The latter work studies the relationship between the non-appearance of a singularity and the temporal integrability of the gradient of the vorticity direction $\xi = \omega/|\omega|$ close to the location of the maximal vorticity. In the reference's terms, a blow-up cannot appear if the Lagrangian displacement of a set of points in the region of maximal vorticity is smoothly directed. More work in this direction was accomplished in [31, 40], see also [66, 65]. Moreover, a large amount of computational effort has been employed to find Euler singularities, where most authors aim to verify the criteria of Beale-Kato-Majda. Prominent works in this respect are those of Grauer and Sideris [60], Pumir and Siggia [99] and Kerr [71]. All of them report a singularity in finite time due to seemingly unbounded vorticity growth, but their results were questioned in later works, which attributed those findings to numerical artifacts or under-resolution (see e.g. [43, 67, 19]). In addition, complex singularities of the incompressible Euler equations with complex initial datum were reported in the work of Caflisch and Siegel [20, 112]. A more recent computational result uses a blow-up criteria which makes use of the Euler-Voigt inviscid regularization that was developed by Larios and Titi [78] (see also [80]). An Euler blow-up can be connected to the behavior of the solution to the aforementioned regularization. An Euler singularity can now be pursued in computing the much better behaved incompressible Euler-Voigt equations. The effectiveness of the method from [78] is demonstrated in [77]. The latter reference treats a periodic Taylor-Green initial velocity in a box, see also the classical references [89, 18]. However, as discussed by Larios and Titi [79], it might be the case that the existence of boundaries plays an important role in the formation of finite-time singularities. Indeed, the thorough numerical investigation of an axisymmetric incompressible Euler flow in an infinite cylinder with boundaries by Lou and Hou [83] supplies ample numerical evidence for a finite time blow-up in the given geometry. Those computational findings have recently inspired proofs of the existence of finite-time singularities with a $C^{1,\alpha}$ initial velocity field that has finite energy [24]. A different proof, in the exterior of an hour-glass shaped domain, provided a similar result for finite-time blow-up [44], see also [44, 28, 119] and references therein for further analytical results involving infinite kinetic energy.

The potential of the Cauchy-Lagrange approach. While the results in [83] are convincing overall, a verification of the latter can only be achieved by a rigorous mathematical proof of the regularity loss in the blowing-up solution, or by another numerical method that is entirely different and produces equal results. In my opinion, the Cauchy-Lagrange algorithm is able to serve such demand due to the following reasons. First, while [83] uses an Eulerian viewpoint, the viewpoint of the CLA is entirely Lagrangian and computes the particle trajectories by recursion relations which stem from the Cauchy-Invariants formula. Second, the methods in [83] rely on an adaptive moving mesh paired with a Runge-Kutta time-stepping scheme and a B-spline Poisson solver, they are hence restricted by a CFL condition and are forced to take very small time steps. Being a Semi-Lagrangian method, the CLA enjoys the advantage that larger, and thus fewer, time steps may be taken to arrive to a certain predefined time. This property has already been demonstrated and amply investigated in the literature on Semi-Lagrangian schemes mostly in the fields of meteorology and plasma

physics (see e.g. [101, 27, 114, 53, 25, 113, 13] and references therein). Our CLA implementation uses a spectral discretization on a Chebyshev-Fourier grid, as displayed in figure 1.1, the time step is executed by direct summation of the time-Taylor series, and the Poisson problems are solved via sophisticated schemes, which allow to solve directly for derivatives up to second order. One can say that the intersection of the numerical methods used in this thesis to those in [83] is minimal. The absence of a CFL condition makes this algorithm a well-suited candidate for singularity detection because the time steps are not resolution restricted, which is advantageous especially in a turbulent flow. Thus, critical time areas are faster approached and may then be tackled by extrapolation methods (using for instance [93]) or analytic continuations (see Section 5.5). It may be remarked here that so far our findings do not contradict the results of [83].

The CLA implementation. The goal of this first implementation of the CLA in a domain with boundaries is to show its effectiveness and probe the findings of [83]. Over the course of three years, I have programmed an efficient implementation of the CLA which aims to provide for accurate simulations of incompressible Euler flow inside of a cylinder. This practical side of the project consists of more than ten thousand lines of Fortran code to account for several hundred numerical routines, which have all been thoroughly tested and probed for their performance. In addition, a large amount of C, C++, Python, and Mathematica[1] code has been generated in the development and analysis of the used methods. The implementation contains novel algorithms to accurately solve the appearing Poisson problems directly for their second order derivatives. The spectral discretization takes place in a Chebyshev-Fourier space based on a shifted Chebyshev extrema grid in the radial direction and a uniform Fourier grid in the vertical direction. The grid sizes are fixed, but methods on refinement techniques are discussed, so that it is in principle possible to continue computations on higher resolutions when necessary. The overall algorithm can be manipulated by various parameters on grid sizes, Poisson solvers, interpolation schemes (with refinements), de-aliasing, parallelization, time-Taylor truncation orders, time-stepping procedures, and so on. Furthermore, the program is functional with most of the parameter settings in different working precisions, namely double, long double, and quadruple precision, see Section 6.1 for an explanation of those terms.

Outline of this thesis. The Cauchy-Lagrange algorithm is described for a general bounded domain in the beginning of Chapter 2. In the sequel of the aforementioned chapter, the algorithm is derived in a cylindrical coordinate system. The concerning notation and more information on axisymmetric flow in cylindrical coordinates is given in Section 2.2. The complete CLA in a cylinder is discussed in detail in Section 2.6. Chapter 3 introduces the spectral discretization and states solving methods for the Poisson problems, which are tested subsequently. Scattered interpolation methods to regain the flow field on the fixed grid from the Lagrangian distorted grid can be found in Chapter 4 together with results from computational tests. Chapter 5 presents possible time-stepping procedures and treats subjects on the convergence radius of (1.1), fixed and adaptive time-stepping, as well as analytic continuation techniques. For testing purposes, stationary solutions are recalled in Chapter 6 and simulated by the CLA together with swirl-free flows that have minimal regularity. Results on the potentially singular flow of [83] can be found in Chapter 7. Conclusions are drawn in Chapter 8, where possible future work is also discussed.

[1] Wolfram Research, Inc., Mathematica, Version 12.1, Champaign, IL (2020)

Chapter 2

The Cauchy-Lagrange Algorithm

2.1 Description of the CLA in a bounded domain

Short Description. The algorithm exploits the Cauchy-invariants formulation of the incompressible Euler equation, which, together with the incompressibility condition $\det(\mathrm{D}X) = 1$, translates to recursion relations for the divergence and rotational[1] of the time-Taylor coefficients $X^{(s)}$ of the particle trajectories X, solution to eq. (1.3). The recursion only involves expressions that consist of coefficients of lower order and starts with $X^{(1)} = v_0$, the initial velocity. The expressions for the divergence and rotational of $X^{(s)}$ are then used to obtain the coefficient $X^{(s)}$ itself via a Helmholtz-Hodge decomposition. Once sufficiently many coefficients have been calculated, a suitable time-step is executed in summing the time-Taylor series directly. The velocity and vorticity field are then readily computed on a distorted grid and interpolated to the original grid before a new recursion cycle starts.

Step-by-Step Description: From [12], we gather the recursion relations obtained from the Cauchy-invariants formulation in Cartesian coordinates. We abbreviate the appearing sums by calligraphic capital letters $\mathcal{F}$ and $\mathcal{G}$ to simplify the notation. Note that in the aforementioned reference the recurrence relations are given in terms of the displacement map $\xi = X - \mathrm{Id}$ (Id denotes the identity map $a \mapsto a$), but since the time-Taylor coefficients are time derivatives at $t = 0$, we have $\xi^{(s)} = X^{(s)}$, for $s > 0$. The first recursion step starts with $X^{(1)} = v_0$, and therefore, $\nabla \cdot X^{(1)} = 0$ and $-\nabla \times X^{(1)} = -\omega_0$, where v_0 and ω_0 are the initial velocity and vorticity field, and ∇ denotes the Nabla-operator. For higher orders ($s > 1$), we have

$$\nabla \cdot X^{(s)} = \mathcal{F}\left((\mathrm{D}X^{(s')})_{0<s'<s} \right) \tag{2.1}$$

$$-\nabla \times X^{(s)} = \mathcal{G}\left((\mathrm{D}X^{(s')})_{0<s'<s} \right) . \tag{2.2}$$

Subsequently, the coefficient is expressed by a Helmholtz-Hodge decomposition $X^{(s)} = \nabla\phi + \nabla \times \Psi$, where the potentials result from two Poisson problems, namely $\Delta\,\phi = \nabla \cdot X^{(s)}$ and $\Delta\,\Psi = -\nabla \times X^{(s)}$, supplemented with appropriate boundary conditions that are specified in Section 2.5. After a user defined number S of recursions, we may proceed with computing

$$X(a, \triangle t) \approx \sum_{s=0}^{S} X^{(s)}(a)\,(\triangle t)^s, \tag{2.3}$$

[1] also known as curl in $\mathbb{R}^3$

with $X^{(0)} = \mathrm{Id}$, provided a suitable time-step has been found. The only limitation of the latter is the radius of convergence of the time Taylor series. Unfortunately, no concise formula for this radius has yet been found. We have achieved good results in setting the time-step at run-time small enough such that the highest order summands do not contribute to the Taylor sum anymore due to limitations in the machine-precision computations. A detailed inspection of the time-stepping is carried out in Chapter 5.
Once the truncated time-Taylor series (2.3) has been summed, we obtain the velocity field on the trajectories via the characteristic equation

$$v\left(X\left(a, \triangle t\right), \triangle t\right) = \partial_t X\left(a, \triangle t\right) \approx \sum_{s=1}^{S} s X^{(s)}\left(a\right) \left(\triangle t\right)^{s-1}. \tag{2.4}$$

The vorticity is similarly obtained in using the vorticity transport formula, as defined in [84, p. 20],

$$\omega\left(X\left(a, \triangle t\right), \triangle t\right) = \mathrm{D}X(a, \triangle t) \cdot \omega_0(a). \tag{2.5}$$

Numerically, the characteristics deform the underlying discrete grid. Because of eqs. (2.4) and (2.5), the velocity and vorticity fields are then only known on that distorted grid. Hence, an interpolation is needed. In principle, this scattered interpolation has to be performed only for the vorticity field to start a new recursion cycle, but it is better to interpolate the velocity field as well, as it reappears in the first-order term in the next time-Taylor series of the characteristics. The interpolated flow fields act as new data and, therefore, the time-Taylor series eq. (1.1) is always developed at $t_0 = 0$.
In this work, an axisymmetric and periodic flow is modeled in a cylindrical domain to understand the challenges and advantages of this novel algorithm for wall-bounded incompressible Euler flow. The axisymmetry of the problem reduces both the Poisson solvers and the interpolation procedure to two dimensions. For this particular flow, the CLA is presented in greater detail, together with a complete mathematical description, in Section 2.6.

2.2 Vector Fields in Cylindrical Coordinates

Let us introduce the cylindrical coordinate system that is used throughout this work as well as the notation for the appearing vector fields in cylindrical notation. The modeled cylindrical domain is denoted by

$$D(1, L) \coloneqq \left\{ a = (a_1, a_2, a_3) \in \mathbb{R} \mid \sqrt{a_1^2 + a_2^2} \leq 1 \text{ and } a_3 \in [0, L] \right\}, \tag{2.6}$$

where $L > 0$ is arbitrary. Note that the flow is defined in the infinite cylinder $\{a = (a_1, a_2, a_3) \in \mathbb{R} \mid \sqrt{a_1^2 + a_2^2} \leq 1\}$ with periodicity length L. For notational convenience, this cylindrical domain is sometime simply referred to as the cylinder. In $D(1, L)$, the following change of variables is in place,

$$a_1 = r\cos\alpha, \quad a_2 = r\sin\alpha, \quad a_3 = z,$$

such that

$$r = \sqrt{a_1^2 + a_2^2}, \quad \alpha = \mathcal{A}(a_1, a_2), \quad z = a_3,$$

with $(r, \alpha, z) \in [0,1] \times (-\pi, \pi] \times [0, L]$ and

$$\mathcal{A}(a_1, a_2) \coloneqq \begin{cases} 0, & \text{if } a_1 = a_2 = 0 \\ \arcsin(\frac{a_2}{r}), & \text{if } a_1 = 0, a_2 \neq 0 \\ \arctan(\frac{a_2}{a_1}), & \text{if } a_1 > 0 \\ \pi - \arcsin(\frac{a_2}{r}), & \text{if } a_1 < 0\,. \end{cases} \tag{2.7}$$

The canonical Cartesian basis vectors e_1, e_2, e_3 get transformed into the local basis

$$e_r = \begin{pmatrix} \cos\alpha \\ \sin\alpha \\ 0 \end{pmatrix}, \quad e_\alpha = \begin{pmatrix} -\sin\alpha \\ \cos\alpha \\ 0 \end{pmatrix}, \quad e_z = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

The solution X of the characteristic equation (1.3) may be written in two basis representations, the initial point coordinate basis and the local coordinate basis, namely

$$X(r, \alpha, z, t) = X^r(r, z, t)e_r + X^\alpha(r, z, t)e_\alpha + X^z(r, z, t)e_z \tag{2.8}$$

$$= \tilde{R}(r, z, t)e_{\tilde{R}} + \tilde{Z}(r, z, t)e_z, \tag{2.9}$$

where

$$\tilde{R} \coloneqq \sqrt{(X^r)^2 + (X^\alpha)^2}, \quad \tilde{\Theta} \coloneqq \mathcal{A}\left(X^r, X^\alpha\right) + \alpha, \quad \tilde{Z} \coloneqq X^z\,. \tag{2.10}$$

and $e_{\tilde{R}} = (\cos\tilde{\Theta}, \sin\tilde{\Theta}, 0)^\top$, $e_{\tilde{\Theta}} = (-\sin\tilde{\Theta}, \cos\tilde{\Theta}, 0)^\top$ and e_z as before. Since $X^\alpha(r, z, 0) = 0$, the two representations in eqs. (2.8) and (2.9) coincide for $t = 0$. The component conversion, from a Cartesian to a cylindrical representation, is given in the form of a rotation of the component vectors by the respective angle about the z-axis, that is

$$\mathcal{R}_0 \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = \mathcal{R}_\alpha \cdot \begin{pmatrix} X^r \\ X^\alpha \\ X^z \end{pmatrix} = \mathcal{R}_{\tilde{\Theta}} \cdot \begin{pmatrix} \tilde{R} \\ 0 \\ \tilde{Z} \end{pmatrix}, \tag{2.11}$$

where

$$\mathcal{R}_\theta \coloneqq \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.12}$$

for $\theta \in \mathbb{R}$. Using the property $\mathcal{R}_\alpha \cdot \mathcal{R}_\theta = \mathcal{R}_{\alpha+\theta}$, one may conveniently translate between the different representations in eq. (2.11) by multiplications with the rotation matrices. To obtain $\tilde{R}, \tilde{Z}$ from X^r, X^α, X^z one simply multiplies eq. (2.11) by $\mathcal{R}_{-\tilde{\Theta}}$ and obtains

$$(\tilde{R}, 0, \tilde{Z})^\top = \mathcal{R}_{\alpha-\tilde{\Theta}} \cdot (X^r, X^\alpha, X^z)^\top, \tag{2.13}$$

which entails an alternative formulation of $\tilde{R}$ to that in eq. (2.10), namely

$$\begin{aligned} \tilde{R} &= X^r \cos\left(\alpha - \tilde{\Theta}\right) - X^\alpha \sin\left(\alpha - \tilde{\Theta}\right) \\ &= X^r \cos\left(\mathcal{A}\left(X^r, X^\alpha\right)\right) + X^\alpha \sin\left(\mathcal{A}\left(X^r, X^\alpha\right)\right). \end{aligned} \tag{2.14}$$

The incompressible Euler equations, eqs. (1.2), in cylindrical coordinates read componentwise

$$\partial_t v^r + v^r \partial_r v^r + v^z \partial_z v^r - \frac{v^2}{r} = -\partial_r p, \tag{2.15}$$

$$\partial_t v^\alpha + v^r \partial_r v^\alpha + v^z \partial_z v^\alpha + \frac{v^r v^\alpha}{r} = 0, \tag{2.16}$$

$$\partial_t v^z + v^r \partial_r v^z + v^z \partial_z v^z = -\partial_z p, \tag{2.17}$$

$$\partial_r v^r + \frac{v^r}{r} + \partial_z v^z = 0, \tag{2.18}$$

for an axi-symmetric velocity field in cylindrical coordinates

$$v(r, \alpha, z, t) = v^r(r, z, t) e_r + v^\alpha(r, z, t) e_\alpha + v^z(r, z, t) e_z \,. \tag{2.19}$$

Equation (2.18) denotes the incompressibility condition $\nabla_c \cdot v = 0$, where

$$\nabla_c \coloneqq e_r \partial_r + \tfrac{1}{r} e_\alpha \partial_\alpha + e_z \partial_z \tag{2.20}$$

is the Nabla-operator in cylindrical coordinates. In $D(1, L)$, the no flow boundary condition becomes simply

$$v \cdot e_r = v^r = 0, \qquad \text{on } \partial D(1, L), \tag{2.21}$$

and the flow components are periodic in vertical direction with periodicity length L. The velocity field in cylindrical coordinates, evaluated on the particle trajectories at $t \geq 0$, takes the form

$$v(\tilde{R}, \tilde{\Theta}, \tilde{Z}, t) = v^r(\tilde{R}, \tilde{Z}, t) e_{\tilde{R}} + v^\alpha(\tilde{R}, \tilde{Z}, t) e_{\tilde{\Theta}} + v^z(\tilde{R}, \tilde{Z}, t) e_z \,. \tag{2.22}$$

The components of v may be obtained from the characteristic equation (1.3). The latter reads in local cylindrical coordinates

$$\partial_t (\tilde{R} e_{\tilde{R}} + \tilde{Z} e_z) = v(\tilde{R}, \tilde{\Theta}, \tilde{Z}, t), \tag{2.23}$$

with

$$\partial_t (\tilde{R} e_{\tilde{R}} + \tilde{Z} e_z) = \dot{\tilde{R}} e_{\tilde{R}} + \tilde{R} \dot{\tilde{\Theta}} e_{\tilde{\Theta}} + \dot{\tilde{Z}} e_z \,. \tag{2.24}$$

On the other hand, in using eq. (2.11), the left hand side (LHS) of eq. (2.23) may be replaced by

$$\partial_t (X^r e_r + X^\alpha e_\alpha + X^z e_z) = \dot{X}^r e_r + \dot{X}^\alpha e_\alpha + \dot{X}^z e_z \tag{2.25}$$

to yield

$$\mathcal{R}_\alpha \cdot (\dot{X}^r, \dot{X}^\alpha, \dot{X}^z)^\top = \mathcal{R}_{\tilde{\Theta}} \cdot (v^r(\tilde{R}, \tilde{Z}, t), v^\alpha(\tilde{R}, \tilde{Z}, t), v^z(\tilde{R}, \tilde{Z}, t))^\top \,. \tag{2.26}$$

Equations (2.23) and (2.24) imply, in conjunction with eqs. (2.10) and (2.22),

$$\begin{aligned} v^r(\tilde{R}, \tilde{Z}, t) &= \dot{\tilde{R}} = \frac{X^r \dot{X}^r + X^\alpha \dot{X}^\alpha}{\sqrt{(X^r)^2 + (X^\alpha)^2}}, \\ v^\alpha(\tilde{R}, \tilde{Z}, t) &= \tilde{R} \dot{\tilde{\Theta}} = \frac{X^r \dot{X}^\alpha - X^\alpha \dot{X}^r}{\sqrt{(X^r)^2 + (X^\alpha)^2}}, \\ v^z(\tilde{R}, \tilde{Z}, t) &= \dot{\tilde{Z}} = \dot{X}^z, \end{aligned} \tag{2.27}$$

provided $(X^r)^2 + (X^\alpha)^2 > 0$, otherwise $v^r = v^\alpha = 0$. Alternatively, $v^r(\tilde{R}, \tilde{Z}, t)$ and $v^\alpha(\tilde{R}, \tilde{Z}, t)$ may be obtained by a left multiplication of the rotation matrix $\mathcal{R}_{-\tilde{\Theta}}$ to both sides of eq. (2.26),

producing

$$
\begin{aligned}
v^r(\tilde{R},\tilde{Z},t) &= \dot{X}^r\cos\left(\alpha-\tilde{\Theta}\right)-\dot{X}^\alpha\sin\left(\alpha-\tilde{\Theta}\right)\\
&= \dot{X}^r\cos\left(\mathcal{A}\left(X^r,X^\alpha\right)\right)+\dot{X}^\alpha\sin\left(\mathcal{A}\left(X^r,X^\alpha\right)\right), \qquad (2.28)\\
v^\alpha(\tilde{R},\tilde{Z},t) &= \dot{X}^r\sin\left(\alpha-\tilde{\Theta}\right)+\dot{X}^\alpha\cos\left(\alpha-\tilde{\Theta}\right)\\
&= -\dot{X}^r\sin\left(\mathcal{A}\left(X^r,X^\alpha\right)\right)+\dot{X}^\alpha\cos\left(\mathcal{A}\left(X^r,X^\alpha\right)\right), \qquad (2.29)\\
v^z(\tilde{R},\tilde{Z},t) &= \dot{X}^z. \qquad (2.30)
\end{aligned}
$$

The Nabla operator from eq. (2.20) may be applied to the cylindrical formulation of X, given as in (2.8), to entail the gradients of the Cartesian components of X in cylindrical coordinates

$$
\begin{aligned}
\nabla X_1 &= (X_r^r\cos\alpha - X_r^\alpha\sin\alpha)e_r+\frac{1}{r}(-X^r\sin\alpha - X^\alpha\cos\alpha)e_\alpha+(X_z^r\cos\alpha - X_z^\alpha\sin\alpha)e_z,\\
\nabla X_2 &= (X_r^r\sin\alpha + X_r^\alpha\cos\alpha)e_r+\frac{1}{r}(X^r\cos\alpha - X^\alpha\sin\alpha)e_\alpha+(X_z^r\sin\alpha + X_z^\alpha\cos\alpha)e_z,\\
\nabla X_3 &= X_r^z e_r + X_z^z e_z,
\end{aligned}
$$

where the subscripts $(\,.\,)_r$ and $(\,.\,)_z$ on the components of X indicate partial derivatives with respect to r and z respectively. The angular derivatives (with respect to α) vanish on X^r, X^α and X^z due to the assumed axisymmetry, and act only on the basis vectors e_r and e_α. Furthermore, from the gradient expressions above we obtain the relations

$$
\begin{aligned}
\partial_{a_1}X_1 &= X_r^r\cos^2\alpha+\tfrac{1}{r}X^r\sin^2\alpha+(\tfrac{1}{r}X^\alpha - X_r^\alpha)\cos\alpha\sin\alpha,\\
\partial_{a_2}X_1 &= -X_r^\alpha\sin^2\alpha+(X_r^r-\tfrac{1}{r}X^r)\cos\alpha\sin\alpha-\tfrac{1}{r}X^\alpha\cos^2\alpha,\\
\partial_{a_3}X_1 &= X_z^r\cos\alpha - X_z^\alpha\sin\alpha,\\
\partial_{a_1}X_2 &= X_r^\alpha\cos^2\alpha+(X_r^r-\tfrac{1}{r}X^r)\cos\alpha\sin\alpha+\tfrac{1}{r}X^\alpha\sin^2\alpha,\\
\partial_{a_2}X_2 &= X_r^r\sin^2\alpha+\tfrac{1}{r}X^r\cos^2\alpha+(X_r^\alpha-\tfrac{1}{r}X^\alpha)\cos\alpha\sin\alpha,\\
\partial_{a_3}X_2 &= X_z^r\sin\alpha + X_z^\alpha\cos\alpha,\\
\partial_{a_1}X_3 &= X_r^z\cos\alpha,\\
\partial_{a_2}X_3 &= X_r^z\sin\alpha,\\
\partial_{a_3}X_3 &= X_z^z\,. \qquad (2.31)
\end{aligned}
$$

The vorticity ω is given in the same notation as v, namely

$$
\begin{aligned}
\omega(r,\alpha,z,t) &= \omega^r(r,z,t)e_r+\omega^\alpha(r,z,t)e_\alpha+\omega^z(r,z,t)e_z, \qquad (2.32)\\
\omega(\tilde{R},\tilde{\Theta},\tilde{Z},t) &= \omega^r(\tilde{R},\tilde{Z},t)e_{\tilde{R}}+\omega^\alpha(\tilde{R},\tilde{Z},t)e_{\tilde{\Theta}}+\omega^z(\tilde{R},\tilde{Z},t)e_z\,. \qquad (2.33)
\end{aligned}
$$

In order to express the vorticity in terms of the Lagrangian components X^r, X^α and X^z, we use the vorticity transport formula as given in [84, p. 20], here in Cartesian coordinates with initial vorticity ω_0,

$$
\omega\left(X\left(a,t\right),t\right) = \mathrm{D}X(a,t)\cdot\omega_0(a)\,. \qquad (2.34)
$$

Equations (2.31) inserted into $\mathrm{D}X \cdot \omega_0$, where $\omega_0 = \omega_0^r e_r + \omega_0^\alpha e_\alpha + \omega_0^z e_z$, gives

$$\begin{pmatrix} (\omega_0^r \cos\alpha - \omega_0^\alpha \sin\alpha)\left(\left(\frac{X^\alpha}{r} - X_r^\alpha\right)\sin\alpha\cos\alpha + \frac{X^r \sin^2\alpha}{r} + X_r^r \cos^2\alpha\right) + (\omega_0^\alpha \cos\alpha + \omega_0^r \sin\alpha) \\ \times\left(-X_r^\alpha \sin^2\alpha - \frac{X^\alpha \cos^2\alpha}{r} + \left(X_r^r - \frac{X^r}{r}\right)\sin\alpha\cos\alpha\right) + \omega_0^z\left(X_z^r \cos\alpha - X_z^\alpha \sin\alpha\right) \\ (\omega_0^\alpha \cos\alpha + \omega_0^r \sin\alpha)\left(\left(X_r^\alpha - \frac{X^\alpha}{r}\right)\sin\alpha\cos\alpha + X_r^r \sin^2\alpha + \frac{X^r \cos^2\alpha}{r}\right) + (\omega_0^r \cos\alpha - \omega_0^\alpha \sin\alpha) \\ \times\left(\frac{X^\alpha \sin^2\alpha}{r} + X_r^\alpha \cos^2\alpha + \left(X_r^r - \frac{X^r}{r}\right)\sin\alpha\cos\alpha\right) + \omega_0^z\left(X_z^\alpha \cos\alpha + X_z^r \sin\alpha\right) \\ X_r^z \sin\alpha\left(\omega_0^\alpha \cos\alpha + \omega_0^r \sin\alpha\right) + X_r^z \cos\alpha\left(\omega_0^r \cos\alpha - \omega_0^\alpha \sin\alpha\right) + \omega_0^z X_z^z \end{pmatrix}, \tag{2.35}$$

which simplifies to

$$\begin{pmatrix} -\frac{X^\alpha \omega_0^\alpha \cos\alpha}{r} - X_r^\alpha \omega_0^r \sin\alpha - X_z^\alpha \omega_0^z \sin\alpha - \frac{X^r \omega_0^\alpha \sin\alpha}{r} + X_r^r \omega_0^r \cos\alpha + X_z^r \omega_0^z \cos\alpha \\ -\frac{X^\alpha \omega_0^\alpha \sin\alpha}{r} + X_r^\alpha \omega_0^r \cos\alpha + X_z^\alpha \omega_0^z \cos\alpha + \frac{X^r \omega_0^\alpha \cos\alpha}{r} + X_r^r \omega_0^r \sin\alpha + X_z^r \omega_0^z \sin\alpha \\ \omega_0^r X_r^z + \omega_0^z X_z^z \end{pmatrix}, \tag{2.36}$$

such that we may infer from eq. (2.34),

$$\mathcal{R}_{\tilde{\Theta}} \cdot \begin{pmatrix} \omega^r(\tilde{R},\tilde{Z},t) \\ \omega^\alpha(\tilde{R},\tilde{Z},t) \\ \omega^z(\tilde{R},\tilde{Z},t) \end{pmatrix} = \mathcal{R}_\alpha \cdot \begin{pmatrix} -\frac{X^\alpha \omega_0^\alpha}{r} + X_r^r \omega_0^r + X_z^r \omega_0^z \\ X_r^\alpha \omega_0^r + X_z^\alpha \omega_0^z + \frac{X^r \omega_0^\alpha}{r} \\ \omega_0^r X_r^z + \omega_0^z X_z^z \end{pmatrix}. \tag{2.37}$$

The result is now obtained in left multiplying eq. (2.37) by $\mathcal{R}_{-\tilde{\Theta}}$ and replacing $\alpha - \tilde{\Theta} = \mathcal{A}(X^r, X^\alpha)$, as in eqs. (2.28) and (2.29), to give the components of ω as

$$\begin{aligned} \omega^r(\tilde{R},\tilde{Z},t) &= \cos(\mathcal{A}(X^r,X^\alpha))\left(-\frac{X^\alpha \omega_0^\alpha}{r} + X_r^r\ \omega_0^r + X_z^r \omega_0^z\right) \\ &\quad + \sin(\mathcal{A}(X^r,X^\alpha))\left(X_r^\alpha\ \omega_0^r + X_z^\alpha \omega_0^z + \frac{X^r \omega_0^\alpha}{r}\right), \qquad (2.38) \\ \omega^\alpha(\tilde{R},\tilde{Z},t) &= -\sin(\mathcal{A}(X^r,X^\alpha))\left(-\frac{X^\alpha \omega_0^\alpha}{r} + X_r^r\ \omega_0^r + X_z^r \omega_0^z\right) \\ &\quad + \cos(\mathcal{A}(X^r,X^\alpha))\left(X_r^\alpha\ \omega_0^r + X_z^\alpha \omega_0^z + \frac{X^r \omega_0^\alpha}{r}\right), \qquad (2.39) \\ \omega^z(\tilde{R},\tilde{Z},t) &= \omega_0^r X_r^z + \omega_0^z X_z^z\,. \qquad (2.40) \end{aligned}$$

Remark. Numerically, the terms $X^r/r, X^\alpha/r$, which appear in the formulas of the vorticity field as well as in the definition of $\mathcal{A}$, together with the division X^α/X^r, may introduce cancellation errors in their numerical computation. Therefore, new variables are introduced, namely

$$X_1^r := \frac{X^r}{r}, \quad X_1^\alpha := \frac{X^\alpha}{r}, \tag{2.41}$$

such that $X^\alpha/X^r = X_1^\alpha/X_1^r$. In our CLA implementation, X_1^r and X_1^α are obtained without explicit numerical divisions by r, but are rather directly constructed. We refer the reader to Section 2.6 for more details on the employed techniques.

2.3 The Cauchy-Invariants Formula in Cylindrical Coordinates

In order to make use of the imposed rotational invariance of the solution to the Euler equations in a given cylindrical domain, we are going to transform the Cauchy-invariants equation, given in Cartesian coordinates in [12], into its cylindrical formulation. By the virtue of this computation, one obtains the axisymmetric recurrence relations for the divergence and rotation of the trajectory's time-Taylor coefficients, denoted as $X^{(s)}$, $s \geq 0$. These recursion relations, supplemented by the Poisson equations for the cylindrical versions of the Helmholtz-Hodge potentials assigned to the time-Taylor coefficients of the trajectory, give rise to the explicit computations of the coefficients $X^{(s)}$.
The computations at hand are executed in a straight forward way and do not require any particular knowledge of the Euler equations or its Cauchy-invariants reformulation. The latter reads

$$\sum_{k=1,2,3} \nabla_a \dot{X}_k(a,t) \times \nabla_a X_k(a,t) = \omega_0(a), \tag{2.42}$$

$$\det\left(\mathrm{D}_a X(a,t)\right) = 1, \tag{2.43}$$

in the Cartesian coordinates $a = (a_1, a_2, a_3)^\top$ with $\dot{X} \coloneqq \partial_t X$. Equations (2.42) and (2.43) get supplemented by the boundary condition

$$\mathcal{S}(X(a,t)) = 0, \tag{2.44}$$

for a boundary point a. Here, $\mathcal{S}$ denotes the boundary map, such that $\ker(\mathcal{S})$, the kernel of $\mathcal{S}$, defines the boundary of the cylindrical domain. In our particular case of a periodic cylinder with radius 1, the boundary map is defined by $\mathcal{S}(a) \coloneqq a_1^2 + a_2^2 - 1$. Thus, eq. (2.44) becomes

$$\mathcal{S}(X(a,t)) = X_1(a,t)^2 + X_2(a,t)^2 - 1 = 0, \tag{2.45}$$

for a boundary coordinate a. We introduce the radial variable $r \in [0,1]$, the angular variable $\alpha \in [0, 2\pi)$ and the vertical variable $z \in [0, L]$, where $L > 0$ assigns a length of periodicity in vertical direction, and will occasionally be referred to as the height of the cylinder. The displacement map, $\xi = \xi(a,t) \coloneqq X(a,t) - a$, in cylindrical coordinates reads

$$\xi(a,t) = R(r,z,t)e_r + A(r,z,t)e_\alpha + Z(r,z,t)e_z, \tag{2.46}$$

such that R, A and Z represent the radial, angular and vertical components of ξ, which gives

$$X(a,t) = (R(r,z,t) + r)e_r + A(r,z,t)e_\alpha + (Z(r,z,t) + z)e_z\,. \tag{2.47}$$

We proceed with the transformation of the Cauchy-invariants formula (2.42) into its cylindrical form. This is done by directly inserting the cylindrical gradients into eq. (2.42). The Nabla-operator in cylindrical coordinates, $\nabla_c \coloneqq e_r\partial_r + \frac{1}{r}e_\alpha\partial_\alpha + e_z\partial_z$, entails the expressions for the rotation and divergence of the axisymmetric displacement map in cylindrical coordinates

$$\nabla_c \times \xi = -A_z e_r + (R_z - Z_r)e_\alpha + \tfrac{1}{r}(rA)_r e_z, \tag{2.48}$$

$$\nabla_c \cdot \xi = \tfrac{1}{r}(rR)_r + Z_z, \tag{2.49}$$

where the subscripts on A, R and Z indicate partial derivatives in r and z. As there is no angular dependence of the components, the α-derivatives only act on the basis vectors e_r, e_α

and vanish on the components.
The gradients of the Cartesian components, X_k, $k = 1, 2, 3$, are just an application of ∇_c to the right hand sides of eq. (2.47) and read

$$\nabla X_1 = ((R_r + 1)\cos\alpha - A_r \sin\alpha)e_r + \frac{1}{r}(-(R + r)\sin\alpha - A\cos\alpha)e_\alpha + (R_z \cos\alpha - A_z \sin\alpha)e_z$$
$$\nabla X_2 = ((R_r + 1)\sin\alpha + A_r \cos\alpha)e_r + \frac{1}{r}((R + r)\cos\alpha - A\sin\alpha)e_\alpha + (R_z \sin\alpha + A_z \cos\alpha)e_z$$
$$\nabla X_3 = (Z_r)e_r + (Z_z + 1)e_z\,.$$

Equally, their time-derivatives read

$$\nabla \dot{X}_1 = (\dot{R}_r \cos\alpha - \dot{A}_r \sin\alpha)e_r + (\frac{1}{r}(-\dot{R}\sin\alpha - \dot{A}\cos\alpha))e_\alpha + (\dot{R}_z \cos\alpha - \dot{A}_z \sin\alpha)e_z,$$
$$\nabla \dot{X}_2 = (\dot{R}_r \sin\alpha + \dot{A}_r \cos\alpha)e_r + (\frac{1}{r}(\dot{R}\cos\alpha - \dot{A}\sin\alpha))e_\alpha + (\dot{R}_z \sin\alpha + \dot{A}_z \cos\alpha)e_z,$$
$$\nabla \dot{X}_3 = \dot{Z}_r e_r + \dot{Z}_z e_z\,.$$

Departing from eq. (2.42), in using the above gradients together with the identities $e_\alpha \times e_z = e_r,\ e_z \times e_r = e_\alpha,\ e_r \times e_\alpha = e_z$, we have

$$\begin{aligned}
\omega_0(a) = \sum_{k=1,2,3} \nabla \dot{X}_k(a,t) \times \nabla X_k(a,t) = \\
& e_r \Big[(\frac{1}{r}(-\dot{R}\sin\alpha - \dot{A}\cos\alpha))(R_z \cos\alpha - A_z \sin\alpha) \\
& \quad - (\dot{R}_z \cos\alpha - \dot{A}_z \sin\alpha)(\frac{1}{r}(-(R + r)\sin\alpha - A\cos\alpha)) \\
& \quad + (\frac{1}{r}(\dot{R}\cos\alpha - \dot{A}\sin\alpha))(R_z \sin\alpha + A_z \cos\alpha) \\
& \quad - (\dot{R}_z \sin\alpha + \dot{A}_z \cos\alpha)(\frac{1}{r}((R + r)\cos\alpha - A\sin\alpha)) \Big] + \\
& e_\alpha \Big[(\dot{R}_z \cos\alpha - \dot{A}_z \sin\alpha)((R_r + 1)\cos\alpha - A_r \sin\alpha) \\
& \quad - (\dot{R}_r \cos\alpha - \dot{A}_r \sin\alpha)(R_z \cos\alpha - A_z \sin\alpha) \\
& \quad + (\dot{R}_z \sin\alpha + \dot{A}_z \cos\alpha)((R_r + 1)\sin\alpha + A_r \cos\alpha) \\
& \quad - (\dot{R}_r \sin\alpha + \dot{A}_r \cos\alpha)(R_z \sin\alpha + A_z \cos\alpha) \\
& \quad + \dot{Z}_z Z_r - \dot{Z}_r(Z_z + 1) \Big] + \\
& e_z \big[(\dot{R}_r \cos\alpha - \dot{A}_r \sin\alpha)(\frac{1}{r}(-(R + r)\sin\alpha - A\cos\alpha)) \\
& \quad - (\frac{1}{r}(-\dot{R}\sin\alpha - \dot{A}\cos\alpha))((R_r + 1)\cos\alpha - A_r \sin\alpha) \\
& \quad + (\dot{R}_r \sin\alpha + \dot{A}_r \cos\alpha)(\frac{1}{r}((R + r)\cos\alpha - A\sin\alpha)) \\
& \quad - (\frac{1}{r}(\dot{R}\cos\alpha - \dot{A}\sin\alpha))((R_r + 1)\sin\alpha + A_r \cos\alpha) \big] \\
& = e_r \frac{1}{r}\big[\dot{R}A_z - \dot{A}R_z + \dot{R}_z A - \dot{A}_z R - \dot{A}_z\big] \\
& \quad + e_\alpha \big[\dot{R}_z R_r + \dot{A}_z A_r - \dot{R}_r R_z - \dot{A}_r A_z + \dot{Z}_z Z_r - \dot{Z}_r Z_z + \dot{R}_z - \dot{Z}_r\big] \\
& \quad + e_z \frac{1}{r}\big[- \dot{R}_r A + \dot{A}_r R - \dot{R}A_r + \dot{A}R_r + r\dot{A}_r + \dot{A}\big]\,. \qquad (2.50)
\end{aligned}$$

We consequently find, under consideration of eq. (2.48),

$$-\nabla_c \times \dot{\xi} = e_r \frac{1}{r}\partial_z\big(\dot{R}A - \dot{A}R\big) + e_\alpha\big[\big\{\dot{R}, R\big\} + \big\{\dot{A}, A\big\} + \big\{\dot{Z}, Z\big\}\big] + e_z \frac{1}{r}\partial_r\big(\dot{A}R - \dot{R}A\big) + \omega_0, \quad (2.51)$$

where we used Poisson brackets for simplifications, namely

$$\big\{f, g\big\} \coloneqq f_z g_r - f_r g_z \quad (2.52)$$

for differentiable functions $f = f(r, z)$ and $g = g(r, s)$. One may insert the formulas from eq. (2.31) directly into the incompressibility condition eq. (2.43) and simplify until a convenient expression is found. Here, however, we only give the result, as the computations involved are basic, but space demanding,

$$\begin{aligned} 1 = \det\big(\mathrm{D}X(a,t)\big) &= \varepsilon_{ijk}\partial_{a_i}X_1\partial_{a_j}X_2\partial_{a_k}X_3 \\ &= \frac{1}{r}(-AA_zZ_r + AA_rZ_z - RR_zZ_r + RR_rZ_z) \\ &\quad - Z_rR_z + R_rZ_z + \frac{1}{r}(AA_r + RR_r + RZ_z) \\ &\quad + \frac{R}{r} + R_r + Z_z + 1, \end{aligned} \quad (2.53)$$

with the usual summation convention on the Levi-Cevita symbol in the first line. By algebraic manipulations of eq. (2.53) and in consulting eq. (2.49), one finds

$$\begin{aligned} \nabla_c \cdot \xi &= \frac{1}{r}(AA_zZ_r - AA_rZ_z + RR_zZ_r - RR_rZ_z) + Z_rR_z - R_rZ_z - \frac{1}{r}(AA_r + RR_r + RZ_z) \\ &= \frac{1}{r}\left(A\big(\big\{A, Z\big\} - A_r\big) + R\big(\big\{R, Z\big\} - R_r - Z_z\big)\right) + \big\{R, Z\big\}. \end{aligned} \quad (2.54)$$

Finally, using eq. (2.11) and (2.47), the boundary condition eq. (2.45) yields

$$\begin{aligned} 0 = \mathcal{S}(X(a,t)) &= X_1^2(a,t) + X_2^2(a,t) - 1 \\ &= ((R(1,z,t)+1)\cos\alpha + A(1,z,t)\sin\alpha)^2 \\ &\quad + ((R(1,z,t)+1)(-\sin\alpha) + A(1,z,t)\cos\alpha)^2 - 1 \\ &= (R(1,z,t)+1)^2 + A(1,z,t)^2 - 1 \\ &= 2R(1,z,t) + R(1,z,t)^2 + A(1,z,t)^2, \end{aligned} \quad (2.55)\ (2.56)$$

for a boundary point a. In confirming the equivalence of eqs. (2.51), (2.54) and (2.56) to eqs. (2.42) to (2.44), we have found the Cauchy-invariants formulation of the axisymmetric, incompressible Euler equations in cylindrical coordinates. The assumption of axisymmetry has led to simplifications, which will be exploited in the next section, where the recursion relations for the time-Taylor coefficients are stated.

2.4 Recursion Relations

For $t > 0$ sufficiently small, the particle displacement map, defined in eq. (2.46), is given by an absolutely converging time-Tayor series

$$\xi(a,t) = \sum_{s\geq 1} \xi^{(s)}(a)\, t^s \quad (2.57)$$

with coefficients

$$\xi^{(s)} := \frac{1}{s!}\partial_t^s \xi \mid_{t=0} = \frac{1}{s!}\partial_t^s X \mid_{t=0},\ s \geq 1. \tag{2.58}$$

Each component of ξ and $\dot{\xi}$ is represented by its own temporal Taylor series

$$\Lambda(r,z,t) = \sum_{s\geq 1} \Lambda^{(s)}(r,z)\, t^s, \qquad \dot{\Lambda}(r,z,t) = \sum_{s\geq 1} s\Lambda^{(s)}(r,z)\, t^{s-1}, \quad \Lambda = R, A, Z, \tag{2.59}$$

thus, we have

$$\xi^{(s)} = R^{(s)}(r,z)e_r + A^{(s)}(r,z)e_\alpha + Z^{(s)}(r,z)e_z, \quad s \geq 1\,. \tag{2.60}$$

Since the time derivatives of ξ equal those of X, the time-Taylor series of X is given by

$$X(a,t) = a + \xi(a,t) = re_r + ze_z + \sum_{s\geq 1}\left(R^{(s)}(r,z)e_r + A^{(s)}(r,z)e_\alpha + Z^{(s)}(r,z)e_z\right) t^s\,.$$

In defining

$$X^{(s)} := \xi^{(s)},\ \text{for } s \geq 1, \quad \text{and} \quad X^{(0)} := R^{(0)}e_r + A^{(0)}(r,z)e_\alpha + Z^{(0)}(r,z)e_z, \tag{2.61}$$

with $R^{(0)}(r,z) := r,\ A^{(0)}(r,z) := 0,\ Z^{(0)}(r,z) := z$, we may write

$$\begin{aligned} X(a,t) &= \sum_{s\geq 0} X^{(s)}(a)\, t^s \\ &= \sum_{s\geq 0}\left(R^{(s)}(r,z)e_r + A^{(s)}(r,z)e_\alpha + Z^{(s)}(r,z)e_z\right) t^s\,. \end{aligned} \tag{2.62}$$

If $s = 1$, then $\xi^{(1)} = \dot{X}(a,0) = v_0(a)$ and, therefore,

$$\nabla_c \times \xi^{(1)} = \omega_0 = -\partial_z v_0^\alpha\ e_r + (\partial_z v_0^r + \partial_r v_0^z)\ e_\alpha + \frac{1}{r}(rv_0^\alpha)_r\ e_z\,. \tag{2.63}$$

It is now in line to find an expression for $-\nabla_c \times \dot{\xi}^{(s)}$ for $s > 0$. In following [12], we insert the power series (2.59) into eq. (2.51) and obtain for the radial component, indicated by superscript r,

$$\left(-\nabla_c \times \dot{\xi}\ \right)^r = \left(-\nabla_c \times \sum\nolimits_{s\geq 1} s\xi^{(s)}t^{s-1}\right)^r = \sum_{s\geq 1} s\,(-\nabla_c \times \xi^{(s)})^r t^{s-1} \tag{2.64}$$

$$= \frac{1}{r}\partial_z \sum_{s\geq 2}\left(\sum_{1\leq m<s}\left[mR^{(m)}A^{(s-m)}\right] - \sum_{1\leq m<s}\left[mA^{(m)}R^{(s-m)}\right]\right) t^{s-1} + \omega_0^r\,. \tag{2.65}$$

In collecting powers of t and applying the same procedure to the angular and vertical components (superscripts α and z resp.), we find

$$(-\nabla_c \times \xi^{(s)})^r = \sum_{1\leq m<s} \frac{m}{rs}\partial_z\left[R^{(m)}A^{(s-m)} - A^{(m)}R^{(s-m)}\right] \tag{2.66}$$

$$(-\nabla_c \times \xi^{(s)})^\alpha = \sum_{1\leq m<s} \frac{m}{s}\left[\left\{R^{(m)}, R^{(s-m)}\right\} + \left\{A^{(m)}, A^{(s-m)}\right\} + \left\{Z^{(m)}, Z^{(s-m)}\right\}\right] \tag{2.67}$$

$$(-\nabla_c \times \xi^{(s)})^z = \sum_{1\leq m<s} \frac{m}{rs}\partial_r\left[A^{(m)}R^{(s-m)} - R^{(m)}A^{(s-m)}\right]. \tag{2.68}$$

Equally, one may express $\nabla_c \cdot \xi^{(s)}$ in terms of $A^{(s')}, R^{(s')}$ and $Z^{(s')}$ with $s' < s$. The divergence expressions are obtained in a similar fashion as before by inserting eq. (2.59) into eq. (2.54), which results in

$$\begin{aligned}\nabla_c \cdot \xi^{(s)} = \\ \sum_{1\le m<s} \frac{A^{(m)}}{r}\left(\{A,Z\}^{(s-m)} - A_z^{(s-m)}\right) \\ + \frac{R^{(m)}}{r}\left(\{R,Z\}^{(s-m)} - R_z^{(s-m)} - Z_r^{(s-m)}\right) + \{R,Z\}^{(s)}\end{aligned} \tag{2.69}$$

for $s \ge 1$, with

$$\{\Lambda,Z\}^{(n)} := \sum_{1\le m<n}\left(\Lambda_z^{(m)} Z_r^{(n-m)} - \Lambda_r^{(m)} Z_z^{(s-m)}\right), \quad n = 1,2,\dots, \tag{2.70}$$

such that $\{\Lambda,Z\} = \sum_{1\le s}\{\Lambda,Z\}^{(s)} t^s$ for $\Lambda = R, A$. By convention, empty sums evaluate to zero. An algorithmic aspect of the formulation of the recurrence sum above with the terms in (2.70) is the reduction of computational complexity when the terms $\{\Lambda,Z\}^{(n)}$ are stored instead of being recomputed every time. This storage allows to compute eq. (2.69) in linear complexity instead of a quadratic one. A verification of the above formulas for the rotation and divergence of $\xi^{(s)}$ consists in applying the more intricate method of directly transforming the recursion relations from [12] into cylindrical coordinates via eqs. (2.31).
The boundary term eq. (2.56), being equivalent to

$$R(1,z,t) = -\frac{1}{2}\left(R(1,z,t)^2 + A(1,z,t)^2\right),$$

becomes at the points $(1,z)$ after insertion of the power series of the components eq. (2.59)

$$\sum_{s\ge 0} R^{(s)} t^s = -\frac{1}{2}\sum_{s\ge 0}\left(\sum_{n=1}^{s-1} R^{(n)} R^{(s-n)} + A^{(n)} A^{(s-n)}\right) t^s .$$

This entails a recursion formula for the boundary value of $R^{(s)}$, namely

$$R^{(s)}(1,z) = -\frac{1}{2}\left(\sum_{n=1}^{s-1} R^{(n)} R^{(s-n)} + A^{(n)} A^{(s-n)}\right)(1,z), \quad s \ge 1 . \tag{2.71}$$

All the recursion formulas that are needed for the implementation of the CLA have been established in cylinrical coordinates, and the RHSs of eqs. (2.66) to (2.69) and (2.71) consist only of orders inferior to s. These recursions for the rotation, divergence and boundary terms allow to proceed with the cylindrical Helmholtz-Hodge decomposition of $\xi^{(s)}$.

2.5 Helmholtz-Hodge Decomposition

In order to exploit the formerly found recursion relations of the divergence and rotation we state the Helmholtz-Hodge decomposition (HHD) of the trajectory's time-Taylor coefficients in cylindrical coordinates. In the following, let $s \ge 1$ be fixed. $\xi^{(s)}$ can be expressed by a HHD in the cylindrical domain $D(1,L)$ as

$$\xi^{(s)} = \nabla\phi + \nabla \times \Psi . \tag{2.72}$$

The potentials ϕ and Ψ should read $\phi^{(s)}$ and $\Psi^{(s)}$, since those potentials depend on s. We omit those superscripts only to increase the readability. The scalar potential ϕ is uniquely described by the Poisson problem

$$\begin{aligned} \Delta\,\phi &= \nabla\cdot\xi^{(s)}, \quad && \text{in } D(1,L),\\ \partial_\nu\phi &= \nu\cdot\xi^{(s)}, && \text{on } \partial D(1,L), \end{aligned} \tag{2.73}$$

and the vector potential Ψ by

$$\begin{aligned} \Delta\,\Psi &= -\nabla\times\xi^{(s)},\\ \nabla\cdot\Psi &= 0, \quad && \text{in } D(1,L),\\ \Psi\times\nu &= 0, && \text{on } \partial D(1,L), \end{aligned} \tag{2.74}$$

where ν denotes the unit outward boundary vector and in our case simply e_r. See [58] for a detailed analysis of the HHD and the above statements for a bounded domain. The fact that our domain is unbounded in vertical direction is compensated by the imposed periodicity of the solutions. The validity of the above statement is verified without difficulty by a slight adaptation of the arguments in [58]. It may be remarked here, that [12] contains a slightly inaccurate statement on the HHD that is corrected here. Indeed, $\Psi = 0$ on the boundary of the domain is not sufficient for a general HHD in a bounded domain but accounts only for a special situation. The correctness of the proof in [12] is unaffected by this change of boundary conditions because the appearing Hölder estimates still hold in this case. For a much more general version of the proof in [12], we refer the reader to [11], where the boundary conditions from above have been addressed in Remark 6. We reformulate the above problems in cylindrical coordinates $a = (a_1, a_2, a_3) \to (r\cos\alpha, r\sin\alpha, z)$, and keep the symbols for the vector potentials. Thus, we are looking for functions $\phi = \phi(r,z)$ and $\Psi = \Psi(r,\alpha,z)$, with

$$\Psi(r,\alpha,z) = \Psi^r(r,z)e_r + \Psi^\alpha(r,z)e_\alpha + \Psi^z(r,z)e_z, \tag{2.75}$$

defined in the domain $D(1,L)$. Under the assumption of axisymmetry we may fix momentarily $\alpha = 0$, which implies $e_r = (1,0,0)^\top$. The boundary conditions for ϕ translate to

$$\partial_r\phi\,|_{r=1} = e_r\cdot\xi^{(s)}\,|_{r=1} = R^{(s)}(1,z),$$

and those for Ψ, namely $\Psi\,|_{r=1}\times e_r = 0$, imply

$$\Psi_z(1,z) = \Psi_\alpha(1,z) = 0\,. \tag{2.76}$$

Moreover, the divergence free condition, $\nabla\cdot\Psi = \frac{1}{r}\Psi^r + \partial_r\Psi^r + \partial_z\Psi^z = 0$, is assumed to hold up to the boundary and delivers, therefore, a boundary condition on Ψ^r,

$$\Psi^r(1,z) + \partial_r\Psi^r(1,z) = -\partial_z\Psi^z(1,z) \overset{(2.76)}{=} 0\,. \tag{2.77}$$

Let us define the linear differential operators

$$\mathcal{L} \coloneqq \frac{1}{r}\partial_r\,(r\,.\,) = \frac{1}{r} + \partial_r\ , \tag{2.78}$$

$$\mathcal{L}_1 \coloneqq \mathcal{L}(\partial_r\,.\,) = \frac{1}{r}\partial_r\,(r\partial_r\,.\,) = \partial_r^2 + \frac{1}{r}\partial_r, \tag{2.79}$$

$$\mathcal{L}_2 \coloneqq \partial_r(\mathcal{L}\,.\,) = \partial_r\left(\tfrac{1}{r}\partial_r(r\,.\,)\right) = \partial_r^2 + \frac{1}{r}\partial_r - \frac{1}{r^2}\ , \tag{2.80}$$

$$\mathcal{G} \coloneqq \mathcal{L}_1 + \partial_z^2 = \partial_r^2 + \frac{1}{r}\partial_r + \partial_z^2 \ , \tag{2.81}$$

$$\mathcal{H} \coloneqq \mathcal{L}_2 + \partial_z^2 = \partial_r^2 + \frac{1}{r}\partial_r - \frac{1}{r^2} + \partial_z^2. \tag{2.82}$$

Subsequently, the Laplace operator in cylindrical coordinates $\Delta_c = \nabla_c \cdot \nabla_c = \frac{1}{r}\partial_r(r\partial_r) + \frac{1}{r^2}\partial_\alpha^2 + \partial_z^2$, applied to ϕ and the vector potential Ψ entails four partial differential equations

$$\begin{aligned} \mathcal{G}\,\phi(r,z) &= \left(\nabla \cdot \xi^{(s)}\right)(r,z) \ , \\ \partial_r\phi(0,z) &= 0, \\ \partial_r\phi(1,z) &= R^{(s)}(1,z), \end{aligned} \tag{2.83}$$

$$\begin{aligned} \mathcal{H}\,\Psi^r(r,z) &= (-\nabla \times \xi^{(s)})^r(r,z), \\ \Psi^r(0,z) &= 0, \\ \Psi^r(1,z) + \partial_r\Psi^r(1,z) &= 0, \end{aligned} \tag{2.84}$$

$$\begin{aligned} \mathcal{H}\,\Psi^\alpha(r,z) &= (-\nabla \times \xi^{(s)})^\alpha(r,z), \\ \Psi^\alpha(0,z) &= 0, \\ \Psi^\alpha(1,z) &= 0, \end{aligned} \tag{2.85}$$

$$\begin{aligned} \mathcal{G}\,\Psi^z(r,z) &= (-\nabla \times \xi^{(s)})^z(r,z), \\ \partial_r\Psi^z(0,z) &= 0, \\ \Psi^z(1,z) &= 0, \end{aligned} \tag{2.86}$$

for $\phi, \Psi^r, \Psi^\alpha, \Psi^z \in C_L^2([0,1]\times[0,L])$, where the subscript L in C_L^2 indicates periodicity in $[0,L]$ and the radial regularity at the boundary is understood to be a limit. The pole conditions stem from the continuity constraints, as the operators must stay bounded in the intervall $[0,1]$. After solving the above equations, one obtains the coefficients $A^{(s)}, R^{(s)}$ and $Z^{(s)}$ in cylindrical coordinates via eq. (2.72), i. e.

$$\begin{aligned} R^{(s)} &= \partial_r\phi - \partial_z\Psi^\alpha, \\ A^{(s)} &= \partial_z\Psi^r - \partial_r\Psi^z, \\ Z^{(s)} &= \partial_z\phi + \tfrac{1}{r}\partial_r(r\Psi^\alpha), \end{aligned} \tag{2.87}$$

where all functions only depend on $(r,z) \in [0,1]\times[0,L]$. In consulting the recursion relations eqs. (2.66) to (2.69), the overall algorithm demands the spatial derivatives of the time-Taylor coefficients $R^{(s)}, A^{(s)}$ and $Z^{(s)}$ in radial and vertical direction, i.e. second order derivatives on the potentials. While eqs. (2.83), (2.85) and (2.86) are pure Neumann or Dirichlet type boundary value problems, is eq. (2.84) a mixed boundary value problem. However, in consulting eps. (2.87), the component Ψ^r is only used for the construction of $A^{(s)}$ and derivatives thereof. In the following, we show that it is not necessary to solve eqs. (2.84) and (2.86) explicitly. Instead of solving for Ψ^r, Ψ^z we may directly solve for $A^{(s)}$ and its derivatives. As a matter of fact, we have

$$\partial_z A^{(s)} = -(\nabla \times \xi^{(s)})^r, \tag{2.88}$$

$$\mathcal{L}A^{(s)} = \frac{A^{(s)}}{r} + \partial_r A^{(s)} = (\nabla \times \xi^{(s)})^z, \tag{2.89}$$

stemming from the definition of the rotational (2.48), the RHSs being given via eqs. (2.66) and (2.68) from Section 2.4. Integrating the operator $\mathcal{L} = \frac{1}{r}\partial_r(r\,.\,)$ in eq. (2.89) directly yields

$$A^{(s)}(r,z) = \frac{1}{r}\int_0^r t\,(\nabla \times \xi^{(s)})^z(t,z)\,\mathrm{d}t, \tag{2.90}$$

$$\frac{A^{(s)}(r,z)}{r} = \frac{1}{r^2}\int_0^r t\,(\nabla \times \xi^{(s)})^z(t,z)\,\mathrm{d}t, \tag{2.91}$$

where we have used that $A^{(s)}(0,z) = 0$ for all $z \in [0,L]$ and $s \in \mathbb{N}_0$. Integrating eq. (2.88) would give $A^{(s)}$ up to a function depending only on r that can be obtained from eq. (2.89) in solving for the 0-th Fourier mode $\hat{A}_0^{(s)}$. However, the expression $A^{(s)}(r,z)/r$ is needed for the algorithm, but an explicit division by r should be avoided numerically. A radial differentiation allows to obtain the latter as $A^{(s)}/r = (\nabla \times \xi^{(s)})^z - \partial_r A^{(s)}$, where the derivative in r may also introduce a fairly large error when resolution is increased, see Figure 3.3 in Section 3.1.3 for the differentiation of shifted Chebyshev expansions. Section 3.2 treats model problems containing the operators $\mathcal{L}, \mathcal{L}_1$ and $\mathcal{L}_2$ from above and explains in detail how eqs. (2.90) and (2.91) are handled numerically in the CLA code.

2.6 The Complete CLA in a Cylinder

In Section 2.4 we have obtained recursion relations in cylindrical coordinates for the divergence and rotational of the time-Taylor coefficients of the displacement map $\xi(a,t) = X(a,t) - a = \sum_{s\geq 1}\left(R^{(s)}e_r + A^{(s)}e_\alpha + Z^{(s)}e_z\right)t^s$. Those time-Taylor coefficients are, for $s \geq 1$, precisely the ones for the trajectory map X itself and present the center of interest of this work. Via a Helmholtz-Hodge decomposition (HHD), stated in Section 2.5, one obtains the coefficients in solving some Poisson equations. The solution of those Poisson problems give rise to the representations

$$R^{(s)} = \partial_r\phi - \partial_z\Psi^\alpha, \quad A^{(s)} = \partial_z\Psi^r - \partial_r\Psi^z, \quad Z^{(s)} = \partial_z\phi + \tfrac{1}{r}\partial_r(r\Psi^\alpha), \qquad \text{for } s \geq 1, \tag{2.92}$$

and one adds $R^{(0)} = r$, $A^{(0)} = 0$ and $Z^{(0)} = z$. We also know that

$$R^{(1)} = v_0^r, \quad A^{(1)} = v_0^\alpha, \quad Z^{(1)} = v_0^z, \tag{2.93}$$

where v_0 is the initial velocity field. The latter relations may serve as a starting point for the recursions, but, in considering the following equations, knowing only the initial vorticity is also sufficient. The potentials ϕ and Ψ fulfill the Poisson eqs. (2.83) to (2.86), the RHSs being replaced by the formerly found recursion relations eqs. (2.66) to (2.69) and (2.71), which gives

$$\begin{aligned}\mathcal{G}\phi = \sum_{1\leq m<s}\Big(&\tfrac{A^{(m)}}{r}\left(\{A,Z\}^{(s-m)} - A_z^{(s-m)}\right)\\ &+ \tfrac{R^{(m)}}{r}\left(\{R,Z\}^{(s-m)} - R_z^{(s-m)} - Z_r^{(s-m)}\right) + \{R,Z\}^{(s)}\Big),\\ \partial_r\phi\,|_{r=1} = -\frac{1}{2}\Big(&\sum_{n=1}^{s-1} R^{(n)}R^{(s-n)} + A^{(n)}A^{(s-n)}\Big)\,|_{r=1},\end{aligned}$$

for the scalar potential, and, for the vector potential component-wise,

$$\mathcal{H}\Psi^r(r,z) = \delta_{1,s}\,\omega_0^r + \sum_{1\le m<s} \frac{m}{rs}\partial_z\Big[R^{(m)}A^{(s-m)} - A^{(m)}R^{(s-m)}\Big],$$

$$\begin{aligned}\mathcal{H}\Psi^\alpha(r,z) =& \delta_{1,s}\,\omega_0^\alpha + \\ &\sum_{1\le m<s} \frac{m}{s}\Big[\big\{R^{(m)},R^{(s-m)}\big\} + \big\{A^{(m)},A^{(s-m)}\big\} + \big\{Z^{(m)},Z^{(s-m)}\big\}\Big],\\ \mathcal{G}\Psi^z(r,z) =& \delta_{1,s}\,\omega_0^z + \sum_{1\le m<s} \frac{m}{rs}\partial_r\Big[A^{(m)}R^{(s-m)} - R^{(m)}A^{(s-m)}\Big].\end{aligned}$$

Moreover, the above equations get supplemented by boundary and pole conditions, specified in Section 2.5.
The appearing derivatives of the coefficients in the recursion sums act directly on the potentials. This fact could, more globally, be seen as an application of Caderon-Zygmund operators of order zero on the initial vorticity. Those operators are in principal just double derivatives on inverse Laplacians. Thus, the only requirement for this mechanism to work, is a Hölder continuous initial vorticity to ensure the differentiability of the solutions to the Poisson problems in the HHD.

In addition to $R^{(s)}, A^{(s)}, Z^{(s)}$, which are already given in eq. (2.92), the terms that are needed in principal by the CLA in a cylindrical domain are:

$$\begin{aligned}
\partial_r R^{(s)} &= \partial_r^2\phi - \partial_r\partial_z\Psi^\alpha, & \partial_z R^{(s)} &= \partial_z\partial_r\phi - \partial_z^2\Psi^\alpha,\\
\partial_r A^{(s)} &= \partial_r\partial_z\Psi^r - \partial_r^2\Psi^z, & \partial_z A^{(s)} &= \partial_z^2\Psi^r - \partial_z\partial_r\Psi^z,\\
\partial_r Z^{(s)} &= \partial_r\partial_z\phi + \partial_r^2\Psi^\alpha + \tfrac{1}{r}\partial_r\Psi^\alpha - \tfrac{1}{r^2}\Psi^\alpha, & \partial_z Z^{(s)} &= \partial_z^2\phi + \partial_z\partial_r\Psi^\alpha + \tfrac{1}{r}\partial_z\Psi^\alpha,\\
A_1^{(s)} &:= \frac{A^{(s)}}{r} = \frac{\partial_z\Psi^r}{r} - \frac{\partial_r\Psi^z}{r}, & R_1^{(s)} &:= \frac{R^{(s)}}{r} = \frac{\partial_r\phi}{r} - \frac{\partial_z\Psi^\alpha}{r}.
\end{aligned} \tag{2.94}$$

In the here presented CLA implementation, the terms $R_1^{(s)}$ and $A_1^{(s)}$ result from a solving technique, which does not imply any division by r in the program itself. The divided derivatives of the potentials, which give $R_1^{(s)}$ and $A_1^{(s)}$, are directly obtained from the source terms of the appearing Poisson problems. Subsequently, the quantities $R_1^{(s)}$ and $A_1^{(s)}$ are used in the computation of $\mathcal{A}\,(X^r, X^\alpha)$ from Section 2.2. For positive X^r, the computation of $\mathcal{A}\,(X^r, X^\alpha)$ comprises the division X^α/X^r, which can lead to cancellation errors for small X^r. To avoid this problem, the terms $R_1^{(s)}$ and $A_1^{(s)}$ are used instead and allow the computation of $\mathcal{A}$ via

$$\mathcal{A}\,(X^r, X^\alpha) = \mathcal{A}\,(X_1^r, X_1^\alpha)\,, \tag{2.95}$$

where

$$X_1^r = \sum_{s\ge 0} R_1^{(s)}(\triangle t)^s, \quad X_1^\alpha = \sum_{s\ge 0} A_1^{(s)}(\triangle t)^s, \tag{2.96}$$

for a time-step $\triangle t$. Moreover, all the terms $X^r/r, X^\alpha/r$ in the vorticity defining equations (eqs. (2.38) to (2.40)) are replaced by X_1^r, X_1^α respectively, as already remarked in the end of Section 2.2. The same replacements take place in the recursion sums eqs. (2.66) to (2.69), such that the divisions by r are completely avoided in our implementation of the CLA. More details on how to solve directly for the corresponding terms is stated in Sections 3.2 and 3.3.

