Matrix-based techniques for (flow-)transition studies

Weiyan Song

ISBN: .....

The research reported in this thesis was supported by the German Aerospace Laboratory (DLR) and the University of Groningen. LOGO's

RIJKSUNIVERSITEIT GRONINGEN

Matrix-based techniques for (flow-)transition studies

Proefschrift

ter verkrijging van het doctoraat in de
Wiskunde en Natuurwetenschappen
aan de Rijksuniversiteit Groningen
op gezag van de
Rector Magnificus, dr. E. Sterken,
in het openbaar te verdedigen op
.... dag .. december 2018
om 12.45 uur

door

Weiyan Song

geboren op .....
te ....., China

Promotor:	Prof. dr. A.E.P. Veldman
Copromotor:	Dr. ir. F.W. Wubs
Beoordelingscommissie:	Prof. dr. ..... Prof. dr. ..... Prof. dr. .....

Contents

Chapter 1

Introduction

In this thesis, numerical techniques for the computation of flow transitions will be introduced and studied. Before we say something about the relevance of such transitions, we need to explain what we mean by it. A flow transition is a qualitative change in the flow pattern when a physical parameter is changed only slightly. One example is the whistling sound which car mirrors used to make at high driving speeds. At low speeds the flow around the mirror will be steady, while at high speeds it will be oscillatory and produce sound waves. So qualitatively the flow changes from steady to oscillatory. Nowadays the shape of such mirrors is adapted such that at all common speeds the flow does not produce sound. Another example is the collapse of the Tacoma Narrows Bridge in 1940 caused by an oscillating flow at high wind speeds. If such an oscillation is close to that of an eigenmode of a structure, then the structure starts to oscillate as well and may even break down as aforementioned bridge. A similar phenomenon appeared at the Erasmus bridge in Rotterdam in 1996, where the suspension cables started to oscillate virulently such that the bridge had to be closed temporarily. A qualitative change in flow also occurs during the onset of turbulence. By making only small changes to a geometry, the flow may change from laminar to turbulent. Depending on different cases, turbulence may increase or decrease the drag of an object in flow. For instance, in a flow around a wing one can delay separation of the flow by forcing the flow to become turbulent near the trailing edge. This results in a slight decrease of the drag, which lowers the fuel consumption of a plane using these wings. Another interesting case of qualitative change in flow pattern occurs when multiple stable steady states exist at one set of physical parameters. This occurs in the climate system and one of the key questions is what kind of perturbations are needed to shift from one state into the other.

From the previous description, it is clear that industrial applications of numerical simulation of these equations can be found in many fields, such as in complex turbulent flows [1], aircraft design [2] and so on. Theoretical studies of transitions in flows of liquids have been studied extensively for decades. Famous classical examples are Poiseuille flow (transitions in a circular pipe) [3], Taylor-Couette flow (flow between rotating cylinders) [4], Rayleigh-Bénard flow (convection in a liquid layer heated from below, more details in Chapter 3) and so on. Transitions occur through bifurcations when parameters are varied. To tackle flow transitions, numerical models for the simulation of the flow are essential. In general, these simulation models are based on fluid flow equations, i.e., the Navier-Stokes equations.

This chapter is organized as follows. First we shall introduce fluid equations and the main techniques for analyzing these equations mathematically. Then we shall also briefly discuss the relevance of linear system solving. Next we will state our research questions and indicate on which problems we will try to answer them. Finally, an overview of the thesis will be presented.

1.1 The Navier-Stokes equations

The Navier-Stokes equations 1.1, derived by Navier, Poisson, Saint-Venant, and Stokes between 1827 and 1845 [5, 6], are the fundamental partial differential equations that govern the motion of the fluid and can be seen as Newton's second law of motion for fluids. In the case of a compressible Newtonian fluid, they are

$$\rho(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{1.1}$$

where ρ is density, μ is the dynamic viscosity, and p is the pressure. $\mathbf{f}$ describes the external force applied to the fluid. The Navier-Stokes equations represent the conservation of momentum, and they are always solved with the continuity equation 1.2, which represents the conservation of mass:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \tag{1.2}$$

Beside these equations, we need an equation of state which specifies the relation between the density ρ and pressure p. The Navier-Stokes equations can model both compressible and incompressible fluids. An example of the former is the fluid air as used in the modeling of the earth atmosphere, while an example of the

latter is the fluid water as used in the simulation of ocean flows. In general, an analytical solution of the Navier-Stokes equations is not available. Therefore, the efficient computation of numerical solutions for these equations is essential. There exist many types of Navier-Stokes solvers, since a solver can be based on a variety of discretization techniques, for instance, finite differences, finite elements, finite volumes or discontinuous Galerkin methods. A comparative introduction to these methods can be found in [7]. In this thesis, we will use the finite volume method for the discretization, which is commonly used in practice, since it discretely preserves the conservation laws from fluid dynamics. We are aware that the simple geometries we consider also admit more sophisticated methods like the pseudo-spectral method. However, since we collaborate with oceanographers, who deal with arbitrary geometries and bottom topologies, we stick to the same discretization technique as they do. This makes it possible to test the quality of finite volume discretization on challenging problems. Moreover, we will focus on the problems related to incompressible flow.

1.2 Incompressible flow equations

The incompressible flow refers to a flow in which the material density is constant and does not link to the pressure. In that case, the continuity equation turns into an equation saying that the velocity field is divergence-free and can be seen as a constraint on the velocity field:

$$\nabla \cdot \mathbf{u} = 0. \tag{1.3}$$

We are interested in the incompressible Navier-Stokes equations combined with a number of transport equations:

$$\begin{aligned} \rho(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}) =& - \nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f}(\phi), \\ \nabla \cdot \mathbf{u} =& 0, \\ \frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi =& \kappa \nabla^2 \phi + g(\mathbf{u}), \end{aligned} \tag{1.4}$$

where ϕ is the entity transported by the flow, for instance, one could think of transport of energy, material, etc. In many cases there is interaction between the flow and the components that are transported; here it is represented by the functions $\mathbf{f}$ and g. In the case of small temperature differences in the flow, one often uses

the well-known Boussinesq equations, which approximate the equations of compressible flow, shown in the previous section, by the above equations in which ϕ will become the temperature (see Chapter 3 for examples).

There are two main difficulties when solving the incompressible Navier-Stokes equations for a Newtonian fluid. The first one is the nonlinear system induced by the convective term. The second one is the constraint of mass conservation and the role of the pressure term which has no thermodynamical significance [8].

Often, one can get rid of the nonlinear system, by using an explicit time-integration method, such as the Euler method, only solving a Poisson equation for the pressure. However, the role of nonlinearity turns out more and more important with the emergence of small structures in the flow dynamics. For instance, when Reynolds number becomes high. This means that the convective effects are getting dominant, and hence the mesh must be fine enough to adequately capture all the solution scales in the flow. This means in practice finer and finer spatial and temporal meshes must be used when the Reynolds number increases. As a consequence, there will be massive computational costs.

If we want to use larger time steps, using an implicit method then the coupled velocity-pressure problem, after linearization, has a saddle-node structure which makes it difficult to solve. It is known that standard preconditioners, based on incomplete factorizations, easily end up to be singular for saddle-point matrices.

Furthermore, we do not constrain ourselves to the calculation of a steady flow at increasing parameter values, e.g. the Reynolds number. We are also interested in the primary transition of a stable steady flow to another stable flow with a different pattern, which can even by transient. For the location of this transition point and the switch to a solution branch with different flow pattern, accurate computation of the most unstable perturbation mode is required. This is represented by the leading eigenvector of the momentum and continuity equations linearized around the steady state. Therefore, eigenvalue analysis is the most natural approach to solve this kind of problems. However, usually because of the large size of the algebraic eigenvalue problem, this takes way too much time and in most of the cases it is not affordable. Thus, many authors use time-dependent methods for stability studies, rather than the eigenvalue analysis [9]. The leading eigenvalues were computed to check the stability of the solution in some studies, for instance, lid-driven cavity [10, 11], but no accurate eigenvalues were computed to locate the critical Reynolds number.

1.3 Two main approaches for bifurcation and stability analysis

In general, for the bifurcation study of fluid flows, one can distinguish between two methods: the time integration approach and the continuation approach. We will discuss both briefly below.

1.3.1 Time integration approach for steady state and stability analysis

The obvious way to study transitions of flows is by just performing time integration for various values of natural parameters, e.g., the Reynolds number. If a steady state is reached, then it is a stable steady state. If a periodic solution is reached, one has found a stable periodic solution. If the latter occurs after an increase of the natural parameter, then one knows that a Hopf-bifurcation has occurred during this increase. Using the time integration in this way, only stable solutions can be found. To be a bit more precise, we find stable solutions of the numerical time integrator, since numerical time integration schemes have their own damping and amplification properties that differ slightly from those of exact time integration.

It is possible to find also unstable steady states using a time integration code. For that it must be incorporated in a fixed point iteration, e.g., take the fixed point function as the flow $\phi(x, T)$ denoting the integration of some initial solution x over a specified time interval T. We know that mildly unstable fixed point iterations can be made to converge with Aitken extrapolation. Such an idea is also present in the Newton-Picard method, which is a generalization of the recursive projection method (RPM) of Shroff and Keller [12]. This method was developed by Lust and Roose [13] and has been implemented in the free software PDECONT(see http://www.kurtlust.net/CODE/r_PDEcont.html). Tiesinga et al. [14] used this method to study the bifurcation behavior of the flow in a driven cavity for Reynolds numbers between 7,500 and 10,000.

Based on time integration, Tuckerman has developed FORTRAN codes to make it more efficient for incompressible flow [15]. In [16], it is shown that the implicit linear step of a time-stepping code can serve as a highly effective preconditioner for solving linear systems involving the full Jacobian via conjugate gradient iteration. This preconditioning can be used in steady-state solving, continuation, direct calculation of bifurcation points by Newton's method, and linear

stability analysis by the inverse power method. In [17, 18], they have thus obtained various coexisting steady, time-dependent flows and a bifurcation diagram of cylindrical Rayleigh-Bénard convection.

A numerical study of several time integration methods for solving the three-dimensional Boussinesq thermal convection equations in rotating spherical shells is presented by Garcia et al. [19]. Both implicit and semi-implicit time integration techniques are considered. It showed that the use of high-order methods, especially those with time step and order control can increase the efficiency of the time integration as well as the accuracy of solutions. Citro et al. [20] propose a new algorithm to compute unstable steady states of a high-dimensional dynamical system efficiently. The method is based on the minimization of the residual norm at each integration step and can be applied as a black-box procedure in any iterative or time marching algorithm.

Note that, due to their simplicity, explicit time integration methods, like Euler's method, are very well suited for computations on parallel computers. Designing an implicit method such that it is efficient on a parallel computer is possible as we will show later on, but requires in general a lot more effort.

1.3.2 Continuation of steady states and eigenvalue computation

Compared to time-marching approaches, the natural parameter continuation method avoids potentially long time integrations to obtain generic and meaningful information and hence can bring down the computation time considerably, especially when calculating steady solutions [21]. Another advantage of natural parameter continuation is that it could be developed as a black box, since only an initial solution is required for the target problem. However, if a turning point is encountered with the increase of the parameter value during natural parameter continuation, the branch cannot be followed. For problems with turning points, pseudo-arclength continuation, a more sophisticated method, needs to be used. Riks and Wempner developed Pseudo-arclength continuation for finite element applications in the late 1960s and it was published in journals in the early 1970s by H.B. Keller [22]. A detailed description of these early developments is provided in the textbook by Crisfield [23].

In each continuation step, a nonlinear system has to be solved. For the solution of the nonlinear system, there is no algorithm to compute the exact solution like Gaussian elimination does for linear systems. We have to resort to iterative

methods to obtain an approximate solution in a finite number of iterations. The most frequently used method is the Newton-Raphson method (Newton's method for short) and the Picard iteration leading to a linear system to be solved in each iteration. Both of them need a sufficiently accurate initial guess to achieve convergence [24].

There has been a lot of work in the area of large-scale continuation on transforming more sophisticated algorithms into black box solvers, for instance, they are available in the software package LOCA (Library Of Continuation Algorithms), which is one of the packages provided in the Trilinos project[1]. LOCA implements among others the pseudo-arclength continuation and this implementation is used in this thesis.

Apart from continuation of solutions, one often also wants to determine their stability under variation of parameters. One has to calculate the most unstable eigenmode along with the steady-state flow, which can be more difficult than the computation of the flow itself. A general method based on linear algebra can be used to study the stability of a fixed point with respect to small perturbations [25]. For large systems, there are mainly three practical options: subspace iterations, Krylov subspace methods and (accelerated) Newton method. The orthogonal subspace iteration method and the Arnoldi method [26] are examples of the first two, both of which are generalizations of the power method for the eigenvalue problem. An example of the third is the Jacobi-Davidson QR (JDQR) method [27]. An excellent review of numerical methods for large-scale eigenvalue problems can be found in [28], in which the author also explains filtering, restart, and preconditioning techniques. In this thesis two methods are used, one is a block Krylov-Schur method with Shift-and-Invert or Cayley transform explained in section 3.4.1 and the other is block JDQR.

Instead of performing continuation meanwhile monitoring stability, Meerbergen [29, 30] constructed an approach in which the parameter for which an Hopf bifurcation occurs can be solved at once from a Lyapunov equation. We did not use this technique but mention it here for completeness.

1.4 Relevance of linear system solving

The effectivity of continuation and stability study depends severely on the efficient and robust solution of linear systems. There have been many solvers around

[1] see http://trilinos.sandia.gov/

for large sparse linear systems, each of them has their advantages and disadvantages. In [31], the authors gave a survey of methods currently used to solve linear systems from fluid flow problems. A saddle-point system is obtained after discretization of incompressible Navier-Stokes equations. In principle, these linear algebraic systems can be solved exactly within a finite number of operations, for instance, Gaussian elimination. They are referred to as direct methods. However, in practice, it is not possible since a typical computational fluid dynamics problem may involve millions of unknowns leading to a huge memory request for the storage of the factorization and also an enormous amount of time to compute it. Therefore, for large applications, iterative methods are preferred, i.e., one performs a finite number of iterations to achieve an approximate solution. But it also has its disadvantages. For instance, iterative methods are not always robust, especially for difficult problems, such as mixed parabolic and hyperbolic PDEs [24]. Convergence may not be achieved and the final approximation solution may not be accurate at all. However, Krylov subspace iteration method can solve saddle point problems efficiently [32] when combined with appropriate preconditioning [31, 33–37].

Many preconditioning techniques have been developed to accelerate the iterative solvers. In [38] and references therein, the authors presented a good overview of the present state of fast solvers for the solution of the incompressible Navier-Stokes equations discretized by the finite element method and linearized by Newton's or Picard's method. Block-oriented preconditioners perform well for the incompressible Navier-Stokes equations. This includes standard additive-Schwarz domain-decomposition methods, aggressive coarsening multigrid, and three preconditioners based on an approximate block LU factorization, specifically SIMPLEC, LSC, and PCD, of which SIMPLEC, LSC, and PCD are implemented in Teko, another software package of Trilinos, see more in [39].

For the separate blocks, occurring in block-oriented preconditioners, often algebraic multigrid methods are employed. Multigrid methods have been proved to be robust and efficient. The first AMG (Algebraic MultiGrid) program was introduced and described by Ruge and Stüben [40]. Since the early 1990s, there was a strong increase of interest in algebraically oriented multilevel methods because of the high demand for efficient "plug-in" solvers driven by increasing problem sizes. An excellent review of AMG is [41]. Another Trilinos package is ML, which has a variety of parallel multigrid schemes for solving the large sparse linear systems following primarily from discretization of elliptic PDEs [42], for example, it has the scheme smoothed aggregation [43], a variant of AMG.

The idea of combining direct and iterative methods has been used by Henon

and Gaidamour [44–46] to solve general sparse linear systems arising from the discretization of scalar PDEs. As in that paper, they reduce the problem to a Schur-complement system on the separators of a domain decomposition. The Schur-complement system is solved iteratively using an ILU factorization. As the structural and numerical properties are not explicitly preserved, robustness and grid-independence cannot be ascertained for indefinite problems.

Thies and Wubs [47] combined ideas from complete and incomplete factorizations to construct a preconditioner that is acting in the divergence free space. By doing this, the preconditioner can deal with the saddle-point structure of the system. For Stokes problems it is guaranteed robust and for Navier-Stokes problems it hardly fails. Also this is a multilevel method and called HYbrid Multi-Level Solver (HYMLS). Another advantage of this hybrid algorithm is that it allows parallel computation on each level [47, 48]. HYMLS is expressed in data structures available in the Trilinos software package Epetra and thereby it is intended for distributed memory computers.

1.5 Our research questions

The research questions in this thesis are threefold.

1. Is the continuation approach a viable alternative to time integration approaches?

2. Can we compute solutions by the continuation approach, which are hard to get by the time-integration approach?

3. How does HYMLS compare to other solvers, like ”physics-based” preconditioners in Trilinos package Teko, and ML with respect to robustness and turnaround time.

As they say: ”The proof of the pudding is in eating it.”, we want to answer our questions by solving a range of test problems with a variety of physics. We also want to use several methods for some of these problems to get a comparison between methods. In the next section, we give some particularities of our test problems.

1.6 Overview of test problems

In this thesis, we firstly test the matrix approach on three well-known widely accepted benchmark problems: one is a flow in a 3D lid-driven cavity, the other two are multi-physics problems: Rayleigh-Bénard convection and differentially heated cavity with and without rotation. Until recently, numerical calculations have predominantly been performed for two-dimensional flows. In our study, both two and three-dimensional problems are considered.

Driven cavity flow serves as a benchmark problem for numerical methods regarding accuracy and numerical efficiency. Although the problem seems simple, it reveals complex phenomena of vortex dynamics, hydrodynamics stability, flow bifurcation, etc. In the literature, it is possible to find numerous studies with different methods on the driven cavity flow, for example in [49] and references therein. Albensoeder et al. [50] gave an excellent survey of studies on this problem. Tiesinga et al. [14] studied the transition from steady to periodic state thoroughly by the Newton-Picard method. Since the 2D lid-driven cavity have been well studied and three-dimensional flow has more physical significance, we will only consider the 3D case in this thesis. Recently, Kuhlmann and Albensoeder found the first bifurcation to be of Hopf-type and slightly subcritical. Above the critical point, the oscillatory flow is symmetric with respect to the symmetric midplane of the cavity. And on a long time scale, the periodic oscillations are interrupted by short bursts [51].

Rayleigh-Bénard convection, which has a great variety of industrial applications, is also an excellent system on which to test new ideas and approaches for understanding nonlinear dynamics. It displays rich dynamics, ranging from stationary patterns to weakly chaotic evolution to highly turbulent states [52, 53]. Numerical parameter continuation and bifurcation methods are needed to study the flow transitions that occur as the Rayleigh number is increased. There are already many results showing the bifurcation diagrams of steady convective flow patterns in a cubical cavity with conducting and insulated lateral walls [54, 55]. In [56], the onset of convection for intermediate aspect ratio is presented in cylindrical containers.

Natural convection flow within a differentially heated cavity (vertical rectangular container with lateral heating in the presence of gravity) also has lots of applications in the industry, such as solar energy storage, building thermal insulation, nuclear reactor core insulation and so on. As a prototypical model, it has been studied for many years [57, 58]. Recently, the instability onset and slightly supercritical oscillatory regimes of air convection in a cubic laterally heated box

are studied in [59] by straightforward time integration of the Boussinesq equations. In [60], the author used a Chebyshev pseudo-spectral discretization for the $8:1$ differentially heated cavity and obtained accurate unsteady simulations at the required Ravalue of 3.4×10^5 also the three first critical bifurcation points were accurately determined.

Rotation can play an important role and has a significant effect on the heat transfer. More studies can be found in [61] and references therein. Recently, in [62], the effect of rotation through Coriolis force on natural convection in the two-dimensional differentially heated rotating enclosure is shown. The numerical investigation is carried out for fixed Prandtl number equal to 0.71, Rayleigh number equal to 1.1×10^5. Results of the effects of rotation on three-dimensional flow in a cavity generated by a horizontal temperature gradient are presented in [63]. We will consider here a 3D differentially heated rotating cavity.

In addition, to test the efficiency and scalability of our solvers for non fluid flow problems, we considered a widely studied model for spatial pattern formation, proposed by Turing in 1952 [64]. Turing showed that a system of two reacting and diffusing chemicals could produce spatial patterns in chemical concentrations from the destabilization of a homogeneous state. Many experimental results have illustrated the formation of striped and spotted patterns, as well as more complicated patterns [65]. The term diffusion-driven instability has occurred in chemical and ecological processes. Turing models can exhibit most of those patterns, and these can be found in many theoretical and experimental papers. For a review, see [66–68] and references therein.

It is known that 3D solutions can display much richer behavior than 2D solutions. There are much more possibilities for spatial multi-stability in three dimensions than those in two dimensions. They are not only interesting from a theoretical point of view but also possible in nature as is shown, e.g., by Bánsági et al. [69]. A decade ago, not many 3D results existed, and the results that existed were on relatively coarse grids, e.g., [70]. Recently, however, quite a few 3D results that were generated on parallel computers were published using time integrations methods, e.g., [71, 72].

The pattern formation behavior in Turing systems is very complex. In this thesis, we have focused on a Turing model called the Barrio-Varea-Aragon-Maini (BVAM) model, proposed by Barrio *et al.*[73] in 1999.

In general, Turing problems have been addressed using the time-dependent method; numerical continuation is rarely used. McCullen and Wagenknecht in 2016 investigated patterns on complex networks computationally using numerical continuation methods for 2D networks [74]. We have not seen a numerical

continuation with multigrid technique applied for 3D analyses in this field.

1.7 Overview of this thesis

In this chapter, we motivate our research, introduce the governing equations and discuss a variety of methods to solve the equations. In Chapter 2, we explain the used algorithms in detail. The parameter continuation schemes and the whole program structure are presented. Next, a description follows of our finite volume discretization module FVM. After that, we describe several linear solvers as well as two eigenvalue computation methods: Arnoldi method and Jacobi-Davidson QR. In Chapter 3, we discuss the particularities of four canonical flow problems: (i) the lid-driven cavity, (ii) the differentially heated cavity, (iii) the Rayleigh-Bénard convection problem, and (iv) a rotating differentially heated thin cavity. Chapter 4 presents computational results of the lid-driven cavity problem, not only the critical Hopf bifurcation is located but also performance results of the linear solver HYMLS. Partial bifurcation diagrams and performance studies are also given in Chapter 5 for one multi-physics problem: Rayleigh-Bénard convection. In Chapter 6, we present bifurcation analysis on a Turing-type Reaction-Diffusion model, showing both two- and three-dimensional bifurcation diagrams. Furthermore, performance studies including comparison with another preconditioner ML are also given. The last chapter outlines conclusions and discusses future work.

Chapter 2

Description of the algorithms

Our continuation program is developed based on the object-oriented software framework Trilinos. It mainly consists of two parts (i) FVM (stands for Finite Volume Method) in which the users can define their problem, i.e., construct the Jacobian matrix, mass matrix and right-hand side (RHS) of the equations to be studied, and (ii) the continuation program, obtaining series of steady solutions as well as performing stability analysis by eigenvalue computation. Currently, FVM can only deal with rectangular domains and Cartesian grids have been used. In FVM, we use an overlapping domain decomposition and create the local part of the Jacobian and RHS in Fortran and then assemble the full matrix and vector using Epetra communication data structures, which are based on MPI. The program uses the Trilinos package LOCA for the actual continuation and is written in C++ using data structures given in the Epetra package of Trilinos. For the eigenvalue computation we used both the Anasazi package of Trilinos and the stand alone package PHIST [75]. In this chapter, we describe briefly the continuation algorithm of LOCA, the layout of our complete code, the way we implemented FVM, the linear system solvers used and the eigenvalue problem solvers employed, respectively.

2.1 Pseudo-arclength continuation algorithm

Continuation is at the heart of our program, therefore we describe here the algorithm used from LOCA. By continuation methods, families of steady states or periodic orbits for a large range of parameter values can be obtained with meaningful and generic information on the local dynamics of the PDEs. Moreover, con-

tinuation methods are able to determine the first bifurcations from the branches of steady states and periodic orbits in an efficient way [21].

Our research is focused on the steady state of the studied incompressible flows. With continuation, a series of approximate solutions can be generated by solving a system of parameterized nonlinear equations $F(\mathbf{u}, \lambda) = 0$, where $\mathbf{u}$ is the solution and λ is the model parameter varied during the continuation. We will first describe *natural* continuation, i.e., having the solution at λ, we progress λ by a certain amount and compute the solution at this new value. This solution is obtained by a predictor-corrector approach. Suppose we have the solution $\mathbf{u}_j$ at a certain value of λ_j, then for the solution at the new value $\lambda_{j+1} = \lambda_j + \Delta\lambda_j$ we first make a prediction $\mathbf{u}'_{j+1}$. This serves as an initial guess for the corrector which solves $F(\mathbf{u}, \lambda_{j+1}) = 0$ by a Newton-type method and leads to $\mathbf{u}_{j+1}$. For the predictor there are various options. If $\mathbf{u}'_{j+1} = \mathbf{u}_j + \Delta\mathbf{u}'_j$, then the prediction $\mathbf{u}'_{j+1}$ can be based on

- constant extrapolation: $\Delta\mathbf{u}'_j = 0$,
- secant approximation: $\Delta\mathbf{u}'_j = \Delta\mathbf{u}_{j-1}\Delta\lambda_j/\Delta\lambda_{j-1}$,
- tangent approximation: $\Delta\mathbf{u}'_j = -\Delta\lambda_j J_F^{-1} F_\lambda$, where J_F and F_λ are the Jacobian matrix of $F(\mathbf{u}, \lambda)$ and the derivative of F with respect to λ at $(\mathbf{u}_j, \lambda_j)$, respectively.

Note that in the first step, the secant predictor cannot be used and usually one resorts to the constant predictor. The tangent predictor is the most accurate method compared to the other two. However, it requires solving an extra linear system.

Concerning the corrector, to solve the nonlinear system of algebraic equations $F(\mathbf{u}, \lambda) = 0$, there are two ways in general. One is Newton's method and the other is Picard iteration. Though Newton's method has quadratic convergence rate, it is not used often in computational fluid dynamics because the linear system with the complete Jacobian matrix may loose favourable properties like a positive definite submatrix for the convection-diffusion part. Picard iteration does not suffer from this problem, but only has linear convergence rate. Normally, the accuracy of an approximate solution can be improved by several digits in one or two Newton steps, while with Picard iteration it may take tens of iterations. Therefore, the increased number of nonlinear iteration counteracts the benefits of a good linear solver [24, 76]. Our hybrid multilevel solver HYMLS, described later on, can deal quite well with the linear system in Newton's method. Hence, we use Newton's method in our program, by which we achieve efficiency in both linear and nonlinear convergence.

Natural continuation usually works fine as long as there are no turning points in the bifurcation diagram. In case of a turning point, the method will fail because locally there is no solution for parameter values λ bigger than its value at the turning point, e.g. in Fig. 2.1 there is no solution at λ_{j+2}, and hence the correction step will fail.

A general approach to circumvent this break down is the *pseudo-arclength* continuation method. In general we can introduce a new parameter and step in that parameter. Since we are not actually interested in the arclength we like to prescribe the step Δs_j only, see Fig. 2.1. Therefore, we write the new system in terms of increments

$$F(\mathbf{u}_j + \Delta\mathbf{u}_j, \lambda_j + \Delta\lambda_j) = 0,\ c(\Delta\mathbf{u}_j, \Delta\lambda_j, \Delta s_j) = 0,$$

Δs_j approximates the arclength if we take $c(\Delta\mathbf{u}_j, \Delta\lambda_j, \Delta s_j) \equiv (\Delta\mathbf{u}_j, \Delta\mathbf{u}_j) + (\Delta\lambda_j)^2 - (\Delta s_j)^2$ by Pythagoras' theorem. This choice is nonlinear in the increments we want to compute. Since it is only needed to get through the turning point we can approximate it by a linear one by defining c as

$$c(\Delta\mathbf{u}_j, \Delta\lambda_j, \Delta s_j) \equiv (\Delta\mathbf{u}_j, \Delta\mathbf{u}_{j-1}) + \Delta\lambda_j\Delta\lambda_{j-1} - \Delta s_j\Delta s_{j-1}.$$

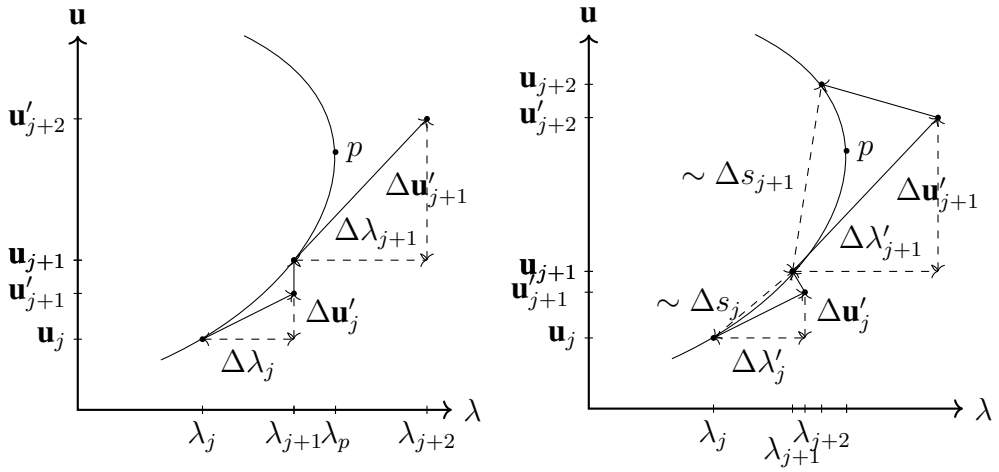
Sometimes weights are also added for the first and/or second term, but this definition of c is the essence of the pseudo-arclength method as first published in [22]. For the new system of nonlinear equations $G(\Delta\mathbf{u}, \Delta\lambda_j, \Delta s_j) \equiv F(\mathbf{u}_j + \Delta\mathbf{u}, \lambda_j + \Delta\lambda) = 0,\ c(\Delta\mathbf{u}, \Delta\lambda, \Delta s_j) = 0$ we can apply Newton's method. The Jacobian of G is now

$$J_G = \begin{pmatrix} J_F(\mathbf{u}_{j+1}^{(k)}, \lambda_{j+1}^{(k)}) & F_\lambda(\mathbf{u}_{j+1}^{(k)}, \lambda_{j+1}^{(k)}) \\ (\Delta\mathbf{u}_{j-1})^T & \lambda_{j-1} \end{pmatrix}.$$

We call this a bordered system since the Jacobian of F is extended by one column on the right and one row at the bottom. Often one has a solver for a system with J_F. One can use that also here if one makes a block LU factorization of the above matrix of the form

$$\begin{pmatrix} J_F & 0 \\ (\Delta\mathbf{u}_{j-1})^T & 1 \end{pmatrix} \begin{pmatrix} I & J_F^{-1}F_\lambda \\ 0 & \lambda_{j-1} - (\Delta\mathbf{u}_{j-1})^T J_F^{-1}F_\lambda \end{pmatrix}.$$

Using this factorization one has to solve twice a system with matrix J_F. Though this is elegant, a problem occurs near a turning point, where the matrix J_F is singular. Therefore, we advocate to deal with the constraint in the solver as indicated in [21], by integrating the border rows and columns into the system and using multilevel preconditioners.



(a) Natural continuation (b) Pseudo-arclength continuation

Figure 2.1: Sketch of (a) natural continuation and (b) pseudo-arclength continuation method. Each step consists of a prediction in the direction of the tangent followed a correction by Newton's method. Natural continuation fails to find correction at parameter value λ_{j+2}.

Compared to time integration methods, a continuation scheme is more efficient, especially when computing the whole bifurcation diagrams or when slowly decaying modes lead to very slow approaches to equilibrium of the time-dependent simulation. Moreover, by continuation, we can investigate the stability of various solution branches and capture the unstable solutions by eigenpair computation of the Jacobian matrix after obtaining the steady state. Our continuation program is developed based on LOCA, which is a generic continuation and bifurcation analysis package of Trilinos designed for large-scale applications. For different problems, the user should generate corresponding right-hand side, Jacobian matrix and mass matrix.

2.1.1 Generating right-hand side, its Jacobian and mass matrix

In this section, we explain how the right-hand side, its Jacobian and the mass matrix are generated. We will take the incompressible Navier-Stokes equations as an example, the inclusion of transport equations is straightforward. We write the

continuous equations in the following form

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathcal{N}(\mathbf{u}, \mathbf{u}) + \frac{1}{\mathrm{Re}}\mathcal{L}\mathbf{u} - \nabla p, \tag{2.1}$$

$$0 = \nabla \cdot \mathbf{u}, \tag{2.2}$$

where $\mathcal{N}(\mathbf{u}, \hat{\mathbf{u}})$ is the bilinear form defined by the convection terms and $\mathcal{L}$ the Laplace operator. This bilinear form can be written in terms of linear operators $\mathcal{A},\ \mathcal{B}$ and $\mathcal{C}$ as $\mathcal{N}(\mathbf{u}, \hat{\mathbf{u}}) = \mathcal{C}((\mathcal{A}\mathbf{u})(\mathcal{B}\hat{\mathbf{u}}))$. We can exploit this structure in the discretization and in the implementation.

The discretization, which will be discussed in section 2.2.2, leads to a system of ordinary differential equations (ODEs)

$$M\frac{d\mathbf{u}}{dt} = -N(\mathbf{u}, \mathbf{u}) + \frac{1}{Re}L\mathbf{u} - Gp, \tag{2.3}$$

$$0 = D\mathbf{u}, \tag{2.4}$$

where here $\mathbf{u}$ and p have become vectors now representing the velocity and pressure in each grid point, respectively. $N(\cdot, \cdot)$ is the discretized variant of $\mathcal{N}(\cdot, \cdot)$; G is the discretization of the gradient operator; D is the discretization of the divergence operator and M is the mass matrix, containing the volumes of the control volumes on its diagonal. If we freeze one of the variables in a bilinear form then it becomes linear, hence one can write

$$N(\mathbf{u}, \hat{\mathbf{u}}) = N_1(\mathbf{u})\hat{\mathbf{u}} = N_2(\hat{\mathbf{u}})\mathbf{u}, \tag{2.5}$$

where we can express $N_1(\mathbf{u})$ and $N_2(\hat{\mathbf{u}})$ in the the discretized variants of the operators $\mathcal{A},\ \mathcal{B}$ and $\mathcal{C}$ as $A,\ B$ and C, respectively. More precisely, $N_1(\mathbf{u}) = C\mathrm{diag}(A\mathbf{u})B$ and $N_2(\hat{\mathbf{u}}) = C\mathrm{diag}(B\hat{\mathbf{u}})A$, where $\mathrm{diag}(v)$ means a square diagonal matrix with the elements of vector v on the main diagonal. The system with the Jacobian, which typically has to be solved in a Newton step is of the form

$$\begin{pmatrix} -N_1(\mathbf{u}) - N_2(\mathbf{u}) + \frac{1}{\mathrm{Re}}L & -G \\ D & O \end{pmatrix} \begin{pmatrix} \Delta\mathbf{u} \\ \Delta p \end{pmatrix} = - \begin{pmatrix} f_{\mathbf{u}} \\ f_p \end{pmatrix}.$$

Matrix Double Purpose (MDP) property Observe that the matrix has a linear part consisting of diffusion, gradient and divergence operators and 2 nonlinear parts. Note that if we skip one of these parts from the Jacobian, then, due to (2.5), when multiplying by $(\mathbf{u}, p)^T$, we get the right-hand side of (2.3). We can exploit

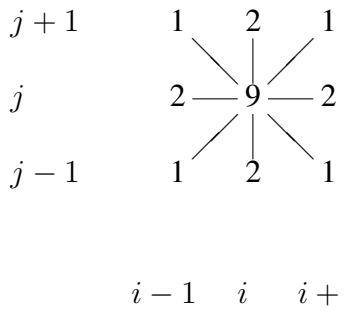


Figure 2.2: Representation of 2D difference $9u_{i,j} + 2(u_{i,j-1} + u_{i,j+1} + u_{i-1,j} + u_{i+1,j}) + u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1}$ in stencil form.

this in the implementation by first building the matrix for the right-hand side, next creating the right-hand side, and finally adding the other term to the matrix to get the full Jacobian. Of course, this holds only if the nonlinear part is bilinear.

