Towards parallel-in-time simulations of turbulent Rayleigh–Bénard convection

Gijs Kooij

Promotie Commissie:

Prof. dr. P.M.G. Apers (voorzitter)	Universiteit Twente
Prof. dr. ir. B.J. Geurts (promotor)	Universiteit Twente
Dr. M.A. Botchev (co-promotor)	Keldysh Institute of Applied Mathematics
Prof. dr. D. Lohse	Universiteit Twente
Dr. ir. R.J.A.M. Stevens	Universiteit Twente
Prof. dr. H.J.H. Clercx	TU Eindhoven
Prof. dr. ir. B. Koren	TU Eindhoven
Prof. dr. A.E.P. Veldman	Rijksuniversiteit Groningen
Dr. D. Ruprecht	University of Leeds

The work in this thesis was carried out at the Multiscale Modeling and Simulation group of the Faculty of Electrical Engineering, Mathematics and Computer Science of the University of Twente. It is part of the research programme of the Foundation for Fundamental Research on Matter (FOM), which is financially supported by the Netherlands Organisation for Scientific Research (NWO).

Nederlandse titel:

Richting parallel-in-tijd-simulaties van turbulente Rayleigh–Bénardconvector

Publisher:

Gijs L. Kooij, Multiscale Modeling and Simulation, University of Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands

Copyright © 2017 Gijs Kooij

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means without the prior written permission of the publisher.

ISBN: 978-90-365-4403-0

DOI: 10.3990/1.9789036544030

TOWARDS PARALLEL-IN-TIME SIMULATIONS OF TURBULENT RAYLEIGH–BÉNARD CONVECTION

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. T.T.M. Palstra,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 10 november 2017 om 14.45 uur

door

Gijs Levi Kooij
geboren op 14 februari 1989
te Tilburg, Nederland

Dit proefschrift is goedgekeurd door de promotor en de co-promotor:

Prof. dr. ir. B.J. Geurts

Dr. M.A. Botchev

Acknowledgements

This thesis concludes four years of my scientific endeavors as a PhD student at the University of Twente. My work would not have been possible without a number of people I am very grateful to.

First of all, I want to thank my supervisors Bernard Geurts and Mike Botchev. It has been a pleasure working with them closely over the years. I thank them for their guidance, their constructive feedback that kept me sharp, and their meticulous proofreading of anything I wrote. Any errors in this thesis that may have slipped through the net should only be blamed on me.

I want to thank the large number of people that contributed to the third chapter of my thesis specifically. I would like to mention Susanne Horn, Detlef Lohse, Erwin van der Poel, Olga Shishkina, Richard Stevens, and Roberto Verzicco. Without their contributions that chapter could not have been written. I am grateful that we have been able to join forces.

I want to thank Arthur Veldman for our interesting discussions on many Thursday afternoons. I thank him for teaching me the virtue of “skew-symmetry”, and sharing plenty of anecdotes about computational fluid dynamics in the good old days.

I thank my direct colleagues in my group: Paolo (with whom I shared the office for four years), Edo, and Milos. We shared a sense of humor and we had many interests in common. We had many in-depth discussions about work and everything besides work, and we also met frequently outside office hours.

I also thank the secretaries of our department Marielle and Linda for all their help with the usual paperwork and administration.

My time in Twente is not only memorable from a purely academic point of view, but also because of the kind and friendly persons I had the pleasure to meet over the years. Many of them were colleagues who I soon got to call friends. I enjoyed the lively conversations over coffee or at the lunch table. I also enjoyed the many times we participated in a pubquiz, usually as the “Mathemagicians”, and our occasional victories. I would like to thank Bettina, Bijoy, Daniela, Felix (who invited me to his lovely home in the Austrian Alps), Gjerrit, Ivana, Koen, Laura, Matthias, Mihaela, Nastya, Nishant, Sjoerd, Wilbert, and many others, for the great time I had with them.

Last but certainly not least, I want to thank my parents, and my brother and sister, for their enduring love and continuous support throughout my PhD studies and my life. Without them I could not have done it.

Gijs Kooij
Enschede, November 2017

Contents

1	Introduction	1
2	Direct numerical simulation of Nusselt number scaling in rotating Rayleigh–Bénard convection	7
2.1	Introduction	7
2.2	Boussinesq approximation	9
2.2.1	Rotating coordinate system	9
2.2.2	Boussinesq approximation	10
2.2.3	Dimensionless formulation	11
2.2.4	Global heat transfer	13
2.3	Spectral element method	13
2.4	Heat transfer scaling and flow structure	15
2.4.1	Scaling of the heat transfer under rotation	16
2.4.2	Change in flow structure	19
2.5	Conclusions	20
3	Comparison of computational codes for direct numerical simulations of turbulent Rayleigh–Bénard convection	23
3.1	Introduction	24
3.2	Governing equations and evaluation of the Nusselt number	25
3.3	Numerical methods	27
3.3.1	AFID/RBFlow	27
3.3.2	NEK5000	27
3.3.3	GOLDFISH	28
3.3.4	OPENFOAM	28
3.4	Performance comparison	29

3.4.1	Convergence test at $Ra = 10^8$	29
3.4.2	Robustness against under-resolution	33
3.5	Conclusions and outlook	34
4	A block Krylov subspace implementation of the time-parallel Pararexp method and its extension for nonlinear partial differential equations	39
4.1	Introduction	39
4.2	Exponential time-integration	41
4.2.1	Exponential block Krylov method	41
4.2.2	Parallelization of linear problems	44
4.2.3	Treatment of nonlinearities	46
4.2.4	Parallelization of nonlinear problems	48
4.3	Advection-diffusion equation	52
4.3.1	Homogeneous PDE	52
4.3.2	Parallel efficiency	56
4.4	Burgers equation	58
4.4.1	Travelling wave	58
4.4.2	Parallel efficiency	61
4.4.3	Multiscale example	62
4.5	Conclusions	64
5	An exponential time integrator for the incompressible Navier–Stokes equation	67
5.1	Introduction	67
5.2	Exponential time integration	69
5.2.1	Exponential block Krylov method	69
5.2.2	Incompressible Navier–Stokes equation	71
5.3	Incompressible flow simulations	74
5.3.1	Taylor–Green vortex	75
5.3.2	Lid-driven cavity flow	76
5.4	Parallelization in time	82
5.4.1	Parallel algorithm	83
5.4.2	Speedup and efficiency	84
5.5	Conclusions	86
6	Conclusions	89
	References	93
	Appendix A	109
	Summary	113
	Samenvatting	117

About the author	121
List of publications	123

Many flows are driven by buoyant forces that are caused by differences in temperature. Relatively hot fluid tends to rise and comparably cool fluid tends to sink. This flow mechanism is known as natural (or free) convection. Various examples of natural convection are found in engineering: heat exchangers, cooling of electronic circuits, industrial furnaces, ventilation of buildings, etc. Natural convection also plays an important role in geophysics, concerning the dynamics of the oceans, the atmosphere, and the core of planets and stars. The fundamentals of natural convection are described in [61, 86, 90].

The driving mechanism of natural convection is often studied in the classical context of Rayleigh–Bénard (RB) convection. In this setting, a layer of fluid is heated from below and cooled from above. If the temperature difference is very small, the fluid remains in a mechanical equilibrium and heat is only transported by diffusion. If the adverse temperature gradient becomes sufficiently large, the equilibrium turns unstable and the fluid is set in motion. Heat is then transported by both diffusion and convection. The existence of RB convection was first demonstrated in experiments by Bénard [7]. Lord Rayleigh subsequently studied the critical point which indicates the onset of convection [141].

In principle, the dynamics of RB convection is governed by the compressible Navier–Stokes equations. Under appropriate conditions simplifications can be made. To start with, the fluid is supposed to be Newtonian, and its material properties (kinematic viscosity, thermal diffusivity) are assumed to be constant. The variations in pressure, density, and temperature are all assumed to be small: the flow is effectively incompressible. The small variations in density are only felt in the gravitational force. The resulting buoyancy is then what drives RB convection. Under these assumptions, one can derive a system of equations that governs RB convection [107]. These are known as the Oberbeck–Boussinesq equa-

tions [18, 137],

$$\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} = \nu \Delta \mathbf{u} - \nabla p + g\beta\theta \mathbf{e}_z, \quad (1.1)$$

$$\partial_t \theta + \mathbf{u} \cdot \nabla \theta = \kappa \Delta \theta, \quad (1.2)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (1.3)$$

where $\mathbf{u} = (u, v, w)^T$ is the velocity, ν the kinematic viscosity, p the kinematic pressure, g the gravitational acceleration, β the thermal expansion coefficient, θ the difference in (actual) temperature with respect to a reference temperature, $\mathbf{e}_z = (0, 0, 1)^T$ the unit vector in the z -direction, κ the thermal diffusivity. Gravity is assumed to act in the negative z -direction.

The dynamics of RB convection are determined by two dimensionless parameters. These are the Rayleigh number, $Ra = \beta g \Delta H^3 / (\kappa \nu)$, and the Prandtl number, $Pr = \nu / \kappa$. Here, H denotes the height of the fluid layer, and Δ the temperature difference between the hot bottom plate and the cold top plate. RB convection sets in when a critical Rayleigh number is reached, $Ra_c = 1708$, see [25] for a detailed stability analysis. When the Rayleigh number is further increased, the flow develops more and more complex structure and eventually turns into turbulence.

The heat transfer between the bottom and the top plate is an important output parameter in RB convection. The heat transfer is measured by the Nusselt number (Nu), which is the ratio of the total heat transfer to the conductive heat transfer. The scaling of Nu with Ra in particular is of great interest, because it allows us to predict Nu at very high Ra . In geophysics for example, Ra is usually much higher than what can be achieved in a laboratory [27]. The scaling of Nu with Ra is traditionally assumed to be of a power law type. Results from different experiments however lead us to believe that a universal power law does not exist [120, 156, 157]. A more general approach is the Grossmann–Lohse theory [68], which accounts for the existence of different regimes in the Ra - Pr parameter space. This model shows excellent agreement with experimental data [164].

Recently, there is much interest in RB convection at very high Rayleigh numbers ($Ra \gtrsim 10^{14}$), also known as the “ultimate” regime predicted by Kraichnan [99]. In this regime, the boundary layers have become fully turbulent. The transition to turbulent boundary layers is associated with increased mixing and enhanced heat transfer [69]. The physics of this regime is not fully understood yet. So far, experiments have produced contradicting results on the scaling of Nu [27, 115, 158].

Direct numerical simulation

In direct numerical simulations (DNS) the governing equations are solved numerically. With the rise of modern supercomputers, DNS has become feasible for more and more practical purposes. Turbulent flow features a broad spectrum of length scales, ranging down to the Kolmogorov length scale in the velocity field and the Batchelor scale in the temperature field [140]. In a DNS, all scales are resolved accurately, without recouring to a turbulence model of some kind. As the Rayleigh

number increases, the length scales become smaller, and a higher spatial resolution is necessary to capture the motions at the smallest scales [154, 165]. Additionally, the time step size is often limited by stability and accuracy restrictions. In that case, the time step size decreases proportionally to the spatial mesh width as well. The increasing computational demand with Rayleigh number limits the practical use of DNS for RB convection to moderate Rayleigh numbers. The highest Rayleigh number simulated with DNS so far is $Ra = 2 \cdot 10^{12}$ (see [163]), which is well below the ultimate regime. Because of the large computational resources needed for DNS, it is important to investigate the accuracy and computational efficiency of existing numerical methods.

In geophysics, natural convection is often accompanied by rotation. It is known that rotation can have a strong effect on the flow structure and heat transfer in Rayleigh–Bénard convection [103]. We present a DNS study of Rayleigh–Bénard convection in a rotating cylinder in Chapter 2. The simulations are performed with the spectral element method from the opensource code Nek5000 [48]. The goal of this study is essentially twofold. First, we test the accuracy and the suitability of the spectral element method for DNS of Rayleigh–Bénard convection. Second, we want to examine the effect of rotation on the flow in a wide range of Rayleigh numbers.

Over the years, many numerical codes have been developed that are capable of simulating RB convection. In Chapter 3, we compare four of these codes based on different discretization techniques: (1) a second-order finite difference method (AFID/RBflow [163, 165, 174, 179, 180]); (2) a fourth-order finite volume method (GOLDFISH [153, 155, 152]); (3) an eighth-order spectral element method (Nek5000 [24, 48, 75, 97, 148]); (4) a second-order finite volume method (OpenFOAM [183]). The first two codes are dedicated to RB convection in simple geometries, whereas the latter two codes are designed for general problems also in complex geometries. We perform simulations in three different geometries: a double-periodic domain, a cubic container, and a cylindrical container. The comparison is mainly focussed on the heat transfer, measured by the Nusselt number. A classical convergence test is presented for $Ra = 10^8$ and $Pr = 1$, using well-resolved simulations as a reference. The accuracy of the methods is assessed in relation to the corresponding computational cost addressing the question—how much does it cost to achieve a certain level of accuracy?

Parallelism in time

With an ever-increasing number of computer cores, additional parallelization techniques beyond subdivision of the computational work in physical space are becoming of interest. In this thesis, we focus on the development of parallel-in-time methods for the incompressible Navier–Stokes equation. Parallel time integration is certainly not a new concept. The first method was already introduced in 1964 [135]. An excellent historic overview of numerous contributions to this field is given in [53]. Earlier overviews are found in [20, 60]. One important characteris-

tic is that time-parallel methods typically perform some redundant computations with respect to a corresponding sequential algorithm. Introducing some degree of redundancy might, however, allow some parts of the algorithm to be executed in parallel. The goal is not to reduce the total amount of computational work, but to reduce the actual (wall-clock) time of the computation.

Different families of time-parallel methods can be distinguished.

- First, some are based on shooting type methods [6, 26, 135, 147]. The shooting method was originally used to solve boundary value problems. These methods led eventually to the development of the Parareal algorithm [112].
- Second, there are domain decomposition methods in space-time. These are based on much older ideas of solving ODEs iteratively by Picard [139] and Lindelöf [110]. This class includes the waveform relaxation method [108], which was developed for the simulation of electronic circuits. Variations of the Schwarz method are suitable for the application to partial differential equations [51, 56].
- Third, we mention multigrid methods in space-time [71, 82, 117, 129]. These also include the PFASST algorithm [46], which is based on the full approximation scheme using a spectral deferred correction method for the integration in time.
- Fourth, some time integration methods allow a form of direct parallelization, without introducing iterations. For example, multiple stages of a Runge–Kutta method can be evaluated in parallel [173, 32]. Also, time can be treated as another dimension, such that all time steps are calculated at once [4, 34, 119, 186, 189]. This can be realized also with space-time elements in (discontinuous) Galerkin methods [22, 83, 175]. Other forms of parallelism in time involve the approximation of exact solvers in implicit methods [16], operator splitting [12], cyclic reduction [187], a Laplace transform [150], or the revisionist integral deferred correction method [31].
- A promising method, which will be a guideline in this thesis, is the Paraexp method [70], which is essentially based on an overlapping time-decomposition in which the original problem is split into homogeneous and nonhomogeneous parts. Parallel speedup is expected from the fact that homogeneous problems can be integrated much faster than nonhomogeneous problems. The integration of the homogeneous problem is realized by using, for example, Krylov subspace methods to evaluate the matrix exponential function in a very efficient way.

Although time-parallel methods have been around for decades, they never became common practice. The introduction of Parareal [112] sparked new interest within the scientific community. A detailed convergence analysis of Parareal is given by [58]. Parareal typically provides decent parallel speedup for parabolic

problems with a dissipative nature. A simple example is the heat equation [159]. Less than satisfactory results are obtained for hyperbolic problems for which Parareal shows poor convergence behaviour [52, 160]. Although, the performance in some cases can be improved by Krylov subspace enhancement [47, 57, 144]. In fluid flow problems, it was demonstrated that the performance of Parareal deteriorates in cases with low viscosity [49, 100, 161]. Direct numerical simulation at high Reynolds number then becomes challenging because convection in the turbulent flow becomes an important term and the high Reynolds number leads to small dissipation. One reason why Parareal works for dissipative problems, is that errors in the initial condition are damped. It is generally known that turbulent flows display chaotic(-deterministic) behavior in the sense that they are extremely sensitive to their initial condition. This leads us to believe that Parareal is not the most suitable algorithm for DNS at high Rayleigh numbers.

An interesting alternative is the Paraexp algorithm, which works well for parabolic and hyperbolic problems [70]. One major issue, however, is that Paraexp is developed for linear (nonhomogeneous) initial value problems, and the application to the incompressible Navier–Stokes equation is not immediately obvious. Assuming a suitable discretization in the spatial dimensions has been chosen, the resulting semi-discrete system is not only nonlinear; it is also a coupled differential-algebraic equation due to the incompressibility constraint. In fact, the continuity equation imposes an algebraic constraint on the velocity field requiring it to be divergence-free (on a discrete level).

In Chapter 4, we introduce an extension of Paraexp to nonlinear partial differential equations. In our approach the nonlinear term is linearized. The nonlinear remainder is evaluated with the solution at the previous iteration and treated as a source term. This approach is very similar to waveform relaxation [108]. Our method relies on the exponential block Krylov (EBK) method [11] which we use to solve both the homogeneous and nonhomogeneous subproblems in the Paraexp framework. The EBK method is a highly efficient exponential integrator that is based on a singular value decomposition (SVD) of the source term, e.g., arising from the linearized convection term at the previous iteration level. Because there is little parallel communication required in Paraexp, the parallel speedup is estimated by timing the parallel processes separately on a serial computer. We present results for the advection-diffusion equation and for the (viscous) Burgers’ equation in various test cases.

Exponential time integrators are an essential feature of Paraexp. In Chapter 5, we discuss exponential time integration for the incompressible Navier–Stokes equation. In our approach, we reformulate the governing equations without the pressure variable by applying a divergence-free projection onto the Navier–Stokes equation. The reformulated equation is an ordinary differential equation that can be integrated with a Krylov subspace method for matrix exponential functions [131, 146]. We adopted the EBK method, like in Chapter 4. The time integration method was tested for several cases including the Taylor–Green vortex and a lid-driven cavity flow. The exponential time integrator fits in the Paraexp framework and

allows parallel-in-time simulations of the incompressible Navier–Stokes equation. We also provide a simple model to give an estimate of the potential speedup and indicate conditions under which the Paraexp approach in combination with EBK can be competitive with traditional time integration methods.

Thesis outline

The outline of this thesis is as follows. Chapter 2 discusses the simulation of rotating Rayleigh–Bénard convection. In Chapter 3 we present a comparison study between several numerical methods for DNS of Rayleigh–Bénard convection. The remaining chapters focus on parallel-in-time methods. We introduce a parallel-in-time method for nonlinear problems in the Chapter 4. We discuss how this method can be applied to the incompressible Navier–Stokes equation in Chapter 5. Concluding remarks are collected in Chapter 6.

Direct numerical simulation of Nusselt number scaling in rotating Rayleigh–Bénard convection*

We report results from Direct Numerical Simulation (DNS) of rotating Rayleigh–Bénard convection, regarding the scaling of heat transfer with the Rayleigh number for rotating systems at a fixed rate of rotation. The Prandtl number, $Pr = 6.4$, is kept constant. We perform simulations, using a spectral element method, for Rayleigh numbers Ra from 10^6 to 10^9 , and Rossby numbers Ro from 0.09 to ∞ . We find that the Nusselt number Nu scales approximately with a power $2/7$ of Ra at sufficiently high Ra for all Ro . The value of Ra beyond which this Nusselt scaling is well established increases with decreasing Ro . Depending on the rotation rate, the Nusselt number can increase up to 18% with respect to the non-rotating case.

2.1 Introduction

Convective heat transfer plays a major role in a wide range of physical phenomena and engineering applications. Rayleigh–Bénard convection is a classic example of convective heat transfer, stimulated by its accessibility to numerical and experimental analysis. In this particular problem, a layer of fluid is heated from below and cooled from above. The thermal expansion of the fluid creates a buoyant force that leads to the convection of heat. We use Direct Numerical Simulation (DNS) to investigate the dependence of the heat transfer efficiency in case the system is in a state of steady rotation.

For the *non-rotating* case the heat transfer, as characterised by the Nusselt number Nu , is predicted to scale with the Rayleigh number Ra as Ra^β in the

*Based on: G.L. Kooij, M.A. Botchev, and B.J. Geurts. *Direct numerical simulation of Nusselt number scaling in rotating Rayleigh–Bénard convection*. Int. J. Heat Fluid Flow **55**, 26–33, 2015.

limit of sufficiently high Ra [68]. In this chapter we present results of an extensive parameter study indicating that asymptotically at high Ra , this scaling of the Nusselt number also accurately describes the *rotating* case. Rotation is shown to introduce considerable variation in the flow structuring [104]. Nevertheless, the simulation results indicate that the scaling exponent β is quite independent of the rotation rate. The main effect of rotation appears through its influence on the value of Ra beyond which the Nusselt number scaling is well expressed.

Rotating Rayleigh–Bénard convection serves as a primary model for understanding the mechanisms of geo- and astrophysical flows. For example, convection inside the core of stars and planets, like the Earth, is believed to generate magnetic fields by a dynamo action [95]. Another example of convection is found in the Earth’s atmosphere [73], and in the core of the Sun [123]. The efficiency with which heat is transported, measured by the Nusselt number, plays an important role in these natural flow phenomena.

Geo- and astrophysical flows are accompanied by the natural rotation of the respective star or planet. Experiments by, e.g., [114], and [134], show that the heat transfer can be affected by rotation. Both physical experiments and numerical simulations are limited in the range of flow scales that can be reproduced. Typically, the interest is in investigating the relevance of scaling laws to extrapolate the Nusselt number to Rayleigh numbers of practical interest. Here, we focus in particular on the effect of rotation on these scaling laws, also to provide reference material for a possible extension to rotating systems of the theory put forward in [68] for the non-rotating case.

There are various studies of the Nusselt number as function of the rotation rate, i.e., the inverse Rossby number [81, 105, 102, 162, 188]. We can roughly distinguish three regimes with respect to the Rossby number. In the weak-rotation regime ($Ro \gtrsim 2.5$), the flow is dominated by a large-scale circulation. The Nusselt number does not increase with respect to the non-rotating case. In the moderate-rotation regime ($0.15 \lesssim Ro \lesssim 2.5$), the large-scale circulation breaks down due to rotation, and the flow organizes itself in vertically aligned vortices. The Nusselt number increases with the rotation rate. Finally in the strong-rotation regime ($Ro \lesssim 0.15$), rotation dominates the flow structure and suppresses heat transport in the vertical direction. The Nusselt number rapidly decreases with the rotation rate. The drastic effect of rotation on the flow structure is visualized in Fig. 2.5 and 2.6 in Section 2.4.2 later.

In this numerical study, we consider Rayleigh–Bénard convection in a rotating vertical cylinder with a width-to-height aspect ratio $\Gamma = D/L = 1$. The goal is to study the influence of the temperature difference between top and bottom walls (characterised by the Rayleigh number) and the rotation rate (characterised by the Rossby number) on the structure of the flow and the heat transfer. We perform direct numerical simulations for a wide range of Rossby and Rayleigh numbers to investigate the dependence of the Nusselt number. The direct numerical simulations are performed with a spectral element method, implemented in the open-source code *Nek5000*, originally developed by [48]. The purpose of

this work is essentially twofold. On the one hand, we assess the performance of the spectral-element method in direct numerical simulations of Rayleigh–Bénard convection. On the other hand, we seek a deeper understanding of the effect of rotation on the transition from an unsteady laminar flow to a developed turbulent flow when increasing the Rayleigh number.

The organization of this chapter is as follows. We first discuss, in Section 2.2, the governing equations for Rayleigh–Bénard convection including rotation. In Section 2.3, we briefly describe the spectral element method used for DNS and justify the spatial resolution we used. Numerical findings are presented in Section 2.4 in which we establish the dependence of the heat transfer on the Rayleigh and Rossby number. We show that the Nusselt number asymptotically maintains strong scaling with the Rayleigh number also in case of rotation, and that rotation and the temperature difference qualitatively change the flow. Concluding remarks are collected in Section 2.5.

2.2 Boussinesq approximation of rotating Rayleigh–Bénard convection

This section describes the equations of motion, regarding Rayleigh–Bénard convection in a rotating cylinder and the evaluation of the Nusselt number from the simulation data.

2.2.1 Rotating coordinate system

The effect of rotation is taken into account by adopting a co-rotating coordinate system and recasting Newton’s laws into this non-inertial coordinate system. The adoption of such a coordinate system introduces additional (fictitious) body forces. We derive the effects of rotation on the evolution of the flow and start from the Navier-Stokes equation in the inertial coordinate system,

$$\rho(\partial_t + \mathbf{v} \cdot \nabla)\mathbf{v} = -\nabla p + \mu \nabla^2 \mathbf{v} + \rho \mathbf{g}. \quad (2.1)$$

Here, ρ is the density, $\mathbf{v}$ the velocity, p the pressure, μ the molecular viscosity and $\mathbf{g}$ the gravitational acceleration. Following the approach by [101], we make use of the kinematic relation,

$$\mathbf{v} = \mathbf{u} + \boldsymbol{\Omega} \times \mathbf{r}. \quad (2.2)$$

Here, $\mathbf{u}$ and $\mathbf{r}$ are the velocity and position vector in the co-rotating coordinate system, and $\boldsymbol{\Omega}$ is the rotation vector. Substituting relation (2.2) into the Navier-Stokes equation (2.1) yields after a little manipulation,

$$\begin{aligned} \rho(\partial_t + \mathbf{u} \cdot \nabla)\mathbf{u} &= -\nabla p + \mu \nabla^2 \mathbf{u} + \rho \mathbf{g} \\ &\quad - 2\rho \boldsymbol{\Omega} \times \mathbf{u} \\ &\quad - \rho \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times \mathbf{r}). \end{aligned} \quad (2.3)$$

Figure 2.1: Geometry of the rotating cylinder.

The rotation of the non-inertial coordinate system introduces two fictitious (or *d'Alembert*) forces, which are the Coriolis force ($2\rho \mathbf{\Omega} \times \mathbf{u}$) and the centrifugal force ($\rho \mathbf{\Omega} \times (\mathbf{\Omega} \times \mathbf{r})$).

In our study, we consider Rayleigh–Bénard in a cylinder rotating about its vertical axis, as illustrated in Fig. 2.1. The coordinate system rotates along with the cylinder and the rotation vector is $\mathbf{\Omega} = (0, 0, \Omega)^T$, where Ω is simply the rate of rotation.

2.2.2 Boussinesq approximation

Rayleigh–Bénard convection is driven by a temperature difference between the “warm” and “cold” plate. The thermal expansion of the fluid generates buoyancy that sets the fluid in motion. This effect of compressibility can be simplified by the *Boussinesq approximation*. In essence, the fluid is regarded to be incompressible and only the leading-order effects of compressibility are taken into account. A comprehensive description of the Boussinesq approximation is given by [107]. To start with, we consider a density variation ρ' from a reference density ρ_0 , $\rho = \rho_0 + \rho'$. The variation in density is assumed to be small in the sense that $\rho'/\rho_0 \ll 1$. Typically, we need an equation of state to close the governing equations. Here, the equation of state is approximated by,

$$\rho = \rho_0 [1 - \beta(T - T_0)], \quad (2.4)$$

where β is the thermal expansion coefficient and T_0 a reference temperature, taken equal to the temperature of the upper plate. This linearisation is only accurate for small fluctuations in temperature, relative to the reference temperature. By

substituting (2.4) into the Navier-Stokes equation (2.3) and collecting terms of similar magnitude, we find,

$$\begin{aligned} \rho_0(\partial_t + \mathbf{u} \cdot \nabla)\mathbf{u} = & -\nabla q + \mu \nabla^2 \mathbf{u} - \rho_0 \beta (T - T_0) \mathbf{g} \\ & - 2\rho_0 \boldsymbol{\Omega} \times \mathbf{u} \\ & + \rho_0 \beta (T - T_0) \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times \mathbf{r}). \end{aligned} \quad (2.5)$$

Here, we define an effective pressure $q = p + \rho_0 \phi - \rho_0 |\boldsymbol{\Omega} \times \mathbf{r}|^2/2$, where ϕ is a scalar field related to gravity $\mathbf{g} = -\nabla \phi$. The concise notation q is possible since the hydrostatic components of the gravitational and centrifugal force can be written as a gradient. Additionally, we have the energy and continuity equation,

$$\rho C_p (\partial_t + \mathbf{u} \cdot \nabla) T = k \nabla^2 T, \quad (2.6)$$

$$\nabla \cdot \mathbf{u} = 0. \quad (2.7)$$

Here, C_p is the specific heat coefficient at constant pressure, and k the thermal conductivity. The governing equations are complemented by the boundary conditions listed in Table 2.1. A no-slip condition is imposed at the wall and the side-wall of the cylinder is assumed to be perfectly insulated, while at the top and bottom walls the temperature is prescribed.

The Boussinesq approximation was first discovered by [137], but is generally attributed to [18]. The Boussinesq approximation is a common practice motivated by cases with relatively small temperature differences of a few degrees Kelvin between the top and bottom walls. The exact validity of this approximation is examined in close detail by, e.g., [66]. One of the necessary assumptions is that fluid properties, μ , k , and β , are independent of temperature. The additional effects of temperature-dependent viscosity and thermal diffusivity are for example studied by [2].

Table 2.1: Boundary conditions

Bottom plate	$\mathbf{u} = 0$	$T = T_0 + \Delta T$
Top plate	$\mathbf{u} = 0$	$T = T_0$
Side-wall	$\mathbf{u} = 0$	$\frac{\partial T}{\partial \mathbf{n}} = 0$

2.2.3 Dimensionless formulation

Following earlier work by [106] for example, we use the height of the cylinder L as the reference length and the *free-fall velocity* $U = \sqrt{g\beta\Delta TL}$ as the reference velocity. Using these typical scales, the governing equations take the dimensionless

form,

$$\begin{aligned}
 (\partial_t + \mathbf{u} \cdot \nabla) \mathbf{u} = & -\nabla q + (Pr/Ra)^{1/2} \nabla^2 \mathbf{u} + T \mathbf{e}_z \\
 & - (1/Ro) \mathbf{e}_z \times \mathbf{u} \\
 & + Fr T \mathbf{e}_z \times (\mathbf{e}_z \times \mathbf{r}),
 \end{aligned} \tag{2.8a}$$

$$(\partial_t + \mathbf{u} \cdot \nabla) T = (Pr Ra)^{-1/2} \nabla^2 T, \tag{2.8b}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{2.8c}$$

Here, we have used $\mathbf{g} = -\mathbf{e}_z$, $\boldsymbol{\Omega} = \mathbf{e}_z$, and the unit vector $\mathbf{e}_z = (0, 0, 1)^T$. The equations are solved numerically in the dimensionless formulation, together with the dimensionless boundary conditions listed in Table 2.2. In the remaining sections, we implicitly assume that quantities are dimensionless unless stated otherwise.

We can identify four dimensionless numbers that characterize rotating Rayleigh–Bénard convection,

$$Ra = g\beta\Delta T L^3 / (\nu\kappa), \tag{2.9}$$

$$Pr = \nu/\kappa, \tag{2.10}$$

$$Ro = U/(2\Omega L), \tag{2.11}$$

$$Fr = \Omega^2 L/g, \tag{2.12}$$

which are the Rayleigh, Prandtl, Rossby and Froude number respectively. In the present study, we assume that $Fr \ll 1$ and the temperature-dependent component of the centrifugal force can be neglected. A study by [188] shows this assumption is valid for a range of realistic configurations. In the experimental study by [106], we find $Fr < 0.04$ for example.

