Multi-level ILU preconditioners and continuation methods in fluid dynamics

Tiesinga, Geesien

Chapter 4

Matrix Renumbering ILU applied to the Navier-Stokes equations

4.1 Matrix Renumbering ILU

In the previous chapters we have considered preconditioners based on an (M)ILU factorization with respect to a repeated red-black ordering. A disadvantage of such a preconditioner is that the numbering of the unknowns is based on the grid. To be able to make such a numbering a structured grid is needed. Furthermore, the decision to drop an element is based on its position only and not on its size. A preconditioner which works with a numbering based on the grid as well but drops the fill when its value is smaller than some threshold is the nested grid ILU (NGILU) factorization [48]. Both kind of preconditioners share the drawback that the numbering of the unknowns is based on the grid and not on the matrix. For this reason a preconditioner, the matrix renumbering ILU factorization (MRILU) [8] has been developed in which both the ordering and the dropping is based on the matrix. The MRILU preconditioner is a generalization of the NGILU preconditioner.

We will describe in short the ordering and dropping strategy of the MRILU preconditioner. The ordering is determined during the construction of the factorization and is based on the sparsity pattern of the matrix and the magnitude of the elements. During the factorization small elements are dropped, resulting in an incomplete LU factorization. To decide whether or not an element will be dropped, a lump space is determined which is based on the diagonal of the factorization. To make sure the lump space is not consumed too fast some additional restrictions are made.

In Algorithm 4.1 the steps of the construction of the MRILU factorization are given. Assume the factorization has progressed to the point that the Schur complement $A^{(i-1)}$ has been computed. The steps needed to compute the next Schur complement will be explained.

In step 1 a reordering of the unknowns and a partitioning of the Schur complement $A^{(i-1)}$ is made. For sparse matrices the unknowns can be divided such that the matrix A_{11} is diagonal, i.e. the unknowns of A_{11} form an independent set. By allowing also weak connections between the unknowns of A_{11}, i.e. they form a nearly independent set, A_{11} can be enlarged. The partitioning is made in such a way that the matrix A_{11} is strongly

```
$A^{(0)} = A$
for i=1:M
    1 Make a reordering and partitioning of $A^{(i-1)}$
        $\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$
    2 Approximate $A_{11}$ by a diagonal matrix $\tilde{A}_{11}$
    3 Drop small elements of $A_{12}$ and $A_{21}$
    4 Make an incomplete factorization
        $\begin{pmatrix} I & 0 \\ \tilde{A}_{21}\tilde{A}_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ 0 & A^{(i)} \end{pmatrix}$
      where $A^{(i)} = A_{22} - \tilde{A}_{21}\tilde{A}_{11}^{-1}\tilde{A}_{12}$
end
Make an exact or incomplete factorization of $A^{(M)}$
```

Algorithm 4.1: construction of MRILU factorization

diagonally dominant, i.e. the elements of A_{11} satisfy

$$\sum_{k \neq i} |a_{ik}| \leq \epsilon |a_{ii}| \quad \text{with } \varepsilon < 1.$$

The ordering of the unknowns is constructed by a greedy algorithm. By keeping track of the absolute sum of the columns belonging to the unknowns selected for A_{11}, it can easily be decided whether or not to add a new unknown to the near independent set selected so far.

In step 2 the matrix A_{11} is replaced by a diagonal matrix $\tilde{A}_{11}$. This is an accurate approximation because by construction the matrix A_{11} is strongly diagonally dominant. The approximation simplifies the construction of the next Schur complement $A^{(i)}$. The unknowns belonging to A_{11} can be eliminated simultaneously, which means that the ordering of these unknowns is not important.

In step 3 small elements will be dropped to limit the number of nonzero elements in the factorization. The dropping outside A_{11} is limited to A_{12} and A_{21}. Gustafsson's modification can be used when an element is dropped. With this modification the dropped element is added to the diagonal. When an element a_{ij} is dropped row i and column j are modified Whether or not it is acceptable to drop this element is measured by the ratio of the element a_{ij} and the diagonal elements a_{ii} and a_{jj} and on the amount dropped so far in row i and column j.

To obtain an accurate dropping strategy these criteria have to be adjusted. A better strategy is obtained when the dropping criteria are not based on the diagonal of A_{22} only. The diagonal of the next Schur complement $A^{(i)}$ may differ significantly from the diagonal of A_{22}, especially for matrices with strongly varying elements. Relatively small modifications compared to the diagonal of A_{22} may be large compared to this new diagonal. Therefore, the dropping criteria should be based on the diagonal D of the complete factorization. The diagonal of A_{11} becomes (slightly modified) a part of D. Therefore, dropping elements outside A_{11} is much more critical. The diagonal D of the complete factorization is not known. Within A_{22} it is approximated by the diagonal of the next Schur complement, which is temporarily calculated without the dropping outside A_{11}. An element on row or column i is dropped when the sum of the absolute values of all discarded elements on this row or column (including those discarded on earlier levels) is smaller than $\varepsilon|d_i|$. In the present implementation of MRILU the dropping strategy is just based on the diagonal of A_{22}.

The available space for dropping has to be restricted even further. When dropping an element a_{ij} of A_{21} row i is modified. On subsequent levels of the construction more elements of this row may be discarded. Therefore, the row space for dropping within A_{21} is restricted by multiplying the remaining space by the number of columns of A_{21} divided by the dimension of A. Furthermore, the available space for lumping on rows of A_{21} should not be consumed by a few large elements but rather by many small elements. Therefore, only entries smaller than a certain fraction of this space are dropped. Similar restrictions are made for the dropping within columns of A_{12}.

Finally, in step 4 the incomplete factorization can be computed, resulting in the next Schur complement. Step 1 to 4 can be repeated until the final Schur complement is of low order. Then, the final Schur complement can be factorized by a standard (I)LU factorization.

Matrices arising from the discretization of a system of partial differential equations can be written in block form, the elements of the matrix are small blocks instead of scalars. For these matrices a block form of the MRILU factorization can be used. To decide whether or not an entry of a block row can be dropped, this block row is multiplied from the left with the inverse of the corresponding diagonal block. This can be somewhat simplified by considering only the absolute maximum in each column of this inverse. The dropping in a column can be handled in a similar way using multiplication from the right.

4.2 Results for the Poisson and convection-diffusion equation

For Laplace-like equations a comparison between MRILU and other linear solvers is made in [7]. MRILU performed well for all test problems. In [8] such a comparison is made for other test problems. In this section we will recapitulate the results given in [8] for the Poisson and convection-diffusion equation. Unless denoted otherwise Bi-CGSTAB [49] is used as linear solver.

First the homogeneous Poisson equation with zero Neumann boundary conditions has been solved on an exponentially stretched grid with the ratio of the maximum and

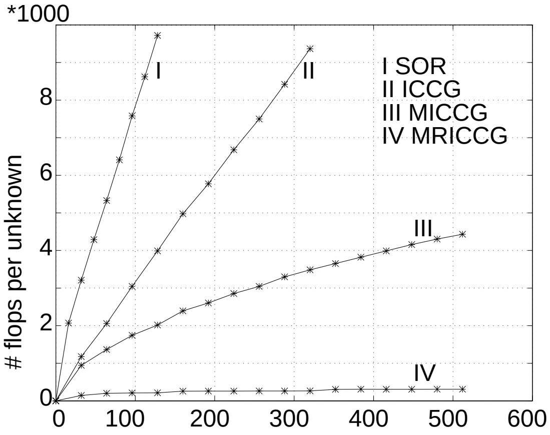

Figure 4.1: Results for the Poisson equation on an exponentially stretched grid.

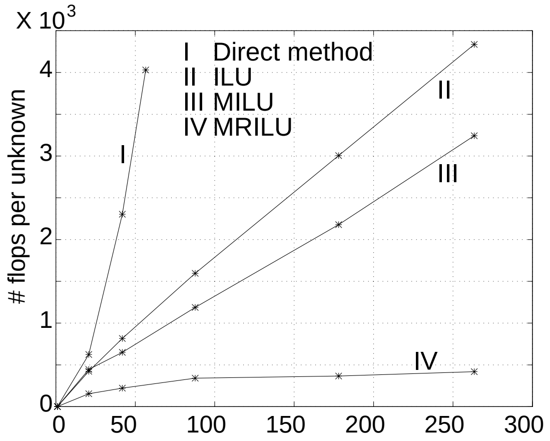

Figure 4.2: Results for the Poisson equation on a finite element grid.

minimum mesh size given by $h_{max}/h_{min} = 100$. As starting vector a nonzero vector has been used. The stopping criterion is given by $(u^{(i)}_{max} - u^{(i)}_{min}) < 10^{-6}(u^{(0)}_{max} - u^{(0)}_{min})$. In Figure 4.1 the symmetric version MRICCG of MRILU (with a simplified dropping strategy) is compared with the methods SOR, ICCG and MICCG. The figure shows for an $M \times M$ grid the number of flops per unknown as a function of M. Clearly, MRILU outperforms the other methods. Moreover, it shows nearly grid-independent convergence.

Then the Poisson equation $-\Delta u = f$ with Dirichlet boundary conditions has been solved on an unstructured finite-element grid. As stopping criterion the 2-norm of the residual of the preconditioned system has to be decreased by at least a factor 10^6. In Figure 4.2 MRILU with a simplified dropping strategy method is compared with a direct method, ILU and MILU. Again MRILU outperforms the other methods and is nearly grid-independent. The performance of MRILU is equally well on structured and unstructured grid. This shows that MRILU can handle general sparsity patterns.

approach	rel. fill	it.	flops
Point	2.0	35	2390
$\kappa = 2417$	2.6	11	911
	2.8	5	451
Block	0.9	8	435
$\kappa = 18$	1.4	4	268
	1.8	3	230

Table 4.1: Convection-diffusion problem (4.1) with $a = 10000, b = 1000$.

Another test case in [8] is the convection-diffusion equation

$$-u_{xx} - u_{yy} + au_x + bu_y = f, \quad a, b \gg 1, \tag{4.1}$$

with dominating convection. A second-order discretization has to be used to obtain sufficient accuracy. The central discretization provides this accuracy. A drawback of this discretization is that the coefficient matrix is not an M matrix. The diagonal is small with respect to the off-diagonal elements. This causes problems in incomplete decompositions. With MRILU these problems can be overcome by using a block approach. The unknowns are divided into pairs by keeping the unknowns of two subsequent grid points in the dominant flow direction together.

The problem is described by equation (4.1) with $a = 10000, b = 1000$ with the unit square as domain and with Dirichlet boundary conditions at $x = 0, y = 0$ and Neumann boundary conditions at $x = 1, y = 1$. The computations have been performed on a 32×32 grid until the preconditioned residual was decreased with a factor 10^{-6}. A convection dominated flow (mesh Peclet number of about 150) has been computed and the standard approach without blocks (which is equal to the approach in [19]) has been compared with the block approach. The results are shown in Table 4.1. In the first column the condition number of the first Schur complement in both approaches is given. The condition number in the block approach is significantly smaller than in the point-wise approach. In the second column the fill of the decomposition relative to the original matrix is given. The third column shows the number of iterations and the last column the flop count for the solution process. The large condition number has a negative effect on convergence. In the point-wise approach more fill-in is needed to get an acceptable number of iterations.

4.3 Results for the Navier-Stokes equations

In the previous section we have considered the performance of the MRILU preconditioner in test cases described by the Poisson and convection-diffusion equation. In this section we will solve the incompressible two-dimensional Navier-Stokes equations. The resulting linear systems are harder to solve than those occurring in the previous test cases.

The incompressible two-dimensional Navier-Stokes equations in conservation form are given by

$$\begin{aligned}
&\int_\Gamma \vec{u}.n \ \mathrm{d}\Gamma = 0,\\
&\int_\Omega \frac{\partial u}{\partial t} \ \mathrm{d}\Omega + \int_\Gamma (u\vec{u} - \nu \nabla u).n \ \mathrm{d}\Gamma = -\int_\Gamma p(n.e_1) \ \mathrm{d}\Gamma,\\
&\int_\Omega \frac{\partial v}{\partial t} \ \mathrm{d}\Omega + \int_\Gamma (v\vec{u} - \nu \nabla v).n \ \mathrm{d}\Gamma = -\int_\Gamma p(n.e_2) \ \mathrm{d}\Gamma,
\end{aligned}$$

with arbitrary domain Ω with boundary Γ. We will consider various discretization methods in order to get insight in the accuracy of the MRILU factorization for different types of matrices.

4.3.1 Symmetry-preserving discretization

In order to preserve the symmetry of the Navier-Stokes equations when discretizing the system, the diffusive terms have to be discretized in a symmetric positive definite way and the convective terms in a skew-symmetric manner. In [53] it is shown that with such a discretization the semi-discrete system is stable. We will show how to obtain a symmetry-preserving discretization.

We will discretize the equations by using a finite-volume approach. Though our starting point are central differences an upwind discretization of the convective terms can easily be obtained by adding artificial diffusion ν_{art} to the real diffusion ν. The discretization is explained by looking at the x-momentum equation and the continuity equation. The grid is given in Figure 4.3, note that $hx_c = \frac{1}{2}(hx_w + hx_e)$ and $hy_c = \frac{1}{2}(hy_n + hy_s)$. The

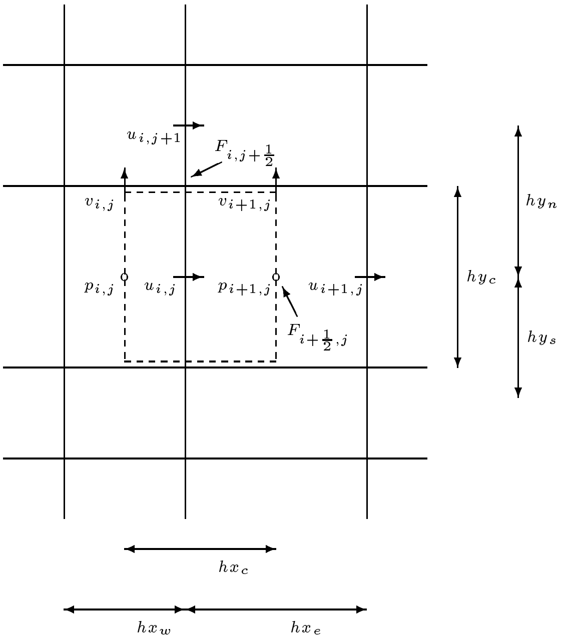

Figure 4.3: The two-dimensional grid and control volume.

convective and diffusive fluxes of the x-momentum equation are given by

$$F(\vec{u}) = (F_1(\vec{u}), F_2(\vec{u})) = (uu - \nu u_x, uv - \nu u_y).$$

The semi-discrete x-momentum equation is

$$hx_c hy_c \frac{du_{i,j}}{dt} + hy_c(F_1(\vec{u})_{i+\frac{1}{2},j} - F_1(\vec{u})_{i-\frac{1}{2},j}) + hx_c(F_2(\vec{u})_{i,j+\frac{1}{2}} - F_2(\vec{u})_{i,j-\frac{1}{2}}) = \\ -hy_c(p_{i+1,j} - p_{i,j}),$$

where

$$F_1(\vec{u})_{i+\frac{1}{2},j} = F_1(\vec{u}_{i+\frac{1}{2},j}) = u_{i+\frac{1}{2},j}\frac{u_{i+1,j} + u_{i,j}}{2} - (\nu + \alpha\ (\nu_{art})_{i+\frac{1}{2},j})\frac{u_{i+1,j} - u_{i,j}}{hx_e},$$

and

$$F_2(\vec{u})_{i,j+\frac{1}{2}} = F_2(\vec{u}_{i,j+\frac{1}{2}}) = v_{i,j+\frac{1}{2}}\frac{u_{i,j+1} + u_{i,j}}{2} - (\nu + \alpha\ (\nu_{art})_{i,j+\frac{1}{2}})\frac{u_{i,j+1} - u_{i,j}}{hy_n},$$

with

$$(\nu_{art})_{i+\frac{1}{2},j} = \frac{|u_{i+\frac{1}{2},j}|hx_e}{2} \quad , \quad (\nu_{art})_{i,j+\frac{1}{2}} = \frac{|v_{i,j+\frac{1}{2}}|hy_n}{2}.$$

The velocities $u_{i+\frac{1}{2},j}$ and $v_{i,j+\frac{1}{2}}$ are not defined in those points. They can be computed by taking an average of their surrounding values

$$u_{i+\frac{1}{2},j} = \frac{1}{2}(u_{i,j} + u_{i+1,j}) \quad , \quad v_{i,j+\frac{1}{2}} = \frac{1}{2}(hx_w v_{i,j} + hx_e v_{i+1,j})/hx_c.$$

The parameter α satisfies $\alpha \in [0, 1]$. By varying α one can adjust the amount of upwind in the scheme. For $\alpha = 1$ this is a first-order upwind scheme, and for $\alpha = 0$ this is a second-order central scheme.

For the continuity equation we take as control volume the standard grid cell. The discretization of the continuity equation at the cell around $p_{i,j}$ yields

$$hy_c(u_{i,j} - u_{i-1,j}) + hx_w(v_{i,j} - v_{i,j-1}) = 0.$$

The above discretization has a property which is important for stability: the central term of the convective part in the momentum equations vanishes, resulting in a skew-symmetric discretization of the convective term. From the equations we observe that this term equals

$$\frac{1}{2}[hy_c(u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}) + hx_c(v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}})]u_{i,j} \\ = \frac{1}{4}[hy_c(u_{i+1,j} - u_{i-1,j}) + hx_w(v_{i,j} - v_{i,j-1}) + hx_e(v_{i+1,j} - v_{i+1,j-1})]u_{i,j}.$$

Note that, apart from the factor 1/4, this central term is precisely the sum of the continuity equations at the cells around $p_{i,j}$ and $p_{i+1,j}$ and hence is zero.

4.3.2 Monotone discretization

Another class of discretizations is formed by the monotonicity preserving schemes. For simplicity we will look at the one-dimensional convection equation

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0.$$

A weak solution of this equation has the following monotonicity preserving properties (see [24]) as a function of t:

- no new local maximum or minimum is created,
- the value of a local minimum is nondecreasing and the value of a local maximum is nonincreasing.

From these properties it follows that the total variation

$$TV(u(t)) := \sup \sum_i |u(x_{i+1}, t) - u(x_i, t)|$$

is a nonincreasing function of t.

The monotonicity properties should be preserved by the discretization scheme. The equation is discretized with a finite volume method. The grid is given in Figure 4.4.

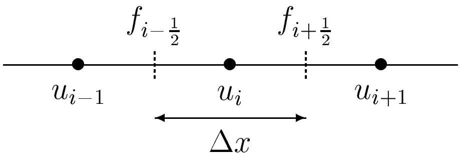

Figure 4.4: One-dimensional grid.

After semi-discretization we obtain

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}(f(u^n)_{i+\frac{1}{2}} - f(u^n)_{i-\frac{1}{2}}). \tag{4.2}$$

By an appropriate discretization of the fluxes at the boundaries of the control volume the discretization can be made total variation diminishing (TVD). A scheme is TVD when

$$TV(u^{n+1}) \leq TV(u^n),$$

with

$$TV(u^n) = \sum_{i=-\infty}^{\infty} |u_i^n - u_{i-1}^n|.$$

In [24] it is shown that a linear TVD scheme is of first-order accuracy. Therefore, to obtain a higher accuracy, nonlinear schemes are needed. For the construction of such schemes the next lemma plays an important role.

Lemma 3 (Harten [24])*: consider a discretization of (4.2) given by*

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}[A^n_{i+\frac{1}{2}}(u^n_{i+1} - u^n_i) + B^n_{i-\frac{1}{2}}(u^n_i - u^n_{i-1})], \tag{4.3}$$

where

$$\begin{aligned} A^n_{i+\frac{1}{2}} &= A(\ldots, u^n_{i-1}, u^n_i, u^n_{i+1}, \ldots), \\ B^n_{i+\frac{1}{2}} &= B(\ldots, u^n_{i-1}, u^n_i, u^n_{i+1}, \ldots). \end{aligned}$$

If the coefficients $A^n_{i+\frac{1}{2}}$ and $B^n_{i+\frac{1}{2}}$ satisfy

$$A^n_{i+\frac{1}{2}} \leq 0, \quad B^n_{i+\frac{1}{2}} \geq 0, \quad 1 - \frac{\Delta t}{\Delta x}(B^n_{i+\frac{1}{2}} - A^n_{i+\frac{1}{2}}) \geq 0,$$

the discretization (4.3) is TVD.