The application scientist has to create a number of basic routines, e.g. for the computation of the right-hand side, the Jacobian matrix and the mass matrix. These routines can be written in FORTRAN90 or C++; the interface is prescribed. In FVM, we implemented this using stencil arrays. A stencil is a common tool to describe the discretization on a Cartesian grid. In Fig. 2.2, an example is given for the 2D-case. This is a 9-point stencil. If we extend this to the 3D case we will get the used 27-point stencil. This is for a scalar case, but if we have a coupled PDE then the coupling to an other type of unknown can also be described by a 27-point stencil. These stencils may vary from point to point so the associated array runs over all grid points. In this way, we obtain the stencil arrays give in Table 2.4. The parallelization of the discretization is performed by domain decomposition. The computational domain is subdivided in rectangular subdomains. In order to find the results for the right-hand side in a certain subdomain, we need also the solution from neighbouring domain due to the fact that application of the stencil near the boundary will ask for an unknown from the neighbouring domain. This is resolved by having the solution available on overlapping subdomains.

We did implement the part, which must be provided by the user, as follows.

FVM.Initialization Based on (i) x-,y- and z-coordinates on a rectangular subdomain, (ii) the initial solution on the corresponding overlapping subdomain

and (iii) a mask array determining the geometry and boundary conditions on the overlapping subdomain, we build the stencil arrays corresponding to second-order accurate finite volume discretization and the bilinear form on the disjoint subdomain, i.e., `Al` and `BiL` in Table 2.4 (to be discussed in section 2.2.2).

FVM.Compute_right-hand_side Using the stencils for the bilinear form, create a stencil for $N_1(\mathbf{u})$, where $\mathbf{u}$ is the current solution, and store in `AnlF` (see Table 2.4). Together with stencils for linear part, current solution and forcing compute the right-hand side.

FVM.Compute_Jacobian_matrix Using the stencils for the bilinear form, create a stencil for $N_2(\mathbf{u})$ and add this to `AnlF` to get `AnlJ` (see Table 2.4). Construct the Jacobian matrix in CSR format, which is the format we use in the solver.

FVM.Compute_mass_matrix Since in our case the mass matrix is diagonal, we just compute the diagonal entries. However, in the Rayleigh-Bénard problem we will split up the Jacobian matrix to find bifurcation points from a generalized eigenvalue problem, see section 5.1. For flexibity, we use CSR format too for the mass matrix within the C++ part.

2.1.2 Parallel continuation progam

A short overview of the parallel continuation algorithm is given below.

Cont.initialization

- Partitioning of the domain into rectangular sub-domains. Based on this, two maps are generated: one for overlapping domains and one for non-overlapping domains.
- Initialize solution on the overlapping domains
- Call FVM.Initialization to compute stencils for linear and bilinear forms for each overlapping domain

Cont.LOCA

- Compute the solution on non-overlapping domains using the Newton method (using the NOX package of Trilinos).
  - Call FVM.Compute_right-hand_side and FVM.Compute_Jacobian_matrix to compute the right-hand side and its Jacobian matrix on the non-overlapping subdomains.
  - Solve the linear system; a solution is found on the non-overlapping subdomains (see section 2.3).
  - Solutions are copied to overlapping domains.
- Eigenvalue computation (using Anasazi package of Trilinos or PHIST, see section 2.4)
  - Call FVM.Compute_Jacobian_matrix and FVM.Compute_mass_matrix to get the needed Jacobian and the mass matrix
  - Solve a linear system with a transformed matrix (see section 2.4).

The linear systems arising in the continuation process and eigenvalue computation can be solved with classical iterative methods, such as GMRES, BiCGStab as well as their variations. In our program, we choose restarted GMRES, which is available in the BELOS package of Trilinos. To accelerate convergence we use a preconditioner (see section 2.3).

2.2 Design of finite volume package FVM

Aim of this section is to show in more detail the implementation of the finite volume discretization in the FVM package. In the first part, the basic description of used parameters and notations is presented. Then, specific details of discretization are described, especially the nonlinear terms. Finally, we will discuss how boundary conditions are treated.

2.2.1 Positioning of the unknowns and intermediate variables

For the positioning of the unknowns we use staggered C-grids which is quite common for the discretization of incompressible fluids. Fig. 2.3 shows the positioning of variables, differences (indicated by δ) and averages (indicated by a bar) of them

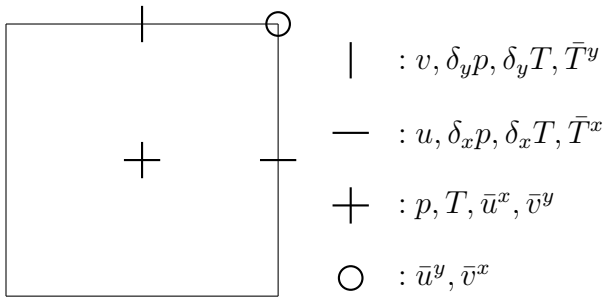


Figure 2.3: Positioning of variables, differences and averages with same index (i, j) in 2D case.

having the same index (i, j) for the 2D case. The 3D case is just an extension in a similar way. The direction of the differencing is indicated by a subscript and for the average it is found at the end of the bar indicating the averaging. The differences and averages are always taken of the two nearest unknowns of the same type, leading to central second-order discretizations on smoothly varying grids.

In Table 2.1, the naming conventions used throughout the program to specify the type of a variable are given. It is shown for the u-unknown, but it holds similarly for the other unknowns. Consider `MxU`. Here, `M` indicates an averaging and the `x` that the averating is in x-direction. The `U` means that u is averaged. Note that due to the positioning conventions made in Fig. 2.3, the averaging will be forward if velocity and coordinate are in the same direction, otherwise it will be backward. So for `MxU` it will be backward. We also introduce a type (hence a `t` is added in front as in `tMxU`) for each variable we have in the program. We will need this later to indicate in the stencil arrays (see Table 2.4) to which unknown the equation is associated and which unknown is targeted in that equation. So `MxU` specifies an array holding the backward averages at the + position of u; `tMxU` is a unique identifier for the `MxU` array. In Table 2.2 we give an overview of more variables with types, names and positions as indicated in Fig. 2.3. In this table, one also define a few operators which can be viewed as the matrix needed to get the result (say `MxU`) from the available variable, e.g., how `MxU` is obtained from `U`. We will need these later on in this section to explain implementation of nonlinear terms.

In Table 2.3 and 2.4, we define the constants and stencil arrays used in the program. The `l` after `A` denotes linear, `nl` after `A` denotes nonlinear, moreover, `F` denotes that it is for the right-hand side and `J` is for the Jacobian of the right-hand side. The function `indg(l,m,n)` defines a unique map between the `l,m,n`,

notation	description	role
$\bar{u}^x_{ijk}$	average of u in x direction	used in the positioning explanation
`MxU(i,j,k)`	backward average of u in x direction	array element holding the value of $\bar{u}^x_{ijk}$
`MyU(i,j,k)`	forward average of u in y direction at point ○	array element holding the value of $\bar{u}^y_{ijk}$
`tMxU`	average of u in point +	parameter in `AL` and `BiL` array referring to $\bar{u}^x_{ijk}$
`tMxVMyU`	vu at point ○	parameter of `BiL` array to get the difference coefficients

Table 2.1: Notations used throughout the program to specify the meaning of a variable, exemplified by the u-unknown.

term	type	variable/ product of variables	location	operator
u	tU	U	–	
v	tV	V	|	
$\bar{u}^x$	tMxU	MxU	+	oMxU
$\bar{v}^y$	tMyV	MyV	+	oMyV
$\bar{T}^x$	tMxT	MxT	–	oMxT
$\bar{T}^y$	tMyT	MyT	|	oMyT
$\bar{u}^x\bar{u}^x$	tMxUMxU	MxU*MxU	+	
$\bar{v}^x\bar{u}^y$	tMxVMyU	MxV*MyU	o	
$u\bar{T}^x$	tUMxT	U*MxT	–	

Table 2.2: Correspondence between the type of unknown, the type as used in the program, the name of the unknown in the program, the location of the unknown and (in some cases) an operator.

all running over the set $\{-1, 0, 1\}$, and the numbers $\{1 : 27\}$, here $np = 27$. So if `(l,m,n)=(0,-1,1)` then $v_{i+l,j+m,k+n}$ indicates $v_{i,j-1,k+1}$. The coefficient multiplying $v_{i,j-1,k+1}$ to give a linear contribution to the right-hand side of the u-equation at (i, j, k) is stored in `Al(i,j,k,indg(0,-1,1),tU,tV)`.

2.2.2 Discretization and its implementation

In the following, more details are given on how the equations are discretized, and how that is implemented using the introduced notations. We use a non-uniform Cartesian grid. This allows us to refine the areas close to boundaries in order to capture boundary layers. For the description we restrict to the xy-plane. The extension in the z-direction is straightforward using similar reasoning. In Fig. 2.4 a picture is given of the non-uniform grid around the point (x_i, y_j), where $\Delta x_i = x_i - x_{i-1}$, $\Delta y_j = y_j - y_{j-1}$, $\widetilde{\Delta x}_i = \frac{1}{2}(x_{i+1} - x_{i-1})$ and $\widetilde{\Delta y}_j = \frac{1}{2}(y_{j+1} - y_{j-1})$. Moreover the control volumes for u_{ij} and T_{ij+1} are also indicated in it; below they

constant	description
`ndim`	number of spatial dimensions
`ndof`	number of degrees of freedom per grid cell
`npar`	number of model parameters
`nx,ny,nz`	number of grid cells in x-, y- and z- direction
`np`	stencil size, here 27

Table 2.3: Constants used in the program.

array	stencil
`Al(1:nx,1:ny,1:nz,1:np,1:ndof,1:ndof)`	linear part
`BiL(1:nx,1:ny,1:nz,1:np,1:2*(ndof-1),1:ndof-1)`	bilinear part operators
`AnlF(1:nx,1:ny,1:nz,1:np,1:ndof-1,1:ndof-1)`	nonlinear part in right-hand side
`AnlJ(1:nx,1:ny,1:nz,1:np,1:ndof-1,1:ndof-1)`	additional nonlinear part in Jacobian matrix

Table 2.4: Arrays used to store the stencils in the program.

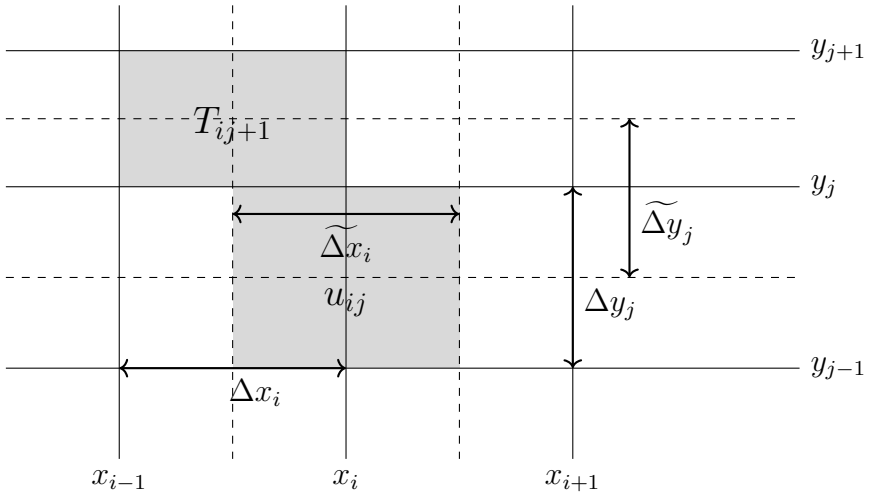


Figure 2.4: Grid and control volumes of u_{ij} and T_{ij+1} in gray, 2D case.

will be referred to as Ω^u_{ij} and Ω^T_{ij+1}, respectively. The starting point for the finite volume discretization is the integral form of the equations written in conservation form. A PDE in conservation form has the shape

$$\frac{\partial \mathbf{u}}{\partial t} + \text{div } \mathbf{F}(\mathbf{u}, \lambda) = q, \tag{2.6}$$

where F denotes the so-called flux functions and q describes sources and sinks. The integral of this equation over a control volume Ω_h with boundary Γ_h and, using Gauss' theorem, can be rewritten to

$$\int_{\Omega_h} \frac{\partial \mathbf{u}}{\partial t} d\Omega_h + \int_{\Gamma_h} \mathbf{F}(\mathbf{u}, \lambda) \cdot \mathbf{n} d\Gamma_h = \int_{\Omega_h} q d\Omega_h. \tag{2.7}$$

Linear part The discretization of the linear terms in the equations, i.e. diffusion terms and gradient and divergence are all using central second-order accurate finite volume discretizations. This approach is straightforward and the results are stored in the `Al` array.

Nonlinear part It can be shown [77] that on closed domains it holds in (2.1) that $\int_\Omega \mathbf{u}\mathcal{N}(\mathbf{u}, \hat{\mathbf{u}}) d\Omega = 0$ for any divergence free $\hat{\mathbf{u}}$. Hence dissipation of kinetic energy in a domain enclosed by walls can only occur by diffusion. We would like to preserve this property in the discretization in order to preclude artificial diffusion.

On the control volumes defined in Fig. 2.4 we have the following discretizations of the nonlinear terms:

$$\begin{aligned}
\int_{\Omega^u_{ij}} (uu)_x \,\mathrm{d}V &\approx \Delta y_j((\bar{u}^x_{i+1,j})^2 - (\bar{u}^x_{i,j})^2),\\
\int_{\Omega^u_{ij}} (vu)_y \,\mathrm{d}V &\approx \widetilde{\Delta x}_i(\bar{v}^x_{i,j}\bar{u}^y_{i,j} - \bar{v}^x_{i,j-1}\bar{u}^y_{i,j-1}),\\
\int_{\Omega^v_{ij}} (uv)_x \,\mathrm{d}V &\approx \widetilde{\Delta y}_j(\bar{u}^y_{i,j}\bar{v}^x_{i,j} - \bar{u}^y_{i-1,j}\bar{v}^x_{i-1,j}),\\
\int_{\Omega^T_{ij}} (uT)_x \,\mathrm{d}V &\approx \Delta y_j(u_{i,j}\bar{T}^x_{i,j} - u_{i-1,j}\bar{T}^x_{i-1,j}).
\end{aligned} \tag{2.8}$$

In the 3D-case, the volume at cell (i, j, k) is just the volume of the 2D-case at (i, j) times Δz_k. It can be shown that this discretization has the desired property. Next we explain how we deal with these nonlinear terms in the program by taking the term $(vu)_y$ as an example. For easy understanding, we introduce operators `oMxV, oMyU` and `oCyMxVMyU` denoting the matrices of forward averaging in x- and y-direction to be applied to `V` and `U`, and backward difference matrix in y-direction to be applied to `MxV*MyU`, respectively.

Using MATLAB notation we can now easily write $(uv)_y$ in three different ways as: (i) `oCyMxVMyU*(MxV.*MyU)`, (ii) `(oCyMxVMyUC*diag(MxV)*oMyU)*U`, and (iii) `(oCyMxVMyU*diag(MyU)*oMxV)*V`. The coefficients for `oMxV` to get `MxV` from `V` and for `oMyU` to get `MyU` from `U` are stored in the `BiL` array as `BiL(:,:,:,:,tMxV,tV)` and `BiL(:,:,:,:,tMyU,tU)`, respectively; `Bil` stands for bilinear.

The coefficients of `oCyMxVMyU` are in `BiL(:,:,:,:,tU,tMxVMyU)`. With the matrix `(oCyMxVMyUC*diag(MxV)*oMyU)` one can create the nonlinear right-hand side using the MDP property described in section 2.1.1 and store its stencil in `AnlF`. In the Jacobian there is one extra nonlinear contribution next to the one already in `AnlF` (see the MDP property); it is precisely the stencil of the second matrix, i.e., `oCyMxVMyU*diag(MyU)*oMxV`, which is stored in `AnlJ(:,:,:,:,tU,tV)`.

Once we have defined this, the storage of other nonlinear terms follows by making appropriate replacements for the unknowns. This is also exploited in the implementation, by defining a generic template that generates all the contributions for the u equation. Using "define"s we let the compiler's preprocessor replace the unknowns and generate the appropriate code. So the template can

also be used for the v and w equations exploiting the rotational invariance of the Navier-Stokes equations. For instance, in three dimension, the derivative of u in x direction u_x is defined as $(u_{i,j,k} - u_{i-1,j,k})/\Delta x_i$. Multiplied by the cell volume, it is $\Delta y_j \Delta z_k (u_{i,j,k} - u_{i-1,j,k})$. We have a source file GenSpf.src, which defines the coefficients of the stencil:

```
ind(i,j,k)=1+ _INDEXPR_
! u_x for divergence
do i=1,nx
  do j=1,ny
    do k=1,nz
      atom(i,j,k,ind(-1,0,0)) =_DELYJ_*_DELZK_
      atom(i,j,k,ind(0,0,0)) =-atom(i,j,k,ind(-1,0,0))
    enddo
  enddo
enddo
#undef  _DELYJ_
#undef  _DELZK_
```

The function `ind` maps a position in the stencil to a number between 1 and 27. In general, the arguments of the function `ind(i1,j1,k1)` can have values -1, 0 and 1, indicating that the coefficient is meant to multiply $u_{i+i1,j+j1,k+k1}$. In the Fortran code, the coefficient `atom()` is given the value as below and used in the subroutine about all the computations of variable u:

```
!Compute u_x
#define _INDEXPR_ (i+1)+(j+1)*3+(k+1)*9
#define _DELYJ_ (Y(J)-Y(J-1))
#define _DELZK_ (Z(K)-Z(K-1))
#include "GenSpf.src"
```

For v_y we use the same template and just redefine directions and also redefining the ind function by interchanging the role of i and j in it:

```
!Compute v_y
#define _INDEXPR_ (j+1)+(i+1)*3+(k+1)*9
#define  _DELYJ_  (X(I)-X(I-1))
#define  _DELZK_  (Z(K)-Z(K-1))
#include "GenSpf.src"
```

Similary, we compute the stencil for w_z.

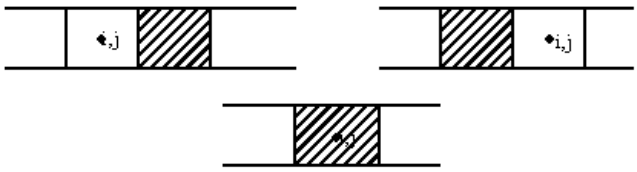


Figure 2.5: Three locations (i, j) where the field stencil needs adjustment due to a nearby no-fluid celll.

2.2.3 Boundary conditions

The geometry is defined by a mask array called `landm`. Currently, an entry of this mask array can have four values: `FLUID, SOLID, MLID, TBC`. If the value of an entry is not `FLUID`, this means that there is no flow in the corresponding mass conservation cell/volume, see Fig. 2.5. If the value of this cell is `SOLID`, all velocities around this cell are zero. A sliding wall or moving lid is indicated by `MLID`, which means that the tangential velocity at the `FLUID-SOLID` interface is prescribed. The standard boundary condition for the temperature at a wall is the no-conduction condition, but if the value is `TBC` the temperature is prescribed at the wall.

In the algorithm, we first enter the field discretization in the stencil arrays `Al` and `BiL`, see Tables 2.5 and 2.6. Next, we can use the entries of the stencil array to implement the boundary conditions. Since we have closed walls, the speed of the moving lid and the temperature at the wall only affect the linear terms. So it modifies the array `Al`. However, if the speed of the moving lid or a temperature prescribed at the wall is not constant during the continuation process, we have to adapt the forcing. Computationally, it is advantageous that the discretization of the boundaries only needs to be done once. From these stencil arrays we can easily compute the right-hand side and the Jacobian matrix.

In the Tables 2.5 and 2.6, we just describe the 2D case; the 3D case is a straightforward generalization. Since in our experiments all our walls are closed, non-trivial boundary conditions only occur in the linear part of the equations. In Table 2.5, it is shown how boundary conditions change the stencil array and the forcing, after the field equations are set. In Table 2.6, we give the stencils for the two linear parts making up for the discretization of the bilinear form.

t1	t2	cond.	loc.	(l,m)	implementation
tU	tU	$u = 0$	-	(1,0)	`Ali(:,:)=0, Ali(0,0)=1`
				(0,0)	`Ali(:,:)=0, Ali(0,0)=1`
				(2,0)	`Ali(1,0)=0`
				(-1,0)	`Ali(-1,0)=0`
tV	tV	$\bar{v}^x = a$	|	(1,0)	`Ali(0,0)=Ali(0,0)-Ali(1,0)`
					`Ali(1,0)=0`
					`F(0,0)=F(0,0)+Ali(1,0)*(2*a)`
				(0,0)	`Ali(:,:)=0, Ali(0,0)=1`
				(-1,0)	`Ali(0,0)=Ali(0,0)-Ali(-1,0)`
					`Ali(-1,0)=0`
					`F(0,0)=F(0,0)+Ali(-1,0)*(2*a)`
tT	tT	$\bar{T}^x = b$	+	(1,0)	`Ali(0,0)=Ali(0,0)-Ali(1,0)`
					`Ali(1,0)=0`
					`F(0,0)=F(0,0)+Ali(1,0)*(2*b)`
				(0,0)	`Ali(:,:,:)=0, Ali(0,0)=1`
				(-1,0)	`Ali(0,0)=Ali(0,0)-Ali(-1,0)`
					`Ali(-1,0)=0`
					`F(0,0)=F(0,0)+Ali(-1,0)*(2*b)`
tT	tT	$\delta_x T = 0$	+	(1,0)	`Ali(0,0)=Ali(0,0)+Ali(1,0)`
					`Ali(1,0)=0`
				(0,0)	`Ali(:,:,:)=0, Ali(0,0)=1`
				(-1,0)	`Ali(0,0)=Ali(0,0)+Ali(-1,0)`
					`Ali(-1,0)=0`

Table 2.5: Boundary implementation in the linear part of the equations. The location of the no-`FLUID` cell is at $(i + l, j + m)$. Here `Ali(l,m)` is a short for `Al(i,j,indg(l,m),t1,t2)` where `t1` and `t2` are the types indicated in the first two columns.

t1	t2	location	(l,m)	stencil
tMxU	tU	+		$[1\ \underline{1}]/2$
			(1,0)	$[\underline{0}\ 1]/2$
			(0,0)	0
			(-1,0)	$[\underline{1}\ 0]/2$
tMxV	tV	o		$[\underline{1}\ 1]/2$
			(1,0)	0
			(1,0)	0
			(0,-1)	0
			(0,0)	0
tCxMxUMxU	tMxUMxU	–		$[-1\ \underline{1}]\Delta y_j$
			(0,0)	0
			(1,0)	0
tCxMxVMyU	tMxVMyU	|		$[-1\ \underline{1}]\widetilde{\Delta y}_j$
			(0,0)	0
			(1,0)	0

Table 2.6: Boundary implementation in the bilinear form determined by two linear parts. The underlining in the stencil notation indicates the central coefficient.The location of the no-FLUID cell is at $(i + l, j + m)$.

2.3 Linear system solvers

In this section, we consider the problem of solving the equation

$$Kx = b, \tag{2.9}$$

where $K \in R^{(n+m)\times(n+m)}$ $(n \geq m)$ is a saddle point matrix that has the form

$$K = \begin{pmatrix} A & G \\ G^T & 0 \end{pmatrix}, \tag{2.10}$$

with $A \in R^{n\times n}$, $G \in R^{n\times m}$. For the Stokes problem discretized on a C-grid (Fig. 2.3), K is a so-called $\mathcal{F}$-matrix (A is symmetric positive definite, and G has row sum zero and at most two entries per row [78]).

In our experiments, we mainly use three linear solvers: smoothed AMG from Trilinos package ML, the Trilinos package Teko designed for multiphysics applications, and our home-made multilevel solver HYMLS. These linear solvers are not exclusively meant for saddle point problems but could also be used for linear systems arising from many other applications, for instance, diffusion-convection systems discussed later in this thesis.

2.3.1 Algebraic multigrid solver

The ML library, the algebraic multilevel preconditioning package of Trilinos contains a variety of parallel multigrid schemes, which includes smoothed aggregation, FAS nonlinear multigrid and a special algebraic multigrid for the eddy current approximations to Maxwell's equations. Smoothed aggregation is used in some of our experiments.

A multigrid solver tries to obtain an approximate solution of the original problem on a hierarchy of grids and uses the approximate solutions from coarser grids to accelerate the convergence on finer grids. A simple multilevel iteration is illustrated in Algorithm 1, which shows a high-level multigrid V-cycle consisting of "Nlevel" grids to solve the equation (2.9), with $K_0 = K$. In the algorithm, S_i^1 and S_i^2 are the approximate solvers corresponding to i steps of pre and post smoothing, respectively. The idea of the smoother is to make the underlying error smooth so that it can be approximated accurately on a coarser grid on which the error can be reduced more efficiently than on the original grid. Smoothers are important for the overall performance of a multigrid method, and they must be supplied on each level. There is a variety of smoothers in ML. For instance, Jacobi, Gauss-Seidel

Algorithm 1 x =multilevel(K_i, b_i, i): solve $K_i x = b_i$, i is the current level.
To solve $Ax = b$ call x =multilevel$(A, b, 0)$.

if $i \neq$ Nlevel **then**
  $x = S_i^1(K_i, b_i, x)$;
  P_i =determine-interpolant(K_i); (only once)
  $r = P_i^T(b_i - K_i x)$;
  $K_{i+1} = P_i^T K_i P_i$; (only once)
  v =multilevel$(K_{i+1}, r, i+1)$;
  $x = x + P_i v$;
  $x = S_i^2(K_i, b_i, x)$;
else
  Solve $K_{Nlevel} x = b_{Nlevel}$ by a direct method;
end if

(GS), symmetric GS and block GS [42]. P_i are the essential operators for transfering solutions from coarse grids to finer grids and its transpose P_i^T can serve as a restriction operator. In ML, it is determined automatically by the algebraic multigrid method [79]. The AMG method is typically superior to other preconditioners for Poisson-like problems. Therefore, we use ML in both the continuation and eigenvalue computations for the Turing problem in chapter 6. In that chapter, we compare also the performance of HYMLS and ML in the continuation.

2.3.2 Block preconditioners in Teko

It is well established that block preconditioners present an appealing alternative. The idea of such a preconditioner is to use a block factorization to segregate the linear operator into smaller groups based on its physical components. The resulting sub-blocks can be solved effectively with available efficient software packages, e.g. ML, which have shown having good parallel scalability.

For the incompressible Navier–Stokes equations (2.9) a block LDU factorization is made:

$$K = \begin{pmatrix} I & 0 \\ G^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & A^{-1}G \\ 0 & I \end{pmatrix}, \tag{2.11}$$

where $S = -G^T A^{-1} G$ is the Schur-complement. All couplings between the physical variables is localized to the Schur-complement operator. As a result, the

challenge of a block preconditioning lies in effectively approximating the inverse Schur-complement.

Teko is a package of Trilinos for development and implementation of block preconditioners. Some generic preconditioners have been implemented in Teko, such as block Jacobi, and block Gauss-Seidel. For the Navier-Stokes equations, Teko has implementations of the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE), the Pressure Convection-Diffusion preconditioners (PCD) and the Least Squares Commutators (LSC) [39].

PCD and LSC preconditioners In our experiments, we will use the LSC preconditioner. This is a variant of the PCD preconditioner. The idea of the PCD is the observation that the operators $\partial/\partial x$ and $\partial/\partial y$ commute approximately with the $a(x,y)$ in $a(x,y)u$, i.e.,

$$\frac{\partial}{\partial x}(a(x,y)u) = a(x,y)\frac{\partial}{\partial x}u + (\frac{\partial}{\partial x}a(x,y))u$$

So if $a(x,y)$ is rather smooth we can neglect its derivative and then the first term in the right-hand side approximates the left-hand term. Now consider the approximation div$\mathcal{C} \approx \mathcal{C}$div where $\mathcal{C}$ is the convection-diffusion operator. Then, in matrix form, we have the approximation

$$G^T M_A^{-1} A \approx B M_B^{-1} G^T, \tag{2.12}$$

where M_A is the part of the mass matrix related to A, so $M_A^{-1}A$ indeed approximates the convection-diffusion operator acting on all the velocity components, and BM_B^{-1} is the convection-diffusion operator acting on the pressure. From (2.12), it follows that $G^T M_A^{-1} \approx BM_B^{-1} G^T A^{-1}$ and therefore $S = -G^T A^{-1} G \approx (BM_B^{-1})^{-1} G^T M_A^{-1} G)$. So

$$S^{-1} \approx -(G^T M_A^{-1} G)^{-1}(BM_B^{-1}).$$

Hence, solving a system with the matrix S means that we need to compute (BM_B^{-1}). In PCD this is explicitly computed, but in LSC one takes the least-squares solution of (2.12):

$$(BM_B^{-1}) \approx \text{argmin}_X(||GX^T - (G^T M_A^{-1} A)^T||_{M_A^{-1}})$$

So $(BM_B^{-1}) \approx (G^T M_A^{-1}(G^T M_A^{-1} A)^T)^T (G^T M_A^{-1} G)^{-1}$ and hence

$$S^{-1} \approx -(G^T M_A^{-1} G)^{-1}(G^T M_A^{-1} A M_A^{-T} G)(G^T M_A^{-1} G)^{-1}.$$

This means we have to solve twice an elliptic equation with the symmetric matrix $G^T M_A^{-1} G$, which is a discrete Laplacian.

2.3.3 HYMLS

In [80] a direct method for the solution of $\mathcal{F}$-matrices was proposed. It reduces fill and computation time while preserving the structure of the equations during the elimination. A hybrid direct/iterative method based on this approach was presented in [81, 82]. It has the advantage that the ordering it defines for the matrix exposes parallelism on each level. It achieves so by partitioning the computational domain into a set of non-overlapping subdomains (the interiors) plus an interface (the separators). After solving the interface problem through the Schur-complement, the problem is reduced to solving the independent systems associated with the subdomains. All the subdomain matrices can be factored independently using sequential sparse direct solvers, and the Schur-complement can be constructed with a minimal amount of communication in an assembly process. The general idea of partitioning can be illustrated as follows.

For a general large (sparse) problem $K\vec{x} = \vec{b}$, after partitioning it into n subdomains and reordering the unknowns the problem can be written in the form

$$\begin{pmatrix} K_{11} & & & K_{1S} \\ & \ddots & & \vdots \\ & & K_{nn} & K_{nS} \\ K_{1S}^T & \dots & K_{nS}^T & K_{SS} \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vdots \\ \vec{x}_n \\ \vec{x}_S \end{pmatrix} = \begin{pmatrix} \vec{b}_1 \\ \vdots \\ \vec{b}_n \\ \vec{b}_S \end{pmatrix}.$$

The diagonal blocks K_{ii} (for $i = 1, \dots, n$) represent the interiors of the n subdomains (i.e. variables $\vec{x}_i$) and the blocks K_{iS} correspond to the couplings between these interior variables and the separator variables x_S. For ease of writing, we combine the block diagonal matrix containing all the interior blocks into a single block K_{II} and their couplings to the separators into K_{IS}, then, elimination of the interior variables formally yields the block LU decomposition

$$\begin{pmatrix} K_{II} & K_{IS} \\ K_{IS}^T & K_{SS} \end{pmatrix} = \begin{pmatrix} I & O \\ K_{IS}^T K_{II}^{-1} & I \end{pmatrix} \begin{pmatrix} K_{II} & K_{IS} \\ O & S \end{pmatrix},$$

where

$$S = K_{SS} - K_{IS}^T K_{II}^{-1} K_{IS} = K_{SS} - \sum_{i=1}^{n} K_{iS}^T K_{ii}^{-1} K_{iS}$$

is the Schur-complement of K_{II}. It should be noted that wherever an inverse of a matrix is written, a solver should be employed rather than explicitly computing

the matrix inverse. From this LU decomposition it follows that the partitioned system can now be solved by first solving

$$Sx_S = b_S - \sum_{i=1}^{n} K_{iS}^T K_{ii}^{-1} b_i$$

followed by solving n independent systems

$$K_{ii}x_i = b_i - K_{iS}x_S, \qquad \forall i = 1, \ldots, n,$$

which can be done in parallel.

The essential part of HYMLS is the construction of a preconditioner for solving the unknowns on the separators occurring in the Schur-complement system. First, the Schur-complement is reduced to keep only one pressure unknown in each subdomain. The preconditioner is constructed by applying an orthogonal transformation (Householder transformation) from the left and right to a block row and block column corresponding to a separator group in the Schur-complement. This also transforms the unknowns of each separator group and we call the first one a V_Σ unknown and the remaining ones non-V_Σ unknowns, since it can be shown that the first one is an average of the original unknowns. Next, dropping all connections between non-V_Σ unknowns and V_Σ unknowns, and between non-V_Σ unknowns in different separator groups yields a block-diagonal preconditioner with dense blocks. Herewith, the fill of the Schur-complement matrix has been reduced significantly.

The difficult task of parallel preconditioning is aided by the above algorithm. The ingredients of the associated incomplete LU factorization are the following:

1. Perform a non-overlapping domain decomposition. These domains are typically small, e.g. edge length 4, and are chosen independently of the partitioning of the computational domain for parallelization.

2. Detect the velocity separators associated to this domain decomposition.

3. Pick for every subdomain one pressure unknown to be kept in the Schur-complement.

4. Eliminate all interior variables of the subdomains and construct the Schur-complement for the velocity separators and the selected pressure unknowns in 3.

5. Perform a Householder transformation on each separator. This decouples most of the velocity unknowns from the remaining pressure unknowns.

6. Identify V_Σ unknowns (separator velocities that still connect to two pressures).

7. Drop all connections between non-V_Σ unknowns and V_Σ unknowns, and between non-V_Σ unknowns in different separator groups. The resulting matrix is block-diagonal with the 'reduced Schur-complement' in the last block defined by the V_Σ and pressure unknowns.