Table 2.2: Dimensionless boundary conditions

Bottom plate	$\mathbf{u} = 0$	$T = 1$
Top plate	$\mathbf{u} = 0$	$T = 0$
Side-wall	$\mathbf{u} = 0$	$\frac{\partial T}{\partial \mathbf{n}} = 0$

The dynamics of rotating Rayleigh–Bénard convection is characterized by the combination of the said dimensionless parameters. A critical Rayleigh number Ra_c indicates the onset of convection [75], beyond which the hydrostatic equilibrium is no longer stable. Higher Rayleigh numbers lead eventually to a regime from “soft” to “hard” turbulence [87].

In combination with the buoyancy effects driving Rayleigh–Bénard convection, rotation can have a significant influence on the overall flow structuring [104]. Early experimental research by [143] shows that rotation could increase the heat transfer with respect to the non-rotating case. This increase in heat transfer is associated

with a qualitative change in the structure of the flow. A relevant parameter is Rossby number: the ratio of the inertial to the Coriolis force. At high Rossby numbers, i.e., very slow rotation, the effect of rotation is limited and the flow is dominated by a *large-scale circulation* (LSC) [105]. For sufficiently low Rossby numbers, the Coriolis force becomes dominant and is capable of breaking up the LSC. In that case, local vortical structures occur, penetrating well into the domain from both the top and the bottom walls. These structures are characteristic of conditions that display a strongly increased transport of heat due to Ekman pumping [39].

In this chapter we consider a system composed of water and take $Pr = 6.4$ throughout. The Rayleigh number is varied ranging from 10^6 to 10^9 , while the Rossby number is varied from ∞ for the non-rotating case to values as low as ≈ 0.1 .

2.2.4 Global heat transfer

The purpose of the direct numerical simulations is to measure the global heat transfer by the flow. The Nusselt number, which is the ratio between the total and the conductive heat flux, provides the following expression of the local heat transfer (here taken in the vertical direction),

$$Nu = (PrRa)^{1/2} u_z T - \partial_z T. \quad (2.13)$$

The global heat transfer is measured, either by averaging over the walls of the cylinder or the entire volume. The wall- and volume-averaged Nusselt numbers are as follows,

$$\langle Nu \rangle_W = \langle \partial_z T \rangle_W \quad (2.14)$$

$$\langle Nu \rangle_V = 1 + (PrRa)^{1/2} \langle u_z T \rangle_V \quad (2.15)$$

where $\langle \cdot \rangle_W$ and $\langle \cdot \rangle_V$ denotes the average over the walls and the volume, respectively. Here, we have already simplified the averages with the boundary conditions given in Table 2.2, following a similar approach by [93]. Both averages should agree if the averaging time and the spatial resolution are sufficient. Naturally, the Nusselt number of the bulk is sensitive to the resolution in the bulk, and the Nusselt number of the wall to the resolution in the near-wall regions. This provides an extra, a posteriori, check on the spatial resolution used in the numerical simulations.

2.3 Spectral element method

The governing equations, given by Eq. (2.8), are solved numerically with the spectral element method (SEM), that is implemented in the open-source code *Nek5000* [48]. The spectral element method is essentially a variation of the finite element method using higher-order piecewise polynomials as basis functions.

Ref. [33] describes in more detail the spectral element method and its application to fluid dynamics.

The basis functions of the velocity are local tensor-product Lagrange interpolants of order p on Gauss-Lobatto-Legendre nodes, whereas the basis functions of the pressure are Lagrange interpolants of order $p - 2$ on Gauss-Legendre nodes. The total number of grid points is then $(p + 1)^3$ (for the velocity) per three-dimensional element.

For time-integration, a semi-implicit third-order BDF3/EXT3 scheme is used [91]. The viscous term is integrated with a third-order backward differencing scheme (BDF3), and the nonlinear convective term with a third-order extrapolation scheme (EXT3). In general, we use adaptive time-stepping with a target CFL number of 0.5, which is in practice more than sufficient to guarantee the stability during the simulation. A CFL number of $\lesssim 1$ is usually advocated [33].

In turbulent flows at high Rayleigh numbers the physics is dominated by convection, as opposed to diffusion in laminar flows. Skew-symmetry of the convective operator is shown to be crucial for the stability of the numerical scheme [142]. The skew-symmetry of the convective operator can be respected by *over-integration*, which means applying quadrature rules with orders higher than N [121]. Quadrature rules of order $3N/2$, instead of N , are in practice sufficient to approximate the skew-symmetry of the convective operator up to machine precision. Over-integration is indispensable to stabilizing the SEM in convection-dominated flows and, consequently, is applied in the numerical simulations presented in this study.

In this numerical study, we use spectral elements with polynomial order $p = 5$. We perform a convergence test to determine the required number of elements, i.e., spatial resolution, for accurate estimates of the Nusselt number. Here, the mesh is characterized by the number of elements in the z -direction, E_z , and in the xy -plane, E_{xy} . We performed several simulations for $Ra = 10^8$ and $Ra = 10^9$ with the different resolutions given in Table 2.3. To capture the sharp gradients in the boundary layers, the meshes are refined in the near-wall region. The grids used in this study vary from about 10^6 degrees of freedom ($E_z \times E_{xy} \times (p + 1)^3$) to $\approx 1.8 \cdot 10^7$.

The goal here is to find an appropriate resolution for the simulations presented in the remaining sections. We assume that cases with the highest rotation rate of $Ro = 0.09$ are the most demanding cases in terms of spatial resolution. We run several simulations for the two Rayleigh numbers, $Ra = 10^8$ and $Ra = 10^9$. The simulations run for a total of 300 time units, starting from a zero-velocity field and a linear temperature profile ($T = z$) as initial conditions. The Nusselt number is only averaged over the last 200 time units, in which the flow has reached an approximately statistically stationary state. We estimate a 95% confidence bound by taking uncorrelated samples from the available history and calculating the standard mean error. The Nusselt numbers for different resolutions are given in Table 2.4.

For $Ra = 10^8$, both Nusselt numbers, $\langle Nu \rangle_V$ and $\langle Nu \rangle_W$, are seen to have converged with the second and third mesh, and agree within their uncertainty

Table 2.3: Number of spectral elements in z -direction, E_z , and in the xy -plane, E_{xy} , with their respective average mesh widths $\bar{h}_z$ and $\bar{h}_{xy}$.

Mesh	E_z	$\bar{h}_z$	E_{xy}	$\bar{h}_{xy}$
1	16	$6.250 \cdot 10^{-2}$	300	$4.976 \cdot 10^{-2}$
2	24	$4.167 \cdot 10^{-2}$	588	$3.555 \cdot 10^{-2}$
3	32	$3.125 \cdot 10^{-2}$	972	$2.766 \cdot 10^{-2}$
4	48	$2.083 \cdot 10^{-2}$	1728	$2.078 \cdot 10^{-2}$

Table 2.4: Convergence of the time-averaged Nusselt number with increasing spatial resolution, in the case $Ro = 0.09$.

Ra	Mesh	$\langle Nu \rangle_W$	$\langle Nu \rangle_V$
10^8	1	37.5 ± 0.4	40.2 ± 1.2
—	2	38.0 ± 0.5	38.1 ± 1.2
—	3	37.9 ± 0.6	38.2 ± 1.6
10^9	1	77.8 ± 0.7	108.5 ± 1.9
—	2	72.2 ± 0.8	88.7 ± 2.2
—	3	73.4 ± 1.0	80.9 ± 3.1
—	4	72.2 ± 0.9	73.8 ± 1.9

bounds. The results also suggest that a higher resolution is required in the case that $Ra = 10^9$. From these results, the formal order of convergence cannot be established given the statistical errors in the time-averaged Nusselt number. The value of $\langle Nu \rangle_V$ appears to be more sensitive than $\langle Nu \rangle_W$ to the number of elements used. The slow convergence of $\langle Nu \rangle_V$ in comparison with $\langle Nu \rangle_W$ could be explained by the relatively lower resolution in the bulk, due to the significant mesh refinement near the wall. Based on these results, we decide to use the third resolution for $Ra \leq 10^8$, and the fourth for $Ra = 10^9$. At $Ra = 10^9$, we have achieved grid convergence for $\langle Nu \rangle_W$ (given the uncertainty bounds) and an agreement with $\langle Nu \rangle_V$, which we assume to be sufficient for our application. We did not fully demonstrate grid convergence for $\langle Nu \rangle_V$ considering the required computational resources.

2.4 Heat transfer scaling and flow structure

In this section, we first analyze the scaling of the Nusselt number with the Rayleigh number, under influence of steady rotation in Subsection 2.4.1. We show that the Nusselt number increases up to 15% with respect to the non-rotating case, depending on the rate of rotation. Subsequently, in 2.4.2 we illustrate the qualitative changes in the flow structure as a result of changes in Ra and Ro .

2.4.1 Scaling of the heat transfer under rotation

In general, we are interested in scaling laws of the form, $Ra = \alpha Nu^\beta$. A universal law that covers the entire parameter space, does not exist, as the coefficients α and β depend on the dimensionless parameters themselves. A comprehensive theory of scaling laws is proposed by [68] that accounts for different regimes in the (Ra, Pr) parameter space. The boundaries between these regimes are not sharp, allowing for transitional scaling laws to prevail. Experiments for low Prandtl numbers by [21] show that the exponent $\beta = 2/7$ holds in a large range of Rayleigh numbers ($Ra > 4 \cdot 10^7$). The existence of a $2/7$ -regime is theoretically supported by [156]. Regarding rotating Rayleigh–Bénard convection, the question arises: is there a scaling law, $Nu \propto Ra^\beta$? If so, what is the scaling exponent β ?

We run simulations for 300 time units, starting with a zero-velocity field and a linear temperature profile ($T = z$) as initial conditions. The Nusselt number of the wall, $\langle Nu \rangle_W$, and of the volume, $\langle Nu \rangle_V$, are averaged over the last 200 time units, in which the flow has reached a statistically stationary state. The time-averaged values of Nu are given in Table 2.5, in which Ra varies from 10^6 to 10^7 and Ro from 0.09 to ∞ . For each simulation, the two Nusselt numbers agree within the 95% confidence bounds. The convergence study presented in Section 2.3, suggests that this is an indication of numerical convergence with respect to the spatial resolution of the simulations. Using these grids, the main structures in the flow associated with heat transfer, e.g., boundary layers near all vertical and horizontal walls, appear well captured.

The convergence study in Section 2.3 also shows that $\langle Nu \rangle_W$ converges faster than $\langle Nu \rangle_V$, when increasing the resolution. The time-average of $\langle Nu \rangle_W$ has a smaller statistical error too. In the remainder of this chapter, we evaluate Nu via $\langle Nu \rangle_W$, as it appears more robust, both numerically *and* statistically, than $\langle Nu \rangle_V$.

The scaling of the Nusselt number with the Rayleigh number is illustrated by plotting the data in a logarithmic scale in Fig. 2.2. We compare the results to the theoretical scaling for non-rotating Rayleigh–Bénard convection by [68], with the updated prefactors by [164]. The results for the non-rotating case, $Ro = \infty$, agree closely with the theoretical predictions. In addition, we observe that the Grossmann-Lohse theory can be approximated with a $2/7$ power law in a certain range of Rayleigh numbers. A least squares fit in the range $10^7 \leq Ra \leq 10^9$, and $Ro = \infty$, produces the power law $Nu \approx 0.15 Ra^{0.29}$, which is close to the theoretical exponent of $2/7$. This result is also in close agreement with the scaling $Nu = 0.145 Ra^{0.294}$, observed in direct numerical simulations by [5]. Those results were obtained with a finite volume method, used by several groups working in the field of Rayleigh–Bénard convection [180, 179]. In the lower range $Ra \lesssim 10^7$ the scaling exponent deviates slightly from $2/7$. This range of Rayleigh numbers is characterized by “soft” turbulence, in which a $2/7$ power law does not hold [21].

The scaling of Nu with Ra is shown in more detail in Fig. 2.3. Here, we have compensated the Nusselt number by a presumed scaling exponent of $2/7$. The effect of *rotation* on the scaling of Nu is not entirely straightforward. As for

Table 2.5: Time-averaged Nusselt numbers, including the 95% confidence bounds, for $Pr = 6.4$, $\Gamma = 1$ and varying Ra and Ro .

Ra	Ro	$\langle Nu \rangle_W$	$\langle Nu \rangle_V$
10^6	0.09	5.7 ± 0.2	5.5 ± 0.2
10^6	∞	9.0 ± 0.1	9.0 ± 0.1
10^7	0.09	16.0 ± 0.3	16.1 ± 0.5
10^7	0.36	18.8 ± 0.1	18.8 ± 0.4
10^7	1.08	17.3 ± 0.1	17.4 ± 0.3
10^7	∞	16.4 ± 0.1	16.5 ± 0.2
10^8	0.09	37.9 ± 0.3	38.2 ± 0.8
10^8	∞	33.0 ± 0.1	33.2 ± 0.4
10^9	0.09	72.2 ± 0.5	73.8 ± 1.0
10^9	0.36	71.2 ± 0.2	72.2 ± 0.9
10^9	1.08	66.8 ± 0.2	67.0 ± 1.6
10^9	∞	64.5 ± 0.3	66.5 ± 1.8

the non-rotating case, a single power law seems to be inadequate in describing the scaling of Nu in the entire range of Rayleigh numbers. For $Ra \gtrsim 10^8$, the scaling appears to be similar to $2/7$, for all Rossby numbers. The existence of a $2/7$ power law in rotating Rayleigh–Bénard convection was also observed in other experiments and simulations [88, 113], independent of the Rossby number. Our results do suggest a weak dependence on the Rossby number. The scaling exponent in the range $10^8 \lesssim Ra \lesssim 10^9$ subtly decreases with the inverse Rossby number. At $Ro = 0.36$ and $Ro = 0.09$ we observe scaling exponents that are slightly below $2/7$. Because of limited computational resources, we have not been able to explore the range $Ra > 10^9$ yet. For $Ra \lesssim 10^7$ we do not observe a uniform scaling at all. At these low Rayleigh numbers various effects of the relatively high viscosity have to be taken into account.

In Fig. 2.4, the Nusselt number is plotted against the inverse Rossby number. Here, the Nusselt number is normalized by the value of the non-rotating case ($Ro = \infty$) to illustrate the relative increase. The results for $Ra = 10^9$ agree with the DNS data by [105], which are performed for identical physical parameters ($Ra = 10^9$, $Pr = 6.4$, and $\Gamma = 1$). We can also distinguish the three regimes of rotation, described by [102]. In the weak-rotation regime, the heat transport does not increase. In the moderate-rotation regime, the Nusselt number increases with the inverse Rossby number. In the strong-rotation regime, the Nusselt number rapidly decreases. This Rossby-number dependence is also observed in experiments by [102] and [188].

We find a maximum increase of 18% at $Ra = 10^7$ and $Ro = 0.18$, and 15% at $Ra = 10^9$ and $Ro = 0.18$. The increase of heat transfer can be attributed to a

Figure 2.2: Scaling of Nu with Ra . $\diamond$: $Ro = \infty$, $\circ$: $Ro = 1.08$, $\triangle$: $Ro = 0.36$, $\times$: $Ro = 0.09$, dashed: $0.15Ra^{0.29}$ (least squares fit $10^7 \leq Ra \leq 10^9$), **solid**: GL theory with updated prefactors [164].

Figure 2.3: Scaling of Nu with Ra , compensated by $Ra^{2/7}$. $\diamond$: $Ro = \infty$, $\circ$: $Ro = 1.08$, $\triangle$: $Ro = 0.36$, $\times$: $Ro = 0.09$, dashed: $0.15Ra^{0.29}$ (least squares fit $10^7 \leq Ra \leq 10^9$), **solid**: GL theory with updated prefactors [164].

phenomenon called *Ekman transport*, first described by [39]. The rotation creates vortices in the boundary layer, that essentially “pump” fluid into the bulk. These vortices provide a more efficient mechanism of transferring heat compared to the pure non-rotating turbulent flow. This is expressed by an increase in the Nusselt number.

The results for $Ra = 10^7$ and $Ra = 10^9$ show a comparable trend in the Nusselt number. For $Ra = 10^7$ the Nusselt number shows initially a stronger increase, but eventually a faster decrease in the strong-rotation regime. The difference might be explained by the fact that, with decreasing Ra , the buoyancy becomes weaker with respect to the Coriolis force. Our results imply that the effect of rotation is more pronounced at lower Rayleigh numbers. A similar observation is made by [182].

Figure 2.4: The Nusselt number as function of the inverse Rossby number. The Nusselt number is normalized by its non-rotating value. $\circ$: $Ra = 10^7$, $\times$: $Ra = 10^9$, $\triangle$: DNS for $Ra = 10^9$, $Pr = 6.4$ [105]. The vertical dashdotted line indicates the transition between the weak- and moderate-rotation regime [182], and the vertical dotted line the transition between the moderate- and strong-rotation regime [102].

2.4.2 Change in flow structure

To visualize the effect of the Rayleigh and Rossby numbers on the three-dimensional flow, we compare the velocity and temperature solutions in the non-rotating case ($Ro = \infty$) with the rapidly rotating one ($Ro = 0.09$). Figures 2.5 and 2.6 show several snapshots of the temperature and the vertical velocity field for $Ro = \infty$ and $Ro = 0.09$, with Ra ranging from 10^6 to 10^9 .

At $Ro = \infty$, convection is dominated by a *large-scale circulation* [105]. Thermal plumes are visible in the temperature field when increasing the Rayleigh number. The flow shows very different patterns at $Ro = 0.09$, i.e., in case of strong steady rotation. Both the temperature and velocity field exhibit long structures in the vertical direction. These structures are generally described as Taylor columns. According to the Taylor-Proudman theorem, the flow tries to align itself with the rotation axis [94]. The vortices are created by Ekman transport in the horizontal boundary layers, which lead to the enhanced transport of heat away from the wall [162]. These vortices become particularly apparent at higher Rayleigh numbers. At $Ra = 10^6$ for example, the effect of rotation is practically indiscernible in the temperature field. These findings corroborate the previous assertion that rotation can increase the heat transfer at $Ra \geq 10^7$, with respect to the non-rotating case.

2.5 Conclusions

We applied direct numerical simulations, on the basis of a spectral element spatial discretisation method, to study the scaling of heat transport in rotating Rayleigh-Bénard convection in a cylindrical container with aspect ratio $\Gamma = 1$. For $Ro = \infty$, we find $Nu \propto Ra^{0.29}$ in the studied range of $10^6 \leq Ra \leq 10^9$, which matches well with the expected $2/7$ scaling from literature. For $0.09 \leq Ro \leq 1.08$, a similar scaling of the Nusselt number seems to apply in the high Rayleigh number regime of $Ra \gtrsim 10^8$. In this regime, the Nusselt number also increases up to 18% for $Ra = 10^7$ and 15% for $Ra = 10^9$ with respect to non-rotating case. The enhanced heat transport is linked to the vertical vortices, created by the rotation of the system, that are observed in the temperature and the velocity field.

Acknowledgements

This project is supported financially by NWO, the Netherlands Organisation for Scientific Research, through FOM, Foundation for Fundamental Research on Matter, as part of the “Ultimate Turbulence” program. The simulations were made possible through grant SH-061 of the Computational Science Board of NWO, and executed on the supercomputers of SURFsara in Amsterdam. The work of MAB is supported in part by the Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow through the Russian Science Foundation grant 14-11-00659.

Figure 2.5: Isosurfaces of temperature field T for Ra from 10^6 to 10^9 , and $Ro = \infty$ and $Ro = 0.09$. Red color corresponds to $T = 0.65$ and blue to $T = 0.35$.

Figure 2.6: Isosurfaces of vertical velocity field u_z for Ra from 10^6 to 10^9 , and $Ro = \infty$ and $Ro = 0.09$. Red color corresponds to $u_z = 0.07$ and blue to $u_z = -0.07$.

Comparison of computational codes for direct numerical simulations of turbulent Rayleigh–Bénard convection

Computational codes for direct numerical simulations of Rayleigh–Bénard (RB) convection are compared in terms of computational cost and quality of the solution. As a benchmark case, RB convection at $Ra = 10^8$ and $Pr = 1$ in a periodic domain, in cubic and cylindrical containers are considered. A dedicated second-order finite-difference code (AFID/RBFLOW) and a specialized fourth-order finite-volume code (GOLDFISH) are compared with a general purpose finite-volume approach (OPENFOAM) and a general purpose spectral-element code (NEK5000). Reassuringly, all codes provide predictions of the average heat transfer that converge to the same values. The computational costs, however, are found to differ considerably. The specialized codes AFID/RBFLOW and GOLDFISH are found to excel in efficiency, outperforming the general purpose flow solvers NEK5000 and OPENFOAM by an order of magnitude with an error on the Nusselt number Nu below 5%. However, we find that Nu alone is not sufficient to assess the quality of the numerical results: in fact, instantaneous snapshots of the temperature field from a near wall region obtained for deliberately under-resolved simulations using NEK5000 clearly indicate inadequate flow resolution even when Nu is converged. Overall, dedicated special purpose codes for RB convection are found to be more efficient than general purpose codes.

*Based on: G.L. Kooij, M.A. Botchev, E.M.A. Frederix, B.J. Geurts, S. Horn, D. Lohse, E.P. van der Poel, O. Shishkina, R.J.A.M. Stevens, and R. Verzicco. *Comparison of computational codes for direct numerical simulations of turbulent Rayleigh–Bénard convection*. Submitted.

3.1 Introduction

Rayleigh-Bénard (RB) convection is the flow driven by buoyancy forces when a fluid layer is heated from below and cooled from above [3, 27, 89, 116]. The main governing parameter for RB convection is the Rayleigh number Ra which is the ratio between the destabilizing buoyancy and the stabilizing viscous and diffusive effects. For sufficiently large Ra , RB flow becomes turbulent. To understand the flow physics, direct numerical simulation (DNS) is in principle a straightforward approach that can be used to study turbulence dynamics and heat transfer. For low Ra this is now routinely done. For higher Ra however, it is much more challenging, though there are many scientific questions. For example, how does the heat transfer scale with the Ra number in the regime of very high Ra numbers ($Ra \gtrsim 10^{14}$)? This regime, referred to as ‘ultimate’, is characterized by an enhanced heat transfer and associated with a transition to fully turbulent boundary layers [68, 69, 74]. DNS, provided it achieves proper accuracy, could be very useful in providing a deeper insight into the nature of this transition and the properties of this ultimate state. Unfortunately, DNS becomes exceedingly demanding as Ra increases, since turbulence produces smaller flow scales that need finer spatial resolution and proportionally small time steps to track their dynamics. For Ra where the ultimate regime is expected the needed computational power is, at present, prohibitive and understanding which numerical code is most cost efficient is a key issue to establish a roadmap for the computer simulations of turbulent RB convection.

Over the years, several codes suitable for DNS of turbulent RB convection have been developed. We compare four of these codes. The first is based on the work by Verzicco *et al.* [180, 179] in which a second-order energy conserving finite-difference method. For the periodic domain simulation we use the AFID code, which was developed by Van der Poel *et al.* [174]. For the cylindrical simulations we use the latest version of RBFLOW, which is an optimized version of the code used by Stevens *et al.* [165, 163]. The second code is GOLDFISH by Shishkina *et al.* [153, 155, 152], which is based on a finite-volume approach and uses discretization schemes of the fourth-order in space. GOLDFISH can be used to study turbulent thermal convection in cylindrical and parallelepiped domains. The third code is a general purpose open-source code NEK5000, based on the spectral-element method described by Fischer [48]. This code is designed to handle a large variety of flow problems, and was also used in the context of RB convection [97, 148]. The fourth code is an open-source software package OPENFOAM [183]. More precisely, its widely used second-order finite-volume scheme was selected for the comparison.

In this study, we compare the four codes in terms of computational efficiency and quality of the results, with a special focus on the heat transport by the turbulent flow, measured by the Nusselt number (Nu). The efficiency of the codes is assessed in relation to their computational costs and the capability to achieve grid converged results. We simulate RB convection in three different geometries, i.e., a periodic domain, a cubic container, and a cylindrical container. Experiments for

RB convection are typically conducted in a cylindrical tank. A major challenge to the DNS is to handle sharp gradients in the boundary layers near the walls as well as to capture thermal structures (plumes) that protrude far into the bulk of the flow.

In this paper we perform a convergence test at $Ra = 10^8$ and $Pr = 1$ in which we compare several levels of mesh refinement. We show that all four codes produce the same Nu number when appropriate spatial resolution is used. At high resolutions, the results become practically identical, taking into account a small uncertainty due to the finite averaging time. Regarding the computational costs per grid point, we observe significant differences among the codes, which illustrates the contrast between general purpose and specialized codes. The special purpose codes AFID/RBFLOW and GOLDFISH appear to be most efficient in terms of cost. When we increase the Ra number for a fixed spatial resolution, we observe, not surprisingly, that for all codes eventually the resolution becomes insufficient for accurately resolving the turbulent flow. However, the Nu number calculation seems to be more robust against deliberate underresolution in the higher order codes like NEK5000 than in the lower order codes. In this context, NEK5000 follows the theoretical scaling of Nu versus Ra better than the others. This might suggest that, for a given number of grid points, the NEK5000 code is capable of correctly capturing the flow physics even when the other codes fail. A study in [59] already hinted on the benefit of using high-order discretizations even in coarsely resolved turbulent flows. A direct inspection of some instantaneous snapshots of temperature in the near wall region, however, clearly shows that this is not the case since the temperature distribution displays the footprint of the underlying discretization. The conclusion is that the evaluation of the Nu alone is not a sufficient criterion to assess the quality of the results that, instead, should be assessed by evaluating more than one quantity. In this paper we also discuss some other advantages and drawbacks of the compared codes.

The remainder of this paper is organized as follows. In Section 3.2, we describe the governing equations of RB convection and the geometries of the domains included in this study. The codes are described in more detail in Section 3.3 and the results are compared in Section 3.4. Last, a summary and conclusions are given in Section 3.5.

3.2 Governing equations and evaluation of the Nusselt number

In this section we present the mathematical model and introduce the methods adopted to evaluate the Nu number. We consider RB convection in three different geometries: a periodic domain, a cube, and a cylinder. Every considered RB cell is characterized by a width D and a height H . A flow in any RB cell is determined by the dimensionless parameters, which are the Rayleigh number $Ra = g\beta\Delta H^3/(\nu\kappa)$, the Prandtl number $Pr = \nu/\kappa$, and the aspect-ratio $\Gamma = D/H$. Here g is grav-

itational acceleration, β the thermal expansion coefficient, Δ the temperature difference between the upper and lower plate, ν the kinematic viscosity, and κ the thermal diffusivity of the fluid. In this study, we consider $\Gamma = 1$ for all geometries, and $Pr = 1$, which means that the inner length scales of the velocity and the temperature fields are of similar order.

In the numerical simulations, we solve the incompressible Navier-Stokes equation with the Boussinesq approximation to account for buoyancy effects. The governing equations read

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \sqrt{\frac{Pr}{Ra}} \nabla^2 \mathbf{u} - \nabla p + \theta \mathbf{e}_z, \quad (3.1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (3.2)$$

$$\frac{\partial \theta}{\partial t} + \mathbf{u} \cdot \nabla \theta = \frac{1}{\sqrt{Pr Ra}} \nabla^2 \theta, \quad (3.3)$$

where $\mathbf{u}$ is the velocity, p the pressure, θ the temperature, and $\mathbf{e}_z$ the unit vector in the vertical direction anti-parallel to the gravitational acceleration. Here lengths are expressed in terms of H , velocities in terms of free fall velocity $U = \sqrt{\beta g \Delta H}$, and temperatures in terms of Δ . No-slip and constant temperature conditions are imposed at the plates and the sidewall, the latter assumed adiabatic. We do not simulate all geometrical configurations with all codes, since not every geometry is feasible in every code. For the periodic domain, we compare AFID and NEK5000, and for the cubic container GOLDFISH and NEK5000. The cylindrical domain is simulated with all four codes included in this study.

One of the main aspects of RB convection is the heat transported by the turbulent flow from the lower to the upper plate. The heat transfer is quantified by the dimensionless heat flux, i.e. the Nu number which is the ratio of the actual specific heat flux to the purely conductive counterpart. Following [165] we consider several ways to compute Nu . First we consider those of them, which are related directly to the gradient of the temperature and to the convective heat transport. As a function of the vertical coordinate z , Nu is defined as the average heat flux through a horizontal cross section of the domain [178], $Nu(z) = -\langle \partial_z \theta \rangle_A + \sqrt{Pr Ra} \langle u_z \theta \rangle_A$ where $\langle \cdot \rangle_A$ denotes the average over a horizontal cross section A and in time. From the no-slip boundary conditions, it follows that the Nu numbers at the lower and upper plate, denoted by Nu_{lo} and Nu_{up} respectively, can be calculated from the average temperature gradient at the plates only. We also define the average of the two as $Nu_{pl} := (Nu_{lo} + Nu_{up})/2$. The third definition is obtained using the volume average $Nu_{vol} := 1 + \sqrt{Ra Pr} \langle u_z \theta \rangle_V$, where $\langle \cdot \rangle_V$ denotes the average over the complete volume of the domain. Note that these definitions of Nu are averaged over time as well.

Two more definitions can be obtained from the global balance of energy. We can derive a relation between the Nu number and the kinetic and thermal dissipation rates [156]. The kinetic and thermal dissipation rates are $\varepsilon := \sqrt{Pr/Ra} (\nabla \mathbf{u})^2$, $\varepsilon_\theta := 1/(\sqrt{Pr Ra}) (\nabla \theta)^2$ and the Nu number can be calculated from the kinetic

and thermal dissipation rate respectively as follows: $Nu_{kin} := 1 + \sqrt{PrRa} \langle \varepsilon \rangle_V$, and $Nu_{th} := \sqrt{PrRa} \langle \varepsilon_\theta \rangle_V$. These relations from the global balance of energy are sometimes used to assess the quality of DNS of RB convection. If the simulation is well resolved, the global balance of energy is respected accurately. When averaged over time Nu calculated from the dissipation rates agrees with the other definitions of the Nu number. Note that the converse is not necessarily true as will be illustrated in Section 3.4.1. In particular, if some definitions of Nu agree with each other then this does not automatically imply that the resolution is adequate.

Being an integral quantity, Nu is one of the main characteristics in RB convection and is, therefore, a natural quantity to investigate. Of course, besides Nu , there are other quantities describing the turbulent RB flow that one could include in a comparison. A good prediction of Nu does not automatically guarantee that other quantities are approximated accurately, in particular higher order moments will converge less easily. However, the converse holds, i.e., Nu predictions will correspond closely if the solution is accurately captured. In the present comparison study, we focus mostly on Nu , because it is one of the most important quantities and it gives a first indication of how well different codes perform.

3.3 Numerical methods

In this section, we provide a brief description of the four codes that are compared. Detailed information can be found in the mentioned references.

3.3.1 AFID/RBflow

The second-order finite-difference scheme has initially been developed by Verzicco *et al.* [180, 179] for cylindrical containers. Time integration is performed with an third order Runge-Kutta method, in combination with a second-order Crank-Nicolson scheme for the viscous terms. RBFlow, which is used for the simulations in the cylindrical domain, computes all viscous terms implicitly. The open-source code AFID, specialized for domains with the two periodic horizontal directions, uses an explicit scheme in the non-bounded directions to improve scalability of the code [174]. In AFID, the pressure is solved using a fast Fourier Transform (FFT) in the horizontal directions by means of a 2D pencil decomposition [174].