With this lemma a higher order monotone scheme can be derived [45]. Assume the flux function $f(u)$ can be split in a forward and backward flux:

$$f(u) = f^+(u) + f^-(u),$$

with

$$\frac{d}{du}f^+(u) \geq 0, \quad \frac{d}{du}f^-(u) \leq 0.$$

A finite volume discretization can then be written as

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}[(f^+(u^{L\ n}_{i+\frac{1}{2}}) + f^-(u^{R\ n}_{i+\frac{1}{2}})) - (f^+(u^{L\ n}_{i-\frac{1}{2}}) + f^-(u^{R\ n}_{i-\frac{1}{2}}))]. \tag{4.4}$$

The left and right values of u at the cell boundaries are taken as

$$\begin{aligned} u^L_{i+\frac{1}{2}} &= u_i + \frac{1}{2}\psi(r_i)(u_i - u_{i-1}), \qquad (4.5) \\ u^R_{i+\frac{1}{2}} &= u_{i+1} + \frac{1}{2}\psi(\frac{1}{r_{i+1}})(u_{i+1} - u_{i+2}), \end{aligned}$$

with

$$r_i = \frac{u_{i+1} - u_i}{u_i - u_{i-1}}.$$

These expressions for u consist of a first-order term and a higher-order term. The function ψ is called a limiter function. By writing the discretization (4.4) in the form of equation (4.3) from lemma 3 restrictions can be put on ψ in order to obtain a TVD scheme. The limiter ψ has to satisfy (see [44])

$$\frac{\psi(r)}{r} - \psi(s) \leq 2.$$

We will consider monotone schemes based on the κ-schemes. With the κ-schemes [51] second-order accuracy can be obtained. These discretization schemes are given by

$$\begin{aligned} u^L_{i+\frac{1}{2}} &= u_i + \frac{1+\kappa}{4}(u_{i+1} - u_i) + \frac{1-\kappa}{4}(u_i - u_{i-1}), \\ u^R_{i+\frac{1}{2}} &= u_{i+1} + \frac{1+\kappa}{4}(u_i - u_{i+1}) + \frac{1-\kappa}{4}(u_{i+1} - u_{i+2}), \end{aligned} \tag{4.6}$$

with $\kappa \in [-1, 1]$. For $\kappa = -1$ one gets the second-order accurate, fully one-sided upwind scheme, for $\kappa = 1$ the standard, second-order accurate scheme and for $\kappa = \frac{1}{3}$ a third-order scheme. Equation (4.6) can be written in the form (4.5), with

$$\psi(r) = \frac{1-\kappa}{2} + \frac{1+\kappa}{2}r.$$

This limiter does not satisfy the conditions for a TVD discretization. Hence, the κ-schemes are not TVD.

The $\kappa = \frac{1}{3}$-scheme, the highest order κ-scheme, will be made TVD. For $\kappa = \frac{1}{3}$ we obtain

$$u^L_{i+\frac{1}{2}} = u_i + \frac{1}{2}(\frac{1}{3} + \frac{2}{3}r_i)(u_i - u_{i-1}). \tag{4.7}$$

Koren [30] constructed a limiter consistent with this discretization. This limiter is given by

$$\psi(r) = \max(0, \min(2r, \min(\frac{1}{3} + \frac{2}{3}r, 2))).$$

A disadvantage of this limiter is that it is not continuously differentiable. Therefore, Koren [29] constructed a continuously differentiable limiter by writing (4.7) as

$$u^L_{i+\frac{1}{2}} = u_i + \frac{1}{2}\Phi(r_i)(\frac{1}{3} + \frac{2}{3}r_i)(u_i - u_{i-1}).$$

Additional to the TVD conditions for the limiter $\Phi(r_i)(\frac{1}{3} + \frac{2}{3}r_i)$, the function Φ has to fulfill the requirements $\Phi(1) = 1, \Phi(0) = 0, \Phi'(1) = 0$ and Φ has to be bounded for large $|r|$. A limiter $\psi(r) = \Phi(r)(\frac{1}{3} + \frac{2}{3}r)$ that satisfies these conditions is

$$\psi(r) = \frac{2r^2 + r}{2r^2 - r + 2}. \tag{4.8}$$

Jacobian from a monotone discretization

The diagonal of the coefficient matrix is of great influence on the accuracy of the MRILU factorization. The stronger the diagonal the better the factorization will be. We will consider the diagonal of the coefficient matrix stemming from a monotone discretization.

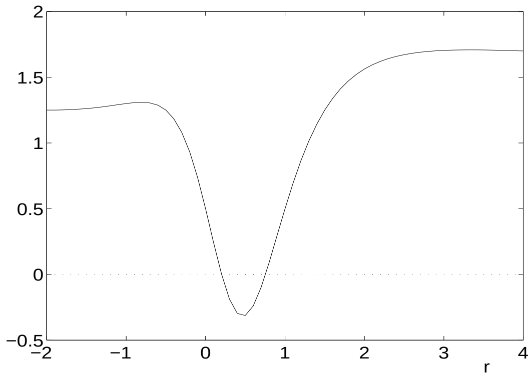

Figure 4.5: The diagonal element of the Jacobian when using the monotone scheme with the continuously differentiable limiter.

With the monotone discretization scheme the convective term $a\frac{\partial u}{\partial x}$ with $a > 0$ is discretized as follows:

$$\frac{a}{\Delta x}(u^L_{i+\frac{1}{2}} - u^L_{i-\frac{1}{2}}) = \frac{a}{\Delta x}(1 + \frac{1}{2}\psi(r_i) - \frac{1}{2}\frac{\psi(r_{i-1})}{r_{i-1}})(u_i - u_{i-1}).$$

This scheme is TVD when

$$(1 + \frac{1}{2}\psi(r_i) - \frac{1}{2}\frac{\psi(r_{i-1})}{r_{i-1}}) \geq 0.$$

This means that the coefficient for u_i is positive and for u_{i-1} is negative, resulting in an M-matrix. However, when a full Newton method is used the coefficient matrix is the Jacobian of the discretized system.

The contribution to the diagonal of the Jacobian is obtained by taking the derivative with respect to u_i:

$$\frac{\partial}{\partial u_i}((1 + \frac{1}{2}\psi(r_i) - \frac{1}{2}\frac{\psi(r_{i-1})}{r_{i-1}})(u_i - u_{i-1})) =$$
$$-\frac{1}{2}(r_i + 1)\frac{\partial\psi}{\partial r}(r_i) - \frac{1}{2}\frac{\partial\psi}{\partial r}(r_{i-1}) + 1 + \frac{1}{2}\psi(r_i).$$

We see that this diagonal is dependent on both ψ and its derivative. To obtain a nice coefficient matrix, which is good to factorize, this diagonal element has to be positive. This gives the condition

$$1 + \frac{1}{2}\psi(r) - \frac{1}{2}(r + 1)\frac{\partial\psi}{\partial r}(r) - \frac{1}{2}\frac{\partial\psi}{\partial r}(s) \geq 0.$$

For the κ-schemes

$$\psi(r) = \frac{1-\kappa}{2} + \frac{1+\kappa}{2}r.$$

Hence, the diagonal coefficient is

$$\frac{3}{4}(1-\kappa),$$

which is positive for κ in $[-1, 1]$. For increasing κ the diagonal of the matrix will become weaker.

For the continuously differentiable limiter $\psi(r) = \frac{2r^2+r}{2r^2-r+2}$ the plot of the diagonal element under the assumption $s = r$, which is a reasonable assumption when the flow is slowly changing in space, is given in Figure 4.5. From this plot we observe that the diagonal element may become negative. This may cause problems when constructing a factorization.

4.3.3 Block form and scaling of the coefficient matrix

We will solve the Navier-Stokes equations simultaneously. Consequently, very large systems have to be solved. The equations of this system, the momentum equations and the continuity equation, are of different character. To avoid that one of the equations will dominate the factorization, the unknowns of a grid cell are put together in a vector $w_{ij} = (u_{ij}, v_{ij}, p_{ij})$. This introduces in a natural way a block form of the discretized system. The elements of this system are three by three blocks.

As preconditioner for this system we will use the block form of the MRILU factorization. To obtain an accurate factorization with a moderate fill, the matrix has to be scaled. In the construction of the MRILU factorization the inverses of the diagonal blocks are needed. To prevent that, as a consequence of the premultiplication with these inverses, the elements of the factorization will differ greatly in size, the elements of the inverses have to be of equal size.

After discretization of the Navier-Stokes equations a typical diagonal block of the coefficient matrix will look like

$$\begin{pmatrix} u^* & 0 & \frac{1}{hx} \\ 0 & v^* & \frac{1}{hy} \\ \frac{1}{hx} & \frac{1}{hy} & 0 \end{pmatrix}$$

We will scale the diagonal block instead of its inverse. When the elements of the diagonal block are of equal size, so are the elements of its inverse. In general u^* and v^* will be of about the same size, and so will $\frac{1}{hx}$ and $\frac{1}{hy}$. But u^* and v^* will differ in size from $\frac{1}{hx}$ and $\frac{1}{hy}$. By scaling the third column and row of the diagonal block with a factor s, its elements can be made of about equal size. The elements will be of equal size when

$$u^* = \frac{s}{hx} \quad , \quad v^* = \frac{s}{hy}.$$

Summing both equations and solving for the scaling factor s, results in

$$s = (u^* + v^*)\frac{hxhy}{hx + hy}.$$

The scaled diagonal block

$$\begin{pmatrix} u^* & 0 & (u^* + v^*)\frac{hy}{hx+hy} \\ 0 & v^* & (u^* + v^*)\frac{hx}{hx+hy} \\ (u^* + v^*)\frac{hy}{hx+hy} & (u^* + v^*)\frac{hx}{hx+hy} & 0 \end{pmatrix},$$

and hence its inverse, has elements of about equal size.

We observed that using this scaling decreased the cpu-time for the linear solver with about a factor 2. More research is needed to improve the scaling.

4.3.4 Results

To test the performance of the MRILU factorization when applied to the Navier-Stokes equations, the lid-driven cavity problem has been solved. In order to solve both convection and diffusion dominated flows, the problem has been solved for Reynolds numbers 1000, 5000 and 10000. The diffusive terms have been discretized with the usual second-order method. For the convective terms various discretization methods have been used: the upwind and central discretization as described in subsection 4.3.1, the κ-schemes for various values of κ and a monotone scheme.

In all the computations as stopping criterion for Newton's method the infinity norm of two succeeding iterates has to be smaller than 10^{-6}. The Bi-CGSTAB method [49] preconditioned with an MRILU factorization has been used to solve the resulting linear systems. As stopping criterion for the linear solver the 2-norm of the residual of the preconditioned system has to be decreased by at least a factor 10^{6}.

The results are displayed in tables showing the relative fill of the MRILU factorization, the number of iterations of the linear solver and the flops per unknown needed to compute a solution. For each case factorizations with a low, medium and high fill have been considered. The parameters determining the MRILU factorizations were tuned such that for the different cases the relative fill of the factorizations remained almost equal. Only when the linear solver needed more than 80 iterations to converge, the relative fill was increased in order to obtain an acceptable number of iterations. The linear solver was considered inadequate (denoted with a dash in the tables) when a relative fill higher than 6 was needed to obtain convergence within 80 iterations.

In Table 4.2 the results for the upwind and central discretization as described in subsection 4.3.1 are given. In this table the relative fill, defined by the quotient of the fill of the MRILU factorization and the fill of the original matrix, the number of iterations and the number of flops needed for the solution process are given. The equations have been discretized on a 128 × 128 grid which has been stretched near the walls. The stretching is determined by the mapping

$$x_s = \frac{1}{2}(1 + \tanh(2s(x_u - \frac{1}{2}))/\tanh(s)).$$

This expression maps grid points x_u from a uniform grid to those of a non-equidistant stretched grid x_s. In both x- and y- direction we use the stretching $s = 1.5$.

For the upwind discretization increasing the Reynolds number results in an increase of the number of iterations when the relative fill in the factorization is kept fixed. Apparently, the flows dominated by convection are harder to compute. This is caused by the fact that the contribution of the discretization of the convection terms to the diagonal of the Jacobian is smaller than that of the diffusion term. A weak diagonal results in a poor factorization. For the central discretization it is not possible to obtain an adequate factorization when the Reynolds number is increased. The discretization of the convection

	upwind			central		
	rel. fill	it.	flops	rel. fill	it.	flops
Re=1000	2.6	36	5497	2.6	55	8270
	3.3	27	4821	3.3	43	7654
	4.0	25	4995	4.0	39	7044
Re=5000	2.6	38	5815	-	-	-
	3.3	33	5930	-	-	-
	4.0	25	5097	-	-	-
Re=10000	2.6	50	7617	-	-	-
	3.3	45	8006	-	-	-
	4.0	40	8046	-	-	-

Table 4.2: Performance of MRILU for upwind and central discretization.

		Re=1000			Re=5000			Re=10000		
$\kappa = -1$	rel.fill	2.2	2.9	3.4	2.9	3.6	4.3	3.5	4.0	4.5
	it.	41	24	20	47	42	37	57	49	39
	flops	7169	4909	4495	9648	9717	9642	12867	12084	10550
$\kappa = 0$	rel.fill	2.2	2.9	3.5	3.5	4.0	4.4	4.5	5.1	5.7
	it.	40	27	22	58	54	45	67	58	42
	flops	7012	5544	5002	13014	13280	11929	18205	17131	13551
$\kappa = \frac{1}{3}$	rel.fill	2.3	2.9	3.3	3.9	4.4	5.1	-	-	-
	it.	36	28	22	59	52	46	-	-	-
	flops	6357	5654	4906	14451	13649	13384	-	-	-
$\kappa = \frac{1}{2}$	rel.fill	2.3	2.9	3.4	-	-	-	-	-	-
	it.	34	26	22	-	-	-	-	-	-
	flops	6040	5364	4972	-	-	-	-	-	-
monotone	rel.fill	2.3	3.0	3.4	4.1	4.5	5.1	-	-	-
	it.	36	26	23	62	44	38	-	-	-
	flops	6356	5397	5207	15495	11902	11250	-	-	-

Table 4.3: Performance of MRILU for κ-schemes and a monotone scheme.

term does not contribute to the diagonal. Therefore, the diagonal of the Jacobian is very small in case of convection dominated flows. Comparing the results at Reynolds number 1000 of the upwind discretization and the central discretization, the number of iterations with the upwind discretization is smaller than with the central discretization. A strong diagonal, as results from the upwind discretization, results in a better factorization.

Table 4.3 shows the performance of the MRILU factorizations when the convection terms are discretized with a κ-scheme or a monotone scheme based on the $\kappa = \frac{1}{3}$-scheme.

The computations have been performed on an 128×128 equidistant grid. The κ-schemes that are considered are those with $\kappa = -1$ (second order upwind), $\kappa = 0$ (Fromm's scheme), $\kappa = \frac{1}{3}$ (third-order accurate) and $\kappa = \frac{1}{2}$ (the Quick method). In the end of Section 4.3.2 we showed that for increasing values of κ the contribution of the discretization of the convective terms to the diagonal of the Jacobian becomes smaller. Therefore, in Table 4.3 for a fixed Reynolds number a drop in the performance of the MRILU factorization is observed when κ is increased. This table also shows the deterioration of the performance as a consequence of the increase of the Reynolds number.

The monotone scheme most consistent with the third-order accurate $\kappa = \frac{1}{3}$-scheme has limiter $\psi(r) = \max(0, \min(2r, \min(\frac{1}{3} + \frac{2}{3}r, 2)))$. This limiter is not continuously differentiable, and hence causes convergence problems for Newton's method. Therefore, we have used the limiter $\psi(r) = \frac{2r^2+r}{2r^2-r+2}$. In the end of Section 4.3.2 the plot of the contribution of this monotone discretization to the diagonal of the Jacobian is shown. This contribution may become small or even negative, which will results in a bad factorization. In Table 4.3 this is seen from the inadequate convergence for Reynolds number 10000. Nevertheless, the performance for the monotone scheme is better than for the $\kappa = \frac{1}{3}$-scheme.

From the results we have seen that a faster convergence is obtained when the Jacobian of the system has a strong diagonal. A way to improve the diagonal is by adding a time derivative, i.e. solving the time-dependent equations. In Table 4.4 the performance of the MRILU factorizations for such time-dependent computations is shown. The time step is taken such that the Courant number is close to 1. In these computations the relative fill can be taken much smaller than in the steady computations. If the time step is sufficiently small the performance of the MRILU factorization is equally well for all discretization methods and Reynolds numbers.

		Re=1000			Re=5000			Re=10000		
upwind	rel.fill	1.2	1.5	2.0	1.2	1.5	2.0	1.2	1.5	2.0
	it.	25	20	15	26	17	16	24	19	16
	flops	2638	2283	2017	2727	1965	2111	2536	2178	2144
central	rel.fill	1.2	1.5	2.0	1.2	1.5	2.0	1.2	1.5	2.0
	it.	24	19	16	25	18	16	25	18	16
	flops	2534	2196	2125	2633	2076	2145	2635	2065	2144
$\kappa = \frac{1}{3}$	rel.fill	1.3	1.6	2.0	1.3	1.6	2.0	1.3	1.6	2.0
	it.	23	17	14	21	18	13	22	17	14
	flops	3176	2576	2387	2888	2739	2205	3029	2588	2372
monotone	rel. fill	1.3	1.6	2.0	1.3	1.6	2.0	1.3	1.6	2.0
	it.	23	17	14	23	16	13	21	15	12
	flops	3185	2587	2397	3172	2457	2232	2891	2289	2036

Table 4.4: Performance of MRILU for time-dependent systems.

4.4 Conclusions

Preconditioning techniques are very important when solving systems iteratively. The matrix renumbering ILU factorization (MRILU) has proven to be very succesful as preconditioner. The ordering of the unknowns is determined during the factorization. Both the ordering and dropping are based on the size of the elements of the matrix. Because of this property MRILU can deal with matrices with an arbitrary sparsity pattern, stemming from a discretization on an unstructured grid.

The MRILU factorization has already been applied successfully to many problems [8], both symmetric, non-symmetric and indefinite. Compared to other advanced iterative methods the number of flops has decreased significantly. For the Poisson problem almost grid independent convergence has been observed.

We have applied the MRILU factorization to the Navier-Stokes equations, which are harder to solve than the Poisson and convection-diffusion equation, and obtained good results. The MRILU factorizations give better results when the system has a strong diagonal. Therefore, the discretization of the convection terms is of great influence on the performance of the MRILU factorization. A strong diagonal can also be obtained by adding a time derivative to the system.

The development of the MRILU factorization is still in progress. Therefore, further improvements can be expected in the future.

Chapter 5

MRILU in a continuation method

5.1 Introduction

In physical applications one is often interested in stationary solutions of partial differential equations and how their behaviour depends on physical parameters. For instance, one would like to know what the possible steady solutions are for particular values of the parameters and whether these solutions are stable or not. A continuation method can be used to trace branches of stable and unstable solutions. The stability of a solution and the bifurcation points of the system can be determined by solving a generalized eigenvalue problem. Since long, continuation methods are used in problems with a small number of degrees of freedom. Only recently they are successfully applied to large systems, resulting for instance from discretization of partial differential equations governing fluid flows. The bottle-neck in applying continuation methods to large systems is solving the occurring linear systems and generalized eigenvalue problems.

The continuation code that we have used has been obtained from the oceanography group of Utrecht University, see [14] for a complete description of this code. In this code a pseudo-arclength continuation [28] is used to step along the solution branch. The respective solutions on such a branch are computed with a predictor-corrector method. Furthermore, the stability of the solutions and the position of the bifurcation points can be determined. A linear solver and an eigenvalue solver had to be added to this continuation code.

For solving large linear systems direct methods are too expensive in both cpu-time and storage requirements. Therefore, iterative methods are preferred. Preconditioned conjugate gradient type methods are widely used. The preconditioner is the main factor determining the quality of this kind of methods. An important class of preconditioners are the incomplete LU factorizations. In the previous chapter we showed that the MRILU factorization is an effective preconditioner. Therefore, as linear solver in the continuation code we use the Bi-CGSTAB method [49] preconditioned with an MRILU factorization.

Small generalized eigenvalue problems can be solved by the QZ method. However, for large systems this method is not practical. Large eigenvalue problems are commonly solved by the Arnoldi method. Recently, Fokkema, Sleijpen and van der Vorst have developed the Jacobi-Davidson QZ (JDQZ) method [20] as an alternative to the Arnoldi method. The JDQZ method is very well suited for computing a number of eigenvalues close to some user specified target. It computes iteratively a partial Schur form, i.e. a partial QZ factorization, with the Jacobi-Davidson method. Unlike the Arnoldi method, the JDQZ method does not have to solve systems exactly. To obtain an acceptable convergence of the JDQZ method a preconditioner is needed. We will use the MRILU factorization as described in the previous chapter.