Furthermore, not all the terms involving the Helmholtz-Hodge potentials in eqs. (2.94)

need to be computed. We may reduce the number of necessary terms by expressing some derivatives of the time-Taylor coefficients of $X^{(s)}$ by algebraically manipulating the definitions of $\nabla \times \xi^{(s)}$ and $\nabla \cdot \xi^{(s)}$ (see eqs. (2.48) and (2.49)). We recall that there holds $X^{(s)} = \xi^{(s)}$ for $s > 0$. Provided we know $\partial_r R^{(s)}, \partial_z R^{(s)}$ and $R_1^{(s)}$, we have

$$\partial_r Z^{(s)} = \partial_z R^{(s)} + (-\nabla \times \xi^{(s)})^\alpha \tag{2.97}$$

$$\partial_z Z^{(s)} = -R_1^{(s)} - \partial_r R^{(s)} + \nabla \cdot \xi^{(s)}, \tag{2.98}$$

where $(-\nabla \times \xi^{(s)})^\alpha$ and $\nabla \cdot \xi^{(s)}$ are the RHSs of eqs. (2.83) and (2.85) respectively and are computed via the recursion relations eqs. (2.67) and (2.69). The computation of all the needed derivatives of $A^{(s)}$, as well as the computation of $A_1^{(s)}$, has been discussed without the use of the HHD in the end of the preceding section. Therefore, we only need to compute those expressions of the Helmholtz-Hodge potentials that appear in the following,

$$\partial_r R^{(s)} = \partial_r^2 \phi - \partial_r \partial_z \Psi^\alpha, \tag{2.99}$$

$$\partial_z R^{(s)} = \partial_z \partial_r \phi - \partial_z^2 \Psi^\alpha, \tag{2.100}$$

$$R_1^{(s)} = \frac{\partial_r \phi}{r} - \frac{\partial_z \Psi^\alpha}{r}, \tag{2.101}$$

$$Z^{(s)} = \partial_z \phi + \partial_r \Psi^\alpha + \tfrac{1}{r} \Psi^\alpha \,. \tag{2.102}$$

The coefficients $R^{(s)}, A^{(s)}$ and $Z^{(s)}$ are not needed for the recurrence mechanism itself, but for the actual computation of the particle trajectories. The coefficients $R^{(s)}$ and $A^{(s)}$ are simply obtained by multiplying $R_1^{(s)}$ and $A_1^{(s)}$ by r respectively. Note that all the components of $-\nabla \times \xi^{(s)}$ need to be computed via the recursion mechanism to construct the required terms, only if the initial angular velocity component is non-zero. In the special case, where the initial angular velocity field is identically zero, all the terms that correspond to $A^{(s)}$ vanish, as well as the components $(-\nabla \times \xi^{(s)})^r$ and $(-\nabla \times \xi^{(s)})^z$, which holds true for all later times. Thus, in this spacial case, those terms need not be computed, which further reduces the numerical complexity of the program. See Section 6.6 for further information and numerical tests on those special (swirl-free) flows.

The recurrence is executed until a predefined maximal order S is reached. Then, all the needed truncated time-Taylor series are ready to be summed and entail the approximated new positions of the particles $X(a, t)$ in cylindrical coordinates after the inserted time. Subsequently, the velocity and vorticity fields are calculated via the characteristic equation (2.23) and vorticity transport formula (2.34) respectively. Since the flow fields are then only known at the endpoints of the Lagrangian particle trajectories, a scattered interpolation back onto the regular Eulerian grid is in place. In the case of our axisymmetric flow, this three-dimensional interpolation reduces to two dimensions as explained in Chapter 4. In the present implementation, both the velocity and vorticity fields are interpolated, they serve as the initial condition for the next iteration. As mentioned before, the knowledge of the new vorticity field would be enough to restart the recursions, but setting the values in eq. (2.93) directly via the interpolated velocity field allows to skip the HHD for the first set of coefficients. Also, the velocity field is most notably needed to track the kinetic energy and helicity conservation of the flow (among others, see Section 6.2).

Discussion. Although, theoretically, the terms in eq. (2.94) are well defined and rather simple in their nature, numerically they do introduce certain complications. The major problem, in applying any straightforward numerical method for the Poisson equations, is the introduction of relatively large errors in the recursion sums due to the second order derivatives on

the potentials, which are inverse Laplacians. These errors accumulate rapidly and lead to an error cascade in the RHSs of the Poisson problems, which are needed for the computation of the next coefficients. Thus, depending on the numerical scheme in use, one might be only able to choose a very limited time-step, such that the apparent errors in the coefficients do not influent the result of the final time-Taylor sum. This in turn limits the convergence rate in time of the overall algorithm. The effect of polluted, or unstable, derivatives is predominantly apparent in spectral approximations on high resolutions as discussed in Section 3.1.3. Therefore, to provide a close to theoretical order of temporal convergence of the algorithm, it is crucial to ensure that the numerical schemes in place do not introduce large errors that are several orders of magnitude worse than those appearing in the actual Poisson solutions. The methods in use in the CLA are presented in Section 3.3, they solve directly for the terms that appear in the RHSs of eq. (2.94). Therefore, all the expressions in the LHSs of eq. (2.94) are saved in the program to sum their respective truncated time-Taylor series.

Chapter 3

Numerical Methods

3.1 Discretization

3.1.1 Spectral Approximation

A crucial part of the investigated algorithm is the accurate computation of the Calderon-Zygmund operators of order zero, namely the double derivatives of an inverse Laplacian, applied to the divergence and rotation of the trajectory's time-Taylor coefficients. The appearing errors from the second derivatives on top of the errors in the solutions themselves may recursively cascade into higher orders s. This amplification of errors can lead to a lower order scheme and a very limited time-step. Thus, the better the above-mentioned operators are computed, the better the time-step can be chosen and the more accurate the overall algorithm becomes with fewer time-steps. At this point, we would like to remind the reader that the Cauchy-Lagrange algorithm is completely independent of the underlying mesh spacing and, thus, not bound to any CFL condition. It is desirable to preserve the possible range of a time-step as much as possible by maximally reducing the accumulation of numerical errors.

We refrain from finite difference methods for the Calderon-Zygmund operators, because of their slow convergence and limited accuracy in the derivatives. Instead, a pseudo-spectral approach was chosen and a representation of the trajectory's coefficients by shifted Chebyshev series in the radial dimension and Fourier series in the vertical dimension is applied,

$$X^{(s)}(r,z) = \sum_{|k|=0}^{\infty} \sum_{l=0}^{\infty} X_{kl}^{(s)} T_l(2r-1)\, \mathrm{e}^{\mathrm{i}2\pi kz/L} \ . \tag{3.1}$$

A truncation to $M+1, N+1 \in \mathbb{N}$ modes yields a spectrally discrete approximation of the latter, i.e.

$$X^{(s)}(r,z) \cong (I_{NM} X^{(s)})(r,z) = \sum_{|k|=0}^{M/2} \sum_{l=0}^{N} X_{kl}^{(s)} T_l(2r-1)\, \mathrm{e}^{\mathrm{i}2\pi kz/L} \ . \tag{3.2}$$

See Section 3.1.2 for a detailed description of the shifted Chebyshev polynomials and series.

The Chebyshev-Fourier basis has been chosen due to the underlying numerically fast discrete Fourier transforms, and also to initialize this novel method for wall-bounded incompressible Euler flow in using widely known and well studied spectral representations. The applied methods allow for a fast numerical convergence and, thus, for an accurate representation of the solutions to the HHD and the flow itself in low to moderate resolutions. Although the choice for a spectral approach introduces difficulties for higher resolutions, as

spectral derivatives become progressively more erroneous and unstable, we have found ways to overcome those pitfalls and find that the overall algorithm is stable, accurate, and fast. Furthermore, the shifted Chebyshev series are obtained from function values known on the shifted Chebyshev extremas, namely $((\cos(i\pi/N)+1)/2)_{i=0,\dots,N}$, which are clustered near (and contain) the boundary of the cylindrical domain as well as the pole $r=0$. This clustering is important especially in view of possibly appearing finite-time singularity, as the numerical studies in [83] indicate a singularity directly at the boundary.

3.1.2 Chebyshev Polynomials and Series Representation

Introductions to Chebyshev polynomials can be found plentiful in the literature (see e.g. [85, 17, 55]). Therefore, we will only recall from the given references the definitions and properties that are important for our algorithm.

Definition. The Chebyshev polynomial of the first kind of order n is defined as

$$T_n(x) = \cos(n \arccos(x)), \quad x \in I,\ n \in \mathbb{N}_0, \tag{3.3}$$

where $I := [-1,1]$ and $\mathbb{N}_0 := \mathbb{N} \cup \{0\} = \{0,1,2,\dots\}$.

In setting $x = \cos(\theta)$, $\theta \in [0,\pi]$, we find

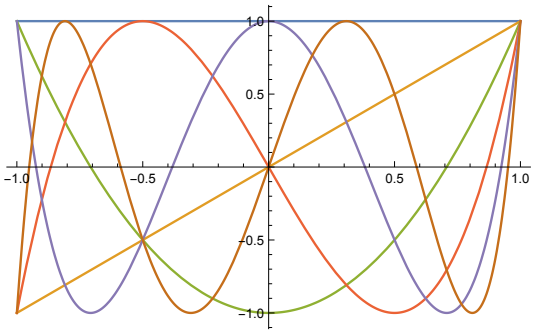
$$T_n(\cos(\theta)) = \cos(n\theta), \tag{3.4}$$

from which all the properties in the following may be derived. The Chebyshev polynomials satisfy the three term recurrence relation

$$\begin{aligned} T_0(x) &= 1 \\ T_{n+1}(x) &= 2xT_n(x) - T_{|n-1|}(x), \quad \forall n \in \mathbb{N}_0, \end{aligned} \tag{3.5}$$

and the first orders in polynomial form are

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \\ T_4(x) &= 8x^4 - 8x^2 + 1 \\ T_5(x) &= 16x^5 - 20x^3 + 5x \\ T_6(x) &= 32x^6 - 48x^4 + 18x^2 - 1 \\ &\dots \end{aligned}$$



(order indicated by number of zeros)

The Chebyshev extrema, that is the local extrema of $T_n(x)$ in the interval I, are given by

$$x_i = \cos\left(\frac{i\pi}{n}\right), \quad i \in \{0,\dots n\}, \tag{3.6}$$

such that $T_n(x_i) = (-1)^i$, the zeros are given by

$$\tilde{x}_i = \cos\left(\frac{2i+1}{2n}\pi\right), \quad i \in \{0,\dots,n-1\}\,. \tag{3.7}$$

Without difficulty, one verifies the nesting property of the Chebyshev extrema, that is

$$\left\{\cos\left(\frac{i\pi}{n}\right)\right\}_{0\le i\le n} \subset \left\{\cos\left(\frac{i\pi}{2n}\right)\right\}_{0\le i\le 2n}, \quad \forall n \in \mathbb{N}, \tag{3.8}$$

or, in other words, the extrema of T_n are also extrema of T_{2n}. Furthermore, there holds the orthogonality relation

$$\int_{-1}^{1} T_l(t)T_k(t)\omega(t)\,\mathrm{d}t = \int_{-1}^{1} \frac{T_l(t)T_k(t)}{\sqrt{1-t^2}}\,\mathrm{d}t = c_l\frac{\pi}{2}\delta_{lk}, \tag{3.9}$$

where

$$c_l \coloneqq \delta_{l0} + 1 = \begin{cases} 2 & ,\text{if } l = 0 \\ 1 & ,\text{if } l > 0\,. \end{cases} \tag{3.10}$$

A discrete version of the latter relation, based on the Chebyshev extrema $(x_i)_{0\le i\le N\in\mathbb{N}}$ from above, reads

$$\sum_{i=0}^{N} \frac{1}{c_i c_{N-i}} T_l(x_i)T_k(x_i) = \frac{c_l c_{N-l}}{2} N\delta_{lk}, \tag{3.11}$$

for $l, k = 0, \ldots, N$. In the literature the above summation $\sum_{i=0}^{N} {}^{1}\!/_{c_i c_{N-i}}$, indicating the halving of the first and last terms, is often denoted by $\sum''$ and $\sum_{i=0}^{N} {}^{1}\!/_{c_i}$ by $\sum'$. Here, we intend to avoid primed sums, the appearing primes in this text represent derivatives with respect to the sole argument of a function.
A multiplication of two Chebyshev polynomials entails

$$2T_nT_m = T_{n+m} + T_{|n-m|}, \quad \forall n, m \in \mathbb{N}_0, \tag{3.12}$$

with the special case

$$xT_n = \frac{1}{2}(T_{n+1}(x) + T_{|n-1|}(x)). \tag{3.13}$$

The derivatives of the Chebyshev polynomials fulfill the relations

$$\begin{aligned} T_0 &= T_1', \quad 4T_1 = T_2' \\ 2T_n &= \left(\frac{{T_{n+1}}'}{n+1} - \frac{{T_{n-1}}'}{n-1}\right), \quad n \ge 2, \end{aligned} \tag{3.14}$$

as well as

$$T_n'' = \sum_{\substack{k=1 \\ k+n \text{ even}}}^{n-2} \frac{1}{c_k} n(n^2-k^2)T_k\,. \tag{3.15}$$

The *shifted Chebyshev polynomial* of the first kind of order $n \in \mathbb{N}_0$ is obtained from the ordinary Chebyshev polynomial by a change of the variable, namely $t = 2r-1$ with $r \in [0,1]$, and is denoted by

$$T_n^*(r) \coloneqq T_n(2r-1)), \quad r \in [0,1]\,. \tag{3.16}$$

The shifted Chebyshev polynomials satisfy the three term recurrence relation

$$\begin{aligned} T_0^*(r) &= 1, \\ T_n^*(r) &= 2(2r-1)T_{n-1}^* - T_{|n-2|}^*, \quad n \in \mathbb{N}, \end{aligned} \tag{3.17}$$

which implies

$$rT_n^* = \frac{1}{4}\left(T_{|n-1|}^* + 2T_n^* + T_{n+1}^*\right), \quad \forall n \in \mathbb{N}_0, \tag{3.18}$$

and the first orders read

$$\begin{aligned}
T_0^*(r) &= 1 \\
T_1^*(r) &= 2r - 1 \\
T_2^*(r) &= 8r^2 - 8r + 1 \\
T_3^*(r) &= 32r^3 - 48r^2 + 18r - 1 \\
T_4^*(r) &= 128r^4 - 256r^3 + 160r^2 - 32r + 1 \\
T_5^*(r) &= 512r^5 - 1280r^4 + 1120r^3 - 400r^2 + 50r - 1 \\
T_6^*(r) &= 2048r^6 - 6144r^5 + 6912r^4 - 3584r^3 + 840r^2 - 72r + 1 \\
&\dots
\end{aligned}$$

At the interval end points, they take the values

$$T_n^*(0) = T_n(-1) = (-1)^n, \tag{3.19}$$

$$T_n^*(1) = T_n(1) = 1, \tag{3.20}$$

and they do not possess any even or odd symmetry about the origin $r = 0$. An explicit representation of the shifted Chebyshev polynomial of order $n > 0$ is given by the polynomial sum

$$T_n^*(r) = \sum_{j=0}^{n} n \frac{4^j (-1)^{n-j} (j+n-1)!}{(2j)!(n-j)!} r^j \,. \tag{3.21}$$

From eq. (3.14), in using eq. (3.16), we directly obtain the derivative relations

$$2T_0^* = {T_1^*}', \quad 8T_1^* = {T_2^*}', \quad 4T_n^* = \left(\frac{{T_{n+1}^*}'}{n+1} - \frac{{T_{n-1}^*}'}{n-1}\right), \quad n \geq 2, \tag{3.22}$$

and with it the integral relations

$$\begin{aligned}
\int T_0^*(r)\mathrm{d}r &= \frac{T_1^*}{2}, \quad \int T_1^*(r)\mathrm{d}r = \frac{T_2^*}{8}, \\
\int T_n^*(r)\mathrm{d}r &= \frac{1}{4}\left(\frac{T_{n+1}^*}{n+1} - \frac{T_{n-1}^*}{n-1}\right), \quad n \geq 2,
\end{aligned} \tag{3.23}$$

which are valid up to an additive constant of integration. The next integral relation follows readily from eqs. (3.18) and (3.23),

$$\begin{aligned}
\tau(n,r) &:= \int rT_n^*(r)\mathrm{d}r = \int \frac{1}{4}\left(T_{|n-1|}^* + 2T_n^* + T_{n+1}^*\right)(r)\mathrm{d}r \\
&= \begin{cases}
\frac{1}{4}T_1^*(r) + \frac{1}{16}T_2^*(r) & , n = 0 \\
\frac{1}{16}T_1^*(r) + \frac{1}{16}T_2^*(r) + \frac{1}{48}T_3^*(r) & , n = 1 \\
-\frac{1}{8}T_1^*(r) + \frac{1}{24}T_3^*(r) + \frac{1}{64}T_4^*(r) & , n = 2 \\
\frac{1}{16}\left(-\frac{T_{n-2}^*(r)}{n-2} + \frac{2T_{n+1}^*(r)}{n+1} - \frac{2T_{n-1}^*(r)}{n-1} + \frac{T_{n+2}^*(r)}{n+2}\right) & , n \geq 3,
\end{cases}
\end{aligned} \tag{3.24}$$

where the constant of integration has again been omitted. In the following we assume to be given function values $u_i^* = u(x_i^*)$, $i = 0, \dots, N$, of a continuous function $u : [0,1] \to \mathbb{R}$ on the

shifted Chebyshev extrema

$$x_i^* := \frac{x_i + 1}{2} \in [0, 1], \quad x_i \text{ as in eq. (3.6)}. \tag{3.25}$$

Note that the shifted Chebyshev extrema fulfill the same nesting property as the non-shifted Chebyshev extrema (i.e. relation (3.8)). An approximation u^N of u is given by the shifted Chebyshev series

$$u^N(r) = \sum_{l=0}^{N} u_l T_l^*(r), \tag{3.26}$$

such that $u^N(x_i^*) = u_i^*$. To find the coefficients u_l of the above series, we write

$$u_i^* = u^N(x_i^*) = \sum_{l=0}^{N} u_l T_l^*(x_i^*) = \sum_{l=0}^{N} u_l T_l(2x_i^* - 1) = \sum_{l=0}^{N} u_l T_l(x_i)\,. \tag{3.27}$$

Multiply eq. (3.27) by $T_k(x_i), k = 0, \ldots, N$, and sum over i in halving the first and last term to find

$$\begin{aligned} \sum_{i=0}^{N} \frac{1}{c_i c_{N-i}} u_i^* T_k(x_i) &= \sum_{l=0}^{N} u_l \sum_{i=0}^{N} \frac{c_l c_{N-l}}{2} T_l(x_i) T_k(x_i) \\ &= \sum_{l=0}^{N} u_l \frac{c_l c_{N-l}}{2} N \delta_{lk}, \end{aligned} \tag{3.28}$$

thus,

$$\sum_{i=0}^{N} \frac{1}{c_i c_{N-i}} u_i^* \cos\left(\frac{ik\pi}{N}\right) = u_k \frac{c_k c_{N-k}}{2} N, \tag{3.29}$$

where we have applied the discrete orthogonality relation (3.11) as well as the definitions (3.3) and (3.6). Consequently, we have derived

$$u_k = \frac{1}{c_k c_{N-k}} \frac{1}{N} \left(u_0^* + (-1)^k u_N^* + 2 \sum_{i=1}^{N-1} u_i^* \cos\left(\frac{ik\pi}{N}\right) \right), \tag{3.30}$$

for $k = 0, \ldots, N$. The bracketed term above is commonly known as the type-I discrete cosine transform, or DCT-I, of the vector $(u_i^*)_{0 \le i \le N}$. It is implemented in numerous open source software[1]. The back-transform, that is the conversion from the series coefficients u_l to function values u_i^*, can also be computed via the DCT-I, which can easily be seen by inserting the shifted Chebyshev extrema into eq. (3.26), giving

$$u_i^* = \sum_{l=0}^{N} u_l \cos\left(\frac{il\pi}{N}\right). \tag{3.31}$$

Thus, we have to perform a DCT-I transform of the vector

$$\left(\frac{c_l c_{N-l}}{2} u_l\right)_{l=0,\ldots,N} = \left(u_0, \frac{u_1}{2}, \ldots, \frac{u_{N-1}}{2}, u_N\right). \tag{3.32}$$

In using a similar approach, one shows that the transform from function values based on the Chebyshev zeros to the Chebyshev coefficients u_l is defined by a DCT-II, whose back-

[1] We use FFTW 3.3.9, see http://www.fftw.org/fftw3_doc/What-FFTW-Really-Computes.html for a definition of their implemented transforms.

transform is a DCT-III. However, for our numerical needs, namely the inclusion of interval end-points (boundary points of the cylinder) and the nesting property (3.8) that is not fulfilled by the Chebyshev zeros, we use the shifted Chebyshev extrema throughout this work. However, the zeros may be used for integration purposes in form of the Gauss-integration scheme, which is reviewed here briefly as well. See [21, 87] for the following statements and proofs thereof.

Gauss integration for Chebyshev Polynomials. Let $w_0, \dots, w_{n-1}$ be the solution of the linear system

$$\sum_{j=0}^{n-1} (\tilde{x}_j)^k w_j = \int_{-1}^{1} x^k \omega(x)\,\mathrm{d}x, \quad 0 \le k \le n-1,$$

where $\tilde{x}_0 < \tilde{x}_1 < \ldots < \tilde{x}_{n-1}$ are the roots of the n-th Chebyshev polynomial from eq. (3.7). There holds

$$w_j = \frac{\pi}{n}, \qquad \text{for } j = 0, \dots, n-1,$$

and

$$\sum_{j=0}^{n-1} p(\tilde{x}_j) w_j = \int_{-1}^{1} p(x) w(x)\,\mathrm{d}x, \quad \text{for all } p \in \mathbb{P}_{2n-1}. \tag{3.33}$$

3.1.3 Numerical Differentiation, Integration and Division

Even though the present implementation of the Cauchy-Lagrange algorithm avoids direct numerical differentiation and division, we recall standard schemes for the latter to compare them to the methods that solve directly for the needed terms, which are presented later in this chapter. The implementation of the Cauchy-Lagrange algorithm needs derivatives of first and second order, therefore, we will restrict ourselves to differentiation and integration up to second order, but the schemes may easily be extended to general orders. In the following, let $M, N \in \mathbb{N}, L > 0$ and $j \in \{-1, 0, 1, 2\}$. We define the generic functions

$$f^{(j)}(z) = \sum_{k=0}^{M} \hat{f}_k^{(j)}\, \mathrm{e}^{\mathrm{i}\tilde{k}z}, \qquad z \in [0, L],\ \tilde{k} = \frac{2\pi k}{L},$$
$$g^{(j)}(r) = \sum_{l=0}^{N-j} g_l^{(j)} T_l^*(r), \quad r \in [0, 1],$$

where $j > 0$ indicates derivatives and $j < 0$ primitives of $f^{(0)}$ or $g^{(0)}$. For consistency, we assume that the modes of f fulfill $\hat{f}_0^{(j)} = 0$, if $j > -1$, and $\hat{f}_0^{(-1)}$ represents a constant of integration. We seek transforms and relations between the coefficients of different orders so that we may differentiate and integrate within the spectral spaces. The recalled methods are applied to test functions, namely

$$s(r) \coloneqq \mathrm{e}^{1-\cos(2\pi r)} - 1, \tag{3.34}$$
$$t(r, z) \coloneqq s(r)\, \mathrm{e}^{\sin(2\pi z/L)}, \tag{3.35}$$

for $(r, z) \in [0, 1] \times [0, L]$. They will also be used to show the numerical convergence and stability of the solving methods for the Poisson problems, which are presented in the next sections. The functions s and t fulfill homogeneous Dirichlet and homogeneous Neumann boundary conditions simultaneously in the radial dimension at $r = 0, 1$. In the program, the test function and related expressions, such as derivatives etc., are evaluated in quadruple precision and rounded to double precision to ensure accurate comparisons. Besides, the

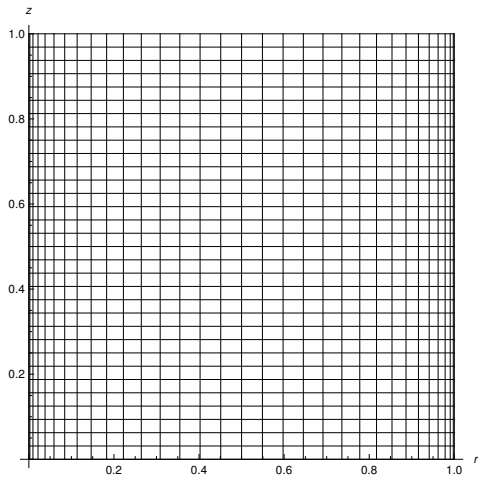


Figure 3.1: The Chebyshev-Fourier grid of size 33×32 for $L = 1$. Every intersection of two lines is a grid point. Cells at the boundary and pole are strongly elongated.

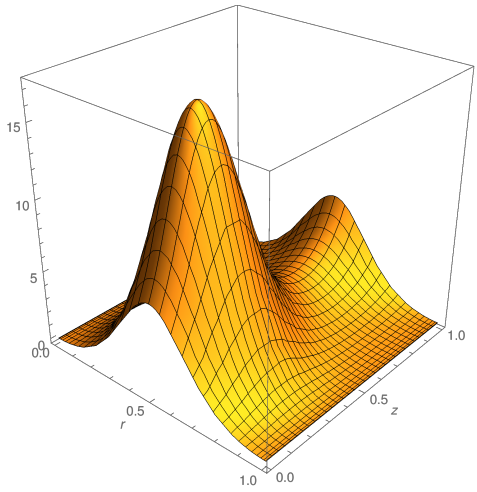


Figure 3.2: The test function t defined in eq. (3.35) for $L = 1$. The Chebyshev-Fourier grid from the left panel is visualized, while the vertical axis is scaled to unit length.

implemented differentiation and solving algorithms have been thoroughly tested with various other test functions, the above just serve as representatives.

Differentiation. The spectral discretization allows to easily obtain the first and second derivatives of an approximated function by simple coefficient transforms. Fix $j > 0$ for the moment, then the Fourier series $f^{(0)}$ is directly differentiated j times to give

$$f^{(j)}(z) = \sum_{k=0}^{M} \hat{f}_k^{(j)} \, \mathrm{e}^{\mathrm{i}\tilde{k}z}, \qquad \text{with } \hat{f}_k^{(j)} = \hat{f}_k^{(0)} (\mathrm{i}\tilde{k})^j \ , \ k = 0, \dots, M \, . \tag{3.36}$$

In the Chebyshev case, the derivative relations for the shifted Chebyshev functions (3.22) imply relations for the coefficients of the first derivative of a shifted Chebyshev series $g^{(0)}$, they read

$$4l g_l^{(0)} = c_{l-1} g_{l-1}^{(1)} - g_{l+1}^{(1)}, \quad l \geq 1, \tag{3.37}$$

from which we get

$$c_{l-1} g_{l-1}^{(1)} = g_{l+1}^{(1)} + 4l g_l^{(0)}, \qquad l = N, N-1, \dots, 1, \tag{3.38}$$

with $g_{N+1}^{(1)} = g_N^{(1)} = 0$ and c_l as in (3.10). To see this, one may argue as for the non-shifted Chebyshev series in [57, p. 71]. First, we write the shifted Chebyshev sum that corresponds to the first derivative of g and then replace the shifted Chebyshev polynomials by the derivative

relations (3.22). We write

$$\begin{aligned} g^{(1)}(r) &= g_0^{(1)} T_0^*(r) + g_1^{(1)} T_1^*(r) + g_2^{(1)} T_2^*(r) + \ldots + \underbrace{g_N^{(1)}}_{=0} T_N^*(r) \\ &= \frac{g_0^{(1)}}{2} T_1^{*\,\prime}(r) + \frac{g_1^{(1)}}{8} T_2^{*\,\prime}(r) + \frac{g_2^{(1)}}{4} \left(\frac{T_3^{*\,\prime}(r)}{3} - \frac{T_1^{*\,\prime}(r)}{1} \right) \\ &\qquad + \ldots + \frac{g_{N-1}^{(1)}}{4} \left(\frac{T_N^{*\,\prime}(r)}{N} - \frac{T_{N-2}^{*\,\prime}(r)}{N-2} \right). \end{aligned} \tag{3.39}$$

An integration of the above sum delivers

$$\begin{aligned} g^{(0)}(r) - g_0^{(0)} &= \left(\frac{g_0^{(1)}}{2} - \frac{g_2^{(1)}}{4} \right) T_1^*(r) + \left(\frac{g_1^{(1)}}{4 \cdot 2} - \frac{g_3^{(1)}}{4 \cdot 2} \right) T_2^*(r) \\ &\quad + \ldots + \left(\frac{g_{N-2}^{(1)}}{4(N-1)} - \frac{g_N^{(1)}}{4(N-1)} \right) T_{N-1}^*(r) + \frac{g_{N-1}^{(1)}}{4N} T_N^*(r), \end{aligned} \tag{3.40}$$

which implies (3.37). The second derivative coefficients are simply obtained via a reapplication of the above recurrence

$$c_{l-1} g_{l-1}^{(2)} = g_{l+1}^{(2)} + 4 l g_l^{(1)}, \quad l = N-1, N-2, \ldots, 1, \tag{3.41}$$

with $g_N^{(2)} = g_{N-1}^{(2)} = 0$. It is both sufficient and efficient to directly implement the recurrence formalism (3.37) and use it twice in order to obtain the coefficients of the second derivatives. For comparison we recall the 6^{th} order finite difference schemes for the first and second derivative (see [46] for instance). For our needs it is sufficient to assume in the following that $\bar{g} = \bar{g}(r)$ is a periodic function on $\mathbb{R}$ with period $[0, 1]$. In setting $r_i = \frac{i}{N}, i = 0, 1, \ldots, N$, $N \in \mathbb{N}$, as well as $\bar{g}_i \coloneqq \bar{g}(r_i)$ with $\bar{g}_{-j} = \bar{g}_{N-j}$ and $\bar{g}_{N+j} = \bar{g}_j$ for $j = 1, 2, 3$ due to periodicity, we have

$$\begin{aligned} \bar{g}'(r_i) &= \frac{-\bar{g}_{i-3} + 9\bar{g}_{i-2} - 45\bar{g}_{i-1} + 45\bar{g}_{i+1} - 9\bar{g}_{i+2} + \bar{g}_{i+3}}{60 N^{-1}} \\ &\qquad + O\left(-\frac{1}{140} \left(\frac{1}{N} \right)^6 \bar{g}^{(7)}(r_i) \right), \end{aligned} \tag{3.42}$$

for the first derivative, and

$$\begin{aligned} \bar{g}''(r_i) &= \frac{2\bar{g}_{i-3} - 27\bar{g}_{i-2} + 270\bar{g}_{i-1} - 490\bar{g}_i + 270\bar{g}_{i+1} - 27\bar{g}_{i+2} + 2\bar{g}_{i+3}}{180 N^{-2}} \\ &\qquad + O\left(-\frac{1}{560} \left(\frac{1}{N} \right)^7 \bar{g}^{(8)}(r_i) \right), \end{aligned} \tag{3.43}$$

for the second derivative. We are going to reference this method as fd6_h, the _h emphasizing the uniformity of the underlying grid with constant grid spacing h. As a matter of fact, this method may be applied to a function that is known on a shifted Chebyshev grid as well. In recalling the form of the shifted Chebyshev extrema $x_i = (\cos(\pi i/N) + 1)/2$, it becomes immediately clear that a function $\tilde{g}((\cos(\pi i/N) + 1)/2) = \tilde{g}((\cos(\pi r_i) + 1)/2)$ is actually a periodic function on the uniform grid $\{r_i = {}^{i}/_{N}\}_{i=0,1,\ldots,N} \subset [0, 1]$. In applying eq. (3.42), by

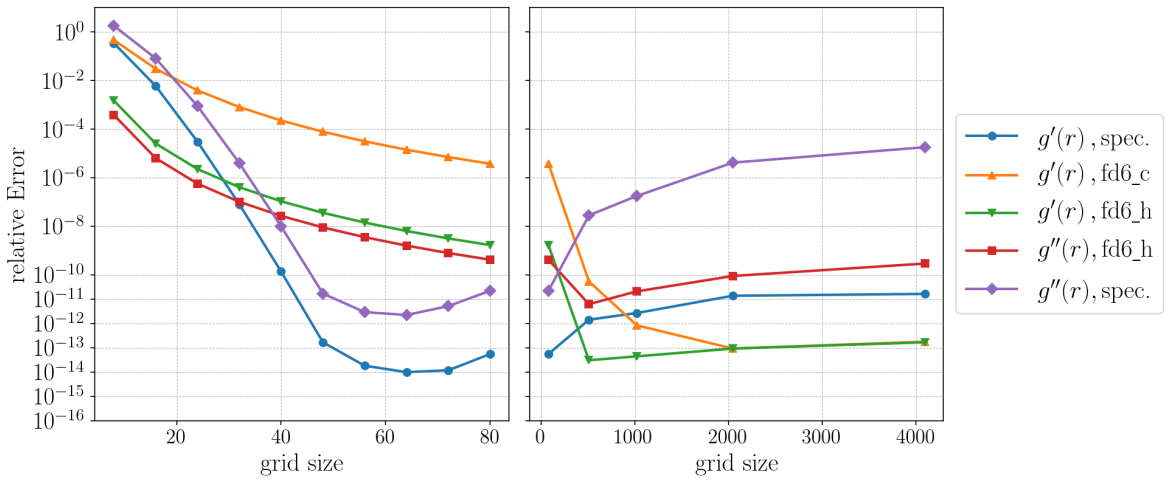


Figure 3.3: Numerical verification of the implemented differentiation methods for $s = s(r)$ defined in eq. (3.34). The left panel shows the (exponential) convergence of the spectral differentiation methods (spec.) versus the finite difference methods of 6-th order on the Chebyshev grid (fd6_c) and a uniform grid (fd6_h) of equal size. The right panel shows the instability of the spectral derivatives with increasing mesh size.

taking care of the inner derivative, one finds for $i = 1, \dots, N-1$,

$$\tilde{g}'(x_i) \cong -\frac{2}{\pi \sin(\pi i/N)} \frac{-\tilde{g}_{i-3} + 9\tilde{g}_{i-2} - 45\tilde{g}_{i-1} + 45\tilde{g}_{i+1} - 9\tilde{g}_{i+2} + \tilde{g}_{i+3}}{60N^{-1}}. \tag{3.44}$$

The inconvenience of the division by zero when $i = 0, N$ can be avoided by taking, instead of the extrema, the shifted Chebyshev zeros $(\tilde{x}_i+1)/2 = (\cos\left(\frac{2i+1}{2N}\pi\right)+1)/2 \quad , i \in \{0, \dots, N-1\}$, which verify $\tilde{x}_i \neq 0, \forall i$. A function $\tilde{g}$ that is given in the shifted Chebyshev spectral space, i.e. by shifted Chebyshev series coefficients, may be readily evaluated on the points $(\tilde{x}_i+1)/2$ via a DCT-3, whose inverse is a DCT-2. Subsequently, the equivalent to eq. (3.44) becomes, for $i = 0, \dots, N-1$,

$$\tilde{g}'(\tilde{x}_i) \cong -\frac{2}{\pi \sin(\frac{2i+1}{2N}\pi)} \frac{-\tilde{g}_{i-3} + 9\tilde{g}_{i-2} - 45\tilde{g}_{i-1} + 45\tilde{g}_{i+1} - 9\tilde{g}_{i+2} + \tilde{g}_{i+3}}{60N^{-1}}. \tag{3.45}$$

Because this differentiation scheme is performed on a shifted Chebyshev grid, we will refer to it by fd6_c. A numerical verification of the convergence of the above schemes is given in Figure 3.3.

Differentiation instabilities. While the spectral differentiation methods from above produce exponential convergence in linear complexity, they lack stability towards higher grid sizes. The reason for this strong error build-up lies in the numerically performed Fourier transforms (FT), where a discrete cosine transform is just a special FT. Instead of returning all modes accurately, the routines return only those which are larger than machine epsilon (10^{-16} for double precision) and the remaining plateau around machine epsilon. Subsequent multiplications of the wavenumbers, squared for the second derivative, amplify the rounding errors and lead to inaccurate results. The appearing inaccuracies are so large that they nearly prohibit a straight forward use of the presented differentiation methods. The right panel in Figure 3.3 shows this error build-up, not only for the spectral Chebyshev differentiation but also for the finite difference scheme for the second derivative. The latter shows a strong error build-up as well and looses quickly several orders of magnitude in accuracy

compared to the actual derivative in machine precision. The behavior seen in Figure 3.3 is common to numerical differentiation schemes and only depends weakly on the choice of the test function. It may be remarked, however, that the error in the Chebyshev differentiation is more distinctive. The Fourier differentiation in eq. (3.46) is more stable but looses a few orders of magnitude for larger grid sizes as well, as shown in Figure 3.4, where the numerical methods from this section are numerically verified for a two-dimensional test function. Those differentiation instabilities are dreadful for the Cauchy-Lagrange algorithm as the recurrence mechanism described in Section 2.6 demands first-order derivatives on the displacement components, thus, second-order derivatives on the Helmholtz-Hodge potentials. Therefore, some effort has been employed to avoid numerical differentiation as far as possible and solve for the needed derivatives directly. The details of these techniques can be found in the proceeding sections.

Integration. The integration methods are obtained in reasoning similarly as above. Here, we search for a series representation of the primitives in the same form as the functions that are integrated. In Fourier space, this is only possible if the zero mode equals 0, since otherwise a linear function appears, which is not periodic in $[0, L]$ and, hence, not representable by a Fourier series. Furthermore, indefinite integration is always defined up to a constant and definite integration, in the form $f(z) = \int_{z_1}^{z} f'(t)\mathrm{d}t$, can only be performed correctly if the primitives value at z_1 is known. To be more precise, sometimes one may obtain a numerical derivative of some order of a solution to a problem in almost the same precision as the solution itself. In that case it is preferred to integrate the derivative instead of differentiating the solution to a certain order. Roughly speaking, for sufficiently high dimensions, one looses one order of magnitude in the relative error for every differentiation in Fourier space and gains one order for every integration. This aspect can clearly be envisioned in writing the primitives $f^{(j)}$, $j < 0$, of the Fourier series $f^{(0)}$, which are

$$f^{(j)}(z) = \sum_{k=1}^{M} \hat{f}_k^{(j)} \, \mathrm{e}^{\mathrm{i}\tilde{k}z}, \qquad \text{with } \hat{f}_k^{(j)} = \frac{1}{(\mathrm{i}\tilde{k})^{|j|}} \hat{f}_k^{(0)} \ , \ k = 1, \dots, M \, . \tag{3.46}$$

The same thinking applies to Chebyshev series. One shows, in using relation (3.37), that

$$g_l^{(-1)} = \frac{c_{l-1} g_{l-1}^{(0)} - g_{l+1}^{(0)}}{4l}, \quad l = N+1, N, \dots, 1, \tag{3.47}$$

with $g_{N+2}^{(0)} = g_{N+1}^{(0)} = 0$. The constants of integration $\hat{f}_0^{(-1)}$ and $g_l^{(-1)}$, which cannot be obtained by coefficient relations, have to be determined in using known values of the primitives. If such values are known, then the zero modes may be determined and one computes the coefficients of the second primitive, except its zero mode, via re-application of the above relation:

$$g_l^{(-2)} = \frac{c_{l-1} g_{l-1}^{(-1)} - g_{l+1}^{(-1)}}{4l}, \quad l = N+2, N+1, \dots, 1, \tag{3.48}$$

where $g_{N+3}^{(-1)} = g_{N+2}^{(-1)} = 0$. In practice, the coefficients above $g_N^{(j)}$ are dropped to preserve the length of the coefficients vector. In contrary to numerical differentiation, the above integration schemes are robust and show exponential convergence. The reason for this stability is the fact that the modes are divided by the wave-numbers instead of multiplied by them. The relative error of the computed primitives to the exact primitives plateaus quickly around machine epsilon, once the function that is being integrated is accurately represented by its truncated spectral series.

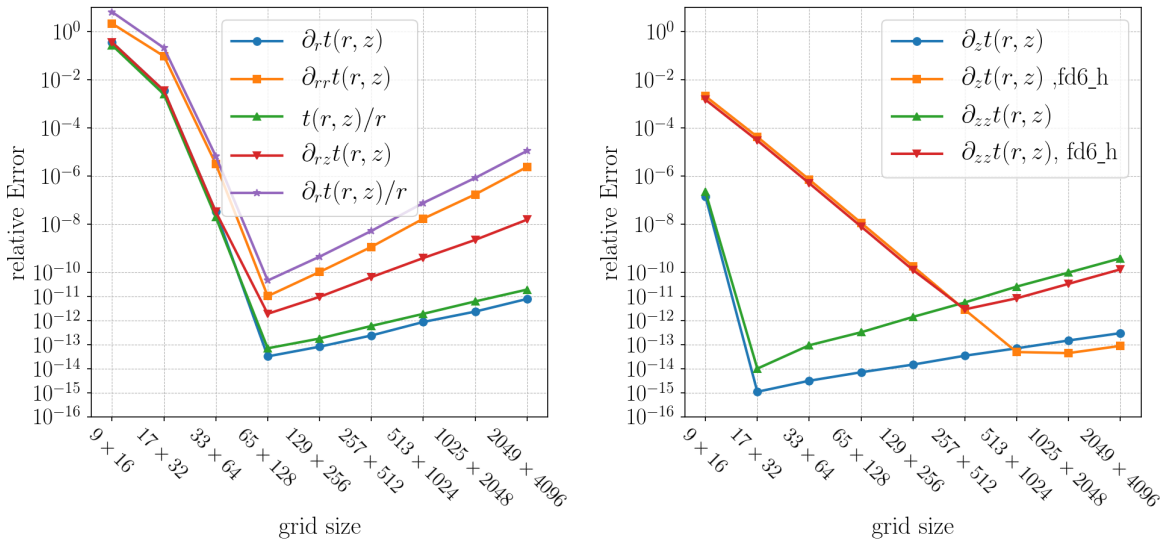


Figure 3.4: Numerical verification of the implemented spectral differentiation methods for $t = t(r, z)$ from eq. (3.35), where $L = 2\pi$. All shown derivatives and divisions by r are obtained through spectral methods except those that are marked by fd6_h indicating the finite difference method of sixth order. The second order derivatives in particular show an unstable behavior for higher grid sizes.

Division by the radial argument. A straightforward implementation of the recurrence relations eqs. (2.66), (2.68) and (2.69) comprises a division by the radial argument $r \in [0, 1]$. This can be inconvenient to deal with, when staying in the physical space, because of the clustering of the Chebyshev grid near zero. The grid also contains zero as a grid point and the corresponding limit has to be found by other means. Hence, it is convenient to have a way to perform a division by r in the shifted Chebyshev spectral space. Although there exist concise coefficient transforms for non-shifted Chebyshev series that solve this problem in spectral space [97, p. 188], those methods may not be applied to a shifted Chebyshev series. Given a shifted Chebyshev series $g = \sum_{l=0}^{N} g_l T_l^*$, we look for a representation of

$$\tilde{g}(r) := \frac{g(r)}{r} \quad r \in [0, 1], \tag{3.49}$$

in form of a Chebyshev series

$$\tilde{g}(r) = \sum_{l=0}^{N} a_l T_l^*(r), \tag{3.50}$$

such that $(a_l)_{0 \le l \le N}$ is obtained from $(g_l)_{0 \le l \le N}$, whenever g may be divided by r. A straight forward way of finding the coefficients a_l is to simply solve for them. We multiply eq. (3.49) (and eq. (3.50)) by r and use eq. (3.17) to obtain

$$\begin{aligned} r\tilde{g}(r) &= \sum_{l=0}^{N} a_l r T_l^*(r) \\ &= \sum_{l=0}^{N} a_l \frac{1}{4} \left(T_{|l-1|}^* + 2T_l^* + T_{l+1}^* \right)(r) = \sum_{l=0}^{N} g_l T_l^*(r) = g(r), \end{aligned} \tag{3.51}$$

which corresponds to the linear system

$$A \cdot (a_l)_l = (g_l)_l, \tag{3.52}$$

where

$$A = \begin{pmatrix} 1/2 & 1/4 & 0 & 0 & \dots \\ 1/2 & 1/2 & 1/4 & 0 & \\ 0 & 1/4 & 1/2 & 1/4 & \\ 0 & 0 & 1/4 & 1/2 & \\ \vdots & & & & \ddots \end{pmatrix} . \tag{3.53}$$

Note that the system (3.52) is solvable only if g can be divided by r. Since the solving matrix A is in tri-diagonal form, the system can be solved in $O(N)$ operations.
However, numerous tests have shown that this spectral division is quite unstable. It becomes valuable, if an explicit limit at $r = 0$ may not be known, because eq. (3.52) may be solved for all the coefficients $\tilde{g}$ without further information. Furthermore, the spectral division scheme does not leave the Chebyshev spectral space and, hence, no transforms are needed, which makes it faster in a fully spectral implementation. Nonetheless, explicit division by r seems to be accurate to machine precision most of the time, depending on the function that is divided of course, but the value at $r = 0$ has to be found additionally. Table 3.1 compares the two methods on a test function, which shows an accuracy loss in the explicit division on higher grid sizes. In all tested functions, the relative error in the (shifted) spectral division was by at least one order of magnitude worse than the error in the explicit division. This seems odd, since the solving matrix A from above is rather well behaved, but a reason for this discrepancy has not been pursued. Both methods, explicit and spectral, are stated for reference only, because the needed expressions that contain a division by r are directly solved for, as described in the next section.

Table 3.1: Numerical comparison of the implemented spectral division method to an explicit division on the shifted Chebyshev extrema grid (explicit limit set at $r = 0$) for $s = s(r)$ from eq. (3.34). The table shows relative errors to the exact values. The spectral division method looses quickly its accuracy for increasing grid sizes, but, before, it shows exponential convergence as expected.

grid size	8	32	64	128	512	2048
spectral	2.52×10^{-1}	3.84×10^{-8}	8.66×10^{-14}	3.30×10^{-14}	4.95×10^{-12}	5.43×10^{-11}
explicit	2.21×10^{-16}	3.58×10^{-16}	1.52×10^{-15}	1.37×10^{-15}	3.72×10^{-14}	1.48×10^{-13}

3.2 Solving Directly for Derivatives

For later reference, this section contains the exact derivatives and divisions by r of the solutions to model problems and how they are implemented in the CLA code. In other words, this section describes how it is possible to avoid numerical differentiation in solving directly for the derivatives and divisions by r. In particular, this method aims at a maximization of accuracy in the derivatives.
Generally speaking, the present method exploits that the inverse of a linear differential operator is also linear and so are the derivatives of it. When working in spectral spaces, then the linear inverse of an operator and its derivatives may directly be applied to the spectral representation of a RHS function and entail the solution in physical space through a matrix-vector multiplication. This operation, however, leaves the spectral space, but produces accurate values in physical space. To be more precise, we assume a uniquely solvable problem $Lu = f$ with a linear operator L that has an inverse L^{-1}, and a RHS function f that is given by a (truncated) spectral series $f(t) = \sum_{l=0}^{N} b_l \Psi_l(t)$, with $t \in I$, for some connected interval $I \subset \mathbb{R}$, and a set of basis functions $\{\Psi_l\}_{0 \le l \le N}, N \in \mathbb{N}$. We seek to know the solution and

its derivatives on a discrete subset $\{x_i\}_{0\leq i\leq N} \subset I$ that is usually related to the set of basis functions via fast transforms from known function values in physical space to spectral space (e.g. Chebyshev series are related to the Chebyshev extrema by a DCT-I). If $L^{-1,(j)}$ represents the derivative of order $j \in \mathbb{N}_0$ of the inverse operator, then the derivative of order j of the solution u to the linear problem on the discrete set of interval points is given by

$$u^{(j)}(x_i) = L^{-1,(j)} f(x_i) = \sum_{l=0}^{N} b_l L^{-1,(j)} \Psi_l(x_i), \quad \forall i \in \{0, \dots, N\}, \tag{3.54}$$

due to the linearity of the inverse operator and its derivatives. The last equation corresponds to a matrix-vector multiplication, namely

$$(u^{(j)}(x_i))_{i=0,\dots,N} = (L^{-1,(j)} \Psi_l(x_i))_{i,l=0,\dots,N} \cdot (b_l)_{l=0,\dots,N}\,. \tag{3.55}$$

The numerical complexity of this method is exactly $2(N+1)^2$ operations, which is in the typical realm of $O(N^2)$ for linear systems with full solving matrices. The accuracy of the solution and derivatives thereof is only reduced by rounding errors in the summation and by inaccuracies in the coefficients of the RHS function. In particular, there is no amplification of errors that stems from multiplications with mode numbers as discussed in Section 3.1.3. The computation of the entries $(L^{-1,(j)} \Psi_l(x_i))_{i,l}$ has to be performed only once, but may be challenging, especially if the inverse operator comprises special functions. The modified Bessel equation, which is treated in Section 3.3.3, is an example of a challenging computation scenario. Instead of taking derivatives, one may apply any other bounded linear operator on the inverse L^{-1}, such as valid division by the argument. In the remainder of this section, we apply the above method to simple model problems with underlying shifted Chebyshev discretization. This will serve as a reference for later purposes.

Recalling the operator $\mathcal{L}$ from Section 2.5, the first problem reads

$$\begin{aligned} \mathcal{L}u(r) = \frac{1}{r}\partial_r \left(ru\right)(r) = \frac{u}{r} + u'(r) = f(r), \\ u(0) = 0, \end{aligned} \tag{3.56}$$

where $r \in [0,1]$ and $f(r) = \sum_{l=0}^{N} b_l T_l^*(r), N \in \mathbb{N}$. This is the one dimensional version of the problem in eq. (2.89). As already stated in eq. (2.90) the inverse of $\mathcal{L}$ reads

$$\mathcal{L}^{-1} f(r) = \frac{1}{r}\int_0^r t\, f(t)\,\mathrm{d}t = \sum_{l=0}^{N} b_l \left(\frac{1}{r}\int_0^r t\, T_l^*(t)\,\mathrm{d}t\right) = u(r)\,. \tag{3.57}$$

One finds the first derivative and division by r via a direct application to $\mathcal{L}^{-1}$, resulting in

$$u'(r) = \partial_r \mathcal{L}^{-1} f(r) = f(r) - \frac{1}{r^2}\int_0^r t\, f(t)\,\mathrm{d}t = \sum_{l=0}^{N} b_l \left(T_l^*(r) - \frac{1}{r^2}\int_0^r t\, T_l^*(t)\,\mathrm{d}t\right), \tag{3.58}$$

$$\frac{u(r)}{r} = \frac{\mathcal{L}^{-1}}{r} f(r) = \frac{1}{r^2}\int_0^r t\, f(t)\,\mathrm{d}t = \sum_{l=0}^{N} b_l \left(\frac{1}{r^2}\int_0^r t\, T_l^*(t)\,\mathrm{d}t\right) = \sum_{l=0}^{N} b_l \frac{\mathcal{L}^{-1} T_l^*(r)}{r}\,. \tag{3.59}$$

We will abbreviate those operators by

$$\mathcal{L}^{-1,(1)} := \partial_r \mathcal{L}^{-1} \tag{3.60}$$

$$\mathcal{L}^{-1,(0)/r} := \frac{\mathcal{L}^{-1}}{r}\,. \tag{3.61}$$

If both terms are needed, then it is preferable to only compute one, the division by r for instance, and find the other one by algebraic modifications of the original equation, i.e. for u' calculate $u' = f - u/r$. Note that the terms in the sums (3.59) and (3.58) are always bounded. At this point, the calculation of $\mathcal{L}^{-1}T_l^*(r)/r$ can be performed by numerical integration and explicit division by r in preferably high precision. A limit has to be taken explicitly at $r = 0$. Modern computer algebra systems, such as Mathematica or Matlab, perform the integration and division readily in arbitrary precision, such that full accuracy in the needed working precision is assured. For a computation on the fly, we may simplify the expression further, using the function τ defined in eq. (3.24), to find

$$\frac{1}{r^2}\int_0^r t\, T_l^*(t)\,\mathrm{d}t = \frac{\tau(l,r)-\tau(l,0)}{r^2}, \tag{3.62}$$

which takes the form, here for $l \geq 3$,

$$\begin{aligned}\frac{\tau(l,r)-\tau(l,0)}{r^2} = \frac{1}{16r^2}\Bigg(&- \frac{T_{l-2}^*(r)-(-1)^{l-2}}{l-2} + \frac{2(T_{l+1}^*(r)-(-1)^{l+1})}{l+1} \\ &- \frac{2(T_{l-1}^*(r)-(-1)^{l-1})}{l-1} + \frac{T_{l+2}^*(r)-(-1)^{l+2}}{l+2}\Bigg),\end{aligned} \tag{3.63}$$

where we have used that $T_l^*(0) = T_l(-1) = (-1)^l$. As mentioned before, this term is bounded for all $l \geq 3$, and the same holds for $l = 0, 1, 2$, where one inserts the respective function values of τ, defined in eq. (3.24). The above expression, eq. (3.63), may be computed directly in modern Fortran, where the Chebyshev polynomials are implemented up to quadruple precision since the 2008 standard. To ensure that the clustering of values close to $r = 0$ poses no potential accuracy loss in the division, one may evaluate the expressions in quadruple precision to obtain accurate values in double precision. The limit $r \to 0$ is easily obtained in applying L'Hôpitals rule

$$\lim_{r\to 0}\frac{1}{r^2}\int_0^r t\, T_l^*(t)\,\mathrm{d}t = \lim_{r\to 0}\frac{rT_l^*(r)}{2r} = \frac{(-1)^l}{2}\,. \tag{3.64}$$

The values u_i of the solution u to eq. (3.56) on the grid points $(r_i)_i$ are obtained by the matrix-vector multiplication

$$A \cdot b = (u_i)_i, \tag{3.65}$$

where b is the vector that contains the shifted Chebyshev coefficients of the RHS function f. The CLA program reads a binary file that contains the matrix $A \coloneqq \{\mathcal{L}^{-1}T_l^*(r_i)/r_i\}_{i,l=0,\ldots,N}$ accurately in quadruple precision for $N = 1024$, that is on 1025 shifted Chebyshev extrema $(r_i)_{i=0,\ldots,N}$. To avoid confusion, the matrix is saved in the binary file in quadruple precision, but in the program itself it is always rounded down (if necessary), such that the matrix-vector multiplications are performed in working precision (double, extended or quadruple). Whenever a Chebyshev dimension of the form $2^N + 1 < 1025$ is needed, then the precise entries are extracted from the saved Matrix, which is possible due to the nesting property of the Chebyshev grids, as stated in (3.8). In this way, the matrix A has to be computed only once and is read into the running program in a matter of milliseconds. A matrix for $N = 2048$ is available too, but rarely needed, because the minimal distance between two neighboring Chebyshev extrema, for the saved grid of size 1025, is

$$1 - \cos(\pi/1024) \approx 4.71{\cdot}10^{-6}, \tag{3.66}$$

Table 3.2: Numerical convergence and accuracy verification of $\mathcal{L}^{-1,(1)}\mathcal{L}^{-1,(0)/r}$ (matrix multiplication) for $s = s(r)$ from eq. (3.34). The table shows relative errors to exact values. The errors for each expression form a plateau at around machine epsilon in double precision.

grid size	9	17	33	65	513	1025
$s(r)$	6.55×10^{-2}	3.33×10^{-4}	1.61×10^{-9}	$4\ .18 \times 10^{-16}$	4.35×10^{-16}	4.17×10^{-16}
$s(r)/r$	6.22×10^{-2}	3.72×10^{-4}	2.03×10^{-9}	$3\ .23 \times 10^{-16}$	3.40×10^{-16}	3.31×10^{-16}
$s'(r)$	3.06×10^{-2}	1.73×10^{-4}	9.47×10^{-10}	$1\ .75 \times 10^{-16}$	1.83×10^{-16}	1.71×10^{-16}

so that, for $L = 1$, roughly 10^6 vertical equi-spaced Fourier grid points would be needed to avoid strongly elongated grid cells near the pole and boundary. A Chebyshev grid size of 513 points ($N = 512$) seems to be sufficient for most cases given the excellent approximation properties of the (shifted) Chebyshev series. Table 3.2 shows the convergence and stability properties of the described method for a test function. As visualized in the Table, the accuracy and stability of the method is striking and one may question if such results can be achieved by any other differentiation method in double precision. We observe, that the relative errors of the first derivative plateau below machine epsilon ($\sim 2.22 \times 10^{-16}$). Similar results are obtained when the test is performed in extended (10 Byte) or quadruple (16 Byte) precision, only that the errors plateau at around the respective machine epsilon once those values have been reached.

The next model problems involve the operators defined in (2.79) and (2.80), namely

$$\mathcal{L}_1 \coloneqq \tfrac{1}{r}\partial_r(r\partial_r), \tag{3.67}$$

$$\mathcal{L}_2 \coloneqq \partial_r\left(\tfrac{1}{r}\partial_r(r\,.\,)\right), \tag{3.68}$$

and read

$$\begin{aligned} \mathcal{L}_1 u_1(r) &= f_1(r), \quad r \in [0,1], \\ u_1'(0) &= u_1'(1) = 0 \end{aligned} \tag{3.69}$$

and

$$\begin{aligned} \mathcal{L}_2 u_2(r) &= f_2(r), \quad r \in [0,1], \\ u_2(0) &= u_2(1) = 0\,. \end{aligned} \tag{3.70}$$

A necessary and sufficient condition for the unique solvability of (3.69) up to an additive constant is

$$\int_0^1 y f_1(y)\,\mathrm{d}y = 0, \tag{3.71}$$

which is assumed to hold in the following. Subsequently, a direct inversion of the operator $\mathcal{L}_1$ entails the solution to (3.69), up to an additive constant, in the form

$$\begin{aligned} u_1(r) = \mathcal{L}_1^{-1} f_1(r) &= \int_0^r \frac{1}{t}\int_0^t y f_1(y)\,\mathrm{d}y\,\mathrm{d}t \\ &= \ln(r)\left(\int_0^r y f_1(y)\,\mathrm{d}y\right) - \int_0^r y f_1(y)\ln(y)\,\mathrm{d}y, \end{aligned}$$

with derivatives

$$\begin{aligned} u_1'(r) &= \partial_r \mathcal{L}_1^{-1} f_1(r) = \frac{\int_0^r y f_1(y)\,\mathrm{d}y}{r}, \\ u_1''(r) &= \partial_r^2 \mathcal{L}_1^{-1} f_1(r) = f_1(r) - \frac{\int_0^r y f_1(y)\,\mathrm{d}y}{r^2}\,. \end{aligned} \tag{3.72}$$

The vanishing of the first derivative at $r = 0$, respectively at $r = 1$, is readily verified under consideration of the continuity of f, respectively condition (3.71).
The inverse $\mathcal{L}_2^{-1}$, applied to the RHS function f_2, produces the solution u_2 to problem (3.70):

$$u_2(r) = \mathcal{L}_2^{-1} f_2(r) = \frac{\int_0^r t \left(c + \int_0^t f_2(x)\, dx \right) dt}{r}, \tag{3.73}$$

where

$$c = -2 \int_0^1 t \int_0^t f_2(x)\, dx\, dt$$

ensures homogeneity at $r = 1$. The vanishing of $\mathcal{L}_2^{-1} f_2$ at $r = 0$ follows readily from the continuity of the RHS function f_2. The first two derivatives read

$$\begin{aligned} u_2'(r) &= \partial_r \mathcal{L}_2^{-1} f_2(r) = -\frac{\int_0^r t \left(c + \int_0^t f_2(x)\, dx \right) dt}{r^2} + c + \int_0^r f_2(x)\, dx, \\ u_2''(r) &= \partial_r^2 \mathcal{L}_2^{-1} f_2(r) = \frac{1}{r} \left(\frac{2 \int_0^r t \left(c + \int_0^t f_2(x)\, dx \right) dt}{r^2} - c - \int_0^r f_2(x)\, dx \right) + f_2(r)\,. \end{aligned} \tag{3.74}$$

The exact solutions u_1, u_2 may be approximated in approximating f_1, f_2 by truncated shifted Chebyshev series. The inverse operators and their derivatives act directly on the Chebyshev basis functions by linearity,

$$\partial_r^j \mathcal{L}_\iota^{-1} \left(\sum_{l=0}^N a_l T_l^* \right)(r) = \sum_{l=0}^N f_{i,l} \partial_r^j \mathcal{L}_\iota^{-1} \left(T_l^* \right)(r), \tag{3.75}$$

where $\iota = 1, 2,\ j = 0, 1, 2$ and $a_0, \ldots, a_N \in \mathbb{R}$. Equally for the division by r, only that $\partial_r^j \mathcal{L}_\iota^{-1}$ in eq. (3.75) gets replaced by

$$\frac{1}{r} \partial_r \mathcal{L}_1^{-1} f_1(r) = \frac{\int_0^r y f_1(y)\, \mathrm{d}y}{r^2}, \tag{3.76}$$

or

$$\frac{1}{r} \mathcal{L}_2^{-1} f_2(r) = \frac{\int_0^r t \left(c + \int_0^t f_2(x)\, dx \right) dt}{r^2}\,. \tag{3.77}$$

As in the treatment of $\mathcal{L}$, most of the involved integrals simplify in using eqs. (3.23) and (3.24) as in eq. (3.62). For instance, we have, for $l > 1$,

$$\begin{aligned} \partial_r \mathcal{L}_2^{-1} T_l^*(r) &= -\frac{\int_0^r t \left(\hat{c} + \int_0^t T_l^*(x)\, dx \right) dt}{r^2} + \hat{c} + \int_0^r T_l^*(x)\, dx \\ &= \frac{\hat{c}}{2} - \frac{1}{4r^2} \int_0^r t \left(\frac{T_{l+1}^*(t)}{l+1} - \frac{T_{l-1}^*(t)}{l-1} \right) - t \left(\frac{(-1)^{l-1}}{l+1} - \frac{(-1)^{l-1}}{l-1} \right) \mathrm{d}t \\ &\quad + \frac{1}{4} \left(\frac{T_{l+1}^*(t)}{l+1} - \frac{T_{l-1}^*(t)}{l-1} \right) - \frac{1}{4} \left(\frac{(-1)^{l-1}}{l+1} - \frac{(-1)^{l-1}}{l-1} \right) \\ &= \frac{\hat{c}}{2} - \frac{1}{4r^2} \left(\left(\frac{\tau(l+1,t) - \tau(l+1,0)}{l+1} \right) - \left(\frac{\tau(l-1,t) - \tau(l-1,0)}{l-1} \right) \right) \\ &\quad + \frac{(-1)^l}{4(l^2-1)}, \end{aligned} \tag{3.78}$$

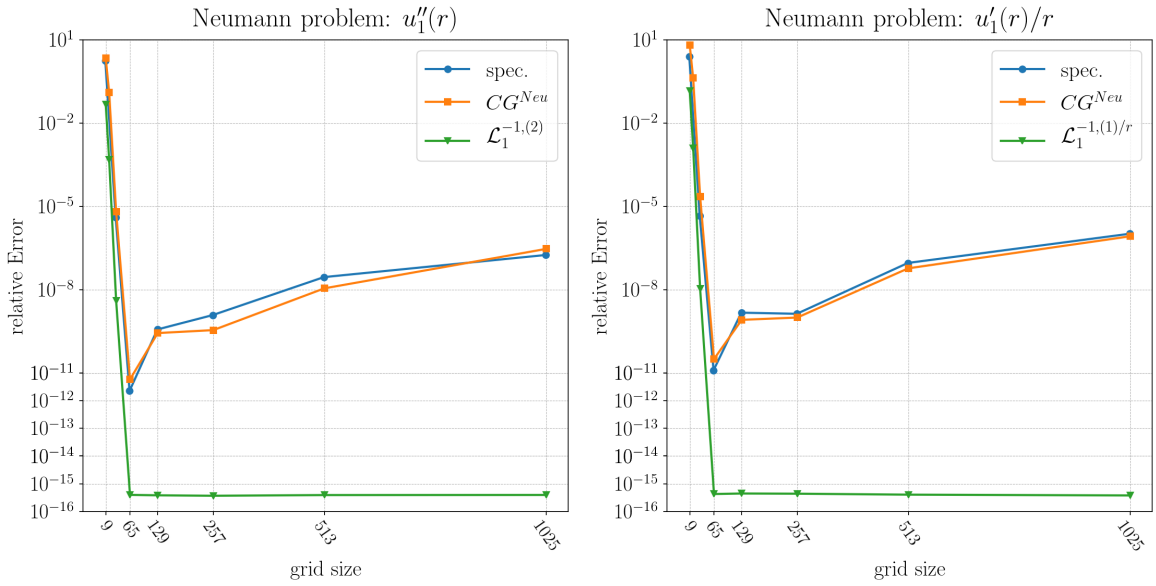


Figure 3.5: Convergence and accuracy verification for the methods $\mathcal{L}_1^{-1,(2)}$ and $\mathcal{L}_1^{-1,(1)/r}$ with the test function $u_1(r) = s(r)$ from eq. (3.35). The RHS function is analytically calculated as $f_1(r) = \mathcal{L}_1 u_1(r)$ and implemented exactly. The legend references the graphs by the method that was used to obtain them. The (CG) methods are presented later in this chapter and present well suited Chebyshev-Galerkin solvers with basis functions that fulfill the boundary conditions individually. Here the (CG) methods are used to solve eqs. (3.69) and (3.70) for the solutions in order to subsequently apply the standard spectral differentiation and division methods from Section 3.1.3. Those spectral differentiation and division methods were also applied directly to the spectral projection of the function s (no solving involved) with results referenced by (spec.)

where τ is defined in (3.24), and

$$\begin{aligned}
\hat{c} &= -2\int_0^1 t \int_0^t T_l^*(x)\,dx\,dt \\
&= -\frac{1}{2}\int_0^1 t\left(\frac{T_{l+1}^*(t)}{l+1} - \frac{T_{l-1}^*(t)}{l-1}\right) - t\left(\frac{(-1)^{l-1}}{l+1} - \frac{(-1)^{l-1}}{l-1}\right) \mathrm{d}t \\
&= -\frac{1}{2}\left(\left(\frac{\tau(l+1,1)-\tau(l+1,0)}{l+1}\right) - \left(\frac{\tau(l-1,1)-\tau(l-1,0)}{l-1}\right) - \frac{(-1)^l}{(l^2-1)}\right).
\end{aligned} \tag{3.79}$$

The expressions that are needed for the CLA code, namely

$$\begin{aligned}
\mathcal{L}_1^{-1,(2)} &:= \partial_r^2 \mathcal{L}_1^{-1}, \\
\mathcal{L}_1^{-1,(1)/r} &:= \frac{\partial_r \mathcal{L}_1^{-1}}{r}, \\
\mathcal{L}_2^{-1,(1)} &:= \partial_r \mathcal{L}_2^{-1}, \\
\mathcal{L}_2^{-1,(0)/r} &:= \frac{\mathcal{L}_2^{-1}}{r},
\end{aligned} \tag{3.80}$$

given explicitly in eqs. (3.72), (3.74), (3.76) and (3.77), may all be evaluated exactly and can be given in terms of shifted Chebyshev polynomials and divisions by r.