3.3.2 Nek5000

The open-source package NEK5000 is based on the spectral element method, which is an essential extension of the standard finite element method to the case of higher-order basis functions. In this case, the basis functions for the velocity and the pressure are tensor product Lagrange polynomials of order $\mathcal{N}$. Details of the code are found in Ref. [48]. The spectral element method has been used successfully for DNS of RB convection [97, 148]. We use the so-called P_N - P_N formulation, based on the splitting scheme in Ref. [169]. This means that the

spectral elements for the velocity components and the pressure are both order $\mathcal{N}$. In the more traditional $P_{\mathcal{N}}-P_{\mathcal{N}-2}$ formulation the pressure is treated with order $\mathcal{N}-2$. Slightly more accurate results can be obtained with the higher order approximation of the pressure in the $P_{\mathcal{N}}-P_{\mathcal{N}}$ formulation for moderately resolved turbulent flows. In our simulations, we use $\mathcal{N} = 8$, which is similar to what others have used in DNS of turbulent flows [40, 138, 148]. The viscous term is treated implicitly with the second-order backward differentiation formula, in combination with an explicit second-order extrapolation scheme for the convective and other terms. The linear system for the velocity is solved with the conjugate gradient method using Jacobi preconditioning. The linear system for the pressure is solved with the generalized minimal residual method, preconditioned with an additive Schwarz method.

3.3.3 Goldfish

The computational code GOLDFISH is based on a finite-volume approach. To calculate the velocity and temperature at the surfaces of each finite volume, it uses higher-order discretization schemes in space, up to the fourth order in the case of equidistant meshes. GOLDFISH has been used to study thermal convective flows in different configurations [152, 153, 155], in cylindrical and parallelepiped domains. For the time integration, the leapfrog scheme is used for the convective term and the explicit Euler scheme for the viscous term. Although formally first order in time, the accuracy is close to the second-order in convection dominated flows [184, Section 5.8]. Note that due to the von Neumann numerical stability of the chosen scheme, the fourth-order spatial discretization requires asymptotically at least $4/3$ times finer time stepping than the second-order scheme [151]. Due to the regularity of the used computational meshes, direct solvers are applied to compute the pressure in cylindrical and Cartesian coordinate systems. Thus, when the RB container is a cylinder, FFT is used in two directions. In the case of a parallelepiped RB container, the grid regularity also allows separation of variables. In this case, the corresponding eigenvalues and eigenvectors for the pressure solver are calculated and stored at the beginning of the simulations. The code is quite flexible in parallelization, including parallel I/O, and is characterized by high modularity and is applicable to different configurations of turbulent thermal convective flows.

3.3.4 OpenFOAM

OPENFOAM is a widely used open-source second-order finite-volume software package [183]. Although OPENFOAM offers many different options, we use conventional choices that are representative for OPENFOAM in order to relate characteristic engineering OPENFOAM usage to the other, more specialized codes. A linear interpolation scheme is used for the convective term. The equations are solved with the PISO algorithm. The default implementation of the second-order Crank-Nicolson scheme is used for time integration. Furthermore, we do not use

Figure 3.1: Nu against the number of grid points for different geometries: (a) Periodic domain (b) Cubic domain (c) Cylindrical domain. Markers: $\times$ AFID, $\diamond$ GOLDFISH, $\square$ NEK5000, $\circ$ OPENFOAM. Lines: $-$ Nu_{vol} , $---$ Nu_{pl} , $- \cdot -$ Nu_{kin} , $\cdots$ Nu_{th} .

the “non-orthogonal” correction for the non-orthogonality of the mesh.

3.4 Performance comparison

In this section, we present the results of the simulations using two specialized RB convection codes (AFID/RBFLOW and GOLDFISH) and two general purpose codes (NEK5000 and OPENFOAM). Our findings shed some light on the issue of the relevance of general purpose codes for moderate Ra number turbulence. We first discuss a convergence test for a moderate $Ra = 10^8$, for which a fully resolved DNS is easily affordable. A comparison in terms of quality of results and cost is made in Section 3.4.1. Finally, some results at higher Ra numbers and fixed spatial resolution are shown in Section 3.4.2, illustrating the inevitable loss of accuracy with significant rise of Ra .

The simulations with AFID/RBFLOW, NEK5000, and OPENFOAM, are all performed on Cartesius (SURFsara). The simulations with GOLDFISH are performed on SuperMUC of the Leibniz-Rechenzentrum (LRZ). The hardware of both machines is very similar, so we do not expect significant differences in performance as a result of hardware architecture differences. Also, the number of cores may also differ per simulation, but the codes run well below their maximum achievable speedup. We therefore assume that the parallel efficiency is high enough, such that it does not play a significant role when comparing computation times.

3.4.1 Convergence test at $Ra = 10^8$

Here we present a classical convergence test in which a number of grid refinements is undertaken to assess the sensitivity of the results to spatial resolution. Because of the fundamental differences between the spatial discretization techniques, the simulations on the different levels of grid refinement need to be performed with

Table 3.1: Nu obtained with the highest spatial resolutions for the periodic domain, where $N_x \times N_y \times N_z = 384 \times 384 \times 384$ for AFID and $N_x \times N_y \times N_z = 379 \times 379 \times 379$ for NEK5000.

	Nu_{lo}	Nu_{up}	Nu_{vol}
AFID	32.24	32.27	32.18
NEK5000	32.29	32.41	32.54
Average	32.32 ± 0.24 (0.73%)		

Table 3.2: Nu obtained with the highest spatial resolutions for the cubic container, where $N_x \times N_y \times N_z = 384 \times 384 \times 384$ for GOLDFISH and $N_x \times N_y \times N_z = 379 \times 379 \times 379$ for NEK5000.

	Nu_{lo}	Nu_{up}	Nu_{vol}
GOLDFISH	31.56	31.49	31.47
NEK5000	31.53	31.58	31.53
Average	31.53 ± 0.08 (0.24%)		

slightly different meshes. For a fair comparison, at any grid refinement level, the total number of grid points or degrees of freedom (which for simplicity we also call grid points) is kept similar for all codes. The mesh refinement is undertaken near the plates. The boundary layer thicknesses are estimated a priori using the scaling theory by Grossmann and Lohse (GL theory) [68, 154, 165, 164]. In the studied case of $Pr = 1$, the thicknesses of the kinetic and thermal boundary layers are similar. At any considered level of grid refinement, the total number of grid points, N , and the number of grid points inside the boundary layers, N_{BL} , of the chosen meshes are similar for all codes. The spatial resolutions and the corresponding values of N and N_{BL} are listed in Appendix A.

The Nu numbers are averaged over 300 dimensionless time units after the solution approaches a statistically stationary state, which takes about 200 time units, depending on the initial conditions. Nu , obtained at different levels of grid refinement, versus the number of the grid points, is presented in Fig. 3.1. We observe that at high grid resolutions all codes converge to the same result within a small time averaging error of about 0.5%. Different ways to calculate Nu and different codes lead to different convergence of the obtained Nu with increasing grid resolution. For example, the results for NEK5000 at very coarse resolutions are quite inaccurate, but they converge quickly to the final value when the resolution is increased. We can interpret the results in Fig. 3.1 as a good indication of convergence to nearly grid independent results, achieved by all codes, albeit at different spatial resolutions. Ultimate convergence assessment is hampered by the degree of time averaging uncertainty that remains, we come back to this momentarily.

Table 3.3: Nu obtained with the highest spatial resolutions for the cylindrical container, where $N_r \times N_\phi \times N_z = 192 \times 512 \times 384$ for RBFLOW/GOLDFISH, $N_{xy} \times N_z = 85009 \times 384$ for NEK5000, and $N_{xy} \times N_z = 110592 \times 384$ for OPENFOAM.

	Nu_{lo}	Nu_{up}	Nu_{vol}
RBFLOW	32.08	32.15	32.24
GOLDFISH	32.19	32.31	32.33
NEK5000	32.26	32.23	32.16
OPENFOAM	32.16	32.10	32.13
Average	32.20 ± 0.14 (0.44%)		

For a fixed time of statistical averaging (and a fixed number of iterations in the case of iterative solvers), the costs are proportional to the mesh size, both in space and time. The Courant-Friedrichs-Lewy (CFL) condition and the numerical stability of the simulations were the two restrictions on the time stepping that were taken into account in all conducted simulations. As one can see in Fig. 3.2, the computational costs scale from $\mathcal{O}(N^{4/3})$ to $\mathcal{O}(N^{5/3})$ in all cases, in a full accordance to the von Neumann stability of the schemes used in the corresponding codes. For the schemes, which are optimal with respect to the von Neumann stability, the time step size τ is taken proportional to the mesh width h , which in turn is proportional to $N^{-1/3}$. This leads to the scaling of the computational costs with the mesh size as $\mathcal{O}(N^{4/3})$. Apart from the von Neumann stability, there exists also another restriction on the time stepping in accurate DNS, which is the resolution of the Kolmogorov time micro-scales; we will come back to this issue in Section 3.5.

Further, in Fig. 3.2, we observe that the general purpose codes designed for unstructured grids in complex geometries are more expensive than those for structured grids, exploiting the periodic directions of the geometry or, at least the possibility to separate variables due to the regularity of the grids. For example, AFID and OPENFOAM are both second-order accurate, but OPENFOAM is much closer to NEK5000 in terms of computational cost. Apart from the grid organization, the order of the schemes, used in the codes, influences the computational load. Obviously, the higher-order schemes need more operations per time step than the lower-order schemes. Also the way to solve the Poisson equation for the pressure-like function determines the efficiency. In general purpose codes only iterative computationally intensive solvers can be employed, while specialized codes can use optimized direct solvers, which are much more efficient and scale well by exploiting the regular structure of the problem. Also the parallel scalability of the codes influences the total computational costs. OPENFOAM, for example, is characterized by quite modest scalability, compared to the other considered codes.

The Nu values obtained in the highest resolution simulations are listed in Tables 3.1, 3.2, and 3.3, for the periodic domain, cube, and cylinder, respectively. These

Figure 3.2: Computational cost against the number of grid points for different geometries: (a) Periodic domain (b) Cubic domain (c) Cylindrical domain. $\times$ AFID/RBFlow, $\diamond$, GOLDFISH, $\square$ NEK5000, $\circ$ OPENFOAM, $---$ $O(N^{4/3})$ and $\cdots$ $O(N^{5/3})$.

data can be used to further quantify the error in Nu versus computational costs. To arrive at this we undertake a number of steps. First, we calculate the average Nu as reference points. We calculate the standard deviation from the data given in the tables and use twice the standard deviation as a 95% confidence interval for the average values. These average Nu are subsequently used as a reference value to calculate the error in Nu_{lo} , Nu_{up} , and Nu_{vol} separately for the different codes and grids. After that, we take the average of those individual errors, and show the average error against the computational cost in Fig. 3.3. The error decreases with increasing cost, until it becomes comparable to the confidence bounds. At that point, the error is dominated by the time averaging error, and no longer due to the spatial discretization. Since the computational costs for simulations of 300 dimensionless time units are already considerably large we did not pursue a further reduction in the time averaging error. In fact, such time averaging error will tend to zero at a rate inversely proportional to the square root of the simulation time. Hence, only at extreme costs one could perceive a significant reduction of the time averaging error. Such resources are not available for this study and are also not required to establish the main conclusions.

In the periodic domain, AFID was found to be much faster than NEK5000. At a given computational cost, a much higher number of grid points can be afforded with AFID. On the other hand, this significant difference in computational cost between NEK5000 and AFID when counting the number of grid points only, is considerably reduced when counting the actually achieved level of precision of the Nu prediction. Clearly, the higher-order method used in NEK5000 is beneficial at reducing the gap with AFID in error versus cost considerations. This is illustrated concisely in Fig. 3.3. For the cubic container we observe a similar situation: the specialized code GOLDFISH is much faster than NEK5000, see Fig. 3.2b. And again, the efficiency of NEK5000 becomes closer to that by GOLDFISH when the convergence of Nu is taken into account. When the error of the Nu calculation is above the confidence bound, GOLDFISH calculates Nu up to tenfold more accu-

Figure 3.3: Average error of Nu against the computational cost for different geometries: (a) Periodic domain (b) Cubic domain (c) Cylindrical domain. $\times$ AFID/RBFLOW, $\diamond$ GOLDFISH, $\square$ NEK5000, $\circ$ OPENFOAM. The dashed line indicates the 95% confidence level of the reference value given in Tables 3.1, 3.2, and 3.3.

rate than NEK5000, for given computational costs. In the cylindrical container, RBFLOW and GOLDFISH are up to a factor ten faster than OPENFOAM for a given level of accuracy, while RBFLOW and OPENFOAM are both of the second order. This illustrates the penalty that comes with the use of a general purpose code compared to a dedicated specialized code. NEK5000 falls roughly in between RBFLOW and OPENFOAM. Overall, Fig. 3.3 shows that the large differences in speed that appear in Fig. 3.2 (with AFID/RBFLOW the most efficient when it comes to costs of simulation with a certain number of grid points) decrease when counting the error in Nu versus computational costs due to the usage of higher-order schemes in the other codes.

In Fig. 3.3 one can also see that the accuracy of the general purpose codes behave non-monotonically with increasing computational time. The strong oscillations in the behavior of the Nu -error versus the costs are explained not only by the restricted time of statistical averaging, but mainly by the usage of the iterative solvers within these codes. While the dedicated codes AFID/RBFLOW and GOLDFISH use direct solvers and get the corresponding solutions at the machine accuracy, the iterative solvers of the general purpose codes stop iterations with a certain residual errors. Furthermore, the used iterative solvers generally do not guarantee a monotonic reduction of the errors with increasing number of conducted iterations. This holds in particular for the generalized minimal residual method used in NEK5000. This makes prediction of the accuracy versus computation costs for general purpose codes less trivial and somewhat uncertain.

3.4.2 Robustness against under-resolution

At $Ra = 10^8$, we are able to compute an accurate reference solution that is converged with respect to the spatial resolution independent of which code was adopted, (see Fig. 3.1). At higher Ra numbers, the computation of such a reference

solution for all codes becomes too expensive, however. As an alternative, we can compare the codes in a different, somewhat more qualitative, way by increasing the Ra number while keeping the spatial resolution fixed. In this case, we use the meshes with the highest resolution adopted for $Ra = 10^8$ in the cylindrical container. As Ra increases, the effect of insufficient resolution will unavoidably show up sooner or later, which indicates the robustness of the codes, against under-resolution. Nu , compensated with $Ra^{-1/3}$, is plotted against Ra in Fig. 3.4. Initially, the results of the three codes at $Ra = 10^8$ are all very close to each other with the differences between the results less than 0.5%. The values of Nu are also very close to the prediction by the GL theory with the deviations between the simulation results and the GL predictions less than 1%. As Ra increases the different robustnesses of the various codes against deliberate under-resolution become apparent. NEK5000 shows the smallest deviation from the theoretical scaling of Nu , and RBFLOW the largest. However, we emphasize that this robustness of Nu for NEK5000 against deliberate underresolution does not imply that other flow features would still be well represented. E.g., in Fig. 3.5 we show temperature snapshots for well resolved and deliberately underresolved simulations with NEK5000. The latter clearly show a pronounced imprint of the computational grid even though the Nu number looks converged. Similar plots obtained from RBFLOW show a similar inadequacy of the grid resolution in the high-end of Ra that, however, is consistent with the deviation for Nu from the expected values.

This comparison clearly indicates that the agreement of Nu with the theoretical prediction (and among the values obtained from the various definitions) is not enough to assess the adequacy of the spatial resolution of the numerical simulation. Additional quantities have to be analyzed, such as the instantaneous temperature snapshots or rms profiles, in order to clarify this issue.

3.5 Conclusions and outlook

In this paper, we have compared several codes for the simulation of turbulent RB convection in a number of typical geometries. Particular attention has been given to the heat transport in the turbulent flow, which is quantified by Nu . The computational efficiency of the codes is determined with reference to fully converged simulations at relatively high spatial resolutions. We observed significant differences between the codes in terms of computational costs, i.e. the specialized AFID/RBFLOW and GOLDFISH code clearly outperform NEK5000 and OPENFOAM. Thus, we note that a considerable saving in computational costs can be achieved by employing an optimized code for a simple geometry compared to general purpose codes designed for complex geometries. The benefit of general purpose codes like NEK5000 and OPENFOAM is of course that they are much wider applicable than specialized codes, which need to be specifically tuned per case.

The usage of unstructured grids in the general purpose codes requires itera-

Figure 3.4: Nu versus Ra in cylindrical container of $\Gamma = 1$. These results of deliberately under-resolved DNS are obtained at a fixed computational mesh, which is too coarse for large Ra . $+$ Nu_{vol} (RBFLOW), $\circ$ Nu_{pl} (RBFLOW), $*$ Nu_{vol} (GOLDFISH), $\square$ Nu_{pl} (GOLDFISH), $\times$ Nu_{vol} (NEK5000), $\diamond$ Nu_{pl} (NEK5000), $---$ Grossmann-Lohse theory [164]. For RBFLOW and GOLDFISH the Nu number gets too high when the simulation is underresolved. In Fig. 3.5 we show the temperature field obtained in underresolved NEK5000 simulation showing that even when the Nu is close to the expected value that the temperature field shows a clear imprint of the used computational grid. We thus stress that the robustness of Nu against deliberate underresolution does not imply the robustness of other flow features.

tive solutions of the governing equations on each time step. This leads to higher computational costs and makes these codes less predictable with respect to the accuracy of the calculation of Nu with growing computational costs. Also the scalability of the OPENFOAM codes on supercomputers leaves much to be desired. All this leads to the fact that AFID/RBFLOW, being also the second-order as OPENFOAM in the considered configuration, is at least ten times faster than OPENFOAM, while providing the same level of accuracy. Therefore we conclude that OPENFOAM, at least in the analyzed configuration, which is the most popular in engineering, is not optimal for scientific investigations of high Ra number thermal convection.

Also, among the other codes, AFID/RBFLOW is clearly the fastest one. It is up to tenfold faster than GOLDFISH and up to hundredfold faster than NEK5000. However, when the accuracy of the Nu calculation is taken into account, the efficiency of GOLDFISH and AFID/RBFLOW is similar, while NEK5000 and OPENFOAM are up to 10 times slower. When in a certain numerical study the point of interest is an integral quantity (zero moment), like Nu or Reynolds number, or when the profiles of the mean temperature or velocity are aimed to be studied (first moments), the advantages of the usage of the second-order code AFID/RBFLOW

Figure 3.5: Temperature field at $z/H = 0.0152$ from the NEK5000 code (top row) and RBFLOW (bottom row) at (from left to right) $Ra = 10^8$, $Ra = 10^9$ and $Ra = 10^{10}$. For the latter two Ra values the chosen grid resolution is insufficient. Note that in the NEK5000 snapshots the imprint of the computational grid is clearly visible in the higher Ra number cases, even though the Nu number from the simulations looks reasonable, see Fig. 3.4. The ripples in the RBFLOW snapshots are observed near sharp gradients when the resolution is insufficient.

are clearly pronounced. It is extremely fast and calculates these quantities precisely on sufficiently fine meshes. This is partly thanks to the implementation of AFID/RBFLOW, which is highly optimized, and scales excellently on large number of cores [190].

Finally, we give a general estimate of the complexity of the DNS of turbulent RB convection in the classical regime and in the ultimate regime, which is to be studied in the future. As we already mentioned in section 3.4.1, apart from the CFL-condition and the von Neumann stability, there exists also another restriction on the time stepping in accurate DNS, which is the resolution of the Kolmogorov time microscales. Note that the Kolmogorov microscale in space, $\eta \equiv (\mu^3/\langle \varepsilon \rangle_V)^{1/4}$, and the microscale in time, $\eta_\tau \equiv (\mu/\langle \varepsilon \rangle_V)^{1/2}$, are related as $\mu \eta_\tau \sim \eta^2$ with $\mu \equiv \sqrt{\text{Pr}/Ra}$. Thus, the optimal (not over-resolved but accurate) DNS, which resolve both, the Kolmogorov time microscale η_τ and the Kolmogorov spatial microscales η , will lead to the scaling of the computational costs with the grid size N at least as $\mathcal{O}(\mu N^{5/3})$. Since $\langle \varepsilon \rangle_V = (Nu - 1)/\sqrt{\text{Pr}Ra}$, for a fixed Pr , the computational costs in accurate DNS must grow at least as $\mathcal{O}(Nu^{5/4} Ra^{3/4})$. Therefore, in the classical regime, where $Nu \sim Ra^{1/3}$, the cost will increase with Ra at least as $\mathcal{O}(Ra^{7/6})$, while for the ultimate regime, where the scaling $Nu \sim Ra^{1/2}$ is expected, the anticipated computational costs in accurate DNS are at least

$\mathcal{O}(Ra^{11/8})$.

Before concluding this paper we wish to point once more out that comparing Nu obtained by the numerical simulation with the expected value is not a reliable criterion to assess its validity. In fact, we showed that deliberately under-resolved simulations performed with higher order codes show a small error in Nu while producing temperature fields with strong unphysical oscillations. Instantaneous snapshots of temperature and profiles of higher order moments have to be evaluated, together with Nu , in order to establish the quality of a numerical simulation.

Acknowledgements

GLK is funded by Foundation for Fundamental Research on Matter (FOM), part of the Netherlands Organization for Scientific Research (NWO). DL and RV are funded by the Netherlands Center for Multiscale Catalytic Energy Conversion (MCEC), an NWO Gravitation program funded by the Ministry of Education, Culture and Science of the government of the Netherlands. OS and SH are funded by the Deutsche Forschungsgemeinschaft (DFG) under the grants Sh 405/4-2 and Ho 5890/1-1, respectively. We thank NWO for granting us computational time on Cartesius cluster from the Dutch Supercomputing Consortium SURFsara under grants SH-061 and SH-015 and also are grateful to the Leibniz Rechenzentrum (LRZ) for providing us computational resources under the grant pr84pu.

A block Krylov subspace implementation of the time-parallel Paraexp method and its extension for nonlinear partial differential equations*

A parallel time integration method for nonlinear partial differential equations is proposed. It is based on a new implementation of the Paraexp method for linear partial differential equations (PDEs) employing a block Krylov subspace method. For nonlinear PDEs the algorithm is based on our Paraexp implementation within a waveform relaxation. The initial value problem is solved iteratively on a complete time interval. Nonlinear terms are treated as a source term, provided by the solution from the previous iteration. At each iteration, the problem is decoupled into independent subproblems by the principle of superposition. The decoupled subproblems are solved fast by exponential integration, based on a block Krylov method. The new time integration is demonstrated for the one-dimensional advection-diffusion equation and the viscous Burgers equation. Numerical experiments confirm excellent parallel scaling for the linear advection-diffusion problem, and good scaling in case the nonlinear Burgers equation is simulated.

4.1 Introduction

Recent developments in parallel computing urge the design of new numerical methods, as well as a major revision of existing numerical algorithms [35]. To be efficient on massively parallel platforms, the algorithms need to employ all possible means to parallelize the computations. When solving partial differential equations (PDEs) with a time-dependent solution, an important way to parallelize the

*Based on: G.L. Kooij, M.A. Botchev, and B.J. Geurts. *A block Krylov subspace implementation of the time-parallel Paraexp method and its extension for nonlinear partial differential equations*. J. Comput. Appl. Math. **316**, 229–246, 2017.

computations is, next to the parallelization across space, parallelization across time. This adds a new dimension of parallelism with which the simulations can be implemented. In this chapter we present a new time-parallel integration method extending the Paraexp method [54] to nonlinear partial differential equations using Krylov methods and waveform relaxation.

Several approaches to parallelize the simulation of time-dependent solutions in time can be distinguished. The first important class of the methods are the waveform relaxation methods [14, 122, 132, 185], including the space-time multigrid methods for parabolic PDEs [20, 71, 82, 117]. The key idea is to start with an approximation to the numerical solution for the whole time interval of interest and update the solution, solving an easier-to-solve approximate system in time. The Parareal method [112], which attracted significant attention recently, is a prime example related to the class of waveform relaxation methods [58].

Parallel Runge–Kutta methods and general linear methods, where the parallelism is determined and restricted by the number of stages or steps, form another class of the time-parallel methods [20, 32, 173]. Time-parallel schemes can also be obtained by treating the time as an additional space dimension and solving for values at all time steps at once [34, 189]. This approach requires significantly more memory and is used, e.g., in multidimensional tensor computations and (discontinuous) Galerkin finite element methods [22, 83, 111, 175]. Recently a “parallel full approximation scheme in space and time” (PFASST) was introduced, which is based on multigrid methods [46]. PFASST was observed to have a generally improved parallel efficiency compared to Parareal. Also, we mention parallel methods that facilitate parallelism by replacing the exact solves in implicit schemes by approximate ones [16], and parallel methods based on operator splitting [12]. Finally, there is the Paraexp method [54] for parallel integration of linear initial-value problems. This algorithm is based on the fact that linear initial-value problems can be decomposed into homogeneous and nonhomogeneous subproblems. The homogeneous subproblems can be solved fast with an exponential integrator, while the nonhomogeneous subproblems can be solved with a traditional time-stepping method.

In this chapter we propose a time parallel method which is based on the matrix exponential time stepping technique and waveform relaxation. Our method adopts the Paraexp method within a waveform relaxation framework, which enables the extension of parallel time integration to nonlinear PDEs. The method is inspired by and relies on recent techniques in Krylov subspace matrix exponential methods such as shift-and-invert acceleration [63, 127, 170], restarting [1, 23, 38, 133, 167] and using block Krylov subspaces [11] to handle nonautonomous PDEs efficiently. The method also relies on a singular value decomposition (SVD) of source terms in the PDE, which is used to construct the block Krylov subspace. To improve the efficiency of the method, the SVD is truncated by retaining only the relatively large singular values. We show in theory and in practice that for a source term, that can be approximated by a smooth function, the singular values decay algebraically with time interval size. In that case, a truncated SVD approximation of the source

term is adequate.

The contribution of this chapter is more specifically as follows. First, to solve systems of linear ODEs, we propose an implementation of the Paraexp method, based on the exponential block Krylov (EBK) method, introduced in [11]. The EBK method is a Krylov method that approximates the exact solution of a system of nonhomogeneous linear ordinary differential equations by a projection onto a block Krylov subspace. Our Paraexp implementation does not involve a splitting of homogeneous and nonhomogeneous subproblems, which leads to a better parallel efficiency. Second, we extend our EBK-based implementation of the Paraexp method to nonlinear initial-value problems. We solve the problem iteratively on a chosen time interval. In our case, the nonlinear term of the PDE is incorporated as a source term, which is updated after every iteration with the latest solution. Third, we show that our Paraexp-EBK (PEBK) implementation has promising properties for time-parallelization. The PEBK method can be seen as a special continuous-time waveform relaxation method, which is typically more efficient than standard waveform relaxation methods [14]. The PEBK method is tested for the one-dimensional advection-diffusion equation and the viscous Burgers equation in an extensive series of numerical experiments. Since we are interested in time-parallel solvers for large-scale applications in computational fluid dynamics (CFD), such as turbulent flow simulations, we also test our method with respect to the diffusion/advection ratio and grid resolution, reflecting high Reynolds-number conditions. For example, the parallel efficiency of the Parareal algorithm was found to decrease considerably with increasing number of processors for the advection equation [52, 144, 160]. In contrast, the EBK method was found to have excellent weak scaling for linear problems.

The chapter is organized as follows. In Section 4.2 we describe the exponential time integration scheme and its parallelization. In Section 4.3, we present numerical experiments with the advection-diffusion equation, and with the viscous Burgers equation in Section 4.4. Finally, a discussion and conclusions are outlined in Section 4.5.

4.2 Exponential time-integration

Our time integration method for linear and nonlinear PDEs is based on the exponential block Krylov (EBK) method [11]. In this section, we first provide a brief description of the EBK method. Then, we extend the EBK method to integrate nonlinear PDEs in an iterative way. Finally, the parallelization of the time integrator is discussed.

4.2.1 Exponential block Krylov method

The EBK method is a time integrator for linear systems of nonhomogeneous ordinary differential equations (ODEs). Details of the method are given in [11]. We

follow the *method of lines* approach [177], i.e., the PDE is discretized in space first. We start with linear PDEs. After applying a spatial discretization to the PDE, we obtain the initial value problem,

$$\begin{aligned}\mathbf{u}'(t) &= A\mathbf{u}(t) + \mathbf{g}(t), \\ \mathbf{u}(0) &= \mathbf{u}_0,\end{aligned}\tag{4.1}$$

where $\mathbf{u}(t)$ is a vector function $\mathbf{u}(t) : \mathbb{R} \rightarrow \mathbb{R}^n$, $A \in \mathbb{R}^{n \times n}$ is a square matrix, and $\mathbf{g}(t) \in \mathbb{R}^n$ a time-dependent source term. The vector $\mathbf{u}$ can readily be identified with the solution to the PDE in terms of its values taken on a computational grid in physical space. In Section 4.2.3, we will introduce a more general term $\mathbf{g}(t, \mathbf{u}(t))$, which contains the nonlinear terms of the PDE. The matrix A is typically large and sparse, depending on the spatial discretization method. The dimension of the system, n , corresponds to the number of degrees of freedom used in the spatial discretization. Exponential integrators approximate the exact solution of the semi-discrete system (4.1), formulated in terms of the matrix exponential.

To treat general source terms, $\mathbf{g}(t)$ is approximated with a piecewise polynomial function. The polynomial approximation of $\mathbf{g}(t)$ is based on a truncated singular value decomposition (SVD) [15], which improves the efficiency of the EBK method. The source term is sampled at s points in time, $0 = t_1 < t_2 < \dots < t_{s-1} < t_s = \Delta T$, in the integration interval $[0, \Delta T]$, and the source samples form the matrix

$$G = [\mathbf{g}(t_1) \quad \mathbf{g}(t_2) \quad \dots \quad \mathbf{g}(t_s)] \in \mathbb{R}^{n \times s}.\tag{4.2}$$

We make a natural assumption that the typical number of time samples s , necessary for a good approximation of $\mathbf{g}(t)$, is much lower than n : $s \ll n$. Since we assume $s \ll n$, it is more economical to calculate the so-called thin SVD of G [64], instead of the full SVD, without loss of accuracy. To obtain the thin SVD of G , we first compute its thin QR factorization, $G = QR$, and then compute the SVD of R . In MATLAB the thin SVD can be calculated with `svd(G,0)`. In this case, the thin SVD of G is

$$G = \tilde{U} \tilde{\Sigma} \tilde{V}^T,\tag{4.3}$$

where $\tilde{\Sigma} \in \mathbb{R}^{s \times s}$ is a diagonal matrix containing the singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_s$, and $\tilde{U} \in \mathbb{R}^{n \times s}$, $\tilde{V} \in \mathbb{R}^{s \times s}$ are matrices with orthonormal columns. The thin SVD can be approximated by a truncation in which the $m < s$ largest singular values are retained. As seen from this truncated SVD, the samples of $\mathbf{g}(t)$ can be approximated by linear combinations of the first m columns of $\tilde{U}$, i.e.,

$$\mathbf{g}(t) \approx U \mathbf{p}(t).\tag{4.4}$$

where $U \in \mathbb{R}^{n \times m}$ is formed by the first m columns of $\tilde{U}$, $\mathbf{p}(t) \in \mathbb{R}^m$ is obtained by an interpolation of the coefficients in these linear combinations [15]. There are several possible choices for $\mathbf{p}(t)$, among which, cubic piecewise polynomials. Then, for a given ΔT , the approximation error in the source term, $\|\mathbf{g}(t) - U \mathbf{p}(t)\|$, can be

easily controlled within a desired tolerance, depending on the number of samples in $[0, \Delta T]$, and the number of singular values truncated (see [11]).

The number of retained singular values, m , is equal to the block width in the block Krylov subspace. The efficiency of the EBK method therefore depends on how many singular values are required for an accurate approximation of the source term (4.4). In applications of the method this parameter m can be varied and practical convergence can be assessed. If the subinterval ΔT is small and the source function $\mathbf{g}(t)$ can be well approximated by a smooth function, then the singular values of the samples of $\mathbf{g}$ decrease rapidly, thus allowing for an efficient low rank approximation in (4.4). Furthermore, let the subinterval length ΔT be small in the sense that $(\Delta T)^2 \ll \Delta T$, or, $\Delta T \ll 1$. Of course, the assumption that ΔT is small, is not always realistic and we comment on how it can be partially relaxed below in Remark 1.