As an application of the continuation code we have computed Rayleigh-Bénard flows. This problem has also been studied in [15]. In that paper the streamfunction-vorticity formulation of the problem has been used. We have applied the continuation method to the formulation in primitive variables. In [15] a preconditioned gradient like method as presented in [47] has been used as linear solver. We have used the Bi-CGSTAB method preconditioned with an MRILU factorization. In [50] it has been shown that for this problem the JDQZ method is more efficient for solving the generalized eigenvalue problems than the SIT method which has been used in [15]. In [50] an exact LU factorization has been used as preconditioner in the JDQZ method. However, for large problems this is not very efficient. Therefore, we have used an MRILU factorization as preconditioner in the JDQZ method.

In the remainder of this chapter we will explain the various aspects of the continuation code. In Section 5.2 the parametrization of the branches and the predictor-corrector method to compute solutions on a branch are treated. Furthermore, in that section it is explained how to detect bifurcation points, switch at these points to another branch and determine the stability of a solution. The arising generalized eigenvalue problems are solved with the JDQZ method, which is described in Section 5.3. As an application we computed Rayleigh-Bénard flows in a rectangular box of aspect ratio ten. The results are given in Section 5.4. Finally, in Section 5.5 we give some conclusions.

5.2 Continuation method

After semi-discretization, an autonomous time-dependent system of nonlinear partial differential equations can be written as

$$B\dot{u}(t) = f(u(t), \lambda),$$

with $\lambda \in R$ a parameter, $u(t) \in R^n$ the solution vector, $B \in R^{n \times n}$ a matrix representing the time dependency (this matrix may be singular) and f a nonlinear mapping from $R^n \times R \rightarrow R^n$.

In this chapter we will consider stationary solutions of this system,

$$f(u, \lambda) = 0.$$

The solutions u depend on the parameter λ and for fixed values of λ more solutions may exist. With a continuation method all branches of solutions can be calculated.

In the remainder of this section the different parts of the continuation method are explained. First, ways to parametrize the branches are given; a parametrization with λ is not always a good choice. Then, a predictor-corrector method is explained. Assuming a solution on a branch is known, this method computes the next solution on this branch. Finally, it is explained how to determine the stability of a solution and the position of branch points.

5.2.1 Parametrization

To be able to follow a branch, a parametrization of this branch is needed. Various parametrizations can be used. An obvious choice is to parametrize it by λ, the problem parameter. With this choice difficulties are encountered at turning points. At such points the system will be singular. With a different parametrization this singularity can be avoided.

A general approach for the parametrization is to choose another variable, say γ, as parameter. The solutions of $f(u,\lambda)=0$ depend on γ and are given by $(u(\gamma),\lambda(\gamma))$. Now, an additional equation is needed to establish the parametrization: $n(u,\lambda,\gamma)=0$. The extended system is given by

$$\begin{aligned} &f(u,\lambda)=0,\\ &n(u,\lambda,\gamma)=0. \end{aligned}$$

The parametrization by λ falls in this setting by taking $n(u,\lambda,\gamma)=\lambda-\gamma$.

As parameter on the branch the arc-length can be used, that is $\gamma=s$. A normalization of the arc-length gives the equation

$$n(u,\lambda,s)=\|\dot{u}(s)\|^2+|\dot{\lambda}(s)|^2-1=0.$$

Assuming a solution on the branch is known, say $(u(s_0),\lambda(s_0))$, this normalization equation can be approximated by

$$n_1(u,\lambda,s)=\|u(s)-u(s_0)\|^2+(\lambda(s)-\lambda(s_0))^2-(s-s_0)^2=0.$$

One likes to avoid the nonlinearity in this equation. This is possible when the derivative $(\dot{u}(s_0),\dot{\lambda}(s_0))$ is known as well. Then the linear equation

$$n_2(u,\lambda,s)=\dot{u}(s_0)^T(u(s)-u(s_0))+\dot{\lambda}(s_0)(\lambda(s)-\lambda(s_0))-(s-s_0)=0$$

can be used in the extended system. The equation n_2 is called a pseudo-arclength normalization. In the used code this parametrization is used, where the derivatives are approximated using the last two computed solutions.

5.2.2 Predictor-corrector method

Assume a solution (u_0,λ_0) is known on the branch. With the continuation method the next solutions $(u_1,\lambda_1),(u_2,\lambda_2),\ldots$ on the branch are computed. In the j-th continuation step the solution (u_{j+1},λ_{j+1}) has to be computed from the solution (u_j,λ_j). With a predictor-corrector method this is split into two steps. The predictor provides an initial guess $(\bar{u}_{j+1},\bar{\lambda}_{j+1})$ for the corrector step, which calculates the solution (u_{j+1},λ_{j+1}) on the branch.

As predictor we use the Euler method. The prediction $(\bar{u}_{j+1},\bar{\lambda}_{j+1})$ of (u_{j+1},λ_{j+1}) is given by

$$(\bar{u}_{j+1},\bar{\lambda}_{j+1})=(u_j,\lambda_j)+\Delta s(\dot{u}_j,\dot{\lambda}_j).$$

The tangent $(\dot{u}_j, \dot{\lambda}_j)$ to the branch can be computed from $\frac{\partial f}{\partial u}\frac{du}{ds} + \frac{\partial f}{\partial \lambda}\frac{d\lambda}{ds} = 0$. But since this involves another solve, the tangent is approximated by

$$(\dot{u}_j, \dot{\lambda}_j) \approx (\frac{u_j - u_{j-1}}{\Delta s}, \frac{\lambda_j - \lambda_{j-1}}{\Delta s}).$$

By this approach the predictor has become a standard extrapolation.

In the corrector step we use the Newton method. The result from the predictor is used as initial guess for the Newton method

$$(u^{(0)}, \lambda^{(0)}) = (\bar{u}_{j+1}, \bar{\lambda}_{j+1}).$$

The k-th step of the Newton method applied to the extended system

$$\begin{aligned} f(u, \lambda) &= 0, \\ n_2(u, \lambda, s) &= 0 \end{aligned}$$

is formulated as

$$\begin{aligned} &\begin{bmatrix} f_u(u^{(k)}, \lambda^{(k)}) & f_\lambda(u^{(k)}, \lambda^{(k)}) \\ \dot{u}_j^T & \dot{\lambda}_j \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -f(u^{(k)}, \lambda^{(k)}) \\ -n_2(u^{(k)}, \lambda^{(k)}, s) \end{bmatrix}, \\ &(u^{(k+1)}, \lambda^{(k+1)}) = (u^{(k)} + \Delta u, \lambda^{(k)} + \Delta \lambda). \end{aligned}$$

The solution of this system is obtained by solving the two systems

$$\begin{aligned} f_u(u^{(k)}, \lambda^{(k)})\ z &= -f(u^{(k)}, \lambda^{(k)}), \\ f_u(u^{(k)}, \lambda^{(k)})\ y &= f_\lambda(u^{(k)}, \lambda^{(k)}). \end{aligned}$$

Then, Δu and $\Delta \lambda$ are given by

$$\begin{aligned} \Delta \lambda &= \frac{-n_2(u^{(k)}, \lambda^{(k)}, s) - \dot{u}_j^T z}{\dot{\lambda}_j - \dot{u}_j^T y}, \\ \Delta u &= z - \Delta \lambda y. \end{aligned}$$

5.2.3 Branch points

With the continuation method a branch of solutions can be traced. Upon varying λ the number of solutions may change. New branches may emerge, branches may end, or branches may intersect. To get a complete picture of all the solution branches it is important to locate the branch points.

Branch points can be divided in turning points and bifurcation points. A turning point occurs when solutions exist for $\lambda < \lambda_0$ (or $\lambda > \lambda_0$) only. At a bifurcation point branches intersect. We will consider only simple bifurcation points. In these points exactly two branches with different tangent intersect. These branch points can be characterized as follows:

Turning point:

(i) f_u has a simple eigenvalue 0 at (u_0, λ_0) or, equivalently, rank $f_u(u_0, \lambda_0)$ is $n-1$,

(ii) $f_\lambda(u_0, \lambda_0) \notin$ range $f_u(u_0, \lambda_0)$.

Simple bifurcation point:

(i) f_u has a simple eigenvalue 0 at (u_0, λ_0) or, equivalently, rank $f_u(u_0, \lambda_0)$ is $n-1$,

(ii) $f_\lambda(u_0, \lambda_0) \in$ range $f_u(u_0, \lambda_0)$.

From the conditions (ii) it follows that the Jacobian of the extended system is singular in bifurcation points and nonsingular in turning points. Therefore, with a pseudo-arclength parametrization turning points cause no problem in the continuation method. In our study of stationary solutions we restrict ourselves to these two types of branch points.

For completeness we mention Hopf bifurcations as well. At these bifurcation points periodic, and hence time-dependent, solutions emerge. These bifurcation points can be characterized by:

Hopf bifurcation:

(i) $f_u(u_0, \lambda_0)$ has a simple pair of purely imaginary eigenvalues $\pm i\beta$ and no other eigenvalue with zero real part.

At a Hopf bifurcation point the emerging periodic solution has period $\frac{2\pi}{\beta}$.

In the remainder of this subsection we will explain how the exact position of a branch point can be found, and how to switch to another branch once a branch point is located.

Detecting branch points

In general, the continuation method will jump over a branch point. A test function $\tau(u(s), \lambda(s))$ is needed to monitor whether a branch point is passed. The test function is chosen such that a branch point is indicated by its zero,

$$\tau(u(s), \lambda(s)) = 0.$$

Because it is unlikely to find this zero (or branch point) exactly, a change of sign of τ in the j-th continuation step

$$\tau(u_{j+1}, \lambda_{j+1})\tau(u_j, \lambda_j) < 0$$

indicates a branch point is passed.

A turning point can be detected by monitoring

$$\tau(u(s), \lambda(s)) = \frac{d\lambda}{ds}.$$

When a turning point is passed this test function changes sign.

To detect a bifurcation point the real part of the eigenvalues of f_u are monitored. If an eigenvalue passes the imaginary axis a bifurcation point is passed. When $\mu_1, \ldots, \mu_n$ are the eigenvalues of f_u the test function is given by

$$\tau(u(s), \lambda(s)) = \max(\mathrm{Re}(\mu_1), \ldots, \mathrm{Re}(\mu_n)).$$

Denoting the eigenvalue with real part equal to zero by μ_0 a Hopf bifurcation is passed if in addition $|\mathrm{Im}(\mu_0)| > 0$.

After a branch point is detected its exact position has to be determined. The secant method is used to determine the zero of τ. If τ changes sign between s_a and s_b the zero of τ can be found with the iterative procedure

$$s_{l+1} = s_l - \tau(s_l)\frac{s_l - s_{l-1}}{\tau(s_l) - \tau(s_{l-1})},$$

with $s_0 = s_a$ and $s_1 = s_b$.

Branch switching

Once the location of a bifurcation point has been determined, a first solution on the emanating branch has to be calculated. This solution can be used as a starting point for the continuation method on this new branch. We denote the first solution on the new branch by (u_{new}, λ_{new}) and the known solution at the bifurcation point by (u_0, λ_0). The Euler predictor is used to obtain a first guess for (u_{new}, λ_{new}):

$$\begin{aligned}\bar{u}_{new} &= u_0 + \Delta s \dot{u}_{new},\\ \bar{\lambda}_{new} &= \lambda_0 + \Delta s \dot{\lambda}_{new}.\end{aligned}$$

Hereafter, the Newton method is used to calculate the solution on the new branch.

The tangent $(\dot{u}_{new}, \dot{\lambda}_{new})$ of the emanating branch is unknown. Differentiating the equation $f(u(s), \lambda(s)) = 0$ with respect to s shows that a tangent $(\frac{du}{ds}, \frac{d\lambda}{ds})$ to a solution branch satisfies

$$f_u(u(s), \lambda(s))\frac{du}{ds} + f_\lambda(u(s), \lambda(s))\frac{d\lambda}{ds} = 0. \tag{5.1}$$

In a simple bifurcation point two branches with different tangent intersect. Hence, at such a bifurcation point the solution space of equation (5.1) is two dimensional. Therefore, we cannot determine the desired tangent from the above equation. In fact, the local behaviour is determined by higher-order terms which lead to the so called bifurcation equation (see [14], [28]). One solution of the bifurcation equation is the tangent to the known branch at the bifurcation point, $(\dot{u}_0, \dot{\lambda}_0)$. Since we need only a way to get away from the current branch we use a crude approximation to $(\dot{u}_{new}, \dot{\lambda}_{new})$, namely a vector $(\frac{du}{ds}, \frac{d\lambda}{ds})$ in the two dimensional solution space of equation (5.1) orthogonal to $(\dot{u}_0, \dot{\lambda}_0)$. This vector is given by

$$\begin{bmatrix} f_u(u_0, \lambda_0) & f_\lambda(u_0, \lambda_0) \\ \dot{u}_0^T & \dot{\lambda}_0 \end{bmatrix} \begin{bmatrix} \frac{du}{ds} \\ \frac{d\lambda}{ds} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

To obtain the solution of this system, the equations

$$f_u(u_0,\lambda_0)z = 0,$$
$$f_u(u_0,\lambda_0)y = f_\lambda(u_0,\lambda_0)$$

are solved. The vector z is the eigenvector corresponding to the eigenvalue 0. If this eigenvector is not known at the bifurcation point, it can be obtained with inverse iteration. The vector orthogonal to $(\dot{u}_0, \dot{\lambda}_0)$ is given by

$$(\frac{du}{ds}, \frac{d\lambda}{ds}) = (z - \frac{d\lambda}{ds}y, -\frac{\dot{u}_0^T z}{\dot{\lambda}_0 - \dot{u}_0^T y}).$$

Now, the Euler predictor step can be performed with $(\dot{u}_{new}, \dot{\lambda}_{new}) = (\frac{du}{ds}, \frac{d\lambda}{ds})$.

5.2.4 Stability of stationary solutions

The continuation method can follow stable as well as unstable branches. Because only the stable solutions are physically relevant, it is important to know whether a solution is stable or not. We will denote a stationary solution by $\hat{u}$ and use

$$\hat{u} \text{ is stable if } \lim_{t\to\infty} u(t) = \hat{u} \text{ for all solutions } u(t) \text{ with } u(0) \text{ close to } \hat{u}.$$

To determine the stability of a solution, we recall that the stationary solutions are solutions of the differential equation

$$B\dot{u}(t) = f(u(t), \lambda).$$

After linearizing f around $\hat{u}$, this equation can be written as

$$B\dot{u}(t) = f(\hat{u}, \hat{\lambda}) + \frac{\partial f}{\partial u}(\hat{u}, \hat{\lambda})\ (u(t) - \hat{u}).$$

Assuming $u(t) = \hat{u} + e^{\mu t}w$ and using that $f(\hat{u}, \hat{\lambda}) = 0$, because $\hat{u}$ is a stationary solution of the differential equation, leads to the generalized eigenvalue problem

$$\mu B w = \frac{\partial f}{\partial u}(\hat{u}, \hat{\lambda})w. \tag{5.2}$$

When the matrix B is nonsingular, which is not the case in our application, this is equivalent to a normal eigenvalue problem.

The real parts of the eigenvalues μ_i $(i = 1, \ldots, n)$ determine the stability of the stationary solution as follows:

$$(i)\ \ \mathrm{Re}(\mu_i) < 0\ \ \forall i \quad \Rightarrow \quad \hat{u} \text{ is stable,}$$

$$(ii)\ \ \mathrm{Re}(\mu_k) > 0 \text{ for some } k \quad \Rightarrow \quad \hat{u} \text{ is unstable.}$$

In the next section we will describe the Jacobi-Davidson QZ method which we use to solve the generalized eigenvalue problem (5.2).

5.3 Eigenvalue solver

In the continuation method both the position of bifurcation points and the stability of solutions are determined. As described in the previous section this can be done by solving a generalized eigenvalue problem

$$\beta Aq = \alpha Bq. \tag{5.3}$$

To solve these generalized eigenvalue problems we will use the Jacobi-Davidson QZ method. Recently, this method has been developed by Sleijpen, van der Vorst and Fokkema. In this section we first explain the Jacobi-Davidson method which is then used in the Jacobi-Davidson QZ method. Our presentation of these methods follows largely the presentation in [20].

5.3.1 Jacobi-Davidson method

Assume an approximate eigenvector $\tilde{q}$ and the corresponding approximate generalized eigenvalue $\langle\tilde{\alpha},\tilde{\beta}\rangle$ of the generalized eigenvalue problem are known.

In each step of the Jacobi-Davidson (JD) method a new approximation $\tilde{q}$ of the eigenvector is selected from a search space span$\{V\}$. This approximate eigenvector $\tilde{q}$ and the corresponding approximate generalized eigenvalue $\langle\tilde{\alpha},\tilde{\beta}\rangle$ are tested with respect to a test space span$\{W\}$:

$$r = \tilde{\beta}A\tilde{q} - \tilde{\alpha}B\tilde{q} \perp \text{span}\{W\}.$$

This results in the following projected generalized eigenvalue problem

$$\tilde{\beta}W^*AVu = \tilde{\alpha}W^*BVu, \tag{5.4}$$

with $\tilde{q} = Vu$. The spaces V and W are of small dimension and hence the projected generalized eigenvalue problem can for instance be solved by the QZ method.

In each step of the JD method the subspaces span$\{V\}$ and span$\{W\}$ are expanded. First compute $\tilde{z} = \nu_0 A\tilde{q} + \mu_0 B\tilde{q}$ and the residual $r = \tilde{\beta}A\tilde{q} - \tilde{\alpha}B\tilde{q}$. Then $\tilde{z}$ and $\tilde{q}$ are normalized such that $\|\tilde{z}\|_2 = \|\tilde{q}\|_2 = 1$. The search space is expanded by the vector v satisfying $v \perp \tilde{q}$ and the correction equation

$$(I - \tilde{z}\tilde{z}^*)(\tilde{\beta}A - \tilde{\alpha}B)(I - \tilde{q}\tilde{q}^*)v = -r.$$

The test space is expanded by the vector w given by $w = \nu_0 A\tilde{v} + \mu_0 B\tilde{v}$. The scalars ν_0 and μ_0 are such that $|\,\nu_0\,|^2 + |\,\mu_0\,|^2 = 1$. We will return later to the choice of these scalars. The vectors v and w are orthogonalized and added to span$\{V\}$ and span$\{W\}$, respectively.

Restart

The projected generalized eigenvalue problem (5.4) is solved with the QZ method. With this method a generalized Schur form is obtained:

$$W^*AVQ = ZS \quad , \quad W^*BVQ = ZT,$$

with Q and Z orthogonal $j \times j$ matrices and S and T $j \times j$ upper triangular matrices. This generalized Schur form will be ordered. This ordering can be used to select the approximations $\langle\tilde{\alpha}, \tilde{\beta}\rangle$ and $\tilde{q}$ and to restart the JD method when the dimensions of the spaces span$\{V\}$ and span$\{W\}$ become too large.

Assume we want an eigenvalue close to some target τ. Then the ordering will be such that

$$|\frac{S(1,1)}{T(1,1)} - \tau| \leq |\frac{S(2,2)}{T(2,2)} - \tau| \leq \ldots \leq |\frac{S(j,j)}{T(j,j)} - \tau|.$$

With this ordering $\langle S(1,1), T(1,1)\rangle$ and the vector $VQ(:,1)$ are the approximations of the eigenvalue closest to the target τ and its corresponding eigenvector. When the dimensions of span$\{V\}$ and span$\{W\}$ extend some value j_{max} the JD method is restarted. The dimensions are reduced to j_{min} by continuing the method with

$$V = VQ(:, 1 : j_{min}) \quad , \quad W = WZ(:, 1 : j_{min}).$$

Choices for the test space

The test space span$\{W\}$ is expanded with the vector $\nu_0 Av + \mu_0 Bv$, where v is the expansion vector of the search space span$\{V\}$. The parameters ν_0 and μ_0 are scaled such that $|\,\nu_0\,|^2 + |\,\mu_0\,|^2 = 1$ and can be chosen in various ways. If $\langle\alpha, \beta\rangle$ is a generalized eigenvalue and q the corresponding eigenvector then $Aq = \alpha z$ and $Bq = \beta z$. By taking

$$\nu_0 = \frac{\bar{\alpha}}{\sqrt{|\alpha|^2 + |\beta|^2}} \quad \text{and} \quad \mu_0 = \frac{\bar{\beta}}{\sqrt{|\alpha|^2 + |\beta|^2}},$$

the quantity $\|\,\nu_0 Aq + \mu_0 Bq\,\|_2 = |\,\nu_0\alpha + \mu_0\beta\,|\,\|\,z\,\|_2$ is maximized. This can be viewed as an attempt to expand the test space optimally in the direction of z.