In principle, one may continue simplifying until a coefficient transform is found, that transforms the shifted Chebyshev coefficients of the RHS functions $f_{1/2}$ (meaning f_1 or f_2) to the coefficients of the needed expression, such as the derivative of the solution to one of the

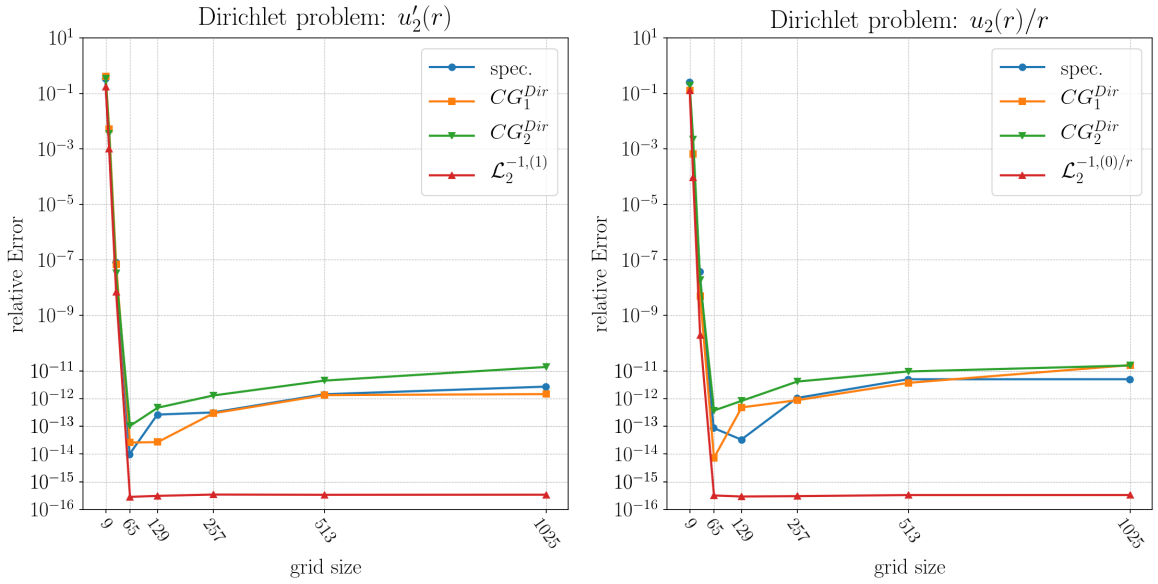


Figure 3.6: Convergence and accuracy verification for the methods $\mathcal{L}_2^{-1,(1)}$ and $\mathcal{L}_2^{-1,(0)/r}$ and the test function $u_2(r) = s(r)$ from eq. (3.35). The legend references the graphs by the method that was used for their creation as in Figure 3.5.

above problems. This would, however, be equivalent to just performing multiplication, division, integration and differentiation on $f_{1/2}$ in spectral space, using the transforms that were described in Section 3.1.3. For example, $\partial_r u_1$ is given by $\partial_r \int_0^r \frac{1}{t} \int_0^t y f_1(y)\,\mathrm{d}y\,\mathrm{d}t$, so that one may execute a multiplication, an integration, a division by the argument, an integration and a differentiation in this order in spectral space via coefficient transforms in linear complexity, while taking care of exactly known values at $r = 0$ for the integration and division. Theoretically, this method seems solid, practically, however, the division by the argument and the differentiation in the shifted Chebyshev space introduce large numerical errors, especially for higher dimensions as discussed in Section 3.1.3 and visualized in Figure 3.3. Because of the latter argument and the applied focus on accuracy rather than speed, the matrix-vector multiplications

$$(\mathcal{D}\, T_l^*(r_i))_{i,l=0,\dots,N} \cdot (a_l)_{l=0,\dots,N}, \tag{3.81}$$

are implemented, where $(a_l)_l$ are the spectral coefficients of the RHS function, $(r_i)_{i=0,\dots,N}$ the shifted Chebyshev extrema and $\mathcal{D}$ any one of the symbols from (3.80). The entries of the matrices were evaluated to high precision via arbitrary precision packages, rounded correctly to quadruple precision and saved to disc for continuous re-utilization.

Notation. We will refer to the above methods, which execute a matrix-vector multiplication to obtain a certain expression, simply by their corresponding operators defined in eqs. (3.60), (3.61) and (3.80), i.e. $\mathcal{L}^{-1,(1)}$, $\mathcal{L}^{-1,(0)/r}$, $\mathcal{L}_1^{-1,(2)}$, $\mathcal{L}_1^{-1,(1)/r}$, $\mathcal{L}_2^{-1,(1)}$ and $\mathcal{L}_2^{-1,(0)/r}$.

Figures 3.5 and 3.6 show a numerical verification of the quality of the presented technique in comparison to spectrally computed expressions. Once the standard spectral derivative methods from Section 3.1.3 are applied to the Chebyshev-projection of the original function and once to the approximated solutions to the problems in eqs. (3.69) and (3.70). The approximated solutions are obtained by well suited Chebyshev-Galerkin solvers that are discussed in detail later in Section 3.3.2. As for $\mathcal{L}^{-1}$, the methods for $\mathcal{L}_{1/2}^{-1}$ produce excellent accuracy results that plateau at around machine epsilon.

3.3 Numerical Methods for the Poisson Problems

The Poisson problems, resulting from the Helmholtz-Hodge decomposition in Section 2.5, play a key role in our algorithm, as they do in most numerical simulations in the field of hydrodynamics. The handling of those problems is crucial for the success of an implementation, general performance criteria being convergence, accuracy and speed. While we achieve exponential convergence in using a spectral discretization, the speed of the methods in use was not prioritized. The absence of a CFL condition, which would enforce small time-steps, makes the CLA generally quite fast when compared to other time-stepping methods such as Runge-Kutta. The greatest attention in the approach presented in this thesis was given to the accuracy of the solutions and their derivatives. Special effort is employed to solve the decoupled equations directly for the needed derivatives as in Section 3.2. Due to the structure of the involved operators and the used Fourier expansions, the Poisson problems decouple, which leads to ordinary differential equations (ODEs) in the Fourier modes. Those ODEs are known as the modified Bessel equations. In the spirit of Section 3.2, we can compute analytically the derivatives of the exact solutions to those equations and may insert a spectral Chebyshev approximation of the source terms, which entails striking accuracy in the results. For, the explicit solutions to the before-mentioned equations are the modified Bessel functions and the solving process comprises simple matrix-vector multiplications with the shifted Chebyshev series coefficients, this method is baptized Bessel-Chebyshev (BC) method. Before we come to the details of this, to the knowledge of the author, unprecedented method in Section 3.3.3, we will introduce the homogeneous model problems and recall a well suited Chebyshev-Galerkin solver from [108].

3.3.1 Decoupled Poisson Problems

Let, in the following, $u = u(r, z)$ and $w = w(r, z)$ be sufficiently smooth z-periodic functions $[0, 1] \times [0, L] \to \mathbb{R}$. In sight of eqs. (2.83) and (2.85) and the fact that in our approach we will not solve eqs. (2.84) and (2.86) explicitly, we will treat the following homogeneous model problems

$$\begin{aligned} \mathcal{G}u \coloneqq \left(\partial_r^2 + \tfrac{1}{r}\partial_r + \partial_z^2\right) u(r,z) &= f(r,z), \\ \partial_r \,|_{r=0,1}\, u(r,z) &= 0, \end{aligned} \tag{3.82}$$

and

$$\begin{aligned} \mathcal{H}w \coloneqq \left(\partial_r^2 + \tfrac{1}{r}\partial_r - \tfrac{1}{r^2} + \partial_z^2\right) w(r,z) &= g(r,z), \\ w(0,z) = w(1,z) &= 0\,. \end{aligned} \tag{3.83}$$

As discussed in Section 3.1, we use the periodicity in vertical direction and anticipate an approximation of the solutions by truncated Fourier series. Let

$$\tilde{k} \coloneqq \frac{2\pi k}{L}\,. \tag{3.84}$$

We may write

$$\mathcal{G}u_M = \mathcal{G}\left(\sum\nolimits_{|k|=0}^{M/2} \hat{u}_k(r)\, \mathrm{e}^{\mathrm{i}\tilde{k}z}\right) = f_M(r,z) = \sum\nolimits_{|k|=0}^{M/2} \hat{f}_k(r)\, \mathrm{e}^{\mathrm{i}\tilde{k}z}, \tag{3.85}$$

$$\mathcal{H}w_M = \mathcal{H}\left(\sum\nolimits_{|k|=0}^{M/2} \hat{w}_k(r)\, \mathrm{e}^{\mathrm{i}\tilde{k}z}\right) = g_M(r,z) = \sum\nolimits_{|k|=0}^{M/2} \hat{g}_k(r)\, \mathrm{e}^{\mathrm{i}\tilde{k}z}, \tag{3.86}$$

or, equivalently,

$$\sum\nolimits_{|k|=0}^{M/2}(\mathcal{L}_1-\tilde{k}^2)\hat{u}_k(r)\,\mathrm{e}^{\mathrm{i}\tilde{k}z}=\sum\nolimits_{|k|=0}^{M/2}\hat{f}_k(r)\,\mathrm{e}^{\mathrm{i}\tilde{k}z}, \tag{3.87}$$

$$\sum\nolimits_{|k|=0}^{M/2}(\mathcal{L}_2-\tilde{k}^2)\hat{w}_k(r)\,\mathrm{e}^{\mathrm{i}\tilde{k}z}=\sum\nolimits_{|k|=0}^{M/2}\hat{g}_k(r)\,\mathrm{e}^{\mathrm{i}\tilde{k}z}\ . \tag{3.88}$$

As a consequence, the equations decouple and for each $k\in[0,{}^{M}\!/_{2}]$ we solve the ordinary differential equations

$$\begin{aligned}(\mathcal{L}_1-\tilde{k}^2)\hat{u}_k(r)=\hat{u}_k''(r)+\tfrac{1}{r}\hat{u}_k'(r)-\tilde{k}^2\hat{u}_k(r)=\hat{f}_k(r),\\ \hat{u}_k'(0)=\hat{u}_k'(1)=0,\end{aligned} \tag{3.89}$$

and

$$\begin{aligned}(\mathcal{L}_2-\tilde{k}^2)\hat{w}_k(r)=\hat{w}_k''(r)+\tfrac{1}{r}\hat{w}_k'(r)-(\tfrac{1}{r^2}+\tilde{k}^2)\hat{w}_k(r)=\hat{g}_k(r),\\ \hat{w}_k(0)=\hat{w}_k(1)=0,\end{aligned} \tag{3.90}$$

where $\mathcal{L}_1$ and $\mathcal{L}_2$ are defined in eqs. (2.79) and (2.80), they are treated analytically in Section 3.2. Since only real valued functions, whose complex modes verify $\hat{u}_k=\overline{\hat{u}_{-k}}$, are treated, we may restrict ourselves to positive modes. Note that eqs. (3.89) and (3.90) are commonly known as the modified Bessel ODEs of order 0 and 1 respectively. Their solutions, well known, yet hard to compute numerically, are stated in Section 3.3.2.

Remark. To obtain the solution of the inhomogeneous problem eq. (2.83), with Neumann boundary data $b=b(z)$ at $r=1$, from a homogeneous problem eq. (3.82), one simply solves the latter with the modified right hand side $\tilde{f}=\tilde{f}(r,z)=f(r,z)-\mathcal{G}(\frac{r^2}{2}b(z))$ for a solution $\tilde{u}=\tilde{u}(r,z)$. The inhomogeneous solution is then given by $\tilde{u}(r,z)+\frac{r^2}{2}b(z)$. The smoothness requirement for b is neglected here, because the procedure is directly applied to the truncated problem, where all involved functions are C^∞.

3.3.2 Modified Galerkin Solver

Let us start with the homogeneous Dirichlet problem eq. (3.90). In [108], the author introduces a Chebyshev-Galerkin scheme for the homogeneous Helmholtz equation in a disc. As a matter of fact, the operators in use are almost identical to ours and, therefore, we may apply his findings to our case. Indeed, the operator in the Helmholtz equation in polar coordinates $(r,\theta)\in[0,1]\times[0,2\pi]$ reads

$$-\Delta+\alpha=-\partial_r^2-\tfrac{1}{r}\partial_r-\tfrac{1}{r^2}\partial_\theta^2+\alpha. \tag{3.91}$$

A Fourier expansion in the angular dimension leads to an application of

$$-\partial_r^2-\tfrac{1}{r}\partial_r+\tfrac{m^2}{r^2}+\alpha \tag{3.92}$$

to the Fourier modes of the solution. If we now let $m=1$ and $\alpha=\tilde{k}^2$, then the negation of the operator (3.92) equates the one in eq. (3.90). Homogeneous Dirichlet conditions at the boundary $r=1$ and at the pole $r=0$ are imposed. The author in [108] further points out that homogeneous Neumann pole conditions are additionally used widely in the literature, but that this choice is non-essential and would introduce inconveniences in the implementation as well as accuracy loss in particular situations. One may add, that while the argumentation for homogeneous Neumann pole conditions stems from the parity argument for the Fourier modes in a disc, the conservative form of our operator reveals readily that only

Dirichlet pole conditions are necessary even without any parity properties. Indeed, the operator $\partial_r^2 + \frac{1}{r}\partial_r - \frac{1}{r^2} = \partial_r(\mathcal{L}\,.\,) = \partial_r(\partial_r + \frac{1}{r})$ implies the differentiablity of the solution divided by r, and thus the vanishing of the latter at $r = 0$, as long as continuity at the pole is imposed.

A Galerkin scheme consists of a variational formulation of a given problem together with a suitable test function space. A finite dimensional sub-space of the latter, spanned by a complete, sometimes orthogonal, set of basis functions, entails a discrete approximation of the solution to the problem. The Lax-Milgram theorem ensures existence and uniqueness if continuity and coercivity conditions on the variational form hold true. For a complete introduction on Galerkin schemes we would like to refer the reader to [109]. Here, we aim directly at a suitable formulation for Chebyshev polynomials.

Notation. The method described below is called Chebyshev-Galerkin (CG) method and abbreviated by CG_1^{Dir}, where the super-/subscripts indicate the Dirichlet boundary condition and the used set of basis functions ($\mathcal{B}_1$ from eq. (3.99)). A second version of this method uses a different set of basis functions (namely $\mathcal{B}_2$ from eq. (3.107)) and will be denoted by CG_2^{Dir}. The Neumann case, which treats another operator, will be referred to as CG^{Neu}.

Starting from eq. (3.90), the radial interval is expanded via the coordinate transform $r = (t+1)/2,\ t \in I := [-1, 1]$, to give

$$y''(t) + \frac{1}{t+1}y'(t) - \left(\frac{1}{(t+1)^2} + \frac{\tilde{k}^2}{4}\right) y(t) = \frac{1}{4}\hat{g}_k((t+1)/2), \tag{3.93}$$

with $y(t) := \hat{w}_k((t+1)/2)$, $k \in [0, M/2]$ fixed, which we multiply by $(t+1)$ to obtain

$$\underbrace{(t+1)y''(t) + y'(t)}_{=((\,.\,+1)y')'(t)} - \left(\frac{1}{t+1} + \frac{\tilde{k}^2}{4}(t+1)\right) y(t) = h(t), \tag{3.94}$$

where $h(t) = \frac{1}{4}(t+1)\hat{g}_k((t+1)/2)$, $\tilde{k} = {}^{(2\pi k)}/_{L}$ as in eq. (3.84), and $y(0) = y(1) = 0$. A suitable test function space is $Y_0 := H^1_{0,\omega}(I)$ with

$$H^1_0(I) := \{\vartheta \in H^1(I) = W^{1,2}(I) \mid \vartheta(-1) = \vartheta(1) = 0 \text{ in the trace sense}\}, \tag{3.95}$$

the subscript ω indicates that the inner product in $H^1_0(I)$ is weighted by $\omega(t) = (1-t^2)^{-1/2}$, i.e. the Chebyshev weight from Section 3.1.2. See [115] for an introduction to weighted Sobolev spaces. In the following let $v \in Y_0$, we apply the inner product $(\,.\,, v)_\omega = \int_I \,.\, v\omega \,\mathrm{d}t$ to both sides of eq. (3.94), execute an integration by parts in the LHS and multiply by (-1) to obtain

$$\begin{aligned}\int_I (t+1)y'(t)(v\omega)'(t)\,\mathrm{d}t + \int_I \frac{1}{t+1}y(t)v(t)\omega(t)\,\mathrm{d}t + \tilde{k}^2 \int_I \frac{1}{4}(t+1)y(t)v(t)\omega(t)\,\mathrm{d}t \\ = -\int_I h(t)v(t)\omega(t)\,\mathrm{d}t\,.\end{aligned} \tag{3.96}$$

The boundary terms from the integration by parts vanish, because of the membership of v in $H^1_0(I)$.
The variational formulation to our problem eq. (3.94) now reads as follows: find $y \in H^1_{0,\omega}(I)$, such that eq. (3.96) holds for each $v \in H^1_{0,\omega}(I)$. In defining

$$Y_0^N := \{p \in P_N \mid p(-1) = p(1) = 0\}, \tag{3.97}$$

with P_N being the polynomial space of maximal degree $N \in \mathbb{N}$, we may state the Galerkin scheme for our homogeneous Dirichlet problem eq. (3.94):

$$\begin{cases} \text{find } y = y^N \in Y_0^N, \\ \text{s.t. eq. (3.96) holds } \forall v = v^N \in Y_0^N, \end{cases} \tag{3.98}$$

in other words, we look for the best approximation of the solution $y \in H^1_{0,\omega}(I)$ in the finite dimensional space Y_0^N. The advantage of the scheme presented in [108] (see also [109]) is that it uses special basis functions that fulfill the boundary conditions exactly and allow additionally for simple structured solving matrices. Those basis functions in the Dirichlet case are given by

$$\begin{aligned} \Phi_l &:= T_l - T_{l+2}, \quad l = 0, \dots, N-2, \\ \mathcal{B}_1 &:= \{\Phi_0, \dots, \Phi_{N-2}\}. \end{aligned} \tag{3.99}$$

One readily verifies the homogeneity via eqs. (3.19) and (3.20) and that $Y_0^N = \operatorname{span}(\mathcal{B}_1)$. In fact, the latter follows immediately from $\mathcal{B}_1 \subset P_N$, its linear independence, and a dimensional argument that goes: $\dim Y_0^N = \dim(\operatorname{span}(\mathcal{B}_1)) = N-1$, because $Y_0^N = \{\sum_{i=0}^N a_i x^i \in P_N \text{ s.t. } \sum_{i=0}^N a_i = 0 \text{ and } \sum_{i=0}^N (-1)^i a_i = 0\}$. Furthermore, the set $P := \{\Phi_l\}_{l \in \mathbb{N}_0}$ is a complete basis of H^1_0, because $\{T_l\}_{l\in\mathbb{N}_0}$ is a complete basis of H^1, but P is not orthogonal. Thus, any function in Y_0^N is given as a linear combination of the basis functions from $\mathcal{B}_1$ and (3.98) must, therefore, only hold for any $v = \Phi_0, \dots, \Phi_{N-2}$. We seek a solution in the form

$$y^N(t) = \sum\nolimits_{l=0}^{N-2} a_l \Phi_l(t), \quad t \in I, \tag{3.100}$$

which we insert into eq. (3.96) to find

$$\begin{aligned} \sum_{l=0}^{N-2} a_l \left(\int_I (t+1)\Phi_l'(t)(\Phi_i\omega)'(t)\,\mathrm{d}t + \int_I \frac{1}{t+1}\Phi_l(t)\Phi_i(t)\omega(t)\,\mathrm{d}t + \tilde{k}^2 \int_I \frac{(t+1)}{4}\Phi_l(t)\Phi_i(t)\omega(t)\,\mathrm{d}t \right) \\ = -\int_I h(t)\Phi_i(t)\omega(t)\,\mathrm{d}t = -\sum_{j=0}^{N} \hat{g}_{kj} \left(\int_I \frac{1}{4}(t+1)T_j(t)\Phi_i(t)\omega(t)\,\mathrm{d}t \right), \end{aligned} \tag{3.101}$$

for each $i = 0, \dots, N-2$. In the last line we have inserted the Chebyshev approximation of g in order to approximate the integral on the RHS. In fact, since we know the values of $\hat{g}_k$ on the shifted Chebyshev nodes, we have $\hat{g}_k(r) = \sum_{j=0}^N \hat{g}_{kj} T_j^*(r)$ and, therefore, $\hat{g}_k((t+1)/2) = \sum_{j=0}^N \hat{g}_{kj} T_j(t)$. To complete the Galerkin scheme CG_1^{Dir}, we solve the following linear system of equations

$$(A + B - \tilde{k}^2 C) \cdot (a_l)_l = -D \cdot (\hat{g}_{kj})_j, \tag{3.102}$$

where the entries of the solving matrices ($\Lambda = (\Lambda_{i,l})$) are given by

$$\begin{aligned} A_{i,l} &= \int_{-1}^{1} (t+1)\Phi_l'(t)(\Phi_i\omega)'(t)\,\mathrm{d}t, \\ B_{i,l} &= \int_{-1}^{1} \frac{1}{t+1}\Phi_l(t)\Phi_i(t)\omega(t)\,\mathrm{d}t, \\ C_{i,l} &= \int_{-1}^{1} \frac{1}{4}(t+1)\Phi_l(t)\Phi_i(t)\omega(t)\,\mathrm{d}t, \\ D_{i,j} &= \int_{-1}^{1} \frac{1}{4}(t+1)T_j(t)\Phi_i(t)\omega(t)\,\mathrm{d}t, \end{aligned} \tag{3.103}$$

for $i, l = 0, \ldots, N-2$ and $j = 0, \ldots, N$. The above integrals integrate exactly and have to be computed only once. From [108] we read the entries of the first three matrices:

$$A_{i,l} = \begin{cases} (i+1)^2\pi, & l = i-1, \\ 2(i+1)(i+2)\pi, & l = i, \\ (i+1)(i+5)\pi, & l = i+1, \\ 4(i+1)\pi, & l \geq i+2, \end{cases} \tag{3.104}$$

where entries are 0 if not specified differently, B is symmetric and tridiagonal with entries

$$B_{i,l} = \begin{cases} 2\pi, & l = i, \\ -\pi, & l = i-1, i+1, \end{cases} \tag{3.105}$$

and C is symmetric and seven-diagonal with super-diagonals

$$C_{i,l} = \begin{cases} (\delta_{i0}+2)\pi/2, & l = i, \\ (\delta_{i0}+1)\pi/4, & l = i+1, \\ -\pi/2, & l = i+2, \\ -\pi/4, & l = i+3, \end{cases} \tag{3.106}$$

where δ denotes the Kronecker delta. We give the matrices for $N = 7$ in the visually more appealing Matrix notation:

$$A = \begin{pmatrix} 4\pi & 5\pi & 4\pi & 4\pi & 4\pi & 4\pi \\ 4\pi & 12\pi & 12\pi & 8\pi & 8\pi & 8\pi \\ 0 & 9\pi & 24\pi & 21\pi & 12\pi & 12\pi \\ 0 & 0 & 16\pi & 40\pi & 32\pi & 16\pi \\ 0 & 0 & 0 & 25\pi & 60\pi & 45\pi \\ 0 & 0 & 0 & 0 & 36\pi & 84\pi \end{pmatrix}, \quad B = \begin{pmatrix} 2\pi & -\pi & 0 & 0 & 0 & 0 \\ -\pi & 2\pi & -\pi & 0 & 0 & 0 \\ 0 & -\pi & 2\pi & -\pi & 0 & 0 \\ 0 & 0 & -\pi & 2\pi & -\pi & 0 \\ 0 & 0 & 0 & -\pi & 2\pi & -\pi \\ 0 & 0 & 0 & 0 & -\pi & 2\pi \end{pmatrix}$$

$$C = \begin{pmatrix} \frac{3\pi}{8} & \frac{\pi}{8} & -\frac{\pi}{8} & -\frac{\pi}{16} & 0 & 0 \\ \frac{\pi}{8} & \frac{\pi}{4} & \frac{\pi}{16} & -\frac{\pi}{8} & -\frac{\pi}{16} & 0 \\ -\frac{\pi}{8} & \frac{\pi}{16} & \frac{\pi}{4} & \frac{\pi}{16} & -\frac{\pi}{8} & -\frac{\pi}{16} \\ -\frac{\pi}{16} & -\frac{\pi}{8} & \frac{\pi}{16} & \frac{\pi}{4} & \frac{\pi}{16} & -\frac{\pi}{8} \\ 0 & -\frac{\pi}{16} & -\frac{\pi}{8} & \frac{\pi}{16} & \frac{\pi}{4} & \frac{\pi}{16} \\ 0 & 0 & -\frac{\pi}{16} & -\frac{\pi}{8} & \frac{\pi}{16} & \frac{\pi}{4} \end{pmatrix}.$$

The proof of the above statements can be found in the given reference and is not repeated here. We shall, however, emphasise the method in calculating the entries of D, which are not stated in [108]. For $i = 0, \ldots, N-2$ and $j = 0, \ldots, N$, we have

$$D_{i,j} = \begin{cases} (\delta_{i1}+1)\pi/16, & j = i-1 \\ (\delta_{i0}+1)\pi/8, & j = i \\ \delta_{i0}\pi/16, & j = i+1 \\ -\pi/8, & j = i+2 \\ -\pi/16, & j = i+3 \end{cases}, \quad D \overset{(N=5)}{=} \begin{pmatrix} \frac{\pi}{4} & \frac{\pi}{16} & -\frac{\pi}{8} & -\frac{\pi}{16} & 0 & 0 \\ \frac{\pi}{8} & \frac{\pi}{8} & 0 & -\frac{\pi}{8} & -\frac{\pi}{16} & 0 \\ 0 & \frac{\pi}{16} & \frac{\pi}{8} & 0 & -\frac{\pi}{8} & -\frac{\pi}{16} \\ 0 & 0 & \frac{\pi}{16} & \frac{\pi}{8} & 0 & -\frac{\pi}{8} \end{pmatrix}.$$

To find the latter, we expand the integrand, using eq. (3.12) from Section 3.1.2, to obtain

$$\begin{aligned}\frac{1}{4}\int_{-1}^{1}(t+1)T_j\Phi_i\omega\,\mathrm{d}t &= \frac{1}{4}\int_{-1}^{1}\left(\frac{T_{j+1}+T_{|j-1|}}{2}+T_j\right)(T_i-T_{i+2})\omega\,\mathrm{d}t \\ &= \frac{1}{8}\int_{-1}^{1}(T_{j+1}T_i+T_{|j-1|}T_i+2T_jT_i-T_{j+1}T_{i+2}-T_{|j-1|}T_{i+2}-2T_jT_{i+2})\omega\,\mathrm{d}t,\end{aligned}$$

and apply the orthogonality property of the Chebyshev polynomials, namely $\int_I T_iT_j\omega\mathrm{d}t = (\delta_{i0}+1)\delta_{ij}\pi/2$. For instance, when $j=i-1, i>0$, the only non-vanishing terms are

$$\frac{1}{8}\int_{-1}^{1}T_{(i-1)+1}T_i\omega\,\mathrm{d}t+\underbrace{\frac{1}{8}\int_{-1}^{1}T_{|(i-1)-1|}T_i\omega\,\mathrm{d}t}_{\neq 0\text{ ,only if }i=1}=(\delta_{i1}+1)\pi/16,$$

where δ denotes the Kronecker symbol. The other entries are computed alike.
The system (3.102) is solved for a solution in the form (3.100) and transformed to the solution $\hat{w}_k^N$ of eq. (3.90) in form of a shifted Chebyshev series, namely $\sum_{j=0}^{N}\hat{w}_{kj}T_j^*$, in using the variable transform $t=2r-1$, $r\in[0,1]$, and the coefficient transform

$$\hat{w}_{kj}=a_j-a_{j-2}\quad,j=0,\dots,N,$$

where we set $a_{-2}=a_{-1}=a_{N-1}=a_N=0$.
The advantage of the structure of the solving matrices is that the system eq. (3.102) can be solved in $O(N)$ operations as described in [107].

One may, however, achieve inherently homogeneous Dirichlet boundary conditions with an even simpler structure of the solving matrices in using a different set of basis functions, namely

$$\begin{aligned}\tilde{\Phi}_l(t) &\coloneqq (1-t^2)T_l(t)\quad,t\in I\\ \mathcal{B}_2 &\coloneqq \{\tilde{\Phi}_l\}_{l=0,\dots,N-2},\end{aligned}\tag{3.107}$$

which are the centerpiece of the second Chebyshev-Galerkin solver CG_2^{Dir}. A Jacobi function version of the latter basis was introduced in [63], where also the Helmholtz equation is studied as a model problem. It turns out that, as for the Jacobi polynomials in [63], the basis (3.107) produces a band-structured solving matrix, which we will specify in the following. Systems with band structure i.e. only the center diagonals are non-zero, can be efficiently solved in $O(N)$ operations, for instance by the LU-decomposition, which preserves the band structure. The only difference of this method to the method CG_1^{Dir} is that the finite dimensional subspace Y_0^N of $H^1_{0,\omega}(I)$ is spanned by the set $\mathcal{B}_2$ instead of $\mathcal{B}_1$ and that the integrals, defining the entries of the solving matrices, take different values. Let $\tilde{A},\tilde{B},\tilde{C},\tilde{D}$ take the places of the matrices A,B,C,D in eqs. (3.102) and (3.103) in which Φ is replaced by $\tilde{\Phi}$. We state the exact entries of the solving matrices, as the author of this thesis could not find them in the literature.

$\tilde{A}$ is a seven-diagonal matrix with non-zero entries

$$\tilde{A}_{i,l} = \begin{cases} -\frac{1}{16}\pi(i-1)^2, & l = i-3, \\ -\frac{1}{8}\pi(i-1)i, & l = i-2, \\ \frac{1}{16}\pi(-\delta_{2,i} + (i-2)i + 5), & l = i-1, \\ \frac{1}{4}\pi\left(i^2+2\right), & l = i, \\ \frac{1}{16}\pi(i(i+2)+5), & l = i+1, \\ -\frac{1}{8}\pi i(i+1), & l = i+2, \\ -\frac{1}{16}\pi(i+1)^2, & l = i+3, \end{cases} \tag{3.108}$$

except that the values have to be multiplied by 2, whenever $i = 0$ or $l = 0$, which applies to the following matrices as well. $\tilde{B}, \tilde{D}$ are symmetric and seven-diagonal and $\tilde{C}$ is symmetric and eleven-diagonal with super-diagonals

$$\tilde{B}_{i,l} = \begin{cases} \frac{1}{8}\pi(2-\delta_{1,i}), & l = i, \\ \frac{1}{16}\pi(\delta_{1,i}-1), & l = i+1, \\ -\frac{\pi}{8}, & l = i+2, \\ \frac{\pi}{16}, & l = i+3, \end{cases}$$

$$\tilde{D}_{i,l} = \begin{cases} \frac{1}{32}\pi(2-\delta_{1,i}), & l = i, \\ \frac{1}{64}\pi(1-\delta_{1,i}), & l = i+1, \\ -\frac{\pi}{32}, & l = i+2, \\ -\frac{\pi}{64}, & l = i+3, \end{cases}$$

$$, \tilde{C}_{i,l} = \begin{cases} \frac{1}{128}\pi(-4\delta_{1,i} + \delta_{2,i} + 6), & l = i, \\ \frac{1}{256}\pi(-3\delta_{1,i} + \delta_{2,i} + 2), & l = i+1, \\ \frac{1}{128}\pi(\delta_{1,i} - 4), & l = i+2, \\ \frac{1}{256}\pi(\delta_{1,i} - 3), & l = i+3, \\ \frac{\pi}{128}, & l = i+4, \\ \frac{\pi}{256}, & l = i+5, \end{cases} . \tag{3.109}$$

The computation of the exact values by hand can be quite tedious. However, modern computer algebra software, capable of performing symbolic integration, can be used to find those expressions. A Mathematica notebook given in Section B.1 demonstrates this capability for the matrix $\tilde{A}$. The results have been verified by numerical arbitrary-precision integration of the actual integrals to $N = 50$. The transform of the solution y^N to the Galerkin problem eq. (3.98), with the representation

$$y^N(t) = \sum\nolimits_{l=0}^{N-2} \tilde{a}_l \tilde{\Phi}_l(t), \quad t \in I, \tag{3.110}$$

into a shifted Chebyshev series is possible, yet not necessary. We obtain the approximation of the physical solution $\hat{w}_k^N \cong y^N$ on the underlying grid as a result of a discrete shifted Chebyshev transform and a multiplication by $(1-(2r-1)^2)$, since $\sum_{l=0}^{N} \tilde{a}_l(1-(2r-1)^2)T_l^* = (1-(2r-1)^2)\sum_{l=0}^{N} \tilde{a}_l T_l^*, r \in [0,1]$.

The quality of the introduced solvers and comparisons thereof in terms of accuracy for fixed $\tilde{k}$ as well as for exemplary Poisson problems can be found in Section 3.3.4.

Let us come to the Poisson problem with Neumann boundary conditions (3.89). The approach is identical to the Dirichlet case and, hence, abbreviated. The same procedure as before turns eq. (3.89) into

$$\begin{aligned} ((\,.\,+1)z')'(t) - \frac{\tilde{k}^2}{4}(t+1)z(t) &= l(t), t \in I, \\ z'(-1) = z'(1) &= 0, \end{aligned} \tag{3.111}$$

with $z(t) = \hat{u}_k((t+1)/2)$ and $l(t) = \frac{1}{4}(t+1)\hat{f}_k((t+1)/2)$, $k \in [0, M/2]$ fixed. A variational

formulation of the latter reads

$$\int_I (t+1)z'(t)(v\omega)'(t)\,\mathrm{d}t + \tilde{k}^2 \int_I \frac{1}{4}(t+1)z(t)v(t)\omega(t)\,\mathrm{d}t = -\int_I h(t)v(t)\omega(t)\,\mathrm{d}t, \tag{3.112}$$

for $v \in Y := \{\vartheta \in H^1_\omega(I) \mid \partial^w\vartheta(-1) = \partial^w\vartheta(1) = 0 \text{ in the trace sense}\}$, where ∂^w denotes the weak derivative. We look for an approximation z^N, $N \in \mathbb{N}$, of the solution z in a finite dimensional polynomial subspace $Y^N := \{p \in P_N \mid p'(-1) = p'(1) = 0\} \subset Y$, that is spanned by the basis functions

$$\Psi_l := T_l - \gamma_l T_{l+2} \quad , l = 0, \ldots, N-2, \tag{3.113}$$

where

$$\gamma_l = \frac{l^2}{(2+l)^2}\,.$$

Inserting the basis functions together with the representations $z(t) = \sum_{l=0}^{N-2} z_l\Psi_l(t)$ and $\hat{f}_k((t+1)/2) = \sum_{j=0}^{N} \hat{f}_{kj}T_j(t)$ into eq. (3.112) entails the system of linear equations

$$(E + \tilde{k}^2 F)\cdot(z_l)_{l\le N-2} = G\cdot(\hat{f}_{kj})_{j\le N}, \tag{3.114}$$

with matrices $E, F \in \mathbb{R}^{N-2\times N-2}$ and $G \in \mathbb{R}^{N-2\times N}$, whose entries are given by

$$\begin{aligned}
E_{i,l} &= \int_{-1}^{1} (t+1)\Psi_l'(t)(\Psi_i\omega)'(t)\,\mathrm{d}t,\\
F_{i,l} &= \int_{-1}^{1} \frac{1}{4}(t+1)\Psi_l(t)\Psi_i(t)\omega(t)\,\mathrm{d}t,\\
G_{i,l} &= \int_{-1}^{1} \frac{(t+1)}{4}T_j(t)\Psi_i(t)\omega(t)\,\mathrm{d}t\,.
\end{aligned}$$

Once eq. (3.114) is solved, we find the Chebyshev series representation of the approximated solution $\hat{u}_k^N = \sum_{j=0}^{N} \hat{u}_{kj}T_j^*$ using the coefficient transform

$$\hat{u}_{kj} = z_j - \gamma_{j-2}z_{j-2} \quad , j = 0, \ldots, N, \tag{3.115}$$

with $\gamma_j = 0$ for $j = -2, -1, N-1, N$. As for the matrices of method CG_2^{Dir}, it is very tedious to find the exact values of the matrix entries. Another, third, proposition on how to obtain those entries numerically accurate is to use Chebyshev-Gauss integration. The latter quadrature rule is accurate for polynomials with a maximal degree of $2n-1$, where n is the number of nodes on which the integrands are evaluated (see eq. (3.33) in Section 3.1.2). As a matter of fact, the above expressions are weighted integrals over polynomials $(t+1)T_iT_j = \frac{1}{2}(t+1)(T_{i+j} + T_{|i-j|})$, where we used eq. (3.12). Therefore, those polynomials are of maximal degree $i+j+1$ and, for the application of the Chebyshev-Gauss quadrature, it suffices to evaluate the integrands on $n = {}^{(i+j)}/_2 + 1$ Chebyshev nodes (i.e. on Chebyshev zeros). The matrices have to be computed only once and, to make sure that the values are correctly stored and to minimize rounding errors, the matrices in the present implementation are computed in quadruple precision and rounded to working precision. Also, the programming effort reduces to a minimum, because this method can be applied to all the matrices in this section. In fact, it is used to double-check in quadruple precision if the exact values of the matrices for $CG_{1/2}^{Dir}$ are implemented correctly. Moreover, once computed for a high number of rows and columns, the matrices can be saved to disc and read in for an extraction of any inferior or equal (Chebyshev-)dimension, using the nesting property of the mesh, which is also part of the present implementation.

To sum up the described CG Poisson solver, we have the following algorithm:

Algorithm 3.1 (Chebyshev-Galerkin)**.**

- *obtain the Fourier-Chebyshev approximation of the RHS function of the problem*
- *solve the linear system that corresponds to the variational problem for every relevant Fourier mode with basis functions that fulfill the boundary conditions exactly*
- *use a coefficient transform to gather the (shifted) Chebyshev series coefficients and back-transform to physical space if needed.*

A slight **modification** of the Chebyshev-Galerkin solvers is undertaken in order to solve directly for the second vertical derivative of the periodic solution $u = u(r, z)$, resp. $w = w(r, z)$. By the help of this modification, one may use the methods from Section 3.2 in order to, subsequently, solve directly for radial derivatives and divisions. This is achieved as follows. Instead of solving for the modes $\hat{u}_k$ of the solution of eq. (3.85), we will solve directly for the terms $-\tilde{k}^2\hat{u}_k$, which are the Fourier modes of the second vertical derivative of u. This is readily achieved in dividing the solving matrix by $-\tilde{k}^2$. More precisely, for the Neumann problem we will solve, instead of eq. (3.114), namely $(E + \tilde{k}^2 F) \cdot (a_l)_l = G \cdot (\hat{f}_{kj})_{j\leq N}$,

$$Q \cdot (\bar{a}_l)_l = G \cdot (\hat{f}_{kj})_{j\leq N}, \tag{3.116}$$

where $Q = -(E+\tilde{k}^2 F)/\tilde{k}^2$ and $\bar{a}_l = -\tilde{k}^2 a_l$. Once we have solved eq. (3.116) for all $k \in [0, {}^M/_2]$, the second vertical derivative of u may be back-transformed to physical space and subtracted from the RHS function to entail

$$\mathcal{L}_1 u^M(r, z) = f^M(r, z) - \partial_z^2 u^M(r, z) =: \bar{f}^M(r, z)\,. \tag{3.117}$$

The latter equation may be accurately solved for radial derivatives and divisions by means of the methods $\mathcal{L}_1^{-1,(2)}$ and $\mathcal{L}_1^{-1,(1)/r}$, which were explained in Section 3.2. Exactly the same procedure may be applied to the Dirichlet boundary problem eq. (3.90) with the associated linear system eq. (3.114). One solves

$$\left(-\tilde{k}^{-2}(A + B - \tilde{k}^2 C)\right) \cdot (\bar{z}_l)_{l\leq N-2} = -D \cdot (\hat{g}_{kj})_j, \tag{3.118}$$

for $\bar{z}_l = -\tilde{k}^2 z_l$, and obtains the radial divisions and derivatives from

$$\mathcal{L}_2 w^M(r, z) = g^M(r, z) - \partial_z^2 w^M(r, z) =: \bar{g}^M(r, z)\,. \tag{3.119}$$

Note that the term $\tilde{k} = {}^{2\pi k}/_L$ may become very large, especially for large grid sizes and small lengths of periodicity $L > 0$, and it may quickly dominate eqs. (3.89) and (3.90). Therefore, a back-transform of the second vertical derivative to physical space before subtraction from the RHS function (also in physical space) is recommended. As a matter of fact, the condition number of the solving matrix is not altered by the division and, hence, the system is solved as accurately as for the solution itself.

Notation. The methods that divide the solving matrices by $-\tilde{k}^2$ and, subsequently, apply the methods from Section 3.2 for the treatment of eqs. (3.117) and (3.119) will be referred to by $CG_{1/2}^{Dir,(1)}$, $CG_{1/2}^{Dir,(0)/r}$, $CG^{Neu,(2)}$ and $CG^{Neu,(1)/r}$, where the superscripts indicate the boundary conditions and the use of the corresponding methods from the definitions in eq. (3.80).

In Figures 3.5 and 3.6 on page 39, a comparison between the Chebyshev-Galerkin solvers from this section and the methods for $\mathcal{L}_{1/2}^{-1}$ for the particular case $\tilde{k} = 0$ can be found. Of course, in this particular case, there is no division by $\tilde{k}^2$ in the solving matrices involved. As visualized in the referenced figures, the solving methods for $\mathcal{L}_{1/2}^{-1}$ produce the needed expressions accurately to machine precision, which implies for the above eq. (3.117), that one obtains the radial derivatives and division by r close to the accuracy of the precedingly found second vertical derivative of the solution, which is, in principle, as accurate as the solution itself, because of the unaltered condition number. A full numerical examination of these modified Chebyshev-Galerkin Poisson solvers can be found in Section 3.3.4.

3.3.3 Exact Solutions to truncated Problems

Initially, set the focus on the homogeneous Neumann Problem (3.82). In the spirit of Section 3.2, we will state the exact analytical solution for an approximated RHS and propose an algorithm that will give rise to a very precise approximation to the discrete solution. Moreover, as we have the analytical solutions at hand, we can get a formula for the second derivatives in a straight forward way and obtain a precise approximation of the latter. The resulting algorithm for this method is of complexity $O(N^2)$ for each (relevant) Fourier mode, but its precision is striking, and efficient parallelization of the code assures the applicability of the method.
Let us depart directly from the Fourier truncated problem eq. (3.89), namely $(\mathcal{L}_1 - \tilde{k}^2)\hat{u}_k(r) = \hat{u}_k''(r) + \frac{1}{r}\hat{u}_k'(r) - \tilde{k}^2\hat{u}_k(r) = \hat{f}_k(r),\ \hat{u}_k'(0) = \hat{u}_k'(1) = 0$, where $\tilde{k} = 2\pi k/L, k \in [0, M/2]$.
• **If $k = 0$**, a direct integration of the one-dimensional operator $\mathcal{L}_1 = \frac{1}{r}\partial_r(r\partial_r\,.\,)$ has been discussed in Section 3.2 and delivers

$$\hat{u}_0(r) = \ln(r)\left(\int_0^r y\hat{f}_0(y)\,\mathrm{d}y\right) - \int_0^r y\hat{f}_0(y)\ln(y)\,\mathrm{d}y \tag{3.120}$$

with the derivatives

$$\hat{u}_0'(r) = \frac{\int_0^r y\hat{f}_0(y)\,\mathrm{d}y}{r}, \quad \hat{u}_0''(r) = \hat{f}_0(r) - \frac{\int_0^r y\hat{f}_0(y)\,\mathrm{d}y}{r^2}\,. \tag{3.121}$$

For, the modes of the RHS function f are at least continuous, the derivatives stay bounded in $[0, 1]$.
• **If $k > 0$**, then eq. (3.89) is well known as the modified Bessel ODE of order 0. Its fundamental solutions are the modified Bessel functions of order 0, namely $I_0(\tilde{k}\,.\,)$ and $K_0(\tilde{k}\,.\,)$, see Section A.1 for a brief presentation of the modified Bessel functions that are used here.

Notation. In order to increase readability, for $j = 0, 1$, we will abbreviate

$$\begin{aligned}\mathcal{J}[I_j h](\tilde{k}, r) &:= \int_0^r tI_j(\tilde{k}t)h(t)\mathrm{d}t,\\ \mathcal{J}[K_j h](\tilde{k}, r) &:= \int_0^r tK_j(\tilde{k}t)h(t)\mathrm{d}t,\end{aligned} \tag{3.122}$$

for a generic function h. If $h = h(x)$ is a power function with exponent n, then we will write x^n in place of h, i.e. $\mathcal{J}[I_j x^n]$ and $\mathcal{J}[K_j x^n]$.

By an application of the method of variation of parameters (see e.g. [72]), under consideration of the homogeneous boundary conditions, one obtains the complete analytical solution

to problem (3.82):

$$\hat{u}_k(r) = I_0(\tilde{k}r) \cdot \mathcal{J}[K_0 \hat{f}_k](\tilde{k}, r) - K_0(\tilde{k}r) \cdot \mathcal{J}[I_0 \hat{f}_k](\tilde{k}, r) + c_k I_0(\tilde{k}r)\,. \tag{3.123}$$

The solution can be verified by a direct insertion into the differential equation, which is done in the proof of Lemma A.1 in Section A.1. From there, we read the exact derivatives

$$\hat{u}_k'(r) = \tilde{k} I_1(\tilde{k}r) \Big(\mathcal{J}[K_0 \hat{f}_k](\tilde{k}, r) + c_k \Big) + \tilde{k} K_1(kr) \mathcal{J}[I_0 \hat{f}_k](\tilde{k}, r), \tag{3.124}$$

$$\begin{aligned} \hat{u}_k''(r) = \frac{1}{2}\tilde{k}\Big(&- \tilde{k}(K_0(\tilde{k}r) + K_2(\tilde{k}r)) \mathcal{J}[I_0 \hat{f}_k](\tilde{k}, r) + \tilde{k}\big(I_0(\tilde{k}r) + I_2(\tilde{k}r)\big) \Big(\mathcal{J}[K_0 \hat{f}_k](\tilde{k}, r) + c_k \Big) \\ &+ 2r \hat{f}_k(r) \Big(I_0(\tilde{k}r) K_1(\tilde{k}r) + I_1(\tilde{k}r) K_0(\tilde{k}r) \Big) \Big)\,. \end{aligned} \tag{3.125}$$

Using eq. (3.124), one easily shows that

$$c_k = -\mathcal{J}[K_0 \hat{f}_k](\tilde{k}, 1) + \frac{K_1(\tilde{k})}{I_1(\tilde{k})} \cdot \mathcal{J}[I_0 \hat{f}_k](\tilde{k}, 1)$$

ensures the Neumann homogeneity. Subsequently, we approximate the Fourier mode $\hat{f}_k$ by a truncated shifted Chebyshev series,

$$\hat{f}_k(r) = \sum_{l=0}^{N} \hat{f}_{kl} T_l^*(r), \tag{3.126}$$

and insert it into the precedingly found formulæ. Since the expressions for the solution, and derivatives thereof, are linear in $\hat{f}_k$, we may pull out the sum and conveniently write

$$\hat{u}_k(r) = \sum_{l=0}^{N} \hat{f}_{kl}\, \mathcal{N}^{(0)}(\tilde{k}, l, r), \tag{3.127}$$

where

$$\mathcal{N}^{(0)}(\tilde{k}, l, r) \coloneqq I_0(\tilde{k}r) \cdot \mathcal{J}[K_0 T_l^*](\tilde{k}, r) - K_0(\tilde{k}r) \cdot \mathcal{J}[I_0 T_l^*](\tilde{k}, r) + c_k^* I_0(\tilde{k}r), \tag{3.128}$$

with

$$c_k^* = -\mathcal{J}[K_0 T_l^*](\tilde{k}, 1) + \frac{K_1(\tilde{k})}{I_1(\tilde{k})} \cdot \mathcal{J}[I_0 T_l^*](\tilde{k}, 1)\,.$$

Similarly, we find expressions for the derivatives, namely

$$\hat{u}_k'(r) = \sum_{l=0}^{N} \hat{f}_{kl}\, \mathcal{N}^{(1)}(\tilde{k}, l, r), \tag{3.129}$$

$$\hat{u}_k''(r) = \sum_{l=0}^{N} \hat{f}_{kl}\, \mathcal{N}^{(2)}(\tilde{k}, l, r), \tag{3.130}$$

with

$$\mathcal{N}^{(1)}(\tilde{k}, l, r) \coloneqq \tilde{k} \Big(I_1(\tilde{k}r) \cdot \mathcal{J}[K_0 T_l^*](\tilde{k}, r) - K_1(\tilde{k}r) \cdot \mathcal{J}[I_0 T_l^*](\tilde{k}, r) + c_k^* I_1(\tilde{k}r) \Big) \tag{3.131}$$

$$\begin{aligned} \mathcal{N}^{(2)}(\tilde{k}, l, r) \coloneqq \frac{1}{2}\tilde{k}\Big(&- \tilde{k}(K_0(\tilde{k}r) + K_2(\tilde{k}r)) \mathcal{J}[I_0 T_l^*](\tilde{k}, r) \\ &+ \tilde{k}\big(I_0(\tilde{k}r) + I_2(\tilde{k}r)\big) \Big(\mathcal{J}[K_0 T_l^*](\tilde{k}, r) + c_k^* \Big) \end{aligned}$$

$$+ 2rT_l^*(r)\left(I_0(\tilde{k}r)K_1(\tilde{k}r) + I_1(\tilde{k}r)K_0(\tilde{k}r)\right)\Big)\,. \tag{3.132}$$

To obtain the expression $\hat{u}_k'(r)/r$, eq. (3.131) and eq. (3.129) are directly divided by r. We extend the definitions of the $\mathcal{N}^{(j)}$ by the $k = 0$ terms using eqs. (3.120) and (3.121):

$$\begin{aligned}
\mathcal{N}^{(0)}(0,l,r) &:= \ln(r)\left(\int_0^r yT_l^*(y)\,\mathrm{d}y\right) - \int_0^r yT_l^*(y)\ln(y)\,\mathrm{d}y \\
\mathcal{N}^{(1)}(0,l,r) &:= \frac{\int_0^r yT_l^*(y)\,\mathrm{d}y}{r} \\
\mathcal{N}^{(2)}(0,l,r) &:= T_l^* - \frac{\int_0^r yT_l^*(y)\,\mathrm{d}y}{r^2}\,.
\end{aligned}$$

For the first derivative divided by r we define furthermore

$$\mathcal{N}^{(1)/r} := \frac{1}{r}\mathcal{N}^{(1)}\,. \tag{3.133}$$

Hence, given a Chebyshev-Fourier approximation of the RHS function in eq. (3.82) in the domain $[0,1]\times[0,L]$, with modes $\hat{f}_{kl}$, the derivatives of the solution $u^{MN} = u^{MN}(r,z)$ to the spectrally truncated Neumann problem take the form

$$\partial_r^j u^{MN}(r,z) = \sum_{|k|=0}^{M}\sum_{l=0}^{N} \hat{f}_{kl}\,\mathcal{N}^{(j)}(\tilde{k},l,r)\,\mathrm{e}^{\mathrm{i}\tilde{k}z}, \quad j = 0,1,2\,. \tag{3.134}$$

Equally, the first derivative divided by r is given by

$$\frac{1}{r}\partial_r u^{MN}(r,z) = \sum_{|k|=0}^{M}\sum_{l=0}^{N} \hat{f}_{kl}\,\mathcal{N}^{(1)/r}(\tilde{k},l,r)\,\mathrm{e}^{\mathrm{i}\tilde{k}z}\,. \tag{3.135}$$

Thus, the Fourier modes of $\partial_r^2 u_k^{MN}(r,z)$ and $\partial_r u_k^{MN}(r,z)/r$ can be obtained on the shifted Chebyshev grid via matrix-vector multiplications in a similar way as described in Section 3.2. Those multiplications entail the Fourier modes on the shifted Chebyshev grid and one simply needs to apply an inverse FFT to obtain the needed expressions on the Chebyshev-Fourier grid.

Notation. These particular methods, which involve matrix-vector multiplications to obtain the Fourier modes of $\partial_r^2 u_k^{MN}(r,z)$ and $\partial_r u_k^{MN}(r,z)/r$, will be referred to as $\mathcal{N}^{(2)}$ and $\mathcal{N}^{(1)/r}$ respectively. Other expressions do not need to be computed in the current implementation of the CLA.

Let us come to the Dirichlet problem (3.83), for which we proceed in the same manner as for the Neumann problem before. We recall the truncated problem eq. (3.90), for all $k \in [0, M/2]$: $(\mathcal{L}_2 - \tilde{k}^2)\hat{w}_k(r) = \hat{w}_k''(r) + \frac{1}{r}\hat{w}_k'(r) - (\frac{1}{r^2} + \tilde{k}^2)\hat{w}_k(r) = \hat{g}_k(r)$, $\hat{w}_k(0) = \hat{w}_k(1) = 0$, where we further assume that $\hat{g}_k(r)$ is approximated by a truncated Chebyshev series $\sum_{l=0}^{N}\hat{g}_{kl}T_l^*(r)$.

• **For $k = 0$**, the operator $\mathcal{L}_2 = \partial_r\left(\frac{1}{r}\partial_r(r\,.\,)\right)$ integrates exactly, as already seen in eq. (3.73), to give

$$
\begin{aligned}
\hat{w}_0(r) &= \frac{\int_0^r t\left(c + \int_0^t \hat{g}_0(x)\,dx\right)\,dt}{r} \\
&= \sum_{l=0}^{N} \hat{g}_{0l} \frac{\int_0^r t\left(c_l + \int_0^t T_l^*(x)\,dx\right)\,dt}{r} \\
&=: \sum_{l=0}^{N} \hat{g}_{kl} \mathcal{D}^{(0)}(0,l,r), \\
\partial_r \hat{w}_0(r) &= \sum_{l=0}^{N} \hat{g}_{0l} \left(-\frac{\int_0^r t\left(c_l + \int_0^t T_l^*(x)\,dx\right)\,dt}{r^2} + c_l + \int_0^r T_l^*(x)\,dx \right) \\
&=: \sum_{l=0}^{N} \hat{g}_{kl} \mathcal{D}^{(1)}(0,l,r), \\
\partial_r^2 \hat{w}_0(r) &= \sum_{l=0}^{N} \hat{g}_{0l} \frac{1}{r} \left(\frac{2\int_0^r t\left(c_l + \int_0^t T_l^*(x)\,dx\right)\,dt}{r^2} - c_l - \int_0^r T_l^*(x)\,dx \right) + T_l^*(r) \\
&=: \sum_{l=0}^{N} \hat{g}_{kl} \mathcal{D}^{(2)}(0,l,r),
\end{aligned}
$$

where

$$
c = -2\int_0^1 t \int_0^t \hat{g}_0(x)\,dx\,dt \quad \text{and} \quad c_l = -2\int_0^1 t \int_0^t T_l^*(x)\,dx\,dt\,.
$$

• **For $k > 0$**, the differential eq. (3.90) is the modified Bessel ODE of first kind, whose fundamental solutions are I_1 and K_1, see Section A.1. The particular solution to the homogeneous Dirichlet problem eq. (3.90) is given by

$$
\hat{w}_k(r) = I_1(\tilde{k}r)\left(\mathcal{J}[K_1\hat{g}_k](\tilde{k},r) + d_k\right) - K_1(\tilde{k}r)\mathcal{J}[I_1\hat{g}_k](\tilde{k},r), \tag{3.136}
$$

with

$$
d_k = -\mathcal{J}[K_1\hat{g}_k](\tilde{k},1) + \frac{K_1(\tilde{k})}{I_1(\tilde{k})} \cdot \mathcal{J}[I_1\hat{g}_k](\tilde{k},1)
$$

to enforce homogeneity at $r = 1$. An insertion of the truncated Chebyshev representation of $\hat{g}_k(r)$ and subsequent simplifications, which exploit the linearity of the involved integrals, give

$$
\begin{aligned}
\hat{w}_k(r) &= \sum_{l=0}^{N} \hat{g}_{kl}\big(I_1(\tilde{k}r)\left(\mathcal{J}[K_1T_l^*](\tilde{k},r) + d_k^*\right) - K_1(\tilde{k}r)\mathcal{J}[I_1T_l^*](\tilde{k},r)\big) \\
&=: \sum_{l=0}^{N} \hat{g}_{kl}\, \mathcal{D}^{(0)}(\tilde{k},l,r),
\end{aligned} \tag{3.137}
$$

with

$$
d_k^* = -\mathcal{J}[K_1T_l^*](\tilde{k},1) + \frac{K_1(\tilde{k})}{I_1(\tilde{k})} \cdot \mathcal{J}[I_1T_l^*](\tilde{k},1)\,.
$$

The same procedure applied to the derivatives of the analytic solution produces

$$\begin{aligned}
\partial_r \hat{w}_k(r) &= \sum_{l=0}^{N} \hat{g}_{kl} \Big(\tfrac{\tilde{k}}{2} \big((I_0(\tilde{k}r) + I_2(\tilde{k}r)) \Big(\mathcal{J}[K_1 T_l^*](\tilde{k}, r) + d_k \Big) \\
&\qquad + (K_0(\tilde{k}r) + K_2(\tilde{k}r)) \mathcal{J}[I_1 T_l^*](\tilde{k}, r) \big) \Big) \\
&=: \sum_{l=0}^{N} \hat{g}_{kl}\, \mathcal{D}^{(1)}(\tilde{k}, l, r) \\
\partial_r^2 \hat{w}_k(r) &= \sum_{l=0}^{N} \hat{g}_{kl} \tfrac{\tilde{k}}{4} \Big[- \tilde{k}(3K_1(\tilde{k}r) + K_3(\tilde{k}r)) \mathcal{J}[I_1 T_l^*](\tilde{k}, r) \\
&\qquad + \tilde{k}(3I_1(\tilde{k}r) + I_3(\tilde{k}r)) \Big(\mathcal{J}[K_1 T_l^*](\tilde{k}, r) + d_k^* \Big) \\
&\qquad + 2r T_l^*(r) \big(I_1(\tilde{k}r) K_0(\tilde{k}r) + I_1(\tilde{k}r) K_2(\tilde{k}r) \\
&\qquad + I_0(\tilde{k}r) K_1(\tilde{k}r) + I_2(\tilde{k}r) K_1(\tilde{k}r) \big) \Big] \\
&=: \sum_{l=0}^{N} \hat{g}_{kl}\, \mathcal{D}^{(2)}(\tilde{k}, l, r)\,.
\end{aligned}$$

To obtain the expression $\hat{w}_k(r)/r$, eq. (3.137) is directly divided by r, and we may define

$$\mathcal{D}^{(0)/r} := \frac{1}{r} \mathcal{D}^{(0)}\,. \tag{3.138}$$

We find the representation of the exact derivatives (and division by r) of the solution to the spectrally discretized version of eq. (3.83), $w^{MN} = w^{MN}(r, z)$, in the form

$$\partial_r^j w^{MN}(r, z) = \sum_{|k|=0}^{M} \sum_{l=0}^{N} \hat{g}_{kl}\, \mathcal{D}^{(j)}(\tilde{k}, l, r)\, \mathrm{e}^{\mathrm{i}\tilde{k}z}, \quad j = 0, 1, 2, \tag{3.139}$$

where $\mathcal{D}^{(0)/r}$ replaces $\mathcal{D}^{(j)}$ to obtain $w^{MN}(r, z)/r$. These expressions are obtained on the Chebyshev-Fourier grid via matrix-vector multiplications as in the Neumann case.

Notation. We will refer to the particular methods of matrix-vector multiplication to obtain $\partial_r w^{MN}(r, z)$ and $w^{MN}(r, z)/r$ as $\mathcal{D}^{(1)}$ and $\mathcal{D}^{(0)/r}$ respectively. Other expressions do not need to be computed in the current implementation of the CLA.

Computation of the solving matrices Above we have stated the exact solutions and their derivatives, eqs. (3.134) and (3.139), to the truncated problems eqs. (3.89) and (3.90) respectively. In order to calculate them numerically, the exact values of the expressions $\mathcal{N}^{(\,.\,)}$ and $\mathcal{D}^{(\,.\,)}$ are needed, where $(\,.\,)$ can be any of $(0), (1), (2), (1)/r$ and $(0)/r$ as defined above. Given a discrete set of points $\{r_0, \ldots, r_N\} \subset [0, 1]$, we aim at the computation of the three-dimensional cubic matrices $N^{(\,.\,)}$ and $D^{(\,.\,)}$ with entries

$$\begin{aligned}
N_{kli}^{(\,.\,)} &:= \mathcal{N}^{(\,.\,)}(\tfrac{2\pi}{L} k, l, r_i), \\
D_{kli}^{(\,.\,)} &:= \mathcal{D}^{(\,.\,)}(\tfrac{2\pi}{L} k, l, r_i),
\end{aligned} \tag{3.140}$$

for $k = 0, \ldots, M/2$ and $l, i = 0, \ldots, N$, where M is the Fourier dimension, and $N + 1$ is the Chebyshev dimension. However, not all the matrices have to be computed in the CLA, since only certain derivatives are needed. Also, inferior orders of differentiation may be obtained

by integration of a higher derivative, provided additional values are known. For instance, one could integrate the second derivative of the Neumann solution to obtain the first derivative, knowing that the latter must vanish at $r = 0$ to keep the operator $\mathcal{L}_1$ bounded. Once the matrices $N^{(\cdot)}$ and $D^{(\cdot)}$ are computed, one may save them to disc and reuse them any number of times. We will see in the following that the numerical computation of the above entries is by no means straightforward or low in complexity, and a non-negligible effort has been employed to obtain them. The needed matrices have been computed for a moderately high number of Chebyshev-Fourier modes on the shifted Chebyshev-extrema grid $\{r_i\}_{0\leq i\leq N}$, which possesses the nesting property mentioned in Section 3.1.2. Since we use N of the form $N = 2^n,\ n \in \mathbb{N}$, the mentioned nesting property enables us to use the matrices for $M/2 = N = 512$ modes (513^3 entries) to solve on all physical Chebyshev-Fourier sub-grids with sizes $(2^p + 1) \times 2^q$, where $p \leq 9,\ q \leq 10$. That is, with the saved cubic matrices $N^{(\cdot)}$ and $D^{(\cdot)}$ of size 513^3 we can achieve grid sizes of up to 513×1024 points in physical space. Note that the odd numbers for the Chebyshev dimension ($N + 1 = 2^n + 1$) have been chosen deliberately, because the underlying fast Chebyshev transform on the Chebyshev-extrema grid, namely the DCT-I in FFTW3, shows best performance for these numbers (see [49, 48] for details on the FFTW3). Indeed, personal tests with the mentioned DCT-I have shown a strong increase in speed if $2^n + 1$ shifted Chebyshev extrema were used instead of 2^n.
The most involved part of the matrix computations is the evaluation of the integrals in (3.122). The modified Bessel functions expose such an extreme behaviour at $r = 0$ and $r = 1$, that, for moderate grid sizes, any standard quadrature formula in double or quadruple precision is not applicable to find the values in the precision needed. The functions K_j, $j = 0, 1, \ldots$ possess (logarithmic) singularities at the pole $r = 0$ and I_j, $j = 0, 1, \ldots$ grow exponentially for large arguments ($\tilde{k}r$), see eqs. (A.7) to (A.11) in Section A.1 for the exact asymptotic terms and a visualization of these functions. Also, the mentioned integrals need to be evaluated to a very large number of digits, because the summands in the exact analytical solutions and their derivatives show violent cancelations. In order to be sure to calculate all entries correctly to more than 36 digits, a precision of roughly $\frac{4}{5}\frac{2\pi}{L}k$ digits is needed,[1] this being especially costly for small heights L or large k. However, the cancellations do dependent on the actual values of $\frac{2\pi}{L}k, l$ and r so that the working precision may be adapted. The amount of needed computational power and time seems to be in no proportion to the seemingly standard goal of inverting a Calderon-Zygmund operator of order zero, namely double derivatives on an inverse Laplacian. For the purpose of very accurate computations of incompressible Euler flow, with a possible goal of detecting finite-time singularities, however, this one-time effort might be appropriate.
Because of steep singularities at $r = 0$ and exponential growth at $r = 1$, the integrals in (3.122) are not gentle to standard quadrature schemes and we will thus proceed in mathematical theory, which will allow for a direct evaluation of the latter integrals in form of Hypergeometrical and Meijer-G functions.