Denote by $\mathbf{g}^{(j)}$ the j th derivative of $\mathbf{g}$ and assume that the constants

$$M_j := \max_{\xi \in [0, \Delta T]} \|g^{(j)}(\xi)\| \quad j = 0, \dots, n,$$

are bounded. Starting from a thin QR factorization of G , we can then show that the singular values of the samples decrease.

Consider the thin QR factorization of G , $G = QR$, where $Q \in \mathbb{R}^{n \times s}$ has orthonormal columns and $R \in \mathbb{R}^{s \times s}$ is an upper triangular matrix with nonnegative diagonal entries. As Theorem 1 in [76] states, the entries r_{jk} of R satisfy

$$|r_{jk}| \leq C_j M_{j-1} (\Delta T)^{j-1}, \quad k \geq j,$$

and $r_{jk} = 0$ for $k < j$ because R is upper triangular. In this estimate the constants C_j depend only on the points $t_1, \dots, t_s$ at which the samples are computed and do not depend on $\mathbf{g}$ and ΔT .

Since G and R have the same singular values, we consider now the singular values of R . According to [80, 3.1.3],

$$\sigma_{j+1} \leq \|R_j\|_2, \quad j = 1, \dots, s-1, \quad (4.5)$$

where R_j is a matrix obtained from R by skipping j rows or columns, chosen arbitrarily. Since the entries in R monotonically decrease rowwise as $|r_{jk}| = O(\Delta T)^{j-1}$, to have the sharpest estimate in (4.5), we choose R_j to be the matrix R with the first j rows skipped. To bound the 2-norm of R_j we use [79, Chapter 5.6]

$$\|R_j\|_2 \leq \sqrt{\|R_j\|_1 \|R_j\|_\infty}$$

and note that for $j = 1, \dots, s-1$

$$\|R_j\|_\infty \leq (s-j) C_j M_{j-1} (\Delta T)^{j-1}, \quad \|R_j\|_1 = O(\Delta T)^{j-1}.$$

Thus, we obtain $\|R_j\|_2 = \sqrt{s-j} O(\Delta T)^{j-1}$, which, together with (4.5), yields the following result.

Theorem 1 *Let $\mathbf{g}(t) : \mathbb{R} \rightarrow \mathbb{R}^n$ be a smooth function such that the constants*

$$M_j := \max_{t \in [0, \Delta T]} \|\mathbf{g}^{(j)}(t)\| \quad j = 0, \dots, n,$$

are bounded. Furthermore, let the subinterval length ΔT be small in the sense that $(\Delta T)^2 \ll \Delta T$. Then for the singular values σ_j , $j = 1, \dots, s$, of the sample matrix

$$G = [\mathbf{g}(t_1) \quad \mathbf{g}(t_2) \quad \dots \quad \mathbf{g}(t_s)] \in \mathbb{R}^{n \times s}$$

holds

$$\sigma_{j+1} = \sqrt{s-j} O(\Delta T)^j, \quad j = 1, \dots, s-1. \quad (4.6)$$

Remark 1 *We may extend the result of Theorem 1 to the case of a large ΔT if the constants M_j are bounded more strongly or even decay with j . Analytic functions for example are infinitely differentiable, which means the M_j are bounded. Indeed, if ΔT is large, we can consider the function $\tilde{\mathbf{g}}(\tilde{\xi}) := \mathbf{g}(t + \xi q)$, with q chosen such that $\tau := \Delta T/q$ is small. As the function $\tilde{\mathbf{g}}$ takes the same values for $\tilde{\xi} \in [0, \tau]$ as the function $\mathbf{g}$ for $\xi \in [t, t + \Delta T]$, Theorem 1 formally holds for $\tilde{\mathbf{g}}$ with ΔT replaced by τ and M_j multiplied by q^j . The coefficients in the O symbol in (4.6) may now grow with the powers of ΔT , thus rendering the result meaningless unless a stricter assumption on M_j is made.*

The truncated SVD approximation of the source term (4.4) facilitates the solution of the initial value problem (IVP) in (4.1) by the block Krylov subspace method [11]. Here, the block Krylov subspace at iteration l is defined as

$$\mathcal{K}_l(A, U) := \text{span} \{U, AU, A^2U, \dots, A^{l-1}U\}. \quad (4.7)$$

Furthermore, the block Krylov subspace method can be accelerated by the shift-and-invert technique [63, 127, 170] and, to keep the Krylov subspace dimension bounded, implemented with restarting [13, 23].

4.2.2 Parallelization of linear problems

In this section, the parallelization of the linear ODE solution is discussed. As we will see, the method is equivalent to the Paraexp method, but there are differences in implementation leading to a better parallel efficiency (see Section 4.3.2). Linear IVPs can be solved parallel in time by using the principle of superposition. First, the time interval is divided into P non-overlapping subintervals,

$$0 = T_0 < T_1 < \dots < T_P = T,$$

where P is also the number of processors that can be used in parallel. We introduce the shift of $\mathbf{u}(t)$ with respect to the initial condition, $\mathbf{u}(t) = \mathbf{u}_0 + \hat{\mathbf{u}}(t)$. The shifted variable $\hat{\mathbf{u}}(t)$ solves the IVP with homogeneous initial conditions,

$$\begin{aligned} \hat{\mathbf{u}}'(t) &= A\hat{\mathbf{u}}(t) + \hat{\mathbf{g}}(t), \quad t \in [0, T], \\ \hat{\mathbf{u}}(0) &= \mathbf{0}, \end{aligned} \quad (4.8)$$

Figure 4.1: Time samples of the source term on the subinterval $[T_j, T_{j+1}]$.

where

$$\widehat{\mathbf{g}}(t) := A\mathbf{u}_0 + \mathbf{g}(t). \quad (4.9)$$

The source term is approximated using s time samples per subinterval, as illustrated in Fig. 4.1. The shifted IVP (4.8) can be decoupled into independent subproblems by the principle of superposition. To each subinterval $[T_{j-1}, T_j]$ we associate a part of the source term $\widehat{\mathbf{g}}(t)$, defined for $j = 1, \dots, P$ as

$$\widehat{\mathbf{g}}_j(t) = \begin{cases} \widehat{\mathbf{g}}(t), & \text{for } T_{j-1} \leq t < T_j, \\ 0, & \text{otherwise.} \end{cases} \quad (4.10)$$

The expected parallel speedup is based on the observation that the solution to (4.8) is given by a variation-of-constant formula (see, e.g., [77]),

$$\widehat{\mathbf{u}}(T) = \int_0^T \exp((T-s)A) \widehat{\mathbf{g}}(s) ds = \sum_{j=1}^P \int_{T_{j-1}}^T \exp((T-s)A) \widehat{\mathbf{g}}_j(s) ds, \quad (4.11)$$

where the integrals on $[T_{j-1}, T]$ can be evaluated independently and in parallel. Here, we emphasize that the integration should be done for $[T_{j-1}, T]$ rather than for $[T_{j-1}, T_j]$ to account for the influence of the source term on the solution at later time. More precisely, by exploiting the linearity of the problem we decompose the IVP (4.8) into P independent subproblems,

$$\begin{aligned} \mathbf{v}'_j(t) &= A\mathbf{v}_j(t) + \widehat{\mathbf{g}}_j(t), \quad t \in [0, T], \\ \mathbf{v}_j(0) &= \mathbf{0}. \end{aligned} \quad (4.12)$$

where $\mathbf{v}_j(t)$ is referred to as a subsolution of the total problem. The subproblems are independent and can be solved in parallel. We integrate the subproblem individually with the EBK method. Observe that the source is only nonzero on the subinterval $[T_{j-1}, T_j]$. We follow the ideas of the Paraexp method [54] and note that the nonhomogeneous part of the ODE requires most of the computational work in the EBK method. An accurate SVD approximation of the source term (4.4) generally requires more singular values to be retained, increasing the dimensions of the block Krylov subspace (4.7). According to the principle of superposition, the solution of the original problem is then given as

$$\mathbf{u}(t) = \mathbf{u}_0 + \sum_{j=1}^P \mathbf{v}_j(t). \quad (4.13)$$

The summation of the subsolutions, $\mathbf{v}_j(t)$ is the only communication required between the parallel processes. The parallel algorithm is summarized in Fig. 4.3. In principle, this algorithm is identical to the Paraexp method [54]. The only practical difference is that in our implementation both the nonhomogeneous and the homogeneous part of the subproblems are solved by the EBK method. The original version of the Paraexp method assumes a conventional time integration method to solve the “difficult” nonhomogeneous part, and a Krylov subspace method for exponential integration of the homogeneous part. Our implementation of the Paraexp method is compared numerically with the original one in Section 4.3.2. The Paraexp method is essentially based on an overlapping time-decomposition method. An illustration for a simple ODE is shown in Fig. 4.2. The influence of the solution in a subinterval on the remaining subintervals is taken into account by exponential propagation of the subsolution, which can be computed very efficiently.

Figure 4.2: Overlapping time-decomposition of the solution of the differential equation $y'(t) = -y(t) + t$, with $y(0) = 0$. In this case the interval is divided into four subintervals. The solution in a subinterval propagates to later subintervals, illustrated by the dashed lines. The complete solution is the sum of the four subsolutions.

Finally, additional parallelism could be exploited within the block Krylov subspace method itself. If the block size is m , then the m matrix-vector products can be executed entirely in parallel, see [109, 136]. This approach could be applied in combination with the parallelization described in this section.

4.2.3 Treatment of nonlinearities

The EBK method, designed to solve a linear system of nonhomogeneous ODEs (4.1), can be extended to handle nonlinear systems of ODEs by including the nonlinearities in the source term. The system is then solved iteratively, in such a way that the iterand $\mathbf{u}_k(t)$ is updated on the entire interval $[0, T]$:

$$\mathbf{u}_k(t) \mapsto \mathbf{u}_{k+1}(t), \quad (4.14)$$

ALGORITHM PARALLEL TIME INTEGRATION.

Given: $A, \mathbf{u}_0, \mathbf{g}(t), \dots$

Solve: $\mathbf{u}'(t) = A\mathbf{u}(t) + \mathbf{g}(t), \quad \mathbf{u}(0) = \mathbf{u}_0.$

for $j = 1, \dots, P$ (in parallel)

 Solve nonhomogeneous part of the IVP (4.12), for $t \in [T_{j-1}, T_j]$.

 Solve homogeneous part of the IVP (4.12), for $t \in [T_j, T_P]$.

end for

Construct solution $\mathbf{u}(t)$, see (4.13).

Figure 4.3: The algorithm for solving linear ODE systems with the Paraexp exponential block Krylov (PEBK) method.

where k denotes the iteration index. We proceed in a few steps. First, the problem is reduced to the form of Eq. (4.1). The nonlinear IVP is therefore approximated by evaluating the source term with the current iterand, $\mathbf{u}_k(t)$. Because the current iterand $\mathbf{u}_k(t)$ is a known function, we can write the source term as an explicit function of time,

$$\mathbf{g}_k(t) := \mathbf{g}(t, \mathbf{u}_k(t)). \quad (4.15)$$

The resulting initial value problem is,

$$\begin{aligned} \mathbf{u}'_{k+1}(t) &= A\mathbf{u}_{k+1}(t) + \mathbf{g}_k(t), \\ \mathbf{u}_{k+1}(0) &= \mathbf{u}_0. \end{aligned} \quad (4.16)$$

This system is solved by the EBK method at each iteration until the solution has sufficiently converged. This approach is similar to applying Picard or fixed-point iterations [130] on the nonlinear term. We note that this approach, when we compute the low-rank approximation of Eq. (4.15) after every iteration, can be improved by using the “dynamical low-rank approximation” [96]. This would allow to reduce the computational work by reusing the low-rank approximations computed for the previous iterands. However, in the presented tests computing the low-rank approximation every time anew is cheap relatively to the other costs.

The convergence behaviour is improved by taking into account the Jacobian matrix, similar to a Newton–Raphson method. In this case, we introduce the average Jacobian matrix,

$$J_k := \frac{1}{T} \int_0^T \left[\frac{\partial \mathbf{g}_k}{\partial u_1} \dots \frac{\partial \mathbf{g}_k}{\partial u_n} \right] dt, \quad (4.17)$$

which is averaged over the interval of integration $[0, T]$, as the EBK method requires a constant matrix. The nonlinear remainder, with respect to the time-averaged state, is contained in the source term. Using this correction, we then

find the recursive relation,

$$\begin{aligned}\mathbf{u}'_{k+1}(t) &= [A + J_k]\mathbf{u}_{k+1}(t) + \mathbf{g}_k(t) - J_k\mathbf{u}_k(t), \\ \mathbf{u}_{k+1}(0) &= \mathbf{u}_0.\end{aligned}\tag{4.18}$$

When converged, $\mathbf{u}_{k+1}(t) = \mathbf{u}_k(t)$, the terms containing J_k eventually disappear. For simplicity, we present the algorithm for a fixed number of iterations K . The convergence can of course also be checked based on the nonlinear residual. We define the nonlinear residual as follows,

$$\mathbf{r}_{k+1}(t) := A\mathbf{u}_{k+1}(t) + \mathbf{g}_{k+1}(t) - \mathbf{u}'_{k+1}(t).\tag{4.19}$$

This expression can be rewritten as

$$\begin{aligned}\mathbf{r}_{k+1}(t) &= \underbrace{[A + J_k]\mathbf{u}_{k+1}(t) + \mathbf{g}_k(t) - J_k\mathbf{u}_k(t) - \mathbf{u}'_{k+1}(t)}_{\approx 0} + \dots \\ &\quad \dots + \mathbf{g}_{k+1}(t) - \mathbf{g}_k(t) - J_k(\mathbf{u}_{k+1}(t) - \mathbf{u}_k(t)), \\ &\approx \mathbf{g}_{k+1}(t) - \mathbf{g}_k(t) - J_k(\mathbf{u}_{k+1}(t) - \mathbf{u}_k(t)).\end{aligned}\tag{4.20}$$

Assuming that the nonlinear term is Lipschitz continuous, we find an upper bound of the nonlinear residual,

$$\|\mathbf{r}_{k+1}(t)\| \leq L\|\mathbf{u}_{k+1}(t) - \mathbf{u}_k(t)\|,\tag{4.21}$$

where L is a Lipschitz constant, accounting for the contributions due to the Jacobian and the dependence of $\mathbf{g}$ on $\mathbf{u}$ as in (4.15). This means that in practice the convergence can be controlled by setting a certain tolerance for $\|\mathbf{u}_{k+1}(t) - \mathbf{u}_k(t)\|$. In the context of waveform relaxation methods, it has been shown that the iterations converge faster, eventually superlinearly, for small ΔT , see [122]. That means convergence is guaranteed by taking ΔT small enough.

Remark 2 *The iteration (4.18) is well known in the literature on waveform relaxation methods [117, 122]. Its convergence is given, e.g., in [13, 130] and, in case of an inexact iteration, in [14]. These results, in particular, show that the iteration (4.18) converges superlinearly on finite time intervals if the norm of the matrix exponential of $t(A + J_k)$ decays exponentially with t , i.e., the eigenvalues of $A + J_k$ lie in the left half-plane. This is typically true for the heat equation and similar parabolic problems such as advection-diffusion problems.*

4.2.4 Parallelization of nonlinear problems

Nonlinear IVPs are solved in an iterative way with the PEBK method. We follow the *waveform relaxation* approach [14, 108, 130], that is, the problem is solved iteratively on the entire time interval of interest, $[0, T]$. The original problem is decomposed into a number of independent, “easier” subproblems on $[0, T]$ that can be solved iteratively, and in parallel.

At each iteration the nonlinear term is treated as source term, which gives us a linear system of ODEs that can be integrated in parallel, see Section 4.2.2. In the case of nonlinear problems, we take the average of the Jacobian matrix on each subinterval, which gives us the piecewise constant function for $j = 1, \dots, P$:

$$J_k(t) := \frac{1}{T_j - T_{j-1}} \int_{T_{j-1}}^{T_j} \left[\frac{\partial \mathbf{g}_k}{\partial u_1} \dots \frac{\partial \mathbf{g}_k}{\partial u_n} \right] dt, \quad t \in [T_{j-1}, T_j]. \quad (4.23)$$

The IVP is then shifted with respect to the initial condition,

$$\begin{aligned} \hat{\mathbf{u}}'_{k+1}(t) &= [A + J_k(t)]\hat{\mathbf{u}}_{k+1}(t) + \hat{\mathbf{g}}_k(t), \quad t \in [0, T], \\ \hat{\mathbf{u}}_{k+1}(0) &= \mathbf{0}, \end{aligned} \quad (4.24)$$

where,

$$\hat{\mathbf{g}}_k(t) := A\mathbf{u}_0 + \mathbf{g}_k(t) + J_k(t)[\mathbf{u}_0 - \hat{\mathbf{u}}_k(t)]. \quad (4.25)$$

The shifted IVP (4.24) can be then solved in parallel. We follow an approach similar to the Paraexp method, as explained in Section 4.2.2. Here, we define

$$\hat{\mathbf{g}}_{j,k}(t) := \begin{cases} \hat{\mathbf{g}}_k(t), & \text{for } T_{j-1} \leq t < T_j, \\ 0, & \text{otherwise.} \end{cases} \quad (4.26)$$

Now, the problem is decomposed into P independent subproblems. The subsolutions $\mathbf{v}_j$ result from the convergence of $\mathbf{v}_{j,k}$ satisfying:

$$\begin{aligned} \mathbf{v}'_{j,k+1}(t) &= [A + J_k(t)]\mathbf{v}_{j,k+1}(t) + \hat{\mathbf{g}}_{j,k}(t), \quad t \in [0, T], \\ \mathbf{v}_{j,k+1}(0) &= \mathbf{0}, \end{aligned} \quad (4.27)$$

which can be integrated with the EBK method. Here, we have introduced a second subindex k to indicate the j th subsolution at the k th iteration. The total solution can then be updated according to the principle of superposition,

$$\mathbf{u}_{k+1}(t) = \mathbf{u}_0 + \sum_{j=1}^P \mathbf{v}_{j,k+1}(t). \quad (4.28)$$

After each iteration, the solution is assembled, and the time-averaged Jacobian matrix is updated at each subinterval, according to Eq. (4.23). Note that for linear systems of ODEs, no iterations are required at all. The parallel algorithm is summarized in Fig. 4.6.

The parallel efficiency of the EBK method is achieved by incorporating the parallelization within the iterations that are required to solve nonlinear IVPs. The differences between the serial and the parallel algorithm are illustrated as flow diagrams in Figures 4.4 and 4.5.

We consider the parallel efficiency in an idealized setting, and estimate a theoretical upper bound here, assuming that communication among the processors can

Figure 4.4: Flow diagram of the serial EBK method for nonlinear PDEs. The algorithm stops after K iterations, after which the solution is assumed to be converged.

Figure 4.5: Flow diagram of the Paraexp-EBK method for nonlinear PDEs. The algorithm stops after K iterations, after which the solution is assumed to be converged.

be carried out very efficiently. The computational cost can then be simply estimated by the number of iterations required for the numerical solution to converge. Suppose the computation time of a single EBK iteration is τ_0 . The maximum parallel speedup is then,

$$\text{speedup} = \frac{K_1 \tau_0}{K_P \tau_0 / P} = \frac{K_1 P}{K_P}$$

where K_1 is number of iterations for serial time integration, and K_P for parallel time integration. The theoretical upper bound of the parallel efficiency is then,

$$\text{efficiency} = \frac{\text{speedup}}{P} = \frac{K_1}{K_P}. \quad (4.29)$$

High parallel efficiency can be achieved if the parallelization does not slow down the convergence of the numerical solution. As will be demonstrated in Section 4.4, we typically observe that K_P is not significantly larger than K_1 , and a near-optimal efficiency is achieved in various relevant cases. For comparison, the parallel efficiency of the Parareal algorithm is formally bounded by $1/K_P$ [125]. In our case, this upper bound is improved by a factor K_1 . Parareal is an iterative method for the parallelization of sequential time integration methods, whereas the EBK method, for nonlinear problems, is an iterative method to start with, and its parallelization does not necessarily increase the total number of iterations. Note the PFASST method [46] has a similar estimate of parallel efficiency as the PEBK method.

ALGORITHM PARALLEL TIME INTEGRATION.

Given: A , $\mathbf{u}_0$, $\mathbf{g}(\mathbf{t}, \mathbf{u}(\mathbf{t}))$, ...

Solve: $\mathbf{u}'(t) = A\mathbf{u}(t) + \mathbf{g}(t, \mathbf{u}(t))$, $\mathbf{u}(0) = \mathbf{u}_0$.

for $k = 1, \dots, K$

for $j = 1, \dots, P$ (in parallel)

 Calculate time-averaged Jacobian matrix $J_k(t)$, see (4.23).

 Solve nonhomogeneous part linearized ODE (4.27), for $t \in [T_{j-1}, T_j]$.

 Solve homogeneous part linearized ODE (4.27), for $t \in [T_j, T_P]$.

end for

 Update solution $\mathbf{u}_k(t)$ following (4.28).

end for

Figure 4.6: The algorithm for solving nonlinear problems with the Paraexp exponential block Krylov (PEBK) method, see also Fig. 4.5.

4.3 Advection-diffusion equation

In this section, we present results of numerical tests where the space-discretized advection-diffusion equation is solved with the EBK method. We demonstrate the consistency and stability of the EBK method for the linear advection-diffusion equation, which is a PDE describing a large variety of transport phenomena [84]. The spatial discretization, using central finite differencing for sake of illustration, of the PDE yields a linear system of ODEs. The time integration can be parallelized as described in Section 4.2.2. We illustrate the principle of parallelization for different physical parameters of the advection-diffusion equation, before we move on to nonlinear PDEs in Section 4.4. Convergence of the numerical solution is observed for different values of the physical parameters.

Here, we denote the relative tolerance of the PEBK solver by `tol`. In our tests, the block Krylov subspace is restarted every 20 iterations. For the truncated SVD approximation (4.4), $\mathbf{p}(t)$ are chosen to be piecewise cubic polynomials, although other types of approximations are possible as well, and are not crucial for the performance of the PEBK method. The sample points per subinterval are Chebyshev nodes. This is not crucial but gives a slightly better approximation than with uniform sample points.

4.3.1 Homogeneous PDE

We consider the advection-diffusion equation with a short pulse initial condition, to clearly distinguish the separate effects of advection and diffusion. The PDE and the periodic boundary conditions are as follows

$$\begin{aligned} u_t + au_x &= \nu u_{xx}, \quad x \in [0, 1], t \in [0, 1], \\ u(x, 0) &= \sin^{20}(\pi x), \\ u(0, t) &= u(1, t), \\ u_x(0, t) &= u_x(1, t), \end{aligned} \tag{4.30}$$

where $a \in \mathbb{R}$ is the advection velocity, and $\nu \in \mathbb{R}$ the diffusivity coefficient. Both parameters are constant in space and time. The PDE is first discretized in space with a second-order central finite difference scheme [128]. The corresponding semi-discrete system of ODEs is then

$$\begin{aligned} \mathbf{v}'(t) &= A\mathbf{v}(t) + A\mathbf{u}_0, \\ \mathbf{v}(0) &= \mathbf{0}, \end{aligned} \tag{4.31}$$

where the matrix A results from the discretization of the spatial derivatives, and represents both the diffusive and the advective term. In (4.31), the substitution $\mathbf{u} = \mathbf{v} + \mathbf{u}_0$ has been applied, with $\mathbf{u}(t)$ being the vector function containing the values of the numerical solution on the mesh at time t , with $\mathbf{u}_0 = \mathbf{u}(0)$. The substitution leads to homogeneous initial conditions. In this case, the source

term is constant in time, and its SVD polynomial approximation (4.4) is exact with $m = 1$. This allows us to focus on the two remaining parameters: the grid resolution and the tolerance of the EBK solver.

The linear IVP (4.31) is decoupled into independent subproblems by partitioning of the source term ($A\mathbf{u}_0$) on the time interval. The superposition of the subsolutions is illustrated in Fig. 4.7, in which the time interval of interest, $[0, 1]$, has been partitioned into four equal subintervals, to be integrated on four processors. The sum of the initial condition and the subsolutions gives the final solution on the entire time interval, see Section 4.2.2.

In the following numerical experiments, $P = 8$ subintervals are used. The advection-diffusion equation is solved for three different combinations of a and ν , see Fig. 4.8. The solutions are computed for different `tol` and Δx . The discretization error is controlled by the time integration with `tol`, and by the spatial discretization with Δx . The error of the numerical solution is measured in the relative ℓ^2 -norm,

$$\|\mathbf{u}_h(T) - \mathbf{u}(T)\| / \|\mathbf{u}(T)\|, \quad (4.32)$$

where the exact solution $\mathbf{u}(T)$ is on the mesh, i.e., it has the entries $u_j(T) = u(x_j, T)$. The exact solution of Eq. (4.30) is

$$u(x, t) = \sum_{n=0}^{20} a_n \cos(n\pi(x - at)) \exp(-(n\pi)^2 \nu t), \quad (4.33)$$

with coefficients

$$a_0 = \frac{1}{2} \int_{-1}^1 \sin^{20}(\pi x) dx, \quad (4.34)$$

$$a_n = \int_{-1}^1 \sin^{20}(\pi x) \cos(n\pi x) dx, \quad (4.35)$$

which can be calculated either analytically or numerically. We note that the coefficients in Eqs. (4.34) and (4.35) are the standard Fourier coefficients for the interval $[0, 1]$, taking into account that $\sin^{20}(\pi x)$ is even. The error in (4.32) shows second-order convergence with Δx , as expected from the truncation error of the finite difference scheme. The convergence plots show that we are able to control the error of the parallel time integration method. In this case, the final error can be made to depend only on the spatial resolution and the tolerance of the EBK method. Also, the EBK method has no principal restrictions on the timestep size, as it directly approximates the exact solution of Eq. (4.31) by Krylov subspace projections. Also, the number of Krylov iterations does not increase with ΔT , because the EBK method with the shift-and-invert technique emphasizes the eigenvalues of small magnitude [11]. The slow dynamics of the solution are associated to those small eigenvalues.

Figure 4.7: Superposition of subsolutions to the advection-diffusion equation, with partitioning $\mathbf{T} = \{0, 0.25, 0.5, 0.75, 1\}$, $\Delta x = 5 \cdot 10^{-3}$, $a = 1$, and $\nu = 10^{-2}$. The color depends on the time coordinate: blue corresponds to $t = 0$ and green to $t = 1$.

(a) Velocity $a = 0$ and diffusivity coefficient $\nu = 10^{-2}$. (b) Velocity $a = 1$ and diffusivity coefficient $\nu = 0$. (c) Velocity $a = 1$ and diffusivity coefficient $\nu = 10^{-2}$.

Figure 4.8: Left: numerical solution on the finest mesh at $t = 0$ (dashed) and $t = 0.25$ (solid). Note that the solution for $a = 1$ and $\nu = 0$ does not show odd-even decoupling because of the high spatial resolution used. Right: convergence of the numerical solution (at final time $t = 1$), for tol : $\square$ 10^{-2} , $\circ$ 10^{-4} , $\times$ 10^{-6} .

4.3.2 Parallel efficiency

In the previous examples, we solved homogeneous IVPs. Nonhomogeneous problems are generally more expensive to solve, because an accurate SVD approximation of the source term (4.4) requires more singular values, which increases the block width of the block Krylov subspace (4.7). Parallel speedup can then be expected by splitting the nonhomogeneous problem into subproblems (4.12), which require less individual effort to solve by the PEBK method. To measure the parallel efficiency of our algorithm for nonhomogeneous IVPs, we introduce a source term $f(x, t)$ in the advection-diffusion equation

$$u_t + au_x = \nu u_{xx} + f(x, t), \quad (4.36)$$

with $\nu = 10^{-2}$, and $a = 1$. The source term is chosen such that the solution is a series of five travelling pulses

$$u(x, t) = \frac{1}{2} - \frac{1}{2} \cos(10\pi(x - t)). \quad (4.37)$$

The mesh width of the spatial discretization is $\Delta x = 10^{-3}$, such that the error due to the spatial discretization is small compared to the time integration error. The mesh width gives a semi-discrete system with a 1000×1000 matrix, which is a suitable problem size for testing the EBK method. The tolerance of the EBK method is set to 10^{-4} . The SVD polynomial approximation is constructed from 100 time samples per subinterval. The singular values are plotted in Fig. 4.10, which reveals that the first two singular values are several orders larger than the rest. Therefore, we retain only the first two singular values in the truncated SVD. The decay of singular values is guaranteed by the upper bound from Theorem 1. We have verified that the SVD approximation is sufficiently accurate, i.e., the approximation error, measured in the L^2 -norm, is less than the tolerance of the EBK method.

We compare the parallel efficiency of the Paraexp-EBK implementation with a conventional implementation of the Paraexp algorithm [54], where the nonhomogeneous problem is integrated with the Crank–Nicolson (CN) scheme. The linear system is solved directly using MATLAB. The matrix exponential propagator, for the homogeneous problem in the Paraexp method, is realized with an Arnoldi method using the shift-and-invert technique [170].

In order to have a Courant number of one, we take $\Delta t = 10^{-3}$ (for $\Delta x = 10^{-3}$ and $a = 1$). According to the Paraexp method, the time step size needs to be decreased in parallel computation by a factor $P^{1/2q}$ [54], where q is the order of the time integration method, in order to control the error. In case of the Crank–Nicolson scheme, we have $q = 2$.

As discussed in Section 4.2.2, there is no communication required between processors, except for the superposition of the solutions to the subproblems at the end. The updated source term and Jacobian matrix can be calculated with the new solution directly, so no additional communication between all processors is needed. We can estimate the possible parallel performance of the algorithm on

a serial computer by measuring the computation time of each independent subproblem, as done for example in [54]. Such an estimate provides a first indication for the parallel speedup. The computation time of the serial time integration is denoted τ_0 . For the parallel time integration, we measure the computation times of the nonhomogeneous and the homogeneous part of the subproblems separately, denoted by τ_1 and τ_2 respectively. The parallel speedup can then be estimated as

$$\text{speedup} = \frac{\tau_0}{\max(\tau_1) + \max(\tau_2)}, \quad (4.38)$$

where we take the maximum value of τ_1 and τ_2 over all parallel processes. The timings of the EBK method and Paraexp are listed in Table 4.1, which show that the runtimes of the parallel EBK method are generally shorter than the Paraexp/Crank–Nicolson combination.

The parallel efficiency of both methods is illustrated in Figure 4.9. Note that the parallel efficiency of the standard Paraexp method steadily decreases, whereas the PEBK method maintains a constant efficiency level around 90%. The decrease in efficiency of the standard Paraexp method is due to the reduced time step size in the Crank–Nicolson scheme, with respect to its serial implementation. The nonhomogeneous part of the subproblems requires more computation time as the number of processors increases.

The numerical test confirms the initial assumption that parallel speedup with the EBK method can be achieved by decomposing the original problem into simpler independent subproblems (4.12). Furthermore, the Paraexp implementation using the EBK method, appears to be more efficient than one using a traditional time-stepping method.

Figure 4.9: Parallel efficiency for solving the advection-diffusion equation. $\square$ PEBK; $\times$ Paraexp/CN.

Figure 4.10: Singular values of the matrix composed of the source term samples.

Table 4.1: Parallel efficiency of the Paraxp-EBK method and the Paraexp/Crank–Nicolson method for the advection-diffusion equation, with number of processors P . Timing τ_0 corresponds to the serial algorithm. For the parallel algorithm, τ_1 and τ_2 are timings of the nonhomogeneous and homogeneous problem respectively.