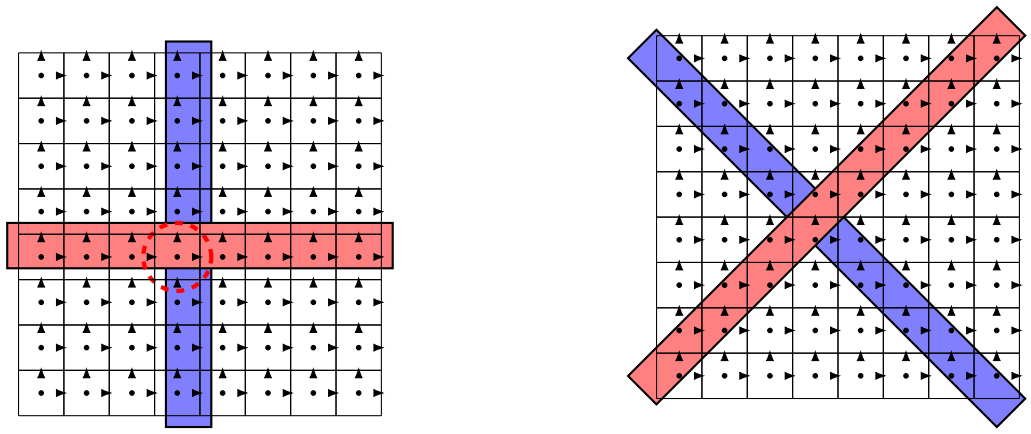
8. Repeat the process on the 'reduced Schur-complement' till the number of levels specified.

9. Make a sparse direct factorization on the last Schur-complement.

To construct a non-overlapping decomposition of the physical domain, we partition the grid geometrically by subdividing the grid manually into equally sized subdomains. The straightforward way to partition the grid is to use *standard* Cartesian partitioning. However, the intersection of horizontal and vertical separators leads to isolated pressure nodes, as indicated by the one in the red dashed circle in Fig. 2.6 (a), all of whose surrounding velocities are separator velocities. This cell is called a *full conservation cell*. To avoid a singular matrix for the interior of the corresponding subdomain, this pressure node cannot be eliminated but instead should be retained in the Schur-complement [24]. Furthermore, the four surrounding velocity nodes in the full conservation cell should also be placed in separate groups as well, which implies that these nodes are retained throughout the multilevel approach. In three dimensions, separators become planes and entire "tubes" of isolated pressure nodes occur at positions where four subdomains meet. It is even more complicated in the corners where eight subdomains meet.

The implementation is intricate and prone to bugs and errors in the code. It also affects the convergence properties of the method. To prevent any occurrences of isolated pressure nodes, *skew* Cartesian separators are the better choice, which are obtained by rotating the separators in standard Cartesian partitioning by 45 degrees. For the 2D case, this is shown in Fig. 2.6 (b). The detailed implementation of skew-Cartesian partition can be found in Van der Klok's bachelor thesis [83].

Fig. 2.7 depicts the process from grid point-of-view, which shows the principle for a coarsening factor of 2, which means that on each subsequent level the separator length is twice that of the previous one. The coarsening factor is an integer bigger than one, and can be chosen by the user. The choice of the separator



(a) Separator length 4

(b) Separator length $4\sqrt{2}$

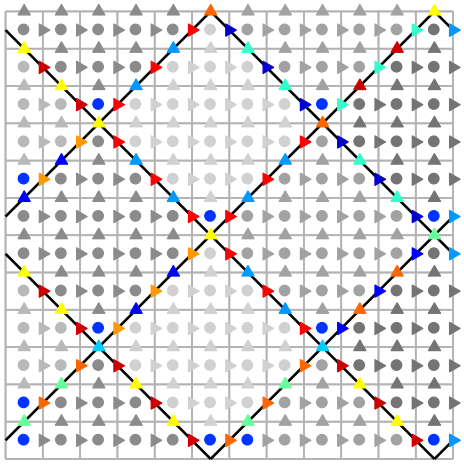
Figure 2.6: Example of partitioning of an 8×8 grid. Separators are highlighted in red and blue. The intersection of separators leads to the isolation of pressure nodes, as highlighted in (a) Cartesian partitioning. (b) Skew partitioning, by placing the separators diagonally on the grid, no longer isolates pressure nodes, but domains along the grid boundary are asymmetric. *Image courtesy of Van der Klok.*

length, the coarsening factor, and the number of levels determine the size of the linear systems on each level.

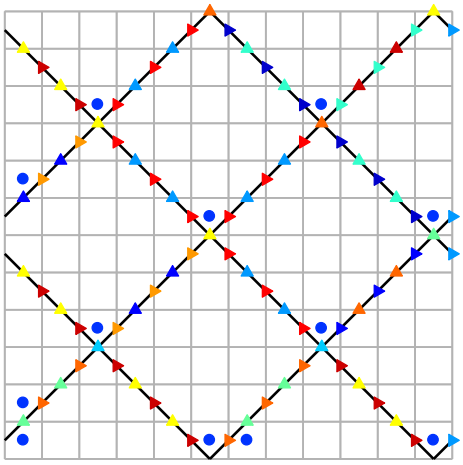
Note that the matrices on each level inherit the structure and numerical properties of the original problem, so the method can be applied recursively. In [81], it is shown that for two levels the amount of iterations is independent of the mesh size. Since we are just repeating the process on the Schur-complement it will be straightforward to show that also for a fixed number of levels the amount of iterations is independent of the mesh size. However with increasing number of levels the number of iterations increases; it does so only very mildly and in a monotonous way. The latter is due to the robustness of the method. The computational complexity depends on the number of iterations and the size of the last Schur-complement. This size increases with the problem size if the number of levels is fixed. By increasing the number of levels it drops significantly.

2.3.4 HYMLS on a Poisson equation

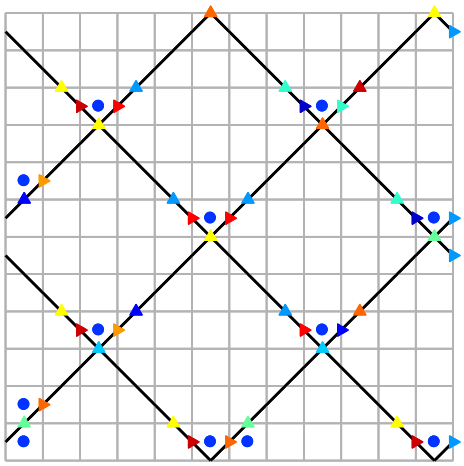
In a later chapter, we will show the behavior of HYMLS on fluid-flow problems. For comparison we give a few results of the method on a Poisson equation de-



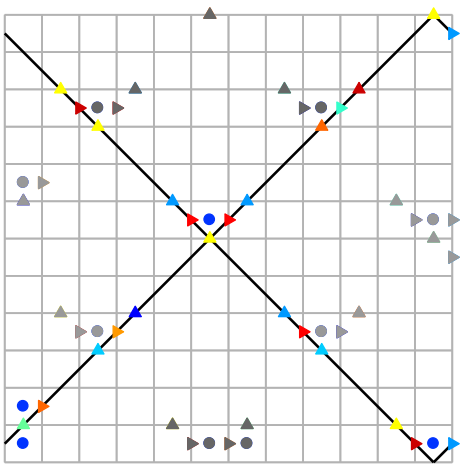
(a) Initial groups



(b) After elimination of interiors only the separators remain



(c) The Householder transformations leaves one V_Σ-node per separator group



(d) Domains on next level

Figure 2.7: Construction of the preconditioner from the grid point-of-view. This example uses a coarsening factor of 2, i.e. the separators on the next level have twice the length of those on the previous level.

np	nx	L	its	t_c	t_s	lss
Cartesian						
1	16	2	22	0.07	0.03	1296
4	32	2	23	0.34	0.07	8379
16	64	2	23	1.02	0.26	76951
128	128	2	24	4.61	1.03	654255
1024	256	2	24	50.89	21.66	5386591
Skew						
1	16	2	19	0.11	0.03	721
4	32	2	28	0.25	0.09	10368
16	64	2	31	0.94	0.43	82944
128	128	2	32	4.30	2.41	663552
1024	256	2	33	181.24	55.50	5308416

Table 2.7: (3D Poisson equation) Typical HYMLS results for constant number of levels L, here $L = 2$, coarsening factor is 2 and separator length is 8; stopping criterion: residual less than 1e-8. *Results courtesy of Baars.*

fined on a cube, using the standard central discretization and Dirichlet boundary conditions. In Tables 2.7 and 2.8, we show results for a constant number of levels and constant last Schur-complement size, respectively. The columns respectively show: the number of cores used (np), the number of grid points in one direction (nx), the number of levels (L), the number of iterations (its), the time needed to compute the factorization (t_c), the time needed to solve the system to 8 digits accuracy (t_s), and the size of the last Schur complement (lss). These experiments have predominantly been performed on the Dutch national computing facility Cartesius, from which we used the so-called thin nodes. Each of them has 2 x 12-core 2.6 GHz Intel Xeon E5-2690 v3 (Haswell) CPUs and 64 GB memory per node. In Table 2.7, we show results for both Cartesian and Skew partition. Both of them work because the problem of isolated pressures does not apply here. We see that the number of iterations (its) is getting constant after a few refinements. This comes at the expense of a growing last Schur-complement (lss). Since for the last Schur-complement we use an exact solver, the factorization of that part dominates the factorization time (t_c) with increasing grid size, and it does also in the solution phase (t_s). Next, we also show some typical results of experiments where we keep the size of the last Schur-complement (lss) fixed. Since our coarsening fac-

np	nx	L	its	t_c	t_s	lss
Cartesian						
4	32	3	26	0.36	0.08	127
16	64	4	34	0.96	0.34	127
128	128	5	42	4.34	0.62	127
1024	256	6	49	23.40	1.26	127
16	64	3	30	0.99	0.29	1647
128	128	4	38	3.00	0.60	1647
1024	256	5	45	9.85	1.78	1647
Skew						
4	32	3	32	0.25	0.27	624
16	64	4	41	1.09	0.37	624
128	128	5	53	4.23	0.84	624
1024	256	6	65	23.25	3.09	624
16	64	3	38	0.81	0.36	4864
128	128	4	50	2.61	0.82	4864
1024	256	5	61	15.08	2.58	4864

Table 2.8: (3D Poisson equation) Typical HYMLS results for constant last Schur-complement size (lss) by increasing number of levels L, coarsening factor is 2 and separator length is 8; stopping criterion: residual less than 1e-8. *Results courtesy of Baars.*

tor (cx) is 2, we have to increase the level by 1 on each refinement to achieve the same last Schur-complement size. We see that the number of iterations increases in the Cartesian case by 7 or 8 on refinement where it does by 8 to 12 in the Skew case. So these experiments show that the number of iterations increases with the logarithm of nx, which is a quite acceptable behavior with increasing problem size.

Table 2.8 also learns us that the weak scalability is not as it should be. For perfect scaling the run time should be constant for problems with nx greater than or equal to 64. We see it increase in the factorization. We found that this is due to communication and improvements will be left to the future.

2.4 Eigenvalue computation

After computing the steady solution, we can analyze its stability by computing the right-most eigenvalues of the Jacobian matrix at this state. Based on the discretization of the incompressible Navier-Stokes equations, we need to solve a generalized eigenvalue problem of the form $Jx = \lambda Mx$ where J is the Jacobian matrix, and M is the mass matrix; the latter matrix is singular. Since we are after eigenvalues close to the imaginary axis we use Shift-and-Invert or the Cayley transform [84] when using Arnoldi's method. These transforms give us back a standard eigenvalue problem. For the Jacobi-Davidson method we exploit that the eigenspace should be in the divergence-free space and use M orthogonality on that space to find a standard eigenvalue problem. We discuss this at the end of the section.

2.4.1 Arnoldi method

The Arnoldi method, based on the Krylov subspace, was first introduced as a direct algorithm for reducing a general matrix into upper Hessenberg form [85]. Then, it was proved that it could be a good iterative technique for approximating eigenvalues of large sparse matrices, especially, when one wants a small number of eigenvalues. The implicitly restarted Arnoldi (IRA) method has been proved successful and implemented in the widely used library ARPACK [26]. To compute the eigenvalues, our program can easily benefit from another Trilinos package Anasazi, which implements several algorithms for the numerical solution of large-scale eigenvalue problems, such as the Block-Krylov-Schur (BKS) method, the Block-Davidson, the TraceMin family and so on[86].

Stewart proposed the Krylov-Schur method in 2001, see details in [87], which is mathematically equivalent to implicitly restarted Arnoldi. However, instead of using the strict upper-Hessenberg form in the Arnoldi decomposition, a Rayleigh quotient matrix is used, which leads to the Krylov decomposition. Compared to IRA, this method has two advantages. The first one is that the converged Ritz vectors are easier to deflate; another advantage is that it can avoid the potential forward instability of the QR algorithm [88], which causes the unwanted Ritz vector remaining in the computation. In our experiments, we use the block variant of the Krylov-Schur method implemented in the Anasazi package with the Shift-and-Invert and Cayley transform strategies to compute the target eigenvalues.

With Shift-and-Invert, one chooses a shift σ such that operator $J - \sigma M$ is not singular. Then the original problem can be transformed into the standard eigenvalue problem

$$Cx = \mu x, \tag{2.13}$$

where $C = (J - \sigma M)^{-1} M$ and $\mu = \frac{1}{\lambda - \sigma}$. It can be seen that the eigenvalues λ of the original problem close to the shift σ are mapped to become exterior eigenvalues of the transformed problem. In Krylov subspace methods, the exterior eigenvalues are the first to be well approximated, which means that they converge firstly. Since we are interested in the eigenvalues close to zero with smallest magnitude, we choose $\sigma = 0$ to transform them into exterior eigenvalues. This approach is especially meaningful if the eigenvalues are real.

With Shift-and-Invert, one can accelerate the convergence towards eigenvalues close to the shift; the Cayley transform can go a step further. One chooses shifts σ and τ such that the matrix $J - \sigma M$ is not singular, then the original problem can be transformed to

$$Cx = \mu x, \tag{2.14}$$

where $C = (J - \sigma M)^{-1}(J - \tau M)$ and $\mu = \frac{\lambda - \tau}{\lambda - \sigma}$.
With the Cayley transform we can emphasize the convergence towards eigenvalues close to the shift σ while suppressing the influence of eigenvalues close to τ, which means we can accelerate the convergence towards interesting parts of the spectrum instead of towards one point in Shift-and-Invert. Cayley transform can be seen as a special form of a general rational transform. For more details see [28].

2.4.2 Jacobi-Davidson QR method

In addition to the Arnoldi method, a great many methods exist for solving eigenvalue problems. Two characteristics of our algorithm guide our choice in particular: the availability of a preconditioner (from continuation process) for the matrix in question and the fact that a sequence of eigenvalue problems for a varying parameter is being solved. In that sense, a better choice is the Jacobi-Davidson QR (JDQR) method or its variant block JDQR.

The Jacobi-Davidson(JD) method, first introduced by Sleijpen and van der Vorst around twenty years ago [27], is the state-of-art method in computing eigenvalues. Since then, many researchers are working on it from a theoretical or an implementation side of view, for a review see [89]. However, for efficient computation and memory usage, a restart strategy has to be applied. This lead to JDQR and JDQZ in 1997 [90] for the standard eigenvalue problem and generalized eigenvalue problem, respectively. JDQR [27, 90], having a quadratic convergence rate, can be started with an approximate subspace (not just one vector) and requires the approximate solution of a linear system, for which the existing solver/preconditioner combination can be used.

Two main phases of JDQR are subspace expansion and subspace extraction. In subspace expansion, the correction equation is solved with Krylov subspace methods, such as GMRES-methods [91], BiCG [92] and so on, as well as their variants, with or without preconditioner. For a review see [93]. The solution of the correction equation will be orthogonalized with respect to the current subspace, and next added to that. In the subspace extraction, the approximate eigenvector is computed, the orthogonal complement of which defines the solution space of the correction to be obtained from the correction equation. This approach can be motivated by the Rayleigh-quotient iteration (RQI) [94].

We will give a concise derivation of the JDQR method in order to explain variants later on. Note that the JDQR method is meant for the standard eigenvalue problem. Later on we will see how we can use that to solve a generalized eigenvalue problem. The starting point for the derivation of the JDQR method is the partial Schur form

$$AQ - QR = 0, \tag{2.15}$$

where Q is an invariant subspace of A and R is an upper-triangular matrix with the eigenvalues associated to the invariant subspace on the diagonal. Following

[28], we can extend Q by solving

$$A \begin{pmatrix} Q & q \end{pmatrix} = \begin{pmatrix} Q & q \end{pmatrix} \begin{pmatrix} R & s \\ 0 & \lambda \end{pmatrix},$$

with orthogonality constraint

$$\begin{pmatrix} Q & q \end{pmatrix}^* \begin{pmatrix} Q & q \end{pmatrix} = I.$$

We can write this as a set of equations into

$$-(A - \lambda I)q + Qs = 0, \tag{2.16}$$
$$Q^*q = 0, \tag{2.17}$$
$$(q^*q)/2 - 1/2 = 0. \tag{2.18}$$

Now define $u = (q, s, \lambda)$ and assume we have an initial guess $u_0 = (q_0, s_0, \lambda_0)$, then we can use Newton's method to solve these equations. First, in Newton's method, we have to solve $\Delta u = (\Delta q_i, \Delta s_i, \Delta \lambda_i)$ in the system

$$\begin{pmatrix} -(A - \lambda_i I) & Q & q_i \\ Q^* & 0 & 0 \\ q_i^* & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta q_i \\ \Delta s_i \\ \Delta \lambda_i \end{pmatrix} = \begin{pmatrix} (A - \lambda_i I)q_i - Qs_i \\ -Q^*q_i \\ -q_i^*q_i/2 + 1/2 \end{pmatrix}, \tag{2.19}$$

and next one updates u_i to compute $u_{i+1} = u_i + \Delta u_i$. This system of equations (2.19) is also referred to as the correction equation. On convergence of the Newton method, q_i and λ_i will become a new eigenpair of A. Also using the found s we can update partial Schur factorization and repeat the process for another eigenpair, etc.

If we make sure q_i has length one and is perpendicular to Q in every step, then the last two entries of the right-hand side become zero. By computing s_{i+1} immediately instead of Δs_i, the system can be rewritten as follows

$$\begin{pmatrix} -(A - \lambda_i I) & Q & q_i \\ Q^* & 0 & 0 \\ q_i^* & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta q_i \\ s_{i+1} \\ \Delta \lambda_i \end{pmatrix} = \begin{pmatrix} (A - \lambda_i I)q_i \\ 0 \\ 0 \end{pmatrix}. \tag{2.20}$$

We can accelerate this by using a subspace spanned by all corrections to compute q. So suppose we have $V_i = (\Delta q_1, \ldots, \Delta q_i)$, we can project the eigenvalue problem $Ax = \lambda x$ to the space spanned by V_i in the following way; we seek a $q_i \in V_i$ and a value λ_i such that

$$V_i^* A q_i = \lambda_i V_i^* q_i,$$

or since $q_i = V_i y_i$

$$V_i^* A V_i y_i = \lambda_i V^* V_i y_i = \lambda_i y_i.$$

This means that q_i is just an approximation of the actual eigenvector on the space V_i, we call it Ritz vector and λ_i Ritz value. Now, we can just keep expanding the space V until the residual $r_i \equiv Aq_i - \lambda_i q_i$ is small enough. The advantage of this is that we do not have to compute the vectors Δq_i with whom we expand V_{i-1} very accurately. Another advantage is that we can keep V when computing the next eigenvector, because we know from Arnoldi that the next eigenvector is also already present in V to some extent.

Also, this method allows for a restart where we only keep an appropriate part of V and discard the rest. Herewith, memory usage can be kept at bay.

As we are only interested in computing Δq_i, it is also convenient to write the system (2.20) without the unknowns s_{i+1} and $\Delta\lambda_i$. Since δq_i should be in the space orthogonal to Q and q_i we find a space of this dimension by premultiplying the first row by $(I - QQ^* - q_i q_i^*)$ which leads to the new correction equation

$$(I - QQ^* - q_i q_i^*)(A - \lambda_i I)\Delta q_i = -(I - QQ^* - q_i q_i^*) r_i. \tag{2.21}$$

As said, this equation does not need to be solved exactly so we can replace $(A - \lambda_i I)$ by an approximating matrix P (for simplicity we assume it is not depending on i). Also from (2.20) we can find the solution of this new correction equation

$$(I - QQ^* - q_i q_i^*) P \Delta q_i = -(I - QQ^* - q_i q_i^*) r_i, \tag{2.22}$$

Define $\tilde{Q} = [Q, q_i]$ and note that $\tilde{Q}\tilde{Q}^* = QQ^* + q_i q_i^*$ and also define $\tilde{Q}_P = P^{-1}\tilde{Q}$. Then we can write

$$P_{\tilde{Q}}^{-1} = (I - \tilde{Q}_P(\tilde{Q}^*\tilde{Q}_P)^{-1}\tilde{Q}^*)P^{-1} \equiv P^{-1}(I - \tilde{Q}(\tilde{Q}^*\tilde{Q}_P)^{-1}\tilde{Q}^* P^{-1}) \tag{2.23}$$

where the equivalence means that we have two equivalent forms. So we find Δq_i from

$$\Delta q_i = -P_{\tilde{Q}}^{-1} r_i. \tag{2.24}$$

It is easy to see that this solution satisfies (2.22) and is orthogonal to $\tilde{Q}$. We call this the uncoupled.

We remark that, though we have pulled apart the application of a preconditioner and the projections, the application of the preconditioner may magnify unwanted components if P is nearly singular. These may be cut out again by the

Algorithm 2 Sketch of JDQR.

```
1: Setup initial subspace
2: while not converged do                          ▷ Outer iteration
3:     Project the problem to a small subspace
4:     Solve the small eigenvalue problem
5:     Calculate an approximation and its residual
6:     Approximately solve the correction equation  ▷ Inner iteration
7:     Orthogonalize the new direction
8:     Enlarge the subspace
9: end while
```

projections, but if the remaining correction may be quite inaccurate hampering the convergence. This maybe even the case when (2.20) is well-conditioned, see [95] for more details.

Formulation (2.24) provides suitable corrections Δq_i for a subspace iteration. If the correction equation (2.21) is solved exactly, then JDQR has asymptotically quadratical convergence rate for the selected Ritz values [27, 96]. The algorithm is shown in Alg. 2. Since the Jacobian matrix from Equ. 1.4 and Equ. 6.1 are not symmetric, and also, some of the eigenvalues have very high multiplicity, it is difficult to obtain target eigenvalues. In [97] and [98], the authors have experimented with block JD, which proved that block JD generally performs well if there is a cluster of eigenvalues.

In our experiments, we choose block variants of the JDQR method, see Alg. 3, which indeed can improve the robustness when it computes multiple or clustered eigenvalues. See results in chapter 6. Instead of calculating one approximated eigenvector in JDQR, it will compute n_b (number of block size) approximated eigenvectors once and solve corresponding n_b correction equations. However, the total number of matrix-vector operations will increase in practice. In [99], the authors have stated that by performance gains through better cache usage on modern CPUs, block-JDQR could be both more efficient and robust than its single vector counterpart.

In solving the n_b correction equations, it is quite similar to a single correction Δq_i from the set of equations (2.21). The difference lies in the fact that we use a deflation of eigenvector approximations that have not converged yet. We use the publicly available PHIST implementation [75] of the block Jacobi-Davidson QR method, which can straightforwardly be used in Trilinos applications. The basic implementation described in [99] was extended to allow using a (left) pre-

Algorithm 3 Sketch of Block-JDQR.

```
1: Setup initial subspace
2: while not converged do                          ▷ Outer iteration
3:     Project the problem to a small subspace
4:     Solve the small eigenvalue problem
5:     Calculate n_b approximations and their residuals
6:     Lock converged eigenvalues
7:     Shrink subspace if required (thick restart)
8:     Approximately solve the n_b correction equations   ▷ Inner iteration
9:     Block-orthogonalize the new directions
10:    Enlarge the subspace
11: end while
```

conditioner as follows: the search space is extended in each outer iteration by n_b corrections obtained as the solutions of the independent left-preconditioned linear systems

$$P_{\tilde{Q}}^{-1}(A - \tilde{\lambda}_j I)\Delta q_j = -P_{\tilde{Q}}^{-1}(Aq_j - \tilde{\lambda}_j q_j), \tag{2.25}$$

where $\tilde{\lambda}_j, q_j, j = 1 \ldots n_b$ are the current approximations to the next few eigenvalues and eigenvectors to converge, respectively, and $\tilde{Q} = [Q, q_1 \ldots q_{n_b}]$ also contains the already converged eigenspace. The right-hand side represents the preconditioned eigenvalue residual. The preconditioner $P_{\tilde{Q}}^{-1}$ is chosen such that it produces vectors in the orthogonal complement of $\tilde{Q}$, i.e., if P^{-1} approximates the action of A^{-1},

$$P_{\tilde{Q}} = (I - \tilde{Q}_P(\tilde{Q}^*\tilde{Q}_P)^{-1}\tilde{Q}^*)P^{-1}, \tag{2.26}$$

$$\tilde{Q}_P = P^{-1}\tilde{Q}. \tag{2.27}$$

Using a proper linear solver can yield good convergence behavior of the overall method.

Since the Jacobian matrix from Equ. 1.4 and Equ. 6.1 are not symmetric, and also, some of the eigenvalues have very high multiplicity, it is difficult to obtain target eigenvalues. In [97] and [98], the authors have experimented with block JD, which proved that block JD generally performs well if there is a cluster of eigenvalues.

From generalized eigenproblem to a standard one In standard JDQR, we cannot solve a generalized eigenvalue problem. We can however formulate a standard

eigenvalue problem on the divergence free space. Based on the discretization of incompressible Navier-Stokes equations, we consider a system of the form

$$\begin{pmatrix} A - \lambda M_A & G \\ G^T & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

where M_A is the nonsingular mass matrix; i.e. a positive definite matrix. We observe that p is just a Lagrange multiplier for solving the system for u with the divergence-free constraint $G^T u = 0$. Now let the columns of W span the kernel of G^T, hence $G^T W = 0$ and if $G^T u = 0$ for some u then there exist a $\hat{u}$ such that $u = W\hat{u}$. With this, the above eigenvalue problem is equivalent to finding a u in the column space of W such that $W^*(A - \lambda M_A)u = 0$, which is a smaller system with non-singular M_A.

Now, say we have some eigenvectors as columns of V,

$$W^* A V = W^* M_A V D, \tag{2.28}$$

with D a diagonal matrix with the eigenvalues on the diagonal. Note that in the case we don't have to restrict to a certain space, we can use $W = I$ bringing it back to the standard case. We can write the partial Schur-form as

$$W^* A Q = W^* M_A Q R,$$

where $V = Q\hat{R}$ is the QR-factorization of V and $R = \hat{R} D \hat{R}^{-1}$ is an upper-triangular matrix with the eigenvalues on the diagonal.

Since the column space of V is a subspace of the column space of W we have from (2.28) that

$$V^* A V = V^* M_A V D.$$

Moreover, D is also unique in this equation if $V^* M_A V$ is nonsingular, because it inherits the positive definiteness of M_A.

In the JDQR method for the standard eigenvalue problem, eigenvalues of $Q^* A Q x = \lambda x$ are computed. We also want the standard eigenvalue problem, therefore, instead of being orthogonal, Q should be M_A-orthogonal, i.e., $Q^* M_A Q = I$. Then we have

$$Q^* A Q = Q^* M_A Q R = R,$$

which reduces to solving the (smaller) standard eigenvalue problem.

A downside of this approach that when solving the correction equation one should always take care the update is in the divergence free space, in order to keep the search space in that space.

Chapter 3

Canonical flow problems

In this chapter, we present literature and recently obtained results, including some of ours, for several problems: 3D lid-driven cavity, differentially heated cavity, Rayleigh-Bénard convection as well as convection in a differentially heated rotating cavity. Apart from the first, all problems use the Boussinesq approximation. Our aim is to show how well matrix-based methods perform on such 3D problems. Therefore, we have performed numerical experiments on those problems with our home-made algorithms and techniques, which has been explained thoroughly in chapter 2. Some of our results will be presented in this chapter to show the agreement with those obtained by other researchers.

3.1 3D lid-driven cavity

Due to its simple geometry, the incompressible flow in lid-driven cavities plays an important role in fundamental fluid mechanics and as a numerical benchmark. We consider the flow of an incompressible Newtonian fluid in a 3D cavity with edges of length L (see Fig. 3.1). At the top, the fluid is driven by a lid moving with constant speed U. At all walls no-slip boundary conditions are applied, hence all boundary conditions are of Dirichlet type. The flow region is defined in the dimensionless Cartesian coordinates x, y and z, each of which varies from -0.5 to 0.5.

The governing equations for this problem are the incompressible Navier-Stokes equations, i.e. the first two equations of Equ. 1.4. If we define the kinematic viscosity by $\nu = \frac{\mu}{\rho}$, then the Navier-Stokes equations, with the velocity vector

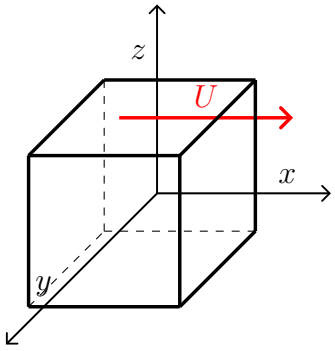


Figure 3.1: (3D lid-driven cavity) Geometry of cavity with moving lid (speed U in x-direction) on the top

$\mathbf{u} = (u, v, w)$, turn into

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \frac{-\nabla p}{\rho} + \nu \nabla^2 \mathbf{u}. \tag{3.1}$$

Using the fact that $\nabla \cdot \mathbf{u} = 0$ for incompressible flow, the non-conservative form can be changed into the conservative form, $\mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{u} \cdot \nabla \mathbf{u} + \mathbf{u}(\nabla \cdot \mathbf{u}) = (\nabla \cdot (\mathbf{u} \cdot \mathbf{u}^T))^T$. The dimensionless conservative form is

$$\frac{\partial \mathbf{u}}{\partial t} + (\nabla \cdot (\mathbf{u} \cdot \mathbf{u}^T))^T = -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u}, \tag{3.2}$$

where Re is the Reynolds number defined as $\mathrm{Re} = \frac{F_{inertia}}{F_{viscous}} = \frac{L\rho U}{\eta}$. Here, U and L are a characteristic velocity and a characteristic length, respectively.

An important step has been made by Feldman and Gelfgat [100] who computed the critical Reynolds number of $\mathrm{Re}_c = 1914$ for the onset of an oscillating perturbation on top of the steady flow. They found that the associated Hopf bifurcation is slightly sub-critical, i.e., for values lower than the critical value there exist two (stable) solutions. Here, a steady one and a transient one. When $\mathrm{Re} = 1970 > \mathrm{Re}_c$, it is observed that the oscillatory flow breaks the mirror symmetry with respect to the cavity midplane $y = 0$. The numerical prediction of Feldman and Gelfgat is consistent with experimental investigations of Liberzon et al. [101] in 2011, who found a critical onset in the range between Re = 1700 and

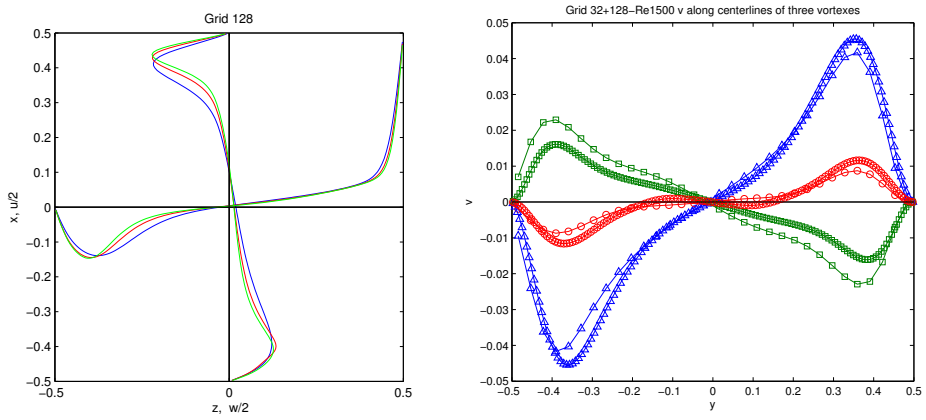


Figure 3.2: (3D lid-driven cavity) Steady state flow profiles obtained for grid 128^3 for Re $= 1000$ (blue), Re $= 1500$ (red) and Re $= 1900$ (green). $u/2$ and $w/2$ velocity components along centerlines (0,0,z) and (x,0,0), respectively (left); v component profiles for Re $= 1500$ along three lines going through the center of downstream secondary vortex(blue), primary vortex(green) and upstream secondary vortex(red), for grid 32^3 and 128^3 (right).

1970. And it was shown that at Re $= 1970$, the flow exhibits oscillations characterized by a dimensionless angular frequency $\omega = 0.575$. Kuhlmann and Albensoeder [51] did very accurate computations using a spectral method and found it is 1919.5. They computed this simply by time integration of the equations and varying the Reynolds numbers. They also studied the sub-critical behavior of this bifurcation in more detail and found that already at Re $= 1921$ complicated dynamics occurs by interfering non-symmetric modes, i.e. modes that do not adopt the mirror symmetry around the plane $y = 0$. After transition to unsteadiness, with further increase of the Reynolds number, the flow becomes turbulent.

For Re ≤ 1900, the computations showed that the steady flow is the only solution. These solutions have a reflection symmetry with respect to the cavity midplane $y = 0$, which for v can be observed in Fig. 3.2-left. Note that y-direction is orthogonal to the main circulation plane xz, see Fig. 3.1. It can be seen from Fig. 3.2 that the velocities do not change so much for Re ranging from 1000 to 1900, but the gradients of them become steeper and steeper close to the boundary with the increase of Reynolds number. Hence, fine grids are needed in the vicinity of the walls. Isosurfaces of oscillation amplitudes of the three velocity components are presented in Fig. 3.3 for developed unstable steady flow at Re $= 2000$.

As expected, the spatial pattern of the amplitude values has a reflection symmetry with respect to the cavity midplane $y = 0$, where amplitudes have the biggest value. The maximum amplitude is defined as $\max(\sqrt{u^2 + v^2 + w^2})$.