However, the generalized eigenvalue $\langle\alpha, \beta\rangle$ is not known in advance. Therefore, an option is to take

$$\nu_0 = \frac{\bar{\tau}}{\sqrt{1 + |\tau|^2}} \quad \text{and} \quad \mu_0 = \frac{1}{\sqrt{1 + |\tau|^2}},$$

where τ is the target.

Another variant would be to adapt ν_0 and μ_0, and use the available approximations of the eigenvalues $\tilde{\alpha}$ and $\tilde{\beta}$. This leads to the choice

$$\nu_0 = \frac{\bar{\tilde{\alpha}}}{\sqrt{|\tilde{\alpha}|^2 + |\tilde{\beta}|^2}} \quad \text{and} \quad \mu_0 = \frac{\bar{\tilde{\beta}}}{\sqrt{|\tilde{\alpha}|^2 + |\tilde{\beta}|^2}},$$

The two variants described so far are called the standard Petrov and the variable standard Petrov approach [20].

A third option would be to take

$$\nu_0 = \frac{1}{\sqrt{1 + |\tau|^2}} \quad \text{and} \quad \mu_0 = -\frac{\tau}{\sqrt{1 + |\tau|^2}},$$

This is called the harmonic Petrov approach. In this way the selection of the appropriate approximations of the eigenpair is optimized instead of expanding the test space optimally as happens in the standard Petrov variants.

5.3.2 Jacobi-Davidson QZ method

In the Jacobi-Davidson QZ (JDQZ) method the Jacobi-Davidson method is used to compute a partial generalized Schur form

$$AQ_k = Z_k S_k \quad , \quad BQ_k = Z_k T_k, \tag{5.5}$$

with Q_k and Z_k $n \times k$ orthogonal matrices and S_k and T_k $k \times k$ upper triangular matrices. A generalized eigenvalue $\langle \alpha, \beta \rangle$ of (S, T), which is easy to obtain, is a generalized eigenvalue of (A, B) as well, and if u is an eigenvector of (S, T) then $Q_k u$ is an eigenvector of (A, B).

Suppose a partial Schur form is already known

$$AQ_{k-1} = Z_{k-1} S_{k-1} \quad , \quad BQ_{k-1} = Z_{k-1} T_{k-1}.$$

To expand this Schur form we need vectors q and z that satisfy

$$A \left[\, Q_{k-1} \ \ q \,\right] = \left[\, Z_{k-1} \ \ z \,\right] \begin{bmatrix} S_{k-1} & s \\ 0 & \alpha \end{bmatrix}$$

and

$$B \left[\, Q_{k-1} \ \ q \,\right] = \left[\, Z_{k-1} \ \ z \,\right] \begin{bmatrix} T_{k-1} & t \\ 0 & \beta \end{bmatrix}.$$

The vector q and $\langle \alpha, \beta \rangle$ have to satisfy

$$q \perp Q_{k-1} \quad , \quad (I - Z_{k-1} Z_{k-1}^*)(\beta A - \alpha B)(I - Q_{k-1} Q_{k-1}^*)q = 0.$$

Hence, they satisfy the generalized eigenvalue problem

$$\beta (I - Z_{k-1} Z_{k-1}^*) A (I - Q_{k-1} Q_{k-1}^*) q = \alpha (I - Z_{k-1} Z_{k-1}^*) B (I - Q_{k-1} Q_{k-1}^*) q,$$

which can be solved with the JD method.

The procedure is as follows. Construct orthogonal $n \times j$ matrices V and W satisfying $V^* Q_{k-1} = W^* Z_{k-1} = 0$ and find an approximate generalized eigenvector $\tilde{q}$ in the search space span$\{V\}$ and test with respect to the test space span$\{W\}$. This leads to the projected generalized eigenvalue problem

$$\tilde{\beta} W^* (I - Z_{k-1} Z_{k-1}^*) A (I - Q_{k-1} Q_{k-1}^*) V u = \tilde{\alpha} W^* (I - Z_{k-1} Z_{k-1}^*) B (I - Q_{k-1} Q_{k-1}^*) V u,$$

or equivalently

$$\tilde{\beta} W^* A V u = \tilde{\alpha} W^* B V u, \tag{5.6}$$

with $\tilde{q} = Vu$. This projected eigenvalue problem is solved with the QZ method, which gives a generalized Schur form $W^* A V Q = ZS$ and $W^* B V Q = ZT$. This Schur form is ordered with respect to the target τ. The first column of VQ is the approximate eigenvector $\tilde{q}$.

In each step of the JD process the search space span$\{V\}$ and test space span$\{W\}$ are expanded. First compute the residual $r = (I - Z_{k-1} Z_{k-1}^*)(\tilde{\beta} A - \tilde{\alpha} B)(I - Q_{k-1} Q_{k-1}^*)\tilde{q}$

and $\tilde{z} = \nu_0 A\tilde{q} + \mu_0 B\tilde{q}$, and scale $\tilde{q}$ and $\tilde{z}$ such that $\|q\|_2 = \|\tilde{z}\|_2 = 1$. The search space span$\{V\}$ will be expanded with the vector v satisfying the correction equation

$$\begin{aligned} &Q_{k-1}^* v = 0 \quad , \quad \tilde{q}^* v = 0, \\ &(I - \tilde{z}\tilde{z}^*)(I - Z_{k-1}Z_{k-1}^*)(\tilde{\beta}A - \tilde{\alpha}B)(I - Q_{k-1}Q_{k-1}^*)(I - \tilde{q}\tilde{q}^*)v = -r, \end{aligned} \tag{5.7}$$

and the test space span$\{W\}$ will be expanded by $w = \nu_0 A\tilde{v} + \mu_0 B\tilde{v}$. The vectors v and w are orthogonalized and added to V and W, respectively. This process is continued until the generalized eigenvalue is computed accurately enough. Then $Q_k = [\, Q_{k-1} \;\; \tilde{q}\,]$ and $Z_k = [\, Z_{k-1} \;\; \tilde{z}\,]$.

If we want to expand the Schur form further, the search and test spaces of the JD part have to be adapted. The conditions $V^* Q_k = 0$ and $W^* Z_k = 0$ are not satisfied anymore. The process is continued with $V = VQ(:, 2 : j)$ and $W = WZ(:, 2 : j)$, where Q and Z are the orthogonal matrices obtained with the QZ method applied to the projected eigenvalue problem (5.6).

5.3.3 Preconditioning

The image space of the operator in the correction equation (5.7) differs from its origin space. Therefore, a Krylov subspace method like GMRES [40] or BiCGstab(l) [43] cannot be applied straightforwardly to this equation. Incorporating an (approximate) inverse solves this problem. Let K be an incomplete LU factorization of $A - \tau B$ and denote

$\tilde{Q}_k = [\; Q_{k-1} \;\; \tilde{q} \;]$, the matrix Q_{k-1} expanded by the vector $\tilde{q}$,

$\tilde{Z}_k = [\; Z_{k-1} \;\; \tilde{z} \;]$, the matrix Z_{k-1} expanded by the vector $\tilde{z}$,

$\tilde{Y}_k = K^{-1}\tilde{Z}_k$, the expanded matrix of preconditioned vectors,

$\tilde{H}_k = \tilde{Q}_k^* \tilde{Y}_k$, the projected preconditioner $\tilde{Q}_k^* K^{-1} \tilde{Z}_k$.

Then the preconditioned correction equation can be written as

$$\tilde{Q}_k^* v = 0 \quad \text{and} \quad (I - \tilde{Y}_k \tilde{H}_k^{-1} \tilde{Q}_k^*) K^{-1} (\tilde{\beta}A - \tilde{\alpha}B) v = -\hat{r},$$

with $\hat{r} = (I - \tilde{Y}_k \tilde{H}_k^{-1} \tilde{Q}_k^*) K^{-1} r$. Since $\tilde{Q}_k^* \hat{r} = 0$ the Krylov space generated by the matrix $(I - \tilde{Y}_k \tilde{H}_k^{-1} \tilde{Q}_k^*) K^{-1} (\tilde{\beta}A - \tilde{\alpha}B)$ and $\hat{r}$ is perpendicular to Q_k. Therefore, this matrix can be used in a Krylov subspace method.

5.3.4 JDQZ in a continuation method

In a continuation code we need to determine the stability of the solutions and the bifurcation points of the system. Therefore, we need to compute the eigenvalues of (5.2). The eigenvalues of interest are those crossing the imaginary axis. In case of a real bifurcation point these eigenvalues are going through zero.

In each continuation step we solve the generalized eigenvalue problem (5.2). Assume we want to compute k_{max} eigenvalues and assume at a certain continuation step we have the generalized Schur form (5.5) with $k = k_{max}$. At the next continuation step the new Jacobian will not differ much from the old one. Therefore, the eigenvalues and eigenvectors will not differ much from those of the previous continuation step as well. We can use the information of the JDQZ method from the previous continuation step to compute the new eigenvalues. The idea is to use this information to obtain a better search space span$\{V\}$ in order to improve the convergence of the JDQZ method. We will consider three variants of starting the construction of the new search space of the JDQZ method at the new continuation step. The first variant is to start the search space with a random vector. The second variant is to take as the first vector of the search space the first Schur vector, i.e. the first column of the matrix $Q_{k_{max}}$, computed in the previous continuation step. The third variant is to start with all k_{max} Schur vectors, that is all columns of $Q_{k_{max}}$, obtained in the previous continuation step.

5.4 Rayleigh-Bénard problem

In order to test the performance of the MRILU factorization in the continuation code, we consider the Rayleigh-Bénard problem [15]. A liquid layer in a rectangular box of aspect ratio 10 is heated from below. The temperature at the top and bottom of the box is constant, the sidewalls are isolated and at all walls the flow satisfies the no-slip condition. The horizontal and vertical velocity are denoted by u and w respectively, the pressure by p and the temperature by T. The governing equations are given by

$$\begin{aligned}
&\frac{1}{\mathrm{Pr}}(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + w\frac{\partial u}{\partial z}) = -\frac{\partial p}{\partial x} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial z^2},\\
&\frac{1}{\mathrm{Pr}}(\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + w\frac{\partial w}{\partial z}) = -\frac{\partial p}{\partial z} + \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial z^2} + \mathrm{Ra}\ T,\\
&\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0,\\
&\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + w\frac{\partial T}{\partial z} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial z^2},
\end{aligned} \tag{5.8}$$

with boundary conditions

$$\begin{aligned}
u = w = 0,&\quad T = 1 \quad \text{at} \quad z = 0,\\
u = w = 0,&\quad T = 0 \quad \text{at} \quad z = 1,\\
u = w = 0,&\quad T_x = 0 \quad \text{at} \quad x = 0, 10,
\end{aligned}$$

where Pr is the Prandtl number and Ra the Rayleigh number. In our calculations Pr=5.5 and Ra is used as the continuation parameter. We are interested in the steady solutions. For all Rayleigh numbers the trivial, motionless solution ($u = w = 0,\ T = 1 - z$) is a solution of the steady partial differential equations. For Ra above some critical value other flow patterns can occur.

The time-independent equations are discretized on a staggered grid. The convective terms are discretized with a central scheme and the diffusive terms with a second-order central scheme. With the mapping

$$y_s = \frac{1}{2}(1 + \tanh(s(y_u - \frac{1}{2}))/\tanh(\frac{s}{2}))$$

a uniform grid y_u can be stretched to obtain a non-equidistant grid y_s. This stretching can be used in both the x- and z-direction, the stretching factors in these directions are denoted by s_x and s_z respectively.

In the remainder of this section we will first determine which grid gives accurate results and compute the bifurcation diagram for that grid. Then we will discuss the effect of the MRILU preconditioner on the performance of the linear solver. Finally, we will consider the performance of the JDQZ method with MRILU as preconditioner for solving the arising generalized eigenvalue problems.

5.4.1 Bifurcation results

The temperature difference between the top and bottom wall causes buoyancy forces. For low Rayleigh numbers the diffusive forces will dominate and the only solution is the motionless solution. When the Rayleigh number is increased at some point the buoyancy forces will dominate the diffusive forces causing the fluid to flow. Bifurcation points on the branch of motionless solutions are called primary bifurcation points.

At the first and second primary bifurcation point a branch of solutions with a 10- and 9-cell flow pattern, respectively, will emerge. In Table 5.1 these two bifurcation points are shown for several grids. From the position of these bifurcation points we will determine which grid is best suited for these computations. The buoyancy forces are caused by the temperature difference between the bottom and top wall of the box. Hence, refining and stretching the grid in z-direction will have more influence than refining and stretching in x-direction. From Table 5.1 we observe that for a fixed number of grid points in z-direction and for fixed stretching factors s_x and s_z the number of grid points in x-direction has hardly any influence on the position of the bifurcation points: for 256 and 128 grid points in x-direction these positions are almost similar. With extrapolation the values of the first two primary bifurcation points for a 128 × ∞ grid can be obtained; these values are 1728.1 and 1732.2. The position of the bifurcation points on a 128 × 32 grid with stretching in z-direction differ about 1 percent with the extrapolated values, and hence are accurate enough. Stretching in x-direction does hardly change the position of the bifurcation points. But, because the changes in number of cells in the solution will be in x-direction we will use stretching in this direction as well. Therefore, the remainder of the calculations will be performed on a 128 × 32 grid with stretching in both directions.

N_x	N_z	s_x	s_z	Ra_1	Ra_2	s_x	s_z	Ra_1	Ra_2	s_x	s_z	Ra_1	Ra_2
128	16	1	1	1694.6	1698.0	1	3	1735.4	1738.5	3	3	1735.2	1738.4
128	32	1	1	1719.5	1723.4	1	3	1730.0	1733.8	3	3	1729.8	1733.7
128	64	1	1	1726.0	1730.1	1	3	1728.6	1732.7	3	3	1728.5	1732.6
256	16	1	1	1695.1	1698.0	1	3	1735.9	1738.6	3	3	1735.9	1738.5
256	32	1	1	1720.0	1723.5	1	3	1730.5	1733.9	3	3	1730.4	1733.9

Table 5.1: The first two bifurcation points for several grids.

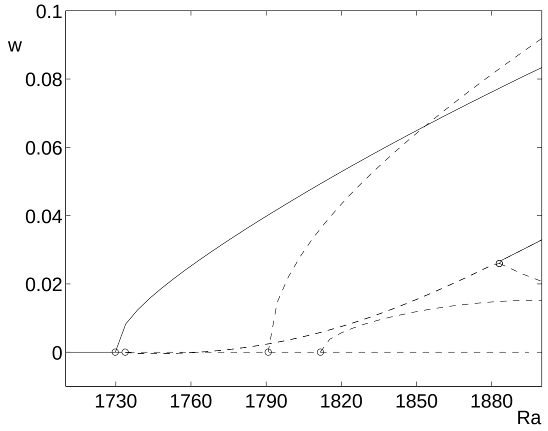

Figure 5.1: Bifurcation diagram for Pr=5.5, solid curves indicate stable states, dashed curves unstable states and circles bifurcation points.

For a 128×32 grid with stretching in both directions the bifurcation diagram is given in Figure 5.1. On the vertical axis the vertical velocity at grid point $(2, 24)$ is plotted. At the first primary bifurcation point (Ra=1729.8) the motionless solution becomes unstable and a stable solution with a 10-cell flow pattern branches off. At the second primary bifurcation point (Ra=1733.7) a solution with 9 cells branches off, this solution is unstable up to the secondary bifurcation point Ra=1883.1, and stable for higher Ra numbers. At this secondary bifurcation point an unstable branch appears. This branch consists of solutions with an asymmetric flow pattern where a new cell develops near the left wall of the box. At the third and fourth primary bifurcation point (Ra=1790.8 and Ra=1811.6) unstable branches of 11- and 12-cell solutions, respectively, branch off. In Figure 5.2 various flow patterns are shown.

5.4.2 MRILU in a linear solver

In this subsection we consider the part of the continuation method which computes the solutions on the branches. This part consists of an Euler prediction method and a Newton method. Assuming a solution is known on a branch, the Euler prediction gives an approximation to the next solution at distance Δs on that branch. This approximation is used as initial vector in the Newton method. To solve the linear systems occurring in the Newton method we make use of the Bi-CGSTAB method [49] preconditioned with an MRILU factorization. We will look at the influence of the step size Δs and the accuracy of the MRILU factorization on the performance of the continuation method.

In all calculations the Newton method is stopped when the updates of the solution and parameter satisfy:

$$\max(\|\Delta u\|_\infty, |\Delta\lambda|) \leq 10^{-6}.$$

The linear systems $Ax = b$ occurring in the Newton process are solved with the Bi-CGSTAB method preconditioned with an MRILU factorization. The linear solver is

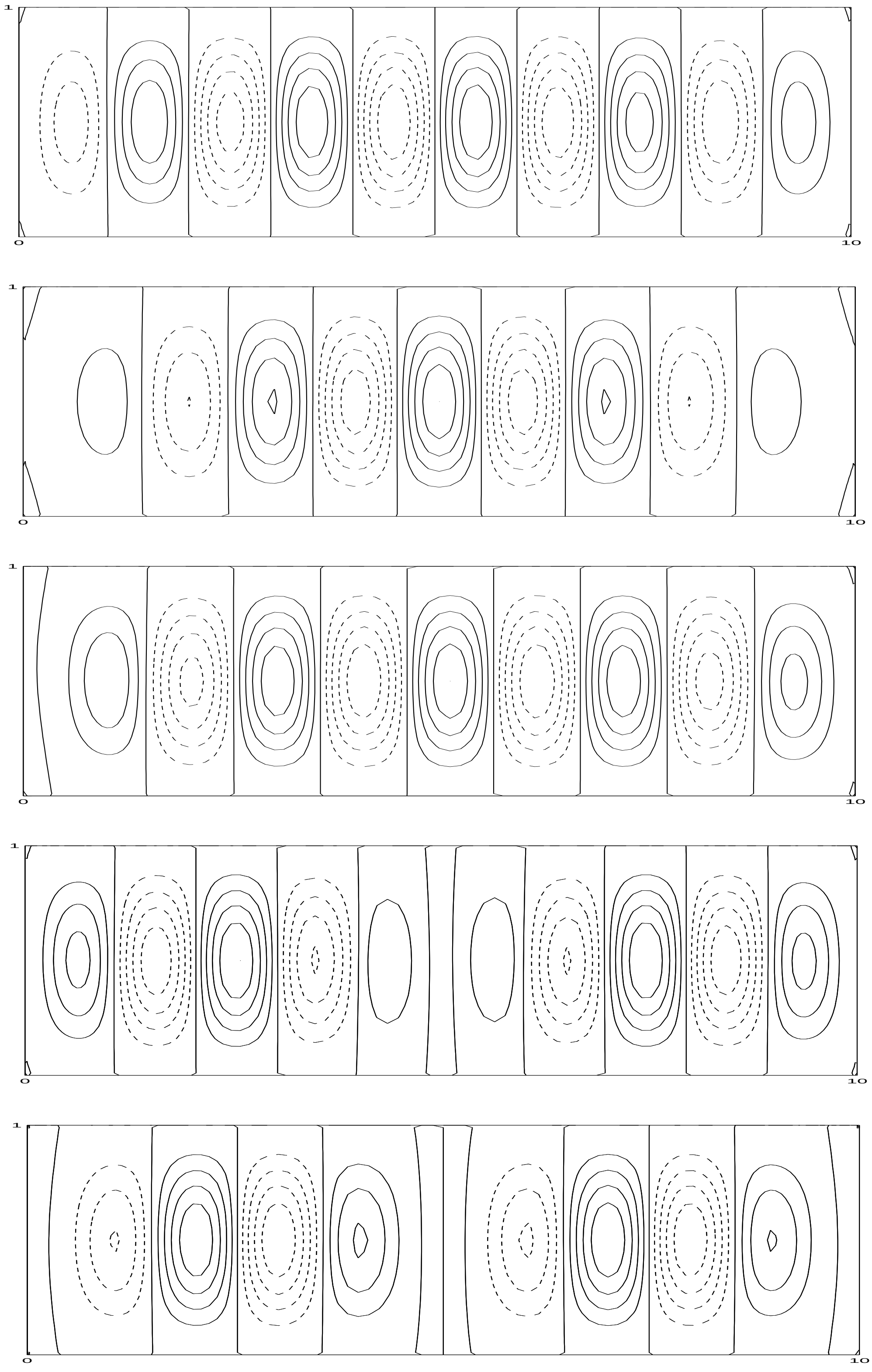

Figure 5.2: Various flow patterns, from top to bottom a stable 10-cell, an unstable 9-cell, an unstable asymmetric 10-cell, an unstable 11-cell and an unstable 12-cell solution.

applied until the preconditioned residual is reduced with three digits:

$$\|P^{-1}(Ax_i - b)\|_2 \leq 10^{-3}\|P^{-1}(Ax_0 - b)\|_2,$$

where P is the MRILU factorization and i the iteration count of the Bi-CGSTAB method.