	P	Serial		Parallel			Eff.
		τ_0	Error	$\max(\tau_1)$	$\max(\tau_2)$	Error	
PEBK	2	1.43e+00	3.05e-04	7.22e-01	6.44e-02	2.70e-04	91 %
	4	2.81e+00	3.05e-04	7.23e-01	7.06e-02	2.68e-04	89 %
	8	5.59e+00	3.05e-04	7.40e-01	6.21e-02	2.88e-04	87 %
	16	1.14e+01	3.05e-04	7.30e-01	6.28e-02	3.22e-04	90 %
	32	2.27e+01	3.05e-04	7.33e-01	6.63e-02	3.65e-04	89 %
Paraexp	2	2.15e+00	4.56e-04	1.29e+00	1.66e-02	4.10e-04	82 %
	4	4.37e+00	4.56e-04	1.52e+00	1.28e-02	3.80e-04	71 %
	8	8.56e+00	4.56e-04	1.85e+00	1.24e-02	3.59e-04	58 %
	16	1.70e+01	4.56e-04	2.18e+00	1.31e-02	3.43e-04	49 %
	32	3.43e+01	4.56e-04	2.62e+00	1.16e-02	3.32e-04	41 %

4.4 Burgers equation

In the previous section, the PEBK method was applied to a linear PDE. In this section, the performance of the PEBK method is tested on a nonlinear PDE, the viscous Burgers equation. In all tests $K = 10$ nonlinear iterations are carried out.

4.4.1 Travelling wave

Consider the viscous Burgers equation,

$$u_t + uu_x = \nu u_{xx} + f(x, t), \quad x \in [0, 1], t \in [0, 1], \quad (4.39)$$

where $\nu \in \mathbb{R}$ denotes the diffusivity (or viscosity) coefficient. In the following experiments, we take $\nu = 10^{-2}$, such that the problem is dominated by the nonlinear convective term. As in the previous example, the boundary conditions are periodic. Exact solutions to the Burgers equation can be found by the Cole–Hopf transformation [78]. In this test case, we construct a desired solution by introducing an appropriate source term, as shown for the advection-diffusion equation in Section 4.3.2. The source term balances the dissipation of energy due to diffusion, and prevents the solution of vanishing in the limit $t \rightarrow \infty$.

The Burgers equation is an important equation in fluid dynamics. It can be seen as a simplification of the Navier–Stokes equation, which retains the nonlinear convective term, uu_x . In the limit $\nu \rightarrow 0$, the nonlinearity may produce discontinuous solutions, or shocks, so that a typical solution may resemble a sawtooth wave. A sawtooth wave can be represented by an infinite Fourier series. A smooth

version of the sawtooth wave can be obtained by the modified Fourier series,

$$u(\xi) = \frac{1}{2} - \sum_{k=1}^{k_{\max}} \frac{1}{\pi k} \sin(2\pi k \xi) \Phi(k, \epsilon), \quad \xi \in \mathbb{R}, \quad (4.40)$$

where k is the wavenumber, $k_{\max}$ a cutoff wavenumber, and $\Phi(k, \epsilon)$ is a smoothing function, which suppresses the amplitudes associated to high wavenumbers:

$$\Phi(k, \epsilon) = \left[\frac{\pi k \epsilon / 2}{\sinh(\pi k \epsilon / 2)} \right], \quad (4.41)$$

Here, ϵ is the smoothing parameter. The smoothing function $\Phi(k, \epsilon)$ is motivated by the viscosity-dependent inertial spectrum of the Burgers equation found by Chorin & Hald [30]. The smoothing function for $\epsilon = 0.1$ is shown in Figure 4.11a. As the smoothing function rapidly decreases with wavenumber, we choose a cutoff wavenumber of $k_{\max} = 100$. The value $\epsilon = 0.1$ is found to produce a smooth version of a sawtooth wave, as shown in Figure 4.11b.

We consider a wave travelling in the positive x -direction by introducing the parametrization $\xi = x - \frac{1}{2}t + \frac{1}{2}$ in (4.39). The source term is then readily found by substituting the chosen solution (4.40) into the Burgers equation (4.39),

$$f(x, t) = u_t + uu_x - \nu u_{xx}. \quad (4.42)$$

The space derivatives in the Burgers equation (in the nonconservative form) are discretized with a second-order central finite difference schemes, using a uniform mesh width $\Delta x > 0$. The error of the numerical solution is again measured in the relative ℓ^2 -norm, cf. (4.32).

(a) Smoothing function (solid line) and (b) Solution for $\epsilon = 0.1$ and $k_{\max} = 100$.
cutoff wavenumber (dashed line).

Figure 4.11: Manufactured solution (4.40) of the Burgers equation.

The tolerance of the PEBK method is set to 10^{-4} . The SVD approximation of the source term, which includes the nonlinear term, is constructed from $s = 50$

samples per subinterval, and reveals that $m = 12$ singular values are sufficient in the truncated SVD, so that the error of the SVD approximation is less than the tolerance of the PEBK method.

The nonlinear system of ODEs is solved iteratively, as outlined in Section 4.2.3. Figure 4.12 shows the error history at different grid resolutions. Here, the error converges to a value that depends on the mesh width. In other words, the final error of the time integration method is much smaller than the error due to the spatial discretization.

Figure 4.12: Error history with $\Delta T = 0.1$. $\square$, $\Delta x = 10^{-2}$; $\circ$, $\Delta x = 5 \cdot 10^{-3}$; $\triangle$, $\Delta x = 2.5 \cdot 10^{-3}$.

The PEBK method is parallelized as discussed in Section 4.2.4. The time interval $[0, T]$ can be partitioned into P subintervals with a uniform subinterval size ΔT . In the following experiment, the complete time interval is extended with each subinterval added, $T = P\Delta T$. The mesh width is $\Delta x = 2.5 \cdot 10^{-3}$. Figure 4.13a shows that increasing the number of processors and hence the simulated time, generally increases the number of iterations required to achieve the same level of accuracy. The increased number of required iterations implies a decrease in theoretical parallel efficiency, which can be estimated by the ratio K_1/K_P , see (4.29).

Next, the final time is kept constant, and the subinterval size reduces with an increase in the number of processors, $\Delta T = T/P$. Figure 4.13b shows that the number of iterations is roughly independent of the number of processors, i.e., in this case, the parallel efficiency, K_1/K_P , does not decrease with P . In fact, there is a slight improvement in convergence because the linearization of the nonlinear term becomes more accurate on smaller subintervals. Parallel speedup might also be improved by the fact that the SVD approximation converges faster on smaller subintervals, see Theorem 1. That is, fewer singular values have to be retained in the truncated SVD in order to achieve a certain accuracy. Also, the number of samples of the source term could be decreased on smaller subintervals.

Figure 4.13: Error history for the Paraxp-EBK method for a fixed subinterval size and a fixed final time. $\square$, $P = 1$; $\circ$, $P = 2$; $\triangle$, $P = 4$; $\star$, $P = 8$; $\diamond$, $P = 16$; $\times$, $P = 32$.

4.4.2 Parallel efficiency

In this section, we test the parallel efficiency of the PEBK method for the viscous Burgers equation. In the following experiments, the source term is chosen such that the solution of the spatially discretized system is equal to the exact solution described in Section 4.4.1. In other words, the source term accounts for the error by the spatial discretization, and we measure only the error due to the time integration. The spatial discretization is here performed with a mesh width of $\Delta x = 2.5 \cdot 10^{-3}$.

If the subinterval size, ΔT , is fixed, the number of iterations generally increases in case the number of processors P increases, see Fig. 4.13. Increasing the number of iterations would reduce the parallel efficiency, according to (4.29). We therefore fix the final time T , such that the subinterval size decreases with increasing P . Also, the total number of samples over $[0, T]$ is fixed, such that the local number of samples decreases with increasing P . To keep the global distribution of samples constant in the parallel computations, the local sample points are uniformly distributed instead of Chebyshev points as used in previous experiments. In our experiments, we use $\Delta T = 0.2/P$ and $s = 128/P$. Furthermore, the number of (retained) singular values is $m = 12$, and the tolerance of the EBK method is set to 10^{-4} . The error history for different P is shown in Fig. 4.14. The results show that the convergence rate of the PEBK method can improve by increasing P . The nonlinear corrections, see (4.18), appear to be more effective on smaller subintervals.

As opposed to the linear problem in Section 4.3.2, communication between the

parallel processes is here required because of the waveform relaxation method. To get a rough estimate of the best possible speedup, we measure the CPU time which would be required by different processors, on a serial computer. Per iteration of the PEBK method, the computation time of each individual process is measured, after which the maximum is stored, i.e., the computation time of the slowest process. The total computation time of the PEBK method is then taken as the sum, over the total number of iterations performed, of the maxima. In this experiment, we have measured a total of ten PEBK iterations.

The computation times for $\nu = 10^{-1}$ and $\nu = 10^{-2}$ are shown in Fig. 4.15a, which clearly demonstrate a considerable parallel speedup. The slightly higher timings at $\nu = 10^{-1}$ could be attributed to the increased stiffness of the problem. Figure 4.15b shows no significant influence of viscosity on the parallel efficiency of the PEBK method. However, at given Δx and ΔT , PEBK does not necessarily converge also at lower ν . For example at $\nu = 10^{-3}$ and $\nu = 10^{-4}$, we only reach convergence when P is high enough ($P \geq 8$ for $\nu = 10^{-4}$). The convergence generally improves with increasing P , because the subintervals become smaller and the nonlinear term is treated more accurately. We also include timings for $\nu = 10^{-3}$ and $\nu = 10^{-4}$ for a range of P values in Fig. 4.15a. These results also indicate that a similar parallel speedup can be achieved for $\nu = 10^{-3}$ and $\nu = 10^{-4}$. The parallel efficiency is not reported however because a reference point for $P = 1$ is not available as convergence is lacking.

The convergence of PEBK for lower values of ν can also be improved by decreasing the final time T . Figure 4.16a shows the timings for $\nu = 10^{-3}$ and $\nu = 10^{-4}$ using $\Delta T = 0.05/P$. In this case, the spatial resolution was $\Delta x = 10^{-4}$ in order to reduce the numerical dissipation with respect to the physical dissipation. Figure 4.16b indicates that a parallel speedup can be also achieved for $\nu = 10^{-3}$ and $\nu = 10^{-4}$. The parallel efficiency of the Parareal algorithm is known to deteriorate for similarly small viscosity coefficients [161]. Furthermore, we note that the actual parallel efficiency of the PEBK method may even be better in practice, since the required number of PEBK iterations could decrease with higher P . Also, the number of singular values, m , could possibly be reduced on smaller subintervals, based on (1), which would further enhance the potential parallel speedup.

The PFASST algorithm [46] shows a good parallel efficiency for the viscous Burgers equation as well. It is unknown whether changing the viscosity coefficient has an impact on the performance of PFASST. Our observations point in the direction of a good parallel efficiency of the PEBK method for simulations of the Navier–Stokes equation at high Reynolds numbers. For these problems there is a high demand for highly efficient parallel solvers [174]. The Parareal algorithm was for example reported to perform poorly in advection-dominated problems [49, 161].

4.4.3 Multiscale example

Turbulent flow is a multiscale phenomenon that is characterized by a wide range of length and time scales. In case of the Burgers equation, we can emulate such

Figure 4.14: Error history for the Paraxp-EBK method with $\Delta T = 0.2/P$. $\square$, $P = 1$; $\circ$, $P = 2$; $\triangle$, $P = 4$; $\star$, $P = 8$.

(a) $\times$ $\nu = 10^{-1}$; $\square$ $\nu = 10^{-2}$; $\diamond$ $\nu = 10^{-3}$; $\circ$ $\nu = 10^{-4}$; dotted line, $\nu = 10^{-1}$ (ideal); dashed line, $\nu = 10^{-2}$ (ideal).

(b) $\times$ $\nu = 10^{-1}$; $\square$ $\nu = 10^{-2}$.

Figure 4.15: Total computation time (left) and parallel efficiency (right) of ten PEBK iterations with $\Delta T = 0.2/P$ and $\Delta x = 2.5 \cdot 10^{-3}$.

(a) $\diamond \nu = 10^{-3}$; $\circ \nu = 10^{-4}$; dotted line, $\nu = 10^{-4}$ (ideal); dashed line, $\nu = 10^{-3}$ (ideal).

(b) $\diamond \nu = 10^{-3}$; $\circ \nu = 10^{-4}$.

Figure 4.16: Total computation time (left) and parallel efficiency (right) of ten PEBK iterations with $\Delta T = 0.05/P$ and $\Delta x = 2.5 \cdot 10^{-4}$.

a solution by combining two functions that are periodic in both space and time, e.g.,

$$u(x, t) = \sin(2\pi x) \sin(2\pi t) + \frac{1}{k_0} \sin(2k_0\pi x) \sin(2k_0\pi t), \quad k_0 > 1. \quad (4.43)$$

The solution features a large scale mode with wavenumber one, and a smaller scale mode with wavenumber $k_0 > 1$. This combination allows the construction of an arbitrarily wide dynamic range. The factor $1/k_0$ is included in compliance with an assumed energy distribution of $[\hat{u}(k)]^2 \propto k^{-2}$ [30], where $\hat{u}(k)$ is the Fourier transform,

$$\hat{u}(k) = \int_{-\infty}^{\infty} u(x, t) \exp(-2\pi i k(x + t)) dx dt, \quad (4.44)$$

where we have used the fact that the solution is periodic in space and time. The previous experiment, see Fig. 4.13a, is repeated for the manufactured solution (4.43) with different values of k_0 . Figure 4.17 illustrates that widening the spectrum does not significantly affect the convergence of the PEBK iterations. Remarkably, the curves appear to form pairs based on P .

4.5 Conclusions

We propose an implementation of the Paraexp method with enhanced parallelism based on the exponential block Krylov (EBK) method. Furthermore, the method,

Figure 4.17: Error history with $\Delta T = 0.1$ and $T = P\Delta T$. $\square$, $P = 1$; $\circ$, $P = 2$; $\triangle$, $P = 4$; $\star$, $P = 8$; $\diamond$, $P = 16$; $\times$, $P = 32$.

Paraexp-EBK (PEBK), is extended to solve nonlinear PDEs iteratively by a waveform relaxation method. The nonlinear terms are represented by a source term in a nonhomogeneous linear system of ODEs. Each iteration the source term is updated with the latest solution. The convergence of the iterative process can be accelerated by adding a correction term based on the Jacobian matrix of the nonlinear term. Each iteration the initial value problem can then be decoupled into independent subproblems, which can be solved parallel in time. Essentially, we implement the Paraexp method within a waveform relaxation approach in order to integrate nonlinear PDEs. Also, the Paraexp-EBK (PEBK) method is used to integrate both the homogeneous and the nonhomogeneous parts of the subproblems. This is in contrast to the original Paraexp method, which assumes a conventional time integration method for the nonhomogeneous parts.

The PEBK method is tested on the advection-diffusion equation for which we demonstrate the parallelization concept for linear PDEs. The parallelization also works in cases without diffusion present, in which the PDE is purely hyperbolic. The parallel efficiency is compared with a Crank–Nicolson (CN) scheme parallelized with the Paraexp algorithm. The parallel efficiency of the PEBK method remains roughly constant around 90%. On the other hand, the parallel efficiency of the CN/Paraexp combination steadily decreases with the number of processors.

As a model nonlinear PDE, the viscous Burgers equation is solved. The number of waveform relaxation iterations required for a certain error tolerance increases, when the relative importance of nonlinearity grows by decreasing the viscosity coefficient. Good parallel efficiency of the EBK method was observed for different values of the viscosity coefficient.

We present numerical experiments using one-dimensional equations, but the

PEBK method is also applicable to equations in two or three dimensions. The parallel performance for higher-dimensional problems is subject of future study. Since the nonlinear convective term in the Burgers equation is shared by the Navier–Stokes equation, the presented results give a hint of the potential of the PEBK method as an efficient parallel solver in turbulent fluid dynamics, where nonlinearity plays a key role. The question remains how to treat the pressure in the incompressible Navier–Stokes equation, which enforces the incompressibility constraint on the velocity field (see for possible approaches in [37, 131]). This will be explored in Chapter 5.

Acknowledgements

Part of the contribution of MAB, in particular in Section 2, was supported via the Russian Science Foundation Grant 14-11-00659.

An exponential time integrator for the incompressible Navier-Stokes equation*

We present an exponential time-integration method for the incompressible Navier–Stokes equation. An essential step in our procedure is the treatment of the pressure by applying a divergence-free projection to the momentum equation. The differential-algebraic equation for the discrete velocity and pressure is then reduced to a conventional ordinary differential equation that can be solved with the proposed exponential integrator. A promising feature of exponential time integration is its potential time parallelism within the Pararexp algorithm. We demonstrate that our approach leads to parallel speedup assuming negligible parallel communication.

5.1 Introduction

With today’s trend towards massively parallel computing, there is a growing interest in parallel-in-time simulations. For the numerical solution of partial differential equations, a parallelization in time could realize additional speedup, when for example the maximum speedup with a parallelization in space is approached, e.g., see [100, 159]. Parallel-in-time simulations have received increased attention after the introduction of the Parareal method [58, 112]; for earlier work see, e.g., [20, 32, 171, 172]. So far, existing methods have not proven themselves to be particularly efficient for parallel-in-time simulations of flows at high Reynolds numbers. In this article, we introduce a time integration method that paves the way to parallel-in-time simulation of incompressible fluid flows at high Reynolds numbers.

*Based on: G.L. Kooij, M.A. Botchev, and B.J. Geurts. *An exponential time integrator for the incompressible Navier-Stokes equation*. Submitted.

A major disadvantage of Parareal is that the parallel efficiency deteriorates for hyperbolic problems [58, 52]. Also for the Navier–Stokes equation at high Reynolds numbers strongly reduced performance was reported [161]. An interesting alternative, the Paraexp method, has been introduced in [54] for the time-parallel solution of linear initial-value problems. With Paraexp, the original problem is decoupled into nonhomogeneous and homogeneous subproblems. Parallel speedup is based on the observation that the homogeneous subproblems can be solved very fast with exponential integrators. These are time integration methods based on the exact integration of linear initial-value problems. Exponential integrators require efficient algorithms to compute the matrix exponential or its product with a vector. There are many methods for computing the matrix exponential. Particularly, Krylov subspace methods prove to be very suitable for the implementation of exponential integrators [36, 54, 126]. A detailed overview of exponential integrators is given in [77].

An extension of Paraexp to nonlinear initial-value problems has been introduced in [98]. Preliminary tests show that parallel speedup is realistic for the viscous Burgers equation, even for low viscosity coefficients. The approach in [98] is based on the exponential block Krylov (EBK) method [11]. A difference with the original formulation of Paraexp is that the exponential integrator, here the EBK method, is used to solve both the nonhomogeneous and the homogeneous subproblems. This unified approach is demonstrated to be effective, as the EBK method can also be very competitive in solving nonhomogeneous subproblems compared to conventional time integration methods.

The application of exponential integrators to fluid dynamics is not straightforward. The incompressible Navier–Stokes equation, discretized in space, is a differential-algebraic equation where a time derivative for the pressure is absent. The continuity equation acts as a constraint equation that imposes a divergence-free condition on the velocity field. The pressure is determined such that the velocity field remains divergence-free. Consequently, special care needs to be taken with exponential integration for the advancement of the pressure in time. One possible approach is to reformulate the governing equations by treating the pressure with a divergence-free projection of the Navier–Stokes equation. This reformulation gives a differential equation for the velocity field that can be solved with Krylov-based exponential integrators, see [37, 131]. Other approaches for Krylov-based exponential integration in the context of fluid dynamics include the method of pseudo compressibility to find steady state solutions [146] and a method for the fully compressible Navier–Stokes equation [149].

In this chapter, we discuss how the EBK method can be extended to the incompressible Navier–Stokes equation. We follow the approach of [37, 131] and treat the pressure by a divergence-free projection. The new time integration method is tested in several numerical experiments including the Taylor–Green vortex and a lid-driven cavity flow. We find that the EBK method can be applied successfully to incompressible flows, also in cases with rather low viscosity. Furthermore, we show that the EBK method can be used within the time-parallel framework

outlined in [98]. We provide a simplified model to analyze how much speedup is feasible with the EBK method for parallel-in-time simulations of the incompressible Navier–Stokes equation. This analysis indicates that with our current implementation of the EBK method a moderate parallel speedup can indeed be expected. This method could provide additional speedup on top of the speedup obtained with a conventional parallelization in space only.

The chapter is organized as follows. In Section 5.2, we explain the basic principle of the EBK method, and discuss how it can be applied to the incompressible Navier–Stokes equation. In Section 5.3, we test the method on a number of test cases, including the Taylor–Green vortex and lid-driven cavity flow. The potential parallel speedup of the EBK method is explored in Section 5.4. Finally, the discussion and conclusions are presented in Section 5.5.

5.2 Exponential time integration

In this section, we describe briefly the exponential block Krylov method, and discuss its application to the incompressible Navier–Stokes equation.

5.2.1 Exponential block Krylov method

Consider the initial value problem (IVP) on $t \in [0, T]$,

$$\mathbf{y}'(t) + A\mathbf{y}(t) = \mathbf{b}(t), \quad \mathbf{y}(0) = \mathbf{y}_0, \quad (5.1)$$

where $\mathbf{y} \in \mathbb{R}^n$ is the unknown function, $\mathbf{b}(t) \in \mathbb{R}^n$, and $A \in \mathbb{R}^{n \times n}$. We are mainly interested in IVPs in the context of the method of lines, in which hyperbolic or parabolic partial differential equations are discretized in the spatial dimensions first. The exact solution of (5.1) is given by the variation-of-constants formula,

$$\mathbf{y}(t) = \exp(-tA)\mathbf{y}_0 + \int_0^t \exp(-(t-\tau)A)\mathbf{b}(\tau) \, d\tau. \quad (5.2)$$

Conventional time integration methods, e.g., the Euler method, use low order approximations based on finite differences to the matrix exponential function. Exponential integrators avoid such approximations by direct evaluation of the matrix exponential function. In this study, we use the *exponential block Krylov* (EBK) method to solve (5.1). Exponential integrators are in general attractive due to their excellent stability and accuracy properties [77]. The EBK method is demonstrated to be particularly competitive for solving (5.1) compared to implicit schemes, and also with respect to other exponential integrators [98]. The EBK method is based on a so-called *block* Krylov subspace, which is generated by the action of a matrix on *multiple* vectors simultaneously. For details of the EBK method, we refer the reader to [11].

An essential step for constructing the block Krylov subspace is a polynomial approximation of the source term of the form

$$\mathbf{b}(t) \approx U\mathbf{q}(t) \quad (5.3)$$

where $U \in \mathbb{R}^{n \times m}$, and $\mathbf{q}(t) \in \mathbb{R}^m$ is a polynomial function of t . It is crucial for successful applications of EBK that m is not too large, so that (5.3) is in fact a low rank approximation, see also [11]. A large m , which is the dimension of the block in the Krylov subspace, is undesirable. In general, the dimension of a Krylov subspace should remain small with respect to the original system for good computational efficiency. Approximation (5.3) can be efficiently constructed in many ways [65, 124]. In this chapter, we use truncated singular value decomposition (SVD) combined with a piecewise polynomial function $\mathbf{q}(t)$ to obtain (5.3). The error of the approximation (5.3) on a given interval $[0, T]$ can be reduced by retaining more singular values (increasing m). If m is found to be too large then we can decrease the time interval $[0, T]$ and repeat the construction of (5.3) which often leads to a much smaller m . In principle, the approximation error in (5.3) can be arbitrarily small, assuming $\mathbf{b}(t)$ is smooth (see for example [166, Section 2.4] for convergence properties of spline interpolation).

We define s nodes on the interval $t \in [0, T]$, $0 = t_0 < t_1 < \dots < t_s = T$. The polynomial approximation of $\mathbf{b}(t)$ is based on the matrix of samples $\tilde{B} = [\mathbf{b}_1 \dots \mathbf{b}_s]$, where $\mathbf{b}_i := \mathbf{b}(t_i)$. For optimal efficiency of the EBK method, we use a truncated SVD of $\tilde{B}$, which reduces the dimension of the block size m substantially with respect to the original dimension of $\tilde{B}$. From the SVD, we find an approximation (5.3). Here, $U\mathbf{q}(t)$ represents the m leading singular vectors of the SVD of $\mathbf{b}(t)$. Based on a block of vectors U instead of a single vector, we can define the block Krylov subspace

$$\mathcal{K}_l(A, U) := \text{span} \{U, AU, A^2U, \dots, A^{l-1}U\}, \quad (5.4)$$

where l is the Krylov iteration index. A linear basis of the block Krylov subspace is generated by the block Arnoldi or by the block Lanczos process, if A is symmetric (see [145]). After the basis has been computed, the original IVP can be projected onto the block Krylov subspace using the orthogonality of the basis vectors [11]. The dimensions of the projected IVP are typically much smaller than of the original problem ($lm \ll n$). Because of its strongly reduced problem size, the projected IVP can be solved cheaply and accurately (to any desired tolerance) with a general ODE solver (see “Method 5” in [126]). This step is performed at every Krylov iteration. The Krylov iterations, incrementing l at every iteration, stop when the exponential residual meets the stopping criterion [13]. The accuracy of the numerical solution is thus controlled by the residual in the block Krylov process and by the low rank approximation (5.3). The approximation (5.3) on $t \in [0, T]$ can be improved by either taking more samples s , or by saving more singular values m .

5.2.2 Incompressible Navier–Stokes equation

In Section 5.2.1, we have introduced the EBK method for solving the linear IVP (5.1). In this section, we present an algorithm in which the EBK method is used for exponential time integration of the incompressible Navier–Stokes equation. We consider the incompressible Navier–Stokes equation on a d -dimensional space domain Ω ($d = 2$ or $d = 3$) with boundary $\partial\Omega$. The governing equations in $\Omega \times [0, T]$ are

$$\vec{u}_t + (\vec{u} \cdot \nabla) \vec{u} = -\nabla p + \nu \Delta \vec{u}, \quad \vec{u}(0) = \vec{u}_0, \quad (5.5)$$

$$\nabla \cdot \vec{u} = 0, \quad (5.6)$$

with appropriate boundary conditions on $\partial\Omega$. Here, $\vec{u} \in \mathbb{R}^d$ is the velocity vector, p the pressure, and ν the kinematic viscosity. We follow the method of lines approach by discretizing in space first. There are many suitable discretization techniques for the Navier–Stokes equation, with popular choices being finite volume and finite element methods [45, 184]. These discretization methods typically result in a semi-discrete system of the general form

$$G\mathbf{u}' + N(\mathbf{u})\mathbf{u} + B^T \mathbf{p} + A_\nu \mathbf{u} = \mathbf{f}, \quad \mathbf{u}(0) = \mathbf{u}^0, \quad (5.7)$$

$$B\mathbf{u} = \mathbf{g}, \quad (5.8)$$

where G is the mass matrix, $N(\mathbf{u})$ is the convective operator, B the divergence operator, B^T the gradient operator, and A_ν the viscous operator. We emphasize that time remains an independent variable and $\mathbf{u} = \mathbf{u}(t)$ is still a function of time with derivative $\mathbf{u}'$. Note that some discretization methods may slightly deviate from the general form. For example, for inf-sup unstable finite element spaces an additional stabilization term is added to the continuity equation [45]. For some discretization methods, it holds that the mass matrix G is the identity matrix. The vectors $\mathbf{f}$ and $\mathbf{g}$ may depend on the boundary conditions on the velocity field, and thus are not necessarily zero. We consider problems with n_u degrees of freedom (DOFs) associated with the velocity field components, and n_p with the pressure field, i.e., $(\mathbf{u}, \mathbf{p}) \in \mathbb{R}^{dn_u \times n_p}$. The DOFs n_u and n_p may differ, depending on the spatial discretization. The pressure and velocity components are located at different gridpoints when for example different finite element spaces are used for the velocity and the pressure, or when a staggered grid is used in a finite volume method.

Equations (5.7) and (5.8) form essentially a differential-algebraic equation in which the discretized continuity equation (5.8) can be seen as an algebraic constraint on the velocity field. The differential-algebraic nature of this equation is different than the basic ODE, given by (5.1). We note that the compressible Navier–Stokes equations are less complicated in that sense, because a time derivative is present also in the continuity equation. Therefore existing exponential integrators can be readily applied to compressible Navier–Stokes equations [149], while for incompressible flow this is more challenging.

Equation (5.7) cannot be integrated directly with the EBK method, because of (a) the nonlinear convection term, and (b) the algebraic constraint on $\mathbf{u}$ in the form of the discrete continuity equation. We now present an approach on how to handle these issues.

The nonlinearity of the problem can be dealt with by incorporating the EBK method into an iterative procedure [98]. Namely, the nonlinear term is linearized about a state $\bar{\mathbf{u}}$, specified later, as follows

$$N(\mathbf{u}_{k+1})\mathbf{u}_{k+1} \approx N(\mathbf{u}_k)\mathbf{u}_k + N(\bar{\mathbf{u}})[\mathbf{u}_{k+1} - \mathbf{u}_k], \quad (5.9)$$

where k is the iteration index. Instead of $N(\bar{\mathbf{u}})$, one could use the Jacobian matrix of the nonlinear term as well. In many codes the matrix $N(\mathbf{u})$ is readily available, which makes the implementation of the linearization above easier in practice. Using this linearization the momentum equation becomes

$$G\mathbf{u}'_{k+1} + [A_\nu + N(\bar{\mathbf{u}})]\mathbf{u}_{k+1} + B^T\mathbf{p}_{k+1} = \mathbf{f}_k, \quad (5.10)$$

$$B\mathbf{u}_{k+1} = \mathbf{g}. \quad (5.11)$$

We will work with $\bar{\mathbf{u}}$ as the average of $\mathbf{u}_k$ on $[0, T]$, such that the matrix $[A_\nu + N(\bar{\mathbf{u}})]$ is independent of time, similar to (5.1). It should be noted that $\mathbf{u}_k(t)$ is the solution at iteration k on the complete interval $[0, T]$. In practice, the solution is stored at multiple temporal points as a matrix $U_k = [\mathbf{u}^1 \dots \mathbf{u}^s]$. The iterative process is usually started with a constant vector $\mathbf{u}_0 = \mathbf{u}(0)$, and for $\bar{\mathbf{u}}$, we normally take a time-average $\bar{\mathbf{u}} = \frac{1}{T} \int_0^T \mathbf{u}_k(t) dt$. The right-hand side $\mathbf{f}_k$ contains the nonlinear remainder,

$$\mathbf{f}_k := \mathbf{f} - [N(\mathbf{u}_k) - N(\bar{\mathbf{u}})]\mathbf{u}_k. \quad (5.12)$$

Next, we deal with the pressure in (5.10). We manipulate (5.10) in such a way that the pressure is eliminated.