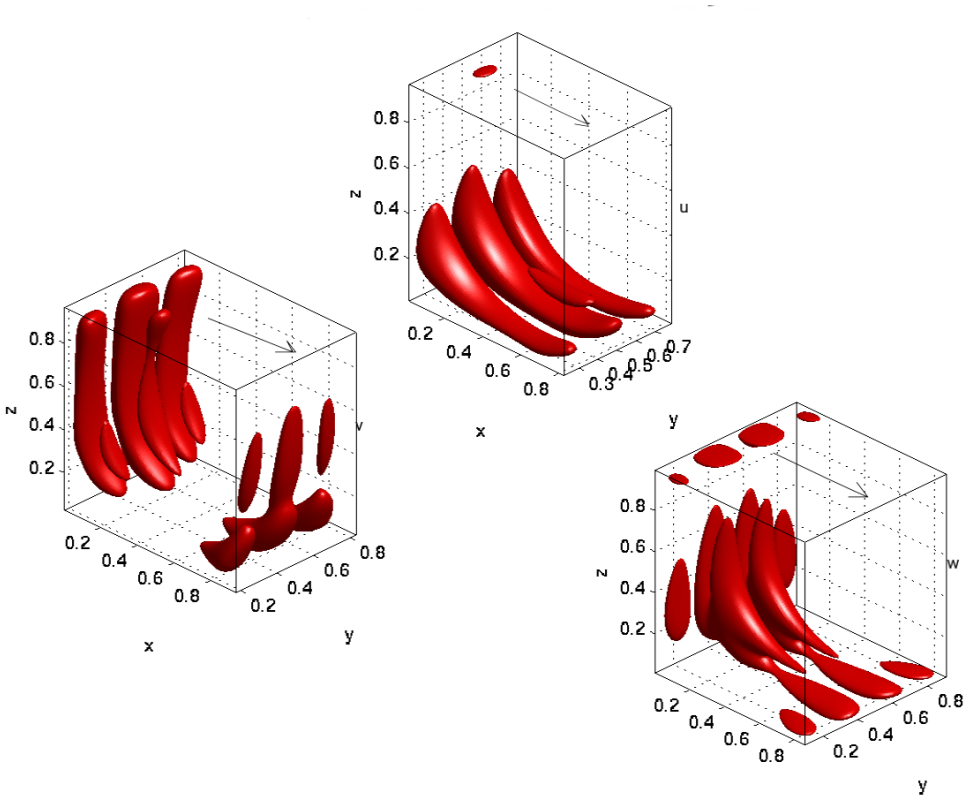


Figure 3.3: (3D lid-driven cavity) Isosurfaces, at 20% of the maximum velocity amplitude level, of the amplitude of the three velocity components (top u, left v and right w) of the most unstable eigenmode at Re $= 2000$, grid=128^3.

3.2 Boussinesq Equations

The Boussinesq approximation is appropriate for almost any compressible fluid with only small variations of the density, so that in the intertial terms, and in the continuity equation, we may approximate the density by a constant. However, even weak density variations are important for buoyancy. Therefore, variations in the buoyancy term are retained in the equation for the vertical component of the

momentum. With this approximation, in 1872, Boussinesq derived the equations known nowadays as the Boussinesq equations, which have been used to model many geophysical phenomena. For instance, it can model large scale atmospheric and oceanic flows that are responsible for cold fronts and the jet stream, see [102, 103]. In addition, the Boussinesq equations also play an important role in the study of the differentially heated cavity problem and Rayleigh-Bénard convection [104, 105].

The standard 3D Boussinesq equations are given by

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} =& \frac{-\nabla p}{\rho} + \nu \nabla^2 \mathbf{u} + g\alpha T \mathbf{e}_z, \\ \frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{T} =& \kappa \nabla^2 T, \end{aligned} \tag{3.3}$$

where the divergence-free constraint equation 1.2 is added to make the equations complete. Here ν is the kinematic viscous coefficient, κ is the thermal diffusivity coefficient and α is the heat expansion coefficient.

3.2.1 Differentially heated cavity

The above Boussinesq equations can be used in differentially heated cavity problems, whose typical configuration is a rectangular domain with two opposite isothermal walls kept at temperatures T_H and T_C ($T_H > T_C$), respectively. The rest of the walls are assumed to be adiabatic. Here, we consider the case that the fluid is confined to a box of size $L_x \times L_y \times L_z$, where L_x, L_y and L_z are the length in x-, y- and z-direction, respectively. Suppose d is the distance between the differentially heated walls, which is either L_x, L_y or L_z.

Choose d as the length scale, d^2/ν as the time scale (and thus $\frac{\nu}{d}$ for the velocity scale), and $\frac{\nu \Delta T}{\kappa}$ ($\Delta T = T_H - T_C$, the difference between the high and low temperature) as a scale for the temperature, then the momentum conservation equations can be put into dimensionless form in Cartesian coordinates with the components of the velocity vector given by $\mathbf{u} = (u, v, w)$ as follows:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} =& -\nabla p + \nabla^2 \mathbf{u} + \mathrm{Ra} T \mathbf{e}_z, \\ \frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T =& \frac{1}{\mathrm{Pr}} \nabla^2 T, \end{aligned} \tag{3.4}$$

where $\mathrm{Ra} = \frac{\alpha g \Delta T d^3}{\nu \kappa}$ is the Rayleigh number and $\mathrm{Pr} = \frac{\nu}{\kappa}$ is the Prandtl number.

There are two important cases of the differentially heated cavity problem: (i) the case that the imposed temperature gradient is perpendicular to the gravity force and (ii) the case that the temperature gradient is parallel to the gravity force. The latter is often called the Rayleig-Bénard problem, and the former is commonly understood as the differentially heated cavity problem, though it in fact is a *laterally* differentially heated cavity problem. In the next section, we will discuss the Rayleig-Bénard problem in more detail.

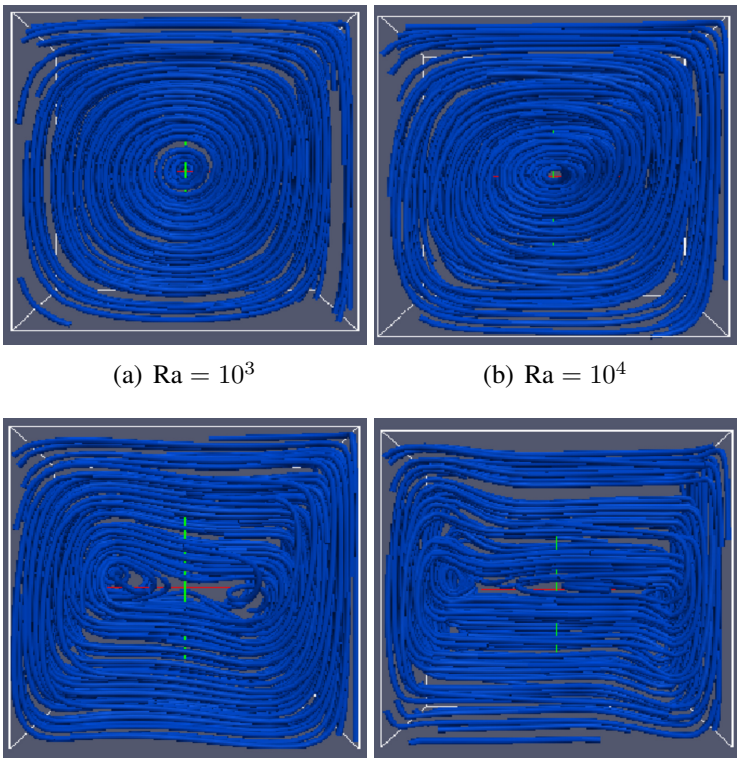
In a differentially heated cavity, just the buoyancy force, which depends on the temperature difference between vertical walls, does supply enough force to get the fluid into motion. The speed is kept at bay by the no-slip walls. Here, we will consider the 2D and 3D flow in a square/cubic cavity, heated from the west side $x = -\frac{1}{2}$ and cooled at the east side, with boundary conditions as follows

$$\begin{aligned} \mathbf{u} = 0 \text{ at } |x| = \frac{1}{2}, T = \frac{1}{2} \text{ at } x = -\frac{1}{2}, T = -\frac{1}{2} \text{ at } x = \frac{1}{2}, \\ \mathbf{u} = \frac{\partial T}{\partial z} = 0 \text{ at } |z| = \frac{1}{2}, \mathbf{u} = \frac{\partial T}{\partial y} = 0 \text{ at } |y| = \frac{1}{2}. \end{aligned} \tag{3.5}$$

Due to the heating (cooling) at the west (east) side, the density of the fluid near the west (east) wall decreases (increases) compared to an initial solution $T = 0$, resulting in a clockwise rotation of the fluid inside the cavity. For a large range of Rayleigh numbers, the 2D velocity field in a square cavity is indistinguishable from that in the symmetry plane ($y = 0$) of the 3D case, due to the negligible effect of the no-slip conditions on the walls at $|y| = \frac{1}{2}$. Fig. 3.4 shows streamlines of the symmetry plane $y = 0$ in a cubical cavity. For low values of the Rayleigh number, the convective effect is very small and the solution is stable; It is found by our methodology that the first critical Rayleigh number occurs around 1.712×10^4, at which the flow gets unstable. When the Rayleigh number increases to 10^5 and 10^6, the buoyancy force increases. As a result, strong circulation of fluid inside the cavity and convective heat transport occur, as shown in the figure. Note that the shown solutions at the bottom are unstable, but steady.

Note that the equations (3.4) have a point symmetry around the origin. At the origin the flow is zero and all variables are odd functions on any line trough the origin. This symmetry is reflected in the solutions in Fig. 3.4. We mention it here because we will later see the case where the cavity is rotated, which causes that this property will be lost.

Note that for low Rayleigh numbers, the velocities will be small and hence viscosity will dominate over inertia. This leads to a symmetry with respect to the



(a) Ra $= 10^3$ (b) Ra $= 10^4$

(c) Ra $= 10^5$ (d) Ra $= 10^6$

Figure 3.4: (Differentially heated cavity) Streamlines for increasing values of the Rayleigh number.

plane $z = 0$, which is clearly present in plot (a) of Fig. 3.4. It is lost when inertia gets more important.

3.2.2 Rayleigh-Bénard convection

The Rayleigh-Bénard convection problem has been widely investigated because of its fundamental interest in relation to the evolution of flow patterns and the onset of unsteadiness. Many engineering problems related to thermal transport in crystal growth, solar collectors, buildings and nuclear reactor core insulation depend on this type of natural convection. The first quantitative experiment was performed by Henri Bénard in the year around 1900 [106]. In his experiment, he studied the stability of a thin fluid layer, which is open to the air and undergoes

a vertical temperature gradient. Fig. 3.5 shows one of Bénard's famous original photographs. For a more detailed review of this problem, readers are kindly in-

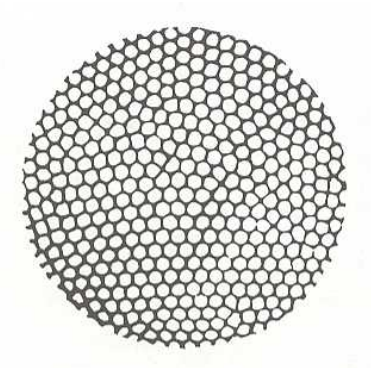


Figure 3.5: (Rayleigh-Bénard convection) One of Bénard's original photographs, low resolution, seen from the top [107].

vited to consult the first chapter of Chandrasekhar's book [108] and more recent reviews in [109, 110].

This prototypical problem not only serves as a benchmark in evaluating computational methods in the laminar regime, but also offers the opportunity to fully understand the transition mechanism and obtain substantial insight into natural convection. The first transition from conduction, i.e. the no-flow case, to convection starts when the Rayleigh number is increased beyond a critical value Ra_c. It is well documented in the literature both for 2D rectangular cavities and 3D enclosures with a variety of width/height aspect ratios [111–113]. The critical Rayleigh number differs with different boundary conditions and aspect ratio's, while independent of the Prandtl number (see later in this section). In Table 3.1 some critical values are shown for infinite domains; these can be derived analytically, using a Fourier component as Ansatz. The resulting critical component will have a certain wavelength or period which also is given in the table. This means that the same critical value will occur if we add free-slip vertical walls one or multiple wavelength apart. The counter side of this is that if we add free-slip walls not equal to a multiple of the wavelength, then the critical Rayleigh number will go up. It is interesting to observe in the first result that a potential unstable situation, i.e. a light fluid below a heavy fluid, is kept in place by viscous forces, not allowing the fluid to get into motion. The second interesting point is that Ra_c is increasing with the length of the no-slip wall, in Table 3.1 from bottom to top

boundary condition at top and bottom wall	Ra_c	wave length	notes
free-slip isothermal	657	2.8285	Viscous force wins from buoyancy force for low values of Ra.
free slip at the top, no-slip at the bottom	1101	2.3427	Less no-slip walls decreases Ra_c.
no-slip isothermal	1707	2.0158	No-slip walls delay the onset of convection.

Table 3.1: (Rayleigh-Bénard convection) Ra_c and corresponding wave length for various combinations of top and bottom boundary conditions for the 2D case on an infinite domain [114, 115].

the length is 0 (all free-slip boundaries), 2.3427 (no-slip at the bottom, hence one wave length), 4.0316 (no-slip at bottom and top, hence 2 wave lengths). At the same time the wavelength is decreasing. Puigjaner et al. have been doing a series of studies on determining the bifurcation diagram and stability of the flow at moderate Rayleigh numbers(Ra $\leq 10^5$) both for insulated and conductive side walls, by means of the Galerkin method with divergence-free basis functions to obtain the solutions and using a parameter continuation algorithm to follow the branch [54, 55]. With insulating side walls, they reported the occurrence of a flow pattern with a single roll around Ra $= 3389$, four rolls around Ra $= 5900$ and two x-rolls plus two y-rolls(toroidal type) around Ra $= 7400$. In [116], they also reported a complex bifurcation diagram of periodic flow patterns within the range $2 \times 10^4 \leq \mathrm{Ra} \leq 1.5 \times 10^5$.

For the solution one can use again (3.4), but in this case the conductive state is a solution for all Rayleigh numbers. This makes it easy to derive the equations for the perturbation of the conductive state. Numerically this is advantageous, since the small perturbations does not suffer from the round-off in the total state. Note that the laterally differentially heated cavity of the previous section does not have a conductive state solution for any Rayleigh number.

Let d be the vertical height of the container. Express the temperature of the dynamical system as a sum of the linear profile of the conduction state and the temperature fluctuation $\theta(x,t)$, i.e., $T(x,t) = T_H - \frac{\Delta T}{d} z + \theta(x,t)$. Then the

Boussinesq equations turn into

$$\begin{aligned}\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} =& \frac{-\nabla p}{\rho} + \nu\nabla^2\mathbf{u} + g\alpha\theta\mathbf{e}_z,\\ \frac{\partial \theta}{\partial t} + \mathbf{u}\cdot\nabla\theta =& \kappa\nabla^2\theta + \frac{\Delta T}{d}\mathbf{u}\cdot\mathbf{e}_z.\end{aligned} \tag{3.6}$$

Rename θ as T again, then the corresponding dimensionless momentum equations governing the Rayleigh-Bénard convection are the same as Equ. 3.4, except that there is one more term in the temperature equation:

$$\frac{\partial T}{\partial t} = -((uT)_x + (vT)_y + (wT)_z) + \frac{1}{\mathrm{Pr}}\nabla^2 T + \frac{1}{\mathrm{Pr}}w. \tag{3.7}$$

Since we solve for the perturbation, the boundary conditions are a homogeneous variant of those for differentially heated cavity, except that the Dirichlet boundary conditions for the temperature occur at bottom and top of the cavity:

$$\mathbf{u} = T = 0 \text{ at } |z| = \frac{1}{2}, \mathbf{u} = \frac{\partial T}{\partial x} = 0 \text{ at } |x| = \frac{1}{2}, \ \mathbf{u} = \frac{\partial T}{\partial y} = 0 \text{ at } |y| = \frac{1}{2}. \tag{3.8}$$

Note that the critical Rayleigh number at which the trivial solution becomes unstable is independent of the Prandtl number. This can be proved by considering the eigenvalue problem associated to the stability study, i.e., $\det(\lambda\mathcal{M} - \mathcal{J}(\mathrm{Ra}, \mathrm{Pr})) = 0$, where $\mathcal{M}$ is the opearator in front of the time derivatives and $\mathcal{J}$ is the continuous variant of the Jacobian of the right-hand side. At the critical Rayleigh number the eigenvalue $\lambda = 0$, which leads to $\det(\mathcal{J}(\mathrm{Ra}, \mathrm{Pr}) = 0$. Considering (3.7), we see that the convective terms cancel out since we consider stability of the trivial solution and then the Prandtl number can simply be divided out, so the critical values follow from $\det(\hat{\mathcal{J}}(\mathrm{Ra})) = 0$, where the hat denotes the modification.

In this thesis, we discuss the primary bifurcations, i.e., branches switching off from the motionless conductive state to the convective state. The algebraic multiplicity of the first eigenvalue, which is also the first critical Rayleigh number, is two. So is the geometric multiplicity, which means it has two linear independent eigenvectors. In fact, one eigenmode corresponds to an x-roll pattern and the other one to a y-roll pattern. These are the straightforward generalization of the first critical eigenmode of the 2D case. We will see in Table 5.3 that for 2D $\mathrm{Ra}_c \approx 2584$ which is higher than the smallest value found in free space (see Table 3.1) due to the no-slip vertical walls. The 2D case is the same as a 3D flow in the case of a free-slip front and back wall. Since these are in fact no-slip walls the critical

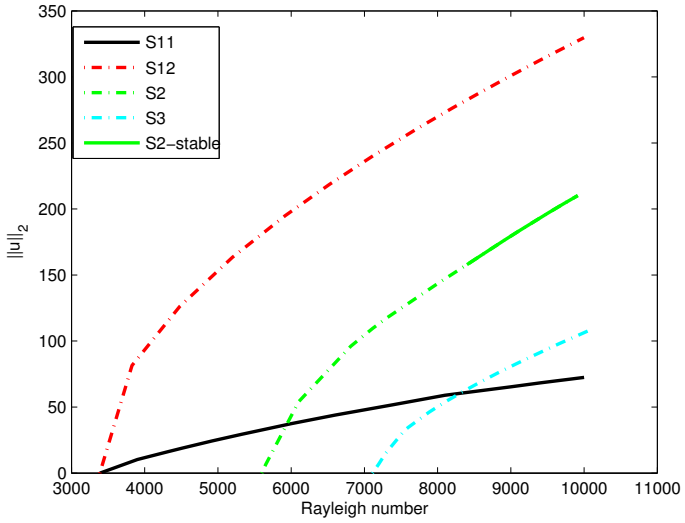
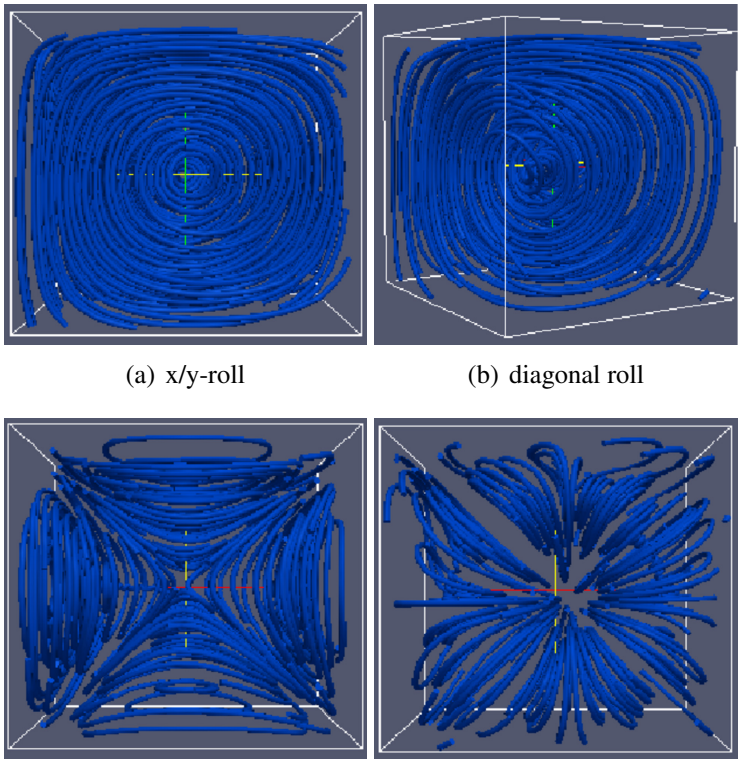


Figure 3.6: (Rayleigh-Bénard convection) Bifurcation diagram of flow in a cube at Pr $= 1.0$. L^2 norm of velocity in x-direction as a function of Rayleigh number. S11, S12, S2, S3 represent x/y roll, diagonal roll, four rolls and toroidal shape solutions, respectively. Stable and unstable flow patterns are depicted with solid lines and dashed lines, respectively.



(a) x/y-roll (b) diagonal roll

(c) four rolls (d) toroidal roll

Figure 3.7: (Rayleigh-Bénard convection) Flow patterns near the first three primary bifurcations.

Rayleigh number increases to approximately 3387 (see Table 5.3), which is also shown in Fig. 3.6). Using the x-roll eigenvector to get on the non-trivial branch, we find an x-roll (depicted as S11 in Fig. 3.6) flow pattern by continuation. So does the y-roll flow pattern. Using the linear combination of these two eigenvectors in the perturbation, we get the single diagonal roll solution (S12). The x-roll and y-roll solutions are stable within the studied Rayleigh number range from 0 till 10^5, while the diagonal roll solution is unstable. From the second critical Rayleigh number, the solution takes the shape of four rolls (S2), which is unstable in the rang of $5900 \leqq \text{Ra} \leqq 8300$ and becomes stable when $\text{Ra} > 8300$. At the third primary bifurcation, an unstable flow pattern develops and has a toroidal shape (S3). A visualization of these flow patterns in streamlines is given in Fig. 3.7.

The flow patterns described above, which are very similar to the eigenvectors associated to the zero eigenvalues of the linearized problem, only hold for Rayleigh numbers slightly above the critical bifurcation point. They just possess the symmetry properties of the linearized problem around the conductive state. It is of interest to analyze how the flow patterns evolve as the Rayleigh number increases. Fig. 3.8 shows contours of the velocity u perpendicular to the plane $x = 0.25$ as well as the velocity vector field in the plane at $\text{Ra} = 3730$ and $\text{Ra} = 20000$. When Ra increases to 20000, the solution still remains basically x-roll configuration, but elongates in the $y = -z$ direction. There is a $\frac{\pi}{2}$ rotation about the x axis. It is noticed that the secondary circulations near the edges of the cavity becomes stronger as the Rayleigh number increases, as what is presented in [54].

3.2.3 Convection in a differentially heated rotating cavity

Rotation can have profound effects on the convection in a rotated enclosure. For instance, in the course of crystal growth, at some point, rotation is needed to stabilize the buoyancy induced flow to get a crystal of higher quality. In the literature, considerable attention has been paid to the rotating Rayleigh-Bénard convection, the convection in an infinite bounded horizontal layer of fluid rotating at a constant angular speed about a vertical axis. Another geometry of great interest is the flow in a vertical closed circular cylinder, which is heated from below and rotates about its axis. Good reviews are written by Ker [117] and by Lee [118], in which they also investigated how the Coriolis force and the centrifugal force act affect a flow in a cavity dominated by thermal buoyancy.

We studied convection of air in a differentially heated rotating cubic cavity

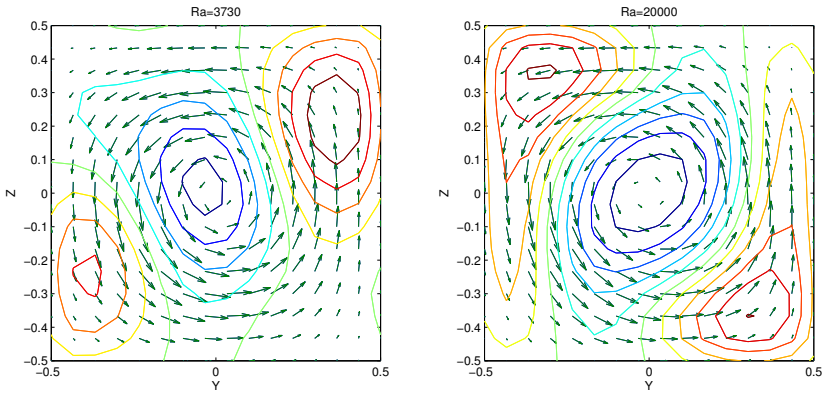


Figure 3.8: (Rayleigh-Bénard convection) Contours of the velocity component normal to the vertical plane $x = 0.25$, and velocity vector distributions for the x-roll flow pattern at $\mathrm{Ra} = 3730$ (left) and $\mathrm{Ra} = 20000$ (right), respectively.

in which the rotating axis coincides with the z-axis and is parallel to the gravitational force. The configuration is depicted in Fig. 3.9. Boundary conditions are as for the case without rotation, i.e., Equ. 3.5. Relative to the Equ. 3.4, there are two extra non-dimensional parameters: the rotational Rayleigh number $\mathrm{Ra}_\omega = \Omega^2\alpha\Delta T d^4/(\nu\kappa)$ (κ is the thermal diffusivity coefficient) and the Taylor number $\mathrm{Ta} = \Omega^2 d^4/\nu^2$. Herewith, the governing momentum equations and the heat equation are as follows:

$$\begin{aligned}
\frac{\partial u}{\partial t} &= -\left((uu)_x + (vu)_y + (wu)_z\right) - p_x + \nabla^2 u + 2\sqrt{\mathrm{Ta}}\, v - \mathrm{Ra}_\omega x T, \\
\frac{\partial v}{\partial t} &= -\left((uv)_x + (vv)_y + (wv)_z\right) - p_y + \nabla^2 v - 2\sqrt{\mathrm{Ta}}\, u - \mathrm{Ra}_\omega y T, \\
\frac{\partial w}{\partial t} &= -\left((uw)_x + (vw)_y + (ww)_z\right) - p_z + \nabla^2 w + \mathrm{Ra} T, \\
\frac{\partial T}{\partial t} &= -\left((uT)_x + (vT)_y + (wT)_z\right) + \frac{1}{\mathrm{Pr}}\nabla^2 T,
\end{aligned} \tag{3.9}$$

where x and y represent spatial coordinate values. Note that apart from the Prandtl number, which has a fixed value for air, there are three dimensionless parameters: the Rayleigh number, the Taylor number and the rotational Rayleigh number.

With $\mathrm{Pr} = 0.7$ for air, $\mathrm{Ra} = 100$ and $\mathrm{Ta} = 100$ fixed, we do continuation on the rotational Rayleigh number, which means that we effectively change the aspect ratio of the cavity. As we increase Ra_ω to 10^6, though the flow is still dominated

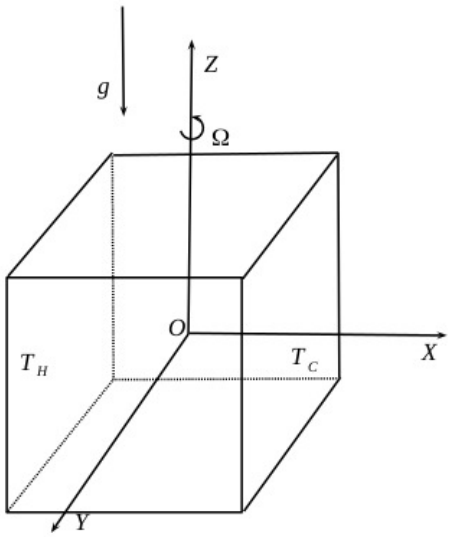


Figure 3.9: (Differentially heated rotating cavity) Geometry of rotating cubic cavity.

by thermal buoyancy, the effects of rotational buoyancy is getting stronger and stronger, which breaks the centro-symmetry of the flow present in the case without rotation (cfg. Fig. 3.4). In the top of the velocity fields, shown in Fig. 3.10, one sees that the magnitude of the dimensionless velocity goes up to about 138. Viewing the flow from the top, a pair of elongated rolls can be observed clearly, whose axes are parallel with the rotating axis of the cavity. These result from the fact that the heavier cold fluid is expelled to the boundaries of the cavity due to the centrifugal force causing that the lighter warm fluid is pressed to the center line. So at the cold wall the fluid moves away from the middle whereas at the hot wall the fluid moves towards the middle and creates the mushroom shape in the isotherms in bottom panel of Fig. 3.10. The plots for $z > 0$ did show a symmetry with respect to the plane $z = 0$ hence we omitted them. This symmetry is analytically there only if the $\mathrm{Ra} = 0$, but since $\mathrm{Ra}_\omega = 10^6$ the centrifugal effect dominates the picture here.

Instead, if we reverse the magnitudes of Ra_ω and Ta, i.e., $\mathrm{Ra}_\omega = 100$ and $\mathrm{Ta} = 10^6$, then the Coriolis force dictates the flow. It is rather weak with $\mathbf{v} < 0.07$.

These results are similar to those in the paper [118]. The only difference is that we use a uniform grid, while they refined the grid near the boundaries.

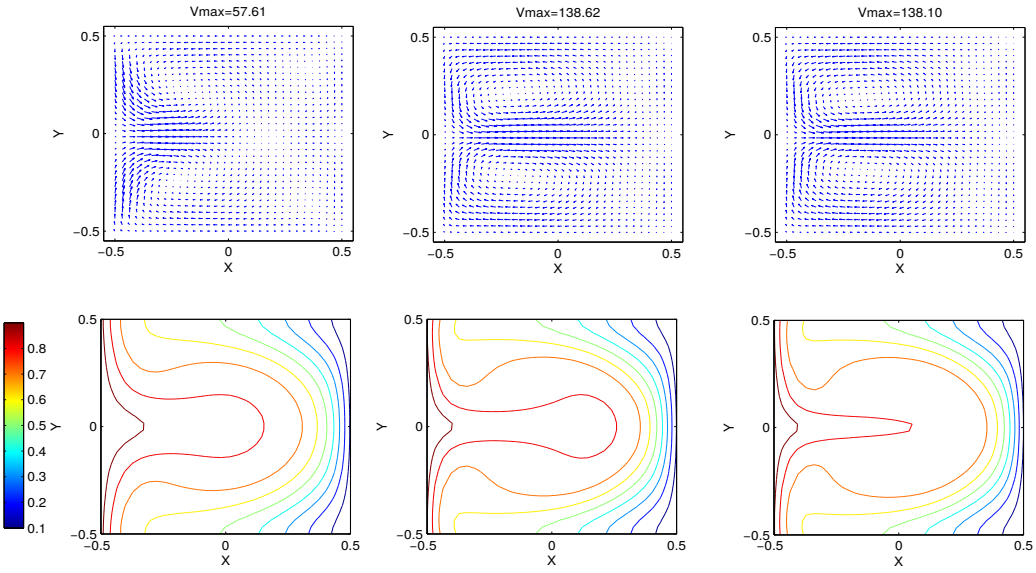


Figure 3.10: (Differentially heated rotating cavity) Velocity vector field and isotherms for planes (from left to right) $z = -0.469$, $z = -0.281$, $z = 0.0$.

Chapter 4

Numerical results for the lid-driven cavity problem

In this chapter, three dimensional numerical experiment results, including the behavior of simulated solutions on different Reynolds number as well as the eigenvalues for the benchmark problem lid-driven cavity are presented. The parallel performance of our algorithm for various grid sizes using HYMLS and Teko, the latter with the Least Squares Commutator (LSC) preconditioner, is also investigated. Moreover, to compare the performance of a time dependent method and continuation, we present results from a time integration method using the backward Euler method as integrator.

4.1 Numerical solutions

In our continuation program, the Reynolds number is the continuation parameter. Starting from a small number, e.g., Re $= 1$, the Reynolds number is increased with step size 500 till Re $= 1900$. Meanwhile, the eigenvalues are computed in order to check the stability of the solution. The results are validated by comparing them with those of Feldman and Gelfgat [100], in which a time-dependent method is used for both 2D and 3D computations. Good agreement of the center line velocity and pressure values is observed for the relevant range of Re, which shows the correctness of the present calculations.

Velocities and pressures on different grids are given in Fig. 4.1. The difference between the results of two close grids is getting smaller and smaller as the grid size is refined. In fact, we see the differences decrease by a factor 4 when

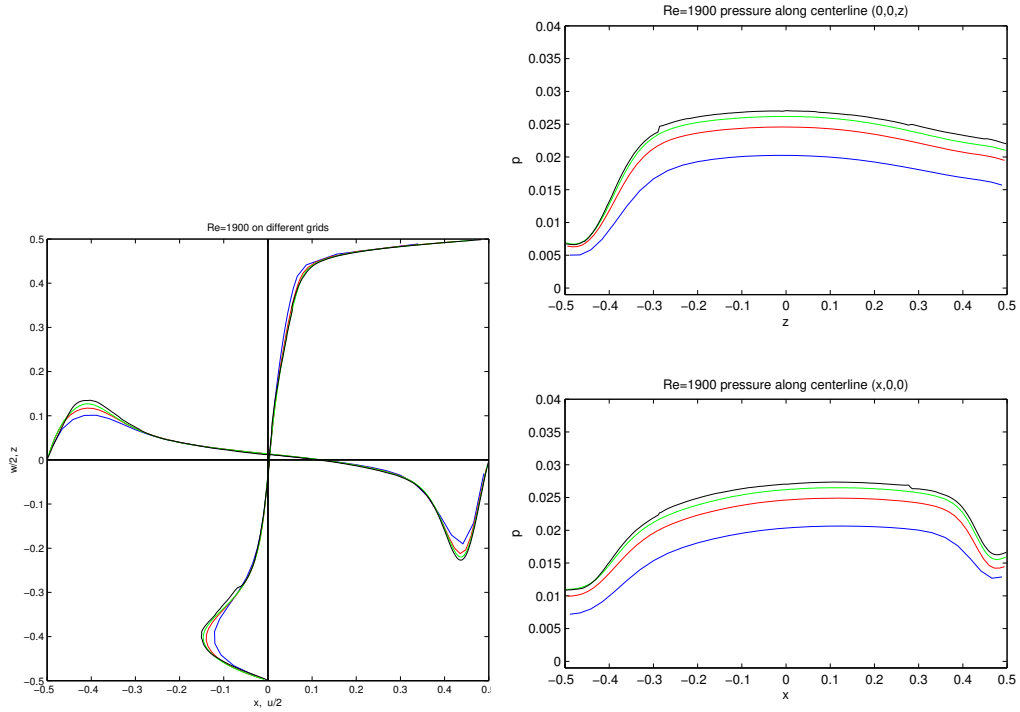


Figure 4.1: (3D lid-driven cavity) Velocity $u/2$ and $w/2$ components (left) and pressure p (right) along centerlines (0,0,z) and (x,0,0) on Re $= 1900$ of grid 32^3 blue, 64^3 red, 128^3 green and 256^3 black.

refining the mesh by a factor 2, which shows the second-order accuracy of the discretization. There is one primary clockwise eddy in the middle and two secondary anti-clockwise eddies in the bottom of the cavity left and right, respectively. These are the same as found in 2D [119], except that there is one more secondary eddy at the top left of the cavity in 2D [120].