Convergence on a branch

We will look at the convergence of the Newton method and the preconditioned Bi-CGSTAB method when computing solutions on the branches avoiding the vicinity of bifurcation points. In this case the occurring systems are nonsingular.

In Table 5.2 the results for computing solutions on the branch of ten-cell solutions are shown. For different step sizes Δs and different MRILU factorizations the number of Newton iterations to compute one solution, the number of Bi-CGSTAB iterations to solve one linear system occurring in the Newton method, the cpu-time needed for the construction of an MRILU factorization, the cpu-time needed for solving one linear system and the total cpu-time needed for computing one solution on a branch are given. In the first step of the construction of the MRILU factorization a quarter of the unknowns is eliminated exactly. The fill per row given in Table 5.2 is the fill of the factorization of the reduced system. The iterative method is applied to the reduced system.

	fill per row	Newton it.	Bi-CGSTAB it.	cpu-time MRILU	cpu-time Bi-CGSTAB	cpu-time cont. step
$\Delta s = 5$	49	3	34	6.5	5.4	44.1
$\Delta s = 10$	49	4	33	6.4	5.3	57.7
$\Delta s = 5$	70	3	23	8.3	4.5	46.1
$\Delta s = 10$	70	4	21	8.2	4.2	59.0
$\Delta s = 5$	76	3	17	9.1	3.6	46.6
$\Delta s = 10$	76	4	16	9.0	3.4	58.5

Table 5.2: The performance of the Newton method and the preconditioned Bi-CGSTAB method at one continuation step for different step sizes Δs and different fills in the MRILU factorizations.

First, we will look at the influence of the step size Δs on the convergence. The convergence of the Newton method is quadratic. When a smaller step size is used the Euler predictor, which is used as starting vector in the Newton method, is a better approximation to the next solution on the branch. Indeed, from Table 5.2 we see that with step size $\Delta s = 5$ three Newton iterations are needed to compute one solution on the branch, whereas for step size $\Delta s = 10$ four Newton iterations are needed. From Table 5.2 it is seen that the step size has no influence on the performance of the preconditioned Bi-CGSTAB method used to solve the linear system within the Newton method: the cpu-time for constructing the MRILU factorization and solving a linear system is equal for both the step sizes. Because the number of Newton iterations is higher for $\Delta s = 10$, the total cpu-time needed for one continuation step, i.e. computing one solution on the branch, will be higher when $\Delta s = 10$. With step size $\Delta s = 10$ a part of a branch is traced with half the number of continuation steps as with step size $\Delta s = 5$. The cpu-time needed for one continuation step is only about a factor 1.3 higher when $\Delta s = 10$. When choosing the step size, one has to find a balance between the number of continuation steps needed to trace a branch and the cpu-time needed for one continuation step. Furthermore, the step size has to be chosen such that all bifurcation points are found and a smooth bifurcation diagram can be drawn.

Next, we will consider the influence of the accuracy of the MRILU factorization on the convergence of the Bi-CGSTAB method. We have considered MRILU factorizations with different fills per row. The different factorizations were constructed with 8 levels. As threshold in the ILU factorization of the last level we used 5.10^{-3}, 10^{-3} and 5.10^{-4}, resulting in a fill per row of the complete factorization of 49, 70 and 76, respectively. The MRILU factorization with a higher fill per row is more accurate, as can be seen in Table 5.2: the number of iterations and the cpu-time needed for the preconditioned Bi-CGSTAB method is lower. The construction of the MRILU factorization is more expensive when the fill per row is higher. From the table we can see that the sum of the cpu-time needed for the preconditioned Bi-CGSTAB method and for the MRILU factorization is almost the same for the different MRILU factorizations. This implies that the accuracy of the preconditioner is not of crucial influence. But, when more fill is allowed in the factorization more storage capacity is demanded. Therefore, it is advisable to use a preconditioner with a lower fill per row.

In each continuation step the starting vector in the Newton method is already a good approximation to the solution. Therefore, the Jacobian will not change much during the Newton process. Hence, the MRILU factorization constructed in the first Newton step can be used in the complete Newton process, this will save a considerable amount of cpu-time. In our computations we have not made use of this feature.

Convergence when switching branches

So far we have considered the convergence on a branch out of the neighbourhood of a bifurcation point. If one is interested in the bifurcation points and one wants to trace the branches emerging from these points as well, the position of and the solution at the bifurcation points have to be determined.

After a bifurcation point is detected by monitoring the eigenvalues, the secant process is used to find its exact position. Only two secant iterations are needed to determine this position. Then the direction orthogonal to the current branch is determined. The JDQZ method which is used to compute the eigenvalues can compute the eigenvectors as well. Therefore, we use the eigenvector to determine this direction.

Once the solution at the bifurcation point and the direction orthogonal to the current branch are known, the continuation method can be started from the bifurcation point in order to compute a solution on the emerging branch. In Table 5.3 the convergence behaviour of the first continuation step starting from the first bifurcation point is shown. At this bifurcation point a solution with a ten-cell flow pattern emerges. As continuation step $\Delta s = 5$ is used.

fill per row	Newton it.	Bi-CGSTAB it.	cpu-time MRILU	cpu-time Bi-CGSTAB	cpu-time cont. step
49	12	41	6.5	6.6	229.3
70	11	31	8.2	6.0	218.2
76	11	25	9.2	5.3	216.5

Table 5.3: The performance of the Newton method and the preconditioned Bi-CGSTAB method for different fills in the MRILU factorizations at a continuation step starting from a bifurcation points with $\Delta s = 5$.

At the bifurcation point the system is singular. Therefore, problems with the convergence can be expected. The number of Newton steps to compute the first solution on the branch with ten-cell flow patterns is significantly higher than the number of Newton steps needed to compute a solution elsewhere on the branch. We observed that the Newton process did not converge optimally: at the beginning of the Newton process the convergence was linear, only when converged close enough to the solution the convergence of the Newton method became quadratic. First of all this is caused by the singularity of the system at the bifurcation point. Secondly, the Euler prediction may be inaccurate because the direction orthogonal to the current branch does not have to be the actual direction of the emerging branch, which will result in a poor starting vector for the Newton method.

The fill per row of the MRILU factorization and the cpu-time needed to construct this preconditioner does not differ from computations elsewhere on the branch. However, the number of Bi-CGSTAB iterations needed has become higher. The construction of the factorization is not affected by the singularity of the system, but the preconditioned Bi-CGSTAB method is.

5.4.3 MRILU in JDQZ

To determine the stability of the solutions and the position of the bifurcation points we need to solve generalized eigenvalue problems of the form

$$Aw = \mu Bw,$$

where A is the Jacobian and B a diagonal matrix incorporating the time-dependency of system (5.8). For the Rayleigh-Bénard problem the matrix B is singular because the mass equation does not depend on time. To solve this generalized eigenvalue problem we use the JDQZ method.

A solution is unstable if at least one of the generalized eigenvalues has real part greater than zero. Because we only consider steady solutions, a bifurcation point occurs when a generalized eigenvalue is zero. Therefore, the generalized eigenvalues of interest are those close to zero. To make sure we compute all eigenvalues with real part greater than zero we take as target in the JDQZ method $\tau = 1$. To determine the first bifurcation points it was sufficient to compute the four eigenvalues closest to this target.

The maximal dimension of the search space span$\{V\}$ is taken as $j_{max} = 20$. When the dimension exceeds j_{max} it is reduced to $j_{min} = 10$. To build the search space span$\{V\}$ in a relatively cheap way the correction equation is solved with GMRES$_1$ until the dimension of the search space is larger than j_{min}. When the dimension becomes larger than j_{min}, the correction equation is solved more accurately with either GMRES$_m$, i.e. full GMRES [40] with a maximum of m steps, or BiCGstab(l) [43]. To expand the test space span$\{W\}$ the harmonic Petrov approach is used.

As stopping criterion for the iterative method used to solve the correction equation we use $\|\tilde{r}_i\|_2 \leq 2^{-j}\|\tilde{r}_0\|_2$, with $\tilde{r}_0$ the initial residual, $\tilde{r}_i$ the residual at the i-th step of the iterative method and j the iteration number for the current eigenvalue approximation in the outer iteration. The outer iteration is stopped when the approximate eigenvalue $\tilde{\mu}$ and its corresponding eigenvector $\tilde{w}$ are accurate enough, $\|A\tilde{w} - \tilde{\mu}B\tilde{w}\|_2 < 10^{-9}|\tilde{\mu}|$.

When solving the correction equation an MRILU factorization of $A - \tau B$, with τ the target, is used as preconditioner. This means that the preconditioner is kept fixed throughout the whole JDQZ method. In [20] a justification of this strategy is given. To obtain convergence when solving the correction equation, the MRILU factorization needs to be very accurate. Therefore, we allow only three levels in the MRILU factorization and use an exact factorization of the last Schur complement. In the first step of the factorization a quarter of the unknowns is eliminated exactly. In the box below the fill per row of the factorization of the reduced system, the total cpu-time needed for the construction of the MRILU factorization and the cpu-time needed for the factorization of the last level are shown.

fill per row	180
cpu-time construction total MRILU factorization	16.6
cpu-time construction LU factorization last level	14.8

At each continuation step this preconditioner is constructed just once and used throughout the whole JDQZ method. Therefore, a lot of effort can be put in the construction of an accurate factorization.

In advance it is not clear whether to use GMRES$_m$ or BiCGstab(l) for solving the preconditioned correction equation. We will compare the results of BiCGstab(2) and GMRES$_{20}$. In addition we will compare the different variants of the JDQZ method, i.e. different ways of starting the construction of the search space span$\{V\}$. The search space is started with a random vector in the first variant (JDQZ1), with the first Schur vector computed in the previous continuation step in the second variant (JDQZ2) and with all four Schur vectors computed in the previous continuation step in the third variant (JDQZ3).

Table 5.4 shows the results of the different JDQZ variants at one continuation step on the branch of unstable nine-cell solutions (results on other branches are similar). The computed eigenvalues are

$$4.84\ 10^{-2}\ ,\ -0.433\ ,\ -0.585\ ,\ -0.981.$$

In the continuation process two different step sizes ($\Delta s = 5$ and $\Delta s = 1$) were used. The table displays the number of matrix-vector multiplications, JDQZ iterations and the cpu-time needed for computing four eigenvalues.

		JDQZ1			JDQZ2			JDQZ3		
		mv	jdqz	cpu	mv	jdqz	cpu	mv	jdqz	cpu
$\Delta s = 5$	GMRES$_{20}$	440	54	531	372	52	457	351	41	401
	BiCGstab(2)	627	47	514	429	47	434	486	42	446
$\Delta s = 1$	GMRES$_{20}$	405	51	509	391	47	473	212	37	313
	BiCGstab(2)	529	45	477	493	41	458	190	30	254

Table 5.4: The performance of the GMRES$_{20}$ and BiCGstab(2) for solving the preconditioned correction equation in different variants of the JDQZ method, in the continuation process different step sizes have been used.

We allowed the BiCGstab(2) method to make maximal 100 matrix-vector multiplications per solve. Therefore, this method can solve the correction equation more accurately than the GMRES$_{20}$ method, which uses maximally 20 matrix-vector multiplications. When the correction equation is solved more accurately, less JDQZ iterations will be needed to compute the eigenvalues. In Table 5.4 this is clearly seen: with the GMRES$_{20}$ method more JDQZ iterations but less matrix-vector multiplications are needed for computing four eigenvalues than with the BiCGstab(2) method.

The computational costs for solving one correction equation will be higher when using the BiCGstab(2) method than when using the GMRES$_{20}$ method because the correction equation is solved more accurately with the BiCGstab(2) method. But, as a consequence of the higher accuracy less JDQZ iterations are needed and hence less correction equations have to be solved. In Table 5.4 we see that for almost all cases the BiCGstab(2) method needs less cpu-time than the GMRES$_{20}$ method. The decrease in cpu-time caused by the decrease of the number of JDQZ iterations is enough to compensate for the increase of cpu-time caused by the increase of the number of matrix-vector multiplications.

From Table 5.4 we see that the cpu-time for the JDQZ3 variant is less than for the other two variants and that the cpu-time for the JDQZ2 method is less than for the JDQZ1 variant. The difference between the cpu-time of the different variants is larger for $\Delta s = 1$ than for $\Delta s = 5$.

To be able to make a good comparison between the different JDQZ variants, in Figure 5.3 the convergence behaviour of the JDQZ variants in the continuation code with step size $\Delta s = 5$ and $\Delta s = 1$ are shown. The BiCGstab(2) method has been used to solve the preconditioned correction equation. The residual of the BiCGstab(2) method when solving the preconditioned correction equation is shown against the number of JDQZ steps. In this way we can observe for different step sizes the influence of using information of the previous continuation step in the JDQZ method.

For $\Delta s = 5$ the differences between the three variants of the JDQZ method are small. Random effects can be the cause of these differences. The Schur vectors from the previous continuation step are not accurate enough approximations of the new Schur vectors to have a positive effect on the convergence of the JDQZ method.

For $\Delta s = 1$ the convergence of the JDQZ method becomes faster when more Schur vectors of the previous continuation step are used to form the search space. With a small

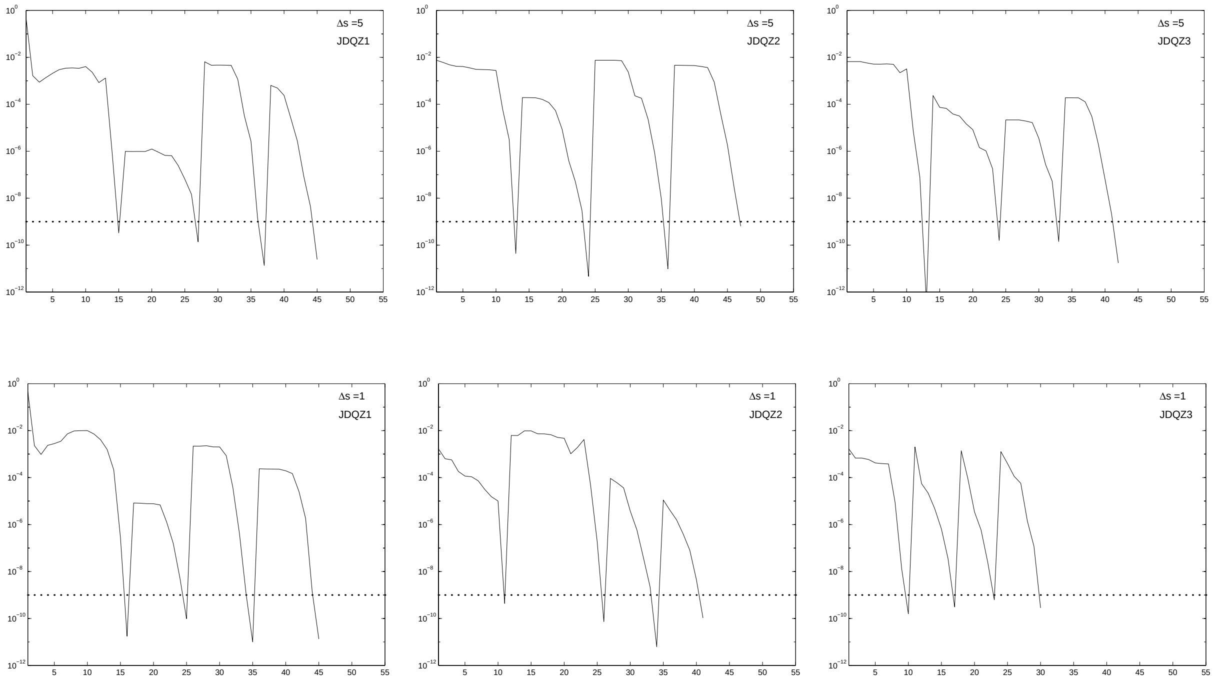

Figure 5.3: Convergence behaviour of the different variants of the JDQZ method for different step sizes in the continuation method.

step size these Schur vectors are accurate approximations of the new Schur vectors and will have a positive effect on the convergence. The JDQZ2 variant converges faster than the JDQZ1 variant when computing the first eigenpair. With the JDQZ2 version the search space is started with the first Schur vector computed in the previous continuation step. Therefore, the search space in the JDQZ2 variant contains more information about the first eigenvector than the search space in the JDQZ1 variant. From the convergence behaviour of the JDQZ2 variant it can be seen that when the JDQZ method converges slowly for one eigenpair, relevant information for the remaining eigenpairs is added to the search space resulting in a fast convergence for these remaining eigenpairs. The JDQZ3 variant converges fast for all eigenpairs. In this variant the search space is started with all four Schur vectors from the previous continuation step. Hence, this search space contains accurate information about all eigenvectors

Concluding, for $\Delta s = 5$ the three JDQZ variants have similar convergence behaviour, whereas for $\Delta s = 1$ the JDQZ3 and JDQZ2 variant converge faster than the JDQZ1 variant. When Δs is small the Schur vectors from the previous continuation step will be a better approximation to those of the current continuation step than when Δs is larger.

5.5 Conclusions

In this chapter we have used a continuation method to compute all solutions of a system of partial differential equations which depends on a parameter. With such a method all branches of solutions can be traced, bifurcation points can be detected and the stability of a solution can be determined. The continuation method can be split in two parts. The first part is the actual continuation part: assuming that at some point on the branch the solution is known a new solution at the next point is computed. An Euler prediction is used to obtain an approximation to the new solution. This approximation is used as starting vector in Newton's method, which is used to compute the new solution. The other part determines the stability of a solution and the position of the bifurcation points by solving a generalized eigenvalue problem. We have used the JDQZ method to solve the generalized eigenvalue problems. In both parts of the continuation method we have used the MRILU factorization: in the Newton method as preconditioner when solving the linear systems and in the JDQZ method to precondition the correction equation.

We have applied the continuation method successfully to the Rayleigh-Bénard problem. On a 128×32 grid with stretching in both directions we have computed a bifurcation diagram. Furthermore, we have considered the convergence of both the linear solver and the eigenvalue solver. The MRILU factorization is an efficient preconditioner for both of these solvers.

When computing a solution on a branch the Newton process converges quadratically. For a smaller step size in the continuation method, the Newton method needs less iterations because for this case the Euler approximation is more accurate. In each Newton step the linear systems are solved with the Bi-CGSTAB method preconditioned with an MRILU factorization. We have compared the performance of MRILU factorizations with a fill per row of 49, 70 and 76. When the fill per row is low the cpu-time needed for the construction of the factorization is low. However, with a low fill the factorization will be less accurate. Therefore, the number of Bi-CGSTAB iterations and hence the cpu-time needed for solving the linear system will be higher. For the different MRILU factorizations the sum of the cpu-time needed for the construction of the factorization and the solve of the linear system is equal. Apparently, when computing a solution on a branch the quality of the preconditioner is not very crucial. But, from the point of view of storage capacity it is advantageous to use a factorization with a low fill per row.

The secant method, which is used to compute the location of a bifurcation point, needed only two steps to converge. Once this location is found, the branch emerging from this bifurcation point can be traced. At a bifurcation point the system is singular. This causes a deterioration of the convergence of the Newton method. In the continuation step starting from a bifurcation point the Newton method converges linearly. The construction of the MRILU factorization is not influenced by the singularity. However, the preconditioned Bi-CGSTAB method needs more iterations near the singularity than elsewhere on the branch.

In the JDQZ method the correction equation is preconditioned with an MRILU factorization. To obtain convergence the factorization needs to be very accurate, resulting in a fill per row of 180. This factorization is made once during the whole JDQZ method, justifying the large effort made to construct the factorization. We have solved the preconditioned correction equation with the BiCGstab(2) method and the GMRES$_{20}$ method. With the BiCGstab(2) method the preconditioned correction equation is solved more accurately (and hence more expensively) than with the GMRES$_{20}$ method. Therefore, the JDQZ method will need less JDQZ iterations when using the BiCGstab(2) method. The cpu-time for the BiCGstab(2) method is lower than for the GMRES$_{20}$ method. The decrease in cpu-time caused by the decrease of the number of JDQZ iterations is enough to compensate for the increase of cpu-time caused by the increase of the number of matrix-vector multiplications.