Lemma 3.1. The following holds for $\tilde{k} = \frac{2\pi}{L}k$, $n \in \mathbb{N}_0$ and $r \in [0, 1]$,

$$\begin{aligned}\mathcal{J}[I_0 x^n](\tilde{k}, r) &= \int_0^r I_0(\tilde{k}t)t^{n+1}\mathrm{d}t \\ &= \frac{r^{n+2}\,{}_1F_2\left(\frac{n}{2}+1; 1, \frac{n}{2}+2; \frac{\tilde{k}^2r^2}{4}\right)}{n+2},\end{aligned} \tag{3.141}$$

[1] The given estimate was found experimentally.

$$\begin{aligned}
\mathcal{J}[K_0 x^n](\tilde{k}, r) &= \int_0^r K_0(\tilde{k}t) t^{n+1} \mathrm{d}t \\
&= \frac{r^{n+2}}{(n+2)^2} \left[(n+2) K_0(\tilde{k}r)\, {}_1F_2\left(1; \frac{n}{2}+1, \frac{n}{2}+2; \frac{\tilde{k}^2 r^2}{4}\right) \right. \\
&\quad \left. + \tilde{k} r K_1(\tilde{k}r)\, {}_1F_2\left(1; \frac{n}{2}+2, \frac{n}{2}+2; \frac{\tilde{k}^2 r^2}{4}\right) \right], \qquad (3.142) \\
\mathcal{J}[I_1 x^n](\tilde{k}, r) &= \int_0^r I_1(\tilde{k}t) t^{n+1} \mathrm{d}t \\
&= \left[\frac{t^{n+1}}{\tilde{k}} I_0(\tilde{k}t) \right]_{t=0}^{t=r} - (n+1) \mathcal{J}[I_0 x^n](\tilde{k}, r) \qquad (3.143) \\
&= \frac{1}{4} \tilde{k} r^{n+3} \frac{\Gamma\left(\frac{n+3}{2}\right)}{\Gamma\left(\frac{n+5}{2}\right)}\, {}_1F_2\left(\frac{n+3}{2}; 2, \frac{n+5}{2}; \frac{\tilde{k}^2 r^2}{4}\right), \qquad (3.144) \\
\mathcal{J}[K_1 x^n](\tilde{k}, r) &= \int_0^r K_1(\tilde{k}t) t^{j+1} \mathrm{d}t \\
&= \frac{1}{4} r^{n+2} G^{2,1}_{1,3}\left(\frac{\tilde{k}r}{2}, \frac{1}{2}; \begin{matrix} -\frac{n}{2} \\ -\frac{1}{2}, \frac{1}{2}, -\frac{n}{2}-1 \end{matrix} \right) \qquad (3.145) \\
&\overset{(n>0)}{=} \left[-\frac{t^{n+1}}{\tilde{k}} K_0(\tilde{k}t) \right]_{t=0}^{t=r} + (n+1) \mathcal{J}[K_0 x^n](\tilde{k}, r), \qquad (3.146) \\
\mathcal{J}[K_1 x^0](\tilde{k}, r) &= \frac{\pi r (L_0(kr) K_1(kr) + L_1(kr) K_0(kr))}{2k}, \qquad (3.147)
\end{aligned}$$

where ${}_1F_2$ is a generalized hypergeometric function, G the Meijer-G function and L the Struve-L function, see [92] for a comprehensive introduction to these special functions.

Proof. The identities eqs. (3.143) and (3.146) are obtained by partial integration. Equation (3.147) is obtained from eq. (6.561-4) on page 676 in [68] by setting $a = kr$ and a change of variables. Equations (3.141) and (3.144) are adapted from eq. (84) on page 198 in [7] again via a change of variables. Equations (3.142) and (3.145) originate from eq. (6.592-2) on page 690 in [68] and references therein, see also [90, page 365]. These identities have additionally been verified by Mathematica's symbolic integration and via its arbitrary precision integration and evaluation of the involved special functions for arbitrarily chosen values of k, r and n. □

We may now use the explicit polynomial expression (3.21) of a shifted Chebyshev polynomial $T_l^*(r) = T_l(2r-1) = \sum_{j=0}^{l} a_{j,l}\, r^j$, where

$$a_{j,l} = \begin{cases} 1, & l = 0, j = 0, \\ \frac{4^j l (-1)^{l-j} (j+l-1)!}{(2j)!(l-j)!}, & l > 0,\ j = 0, \ldots, l, \end{cases}$$

and find the explicit formulas for the integrals in (3.122) by inserting I_0, I_1, K_0 and K_1 into

$$\mathcal{J}[\,.\,T_l^*](\tilde{k}, r) = \sum_{j=0}^{l} a_{j,l}\, \mathcal{J}[\,.\,x^j](\tilde{k}, r)\,. \qquad (3.148)$$

The cubic matrices $N^{(\cdot)}$ and $D^{(\cdot)}$ from eq. (3.140) can then be obtained as a sum of special functions which are evaluated to a high enough precision to provide for the range, which the

cancellations need.

Algorithm 3.2 (Bessel-Chebyshev)**.**

- *For a fixed height $L > 0$ and Chebyshev-, Fourier dimension $M, N+1$ respectively, compute the cubic matrices $N^{(\cdot)}$, $D^{(\cdot)}$ for the methods $\mathcal{N}^{(\cdot)}$, $\mathcal{D}^{(\cdot)}$.*
- *For all $k \in [0, M/2]$, multiply the k-th slice of the cubic matrices by the vector of Chebyshev coefficients of the k-th mode of the RHS function to obtain the corresponding expression in physical space.*

The evaluation of the involved special functions is very costly and can only be achieved by modern arbitrary precision packages, which have the needed functions implemented. An efficient parallization is also mandatory for moderate to high resolutions. To reduce the computational amount, eqs. (3.143) and (3.146) are implemented and reuse the entries of $\mathcal{J}[I_0 x^n](\tilde{k}, r)$ and $\mathcal{J}[K_0 x^n](\tilde{k}, r)$, $n \geq 1$. Currently, eq. (3.145) is implemented for $\mathcal{J}[K_0 x^n](\tilde{k}, r)$, $n = 0$, but eq. (3.147) is faster and the involved Struve-L functions are easier to find as components of arbitrary precision libraries. The Library that is used to compute the values of the solving matrices $N^{(\cdot)}$ and $D^{(\cdot)}$ is *ARBLIB* [69]. This C library uses ball arithmetic and has a large number of special functions implemented. The author of this thesis has written a computer program in the C++ Language to compute the correct matrices in sufficiently high precision, such that the results can be saved to disc correctly rounded to quadruple precision for continuous re-utilization. The code is quite long, since the execution of simple arithmetic operations with this library requires complex function calls. Moreover, the code is efficiently parallelized by a hybrid parallization that uses both OpenMP and MPI protocols. The vast amount of digits that is needed for the cancellations, however, still demands for very long computation times even when 400-600 computational cores are employed and only moderate resolutions are sought.

Remark. By the virtue of lemma 3.1, the presented methods for the computation of the analytical solutions to truncated problems may also be applied to other polynomial basis functions. Instead of Chebyshev polynomials, one may use Lagrange, Laguerre, Jacobi, or any other useful polynomials in the approximation of the RHS function. One only needs to replace the basis functions in eq. (3.148) together with the corresponding polynomial coefficients $a_{j,l}$. Also, the choice of evaluation points may be deliberately chosen and may differ from the grid points that are related to the basis functions. The CLA code uses Chebyshev polynomials, mainly because of their connection to Fourier transforms that are computed by the FFT algorithm.

3.3.4 Numerical Verification of the Poisson Solvers

Here, we will study the quality of the Poisson solvers from Sections 3.3.2 and 3.3.3. Some tests of the methods applied to the ordinary differential eqs. (3.89) and (3.90) with different values for $\tilde{k}$ are executed, before we take a look at the 2-dimensional Poisson problems (3.85) and (3.86). The tests are performed on the earlier introduced test functions

$$s(r) = e^{1-\cos 2\pi r} - 1, \tag{3.149}$$

$$t(r, z) = s(r)\, \mathrm{e}^{\sin(2\pi z/L)}, \tag{3.150}$$

for $r \in [0, 1]$ and $z \in [0, L]$. The RHS functions from eqs. (3.82), (3.83), (3.89) and (3.90), as well as the derivatives of s and t, are computed analytically and implemented in quadruple precision to obtain accurate function implementations in double precision. The reason for the

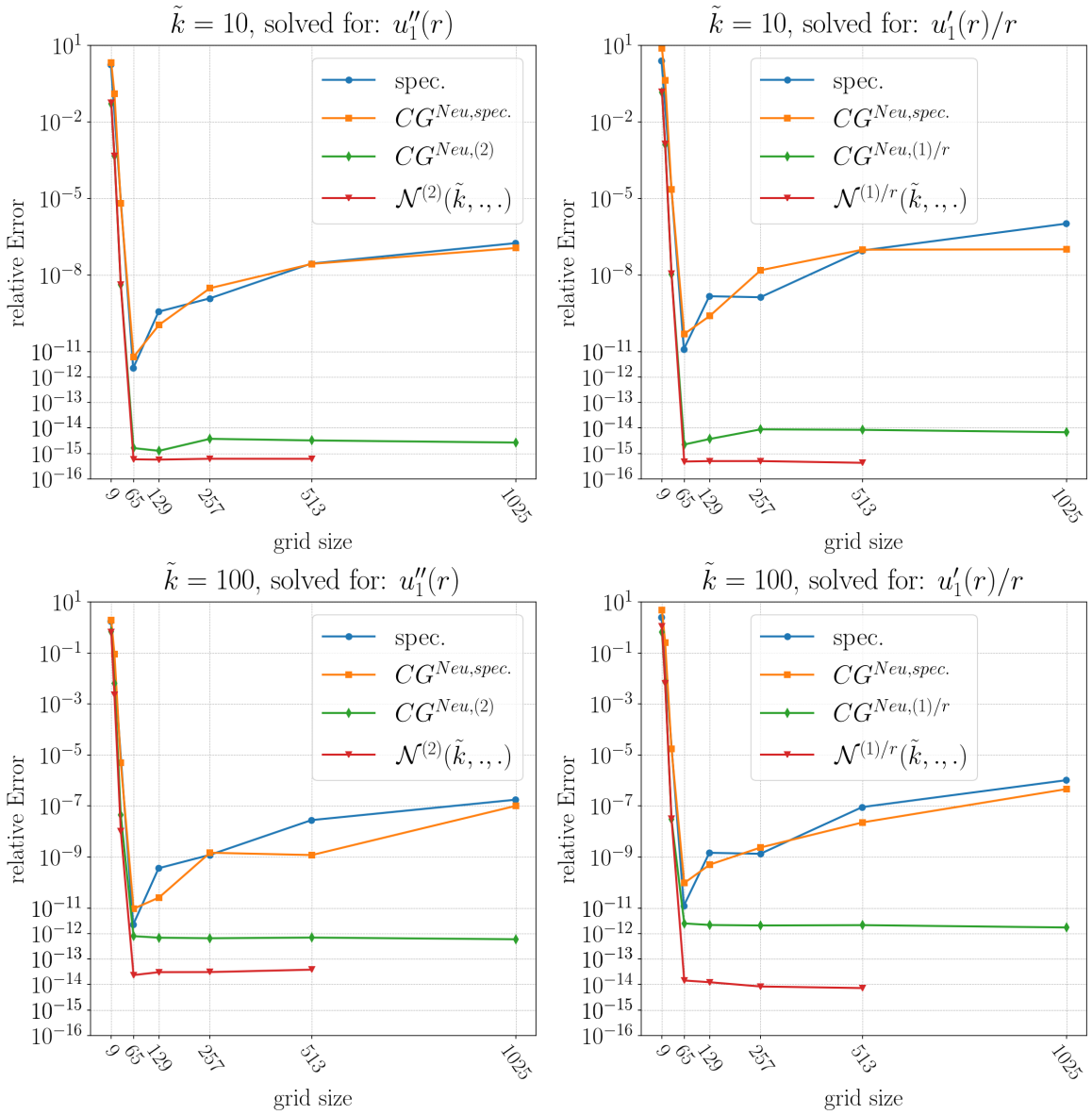


Figure 3.7: Relative errors in the double derivative of the approximated solution to eq. (3.85) and derivative divided by r for two different values of $\tilde{k}$. The accuracy for both expressions obtained by $CG^{Neu,(2)}$ and $CG^{Neu,(1)/r}$ respectively plateau at a certain level due to their origin from $-\tilde{k}^2 u_1$ that was solved for and subtracted from the RHS function. The Bessel-Chebyshev method is sill more accurate by one to two orders of magnitude, but currently restricted to 513 shifted Chebyshev grid points.

quadruple precision evaluation is the presence of the terms ∂_r/r and $1/r^2$ in the operators $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively. One may of course use different test functions such as the above functions multiplied by r^2 to avoid any division by r inside of the program, but it was found that for such functions the explicit division by r in the Fortran implementation as a method is accurate to machine precision even for high point densities. Therefore, the argumentation against explicit division stands on loose ground, see Table 3.1 for a comparison of the explicit division to the spectral division for our test function. The limits at $r = 0$ that appear in the operators themselves and in other expressions were implemented exactly.
The labels $CG^{(\cdot)}$ represent the methods that were introduced in Section 3.3.2 (see the notation paragraphs). They are Chebyshev-Galerkin methods, which use basis functions that fulfill the boundary conditions exactly, and whose solving matrices are divided by the term

$-\tilde{k}^2$ in order to solve for the second z-derivative instead of the solution. Once all the modes have been solved, the second z-derivative is transformed to physical space and subtracted from the RHS function as explained in the end of Section 3.3.2. The remaining equations after the subtraction are solved via the methods from Section 3.2 directly for their radial derivatives and divisions y r. If the CG symbol has a *spec.* in its superscript, then the CG method is used only to solve for the solution and the plotted expressions are found via spectral differentiation and division methods as given in Section 3.1.3. The methods $\mathcal{N}^{(\cdot)}$ and $\mathcal{D}^{(\cdot)}$ from Section 3.3.2 are directly used as labels in the plots, those methods are evaluations of the exact derivatives of the solutions to eqs. (3.89) and (3.90) via matrix-vector multiplication with the series coefficients of the RHS function. The label *spec.* refers to the direct application of the differentiation and division methods from Section 3.1.3 to the spectral projection of the function s or t.
Note that the graphs for the Bessel-Chebyshev methods $\mathcal{N}^{(\cdot)}$ and $\mathcal{D}^{(\cdot)}$ are only displayed until a grid size of 513×1024 points. This is because the solving matrices for those methods were only computed to 513 shifted Chebyshev points and 513 Fourier modes $(0, \ldots, 512)$ to account for a total grid size of 513×1024 points. In fact, the case where $L = 2\pi$ poses no problem for higher grid size computations, the value $L = 1/6$ in contrary, used in [83] for singularity detection, poses serious difficulties in the matrix computations. As a matter of fact, $\tilde{k}^2 = (2\pi k/L)^2$ grows quickly to very large numbers for small L and the extreme behavior of the modified Bessel functions in the exact solutions display violent cancellations that demand for very high precision computing to account for their compensation. Generally speaking, the higher the wave number, the more digits are used to produce a quadruple precision output and the longer it takes to compute the corresponding slice of the three-dimensional matrix. However, the effort might be worth it, if accuracy is a priority.

Figure 3.7 shows the convergence and stability properties of the used methods for $\tilde{k} = 10,100$ in the Neumann boundary value problem (3.89) for $u_1(r) = s(r)$. The plot can be understood as the case where the mode $\hat{u}_{10}(r)$ (resp. $\hat{u}_{100}(r)$) equals the test function s, where $L = 2\pi$. The Bessel-Chebyshev (BC) method is the most accurate among the tested methods, far more accurate than a direct differentiation via coefficient transforms in the shifted Chebyshev spectral space. Both methods, CG and BC, display stability for increasing grid sizes in forming a plateau close to their respective error minima. Note that these tests were performed with various other test functions with similar results. If the test function sits on a higher mode, then the plateaus are still formed, but slightly elevated, this is due to the increasing influence of the term $\tilde{k}^2$ in the condition numbers of the solving matrices. Also, the radial expressions in the CG methods are obtained from the RHS functions minus the second vertical derivative of the solution, which contain inaccuracies. The same behavior can be observed for the methods that are used to solve the Dirichlet boundary value problem (3.90) for the needed radial derivatives and divisions. See Figure 3.8 for a numerical verification of those methods, again for two values of $\tilde{k}$, one that corresponds to the mode $\hat{w}_{500}(r)$ where $L = 2\pi$, and one for the same mode but with $L = 1/6$. This case, however, is rather extreme, but good for testing purposes. The test visualized in Figure 3.8 shows that the method CG_1^{Dir} should be preferred over CG_2^{Dir} when it comes to accuracy of the outcome. An application of the radial spectral differentiation methods from Section 3.1.3 on the numerical solutions of the Chebyshev-Galerkin schemes should be avoided completely due to their unstable behavior for increasing grid sizes.

In the sequel of this section we investigate the model Poisson problems eqs. (3.82) and (3.83) by inserting the test function $t = t(r, z)$. See the upper panels of Figure 3.9 for the first derivative of the solution to the Neumann problem (3.82) divided by r for both lengths of periodicity which are treated in this work ($L = 2\pi$ and $L = 1/6$). The lower panels show the

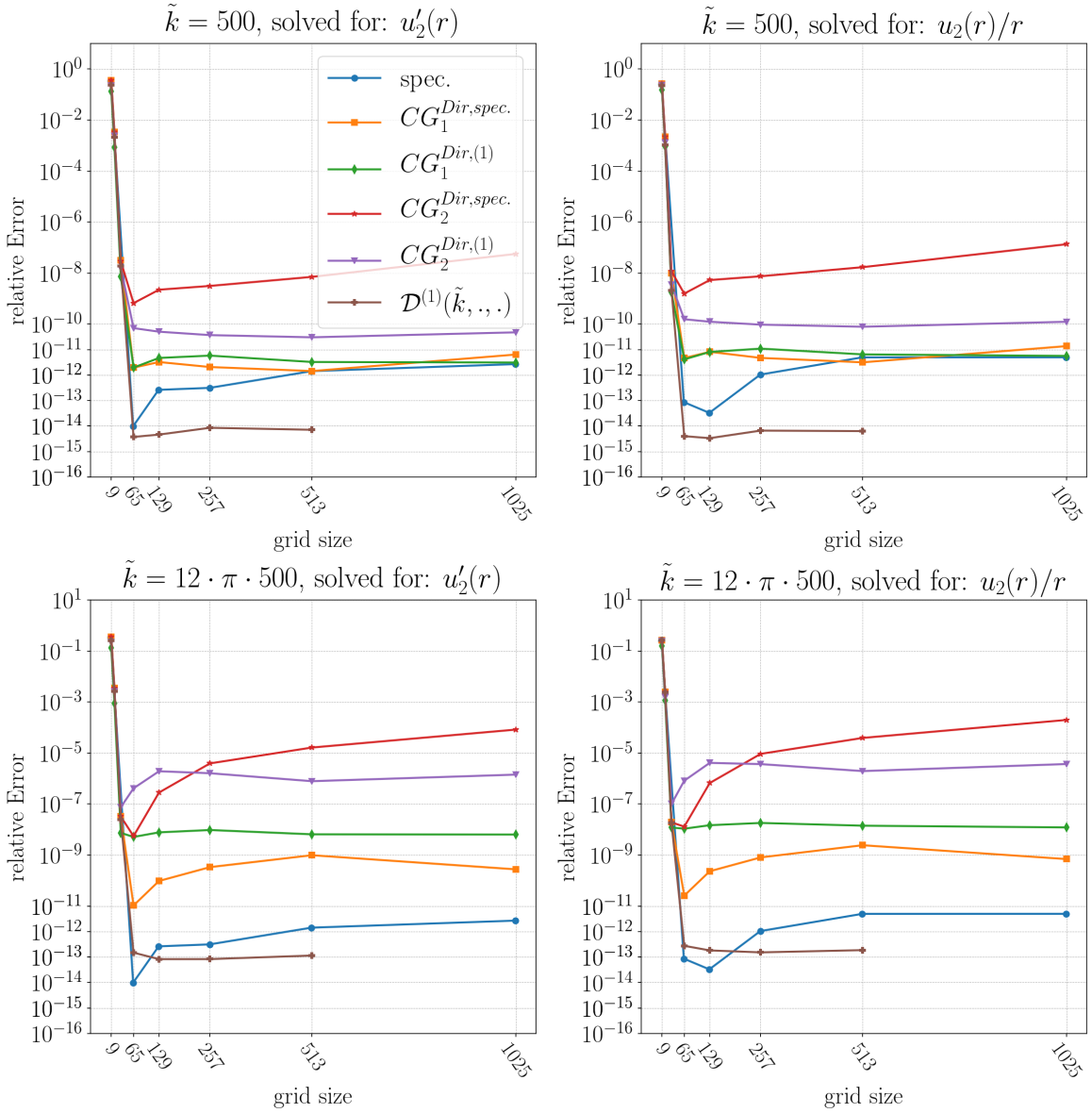


Figure 3.8: Relative errors for the expressions in the title that are obtained by the methods in the legend (upper left), which is valid for all the panels. The plots represent the case where the test function $u_2(r) = s(r)$ sits on the 500^{th} mode for $L = 2\pi$ (upper panels) and $L = 1/6$ (lower panels). The graphs labeled by *spec.* are independent of $\tilde{k}$, the corresponding expressions were directly obtained by spectral differentiation and division methods from the known function. The latter is not used in the program, here it is only plotted to show the strength of the method $\mathcal{D}^{(\cdot)}$. Even if the solution is known, it is still better to solve for the plotted expressions by the presented solving methods instead of using standard spectral methods.

relative error for the methods to obtain the first radial derivative of the Dirichlet problem (3.83). The other expressions that contain radial derivatives and divisions by r show similar behavior. The CG and BC methods prove to be stable and accurate, whenever $L = 2\pi$. If $L = 1/6$, then the accuracy levels shift higher but the methods stay stable and comparatively accurate.

To obtain the mixed derivatives $\partial_r \partial_z$, which are needed in the CLA code, the author chose to take the Fourier spectral derivative directly on the result of the methods $\mathcal{D}^{(1)}, CG^{Dir,(1)}$, giving $\partial_r \partial_z w(r,z)$, and to the results of $\mathcal{N}^{(1)/r}, CG^{Neu,(1)/r}$, which are then multiplied by r, giving $\partial_r \partial_z u(r,z)$. While the results of the Chebyshev-Galerkin methods suffer from the

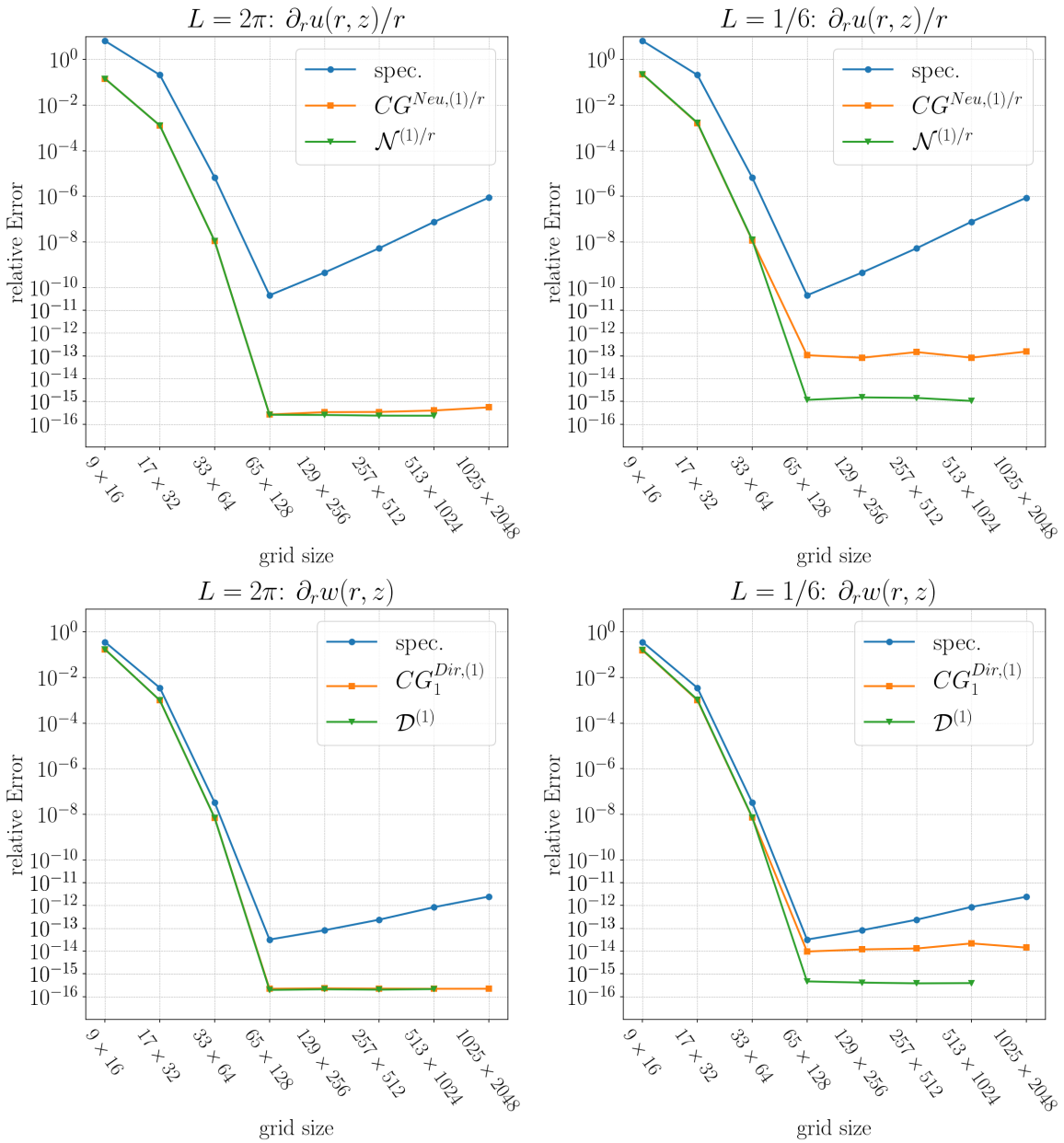


Figure 3.9: Derivative of the approximated solution to eq. (3.82) divided by r and derivative of the approximated solution to eq. (3.83) for $L = 2\pi$ and $L = 1/6$.

multiplication of the modes by the wave numbers $\tilde{k}$, the Bessel-Chebyshev method takes it easily as explained in the following. The reason lies in the character of the BC method which handles the exact inverses of second-order operators. The entries of the k-th slice of the cubic matrix for $\mathcal{D}^{(0)}$ behave roughly as $1/\tilde{k}^2$, and those of $\mathcal{D}^{(1)}$ as $1/\tilde{k}$, such that the absolute values of the entries of $\tilde{k} \cdot \mathcal{D}^{(1)}$ lie around 1. Because of this fact, the spectral differentiation in the Fourier dimension does not introduce instabilities in the Bessel-Chebyshev method that might stem from inaccuracies in the spectral coefficients of the RHS function, because erroneous modes are only multiplied by numbers of modulus ~ 1 instead of $\tilde{k}$. One could say that the solving matrices prepare the solution for a spectral differentiation in z. Figure 3.10 is showing this behavior for the test function case. Second order z-derivatives are needed in the CLA code as well. Those are obtained algebraically in case of the Poisson problem with Neumann boundary conditions, simply by computing $\partial_z^2 u = f - \partial_r^2 u - \partial_r u/r$. Note

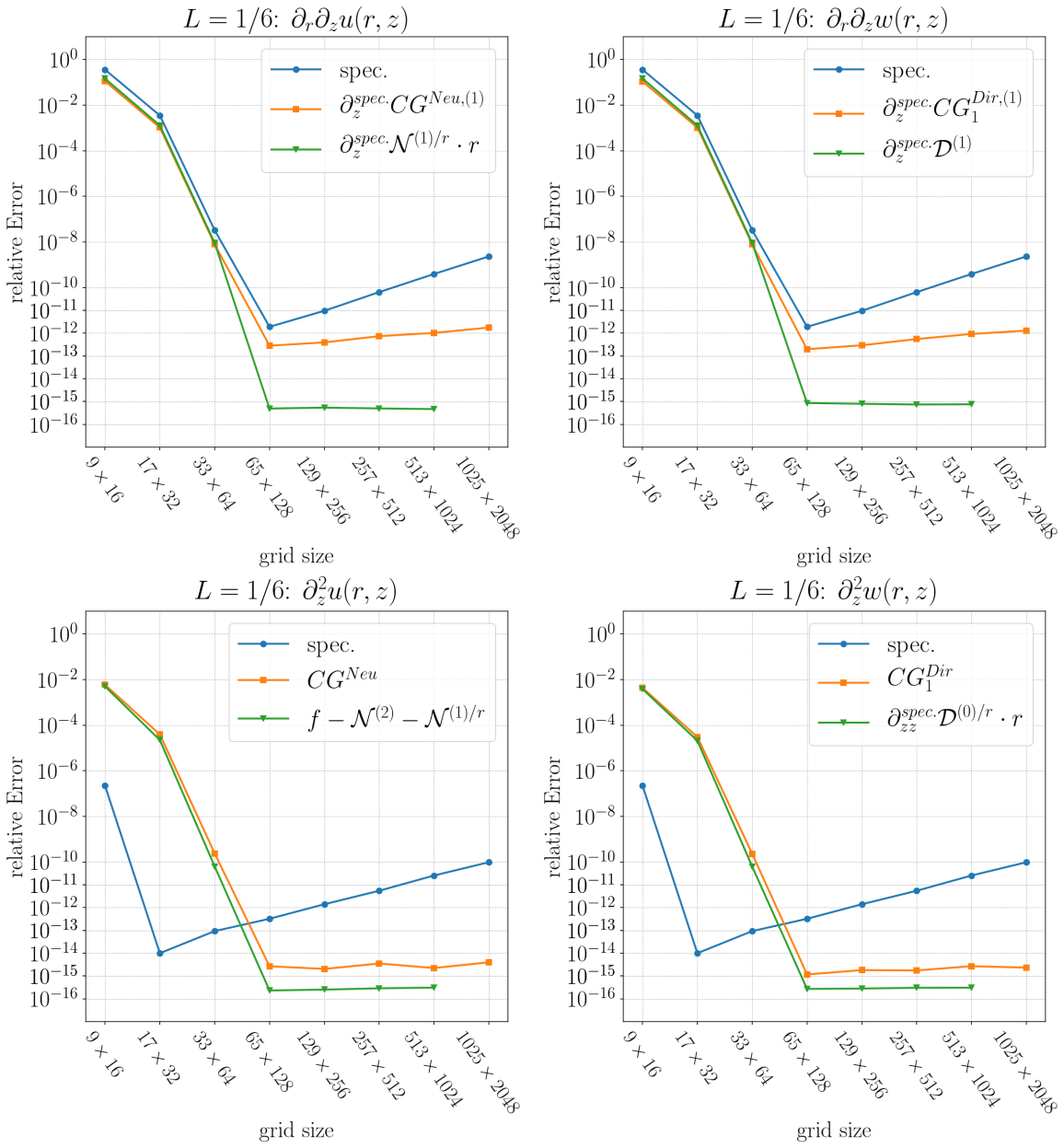


Figure 3.10: Mixed and double z-derivatives of the approximated solutions to eqs. (3.82) and (3.83) for $L = 1/6$. $\partial_z^{spec.\cdot}$ means that the derivative is obtained by spectral Fourier differentiation, i.e. by a multiplication of the modes by $\mathrm{i}\tilde{k}$, where i is the imaginary unit.

that, in the CLA, we need to solve anyway for the expressions that are subtracted. In the case of the Poisson problem with Dirichlet boundary conditions, an additional term in the equation hinders the described algebraic modification. The term w/r^2 cannot be obtained by spectral division, because of the strong instabilities and low accuracy. One could create another solving matrix (corresponding to $\mathcal{D}^{(0)/r^2}$) in order to solve for the double division, but at the present moment, the computation time of these solving matrices is just too long. Fortunately, as described above, the k-slices of the matrix that is used in the method $\mathcal{D}^{(0)/r}$ may readily be multiplied by $-\tilde{k}^2$ as the absolute values of their entries lie around $1/\tilde{k}^2$ and below. The methods $CG^{Dir/Neu}$ solve directly for the second order z-derivative, since the solving matrices for every mode are divided by $-\tilde{k}^2$.

Chapter 4

Interpolation

The Cauchy-Lagrange Algorithm (CLA) computes the truncated time-Taylor series of the Lagrangian particle trajectories of an incompressible ideal Euler flow in the form

$$X(a,t) \cong \sum_{s=0}^{S} X^{(s)}(a)\, t^s = \sum_{s=0}^{S} \left(R^{(s)}(r,z)e_r + A^{(s)}(r,z)e_\alpha + Z^{(s)}(r,z)e_z \right) t^s, \tag{4.1}$$

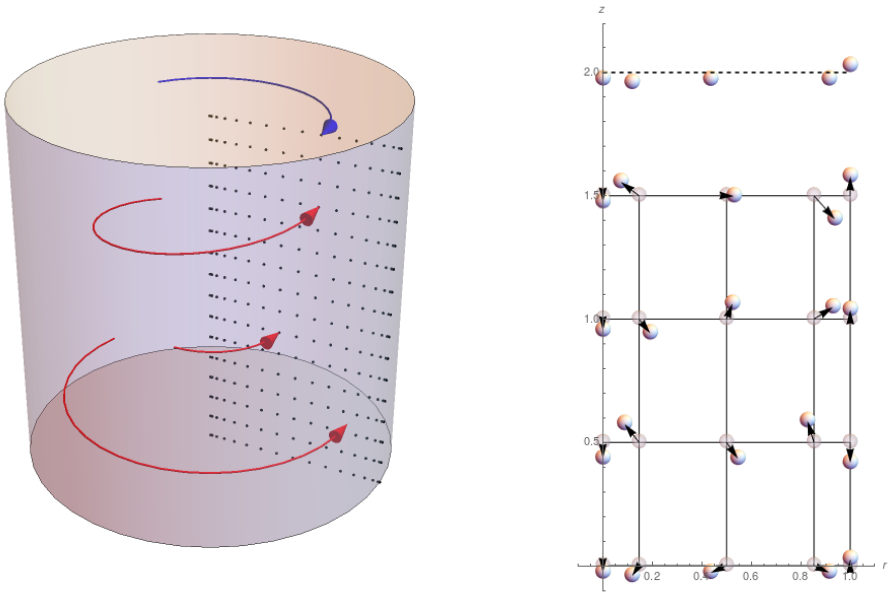
where $S \in \mathbb{N}$ denotes the truncation order. In our case the flow is axisymmetric, periodic along the z-axis and confined in a vertically unbounded cylindrical domain with a fixed radius $R = 1$. Because of the periodicity of the flow, only a sub-cylinder $D(1, L)$ (as defined in eq. (2.6)) with height $L > 0$ matters, where L is the periodicity length of the flow. A cylinder slice usually refers to a slice of the sub-cylinder as illustrated in Figure 4.1 (a) and we may choose a slice, say at $\alpha = 0$.
Once sufficiently many time-Taylor coefficients of X have been computed on a discrete set of Chebyshev-Fourier grid points via the recurrence mechanism stated in Section 2.6, a suitable time-step may be inserted into the time-Taylor series to compute the new particle positions in the cylinder. Due to the assumed axi-symmetry of the underlying flow, it is sufficient to compute the time-Taylor coefficients on a slice of the sub-cylinder $D(1, L)$. The discrete grid points on that slice, seen as distinct particles, disperse into the cylinder with evolving time. It is important to note, that the angular movement can be neglected and only the radial and vertical movements of the particles must be taken into account. To be more precise, the particles that lie on a circle in a plane perpendicular to the vertical axis show all the same movements due to the axi-symmetry. Thus, one may choose the particular particle that arrives directly on our chosen slice (at $\alpha = 0$) after the inserted time-step. In this way, the radial and vertical displacement of a particle can be reduced to a movement on the chosen slice departing from the point where the corresponding circle intersects with the slice. Since there exists such a circle to every grid point on a slice, we may choose those particles on the circle which arrive precisely on the slice. Mathematically, this becomes apparent by the fact that the angle of the initial point $re_r + ze_z$ is absent in the computation of the trajectory components as displayed in eq. (4.1), such that the choice of α in $e_r = (\cos\alpha, \sin\alpha, 0)^\top$ does not interfere in the computation of the dynamics.

As discussed in Section 2.2, X may be written as

$$X(r,\alpha,z,t) = X^r(r,z,t)e_r + X^\alpha(r,z,t)e_\alpha + X^z(r,z,t)e_z = \tilde{R}(r,z,t)e_{\tilde{R}} + \tilde{Z}(r,z,t)e_z, \tag{4.2}$$

where $e_{\tilde{R}} = (\cos\tilde{\Theta}, \sin\tilde{\Theta}, 0)^\top$ and $e_{\tilde{\Theta}} = (-\sin\tilde{\Theta}, \cos\tilde{\Theta}, 0)^\top$ with

$$\tilde{R} := \sqrt{(X^r)^2 + (X^\alpha)^2}, \quad \tilde{\Theta} := \mathcal{A}(X^\alpha, X^r) + \alpha, \quad \tilde{Z} := X^z, \tag{4.3}$$



(a) Particles move onto the slice that contains the Chebyshev-Fourier grid.

(b) Particle displacement on the (r, z)-plane on a 5×4 grid.

Figure 4.1: The modeled slice of the cylindrical domain $D(1, 2)$ with the underlying Chebyshev-Fourier grid is shown. Only the radial and vertical movements of the particles are relevant. The top particle positions in (b) are obtained by periodicity of the flow from those that correspond to the first horizontal grid line.

where $\mathcal{A}$ is defined in eq. (2.7). From eq. (4.3) it can easily be seen that the angle α of the initial particle only influences the angle $\tilde{\Theta}$ of the particle position after a finite time and may always be chosen such that $\tilde{\Theta} = 0$. Assume an axisymmetric and time dependent velocity field $v(r, \alpha, z, t) = v^r(r, z, t)e_r + v^\alpha(r, z, t)e_\alpha + v^z(r, z, t)e_z$, which is evaluated at time t_1 on the particle trajectory that originated from $re_r + ze_z$ at time t_0,

$$v(\tilde{R}, \tilde{\Theta}, \tilde{Z}, t_1) = v^r(\tilde{R}, \tilde{Z}, t_1)e_{\tilde{R}} + v^\alpha(\tilde{R}, \tilde{Z}, t_1)e_{\tilde{\Theta}} + v^z(\tilde{R}, \tilde{Z}, t_1)e_z\,. \tag{4.4}$$

If the initial grid points $(r_i, z_j)_{i,j}$ correspond to the particle positions at time t_0, then an interpolation may be performed in order to obtain the values of the components of v on the grid points $(r_i, z_j)_{i,j}$ at time t_1 from the values that are known on $(\tilde{R}(r_i, z_j, t_1), \tilde{Z}(r_i, z_j, t_1))_{i,j}$. In other words, we aim to achieve

$$v^\lambda(\tilde{R}(r_i, z_j, t_1), \tilde{Z}(r_i, z_j, t_1), t_1) \overset{\text{interpolation}}{\longrightarrow} v^\lambda(r_i, z_j, t_1), \tag{4.5}$$

for $(i, j) \in \{0, \ldots, N\} \times \{0, \ldots, M-1\}$, and $\lambda = r, \alpha, z$. The vorticity $\omega(r, \alpha, z, t) = \omega^r(r, z, t)e_r + \omega^\alpha(r, z, t)e_\alpha + \omega^z(r, z, t)e_z$ is interpolated in the same way, once it is obtained via the vorticity transport formula as explained in Section 2.2. The displacement inside the cylinder and the relevant radial and vertical movement inside of the slice in the cylinder is visualized in Figure 4.1. Thus, the axi-symmetry of the underlying flow reduces effectively all involved computations to two dimensions. This allows us to perform a two-dimensional interpolation instead of a three-dimensional one. However, the difficulty lies here in the fact that one has to interpolate from an irregular grid onto a regular grid, i.e. one has to perform a two-dimensional scattered interpolation.
Furthermore, particles may leave the convex hull of the Chebyshev-Fourier grid points in

vertical direction or show a contraction, such that there are grid points that do not lie in the convex hull of the scattered particles. In the latter case one could perform an extrapolation, but it is better to use the periodicity of the flow and simply include more particles from above and below the first and last horizontal grid lines. If $z_j = Lj/M$ are the original vertical grid points, then the top layers of particles above the last horizontal grid line at $L(M-1)/M$ are simply added by periodicity,

$$\tilde{R}\left(r_i, \tfrac{L(M+m)}{M}\right) = \tilde{R}\left(r_i, \tfrac{Lm}{M}\right), \tag{4.6}$$

$$\tilde{Z}\left(r_i, \tfrac{L(M+m)}{M}\right) = \tilde{Z}\left(r_i, \tfrac{Lm}{M}\right) + L, \tag{4.7}$$

for all $i = 0, \ldots, N$ and $m = 0, \ldots, M_1$, where $M_1 < M$ is the number of top layers to be added. Several layers below the first horizontal grid line are added similarly via

$$\tilde{R}\left(r_i, -\tfrac{Lm}{M}\right) = \tilde{R}\left(r_i, \tfrac{L(M-1-m)}{M}\right), \tag{4.8}$$

$$\tilde{Z}\left(r_i, -\tfrac{Lm}{M}\right) = \tilde{Z}\left(r_i, \tfrac{L(M-1-m)}{M}\right) - L, \tag{4.9}$$

where $m = 0, \ldots, M_2$ and $M_2 < M$ the number of bottom layers. Figure 4.2 shows an application of this procedure with one top and one bottom layer added to the original grid.

4.1 2D Scattered Interpolation

A scattered interpolation method seeks to obtain function values on points that lie in between irregularly distributed points on which the function values are known. The palette of such methods is not as vast as for regularly distributed points, simply because this setting is much more complex, as it aims to control a seemingly random distribution of points. Indeed, the position of the particles after a finite time is completely dependent on the underlying flow, which could place them literally anywhere. The only control parameter in our possession is the time-step that we may choose.

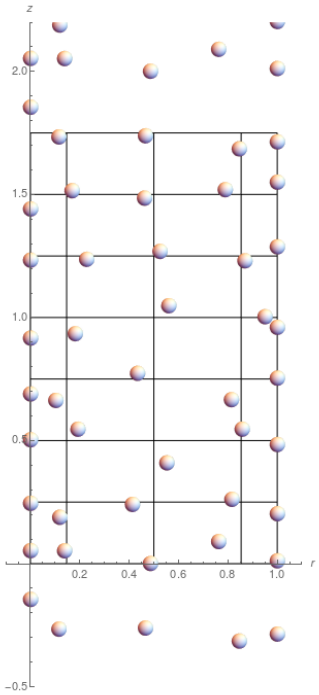


Figure 4.2: Particle layers are added by periodicity to include the underlying 5×8 grid in their convex hull.

The easiest, and possibly the most inaccurate, of scattered interpolation methods might be the method of the nearest neighbor. As the name indicates, the interpolation value is chosen among the function values on the points that lie the closest to the interpolation point. Interestingly, given a moderate flow that doesn't displace the particles far from their origin, i.e. by inserting a small time-step, the interpolation by nearest neighbor would simply take the function values on the scattered grid to use them as the function values on the original grid at time t_1 and produce fairly accurate results for the flow in a blistering time, because this step only involves a copying of one array to another. In this way, one cycle of the algorithm gets very fast, but can only use a very small time-step and the overall accuracy is limited. Needless to say, for ambitious Euler computations, especially towards possible finite time singularity detection, more sophisticated scattered interpolation

techniques must be applied. Common methods, which we consider here for an application in the CLA, are:

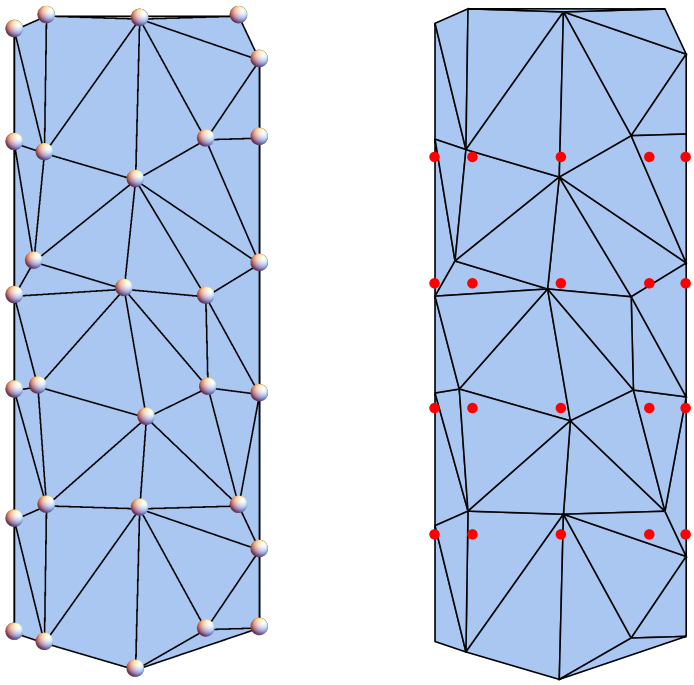
- Cascade interpolation
- Fitting of bicubic splines to the irregular data in the least squares sense
- Delaunay tessellation and subsequent linear barycentric interpolation
- Interpolation by radial basis functions (RBF)

To suit the context, those methods have not been tested, nor compared, for randomly scattered points in the (r, z)-plane, but rather for real displacements of particles induced by a stationary flow. It is important to remark that the (r, z)-positions of the particles are not seemingly randomly distributed on the cylinder slice in our case. Indeed, the underlying axi-symmetry of the studied flows prevents the particles from mixing "chaotically". In a general Euler flow in a cylindrical domain, as in any other domain, the particle trajectories cannot cross or touch each other. The mapping of those trajectories onto the (r, z)-plane of a slice, on any vertical slice of the cylinder, however, may cross on this plane and produce a seemingly random distribution of particles after a finite time-step. *On the contrary, in axisymmetric flows in a cylinder, any two mappings of distinct particle trajectories onto the (r, z)-plane cannot cross, because this would imply that other particles touch in the body of the cylinder.* One may argue that for this reason, the best adapted interpolation method for our particular problem is the one that exploits the aforementioned property of the axisymmetric flow in our cylindrical domain. The cascade interpolation presented in [100] is by far the best suited method in this respect. It was developed explicitly for the Lagrangian displacement in a fluid flow and reduces the two-dimensional problem into a sequence of one-dimensional problems. This method is discussed in detail further below. Here, we proceed with a brief description of the other methods from the above list.
A *least square fitting of bi-cubic B-Splines* to the scattered Lagrangian grid is implemented in the public library *ALGLIB* [14]. Unfortunately, a detailed mathematical description or scientific references for their methods seems not to be provided by its authors. It may also be noted that least square fittings are actually only approximations of surfaces with a certain regularity to given data points. That is, the resulting surface may not necessarily include the original points. In contrary, the interpolating surface that we demand should agree with all data points. Nonetheless, the least squares fitting routine from ALGLIB performs better than the RBF or Delaunay interpolation in our test scenario, but still stays far behind the cascade interpolation. In order to make sense out of the rather scarce explanations in [14], we would like to refer to the algorithms of curve and surface fitting by B-splines in [94, chapter 9], where a least squares surface fitting is explained via a sequence of fitted curves. For a fitted curve to data points $(Q_k)_{k=1,\dots,m}$, a least squares fitting procedure in one dimension using cubic B-splines seeks to minimize the expression

$$\sum_{k=1}^{m-1} |Q_k - C(u_k)|^2, \tag{4.10}$$

with respect to the variable points, or weights, $P_0, \dots, P_n$, $n < m$, that define

$$C(u) = \sum_{i=0}^{n} N_{i,3}(u) P_i, \quad u \in [0,1], \text{ s.t. } C(0) = Q_0 \text{ and } C(1) = Q_m\,. \tag{4.11}$$



(a) **(b)**

Figure 4.3: (a) A Delaunay tesselation of the interpolation region for a cylinder $D(1, 2)$; the particles give the vertices of the triangles. (b) The tesselation together with the interpolation points on a 5×4 Chebyshev-Fourier grid. Every interpolation point lies in a triangle due to the particle layers that were added in a previous step.

The $N_{i,3}$ are the cubic B-splines defined in Section A.3 and the $(u_k)_k$ in eq. (4.10) are the corresponding knots. For further details on the least squares fitting procedure with B-splines, see [94, 37, 117] and references therein. A more general treatment of curve and surface fitting is given in [76]. The reason why this fitting is quite accurate after all, might be connected to the fact that the scattered grid is not too distorted for small time-steps and stays close to a product grid, i.e. a grid that is made of rows and columns of rectangles. Another method in this spirit is scipy's *griddata-spline* method. It consists of a surface fitting via piece-wise cubic polynomials, where the surface minimizes the curvature approximately[1].
The *Delaunay tesselation*, or Delaunay triangulation, of randomly distributed points is a common method for scattered interpolation purposes. This tesselation, initiated in [39], maps a mesh of triangles onto the distorted grid such that the scattered points, also called *sites*, give the vertices of the triangles. An exact definition can be found in [47], it reads: "The Delaunay tessellation is obtained by connecting with a line segment any two points p, q, for which a circle exists that passes through p and q but does not contain any other site in its interior or boundary." Such a partition also yields the property that the minimal inner angle of all involved triangles is maximized. Many more features of the Delaunay tesselation, one being the fact that it can usually be obtained in $O(n \log(n))$ operations, where n is the number of sites, are stated and proved in [91, 47, 96]. An example of such a triangulation is shown in Figure 4.3. It is used to interpolate onto the underlying Chebyshev-Fourier grid via linear barycentric interpolation. A grid point a in the (r, z)-plane lies in a triangle which is defined by three point vertices a_1, a_2, a_3. The barycentric coordinates are then given by the

[1] for details see
https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.griddata.html and references therein

real numbers $\beta_1, \beta_2, \beta_3$, such that $\beta_1 + \beta_2 + \beta_3 = 1$, and

$$\sum_{i=1,2,3} \beta_i a_i = a\,. \tag{4.12}$$

We refer to [34, p. 216] for further details and a method on how to obtain the β_i. The scalar function values that are known on the vertices, namely $f(a_1), f(a_2), f(a_3)$ are then used to obtain the value at a by a linear combination of the latter with coefficients β_i, i.e.

$$f(a) = \sum_{i=1,2,3} \beta_i f(a_i), \tag{4.13}$$

so that this interpolation is only exact for affine functions.
The use of *radial basis functions (RBF)* is another common scattered interpolation method in meteorology and geography, but only really applicable for a rather small amount of data points. In its ordinary form, this method needs $O(\tilde{N}^3)$ operations, where $\tilde{N}$ is the number of data sites, and thus, $\tilde{N} \geq (N+1)M$, where $N+1$ and M are our radial and vertical grid dimensions. Also, it locates at the lower end of the performance scale, comprising complexity and accuracy, of all tested methods for the CLA. The method is rather simple in its nature and here briefly reviewed for completeness; see [118, 98] for detailed introductions and examinations of this method together with code snippets for its implementation. As the name might indicate, the radial basis function method uses scalar functions that solely depend on the distance between data sites. The functions are linearly combined in order to approximate interpolation values, i.e. for a set of scattered data sites $\{a_0, \ldots, a_{\tilde{N}-1}\}$ in a region $\mathcal{R}$ of any dimension, a function $\varphi = \varphi(r)$ is defined, such that

$$w(a) \coloneqq \sum_{i=0}^{\tilde{N}-1} w_i \varphi(\|a - a_i\|_2), \quad a \in \mathcal{R}, \tag{4.14}$$

interpolates a function y that is known only on the scattered data sites. That is, an equality of the form

$$y(a_j) = \sum_{i=0}^{\tilde{N}-1} w_i \varphi(\|a_j - a_i\|_2), \quad j = 0, \ldots, \tilde{N}-1, \tag{4.15}$$

is pursued and gives rise to a linear system of equations for the $\tilde{N}$ unknown weights w_i. Indeed, the RBF φ can be chosen such that the set of equations (4.15) is always solvable for distinct points. Common choices for the RBF φ are the Gaussians ($\varphi(r) = \mathrm{e}^{-cr^2}$), the inverse multiquadric ($\varphi(r) = 1/\sqrt{c^2 + r^2}$), and the multiquadric ($\varphi(r) = \sqrt{c^2 + r^2}$), where $c > 0$. For a fixed dimension it may also be chosen to entail a positive definite interpolation matrix $\varphi(\|a_j - a_i\|)_{ji}$ (see [88]). The complexity may further be reduced by using layer techniques as implemented in the public library ALGLIB, which claims to be very accurate and to perform a Gaussian RBF method in $O(N \log(N))$ operations[1]. In the performed tests, however, this was the slowest and most inaccurate method. Other tested RBF interpolations[2] produced very similar results. A version of the RBF method is the so called Shepard method (see [98]), which was also tested for the CLA without major improvements.

[1] see https://www.alglib.net/interpolation/fastrbf.php

[2] notably that from
https://people.math.sc.edu/Burkardt/cpp_src/rbf_interp_2d/rbf_interp_2d.html and
https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.Rbf.html

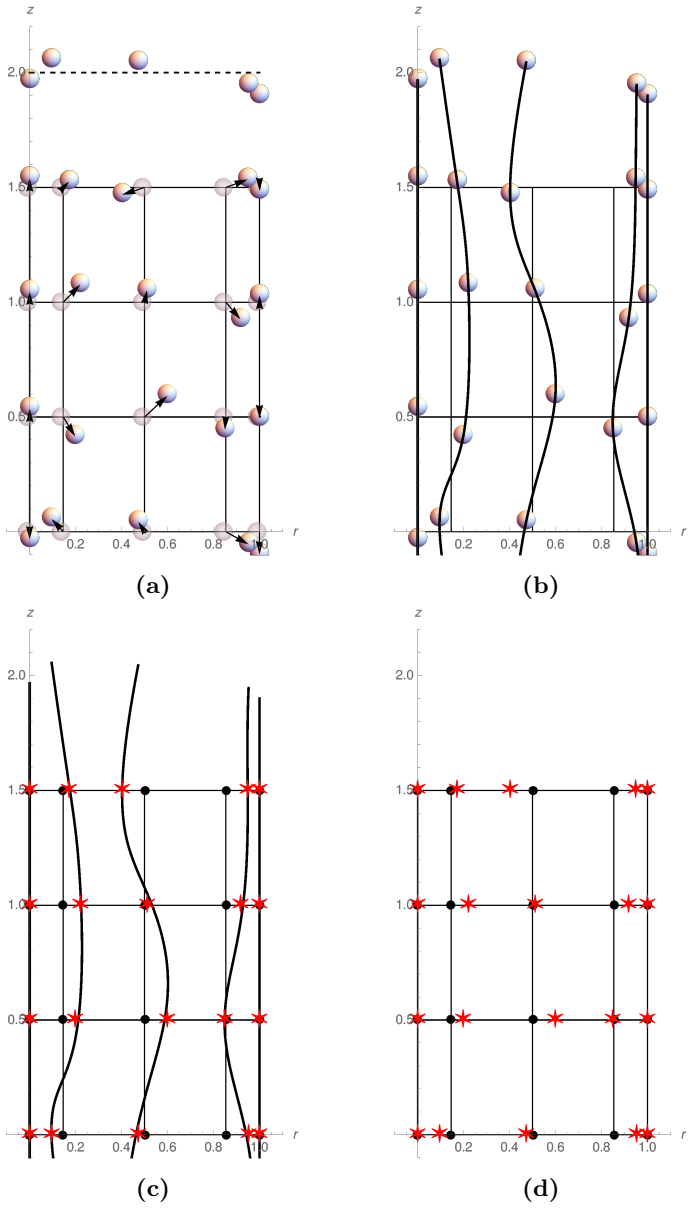


Figure 4.4: Process visualization of the cascade interpolation in a slice of the cylinder. (a) particle layers are added (b) particles that originate from a column are interpolated (c) the intersections with horizontal grid lines form a hybrid grid indicated by red stars (d) the values on the original grid (black dots) are interpolated from the hybrid grid. Only one-dimensional (scattered) interpolations are needed in this method.

The cascade interpolation with Lagrange polynomials and B-spline functions The cascade interpolation, introduced in [100], was intended to accelerate Lagrangian weather simulations (i.e. [8, 86] and references therein). This method has proven to be quite effective for Lagrangian displacements as shown in [95], where the CLA has been developed for a two-dimensional periodic Euler flow. In fact, the interpolation problem in the latter reference and in this thesis are almost identical, the only difference is that the displacements in the cylinder slice are periodic only in the vertical direction and not in the radial direction. The cascade interpolation is basically a composition of one-dimensional interpolations. The name is anticipating the passing of information given on scattered points down to more regular hybrid grids with each dimension until a regular point distribution is attained. The authors of both references [100, 95] explain the method via mutual variable dependencies of

Lagrangian and Eulerian coordinates which involves notational inconveniences and can be confusing to follow. The description we give here is for that matter intended to be rather visual, which shall also exhibit the simplicity of the method. In two dimensions, suppose we are given a function that is known on the distorted Lagrangian particle grid, $w(\tilde{R}, \tilde{Z})$, and seek to interpolate this function on the original regular grid $(r_i, z_j)_{i,j}$, which is equipped with horizontal and vertical grid lines as in Figure 4.4 (a). One may pick a column of the grid in fixing $i = i^*$ and plot the displacement of the corresponding grid line. This plot then displays a function known on the (vertical) abszissæ $\tilde{Z}(r_{i^*}, z_j)$ with ordinates $\tilde{R}(r_{i^*}, z_j)$ to which the function values $w(\tilde{R}(r_{i^*}, z_j), \tilde{Z}(r_{i^*}, z_j))$ are assigned. The plotted lines of all such columns are given exemplary in Figure 4.4 (b). Subsequently, as the $\tilde{Z}(r_{i^*}, z_j)$ are irregularly distributed on the vertical grid line that corresponds to i^*, a scattered one-dimensional interpolation is at hand to find the values of the intersections with the plotted lines and the horizontal grid lines, that is at the abszissæ $(z_j)_j$. These intersections are denoted by $\bar{r}_{i^*,j}$, because of their correspondence to the z_j and the column at r_{i^*}, they give rise to a hybrid grid with new coordinates, which are indicated by red stars in Figure 4.4 (c). A second scattered interpolation along the vertical grid line produces the values $w(\bar{r}_{i^*,j}, z_j)$. Once this is done for every vertical grid line, one may fix a horizontal line at $z_j = z_{j^*}$ and perform another scattered one-dimensional interpolation on the irregularly distributed $\bar{r}_{i,j^*}$ in order to obtain the function values $w(r_i, z_{j^*})$. Thus, in order to obtain all function values on the original grid, one has to perform $2(N + 1)$ interpolations on the vertical grid lines and subsequently, M interpolations on the horizontal lines of the hybrid grid.
In three dimensions with coordinates $(x, y, z) \in \mathbb{R}^3$, one performs the described method first on every parallel plane that corresponds to a grid point dimension, i.e. (x, y)-planes through the $(z_j)_j$ coordinates of the grid, and performs then a scattered interpolation along all grid lines that are perpendicular to those planes and pass through the formerly fixed grid point coordinates $(z_j)_j$.

The authors of [100] state that the associated algorithm is of linear complexity in the order of accuracy per point, and exactly $3\tilde{O}$ for an order $\tilde{O}$. That is, if a cubic Lagrange polynomial (order= 4) is used, then the operations count is $3 \cdot 4$ for one single point value. However, this seems to be incorrect, as one needs $\tilde{O}$ points for one interpolation point, giving one new coordinate on a hybrid grid. To arrive at the next coordinate one needs again $\tilde{O}$ points from the hybrid grid that were all obtained by interpolations, which brings the operation count to a quadratic term, etc. However, this operation count is per point and in terms of the approximation order. In return, for a fixed prescribed accuracy, the associated algorithm in a plane, with a grid size of NM points, has an operation count of $O(MN)$, which is optimal. In the cascade interpolation method, the composition of one-dimensional scattered interpolations also offers a wide range of possibilities, as one-dimensional scattered interpolations are much easier to handle. For instance, one may use one-dimensional B-spline functions, which have no canonical counterpart in higher dimensions. Also Lagrange polynomials of arbitrary order may be used here and allow for fast scattered one-dimensional interpolations. The latter was already applied in [95] with robust results, but also the B-spline function interpolation is a suitable candidate, even though it is much slower than the Lagrange interpolation due to the appearing linear systems, which have to be solved.

Inside of the cascade interpolation, any other one-dimensional scattered interpolation technique and even alterations of them are also applicable. For instance, one may apply one interpolation method in one dimension and a second in the other dimension. This, however, only makes sense if there are special reasons, like apparent discontinuities in one dimension. The current implementation of the CLA contains a mechanism that allows the user to switch between Lagrange and B-spline interpolations with arbitrary orders of accuracy.

These alternations of accuracy orders may even take place within one program execution. The purpose of such an implementation was to allow for the so called *p-adaptive cascade interpolations*, which alter the degree of the interpolation polynomials in order to satisfy a certain condition. Such a condition is given, for example, by the minimization of the total kinetic energy loss and total helicity loss[1]. More precisely, the interpolation orders inside the cascade interpolation for the velocity field components are cycled through a given range and, eventually, that interpolation result is chosen, which yields a minimal energy loss. Subsequently, the same interpolation loop is applied to the vorticity field with the aim of minimizing the helicity loss, for which the formerly found, energy-loss minimizing, velocity field is used. The interpolation orders must not necessarily be the same in radial and vertical direction, such that for each radial interpolation order a loop can be performed over all the available vertical interpolation orders (extensive p-adaptive cascade interpolation). However, those p-adaptive interpolation techniques were later neglected, because the preservation of total kinetic energy or helicity are only necessary (and not sufficient) conditions for a solution to the incompressible Euler equations. Also, those p-adaptive interpolation methods imply an elevated numerical complexity and slow down the program execution without delivering much better results compared to a well chosen fixed order cascade interpolation.

For brevity, we recall the Lagrange polynomial interpolation here and move the definition and properties of the B-splines interpolation method to Section A.3. For a number $n \in \mathbb{N}$ of strictly increasing sites $x_0 < x_1 < \ldots < x_{n-1}$, the Lagrange basis polynomials of order n (and degree $n-1$) are defined by

$$I_l(x) \coloneqq \sum_{\substack{0 \le i \le n-1 \\ i \neq l}} \frac{x - x_i}{x_l - x_i}, \quad l = 0, \ldots, n-1\,. \tag{4.16}$$

A function y, with function values $y(x_0), \ldots, y(x_{n-1})$, may be interpolated by a Lagrange polynomial in the form

$$y(x) \cong \sum_{l=0}^{n-1} y(x_l) I_l(x)\,. \tag{4.17}$$

The fact that this method doesn't involve a linear system, which needs to be solved, makes it very fast, and numerous implementations can be found in public libraries. For the CLA, a Fortran implementation by John Burkardt[2] has been adapted to Fortran 2008 standards and transformed into a multi-precision version. In this routine, the sites are chosen such that the interpolation point is in their center, if possible, i.e. a centered Lagrange polynomial is computed with the same amount of sites to one side as to the other side of the interpolation point. In vertical direction, this can always be fulfilled, because of possibility to add particle layers using the periodicity of the interpolated functions. In the step where additional layers are added, the interpolation order is taken into account, such that sufficiently many points are available to allow for a centered lagrange interpolation. In radial direction, especially close to the boundary or pole, a periodicity, or parity with respect to the interval end points, is not known and thus no valid extension outside of the interval $[0, 1]$ can be constructed. In this case, the interpolation polynomial is still constructed as above with sites that lie closest to the boundary or pole. In this scenario, the Chebyshev extrema (also their zeros) are well known to reduce the Runge phenomenon [17, 37], such that the Lagrange interpolation is stable even if an un-centered site distribution is used close to the interval end points.