In our study, we want to use continuation of equilibria to find the critical point where the solution becomes unstable. To find the critical point, one has to compute the critical modes by solving an algebraic eigenvalue problem. Doing so, one observes that a conjugate pair of eigenvalues is crossing the imaginary axis near Re $= 2000$, see Fig. 4.2. Hence, the flow loses its stability there. Table 4.1 shows the eigenvalues with smallest real part on a sequence of grids. By extrapolation, based on the second-order behavior of the error, the critical Reynolds number is

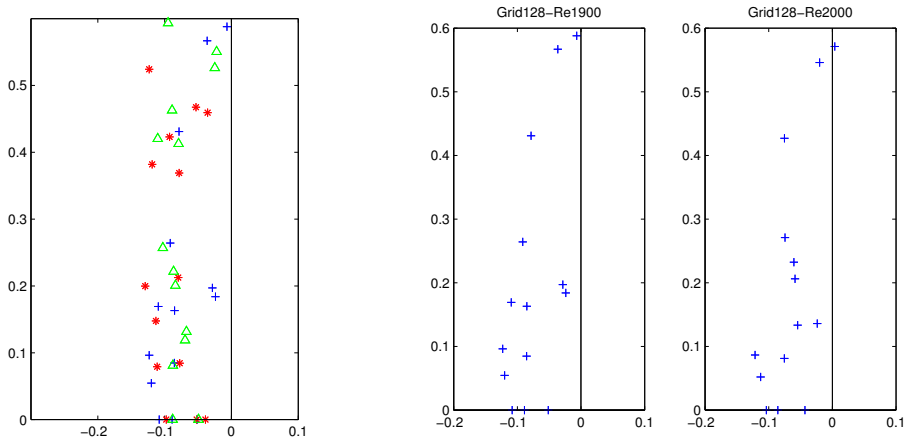


Figure 4.2: (3D lid-driven cavity) Eigenvalues at Re=1900 of grid 32^3 red, 64^3 green and 128^3 blue (left); eigenvalues closest to zero of grid 128^3 at Re $= 1900$ and Re $= 2000$ (right). *Results courtesy of Thies.*

Re	λ_{32}	λ_{64}	λ_{128}
1900	$-53.05 + 0.468i$	$-22.28 + 0.550i$	$-6.762 + 0.588i$
2000	$-42.35 + 0.463i$	$-10.73 + 0.545i$	$3.901 + 0.571i$
Re_c	2395	2093	1963 extrapolated value 1917

Table 4.1: (3D lid-driven cavity) Eigenvalues obtained on different grids at Re $=$ 1900 and Re $= 2000$.

expected around 1917. At that point, the eigenvalue is between $0.571i$ and $0.588i$, which is in great agreement with the angular frequency obtained by Liberzon et al.. Eigenvalues have been computed by the Anasazi block Krylov-Schur method using Shift-and-Invert as well as the Cayley transform to target the eigenvalues near 0. One eigenvalue's real part becomes positive between Re $= 1900$ and Re $= 2000$, which means there is a Hopf bifurcation occurring in this interval. In Fig. 4.2 the two eigenvalues closest to the imaginary axis are very close to each other.

With the current continuation technique, the stable periodic orbit cannot be followed after the Hopf bifurcation point. In order to do that, a time-dependent computation is needed. This should be built in for instance PDECONT, so that

Re	Newton				Picard			
	nx=8		nx=16		nx=8		nx=16	
	Nwt its.	HGMR its.	Nwt its.	HGMR its.	Picrd its.	HGMR its.	Picrd its.	HGMR its.
0	1	72	1	112	1	72	1	112
200	4	92	4	160	58	86	74	147
400	4	104	3	202	1e-3	97	1e-3	172
600	4	122	3	226				
800	3	135	3	252				
1000	3	148	3	277				

Table 4.2: (3D lid-driven cavity) Comparison of Newton (Nwt) and Picard (Picrd) iteration; stopping criterion nonlinear iteration and HYMLS-preconditioned GMRES (HGMR): 1.0e-8.

Floquet multipliers can be computed. The name of bifurcation occurred here should be cyclic Hopf bifurcation. Hopf bifurcations are described in more detail in [121] and in [122] by Golubitsky and Schaefer.

4.2 Newton versus Picard iteration

In this section, we show in Table 4.2 a comparison of performance of Newton versus Picard iteration. In our code, Picard iteration is simply obtained by giving the Newton process just the matrix needed to compute the right-hand side (see the MDP property in section 2.1.1). We just increased the Reynolds number by step size 200. One observes that the number of Newton iterations (Newton its.) is low in this range, indicating that the behavior is not far from linear. The number of HYMLS iterations is slowly increasing with the Reynolds number. Next to it we see that the number of Picard iterations is already high for Re=200. At Re=400 the convergence is stagnated around 1e-3. This occurs also at Re=300. We know that the matrix in the Picard iteration has negative eigenvalues for all Reynolds numbers, (see [77]); the real part of the eigenvalues is determined by the eigenvalues of the diffusion part. This makes the linear solve somewhat easier as we can see from the number of iterations of HYMLS. It is disappointing that Picard iteration already fails at these low Reynolds numbers, since we know that

the first bifurcation is occurring much later at Re ~ 2000. At that value the complete Jacobian gets its first positive eigenvalue.

4.3 Parallel performance

To test the scalability of HYMLS, we did experiments in 3D for different numbers of grid points $nx = ny = nz$ with skew partitioning. The grid is exponentially stretched, with cells near the walls 4 times smaller than those at the centerlines. Tables 4.3 and 4.4 show results of Re $= 0$ and Re $= 500$, respectively, including the running time of the main phases with different value of coarse factor cf (explained in chapter 2 section 2.3.3), number of level L, number of separator length sx and different number of processors np. t_c represents the time it takes to compute the preconditioner and t_s means the time the linear solver takes, stopping criterion is 1e-8. it_{tot} represents the number of GMRES(50) iterations performed by HYMLS in the last iteration of the Arnoldi process. To have a better idea of how the factors cf, np, L and sx interact, the tables also show the size of the Schur-complement on the last level. The simulations are performed on the cluster of University of Groningen called Peregrine, which has 162 nodes and each node has 24 Intel Xeon 2.5 GHz cores and 128 GB internal memory. The nodes are coupled through an 56 Gb/s Infiniband network.

Observe that the last Schur-complements for $nx = 32$ and $nx = 128$ are of similar size (row of color light cyan). The number of iterations doubles. So this means that the number of iterations behaves like $\sqrt{nx}$.

Note that the Stokes problem is merely a Poisson problem restricted to the divergence-free space. For a Poisson problem, we know that without scaling or with diagonal scaling, or with ILU(0) preconditioning, we just get the nx behavior. A symmetric Gauss-Seidel or MILU(0) preconditioning should bring it to $\sqrt{nx}$. The present method shows this behavior if we keep the last Schur-complement of equal size when refining(see rows of color light cyan). On the other hand, if we keep the number of levels fixed as 3 we see that the number of iterations goes to an upper bound 403(red in Table 4.3). This is according to the theory in [47]. Note that the number of processors does not influence the number of iterations as it should, because the algorithm does not depend on that.

Starting with the zero-solution until a steady solution is reached at Re $= 500$, it takes 5 or 6 Newton steps to converge, while for Re $= 0$ one only needs 2 steps due to linearity. That explains why more time and iterations are spent on the linear solver for Re $= 500$. Observe that the number of iterations goes up by a factor

nx	cf	np	L	sx	lss	t_c	t_s (it_{tot})
16	4	8	2	4	6360	3.2	0.7(75)
32	4	1	3	4	9520	15	17.5(150)
32	4	8	3	4	9520	3.6	2.7(150)
32	4	16	3	4	9520	3.0	2.7(150)
64	4	8	3	8	9648	13.4	32.5(213)
64	2	8	3	8	17176	15	53(270)
64	4	16	3	4	76216	18.3	23.8(211)
64	4	16	4	4	1221	18.1	24.9(216)
64	4	32	4	4	1221	20	42.7(216)
128	4	64	3	8	76728	18.7	81.2(289)
128	4	32	4	4	9648	90	187(305)
256	4	64	3	8	612048	855	234.5(403)

Table 4.3: (3D lid-driven cavity) Scalability test results of HYMLS at Re $= 0$.

3 when going from $nx = 32$ to $nx = 128$. If we bound that factor by 4, though based on the Stokes case we expect a factor 2, we get a complexity of order nx^4 or, expressed in $N = nx^3$, $N^{4/3}$.

The performance compared with Teko at Re $= 500$ is showed in Table 4.5. We chose Teko for comparison because it is available in Trilinos and can therefore easily be coupled to our code. The stopping criterion of the linear solver is the same as before 1e-8. And the experiments are performed on the same cluster. Compared to HYMLS, at grid 32^3, it has better performance, taking less time to solve the linear system and less iterations. However, as grid refinement at grid 64^3, it takes around one hundred times more time to compute the preconditioner and solve. Meanwhile, more iterations are needed for Teko to converge. For higher resolutions, Teko does not converge.

4.4 Behavior of a time integration method

To study the behavior of time integration, we ran the program Heat97 using the backward Euler method as integrator. The pressure Poisson equation and the momentum (transport) equations are solved separately, using MICCG and BiCGStab,

nx	cf	np	L	sx	lss	t_c	t_s (it_{tot})
16	4	8	2	4	6360	3.2	0.7(75)
32	4	1	3	4	9520	15	332(282)
32	4	16	3	4	9520	3.0	42.7(282)
64	4	8	3	8	9648	13	561(460)
64	4	32	3	4	76216	23	760(420)
128	4	64	3	8	76728	18.5	1098(659)
128	4	64	4	4	9648	41	1383(743)
256	4	256	3	8	612048	833.6	3724(900)

Table 4.4: (3D lid-driven cavity) Scalability test results of HYMLS at Re $= 500$.

nx	np	L	t_c	t_s (it_{tot})
32	1	3	11.8	189.3(198)
64	8	3	1540	5498(551)
128	64	3	4544	-

Table 4.5: (3D lid-driven cavity) Performance of Teko with LSC preconditioner at Re $= 500$.

nx	Δt (sec)	Poisson iters./time step	CPU time (sec)
64	0.05	29	660
128	0.025	35	9420
256	0.01	43	167340

Table 4.6: (3D lid-driven cavity) Performance of time integration at Re $= 500$ on different grid size. *Results courtesy of Veldman.*

respectively. So for the pressure Poisson equation we use a modified Incomplete Cholesky factorization as preconditioner in the Conjugate Gradients method. For the momentum equation we do not employ a preconditioner as for limited time step size used here, the CFL number is between 4 and 5, the matrix is rather close to identity. Unfortunately, the Heat97 code is just sequential and hence it does not run in parallel.

Table 4.6 shows the performance of this method on a 3D lid-driven cavity problem on one node of Peregrine (see beginning of previous section) for different grid sizes, starting with the zero-solution until a steady solution is reached. For Re $= 500$, 50 physical seconds are required to obtain the steady solution in about 8 digits. The grid is exponentially stretched, with cells near the walls 4 times smaller than at the centerlines as we did in the experiments with FVM/HYMLS. The experiments are performed on University of Groningen's cluster Peregrine, too.

The convergence of MICCG method behaves as expected. We see that the number of iterations is proportional to $\sqrt{nx}$, even slower than that. Probably this is due to the averaging over a large number of time steps in which the solution approaches a steady limit, hence the initial guesses are getting better and better. The solution time of the momentum systems is negligible because on average between 1 and 2 iterations per time step suffice. Note that the memory consumption is increasing almost linear with the amount of unknowns as expected. The CPU time increases by roughly $nx^{9/2}$ or $N^{3/2}$. The extra $3/2$ on top of N is due to the square root increase of the number of iterations of the Poisson solver and the linear increase of number of time steps to reach the end time (note that the time step Δt is about halved when nx is doubled). The decrease of the time step helps to keep the number of iterations for the momentum equation low.

4.5 Comparison of results

Comparing the time integration to the continuation, we observe that the continuation complexity is about $N^{4/3}$ at Re $= 500$ while that of the time stepper is about $N^{3/2}$. Since, the continuation has quite some overhead. Let us assume for a moment that the time integration method is perfectly parallel scalable. Then for $nx = 256$ we need about 650 seconds to get the result, where HYMLS needs about 4500 seconds. Then this shows that solving with HYMLS is slower by a factor 6. However, in the time integration approach the preconditioner is an ILU(0) factorization and the solution of the linear equations dominates the turnaround time. Since, ILU(0) contains recurrence and it is not straightforward to parallelize. To overcome these reorderings have to be done which may easily give rise to overhead. So in reality the factor 6 will be less. Of course, both programs allow for efficiency improvements. We believe that with a substantial effort one could speedup the continuation more than what is possible in the time integration approach. We know for instance that quite some data is moved in the continuation code and that redundant computations are done. Moreover, a better tuning of inner- and outer iterations is possible. So our conclusion is that the approaches are competitive for the 3D lid-driven cavity problem.

Chapter 5

Numerical results for Rayleigh-Bénard convection

In this chapter[1], besides results in chapter 3, more numerical experiment results are presented for Rayleigh-Bénard convection. Critical Rayleigh numbers are computed both for two and three dimensions on different grid size. It shows *weak scalability* of our algorithm in two-dimension. Primary bifurcations from the conductive states are also studied.

5.1 Numerical procedure and performance

In the experiments, we set the Rayleigh number as the continuation parameter and run on different grids, with the same number of grid points in each direction. For the 2D case, Table 5.1 shows horizontally how the iteration numbers and eigenvalues vary with the Rayleigh number, and vertically the influence of the grid on these numbers. Observe that Ra_c is between 2400 and 2600, and that more iterations of the Arnoldi process are needed in that interval. The reason for that is that the Jacobian matrix is getting almost singular as the Rayleigh number gets closer to its critical value. Note also that the critical eigenvalue varies smoothly with the continuation parameter Ra, which means that no jumping between solution branches occurs.

In Fig. 3.7 we have already shown the first three primary bifurcation flow

[1] A large part of this chapter has been published in the Proceedings of the 11th world congress on computational mechanics [48]

nx		2000	2200	2400	2600	2800	3000
32	eig.	-12.01	-7.7	-3.5	0.686	4.89	9.13
	ite.	47	48	49	53	49	49
64	eig.	-12.06	-7.92	-3.77	0.407	4.60	8.8
	ite.	65	67	70	76	70	69
128	eig.	-12.12	-7.98	-3.83	0.363	4.52	8.72
	ite.	72	74	77	85	77	76
256	eig.	-12.14	-8.00	-3.85	0.319		
	ite.	76	77	79	89		
512	eig.	-12.14	-8.01				
	ite.	77	78				
1024	eig.	-12.14					
	ite.	80					

Table 5.1: (2D Rayleigh-Bénard convection) Eigenvalues (with the number of iterations in last step of Arnoldi process) for different grids and Rayleigh numbers.

Flow pattern	nx=16	nx=32	nx=64	nx=128	Δ_{128-64}
x/y/single diagonal roll	3283	3360	3381	3387	0.0018
four rolls	5758	5840	5882	5898	0.0027
toroidal roll	7185	7260	7388	7400	0.0016

Table 5.2: (3D Rayleigh-Bénard convection) Convergence of the critical Rayleigh number as a function of used grid $nx = ny = nz$. The last column includes the relative differences between $nx = 128$ and $nx = 64$.

patterns in 3D; in Table 5.2, we show the Rayleigh numbers at which these bifurcations of the conductive state occur at various mesh sizes.

As written in section 3.2.2 we can find the critical Rayleigh numbers from $\det(\mathcal{J}(\mathrm{Ra})) = 0$. After discretization this leads to $\det(J(\mathrm{Ra})) = 0$. We would like to know when this will have a solution. To show this existence, note that the Rayleigh number appears as a coefficient in the equations. So, we can transform the determination of the critical Rayleigh number to an eigenvalue problem; instead of $\det(J(\mathrm{Ra})) = 0$, we can write $\det(\mathrm{Ra}\, J_1 - J_2) = 0$. By a similarity transformation, in fact replacing T by $\sqrt{\mathrm{Ra}}T$ we can transform this into a generalized eigenvalue problem for $\sqrt{\mathrm{Ra}}$ in which both matrices are symmetric $\sqrt{\mathrm{Ra}}\hat{J}_1 - \hat{J}_2$. Moreover, this is an eigenvalue problem on the space of divergence free velocities and on this space $\hat{J}_2$ is definite. Hence, all the eigenvalues $\sqrt{\mathrm{Ra}}$ will be real. For the implementation of this eigenvalue problem, we adapted the computation of the Jacobian to give us two matrices instead of one. The eigenvalue problem in itself is similar to what we had for the computations of the stability of the solutions where the Jacobian and the mass matrix are needed.

To solve this eigenvalue problem we used the Anazasi solver. We choose "Shift-and-invert" with shift zero to find the eigenvalues. HYMLS is used as preconditioner to solve the according linear system. Table 5.3 presents the results. By this approach, we can directly locate $\mathrm{Ra}_c = 2584$ for the 2D case, which is the same as that found by Gelfgat in[123], and $\mathrm{Ra}_c = 3387$ for the 3D case, which is in agreement with the $\mathrm{Ra}_c = 3446$ predicted by Catton (1970) using linear stability analysis and the $\mathrm{Ra}_c = 3800$ determined experimentally by Heitz and Westwater (1971) in a cubical cavity with nearly adiabatic lateral walls [124]. Besides, the critical value corresponds to the eigenvalue, which has geometric multiplicity two. This is consistent with Puigjaner's prediction (2004), in which she also showed that the two associated linear independent eigenvectors take the form of an x-roll and an y-roll. One observes that the Rayleigh numbers are converging nicely with the grid refinement. Moreover, the number of iterations increases monotonously and only very mildly with grid refinement for both 2D and 3D case. This shows the robustness of the preconditioner.

To study the *weak scalability* of the method on a parallel computer, we perform computations for a 2D case. The computer used is an opteron cluster with inifiniband connection between the nodes; every node contains 12 cores. The results are shown in Table 5.4. Again we have a square cavity, with equal number of grid points in each direction. L and np denote the number of levels in the preconditioner and the number of cores used, respectively. The number of GMRES(50) iterations performed by HYMLS in the last iteration of the Arnoldi

nx		iterations	Ra_c
16	2D	27	2517
	3D	54	3283
32	2D	39	2567
	3D	60	3360
64	2D	55	2581
	3D	89	3381
128	2D	59	2584
	3D	129	3387

Table 5.3: (2D and 3D Rayleigh-Bénard convection) Eigenvalues, number of iterations in last step of Arnoldi process and Ra_c for different grids sizes.

nx	L	np	it_{tot}	t_c	t_s
64	3	16	65	0.32	0.20
128	3	32	72	0.72	0.48
256	3	32	76	2.85	2.58
512	3	32	77	38.40	12.30
1024	3	128	80	728.00	54.40
1024	4	128	122	40.00	33.00
2048	4	128	124	56.00	104.00

Table 5.4: (2D Rayleigh-Bénard convection) Scalability test.

process is indicated by it_{tot}. t_c means the time the HYMLS takes to computes LU factorization and t_s means the time the linear solver takes. Stopping criterion for the linear solver is 1e-8. It appears that the eigenvalues come in pairs, i.e., if λ is an eigenvalue, then $-\lambda$ is also an eigenvalue. This can be proved similarly as above. If $(\lambda J_1 - J_2)v = 0$, then $(\lambda P J_1 P^{-1} - P J_2 P^{-1})Pv = 0$, where P is identity matrix except those elements in the temperature rows are -1 but 1. Looking into the matrix of J_1 and J_2, we can see $PJ_1P^{-1} = -J_1$ and $PJ_2P^{-1} = J_2$. So $(-\lambda J_1 - J_2)Pv = 0$, that means $-\lambda$ is also an eigenvalue, and the corresponding eigenvector is Pv.

Observe that the number of iterations is reaching a limit if the number of levels (L) is kept constant. We have proven this behavior in [81]. However, at the same time the size of the last Schur-complement increases on refinement by a factor 4,

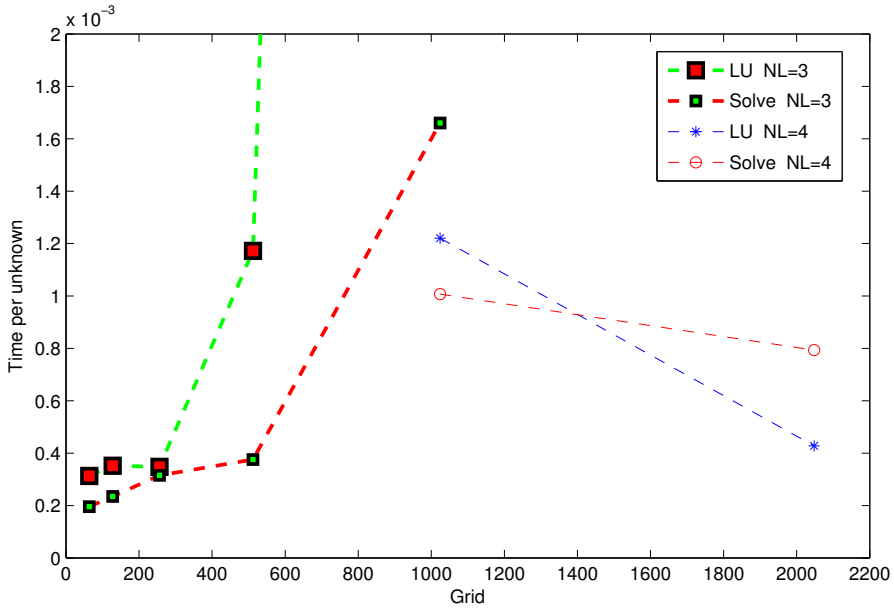


Figure 5.1: (2D Rayleigh-Bénard convection) Time per unknown for computation of LU factorization and Solve.

and the time of its factorization will dominate the computations. Therefore, we use an additional level and since the separator size in this case is 4, the size of the last Schur-complement will be 16 times smaller. This explains the large drop in computation time when we go from 3 to 4 levels keeping the same grid (1024^2). For the solve time, we see a similar but less pronounced behavior, since solving with a rather full L and U-factor is much cheaper than creating them.

In Fig. 5.1, we depicted the time which is needed per unknown to make an LU factorization and the time to solve the equations. These are obtained from the table by computing $(np \times tc)/(4nx^2)$, so the total CPU time consumed divided by the total number of unknowns. Hence, in case of an optimal speedup and grid independent convergence, one would see a horizontal line. In this case, we see for number of levels 4, NL=4, clearly the effect of keeping the number of levels constant; for small problems the effort needed for the Schur complement is little, but it increases rapidly once it dominates the computation time. This will eventually also occur for NL=4, but here it is still negligible and apparently parallelism is better exploited on the finer grid $nx = 2048$. The interesting point is that apart from the mentioned effect the time per unknown for the factorization does not in-

crease strongly if we increase the number of levels, a factor 2 for the last problem ($nx = 2048$) which is about 1000 times bigger than the first ($nx = 64$) and a factor 4 for the solve. These computations were carried out during normal operation of the cluster, and we did not have exclusive access to the computer resources for these runs and the timing results may be affected due to the distribution of the program over the various nodes, time sharing, etc. From this point of view, we consider the results obtained as very reasonable.

5.2 Away from the conductive state

Until now, computations for the trivial solution branch were presented. One way to get away from the trivial branch towards a *non-trivial stable branch* is to add a perturbation to the system just before it is meeting a bifurcation point. After switching to the non-trivial solution, the perturbation can be turned off again. We will show that a good choice for the perturbation is a small multiple of the critical eigenvector near the bifurcation point. Assume $F(u, \mathrm{Ra})$ is the right-hand side and it holds that $F(0, \mathrm{Ra}) = 0$. Now, we want to find the solution of $F(u, \mathrm{Ra}) = p$, where p is the perturbation. A Taylor expansion of $F(u, \mathrm{Ra})$ is given by $F(u, \mathrm{Ra}) \approx F(0, \mathrm{Ra}) + \frac{\partial F}{\partial u}(0, \mathrm{Ra})u$, which reduces to $F(u) \approx \frac{\partial F}{\partial u}(0, \mathrm{Ra})u$,

$$\frac{\partial F}{\partial u}(0, \mathrm{Ra})u \approx p \tag{5.1}$$

Now we choose $p = \varepsilon M v$, where M is the mass matrix, and v is the eigenvector obtained at $\mathrm{Ra} = \mathrm{Ra}_c$, which satisfies $\frac{\partial F}{\partial u}(0, \mathrm{Ra}_c)v = 0$. For this eigenvector, $\frac{\partial F}{\partial u}(0, \mathrm{Ra})v \approx \lambda(\mathrm{Ra})Mv$ with $\lambda(\mathrm{Ra}_c) = 0$ and λ is a continuous function of Ra. Solving (5.1), we get $u \approx \varepsilon v/\lambda(\mathrm{Ra})$. Though this equation is derived under the assumption that u is small, it shows that u grows rapidly if Ra gets near Ra_c. In Fig. 5.2 we see how this works out in practice. Here, the norm of the velocity and the opposite of the real part of the largest real eigenvalue $-\lambda$ are depicted with respect to the Rayleigh number. The red dot on the x-axis is the actual critical point, but the perturbation causes that the curve goes around it. From the real part of eigenvalues, shown in the left panel, we can see that the non-trivial branch is stable for all shown Rayleigh numbers. Similar behavior is found in the 3D case.

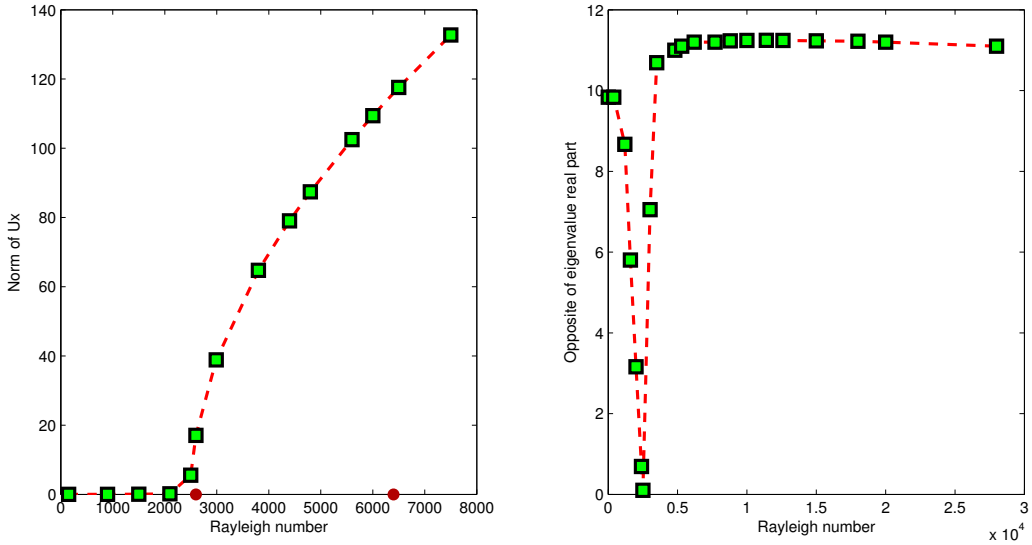


Figure 5.2: (2D Rayleigh-Bénard convection) Norm of velocity in x direction (left, two red points on x-axis indicate the first two critical Rayleigh numbers), and $-\lambda$ (right) w.r.t Rayleigh number.

5.3 Summary

We have discussed an implementation of a package for analyzing the dynamics of fluid flows coupled to transport equations. In this implementation we solve the coupled equations at once. This is quite natural when considering non-overlapping domain decomposition using separators (in fact leading to a kind of Nested Dissection approach). The problem that leads to fuller and fuller Schur-complement matrices is solved by the transform-and-drop approach used in HYMLS. This allows for a significant reduction of the number of unknowns on the separators, which is similar to an aggressive coarsening in multigrid context. At the same time we can keep the nice properties like symmetry and positiveness of matrices. This robust approach also admits a parallel implementation, which we performed using Trilinos packages. Experiments in this thesis show an acceptable weak scaling. These results were obtained on a time sharing computer. Improvements of the implementation are under way.

We have also shown that the results for the two benchmark problems obtained by the hybrid solver FVM/HYMLS are consistent with those in related literature. From this we conclude that the FVM/HYMLS solver makes it possible to per-

form bifurcation analysis and steady state computations on large CFD problems. Another advantage of it is that it is easy to extend it with more physics, for instance, temperature, salt etc.. It is known that the Arnoldi method needs accurate solutions of inner systems, which implies that much numerical effort is needed in an iterative solver. From the experiments, it is noticed that it takes a lot of iterations and time to converge an eigenvalue computation compared to solving steady states. To improve it, we resort to the state-of-art method Jacobi-Davidson QR (JDQR). The performance of JDQR will be described specifically in the next chapter.

Chapter 6

Numerical bifurcation analysis of a Turing-type reaction-diffusion model

In this chapter[1], we consider a widely studied model for spatial pattern formation, proposed by Turing in 1952 [64]. Turing showed that a system of two reacting and diffusing chemicals could produce spatial patterns in chemical concentrations from the destabilization of a homogeneous state. Many experimental results have illustrated the formation of striped and spotted patterns, as well as more complicated patterns [65]. The term diffusion-driven instability has occurred in chemical and ecological processes. Turing models can exhibit most of those patterns and they can be found in many theoretical and experimental papers. For an overview, see [66–68]. Bifurcation and stability studies of the steady solutions are also of great interest: Callahan gave the bifurcation diagram of the Brusselator model and the Lengyel–Epstein model [126, 127]. However, most of the results are more theoretical and computed on very coarse grids.

The general form of a Turing system for modeling the evolution of the concentrations of two chemicals is as follows

$$\begin{aligned} \frac{\partial U}{\partial t} &= D_U \nabla^2 U + f(U,V) \\ \frac{\partial V}{\partial t} &= D_V \nabla^2 V + g(U,V) \end{aligned} \tag{6.1}$$

[1] A large part of this chapter has been published in Communications in Nonlinear Science and Numerical Simulation [125]

where $U = U(x,t)$ and $V = V(x,t)$ are the two concentrations, and D_U and D_V are the diffusion coefficients, respectively. The scalar functions f and g represent the reactions between the components, which are usually nonlinear.

There are various Turing models with different reaction kinetics based on different applications, e.g., the Brusselator model [128], Gray-Scott model [129] and Lengyel- Epstein model [130]. In this thesis, we study the model brought up by Barrio *et al.* [73] in 1999, the Barrio-Varea-Aragon-Maini (BVAM) model. Our results can also be found in the paper [125]. As a general Turing model, it has applications in imitating the pattern formation on various fish species' skin [131]. The equations are obtained by expressing (6.1) in terms of a perturbation with respect to the stationary uniform solution (U_c, V_c), and then solving $f(U_c, V_c) = g(U_c, V_c) = 0$. Neglecting terms of order higher than 3, the equations are given by

$$\begin{aligned} \frac{\partial u}{\partial t} &= D\delta\nabla^2 u + \alpha u(1 - r_1 v^2) + v(1 - r_2 u), \\ \frac{\partial v}{\partial t} &= \delta\nabla^2 v + v(\beta + \alpha r_1 uv) + u(\gamma + r_2 v), \end{aligned} \tag{6.2}$$

where $u = U - U_c$ and $v = V - V_c$, so the point $(u, v) = (0, 0)$ is the stationary solution. The constant δ is a scaling factor and D is the ratio between the diffusion coefficients of the two chemicals. We note that D must not be equal to one in order to make the diffusion-driven instability occur [73]. There are two parameters r_1 and r_2 in the nonlinear interactions, affecting a cubic and quadratic term, respectively. In [73], it is observed that the cubic term favors stripe patterns and the quadratic ones spot patterns. In our experiments, we use periodic boundary conditions, and to make the investigation as simple as possible, we set $\alpha = -\gamma$, so that $(0, 0)$ is the only spatially uniform steady solution. In this work, we study only one set of parameters as indicated in Table 6.1. This set is one of the choices made in [73]. In the following we will determine branches of steady solutions of the equations above as a function of r_2 on, respectively, an interval, a square and a cube with edge length $L = 30$.

In Section 6.1, we analyze the model theoretically in terms of the linear stability of the trivial solution and determine unstable modes. These modes will later be used to get onto non-trivial solution branches. Furthermore, we discuss some properties of the non-trivial solutions, such as degrees of freedom, symmetry and similarity of the solutions. The main numerical methods we use are introduced in Section 6.2: the continuation methodology and the Jacobi-Davidson method for computing eigenpairs for the linear stability analysis. Both techniques require the

D	α	β	δ	r_1	L
0.516	0.899	-0.91	2	3.5	30

Table 6.1: Parameter values.

solution of large and sparse linear systems. To solve these systems iteratively, a multigrid preconditioned Krylov solver is used.

In Section 6.3 we present numerical results in 1D, 2D and 3D, including bifurcation diagrams, stability of various solution patterns and an overview of the performance of the algorithms and the parallel implementation.

6.1 Model Analysis

In the following part, a mathematical analysis will be carried out in order to get a better idea of the characteristics of the solutions of Eq. 6.2.

6.1.1 Linear stability analysis of the trivial solution

We follow the exposition in [73]. We start with noting that the nonlinear part of the right-hand side of (6.2) will not contribute to its Jacobian. Now, in the absence of diffusion, standard linear analysis predicts exponentially growing solutions of the form $(u, v) = (u_0 \exp(\lambda t), v_0 \exp(\lambda t))$ where λ is an eigenvalue, with

$$\lambda = \frac{1}{2}[(\alpha + \beta) \pm \sqrt{(\alpha + \beta)^2 - 4\alpha\beta - \gamma}]. \tag{6.3}$$

In the presence of diffusion, the spatial variation of the functions u and v is of the form $\exp(i\overline{k} \cdot \overline{x})$, and the dispersion relation of the linearized equations is given by

$$\lambda^2 + B\lambda + C = 0, \tag{6.4}$$

with $B = -k^2\delta(1 + D) + \alpha + \beta$ and $C = (\alpha - \delta D k^2)(\beta - \delta k^2) + \alpha$, $k^2 = \overline{k} \cdot \overline{k}$.

Figure 6.1 shows the real and imaginary parts of the two eigenvalues of equation (6.4) as a function of the wave number k with the parameter values given in Table 6.1. For $k \leq 0.34$, there is a complex pair of conjugate eigenvalues, the real part of which is less than zero, hence the trivial solution is stable there. For

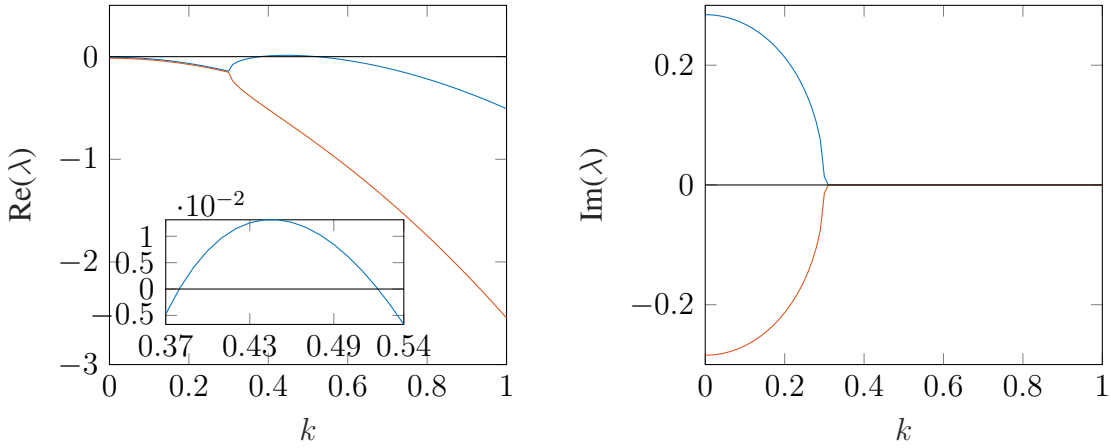


Figure 6.1: Two eigenvalues of the linearized equations (6.4) with respect to the wave number k.

$k \approx 0.34$ these two eigenvalues turn into two real eigenvalues, one of which is positive in the range

$$[0.39, 0.52]. \tag{6.5}$$

So if the size of the d-dimensional domain allows wave numbers k in this interval then the trivial solution will be unstable, otherwise it will be stable. Next we will study whether they can occur on generalized squares of size $L = 30$ in 1, 2 and 3 dimensions, respectively.