We have considered three variants to start the search space of the JDQZ method: variant one starts with a random vector, variant two with the first Schur vector from the previous continuation step and variant three with all Schur vectors from the previous continuation step. With the third variant the JDQZ method needs the least iteration steps. It is beneficial to start with a search space that already contains information about the Schur vectors. The results for this variant are even better when a small step size is used in the continuation method: in this case the previous Schur vectors are a better approximation to the new Schur vectors.

When the step size is taken smaller in the continuation method the convergence of both the Newton method and the JDQZ method improves. The cpu-time needed for one continuation step will be lower. But, the total number of continuation steps needed to trace a given part of a branch will be higher. Consequently, the step size should not be taken too small or large.

Chapter 6

Bifurcation analysis of driven cavity flow by the Newton-Picard method

6.1 Introduction

In this chapter we will study the transition from steady to (quasi) periodic flow. The flows under consideration are governed by the incompressible Navier-Stokes equations. Such studies are not only interesting from a theoretical point of view, but from a practical one as well. For example, consider a flow around an object which becomes periodic at a certain Reynolds number. At that stage the flow will exert a periodic force on the object, which may give such strong vibrations that, due to fatigue of the material, the object will eventually get damaged. In such a case one could change the form of the object in order to improve its aerodynamic properties such that the periodic behaviour is delayed to a much higher Reynolds number which does not or only rarely occur in practice.

The study on the transition will be done for the lid-driven cavity. This problem is a well known test case for numerical methods for solving the Navier-Stokes equations. Nevertheless, little is known about its bifurcation behaviour. Cazemier et al. [12] have performed a bifurcation analysis for the lid-driven cavity using a low-dimensional dynamical system resulting from a Galerkin projection of the Navier-Stokes equations on a basis obtained by a proper orthogonal decomposition (or singular value decomposition) of snapshots of the flow on a very fine grid (see Section 6.2.2). The low-dimensional system admits to compute the monodromy matrix, the order of which is equal to the number of degrees of freedom in the system, whose eigenvalues determine the bifurcation behaviour. They observed that the behaviour of the low-dimensional dynamical system predicted the behaviour of the high-dimensional system on the very fine grid reasonably well.

For our bifurcation analysis we have used the software tool PDECONT, which has been developed by Lust and Roose [32]. This tool is based on the Newton-Picard method which in itself is a generalization of the RPM of Shroff and Keller [42]. In this method the monodromy matrix restricted to the space consisting of the most critical modes is constructed. Therefore, the tool allows to study the bifurcation behaviour of the high-dimensional system itself. In the tool the user has to specify the right-hand side and the flow, i.e. a map which gives the time-dependent solution of the PDE for a given time, Reynolds number and initial solution. We have computed this flow by using an implicit time-integration method where MRILU (Chapter 4 and [8]) has been applied to solve the occurring linearized systems. Due to limited computer resources we have used a somewhat coarser grid than Cazemier et al. Nevertheless, the results are very similar.

6.2 Problem description and results of others

In this section we will describe the problem and summarize the information about its bifurcation behaviour found by others.

6.2.1 Lid-driven cavity

The problem under consideration is the lid-driven cavity problem; the flow in a square cavity with constant moving lid. The domain and boundary conditions of this problem are shown in Figure 6.1.

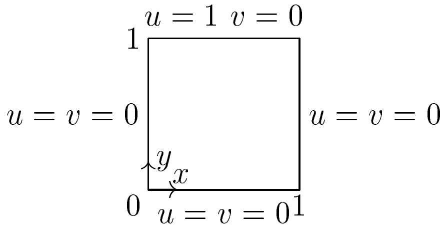

Figure 6.1: Geometry for the lid-driven cavity problem.

The flow is described by the two-dimensional incompressible Navier-Stokes equations, which in conservation form are given by

$$\begin{aligned}
&\int_\Gamma \vec{u}.n \ \mathrm{d}\Gamma = 0,\\
&\int_\Omega \frac{\partial u}{\partial t} \ \mathrm{d}\Omega + \int_\Gamma (u\vec{u} - \nu \nabla u).n \ \mathrm{d}\Gamma = -\int_\Gamma p(n.e_1) \ \mathrm{d}\Gamma,\\
&\int_\Omega \frac{\partial v}{\partial t} \ \mathrm{d}\Omega + \int_\Gamma (v\vec{u} - \nu \nabla v).n \ \mathrm{d}\Gamma = -\int_\Gamma p(n.e_2) \ \mathrm{d}\Gamma,
\end{aligned}$$

for arbitrary domain Ω with boundary Γ.

6.2.2 Results by Cazemier

The flow in a lid-driven cavity was also studied by Cazemier et al. [11, 12]. In their work a direct numerical simulation is performed at Reynolds number 22,000 on a 250×250 grid (about 200,000 unknowns) with a fourth-order finite-volume discretization. This Reynolds number is far beyond the transition point from steady to periodic flow; the flow is already chaotic. From this simulation 700 snapshots are taken at constant time intervals (5 sec). By a proper orthogonal decomposition (POD), or in linear algebra terms a singular value

f	0.27	0.44	0.11	0.16	0.32	0.53	0.70	0.06	0.38	0.60	0.88

Table 6.1: Occurring frequencies (Hz) in DNS calculation at Re=10,000.

decomposition, of these snapshots an orthogonal basis is constructed. They showed that a basis of dimension 80 is enough to capture 95% of the energy in the system. Then, these 80 basis functions are used for a Galerkin approximation of the Navier-Stokes equations. The thus derived low-dimensional dynamical system is used to study the bifurcation behaviour. At certain Reynolds numbers the results of this dynamical system were verified by performing a DNS. The results were qualitatively similar, but quantitatively different.

The DNS results in [11] yield some important data to compare our results with. First, the transition from a stationary to a periodic flow is observed at Re=7972, the emerging period T is 2.2 seconds. Secondly, at Re=10,000 a plot is given of the power spectral density of the time signal. We clearly see a frequency 0.44 Hz (about the reciprocal of the period 2.2) in the plot. In Table 6.1 the most important frequencies, as good as we can read it from the plot, in that signal are listed in order of significance. The signal contains already a lot of frequencies, among which there will be combination frequencies. In our experiments (see Section 6.5) we will see that for this Reynolds number the steady state solution has already 5 unstable modes. Many of the occurring frequencies shown above can be found as combination frequencies of these modes.

The analysis with the low-dimensional dynamical system showed at Re=7819 a transition from a stationary to a periodic flow with time period 1.63 seconds, which corresponds to a frequency of about 0.60 Hz. This frequency is quite different from the frequency 0.44 Hz obtained from the DNS, which is explained in Cazemier's thesis [11] from the fact that the POD basis has been obtained at Re=22,000. However, our experiments also indicate that the frequency 0.60 Hz is rather important.

6.2.3 Results by Poliashenko and Aidun

Poliashenko and Aidun described in [38] a direct method to compute a simple bifurcation. They solve an augmented system which give the critical parameter value at once, avoiding a continuation process. Similar techniques are discussed in [1]. This direct method was applied to compute the first bifurcation of the flow in the lid-driven cavity. Based on the results on some grids they concluded that this critical value is at $7763 \pm 2\%$. By using Richardson extrapolation on the results of their finest two grids we come to the value 7952 which is quite close to the value 7972 found by Cazemier et al. It surprised us that on a 57×57 grid using biquadratic finite elements such an accurate determination of the bifurcation point is possible.

Another interesting result they obtained is that for a square cavity the first (Hopf) bifurcation is supercritical, which means that the solution is unique before the first bifurcation. They show a result of a non-square cavity with height-width ratio 1.5 where this is not the case; next to the steady state there are a stable and an unstable periodic solution.

6.3 Discretization

In general a parameter dependent autonomous partial differential equation (PDE) is considered. With the method of lines this PDE is first discretized in space, resulting in a large system of ODEs

$$\begin{aligned} \frac{dx}{dt} &= f(x, \gamma) \quad t > 0, \\ x(0) &= x_0, \end{aligned}$$

with $x \in R^n$ and γ the parameter. To solve this system and perform a bifurcation analysis we use the software tool PDECONT. This tool requires from the user two subroutines: one which computes the right-hand side $f(x, \gamma)$ and one which computes the flow $\varphi(x_0, t, \gamma)$, i.e. the solution of the PDE at time t for a given initial solution x_0. To compute this flow we employ as time integration the θ-method. In the following we treat, according to the method of lines, the discretization of the Navier-Stokes equations in space and time separately.

6.3.1 Space-discretization

In order to perform an accurate bifurcation analysis, we wish to preserve the stability of the Navier-Stokes equations when discretizing the system. In order to obtain this, the convective terms have to be discretized skew-symmetric [53]. In the continuous case convection corresponds to a skew-symmetric and diffusion to a symmetric positive definite operator. When preserving these properties in the discretized case, the resulting coefficient matrix will be positive definite. In [53] it is shown that if after semi-discretization the coefficient matrix is positive definite the energy of the solution is decreasing in time. This implies that the semi-discretized system is stable. We will use the symmetry-preserving finite-volume discretization as described in Section 4.3.1.

The starting point in this discretization is a central discretization of the convective terms, but by adding artificial diffusion to the real diffusion an upwind discretization can be obtained. In our computations we want to eliminate the influence of artificial diffusion on the results. Hence, we set it equal to zero. We will use a 128 × 128 grid, and refine the grid near the walls in order to prevent unphysical wiggles. This refinement is determined by the mapping

$$y_s = \frac{1}{2}(1 + \tanh(2s(y_u - \frac{1}{2}))/\tanh(s)).$$

This expression maps grid points y_u from a uniform grid to those of a stretched grid y_s. As stretching factor we use $s = 1.5$ in both x- and y-direction.

6.3.2 Time-discretization

For the time-discretization we used the θ-method, which is given by

$$x^{n+1} = x^n + \Delta t f((1-\theta)x^n + \theta x^{n+1}),$$

with $\theta \in [0, 1]$. For $\theta = 1/2$ and $\theta = 1$ this is the implicit midpoint and backward Euler method, respectively. The choice of θ in the application is delicate and will be considered separately below. As initial guess for the solution at time step $n + 1$ we used the extrapolation formula $x^{n+1} = 2x^n - x^{n-1}$. The implicit relations are solved by the Newton method and the thereby occurring linear equations by MRILU. In the Newton-Picard method we are computing perturbations of a steady state over a certain time period. It appeared that those perturbations are so small that one MRILU factorization for the whole time period sufficed.

Choice of θ

In the time-continuous case the high-frequency components are highly damped, and components with eigenvalues with a large imaginary part and a small positive real part are unstable. When using a time discretization, for instance the θ-method, we wish to preserve these properties. We will consider the influence of the choice of θ on the damping of the different components.

Suppose we want to study the effect of perturbations of a steady state over a time period T with n equal time steps, hence $\Delta t = T/n$. Then, for each eigenvalue λ of the Jacobian matrix of f we have the following amplification factor r

$$r = [(1 + (1 - \theta)\lambda\Delta t)/(1 - \theta\lambda\Delta t)]^n. \tag{6.1}$$

The Newton-Picard method, discussed below, computes the most dominant r's, i.e. those r's which are in magnitude larger than a certain user specified value. We want that the corresponding dominant subspace contains the same perturbations as those dominating in the time-continuous case. So on the one hand we want $\theta = 1/2$, since then the important eigenvalues near the imaginary axis are not damped too strongly, avoiding that they will not be treated as dominant by the Newton-Picard method and consequently will become invisible. On the other hand we want θ larger than $1/2$ in order to damp the λ's large in magnitude. So we try to find θ close to $1/2$ such that for λ's large in magnitude the magnitudes of the corresponding r's are less than those corresponding to λ's close to the imaginary axes. For large $\lambda\Delta t$ we have from (6.1) the approximation

$$|r| \approx [(1 - \theta)/\theta]^n \approx \exp[-4(\theta - 1/2)n]. \tag{6.2}$$

We will clarify the above with a numerical example. Assume we wish to compute all r's which are in magnitude larger than 0.95. In our computations the number of time steps we used is about 10 to 20. For 10 time steps and $\theta = 0.51$ we find for λ's large in magnitude $|r| \approx 0.7$, which is only slightly below 0.95 and therefore a further decrease of θ is not desirable, since this would increase r. For eigenvalues with imaginary part of about 2 to 4 and close to the imaginary axis we observed in our computations an influence of the time discretization on the damping when $\theta = 0.51$. Hence, on this ground we would like to decrease θ. We see that these considerations for the choice of θ are in conflict. When the time step is halved n is doubled and we are allowed (see (6.2)) to decrease θ to 0.505 in order to keep the same damping behaviour for the high-frequency components.

Computation of eigenvalues from r

At this place we discuss an application of equation (6.1) which can be used to compute where the stationary solutions of the semi-discretized equations become unstable. As mentioned before the Newton-Picard method will compute the value of r. From equation (6.1) we can compute the corresponding eigenvalue λ. Due to the n-th root that has to be taken λ is multi-valued. Only one of these values is the sought eigenvalue. This true eigenvalue is of course unchanged under perturbation of the time period T. The others values do change under such a perturbation. To see this, consider the expression

$$r = \exp(\lambda T),$$

which is the continuous variant of equation (6.1) which we use here to simplify the analysis. Now, for the true eigenvalue $\hat{\lambda}$ we have

$$\frac{dr}{dT} = \hat{\lambda} \exp(\hat{\lambda} T), \tag{6.3}$$

which we will observe after a perturbation of T. The pseudo eigenvalues mathematically yield after a perturbation of T

$$\frac{dr}{dT} = T \exp(\lambda T) \frac{d\lambda}{dT} + \lambda \exp(\lambda T).$$

When we equate this to the observed one (6.3) we find

$$\frac{d\lambda}{dT} = (\hat{\lambda} \exp[(\hat{\lambda} - \lambda)T] - \lambda)/T.$$

So in general $\frac{d\lambda}{dT}$ differs from zero if λ differs from $\hat{\lambda}$ and hence we can detect $\hat{\lambda}$ by changing T a little for a fixed parameter (the Reynolds number). The most constant value among the solutions is the sought eigenvalue.

This can be used during the continuation process as well. For the steady state we see that the Newton-Picard algorithm also changes T. By picking the most constant value in the set of computed values for λ we can identify the eigenvalue.

6.4 PDECONT

In this section the so-called Newton-Picard method which is at the basis of PDECONT will be described in short (see [32] for an extensive treatment). The underlying idea of the method can already be found in [27, 42], where it is applied to steady computations. Shroff and Keller called their method the recursive projection method (RPM).

To introduce the underlying idea we confine ourselves to the steady case. If we use a time-integration method, eventually the error is dominated by a set of slowly decaying (periodic) components. It is possible to make a basis for this set of slowly decaying components which forms an invariant subspace of limited dimension. The update in this invariant subspace and its orthogonal complement will be computed separately. In the former Newton's method is used and in the latter the time integration is still employed.

To explain the Newton-Picard method we begin with a description of the RPM, in order to point out the main ideas.

6.4.1 RPM

Suppose we have a fixed-point problem

$$x = g(x).$$

One may think here of a steady-state computation by means of a time-integration method over some period, or of detecting a periodic solution of a PDE. The fixed-point iteration $x_{n+1} = g(x_n)$ may converge slowly or even diverge when the eigenvalues of the Jacobian of g are in magnitude close to 1 or larger than 1.

To stabilize this procedure the space will be split in two orthogonal subspaces $\mathcal{P}$ and $\mathcal{Q}$ with projectors P and $Q = I - P$. The subspace $\mathcal{P}$ is of low dimension, and corresponds to the slowly converging and unstable modes. Let $p_0 = Px_0$ and $q_0 = Qx_0$, then we iterate according to

$$\begin{aligned} p_{i+1} &= p_i + C_1(Pg(x_i) - p_i), \\ q_{i+1} &= q_i + C_2(Qg(x_i) - q_i), \\ x_{i+1} &= p_{i+1} + q_{i+1}. \end{aligned}$$

Here, C_1 and C_2 should be chosen such that good convergence is obtained in both subspaces. In the paper of Shroff and Keller [42] C_2 is the identity matrix, yielding a Picard iteration in $\mathcal{Q}$, and C_1 is minus the inverse of the Jacobian of $Pg(x_i) - p_i$,

$$-(\frac{\partial(Pg(p_i + q_i) - p_i)}{\partial p_i})^{-1},$$

resulting in the Newton method in $\mathcal{P}$.

In the iteration we need the projectors P and Q. Let the columns of V be the orthogonal vectors spanning $\mathcal{P}$. Then the projectors are given by $P = VV^T$ and $Q = I - P = I - VV^T$. In the actual computations we do not work with p but with its representation $\hat{p}$ in the basis V, every $p \in \mathcal{P}$ can be written as $p = V\hat{p}$. Hence, we can rewrite the p update equation as

$$V\hat{p}_{i+1} = V\hat{p}_i + C_1(VV^Tg(V\hat{p}_i + q_i) - V\hat{p}_i).$$

Premultiplying this equation by V^T yields

$$\hat{p}_{i+1} = \hat{p}_i + V^TC_1V(V^Tg(V\hat{p}_i + q_i) - \hat{p}_i).$$

In order to obtain a good convergence in the subspace $\mathcal{P}$ we zero the Jacobian of the right-hand side in this formulation. This gives

$$V^TC_1V = (I - V^Tg_xV)^{-1}.$$

Here, the directional derivatives g_xV_i where V_i is the i-th column of V can be found by numerical differentiation

$$g_x(x_n)V_i = (g(x_n + \varepsilon V_i) - g(x_n))/\varepsilon. \tag{6.4}$$

Remark In the periodic case the matrix $V^T g_x V$ is an approximation to the monodromy matrix restricted to $\mathcal{P}$.

We are left with the problem of finding V. Its columns build the invariant subspace $\mathcal{P}$ which usually is the space corresponding to the dominant eigenvalues of g_x. V can be found in various ways. We will return to this problem in the next section.

Note that the presented iteration is in general more contractive than the original fixed-point problem. Even if the fixed-point problem is mildly unstable, i.e. only a few eigenvalues of g_x are in magnitude larger than one, then the RPM is converging if the eigenvectors corresponding to those unstable eigenvalues are in $\mathcal{P}$.

6.4.2 Newton-Picard method

Lust and Roose reconsidered and generalized the RPM for periodic problems starting from a linear algebra point of view and called their method the Newton-Picard method. We will describe the part of the algorithm that we have used in our computations. For the other features we refer to [32, 31]. An outline of the algorithm including a continuation process is given in Algorithm 6.1. In the remainder of this section we will clarify the main part of this algorithm.

```
Given an initial guess for a solution at a given parameter
  1. Compute the solution accurately
     (a) Construct an invariant subspace
     (b) Compute the correction
     (c) Check convergence
     (d) If not converged go to (a)
  2. Increment the parameter
  3. Generate a guess for the solution at the new parameter value
  4. Go to 1 or stop if goal is reached
```

Algorithm 6.1: Newton-Picard method.

Construction of the invariant subspace

The invariant subspace is found by a variant of orthogonal iteration with Ritz acceleration (see [21, page 423]) or subspace iteration with projection [39]. The iteration process is given by Algorithm 6.2. Here, M stands for the monodromy matrix or in terms of the previous section g_x. If Steps 2 to 4 are omitted and $V = W$, we just have the standard orthogonal iteration process (a generalization of the Power method). The effect of these

```
Given some $V_0$ with $V_0^T V_0 = I$ and $M$
Set $k = 1$ and perform the following steps

  1. Apply $M$ to $V_{k-1}$: $W_k = MV_{k-1}$

  2. Compute the restriction of $M$ to the space spanned by $V_{k-1}$:
     $M_k = V_{k-1}^T W_k (\equiv V_{k-1}^T M V_{k-1})$

  3. Convert $M_k$ to its real Schur normal form where the eigenvalues
     on the diagonal of R are in decreasing order of magnitude:
     $U^T M_k U = R$

  4. Combine columns in $W_k$ such that they correspond to the order
     of the eigenvalues: $V_k = W_k U$

  5. Orthogonalize $V_k$

  6. Increment $k$ and go to 1 or stop if converged
```

Algorithm 6.2: Construction of invariant subspace.

steps is the following. They construct a basis for the subspace such that the eigenvector (eigenspace) corresponding to the eigenvalue (eigenvalue pair) largest in magnitude is on the first position, repeating itself for the remaining eigenvalues. This is an acceleration of the orthogonal iteration where eventually also the eigenvectors will appear in the desired order.