[1] See Section 6.2 for details on the preservation of kinetic energy and helicity for the incompressible Euler equations.

[2] https://people.sc.fsu.edu/~jburkardt/f_src/lagrange_interp_1d/lagrange_interp_1d.html

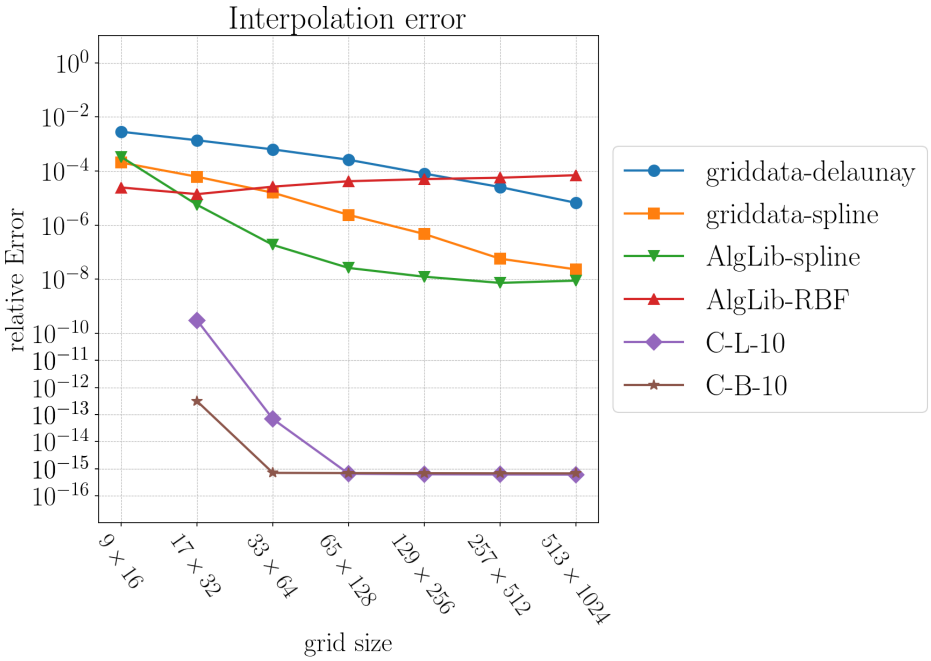


Figure 4.5: Comparison of the reviewed scattered interpolation methods in the (r, z)-plane. The abbreviations C-L-10 and C-B-10 stand for the cascade interpolation with Lagrange and B-spline functions respectively, both with degree 10.

In the B-spline interpolation, briefly recalled in Section A.3, one has to solve for the coefficients α_i in

$$\sum_{i=0}^{n-1} \alpha_i N_{i,p-1}(x_j) = y(x_j), \tag{4.18}$$

where $N_{i,p-1}$ are the B-splines of order p and degree $p-1$. A B-spline interpolation module, which is based on routines from [37], is publicly available on Github[1]. Parts of this module were extracted and included in the CLA code after an adaptation to our needs. In these routines, the $n+p$ knots of the B-splines are distributed in the interpolation interval, such that $2p$ knots are placed on the interval end points, p on each end point, and the rest is placed on the data points starting from $x_{p/2}$, if p is even, and in between the data sites, if p is odd, starting after $x_{(p-1)/2}$. See [37] for more information on knot placing.

Numerical comparison of the interpolation methods As mentioned before, it is not necessary to compare the scattered interpolation methods from this section on a set of randomly distributed points. Instead, a real displacement field is taken to test and compare the methods under realistic circumstances. For this purpose, a stationary flow, presented in [81] and described in detail in Section 6.3, was chosen to define a particle dispersion after an appropriate time-step. The coefficients $X^{(s)}$ were calculated in their cylindrical form via the recurrence mechanism described in Section 2.6. Subsequently, the truncated Taylor series were summed with a time-step small enough to ensure that the calculated velocity and vorticity fields show a relative error below 10^{-15} in double precision on the distorted grid. Furthermore, it was verified that at least one interpolation method is accurate enough to produce a relative error below 10^{-15} on the original grid, to ensure that the test data is

[1] The author of this module is Jacob Williams, it can be obtained from https://github.com/jacobwilliams/bspline-fortran/blob/master/src/bspline_sub_module.f90. Note that the one-dimensional interpolation there applies to scattered data sites, even though it is stated otherwise.

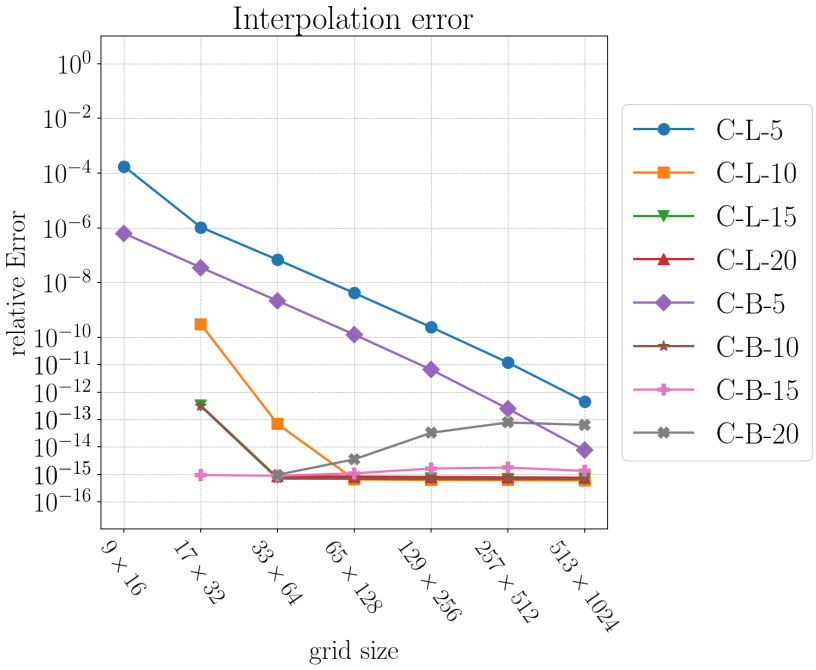


Figure 4.6: Comparison of Lagrange (C-L) and B-spline (C-B) cascades of different degrees.

accurate enough.
The conclusion of all executed tests on a number of different stationary flows is that the performance of the treated methods may be ordered as in the list on page 65. In our setting, the cascade interpolations with one-dimensional Lagrange and B-spline interpolation are by far the most accurate methods and they are the only ones which achieve relative errors below 10^{-15} in a double precision computation. Furthermore, they are not only the most accurate but also the fastest of all tested methods (nearest neighbour method excluded). Where ALGLIB's least square fitting of cubic splines takes for all six components of the velocity and vorticity field an average of 169 seconds on one single thread, a 10th degree B-spline cascade takes only 3.2 seconds and a 10th degree Lagrange cascade 1.2 seconds. The produced relative errors, i.e. for the vertical vorticity component, in this particular run at a moderately high resolution are of 2.9×10^{-8}, 6.6×10^{-16} and 6.1×10^{-16} for the ALGLIB [1], B-Spline cascade and Lagrange cascade method respectively . Additionally, as the cascade interpolation is a composition of several one-dimensional interpolations, the implementation is efficiently parallelized and makes the cascade interpolation the method of choice for the CLA.

4.2 Refinement Methods

In reference [95], it was already noted that the interpolations may be performed on a finer grid or to increase the point density in regions, where large gradients reside. This is achieved as follows: a (pseudo-)spectral approach implies that, at any given time, we know a fully spectral approximation of all involved functions. A good spectral approximation of a given function allows to use padding of the coefficient vectors to evaluate the functions on as many grid points as needed to perform an interpolation, in the same manner as one pads the spectral projections with zeros to avoid aliasing errors in spectral methods.
The point is, that under sufficient refinement, any interpolation scheme should attain its min-

[1] The minimal relative error of the ALGLIB method that was achieved on the test flow with very high resolution was about $1. \times 10^{-12}$.

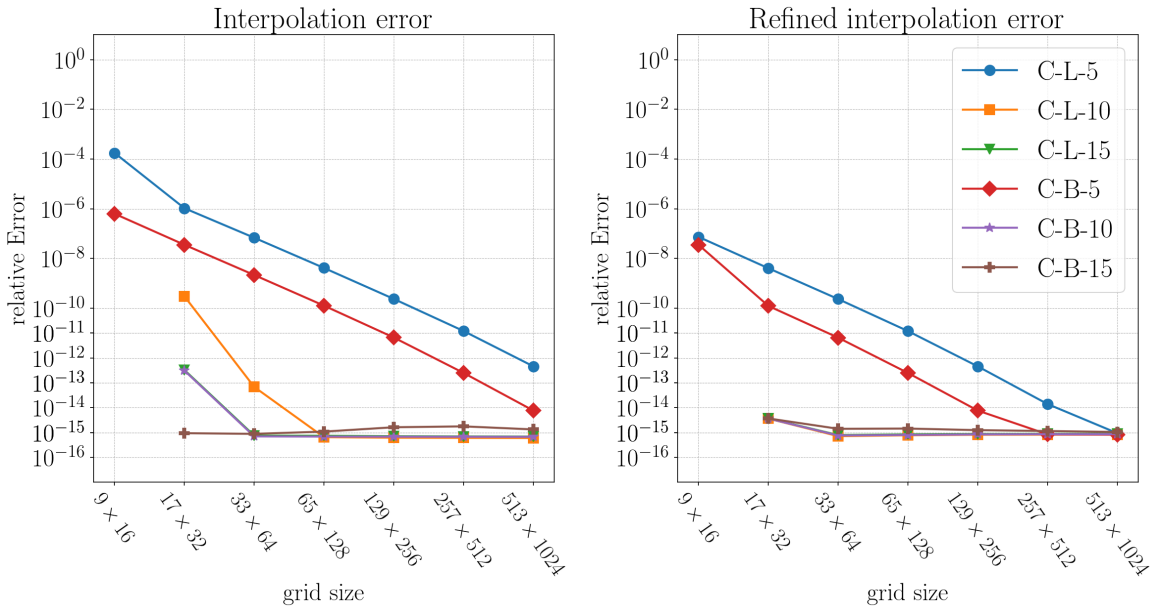


Figure 4.7: (a) The Cascade interpolation with Lagrange and B-spline interpolations is applied without refinements. (b) The refinement factor powers $\text{rfp}_r = \text{rfp}_z = 2$ have been applied. The low order schemes show a relative error that is several orders of magnitude lower than without the refinement. The higher order schemes produce minimal errors even on very low resolutions.

imal error, even if it has only a low convergence rate, such as low order Lagrange polynomials. Clearly, with this method one does not obtain any further information of the interpolated function.

To be more precise, the functions $\tilde{R}(r,z), \tilde{Z}(r,z)$ and $w(\tilde{R}(r,z), \tilde{Z}(r,z))$, where w can be any cylindrical component of v or ω, are known approximately in the form of Chebyshev-Fourier coefficients. The length of the coefficient vectors in one dimension corresponds directly to the number of Chebyshev or Fourier points via the underlying fast Fourier transforms (FFTs). By adding any number of zeros to the coefficient vectors, one increases the number of corresponding physical grid points together with the related function values. As the interpolations are carried out in physical space, one may simply evaluate the interpolating functions on the original grid points. As a matter of fact, these refinement techniques are, at least in one dimension, not much slower than the interpolations on the original grid due to the unchanged number of interpolation points. The Lagrange interpolation of order n uses a fixed number (n) of grid points for one interpolation value, no matter if in one scenario the grid points are denser than in the other. Hence, the number of executed evaluations does not increase in one dimension and only the increased length of the coefficient vectors slows the execution as it enters into the transforms as $O((N+m)\log(N+m))$, if $m \in \mathbb{N}$ is the discrete number of length increments of the spectral series (via padding) and N the original number of physical sites or coefficients. In the cascade interpolation in 2D, however, it is mandatory to perform $N+m$ interpolations in the first dimension in order to obtain the higher point density in the hybrid grid in the second dimension, where then only N interpolations are necessary. On the other hand, it is possible to simply evaluate the spectral series in additional points by direct summations to locally increase the point density where needed.

This refinement method can in particular be used for low resolution runs, such that the Poisson problems are solved in lower, yet sufficient, resolution, while tracking the absolute values of high modes. If the high modes fill up, e.g. when the flow gets more turbulent, the

resolution may be increased globally, in using this refinement technique. *In other words, the presented technique allows for an adaptive mesh implementation of the spectral CLA.*
Here, thanks to re-applied refinements in the interpolation step, the interpolation methods may always be executed with a sufficient point density. The current implementation of the CLA comprises the capability of increasing the total grid size by factors of $2^p, p \in \mathbb{N}$, in radial or vertical direction independently. The choice for these factors is made in sight of the formerly discussed complexity increment and the fact that the transforms still take a large portion of the overall complexity. The underlying Chebyshev-Fourier grid is of the form $(2^n + 1) \times 2^m$, which, among other features, maximizes the speed of the related FFTs as discussed in Section 3.1. Hence, it is favourable to keep this form of grid sizes. The finer grids are then of the form

$$(2^{n+p_1} + 1) \times 2^{m+p_2}, \tag{4.19}$$

with $p_1, p_2 = 1, 2, \ldots$. The local increase of point densities via direct summations of the spectral series as described above has not yet been implemented.

Notation. The natural numbers p_1, p_2 will be called the *refinement factor powers* and abbreviated by rfp_r or rfp_z with respect to the corresponding dimension.

See Figure 4.7 for an application of the described refinement method, where the refined grids are four times larger than the original grid ($\mathrm{rfp}_r = \mathrm{rfp}_z = 2$).

Chapter 5

Time-Stepping

5.1 The Radius of Convergence

The time-Taylor series of the characteristic curves, or particle trajectories, to an incompressible Euler flow v in a compact domain Ω, given by

$$X(a,t) = \sum_{s=0}^{\infty} X^{(s)}(a) t^s, \tag{5.1}$$

converges point-wise absolutely for small enough values of t, whenever the gradient of $X^{(1)} = v_0$ is Hölder continuous of exponent $\alpha > 0$. The convergence can also be examined in the L^p sense, i.e.

$$\lim_{S\to\infty} \|X^{(s)}(\,.\,,t) - \sum_{s=0}^{S} X^{(s)}(\,.\,)t^s\|_{L^p} = 0, \tag{5.2}$$

which is assured if

$$\|X(\,.\,,t)\|_{L^p} \leq \sum_{s=0}^{\infty} \|X^{(s)}\|_{L^p} t^s < \infty, \tag{5.3}$$

for $t > 0$. A radius of convergence $\varrho_p > 0$ to the sum in eq. (5.3) may be defined via the Cauchy-Hadamard formula

$$\frac{1}{\varrho_p} = \limsup_{s\to\infty} \|X^{(s)}\|_{L^p}^{\frac{1}{s}}, \tag{5.4}$$

another definition being given by the ratio rule

$$\varrho_p = \lim_{s\to\infty} \frac{\left\|X^{(s-1)}\right\|_{L^p}}{\left\|X^{(s)}\right\|_{L^p}}, \tag{5.5}$$

if the latter limit exists. Clearly, if (5.1) converges absolutely in one L^p space, then the convergence also holds in any L^q with $1 \leq q < p \leq +\infty$. Also, the associated convergence radii behave as $\varrho_q \geq \varrho_p$ for $p \leq q$. To see the latter, one makes use of the well known estimate $\|f\|_{L^q} \leq |\Omega|^{1/q-1/p} \|f\|_{L^p}$, where $|\Omega|$ denotes the Lebesgue measure of Ω, and the definition eq. (5.4) of ϱ_p, as well as the fact that $\lim_{s\to\infty} |\Omega|^{(1/q-1/p)/s} = 1$. Consequently, $\varrho_\infty > 0$ implies the convergence in any L^p space and $\varrho_\infty = \inf_{p\in\mathbb{N}} \varrho_p$. On the other hand, the point-wise radius of convergence of the series (5.1) can be determined by

$$\varrho_a = \frac{1}{\limsup_{s\to\infty} \|X^{(s)}(a)\|^{\frac{1}{s}}} = \lim_{s\to\infty} \frac{\left\|X^{(s)}(a)\right\|}{\left\|X^{(s+1)}(a)\right\|}, \quad a \in \Omega, \tag{5.6}$$

where the involved norm can be any of $\mathbb{R}^d \supset \Omega$. As in [95], a global radius of convergence $\bar{\varrho}$ is then obtained by minimizing over a, i.e.

$$\bar{\varrho} = \min_{a \in \Omega} \varrho_a \,. \tag{5.7}$$

The minimum is attained, because we deal with regular functions on a compact domain. In a numerical setting, this may produce acceptable results for high resolutions, even though the numerical complexity is quite high. For the numerical investigations in this work, the focus has been set on the radii ϱ_∞ and ϱ_2. Without difficulty, one verifies that

$$\varrho_\infty \leq \varrho_p \leq \bar{\varrho}, \tag{5.8}$$

for all $1 \leq p \leq \infty$. Furthermore, in [95] it is stated that if all $X^{(s)}$ belong to L^p and $X(\,.\,,t)$ belongs to L^p for all $t < \bar{\varrho}$, then there holds $\bar{\varrho} = \varrho_p$. This implies equality in eq. (5.8), for the case of continuous functions on a compact domain. Indeed, $t < \bar{\varrho}$ always implies $X(\,.\,,t)$, $X^{(s)} \in L^\infty(\Omega) \subset L^p(\Omega)$ and, hence, $X(\,.\,,t)$, $X^{(s)} \in L^p(\Omega)$ and $\bar{\varrho} = \varrho_p, \forall p$. Note that in [95] a two-dimensional periodic flow has been treated, but their results hold in the case of a bounded domain too, because in the periodic case the involved norms are integrals over the bounded periodicity domain. Thus, their argumentation directly translates to bounded domains, also in higher dimensions.

A central question to the CLA is how to obtain this radius of convergence in an accurate way, or, if that's not possible, how to choose an appropriate time-step. Unfortunately, there is currently no explicit formula for the convergence radius available. Thus, the choice of the time-step becomes a challenging aspect of this algorithm. In principle, it would be possible to obtain an explicit formula for the convergence radius from the constructive proof in [12]. However, the estimates that are used in [12] involve constants which originate from Schauder estimates of the Newton potentials. Those constants are in general hard to compute and slight variations in their determined value might have strong influence on the value of the later computed radius of convergence. In the space $C^{1,\alpha}$, this convergence radius $\varrho > 0$ for (5.1) is typically of the form

$$\varrho = \frac{C}{\|\omega_0\|_\alpha}, \tag{5.9}$$

where $C = C(\Omega, \alpha)$ is a positive constant that depends only on the geometry of the domain Ω and $\alpha > 0$. The initial vorticity ω_0 can be replaced by $\nabla \times X^{(1)}$, since $X^{(1)} = v_0$. Given the constant C, one would still have to compute a Hölder-norm numerically during the simulation, which can be inaccurate, depending on the resolution as well as on the Hölder exponent α. For some flows it is possible to approximate the radius of convergence by techniques discussed below (in Section 5.2). Such flows are for example the stationary flows also discussed below. Even though the constant $C(= \varrho \, \|\omega_0\|_\alpha)$ in eq. (5.9) is difficult to obtain analytically from the proofs of analyticity, but it can be calculated provided a convergence radius for some (smooth) flow is known. One only needs to additionally compute the Hölde-norm of the initial vorticity field. If an exact radius is not known, then it is possible to use the techniques presented below to approximate some radii of convergence for some suitable flows. The vorticity fields of such flows should be at least of regularity C^1, then a fixed α (e.g. $\alpha = 1/2$) may be chosen accordingly. If the approximations of the convergence radii are sufficiently accurate, then the computation of the constant C should, for fixed α, always produce the same result because then it depends only on the fixed geometry of the domain. Initially, the vorticities are known analytically and equally their Hölder norms. During the simulation, it could be better to aim for convergence in $H^s, s > 5/2$, because the norms $\|\omega_0\|_{H^s}$ might be easier to compute

numerically. This is the case for our Chebyshev-Fourier spectral discretization. However, no effort has yet been undertaken to compute the constant C via the above method for our cylindrical domain.
In the sequel of this chapter we try to find appropriate time-steps without knowing the exact convergence radius. For this we get a start from [95], where the authors choose the time-step such that the highest order term in a truncation of (5.1) is below a given accuracy, which is meant to bound the overall error below a given accuracy. The argumentation for this choice in the aforesaid reference is unfortunately somewhat opaque to the author of this thesis, but we will follow the line of reasoning momentarily. A truncation of the time-Taylor series of X is denoted by

$$X^S(a,t) = \sum_{s=0}^{S} X^{(s)}(a)t^s, \quad S \in \mathbb{N}\,. \tag{5.10}$$

The reminder then takes the form

$$X(a,t) - X^S(a,t) = \sum_{s=S+1}^{\infty} X^{(s)}(a)t^s, \tag{5.11}$$

which is majorized in a formal way in L^2 as

$$\left\|X - X^S\right\|_{L^2} \leq \sum_{s=S+1}^{\infty} \|X^{(s)}\|_{L^2} t^s\,. \tag{5.12}$$

Subsequently, if the radius of convergence ϱ in eq. (5.4) is positive, then there exists a constant $c > 0$ such that the RHS sum in estimate (5.12) is bounded, for $t < \varrho$, by

$$\sum_{s=S+1}^{\infty} \|X^{(s)}\|_{L^2} t^s \leq c \sum_{s=S+1}^{\infty} \left(\frac{t}{\varrho}\right)^s = c\left(\frac{1}{1-r} - \frac{1-r^{S+1}}{1-r}\right) = c\frac{r^{S+1}}{1-r}, \tag{5.13}$$

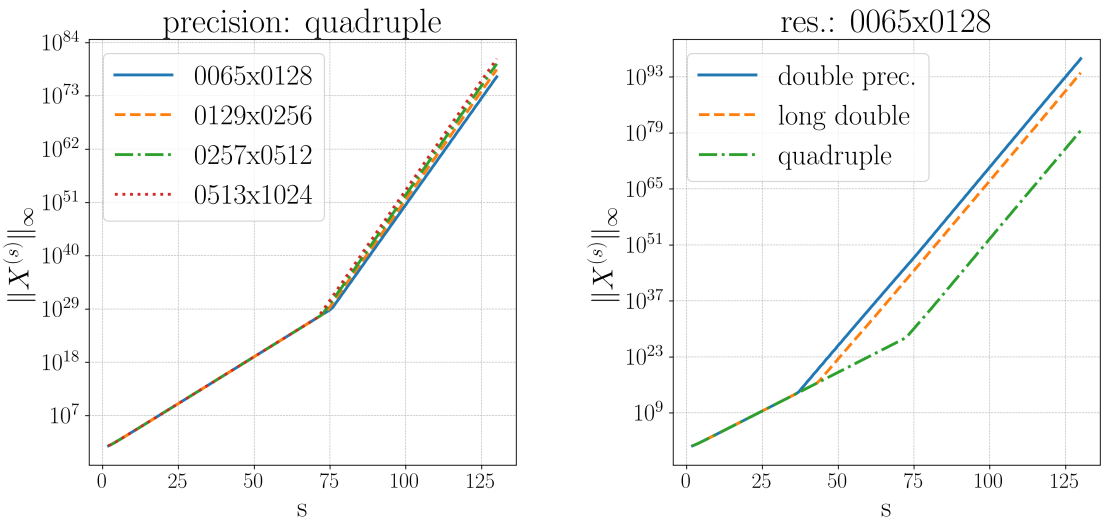
with $r = t/\varrho$. However, without knowing neither the constant c nor the radius ϱ, this estimate might not be of great use for our purposes.

5.2 Numerical Estimation of the Radius of Convergence

If it is momentarily not possible to know the radius of convergence analytically, then one may try to approximate it numerically. A straight forward attempt can be made in computing a high number S of time-Taylor coefficients and use extrapolation techniques to approximate the limits in eq. (5.4) or (5.5). This method is emphasized here for a stationary flow from [81], which is discussed in detail in Section 6.3. The regarded stationary vorticity in the periodic cylinder $D(1, 2\pi)$ reads

$$\begin{aligned}
\omega^r(r,z) &= \sqrt{c_1^2+1}\sin(z)J_1\left(rc_1\right),\\
\omega^\alpha(r,z) &= \left(c_1^2+1\right)\cos(z)J_1\left(rc_1\right),\\
\omega^z(r,z) &= c_1\sqrt{c_1^2+1}\cos(z)J_0\left(rc_1\right),
\end{aligned}$$

where $r \in [0,1], z \in [0,2\pi]$ and c_1 is the first positive root of the Bessel function J_1. This information suffices to start the recurrence mechanism described in the first part of Section 2.6. For testing purposes, the time-Taylor coefficients of the trajectories of the above



(a) **(b)**

Figure 5.1: Order dependent growth of the time-Taylor coefficients $X^{(s)}$ in (a) various resolutions and (b) various working precisions. A break appears in the linear growth due to cancellation effects. The lines overlap almost exactly until the appearance of the first break.

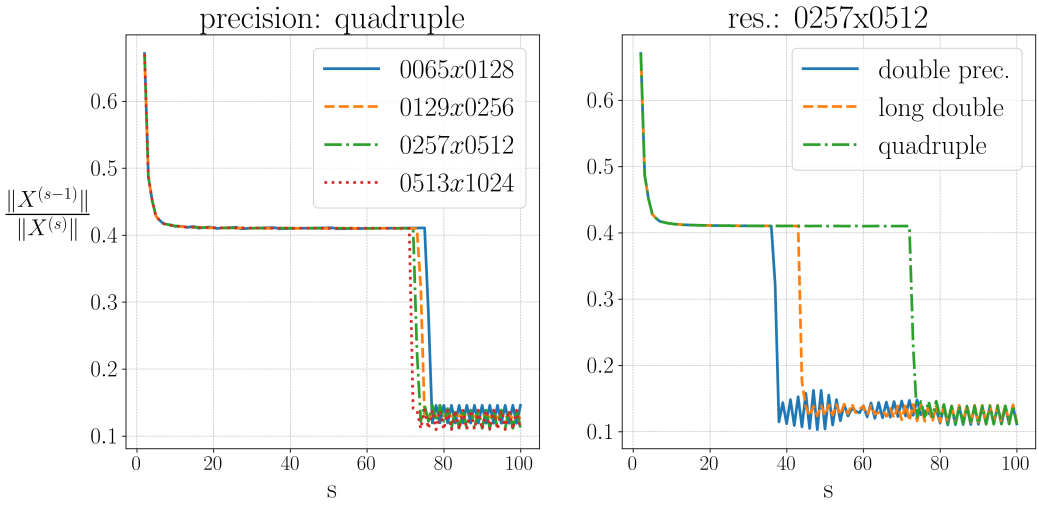
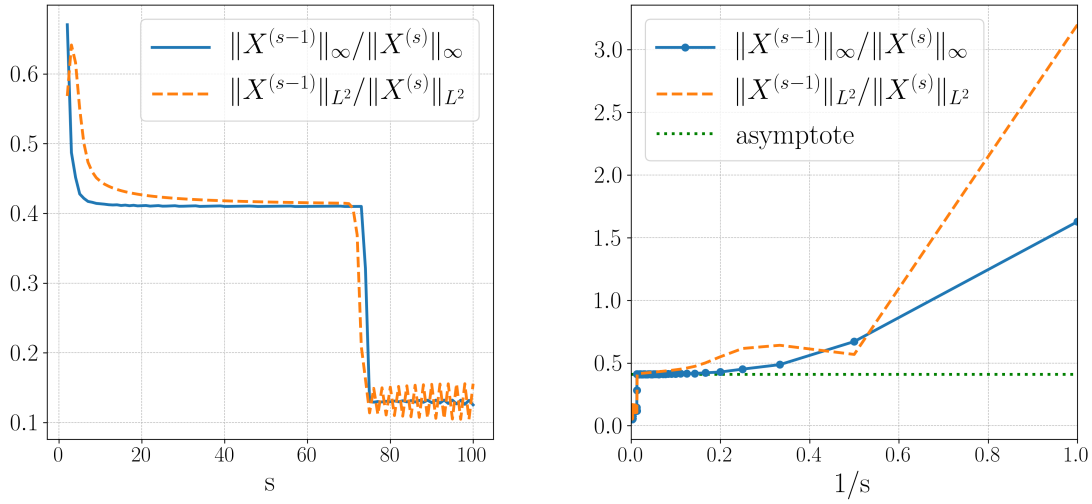


Figure 5.2: Due to the growth change in the time-Taylor coefficients, seen in Figure 5.1, also the ratios of the subsequent time-Taylor coefficients show breaks. Those are even more prominent and show little dependence on the resolution, but their locations vary strongly with working precision. The curves overlap to at least $s = 30$, from which a possible limit at around 0.4 can be inferred.

flow have been computed to $S = 350$ in double, long double, and quadruple precision on various resolutions, see Section 6.1 for an explanation of the used precision types. The radii ϱ_∞ and ϱ_2 were approximated by the ratio formula eq. (5.5). The Cauchy-Hadamard formula was discarded, because the root in eq. (5.4) becomes quickly computationally expensive and inaccurate. It was realized that a seemingly unnatural break in the growth behaviour of the coefficients takes place and therefore, orders of up to $S = 100$ are sufficient. In fact, it appears that the values of $\|X^{(s)}\|_\infty$ grow linearly with s at first, but show a change, or break, in their behavior eventually. After this break, the mentioned values continue to grow linearly with s, but with a steeper inclination, see Figure 5.1. We observe in the aforesaid figure



(a) **(b)**

Figure 5.3: (a) An attempt to approximate the radii of convergence ϱ_∞ and ϱ_2 via the ratio formula (5.5) with a grid size of 65×128 points. The break in the components growth is clearly visible and sets in even earlier in the L^2 norm. (b) A Domb-Sykes plot, which maps the ratios as a function of $1/s$. In both plots the last values before the breaks align almost horizontally to point to a radius $\varrho_\infty \sim 0.4$.

that this break does not dependent on the resolution, even though for higher grid sizes the break appears slightly earlier due to rounding errors. More prominent is the break shift with changing working precisions. In quadruple precision, the break shows roughly at order $2s'$, if s' is the order at which the break shows in double precision. This behavior strongly points to cancellation effects, which is not surprising given the magnitude of the values $\left\|X^{(s)}\right\|_\infty$ for this flow. In Figure 5.3 (a), the quotient of eq. (5.5) is plotted as a function of s and shows a drop of values once the break point is reached. Nonetheless, a potential limit can be found by extrapolating the part of the graph that lies before the break. In the particular case of the above stationary flow, this limit lies at, or slightly above, 0.4. Moreover, Figure 5.3 indicates that the L^∞-norm is for this flow better suited than the L^2-norm, because the corresponding graph displays a more stringent convergence to a possible limit. A validation for the correctness of the plotted values can be seen in the overlap of the ratio curves in several resolution runs with varying working precisions as visualized in Figure 5.2. Thus, the chosen example flow allows for a determination of the convergence radius of the sum in eq. (5.3) by straight forward computations of the ratios in the limit (5.5) to around $S = 30$. Certainly, this strong asymptotic behavior is a feature of the chosen flow and not a general characteristic of an incompressible Euler flow. In principle, a clear asymptotic behavior can manifest very slowly, and at orders which are out of reach of the recurrence algorithm, even in quadruple precision, because of the cancellations.

In the case where a horizontal line may not be drawn clearly at a possible asymptotic level, another plotting method might be more useful. Introduced in [42], the Domb-Sykes plot maps the ratio $X^{(s-1)}/X^{(s)}$ to the inverse order $1/s$, an example can be found in Figure 5.3 (b). An asymptote will eventually show as a linear function at small $1/s$ and the intersection of that function with the ordinate axis exhibits an approximation of the convergence radius. An advantage of this plotting method lies in the density of the values $1/s$ for moderately high truncation numbers S, which simplifies the fitting of an asymptote and produces a definitive

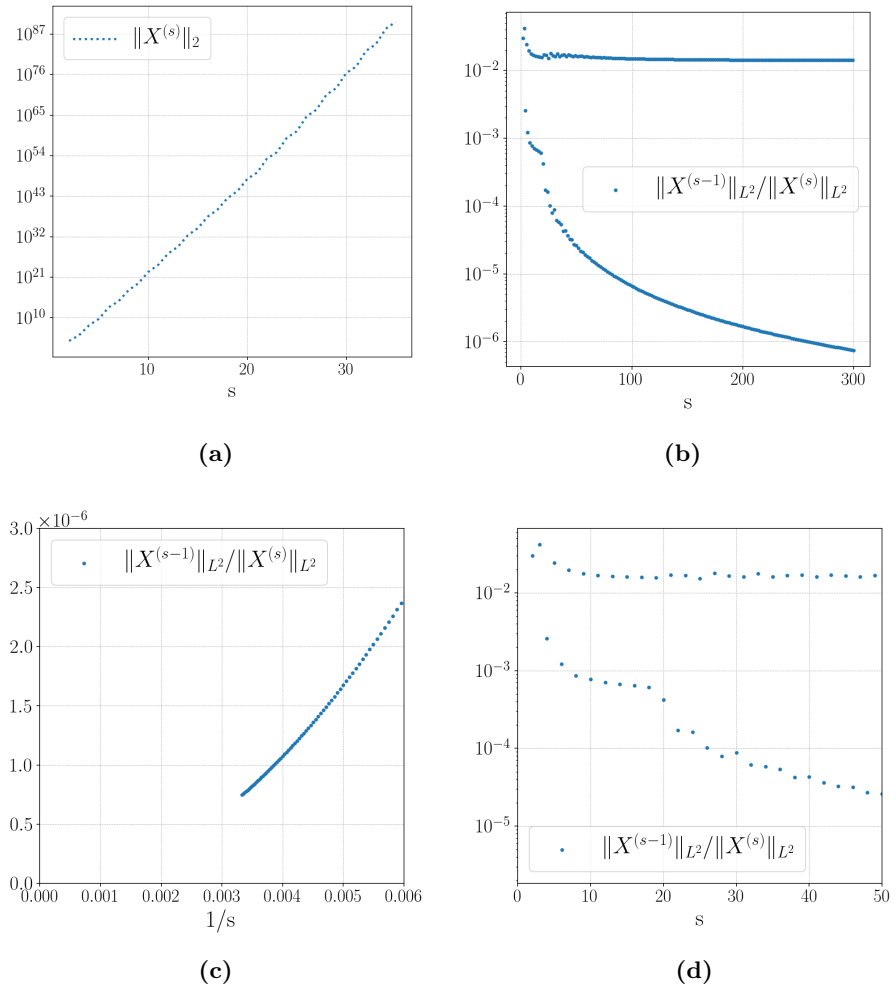


Figure 5.4: The growth of the time-Taylor coefficients, computed in a quadruple precision run, is plotted in (a), together with the ratio rule in the L^2-norm in (b), and a Domb-Sykes plot in (c). Panel (d) shows a zoom into the region of panel (b), where the ratio shows a drop. It seems improbable that the coefficients are correctly computed after $s = 20$. A reason for this may be found in the large deviation between the maximal values of the coefficients, which leads to strong cancellation errors in the recurrence sums. In this respect, note that the drop in panel (d) appears at an order at which the range of quadruple precision numbers (~ 36) has just been depleted in the maximal value plot in panel (a).

value for an approximation of ϱ as an intersection of a linear function with an axis. In our particular case, we must simply pay attention that this extrapolation is done before a break appears, i.e. before the cancellations take place. Furthermore, since the here investigated flow is stationary, the radius stays fixed and has to be found only once in the beginning. However, a re-application of the above methods after a few time-steps can indicate the robustness of the implementation and may be used for validation purposes. This validation technique might be more applicable to non-stationary flows, as in the stationary case one can always directly obtain the relative errors between the calculated and the known flow fields.

In the current implementation of the CLA, the search for possible convergence radii of non-stationary test flows is done only once to have an idea of possible values and to test the robustness of the implementation of the recurrence formalism, especially in the first time-steps. We will see later that one should not choose a time-step too close to an approximated radius of convergence even if it is exact. The reason for this lies in the error cascade that has

already been discussed in Section 2.6. The recurrence mechanism implies a re-insertion of functions that are applications of Calderon-Zygmund operators of order zero to former RHS functions of some Poisson problems. The latter comprise errors of some magnitude, which will enter in the new RHSs of the Poisson problems, on which again the aforesaid operators are applied and so forth. Simplified, one can say that the error in the time-Taylor coefficients accumulates, on top of rounding errors, as in continuous re-applications of the algorithm for the zero-order Calderon-Zygmund operators. This, however, is usually compensated by time-steps which are fractions of the convergence radius, because a Taylor series in a finite precision computation only needs a finite number of terms to arrive at the corresponding number of significant digits. The lower the time-step the fewer terms are needed. In other words, the time-step acts like a pair of scissors, which cut off the accumulated error in each summation term. Here a simple example to emphasize how numerical errors in the time-Taylor coefficients are not showing in the final sum for a given hypothetical precision of 7 significant digits (roughly single precision). Suppose we are given a power series

$$\sum_{s=0}^{\infty} a_s t^s, \tag{5.14}$$

of which we know that the entries verify $a_s = 1$ and thus the convergence radius is $\varrho = 1$. Assume further that a numerical truncation is computed somehow to 7 coefficients, producing the erroneous values

$$\begin{aligned} &\tilde{a}_0 = 1.000000, \quad \tilde{a}_1 = 1.000004, \quad \tilde{a}_2 = 1.000065, \quad \tilde{a}_3 = 1.000567, \\ &\tilde{a}_4 = 1.009567, \quad \tilde{a}_5 = 1.049567, \quad \tilde{a}_6 = 1.135648\,. \end{aligned}$$

Hence, the last computed entry is accurate to only 1 digit and a value of t close to the convergence radius would produce an overall accuracy of only 1 digit as well. However, if the coefficients are summed with $t = 0.1$, then the truncated sum takes the value 1.111111, which is accurate in the working precision, which means accurate up to additions of numbers with modulus around machine epsilon (here 10^{-7}). The point is that finite precision operations can only operate in a certain range and only this range has to be controlled, for example via small time-step insertions in the time-Taylor series. To allow for a larger time-step in our time-Taylor series, it is, therefore, crucial to minimize the error in the Calderon-Zygmund operators of order 0. This insight explains the efforts that have been undertaken in Section 3.3 (and Section 3.3.3 in particular) to solve directly for the second derivatives of the solutions to the Poisson problems.

5.3 Fixed Time-Steps

For non-stationary flows, the radius of convergence changes in dependence of the regularity of the underlying flow. This implies that a fixed time-step can only be chosen if it is smaller than the minimal convergence radius in a given time span. We have an interest to choose a small time-step also to assure that the fluid particles don't move too far off the regular grid, so that the two-dimensional scattered interpolations, discussed in Chapter 4, give accurate results.

Remark. The aforementioned time-steps, which must stay below a minimal radius of convergence in a given time span, may still be quite large compared to those in traditional explicit time integration schemes, such as Runge-Cutta in a purely Eulerian viewpoint. The latter are bound by the well known Courant-Friedrichs-Lewi (CFL) condition, see [32], which have

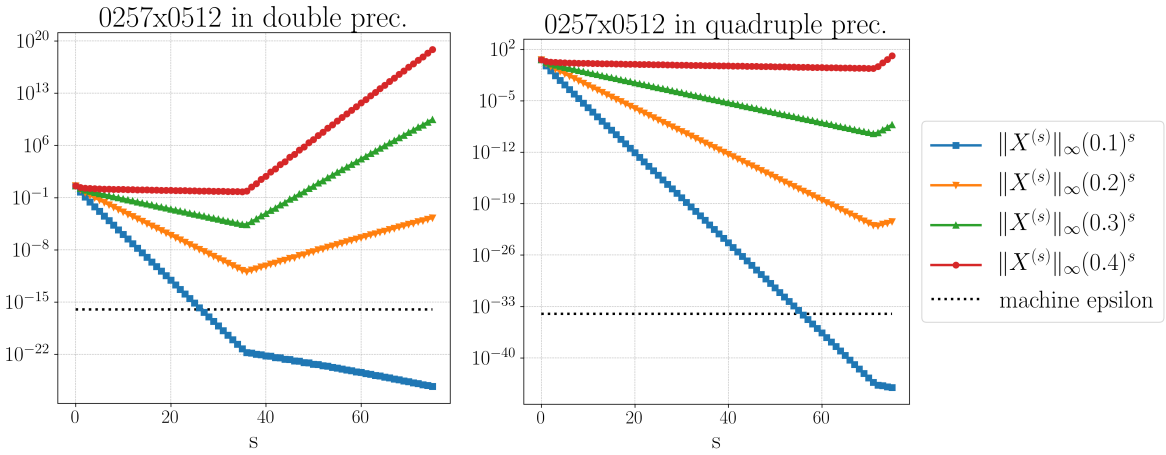


Figure 5.5: Plots of the values $\|X^{(s)}\|_\infty(\triangle t)^s$ as a function of s for time-steps $\triangle t = 0.1, 0.2, 0.3, 0.4$. The left panel visualizes a double precision computation. The first terms are of magnitude 1, thus, only the smallest inserted time-step achieves a precise summation, and the last terms (after $s \sim 30$) can be neglected due to their low magnitude. The same conclusions may be drawn from the quadruple precision computation in the right panel. Due to cancellations in the coefficient computations, higher time-steps cannot reach the prescribed precision goal.

a dependence on the minimal mesh spacing of the underlying grid. The time-step in the CLA is only bound by the radius of convergence of the time-Taylor series of X, which does not depend on the underlying mesh. The absence of a CFL condition in our (semi-)Lagrangian time-stepping scheme allows us to take the same time-steps in any resolution. This is particularly fortunate for high resolutions, where mesh spacing is tiny, e.g. at the boundary of our cylinder in the Chebyshev-Fourier grid. Small mesh spacing forces CFL restricted time integration schemes to small time-steps, which increases the overall numerical complexity. Moreover, in the CLA, small resolutions allow, in principle, to choose the time-step even larger than for higher resolutions, if accuracy on the common points of a lower and a higher resolution is comparable, i.e. when the flow is accurately represented by low order spectral series. In theory, a larger time-step cannot be chosen on lower resolutions, the actual convergence radius is independent of resolution, but, in practice, one needs to ensure only that the time-step is chosen small enough such that the truncated time-Taylor sums are accurate in working precision on the grid points. This can produce a possibly larger time-step because fewer data sites have to be accounted for. Since the performance of scattered integration schemes deteriorates with smaller resolutions, one could use the refinement methods examined in Section 4.2 for the interpolations at hand. The latter interpolate accurately from a distorted Lagrangian mesh to a regular Eulerian grid even in low physical resolutions, if the spectral resolution suffices, as visualized in Figure 4.7.

If the radius of convergence drops below an initially set time-step or even approaches zero in an envisaged time span, e.g. because of strong regularity loss of the flow, then this can be observed in the growth of the time-Taylor coefficients $X^{(s)}$ with respect to the order s. Reconsider the strongly simplified example from above (eq. (5.14)) and suppose that, at a later time-step, the coefficients in eq. (5.14) are given by $a_0 = 1$, $a_s = 2s$, $s > 0$, which reduces the convergence radius to $\varrho = 0.1$. If we now insert the time-step from before, i.e. $t_1 = 0.1$, then we have $a_6 t_1^6 = 1.2 \times 10^{-6}$ and thus the relative error $|a_0 - a_6 t_1^6|/|a_0|$ is larger than machine epsilon. The latter fact is visible in a computation and can be tracked, it clearly indicates, that either the truncation order S is too small, or the time-step too large. See Figure 5.5 for plots of the norms of the summation terms with respect to the corresponding order. This

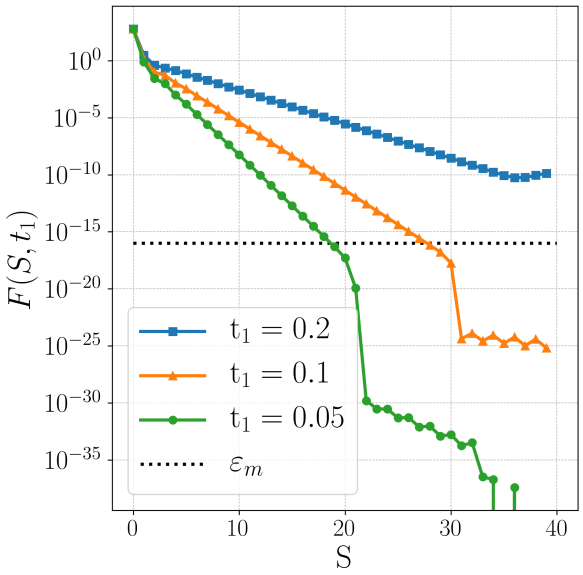


Figure 5.6: Plot of the function F defined in (5.17) for different time-steps in dependence of the orders s. The computation may be stopped when the values of F drop below machine epsilon ε_m, here in double precision and indicated by a dotted line in the plot. The break of the coefficient growth is clearly visible, but appears after the criterion has been met. All higher order terms, after the criterion has been met, do not alter the summation result of the time-Taylor series in a double precision computation.

thinking applies mostly to an asymptotic and monotonical growth of the coefficients entries, which has been observed in all the investigated flows. In a CLA simulation the coefficients a_s take the values of $\left\|X^{(s)}\right\|_\infty$, or any other L^p norm, and the above insights are equivalently applied in higher working precisions.

Here, a description of what exactly has been implemented in the current CLA code concerning fixed time-steps. At the beginning, a user defined truncation order S is set and a suitable initial time-step t_1 is given, which lies below an estimated radius of convergence, but also implies minimal relative errors in the conserved quantities, see Section 6.2. The behavior of the terms $\left\|X^{(s)}\right\|_\infty t_1^s$ is tracked and the computation of new coefficients after $X^{(\sigma)}$, $\sigma < S$, is stopped, if there holds

$$\varepsilon_m > \frac{\|b_{\sigma-2}(t_1) - b_{\sigma-1}(t_1)\|_{max}}{\|b_{\sigma-2}(t_1)\|_{max}} \geq \frac{\|b_{\sigma-1}(t_1) - b_{\sigma}(t_1)\|_{max}}{\|b_{\sigma-1}(t_1)\|_{max}}, \tag{5.15}$$

where

$$b_\sigma(t_1) = \sum_{s=0}^{\sigma} X^{(s)} t_1^s, \tag{5.16}$$

and $\|\,.\,\|_{max}$ denotes the maximal value of the l_2-norms of the three components on the underlying discrete grid. The number ε_m denotes machine epsilon, e.g. 2.22×10^{-16} in double precision. In fact, this method also accounts for alternating signs in the entries of $X^{(s)}(a)$, which is internally given by a two dimensional array. The additional numerical complexity is quite low, since the sums are computed anyways. All the investigated flows showed a moderate monotonical growth in the coefficients. One may also set the focus on the time derivative of a series, which is crucial to calculate the new velocity field via the characteristic

equation. Thus, the following criterion may be taken instead of that in (5.15),

$$\varepsilon_m > F(\sigma, t_1) := \frac{\|b'_{\sigma-1}(t_1) - b'_\sigma(t_1)\|_{max}}{\|b'_{\sigma-1}(t_1)\|_{max}}, \tag{5.17}$$

where b'_σ denotes the temporal derivative of b_σ, i.e.

$$b'_\sigma(t_1) = \sum_{s=1}^{\sigma} sX^{(s)}t_1^{s-1}\,. \tag{5.18}$$

Any fraction of ε_m in eq. (5.17) may be taken too, $\varepsilon_m/2$ is implemented as an alternative to eq. (5.15). This requirement ensures that the derived time-Taylor series is computed precisely and implies eq. (5.15) for monotonical growth of the coefficients on the grid points as long as t_1 is sufficiently small. Figure 5.6 shows a plot of the criterion function $F(S, t)$. The calculation of new coefficients is suspended once the criterion is matched. Should the described situation not appear before the maximal order S is reached, then the time-step is halved until the conditions are met. This is only a first instance to control the validity of a time-step. After this procedure the time-step may be further reduced, if other control mechanisms set in. Such mechanisms are for example the validation of total kinetic energy or total helicity conservation (among others that are described in Section 6.2). In this method, a truncation order should not be set too high, because the fact that the relative errors in eq. (5.17) are below machine epsilon does not imply that the sum produces accurate results with respect to the flow. It only suggests that any additional terms in the time-Taylor sum would not alter the summation result itself in the working precision. For short time spans, it is usually possible to set an order and a time-step that is maintained over the whole time span.

5.4 Adaptive Time-Stepping

Using the insights from the preceding section, it is possible to adapt a time-step at every iteration in an automated manner. For instance, the time-step can be adapted until a specific precision requirement is fulfilled precisely for a fixed truncation order S. Such a strategy can be seen as an opposite to the previously discussed fixed time-stepping, where the computation of further coefficients is suspended, once a precision criterion is met. Here, the number of computed coefficients is constant and the time-step is adapted to fulfill such criteria. One requirement can be for instance the criterion eq. (5.17), where σ gets replaced by the fixed order S. Here, the implemented method:

Algorithm 5.1 (adaptive time-stepping)**.**

- *compute the coefficients* $X^{(s)}$, $s = 0, \ldots, S$, *for fixed* S.
- *In the first iteration, reduce the time-step, departing from an initial guess until the precision criterion is met closely.*
- *In a later iteration, replace the initial guess by the time-step that has been used in the preceding iteration and reduce it, if necessary, until the criterion is met.*

This adaptation of a time-step can be quite demanding in resources especially in high resolutions or high precision because for every decrement of the inserted time the truncated sums have to be recalculated to check the fulfillment of the precision criterion. The time-step

is reduced by small fractions of an initial value, currently as $t_1 - t_1/20$, which is meant to keep the time-step as large as possible to meet a criterion more or less exactly. Surely, the time-step could also be halved to reduce the number of summations that are needed and so to meet the criterion faster, which reduces the numerical complexity. The criteria should mainly be used to verify if the precision range is exceeded, i.e. that the last term does not alter the summation result in the truncated time-Taylor series in the given working precision. A last term in the truncated time-Taylor sum that only alters the relative error with respect to the sum minus this term by a number below machine epsilon can be neglected and, if asymptotic growth of the coefficients is assumed, all higher order terms as well. Note that it is important to use the relative error in eq. (5.17) instead of only checking that the last term has entries of modulus below machine epsilon ε_m. Indeed, the first coefficients could start at small numbers, even close to ε_m, then the last term's entries would need to be below ε_m^2 in modulus to drop out of the range of significant digits.
Tables 5.1 and 5.2 list the computed time-step in dependence of the truncation order S for the stationary flow on 129×256 grid points, the appearing Poisson problems were solved with the Bessel-Chebyshev solver. In addition, the tables, show next to the values of $F(S, t_1)$ from criterion (5.17), also the relative errors of the velocity and vorticity fields on the particle trajectories, which originate from the Chebyshev-Fourier grid $D^{\#}$. More precisely, after the trajectories have been advanced by one time-step t_1 to yield $X(D^{\#}, t_1)$, the velocity field, here abbreviated by $v^{\#}(X(D^{\#}, t_1), t_1)$, is computed via eqs. (2.28) to (2.30). Subsequently, the values of the particle trajectories $X(D^{\#}, t_1)$ are inserted into the analytically known stationary velocity field v to calculate the relative error

$$\delta_v = \frac{\left\| v(X(D^{\#}, t_1), t_1) - v^{\#}(X(D^{\#}, t_1)) \right\|_{l^2(D^{\#})}}{\| v(X(D^{\#}, t_1), t_1) \|_{l^2(D^{\#})}} . \tag{5.19}$$

The relative error δ_ω for the vorticity field is obtained in the same manner in using eqs. (2.38) to (2.40) and the known stationary vorticity function.

On the one hand, higher orders allow for larger time-steps and reduce their total number until a predefined finite time, but may also introduce larger errors and could potentially limit the possible total integration period. On the other hand, a lower order implies smaller time-steps, but increases accuracy in the interpolations and reduces the numerical complexity of one iteration, but a larger number of iterations is needed to arrive to a predefined final time. Additionally, more iterations imply more scattered interpolations, which increase the total error faster. Indeed, in sight of the test results from later chapters, the reality is, that larger time-steps are more accurate in a given time span.

Table 5.1: The time-steps for the stationary flow on a grid size of 129×256 points in the table were determined by the adaptive method described above for different truncation orders S. Since criterion (5.17) is used, the values of F are given in the second row. The relative error of the velocity and vorticity field on the distorted grid (see eq. (5.19)) are given in the third and fourth row respectively.

S	4	6	8	10	12
t_1	9.51×10^{-7}	1.66×10^{-4}	1.51×10^{-3}	4.88×10^{-3}	1.11×10^{-2}
$F(S, t_1)$	9.96×10^{-17}	1.04×10^{-16}	1.09×10^{-16}	7.16×10^{-17}	1.03×10^{-16}
δ_v	4.48×10^{-16}	5.48×10^{-16}	5.99×10^{-16}	6.52×10^{-16}	6.92×10^{-16}
δ_ω	4.63×10^{-16}	5.66×10^{-16}	6.18×10^{-16}	6.69×10^{-16}	7.13×10^{-16}

Table 5.2: Same as in Table 5.1 for higher truncation orders S.

S	15	20	25	30	35
t_1	2.31×10^{-2}	4.78×10^{-2}	7.54×10^{-2}	9.75×10^{-2}	1.19×10^{-1}
$F(S, t_1)$	7.67×10^{-17}	5.60×10^{-17}	7.94×10^{-17}	3.51×10^{-17}	3.57×10^{-17}
δ_v	7.44×10^{-16}	8.92×10^{-16}	1.06×10^{-15}	1.21×10^{-15}	1.34×10^{-15}
δ_ω	7.63×10^{-16}	8.20×10^{-16}	9.57×10^{-16}	8.96×10^{-16}	1.02×10^{-15}

5.5 Advance by Analytical Continuation

Analyticity is one of the strongest properties a function may have and should be exploited as far as possible. As a matter of fact, the radius of convergence is only bound by singularities, or poles, which lie in the complex plane, but not necessarily on the real axis. Furthermore, if a function is analytic with a positive radius of convergence, then this function is locally analytic in every point in the disc of convergence. In other words, the function may be developed into a converging power series around any point whose modulus is smaller than the convergence radius, and the new radius of convergence can even be larger than the original. This well-known technique is called *analytic continuation* and presents here a possibility to advance the trajectories without the need to re-interpolate from the Lagrangian to the original grid. The flow fields will always be known on the evolving trajectories. Moreover, the coefficients have to be computed only once until a predefined truncation order, i.e. the Poisson problems from Section 2.5 have to be solved only once. In this method, the flow fields are only known on the Lagrangian trajectories. Some precision is lost due to high temporal derivative computations, but the speed is striking. To be more precise, let us recall the time-Taylor series of the particle trajectories developed around a time t_0, here in the notation

$$X(a, t; t_0) = \sum_{s=0}^{\infty} X^{(s)}(a)(t - t_0)^s, \quad \text{for } |t| < \varrho_0. \tag{5.20}$$

There holds

$$X^{(s)} = \frac{1}{s!} \partial_t^s X(t) \mid_{t=t_0}, \tag{5.21}$$

where the spatial argument has been left out for convenience and ϱ_0 is a formal convergence radius. The above information on the structure of a time-Taylor series, is sufficient to state a formal analytic continuation to $t_1 < \varrho_0$ by developing $X(t)$ around t_1. This is simply achieved in writing

$$X(t; t_1) = \sum_{s=0}^{\infty} \frac{1}{s!} \partial_t^s X(t; t_0) \mid_{t=t_1} (t - t_1)^s = \sum_{s=0}^{\infty} \bar{X}^{(s)} (t - t_1)^s, \quad \text{for } |t| < \varrho_1, \tag{5.22}$$

where eq. (5.20) implies

$$\bar{X}^{(s)} := \frac{1}{s!} \partial_t^s X(t; t_0) \mid_{t=t_1} = \sum_{k=s}^{\infty} \frac{k!}{(k-s)!s!} X^{(k)} (t_1 - t_0)^{k-s}. \tag{5.23}$$

Numerically, the sums in eqs. (5.20) and (5.22) can be approximated by partial sums to a finite order S and the freshly obtained coefficients can then be used to find the new convergence radius as in Section 5.2, or simply a new time-step via the precision criteria eqs. (5.15) and (5.17). As mentioned before, the sums loose accuracy as the higher temporal derivatives (5.23) loose precision. Inaccuracies of the truncated series approximation enter directly into

the new coefficients. Remember, the Taylor series can be computed precisely for a good time-step and a given precision, but this does not necessarily imply that the result is accurate in the way that the error to the real solution is small. Thus, once more the precision of the Calderon-Zygmund operators of order zero is essential. The less errors arise in the derivatives on the inverse Laplacians in the problems (2.73) and (2.74) the better the truncated analytic continuation method will work. Here, the algorithmic form of the method.

Algorithm 5.2 (analytic continuation)**.**

- *compute the coefficients* $X^{(s)}$*,* $s = 0, \ldots, S$*, for fixed* S
- *find a suitable time-step* $\triangle t$
- *compute the new coefficients approximately as*

$$\bar{X}^{(s)} = \sum_{k=s}^{S} \frac{k!}{(k-s)!s!} X^{(k)} (\triangle t)^{k-s}, \quad 0 \leq s \leq S\,. \tag{5.24}$$

Furthermore, an application of the described method is much easier in Cartesian coordinates, because the velocity field can directly be written as a (truncated) time-Taylor series, i.e. for the first component $v_1(X(a,t),t) = \sum_{s=1}^{S} sX^{(s)}(a)(t-t_0)^{s-1}$. In cylindrical coordinates, the latter is only possible for the vertical components of the velocity and vorticity fields. The remaining components are multiplications and divisions of the components X^r and X^α of X, which can, in principle, be transformed into a convergent power series. Here, it is easier to apply the analytic continuation directly to the components X^r, X^α and X^z and form the non-vertical components of the velocity and vorticity in the usual way (i.e. as in eqs. (2.28), (2.29), (2.38) and (2.39)). This method is implemented in the current CLA code but is still experimental.

Chapter 6

Testing the CLA

6.1 Overview of the Complete Implementation

This section gives a complete overview of the Fortran implementation of the Cauchy-Lagrange algorithm in a cylindrical domain with boundary and lists the parameters that can be modified. Those parameters were thoroughly tested on various flows with results presented in the next sections.

Precision. The working precision follows the IEEE standards [1, 2] and can be set in a make file, which contains compilation instructions. The precision must be fixed at compile time. Here are the options:

- 8 Byte double precision, presenting numbers that lie between 10^{-308} and 10^{+308} with roughly 16 significant decimal digits
- 10 Byte long, or extended, double precision[1] that accounts for numbers between 10^{-4932} and 10^{+4932} with roughly 20 significant decimal digits
- 16 Byte quadruple precision[2], handles numbers between 10^{-4932} and 10^{+4932} and roughly 34 significant decimal digits[3]

The make file selects the Intel compiler `ifort` for double and quadruple precision. The GNU compiler `gfortran` is used for long double precision.

Parallelization. The CLA implementation comprises a hybrid Parallelization that uses both MPI and OpenMP interfaces, see [26, 36] as references. While MPI instances span computations across several computational nodes, OpenMP instances span them over the CPUs (central processing units), or cores, in one node. For a discrete and finite set of problems, the implemented hybrid parallelization organizes the involved computation first by distributing equal chunks of the set to the different nodes, which then subdivide their chunk into local portions that are distributed and solved by the CPUs on the node. These instances together with the number of computation nodes and number of CPUs are also set in the make file at compile time.

[1] This is a specification of the GNU compiler and is for instance not available for Intel compilers.

[2] Available in GNU and Intel compilers, Intel refers to quadruple precision as long double, but here we will always call it quadruple to avoid confusion with the GNU standard for 10 Byte numbers.

[3] The exponent range is the same for long double and quadruple precision number formats, they reserve 15 bits.

The remaining parameters are set at run time and most of them can be modified even within the running program. The initial parameters are set in a control file with the following content. The terms in the brackets refer to the type of the parameter.

```
&parameters
! 2 -> 2*pi | 6 -> 1/6
nheight                = (int)
dims                   = (int)

dimr                   = (int)
dimz                   = (int)
smax                   = (int)

! interp_method:
!| 0-none | 1-timetravel | 2-cascade-burk-cubic | 10-alglib_spline
!| 4-cascade-lagrange | 5-cascade-bspline-boor
!| 6-padaptive-cascade-lagrange | 7-padaptive-cascade-bspline-boor
!| 8-extensive-padaptive-cascade-lagrange | 9-extensive-padaptive-cascade-bspline-boor
interp_method          = (int)
interp_order           = (int)
ref_fact_pow_r         = (int)
ref_fact_pow_z         = (int)

nb_tmstp               = (int)
tmstp_incr             = (float)

nb_tmadv               = (int)
tmadv_incr             = (float)

use_hsmax              = (boolean)

nprint_to_files        = (int)
ncreate_plots          = (int)
nprint_velo_vort       = (int)

! npslv: 1->Poisson-Galerkin, 2->Poisson-Besselmat solver;
npslv                  = (int)

! nbesseltst chooses the test function, the others are 0:off, 1:on
nbesseltst             = (int)
nstationary            = (int)
ntstswitch             = (int)
nluohou                = (int)
ngenerique             = (int)

nsetdivbndry           = 1
dealen                 = (int)
! dealen: padding, 0: no dealias, 1: 2*dimz/3, 2: 2*dimz, 3: implicit dealiasing via fftwpp

tmstp_lmt              = (float)
enloss_lmt             = (float)
helloss_lmt            = (float)
comp_add_lmt           = (float)
particle_out_lmt       = (float)

verbose                = (boolean)
```

The most important of the above parameters are explained below.

Periodicity length, grid size, and Poisson solvers. The parameter `nheight` chooses between a periodicity length of 2π and $1/6$, the latter is used to investigate the flow, which has been reported to develop a finite-time singularity in reference [83]. The Poisson solver is chosen by `npslv`. If `npslv`$= 1$, then the Poisson problems get solved by the Chebyshev-Galerkin method from Section 3.3.2, it can handle any other periodicity length and, in principle, any grid size with `dimr` radial Chebyshev points and `dimz` vertical Fourier points. For `npslv`$= 2$ the Bessel-Chebyshev method, i.e. Algorithm 3.2, is used to solve the appearing Poisson problems. The needed cubic matrices for this method were computed up to `dims` $= 513$, in one dimension, for $L = 1/6$ and `dims` $= 1025$ for $L = 2\pi$. The parameter setting `dims` $= 513$, resp. 1025, implies a maximal grid size of 513×1024, resp. 1025×2048, points. The matrices are read into the program in quadruple precision in the binary format and are rounded to working precision. Matrices with 65^3 and 257^3 entries are also available and selected by `dims`, which decreases the reading time in test scenarios, even though the largest matrices only need some seconds to arrive in the RAM (~ 2 sec. on a modern architecture). Because of the nesting properties of the underlying discretisation, any sub grids of the form $(2^n + 1) \times 2^m, n, m = 1, 2, \ldots$ can be extracted from those matrices, where $2^n + 1, 2^{m-1} \leq$ `dims`. If the Chebyshev-Galerkin solver is used with any other precision than double, then the execution is aborted. The package that is used for solving the linear systems, namely LAPACK (see [4]), provides only double precision support. One could recompile this open-source package into a different working precision, but this is not recommended, since the package code contains many tuning parameters explicitly for double precision computations. Instead, it was envisaged to implement directly the solving algorithm from the given reference which has linear complexity. The corresponding algorithm is given in [107] for the Dirichlet problem, but still has to be found for the Neumann problem, which is the reason for the delayed implementation.

Truncation order. The truncation order S is controlled by the parameter `smax`. Whether this truncation order is always reached in a computation of the coefficients is controlled by the boolean parameter `use_hsmax`. If `use_hsmax = t` (true) then the computation is aborted at the order where the precision criterion from eq. (5.15) is met. The time-Taylor series is summed only until the internal order `hsmax`, where the criterion has been met. If this mechanism does not abort before `smax` is reached, then the time-step is halved until the latter criterion is met at `smax`. If `use_hsmax = f` (false), the coefficients are always calculated until the order `smax`, the time-step might then be modified by other control mechanisms. Furthermore, to avoid numerical differentiation and their underlying errors as far as possible, all the coefficients in (2.94) on page 20 are stored and explicitly summed to the maximal order `smax` or `hsmax`.

Control mechanisms. As the energy and helicity are conserved in an incompressible Euler flow (see e.g. [50]), their change is used as a control instance. The total kinetic energy loss and total helicity loss can be bound by `enloss_lmt` and `helloss_lmt` respectively and are usually set to 10^{-5} (close to single precision machine epsilon) or $10^{-3} = 0.1\%$. A time-step gets halved if those limits are exceeded, while the minimal possible time-step is set to `tmstp_lmt`. The simulation is aborted if the time-step drops below `tmstp_lmt`. Another bound is the `particle_out_lmt`, it controls how far a boundary particle may leave the boundary due to numerical errors. It is important to remark here, that the boundary value of a trajectory as well as the divergence of the velocity field are set to one and zero respectively after every time-step. The default for all the computations in this work sets those values explicitly, but this choice, being debatable, can be modified by setting `nsetdivbndry=0`. The reason for

this choice is the aim of reducing the overall error, the boundary value and the divergence are corrected to their true values. Also, the divergence term would need to be computed via numerical differentiation, which distorts the results, especially for a shifted Chebyshev spectral discretization.