Multiplying (5.10) with BG^{-1} , we find

$$B\mathbf{u}'_{k+1} + BG^{-1}([A_\nu + N(\bar{\mathbf{u}})]\mathbf{u}_{k+1} + B^T\mathbf{p}_{k+1}) = BG^{-1}\mathbf{f}_k. \quad (5.13)$$

To simplify this equation, we use the time derivative of the continuity equation (5.11), which reads

$$B\mathbf{u}'_{k+1} = \mathbf{g}'. \quad (5.14)$$

Substituting relation (5.14) into (5.13) gives us an algebraic equation for the pressure,

$$BG^{-1}B^T\mathbf{p}_{k+1} = [BG^{-1}(\mathbf{f}_k - (A_\nu + N(\bar{\mathbf{u}}))\mathbf{u}_k) - B\mathbf{g}'], \quad (5.15)$$

which is essentially a discrete Poisson equation, since $BG^{-1}B^T$ represents a discrete Laplace operator. Substituting the solution of (5.15) into (5.10) removes the pressure from the momentum equation,

$$\mathbf{u}'_{k+1} + P[A_\nu + N(\bar{\mathbf{u}})]\mathbf{u}_{k+1} = P\mathbf{f}_k - G^{-1}B^T(BG^{-1}B^T)^{-1}\mathbf{g}', \quad (5.16)$$

where, to simplify notation, we have introduced the projection operator,

$$P := G^{-1} - G^{-1}B^T(BG^{-1}B^T)^{-1}BG^{-1}. \quad (5.17)$$

Note that P projects discrete velocity fields onto a (discretely) divergence-free space that satisfies the condition $B\mathbf{u} = 0$. We note that for some discretizations the explicit calculation and storage of G^{-1} is not practical, but in general the action of G^{-1} on a vector is easily computed with direct or iterative linear solvers. The matrix P is also never calculated explicitly for the same reason. Our procedure of treating the pressure in the momentum equation is in the same vein as classic pressure correction or fractional step methods (see, e.g., [28, 72, 85, 168, 176]). In fact, the projection operator in [176] is identical to (5.17) in the case of $G = I$.

The projection P is essentially the discrete equivalent of the Leray projection [29, 50]. It is easy to verify that the projection P has the analogous properties

1. $P^2\mathbf{u} = P\mathbf{u}$, for all $\mathbf{u} \in \mathbb{R}^{dn_u}$,
2. $BP\mathbf{u} = 0$, for all $\mathbf{u} \in \mathbb{R}^{dn_u}$,
3. $P\mathbf{u} = \mathbf{u}$, for all $\{\mathbf{u} \in \mathbb{R}^{dn_u} | B\mathbf{u} = 0\}$,
4. $PB^T\mathbf{p} = 0$, for all $\mathbf{p} \in \mathbb{R}^{n_p}$.

Equation (5.16) is a linear IVP that can be solved with the EBK method, cf. (5.1). A similar approach for exponential integrators for the incompressible Navier–Stokes equation is given by [131]. In our case, the projection operator P is derived directly from the semi-discretization in (5.7).

At each iteration k the EBK method is employed to find $\mathbf{u}_{k+1}$ by solving (5.16). Therefore, EBK can be seen as an inner iterative process, and iterations k as outer iterations. The convergence of the outer iterations has been studied in [122, 185] (in a slightly different context of waveform relaxation methods). The convergence is generally slower if the nonlinear term becomes larger, and it will be investigated whether the convergence is still sufficient for our application [98]. The outer iterations can be stopped if they are converged within a certain tolerance,

$$\|\mathbf{u}_{k+1} - \mathbf{u}_k\|_\infty < \text{tol}, \quad (5.18)$$

which does not only show the stagnation in iterations but also controls the nonlinear residual [98]. Generally, the tolerance for the nonlinear residual should be in the same order as the tolerance for the exponential residual, i.e., the stopping criterion for the inner iterations of the EBK methods. After the outer iterations are ended, we form the matrix $U = [\mathbf{u}^1 \dots \mathbf{u}^s]$ that is comprised of s samples of the solution $\mathbf{u}(t)$ on the interval $[0, T]$. This time interval is typically much larger than the time step size Δt of traditional time integration methods restricted by a Courant–Friedrichs–Lewy (CFL) condition. In numerical experiments in [98], we find $T/\Delta t = \mathcal{O}(10^3)$ for example.

5.3 Incompressible flow simulations

In Section 5.2, we covered the basics of the EBK method, and how it can be applied to the incompressible Navier–Stokes equation. In this section, we now present several numerical experiments with our time integration method. These experiments involve a Taylor–Green vortex and lid-driven cavity flow.

The governing equations are discretized in space with a finite element method implemented in the MATLAB package IFISS [42, 43]. In our numerical experiments, we are using *Taylor–Hood* Q_2 – Q_1 elements. This means that the velocity components are approximated with continuous piecewise quadratic polynomials, and the pressure with continuous piecewise linear polynomials. Taylor–Hood elements are *inf-sup* stable [45].

The projection operator (5.17) requires the solution of a linear system. Because the test and trial functions are nonorthogonal, the mass matrix G is not diagonal in our case. Therefore, it is not practical to explicitly construct the discrete Laplace operator, $BG^{-1}B^T$. Instead of constructing this matrix, we simply solve the block-coupled system,

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{pmatrix} \mathbf{v} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^* \\ 0 \end{pmatrix}, \quad (5.19)$$

where $\mathbf{v}, \mathbf{v}^* \in \mathbb{R}^{dn_u}$, and $\boldsymbol{\lambda} \in \mathbb{R}^{n_p}$. The linear system is solved with a direct solver in MATLAB. It is not hard to verify that the solution of (5.19) is equivalent to the action of the projection operator, $\mathbf{v} = P\mathbf{v}^*$, as defined in (5.17). With a block Gaussian elimination, the system can be reduced to

$$\begin{bmatrix} G & B^T \\ 0 & BG^{-1}B^T \end{bmatrix} \begin{pmatrix} \mathbf{v} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^* \\ BG^{-1}\mathbf{v}^* \end{pmatrix}, \quad (5.20)$$

After back-substitution, we find the solution for $\mathbf{v}$,

$$\mathbf{v} = G^{-1}\mathbf{v}^* - G^{-1}B^T(BG^{-1}B^T)^{-1}BG^{-1}\mathbf{v}^*. \quad (5.21)$$

This solution is indeed identical to $\mathbf{v} = P\mathbf{v}^*$, according to the definition of P (5.17). If the problem size of (5.19) becomes too large for direct solvers, iterative solvers can be used. In practice, variations of the Uzawa algorithm are a possible choice for solving the saddle-point problem (5.19) by decoupling $\mathbf{v}$ and $\boldsymbol{\lambda}$, see [118]. Alternatively, block coupled solvers are feasible with efficient preconditioners [8, 9, 41, 44]. For a recent overview see [45].

To ensure that the initial condition satisfies the discrete continuity equation at $t = 0$, the velocity field is projected onto a divergence-free space. We correct the initial condition by solving the linear system

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{pmatrix} \mathbf{u}^0 \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{u}}^0 \\ \mathbf{g}(0) \end{pmatrix}, \quad (5.22)$$

where $\tilde{\mathbf{u}}^0$ is the uncorrected initial condition. As a result, the initial condition satisfies $B\mathbf{u}^0 = \mathbf{g}(0)$.

5.3.1 Taylor–Green vortex

In the first test case, we consider the so-called Taylor–Green vortex, for which an exact solution to the unsteady Navier–Stokes equation is known, given by

$$\vec{u}(x, y, t) = \begin{pmatrix} -\sin(\pi x) \cos(\pi y) \exp(-2\pi^2 \nu t) \\ \cos(\pi x) \sin(\pi y) \exp(-2\pi^2 \nu t) \end{pmatrix}. \quad (5.23)$$

We solve the problem numerically on a square domain, $\Omega = [-1, 1] \times [-1, 1]$, with time-dependent Dirichlet boundary conditions and initial condition consistent with (5.23). In other words, the values of the boundary conditions and initial condition follow from the evaluation of the exact solution at the boundaries or at time zero.

The right-hand side (RHS) in (5.16) is calculated on N_t nodes (time moments), which is necessary to construct the low rank polynomial approximation in (5.3). The nodes are equally spaced in time with a distance Δt . The step size Δt is not restricted by stability conditions, but we use the Courant number to get a natural estimate of an appropriate Δt . Here, we choose Δt to satisfy the condition $C = u\Delta t/\Delta x \leq 1$, where $u = 1$ (initially) and Δx is the spatial distance between the finite element nodes. Also, we preserve $m = 4$ leading singular values from the SVD, which appears to be sufficient for our purpose. This initial choice of parameters is based on prior experiments [98], but could still be optimized. We discuss the influence of m and N_t later in Section 5.3.2.

Equation (5.16) is integrated by the EBK method using a residual tolerance of $\text{tol} = 10^{-3}$, which is sufficient in this case. The outer iterations are stopped when the stopping criterion (5.18) is reached, using the same tolerance of 10^{-3} . To reach the tolerance, typically four outer iterations in total are required.

We integrate from $t = 0$ to $t = 1$, and measure the numerical error in the x -component of the velocity at the final time, measured in the relative L^2 -norm, $\|u - u_{\text{ref}}\|/\|u_{\text{ref}}\|$. In this case, the numerical error is mostly determined by the spatial discretization. Figure 5.1 shows the error varying with the spatial resolution for several values of ν . The error is generally larger when ν decreases, which is related to the tendency of central discretization schemes to produce small “wiggles” at higher mesh Reynolds numbers, i.e., for advection-dominated flows. Similar wiggly behavior has been observed using the default Crank–Nicolson scheme of IFISS. In such cases, the accuracy can be improved by using a finer mesh or by using a higher-order upwind-type discretization of the advection term [67]. Furthermore, the error shows approximately fourth-order convergence with Δx , which is one order higher than expected from typical error estimates [45]. This form of superconvergence might be attributed to the high regularity of the exact solution [181].

The numerical error in this test case is primarily dominated by the spatial discretization. This validates the convergence of the underlying spatial discretization. Variations of the EBK parameters (m , N_t , tol) have little influence on the error here. In the next section, we examine these parameters in more detail.

Figure 5.1: Taylor–Green vortex: error in x -velocity at $t = 1$ versus spatial resolution. $\times \nu = 10^{-3}$; $\circ \nu = 10^{-2}$; $\square \nu = 10^{-1}$; $-- O(\Delta x^4)$.

5.3.2 Lid-driven cavity flow

Lid-driven cavity flow is a well-known benchmark problem used to test solvers for the Navier–Stokes equation, see for example [17, 62]. In this section, we analyze the performance of the EBK method in simulations of lid-driven cavity flow. In this case, the exact solution to the Navier–Stokes equation is not known, but we compare EBK-based results with the Crank–Nicolson method, which is implemented as the default timestepping method in IFISS [92].

We study lid-driven cavity flow in a square domain, $\Omega = [-1, 1] \times [-1, 1]$. No-slip boundary conditions apply at the side and bottom walls. The velocity in x -direction at the top boundary, the “lid” of the cavity, is prescribed by the function $U(x, t)$. The geometry and the boundary conditions of the problem are sketched in Fig. 5.2.

Stokes flow

First, we simulate unsteady Stokes flow in a cavity, i.e., ignoring the nonlinear convection term in (5.7), which would approximate laminar flow at very low Reynolds numbers. The initial condition is $\vec{u} = 0$, and the lid is accelerated gradually from zero to a steady velocity, spinning up the flow in the cavity. The lid velocity is given by

$$U(x, t) = (1 - x^2)(1 + x^2)(1 - e^{-t}). \quad (5.24)$$

Note that the boundary conditions are *regularized* to prevent corner singularities, i.e., the velocity discontinuities at the top corners, see for example [19]. Otherwise very high spatial resolution is required to accurately resolve the solution near those points. However, this is mainly irrelevant for the performance of the time integration method. In this case, the time-dependency of the RHS in (5.10) only

Figure 5.2: Geometry and boundary conditions of a lid-driven cavity flow. The domain is $\Omega = [-1, 1] \times [-1, 1]$.

comes from the transient boundary condition (5.24), and not from the nonlinear term.

Without the presence of a nonlinear term, no outer iterations are needed, and we can focus on certain aspects of the EBK method in more detail. The accuracy of the EBK method mainly depends on the low rank approximation of the source term (5.3). The main parameters are (a) the number of source term samples, N_t , and (b) the number of modes from the truncated SVD, m .

As explained in Section 5.2.1, the numerical error of the EBK method depends on the residual tolerance and the error in low rank approximation of the RHS (5.3). In the following test, we use a low residual tolerance of $\text{tol} = 10^{-10}$, such that the numerical error mainly depends on the accuracy of the low rank approximation. For the spatial discretization, we use 16×16 finite elements, which amounts to 33×33 grid points for the velocity field components.

We can define a Reynolds number based on $Re = U_{ref}L/\nu$, where $U_{ref} = 1$ is the maximum lid velocity and $L = 2$ is the cavity length. In this test case, the viscosity coefficient is $\nu = 0.02$, which corresponds to $Re = 100$. We integrate from $t = 0$ to $t = 1$. A reference solution is computed with the Crank–Nicolson method using a small time step size and Richardson extrapolation. The error in the x -velocity is measured in the relative L^2 -norm. Figure 5.3 shows the numerical error at $t = 1$. As expected, the accuracy increases with the number of samples N_t . Also, the error in the continuity equation decreases with N_t , because the RHS (5.10) is approximated more accurately. Both errors display fifth-order convergence with N_t here. Because the approximation (5.3) is based on cubic spline interpolation, we should expect at least a fourth-order convergence.

A time-consuming part in the EBK method is the block Arnoldi process, which generates an orthonormal basis of the block Krylov subspace. At every Krylov iteration, the action of the matrix $P(A + J(\bar{\mathbf{u}}))$ on m vectors is required. This matrix

action is not a simple matrix-vector multiplication, as the matrix $P(A + J(\bar{\mathbf{u}}))$ is not calculated explicitly. Because P involves the solution of a discrete Poisson problem, see (5.17), each Krylov iteration is at least as expensive as a Poisson solve. An SVD shows however that the singular values may rapidly decrease in magnitude, see Fig. 5.4 for example. The difference between the second and third value is several orders of magnitude in this example. Thus, in this case, we can save considerable computational work by truncating the SVD with $m = 2$ without losing any accuracy in practice.

Figure 5.5 shows the number of matrix actions required by EBK against the number of samples. The number stays constant for $m = 2$, but increases strongly when increasing $m = N_t$, i.e., no truncation. Increasing N_t is not expected to produce a larger truncation error necessarily, because it does not introduce more dynamics in the RHS function. It could be that a limited number of modes is sufficient to represent the RHS function accurately. Adding more snapshots, indicated by N_t , does not affect that. In this case, a significant number of matrix actions can be saved with a truncated SVD, which improves the computational efficiency of the EBK method significantly, with practically no effect on the accuracy of the solution. In some cases, the number of matrix actions can be reduced from 1248 to 114, see Fig. 5.5. This shows that a truncated SVD is instrumental in realizing good computational efficiency of the EBK method in practice.

Figure 5.3: Numerical error at $t = 1$ versus number of samples N_t . $\square$ relative L^2 -error of x -velocity, $\times$ L^∞ -error of continuity equation ($\|B\mathbf{u} - \mathbf{g}\|_\infty$), $---$ $O(N_t^{-5})$. Stokes flow in a lid-driven cavity with $\nu = 0.02$.

Steady state

Steady state solutions to lid-driven cavity flow are well-studied in literature [17, 62]. In the following test case, we use the EBK method to solve the *steady* Navier–Stokes equation. We verify that the EBK method indeed converges to the correct steady state. In this case, we use an equidistant grid of 16×16 elements, i.e.,

Figure 5.4: Singular values σ_j for $N_t = 32$. Stokes flow in a lid-driven cavity with $\nu = 0.02$.

Figure 5.5: Actions of the matrix $P[A + J(\bar{\mathbf{u}})]$ versus number of samples N_t . $\times$ $m = N_t$, $\square$ $m = 2$. Stokes flow in a lid-driven cavity with $\nu = 0.02$.

33×33 gridpoints. The flow is initialized as a solution to Stokes flow. Using this initial condition, the Navier–Stokes equation is integrated over a long time interval $t \in [0, T_{end}]$ until the solution reaches a steady state.

Based on the results in Fig. 5.4, we use $m = 4$ to capture the relevant modes. Also, we use $N_t = 33$, which corresponds to a CFL number of $C = 5$ in case of $T_{end} = 10$. Also, we use $\text{tol} = 10^{-3}$ for controlling the exponential and the nonlinear residuals. The precise choice of parameters matters very little in this case. We have observed that the accuracy of the time integration method has a marginal influence on the error of the steady state solution.

The steady solution at $Re = 100$ is plotted in Fig. 5.6, for which we used $T_{end} = 10$. The results obtained with the EBK method are in good agreement with the reference data from [62]. The EBK method yields the correct steady state solution. We also compare the EBK method to the default steady Navier–Stokes solver of IFISS, which provides us with a reference solution (on the same mesh). The errors of the velocity profiles, measured in the relative L^2 -norm, are given in Table 5.1. In general, the EBK method agrees well with the default steady Navier–Stokes solver in IFISS. The final time appears to be the main parameter influencing the error. Increasing T_{end} decreases the error as the solution becomes closer to the true steady state. Long time intervals are needed because the transient solution approaches the steady state rather slowly, especially in cases with low viscosity.

Oscillating lid

In this section, we examine the transient solution of a lid-driven cavity flow. In this case, the lid velocity oscillates in time. As such, the solution is intrinsically unsteady and will not tend to a steady state asymptotically. The lid velocity is

Figure 5.6: Steady state solution of lid-driven cavity flow at $Re = 100$, shown by the velocity components along lines through the geometric center of the cavity. — EBK (33×33 grid), $\square$ Ghia et al. [62].

Table 5.1: Error in L^2 -norm of the steady state solution of lid-driven cavity flow at $Re = 100$.

T_{end}	Error u	Error v
10	7.96E-3	1.55E-2
20	1.36E-3	2.32E-3
40	1.72E-4	3.12E-4
80	4.63E-5	8.54E-5

prescribed as

$$U(x, t) = (1 - x^2)(1 + x^2) \left(1 + \frac{1}{2} \sin(2\pi t)\right). \quad (5.25)$$

We use a 33×33 grid, which was shown in Section 5.3.2 to be an adequate resolution for resolving the steady state solution at $Re = 100$. In this case, we want to calculate a time-accurate solution, and the RHS in (5.16) is evaluated at $N_t = 33$ points on the interval $t \in [0, T_{end}]$. This corresponds to a Courant number of approximately $C = 0.5$, which is below the typical stability limit of explicit time-stepping methods. We calculate the solution at $T_{end} = 1$, i.e., after one oscillation of the lid. An accurate reference solution is computed with the CN method, which is the default timestepping method of IFISS. As in Section 5.3.2, we measure the error in the u -profile along the vertical centerline of the cavity, and in the v -profile along the horizontal centerline. The errors are measured in the relative L^2 -norm.

The errors are shown in Fig. 5.7 for two different Reynolds numbers. The error decreases as the number of SVD modes increases, until the error is limited by the tolerance of the EBK method. This shows that a stricter tolerance is needed for higher m in order to achieve a higher accuracy. For example, at $m = 4$ and $Re = 100$ the choice of the tolerance does not influence the total accuracy, and a tolerance of 10^{-2} would suffice. Note that the matrix-vector multiplications are expensive in the EBK method, because they involve the projection matrix P . The number of matrix actions is proportional to m . Therefore it would be beneficial not to take m too large. The results for $Re = 100$ and $Re = 1000$ are both very similar, which suggests that the Reynolds number has a limited influence on the accuracy of the EBK method.

Figure 5.7: Error at $t = 1$ versus m for several tolerances (`tol`) of the exponential and nonlinear residual. Solid: error in u , dashed: error in v . $\circ$ `tol` = 10^{-2} , $\square$ `tol` = 10^{-3} , $\times$ `tol` = 10^{-4}

Instead of computing the solution on the entire interval $[0, T_{end}]$ directly, one could divide the interval into multiple subintervals of size ΔT . The solution on

the subintervals are then computed sequentially. The number of outer iterations depends largely on the size of the time interval ΔT . Figure 5.8 shows the number of iterations as function of the time interval size ΔT . As ΔT increases, more iterations are required to achieve a given tolerance. The number of matrix actions, which are an important indication of the computational costs, does not necessarily increase by increasing ΔT however. Also, the Reynolds number plays a role. As the Reynolds number increases, the nonlinearity becomes stronger and more iterations are needed. At $Re = 3000$ we see a significant increase for $\Delta T > 0.25$ for example. We also measure the total number of matrix actions, N_{mat} , which involves the matrix $P[A + J(\bar{\mathbf{u}})]$. This provides a reasonable estimate of the computational cost. For $Re = 1000$ and $Re = 3000$ the number of matrix actions starts to increase from $\Delta T > 0.25$, due to the considerable increase in the number of iterations. In practice, the optimal ΔT can be determined by experimentation. These results suggest that it is preferable to choose ΔT as large as possible, without the number of iterations becoming excessively large. However, it is expected that the number of iterations increases with the Reynolds number for a given ΔT .

Figure 5.8: Number of outer iterations and matrix actions versus the time interval size ΔT , using $m = 4$ and $\text{tol} = 10^{-2}$. The matrix actions are compensated with ΔT ($N_{mat}/\Delta T$). $\times$ $Re = 100$, $\square$ $Re = 300$, $\circ$ $Re = 1000$, $\diamond$ $Re = 3000$.

5.4 Parallellization in time

In this section, we describe how the EBK method can be parallelized. We provide a basic model to study the potential scaling for the incompressible Navier–Stokes equation and determine the conditions under which EBK can become competitive in terms of computing time.

5.4.1 Parallel algorithm

The EBK method can be parallelized in time naturally with the Paraexp method, which is based on “parallel exponential propagation”. The Paraexp method was introduced as an efficient parallel-in-time algorithm for nonhomogeneous differential equations [54]. One advantage of Paraexp is that its parallel performance does not deteriorate for hyperbolic PDEs or parabolic PDEs with little dissipation. This deterioration is shown to be a major weakness of the earlier and well-known Parareal method [52, 161]. The EBK method allows for a natural extension of the Paraexp method to nonlinear PDEs. Kooij et al. [98] show that realistic speedup can be expected for the advection–diffusion equation and the viscous Burgers equation.

The idea behind the Paraexp method is that a linear problem can be decoupled into independent subproblems using the principle of superposition. In our case, we solve (5.16) in parallel at every outer iteration. This equation is of the form,

$$\mathbf{y}'(t) = \mathcal{A}\mathbf{y}(t) + \mathbf{g}(t), \quad \mathbf{y}(0) = 0, \quad t \in [0, T]. \quad (5.26)$$

Here, without loss of generality, we assume that the problem is transformed to have a zero initial condition. This means using a substitution $\mathbf{y}(t) = \hat{\mathbf{y}}(t) - \mathbf{y}_0$, where $\hat{\mathbf{y}}(t)$ is the solution to the original problem with the nonzero initial condition $\mathbf{y}_0$. In the case of nonlinear problems, (5.26) is updated at every outer iteration. The matrix $\mathcal{A}$ contains the Jacobian matrix of the nonlinear term averaged over $t \in [0, T]$, and the source term $\mathbf{g}(t)$ contains the nonlinear remainder. The Jacobian matrix is time-averaged, such that $\mathcal{A}$ does not depend on t . The time interval is divided into $\mathcal{P}$ non-overlapping subintervals, $0 = T_0 < T_1 < \dots < T_{\mathcal{P}} = T$. Based on these subintervals, we define $\mathbf{g}_j(t)$ as the piecewise function

$$\mathbf{g}_j(t) = \begin{cases} \mathbf{g}(t), & \text{for } T_{j-1} \leq t < T_j, \\ 0, & \text{otherwise.} \end{cases} \quad (5.27)$$

Next, the original problem can be written as the combination of $\mathcal{P}$ independent problems,

$$\mathbf{v}_j'(t) = \mathcal{A}\mathbf{v}_j(t) + \mathbf{g}_j(t), \quad \mathbf{v}_j(0) = 0, \quad t \in [0, T]. \quad (5.28)$$

These problems can be solved in parallel on $\mathcal{P}$ processors and do not require any communication during the solution procedure. It is easy to see that the solution to (5.26) is the sum of the solutions to the subproblems (5.28),

$$\mathbf{y}(t) = \sum_{j=1}^{\mathcal{P}} \mathbf{v}_j(t). \quad (5.29)$$

Parallel speedup is expected on the observation that the source term is zero in all but one subinterval. The closed-form solution in the homogeneous part of (5.28) is given by

$$\mathbf{v}_j(t) = \exp((t - T_j)\mathcal{A}) \mathbf{v}_j(T_j), \quad t \geq T_j, \quad (5.30)$$

which is also called the *exponential propagation* of the solution. The matrix exponential in (5.30) can be evaluated very efficiently [126]. The computation time of (5.30) is typically negligible compared to the time needed for solving the nonhomogeneous part of (5.28) [54]. For a fast evaluation of (5.30), efficient algorithms for calculating the matrix exponential or its action on vectors are needed. These methods are typically much faster than traditional timestepping methods. Each parallel processor only deals with a part of the original nonhomogeneous problem. Parallel speedup is based on this premise. In the original formulation of the Paraexp method the nonhomogeneous part is solved by an off-the-shelf ODE solver. With the EBK method, we can take a unified approach by solving both the homogeneous and the nonhomogeneous part of (5.28) with the same time integrator. Such a unified approach is shown to be more effective than an original implementation of the Paraexp method [98].

5.4.2 Speedup and efficiency

We consider strong scaling when integrating on a fixed interval $t \in [0, \Delta T]$. The wall clock time of the sequential algorithm is $\mathcal{T}_1 = \tau_1 K_1$, where τ_1 is the time of one EBK solve on the interval ΔT and K_1 is the number of outer iterations. The wall clock time of the parallel algorithm is $\mathcal{T}_{\mathcal{P}} = \tau_{\mathcal{P}} K_{\mathcal{P}} = (\tau_{nh} + \tau_h + \tau_c) K_{\mathcal{P}}$, where $\tau_{\mathcal{P}}$ is the wall-clock time of one outer iteration, τ_{nh} the solver time for the nonhomogeneous part of the IVP, and τ_h for the homogeneous part. Here, τ_c denotes the communication time between the parallel processes per outer iteration, and $K_{\mathcal{P}}$ the number of outer iterations of the parallel EBK method. In the case of $\mathcal{P} = 1$, we simply have $\tau_{nh} = \tau_1$ and $\tau_h = \tau_c = 0$. After every outer iteration, the solutions of the subproblems are summed and the source term, containing the nonlinear terms, is updated with the latest solution. These operations are included in τ_c . The parallel efficiency is then calculated as

$$\eta = \frac{\mathcal{T}_1}{\mathcal{P}\mathcal{T}_{\mathcal{P}}} = \frac{\tau_1 K_1}{\mathcal{P}(\tau_{nh} + \tau_h + \tau_c) K_{\mathcal{P}}}. \quad (5.31)$$

This expression shows that ideal efficiency of $\eta \approx 1$ is obtained under several conditions. First, the time spent on the nonhomogeneous part should be proportional to $\mathcal{P}$, $\tau_{nh} = \tau_1/\mathcal{P}$. Second, the time spent on the homogeneous part should be much less than the nonhomogeneous part, $\tau_h \ll \tau_{nh}$. Third, the communication time should be relatively small, $\tau_c \ll \tau_h + \tau_{nh}$. Finally, the required number of outer iterations should not increase, $K_{\mathcal{P}} \leq K_1$. The communication pattern of the Paraexp/EBK method is different than the Parareal method, because it does not have a “coarse propagator” which is an intrinsically sequential part of the Parareal method.

Under some assumptions, we can propose a simple model for predicting the maximal speedup that can be achieved. In the ideal case of perfect parallel efficiency, we have $\tau_h \ll \tau_{nh}$, but in practice τ_h needs to be taken into account. We can write τ_h as a fraction α of τ_{nh} , i.e., $\tau_h = \alpha\tau_{nh}$. In practice, α is greater than

zero, depending on the number of subintervals/processors $\mathcal{P}$, i.e., $\alpha = \alpha(\mathcal{P}) > 0$. Based on observations shown later, we expect α to be around 0.1. The parallel speedup is

$$S = \frac{\tau_1}{\tau_{\mathcal{P}}} = \frac{\tau_1 K_1}{((1 + \alpha(\mathcal{P}))\tau_{nh} + \tau_c)K_{\mathcal{P}}} \quad (5.32)$$

The Jacobian matrix is time-averaged over each subinterval individually. It is safe to assume that $K_{\mathcal{P}} \leq K_1$, because the nonlinear correction, using a time-averaged Jacobian matrix, is more accurate on smaller subintervals. For now, we assume that the communication is negligible, $\tau_c = 0$, and that $\tau_{nh} = \tau_1/\mathcal{P}$, corresponding to the fact that τ_1 relates to EBK on the whole interval and $\tau_{\mathcal{P}}$ to EBK on one of the subintervals. We assume that $\tau_{\mathcal{P}} < \tau_1$, because the nonhomogeneity is limited to a smaller subinterval $(\Delta T/\mathcal{P})$. Under these assumptions, the speedup can be simplified to

$$S = \frac{\mathcal{P}}{1 + \alpha(\mathcal{P})}. \quad (5.33)$$

A low value for α is important for high parallel speedup. In a worst-case scenario, we could have $\tau_h + \tau_{nh} = \tau_1$. Assuming $\tau_{nh} = \tau_1/\mathcal{P}$, we would then find $\alpha = \mathcal{P} - 1$, and hence no speedup at all with $S = 1$. The value $\alpha = \mathcal{P} - 1$ is maximum value that can be expected realistically. This situation occurs if traditional time-stepping methods, instead of exponential integrators, are used for calculating the exponential propagation (5.30). In other words, the homogeneous part of (5.28) is not solved substantially faster than the nonhomogeneous part. However, in our case we use EBK and this effect does not play a role. This is a key motivation for adhering to EBK as a method for time-parallel integration.

The wall-clock times of the EBK and the CN method from the experiment of the Taylor–Green vortex, see Section 5.3.1, are listed in Table 5.2. Those computations are performed for $\Delta T = 1$ and $Re = 100$. The time of the EBK method corresponds to τ_1 here. The EBK method is generally slower than the CN method: a factor 7 for $\Delta x = 1/32$ for example. In other words, the parallel speedup needs to be at least 7 for the EBK method to break even with the CN method in terms of computation time. The parallel speedup is illustrated for several $\alpha(\mathcal{P})$ in Fig. 5.9, taking $\alpha(\mathcal{P}) = c(\mathcal{P} - 1)$ and considering the dependence of S on a constant c . The case $\alpha = 0$ corresponds to ideal speedup, i.e., the homogeneous parts have zero cost. As the cost of the homogeneous part increases with respect to the nonhomogeneous ones, α increases and the parallel efficiency decreases. The curves move away from the ideal line as α increases. This figure illustrates that the EBK method can only break even when α is sufficiently small.

When we assume that $\alpha = c(\mathcal{P} - 1)$, where $c > 0$ is a constant, we can examine the speedup as $\mathcal{P}$ approaches infinity. In that case, the limit of the speedup is

$$\lim_{\mathcal{P} \rightarrow \infty} S = \frac{\mathcal{P}}{1 + c(\mathcal{P} - 1)} = \frac{1}{c}. \quad (5.34)$$

For the current example, this means that α should approximately be below $\alpha = 0.08(\mathcal{P} - 1)$, for the possibility of the EBK method to break even with the CN

method. The actual α can be estimated dividing the time interval into subintervals and limit the nonhomogeneity to the first subinterval. We measured α in a numerical experiment with a Taylor–Green vortex ($Re = 100$) and estimated the corresponding speedup, see Fig 5.9. In this figure, we show the actual speedup measurements and plot the speedup for several theoretical values of α . The speedup can be improved slightly by averaging the Jacobian matrix over the complete time interval, instead of averaging over each subinterval individually. The homogeneous part can then be solved by the EBK method as a single problem, because the Jacobian matrix does not change per subinterval. The parallel efficiency appears to remain relatively low, because the homogeneous parts are not solved substantially faster than the nonhomogeneous parts. The homogeneous parts are relatively expensive, even without a source term, because the application of the projection P at every Krylov iteration is expensive. The EBK method may be improved by the so-called shift-and-invert (SaI) technique [11]. The SaI-technique is based on a rational Krylov subspace, which improves the convergence by emphasizing small eigenvalues [127, 170]. The expected faster convergence would make the EBK method more competitive with the CN method already at lower numbers of processors. Also, we might get closer to the ideal situation in which $\tau_h \ll \tau_{nh}$. This would increase α and therefore reduce the parallel speedup. The realization of the SaI-technique is not straightforward, because of the differential-algebraic nature of the Navier–Stokes equation. The possible implementation and improvements of the SaI-technique will be explored in future studies.