Unstable modes in 1D, 2D and 3D. For the parameters in Table 6.1, and a specific size of the domain, say $L = 30$, we can now specify in the various dimensions which modes become unstable. First we define the constant $\kappa = 2\pi/L \approx 0.2094$, which will be used throughout this chapter. In 1D the wave numbers are given by $k = \kappa m$, with $m = 1, 2, \cdots$. For $m = 2$ we find the sole wave number in the interval (6.5): $k = 0.4189$. In 2D, the wave numbers are given by $k = \kappa\sqrt{m^2 + n^2}$, with $m, n = 0, 1, 2, \cdots$ (and m and n not both zero) and in 3D similarly $k = \kappa\sqrt{m^2 + n^2 + l^2}$. Note that this relates to wave vectors $\mathbf{k} = \kappa[m; n]$ and $\mathbf{k} = \kappa[m; n; l]$ in 2 and 3 dimensions, respectively. In Table 6.2, we list the wave numbers of the modes making the zero solution unstable together with those of surrounding stable modes.

Due to the fact that we are working on squares and cubes, the values may be randomly permuted over m, n and l. We also gave names to the unstable

1D, 2D, 3D

type	l, m, n	k	name
stable	1,0,0	0.2094	
unstable	2,0,0	0.4189	S2
stable	3,0,0	0.6283	

2D,3D

type	l, m, n	k	name
stable	1,1,0	0.2962	
unstable	2,1,0	0.4683	S3,S4
stable	2,2,0	0.5924	S1

3D

type	l, m, n	k	name
stable	1,1,1	0.3628	S6
unstable	2,1,1	0.5130	S7,S8,S9
stable	2,2,0	0.5924	
stable	2,2,1	0.6283	

Table 6.2: Stable and unstable modes with respect to the trivial solution. Unstable modes occur in the interval (6.5).

modes, because we will use these later to name the branches of solutions that they generate. Observe that the (2,1,0) mode has two names. This is due to the fact that for a square there is a rotated 1D mode and a genuine 2D mode, which have the same wave number. On the one hand we have $\cos((2x + y)\kappa)$, which we will call S3, and on the other hand $\cos(2x\kappa)\cos(y\kappa)$, which we call S4. Note that due to the periodic boundary conditions all cosines may have a different phase shift. Note also that $\cos((2x + y)\kappa) = \cos(2x\kappa)\cos(y\kappa) - \sin(2x\kappa)\sin(y\kappa)$, which relates the two modes. The product of sines here is just a phase shifted version of the product of cosines. Moreover, in the right-hand side we could differentiate the wave numbers for the x and y direction to adjust to a non square domain. Since we cannot do this in the left-hand side, the rotated mode will not appear on a non-square domain. However, due to the combination in the right-hand side we

are able to see something almost of that shape if the lengths of the sides differ not much from each other.

Similar phenomena occur in 3D. Observe that the mode (2,1,1) has three names: S7, S8 and S9, which are a truly 3D droplet pattern started with initial solution $\cos(2\kappa x)\cos(\kappa y)\cos(\kappa z)$, a tilted lamellae pattern started with $\cos(2\kappa x + \kappa y + \kappa z)$ and a tilted cylinder started with $\cos(2\kappa x)\cos(\kappa y + \kappa z)$, respectively. These are also related to goniometric identities, e.g., $\cos(2\kappa x + \kappa y + \kappa z) = \cos(2\kappa x)\cos(\kappa y + \kappa z) - \sin(2\kappa x)\sin(\kappa y + \kappa z)$.

We remark that the choice of L also determines how many unstable modes will occur. For instance, if L is doubled (as is done in [70, 73]), the modes in the table reoccur but with wave number twice as big in all directions. So l, m, n in Table 6.2 become even numbers, which opens up the possibility that a combination with an odd number will drop into the interval (6.5). A simple computation reveals that this is indeed the case.

6.1.2 Magnitude estimate of non-trivial solution

A mode that makes the trivial solution unstable may turn into a stable solution of the nonlinear equations. By performing a Galerkin projection of the problem, an estimate of the magnitude of such a solution can be obtained. We show this process for the 1D case, but equally well it can be done in the 2D and 3D case.

We write the steady state system related to (6.2) as the vector equation $F(\mathbf{u}) = J\mathbf{u} + N(\mathbf{u}) = 0$, where J is the Jacobian matrix of F and N is the remaining nonlinear part. Here, the Jacobian is the same matrix for which the eigenvalues have been determined. Let $\mathbf{v}$ denote the eigenvector associated with the positive eigenvalue; note that eigenvalue and eigenvector are both real. Next we express $\mathbf{u}$ in this unstable mode and call this specific choice $\mathbf{u}_k$, so

$$\mathbf{u}_k \equiv \epsilon \mathbf{v} \sin(kx)$$

with $||\mathbf{v}|| = 1$. By the Galerkin approach, it is required that

$$\int_{x=0}^{L_x} \sin(kx)\mathbf{v}^T F(\mathbf{u}_k)dx = 0,$$

which results in

$$\int_{x=0}^{L_x} \sin(kx)\mathbf{v}^T[\lambda \mathbf{v}\sin(kx) + N(\mathbf{u}_k)/\epsilon)]dx = 0.$$

Suppose $N(\mathbf{u}) = Q(\mathbf{u}) + C(\mathbf{u})$, where Q and C are quadratic and cubic in $\mathbf{u}$, respectively. Then we can write

$$N(\mathbf{u}_k)/\epsilon = \epsilon \sin^2(kx)Q(\mathbf{v}) + \epsilon^2 \sin^3(kx)C(\mathbf{v}).$$

Suppose integrals over the domain of $\sin^2, \sin^3, \sin^4$ are respectively $a,\ 0,\ b$ then we end up with an equation of the form

$$\lambda a + \epsilon^2 b \mathbf{v}^T C(\mathbf{v}) = 0,$$

with

$$\epsilon = \sqrt{-\lambda a/(b\mathbf{v}^T C(\mathbf{v}))}.$$

In our case $C(\mathbf{u}) = \alpha r_1 u_1 u_2^2[-1; 1]$ and $\mathbf{v}^T C(\mathbf{v}) = \alpha r_1 v_1 v_2^2(v_2 - v_1)$, where $\mathbf{u} = [u_1; u_2]$ and similar for $\mathbf{v}$. To get a real solution, ϵ should be real. This is indeed the case for $k = 0.4189$ with the parameters specified in Table 6.1. For these values $\mathbf{v} = [0.8167, -0.5771]$ and $\epsilon = 0.1122$.

The knowledge gained in this section helps us to start the continuation and we will come back to this later.

6.1.3 Some properties of the solutions

Nonuniqueness of the solutions. Due to the periodic boundary conditions one can shift the solutions around in the plane. In principle, one has to apply a (phase-) condition to prohibit this. This also results in zero eigenvalues of the Jacobian at the solutions. The number of zero eigenvalues depends on the solution. A 1D solution can only be shifted in one direction so it has a single eigenvalue zero. This also holds for generalizations of 1D solutions to 2D and 3D. A genuine 2D solution, i.e. one which cannot be found from a generalization of a 1D solution, can be shifted in two directions, each shift giving another solution, so here we have two zero eigenvalues. Similarly, in three dimensions there are three zero eigenvalues for a genuine 3D solution.

Symmetries and coinciding eigenvalues. Apart from multiple zero eigenvalues one also finds equal non-zero eigenvalues. For instance, if in 2D the solution corresponding to the (2,0) mode is studied, we will find an unstable mode of genuine two dimensional shape. This mode may be shifted arbitrarily in the y-direction and gives rise to an independent eigenvector. Hence, this builds a two-dimensional subspace of unstable modes and consequently leads to a double positive eigenvalue.

Similarly, the 3D mode (2,1,1) has in itself already three representations, as we have seen above. Moreover, the 2 can be at three positions, leading to at least 9 equal eigenvalues. In a numerical computation we found even up to 24 equal eigenvalues. We will get back to that in Section 6.3.3 where we study some of the 3D modes.

Symmetry in solutions is a much studied subject, see for instance [132–134]. It is possible to solve the problem on a small portion of the domain with various boundary conditions, whereupon various combinations of these solutions gives the whole range of solutions. We did not exploit this in the thesis.

Similarity solutions. In Table 6.1 we fixed $r_1 = 3.5$. We will show now that if we compute the solutions for all r_2, then we do so for any pair (r_1, r_2).

Suppose at certain parameter values $(\hat{r}_1,\ \hat{r}_2)$ we have a steady state $(\hat{u},\ \hat{v})$. Now we wonder for which values $(r_1,\ r_2)$, $(\mu\hat{u},\ \mu\hat{v})$ is a solution. If we substitute this into Equation (6.2) we have that

$$\begin{aligned} 0 =& \mu D\delta\nabla^2\hat{u} + \alpha\mu\hat{u}(1 - r_1\mu^2\hat{v}^2) + \mu\hat{v}(1 - r_2\mu\hat{u}), \\ 0 =& \mu\delta\nabla^2\hat{v} + \mu\hat{v}(\beta + \alpha r_1\mu^2\hat{u}\hat{v}) + \mu\hat{u}(\gamma + r_2\mu\hat{v}). \end{aligned} \tag{6.6}$$

After dividing by μ we find

$$\begin{aligned} 0 =& D\delta\nabla^2\hat{u} + \alpha\hat{u}(1 - r_1\mu^2\hat{v}^2) + \hat{v}(1 - r_2\mu\hat{u}), \\ 0 =& \delta\nabla^2\hat{v} + \hat{v}(\beta + \alpha r_1\mu^2\hat{u}\hat{v}) + \hat{u}(\gamma + r_2\mu\hat{v}). \end{aligned} \tag{6.7}$$

So $(\mu\hat{u},\ \mu\hat{v})$ is a solution if both $\mu^2 r_1 = \hat{r}_1$ and $\mu r_2 = \hat{r}_2$. The family of solutions defined by $\hat{r}_1,\ \hat{r}_2$ and μ has the same stability behavior, with exactly the same eigenvalues.

6.2 Numerical Methods

We use a standard second order central finite difference scheme (3-point for 1D, 5-point for 2D and 7-point for 3D) to discretize (6.2) in space. An equidistant grid is used in all of our experiments. Rather than discretizing the time dimension as well, we focus on the direct computation of steady states using pseudo-arclength continuation [22]. An implementation of the algorithm is available in the Trilinos [135] library LOCA ("Library of Continuation Algorithms"). The arising linear systems are solved using the well-known GMRES method (Trilinos package Belos) with preconditioner HYMLS or an algebraic multigrid preconditioner

(ML package). Eigenvalues and -vectors are computed using PHIST, a recent implementation of the Jacobi-Davidson method [75] that allows for non-symmetric matrices and easy integration with Trilinos applications. Below, the methods are briefly outlined in order to make the text more accessible.

6.2.1 Continuation approach

Our research is focused on the steady state of system 6.2. We use the same continuation algorithm described above in Chapter 2 section 2.1. By continuation in a specific parameter r_1 or r_2, a series of approximate solutions can be generated by solving a system of parameterized nonlinear equations.

6.2.2 Solution of linear systems

The linearized systems that arise in the continuation process take the form of two independent Laplace operators on the diagonal with off-diagonals for the coupling of the unknowns u and v. The coupling makes the matrix non-symmetric, so the solver of choice is the GMRES Krylov subspace method. We observe that the coupling terms are constant, independent of the grid size $\Delta x = x_{i+1} - x_i$. Therefore, at sufficiently high resolution, the Laplace terms dominate the convergence of the iterative solver. Multigrid preconditioning is an obvious choice to keep the number of linear solver iterations at bay. Being readily available in Trilinos, we choose the smoothed aggregation AMG solver ML, see [42] for documentation. Although the systems solved are all singular as discussed in Section 6.1.3, the multigrid method will converge to a particular solution in the solution space. This often occurs with iterative procedures, as long as matrices inverted in those procedures are non-singular. We have experienced that a direct method does indeed fail.

6.2.3 Linear stability analysis by eigenvalue computation

We use the block variant of JDQR implemented in PHIST to compute the eigenvalues to determine the stability of steady state and most unstable eigenmode to locate the bifurcations. It is found that using an AMG cycle on the linear part of Eqn. 6.2 for the action of P^{-1} in Equ. 2.25 achieves good convergence behavior of the overall method. This choice is motivated by the fact that the sought eigenvalues are close to 0, so that neglecting the shifts $\tilde{\lambda}_j$ in the preconditioner is a reasonable approximation.

6.2.4 Branch switching

Leaving the trivial branch. The general way to get on the branch of non-trivial solutions is to solve the eigenvalue problem for the Jacobian matrix of the discretized problem and select those modes that have a positive eigenvalue. With each of these one can create a Galerkin projection as indicated in Section 6.1.2 and find its approximate magnitude. At the end of that section a specific solution is given. Based on this our starting solution will have the form

$$0.1 \cdot \phi(x, y, z)[1; -1], \tag{6.8}$$

where ϕ is a combination of goniometric functions related to the unstable modes in Table 6.2. This guess appeared to be good enough to get convergence to the associated mode of the nonlinear system.

Switching between non-trivial branches. Once we are on a non-trivial branch we might also need to get onto another branch. The direction where to go is given by the eigenvector $\mathbf{v}$ associated with the unstable eigenvalue. If $\mathbf{u}$ is the steady state solution on the current branch near the bifurcation point, we use as an initial guess $\mathbf{u} + \epsilon\mathbf{v}$ for some small values of ϵ for the next Newton iteration, which will then typically converge to a nearby solution on the new branch.

6.2.5 Differences with time integration approach

In the continuation approach, the time derivative disappears, but is implicitly still there, as we will explain here. In order to study the stability we want to find solutions of the form $\mathbf{u}(\mathbf{x}, t) = \bar{\mathbf{u}}(\mathbf{x}) + \epsilon(t)\mathbf{v}(\mathbf{x})$, where $\bar{\mathbf{u}}(\mathbf{x})$ is a steady state solution. Inserting this in the equation $d\mathbf{u}/dt = F(\mathbf{u})$, we obtain for small $\epsilon(t)$ the equation

$$\frac{d\epsilon}{dt}\mathbf{v} = \epsilon J(\bar{\mathbf{u}})\mathbf{v}.$$

This linear equation allows for separation of variables, leading to

$$\frac{d\epsilon}{dt} = \lambda\epsilon, \quad \lambda\mathbf{v} = J(\bar{\mathbf{u}})\mathbf{v}.$$

The former has the solution $\epsilon(t) = \epsilon(0)\exp(\lambda t)$, and the latter has eigenvalues and eigenvectors as its solution. Once the eigenproblem is solved we find solutions of the specified form $\mathbf{u}(\mathbf{x}, t) = \bar{\mathbf{u}}(\mathbf{x}) + \epsilon(0)\exp(\lambda t)\mathbf{v}(\mathbf{x})$.

For a time integrator one can perform the analogous process. For instance, for the forward Euler method one finds a different equation for $\epsilon(t)$

$$\epsilon(t + \Delta t) = \epsilon(t) + \lambda \Delta t \epsilon(t) = (1 + \lambda \Delta t)\epsilon(t).$$

In this case, the solution is $\mathbf{u}(\mathbf{x}, n\Delta t) = \bar{\mathbf{u}}(\mathbf{x}) + \epsilon(0)(1 + \lambda\Delta t)^n \mathbf{v}(\mathbf{x})$. Note that $(1 + \lambda\Delta t)$ is a crude approximation of the exponential $\exp(\lambda\Delta t)$; it is called the amplification factor below.

So in reality, the perturbation damps out (amplifies) if $\mathcal{R}(\lambda) < 0$ ($\mathcal{R}(\lambda) > 0$), but for the Euler integrated version this occurs if $|1 + \lambda\Delta t| < 1$ ($|1 + \lambda\Delta t| > 1$), which is quite different if Δt is big.

Now, let us return to the original problem, where one gives an initial guess and performs a stable integration with the forward Euler method. Then by varying the initial conditions one may find various steady states. However, the consequence of the above is that only steady states can be found for which $|1+\lambda\Delta t| < 1$, for all eigenvalues λ that occur. From this, it is immediately clear that an unstable steady state (at least one positive eigenvalue) cannot be found. Moreover, solutions which are stable but with an eigenvalue containing a relative large imaginary component, i.e. near a Hopf bifurcation, cannot be found if for that eigenvalue $|1 + \lambda\Delta t| > 1$. So even if we would start with such a steady solution we would be repulsed from it. This is a peculiarity one should be aware of when using the forward Euler method.

Another popular scheme is the backward Euler method. In that method, the amplification is $1/|1 - \lambda\Delta t|$, and one can take arbitrarily large time steps. However, one may find some stationary solutions even if they are unstable, wrongly concluding that they are stable since the numerical method converged.

As will be clear by now every time integration method has its own peculiarities, which can influence our conclusions with respect to stability. This cannot happen with the continuation approach, which makes it mathematically superior here.

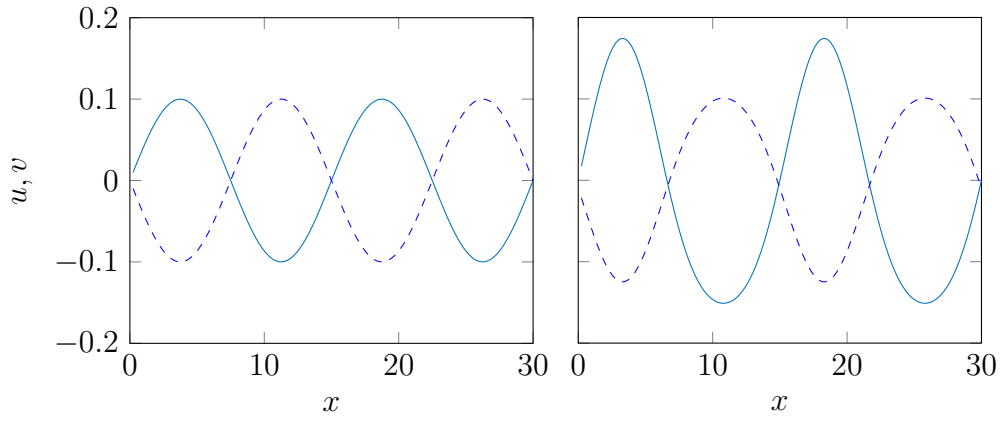


Figure 6.2: Non-trivial 1D solution components u (solid) and v (dashed), left for $r_2 = 0$ and right for $r_2 = 1$.

6.3 Numerical Experiments

In this section, we will show results for the 1D, 2D and 3D case respectively. In all cases we use a grid with 128 nodes in each direction. A justification for this is given in the paragraph "Refinement tests" at the end of Section 6.3.2.

6.3.1 Non-trivial solution in 1D

Table 6.2 shows that we have one stable nontrivial solution S2. If we choose $\phi(x) = \sin(2\kappa x)$ in (6.8) we indeed find it as a solution. In Fig. 6.2 this solution is shown for $r_2 = 0$ and $r_2 = 1$. Observe that u gets narrower tops and wider valleys where for v this is vice versa. Below we will only consider the interval [0,1], but here we remark that this solution is stable until about 1.58. At that point we find a Hopf-bifurcation leading to a stable periodic solution, with period $2\pi/0.1772$. This solution is stable for all values of r_2 between 0 and 1. It will be clear that this will also be a solution of the 2D and 3D case. However, we may not conclude that it is stable in the 2D and 3D case. But if it is unstable in the 2D case, then it must be due to a field that is genuinely 2D.

6.3.2 Bifurcation diagram of the 2D case

In our experiment, except for the uniform zero solution, five non-zero branches S1–S5 are found with appropriate initial vectors based on the model analysis.

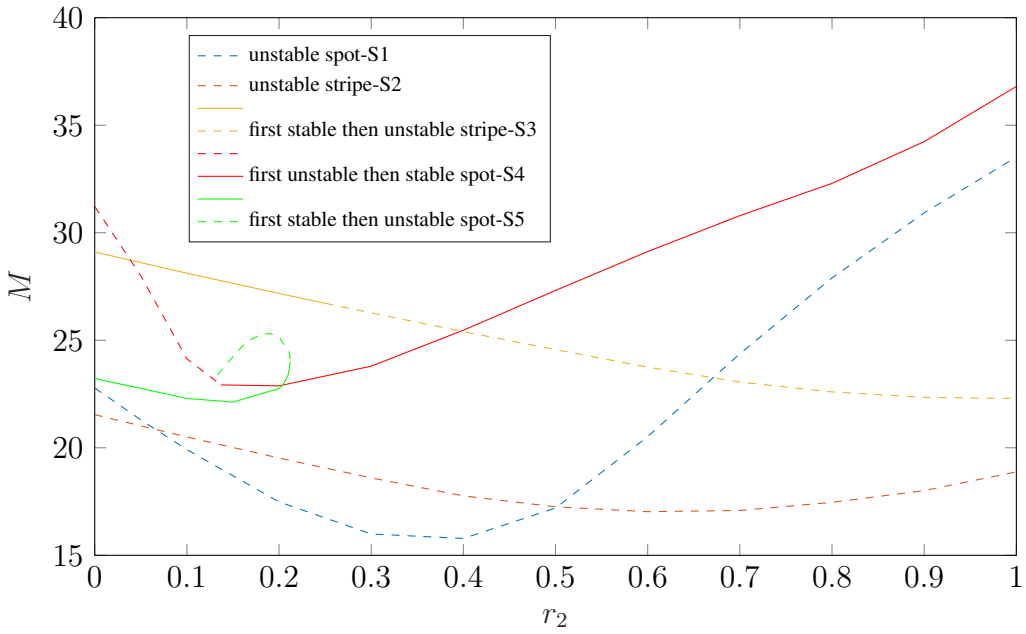


Figure 6.3: Bifurcation diagram in 2D with parameter r_2 varying from 0.0 to 1.0. Stable and unstable branches are depicted by solid and dashed lines, respectively.

There are both stable and unstable stripe and spot patterns, see the bifurcation diagram in Fig. 6.3. The monitor we use here and in the following is defined by

$$M = \max_{x,y} \Delta u, \tag{6.9}$$

where Δ is the Laplacian. Being sensitive to curvature, M gives an insightful bifurcation diagram, but other choices are also possible.

The S2/S4 solution range. Below we start with a mode of the form S2 and will discover that also a mode of the form S4 is playing an important role. We show that the nonlinear behavior is occurring in a space built up by deformations of these two modes.

S2 Taking $\phi(x, y) = \sin(2\kappa x)$ gives the generalization of the 1D case, see Fig. A.1. In 2D it is unstable, as expected, due to a mode with a genuinely 2D pattern resembling the S4 mode, see Fig. A.2. Since this mode makes S2 unstable we guess that a combination of the two becomes the stable one.

S5 By starting with the sum of the solution of $r_2 = 0.0$ on S2 and its eigenvector (an S4 mode), corresponding to the positive eigenvalue which makes the stripe pattern S2 unstable (see Fig. A.2), the system converges to a solution on a stable branch which we will call S5, see Fig. A.3. At $r_2 = 0.0$ it is a wavy striped pattern. Increasing r_2 to 0.21227, the spot (S4 mode) is getting more pronounced. At this value of r_2 there is a turning point which shows up by a new zero eigenvalue. See the bifurcation diagram for the 2D case (Fig. 6.3). When r_2 is decreasing after the turning point, the spot pattern (the S4 mode) is becoming even more pronounced in the solution; see Fig. A.4. The continuation parameter r_2 goes back until 0.129, where the branch connects to the S4 branch, which we will study next.

S4 When choosing $\phi(x, y) = \cos(2\kappa x)\cos(\kappa y)$ in (6.8), the S4 mode with hexagonal spots is excited and Newton converges to a non trivial solution with the same pattern. It is unstable up to the bifurcation point $r_2 = 0.129$, where S5 splits off, and after that it is stable; see Fig. A.5. Observe that the second plot, i.e. the one for $r_2 = 0.1$ is similar to the last of Fig. A.4, the solution of S5 near the bifurcation point. Also note the symmetry that is growing in the y direction when r_2 is decreasing (Fig. A.4). It is clear that a sister of S5 would be the one where the yellow band is just going along the other sides of the spots. This will be the other solution emanating at the bifurcation point. Hence we have a pitchfork bifurcation here, and because the symmetry breaks, it is also a symmetry breaking bifurcation.

Summarizing, we see that the modes S2 and S4 and their combination give a partial solution to the problem. At $r_2 = 0$, a combination (S5) is stable until 0.212. When in a time-dependent simulation we increase r_2 from there, one will always converge to the stable S4 branch. The interesting thing is that S4 is also stable before 0.212, starting at 0.129, so there are two stable solutions (S4 and S5) on the interval $[0.129, 0.212]$. Before 0.129, S4 is unstable and hence S5 is the only stable solution there. We also understood why S4 changes stability at 0.129 and why S5 ceases to be stable at 0.212: S5 has a turning point at 0.212 and returns to S4 at 0.129, being unstable along that branch. For large r_2 only S4 is stable which also agrees with the observation in [73] that the quadratic term favors spots.

The S3 solution area The S3 solution is a tilted version of the S2 solution. One might expect a similar behaviour but that is actual not the case as we will see below

S3 When taking $\phi = \cos(\kappa(2x + y))$ we find again a stripe solution which is oriented in the diagonal direction (Fig. A.6). Note that the stripe can be tilted in any direction, depending on the initial solution. It is observed that at the beginning all the eigenvalues are negative. With increase of r_2, around $r_2 = 0.25$, there is a pair of positive eigenvalue occurs, having the same value. In Fig. 6.4, distribution of the first 10 eigenvalues with largest real part is shown. All of them are real and some of them have algebraic multiplicity more than two. The corresponding eigenvectors are spot pattern, which make the stripe unstable, see Fig. A.7.
Besides, there is a pair of complex conjugate eigenvalues. With the increase of r_2, its real part is getting closer and closer to zero. In the range $1.625 < r_2 < 1.65$, it crosses the imaginary axis, the eigenvalue becomes pure imaginary 0.2, which means there is Hopf bifurcation and periodic solution will emerge with frequency $f = 0.2$. However, the periodic solution can not be computed by solving the steady state. The bifurcation occurring at the double zero eigenvalue (a so-called Bogdanov-Takens bifurcation) is difficult to deal with in the approach we use. Hence, we do not present results for that. One could imagine though what is going to happen. One expects a stable and unstable branch in the two dimensional space splitting off. On the stable branch we will have to deal with a Jacobian that is nearly singular, which may hamper the convergence of Newton's method. Indeed, we tried to get on the branch but it jumps always to solution S4.

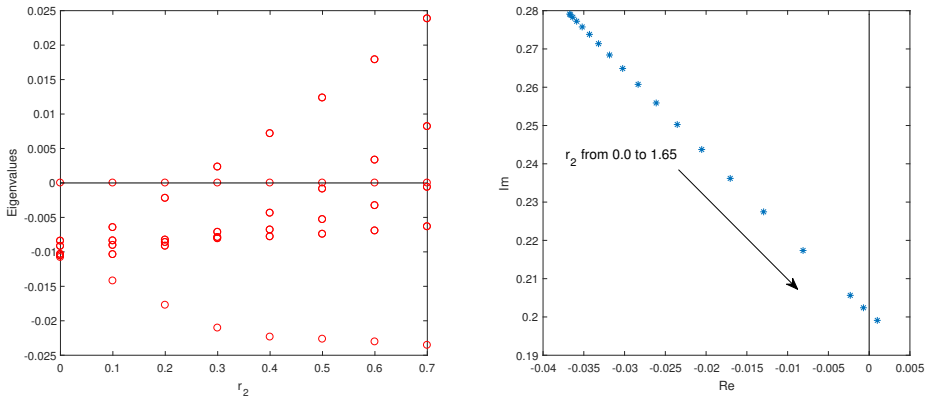


Figure 6.4: Evolution of eigenvalues with largest real part as a function of the r_2 on branch S3 (left) and one of the conjugate complex eigenvalues (right).

Summarizing, the S3 solution is stable only for values of r_2 up to 0.25. This means that together with the previous solutions, there are three stable solutions on the interval [0.129, 0.212]. The eigenvectors appearing at 0.25 resemble a tilted S4, but we were not able to find a solution when starting with a tilted field resembling the eigenvector pattern: $\cos(\kappa(x/2+y))\cos(\kappa(y-3x/2)) = \cos(\kappa(x-2y)) - \cos(2\kappa x)$. This field has the right periodicity conditions but is not an eigenmode of the linear part of the equation, since the two cosines in the last expression have different eigenvalues ($5\kappa^2$ and $4\kappa^2$ for the Laplace operator, respectively).

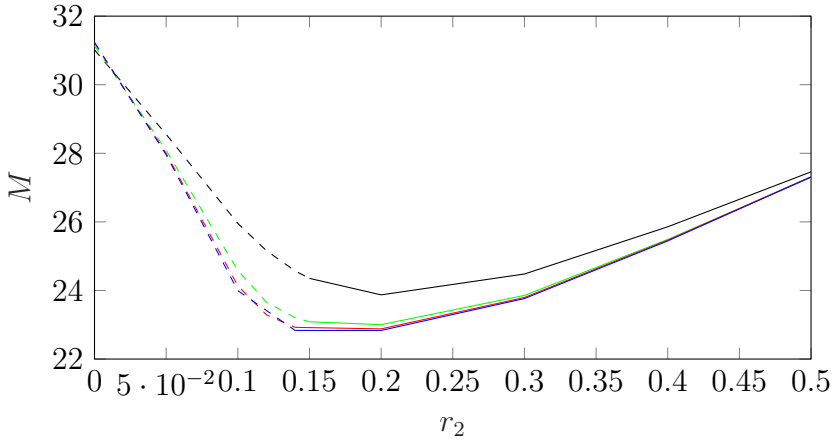


Figure 6.5: Bifurcation diagram obtained on a 32^2 (black), 64^2 (green), 128^2 (red) and grid 256^2 (blue) grid on branch S4.

Refinement test Fig. 6.5, following branch, shows the expected second-order accuracy of the discretization with grid refinement. Observe that the solution on grid 256^2 is almost the same as that on grid 128^2. Hence, it is enough to use the 128 grid points for our experiments. To get an impression of the accuracy of the position of the bifurcation points, we perform a sensitivity analysis for the eigenvalues of S4 for the number of grid points. So we want to locate the bifurcation parameter r_2, when the solution becomes stable, in other words, when the positive eigenvalue crosses the imaginary axis, the eigenvalues at different grids are computed. In Table 6.3, it shows the eigenvalue, with largest real part except zero, changing with the increase of r_2, which also implies the property of second-order convergence of our algorithm. By extrapolation based on the second-order behaviour of the error, the estimated critical r_2 can be computed, presented in the last column.

$grid$	$r_2 = 0.14$	$r_2 = 0.145$	$r_2 = 0.15$	critical r_2
32^2	1.308e-03	1.010e-03	7.146e-04	0.1621
64^2	2.476e-04	-4.953e-05	-3.409e-04	0.1442
128^2	-1.040e-04	-3.955e-04	-6.479e-04	0.1384
256^2	-2.010e-04	-4.900e-04	-7.30e-04	0.1369

Table 6.3: The positive eigenvalue as a function of used grid.

Comparison to results from literature [73] and [136] also report 2D numerical results of Eq. 6.2. They started with random initial solutions and, after hundreds of thousands time steps using the Euler method, spot, stripe and hexagonal spot patterns are found for different parameter values r_1 and r_2. They observed that the cubic term favors stripe patterns while the quadratic term favors spot patterns. However, the cubic term scaled by r_1 does not completely suppress the occurrence of spot patterns. We also found the unstable spot patterns when $r_1 = 3.5$ and $r_2 = 0.0$, which is impossible by a time integration approach. Our results also show that when r_2 increases, the spot pattern is the only stable solution occurring and therefore it will be very robust in a time integration approach. Here, we have seen that there are regions in which 2 or even 3 stable solutions exist for the same parameter value.

6.3.3 Results for 3D

There are more morphologies in three dimensions, since besides the generalizations of the 2D solutions there will be new solutions. The 3D bifurcation diagram of branches S1–S5, including their stability properties, appeared to be exactly the same with those of 2D case. Therefore, in the bifurcation diagram for the 3D case we only plot the branches of genuine 3D solutions, i.e., in Fig. 6.6 we only plot the branches S6–S9, which are all unstable in the computed range. Next we discuss the classes of solutions separately.

Generalized 1D mode. We start off with the mode occurring already in the 1D case: S2. As in 2D, taking $\phi(x, y) = \sin(2\kappa x)$ again, we obtain the generalization of the 2D stripe pattern, which is also unstable, as expected. Since all constants have remained the same, the eigenvalues are also the same as those in 2D. But there are two additional independent eigenvectors appearing, i.e. four in total

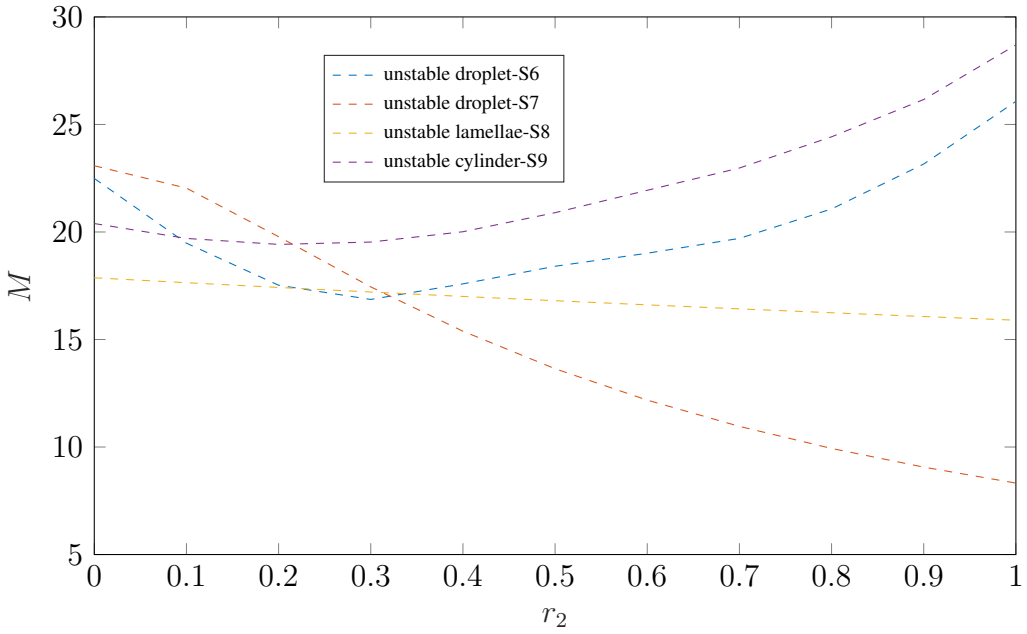


Figure 6.6: Bifurcation diagram in 3D with parameter r_2 varying from 0.0 to 1.0, obtained on a 128^3 grid. For branches S1-S5 we refer to Figure 6.3

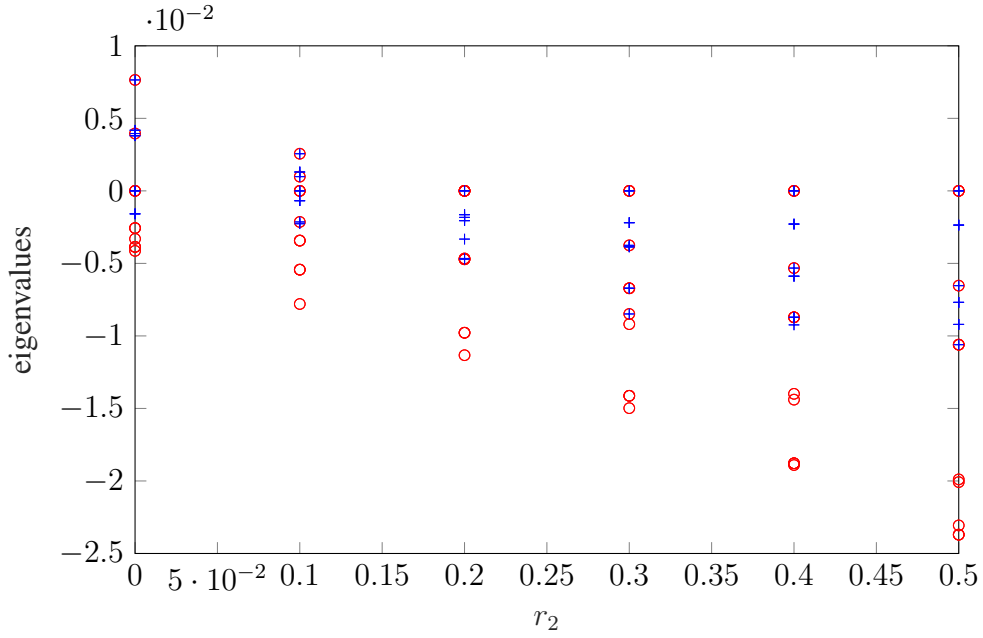


Figure 6.7: 2D eigenvalues (○) and 3D eigenvalues (+) as a function of r_2 on the S4 branch.

corresponding to the positive eigenvalue. In fact we find two pairs of eigenspaces, where each pair is an exact generalization of the 2D case: the S4 mode extended cylindrically. The pairs are just rotated 90 degrees with respect to each other. Since these modes make S2 unstable, a combination of the two becomes the stable one, which is S5 explained below.