Interpolation parameters. The user chooses moreover which interpolation method from Section 4.1 is employed. The options for `interp_method` are listed in the control file above and are mostly self-explanatory, except the time-travel method and the p-adaptive cascades. The time-travel method evaluates the trajectories at negative times, which yields the original positions of the particles in a stationary (!) flow that will arrive exactly on the grid points. The interpolation becomes by the use of this method a mere evaluation of the spectrally given flow fields on these origins. This method is very accurate, but also quite slow and limited to stationary solutions. It is not further pursued in this work, because the stationary solutions are only used for testing and to validate some parameter settings, which will be used for potentially singular flows. The p-adaptive cascades, briefly discussed in Section 4.1, use interpolation orders for the velocity field from 3 to `interp_order` to minimize energy loss, and equally, cycle through those orders to minimize helicity loss when interpolating the vorticity utilizing the formerly found energy minimizing velocity field. While the non-extensive versions apply the same interpolation order in the vertical and radial direction, the extensive version tries combinations of orders from the aforesaid range in radial and vertical direction. The loops over the interpolation orders are aborted if energy or velocity loss is found to be exactly zero. The fixed cascade interpolation orders for `interp_method=4,5` are chosen by `interp_order` and range from 3 to 20 for the B-spline version and higher for the Lagrange version. The refinement methods discussed in Section 4.2 are selected here in setting rfp_r by `ref_fact_pow_r` and rfp_z by `ref_fact_pow_z`.

Time-step parameters. The user has the choice of setting a fixed number of time-steps with `nb_tmstp` or, by setting `nb_tmstp=0`, to let the program end itself, when any control parameter cannot be verified anymore, i.e. when the time-step drops below minimal `tmstp_lmt`. The initial fixed time-step (see Section 5.3 for details) is set with `tmstp_incr`. If `nb_tmadv` is non-zero, then the time is advanced by `tmadv_incr` via the analytic continuation method that is explained in Section 5.5. It is possible to combine those two time-stepping schemes, for instance, in every iteration, before an interpolation to Eulerian coordinates is executed, the flow can be advanced by a number of analytic continuations. Such a combination makes the algorithm very fast on the one hand, because it favors easy summations over costly interpolations. But on the other hand, it might reduce the precision of the scattered interpolations because the particles move further away from the grid. For the adaptive time-stepping method from Section 5.4, `tmstp_incr` has to be set to 0.0. The initial guess for the adaptive time-step is then given by `tmadv_incr`.

Dealiasing. The parameter `dealen` sets the dealiasing method. The user has the choice between no aliasing, the 2/3-rule, padding to double length, or implicit dealiasing via the public package fftwpp[1]. Note that only the Fourier modes are dealiased, the Chebyshev modes stay unchanged.

[1] see `http://fftwpp.sourceforge.net/` and [16]

6.2 How to Validate the Code

This section is intended to recall the de facto standard methods for validating numerical simulations of the (axisymmetric) incompressible Euler equations and adds original methods for the CLA. To date, there is no reliable method to verify the validity of a non-stationary incompressible Euler flow. There are, however, many propositions to approach this problem. A non-exhaustive list of numerical validation techniques can be found in [83], it may be extended by the conservation of Casimirs and general helicities discussed below. The most prominent quantity to watch in an incompressible Euler simulation is the total kinetic energy[1], given by

$$E = \frac{1}{2}\int_{D(1,L)} |v|^2 \,\mathrm{d}x = \frac{1}{2}\int_0^1 \int_0^L ((v^r)^2 + (v^\alpha)^2 + (v^z)^2) r \,\mathrm{d}r\mathrm{d}z. \tag{6.1}$$

If the velocity field v is a solution to the Euler equations (1.2), then the *energy is conserved* (see e.g. [50]). The converse is clearly not true. The check for this conservation can, therefore, serve as a control mechanism to verify that the grip on the actual solution has not been lost in a computation. Same goes for the *conservation of helicity*, it is given by

$$H = \int_{D(1,L)} v \cdot \omega \,\mathrm{d}x = \int_0^1 \int_0^L (v^r\omega^r + v^\alpha\omega^\alpha + v^z\omega^z) r \,\mathrm{d}r\mathrm{d}z\,. \tag{6.2}$$

However, in the author's opinion, the conservation of such quantities should not be trusted too much. Indeed, tests with stationary solutions show that the energy loss does not correlate well to the actual error in the flow fields. For this reason the p-adaptive interpolation methods described briefly in Section 6.1 were later discarded. As an extreme (but not excluded) example, if the velocity and vorticity have an initial helicity of zero, then the interpolating functions can be chosen such that the integral (6.2) is minimized with functions that actually vanish between the grid points, e.g. by choosing suitable Gaussians only at the grid points. The authors in [81] describe more general conserved quantities for axisymmetric flows. We shall recall their findings here briefly, also for later reference. The 3D axisymmetric Euler equations in cylindrical coordinates, eqs. (2.15) to (2.18), can be rewritten in terms of the Stokes stream function Ψ, the angular momentum $\sigma = rv^\alpha$ and the variable $\xi = \omega^\alpha/r$. This reformulation reads

$$\begin{aligned} \partial_t\sigma + \{\Psi,\sigma\} &= 0, \\ \partial_t\xi + \{\Psi,\xi\} &= \partial_z\left(\frac{\sigma^2}{4y^2}\right), \\ \frac{1}{2y}\partial_z^2\Psi + \partial_y^2\Psi &= -\xi, \end{aligned} \tag{6.3}$$

with $y = r^2/2$ and appropriate boundary conditions. Here, $\{\,,\}$ denotes the Poisson bracket, i.e. eq. (2.52) in the variables y and z. In [81], the conservation of the following integrals is proved for arbitrary functions f and g,

$$I_f = \int f(\sigma)\,\mathrm{d}y\,\mathrm{d}z, \tag{6.4}$$

$$H_g = \int \xi\, g(\sigma)\,\mathrm{d}y\,\mathrm{d}z\,. \tag{6.5}$$

[1] For brevity, we will often refer to this term simply as energy.

These integrals are commonly known as Casimirs and general helicities respectively, see also [74, 3] and references therein. Particular choices for f and g yield, for instance, the total angular momentum

$$I_1 = \int_{D(1,L)} \sigma \, \mathrm{d}y \, \mathrm{d}z = \int_0^1 \int_0^L v^\alpha r^2 \, \mathrm{d}r\mathrm{d}z, \tag{6.6}$$

or the total angular helicity

$$H_1 = \int_{D(1,L)} \xi\sigma \, \mathrm{d}y \, \mathrm{d}z = \int_0^1 \int_0^L v^\alpha \omega^\alpha r \, \mathrm{d}r\mathrm{d}z \, . \tag{6.7}$$

In the literature, the majority of Euler flow simulations is carried out on an Eulerian grid. The Lagrangian view point allows in return to carry out additional quality measurements regarding an implementation. One of those is the *movement of the boundary particles*. Theoretically, they stay on the boundary, but numerically, an error in the distance to the boundary after every iteration can be measured. In our cylindrical domain, this is easily achieved in computing

$$\max_{z\in[0,L]} |1 - \sqrt{R(1,z,t_1)^2 + A(1,z,t_1)^2}|, \tag{6.8}$$

where t_1 is a time-step. In the aftermath, this error is usually very close to the error in the energy conservation from further above, even though the current implementation of the CLA resets by default the boundary value to 1 after every iteration.

Unique to the implemented Cauchy-Lagrange formalism is also the possibility to *verify directly the Cauchy-invariants equation* (CIE), eq. (2.50), together with the *incompressibility condition* eq. (2.53) after every time-step insertion. Instead of checking only implications of a solution, this method verifies directly the CIE and thus, the Euler equations. Since, for accuracy reasons, all the needed terms in eqs. (2.50) and (2.53) are stored and summed in time anyway, the verification of the CIE can easily be achieved. On the contrary, in a purely Eulerian setting, the verification of the Euler equations appears to be tricky and impractical. Note, however, that the initial vorticity in eq. (2.50) is replaced with the current vorticity after every time-step, hence, the CIE is not verified with the initial vorticity from $t = 0$.

The authors of reference [83] verified for their flow that the energy and other quantities are conserved, and proposed additionally a *resolution study*. It aims to compare the values of a flow field component between computations with different resolutions at a fixed time. In other words, at a fixed time, it computes the sup-norm relative error between a computed solution on a finer grid with a numerical solution on a coarser grid. The solution on the higher resolution is for this purpose interpolated to the lower resolution grid before the maximal modulus of the difference is measured and consequently divided by the sup-norm of the higher resolution solution on its original finer grid. While dividing by the sup-norm measured on the higher resolution enables convergence studies, it is less suited for accuracy or precision validations, simply because this division represents a distorted relative error of maximal values on differently sized grids. Furthermore, the choices made in [83] originate from the adaptive moving mesh, which is designed to follow high gradients. Better results might be obtained with a division by the norm of the finer resolved solution evaluated on the lower resolution grid (by sub-grid extraction or interpolation). More precisely, let $v^\#$ be a numerical solution on a fine product grid $D^\#$ in the domain $D(1, L)$ after a fixed number

of time-steps. Suppose also a series of computed solutions $v^{\#_n}, n \in \mathbb{N}$, on coarser grained product grids $D^{\#_n}$ such that $D^{\#_n} \subseteq D^{\#_{n+1}} \subseteq \cdots \subseteq D^{\#}$. In this discrete setting, there holds eventually, $D^{\#_m} = D^{\#}$ for high enough m. Then, the quality of the convergence of

$$\tau_n \coloneqq \frac{\|v^{\#} - v^{\#_n}\|_{max,D^{\#_n}}}{\|v^{\#}\|_{max,D^{\#_n}}} \tag{6.9}$$

to zero provides insights on the convergence and precision of the flow on the grid $D^{\#_n}$. The norm $\|\,.\,\|_{max,D^{\#_n}}$ denotes the maximal value of the modulus on the grid $D^{\#_n}$. For comparison, in [83], $\|v^{\#} - v^{\#_n}\|_{max,D^{\#_n}}/\|v^{\#}\|_{max,D^{\#}}$ is taken instead of τ_n. The norm can be replaced by the l^2-, or any other l^p- norm. In the given scenario, it is assumed that higher resolutions yield better results, which is not always true. Spectral derivatives for instance degrade with increasing resolutions and, thus, the above strategy must be handled with care. This is also said in sight of the apparent discrepancy between the sup-norm relative error and the energy conservation in [83]. In the aforesaid reference, at a fixed time, a sup-norm relative error of a lesser resolved to a finer resolved angular velocity component was reported to be of magnitude 10^{-7}, while the total energy loss at this fixed time and both, lesser and finer, resolutions is reported to be $\sim 10^{-13}$.

Another conserved quantity is the *circulation* about a closed material curve C that moves with the fluid ([75, 10]). Kelvin's circulation theorem states that

$$\partial_t \Gamma(t) = 0, \tag{6.10}$$

with

$$\Gamma(t) = \int_{X(C,t)} v(x,t) \cdot \mathrm{d}\sigma(x) = \int_0^1 v(X(C(s),t),t) \cdot \partial_s X(C(s),t)\,\mathrm{d}s, \tag{6.11}$$

if C is parametrized by $s \in [0,1]$ such that $C(0) = C(1)$. Differently formulated, the quantity

$$\int_C v(a,t) \cdot \mathrm{d}\sigma(a) \tag{6.12}$$

is constant under the material derivative $\mathrm{D}_t = \partial_t + (v \cdot \nabla)$. Numerically, this conservation is difficult to verify for arbitrary closed curves, but is in principle possible with spectral representations and the ability to evaluate them anywhere in the covered domain. An Evaluation of a non-trivial closed curve in the cylindrical domain $D(1,L)$ would need to be done very thoroughly in order to let the integral converge accurately to its conserved value. A considerable numerical complexity and large errors seem to be implied, which practically rules out this circulation method for general curves in ambitious Euler flow simulations. In return, for simple circular curves around the axis, this task is readily feasible. Such a curve is of the form

$$\bar{r}e_r + \bar{z}e_z, \quad \text{for fixed } \bar{r} \in [0,1], \bar{z} \in [0,L], \tag{6.13}$$

and is parametrized by $\alpha \in [0, 2\pi]$. In the notation of Section 2.2, Γ in cylindrical coordinates takes the form

$$\begin{aligned}\int_0^{2\pi} v(\tilde{R},\tilde{\Theta},\tilde{Z},t) \cdot \partial_\alpha(\tilde{R}e_{\tilde{R}} + \tilde{Z}e_z)\,\mathrm{d}\alpha &= \int_0^{2\pi} v(\tilde{R},\tilde{\Theta},\tilde{Z},t) \cdot \tilde{R}e_{\tilde{\Theta}}\mathrm{d}\alpha \qquad (6.14)\\ &= \int_0^{2\pi} \left(v^r(\tilde{R},\tilde{Z},t)e_{\tilde{R}} + v^\alpha(\tilde{R},\tilde{Z},t)e_{\tilde{\Theta}} + v^z(\tilde{R},\tilde{Z},t)e_z\right) \cdot \tilde{R}e_{\tilde{\Theta}}\,\mathrm{d}\alpha\\ &= 2\pi\tilde{R}\,v^\alpha(\tilde{R},\tilde{Z},t)\,. \qquad (6.15)\end{aligned}$$

Kelvins circulation theorem tells us that eq. (6.15) stays constant, which can be verified easily in every time iteration via

$$\frac{\|rv^\alpha(r,z,t_i) - \tilde{R}(r,z,t_{i+1})v^\alpha(\tilde{R}(r,z,t_{i+1}),\tilde{Z}(r,z,t_{i+1}),t_{i+1})\|_{l^2(D^\#)}}{\|rv^\alpha(r,z,t_i)\|_{l^2(D^\#)}}, \tag{6.16}$$

where t_i is the time after i iterations and $D^\#$ the underlying grid. This verification method is quite simple, low in numerical complexity and implemented in the CLA code. Note that in eq. (6.16) one cannot use the initial velocity $v_0^\alpha = v_0^\alpha(r,z,0)$, because, in the CLA implementation, $\tilde{R}(r,z,t_{i+1})$ is the spatial displacement with respect to the underlying grid within one time-step and not within the whole integration period $[0,t_{i+1}]$. In order to compare to the initial velocity v_0^α, one would need to track the particle positions $(\tilde{R}(r,z,t),\tilde{Z}(r,z,t))$ from $t=0$ to the time where the conservation of eq. (6.16) is checked. The method of analytic continuation allows for such a tracking, because the flow fields are always known on the evolving trajectories without any interpolation involved.

6.3 Stationary Solutions

This section recalls some stationary solutions to the three dimensional incompressible axisymmetric Euler equations in a cylindrical domain for testing purposes. A well known example of such a steady state axisymmetric flow is a vortex, given by the simple form

$$v = re_\alpha. \tag{6.17}$$

It is a flow with swirl velocity $v^\alpha = r$, where a constant vertical component can be added to entail some more particle movement. This flow allows to state the analytical form of the time-dependent particle trajectories $X = X(r,\alpha,z,t)$ as

$$X(r,\alpha,z,t) = r\cos(t)e_r + r\sin(t)e_\alpha + ze_3, \tag{6.18}$$

see [84]. The angle α only appears in the basis and may be set to zero to simplify the representation. Subsequently, from eq. (6.18), we find,

$$X(r,0,z,t) = \begin{pmatrix} r \\ 0 \\ z \end{pmatrix} + \begin{pmatrix} 0 \\ r \\ 0 \end{pmatrix} t + \begin{pmatrix} -\frac{r}{2!} \\ 0 \\ 0 \end{pmatrix} t^2 + \begin{pmatrix} 0 \\ -\frac{r}{3!} \\ 0 \end{pmatrix} t^3 + \begin{pmatrix} \frac{r}{4!} \\ 0 \\ 0 \end{pmatrix} t^4 + \dots \,. \tag{6.19}$$

A set of more sophisticated stationary solutions is presented in reference [81] and is obtained from set of equations (6.3), which are a reformulation of the incompressible axisymmetric Euler equations. In following the given reference, stationary solutions are characterized, in the notation of eqs. (6.3), by

$$\{\Psi,\sigma\} = 0 \quad \text{and} \quad \{\Psi,\xi\} + \left\{\frac{\sigma^2}{2y},\sigma\right\} = 0\,. \tag{6.20}$$

The authors of [81] then show that steady state solutions take the form

$$\sigma = f(\Psi), \tag{6.21}$$

$$\frac{1}{2y}\partial_z^2\Psi + \partial_y^2\Psi = \frac{f(\Psi)}{2y}f'(\Psi) + g(\Psi), \tag{6.22}$$

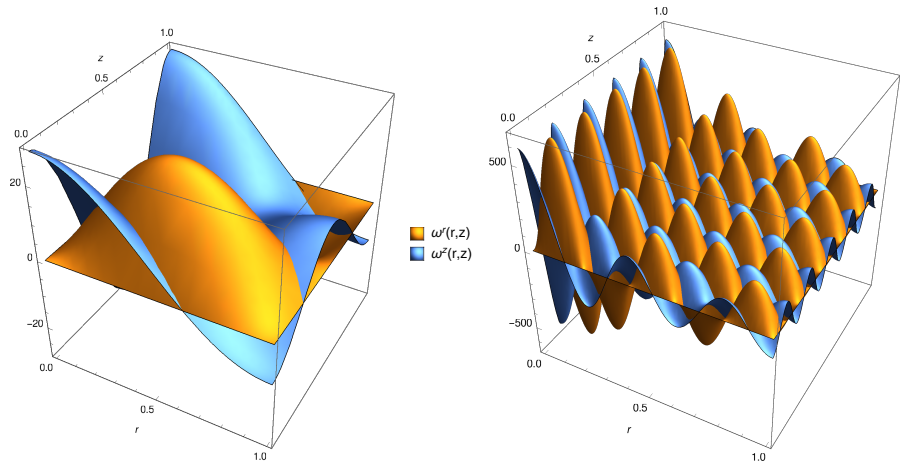


Figure 6.1: Stationary flow components. Left: radial and vertical vorticity for $\kappa = 1$ and with c_1. Right: same as left with $\kappa = 5$ and c_5.

for arbitrary functions f and g. Explicit results for simple shapes of those arbitrary functions are also given. If the function f is linear and g is constant, then

$$\sigma = A + B\Psi, \tag{6.23}$$

$$\Psi = \mu r J_1\left(\sqrt{(B^2 - \kappa^2)}r\right)\cos(\kappa z + \phi) - \frac{A}{B} - \frac{Cr^2}{B^2}, \tag{6.24}$$

with constants $\mu, \kappa, \phi, A, B, C$. Since Ψ is the stream function, the radial velocity component is given by

$$v^r(r,z) = -\frac{1}{r}\partial_z\Psi(r,z) = \mu J_1\left(\sqrt{(B^2 - \kappa^2)}r\right)\kappa\sin(\kappa z + \phi) - \frac{A}{rB} - \frac{Cr}{B^2}, \tag{6.25}$$

from which we infer that $A = 0$. Moreover, due to the no-flow boundary condition $v \cdot \nu = v \cdot e_r = v^r = 0$ on the boundary, the formula simplifies further to

$$v^r(r,z) = \mu\kappa J_1\left(\sqrt{(B^2 - \kappa^2)}r\right)\sin(\kappa z + \phi), \tag{6.26}$$

and B is chosen such that $\sqrt{(B^2 - \kappa^2)} =: c_j$ is the j-th positive zero of the Bessel function J_1. The constant κ must be of the form $\kappa = 2\pi k/L$, $k \in \mathbb{Z}$, to ensure periodicity, ϕ can be set to zero. Ultimately, the constant μ can be set to one. With these choices and insights, the vertical velocity component takes the form

$$v^z(r,z) = \frac{1}{r}\partial_r\Psi = \left(\tfrac{1}{r}J_1(c_j r) + c_j J_1'(c_j r)\right)\cos(\kappa z)\,. \tag{6.27}$$

The angular velocity can be determined via eq. (6.23). Since $\sigma = rv^\alpha$, we have

$$v^\alpha(r,z) = \sqrt{(c_j + \kappa^2)}J_1(c_j r)\cos(\kappa z). \tag{6.28}$$

In summary, the stationary velocity, which is used as the principal flow for testing the CLA in this work, takes the (simplified) form :

$$
\begin{aligned}
v^r(r,z) &= \kappa J_1(c_j r)\sin(\kappa z),\\
v^\alpha(r,z) &= \sqrt{(c_j+\kappa^2)}J_1(c_j r)\cos(\kappa z),\\
v^z(r,z) &= c_j J_0(c_j r)\cos(\kappa z)\,.
\end{aligned}
\tag{6.29}
$$

In taking the cylindrical curl of the velocity field, one obtains the corresponding vorticity,

$$
\begin{aligned}
\omega^r(r,z) &= \kappa\sqrt{c_j^2+\kappa^2}J_1\left(rc_j\right)\sin(\kappa z),\\
\omega^\alpha(r,z) &= \left(c_j^2+\kappa^2\right)J_1\left(rc_j\right)\cos(\kappa z),\\
\omega^z(r,z) &= c_j\sqrt{c_j^2+\kappa^2}J_0\left(rc_j\right)\cos(\kappa z)\,.
\end{aligned}
\tag{6.30}
$$

The J_k denote the k-th order Bessel functions of first kind. A visualization of the initial radial and vertical vorticity components on the (r,z)-plane for $L=1$ can be seen in Figure 6.1. In comparing the involved functions in eqs. (6.29) and (6.30), the velocity field components take a very similar shape. In the case of a vortex, X can be obtained by direct time integration of the characteristic equation $\dot{X}=v(X)$. This is much more involved for the present stationary solution. Therefore, we apply a different approach for this case in the next section and show that it is still possible to obtain analytic representations of the time-Taylor coefficients $X^{(s)}$ until a certain order for any stationary flow.

6.4 Explicit Computation of the Time-Taylor Coefficients for Stationary Solutions

We depart from the characteristic equation and leave the dependence on the arguments out for convenience (the following is true in any coordinate system),

$$
\dot{X}(t) = V(X(t)), \quad X(0)=\mathrm{Id},
\tag{6.31}
$$

for a classical stationary solution $V=V(a)$ to the three dimensional incompressible axisymmetric Euler equations. If $X(t)$ is analytic in time, then so is the velocity along the trajectories $V(X(t))$ due to eq. (6.31). Let $g(t) \coloneqq V(X(t))$, then

$$
g(t) = \sum_{s\geq 0} g^{(s)}t^s, \quad \text{with } g^{(s)} = \tfrac{1}{s!}\partial_t^s V(X(t))|_{t=0}\,.
\tag{6.32}
$$

One finds

$$
\sum_{s\geq 1} sX^{(s)}t^{s-1} = \sum_{s\geq 0} g^{(s)}t^s,
\tag{6.33}
$$

and after collecting powers of t,

$$
X^{(s)} = g^{(s-1)}, \quad s\geq 1\,.
\tag{6.34}
$$

The first coefficients are, thus, given by $X^{(0)}=\mathrm{Id}$, $X^{(1)}=V$, $(2!)X^{(2)}=DV\cdot X^{(1)}=DV\cdot V$, $(3!)X^{(3)}=X^{(1)\top}\cdot D^2V\cdot X^{(1)}+DV\cdot X^{(2)}=V^\top\cdot D^2V\cdot V+DV\cdot(DV\cdot V)$,...
Despite the fact that the general form may be given by a recursion formula and the formula of Faa di Bruno [41, 62], here, we are mostly interested in the explicit calculation of the time Taylor coefficients in cylindrical coordinates. This is possible by hand in the case of the

vortex, but rather tedious for higher orders in the case of more intricate solutions, such as those recalled in Section 6.3. We will therefore make use of computer algebra systems, but first state the recursion in cylindrical coordinates.

Remark. This method is only applicable to non-stationary solutions, if one knows all the time derivatives of the solution at $t = 0$, as $\partial_t V(X(t), t)|_{t=0} = D_x V_0(x) \cdot \dot{X} + \partial_t V(x, t)|_{t=0}$, where V_0 is the initial velocity.

To simplify notation, we write a generic stationary axisymmetric velocity field here as

$$V(r, \alpha, z) = u(r, z)e_r + v(r, z)e_\alpha + w(r, z)e_z. \tag{6.35}$$

With the definitions of Section 2.2 we may directly take the first derivative with respect to t of $V(X(t))$ and find

$$\begin{aligned}
\partial_t v(\tilde{R}, \tilde{\Theta}, \tilde{Z}) &= \partial_t \left[u(\tilde{R}, \tilde{Z}) e_{\tilde{R}} + v(\tilde{R}, \tilde{Z}) e_{\tilde{\Theta}} + w(\tilde{R}, \tilde{Z}) e_3 \right] \\
&= \left(u_r(\tilde{R}, \tilde{Z})\dot{\tilde{R}} + u_z(\tilde{R}, \tilde{Z})\dot{\tilde{Z}} - v(\tilde{R}, \tilde{Z})\dot{\tilde{\Theta}} \right) e_{\tilde{R}} \\
&+ \left(v_r(\tilde{R}, \tilde{Z})\dot{\tilde{R}} + v_z(\tilde{R}, \tilde{Z})\dot{\tilde{Z}} + u(\tilde{R}, \tilde{Z})\dot{\tilde{\Theta}} \right) e_{\tilde{\Theta}} \\
&+ \left(w_r(\tilde{R}, \tilde{Z})\dot{\tilde{R}} + w_z(\tilde{R}, \tilde{Z})\dot{\tilde{Z}} \right) e_3 \,.
\end{aligned} \tag{6.36}$$

Setting $t = 0$ and comparing the radial, angular and vertical components in eq. (6.34), the LHSs of the cylindrical Euler equations appear as expected for $s = 2$:

$$\begin{aligned}
\bar{R}^{(2)} &= u_r u + u_z w - \frac{v^2}{r}, \\
\bar{A}^{(2)} &= v_r u + v_z w + \frac{uv}{r}, \\
\bar{Z}^{(2)} &= w_r u + w_z w \,.
\end{aligned} \tag{6.37}$$

Higher derivatives however are more difficult to obtain by hand, but do not pose any problem to modern computer algebra systems. Here we give a Mathematica code, which computes the time derivatives of the components of the displacement map up to the order `smax`, where *wlog* $\alpha = 0$ is chosen:

```
ut[r_, z_] := u[r, z]
vt[r_, z_] := v[r, z]
wt[r_, z_] := w[r, z]

\[Theta][t_] := ArcTan[A[t]/R[t]]
v0[t_] :=
    ut[Sqrt[R[t]^2 + A[t]^2], Z[t]] {Cos[\[Theta][t]],Sin[\[Theta][t]], 0}+
    vt[Sqrt[R[t]^2 + A[t]^2], Z[t]] {-Sin[\[Theta][t]],Cos[\[Theta][t]], 0}+
    wt[Sqrt[R[t]^2 + A[t]^2], Z[t]] {0, 0, 1}

R[0] = r; A[0] = 0; Z[0] = z;

Do[{Derivative[k][R][0], Derivative[k][A][0], Derivative[k][Z][0]} =
    FullSimplify[Derivative[k - 1][v0][0], r > 0];

Print[{Derivative[k][R][0]/k!,
       Derivative[k][A][0]/k!,
       Derivative[k][Z][0]/k!}];
, {k, 1, smax}]
```

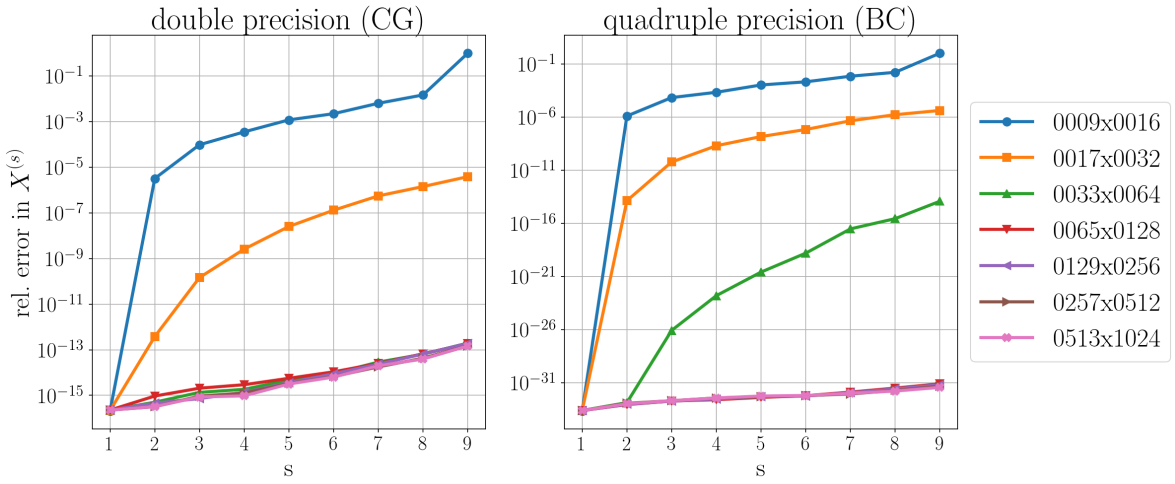


Figure 6.2: Relative errors in the computed coefficients $X^{(1)}, \ldots, X^{(9)}$, which belong to the flow in eq. (6.29) with $\kappa = 1$ and c_1. The Poisson problems were solved in double precision via the Chebyshev-Galerkin solver (left) and in quadruple precision via the Bessel-Chebyshev solver (right). The true coefficients were obtained via symbolic differentiation inside a computer algebra system by the method described in this section.

The `R,A,Z` in the code above represent $s!R^{(s)}, s!A^{(s)}, s!Z^{(s)}$ respectively. Here, $\arctan(A(t)/R(t))$ instead of $\mathcal{A}(R(t), A(t))$ may be taken for $\Theta(t)$, because of the time-derivative at $t = 0$. If we run the above code with $u(r, z) = 0, v(r, z) = r$ and $w(r, z) = 0$, as to say the case of a vortex, we readily obtain

$$\begin{aligned}
\{R^{(1)}, A^{(1)}, Z^{(1)}\}/1! &= \{0 \quad , r \quad , 0\} \\
\{R^{(2)}, A^{(2)}, Z^{(2)}\}/2! &= \{-\tfrac{r}{2}, 0 \quad , 0\} \\
\{R^{(3)}, A^{(3)}, Z^{(3)}\}/3! &= \{0 \quad , -\tfrac{r}{6}, 0\} \\
\{R^{(4)}, A^{(4)}, Z^{(4)}\}/4! &= \{\tfrac{r}{24} , 0 \quad , 0\} \\
\{R^{(5)}, A^{(5)}, Z^{(5)}\}/5! &= \{0 \quad , \tfrac{r}{120}, 0\} \\
&\ldots
\end{aligned}$$

In case of the stationary solution with Bessel functions, we will not give higher coefficients analytically, but we verify the appearance of the LHSs of the cylindrical Euler equations in $R^{(2)}/2\ e_r + A^{(2)}/2\ e_\alpha + Z^{(2)}/2\ e_z$ in taking the cylindrical curl of the last formula,

```
kk=2 ;
FullSimplify@Curl[
{Derivative[kk][R][0]/(kk!), Derivative[kk][A][0]/(kk!), Derivative[kk][Z][0]/(kk!)}
, {r, \[Alpha], z}, "Cylindrical"]
```

with the result `{0 , 0 , 0 }` as expected, because a curl of a gradient $(-\nabla p)$ has been taken, see eqs. (6.37). This method has been used to compute some coefficients of the stationary flow (6.29), with $\kappa = 1$ analytically within Mathematica to evaluate those on a 513×1024 grid. The result has been printed to file and can be read into the program. Figure 6.2 is showing the relative errors of the CLA-computed coefficients to the evaluated analytical coefficients up to $s = 9$. The computation of the analytical terms is quite expensive for higher values of s, because of the recursive character of the computation, and, unfortunately, the symbolic differentiation cannot be parallelized in Mathematica (at least to the knowledge of the author). We see in Figure 6.2 that the relative errors attain quickly their minimum

when the resolution is increased, which is mainly due to the choice of $\kappa = 1$ and c_1, where the latter is the first root of J_1. The left panel of the mentioned figure shows results that were obtained in using the Chebyshev-Galerkin (CG) solver and the right panel used the Bessel-Chebyshev (BC) solver. There are only very minor differences when the two solvers are compared directly. If the BC solver was plotted as well in the left panel for the same working precision, then the plots would overlap for all the tested grid sizes. However, the CG solver is not yet implemented in quadruple precision, due to the lack of a well suited quadruple precision solver for involved linear systems. Nevertheless, the BC solver can stem this weight and produces accurate results for already moderately large grid sizes in the case of this stationary flow. The fact that we can compute the coefficients analytically allows for an ideal testing environment for the Poisson solvers. For the plots in Figure 6.2, only the recursions from Section 2.4 and the Poisson solvers are used.

6.5 Test Results for Stationary Solutions

Stationary solutions are excellent testing candidates for the CLA code because, despite being in a steady-state, the flow can have non-trivial Lagrangian dynamics. In knowing the analytical flow fields, we can verify the computed velocity and vorticity on the trajectories directly even before the interpolation to the fixed original grid. Also, tracking the actual relative errors of those fields allows us to probe the other discussed quality criteria from Section 6.2, which may reveal the weaknesses and strengths of those criteria.

The implemented methods are tested here on the stationary flow from Section 6.3, namely eqs. (6.29) and (6.30), with c_1 being the first positive root of J_1 and $\kappa = 1$. The length of periodicity is hence fixed to $L = 2\pi$. In the course of the development of the current implementation of the CLA, a wide variety of parameter settings were tested to find the most suitable ones, see Section 6.1 for a list of run-time parameters. Here, we discuss the most important parameter values and their implications on the stability and convergence of the algorithm. If not stated otherwise, then the data sets were obtained by double precision computations, without de-aliasing, and no refinement was used for the interpolations. The simulations for the here presented test results had an implemented stopping criteria, which aborted the computation when the relative error in the velocity field became larger than 1.0×10^{-5}. The grids are always of the form $(2^n + 1) \times 2^{n+1}$, which was intended to suit the BC method, for which the needed Bessel-Chebyshev matrices (see eq. (3.140)) are saved to provide for a maximal resolution of 513×1024 grid points. Also, the cells at the boundary and pole of the cylinder become less elongated as in a square grid.

From Figure 6.2 we see that the coefficients of the stationary flow are already well resolved in low resolutions. Therefore, the resolution depended results displayed in Figure 6.3 for the velocity and vorticity relative errors can rather be explained by the convergence of the underlying B-spline cascade interpolation. Indeed, we can verify this in using the refinement techniques discussed in Section 4.2. The second row of Figure 6.3 shows the result of a simulation on 33×64 grid points that was stepwise increased within the interpolation procedure to account for 513×1024 interpolation points at the final step. We infer from Figure 6.3 also a quite large gap between the error in the velocity field and that in the vorticity field. This gap seems to widen the longer the simulation runs, such that at the end time in the highest resolution the difference has grown to over two orders of magnitude. The reason for this might lie in the way those fields are computed. The vorticity transport formula is used as in eqs. (2.38) to (2.40), several terms get multiplied with the initial vorticity at one time-step, which can build up errors faster. The velocity field is simply obtained by the characteristic equation as in eqs. (2.28) to (2.30), where an encounter with the initial value is damped

by the multiplication of the time-step. Remember, that the velocity field becomes the first coefficient in the new time-Taylor series.

Turning now to the interpolation methods, the cascade interpolation for the CLA via Lagrange polynomials or B-spline functions is far superior to all other tested scattered interpolation methods as discussed in the end of Section 4.1. Therefore, we may skip the comparison of different methods here and lay our eyes directly on different polynomial orders inside the cascade interpolation. Figure 6.4 shows the impact on the accuracy of the computed velocity field when only the interpolation orders vary. The used resolution of 513×1024 points ensures a high enough point density, such that at least one interpolation order gives accurate results with respect to machine precision, i.e. at least one interpolation order reached the minimal relative error of 2.2×10^{-16} for the velocity field on the fixed Chebyshev-Fourier grid. See the caption of Figure 6.4 for the other fixed parameters used in these runs, such as time-step and truncation order. One could argue, that higher interpolation orders produce a better outcome because the underlying functions are represented by high order polynomials and combinations of trigonometric functions. The contrary is true. From the aforesaid figure we infer, that lower orders are much more stable than higher orders in the B-spline cascade and moderate orders seem to prevail in the Lagrange cascade. Third-order B-spline functions

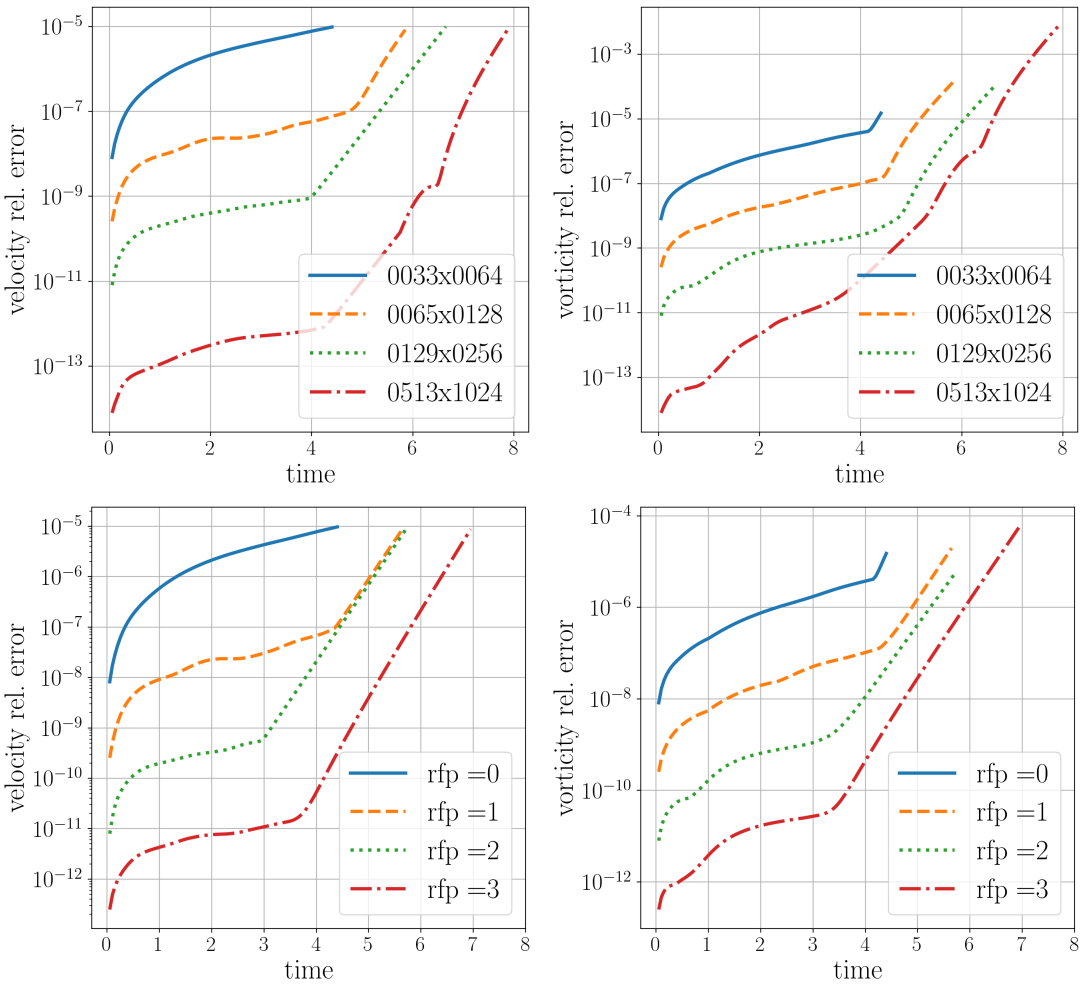


Figure 6.3: Top row: A visualization of relative errors in velocity (left) and vorticity field (right) due to different resolutions. The fixed run-time parameters are: $S = 25$, time-step 0.05, interpolation method C-B-5 (5-th order B-spline cascade), no refinement. Bottom row: The flow is computed on a 33×64 grid and the interpolations are refined by $\mathrm{rfp}_r = \mathrm{rfp}_z = \mathrm{rfp}$.

cannot compete with the fifth-order equivalents, indicating that there are optimal orders. However, this behavior is surely dependent on the underlying flow and the chosen resolution. The problem with interpolating functions of a higher degree is that they amplify errors too strongly and lead quickly to unstable behavior. A similar outcome can be observed in the first panel of Figure 6.7 further below, where the flow parameters have been changed, namely to $\kappa = 5$ and the fifth root c_5, to form more wavy flow fields see Figure 6.1 for a function plot in unified length. Also for this flow, lower and moderate orders give the best results.

Apart from resolution and interpolation methods, the time-step influences the outcome too, but to a lesser extent. If we take a look at Figure 6.5, then we see a convenient property of the CLA. Larger time-steps bring better results. This fact has been observed in all the computed flows when all other run-time parameters are fixed and can be measured by other quality methods from Section 6.2 if the flow is non-stationary. This behavior is somewhat counter-intuitive because one could expect that the smaller the time-steps the more precise the time-Taylor summations and the better the interpolations. This is true in one iteration, but more time-steps have to be executed to arrive to a certain time and more interpolations have to be carried out. Scattered interpolation methods are not known for their accuracy and indeed this accumulation of errors may be seen in the plot. Needless to say that larger time-steps also decrease the total run time of the simulation for a fixed integration period. The time-steps in Figure 6.5 were maintained over the whole execution time, thus for the run with the shortest time-step 720 steps had to be made, while the run with the largest time-step only needed 83 iterations to arrive to its final time. Other time-steps have been tested on a much larger grid of 1025×8192 points, the results can be found in Figure C.1 in Chapter C.

Next we discuss the quality criteria from Section 6.2. Figure 6.6 shows how well some discussed conserved quantities are preserved in the simulation. Energy conservation is quite stable in the beginning when compared to the actual velocity error, but between $t = 5$ and $t = 7$ the computational indication on the energy conservation is too good. The measured energy loss does not mirror the relative error in the velocity field. The difference between the final errors climbs to about 5 orders of magnitude: 10^{-5} for the velocity relative error, and

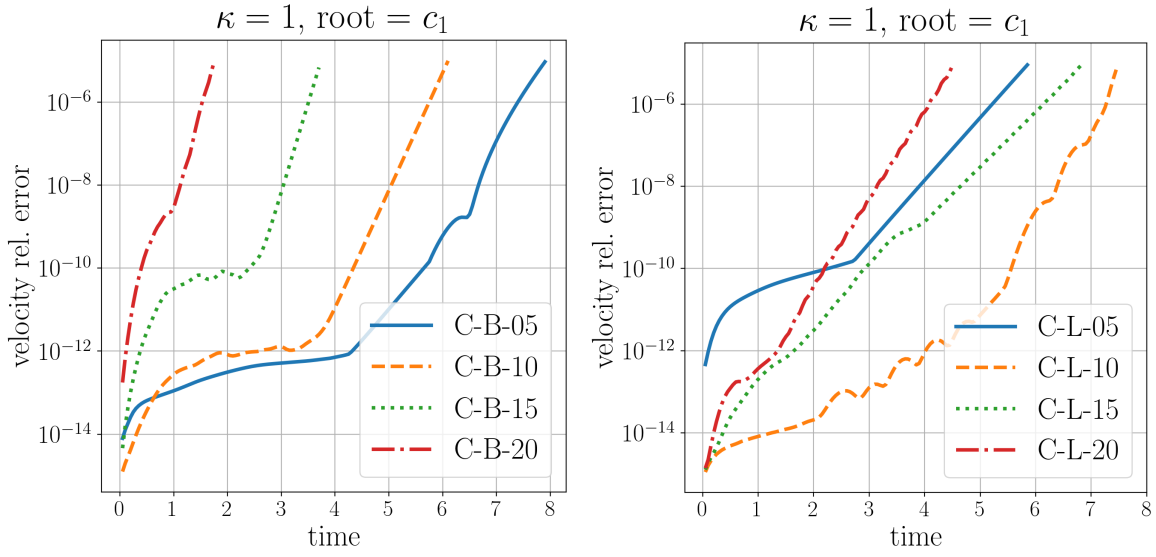


Figure 6.4: The effect of different orders inside the B-spline (left) and Lagrange cascade interpolation (right). Only the relative error in the velocity field is plotted. The run-time parameters are as in Figure 6.3.

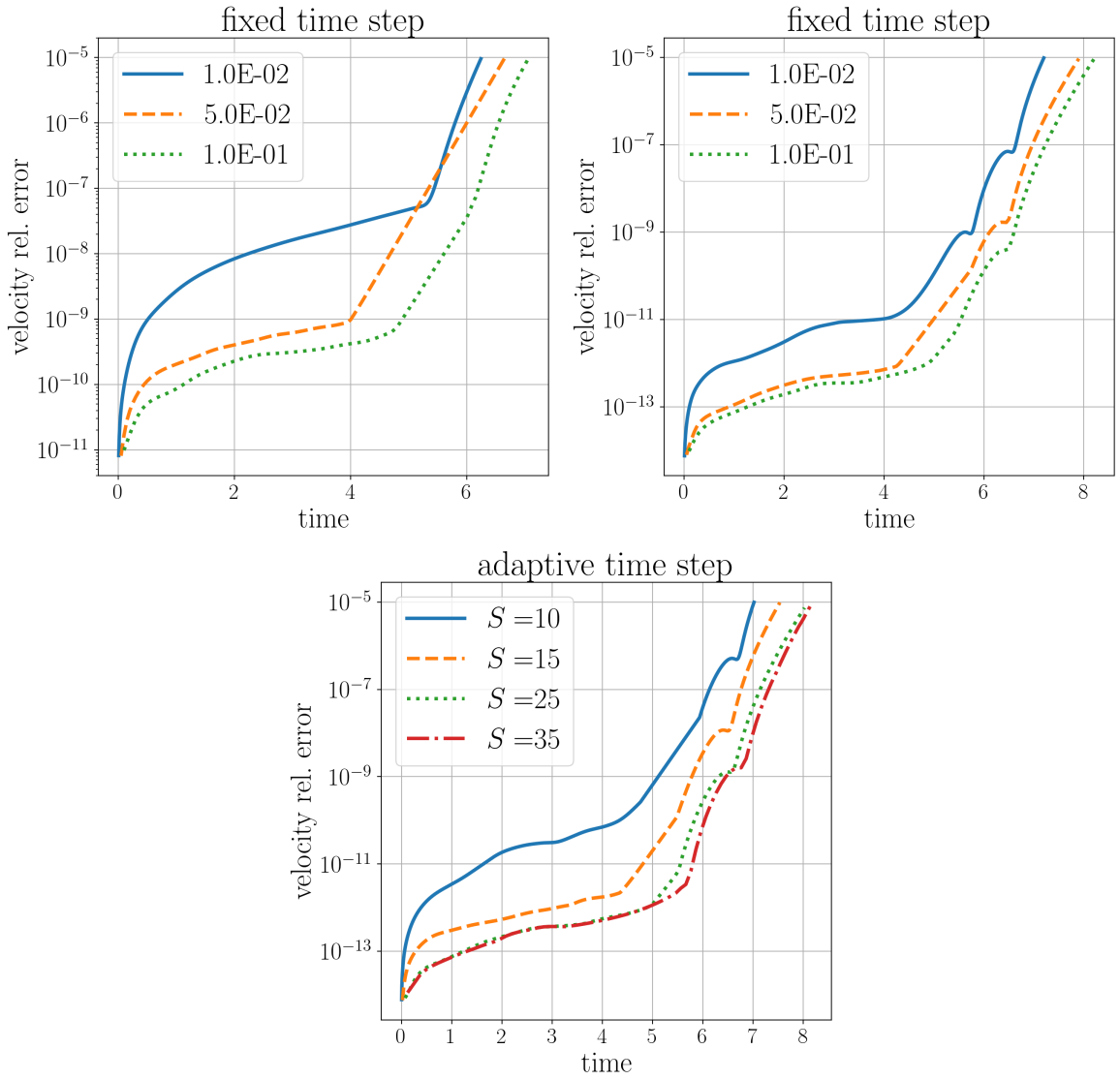


Figure 6.5: Top row: effect of fixed time-stepping on a 129×256 (left) and 513×1024 grid (right). Larger time-steps yield better results. Bottom: adaptive time-stepping with truncation orders given in the panel. All flows used the Chebyshev-Galerkin (CG) solver, B-C-5, and no refinement nor dealiasing was used.

10^{-10} in the relative change of total kinetic energy. In the case of the helicity conservation, an even higher difference is measurable. The total loss of helicity is close to the energy loss for this particular flow, but the vorticity error builds up more quickly than that of the velocity field. Concerning the conservation of Casimirs from eq. (6.4) and general helicities (6.5), the plot shows the angular momentum (6.6) and angular helicity (6.4). We can see that the preservation of the angular momentum aligns better with the error developments of the flow fields and the angular helicity behaves similar to energy loss. On the one hand, a possible downside of verifying Casimir and general helicity conservation is that they only take the angular components of the flow fields into account and are in principle unaffected by the radial or vertical components. On the other hand, one may argue that, in general flows with swirl, the errors from one component will soon appear in the others as well, such that it might be reasonable and effective to track the above conserved angular quantities.

An even better alignment with the relative error in the velocity field is found in the preservation of Kelvin circulation, namely the verification of Kelvin's theorem eqs. (6.10)

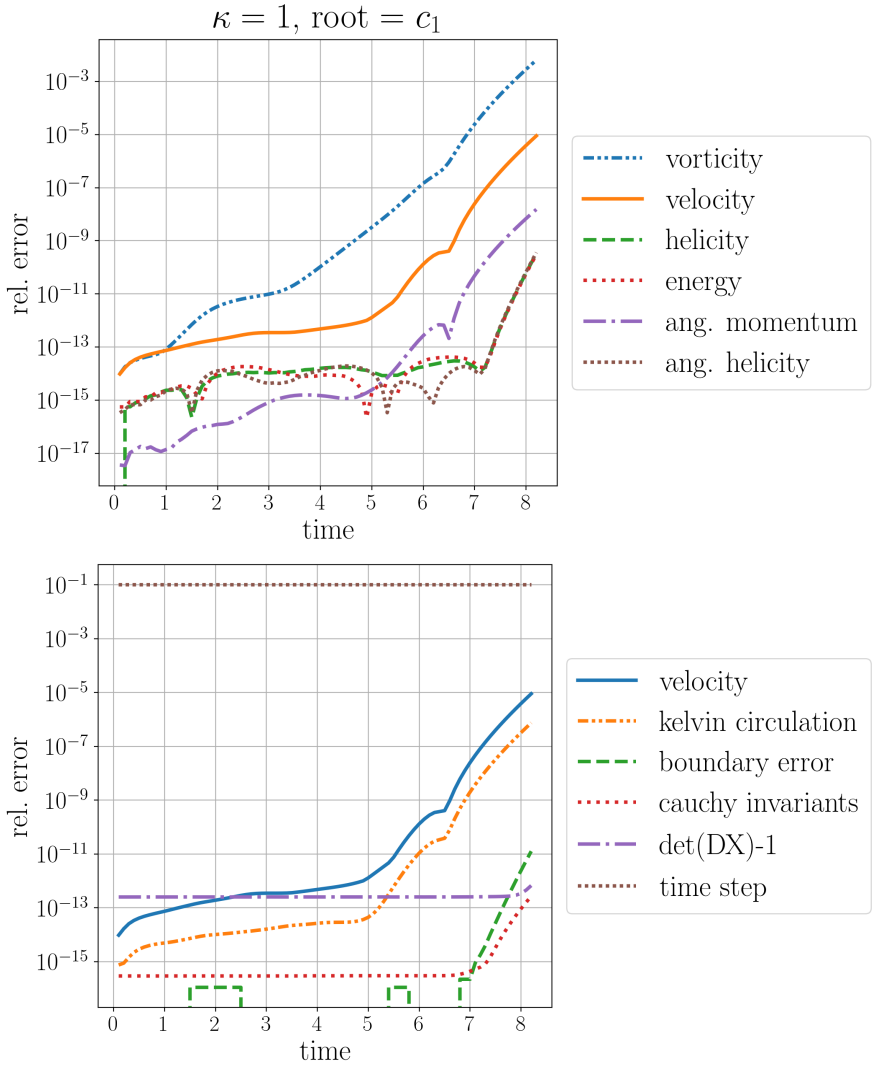


Figure 6.6: Test flow on a 513×1024 grid with $S = 40$, time-step 0.1, CG solver and B-C-5 interpolation, no refinement or dealiasing.

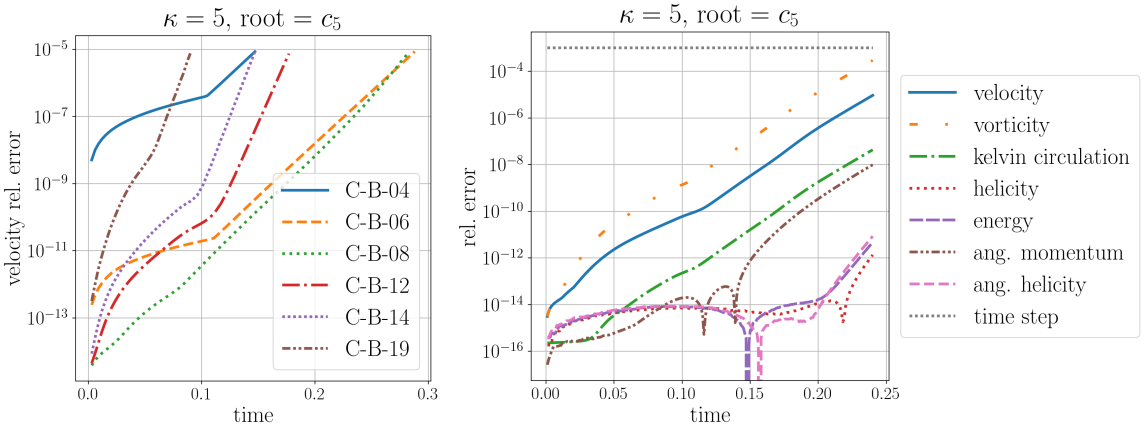


Figure 6.7: Test results for the stationary flow with $\kappa = 5$ and root c_5 as plotted in Figure 6.1. Left: effect of interpolation orders in the B-spline cascade. Right: overview of conserved quantities, where the time-step is 0.001 and $S = 25$, everything else is as in Figure 6.6.

and (6.11) via (6.16). Figure 6.6 visualizes the relative change of the latter quantity. This

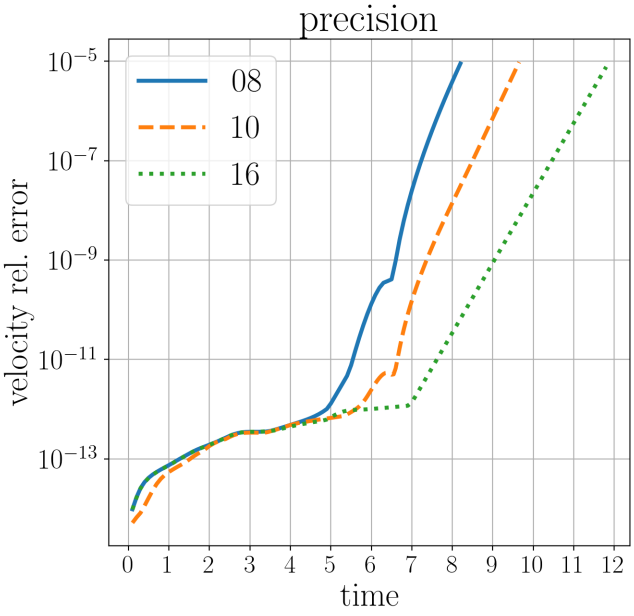


Figure 6.8: Comparison of the influence of double (08), long double (10), and quadruple precision (16) on the relative error in the velocity field. Higher precision computations improve the covered time span in which the tracked quantity stays below a certain threshold.

computational conservation of circulation behaves almost exactly like the slightly shifted relative error in the velocity and is computed in a simple manner in the CLA. Note that the expression (6.16) is well suited for Lagrangian computations, but difficult to obtain in Eulerian simulations. It is tempting to use the Kelvin conservation as principal quality criteria, but it is not ruled out that the good alignment here fails in other flows. Another important conserved quantity in the CLA is of course the Cauchy invariants formula (CIF) (2.42) itself. A relative error in this quantity is also visualized in Figure 6.6, together with the relative change in the incompressibility condition ($\det(\mathrm{D}X) = 1$). Both quantities are very well maintained and only show outbreaks towards the end of the simulation when the errors in the vorticity field are already quite large. Therefore, it appears that those quantities are less well suited to track the accuracy of the computation. Besides, the calculation of the expression in the left-hand side of the CIF is very costly, because the temporal derivatives of ∇X have to be computed additionally, which is not done in the CLA otherwise. The evaluation of $\det(\mathrm{D}X)$ is also quite expensive and slows down the simulation simply due to the number of terms that are involved in its computation, see eq. (2.53) for this determinant in cylindrical coordinates. The amount of terms can also be responsible for the relatively high relative error that is seen for $\det(\mathrm{D}X) = 1$ in Figure 6.6. Because of the given arguments, and the experience that the described behavior is observable in all the tested flows, the verification of those criteria was later neglected in most of the CLA runs to speed up the computation.
Another criterion that is easily verified is the movement of the boundary particles. The relative change of $\tilde{R}(1, z) - 1$ is plotted in Figure 6.6 as well and shows to be zero for most of the time, which is surely connected to the fact that the value $\tilde{R}(1, z)$ is explicitly set to 1 before the start of a new iteration. The radial movement of the pole particles is intrinsically non-existent. The movement of the second column of particles, close to the pole, is tracked too, and a run is aborted when those particles arrive at, or even cross the pole, but this actually never happens. The same is monitored for the second row of particles at the boundary.

The above criteria can additionally be inspected for the more wavy stationary flow where $\kappa = 5$ and c_5 is used instead of c_1, see the second panel of Figure 6.1. The concerning relative

errors are plotted in Figure 6.7 and similar conclusions can be drawn for this flow.

6.6 Test Results for Solutions without Swirl

Another possibility to further test the CLA is given by *swirl-free* flows, whose angular component stays exactly zero at all times. As the here discussed swirl-free flows are non-stationary, one cannot directly measure a relative error in the flow fields, but fortunately they are better understood compared to general three-dimensional flows. For instance, it is shown in [110] that the axisymmetric Euler equations in a (wall-bounded) cylindrical domain Ω are well-posed globally in time for swirl-free initial data in $H^s(\Omega)$ for $s \geq 3$. It is shown furthermore that $s > 5/2$ is sufficient in case $\Omega = \mathbb{R}^3$. See also [116, 102, 110, 84] for connected results in $\mathbb{R}^3$ with $s > 7/2$. Moreover, we can verify numerically, next to the fact that the angular component stays zero at all times, that the pseudo-vorticity ω^α/r stays constant along the trajectories and verifies a maximum principle (see e.g. [84]). More precisely, in the notation of eq. (4.2), we have

$$\frac{\omega^\alpha(\tilde{R}(r,z,t),\tilde{Z}(r,z,t),t)}{\tilde{R}(r,z,t)} = \frac{\omega^\alpha(r,z,0)}{r}\,. \tag{6.38}$$

In order to verify these facts, we simulate a swirl- and divergence-free flow with initial components

$$\begin{aligned}
v_0^r(r,z) &= \frac{2\pi}{L}(1-r)^n r^{a+2}\cos(2\pi z/L)\,,\\
v_0^\alpha(r,z) &= 0,\\
v_0^z(r,z) &= -(1-r)^{n-1} r^{a+1}\sin(2\pi z/L)\,(a - r(a+n+3)+3),
\end{aligned}$$

where $a \geq 1$ and $n \in \mathbb{N}$. As $\omega = \nabla_c \times v$, the radial and vertical component of the vorticity field vanish identically and the initial angular vorticity component reads

$$\begin{aligned}
\omega_0^\alpha(r,z) = \frac{1}{L^2}(1-r)^{n-2} r^a \sin(2\pi z/L)\,\big(&L^2((a+1)(a+3)+r^2(a+n+1)(a+n+3)\\
&- r(2a^2+2a(n+4)+5n+6)) - 4\pi^2(r-1)^2 r^2\big)\,.
\end{aligned}$$

As a matter of fact, there holds

$$v^\alpha(r,z,t) = \omega^r(r,z,t) = \omega^z(r,z,t) = 0 \tag{6.39}$$

for all times t. These identities are exactly preserved by our implementation of the CLA, which is visualized exemplary in Figure 6.9, right panel. Therefore, all the conserved quantities which only depend on those components, such as Kelvin circulation etc., stay exactly zero in the computation. Also, the helicity evaluates exactly to zero, since for its computation the term $v^r\omega^r + v^\alpha\omega^\alpha + v^z\omega^z$ $(= 0)$ is integrated over the (r,z)-plane. As for the stationary solution in Section 6.5, the results for the swirl-free flow dependent on resolution, the interpolation order, and the chosen time-step. Higher resolutions, lower to moderate interpolation orders, and larger time-steps increase the integration period until a critical energy loss occurs. The time-step $\triangle t = 0.1$ was maintained over the whole simulation and the run was stopped, when energy conservation exceeded 10^{-6}. The total kinetic energy, in Figure 6.9 left panel, seems to be well preserved until $t = 1.5$. However, we know from the tests with the stationary solution (see the upper panel of Figure 6.6 in particular), that the actual error in the flow fields can behave differently to the error in the energy conservation. The min/max-principle mentioned above is well demonstrated in Figure 6.9 right panel, even though a final shift in

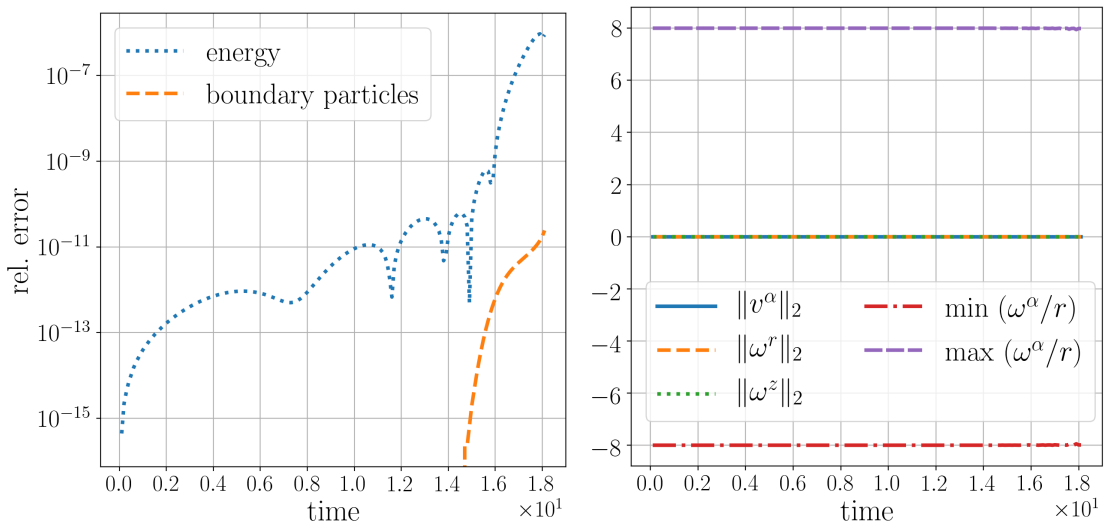


Figure 6.9: Test results for the given swirl-free flow with $a = 1$ and $n = 2$, and run-time parameters: $S = 20$, $\triangle t = 0.1$, Chebyshev-Galerkin solver, 5th order B-spline cascade interpolation, no dealiasing, on a 513×1024 Chebyshev-Fourier grid. Left: kinetic energy conservation and movement of the boundary particles in radial direction, all the conserved quantities, which are not plotted here, stayed identically zero in the simulation; Right: verification of the min/max principle for $\omega^\alpha/r \mid_{r\neq 0}$, and verification of eq. (6.39).

the min/max values on the fixed grid of around 10^{-5} is observed. This shift, however, might simply stem from the fact that the real extrema do not lie on the grid and move towards, or away from the closest grid points. Higher resolutions yield better results also with respect to this point, although very large resolutions imply other inaccuracies, such as accumulating rounding errors.

Chapter 7

Application to a Potentially Singular Flow

The presented implementation of the Cauchy-Lagrange algorithm in a domain with boundaries was developed in sight of the computational blow-up result that was reported in [83]. As briefly discussed in the introductory chapter, it is crucial to verify the singular vorticity behavior from the given reference by entirely different numerical methods. This first implementation aims to provide such a method and some computational results are presented in this chapter.