Table 5.2: Computation time of the serial execution of the EBK and the CN method for the Taylor–Green vortex at $Re = 100$ and several spatial resolutions. The (maximum) mesh Reynolds number Re_h is also indicated.

Δx	Re_h	EBK	CN
1/32	1.56	9.00 s	1.24 s
1/64	0.78	26.3 s	3.71 s
1/128	0.39	194 s	21.2 s

5.5 Conclusions

The EBK method is an exponential time integration method which we have extended in this chapter to the incompressible Navier–Stokes equation. The pressure is treated by introducing a divergence-free projection operator, whereas the nonlinear convection term is taken care of by a process of outer iterations. In these iterations, the nonlinear term is handled as a source term, evaluated with the solution of the previous iteration (on the complete time interval). This approach is very similar to waveform relaxations.

In several numerical experiments, we demonstrate that the EBK method can

Figure 5.9: Parallel speedup for several $\alpha(\mathcal{P})$, where $\alpha = \tau_h/\tau_{nh}$. The dashed line indicates the break-even point of the EBK method with the CN method. $\circ$ Jacobian matrix averaged over each subinterval. $\times$ Jacobian matrix averaged over the complete time interval.

produce highly accurate time solutions. The spatial discretization of the Navier–Stokes equation is carried out by a finite element method. Furthermore, a major advantage is that the EBK method is suitable for parallel-in-time simulations. With a simplified model we have analyzed the potential speedup. We establish that parallel speedup for parallel-in-time simulations of the incompressible Navier–Stokes equation is indeed feasible with the EBK method. The efficiency of the EBK method itself might be further improved by using a rational Krylov subspace [10, 70] with the shift-and-invert technique [170]. Future research will be directed toward an actual application of the time-parallel method, measuring the performance benefit as function of problem size and number of processors.

Conclusions

The aim of this thesis was to explore the possibility of parallel-in-time simulation of turbulent Rayleigh–Bénard (RB) convection. The first two chapters are focussed on direct numerical simulations (DNS) of turbulent RB convection. The latter two chapters discuss contributions to the development of a parallel-in-time method for the incompressible Navier–Stokes (NS) equations.

In Chapter 2, we studied the effect of rotation on the scaling of the Nusselt number on RB convection in a cylinder with aspect ratio $\Gamma = 1$. We also demonstrated the application of a spectral element method to DNS of RB convection. Rotation was found to increase heat transfer, but it has little effect on the scaling of Nu at higher Rayleigh numbers. The simulations were performed with the spectral element method from the opensource code Nek5000. With a grid convergence test, we showed that the spatial resolution was sufficient to calculate the Nusselt number accurately at a high rotation rate ($Ro = 0.09$). Furthermore, we found good agreement with data from literature that were obtained with a frequently used finite difference method. We calculated the Nusselt number for a range of Rossby numbers ($0.09 \leq Ro \leq 1.08$) and Rayleigh numbers ($10^6 \leq Ra \leq 10^9$). The scaling of Nu with Ra in the nonrotating case, $Ro = \infty$, matched well the Grossmann-Lohse theory. In this parameter range a typical approximation of $Nu \sim Ra^{2/7}$ holds. Our findings suggest that this approximation also holds in rotating flows at higher Rayleigh numbers ($Ra \gtrsim 10^8$). It should be noted that the simulations were performed for a relatively small range of Rayleigh numbers. Additional simulations are recommended to determine how the scaling of Nu continues at higher Ra .

In Chapter 3, we compared several numerical methods than can be used for DNS of RB convection. We demonstrated that specialized codes, designed for structured grids in simple geometries, are the most efficient in terms of accuracy against computational cost. We compared four different codes: a second order finite difference method (AFID/RBflow), a fourth order finite volume method

(Goldfish), an eighth order spectral element method (Nek5000), and a second order finite volume method (OpenFOAM). We performed simulations in three different geometries: a periodic domain, and a cubic and cylindrical container. All geometries had a unit aspect ratio. The main parameter of interest was the heat transport in the turbulent flow, measured by the Nusselt number. The accuracy of the numerical methods was determined using fully converged simulations at high resolution as reference. Significant differences were observed in the computational cost, i.e., CPU hours at a given number of grid points. AFID/RBflow for example, was found to be about a hundred times faster than Nek5000 at a given spatial resolution. The specialized codes clearly outperformed the general codes in terms of computing time per grid point. In other words, a considerable saving in computation time can be achieved by optimizing a code for simple geometries compared to codes for general complex geometries. The difference is largely caused by the way the Poisson equation for the pressure is solved. For structured meshes with one or more periodic directions fast Fourier transform (FFT) solvers can be used, which are arguably the most efficient choice for that application. For unstructured meshes, iterative methods, e.g., the GMRES or the conjugate gradient method, have to be used. General purpose codes, such as Nek5000 and OpenFOAM, offer flexibility in geometry, but are therefore more expensive than specialized codes.

However, the differences in computational efficiency were smaller when taking the achieved accuracy into account. AFID/RBflow and Goldfish were the most efficient codes, but Nek5000 followed relatively closely, being about a factor of 5–10 more expensive for achieving the same level of accuracy. This illustrates the possible benefit of higher-order discretization methods that would also be applicable to general conditions. It is possible that higher-order methods are more efficient when considering higher moment statistics of the flow. This could be an interesting subject for future studies.

Regarding parallelization in time, we introduced a parallel-in-time method for nonlinear partial differential equations in Chapter 4. Numerical experiments showed that parallel speedup with this method is feasible. Our time integration method assumes a semi-discrete system in which the equations are discretized in space first. This method is therefore compatible with many spatial discretization techniques, including the ones analysed in Chapters 2 and 3. A parallel-in-time code could for example be based on the spectral element method.

The method we proposed, is an implementation of the Paraexp method based on the exponential block Krylov (EBK) method. The Paraexp-EBK (PEBK) method is extended to solve nonlinear PDEs iteratively using a waveform relaxation method. The nonlinear term is treated as a source term in a nonhomogeneous initial value problem. At each iteration the source term is updated with the latest solution. The convergence of the nonlinear iterations are improved considerably by adding a correction term based on a linearization of the nonlinear term. At each iteration, the nonhomogeneous initial value problem can then be decoupled into independent subproblems that can be solved parallel in time. Each subproblem only treats a part of the inhomogeneity of the original problem. In the original

Paraexp algorithm, the nonhomogeneous subproblems are solved with a conventional time integration method. On the contrary, we use the EBK method to solve both the homogeneous and the nonhomogeneous parts of the subproblems.

To demonstrate the parallelization concept for linear PDEs, we tested PEBK on the advection-diffusion equation. The method was shown to be efficient, even in cases with little or no diffusion. We compared PEBK with a Crank–Nicolson scheme parallelized with Paraexp. Here, PEBK showed a consistently higher parallel efficiency. We also demonstrated PEBK for nonlinear PDEs by solving the viscous Burgers equation. Also in this case, good parallel efficiency was observed. Decreasing the viscosity coefficient did not have a significant effect on the parallel efficiency.

In Chapter 5, we discussed how the EBK method can be applied to the incompressible NS equations. We demonstrated the time integration method for several two-dimensional test cases and explored the potential of a parallelization in time. In our approach, we reformulated the NS equations by applying a “divergence-free” projection to the momentum equation, such that the continuity equation is no longer explicitly arising as an algebraic constraint. The projection by itself ensures a solenoidal or divergence-free velocity field (in a discrete sense). The nonlinear convective term was treated iteratively, like for the Burgers equation in Chapter 4. We tested the EBK method for several numerical experiments, including the Taylor–Green vortex and lid-driven cavity flow. A finite element method was used for the discretization in the spatial dimensions.

Parallelization in time is possible with the framework proposed in Chapter 4. With a simplified model, we analyzed the potential parallel speedup, and established that (limited) speedup is indeed possible. As for most Krylov subspace methods, it is highly recommended to use some kind of acceleration of the EBK method to ensure fast convergence in practical applications. One way would be to use a rational Krylov subspace with the shift-and-invert technique [11, 127, 170]. This method ensures a practically mesh-independent convergence behaviour, which becomes essential for meshes with very high spatial resolution or with strong local mesh refinement.

The exponential time integration method from Chapter 5 provides the basis for a time-parallel method for incompressible flows with a parallelization as discussed in Chapter 4. The extension to three-dimensional problems is straightforward because we follow the method of lines. In other words, the partial differential equation is discretized in the spatial dimensions first. The time integration method can then be applied to the corresponding semi-discrete system with time as the remaining continuous variable. The EBK method turns out to be relatively expensive in this case, because for every Krylov iteration a saddle-point problem or an equivalent Poisson equation needs to be solved. Also, the number of iterations increases when the spatial resolution increases, i.e., when the condition number of the matrix increases. Accelerating the Krylov method (using the shift-and-invert technique) could remedy the situation, and reduce the total number of Krylov iterations. Not only would that accelerate the convergence, it also leads to (perhaps

almost) mesh-independent convergence behavior. As DNS of turbulent flows usually requires very fine meshes, acceleration of the EBK method is likely to be an essential tool. Using the shift-and-invert technique is a possibility, but its application in this case is not trivial, because the projection operator for incompressible flows somewhat complicates the procedure. Future study is recommended to study the acceleration of Krylov subspace methods for exponential integrators of the incompressible NS equations. Another point is the inclusion of the temperature equation and its coupling with the momentum equation following the Boussinesq approximation. Solving the temperature equation is necessary for the simulation of Rayleigh–Bénard convection. The temperature could be included by expanding the semi-discrete system with the discretised transport equation for the temperature and introducing the temperature as an additional state variable. The coupling with velocity field can be taken care of with our approach of waveform relaxation. Finally, a variant of Paraexp for nonlinear initial value problems has recently been proposed in [55]. A comparison with our method would be interesting as well.

References

- [1] M. Afanasjew, M. Eiermann, O. G. Ernst, and S. Güttel. Implementation of a restarted Krylov subspace method for the evaluation of matrix functions. *Linear Algebra Appl.*, 429:2293–2314, 2008.
- [2] G. Ahlers, E. Brown, F. F. Araujo, D. Funfschilling, S. Grossmann, and D. Lohse. Non-Oberbeck–Boussinesq effects in strongly turbulent Rayleigh–Bénard convection. *J. Fluid Mech.*, 569:409–445, 2006.
- [3] G. Ahlers, S. Grossmann, and D. Lohse. Heat transfer and large scale dynamics in turbulent Rayleigh–Bénard convection. *Rev. Mod. Phys.*, 81(2):503, 2009.
- [4] A. Axelsson and J. Verwer. Boundary value techniques for initial value problems in ordinary differential equations. *Math. Comp.*, 45(171):153–171, 1985.
- [5] J. Bailon-Cuba, M. S. Emran, and J. Schumacher. Aspect ratio dependence of heat transfer and large-scale flow in turbulent convection. *J. Fluid Mech.*, 655:152–173, 2010.
- [6] A. Bellen and M. Zennaro. Parallel algorithms for initial-value problems for difference and differential equations. *J. Comput. Appl. Math.*, 25(3):341–350, 1989.
- [7] H. Bénard. Les tourbillons cellulaires dans une nappe liquide. *Rev. Gén. Sci. Pures Appl.*, 11:1261–1274, 1900.
- [8] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numer.*, 14:1–137, 2005.

- [9] M. Benzi and Z. Wang. A parallel implementation of the modified augmented Lagrangian preconditioner for the incompressible Navier-Stokes equations. *Numer. Algorithms*, 64(1):73–84, 2013.
- [10] M. Berljafa and S. Güttel. Generalized rational Krylov decompositions with an application to rational approximation. *SIAM J. Matrix Anal. Appl.*, 36(2):894–916, 2015.
- [11] M. A. Botchev. A block Krylov subspace time-exact solution method for linear ordinary differential equation systems. *Numer. Linear Algebra Appl.*, 20(4):557–574, 2013.
- [12] M. A. Botchev, I. Faragó, and A. Havasi. Testing weighted splitting schemes on a one-column transport-chemistry model. *Int. J. Environ. Pollut.*, 22(1/2):3–16, 2004.
- [13] M. A. Botchev, V. Grimm, and M. Hochbruck. Residual, restarting, and Richardson iteration for the matrix exponential. *SIAM J. Sci. Comput.*, 35(3):A1376–A1397, 2013.
- [14] M. A. Botchev, I. V. Oseledets, and E. E. Tyrtysnikov. Iterative across-time solution of linear differential equations: Krylov subspace versus waveform relaxation. *Comput. Math. Appl.*, 67(12):2088–2098, 2014.
- [15] M. A. Botchev, G. L. G. Sleijpen, and A. Sopaheluwakan. An SVD-approach to Jacobi–Davidson solution of nonlinear Helmholtz eigenvalue problems. *Numer. Linear Algebra Appl.*, 431(3):427–440, 2009.
- [16] M. A. Botchev and H. A. van der Vorst. A parallel nearly implicit scheme. *J. Comput. Appl. Math.*, 137:229–243, 2001.
- [17] O. Botella and R. Peyret. Benchmark spectral results on the lid-driven cavity flow. *Comput. & Fluids*, 27(4):421–433, 1998.
- [18] J. Boussinesq. *Théorie analytique de la chaleur: mise en harmonie avec la thermodynamique et avec la théorie mécanique de la lumière*, volume 2. Gauthier–Villars, 1903.
- [19] C.-H. Bruneau and M. Saad. The 2D lid-driven cavity problem revisited. *Comput. & Fluids*, 35(3):326–348, 2006.
- [20] K. Burrage. *Parallel and sequential methods for ordinary differential equations*. Clarendon Press, New York, 1995.
- [21] B. Castaing, G. Gunaratne, F. Heslot, L. Kadanoff, A. Libchaber, S. Thomae, X.-Z. Wu, S. Zaleski, and G. Zanetti. Scaling of hard thermal turbulence in Rayleigh–Bénard convection. *J. Fluid Mech.*, 204:1–30, 1989.

- [22] A. Cella and M. Lucchesi. Space-time finite elements for the wave propagation problem. *Meccanica*, 10(3):168–170, 1975.
- [23] E. Celledoni and I. Moret. A Krylov projection method for systems of ODEs. *Appl. Numer. Math.*, 24(2):365–378, 1997.
- [24] M. Chandra and M. K. Verma. Dynamics and symmetries of flow reversals in turbulent convection. *Phys. Rev. E*, 83(6):067303, 2011.
- [25] S. Chandrasekhar. *Hydrodynamic and hydromagnetic stability*. Courier Dover Publications, 2013.
- [26] P. Chartier and B. Philippe. A parallel shooting technique for solving dissipative ODE's. *Computing*, 51(3):209–236, 1993.
- [27] F. Chillà and J. Schumacher. New perspectives in turbulent Rayleigh–Bénard convection. *Eur. Phys. J. E*, 35(7):1–25, 2012.
- [28] A. J. Chorin. Numerical solution of the Navier–Stokes equations. *Math. Comp.*, 22(104):745–762, 1968.
- [29] A. J. Chorin. On the convergence of discrete approximations to the Navier–Stokes equations. *Math. Comp.*, 23(106):341–353, 1969.
- [30] A. J. Chorin and O. H. Hald. Viscosity-dependent inertial spectra of the Burgers and Korteweg–deVries–Burgers equations. *Proceedings of the National Academy of Sciences of the United States of America*, 102(11):3921–3923, 2005.
- [31] A. J. Christlieb, C. B. Macdonald, and B. W. Ong. Parallel high-order integrators. *SIAM J. Sci. Comput.*, 32(2):818–835, 2010.
- [32] J. J. B. de Swart. *Parallel Software for Implicit Differential Equations*. PhD thesis, University of Amsterdam, Nov. 1997.
- [33] M. Deville, P. Fischer, and E. Mund. *High-Order Methods for Incompressible Fluid Flow*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004.
- [34] S. V. Dolgov, B. N. Khoromskij, and I. V. Oseledets. Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the Fokker–Planck equation. *SIAM J. Sci. Comput.*, 34(6):A3016–A3038, 2012.
- [35] J. Dongarra et al. The international exascale software project roadmap. *Int. J. High Perform. Comput. Appl.*, page 1094342010391989, 2011.
- [36] V. Druskin and L. Knizhnerman. Extended Krylov subspaces: approximation of the matrix square root and related functions. *SIAM J. Matrix Anal. Appl.*, 19(3):755–771, 1998.

- [37] W. Edwards, L. Tuckerman, R. Friesner, and D. Sorensen. Krylov methods for the incompressible Navier–Stokes equations. *J. Comput. Phys.*, 110(1):82–102, 1994.
- [38] M. Eiermann, O. G. Ernst, and S. Güttel. Deflated restarting for matrix functions. *SIAM J. Matrix Anal. Appl.*, 32(2):621–641, 2011.
- [39] V. W. Ekman. On the influence of the Earth’s rotation on ocean currents. *Ark. Mat. Astron. Fys.*, 2:1–53, 1905.
- [40] G. K. El Khoury, P. Schlatter, A. Noorani, P. F. Fischer, G. Brethouwer, and A. V. Johansson. Direct numerical simulation of turbulent pipe flow at moderately high Reynolds numbers. *Flow Turbul. Combust.*, 91(3):475–495, 2013.
- [41] H. Elman and D. Silvester. Fast nonsymmetric iterations and preconditioning for Navier–Stokes equations. *SIAM J. Sci. Comput.*, 17(1):33–46, 1996.
- [42] H. C. Elman, A. Ramage, and D. J. Silvester. Algorithm 866: IFISS, a Matlab toolbox for modelling incompressible flow. *ACM Trans. Math. Softw.*, 33(2):14, 2007.
- [43] H. C. Elman, A. Ramage, and D. J. Silvester. IFISS: A computational laboratory for investigating incompressible flow problems. *SIAM Rev.*, 56(2):261–273, 2014.
- [44] H. C. Elman, D. J. Silvester, and A. J. Wathen. Performance and analysis of saddle point preconditioners for the discrete steady-state Navier–Stokes equations. *Numer. Math.*, 90(4):665–688, 2002.
- [45] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press, 2014.
- [46] M. Emmett and M. Minion. Toward an efficient parallel in time method for partial differential equations. *Comm. Appl. Math. Comput. Sci.*, 7(1):105–132, 2012.
- [47] C. Farhat, J. Cortial, C. Dastillung, and H. Bavestrello. Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses. *Int. J. Numer. Methods Eng.*, 67(5):697–724, 2006.
- [48] P. F. Fischer. An overlapping Schwarz method for spectral element solution of the incompressible Navier–Stokes equations. *J. Comput. Phys.*, 133(1):84–101, 1997.
- [49] P. F. Fischer, F. Hecht, and Y. Maday. A Parareal in time semi-implicit approximation of the Navier–Stokes equations. In *Domain decomposition methods in science and engineering*, pages 433–440. Springer, 2005.

- [50] H. Fujita and T. Kato. On the Navier–Stokes initial value problem. I. *Arch. Ration. Mech. Anal.*, 16(4):269–315, 1964.
- [51] M. J. Gander. Overlapping Schwarz for linear and nonlinear parabolic problems. In *Proceedings of the 9th International Conference on Domain Decomposition*, pages 97–104, 1996.
- [52] M. J. Gander. Analysis of the parareal algorithm applied to hyperbolic problems using characteristics. *Bol. Soc. Esp. Mat. Apl.*, 42:21–35, 2008.
- [53] M. J. Gander. *50 Years of Time Parallel Time Integration*, pages 69–113. Springer International Publishing, Cham, 2015.
- [54] M. J. Gander and S. Güttel. PARAEXP: A parallel integrator for linear initial-value problems. *SIAM J. Sci. Comput.*, 35(2):C123–C142, 2013.
- [55] M. J. Gander, S. Güttel, and M. Petcu. A nonlinear ParaExp algorithm. MIMS EPrint: 2017.17.
- [56] M. J. Gander, L. Halpern, and F. Nataf. Optimal convergence for overlapping and non-overlapping schwarz waveform relaxation. In *Proceedings of the 11th International Conference on Domain Decomposition*, 1999.
- [57] M. J. Gander and M. Petcu. Analysis of a Krylov subspace enhanced parareal algorithm for linear problems. In *Paris-Sud Working Group on Modelling and Scientific Computing 2007–2008*, volume 25 of *ESAIM Proc.*, pages 114–129. EDP Sci., Les Ulis, 2008.
- [58] M. J. Gander and S. Vandewalle. Analysis of the Parareal time-parallel time-integration method. *SIAM J. Sci. Comput.*, 29(2):556–578, 2007.
- [59] G. J. Gassner and A. D. Beck. On the accuracy of high-order discretizations for underresolved turbulence simulations. *Theoret. Comput. Fluid Dyn.*, 27(3):221–237, 2013.
- [60] C. W. Gear. Parallel methods for ordinary differential equations. *Calcolo*, 25(1):1–20, 1988.
- [61] B. Gebhart, Y. Jaluria, R. L. Mahajan, and B. Sammakia. *Buoyancy-Induced Flows and Transport*. Hemisphere, New York, 1988.
- [62] U. Ghia, K. Ghia, and C. Shin. High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method. *J. Comput. Phys.*, 48(3):387–411, 1982.
- [63] T. Gökler and V. Grimm. Uniform approximation of φ -functions in exponential integrators by a rational Krylov subspace method with simple poles. *SIAM J. Matrix Anal. Appl.*, 35(4):1467–1489, 2014.

- [64] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.
- [65] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. How to find a good submatrix. In *Matrix methods: theory, algorithms and applications*, pages 247–256. World Sci. Publ., Hackensack, NJ, 2010.
- [66] D. D. Gray and A. Giorgini. The validity of the Boussinesq approximation for liquids and gases. *Int. J. Heat Mass Transfer*, 19(5):545–551, 1976.
- [67] P. M. Gresho and R. L. Lee. Don’t suppress the wiggles—they’re telling you something! *Comput. & Fluids*, 9(2):223–253, 1981.
- [68] S. Grossmann and D. Lohse. Scaling in thermal convection: a unifying theory. *J. Fluid Mech.*, 407:27–56, 2000.
- [69] S. Grossmann and D. Lohse. Multiple scaling in the ultimate regime of thermal convection. *Phys. Fluids*, 23(4):045108, 2011.
- [70] S. Güttel and L. Knizhnerman. A black-box rational Arnoldi variant for Cauchy-Stieltjes matrix functions. *BIT*, 53(3):595–616, 2013.
- [71] W. Hackbusch. Parabolic multigrid methods. In *Computing methods in applied sciences and engineering, VI (Versailles, 1983)*, pages 189–197. North-Holland, Amsterdam, 1984.
- [72] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, 8(12):2182, 1965.
- [73] D. L. Hartmann, L. A. Moy, and Q. Fu. Tropical convection and the energy balance at the top of the atmosphere. *J. Clim.*, 14(24):4495–4511, 2001.
- [74] X. He, D. Funfschilling, H. Nobach, E. Bodenschatz, and G. Ahlers. Transition to the ultimate state of turbulent Rayleigh-Bénard convection. *Phys. Rev. Lett.*, 108(2):024502, 2012.
- [75] F. Hébert, R. Hufschmid, J. Scheel, and G. Ahlers. Onset of Rayleigh-Bénard convection in cylindrical containers. *Phys. Rev. E*, 81:046318, Apr 2010.
- [76] M. Hochbruck and J. Niehoff. Approximation of matrix operators applied to multiple vectors. *Math. Comput. Simulation*, 79(4):1270–1283, 2008.
- [77] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numer.*, 19:209–286, 2010.
- [78] E. Hopf. The partial differential equation $u_t + uu_x = \mu u_{xx}$. *Comm. Pure Appl. Math.*, 3(3):201–230, 1950.

- [79] R. A. Horn and C. R. Johnson. Matrix Analysis. *Cambridge University Press*, 1986.
- [80] R. A. Horn and C. R. Johnson. Topics in Matrix Analysis. *Cambridge University Press*, 1991.
- [81] S. Horn and O. Shishkina. Rotating non-Oberbeck-Boussinesq Rayleigh-Bénard convection in water. *Phys. Fluids (1994-present)*, 26(5), 2014.
- [82] G. Horton and S. Vandewalle. A space-time multigrid method for parabolic partial differential equations. *SIAM J. Sci. Comput.*, 16(4):848–864, 1995.
- [83] T. J. R. Hughes and G. M. Hulbert. Space-time finite element methods for elastodynamics: formulations and error estimates. *Comput. Methods Appl. Mech. Eng.*, 66(3):339–363, 1988.
- [84] W. Hundsdorfer and J. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*, volume 33 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2003.
- [85] R. I. Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *J. Comput. Phys.*, 62(1):40–65, 1986.
- [86] Y. Jaluria. *Natural Convection: Heat and Mass Transfer*. Pergamon, New York, 1980.
- [87] K. Julien, S. Legg, J. McWilliams, and J. Werne. Hard turbulence in rotating Rayleigh-Bénard convection. *Phys. Rev. E*, 53(6):R5557, 1996.
- [88] K. Julien, S. Legg, J. McWilliams, and J. Werne. Rapidly rotating turbulent Rayleigh-Bénard convection. *J. Fluid Mech.*, 322:243–273, 1996.
- [89] L. P. Kadanoff. Turbulent heat flow: Structures and scaling. *Phys. Today*, 54(8):34–39, 2001.
- [90] S. Kakaç, W. Aung, and R. Viskanta. *Natural Convection: Fundamentals and Applications*. Taylor & Francis, Washington, D.C., 1985.
- [91] G. E. Karniadakis, M. Israeli, and S. A. Orszag. High-order splitting methods for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 97(2):414–443, 1991.
- [92] D. A. Kay, P. M. Gresho, D. F. Griffiths, and D. J. Silvester. Adaptive time-stepping for incompressible flow part ii: Navier-Stokes equations. *SIAM J. Sci. Comput.*, 32(1):111–128, 2010.
- [93] R. M. Kerr. Rayleigh number scaling in numerical convection. *J. Fluid Mech.*, 310:139–179, 1996.

- [94] E. M. King and J. M. Aurnou. Thermal evidence for Taylor columns in turbulent rotating Rayleigh–Bénard convection. *Phys. Rev. E*, 85(1):016313, 2012.
- [95] E. M. King, K. M. Soderlund, U. R. Christensen, J. Wicht, and J. M. Aurnou. Convective heat transfer in planetary dynamo models. *Geochem. Geophys. Geosyst.*, 11(6), 2010.
- [96] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM J. Matrix Anal. Appl.*, 29(2):434–454, 2007.
- [97] G. L. Kooij, M. A. Botchev, and B. J. Geurts. Direct numerical simulation of Nusselt number scaling in rotating Rayleigh–Bénard convection. *Int. J. Heat Fluid Flow*, 55:26–33, 2015.
- [98] G. L. Kooij, M. A. Botchev, and B. J. Geurts. A block Krylov subspace implementation of the time-parallel Paraexp method and its extension for nonlinear partial differential equations. *J. Comput. Appl. Math.*, 316:229–246, 2017.
- [99] R. H. Kraichnan. Turbulent thermal convection at arbitrary Prandtl number. *Phys. Fluids*, 5(11):1374–1389, 1962.
- [100] R. Krause and D. Ruprecht. Hybrid space–time parallel solution of Burgers’ equation. In *Domain Decomposition Methods in Science and Engineering XXI*, pages 647–655. Springer, 2014.
- [101] P. Kundu and I. Cohen. *Fluid Mechanics*. Elsevier Science, 2010.
- [102] R. P. Kunnen, R. J. Stevens, J. Overkamp, C. Sun, G. F. van Heijst, and H. J. Clercx. The role of Stewartson and Ekman layers in turbulent rotating Rayleigh–Bénard convection. *J. Fluid Mech.*, 688:422–442, 2011.
- [103] R. P. J. Kunnen. *Turbulent rotating convection*. PhD thesis, TU Eindhoven, 2008.
- [104] R. P. J. Kunnen, H. J. H. Clercx, and B. J. Geurts. Heat flux intensification by vortical flow localization in rotating convection. *Phys. Rev. E*, 74:056306, Nov 2006.
- [105] R. P. J. Kunnen, H. J. H. Clercx, and B. J. Geurts. Breakdown of large-scale circulation in turbulent rotating convection. *Europhys. Lett.*, 84(2):24001, 2008.
- [106] R. P. J. Kunnen, B. J. Geurts, and H. J. H. Clercx. Experimental and numerical investigation of turbulent convection in a rotating cylinder. *J. Fluid Mech.*, 642:445–476, 1 2010.

- [107] L. D. Landau and E. M. Lifshitz. *Fluid Mechanics*, volume 6 of *Course of Theoretical Physics*. Pergamon Press, 2 edition, 1987.
- [108] E. Lelarasmee, A. E. Ruehli, and A. Sangiovanni-Vincentelli. The waveform relaxation method for time-domain analysis of large scale integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1(3):131–145, July 1982.
- [109] G. Li. A block variant of the GMRES method on massively parallel processors. *Parallel Comput.*, 23(8):1005–1019, 1997.
- [110] E. Lindelöf. Sur l’application des méthodes d’approximations successives à l’étude des intégrales réelles des équations différentielles ordinaires. *J. Math. Pures Appl.*, 10:117–128, 1894.
- [111] J.-L. Lions. *Équations différentielles opérationnelles et problèmes aux limites*. Die Grundlehren der mathematischen Wissenschaften, Bd. 111. Springer-Verlag, Berlin-Göttingen-Heidelberg, 1961.
- [112] J.-L. Lions, Y. Maday, and G. Turinici. Résolution d’edp par un schéma en temps “pararéel”. *C. R. Acad. Sci. Paris Sér. I Math.*, 332(7):661–668, 2001.
- [113] Y. Liu and R. E. Ecke. Heat transport scaling in turbulent Rayleigh-Bénard convection: effects of rotation and Prandtl number. *Phys. Rev. Lett.*, 79(12):2257, 1997.
- [114] Y. Liu and R. E. Ecke. Heat transport measurements in turbulent rotating Rayleigh-Bénard convection. *Phys. Rev. E*, 80(3):036314, 2009.
- [115] D. Lohse and F. Toschi. Ultimate state of thermal convection. *Phys. Rev. Lett.*, 90(3):034502, 2003.
- [116] D. Lohse and K.-Q. Xia. Small-scale properties of turbulent Rayleigh-Bénard convection. *Ann. Rev. Fluid Mech.*, 42:335–364, 2010.
- [117] C. Lubich and A. Ostermann. Multi-grid dynamic iteration for parabolic equations. *BIT Numer. Math.*, 27:216–234, 1987.
- [118] Y. Maday, D. Meiron, A. T. Patera, and E. M. Rønquist. Analysis of iterative methods for the steady and unsteady Stokes problem: Application to spectral element discretizations. *SIAM J. Sci. Comput.*, 14(2):310–337, 1993.
- [119] Y. Maday and E. M. Rønquist. Parallelization in time through tensor-product space-time solvers. *C. R. Math. Acad. Sci. Paris*, 346(1-2):113–118, 2008.