Generalized 2D modes. Next we considered the modes which were really of 2D shape. It appears that all modes S3, S4 and S5, have a similar behavior as in the 2D case. It seems that for all these cases the 3D-generalized 2D eigenvalues dominate over new eigenvalues corresponding to modes with truly 3D patterns. An example of that is shown in Fig. 6.7 for S4. The eigenvalues from the 2D case are indicated by a '○' while the new ones entering in the 3D case are indicated by a '+'.

The observation that for increasing parameter r_2 the spot pattern prevails does not seem to generalize to the 3D droplet solution. Instead the generalized S4 mode, which has a cylindrical pattern, prevails over a genuine 3D droplet pattern, most likely because the latter is not an unstable mode of the zero solution.

name	function ϕ for initial guess (u, v)	pattern
S6	$\sin(\kappa x)\sin(\kappa y)\sin(\kappa z)[1,-1]$	droplet (Fig. A.8)
S7	$\cos(2\kappa x)\cos(\kappa y)\cos(\kappa z)[1,-1]$	droplet (Fig. A.9)
S8	$\cos(2\kappa x+\kappa y+\kappa z)[1,-1]$	tilted lamellae (Fig. A.10)
S9	$\cos(2\kappa x+\kappa y)\cos(\kappa z)[1,-1]$	tilted cylinder (Fig. A.11)

Table 6.4: List of genuinely 3D solutions and initial conditions used to find them. All of these modes are unstable in the the range $0 \le r_2 \le 1$.

Genuine 3D modes. Finally we consider the genuine 3D modes from Table 6.2. We found four different solutions S6–S9, but all of them are unstable in the parameter range studied. Table 6.4 lists the initial guesses used and points to the corresponding figures in the appendix. According to Table 6.2 we know that the zero solution is unstable with respect to modes S7-S9. The eigenvalue has a high geometric multiplicity: we computed it numerically and found 24 equal positive eigenvalues. To single out the stable branch originating from the bifurcation is therefore not trivial. To do so one should exploit the symmetry of the problem as indicated in the paragraph on symmetries in Section 6.1.

The results presented above agree with the observation by De Wit *et al.* [137], who studied the well known Brusselator reaction-diffusion model in 3D and demonstrated possible symmetry structures with high dimension, i.e. body centered cubic (BCC), hexagonally packed cylinders (HPC), and also lamellae structures. Our results provide richer information on the pattern formation and change depending on different parameters. In [70] and [136] model equation 6.2 introduced by Barrio *et al.* is investigated. Their results agree with ours as well. The HPC and lamellae patterns are generalizations of 2D solutions and have the same bifurcation diagram as in 2D.

6.3.4 Performance studies

In this section we want to give an impression on the effectiveness of the continuation process, and in particular the linear and eigenvalue solvers, which constitute most of the runtime in 3D. We use the number of sparse matrix-vector products (matvecs) as a rough indicator of the cost of our solvers. Equally roughly one could say that one time step with Euler's method, used by many other authors, would cost about one matvec per time step. We note that the number of time steps

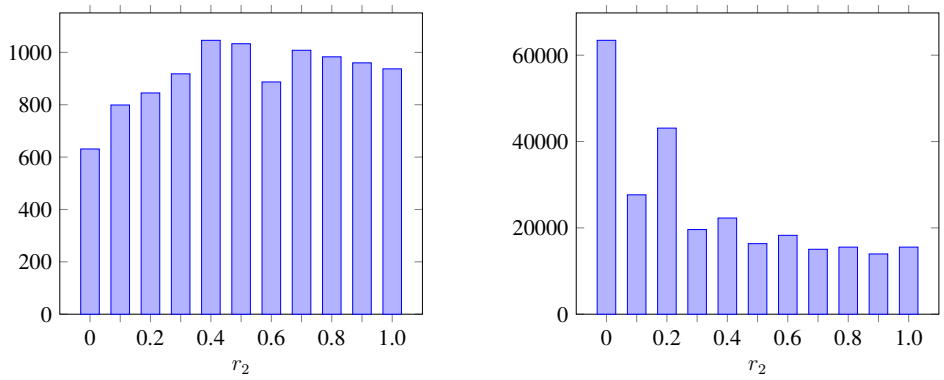


Figure 6.8: Number of matvecs with preconditioner ML for computing a sequence of steady states (left) and their associated 10 right-most eigenpairs (right) on branch S3 with parameter r_2 varying from 0.0 to 1.0 on a 128^3 grid.

reported in papers such as [70, 73] is on the order of hundred thousands or even millions for reaching a single steady state for this kind of problem.

In Fig. 6.8, the number of matvecs is shown for the solution of the nonlinear system (left) and the eigenvalue computation (right) at the different parameters. The number of Newton steps per parameter value is typically 3 or 4, and an adaptive tolerance of the inner GMRES solver is used to save some iterations. The GMRES method is restarted after 50 iterations to save memory and orthogonalization time. In the Jacobi-Davidson eigensolver we use a block size of 4 and allow at most 25 inner GMRES iterations to achieve an adaptively computed tolerance for the correction equation. Default settings are used for the smoothed aggregation (SA) AMG preconditioner ML from Trilinos version 11.12.1.

On the left one sees that the number of matvecs increases until the bifurcation point $r_2 = 0.25$ where the branch becomes unstable. After that the number of iterations decreases slightly, possibly indicating that we are getting away from the singularity. In the right panel, note that the first eigenvalue computation is particularly expensive. This is because no approximate eigenspace is available for starting the block JDQR method. Hence it also shows that reusing the space from the previous step is advantageous, because in the end it reduces the amount of work by a factor 3. Comparing the left and right panel in Fig. 6.8 one observes that eventually the cost for the linear stability analysis (i.e. the eigenvalue computation) is about twenty times that of the actual solution of the non-linear problem. Of course this can be reduced by requiring fewer eigenvalues. Moreover, one could make the amount dependent on the situation, e.g. all the eigenvalues that

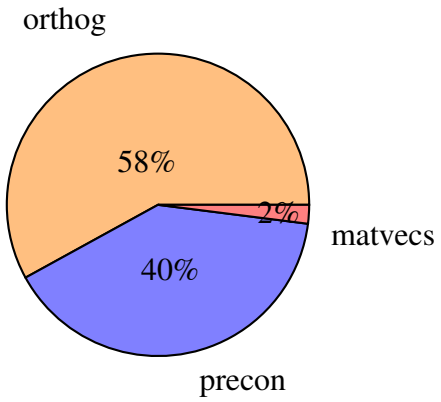


Figure 6.9: The runtime percentage of the three main operations for computing a sequence of steady states on branch S3 with parameter r_2 varying from 0.0 to 1.0 on a 128^3 grid.

are within a certain distance from the imaginary axis.

In order to give an impression of the performance of the linear solver, we followed a branch of steady states on a 128^3 grid, running on the 64 cores of an Intel Xeon Phi 7210 ("Knight's Landing") many-core processor (core frequency 1.3 GHz and configured in cluster/cache mode). In Fig. 6.9, we give a breakdown of the actual runtime. A thorough performance analysis is not the goal here as all of the building blocks are freely available software and their performance has been studied elsewhere. The average time required for solving one linear system is 36 seconds. We display the runtime percentage of the three most expensive operations: orthogonalization of the subspace in GMRES(50) (orthog), preconditioner applications (precon) and matrix-vector products (matvecs). Most of the time is spent in orthogonalization. This can be reduced by decreasing the restart parameter m in GMRES(m) at the cost of more iterations and hence more matvecs and preconditioner applications. The matrix-vector product can be executed very efficiently in parallel here because of the simple matrix structure and therefore only has a minor contribution to the overall runtime.

Figure 6.10 shows the effect of using the ML preconditioning in the Jacobi-Davidson eigensolver. We see a significant decrease of matvecs for large grid sizes. Note that the Jacobi-Davidson method performs some preconditioning of the equations solved by projecting out approximate and converged eigenmodes even if only GMRES is used as a correction solver. Therefore we expect the gap between the unpreconditioned and preconditioned solver to be even larger when

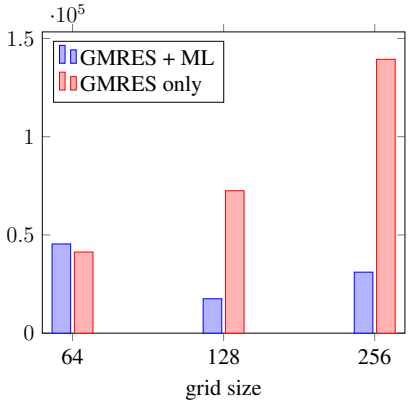


Figure 6.10: Number of matvecs in an eigenvalue computation with and without preconditioning for different grid sizes.

computing the steady states on the branch. The actual time spent solving both linear and eigenvalue problems depends on the balance between orthogonalization and preconditioner applications and can be optimized by tuning solver settings in production runs.

Instead of ML we also employed HYMLS to compare the two preconditioners. The behaviour of the linear solver with HYMLS as preconditioner during continuation process is quite robust and and shows grid independent convergence. Following the branch S3 with parameter r_2 from 0.0 to 1.0, Fig. 6.11 shows that the average number of iterations in the Newton process is becoming smaller and smaller with grid refinement. The reason is that with grid refinement the problem gets closer and closer to Laplace's equation, which is easier to solve. In addition, HYMLS shows its benefit over ML saving about 30% of the operations with respective to total matrix-vector products.

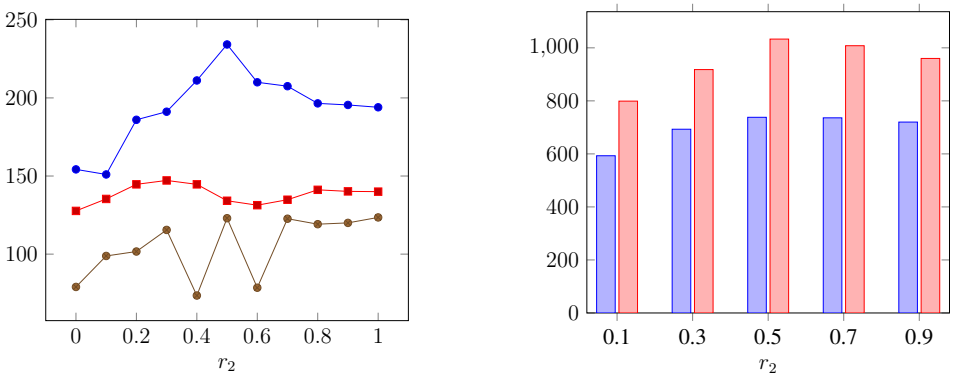


Figure 6.11: Average number of iterations in continuation process following branch S3 with parameter r_2 varying from 0.0 to 1.0 with HYMLS using a fixed number of levels (3); results are obtained on different grids, from top to bottom: 32^3, 64^3 and 128^3, respectively (left); comparison between HYMLS (blue) and ML (red) w.r.t. total matrix-vector products (right).

6.4 Summary and discussion

We have shown that numerical continuation techniques, combined with efficient multigrid and Jacobi-Davidson solvers, are a very effective way to analyze nonlinear PDEs describing reaction-diffusion processes. Compared to previous results on this type of problem, we presented richer bifurcation diagrams in 2D and 3D with higher spatial resolution. In conclusion, the continuation approach is not only efficient in solving the steady solution of large dynamical systems but also can give more insight about what is happening at the critical points where bifurcations may occur. With this method, fruitful and insightful numerical results such as pattern selections as well as bifurcation diagrams can be computed, which definitely will pave the way to a deeper understanding of the experimental phenomenon.

The numerical performance of the overall approach was demonstrated by reporting the number of operations required and giving an indication of the run time. We note that much optimization can be done to solve such problems more efficiently: geometric multigrid and matrix-free methods for structured grids, hybrid parallelization and SIMD usage (see [99]), exploiting the many symmetries in 3D, etc. In 3D we did not investigate the complete bifurcation structure as systematically as in 2D. Instead we demonstrated the feasibility of such an investigation and leave it to domain scientists with a concrete application in e.g. computational biology or chemistry.

Chapter 7

Conclusions and outlook

In this chapter, we will look back to the work done in this thesis in order to draw conclusions and to distill subjects for further research.

7.1 Conclusions

In Section 1.5, we set out the research questions for this thesis. Here we want to assess how well these have been answered by the results in this thesis.

The first question was: *Is the continuation approach a viable alternative to time integration approaches?*

The numerical experiments on a variety of multi-physics problems show that continuation approach is a practical and efficient way to solve series of steady states as a function of parameters and to do bifurcation analysis. Starting with a proper initial guess, Newton's method converges in a few steps. Since solving the linear systems arising from the discretization takes most of the computational work, efficiency is determined by how fast the linear systems can be solved. For the lid-driven cavity we have seen in section 4.5 that comparable turnaround times were obtained by a time integration method using the pressure correction approach solving a Poisson equation for the pressure and a convection-diffusion equation for the velocities at Re=500.

The second question was: *Can we compute solutions by the continuation approach, which are hard to get by the time-integration approach?* In Fig. 3.6, we showed to be able to perform continuation of unstable steady solutions, which are impossible to get by only using a time integration approach. Stable and unstable modes give interesting insight in the dynamics of solution.

The third question was: *How does HYMLS compare to other solvers, like "physics-based" preconditioners in Trilinos package Teko, and ML in terms of robustness and turnaround time?* Compared with other preconditioners, such as Teko and ML, HYMLS can compute solutions at higher Reynolds numbers. We saw in table 4.5 that, for Re=500 in the lid-driven cavity problem, Teko with LSC preconditioner did not converge. Here, ML is used for the subblocks in the LSC preconditioner. We have also shown in table 4.2 that Picard iteration does not converge at Re=500. In that case an Oseen equation needs to be solved, the matrix of which is only a part of the Jacobian. It is clear that one needs more of the Jacobian to get convergence. HYMLS can deal with the full Jacobian. Despite the fact that for the Poisson equation HYMLS shows a number of iterations that increases with the logarithm of the total of unknowns in section 2.3.4 , this is not reflected in the Stokes and Navier-Stokes solution results in tables 4.3 and 4.4. Though the number of iterations is still acceptable, one needs to look into it further to get the desired property of near grid independent convergence for flow problems. In fact, there is a lot of room for improvement of the method. The main thing shown here is its robustness, which is due to the iteration in the divergence-free space.

To test the efficiency of linear solvers for non-flow problems, we studied a well-known reaction-diffusion system, i.e., the BVAM model of the Turing problem. HYMLS still performs well with respect to ML in the continuation process in the Newton process as showed in fig. 6.11. Compared to ML, it needs fewer iterations in GMRES. However, it did not perform very well in the eigenvalue computation for solving the correction equation in JDQR. When solving the correction equation, the linear iteration process may stall or diverge. The reason for this is unclear at this moment and will be left for future research. A difficulty in the eigenvalue problems appears to be the high multiplicity of some of the eigenvalues, sometimes up to 20.

There is a limitation on HYMLS that should be mentioned. It has been constructed for (Navier-)Stokes equations which have a discretization with only two nonzeros in each row of that part of the matrix representing the gradient. Moreover the discretization of the divergence should be the exact transpose of the gradient.

Since HYMLS is based on data structures from the Epetra package, it will immediately take advantage of any improvements in this package. One could also easily switch to the Tepetra package, which contains complex arithmetic. This arithmetic will come in handy for eigenvalue computations.

In general we find that thanks to the highly parallel computation, our program can deal with very fine grids giving insightful knowledge about the flow struc-

ture topology, transition to unsteadiness as well as flow characteristics beyond the transition. Especially three-dimensional results, which are difficult to capture at high Reynolds and Rayleigh numbers can be obtained. In addition, the application to the Turing system not only proved our program's ability in doing nonlinear bifurcation analysis efficiently but also provided insightful information on pattern formation, which plays an essential role in chemistry and biology.

7.2 Ideas for future research

In future we like to add more classical problems to the ones mentioned in chapter 3. One important and promising application of our program is to simulate the ocean circulation on the surface of the earth. A model of this Atlantic ocean would be a stretched box, which is very thin compared to the length of earth pole to the equator. The model should be driven by gradients in surface temperature and salinity, and by wind. Another interesting one would be a Taylor-Couette flow.

Also our techniques can be improved. For instance, use the eigenspace coming about from the eigenvalue computation in a deflation process to speed up the linear solves. The other way around, one could also use part of the Krylov subspace of the linear solver to start the eigenvalue solver process. In the Jacobi-Davidson method, we can already already import a guess for the basis and it appears to be fruitful to use the eigenspace computed at the previous continuation step. It is tempting to try to combine the eigenvalue computation and the solution of the linear system and have only one subspace to solve both problems. These can be viewed as subspace accelerated inexact Newton methods, see [138].

The number of iterations in HYMLS can be brought down by making the factorization more accurate. Currently, a lot of entries have been dropped. Here, a trade-off between iteration costs and accuracy costs should be pursued. Experience with MRILU [139] showed that the choice is not very critical and that it is possible to increase the accuracy till only one iteration is needed to get the desired reduction in the error. Also solving slightly indefinite equations occurring in eigenvalue problems needs some study to improve HYMLS' behavior on.

The restriction to a C-grid discretization of the Navier-Stokes equations can also be alleviated. In principle it can also work for so-called A- and B-grids; in the former we have a collocated grid and in the latter the velocities are collocated). So far the grids have been structured, but it would be interesting to extend it to finite volume discretizations on unstructured meshes.

In this thesis, we did not consider the stability of periodic solutions that may occur after an Hopf-bifurcation. One could extend the technology by adding continuation of periodic solutions. At the Hopf-bifurcation point the frequency of such a solution is known, hence, it is tempting to build a special purpose time integrator which is accurate near this frequency instead of being that at frequency zero, the latter is the standard case.

Another addition to the techniques will be solvers for flow problems with noise. Especially, in the neighborhood of bifurcation points this can be tricky. In the end, we would like to be able to compute transition times and probabilities for switching from one stable state into another. For instance, these are relevant in estimating the chance on getting a rapid climate change.

Appendices

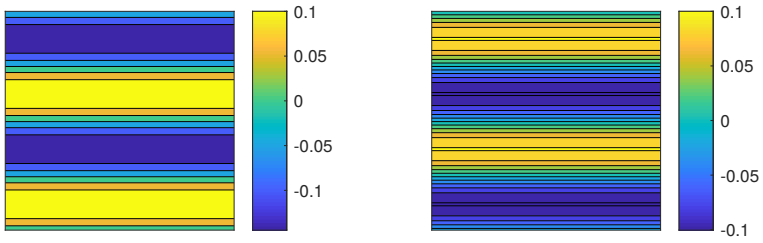
Appendix A

Solutions of BVAM model

In the appendices we present some of the solution patterns found in 2D and 3D. In all of the plots, the left column depicts the u-variable and the right column the v-variable. All results were obtained using a spatial resolution of 128 grid points per direction.

A.1 Patterns of 2D Solutions

Figure A.1: S2 unstable stripe solution at r_2=0.0.



A.2 3D solutions

Figure A.2: Two independent eigenvectors corresponding to the positive eigenvalue of branch S2 with a pattern of an S4 mode, with $r_2 = 0.0$.

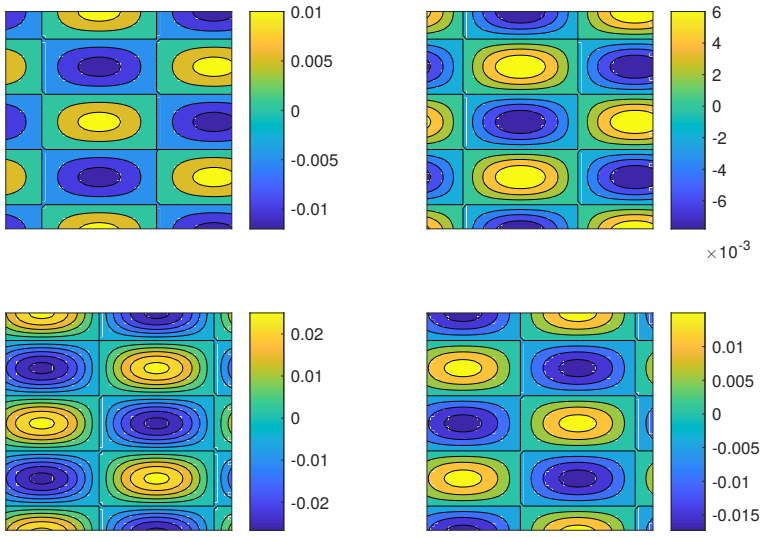


Figure A.3: S5 stable mixed stripe and spot solution at r_2=0.0, 0.1, 0.2, 0.21227 from top to bottom.

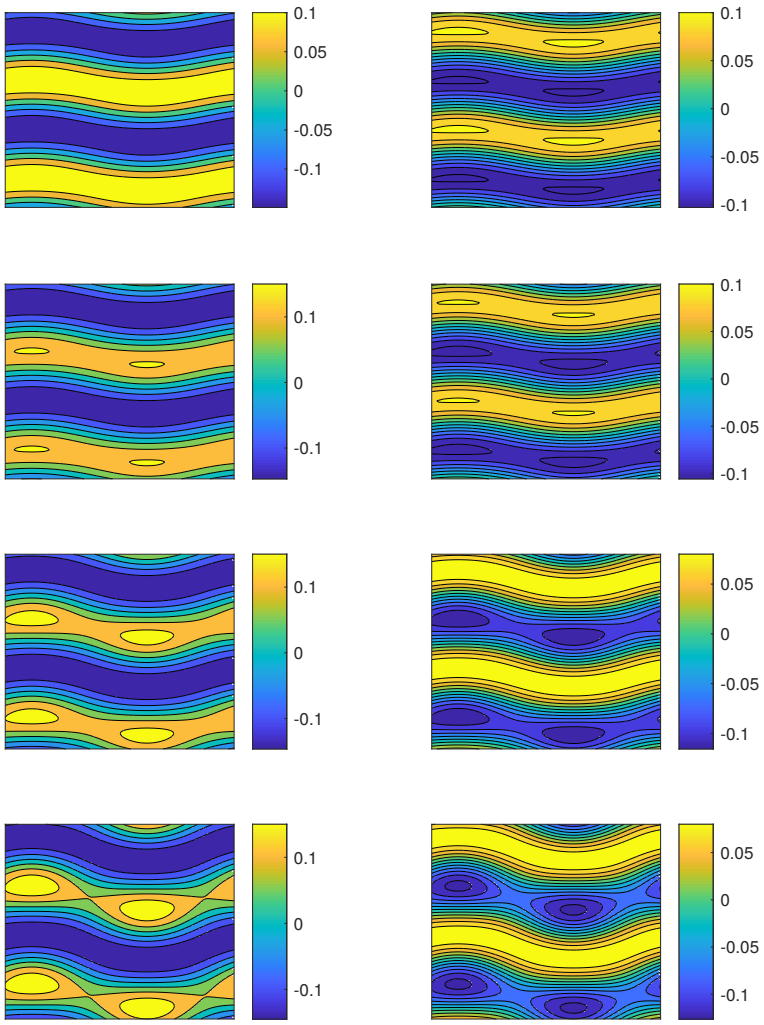


Figure A.4: S5 after turning point: unstable mixed stripe and spot solution at r_2=0.21, 0.19, 0.15, 0.129 from top to bottom.

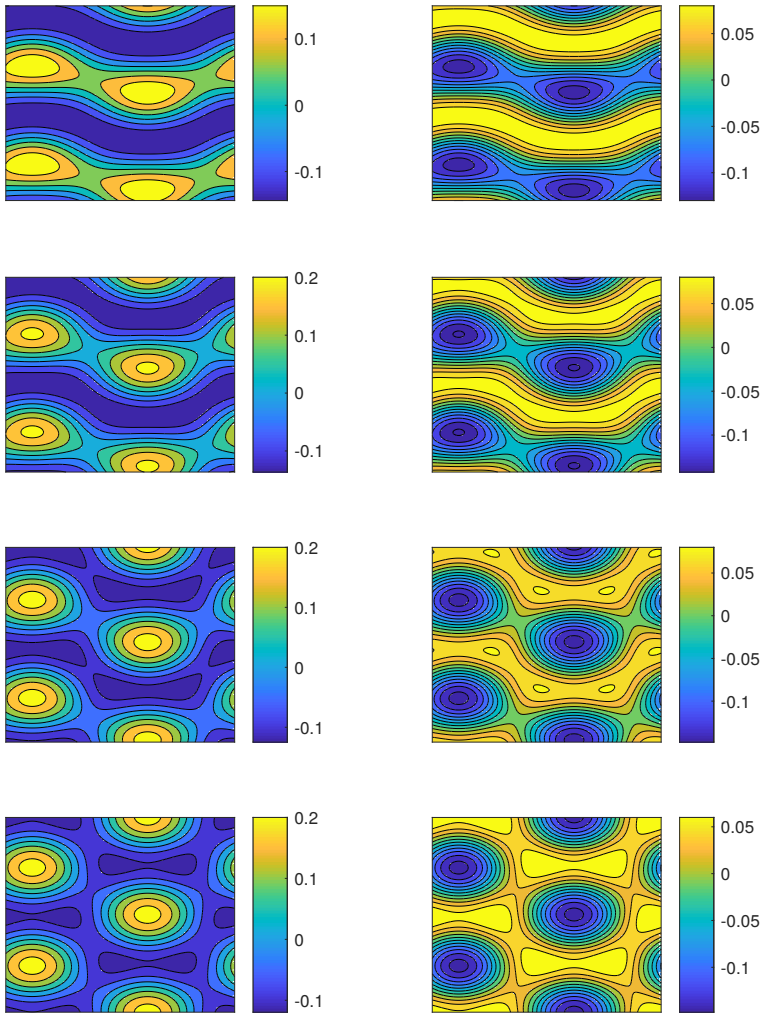


Figure A.5: S4 spot solution becoming stable (shown at r_2=0.0, 0.2, 0.5 from top to bottom).

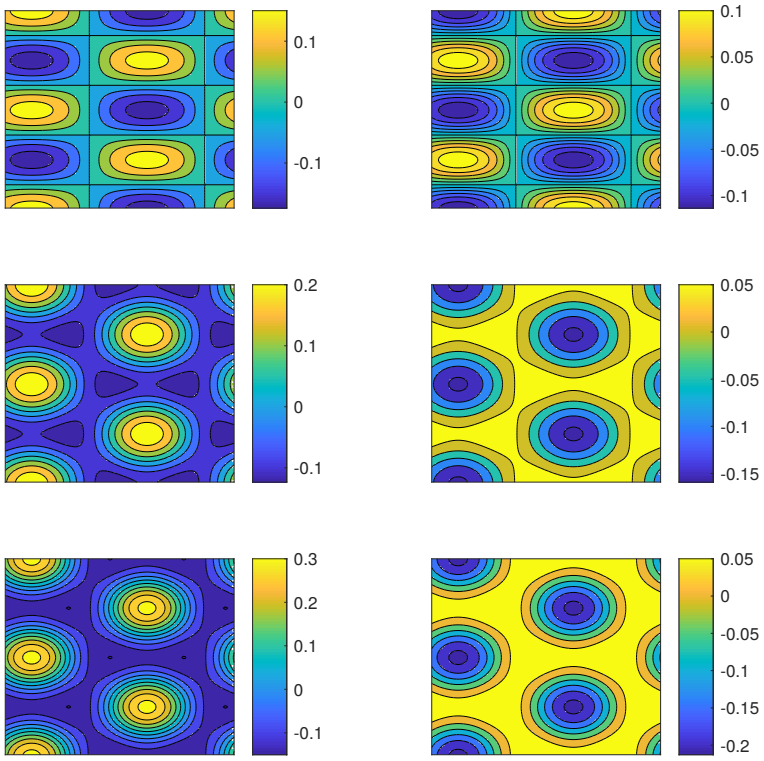


Figure A.6: S3 stable stripe solution at r_2=0.0 that will eventually lose stability.

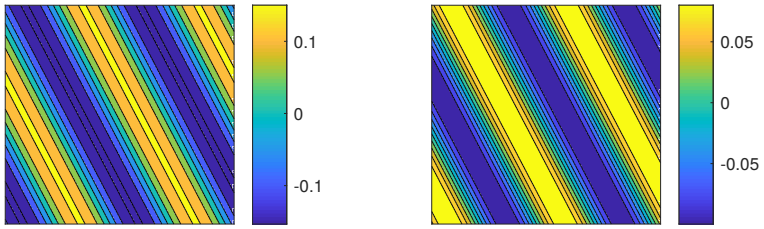


Figure A.7: Two independent eigenvectors corresponding to the positive eigenvalue of branch S3, $r_2 = 0.25$.

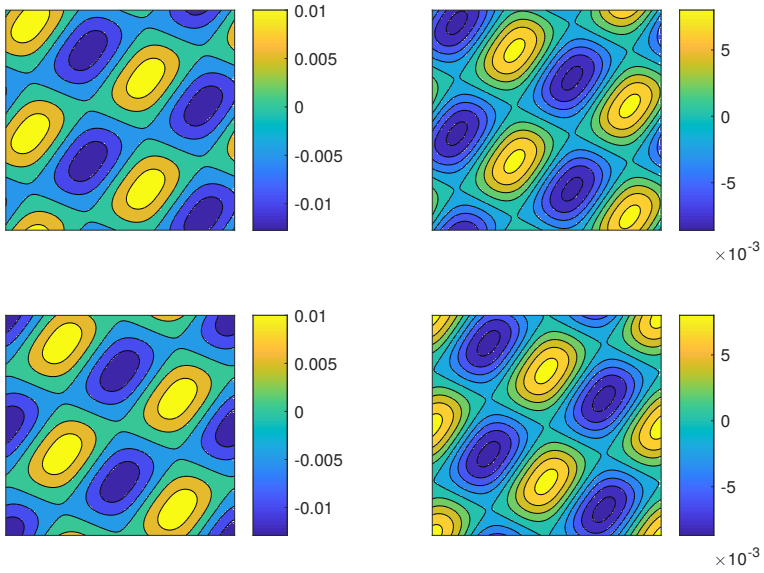


Figure A.8: Isosurface of 3D solution of pattern S6, with r_2=0.0, 0.5, 1.0 from top to bottom.

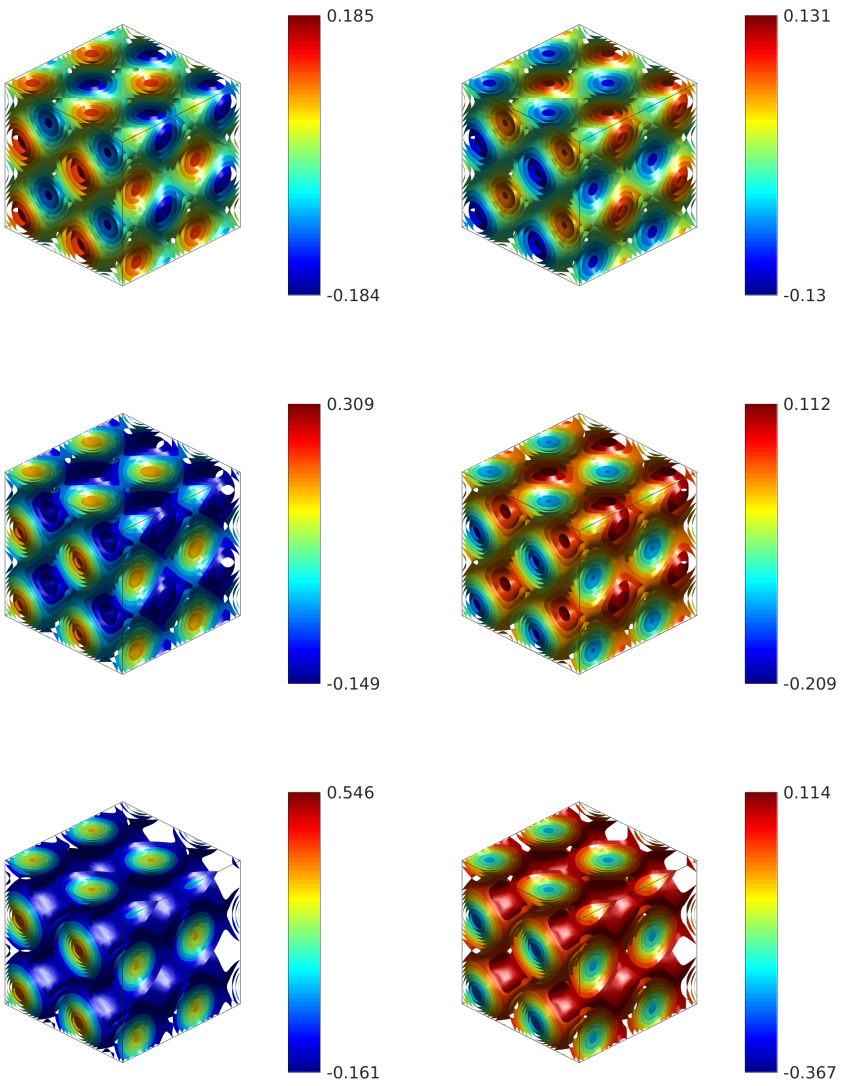


Figure A.9: Isosurface of 3D solution of pattern S7, with r_2=0.0, 0.5, 1.0 from top to bottom.