7.1 Algorithm and Initial Data of G. Luo and T. Hou in [83]

The authors of reference [83] compute an axisymmetric and incompressible Euler Flow in the lower quarter of the cylinder $D(1, 1/6)$, and claim a finite time singularity of the maximal vorticity modulus for the initial swirl-only velocity field with $v_0^r = v_0^z = 0$ and

$$v_0^\alpha(r, z) = 100\, r\, \mathrm{e}^{-30(1-r^2)^4} \sin(2\pi z/L), \quad L = 1/6\,. \tag{7.1}$$

In [83], a 3×10^8-fold increase of the quantity $\|\omega\|_{L^\infty}$ is reported, where the maximum of the vorticity modulus is attained on the boundary $\partial D(1, 1/6)$ (at $r = 1$) and, more precisely, on the symmetry axis at $z = 0$. Their investigations depart from a totally different formulation of the incompressible Euler equations than that in the CLA, using a transform of the stream-vorticity formulation of the incompressible Euler equations. This transform reads in our notation

$$\begin{aligned} v_{1,t} + v^r v_{1,r} + v^z v_{1,z} &= 2 v_1 \psi_{1,z}, \\ \omega_{1,t} + v^r \omega_{1,r} + v^z \omega_{1,z} &= (v_1^2)_z, \\ -\left[\partial_r^2 + (3/r)\partial_r + \partial_z\right] \psi_1 &= \omega_1, \end{aligned} \tag{7.2}$$

with

$$v_1 = v^\alpha / r, \quad \omega_1 = \omega^\alpha / r, \quad \psi_1 = \psi^\alpha / r, \tag{7.3}$$

where ψ^α is the angular component of the vector stream function, satisfying $-\Delta\, \psi = \omega = \nabla \times v$. The sub-scripts, different from 1, indicate partial derivatives as usual. See the referenced article and [84] for further details on the construction of the above formulation. The radial and vertical components of the velocity field are then given by

$$v^r = r \psi_{1,z}, \quad v^z = 2\psi_1 + r \psi_{1,r}. \tag{7.4}$$

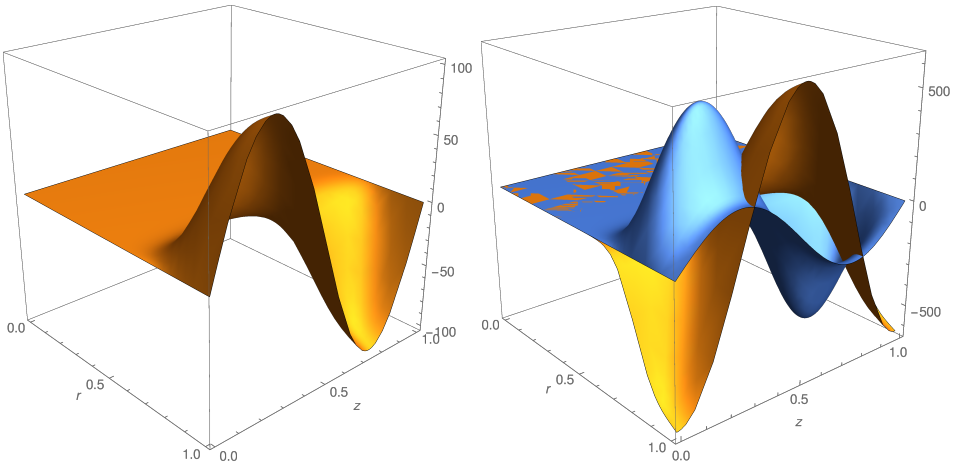


Figure 7.1: Left, initial swirl velocity (7.1). Right, initial radial (orange) and vertical (blue) vorticity components.

The Euler equations in this form are numerically convenient, because they avoid divisions by the radial argument r. Note that the CLA also avoids divisions in using the terms R_1 and A_1 from eq. (2.94).

In [83], the system (7.2) the third equation is solved via a 6th order B-Spline based Galerkin solver and a 4th order Runge-Kutta time-stepping scheme is applied to the first two equations, the appearing derivatives are obtained by a 6th order centered difference formula. However, the most important part of their method is the adaptive moving mesh, which keeps shifting grid points into the region with peak vorticity values. This adaptive mesh allows them to reach an effective maximal resolution of around $(10^{12})^2$ per unit square near the location of the potential singularity. Another important facet of their implementation is that they use the preserved symmetry of the initial velocity field, which allows to only model the quarter cylinder and start off with a much higher point density. The CLA has not yet been adapted to exploit those flow symmetries, it always computes the flow in the full cylindrical domain $D(1, L)$, here with $L = 1/6$. This can be seen as a draw-back of the current implementation, because a much higher grid size has to be employed in order to obtain similar initial local resolutions. However, even with this draw-back, the computational results are convincing, which highlights the potential of the CLA as a whole.

7.2 Computational Insights

As in Sections 6.5 and 6.6, we will investigate different parameter settings to show their influence on the outcome of a simulation. In all the presented runs, the computations were aborted if the time-step dropped below 5×10^{-8}. The time-step is halved whenever the truncation order S is reached and the precision criterion in eq. (5.17) is not fulfilled, i.e. if $F(S, \triangle t) > \varepsilon_m$, for a time-step $\triangle t$ and machine epsilon ε_m. This halving is done until the criterion is matched or the code aborted. The time-step is also halved, if the helicity or energy[1] loss reaches a certain limit (mostly 10^{-5}, but also 10^{-3} was used). The latter is usually responsible for a stop of the execution and not the precision criterion, such that at the stopping time a time-step above the given limit could not assure a helicity or energy loss below the given bound anymore. No refinement was used in the interpolations, except if it is stated otherwise, and the coefficients were computed until a precision criterion was reached in

[1] We simply write energy, but always mean total kinetic energy.

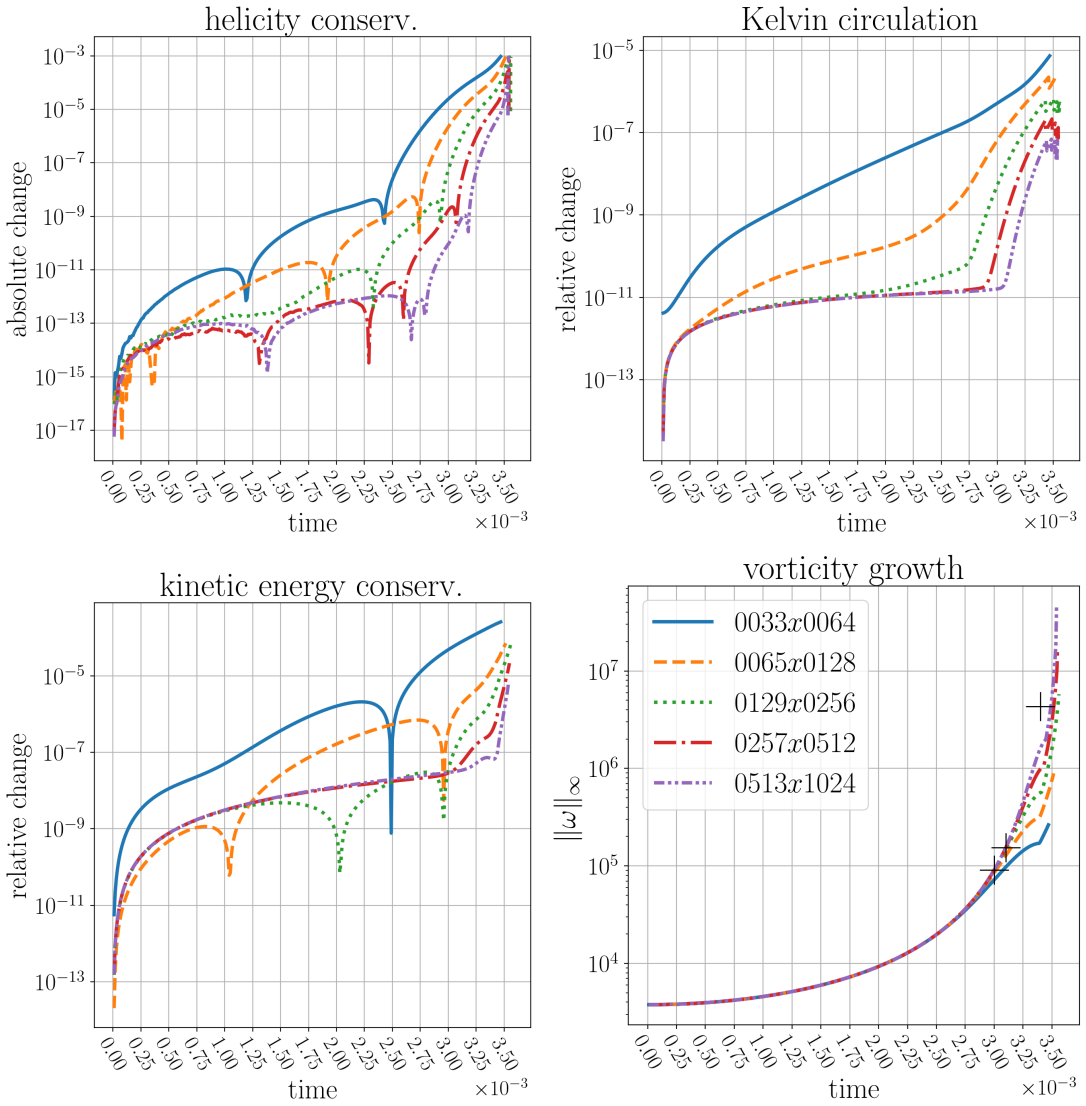


Figure 7.2: Resolution study with fixed parameters: $S = 15$, $\triangle t = 10^{-5}$, Bessel-Chebyshev solver, interpolation method C-B-5. no refinement, no dealiasing. The legend in the vorticity growth plots is valid for all other panels. Here and in the following plots, the black crosses in the vorticity plot are the exact coordinates from Table 7.1 at times $t = 0.003$, $t = 0.0031$ and $t = 0.0034$. Also the helicity panel displays absolute errors, because the initial helicity is zero for this flow.

the time-Taylor sums as explained in Section 5.3, see eq. (5.17) in particular. The results will be mainly measured by the conservation of helicity, energy and Kelvin circulation. Energy and helicity have not proven to be particularly good indicators for correctness, at least for the stationary solutions examined in Section 6.5, but they are very common in computational fluid dynamics and, in using them, the results become more comparable to other numerical methods. However, if the limit of 10^{-5} in relative energy or absolute helicity change is reached,

Table 7.1: The approximate maximal vorticity values in this table are taken from [83] and [6]. They will serve as a benchmark for our simulations. At least two algorithms (those from the given references) show good agreement on the value at $t = 3.1 \cdot 10^{-3}$.

time	0.0	$3.0 \cdot 10^{-3}$	$3.1 \cdot 10^{-3}$	$3.4 \cdot 10^{-3}$	$3.5 \cdot 10^{-3}$	$3.505 \cdot 10^{-3}$
$\|\omega\|_\infty$	$3.76999 \cdot 10^3$	$9.0847 \cdot 10^4$	$1.54276901 \cdot 10^5$	$4.3127 \cdot 10^6$	$5.8413 \cdot 10^9$	$1.2401 \cdot 10^{12}$

then the solution is likely to be inaccurate. The absolute change ($|H(t) - H(0)|$ instead of $|H(0) - H(t)|/|H(0)|$) in helicity is used, because the initial helicity is zero. The maximal vorticity values ($\|\omega\|_\infty$) at specific time instances, however, seem to be a better indicator for correctness, as some of those values are stated in [83], hence, they are directly comparable. The list of those maximal values is given in Table 7.1, it is appended by an additional value taken from [6]. The author of the latter reference has repeated the simulation of the same equations as in [83], i.e. system (7.2), with a spectral Chebyshev-Fourier solver on a fixed grid with 769 radial and 2048 vertical points. It is important to note that he also solved the equations on the quarter cylinder. It is further stated in [6] that their maximal vorticity value was communicated by one of the authors of reference [83]. This value at $t = 0.0031$, given in the third column of Table 7.1, is verified in [6] with a relative difference of 1.9×10^{-8}. Thus, there are at least two independent simulations which agree on the leading digits of a maximal vorticity value at a time instance not too far away from the reported temporal singularity. In the following plots of the maximal vorticity, the second, third and fourth values of Table 7.1 are indicated by black crosses to show qualitatively how far the CLA can approach them in those test runs. The insights from the following pages are then used to execute the CLA code on higher resolutions with results also reported further below.

We will go roughly as in Section 6.5 and start with a resolution study. Figure 7.2 shows the behavior of the error in total kinetic energy, helicity, and Kelvin circulation as described in Section 6.2, as well as the vorticity growth. Later figures are organized in the same manner, but sometimes the vorticity growth is narrowed to a smaller time span before the stopping time of the runs. As seen in the aforesaid figure, the result strongly depends on the chosen resolution, when all other run-time parameters from Section 6.1 stay fixed. The curves of the Kelvin circulation error, which will henceforth only be referred to as the Kelvin conservation, allow to easily distinguish between the low and moderate resolutions. The helicity conservation also shows noticeable differences, while the energy conservation curves are less clearly attributable to the different resolutions, which might be connected to the different error computation (absolute/relative). Also, all curves within one plot show very similar behavior with respect to their measured quantity. The appearing sudden drops in energy and helicity conservation are usually present, but their cause is not yet known to the author of this thesis. The maximal vorticity values that were computed on the largest grid agree qualitatively with the first values in Table 7.1, but their exact fulfillment will be studied later together with runs in higher resolutions. Note that even for the largest grid in Figure 7.2, only 256 vertical grid points lie in the modeled region of [83] and [6], i.e. in the quarter of the cylinder. For all the curves in Figure 7.2, the Bessel-Chebyshev solver (BC) was used in their creation process, but the Chebyshev-Galerkin (CG) method produces very similar results, the curves would mostly overlap if plotted. Another observation can be made regarding the vorticity growth when comparing the second and fourth panel of Figure 7.2. The maximal vorticity seems to fall off from its strong growth after a sudden change in the behavior of the corresponding Kelvin conservation curve. This could indicate that the breaks in the Kelvin conservation, together with the fall-off in vorticity growth, reveal inaccurate computations.

Figure 7.3 visualizes the effect of different interpolation orders. Here, we only discuss the changes induced by different orders in the B-spline cascade, as they appear to be more prominent than in the Lagrange cascade. Different orders in the Lagrange cascade show much less influence on the outcome of a simulation than in the B-spline case. However, the Lagrange cascade seems to be slightly less accurate in all the tested flows, but it is much faster than the B-spline variant, since no solving of linear systems is involved. As we aim for accuracy, we stick with the B-spline cascade in most of the here presented results. Generally,

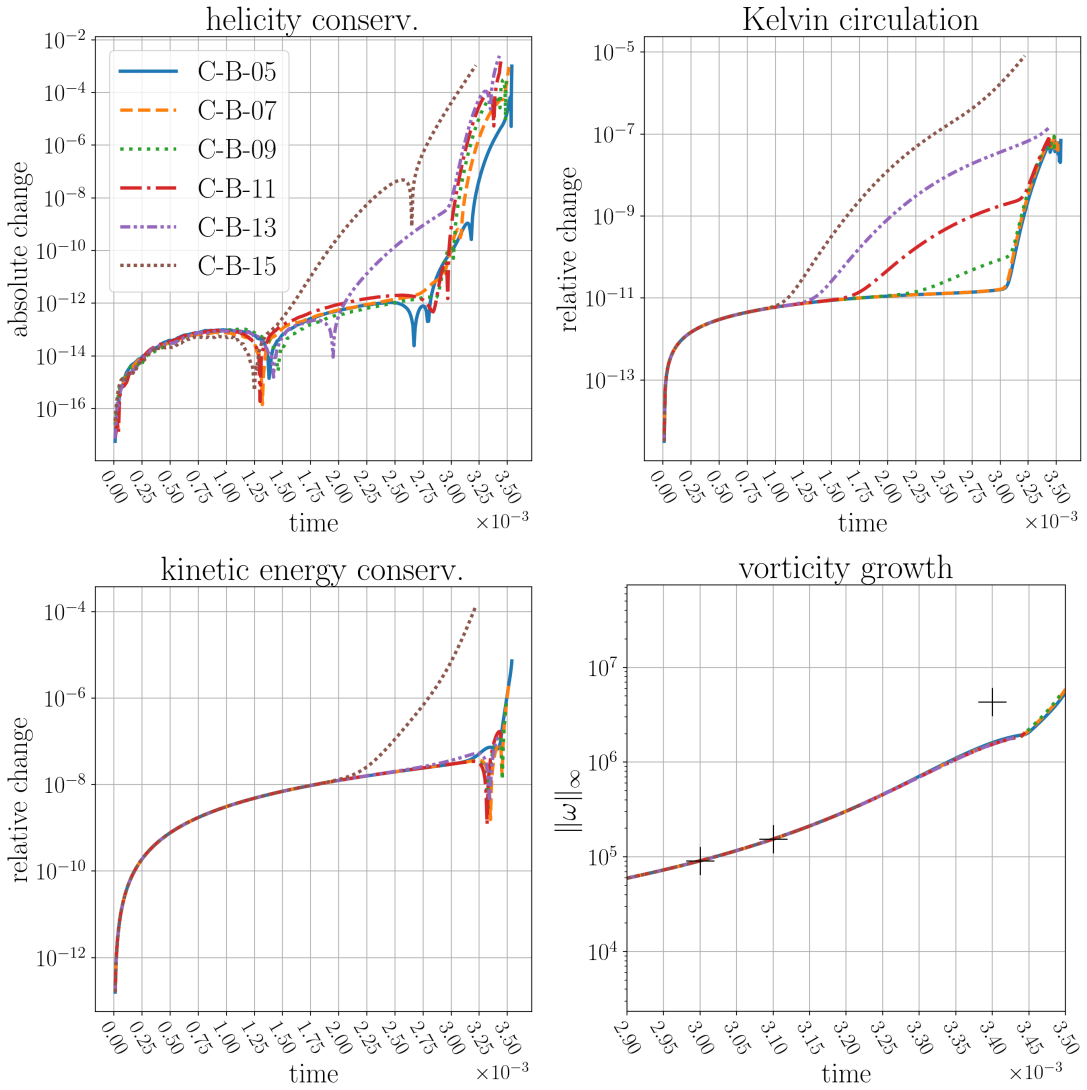


Figure 7.3: Interpolation orders are inspected with the same parameters as in Figure 7.2 on 513×1024 grid points. Lower orders seem to give better results with respect to helicity and Kelvin circulation. The vorticity growth plot is on a narrowed time span, but differences are hard to make out on a logarithmic scale.

as for the stationary flows, smaller interpolation orders seem to produce better results. This can be seen most noticeably in the Kelvin conservation, but also in the helicity conservation. The location of the relatively sharp break in the Kelvin conservation is connected to the low resolution of 1024 vertical points over the full length of periodicity L. In the flow, small Fourier modes fill up continuously and the resolution is eventually not capable of an accurate representation of the flow field components due to under resolution. The panel, which shows the vorticity growth, zooms into the region of most interest, all orders seem to agree well with the first values in Table 7.1, but as the vertical axis in this vorticity plot has a logarithmic scale, those agreements can only be understood qualitatively.

Concerning the time-steps, the initial coefficients have been used in Section 5.2 to obtain a guess on the possible convergence radius in Figure 5.4. Unfortunately, a sudden drop of values in the approximation of the convergence radius (Figure 5.4 (d)) hinders a good vision on a possible limit in the quotient rule for the radius (eq. (5.5)). The early development of the quotient $X^{(s-1)}/X^{(s)}$ seems to point to 10^{-4}, but this is momentarily hard to verify. However, time-steps between 5×10^{-6} and 5×10^{-5} have proven to provide good results. The time-step

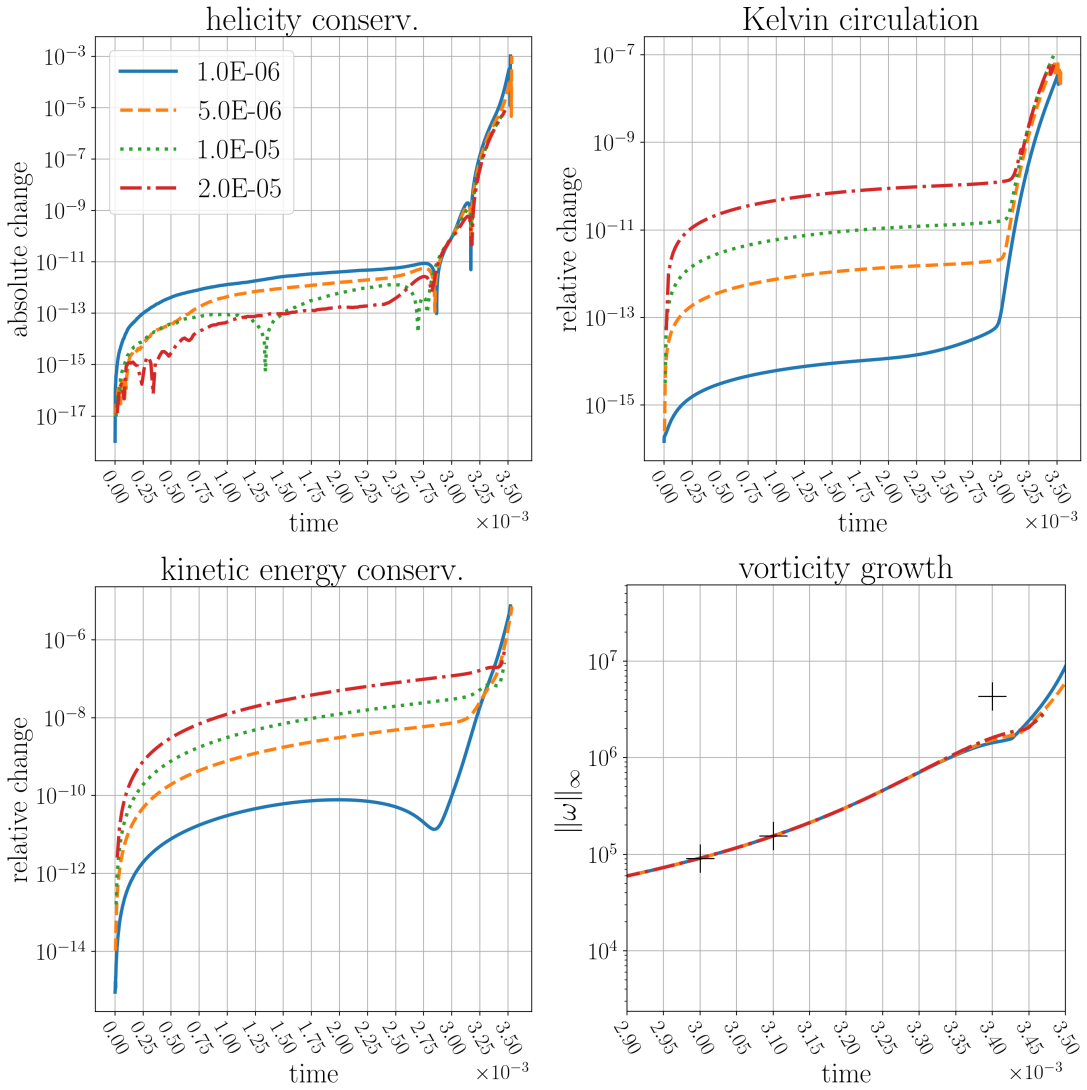


Figure 7.4: Several fixed time-steps are compared with the same run-time parameters as in Figure 7.3. Larger time-steps deliver good and fast results. The legend in the first panel is valid for the others too.

dependence of the outcome is plotted in Figure 7.4. The conservation of energy, helicity, and Kelvin circulation show qualitatively the same behavior, even though smaller time-steps maintain lower errors in the conservation of most of the monitored quantities. Remarkably, the absolute helicity change is lowest for the largest time-step. Also, if we consider the vorticity growth, then large time-steps and their corresponding vorticity growth falls off later than the growth with smaller time-steps. When searching for sudden breaks in the Kelvin conservation, then it appears that those happen later for larger time-steps, and the same is true for the energy conservation curves. Moreover, the runs with larger time-steps take a fraction of the total execution time compared to smaller time-steps, while still providing adequate, if not better, results. The latter property is quite common among Semi-Lagrangian methods and helps us here to achieve very large resolutions (see further below). The initial time-steps are not maintained throughout the computation, they are adapted to allow for precision and conservation criteria as mentioned before. However, these adaptations do not happen very often and the initial time-step can usually be maintained over many iterations.

The adaptive time-stepping procedure from Section 5.4 is examined in Figure 7.5 for

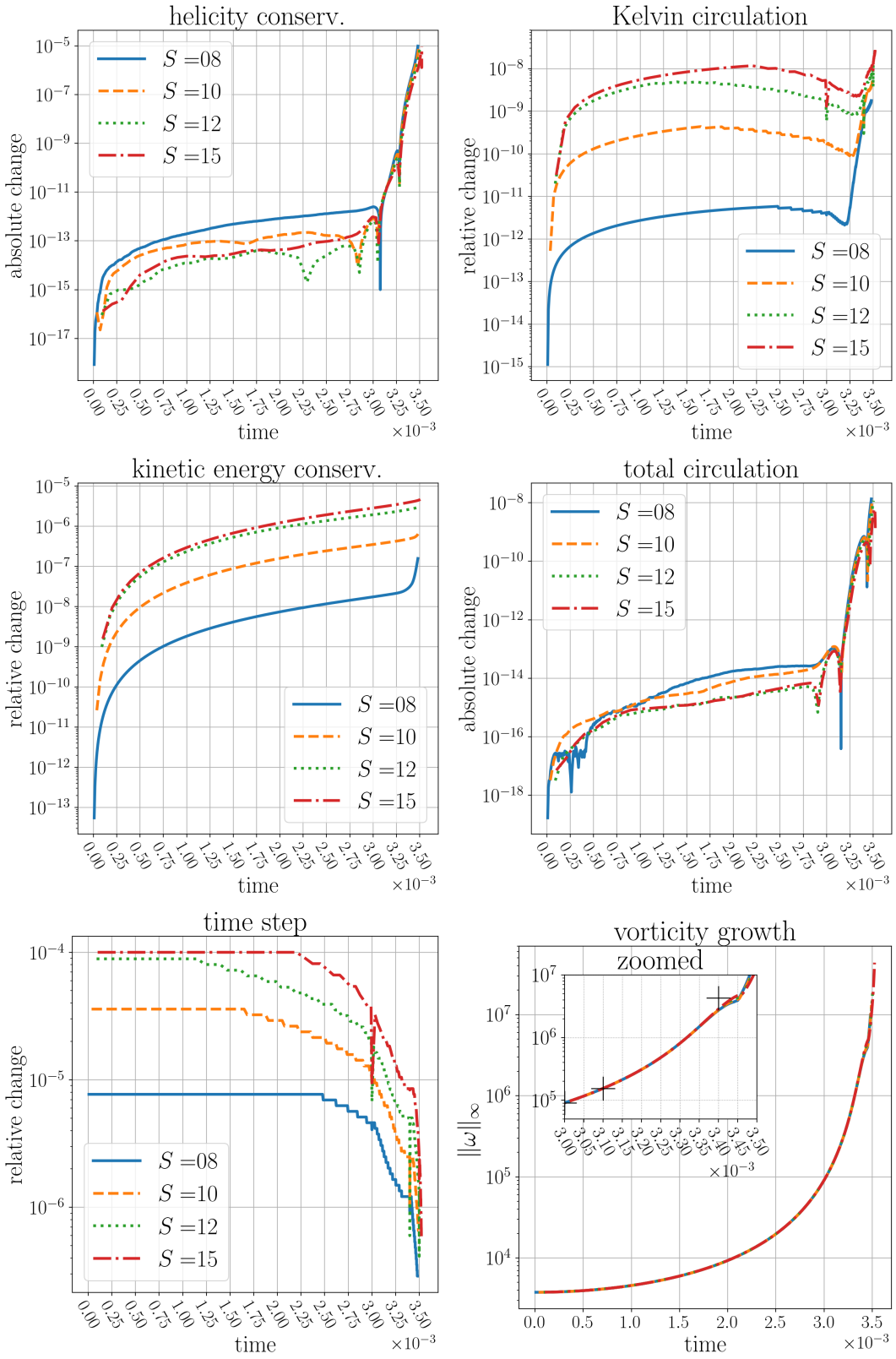


Figure 7.5: The effect of different truncation orders in the adaptive time-stepping from Section 5.4 is visualized. Higher orders imply larger time-steps and, thus, fewer iterations. These runs were executed on a 513×3072 grid, the other parameters are the same as in the plots from above. Sudden drops in the size of the time-steps in the lower left panel at $t = 0.003$ (and selected later times) are forced time-steps to arrive exactly at that time for a comparison to the values in Table 7.1.

several truncation orders. Higher numbers of S imply a larger time-step due to the growth behavior of the coefficients. It holds $|X^{(s)}| < c/\varrho^s$ for s large enough, where c is a constant and ϱ the radius of convergence. The time-step is chosen such that the time-Taylor sum and its first temporal derivative, truncated to S coefficients, is not altered by the last terms. This implies a bound on the last term in the time-Taylor sum of the from $|X^{(S)}|t^S < \varepsilon$, for some $\varepsilon > 0$. Thus, the bound from before allows choosing t closer to ϱ for larger S
As in Figure 7.4, the curves, belonging to a conserved quantity with different truncation orders, display very similar behavior. Higher truncation orders, thus larger time-steps, show a break out in the Kelvin conservation curves slightly later and seem to maintain the strong vorticity growth slightly longer than their lower order competitors. The helicity conservation curves are very close to each other, but the high orders give lower absolute changes than the small orders. The energy and Kelvin conservation curves behave seemingly opposite the helicity behavior. There, the relative errors are lower for lower orders, but those values only depend on the velocity field and are measured by a quotient, not a difference. In the CLA, it is the vorticity field from which a new iteration departs and therefore the focus should be laid more on the conserved quantities which concern the latter. Indeed, the total circulation $(\int \omega^\alpha \mathrm{d}x)$ behaves similarly to the helicity conservation.

The influence of the working precision is visualized in Figure 7.6, which shows the helicity error and vorticity growth for a run on 513×1024 points in all implemented working precisions. Unfortunately, we cannot go higher in resolution yet, because only the Bessel-Chebyshev solver works in multiple precisions and the Bessel matrices, which are needed in the solver, have only been calculated for the mentioned grid size. In the plots, we can see that the helicity values (which should be zero), start off at a lower level for a higher working precision, but, later, all values align and are indistinguishable. The different curves of maximal vorticity can also not be told apart, but a common fall-off and sudden break indicate their inaccurate behavior. The very similar outcomes might be tied to the low resolution of those runs, as higher precision cannot improve under-resolved simulations. If the resolution was sufficient, then we would be able to observe at least a slight increase of the total covered time span before the error bounds are reached, as it was seen for the stationary flow in Figure 6.8.

Dealiasing in the Fourier modes is investigated in Figure 7.7. Here, the much larger grid

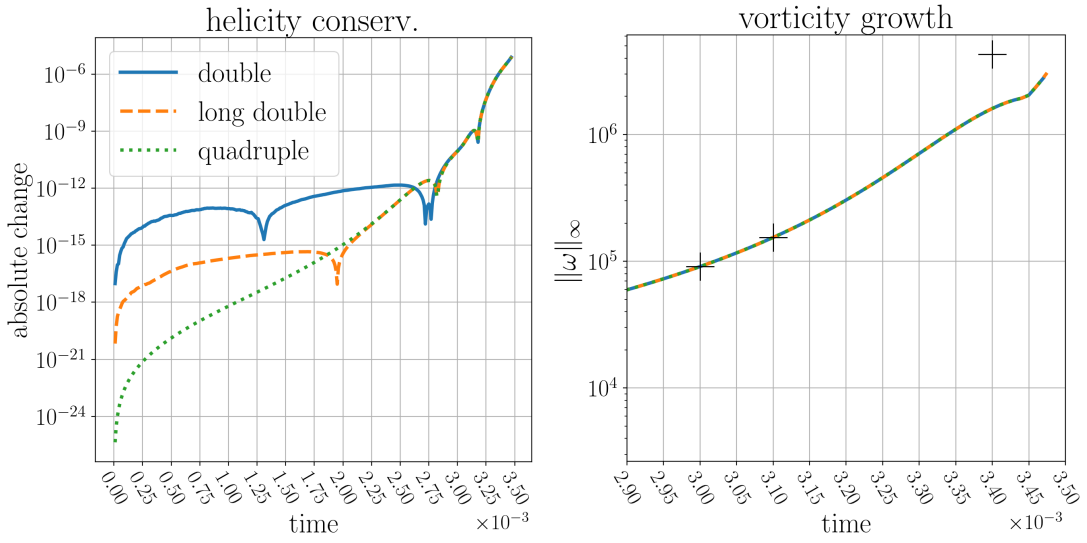


Figure 7.6: Multiple precision. The difference in vorticity growth for varying working precisions is merely noticeable. The low grid size of 513×1024 points seems to be the reason for this. The run-time parameters, apart from precision, were fixed and are as in Figure 7.2.

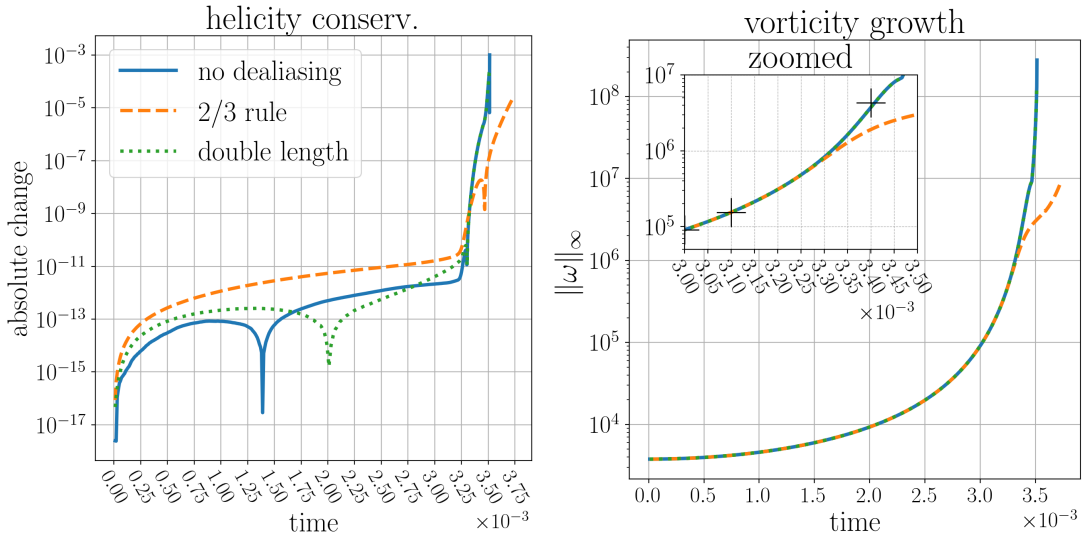


Figure 7.7: The effect of dealiasing is shown. The 2/3 rule crops the highest modes and leads to under resolution sooner. The double length method brings no improvement to the non-dealiased version.

size 1025×8192 has been chosen to see the difference more clearly. While the vorticity growth curves, which correspond to no aliasing and the double-length method, literally overlap, shows the 2/3 rule a fall-off. This is due to under-resolution in later times when the high Fourier modes of the flow fill up. The 2/3 rule fills the last third of the Fourier coefficient vector with zeros (in a representation without complex conjugates) and crops, therefore, the important smaller modes, which are needed to represent the small scales of the flow. For this reason, no dealiasing is used mostly.

So far we have seen that the result strongly depends on the parameter setting. Most important for a good outcome is a low to moderate interpolation order, as well as a not too small time-step. The strongest influence is given by the resolution and will be further investigated. Figure 7.8 shows a comparison between a few higher resolutions. The largest grid size, which has been plotted, is 1025×10240. The grid spacing at the boundary of this flow is $2.35 \cdot 10^{-6}$ in radial direction (due to the clustering of the Chebyshev extrema) and $1.63 \cdot 10^{-5}$ in vertical direction. Our mesh spacings are of course still very far from the mesh spacings that were achieved in [83] close to the reported singularity, but they are comparable to those in [6]. For higher resolutions, the curves belonging to conserved quantities stay longer in a regular shape and break their behavior later, when compared to lower resolution runs, which is also visible in Figure 7.2 from before. The later those breaks appear, the better is the approach of the maximal vorticity to the value at $t = 0.0034$ given in [83] and recalled in Table 7.1. This qualitative observation can be examined more closely, together with other values from Table 7.1, by comparing the exact values in Table 7.2.

Table 7.2: The body of this table states the computed maximal vorticity at selected times and grid sizes. The values in the last row are those of Table 7.1 and come from the given references.

time	$3.0 \cdot 10^{-3}$	$3.1 \cdot 10^{-3}$	$3.4 \cdot 10^{-3}$
513×1024	$9.08451276 \cdot 10^4$	$1.54167635 \cdot 10^5$	$1.60528443 \cdot 10^6$
513×4096	$9.08466826 \cdot 10^4$	$1.54277249 \cdot 10^5$	$3.13597705 \cdot 10^6$
1025×8192	$9.08466912 \cdot 10^4$	$1.54277317 \cdot 10^5$	$3.82533611 \cdot 10^6$
1025×10240	$9.08466912 \cdot 10^4$	$1.54277318 \cdot 10^5$	$3.94749487 \cdot 10^6$
ref. [83, 6]	$9.0847 \cdot 10^4$	$1.54276901 \cdot 10^5$	$4.3127 \cdot 10^6$

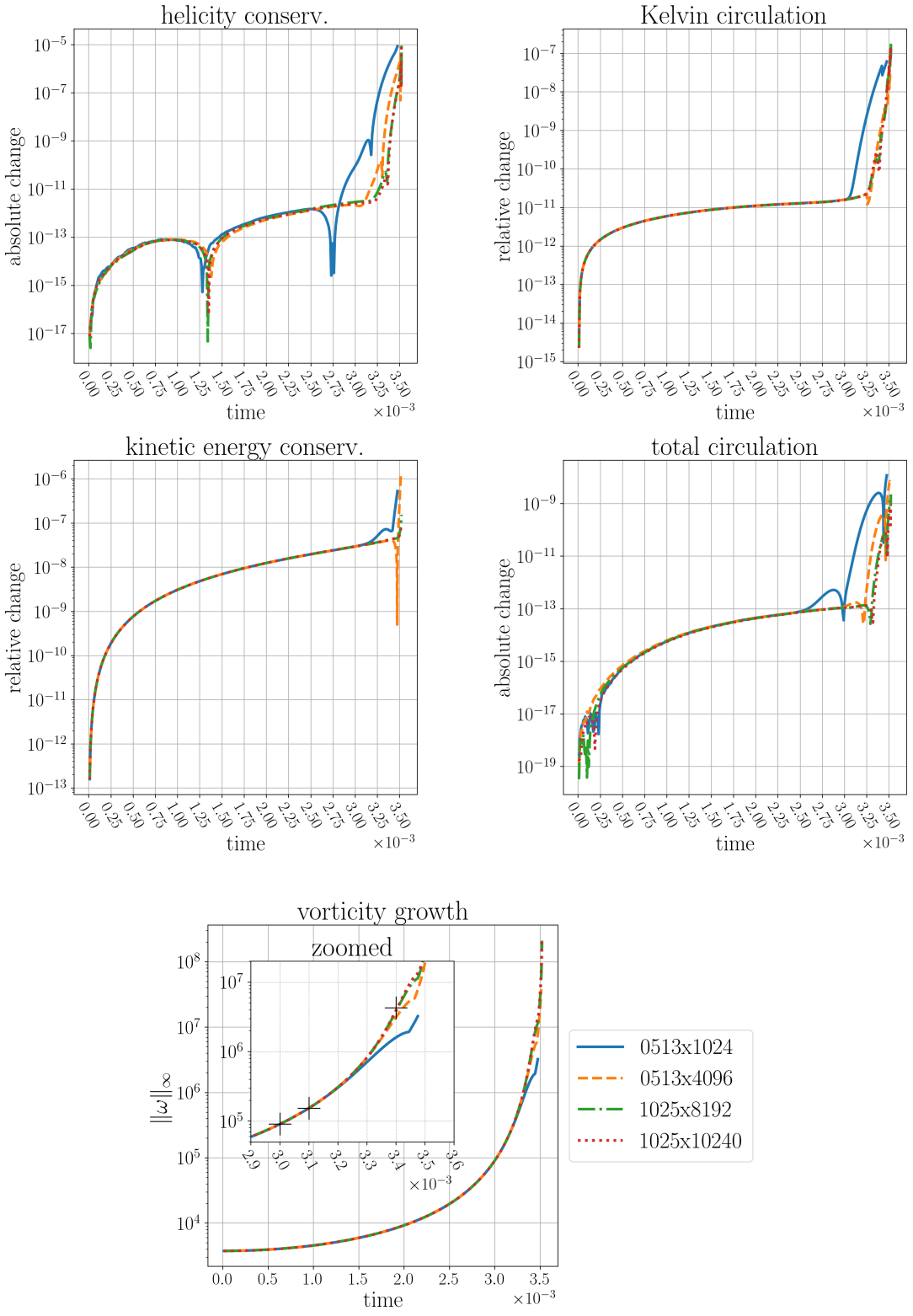


Figure 7.8: Higher resolution runs are shown. Here, the truncation order was set to $S = 18$ to maintain the time-step $\triangle t = 10^{-5}$ longer. The applied interpolation method is B-C-5, no refinement nor dealiasing was used. Sudden breaks in the conserved circulation quantities announce a fall-off in the vorticity growth.

Our maximal vorticity values in Table 7.2 show good agreement with those from the references for $t = 3 \cdot 10^{-3}$ and $t = 3.1 \cdot 10^{-3}$. The value at $3.4 \cdot 10^{-3}$ is approached, but only agrees in magnitude. Those are exceptional results, in particular, if the execution time of the simulation is considered. All of the computations presented in Table 7.2 and Figure 7.8 were carried out on a single CPU[1] and finished within one day. Typically, implementations of CFL bound algorithms need multiples of that time span and execute thousands of time-steps. The run of the CLA on the highest resolution in Table 7.2 reached $t = 3.1 \cdot 10^{-3}$ in only 310 iterations and $t = 3.4 \cdot 10^{-3}$ in 344.

Even larger grid sizes and runs with a higher working precision at those high resolutions are attainable by the use of MPI routines in comparable run times[2]. However, larger grid sizes or a higher working-precision certainly increase the memory consumption of the CLA. The current implementation is particularly memory demanding, because all the components of the coefficients, as well as their derivatives and divisions by r, as listed in (2.94), up to the truncation order S, are stored for direct insertion into the recurrence relations to avoid numerical errors from spectral differentiation schemes. That is, the terms listed in (2.94) are not obtained by spectral differentiation or division methods but constructed explicitly from the derivatives of the corresponding potentials, which are directly solved for. The wide palette of parameter settings, described in Section 6.1, also leaves a noticeable memory footprint. For the particular flow that is investigated here and in the given references, namely a blow-up candidate for the incompressible Euler equations, it is much better and more efficient to exploit the underlying flow symmetries as in [83] and [6], which allows modeling the flow in only a quarter of the periodicity domain. In the case of the CLA, such symmetry considerations would cut the memory demands tremendously and would thus allow to achieve highly resolved Euler simulations in a fraction of the time of CFL bound algorithms for the same problem. As the presented work gives a first realization of the CLA in a wall-bounded domain, it was intended to provide a code that can simulate any axisymmetric, periodic flow,

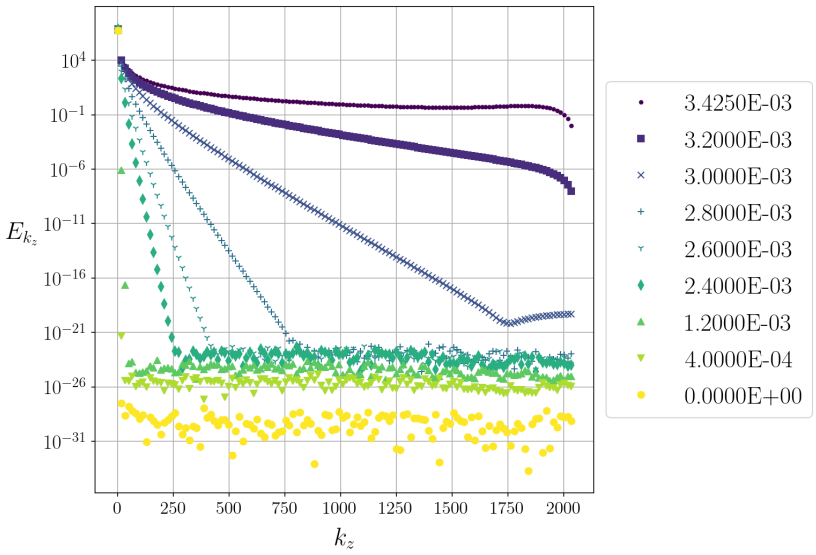


Figure 7.9: Energy spectrum at the boundary $r = 1$ for various time instances. Small modes fill up rapidly and the flow can soon not be resolved accurately anymore. The underlying grid size was fixed to 513×4096 Chebyshev-Fourier points. All other fixed run time parameters are given in the text.

[1] The cluster in use provides Intel Xeon Gold 6148 CPUs with 20 cores and 40 possible threads, which are individually supplemented by 192GB of memory (RAM).

[2] Technical issues after a cluster maintenance in the period when the here presented data was collected prevented the execution of runs through MPI routines.

which does not necessarily possess special symmetries.

Nonetheless, spectral simulations on a fixed grid might eventually not be enough to adequately reproduce the results from [83]. Indeed, the following investigations of the energy distribution suggest that the small scales of the potentially blowing-up flow fill up ever more rapidly the closer the predicted singularity time is approached. Figures 7.9 and 7.10 show the evolution of the energy spectrum along the z-direction

$$E_{k=k_z}(r) = \frac{1}{2}(|\hat{v}_k^r(r)|^2 + |\hat{v}_k^\alpha(r)|^2 + |\hat{v}_k^z(r)|^2), \tag{7.5}$$

where $\hat{v}_k^{(\cdot)}(r)$ are the Fourier transforms of the velocity components with wave number $k = k_z$, and r the radial argument.
All the figures that are referenced in the following were obtained from one program execution with the following run time parameters: the grid was set to 513×4096 Chebyshev-Fourier points in double precision, the 5th order B-spline cascade B-C-5 was used for the interpolations and no refinement or aliasing was enabled. The initial time-step was of the size 1.0×10^{-5} with truncation order $S = 15$, the Poisson problems were solved by the Chebyshev-Galerkin solver.
From Figure 7.9 we read that each increment of $\triangle t = 2 \times 10^{-4}$ in the time interval from 2.6×10^{-3} to 3.0×10^{-3} doubles the needed Fourier modes to accurately represent the velocity at the boundary. The velocity components in physical space are plotted in Figure 7.11, also for various times, and Figure 7.12 shows the corresponding vorticity components. See also Figures 7.14 and 7.15 for a display of their evolution in the (r, z)-plane. The top row of Figure 7.10 shows the vorticity growth as well as the computational conservation of some quantities next to the total kinetic energy of the flow. The other panels in the aforesaid figure show $\log_{10}(E_{k_z}(r))$ on the (r, k_z)-plane. We see a strong concentration of energy close to the boundary at $r = 1$, and, once the highest modes are filled, a rising error plateau is forming, which indicates a beginning numerical thermalization of the simulated flow. The error that stems from the inaccurate representation of the flow fields at later times is distributed over the whole (r, k_z)-plane. In the bottom row of Figure 7.10, actually two plateaus can be seen. The lower one corresponds to the even modes which should stay zero due to the preserved symmetries from before.
The filling of the (shifted) Chebyshev-Fourier modes of the vorticity field is visualized in Figure 7.13, where the logarithm of W_{kl} is plotted, where

$$W_{kl} = \sqrt{|\hat{\omega}_{kl}^r|^2 + |\hat{\omega}_{kl}^\alpha|^2 + |\hat{\omega}_{kl}^z|^2}, \quad k \in [0, M/2 + 1],\ l \in [0, N], \tag{7.6}$$

with $M = 4096$ and $N = 513$. The $\hat{\omega}_{kl}^r$ are the Chebyshev-Fourier modes of the vorticity components, as in eq. (3.2) with $X^{(s)}$ replaced by ω, only that here the modes are divided by $\sqrt{MN}$ after the unnormalized Chebyshev-Fourier transforms. In Figure 7.13, the Chebyshev modes seem to fill up more rapidly than the Fourier modes at first, but eventually the number of occupied Fourier modes strongly exceeds those of the Chebyshev spectrum, note the uniform ratio of the displayed axis. After the time instance $t = 3.0 \times 10^{-3}$, the resolution in z-direction is insufficient to represent the vorticity field accurately. An adaptive moving mesh seems to be an adequate, even necessary, tool to further investigate the flow closer to the reported blow-up time in [83]. Again, nothing hinders an implementation of the CLA with such refinement methods. An adaptive moving mesh might further exhibit the strengths of the CLA due to the improvement of accuracy in the interpolations that can be expected, paired with the large time-steps, which allow a low number of total iterations. $\sim$

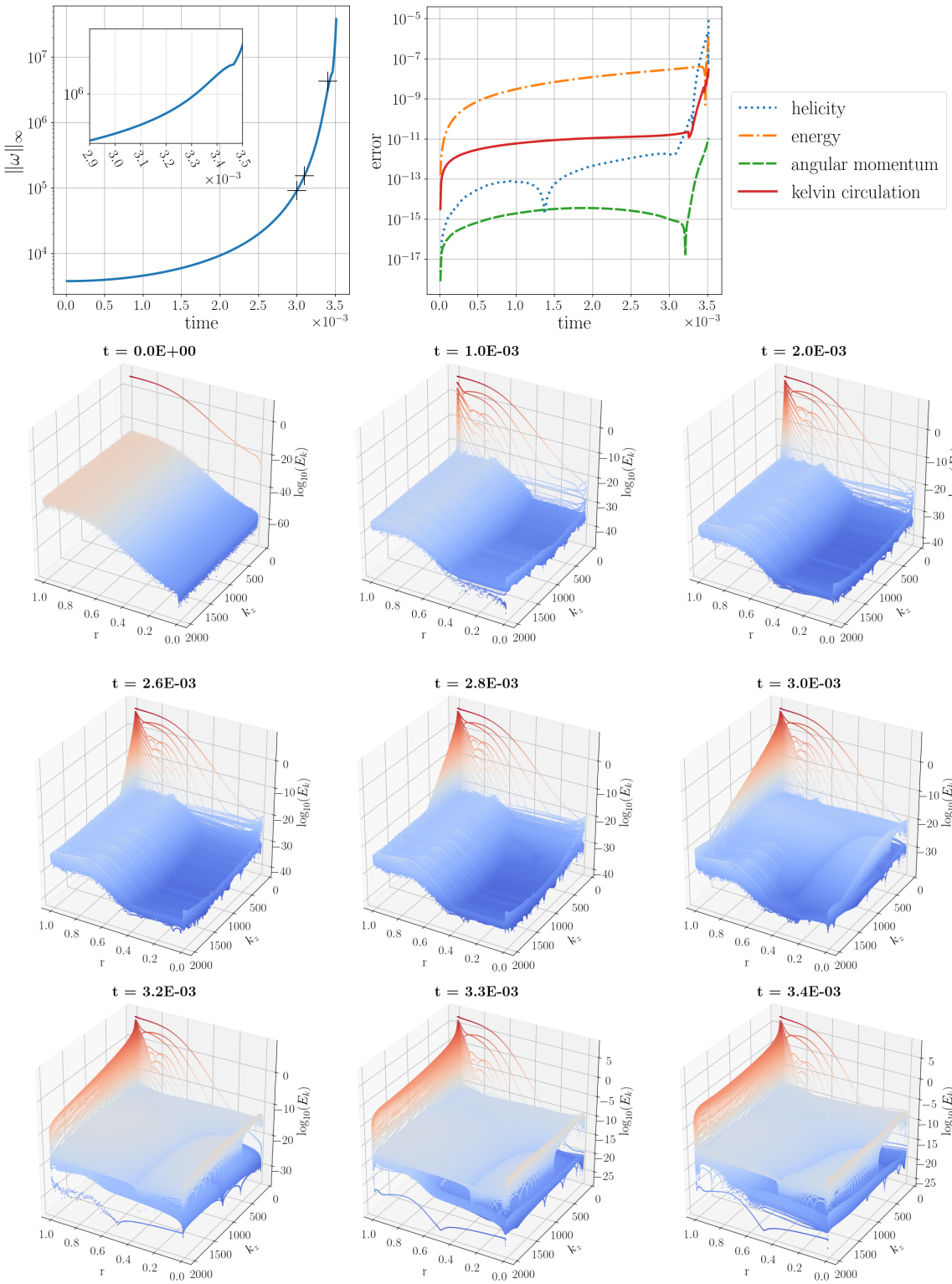


Figure 7.10: The top row shows the vorticity growth and conserved quantities for the run with underlying grid size 513×4096 and parameters as described in the text. Below, the energy spectrum in z-direction is plotted in dependence of r for different time instances. Only positive modes are displayed, and the top left panel only shows odd modes.

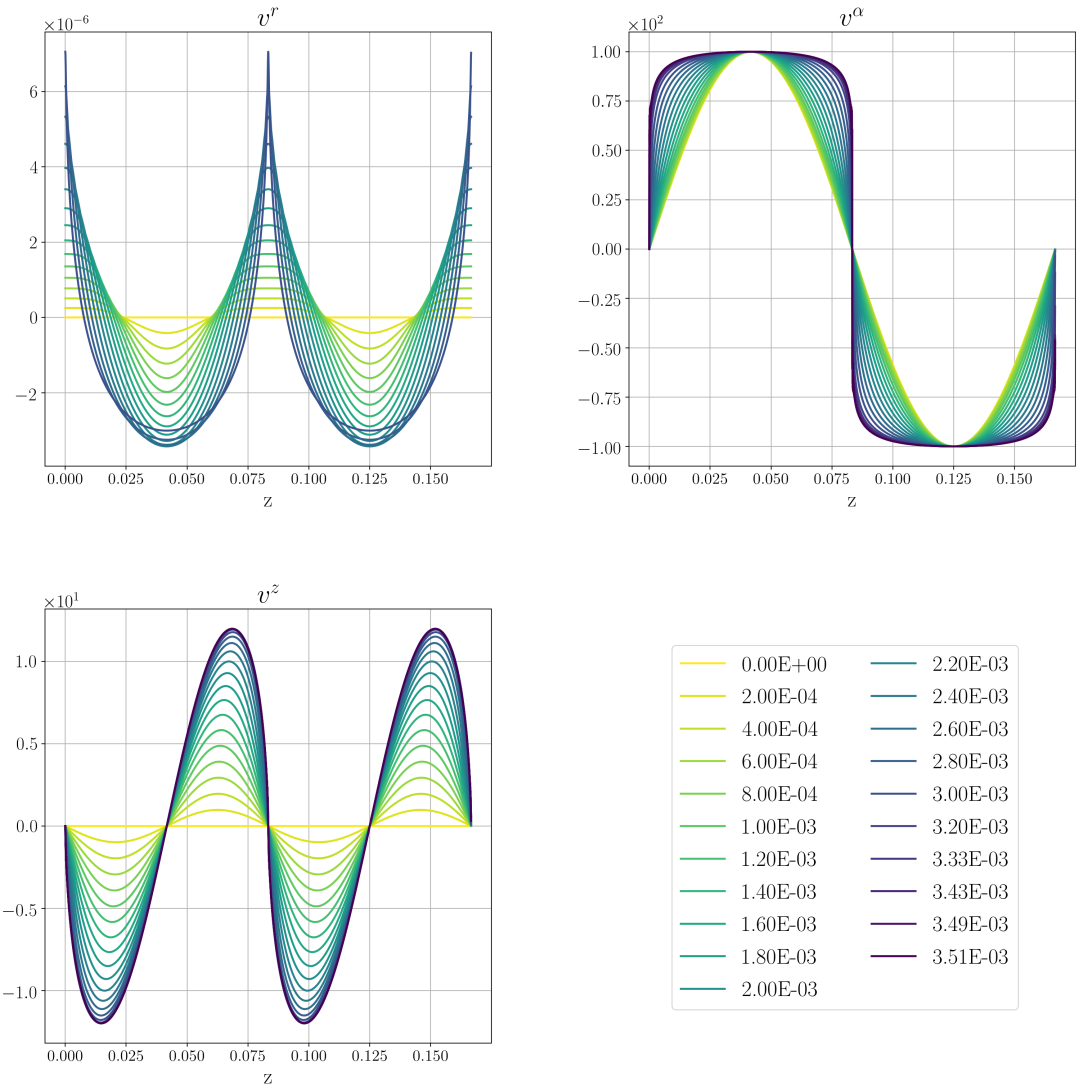


Figure 7.11: Plot of the velocity components at $r = 1$ for different times. The values were obtained from a simulation with 513×4096 grid points, the other run time parameters were the same as in the preceding figure. The velocity component v^r in the top left panel should stay zero at $r = 1$ due to the no-flow boundary condition, but large errors appear eventually at the symmetry axes of the flow.

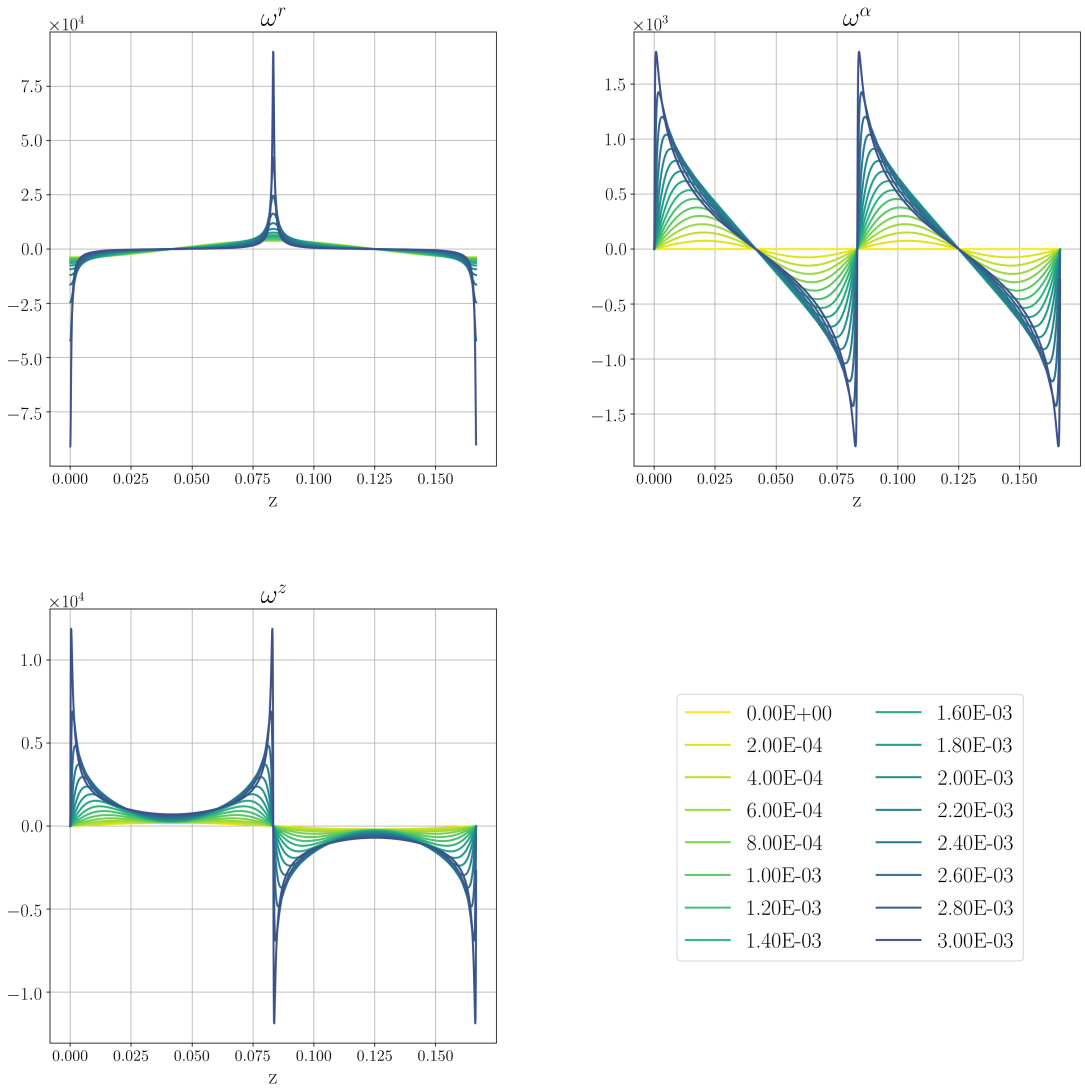


Figure 7.12: Plot of the vorticity components at $r = 1$ for different times. The underlying grid size for their computation was 513×4096 as in the preceding figures. The evolution of the components is only displayed until 3.0×10^{-3}, the growth of the local extrema is too extreme to be conveniently visualized.

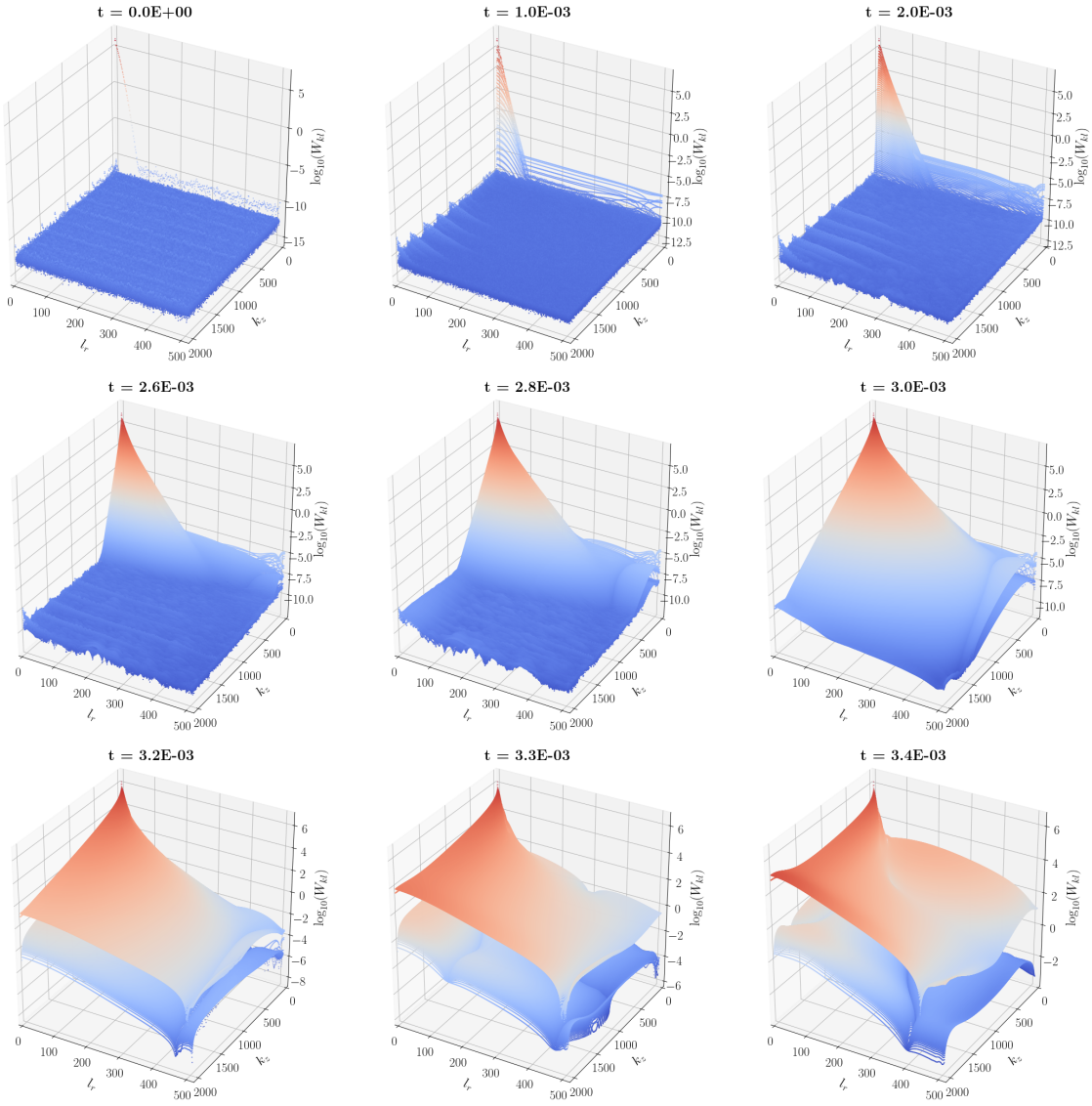


Figure 7.13: Visualization of the Chebyshev-Fourier modes of the vorticity field via the quantity W_{kl} from eq. (7.6). The Fourier modes fill up much more rapidly than the Chebyshev modes and are soon insufficient to accurately represent the vorticity field.