- [120] W. V. R. Malkus. The heat transport and spectrum of thermal turbulence. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 225, pages 196–212. The Royal Society, 1954.
- [121] J. Malm, P. Schlatter, P. F. Fischer, and D. S. Henningson. Stabilization of the spectral element method in convection dominated flows by recovery of skew-symmetry. *J. Sci. Comput.*, 57(2):254–277, 2013.
- [122] U. Mikkala and O. Nevanlinna. Convergence of dynamic iteration methods for initial value problems. *SIAM J. Sci. Stat. Comput.*, 8(4):459–482, 1987.
- [123] M. S. Miesch. Large-scale dynamics of the convection zone and tachocline. *Living Rev. Sol. Phys.*, 2(1):1–139, 2005.
- [124] A. Y. Mikhalev and I. V. Oseledets. Iterative representing set selection for nested cross approximation. *Numer. Linear Algebra Appl.*, 23(2):230–248, 2016.
- [125] M. Minion. A hybrid parareal spectral deferred corrections method. *Comm. Appl. Math. Comput. Sci.*, 5(2):265–301, 2011.
- [126] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev.*, 45(1):3–49, 2003.
- [127] I. Moret and P. Novati. RD rational approximations of the matrix exponential. *BIT*, 44:595–615, 2004.
- [128] K. W. Morton and D. F. Mayers. *Numerical Solution of Partial Differential Equations: An Introduction*. Cambridge University Press, 2005.
- [129] M. Neumüller. *Space-time methods: fast solvers and applications*. PhD thesis, University of Graz, 2013.
- [130] O. Nevanlinna. Remarks on Picard-Lindelöf iteration. *BIT Numer. Math.*, 29(2):328–346, 1989.
- [131] C. K. Newman. *Exponential Integrators for the Incompressible Navier–Stokes Equations*. PhD thesis, Virginia Polytechnic Institute and State University, 2004.
- [132] A. R. Newton and A. L. Sangiovanni-Vincentelli. Relaxation-based electrical simulation. *IEEE Transactions on Electron Devices*, 30(9):1184–1207, 1983.
- [133] J. Niehoff. *Projektionsverfahren zur Approximation von Matrixfunktionen mit Anwendungen auf die Implementierung exponentieller Integratoren*. PhD thesis, Heinrich-Heine-Universität Düsseldorf, 2006.

- [134] J. Niemela, S. Babuin, and K. Sreenivasan. Turbulent rotating convection at high Rayleigh and Taylor numbers. *J. Fluid Mech.*, 649:509–522, 2010.
- [135] J. Nievergelt. Parallel methods for integrating ordinary differential equations. *Commun. ACM*, 7(12):731–733, Dec. 1964.
- [136] A. Nikishin and A. Y. Yeremin. Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers, I: General iterative scheme. *SIAM J. Matrix Anal. Appl.*, 16(4):1135–1153, 1995.
- [137] A. Oberbeck. Ueber die Wärmeleitung der Flüssigkeiten bei Berücksichtigung der Strömungen infolge von Temperaturdifferenzen. *Ann. Phys.*, 243(6):271–292, 1879.
- [138] J. Ohlsson, P. Schlatter, P. F. Fischer, and D. S. Henningson. Direct numerical simulation of separated flow in a three-dimensional diffuser. *J. Fluid Mech.*, 650:307–318, 2010.
- [139] E. Picard. Sur l’application des méthodes d’approximations successives à l’étude de certaines équations différentielles ordinaires. *J. Math. Pures Appl.*, 9:217–272, 1893.
- [140] S. B. Pope. *Turbulent Flows*. Cambridge University Press, New York, 2000.
- [141] L. Rayleigh. On convection currents in a horizontal layer of fluid, when the higher temperature is on the under side. *Phil. Mag.*, 32(192):529–546, 1916.
- [142] E. M. Rønquist. Convection treatment using spectral elements of different order. *Int. J. Numer. Methods Fluids*, 22(4):241–264, 1996.
- [143] H. T. Rossby. A study of Bénard convection with and without rotation. *J. Fluid Mech.*, 36:309–335, 1969.
- [144] D. Ruprecht and R. Krause. Explicit parallel-in-time integration of a linear acoustic-advection system. *Comput. & Fluids*, 59:72–83, 2012.
- [145] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, 2nd edition, 2003.
- [146] Y. Saad and B. Semeraro. Application of Krylov exponential propagation to fluid dynamics equations. Technical Report 91.06, Research Institute for Advanced Computer Science, 1991.
- [147] P. Saha, J. Stadel, and S. Tremaine. A parallel integration method for solar system dynamics. *Astron. J.*, 114:409–415, 1997.
- [148] J. D. Scheel, M. S. Emran, and J. Schumacher. Resolving the fine-scale structure in turbulent Rayleigh–Bénard convection. *New J. Phys.*, 15(11):113063, 2013.

- [149] J. C. Schulze, P. J. Schmid, and J. L. Sesterhenn. Exponential time integration using Krylov subspaces. *Int. J. Numer. Methods Fluids*, 60(6):591–609, 2009.
- [150] D. Sheen, I. H. Sloan, and V. Thomée. A parallel method for time discretization of parabolic equations based on Laplace transformation and quadrature. *IMA J. Numer. Anal.*, 23(2):269–299, 2003.
- [151] O. Shishkina. The Neumann stability of high-order symmetric schemes for convection-diffusion problems. *Sib. Math. J.*, 48(6):1141–1146, 2007.
- [152] O. Shishkina and S. Horn. Thermal convection in inclined cylindrical containers. *J. Fluid Mech.*, 790:R3, 2016.
- [153] O. Shishkina, S. Horn, S. Wagner, and E. S. Ching. Thermal boundary layer equation for turbulent Rayleigh–Bénard convection. *Phys. Rev. Lett.*, 114(11):114302, 2015.
- [154] O. Shishkina, R. J. A. M. Stevens, S. Grossmann, and D. Lohse. Boundary layer structure in turbulent thermal convection and its consequences for the required numerical resolution. *New J. Phys.*, 12(7):075022, 2010.
- [155] O. Shishkina and S. Wagner. Prandtl-number dependence of heat transport in laminar horizontal convection. *Phys. Rev. Lett.*, 116(2):024302, 2016.
- [156] B. I. Shraiman and E. D. Siggia. Heat transport in high-Rayleigh-number convection. *Phys. Rev. A*, 42(6):3650, 1990.
- [157] E. D. Siggia. High Rayleigh number convection. *Annu. Rev. Fluid Mech.*, 26(1):137–168, 1994.
- [158] J. Sommeria. Turbulence: The elusive ‘ultimate state’ of thermal convection. *Nature*, 398(6725):294–295, 1999.
- [159] R. Speck, D. Ruprecht, M. Emmett, M. Bolten, and R. Krause. A space-time parallel solver for the three-dimensional heat equation. In *Parallel Computing: Accelerating Computational Science and Engineering (CSE)*, volume 25, pages 263–272. IOS Press, 2014.
- [160] G. A. Staff and E. M. Rønquist. Stability of the parareal algorithm. In *Domain decomposition methods in science and engineering*, pages 449–456. Springer, 2005.
- [161] J. Steiner, D. Ruprecht, R. Speck, and R. Krause. Convergence of Parareal for the Navier–Stokes equations depending on the Reynolds number. In *Numerical Mathematics and Advanced Applications-ENUMATH 2013*, pages 195–202. Springer, 2015.

- [162] R. J. Stevens, J.-Q. Zhong, H. J. Clercx, G. Ahlers, and D. Lohse. Transitions between turbulent states in rotating Rayleigh–Bénard convection. *Phys. Rev. Lett.*, 103(2):024503, 2009.
- [163] R. J. A. M. Stevens, D. Lohse, and R. Verzicco. Prandtl and Rayleigh number dependence of heat transport in high Rayleigh number thermal convection. *J. Fluid Mech.*, 688:31–43, 2011.
- [164] R. J. A. M. Stevens, E. P. van der Poel, S. Grossmann, and D. Lohse. The unifying theory of scaling in thermal convection: the updated prefactors. *J. Fluid Mech.*, 730:295–308, 9 2013.
- [165] R. J. A. M. Stevens, R. Verzicco, and D. Lohse. Radial boundary layer structure and Nusselt number in Rayleigh–Bénard convection. *J. Fluid Mech.*, 643:495–507, 1 2010.
- [166] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*, volume 12 of *Texts in Applied Mathematics*. Springer-Verlag, New York, third edition, 2002.
- [167] H. Tal-Ezer. On restart and error estimation for Krylov approximation of $w = f(A)v$. *SIAM J. Sci. Comput.*, 29(6):2426–2441, 2007.
- [168] R. Témam. Sur l’approximation de la solution des équations de navier–stokes par la méthode des pas fractionnaires, i, ii. *Arch. Rational Mech. Anal.*, 33(5):377–385, 1969.
- [169] A. Tomboulides, J. Lee, and S. Orszag. Numerical simulation of low Mach number reactive flows. *J. Sci. Comput.*, 12(2):139–167, 1997.
- [170] J. van den Eshof and M. Hochbruck. Preconditioning Lanczos approximations to the matrix exponential. *SIAM J. Sci. Comput.*, 27(4):1438–1457, 2006.
- [171] P. J. van der Houwen. Parallel step-by-step methods. *Appl. Numer. Math.*, 11(1-3):69–81, 1993.
- [172] P. J. van der Houwen and B. P. Sommeijer. Iterated Runge–Kutta methods on parallel computers. *SIAM J. Sci. Stat. Comput.*, 12(5):1000–1028, 1991.
- [173] P. J. van der Houwen and B. P. Sommeijer. CWI contributions to the development of parallel Runge–Kutta methods. *CWI Quarterly*, 11(1):33–53, 1998. Solving differential equations on parallel computers.
- [174] E. P. van der Poel, R. Ostilla-Mónico, J. Donners, and R. Verzicco. A pencil distributed finite difference code for strongly turbulent wall-bounded flows. *Comput. & Fluids*, 116(0):10–16, 2015.

- [175] J. J. W. van der Vegt and H. van der Ven. Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows. I. General formulation. *J. Comput. Phys.*, 182(2):546–585, 2002.
- [176] J. J. I. M. van Kan. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM J. Sci. Stat. Comput.*, 7(3):870–891, 1986.
- [177] J. G. Verwer and J. M. Sanz-Serna. Convergence of method of lines approximations to partial differential equations. *Computing*, 33(3-4):297–313, 1984.
- [178] R. Verzicco and R. Camussi. Prandtl number effects in convective turbulence. *J. Fluid Mech.*, 383:55–73, 1999.
- [179] R. Verzicco and R. Camussi. Numerical experiments on strongly turbulent thermal convection in a slender cylindrical cell. *J. Fluid Mech.*, 477:19–49, 2003.
- [180] R. Verzicco and P. Orlandi. A finite-difference scheme for three-dimensional incompressible flows in cylindrical coordinates. *J. Comput. Phys.*, 123(2):402–414, 1996.
- [181] J. Wang and X. Ye. Superconvergence of finite element approximations for the Stokes problem by projection methods. *SIAM J. Numer. Anal.*, 39(3):1001–1013, 2002.
- [182] S. Weiss and G. Ahlers. Heat transport by turbulent rotating Rayleigh–Bénard convection and its dependence on the aspect ratio. *J. Fluid Mech.*, 684:407–426, 2011.
- [183] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Comput. Phys.*, 12(6):620–631, 1998.
- [184] P. Wesseling. *Principles of Computational Fluid Dynamics*, volume 29 of *Springer Series in Computational Mathematics*. Springer, Berlin, 2001.
- [185] J. White, F. Odeh, A. L. Sangiovanni-Vincentelli, and A. Ruehli. Waveform relaxation: Theory and practice. Technical Report UCB/ERL M85/65, EECS Department, University of California, Berkeley, 1985.
- [186] D. E. Womble. A time-stepping algorithm for parallel computers. *SIAM J. Sci. Stat. Comput.*, 11(5):824–837, 1990.
- [187] P. Worley. Parallelizing across time when solving time-dependent partial differential equations. In *Proceedings of the 5th SIAM Conference on Parallel Processing for Scientific Computing*, 1991.

- [188] J.-Q. Zhong, R. J. A. M. Stevens, H. J. H. Clercx, R. Verzicco, D. Lohse, and G. Ahlers. Prandtl-, Rayleigh-, and Rossby-number dependence of heat transport in turbulent rotating Rayleigh-Bénard convection. *Phys. Rev. Lett.*, 102:044502, Jan 2009.
- [189] J. Zhu. A new parallel algorithm for the numerical solutions of time dependent partial differential equations (unpublished note). Engineering Research Center, Mississippi State University, 2000.
- [190] X. Zhu, E. Phillips, V. Spandan, J. Donners, G. Ruetsch, J. Romero, R. Ostilla-Mónico, Y. Yang, D. Lohse, R. Verzicco, M. Fatica, and R. Stevens. AFiD-GPU: a versatile Navier-Stokes solver for wall-bounded turbulent flows on gpu clusters. *Comput. Phys. Commun.*, 2017 (submitted).

Appendix A

Spatial resolution and Nusselt number at $Ra = 10^8$ and $\Gamma = 1$.

Table 6.1: Spatial resolution and Nu numbers for AFID in the periodic domain.

N	N_{BL}	Nu_{lo}	Nu_{up}	Nu_{vol}	Nu_{kin}	Nu_{th}
64^3	3	37.02	37.00	37.06	36.94	37.24
96^3	4	33.73	33.81	33.74	33.32	33.80
128^3	5	33.11	33.12	33.27	32.76	33.12
192^3	8	32.57	32.62	32.76	32.24	32.60
256^3	11	32.57	32.62	32.75	32.69	32.59
384^3	16	32.24	32.27	32.18	32.10	32.25

Table 6.2: Spatial resolution and Nu numbers for NEK5000 in the periodic domain.

E	N	N_{BL}	Nu_{lo}	Nu_{up}	Nu_{vol}	Nu_{kin}	Nu_{th}
5^3	36^3	2	43.78	43.63	33.47	28.01	32.25
7^3	50^3	2	36.47	36.48	32.86	29.72	31.29
9^3	64^3	3	32.01	32.04	32.27	30.70	31.33
14^3	99^3	4	31.57	31.53	32.22	31.83	31.95
18^3	127^3	5	32.30	32.22	32.39	32.19	32.23
27^3	190^3	8	32.41	32.46	32.41	32.38	32.41
36^3	253^3	11	32.31	32.43	32.45	32.35	32.38
54^3	379^3	16	32.29	32.41	32.54	32.35	32.37

Table 6.3: Spatial resolution and Nu numbers for GOLDFISH in the cubic container.

N	N_{BL}	Nu_{lo}	Nu_{up}	Nu_{vol}	Nu_{kin}	Nu_{th}
64^3	3	34.93	34.99	34.91	30.13	31.47
96^3	4	32.38	32.36	32.34	29.88	30.46
128^3	5	31.56	31.70	31.66	30.16	30.44
192^3	8	31.67	31.51	31.58	30.89	31.01
256^3	11	31.65	31.63	31.64	31.24	31.28
384^3	16	31.56	31.49	31.47	31.31	31.36

Table 6.4: Spatial resolution and Nu numbers for NEK5000 in the cubic container.

E	N	N_{BL}	Nu_{lo}	Nu_{up}	Nu_{vol}	Nu_{kin}	Nu_{th}
5^3	36^3	2	43.44	43.38	33.29	29.58	32.86
7^3	50^3	2	36.13	36.28	32.47	30.55	31.48
9^3	64^3	3	31.76	31.88	31.83	30.77	31.12
14^3	99^3	4	30.93	30.93	31.55	31.36	31.38
18^3	127^3	5	31.53	31.64	31.68	31.56	31.62
27^3	190^3	8	31.40	31.41	31.37	31.36	31.38
36^3	253^3	11	31.56	31.71	31.60	31.62	31.63
54^3	379^3	16	31.53	31.58	31.53	31.52	31.55

Table 6.5: Spatial resolution and Nu numbers for RBFLOW in the cylindrical container.

N_r	N_ϕ	N_z	N_{BL}	Nu_{lo}	Nu_{up}	Nu_{vol}
48	128	96	4	34.95	34.86	34.85
96	256	192	8	32.59	32.76	32.58
192	512	384	16	32.08	32.15	32.24

Table 6.6: Spatial resolution and Nu numbers for GOLDFISH in the cylindrical container.

N_r	N_ϕ	N_z	N_{BL}	Nu_{lo}	Nu_{up}	Nu_{vol}
48	128	96	4	33.20	33.02	33.18
96	256	192	8	32.40	32.19	32.26
192	512	384	16	32.19	32.31	32.33

Table 6.7: Spatial resolution and Nu numbers for NEK5000 in the cylindrical container. E_{xy} denotes the number of spectral elements in a horizontal cross section, and E_z in the vertical direction.

E_{xy}	E_z	N_{xy}	N_z	N_{BL}	Nu_{lo}	Nu_{up}	Nu_{vol}
48	5	2409	36	2	41.78	41.85	33.10
48	7	2409	50	2	36.91	36.62	32.87
48	9	2409	64	3	32.68	33.06	32.52
108	14	5377	99	4	31.47	31.41	32.29
192	18	9521	127	5	32.07	32.12	32.24
432	27	36100	190	8	32.14	32.18	32.16
768	36	64009	253	11	32.11	32.08	32.03
1728	54	85009	379	16	32.26	32.23	32.16

Table 6.8: Spatial resolution and Nu numbers for OPENFOAM in the cylindrical container.

N_{xy}	N_z	N_{BL}	Nu_{lo}	Nu_{up}	Nu_{vol}
3072	64	3	35.43	35.30	37.10
6912	96	4	33.33	33.27	34.00
12288	128	5	32.62	32.52	32.78
27648	192	8	32.17	32.09	32.21
49152	256	11	31.90	32.10	32.00
110592	384	16	32.16	32.10	32.13

Summary

Rayleigh–Bénard (RB) convection is a type of natural convection that occurs when a layer of fluid is heated from below and cooled from above. Flows of this kind appear frequently in engineering and in nature. One could think of the Earth’s atmosphere for example. This thesis is focussed on the development of numerical methods for simulations of RB convection. With the rise of modern computers direct numerical simulation (DNS) has become common practice for (relatively) moderate Rayleigh numbers. In other words, the equations of motion are solved accurately without modelling the small-scale features of the turbulent flow. In the first two chapters we discuss several aspects of DNS of RB convection. In the latter two chapters we explore the possibility of parallelization in time.

In Chapter 2, we study the effect of rotation on the heat transfer in RB convection. With use of a spectral element method, we perform a series of simulations at different Rayleigh and Rossby numbers. Rotation can have a large influence on the structure of the flow. This effect is visible in the appearance of numerous vertical vortices that reach from the boundary layers to deep into the domain, which leads to enhanced transport of heat. The rate of rotation has however no observable effect on the scaling of the Nusselt number with the Rayleigh number, beyond the transition to turbulence ($Ra \gtrsim 10^8$). In this study, we also demonstrated that the spectral element method is suitable for simulating (rotating) RB convection, yielding accurate results at appropriate resolution.

Over the years, many different numerical methods have been developed for DNS of turbulent flows. In Chapter 3, we analyse four such methods. The first two are specifically designed for RB convection in simple geometries: AFiD/RBflow, a second-order finite difference method and GOLDFISH, a fourth-order finite volume method. The other two methods are intended for general applications in complex geometries: Nek5000, a spectral element method, and OpenFOAM, a second-order

finite volume method. With Nek5000 we have chosen spectral elements of order eight. We perform a convergence test at $Ra = 10^8$ for meshes with different resolutions. All methods converge to the same Nusselt number at sufficient resolution. Large differences are however observed in the computational cost (measured per gridpoint), which mainly illustrates the contrast between specialized and general codes. The specialized codes are optimized for simple geometries. The differences are considerably reduced when the achieved accuracy is taken into account though. At equal resolution, Nek5000 is for example the most expensive code, but also the more accurate one. All in all, AFiD/RBflow and GOLDFISH appear to be the most efficient choices, concerning accuracy against computational cost, with computational costs up to ten times lower compared to Nek5000 and OpenFOAM.

Because the number of cores in new supercomputers keeps increasing, parallelization becomes even more important. An interesting approach is a parallelization in time, especially when the traditional parallelization in space is already saturated. Such an approach would allow to reduce the wall-clock time of a desired simulation. A recent algorithm for parallelization in time is the Paraexp algorithm, developed for linear initial value problems. In Chapter 4, we present an extension of the Paraexp algorithm to nonlinear partial differential equations (PDE) using the exponential block Krylov (EBK) method as a base approach. This parallel-in-time method is also suitable for linear PDEs. For a first impression of the parallel efficiency, we estimate the computational time by measuring the computational processes individually on a serial computer. Here, the parallel communication is assumed to be very small. We first test the method for the advection-diffusion equation. Even with a low diffusion coefficient, the parallel efficiency remains high. The application to nonlinear PDEs is illustrated with the viscous Burgers equation. Our method also shows good parallel efficiency in that case at different values of the viscosity coefficient.

An essential element of the Paraexp algorithm is the so-called exponential integrator, which can solve homogeneous initial value problems very fast. The integration methods are based on the actions of the matrix exponential or related matrix functions on a vector. This can be done, for example, using Krylov subspace methods. In Chapter 5, we introduce an exponential integrator for the incompressible Navier–Stokes equations based on the EBK method. The application to the Navier–Stokes equation is not straightforward because of the presence of the continuity equation. This imposes a constraint on the velocity field for it to be divergence-free. We show that the governing equations can be reduced to an ordinary differential equation using a projection operator on divergence-free vector fields. Several numerical experiments, including a lid-driven cavity flow, show that the EBK method converges to the reference solution. An important property of the EBK method is the possibility to parallelize it in time with the framework discussed in Chapter 4. Initial estimates show that parallel speedup is possible, even with the proposed prototype method.

For practical application the acceleration of the EBK method is certainly recommended. This would perhaps be possible with a rational Krylov subspace, using

the shift-and-invert technique. The application to incompressible flow simulation is not clear at present. The divergence-free projector, introduced in Chapter 5, complicates the usual procedure in particular. Such methods are subject to future studies.

Samenvatting

Rayleigh–Bénard (RB) convectie is een vorm van natuurlijke convectie die plaatsvindt wanneer een laag vloeistof of gas van onderen wordt verwarmd en van boven gekoeld. Stromingen van deze soort komen dikwijls voor in de techniek of in de natuur. Men kan bijvoorbeeld denken aan de atmosfeer van de Aarde. Dit proefschrift is gericht op de ontwikkeling van numerieke methodes voor het simuleren van RB convectie. Dankzij de komst van moderne computers is de *direct numerical simulation* (DNS) gemeengoed geworden voor (relatief) bescheiden getallen van Rayleigh. Met andere woorden, de bewegingsvergelijkingen worden nauwkeurig opgelost zonder het modelleren van de kleinschalige kenmerken van de turbulente stroming. In de eerste twee hoofdstukken behandelen we verschillende facetten van DNS van RB convectie. In de overige twee hoofdstukken verkennen we de mogelijkheid van een parallelisatie in de tijd.

In Hoofdstuk 2 bestuderen we het effect van rotatie op de warmteoverdracht in RB convectie. Met behulp van een spectrale-elementenmethode, voeren we een serie simulaties uit bij verschillende getallen van Rayleigh en Rossby. Rotatie kan een grote invloed hebben op de structuur van de stroming. Dit effect is zichtbaar in de aanwezigheid van talrijke verticale wervels die vanuit de grenslagen tot ver in het domein reiken, wat leidt tot een toename in de warmteoverdracht. De rotatiesnelheid blijkt echter geen waarneembare invloed te hebben op de schaling van het getal van Nusselt met het getal van Rayleigh, voorbij de transitie naar turbulentie ($Ra \gtrsim 10^8$). Met deze studie hebben we ook aangetoond dat de spectrale-elementenmethode een geschikte methode is voor het simuleren van (roterende) RB convectie waarmee nauwkeurige resultaten kunnen worden behaald bij voldoende resolutie.

In de loop der jaren, zijn er vele verschillende numerieke methodes ontwikkeld voor DNS van turbulente stromingen. In Hoofdstuk 3 analyseren we vier nu-

merieke codes waarin zulke methodes zijn opgenomen. De eerste twee codes zijn specifiek ontwikkeld voor RB convectie in simpele geometrieën: AFID/RBflow, een tweede-orde-eindige-differentiemethode en GOLDFISH, een vierde-orde-eindige-volumemethode. De overige twee zijn bedoeld voor algemenere toepassingen in complexe geometrieën: Nek5000, een spectrale-elementenmethode, en OpenFOAM, een tweede-orde-eindige-volumemethode. Met Nek5000 kiezen we voor spectrale elementen van de achtste orde. We voeren een convergentietest bij $Ra = 10^8$ uit voor roosters op verschillende resoluties. Alle methodes convergeren uiteindelijk naar het hetzelfde getal van Nusselt bij voldoende resolutie. In de rekenkosten (gemeten per roosterpunt) zijn echter grote verschillen waar te nemen, wat met name het contrast tussen de gespecialiseerde en de algemene codes illustreert. De gespecialiseerde codes zijn namelijk geoptimaliseerd voor simpele geometrieën. De verschillen in rekenkosten worden echter aanzienlijk kleiner wanneer de nauwkeurigheid wordt meegewogen. Bij gelijke resolutie is Nek5000 bijvoorbeeld de duurste code, maar eveneens de meest nauwkeurige. Al met al, lijken AFID/RBflow en GOLDFISH de meest efficiënte keuzes te zijn als het gaat om nauwkeurigheid tegen rekenkosten voor RB simulaties, waarbij de rekenkosten tot tien keer lager liggen ten opzichte van Nek5000 en OpenFOAM.

Omdat nieuwe supercomputers steeds meer cores hebben, wordt de parallelisatie steeds belangrijker. Een interessante aanpak is een parallellisatie in de tijd, met name als de gebruikelijke parallellisatie in de ruimte verzadigd is. Een recent algorithm voor een parallellisatie in tijd is het Paraexp-algorithm, ontwikkeld voor lineaire beginwaardeproblemen. In Hoofdstuk 4, presenteren we een uitbreiding van het Paraexp-algorithm naar niet-lineaire partiële differentiaalvergelijkingen (PDV) op basis van de exponential block Krylov (EBK) methode. Deze parallel-in-tijdmethode is eveneens geschikt voor lineaire PDVs en kan ook worden ingezet voor niet-lineaire problemen. Om een eerste indruk te krijgen van de parallele efficiëntie, schatten we de rekestijd door de parallele processen afzonderlijk te meten op een seriële computer. Hierbij wordt aangenomen dat de parallele communicatie gering is. We testen de methode eerst uit voor de advection-diffusievergelijking. Zelfs bij een lage diffusiecoëfficiënt, blijft de parallele efficiëntie hoog. De toepassing op niet-lineaire PDVs wordt geïllustreerd met behulp van de viskeuze Burgersvergelijking. Ook hier laat onze methode goede parallele efficiëntie zien bij verschillende waarden van de viscositeitscoëfficiënt.

Een onmisbaar element van het Paraexp-algorithm is de zogeheten exponentiële integrator, waarmee homogene beginwaardeproblemen bijzonder snel kunnen worden opgelost. Deze integratiemethodes zijn gebaseerd op acties van de matrixexponentiële functie of gerelateerde functies. Dit kan bijvoorbeeld worden gedaan door middel van Krylovmethodes. In Hoofdstuk 5 introduceren we een exponentiële integrator voor de incompressibele Navier–Stokesvergelijking op basis van de EBK methode. De toepassing op de Navier–Stokesvergelijking is niet vanzelfsprekend vanwege de aanwezigheid van de continuïteitsvergelijking. Na discretisatie levert dit een algebraïsche vergelijking die de voorwaarde dat het snelheidsveld divergentie-vrij moet zijn weergeeft. We laten zien dat de differentiaal-

algebraïsche vergelijking kan worden gereduceerd tot een gewone differentiaalvergelijking met behulp van een projectieoperator op divergentie-vrije vectorvelden. Verschillende numerieke experimenten, met bijvoorbeeld een *lid-driven cavity* stroming, laten zien dat de EBK methode convergeert naar de referentie-oplossing. Een belangrijke eigenschap van de EBK methode is dat deze kan worden geparalleliseerd volgens het raamwerk uitgezet in Hoofdstuk 4. Eerste schattingen laten zien dat parallelle speedup mogelijk is, zelfs met het voorgestelde eenvoudige prototype van de methode.

Voor praktische toepassingen is het accelereren van de EBK methode zeker aan te raden. Dit zou wellicht kunnen met bijvoorbeeld rationale Krylovruimtes waarbij men gebruik maakt van de *shift-and-invert*-techniek. De toepassing op incompressibele stromingen ligt echter niet voor de hand vooralsnog. De divergentie-vrije projectie, zoals in Hoofdstuk 5, compliceert namelijk de gebruikelijke procedure. Zulke methodes moeten in de toekomst nader worden onderzocht.

About the author

Gijs Kooij was born in Tilburg, The Netherlands, on the 14th of February, 1989. He received his high school diploma at the St.-Odulphuslyceum in Tilburg. Afterwards, he obtained a BSc and a MSc degree in Aerospace Engineering from Delft University of Technology. During this time he did an internship at the German Aerospace Centre in Göttingen, Germany. He finally pursued a PhD degree in Applied Mathematics at the University of Twente under the supervision of Prof. Bernard Geurts and Dr. Mike Botchev. His work was focussed on the development of numerical methods for simulations of Rayleigh–Bénard convection.

List of publications

Refereed publications

- G.L. Kooij, M.A. Botchev, and B.J. Geurts. *Direct numerical simulation of Nusselt number scaling in rotating Rayleigh–Bénard convection*. Int. J. Heat Fluid Flow **55**, 26–33, 2015.
- G.L. Kooij, M.A. Botchev, and B.J. Geurts. *A block Krylov subspace implementation of the time-parallel Paraexp method and its extension for non-linear partial differential equations*. J. Comput. Appl. Math. **316**, 229–246, 2017.
- G.L. Kooij, M.A. Botchev, and B.J. Geurts. *An exponential time integrator for the incompressible Navier–Stokes equation*. Submitted.
- G.L. Kooij, M.A. Botchev, E.M.A. Frederix, B.J. Geurts, S. Horn, D. Lohse, E.P. van der Poel, O. Shishkina, R.J.A.M. Stevens, and R. Verzicco. *Comparison of computational codes for direct numerical simulations of turbulent Rayleigh–Bénard convection*. Submitted.

Conference proceedings (abstracts)

- G.L. Kooij, M.A. Botchev, and B.J. Geurts. *Direct numerical simulations of rotating Rayleigh–Bénard convection*. 10th Int. ERCOFTAC Symp. on Engineering Turbulence Modelling and Measurements (ETMM10), Marbella, Spain, Sept. 17–19, 2014.

- G.L. Kooij, M.A. Botchev, and B.J. Geurts. *Simulations of turbulent Rayleigh–Bénard convection with a spectral element method*. GAMM 86th Annual Scientific Conference, Lecce, Italy, March 23–27, 2015.
- G.L. Kooij, M.A. Botchev, and B.J. Geurts. *Comparison of a spectral element and a finite volume method for DNS for Rayleigh–Bénard convection*. 10th ERCOFTAC Workshop on Direct and Large-Eddy Simulation (DLES10), Limassol, Cyprus, May 27–29, 2015.
- G.L. Kooij, M.A. Botchev, and B.J. Geurts. *A Krylov-based exponential time integrator of the incompressible Navier–Stokes equation*. European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS), Hersonissos, Greece, June 5–10, 2016.
- G.L. Kooij, S. Horn, R.J.A.M. Stevens, E.P. van der Poel, M.A. Botchev, O. Shishkina, D. Lohse, R. Verzicco, and B.J. Geurts. *Comparison study of numerical codes for turbulent Rayleigh–Bénard convection*. 11th Int. ERCOFTAC Symp. on Engineering Turbulence Modelling and Measurements (ETMM11), Palermo, Italy, Sep. 21–23, 2016.
- G.L. Kooij, S. Horn, R.J.A.M. Stevens, E.M.A. Frederix, E.P. van der Poel, M.A. Botchev, O. Shishkina, D. Lohse, R. Verzicco, and B.J. Geurts. *Benchmark of numerical methods for DNS of Rayleigh–Bénard convection*. 11th Int. ERCOFTAC Workshop on Direct and Large-Eddy Simulation (DLES11), Pisa, Italy, May 21–23, 2017.