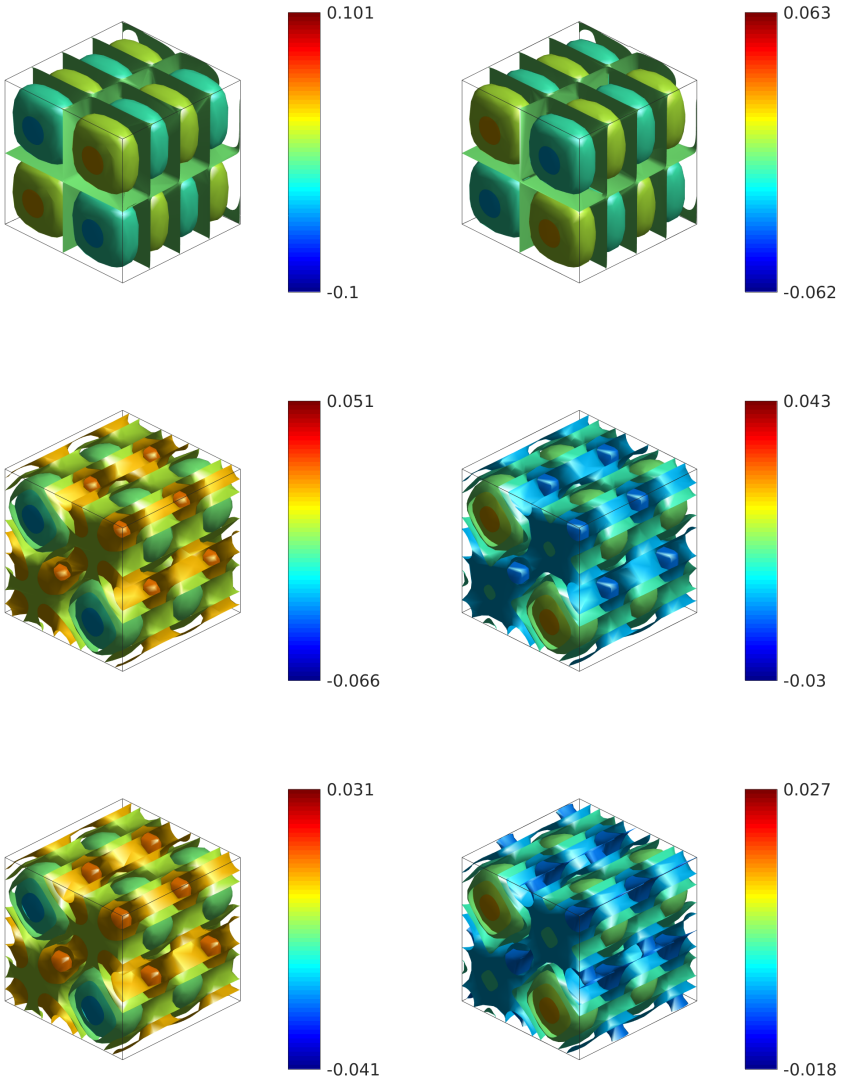


Figure A.10: Isosurface of 3D solution of pattern S8, with r_2=0.0.

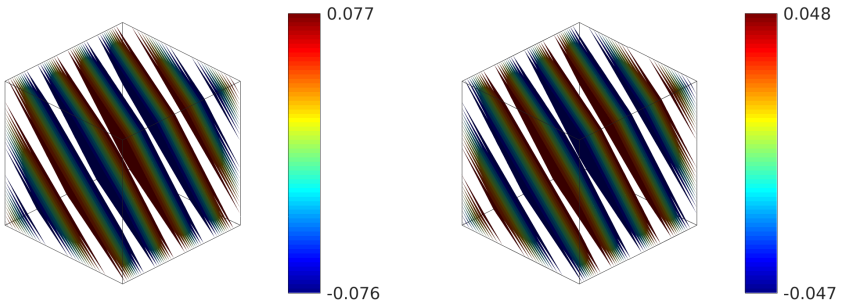
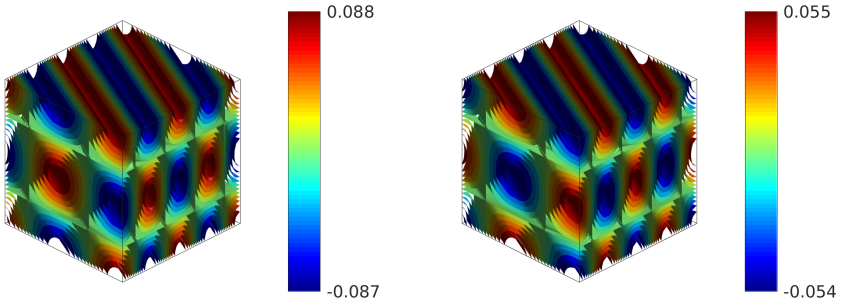


Figure A.11: Isosurface of 3D solution of pattern S9, with r_2=0.0.



Bibliography

[1] C. Foias, O. Manley, R. Rosa, and R. Temam. *Navier-Stokes Equations and Turbulence*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, 2001.

[2] J. B. Vos, A. Rizzi, D. Darracq, and E. H. Hirschel. Navier-Stokes solvers in European aircraft design. *Progress in Aerospace Sciences*, 38(8):601 – 697, 2002.

[3] S. P. Sutera. The history of Poiseuille flow. *Annu. Rev. Fluid Mech.*, 25:1–19, 1993.

[4] J. Hwang and K. Yang. Numerical study of Taylor–Couette flow with an axial flow. *Computers and Fluids*, 33(1):97 – 118, 2004.

[5] D. J. Tritton. *Physical Fluid Dynamics, 2nd ed.*, pages 52–53 and 59–60. Oxford, England: Clarendon Press, 1988.

[6] T. E. Faber. *Fluid Dynamics for Physicists*, pages 31–62. Cambridge University Press, New York, 1995.

[7] L. Quartapelle. *Numerical solution of the incompressible Navier–Stokes equations*, volume 113, pages 2–10. Birkhäuser-Verlag, Basel, 1993.

[8] Y. Wang. *Solving incompressible Navier-Stokes equations on heterogeneous parallel architectures*, pages 14–15. Ph.D. Dissertation, University Paris Sud-Paris XI, 2015.

[9] A. Y. Gelfgat. Linear instability of the lid-driven flow in a cubic cavity. *ArXiv e-prints*, April 2017.

[10] F. Gómez, R. Gómez, and V. Theofilis. On three-dimensional global linear instability analysis of flows with standard aerodynamics codes. *Aerospace Science and Technology*, 32(1):223 – 234, 2014.

[11] J-Ch Loiseau, J-Ch Robinet, and E. Leriche. Intermittency and transition to chaos in the cubical lid-driven cavity flow. *Fluid Dynamics Research*, 48(6):061421, 2016.

[12] G. M. Shroff and H. B. Keller. Stabilization of unstable procedures: The recursive projection method. *SIAM Journal on Numerical Analysis*, 30(4):1099–1120, 1993.

[13] K. Lust and D. Roose. *Computation and Bifurcation Analysis of Periodic Solutions of Large-Scale Systems*, pages 265–301. Springer New York, New York, NY, 2000.

[14] G. Tiesinga, F.W. Wubs, and A.E.P. Veldman. Bifurcation analysis of incompressible flow in a driven cavity by the Newton–Picard method. *Journal of Computational and Applied Mathematics*, 140(1):751 – 772, 2002. Int. Congress on Computational and Applied Mathematics 2000.

[15] L. S. Tuckerman. Divergence-free velocity fields in nonperiodic geometries. *Journal of Computational Physics*, 80(2):403 – 441, 1989.

[16] L. S. Tuckerman and D. Barkley. *Bifurcation Analysis for Timesteppers*. Springer New York, New York, NY, 2000.

[17] K. Borońska and L. S. Tuckerman. Extreme multiplicity in cylindrical Rayleigh-Bénard convection. i. time dependence and oscillations. *Phys. Rev. E*, 81:036320, Mar 2010.

[18] K. Borońska and L. S. Tuckerman. Extreme multiplicity in cylindrical Rayleigh-Bénard convection. ii. bifurcation diagram and symmetry classification. *Phys. Rev. E*, 81:036321, Mar 2010.

[19] F. Garcia, M. Net, and J. Sánchez. A comparison of high-order time integrators for highly supercritical thermal convection in rotating spherical shells. In Mejdi Azaïez, Henda El Fekih, and Jan S. Hesthaven, editors, *Spectral and High Order Methods for Partial Differential Equations - ICOSAHOM 2012*, pages 273–284, Cham, 2014. Springer International Publishing.

[20] V. Citro, P. Luchini, F. Giannetti, and F. Auteri. Efficient stabilization and acceleration of numerical simulation of fluid flows by residual recombination. *Journal of Computational Physics*, 344, 05 2017.

[21] H.A. Dijkstra, F.W. Wubs, A.K. Cliffe, E. Doedel, I.F. Dragomirescu, B. Eckhardt, A.Y. Gelfgat, A. Hazel, V. Lucarini, A.G. Salinger, E.T. Phipps, J. Sanchez-Umbria, H. Schuttelaars, L. Tuckerman, and U. Thiele. Numerical bifurcation methods and their application to fluid dynamics: Analysis beyond simulation. *Communications in Computational Physics*, 15(1):2–38, 2014.

[22] H. B. Keller. Numerical solution of bifurcation and nonlinear eigenvalue problems. *Applications of bifurcation theory*, 1(38):359–384, 1977.

[23] M. A. Crisfield. *Nonlinear Finite Element Analysis of Solids and Structures, Vol 1: Basic Concepts*, chapter 1. Wiley, 1991.

[24] J. Thies. *Scalable algorithms for fully implicit ocean models*, chapter 1. Ph.D. Dissertation, University of Groningen, 2011.

[25] F. Charru. *Hydrodynamic Instabilities*. Cambridge University Press, 2011. Cambridge Books Online.

[26] R. Lehoucq, D. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, 1998.

[27] G. L. G. Sleijpen and H. A. Van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 17(2):401–425, 1996.

[28] Y. Saad. *Numerical Methods for Large Eigenvalue Problems, Revised Edition*, volume 66, pages 128–134. Siam, 2011.

[29] K. Meerbergen and A. Spence. Shift-and-invert iteration for purely imaginary eigenvalues with application to the detection of Hopf bifurcations in large scale problems. *SIAM J. Matrix Anal. Appl.*, 31:1463–1482, 01 2010.

[30] K. Meerbergen. An implicitly restarted rational Krylov strategy for Lyapunov inverse iteration. *IMA Journal of Numerical Analysis*, 36(2):655–674, 2016.

[31] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numer.*, 14:1–137, 2005.

[32] H.A. van der Vorst. *Iterative Krylov methods for large linear systems, volume 13 of Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, 2003.

[33] M. Benzi and M. A. Olshanskii. An augmented Lagrangian-based approach to the Oseen problem. *SIAM J. Scientific Computing*, 28:2095–2113, 2006.

[34] A. C. De Niet and F. W. Wubs. Two saddle point preconditioners for fluid flows. *Int. J. Numer. Methods Fluids*, 54:355–377, 2007.

[35] H. C. Elman, D. Silvester, and A. J. Wathen. Performance and analysis of saddle point preconditioners for the discrete steady-state Navier-Stokes equations. *Numerische Mathematik*, 90(4):665–688, Feb 2002.

[36] D. Kay, D. Loghin, and A. Wathen. A preconditioner for the steady-state Navier-Stokes equations. *SIAM Journal on Scientific Computing*, 24(1):237–256, 2002.

[37] H. Elman, V.E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 227(3):1790 – 1808, 2008.

[38] A. Segal, M. Rehman, and C. Vuik. Preconditioners for incompressible Navier-Stokes solvers. *Numerical Mathematics-theory Methods and Applications*, 3, 08 2010.

[39] E. C. Cyr, J. N. Shadid, and R. S. Tuminaro. Stabilization and scalable block preconditioning for the Navier-Stokes equations. *Journal of Computational Physics*, 231(2):345 – 363, 2012.

[40] J. Ruge and K. Stüben. *Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG)*. Arbeitspapiere der GMD. GMD, 1984.

[41] K. Stüben. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128(1):281 – 309, 2001. Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

[42] M. W. Gee, C. M. Siefert, J. J. Hu, R. S. Tuminaro, and M. G. Sala. ML 5.0 smoothed aggregation user's guide. Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

[43] P. Vanek, M. Brezina, and J. Mandel. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik*, 88(3):559–579, May 2001.

[44] P. Hénon and Y. Saad. A parallel multistage ILU factorization based on a hierarchical graph decomposition. *SIAM Journal on Scientific Computing*, 28(6):2266–2293, 2006.

[45] J. Gaidamour, P. Hénon, J. Roman, and Y. Saad. An hybrid direct-iterative solver based on the Schur complement approach. In *8th Workshop of the ERCIM Working group*, Salerne, Italy, September 2006.

[46] J. Gaidamour and P. Hénon. A parallel direct/iterative solver based on a Schur complement approach. In *2008 11th IEEE International Conference on Computational Science and Engineering*, pages 98–105, July 2008.

[47] J. Thies and F. W. Wubs. Design of a parallel hybrid direct/iterative solver for CFD problems. In *Proceedings of the 2011 IEEE Seventh International Conference on eScience*, ESCIENCE '11, pages 387–394, Washington, DC, USA, 2011. IEEE Computer Society.

[48] W. Song, F. W. Wubs, and J. Thies. A highly parallel code for strongly coupled fluid-transport equations. In *Proceedings of the 11th world congress on computational mechanics*, WCCM XI, pages 199–210. Barcelona, Spain, 2014.

[49] U. Ghia, K. N. Ghia, and C. T. Shin. High-resolutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48(3):387 – 411, 1982.

[50] S. Albensoeder and H. C. Kuhlmann. Accurate three-dimensional lid-driven cavity flow. *Journal of Computational Physics*, 206(2):536 – 558, 2005.

[51] H. C. Kuhlmann and S. Albensoeder. Stability of the steady three-dimensional lid-driven flow in a cube and the supercritical flow dynamics. *Physics of Fluids*, 26, 2014.

[52] N. C. Markatos and K. A. Pericleous. Laminar and turbulent natural convection in an enclosed cavity. *International Journal of Heat and Mass Transfer*, 27(5):755 – 772, 1984.

[53] A. C. Perkins. *Mechanisms of instability in Rayleigh-Bénard convection*. Phd thesis no. 20599, Georgia Institute of Technology, 2011.

[54] D. Puigjaner, J. Herrero, F. Giralt, and C. Simó. Stability analysis of the flow in a cubical cavity heated from below. *Physics of Fluids*, 16:3639–3655, 2004.

[55] D. Puigjaner, J. Herrero, C. Simó, and F. Giralt. Bifurcation analysis of steady Rayleigh–Bénard convection in a cubical cavity with conducting sidewalls. *Journal of Fluid Mechanics*, 598:393–427, 3 2008.

[56] J. Yu, A. Goldfaden, M. Flagstad, and J. D. Scheel. Onset of Rayleigh-Bénard convection for intermediate aspect ratio cylindrical containers. *Physics of Fluids*, 29(2):24–107, 2017.

[57] D. R. Chenoweth and S. Paolucci. Gas flow in vertical slots with large horizontal temperature difference. *Phys. Fluids*, 28:2365–2374, 1985.

[58] S. Paolucci. The differentially heated cavity. *Sadhana*, 19(5):619–647, 1994.

[59] A. Y. Gelfgat. Time-dependent modeling of oscillatory instability of three-dimensional natural convection of air in a laterally heated cubic box. *Theoretical and Computational Fluid Dynamics*, 31(4):447–469, Aug 2017.

[60] S. Xin and P. L. Quéré. An extended Chebyshev pseudo-spectral benchmark for the 8:1 differentially heated cavity. *International Journal for Numerical Methods in Fluids*, 40(8):981–998, 2002.

[61] M. F. Baig and A. Masood. Natural convection in a two-dimensional differentially heated square enclosure undergoing rotation. *Numerical Heat Transfer, Part A: Applications*, 40(2):181–202, 2001.

[62] M. Narendra Kumar, G. Pundarika, K. Rama Narasimha, and K. N. Seetharamu. Effect of rotation on natural convection in differentially heated rotating enclosure by numerical simulation. *Journal of Applied Fluid Mechanics*, 9(3):1265–1272, 2016.

[63] A. Medelfef, D. Henry, A. Bouabdallah, R. Boussaa, S.Kaddeche, and V. Botton. A three dimensional numerical study of rotating buoyant convection in a side heated cavity. In 23^th *Congrès Français de Mécanique*, 2017.

[64] A. M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 237(641):37–72, 1952.

[65] A. M. Zhabotinsky L. Yang, M. Dolnik and I. R. Epstein. Turing patterns beyond hexagons and stripes. *Chaos*, 16(3), 2006.

[66] J. Boissonade, E. Dulos, and P. De Kepper. *Turing Patterns: From Myth to Reality*, pages 221–268. Springer Netherlands, Dordrecht, 1995.

[67] E. Dulos, J. Boissonade, J. J. Perraud, B. Rudovics, and P. De Kepper. Chemical morphogenesis: Turing patterns in an experimental chemical system. *Acta Biotheor.*, 44(3-4):249–61, 1996.

[68] P. K. Maini, K. J. Painter, and H. Nguyen Phong Chau. Spatial pattern formation in chemical and biological systems. *J. Chem. Soc., Faraday Trans*, 93:3601–3610, 1997.

[69] T. Bánsági, V. K. Vanag, and I. R. Epstein. Tomography of reaction-diffusion microemulsions reveals three-dimensional Turing patterns. *Science*, 331(6022):1309–1312, 2011.

[70] T. Leppänen, M. Karttunen, K. Kaski, R. A. Barrio, and L. Zhang. A new dimension to Turing patterns. *Physica D Nonlinear Phenomena*, 168:35–44, August 2002.

[71] F. Molnár, F. Izsák, R. Mészáros, and I. Lagzi. Simulation of reaction-diffusion processes in three dimensions using CUDA. *Chemometrics and Intelligent Laboratory Systems*, 108(1):76–85, 2011. cited By 10.

[72] H. Shoji and T. Ohta. Computer simulations of three-dimensional Turing patterns in the Lengyel-Epstein model. *Phys. Rev. E (3)*, 91(3):032913, 11, 2015.

[73] R. A. Barrio, C. Varea, J. L. Aragón, and P. K. Maini. A two-dimensional numerical study of spatial pattern formation in interacting Turing systems. *Bulletin of Mathematical Biology*, 61(3):483 – 505, 1999.

[74] N. McCullen and T. Wagenknecht. Pattern formation on networks: from localised activity to Turing patterns. *Scientific Reports*, 6(27397), 2016.

[75] The PHIST software repository. `https://bitbucket.org/essex/phist/`. Accessed 2017-09-13.

[76] R. Seydel. *Practical Bifurcation and Stability Analysis*, pages 38–48. Springer, 2010.

[77] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-preserving discretization of turbulent flow. *J. Comput. Phys.*, 187(1):343–368, May 2003.

[78] M. Tuma. A note on the LDLT decomposition of matrices from saddle-point problems. *SIAM J. Matrix Anal. Appl.*, 23(4):903–915, April 2001.

[79] P. B. Bochev, C. J. Garasi, J. J. Hu, A. C. Robinson, and R. S. Tuminaro. An improved algebraic multigrid method for solving Maxwell's equations. *SIAM Journal on Scientific Computing*, 25(2):623–642, 2003.

[80] A. C. De Niet and F. W. Wubs. Numerically stable LDL^T-factorization of F-type saddle point matrices. *IMA Journal of Numerical Analysis*, 29:208–234, 2009.

[81] F. W. Wubs and J. Thies. A robust two-level incomplete factorization for (Navier-) Stokes saddle point matrices. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1475–1499, 2011.

[82] J. Thies and F. W. Wubs. A robust parallel ILU solver with grid-independent convergence for the coupled steady incompressible Navier-Stokes equations. In *Proceedings of ECCOMAS CFD*, ESCIENCE '11, pages CD–ROM paper 1421. J.C.F. Pereira and A. Sequeira (eds.), 2010.

[83] M. van der Klok. *Skew partitioning for the hybrid multilevel solver*, pages 19–22. Bachelor thesis, University of Groningen, 2017.

[84] G. Sleijpen, H. van der Vorst, A. Ruhe, Z. Bai, T. Ericsson, T. Kowalski, B. Kågström, and R. Li. *Templates for the Solution of Algebraic Eigenvalue Problems*, chapter 8. Generalized Non-Hermitian Eigenvalue Problems, pages 233–279. SIAM, 2000.

[85] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9(1):17–29, 1951.

[86] C. G. Baker, U. L. Hetmaniuk, R. B. Lehoucq, and H. K. Thornquist. Anasazi software for the numerical solution of large-scale eigenvalue problems. *ACM Trans. Math. Softw.*, 36(3):13:1–13:23, July 2009.

[87] G. W. Stewart. A Krylov–Schur algorithm for large eigenproblems. *SIAM J. Matrix Anal. Appl.*, 23(3):601–614, March 2001.

[88] B. N. Parlett and J. Le. *QR; Its Forward Instability and Failure to Converge*, pages 177–189. Birkhäuser Basel, Basel, 1991.

[89] M. E. Hochstenbach and Y. Notay. The Jacobi–Davidson method. *GAMM-Mitteilungen*, 29(2):368–382, 2006.

[90] G. L.G. Sleijpen D. R. Fokkema and H. A. Van der Vorst. Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils. *SIAM Journal on Scientific Computing.*, 20(1), 1996.

[91] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[92] V. Faber and T. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.*, 21(2):315–339, 1984.

[93] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, second edition, 2003.

[94] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, NJ, 1980. Reprinted as Classics in Applied Mathematics 20, SIAM, Philadelphia, 1997.

[95] G. L. G. Sleijpen and F. W. Wubs. Exploiting multilevel preconditioning techniques in eigenvalue computations. *SIAM: Journal on Scientific Computing*, 25(4):1249–1272, 2003.

[96] G. L. G. Sleijpen, A. G. L. Booten, D. R. Fokkema, and H. A. van der Vorst. Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems. *BIT Numerical Mathematics*, 36(3):595–633, Sep 1996.

[97] R. Geus. *The Jacobi-Davidson algorithm for solving large sparse symmetric eigen-value problems*. Phd thesis no. 14734, ETH Zurich, 2002.

[98] J. H. Brandts. *Solving Eigenproblems: From Arnoldi via Jacobi-Davidson to the Riccati Method*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[99] M. Röhrig-Zöllner, J. Thies, M. Kreutzer, A. Alvermann, A. Pieper, A. Basermann, G. Hager, G. Wellein, and H. Fehske. Increasing the performance of the Jacobi–Davidson method by blocking. *SIAM Journal on Scientific Computing*, 37(6):C697–C722, 2015.

[100] Y. Feldman and A. Y. Gelfgat. Oscillatory instability of a three-dimensional lid-driven flow in a cube. *Physics of Fluids*, 22(November 2009):1–9, 2010.

[101] A. Liberzon, Y. Feldman, and A. Y. Gelfgat. Experimental observation of the steady-oscillatory transition in a cubic lid-driven cavity. *Physics of Fluids*, 23:1–14, 2011.

[102] A. E. Gill. *Atmosphere-Ocean Dynamics*, chapter 1. Academic Press, London, 1982.

[103] J. Pedlosky. *Geophysical Fluid Dynamics*, chapter 1. Springer-Verlag, New York, 1987.

[104] P. Constantin and C. R. Doering. Infinite Prandtl number convection. *Journal of Statistical Physics*, 94(1):159–172, Jan 1999.

[105] P. Constantin, G. Iyer, and J. Wu. Global regularity for a modified critical dissipative quasi-geostrophic equation. *Indiana University Mathematics Journal*, 57(6):2681–2692, 2008.

[106] H. Bénard. Les tourbillons cellulaires dans une nappe liquide. *Rev. Gén. Sci. pures et appl*, 11:1261–1271, 1900.

[107] M. Van Dyke. *An Album of Fluid Motion*. The Parabolic Press, 1982.

[108] S. Chandrasekhar. *Hydro dynamic and hydromagnetic stability*, chapter 1. Clarendon Press, Oxford, 1961.

[109] M. C. Cross and P. C. Hohenberg. Pattern formation outside of equilibruim. *Rev. Mod. Phys.*, 65:851–1112, 1993.

[110] E. Bodenschatz, W. Pesch, and G. Ahlers. Recent development in Rayleigh-Bénard convection. *Ann. Rev. Fluid Mech.*, 32:708–778, 2000.

[111] I. Catton. The effect of insulating vertical walls on the onset of motion in a fluid heated from below. *International Journal of Heat and Mass Transfer*, 15(4):665–672, 1972.

[112] R. M. Clever and F. H. Busse. Transition to time-dependent convection. *J. Fluid Mech.*, 65(4):625–645, 1974.

[113] R. W. M. Henkes and P. L. Quéré. Three-dimensional transition of natural-convection flows. *J. Fluid Mech.*, 319(218), 1996.

[114] M. Lappa. *Thermogravitational Convection: The Rayleigh–Bénard Problem*, pages 119–130. John Wiley & Sons, Ltd, 2009.

[115] Rayleigh-Bénard instability. `http://hmf.enseeiht.fr/travaux/CD0001/travaux/optmfn/hi/01pa/hyb72`. Accessed 3-April-2018.

[116] D. Puigjaner, J. Herrero, C. Simó, and F. Giralt. From steady solutions to chaotic flows in a Rayleigh-Bénard problem at moderate Rayleigh numbers. *Physica D: Nonlinear Phenomena*, 240(11):920–934, 2011.

[117] Y. T. Ker and T. F. Lin. A combined numerical and experimental study of air convection in a differentially heated rotating cubic cavity. *International Journal of Heat and Mass Transfer*, 39(15):3193–3210, 1996.

[118] T. L. Lee and T. F. Lin. Transient three-dimensional convection of air in a differentially heated rotating cubic cavity. *International Journal of Heat and Mass Transfer*, 39(6):1243–1255, April 1996.

[119] H. C. Ku, R. S. Hirsh, and T. D. Taylor. A pseudospectral method for solution of the three-dimensional incompressible Navier-Stokes equations. *Journal of Computational Physics*, 70(2):439 – 462, 1987.

[120] K. Poochinapan. Numerical implementations for 2D lid-driven cavity flow in stream function formulation. *ISRN Applied Mathematics*, 2:536 – 558, 2012.

[121] M. Golubitsky and W. F. Langford. Classification and unfoldings of degenerate Hopf bifurcations. *Journal of Differential Equations*, 41(3):375 – 415, 1981.

[122] M. Golubitsky, I. Stewart, and D. G. Schaeffer. *Singularities and Groups in Bifurcation Theory. Vol. II*, chapter 2. Applied Mathematical Sciences, vol. 69. Springer, New York, 1988.

[123] A. Y. Gelfgat. Different Modes of Rayleigh – Bénard Instability in Two- and Three-Dimensional Rectangular Enclosures. *Journal of Computational Physics*, 324:300–324, 1999.

[124] J. Pallares, M. P. Arroyo, F. X. Grau, and F. Giralt. Experimental laminar Rayleigh-Bénard convection in a cubical cavity at moderate Rayleigh and Prandtl numbers. *Experiments in Fluids*, 31(August 2000):208–218, 2001.

[125] W. Song, F. W. Wubs, J. Thies, and S. Baars. Numerical bifurcation analysis of a 3D Turing-type reaction–diffusion model. *Communications in Nonlinear Science and Numerical Simulation*, 60:145 – 164, 2018.

[126] T. K. Callahan and E. Knobloch. Pattern formation in three-dimensional reaction–diffusion systems. *Physica D: Nonlinear Phenomena*, 132(3):339 – 362, 1999.

[127] T. K. Callahan. Turing patterns with O(3) symmetry. *Physica D: Nonlinear Phenomena*, 188(1–2):65 – 91, 2004.

[128] P. Yu and A.B. Gumel. Bifurcation and stability analyses for a coupled Brusselator model. *Journal of Sound and Vibration*, 244(5):795 – 820, 2001.

[129] P. Gray and S. K. Scott. Autocatalytic reactions in the isothermal, continuous stirred tank reactor. *Chemical Engineering Science*, 38(1):29 – 43, 1983.

[130] I. Lengyel, G. Rábai, and I. R. Epstein. Batch oscillation in the reaction of chlorine dioxide with iodine and malonic acid. *Journal of the American Chemical Society*, 112(11):4606–4607, 1990.

[131] T. Leppänen. *Computatioanl studies of pattern formation in Turing Systems*, pages 25–28. Ph.D. Dissertation, Helsinki University of Technology, 2004.

[132] E. L. Allgower, K. Georg, and R. Miranda. *Exploiting Symmetry in Applied and Numerical Analysis*. American Mathematical Society, 1993.

[133] M. Golubitsky, I. Stewart, and D. G. Schaeffer. *Singularities and groups in bifurcation theory. Vol. II*, volume 69 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 1988.

[134] P. E. Hydon. *Difference equations by differential equation methods*, volume 27 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2014.

[135] M. Heroux, R. Bartlett, V. H. R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[136] T. Leppänen, M. Karttunen, R. A. Barrio, and K. Kaski. Turing systems as models of complex pattern formation. *Brazilian Journal of Physics*, 34:368 – 372, 2004.

[137] A. de Wit, P. Borckmans, and G. Dewel. Twist grain boundaries in three-dimensional lamellar Turing structures. In *Proc. Natl. Acad. Sci. USA*, 94, pages 12765–12768, 1997.

[138] D. R. Fokkema, G. L. G. Sleijpen, and H. A. Van der Vorst. Accelerated inexact Newton schemes for large systems of nonlinear equations. *SIAM Journal on Scientific Computing*, 19(2):657–674, 1998.

[139] E. F. F. Botta and F. W. Wubs. Matrix renumbering ILU: An effective algebraic multilevel ILU preconditioner for sparse matrices. *SIAM J. Matrix Analysis Applications*, 20:1007–1026, 1999.

Summary

Numerical flow simulations are playing a more and more important role not only in the scientific field but also in the industrial area, where an experimental practice is either difficult or expensive to realize. Numerical simulation methods have been studied intensively in the past decades to achieve good accuracy as well as low turnaround times. There is no numerical technique which is generally suitable for every problem. Usually, there are specific techniques for a given type of problem. For instance, a preconditioner is created depending on the structure and properties of the matrix arising from the system. The motion of a fluid is governed by the Navier-Stokes equations, which can model different types of fluid flows. For example, the fluid air in the modeling of the earth atmosphere is considered as compressible, while in the simulation of ocean tides, the fluid water is regarded as incompressible. Analytical solutions of the Navier-Stokes equations only exist for a limited number of cases, hence, for general cases, computing efficiently numerical solutions for these equations is essential. We focused on steady-state problems related to incompressible flows.

There are basically two approaches to find steady states. The first is a time-integration method which just finds it by integrating to the steady state. The second is to solve the nonlinear system directly by Newton's method. The goal here is to get an insight into what kind of problems can be solved readily with a continuation method and which of the approaches, time integration or direct solution, is favorable. For the latter we developed our own parallel continuation program, which is also presented in this thesis.

Methods

There are two main difficulties when solving the incompressible Navier-Stokes equations for a Newtonian fluid. The first one is the nonlinear system induced by the convective terms. The second is the constraint of mass conservation combined with the role of the pressure term which has no thermodynamical significance. Concerning the nonlinear system, it is attractive to use explicit time-integration methods to find steady-state solutions of the equations, e.g., by the forward Euler method, which avoids the need of solving a full set of nonlinear equations. The role of nonlinearity turns out to be more and more critical with the emergence of small structures in the flow dynamics. For instance, when the Reynolds number becomes high, which means that the convective effects are dominant relative to the viscous terms, the grid must be fine enough to capture all the scales in the flow accurately. This means in practice that finer and finer spatial and temporal grids must be used when the Reynolds number increases. As a consequence, there will be large computational costs.

The enforcement of the mass conservation leads to an algebraic constraint on the velocity field, and the pressure can be considered as the Lagrange multiplier associated to this constraint. The coupled velocity-pressure problem, after linearization, has a saddle-node structure which makes it difficult to solve. The matrix will be indefinite and when generating a standard preconditioner, say an incomplete factorization with fill determined by a drop tolerance, there is a considerable chance that it will be nearly singular leading to slow convergence or stagnation.

Apart from time integration, Newton-Krylov methods are the standard way to compute the steady state of the fully-coupled incompressible Navier-Stokes equations. The most challenging part is to solve the linear system, arising after discretization, efficiently. Usually, the problem is of huge scale, involving millions of unknowns and the amount of memory required for the factorization is not linear in the number of unknowns and computing time increases quite sharply because of the computation of all new elements during the factorization, especially for 3-dimensional problems. Therefore, robust solvers are essential.

A numerical continuation approach with a robust linear solver can gain accurate steady solutions with high efficiency. We have developed a parallel continuation program using data structures from the Epetra package available in the Trilinos library. The program uses the Trilinos package LOCA for the continuation in which the Newton-Krylov method is employed to solve the nonlinear equations. We used the Arnoldi method from the Anasazi package and the Jacobi-Davidson

method from the PHIST software to do the eigenvalue computation for the stability analysis. In solving the linear system in each Newton step and eigenvalue computation, proper preconditioners are needed to achieve fast convergence, for instance, LSC in Teko, ML, and HYMLS, among which HYMLS is our homemade code. HYMLS is a hybrid direct/iterative approach, aiming to combine the robustness of direct solvers with the memory and computational efficiency of iterative methods.

Focused Problems

Transitions in flows of liquids and gases are of great interest. We like to construct bifurcation diagrams showing for which parameter values transitions are to be expected. Such diagrams can be obtained by computing a series of steady states as a function of parameter values and the eigenvalues at the corresponding states. In the thesis, we analyzed four canonical flow problems: the lid-driven cavity, the differentially heated cavity, Rayleigh-Bénard convection, and differentially heated rotating cavity.

Another interesting problem studied in this thesis is a Turing-type reaction-diffusion model. The pattern formation behavior in Turing systems is very complex. It is known that 3D solutions can display much richer behavior than 2D solutions because there are many more possibilities for spatial multi-stability in 3D than there are in 2D. In this thesis, we have focused on a Turing model called the Barrio-Varea-Aragon-Maini (BVAM) model and gave a rich 3D bifurcation diagram as well as a stability analysis.

Results in this thesis

Our continuation program was able to perform the analysis for the flow problems and the Turing problem. Moreover, we could compute unstable steady states, which is impossible by time-integration methodology. We also showed that Picard iteration, in which an Oseen problem needs to be solved, does not converge for already modest values of the Reynolds number in the lid-driven cavity problem. Hence, the full Jacobian is necessary in Newton's method. Unfortunately, the linear system solver Teko using the LSC preconditioner did also not converge for

modest values of the Reynolds number. On the other hand, HYMLS performed well for a vast range of Reynolds numbers.

For the lid-driven cavity problem, we also compared our approach to the time-integration approach at Reynolds number 500. Unfortunately, that approach was not parallelized, but, if it would, we expect that the turnaround time would be similar to that of our approach. This leads to our main conclusion that the continuation approach, using a robust preconditioner like HYMLS, is competitive with the time-integration approach for the steady-state flow problems considered here.

Acknowledgments

Weiyan Song
Groningen
August 17, 2018