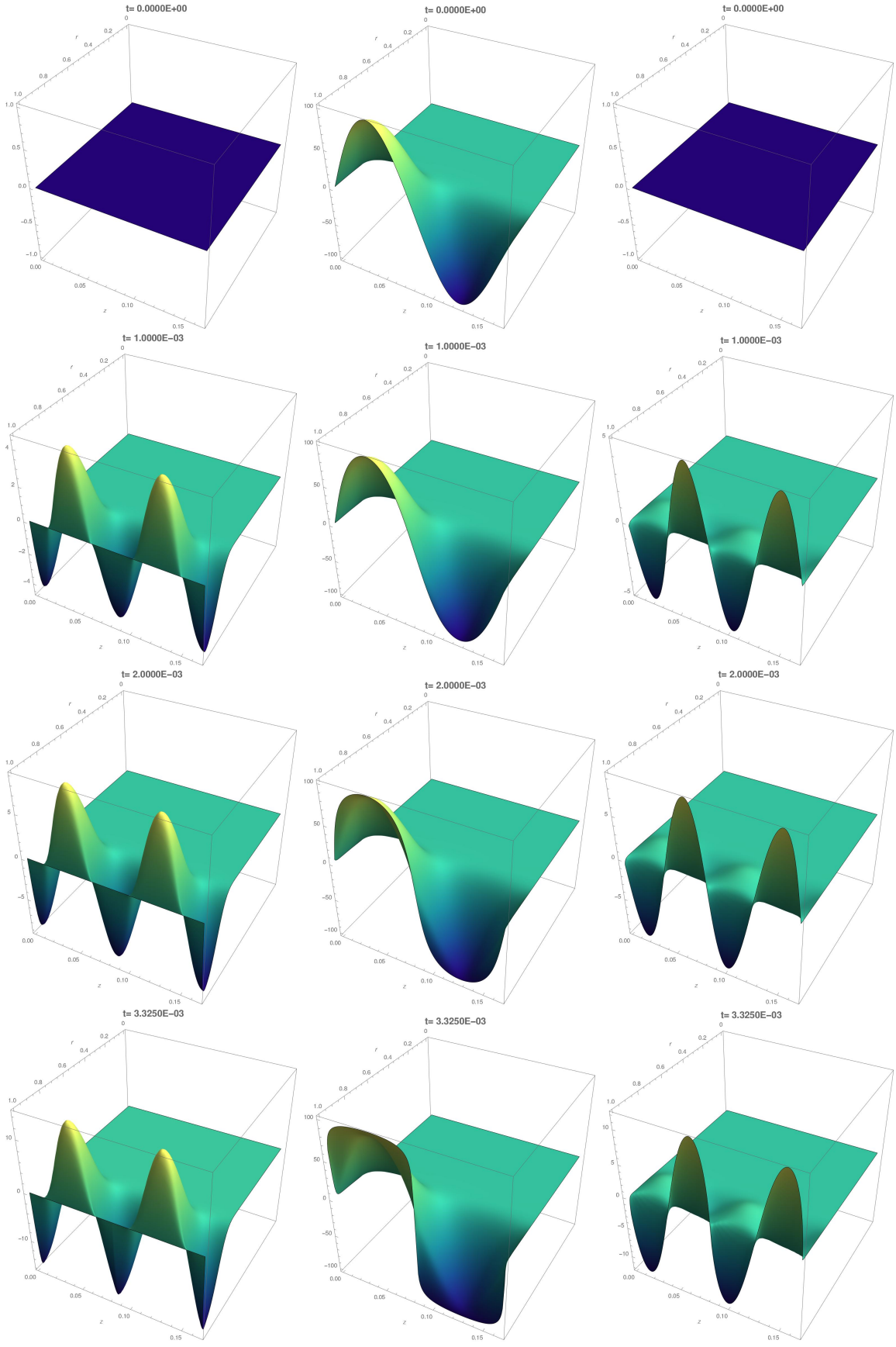


Figure 7.14: Evolution of the velocity components for the initial condition from reference [83]. The columns from left to right show the radial, angular and vertical components respectively. The axes ratios are uniform and the error of the radial component at $r = 1$ from Figure 7.11 can therefore not be seen.

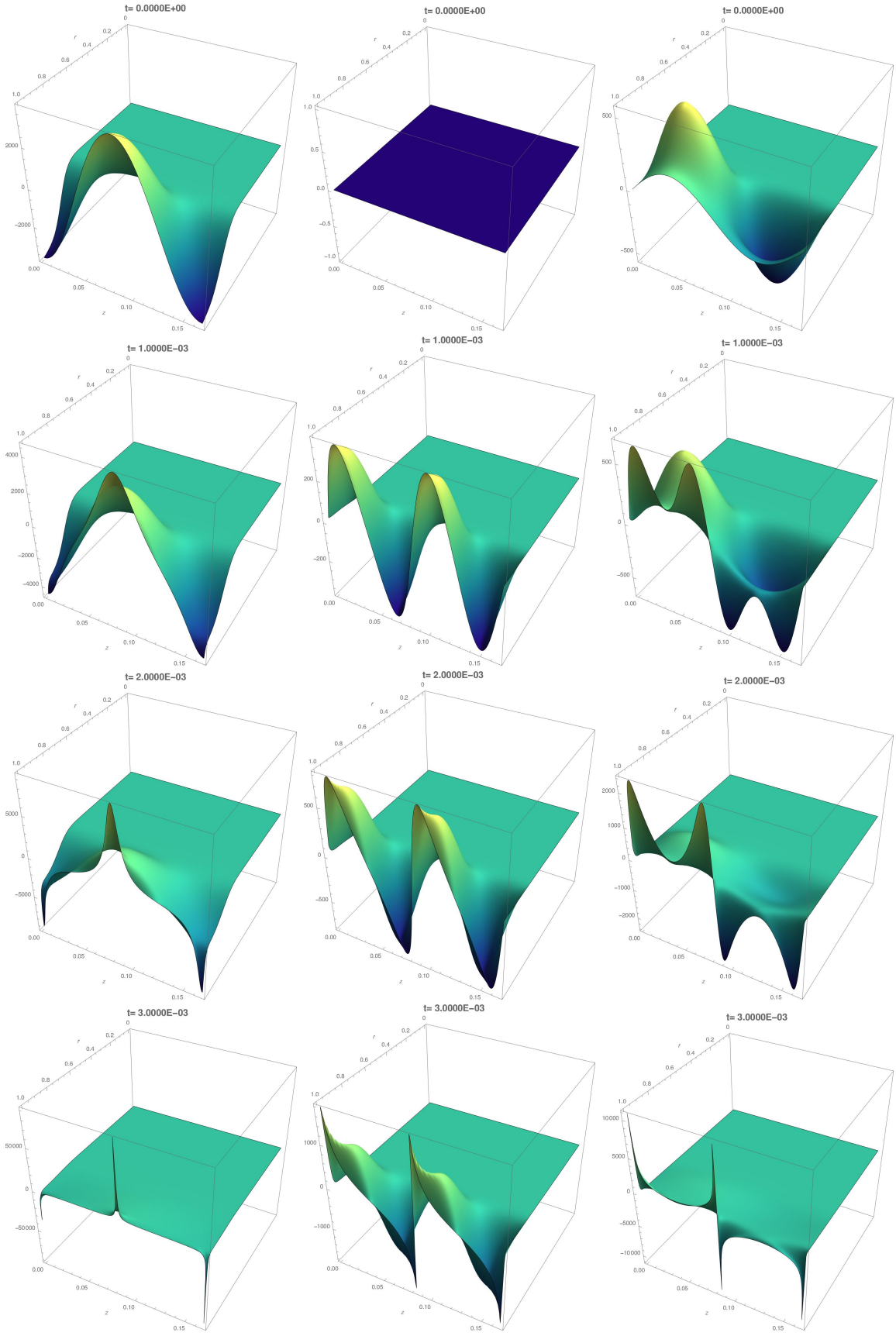


Figure 7.15: The corresponding vorticity field for Figure 7.14. The columns from left to right show the radial, angular and vertical vorticity components respectively.

Chapter 8

Conclusions and Outlook

This thesis provides a detailed numerical analysis of the novel Cauchy-Lagrange algorithm (CLA) for axisymmetric incompressible Euler flow. To the knowledge its author, this is the first time that the CLA is applied to a flow in a domain with boundaries. The algorithm is described for general smooth and bounded domains in Section 2.1 and, in Section 2.6, stated more precisely for the particular geometry of an infinite cylinder which contains a periodic and axisymmetric flow. For this purpose, the recursion relations for the time-Taylor coefficients of the particle trajectories $X = \sum_{s\geq 0} X^{(s)} t^s$, stated in [12] in Cartesian coordinates, were obtained from the cylindrical transform of the Cauchy-Invariants formula as stated in Section 2.3. The computational complexity of those recursion sums was reduced from $O(s^2)$ to $O(s)$ by the use of Poisson brackets. This reduction adds to the already simplified problems, which only depends on the radial and vertical argument due to the imposed axisymmetry. The Helmholtz-Hodge decomposition of the time-Taylor coefficients $X^{(s)}$ from [12] is given in its correct form in Section 2.5 and transformed into the cylindrical coordinate system. In a spectral discretisation with a (shifted) Chebyshev-Fourier basis, the connected Poisson problems decouple into sets of ordinary differential equations for the Fourier modes of the Helmholtz potentials (Section 3.3.1). Second order derivatives of those potentials, which are Calderon-Zygmund operators of order zero applied to the RHS function, compose the cylindrical components of $X^{(s)}$ and have to be found accurately to avoid an error cascade in the recurrence sums. For this reason, two numerical schemes were proposed to compute the Calderon-Zygmund operators. One of them is a modification of the Chebyshev-Galerkin solver from [108] and the second is a novel algorithm, which was developed in this thesis to solve directly for the second order derivatives of a Poisson solution. The latter algorithm can be seen as the numerical manifestation of the involved Calderon-Zgmund operators of order zero, as it implements the exact derivatives of an inverse Laplacian in a finite dimensional Chebyshev-Fourier spectral space. The linearity of the involved operators allows to obtain the result by simple matrix-vector multiplications. The various aspects of those two solvers together with their development were discussed in Section 3.3.1. Furthermore, the two dimensional interpolation, which is needed to gather the function values of the time advanced flow fields back on a fixed grid to restart the recursion process, is examined thoroughly in Chapter 4. Many interpolation schemes have been considered and tested for actual particle displacements. In this process, the cascade interpolation, which was already used in [95], has proven to give the best results with respect to accuracy and speed. Moreover, the complex matter of valid time-stepping techniques has been discussed in Chapter 5. Next to standard methods, as finding the convergence radius via Domb-Sykes plots, adaptive time stepping methods were proposed, which do not need to compute the convergence radius of the time-Taylor series of X. In addition, a time-stepping technique based on analytic continuation

was suggested in the aforesaid chapter. It allows to advance the trajectories and flow fields without the use of an interpolation onto a fixed grid between the time steps, the computed flow fields are then always known on the Lagrangian paths that depart from the original grid.

On the practical side, the author of this thesis has implemented the CLA for the given geometry from scratch and has produced a large amount of Fortran, C, C++, Python and Mathematica code to realize many numerical methods as part of the CLA implementation. The code itself is written in Fortran with parts in C/C++. Many routines and schemes were probed in the CLA and the most effective have been presented in this thesis. In the earlier development, other Poisson solvers have been investigated and developed but were later discarded because of large numerical overheads. The result of the programming efforts is a versatile code which runs in multiple working precisions and in parallel through a hybrid MPI-OpenMP parallelization. A number of parameters can be set before and within a program execution, an overview and description of the most important parameters can be found in Section 6.1. Many other switches can be found inside the code and allow plentiful variations of the CLA, but a complete representation of all the possible settings inside and outside of the code was not envisaged here. A minimal working example of the code will be made publicly available at a later time.

The implementation of the CLA and its components have been tested carefully and on many platforms. A number of different Poisson problems were regularly tested in order to ensure the functionality of the Poisson solvers, which are the heart of the CLA code. Apart from that, the code has mostly been tested with the stationary solutions from Section 6.3, the results have been discussed in Section 6.5. The implementation is found to be fast and effective. The speed is due to an efficient OpenMP parallelization, but, to a greater extent, to the possibility of making larger time steps within the CLA, when compared to CFL bound algorithms in computational fluid dynamics. Indeed, the absence of a CFL condition makes the CLA a strong competitor to traditional schemes. It produces equal results in a much shorter time, which means in return that higher resolutions can be achieved. Apart from speed gains, the CLA allows to investigate computationally the complex temporal plane of the particle trajectories and Euler solutions along those trajectories through the possibility of computing a high number of time-Taylor coefficients. However, it was found in Section 5.2, that the coefficients should be computed with higher precision for this task.

The tests with the stationary solutions showed robustness and fast convergence to the real steady-state flows. A method for computing analytically the time-Taylor coefficients of trajectories belonging to such stationary solutions was presented in Section 6.4. With the help of computer algebra systems, it is possible to compute those coefficients by symbolic differentiation, print their discrete values to file, and read them into the Fortran program for comparison. In this way, the effectiveness of the implemented methods was tested and verified in Section 6.4. In particular, the CLA, which uses the Bessel-Chebyshev solver in quadruple precision, has proven to give very accurate results with respect to the limited number of coefficients that were available. In summary, both implemented Poisson solvers produce accurate results and in the complete CLA with several time steps no difference could yet be spotted in the outcome.

The potentially singular solution from [83] was computed with widely varying parameter sets and good agreement has been found at least for times which are not too far from the reported singularity. The current implementation can momentarily not reproduce the extreme vorticity growth close to the singularity time, but the potential of the CLA as a whole is demonstrated, especially in view of the speed gains due to large time steps. A considerable difference between the algorithm in [83] and the here presented CLA is that our results are

obtained by computing the flow in the full cylinder, while the computations in the given reference were carried out on the quarter cylinder. Thus, our modeled domain is four times larger, which increases the numerical complexity and storage demand of the program. In spite of this disadvantage, we were sill able to verify a particular maximal vorticity number close to the reported singularity time, which has already been verified in [6] (also via computations on the quarter cylinder). In future implementations of the CLA, a restriction to the quarter cylinder, paired with an adaptive moving mesh as in [83] is imaginable and could produce even better agreement with the data given in the reference. Before that, a spectral discretisation that exploits the preserved symmetries of the potentially blowing-up flow can be tried on a fixed grid and would allow to reach very high resolutions in the quarter cylinder. While such an exploitation of symmetries is readily achieved in traditional Eulerian computations of the flow fields (by limiting the computations only to even or odd modes), the computations of the time-Taylor coefficients of the trajectories in the CLA do not allow such a simple approach and need a tracking of the parity throughout the whole simulation. A splitting of the corresponding equations of the recursion relations from Section 2.4 has been envisaged in the beginning of this Ph.D. project, but was later not further pursued in favor of a more general approach.

More generally, the current code of the CLA can still be improved by implementing the special linear system solvers for the Chebyshev-Galerkin method that are proposed in [107, 108]. Those methods allow to solve the resulting linear systems in the Chebyshev-Galerkin solver in linear complexity and could be implemented for multiple working precisions. The linear complexity of those solvers opens the door to quadruple precision simulations on high resolutions in using the general speed advantage of the CLA. Apart from those aspirations, it could be useful to study the CLA on a theoretical level and prove the mathematical convergence of the global method. This convergence study has not been carried out in this thesis, the ambition to build an accurate, fast and efficient CLA implementation, which could possibly verify the reported singularity in [83], prevailed. Another important question is how the CLA would perform in a fully three-dimensional flow, as the implemented algorithm here is quasi-2D due to the imposed axisymmetry of the flow. Possible issues with a CLA in full 3D can come from storage shortages due to the large amount of data that is needed to store the coefficients, but this problem might be solved soon with next generation supercomputers. The cascade interpolation would probably perform well in three-dimensions as well, and proper parallelization can help with potential speed problems.

In short, the research presented in this thesis has built upon previous work, contributed novel results, insights and numerical tools to the field, and at the same time it has prepared the ground for many natural future developments.

Appendices

Appendix A

Special Functions

A.1 Modified Bessel Functions

The knowledge on (modified) Bessel functions is extensive and so is the literature on it, see [15, 73, 92] as standard references. Here, we solely gather, from the given references, the definitions and basic properties of the modified Bessel functions I_ν and K_ν, for $\nu = 0, 1, \ldots$. Those are used in the current implementation of the CLA as part of the exact solutions to the truncated Poisson problems from Section 2.5. Let $n \in \mathbb{N}_0 = \{0, 1, \ldots\}$, then I_n and K_n are the fundamental solutions to

$$z^2 w''(z) + z w'(z) - (z^2 + n^2) w(z) = 0, \tag{A.1}$$

for $z \in \mathbb{C} \setminus (-\infty, 0]$. As our computations do not leave the real number space, we assume furthermore, that $z \in (0, +\infty)$. A series representation serves as an explicit definition for I_n,

$$I_n(z) = (\tfrac{1}{2}z)^n \sum_{k=0}^{\infty} \frac{(\frac{1}{4}z^2)^k}{k!\Gamma(n+k+1)}, \tag{A.2}$$

where Γ denotes the Gamma function, such that $\Gamma(n+k+1) = (n+k)!$. K_n is commonly defined as

$$K_n(z) = \lim_{\nu \to n} \tfrac{1}{2}\pi \frac{I_{-\nu}(z) - I_\nu(z)}{\sin(\nu\pi)},$$

and is explicitly given by

$$\begin{aligned} K_n(z) = \tfrac{1}{2}(\tfrac{1}{2}z)^{-n} \sum_{k=0}^{n-1} \frac{(n-k-1)!}{k!}(-\tfrac{1}{4}z^2)^k + (-1)^{n+1} \ln\left(\tfrac{1}{2}z\right) I_n(z) \\ + (-1)^n \tfrac{1}{2}(\tfrac{1}{2}z)^n \sum_{k=0}^{\infty} \left(\frac{\Gamma'(k+1)}{\Gamma(k+1)} + \frac{\Gamma'(n+k+1)}{\Gamma(n+k+1)} \right) \frac{(\frac{1}{4}z^2)^k}{k!(n+k)!}, \end{aligned}$$

which simplifies, for $n = 0$, to

$$K_0(z) = -\left(\ln\left(\tfrac{1}{2}z\right) + \gamma\right) I_0(z) + 2\sum_{k=1}^{\infty} \frac{I_{2k}(z)}{k} \tag{A.3}$$

$$= -\left(\ln\left(\tfrac{1}{2}z\right) + \gamma\right) I_0(z) + \frac{\frac{1}{4}z^2}{(1!)^2} + (1 + \tfrac{1}{2}) \frac{(\frac{1}{4}z^2)^2}{(2!)^2} + (1 + \tfrac{1}{2} + \tfrac{1}{3}) \frac{(\frac{1}{4}z^2)^3}{(3!)^2} + \cdots, \tag{A.4}$$

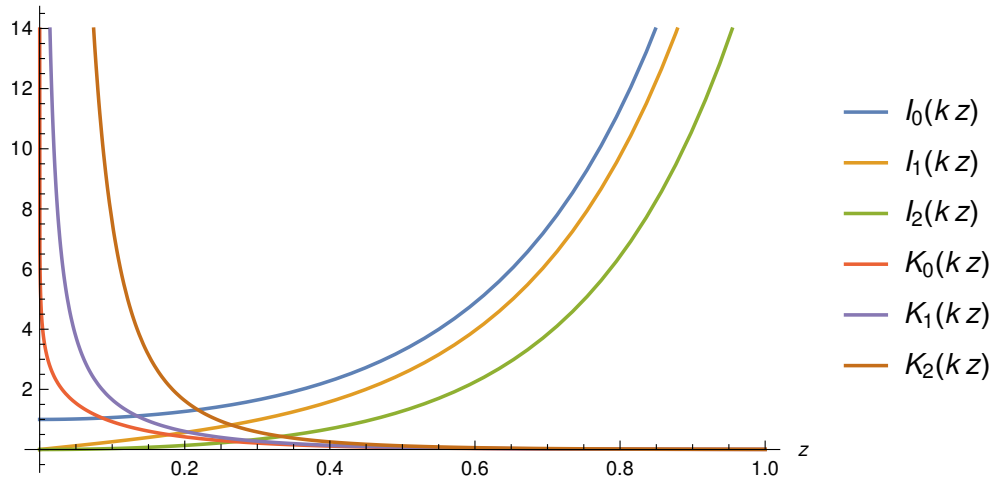


Figure A.1: The modified Bessel functions of orders 0 to 2 in the interval $[0, 1]$ with the argument (kz), where $k = 5$.

with Euler's constant $\gamma = \lim_{n\to\infty}\left(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} - \ln n\right) = 0.57721\ \ldots$.

Also, concise integral forms can be given,

$$I_n(z) = \frac{1}{\pi}\int_0^{\pi} e^{z\cos\theta}\cos(n\theta)\,\mathrm{d}\theta, \tag{A.5}$$

$$K_n(z) = \int_0^{\infty} e^{-z\cosh t}\cosh(nt)\,\mathrm{d}t\,. \tag{A.6}$$

The above series representations imply asymptotic terms for small arguments $z << 1$

$$I_0(z) \sim 1$$

$$I_n(z) \sim \tfrac{1}{n!}(\tfrac{1}{2}z)^n,\ n > 0 \tag{A.7}$$

$$K_0(z) \sim -\ln z \tag{A.8}$$

$$K_n(z) \sim \tfrac{1}{2}(n-1)!(\tfrac{1}{2}z)^{-n},\ n > 0, \tag{A.9}$$

and, for $z \to \infty$,

$$I_n(z) \sim e^z/\sqrt{2\pi z} \tag{A.10}$$

$$K_n(z) \sim \sqrt{\pi/(2z)}e^{-z}\,. \tag{A.11}$$

Furthermore, there hold the derivative relations

$$I_n'(z) = \frac{1}{2}(I_{|n-1|}(z) + I_{n+1}(z)) \tag{A.12}$$

$$K_n'(z) = -\frac{1}{2}(K_{|n-1|}(z) + K_{n+1}(z)), \tag{A.13}$$

such that $I_0' = I_1$ and $K_0' = -K_1$. Other interesting relations are the Wronskian

$$\mathcal{W}\{K_n(z), I_n(z)\} = I_n(z)K_{n+1}(z) + I_{n+1}(z)K_n(z) = 1/z. \tag{A.14}$$

and

$$\mathcal{Z}_{n-1}(z) - \mathcal{Z}_{n+1}(z) = \frac{2n}{z}\mathcal{Z}_n(z), \tag{A.15}$$

for $\mathcal{Z}_n \in \{I_n, \mathrm{e}^{n\pi\mathrm{i}}\, K_n\}$.

A.2 Verification of the Analytical Solutions to Equations (3.89) and (3.90)

Instead of deriving the explicit solutions to eqs. (3.89) and (3.90) in applying the method of variation of parameters, we simply verify for the Neumann problem that the terms in Section 3.3.3 actually solve the equations and verify the boundary conditions. The proof for the solution to the Dirichlet problem is done in an analogous way and omitted here. Note, that eq. (A.1) has to be modified in order to represent the operators in the referenced problems. For clarification, if I_n and $K_n, n \in \mathbb{N}_0$, are fundamental solutions to eq. (A.1), then $I_n(k\,.\,)$ and $K_n(k\,.\,)$ are fundamental solutions to

$$z^2 w''(kz) + zw'(kz) - (k^2 z^2 + n^2) w(kz) = 0, \tag{A.16}$$

due to the variable transform $z \to kz$. Furthermore, if the boundary terms are fixed in eqs. (3.89) and (3.90), then those equations may be safely multiplied by r^2 and we obtain the non-homogeneous version of eq. (A.16) to which the method of variation of parameters may be applied. A direct verification of the solutions, however, will not only verify the solutions but also produce the derivatives with respect to r.

Lemma A.1. The solution to the problem (3.89), here in similar but not identical notation, $u''(r) + u'(r)/r - k^2 u(r) = f$, $u'(0) = u'(1) = 0$, is given by

$$u(r) = I_0(kr) \cdot \mathcal{J}[K_0 f](k,r) - K_0(kr) \cdot \mathcal{J}[I_0 f](k,r) + c I_0(kr), \tag{A.17}$$

with $c = -\mathcal{J}[K_0 f](k,1) + \frac{K_1(k)}{I_1(k)} \cdot \mathcal{J}[I_0 f](k,1)$, where $\mathcal{J}[I_0 f](k,r) = \int_0^r t I_0(kt) f(t)\,\mathrm{d}t$ and similar for $\mathcal{J}[K_0 f](k,r)$. Here, we replaced $\hat{u}_k$ and $\hat{f}_k$ by u and f respectively, as well as $\tilde{k}$ by k in (3.89) for simplification.

Proof. The function eq. (A.17) may directly be differentiated in using eqs. (A.12) and (A.13) to give

$$\begin{aligned} u'(r) &= kI_1(kr)\mathcal{J}[K_0 f](k,r) + rI_0(kr)K_0(kr)f(r) \\ &\qquad + kK_1(kr)\mathcal{J}[I_0 f](k,r) - rK_0(kr)I_0(kr)f(r) + ckI_1(kr) \\ &= kI_1(kr)\left(\mathcal{J}[K_0 f](k,r) + c\right) + kK_1(kr)\mathcal{J}[I_0 f](k,r)\,. \end{aligned} \tag{A.18}$$

The latter is differentiated once more to entail

$$\begin{aligned} u''(r) &= kI_1'(kr)\left(\mathcal{J}[K_0 f](k,r) + c\right) + rkI_1(kr)K_0(kr)f(r) \\ &\qquad + kK_1'(kr)\mathcal{J}[I_0 f](k,r) + rkK_1(kr)I_0(kr)f(r) \\ &= \tfrac{1}{2}k^2(I_0(kr) + I_2(kr))\left(\mathcal{J}[K_0 f](k,r) + c\right) + rkI_1(kr)K_0(kr)f(r) \\ &\qquad - \tfrac{1}{2}k^2(K_0(kr) + K_2(kr))\mathcal{J}[I_0 f](k,r) + rkK_1(kr)I_0(kr)f(r) \\ &= \frac{1}{2}k\Big(- k(K_0(kr) + K_2(kr))\mathcal{J}[I_0 f](k,r) + k\big(I_0(kr) + I_2(kr)\big)\left(\mathcal{J}[K_0 f](k,r) + c\right) \\ &\qquad + 2rf(r)\left(I_0(kr)K_1(kr) + I_1(kr)K_0(kr)\right)\Big). \end{aligned} \tag{A.19}$$

Before we insert these derivatives into the differential equation, which is restated in the

lemma, we will examine the term $u'(r)/r$. We have

$$\begin{aligned}\frac{u'(r)}{r} &= \frac{1}{r}\Big(kI_1(kr)(\mathcal{J}[K_0 f](k,r)+c) + kK_1(kr)\mathcal{J}[I_0 f](k,r)\Big)\\ &= k^2\frac{I_1(kr)}{kr}(\mathcal{J}[K_0 f](k,r)+c) + k^2\frac{K_1(kr)}{kr}\mathcal{J}[I_0 f](k,r)\\ &= \tfrac{1}{2}k^2(I_0(kr)-I_2(kr))(\mathcal{J}[K_0 f](k,r)+c) - \tfrac{1}{2}k^2(K_0(kr)-K_2(kr))\mathcal{J}[I_0 f](k,r),\end{aligned} \tag{A.20}$$

where we have used eq. (A.15). An insertion of eqs. (A.19) and (A.20) into the differential equation produces, under consideration of eq. (A.14),

$$\begin{aligned}&\Big(\tfrac{1}{2}k^2(I_0(kr)+I_2(kr))\,(\mathcal{J}[K_0 f](k,r)+c) - \tfrac{1}{2}k^2(K_0(kr)+K_2(kr))\mathcal{J}[I_0 f](k,r)\\ &\qquad + krf(r)\,(I_0(kr)K_1(kr)+I_1(kr)K_0(kr))\Big)\\ &\qquad + \tfrac{1}{2}k^2(I_0(kr)-I_2(kr))(\mathcal{J}[K_0 f](k,r)+c) - \tfrac{1}{2}k^2(K_0(kr)-K_2(kr))\mathcal{J}[I_0 f](k,r)\\ &\qquad - k^2\Big(I_0(kr)(\mathcal{J}[K_0 f](k,r)+c) - K_0(kr)\cdot\mathcal{J}[I_0 f](k,r)\Big)\\ &\qquad = f(r),\end{aligned}$$

and leaves us with the verification of the boundary contitions. The equality $u'(1)=0$ is readily verified by inserting the constant c into $u'(1)$. For $u'(0)$, a limit has to be taken

$$\begin{aligned}&\lim_{r\to 0}\Big(kI_1(kr)(\mathcal{J}[K_0 f](k,r)+c) + kK_1(kr)\mathcal{J}[I_0 f](k,r)\Big)\\ &= \lim_{r\to 0}\Big(k\frac{z}{2}(\int_0^r t(-\ln t)f(t)\,\mathrm{d}t + c) + k\frac{1}{r}\int_0^r t(\tfrac{t}{2})f(t)\,\mathrm{d}t\Big)\\ &= 0,\end{aligned}$$

where we have inserted the asymptotic values from eqs. (A.7) to (A.9) and used the fact that f is at least Hölder continuous in $[0,1]$. □

A.3 B-spline Functions and Curves

The B-spline functions were originally defined via divided differences of truncated power functions [35, 103], but a defining recurrence formalism, introduced in [33, 38], has prevailed. The following definitions and properties can be found, for instance, in [94, 37].

Definition. For $m > 0$, let $U = \{u_0, ..., u_m\}$ be a set of a non-decreasing sequence of real numbers, such that $u_i \leq u_{i+1}, i = 0, \ldots, m-1$. U is called the knot vector with knots u_i. The *i-th B-spline function* of order $p+1$ and degree p, denoted by $N_{i,p}(u)$, is defined as

$$N_{i,0}(u) := \begin{cases} 1, & \text{if } u_i \leq u < u_{i+1} \\ 0, & \text{other-wise} \end{cases} \tag{A.21}$$

$$N_{i,p}(u) := \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u), \tag{A.22}$$

where the quotients are defined to be zero, if the denominators vanish.

From the definition we read the following properties of the B-spline functions:

- $N_{i,0}$ is a step function, equal to zero everywhere except in the half-open interval $[u_i, u_{i+1})$, which is also called the knot span. Note that the knot span may be of length zero.
- $N_{i,p}, p > 0$, is a linear combination of two $(p-1)$-degree basis functions.
- $N_{i,p}(u) = 0$, if u is outside the interval $[u_i, u_{i+p+1})$
- In any given knot span $[u_j, u_{j+1}), j \in \{0, \ldots, m-1\}$, at most $p+1$ of the $N_{i,p}$ are non-zero, namely $N_{j-p,p}, \ldots, N_{j,p}$.
- $N_{i,p}(u) \geq 0$ for all i, p, and u.
- Partition of unity: for fixed i and any $u \in [u_i, u_{i+1})$, there holds

$$\sum_{j=i-p}^{i} N_{j,p}(u) = 1. \tag{A.23}$$

- At a knot u_j, $N_{i,p}(u)$ is $p-k$ times continuously differentiable, where k is the number of times that u_j appears in the knot vector U (multiplicity of u_j in U). In the interior of any knot span the B-spline functions are of class C^∞.
- The derivative of a B-spline basis function is given by

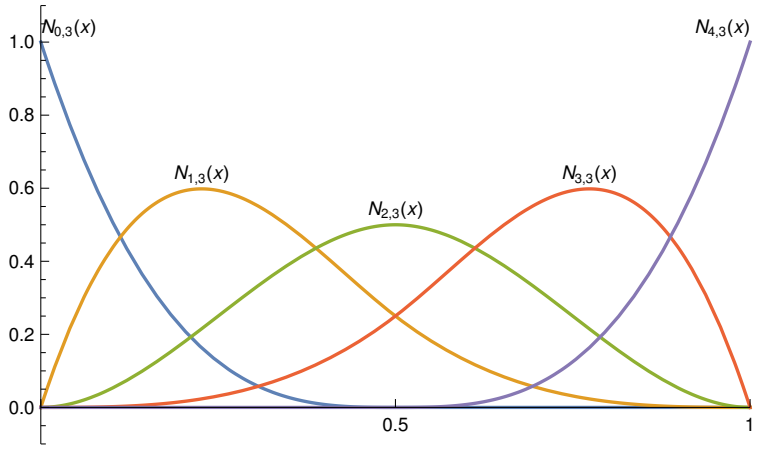
$$N'_{i,p}(u) = \frac{p}{u_{i+p} - u_i} N_{i,p-1}(u) - \frac{p}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u), \tag{A.24}$$

  for $p > 0$. A repeated application of this formula yields

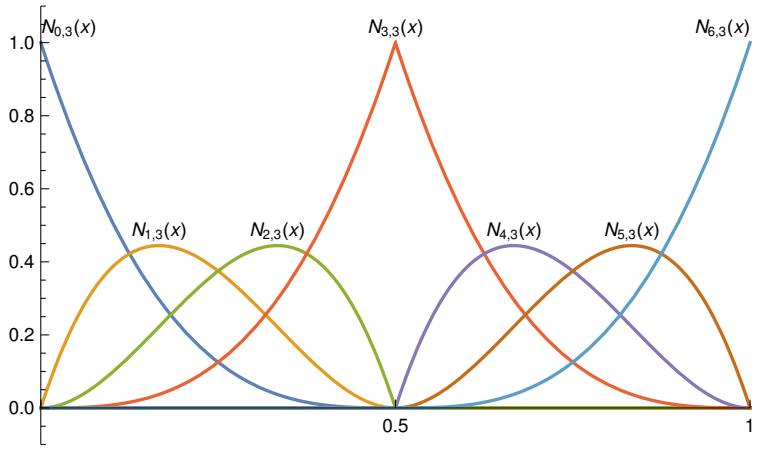
$$N^{(k)}_{i,p}(u) = \frac{p}{u_{i+p} - u_i} N^{(k-1)}_{i,p-1}(u) - \frac{p}{u_{i+p+1} - u_{i+1}} N^{(k-1)}_{i+1,p-1}(u), \tag{A.25}$$

  for $k \leq p$.

A visualization of some B-spline functions is given in Figure A.2.



(a)



(b)

Figure A.2: B-spline basis functions of order 4 (degree 3) on the knot vector (a) $U = \{0, 0, 0, 0, 1/2, 1, 1, 1, 1\}$ and (b) $U = \{0, 0, 0, 0, 1/2, 1/2, 1/2, 1, 1, 1, 1\}$

1D interpolation using B-splines Suppose we are given $n \in \mathbb{N}$ coordinates $p_j = (\tau_j, g(\tau_j)$ of a function g evaluated on a strictly increasing one dimensional grid $\tau_0 < \tau_1 < \ldots < \tau_{n-1}$. In order to interpolate the function g between the abscissæ one tries to find weights α_i such that

$$\sum_{i=0}^{n} \alpha_{i-1} N_{i,p}(\tau_j) = g(\tau_j), \tag{A.26}$$

where the B-splines $N_{i,p}$ of degree p are associated to a knot vector $U = \{u_0, \ldots, u_m\}, m \in \mathbb{N}$, such that $\dim\{(N_{i,p})_{i=0,\ldots,m}\} = n$. The following theorem states when this is possible.

Theorem 1 (Schoenberg-Whitney)**.** *Under the assumptions made above, the linear system that corresponds to eq. (4.17) is uniquely solvable, if and only if*

$$N_{i,p}(\tau_i) > 0, \quad \forall i = 0, \ldots, n-1. \tag{A.27}$$

This holds in particular if $u_i < \tau_i < u_{i+p+1}$ or, if $\tau_i = u_i$ and u_i has multiplicity $p+1$ in U.

See [37] for a proof. As a matter of fact, the matrix that corresponds to the linear system has a band structure with less than p sub-/super-diagonals, which implies a linear complexity for the underlying computational algorithm. In [37], a number of Fortran code pieces are

given, which implement the described interpolation scheme along with numerous other results on B-splines, such as differentiation and efficient evaluation of linear combinations of B-spline functions.

Appendix B

Useful Mathematica Notebooks

B.1 Exact Integration of Matrix Entries

The following notebook demonstrates the usage of the symbolic integration capability of Mathematica to obtain the exact analytical entries of the matrix $\tilde{A}$ from section 3.3.2. The band structure of the latter matrix is exploited in the way that the diagonal entries, diagonals seen as vectors, only depend on the row number. The first row and first column of the matrix have to be verified additionally, because of the special values $i = 0$ or $l = 0$. Here, only the second entry in the first column, corresponding to the first entry of the first sub-diagonal, has to be slightly adapted as indicated in eq. (3.108). One easily shows that the entries outside the seven main diagonals evaluate to zero. The matrices $\tilde{B}$ and $\tilde{C}$ are computed in the exact same way, only that the integrand is changed accordingly. The commands can be copied from the verbatim text further below.

In[29]:=
```
(* Definition of the basis functions *)
phi[l_, x_] := (1 - x^2) * ChebyshevT[l, x]
ω[t_] := 1/Sqrt[1 - t^2]

(* The following integrand is that of Matrix Ã *)
integrand[t_, m_, l_] := (t + 1) D[phi[l, t], {t, 1}] * D[phi[m, t] * ω[t], t]

(* A change of variables has to be
 performed in order to make this method work *)
changed = Evaluate@integrand[t, m, l] /. t → Cos[θ];

syma[θ_, m_, l_] := Evaluate@FullSimplify[Sin[θ] * changed, Sin[θ] > 0]

Print["The simplified integrand ", syma[θ, m, l],
 " is integrated over the interval [0,Pi]."]

$Assumptions = i ∈ Integers && i ≥ 0;

Print[" The diagonal entries (i=0,1,2,...) :"]
Integrate[syma[θ, i, i], {θ, 0, Pi}]

Print["The first super-diagonal:"]
Integrate[syma[θ, i, i + 1], {θ, 0, Pi}]

Print["The second super-diagonal:"]
Integrate[syma[θ, i, i + 2], {θ, 0, Pi}]

Print["The third super-diagonal:"]
Integrate[syma[θ, i, i + 3], {θ, 0, Pi}]

Print["The fourth super-diagonal:"]
Integrate[syma[θ, i, i + 4], {θ, 0, Pi}]

Print[" *** "]

Print["The first sub-diagonal:"]
Integrate[Refine[syma[θ, i, i - 1], i ≥ 2], {θ, 0, Pi}]

Print["The second sub-diagonal:"]
Integrate[Refine[syma[θ, i, i - 2], i ≥ 3], {θ, 0, Pi}]

Print["The third sub-diagonal:"]
Integrate[Refine[syma[θ, i, i - 3], i ≥ 4], {θ, 0, Pi}]
```

```
Print["The fourth sub-diagonal:"]
Integrate[Refine[syma[θ, i, i - 4], i ≥ 5], {θ, 0, Pi}]
```

```
The simplified integrand
 (1 + Cos[θ]) (2 Cos[θ] Cos[l θ] - l Sin[θ] Sin[l θ]) (Cos[θ] Cos[m θ] - m Sin[θ] Sin[m θ])
  is integrated over the interval [0,Pi].

 The diagonal entries (i=0,1,2,...) :
```

Out[37]= $\frac{1}{4}\left(2+i^2\right)\pi$

```
The first super-diagonal:
```

Out[39]= $\frac{1}{16}\left(5+i\left(2+i\right)\right)\pi$

```
The second super-diagonal:
```

Out[41]= $-\frac{1}{8}i\left(1+i\right)\pi$

```
The third super-diagonal:
```

Out[43]= $-\frac{1}{16}\left(1+i\right)^2\pi$

```
The fourth super-diagonal:
```

Out[45]= 0

```
 ***
The first sub-diagonal:
```

Out[48]= $\frac{1}{16}\left(5+\left(-2+i\right)i\right)\pi$

```
The second sub-diagonal:
```

Out[50]= $-\frac{1}{8}\left(-1+i\right)i\,\pi$

```
The third sub-diagonal:
```

Out[52]= $-\frac{1}{16}\left(-1+i\right)^2\pi$

```
The fourth sub-diagonal:
```

Out[54]= 0

Verbatim text of the above Mathematica code (for copy and paste):

```
(* Definition of the basis functions *)
phi[l_,x_]:=(1-x^2)*ChebyshevT[l,x]
\[Omega][t_]:=1/Sqrt[1-t^2]

(* The following integrand is that of Matrix Overscript[A, ~] *)
integrand[t_,m_,l_]:= (t+1)D[phi[l,t],{t,1}]*D[phi[m,t]*\[Omega][t],t]

(* A change of variables has to be performed in order to make this method work *)
changed=Evaluate@integrand[t,m,l]/.t->Cos[\[Theta]];

syma[\[Theta]_,m_,l_]:=Evaluate@FullSimplify[Sin[\[Theta]]*changed,Sin[\[Theta]]>0]

Print["The simplified integrand ",syma[\[Theta],m,l]," is integrated over the interval [0,Pi]."]

$Assumptions=i\[Element]Integers&&i>=0;

Print[" The diagonal entries (i=0,1,2,...) :"]
Integrate[syma[\[Theta],i,i],{\[Theta],0,Pi}]

Print["The first super-diagonal:"]
Integrate[syma[\[Theta],i,i+1],{\[Theta],0,Pi}]

Print["The second super-diagonal:"]
Integrate[syma[\[Theta],i,i+2],{\[Theta],0,Pi}]

Print["The third super-diagonal:"]
Integrate[syma[\[Theta],i,i+3],{\[Theta],0,Pi}]

Print["The fourth super-diagonal:"]
Integrate[syma[\[Theta],i,i+4],{\[Theta],0,Pi}]

Print[" *** "]

Print["The first sub-diagonal:"]
Integrate[Refine[syma[\[Theta],i,i-1],i>=2],{\[Theta],0,Pi}]

Print["The second sub-diagonal:"]
Integrate[Refine[syma[\[Theta],i,i-2],i>=3],{\[Theta],0,Pi}]

Print["The third sub-diagonal:"]
Integrate[Refine[syma[\[Theta],i,i-3],i>=4],{\[Theta],0,Pi}]

Print["The fourth sub-diagonal:"]
Integrate[Refine[syma[\[Theta],i,i-4],i>=5],{\[Theta],0,Pi}]
```

Appendix C

Additional Plots

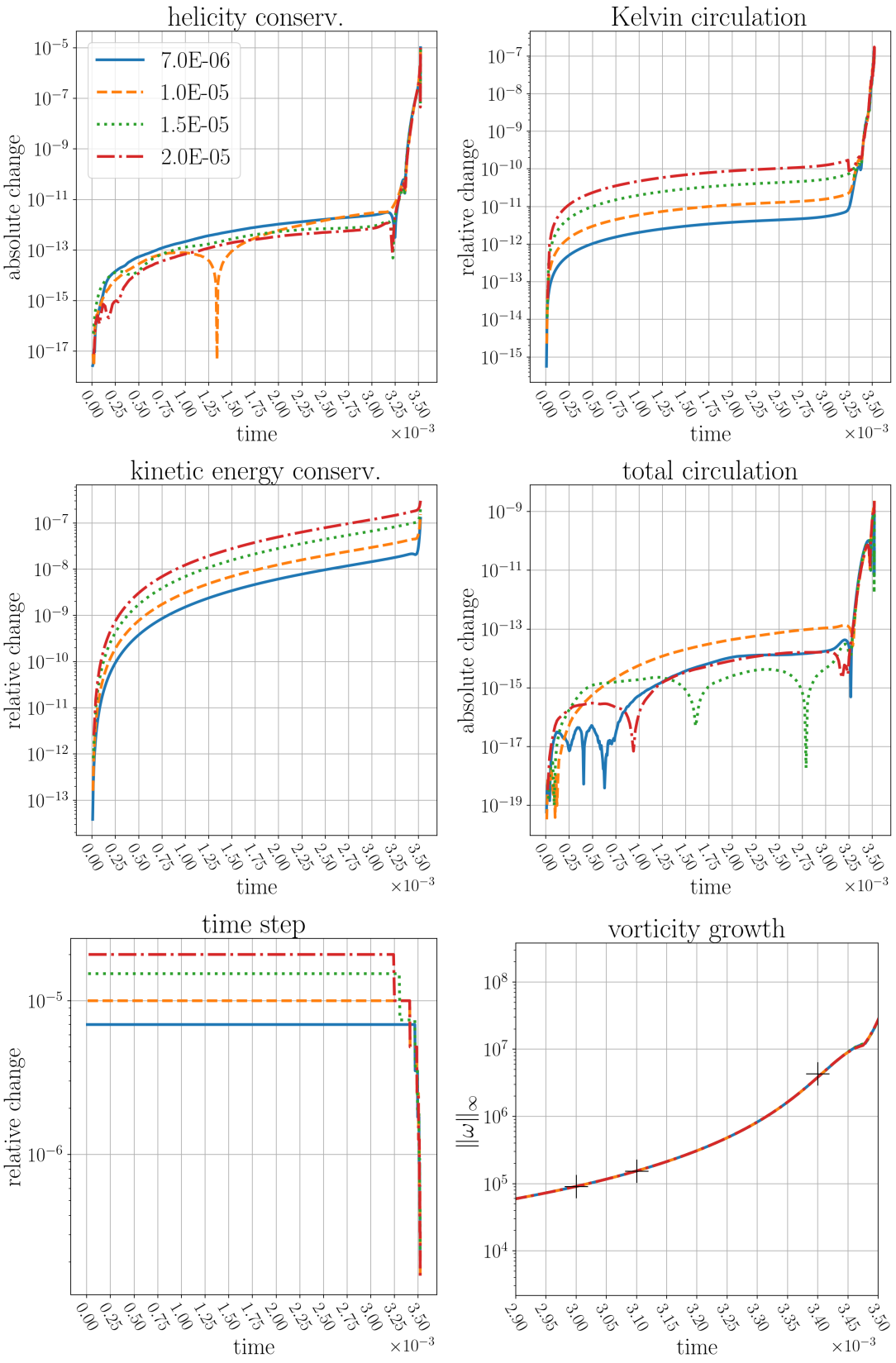


Figure C.1: A fixed time stepping for the flow from [83] is repeated here for a 1025×8192 grid. Breaks appear later but more prominent with respect to the run with a smaller resolution, with first signs of irregular behavior at $t = 0.00325$. This is the time, when the Fourier dimension is exhausted and smaller scales cannot be represented anymore.

References

[1] "754-1985-IEEE Standard for Binary Floating-Point Arithmetic". In: *ANSI/IEEE* (1985).

[2] "754-2008-IEEE Standard for Floating-Point Arithmetic". In: *IEEE Computer Society* (2008).

[3] H. D. I. Abarbanel, D. D. Holm, J. E. Marsden, and T. S. Ratiu. "Nonlinear stability analysis of stratified fluid equilibria". In: *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences* 318.1543 (1986), pp. 349–409.

[4] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[5] C. Bardos and E. S. Titi. "Loss of smoothness and energy conserving rough weak solutions for the 3d Euler equations". In: *arXiv preprint arXiv:0906.2029* (2009).

[6] D. Barkley. "A fluid mechanic's analysis of the teacup singularity". In: *Proceedings of the Royal Society A* 476, 20200348 (2020).

[7] H. Bateman. *Tables of integral transforms*. Ed. by A. Erdélyi. Vol. II. McGraw-Hill Book Company, 1954.

[8] J. R. Bates. "An efficient semi-Lagrangian and alternating direction implicit method for integrating the shallow water equations". In: *Monthly weather review* 112.10 (1984), pp. 2033–2047.

[9] J. T. Beale, T. Kato, and A. Majda. "Remarks on the breakdown of smooth solutions for the 3-D Euler equations". In: *Communications in Mathematical Physics* 94.1 (1984), pp. 61–66.

[10] A. Bennett. *Lagrangian fluid dynamics*. Cambridge University Press, 2006.

[11] N. Besse. "Regularity of the Geodesic Flow of the Incompressible Euler Equations on a Manifold". In: *Communications in Mathematical Physics* 375 (2020), pp. 2155–2189.

[12] N. Besse and U. Frisch. "A constructive approach to regularity of Lagrangian trajectories for incompressible Euler flow in a bounded domain". In: *Communications in Mathematical Physics* 351.2 (2017), pp. 689–707.

[13] N. Besse and E. Sonnendrücker. "Semi-Lagrangian schemes for the Vlasov equation on an unstructured mesh of phase space". In: *Journal of Computational Physics* 191.2 (2003), pp. 341–376.

[14] S. Bochkanov. *ALGLIB (www.alglib.net)*.

[15] F. Bowman. *Introduction to Bessel functions*. Courier Corporation, 2012.

[16] J. C. Bowman and M. Roberts. “Efficient dealiased convolutions without padding”. In: *SIAM Journal on Scientific Computing* 33.1 (2011), pp. 386–406.

[17] J. P. Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[18] M. E. Brachet, D. I. Meiron, S. A. Orszag, B. G. Nickel, R. H. Morf, and U. Frisch. “Small-scale structure of the Taylor–Green vortex”. In: *Journal of Fluid Mechanics* 130 (1983), pp. 411–452.

[19] M. D. Bustamante and R. M. Kerr. “3D Euler about a 2D symmetry plane”. In: *Physica D: Nonlinear Phenomena* 237.14-17 (2008), pp. 1912–1920.

[20] R. E. Caflisch. “Singularity formation for complex solutions of the 3D incompressible Euler equations”. In: *Physica D: Nonlinear Phenomena* 67.1-3 (1993), pp. 1–18.

[21] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral methods: fundamentals in single domains*. Springer Science & Business Media, 2007.

[22] A.-L. Cauchy. “Sur l’état du fluide à une époque quelconque du mouvement”. In: *Mémoires extraits des recueils de l’Académie des sciences de l’Institut de France, Sciences mathématiques et physiques* 1 (1827), pp. 33–73.

[23] J.-Y. Chemin. *Perfect incompressible fluids*. Vol. 14. Oxford University Press, 1998.

[24] J. Chen and T. Y. Hou. “Finite time blowup of 2D Boussinesq and 3D Euler equations with $C^{1,\alpha}$ velocity and boundary”. In: *arXiv preprint arXiv:1910.00173* (2019).

[25] C.-Z. Cheng and G. Knorr. “The integration of the Vlasov equation in configuration space”. In: *Journal of Computational Physics* 22.3 (1976), pp. 330–351.

[26] L. Clarke, I. Glendinning, and R. Hempel. “The MPI message passing interface standard”. In: *Programming environments for massively parallel distributed systems*. Springer, 1994, pp. 213–218.

[27] P. Colella and P. R. Woodward. “The piecewise parabolic method (PPM) for gas-dynamical simulations”. In: *Journal of computational physics* 54.1 (1984), pp. 174–201.

[28] P. Constantin. “The Euler equations and non-local conservative Riccati equations”. In: *arXiv preprint nlin/0004019* (2000).

[29] P. Constantin, C. Fefferman, and A. J. Majda. “Geometric constraints on potentially singular solutions for the 3-D Euler equations”. In: *Communications in Partial Differential Equations* 21.3-4 (1996).

[30] P. Constantin, V. Vicol, and J. Wu. “Analyticity of Lagrangian trajectories for well posed inviscid incompressible fluid models”. In: *Advances in Mathematics* 285 (2015), pp. 352–393.

[31] D. Cordoba and C. Fefferman. “On the collapse of tubes carried by 3D incompressible flows”. In: *arXiv preprint math/0101253* (2001).

[32] R. Courant, K. Friedrichs, and H. Lewy. “Über die partiellen Differenzengleichungen der mathematischen Physik”. In: *Mathematische annalen* 100.1 (1928), pp. 32–74.

[33] M. G. Cox. “The numerical evaluation of B-splines”. In: *IMA Journal of Applied Mathematics* 10.2 (1972), pp. 134–149.

[34] H. S. M. Coxeter. “Introduction to geometry”. In: (1961).

[35] H. B. Curry and I. J. Schoenberg. “On spline distributions and their limits-the polya distribution functions”. In: *Bulletin of the American Mathematical Society*. Vol. 53. 11. amer. mathematical soc. 201 Charles st, providence, ri 02940-2213, 1947, pp. 1114–1114.

[36] L. Dagum and R. Menon. “OpenMP: an industry standard API for shared-memory programming”. In: *Computational Science & Engineering, IEEE* 5.1 (1998), pp. 46–55.

[37] C. De Boor. *A practical guide to splines*. Applied Mathematical Sciences 27. Springer, 2001.

[38] C. De Boor. “On calculating with B-splines”. In: *Journal of Approximation theory* 6.1 (1972), pp. 50–62.

[39] B. Delaunay. “Sur la sphere vide”. In: *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7.793-800 (1934), pp. 1–2.

[40] J. Deng, T. Y. Hou, and X. Yu. “Geometric properties and nonblowup of 3D incompressible Euler flow”. In: *Communications in Partial Difference Equations* 30.1-2 (2005), pp. 225–243.

[41] F. Di Bruno. “Note sur une nouvelle formule de calcul différentiel”. In: *Pure and Applied Mathematics Quarterly* 1 (1857), pp. 359–360.

[42] C. Domb and M. F. Sykes. “On the susceptibility of a ferromagnetic above the Curie point”. In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 240.1221 (1957), pp. 214–228.

[43] W. E and C.-W. Shu. “Small-scale structures in Boussinesq convection”. In: *Physics of Fluids* 6.1 (1994), pp. 49–58.

[44] T. M. Elgindi and I.-J. Jeong. “Finite-time singularity formation for strong solutions to the axi-symmetric 3D Euler equations”. In: *Annals of PDE* 5.16 (2019).

[45] A. B. Ferrari. “On the blow-up of solutions of the 3-D Euler equations in a bounded domain”. In: *Communications in Mathematical Physics* 155.2 (1993), pp. 277–294.

[46] B. Fornberg. “Generation of finite difference formulas on arbitrarily spaced grids”. In: *Mathematics of computation* 51.184 (1988), pp. 699–706.

[47] S. Fortune. “Voronoi diagrams and Delaunay triangulations”. In: *Computing in Euclidean geometry*. World Scientific, 1995, pp. 225–265.

[48] M. Frigo and S. G. Johnson. “FFTW3, 2006”. In: *Available from WWW: http://www.fftw. org/# documentation* (2007).

[49] M. Frigo and S. G. Johnson. “The design and implementation of FFTW3”. In: *Proceedings of the IEEE* 93.2 (2005), pp. 216–231.

[50] U. Frisch. *Turbulence: the legacy of AN Kolmogorov*. Cambridge university press, 1995.

[51] U. Frisch and B. Villone. “Cauchy’s almost forgotten Lagrangian formulation of the Euler equation for 3D incompressible flow”. In: *The European Physical Journal H* 39.3 (2014), pp. 325–351.

[52] U. Frisch and V. Zheligovsky. “A very smooth ride in a rough sea”. In: *Communications in Mathematical Physics* 326.2 (2014), pp. 499–505.

[53] R. R. Gagné and M. M. Shoucri. “A splitting scheme for the numerical solution of a one-dimensional Vlasov equation”. In: *Journal of Computational Physics* 24.4 (1977), pp. 445–449.

[54] P. Gamblin. “Système d’Euler incompressible et régularité microlocale analytique”. In: *Annales de l’Institut Fourier*. Vol. 44. 5. 1994, pp. 1449–1475.

[55] W. Gautschi. *Orthogonal polynomials*. Oxford University Press, 2004.

[56] J. D. Gibbon. “The three-dimensional Euler equations: Where do we stand?” In: *Physica D: Nonlinear Phenomena* 237.14-17 (2008), pp. 1894–1904.

[57] A. Gil, J. Segura, and N. M. Temme. *Numerical methods for special functions*. SIAM, 2007.

[58] V. Girault and P.-A. Raviart. *Finite element methods for Navier-Stokes equations: theory and algorithms*. Vol. 5. Springer Science & Business Media, 2012.

[59] O. Glass, F. Sueur, and T. Takahashi. “Smoothness of the motion of a rigid body immersed in an incompressible perfect fluid”. In: *Annales Scientifiques de l’École Normale Supérieure* 45.1 (2012), pp. 1–51.

[60] R. Grauer and T. C. Sideris. “Numerical computation of 3D incompressible ideal fluids with swirl”. In: *Physical Review Letters* 67.25 (1991), pp. 3511–3514.

[61] N. M. Günther. “Über ein Hauptproblem der Hydrodynamik”. In: *Mathematische Zeitschrift* 24.1 (1926), pp. 448–499.

[62] H. Gzyl. “Multidimensional extension of Faa di Bruno’s formula”. In: *Journal of Mathematical Analysis and Applications* 116.2 (1986), pp. 450–455.

[63] W. Heinrichs. “Spectral methods with sparse matrices”. In: *Numerische Mathematik* 56.1 (1989), pp. 25–41.

[64] T. Hertel. “On the time-analytic behavior of particle trajectories in an ideal and incompressible fluid flow”. MA thesis. Leipzig, 2016. URL: https://nbn-resolving.org/urn:nbn:de:bsz:15-qucosa2-170523.

[65] T. Y. Hou. “Blow-up or no blow-up? A unified computational and analytic approach to 3D incompressible Euler and Navier–Stokes equations”. In: *Acta Numerica* 18 (2009), pp. 277–346.

[66] T. Y. Hou and R. Li. “Blowup or no blowup? The interplay between theory and numerics”. In: *Physica D: Nonlinear Phenomena* 237.14-17 (2008), pp. 1937–1944.

[67] T. Y. Hou and R. Li. “Dynamic depletion of vortex stretching and non-blowup of the 3-D incompressible Euler equations”. In: *Journal of Nonlinear Science* 16.6 (2006), pp. 639–664.

[68] A. Jeffrey and D. Zwillinger. *Table of integrals, series, and products*. Elsevier, 2007.

[69] F. Johansson. “Arb: efficient arbitrary-precision midpoint-radius interval arithmetic”. In: *IEEE Transactions on Computers* 66 (8 2017), pp. 1281–1292.

[70] T. Kato. “On the Smoothness of Trajectories in Incompressible Perfect”. In: *Nonlinear Wave Equations: A Conference in Honor of Walter A. Strauss on the Occasion of His Sixtieth Birthday, May 2-3, 1998, Brown University*. Vol. 263. American Mathematical Soc. 2000, p. 109.

[71] R. M. Kerr. “Evidence for a singularity of the three-dimensional, incompressible Euler equations”. In: *Physics of Fluids A: Fluid Dynamics* 5.7 (1993), pp. 1725–1746.

[72] A. C. King, J. Billingham, and S. R. Otto. *Differential equations: linear, nonlinear, ordinary, partial*. Cambridge University Press, 2003.

[73] B. G. Korenev. *Bessel functions and their applications*. CRC Press, 2002.

[74] Y. Kuroda. “Symmetries and Casimir invariants for perfect fluid”. In: *Fluid dynamics research* 5.4 (1990), p. 273.

[75] S. H. Lamb. *Hydrodynamics*. Ed. by R. Caflisch. 6th ed. Cambridge Mathematical Library. Cambridge University Press, 1975.

[76] P. Lancaster and K. Salkauskas. *Curve and surface fitting. An introduction*. 1986.

[77] A. Larios, M. R. Petersen, E. S. Titi, and B. Wingate. “A computational investigation of the finite-time blow-up of the 3D incompressible Euler equations based on the Voigt regularization”. In: *Theoretical and Computational Fluid Dynamics* 32.1 (2018), pp. 23–34.

[78] A. Larios and E. S. Titi. “A Blow-Up Criterion for the 3D Euler Equations Via the Euler-Voigt Inviscid Regularization”. In: *arXiv preprint arXiv:1507.08203* (2015).

[79] A. Larios and E. S. Titi. “Global regularity versus finite-time singularities: some paradigms on the effect of boundary conditions and certain perturbations”. In: *Recent Progress in the Theory of the Euler and Navier-Stokes Equations* 430 (2016), pp. 96–125.

[80] A. Larios and E. S. Titi. “On the higher-order global regularity of the inviscid Voigt-regularization of three-dimensional hydrodynamic models”. In: *arXiv preprint arXiv:0910.3354* (2009).

[81] N. Leprovost, B. Dubrulle, and P.-H. Chavanis. “Dynamics and thermodynamics of axisymmetric flows: Theory”. In: *Physical Review E* 73.4, 046308 (2006).

[82] L. Lichtenstein. “Über einige Existenzprobleme der Hydrodynamik homogener, unzusammendrückbarer, reibungsloser Flüssigkeiten und die Helmholtzschen Wirbelsätze”. In: *Mathematische Zeitschrift* 23.1 (1925), pp. 89–154.

[83] G. Luo and T. Y. Hou. “Toward the Finite-Time Blowup of the 3D Axisymmetric Euler Equations: A Numerical Investigation”. In: *Multiscale Modeling & Simulation* 12.4 (Jan. 2014), pp. 1722–1776.

[84] A. J. Majda and A. L. Bertozzi. *Vorticity and Incompressible Flow*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2001.

[85] J. C. Mason and D. C. Handscomb. *Chebyshev polynomials*. CRC press, 2002.

[86] A. McDonald. “A semi-Lagrangian and semi-implicit two time-level integration scheme”. In: *Monthly Weather Review* 114.5 (1986), pp. 824–830.

[87] B. Mercier. *Analyse numérique des méthodes spectrales*. Tech. rep. 1981.

[88] C. A. Micchelli. “Interpolation of scattered data: distance matrices and conditionally positive definite functions”. In: *Constructive approximation* 2.1 (1986), pp. 11–22.

[89] R. H. Morf, S. A. Orszag, and U. Frisch. “Spontaneous singularity in three-dimensional inviscid, incompressible flow”. In: *Physical Review Letters* 44.9 (1980), p. 572.

[90] F. Oberhettinger. *Tables of Bessel Transforms*. Springer Berlin Heidelberg, 1972.

[91] A. Okabe, B. Boots, and K. Sugihara. “Spatial tessellations”. In: *Geographical information systems* 1 (1999), pp. 503–526.

[92] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark. *NIST handbook of mathematical functions hardback and CD-ROM*. Cambridge University Press, 2010.

[93] W. Pauls and U. Frisch. “A Borel transform method for locating singularities of Taylor and Fourier series”. In: *Journal of Statistical Physics* 127.6 (2007), pp. 1095–1119.

[94] L. Piegl and W. Tiller. *The NURBS book*. Springer Science & Business Media, 2012.

[95] O. Podvigina, V. Zheligovsky, and U. Frisch. "The Cauchy–Lagrangian method for numerical analysis of Euler flow". In: *Journal of Computational Physics* 306 (2016), pp. 320–342.

[96] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Texts and Monographs in Computer Science. Springer, 1985.

[97] W. H. Press, S. A. Teukolsky, B. P. Flannery, and W. T. Vetterling. *Numerical recipes in Fortran 77: volume 1, volume 1 of Fortran numerical recipes: the art of scientific computing*. Cambridge university press, 1992.

[98] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes: the art of scientific computing*. 3rd ed. Cambridge University Press, 2007.

[99] A. Pumir and E. D. Siggia. "Development of singular solutions to the axisymmetric Euler equations". In: *Physics of Fluids A: Fluid Dynamics* 4.7 (1992), pp. 1472–1491.

[100] R. J. Purser and L. M. Leslie. "An efficient interpolation procedure for high-order three-dimensional semi-Lagrangian models". In: *Monthly Weather Review* 119.10 (1991), pp. 2492–2498.

[101] A. Robert. "A stable numerical integration scheme for the primitive meteorological equations". In: *Atmosphere-Ocean* 19.1 (1981), pp. 35–46.

[102] X. Saint Raymond. "Remarks on axisymmetric solutions of the incompressible Euler system". In: *Communications in Partial Differential Equations* 19.1-2 (1994), pp. 321–334.

[103] I. J. Schoenberg. "Contributions to the problem of approximation of equidistant data by analytic functions". In: *Quarterly of Applied Mathematics* 4.1 (1946), pp. 45–99.

[104] P. Serfati. "Équation d'Euler et holomorphies à faible régularité spatiale". In: *Comptes Rendus de l'Academie des Sciences-Serie I-Mathematique* 320.2 (1995), pp. 175–180.

[105] P. Serfati. "Étude mathématique de flammes infiniment minces en combustion. Résultats de structure et de régularité pour l'équation d'Euler incompressible". PhD thesis. Thèse de Doctorat de l'Université Paris 6, 1992.

[106] P. Serfati. "Structures holomorphes à faible régularité spatiale en mécanique des fluides". In: *Journal de Mathématiques Pures et Appliqués* 74 (Jan. 1995), pp. 95–104.

[107] J. Shen. "Efficient spectral-Galerkin method II. Direct solvers of second-and fourth-order equations using Chebyshev polynomials". In: *SIAM Journal on Scientific Computing* 16.1 (1995), pp. 74–87.

[108] J. Shen. "Efficient spectral-Galerkin methods III: Polar and cylindrical geometries". In: *SIAM Journal on Scientific Computing* 18.6 (1997), pp. 1583–1604.

[109] J. Shen, T. Tang, and L.-L. Wang. *Spectral methods: algorithms, analysis and applications*. Vol. 41. Springer Science & Business Media, 2011.

[110] T. Shirota and T. Yanagisawa. "Note on global existence for axially symmetric solutions of the Euler system". In: *Proceedings of the Japan Academy, Ser. A, Mathematical Sciences* 70.10 (1994), pp. 299–304.

[111] A. Shnirelman. "On the analyticity of particle trajectories in the ideal incompressible fluid". In: (). arXiv: 1205.5837v1 [math.AP].

[112] M. Siegel and R. E. Caflisch. “Calculation of complex singular solutions to the 3D incompressible Euler equations”. In: *Physica D: Nonlinear Phenomena* 238.23-24 (2009), pp. 2368–2379.

[113] E. Sonnendrücker, J. Roche, P. Bertrand, and A. Ghizzo. “The semi-Lagrangian method for the numerical resolution of the Vlasov equation”. In: *Journal of computational physics* 149.2 (1999), pp. 201–220.

[114] A. Staniforth and J. Côté. “Semi-Lagrangian integration schemes for atmospheric models—A review”. In: *Monthly weather review* 119.9 (1991), pp. 2206–2223.

[115] B. O. Turesson. *Nonlinear potential theory and weighted Sobolev spaces*. Vol. 1736. Springer Science & Business Media, 2000.

[116] M. R. Ukhovskii and V. I. Yudovich. “Axially symmetric flows of ideal and viscous fluids filling the whole space”. In: *Journal of Applied Mathematics and Mechanics* 32.1 (1968), pp. 52–62.

[117] J. S. Vandergraft. *Introduction to Numerical Computations*. 2nd. Computer Science & Applied Mathematics Monograph. Academic Press Inc, 1983.

[118] H. Wendland. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004.

[119] V. I. Yudovich. “On the loss of smoothness of the solutions of the Euler equations and the inherent instability of flows of an ideal fluid”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 10.3 (2000), pp. 705–719.