In the standard acceleration [21, page 423] Step 2 is replaced by first orthogonalizing W, which allows to omit Step 5, and then computing $M_k = \tilde{W}_k^T M \tilde{W}_k$. This requires a new application of M, which will slow down the construction by almost a factor two. The difference is that in the algorithm used the reordering is based on the old V, this difference has only a limited effect on the convergence. Eventually, when the respective subspaces are converging, U tends to the identity matrix, and the eigenvalues on the diagonal of R, i.e. the eigenvalues of the monodromy matrix M restricted to the invariant subspace, are the amplification factors mentioned in subsection 6.3.2.

Convergence behaviour In exact arithmetic we could, by omitting the orthogonalization, simplify the construction to the iteration $V_k = MV_{k-1}$, requiring only that the distance [21, page 76] of V_0 and the dominant eigenspace is less than one. If d is the dimension of the space spanned by V_k then the i-th Ritz value converges as $(\lambda_{d+1}/\lambda_i)^k$. Here the Ritz value can be obtained by orthogonalizing V_k, restricting M to that space as in Step 2, and computing the eigenvalues of that restriction in decreasing order (the i-th eigenvalue of this matrix is the i-th Ritz value).

In order to get the same result in the presence of rounding errors, V_k has to be orthogonalized now and then (usually at each step), resulting in orthogonal iteration. In orthogonal iteration the eigenvectors will converge in order of decreasing magnitude of the eigenvalues. In order to accelerate the convergence of the eigenvectors, one recombines the columns in V_k when the Ritz values are computed (here at every step), yielding the same convergence in the eigenvectors as in the Ritz values. For more considerations we refer to [39, Ch.5].

Parallelization The computation of MV is well parallelizable. Since this is the most time consuming operation of the code this is very attractive. V consists of the order of 30 columns. So almost a factor 30 can be gained, since the work is strongly balanced.

Stopping criterion and locking For the stopping criterion the matrix

$$(I - VV^T)MV$$

is considered. Here $(I - VV^T)$ is the projector on the orthogonal complement of V. Hence, if V is an invariant subspace then this expression is zero. It is not necessary that all columns of V have converged. The user specifies a lower bound for the magnitude of the eigenvalues that must converge and a tolerance which is used as follows. If the eigenvalues corresponding to the space spanned by the first k columns of V are larger than the lower bound and if the norm of the first k columns of $(I - VV^T)MV$ is less than the tolerance then these k columns are assumed to have converged. Of course, the space V must be taken large enough such that all wanted eigenvectors are in it. A larger space than strictly necessary is even beneficial as we have seen in the paragraph on the convergence behaviour.

As we have seen the eigenvectors converge in order, this can be exploited by freezing (or locking) a converged vector. In this case Step 1 of Algorithm 6.2 is only performed for the not-converged part. The other steps are still applied to all vectors in V.

Remark We did not make use of the ability of the program to add and delete vectors in V. In the case of adding, random vectors are employed. We observe an algebraic effect in the convergence of such a vector which delays the overall convergence considerably. In our view first a number of iterations on new (random) vectors have to be done while the old ones are locked. In the course of our study we got sufficient information on how to choose the dimension of the space spanned by the columns of V.

Computation of the correction

In order to describe the computation of the correction we rewrite our problem. Let us describe the solution of the system in terms of the flow φ: $x(t) = \varphi(x(0), t, \gamma)$. So the solution depends on the initial solution and the parameter γ. If a periodic solution exists then the flow must have a fixed point

$$x = \varphi(x, T, \gamma). \tag{6.5}$$

Note that the fixed point is not unique, if x is a fixed point then $\varphi(x, t, \gamma)$ is one as well. The solution is often made unique by imposing a so-called phase condition, which we write formally as

$$s(x, T, \gamma) = 0. \tag{6.6}$$

Remark By substituting $\varphi(x,t,\gamma)$ in the fixed-point equation and taking the derivative with respect to t we see immediately that $\varphi_t(x,t,\gamma)$ is an eigenvector of $\varphi_x(\varphi(x,t,\gamma),T,\gamma)$ with eigenvalue 1. For $t=0$, this means we have the eigenvector $\varphi_t(x,0,\gamma)$ and the matrix $\varphi_x(x,T,\gamma)$. In the latter case the eigenvector is equal to $f(x,\gamma)$. We will use this result later on.

Of course, one likes to solve the above two coupled equations (6.5) and (6.6) with the Newton method. The Newton-correction equation is given by

$$\begin{bmatrix} \varphi_x - I & \varphi_T \\ s_x & s_T \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta T \end{bmatrix} = - \begin{bmatrix} r \\ s \end{bmatrix},$$

where r is the residual of the fixed-point equation. Here, φ_x is an approximation to the so-called monodromy matrix, therefore it is usually denoted by M. As explained in the remark above, φ_T can be approximated by $f(x,\gamma)$. The matrix φ_x is full, and hence, for large systems, the Newton method will be too expensive.

To obtain a rapidly linearly converging method ideas from the RPM are used. Write Δx as $\Delta x = V\Delta\hat{p} + \Delta q$, and note that $\Delta q = Q\Delta q$ where Q is the orthogonal projector on the complement of span$\{V\}$, $Q = I - VV^T$. Herewith, the correction equation is rewritten as

$$\begin{bmatrix} Q(M-I)Q & Q(M-I)V & Q\varphi_T \\ V^T(M-I)Q & V^T(M-I)V & V^T\varphi_T \\ s_xQ & s_xV & s_T \end{bmatrix} \begin{bmatrix} \Delta q \\ \Delta\hat{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} Qr \\ V^Tr \\ s \end{bmatrix}.$$

If V is an invariant subspace of M then $Q(M-I)V = 0$, we will assume that this is the case. Moreover, we will assume that $Q\varphi_T$ is negligible; for the periodic solution φ_T is an eigenvector of M with eigenvalue 1 (see the remark above) and hence a vector of the subspace $\mathcal{P}$, therefore $Q\varphi_T$ will be small during the Newton process. This results in a Gauss-Seidel like iteration. First we have to solve for Δq and then the other unknowns follow from the reduced system

$$\begin{bmatrix} V^T(M-I)V & V^T\varphi_T \\ s_xV & s_T \end{bmatrix} \begin{bmatrix} \Delta\hat{p} \\ \Delta T \end{bmatrix} = - \begin{bmatrix} V^Tr + V^TM\Delta q \\ s + s_x\Delta q \end{bmatrix}. \tag{6.7}$$

In order to solve for Δq we need to solve a system with the matrix $Q(M-I)Q$. This forms the bottle-neck in the computations since we do not want to compute this full matrix explicitly. However, we can compute a multiplication of M with a vector by numerical differentiation as in (6.4). Hence we can try to approximate the inverse of $Q(M-I)Q$ by a polynomial. In the present version a truncated Neumann series is used

$$[Q(M-I)Q]^+ \approx Q + QMQ + (QMQ)^2 + ... + (QMQ)^k, \tag{6.8}$$

where + denotes the pseudo-inverse; this yields exactly the inverse of $M - I$ restricted to the orthogonal complement of V. The value of k is specified by the user (in our applications we have used $k=10$ and $k=20$).

Once the reduced system is computed, its solution can be found by a direct method since the system is small (in our case of order 30).

Automatic phase condition through pseudo-inverse In order to avoid giving a phase condition a variant based on least-squares is also available. This variant has been used in our computations. Here, the reduced system to be solved is

$$\left[\begin{array}{cc} V^T(M-I)V & V^T\varphi_T \end{array}\right]\left[\begin{array}{c} \Delta\hat{p} \\ \Delta T \end{array}\right] = -\left[\begin{array}{c} V^Tr + V^TM\Delta q \end{array}\right]. \tag{6.9}$$

This system is underdetermined. Hence, the solution is not unique. A way to make it unique is to find the solution with minimal two-norm. This solution is found by employing the pseudo-inverse, which implicitly defines a phase condition. We will make the form of this condition more explicit in the following.

In case of a periodic solution φ_T is the eigenvector of M with eigenvalue 1. In general this is a simple eigenvalue and due to the construction in the Newton-Picard process it is contained in the invariant subspace V. Hence, if $[\,\overline{\Delta\hat{p}},\ \overline{\Delta T}\,]^T$ is a solution of the underdetermined system then $[\,\overline{\Delta\hat{p}}+\mu V^T\varphi_T, \overline{\Delta T}\,]^T$ is a solution as well. Minimization of the length of this vector leads to the condition that the inner product $(V^T\varphi_T, \overline{\Delta\hat{p}}+\mu V^T\varphi_T) = 0$. Or in other words, the unique solution of the system has to be perpendicular to $[V^T\varphi_T, 0]^T$. This is the implicit phase condition when the system is solved using the pseudo-inverse. In [31, 32] it is shown that this condition is close to a desirable one.

We have also used the code on the steady branch. In that case, $V^T(M-I)V$ is in general non-singular, but $\varphi_T = 0$ and hence $V^T\varphi_T = 0$. Then the solution with minimal two-norm is the one with $\Delta T = 0$, which is precisely what we want.

As usual in the application of the pseudo-inverse the user has to specify a drop tolerance for the singular values. Here, a singular value is dropped if it is less than the drop tolerance times the maximum singular value.

Stopping criterion The stopping criterion of the Newton-Picard process is based on the residual and the corrections. More precise, the root mean square of both the residual, Δp and Δq and the magnitude of ΔT should be less than a user specified tolerance.

Pseudo-arclength method In continuation often pseudo-arclength parametrization is used in order to be able to pass through a fold point. Then, an equation has to be added to the system. When the pseudo-arclength parameter is denoted by μ, this equation is given by

$$n(x, T, \gamma, \mu) = 0.$$

The precise form is given in formula (2.1) of Lust and Roose. In our computations we did not use this pseudo-arclength variant.

6.5 Numerical results

The PDECONT tool can be used to compute both steady and periodic flows and their stability. For the lid-driven cavity problem depending on the Reynolds number both types of flows can occur. In this section we will present results from computations on both a steady and a periodic branch. All computations have been performed on a 128 × 128 stretched grid.

6.5.1 Along the steady solution branch

As a starting solution for the Newton-Picard process we computed the steady-state solution at Re=7500. This was done by a separate program using the backward Euler method ($\theta = 1$).

Setting of the parameters in PDECONT

The tool PDECONT is a cascade of iterative procedures, we will shortly indicate the criteria used for each procedure.

Continuation process In the continuation process the Reynolds number is used as the continuation parameter. The process is started at Re=7500 with step size 250. During the process the step size is automatically adjusted. The smallest and largest step size we allow are 25 and 500, respectively.

Subspace iteration During the computation we keep the dimension of span$\{V\}$ fixed at 22. For the invariant subspace we require that the stopping criterion is satisfied with tolerance 0.01 for all eigenvalues greater than 0.95. The parameter used in the numerical differentiation is 10^{-5}. This is small but necessary for two reasons. First of all, we obviously want that the error is small. Secondly, we have to deal with the non-linearity in case of a complex eigenvalue. To be more precise, suppose that a certain eigenvalue is complex then the two eigenvectors span a two-dimensional space. In the algorithm we iterate on two real orthogonal vectors which approximately span this space. Application of the nonlinear matrix M gives as result two new real vectors which are orthogonalized in order to give the new basis. Since M is nonlinear the respective spaces built in this way will not converge when the parameter in the numerical differentiation is too large. The criterion in the time stepping process is also very important in this respect.

Time integration We set the time step at 0.125. This is adapted by the program such that its multiple fits in the period. For the Newton method a stopping criterion of 10^{-12} is imposed on the correction (maximum norm). This is needed for the numerical differentiation in order to have a sufficient small error in MV. Since the periods we are computing are of order 1 about ten time steps are needed yielding an error in the integration over one period of about 10^{-11}. Due to the numerical differentiation this error will become $10^{-6}(= 10^{-11}/10^{-5})$. We experimented with this stopping criterion and observed that this high accuracy was needed to maintain convergence of the eigenvalues. We think that also the non-normality of M will play an important role here.

The problem to find the magnitude of variation in the numerical differentiation is absent when using the differential equation for the perturbation, using the exact Jacobian. We are inclined to do this in future experiments. We need only half of the accuracy for the time integration. So it is equally expensive.

Solving correction equation The value of k in the Neumann series (6.8) in the Picard step is set to 10. For the computation of the pseudo-inverse of the reduced system (6.9) the drop tolerance for the singular values is 10^{-4}. The tolerance for the stopping criterion of the Newton-Picard process is 0.005.

Computation of eigenvalue data

In Table 6.2 eigenvalues and corresponding periods and frequencies are given. In each column one can see how an eigenvalue evolves as a function of the Reynolds number. A plot of this data is given in Figure 6.2. These eigenvalues are computed from equation (6.1) as discussed in Section 6.3.2. Hence, the eigenvalues are independent of θ. The results indicated with † in the table are not fully converged. These results usually belong to an eigenvalue with a large imaginary part for which the amplification factor (see Section 6.3.2) is not in the range of accurately computed amplification factors.

Re	λ_1	T_{est}	$1/T_{est}$	λ_2	T_{est}	$1/T_{est}$
7500	-0.0159± 0.9406i	6.6799	0.1497			
7750	-0.0152± 0.9398i	6.6858	0.1496	-0.0163± 2.7813i	2.2590	0.4427
8000	-0.0144± 0.9392i	6.6897	0.1495	-0.0068± 2.7762i	2.2633	0.4418
8375	-0.0138± 0.9371i	6.7049	0.1491	0.0009± 2.7640i	2.2732	0.4399
8875	-0.0130± 0.9357i	6.7152	0.1489	0.0189± 2.7518i	2.2833	0.4380
9375	-0.0124± 0.9345i	6.7233	0.1487	0.0313± 2.7354i	2.2970	0.4353
9875	-0.0115± 0.9322i	6.7403	0.1484	0.0473± 2.7247i	2.3060	0.4337
10375	-0.0110± 0.9330i	6.7346	0.1485	0.0608± 2.7148i	2.3144	0.4321
Re	λ_3	T_{est}	$1/T_{est}$	λ_4	T_{est}	$1/T_{est}$
7750	-0.0245± 1.8672i	3.3650	0.2972	-0.0391± 0.9355i	6.7166	0.1489
8000	-0.0268± 1.8667i	3.3660	0.2971	-0.0375± 0.9347i	6.7222	0.1488
8375	-0.0241± 1.8632i	3.3723	0.2965	-0.0360± 0.9335i	6.7310	0.1486
8875	-0.0223± 1.8598i	3.3784	0.2960	-0.0336± 0.9312i	6.7471	0.1482
9375	-0.0220± 1.8571i	3.3833	0.2956	-0.0314± 0.9301i	6.7556	0.1480
9875	-0.0214± 1.8545i	3.3881	0.2952	-0.0296± 0.9289i	6.7641	0.1478
10375	-0.0194± 1.8489i	3.3983	0.2943	-0.0274± 0.9264i	6.7822	0.1474
Re	λ_5	T_{est}	$1/T_{est}$	λ_6	T_{est}	$1/T_{est}$
8375	-0.0312± 3.7956i†	1.6554	0.6041			
8875	0.0275± 3.8058i	1.6509	0.6057	-0.0151± 3.3252i†	1.8896	0.5292
9375	0.0583± 3.7903i	1.6577	0.6032	0.0259± 3.2788i	1.9163	0.5218
9875	0.0871± 3.7818i	1.6614	0.6019	0.0490± 3.2702i	1.9214	0.5205
10375	0.1136± 3.7755i	1.6642	0.6009	0.0717± 3.2565i	1.9294	0.5183
Re	λ_7	T_{est}	$1/T_{est}$	λ_8	T_{est}	$1/T_{est}$
8875	-0.0348± 4.4752i†	1.4040	0.7123			
9375	0.0303± 4.3597i	1.4412	0.6939			
9875	0.0640± 4.3418i	1.4471	0.6910	-0.0120± 4.8817i‡	1.2871	0.7769
10375	0.0952± 4.3249i	1.4528	0.6883	0.0291± 4.8370i	1.2990	0.7698

Table 6.2: Eigenvalues and corresponding estimated periods and frequencies.

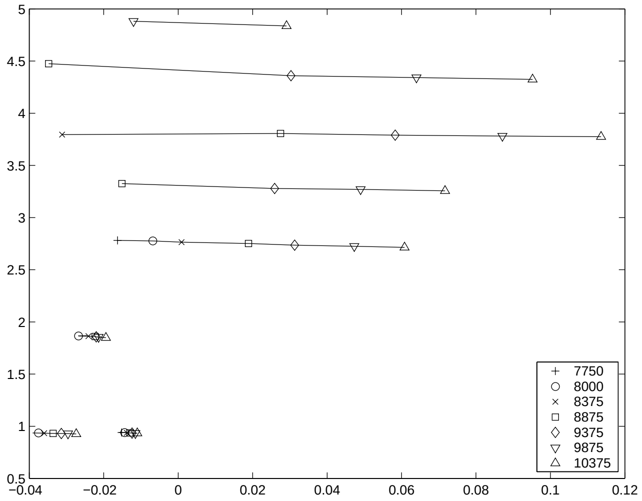

Figure 6.2: Eigenvalues in the complex plane for various Reynolds numbers.

We verified the results by requiring two extra digits of accuracy for the solution at Re=9875. It appeared that the maximal difference in the eigenvalues was less than 14e-4. Hence the above results are rather accurate. We also halved the time step and decreased θ to 0.505. Now, we observed the result indicated with a ‡ in the table. This component is unstable for Re=10,375 as we can see from the last result in the table. Such a component with a large imaginary part is indeed not seen if θ is too large.

Comparison to Cazemier's results

From Table 6.2 we see that in our computations the first bifurcation occurs in the neighbourhood of Re=8375. The difference with Cazemier's DNS value 7972 is due to our coarser grid and lower-order discretization.

In Cazemier's analysis with the low-dimensional dynamical system the frequency 0.60 Hz appeared at the first transition point, whereas with both Cazemier's DNS and our analysis the frequency 0.44 Hz occurred. In our results the frequency 0.60 Hz seems to be connected to the eigenvalue which causes the second Hopf bifurcation and has the largest real part for the higher Reynolds numbers. Hence, it does not surprise us that this frequency plays a prominent role.

Comparing our results for Reynolds number 10,000 to those of Table 6.1 we observe many common components, the remaining frequencies in that table can almost always be formed by a combination of the frequencies that we have found. To be more precise, from Table 6.2 we find for Re=10,000 the unstable modes with frequencies 0.43, 0.52, 0.60, 0.69 Hz which correspond to the frequencies 0.44, 0.53, 0.60, 0,70 Hz in Table 6.1. These are the frequencies of the Hopf bifurcations, hence the others must be integer combinations of them. In Table 6.3 we give a possible realization of the remaining frequencies in the same order as in Table 6.1. The most important frequency in Table 6.1 is 0.27 Hz, which has a large amplitude in Cazemier's DNS computations and which was believed to originate from a Hopf bifurcation. From our computations we find that it is a combination frequency.

This is confirmed by the computation along the periodic solution branch starting from the period corresponding to the frequency 0.44 Hz, see Section 6.5.2. The first unstable mode on this branch is the one corresponding to the frequency 0.70 Hz.

f	reconstr.
0.27	0.70-0.44
0.44	Hopf
0.11	0.70-0.60
0.16	0.60-0.44 or 0.70-0.53
0.32	2(0.60-0.44)
0.53	Hopf
0.70	Hopf
0.06	0.60-0.53
0.38	(0.70-0.44)+(0.70-0.60)
0.60	Hopf
0.88	2(0.44)

Table 6.3: Reconstruction attempt of unstable modes.

6.5.2 Along periodic solution branches

As mentioned before the first bifurcation in our computations occurs in the neighbourhood of Re=8375. This is determined by computing the eigenvalues of the Jacobian with equation (6.1), and looking when they pass the imaginary axis. However, if we want to switch to the periodic solution branch, we have to consider the transition of the time-integration process, i.e. to determine where the amplifier r leaves the unit circle. This may shift the transition point, because the eigenvalues do not depend on the time-integration method whereas the amplification factor r does.

We increased the accuracy of the time integration by using $\theta = 0.502$ and $\Delta t = 0.06125$. The continuation process was started again at Re=8000 with step size 100. We increased the dimension of the invariant subspace to 30 and used $k = 20$ in the truncated Neumann series (6.8). At the start of this extension we see an algebraic effect on the convergence behaviour. After a while the amplification factors of the additional vectors settle down at about 0.78 which complies with equation (6.2) which gives approximately 0.74. This is quite close to our region of interest.

We again observed that the steady solution becomes unstable at Re=8400 for a perturbation with period about 2.3 seconds. We started on the periodic branch at Re=8550 using the steady solution plus 0.5 times the first vector in the basis, i.e. the vector belonging to λ_2. The factor was found after some trials (0.05 still returned the steady solution). A plot of how the Floquet multipliers move is shown in Figure 6.3. The marks in this plot make it possible to follow the Floquet multipliers for increasing Reynolds numbers; they follow two consecutive sequences of $\circ, \times, +, *, \square, \diamond$. All occurring multipliers can be traced back to the components given in Table 6.2. The radius of the Floquet multipliers is scaled in order to stretch the plot near the unit circle. On the dash-dotted and dashed line the radius is 0.90 and 0.95, respectively.

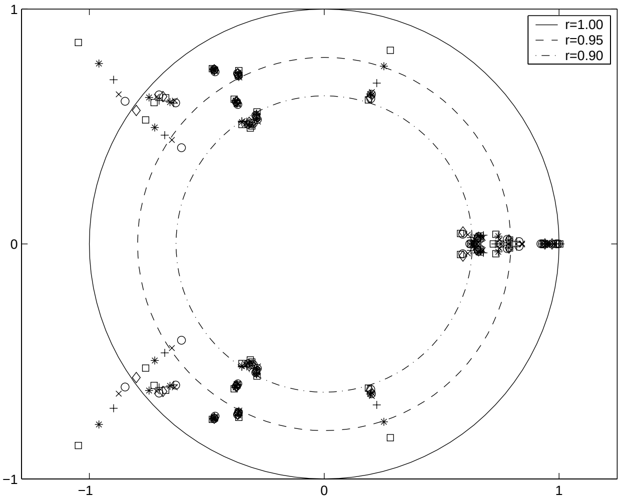

Figure 6.3: Floquet multipliers of the periodic solution emerging at the first Hopf bifurcation for Reynolds numbers in the range 8550-10,000. The radius of the values is scaled with 100^{r-1}.

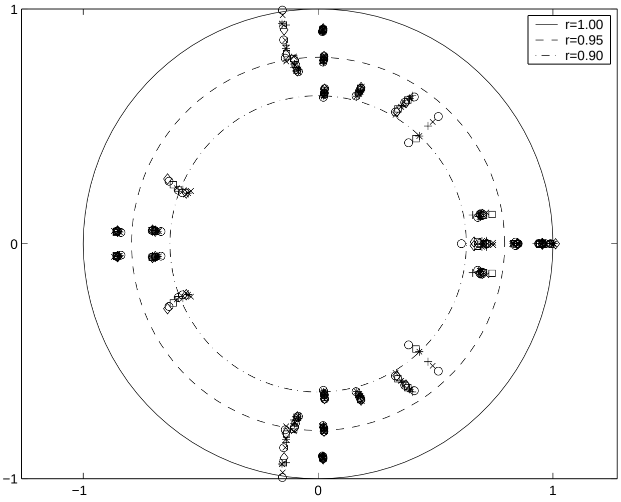

Figure 6.4: Floquet multipliers of the periodic solution emerging at the second Hopf bifurcation for Reynolds numbers in the range 8700-10,000. The radius of the values is scaled with 100^{r-1}.

From Figure 6.3 we observe that at Re=9150 a pair of complex conjugate Floquet multipliers leaves the unit disc and hence the periodic solution becomes unstable. The frequency of the unstable mode is about 0.70. This is again computed with equation (6.1), which holds approximately now. This component is clearly generated from λ_7 in Table 6.2. Since a pair of complex conjugate Floquet multipliers has left the unit disc a 2-periodic or quasi-periodic solution will emerge, we can not compute such solutions

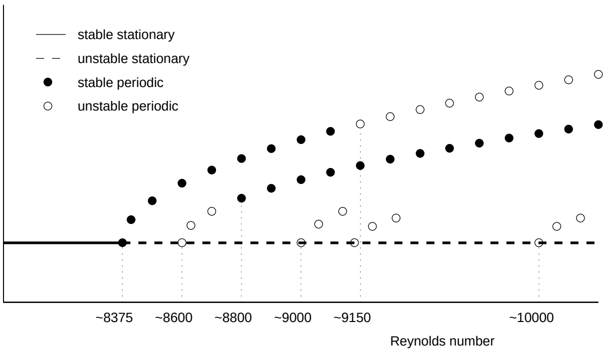

Figure 6.5: Bifurcation diagram.

with our method. From Re=9250 we followed the unstable periodic solution branch with increments 250 until we reached Re=10,000. No new unstable modes occur.

To compute the periodic branch emerging from the second bifurcation point at the steady-state branch, we started the computation at Re=8700 using the steady solution plus 0.5 times the vector corresponding to the eigenvalue λ_5 which causes the bifurcation. The Reynolds number is increased with step size 100 until Re=10000. The movement of the Floquet multipliers is shown in Figure 6.4.

At this second bifurcation point the steady-state solution remains unstable. The periodic solution has period 1.7 seconds, and from the Floquet multipliers in Figure 6.4 we see that the emerging periodic solution is unstable. However, we see that at about Reynolds number 8800 a pair of complex conjugate Floquet multipliers reenters the unit disc and hence at that Reynolds number the periodic solution becomes stable and a 2-periodic or quasi-periodic solution will disappear. The periodic solution remains stable for the remainder of the computation to Re=10000. Since the DNS results from Cazemier show a stable periodic solution with frequency 0.6 Hz. (period 1.7 seconds) at Re=12000 we think that in between this periodic solution is stable as well.

6.5.3 Bifurcation diagram

In Figure 6.5 a partial bifurcation diagram based on our results is shown. This picture only gives a qualitative behaviour of the system. We see that when the stationary solution becomes unstable at Reynolds number 8375 a periodic branch occurs. This periodic branch is stable up to a Reynolds number of about 9150. At Reynolds number of about 8600 a second periodic branch emerges from the steady-state branch. This periodic branch is unstable up to Reynolds number of about 8800, where it becomes stable. We observe that for Reynolds numbers in the range from 8800 to 9150 two stable periodic solutions exist. From the unstable stationary branch other periodic branches occur at Reynolds numbers of about 8600, 9000, 9100 and 10,000. These branches are all (initially) unstable.

6.6 Conclusions

We split our conclusions in two parts: the numerical approach and the bifurcation behaviour. With respect to the numerical approach we mention:

- Large scale bifurcation analysis of periodic solutions is possible. However, the current version of PDECONT consumes a lot of computer time, which is due to the slow convergence of the invariant subspace.

- The use of an implicit method is possible with good iterative solvers, such as CG type methods preconditioned by an MRILU factorization. Special care has to be taken that only the Floquet multipliers of interest are found.

- The space discretization is chosen such that it remains stable if no artificial viscosity is used, this is an important reason why we find bifurcations at low Reynolds numbers.

With respect to the bifurcation behaviour on the range Re=7500 to 10,000 we find the following:

- The first Hopf bifurcation occurs with our model at about Re=8375. This is a bit higher than the value of Cazemier Re=7972, which we attribute to the fact that our grid is twice as coarse and only second-order accurate discretizations are used.

- The next Hopf bifurcations occur at about Re=8600, 9000, 9100 and 10,000. The frequencies corresponding to the corresponding modes are also encountered on the branches of periodic solutions. They can also be used to explain the time signals found by Cazemier.

- The periodic solution occurring at Re=8375 is stable to about Re=9150. Thereafter, no new unstable modes occur before Re=10,000.

- The periodic solution emerging at Re=8600 is unstable until about Re=8800. After that point it is stable to at least Re=10,000.

Chapter 7

Conclusions

In this thesis we have considered preconditioners based on an incomplete LU factorization, and applied them in solving problems originating from the field of computational fluid dynamics. The flows that we have computed are described by the Navier-Stokes equations. Instead of using the popular pressure-correction approach, we have solved the equations fully coupled. With this approach non-symmetric large sparse systems have to be solved. With the increase of computer power and the development of fast linear solvers this approach has become feasible.

A good preconditioner is essential when solving large sparse systems, originating for instance from the discretization of partial differential equations, iteratively. An important class of preconditioners is formed by the (modified) incomplete LU factorizations ((M)ILU). The quality of these preconditioners strongly depends on the ordering of the unknowns and the dropping strategy employed during the factorization.

In Chapter 2 and 3 we have considered block (M)ILU factorizations with respect to a repeated red-black ordering. The ordering in such factorizations is fixed, and the dropping is based only on the position of the fill and not on the size of the elements. This type of preconditioners is attractive because the construction of the factorization is cheap and can easily be vectorized.

In Chapter 2 we have applied several variants of the block RRB preconditioner to some test cases. We used either an ILU or MILU factorization, of which the last level was factorized exactly or approximated by a block diagonal matrix. The fastest convergence was obtained with the MILU factorization with an exact factorization of the last part. For the Poisson system this method was almost grid independent. However, for the (Navier-)Stokes equations the convergence stagnated when the number of levels was taken too high in this factorization.

In Chapter 3 we have given theoretical estimates for the condition number of the preconditioned system. Furthermore, we have estimated for some test cases the eigenvalues of the preconditioned system, and with these results explained the convergence behaviour observed in Chapter 2.

The factorizations can be improved by using in the MILU factorizations a relaxed modification, and by replacing the diagonal approximation at the last level with a more advanced one. Furthermore, the method can be made more competetive when a time derivative is added to the system. A disadvantage of this method is that the ordering is fixed. Therefore, the grid has to be structured. Another disadvantage is that the dropping is based on the position of the fill and not on its size. The factorization will become worse when the elements of the matrix will vary largely in size.

In Chapter 4 we consider the MRILU factorization. With this method the ordering of the unknowns and the dropping are determined during the factorization, and are both based on the size of the elements in the factorization. This factorization has succesfully been applied in a variety of problems. Because both the ordering and dropping are based on the matrix, the method can handle matrices with a general sparsity, for instance matrices stemming from the discretization on an unstructured grid. For the Poisson problem the method shows grid independent convergence. We have applied the MRILU factorization in solving the Navier-Stokes equations. We observed that the performance of the factorization is better when the diagonal of the system is stronger, which can be obtained by using a suitable discretization for the convection terms or adding a time derivative to the system. In the future the MRILU factorization will be improved further.

Continuation methods are used to perform a bifurcation analysis of a parameter dependent system. Until recently, these methods have only be applied to low-dimensional systems (about 10 degrees of freedom). The bottle-neck in computations on high-dimensional systems (about 10^5 degrees of freedom) is formed by the necessity to solve large linear systems and eigenvalue problems. With the development of fast linear solvers and eigenvalue solvers such computations have become feasible.

In Chapter 5 we have used a pseudo-arclength continuation method to perform a bifurcation analysis of the Rayleigh-Bénard problem. For this problem the solutions are stationary. The MRILU factorization has been used to compute the solutions, and turned out to be very robust. The Jacobi-Davidson QZ method has been used to solve the eigenvalue problem which determines the stability of the solutions and the position of the bifurcation points. This method allows to compute a few eigenvalues near a user specified target, and hence is well suited for application in a continuation method. The MRILU factorization can be used as preconditioner in the JDQZ method.

We have used the Newton-Picard method in Chapter 6 to perform a bifurcation analysis of the lid-driven cavity problem. With this method both stationary and periodic solutions of high-dimensional systems can be computed and their stability determined.

For the time discretization of the Navier-Stokes equations we have used an implicit method, which is possible when a good preconditioner as MRILU is used. We have used a second-order spatial discretization with no artificial diffusion.

On a 128×128 grid we found that the first Hopf bifurcation is at about Re=8375. The next Hopf bifurcations are at about Re=8600, 9000, 9100, 10,000. The frequencies corresponding to the corresponding unstable modes are also encountered on the branch of periodic solutions occurring at Re=8375. This periodic solution is stable up to Re=9150, and no new unstable modes occur before Re=10,000. The periodic branch emerging from the second Hopf bifurcation point at Re=8600 is unstable up to Re=8800, and thereafter stays stable to at least Re=10,000. Our results are in good comparison with results obtained by others.

More accurate results can be obtained when using a higher-order discretization and a finer grid. To make the latter possible the Newton-Picard method has to become faster, especially the convergence of the invariant subspace has to be improved.

Bibliography

[1] E.L. Allgower and H. Schwetlick. A general view of minimally extended systems for simple bifurcation points. *Z. Angew. Math. Mech.*, 77(2):83–97, 1997.

[2] K.E. Atkinson. *An introduction to numerical analysis.* Wiley, second edition, 1989.

[3] O. Axelsson and V. Eijkhout. The nested recursive two-level factorization for nine-point difference matrices. *SIAM J. Sci. Stat. Comput.*, 12:1373–1400, 1990.

[4] O. Axelsson and G. Lindskog. On the eigenvalue distribution of a class of preconditioning methods. *Numer. Math.*, 48:479–498, 1986.

[5] O. Axelsson and N. Munksgaard. Analysis of incomplete factorizations with fixed storage allocation. In D.J. Evans, editor, *Preconditioning methods: analysis and applications*, pages 219–241. Gordon & Breach, New York, 1983.

[6] R.E. Bank and C. Wagner. Multilevel ILU decomposition. *Numer. Math.*, 82:543–576, 1999.

[7] E.F.F. Botta, K. Dekker, Y. Notay, A. van der Ploeg, C. Vuik, F.W. Wubs, and P.M. de Zeeuw. How fast the Laplace equation was solved in 1995. *Appl. Numer. Math.*, 24(4):439–455, 1997.

[8] E.F.F. Botta and F.W. Wubs. Matrix Renumbering ILU: An effective algebraic multi-level ILU-preconditioner for sparse matrices. *SIAM J. Matrix Anal. Appl.*, 20(4):1007–1026, 1999.

[9] C.W. Brand. An incomplete factorization preconditioning using repeated red-black ordering. *Numer. Math.*, 61:433–454, 1992.

[10] N.I. Buleev. A numerical method for the solution of two-dimensional and three-dimensional equations of diffusion. *Math. Sb.*, 51:227–238, 1960.

[11] W. Cazemier. *Proper Orthogonal Decomposition and low dimensional models for turbulent flow.* PhD thesis, University of Groningen, Groningen, The Netherlands, 1997.

[12] W. Cazemier, R.W.C.P. Verstappen, and A.E.P. Veldman. POD and low-dimensional models for driven cavity flows. *Physics of fluids*, 10:1685–1699, 1998.

[13] E.F. D'Azevedo, P.A. Forsyth, and W.P. Tang. Towards a cost-effective ILU preconditioner with high level fill. *BIT*, 32(3):442–463, 1992.

[14] H.A. Dijkstra. *Mass transfer induced convection near gas-liquid interfaces.* PhD thesis, University of Groningen, Groningen, The Netherlands, 1988.

[15] H.A. Dijkstra, M.J. Molemaker, A. van der Ploeg, and E.F.F. Botta. An efficient code to compute non-parallel steady flows and their linear stability. *Comp. Fluids*, 24:415–434, 1995.

[16] I.S. Duff and G.A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT*, 29(4):635–657, 1989.

[17] T. Dupont, R. Kendall, and H. Rachford. An approximate factorization procedure for solving self-adjoint elliptic difference equations. *SIAM J. Numer. Anal.*, 5:559–573, 1968.

[18] S.C. Eisenstat. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM J. Sci. Statist. Comput.*, 2(1):1–4, 1981.

[19] H.C. Elman and G.H. Golub. Line iterative methods for cyclically reduced discrete convection-diffusion problems. *SIAM J. Sci. Statist. Comput.*, 13(1):339–363, 1992.

[20] D.R. Fokkema, G.L.G. Sleijpen, and H.A. van der Vorst. Jacobi-Davidson style QR and QZ algorithms for the partial reduction of matrix pencils. *SIAM J. Sc. Comput.*, 20:94–125, 1998.

[21] G.H. Golub and C.F. van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[22] I. Gustafsson. A class of first order factorization methods. *BIT*, 18:142–156, 1978.

[23] P.R. Halmos. *A Hilbert space problem book.* D. van Nostrand Company, Inc., 1967.

[24] A. Harten. High resolution schemes for hyperbolic conservation laws. *J. Comp. Phys.*, 49:357–393, 1983.

[25] R.A.W.M. Henkes. *Natural-Convection Boundary layers.* PhD thesis, University of Delft, Delft, The Netherlands, 1990.

[26] R.A. Horn and C.R. Johnson. *Topics in matrix analysis.* Cambridge University press, 1991.

[27] H. Jarausch and W. Mackens. Solving large nonlinear systems of equations by an adaptive condensation process. *Numer. Math.*, 50(6), 1987.

[28] H.B. Keller. *Numerical solution of bifurcation and nonlinear eigenvalue problems, Applications of bifurcation theory (edited by P.H. Rabinowitz).* Academic Press, 1977.

[29] B. Koren. Upwind discretization of the steady Navier-Stokes equations. *Int. J. Num. Meth. Fl.*, 11:99–117, 1990.

[30] B. Koren. A robust upwind discretization method for advection, diffusion and source terms. In *Numerical methods for advection-diffusion problems*, pages 117–137, 1993.

[31] K. Lust. *Numerical bifurcation analysis of periodic solutions of partial differential equations*. PhD thesis, K.U. Leuven, Leuven, Belgium, 1997.

[32] K. Lust and D. Roose. Computation and bifurcation analysis of periodic solutions of large-scale systems. Technical report, University of Minnesota, 1998. IMA Preprint 1536.

[33] J.A. Meijerink and H.A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31(137):148–162, 1977.

[34] J.A. Meijerink and H.A. van der Vorst. Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems. *J. Comput. Phys.*, 44(1):134–155, 1981.

[35] Y. Notay and Z. Ould Amar. A nearly optimal preconditioning based on recursive red-black orderings. *Num. Lin. Alg. Appl.*, 4:369–391, 1997.

[36] T.A. Oliphant. An implicit, numerical method for solving two-dimensional time-dependent diffusion problems. *Quart. Appl. Math.*, 19:221–229, 1961.

[37] T.A. Oliphant. An extrapolation procedure for solving linear systems. *Quart. Appl. Math.*, 20:257–265, 1962.

[38] M. Poliashenko and C.K. Aidun. A direct method for computation of simple bifurcations. *J. Comput. Phys.*, pages 246–260, 1995.

[39] Y. Saad. *Numerical methods for large eigenvalue problems*. Manchester University Press, 1992. We used a revised version (available on internet via the homepage of Saad, http://www-users.cs.umn.edu/~ saad).

[40] Y. Saad and M.H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, 1986.

[41] R. Seydel. *Practical bifurcation and stability analysis : from equilibrium to chaos*. Springer-Verlag, second edition, 1994.

[42] G.M. Shroff and H.B. Keller. Stabilization of unstable procedures: the recursive projection method. *SIAM J. Numer. Anal.*, 30(4):1099–1120, 1993.

[43] G.L.G. Sleijpen and D.R. Fokkema. BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum. *ETNA*, 1:11–32, 1993.

[44] S.P. Spekreijse. *Multigrid solution of the steady Euler equations*. Stichting Mathematisch Centrum Centrum voor Wiskunde en Informatica, Amsterdam, volume 46 edition, 1988.

[45] P.K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM J. Numer. Anal.*, 21(5):995–1011, 1984.

[46] G. Tiesinga. Block preconditioned BiCGstab(2) for solving the Navier-Stokes equation. *ZAMM*, 76:563–564, 1996.

[47] A. van der Ploeg. Preconditioning techniques for non-symmetric matrices with application to temperature calculations of cooled concrete. *Int. J. Num. Methods Eng.*, 35(6):1311–1328, 1992.

[48] A. van der Ploeg, E.F.F. Botta, and F.W. Wubs. Grid-independent convergence based on preconditioning techniques. Technical Report W-9310, Department of Mathematics, Groningen, 1993.

[49] H.A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13(2):631–644, 1992.

[50] J.L.M. van Dorsselaer. Computing eigenvalues occurring in continuation methods with the Jacobi-Davidson QZ method. *J. Comp. Phys.*, 138:714–733, 1997.

[51] B. van Leer. Upwind-difference methods for aerodynamic problems governed by the Euler equations. In *Large-scale computations in fluid mechanics, Part 2 (La Jolla, Calif., 1983)*, pages 327–336. Amer. Math. Soc., Providence, R.I., 1985.

[52] H. van Santen, D. Lathouwers, C.R. Kleijn, and H.E.A. van den Akker. Influence of segregation on the efficiency of finite volume methods for the incompressible Navier-Stokes equations. In *Proc. of the Fluids Engng Division of ASME Volume 3*, pages 151–158. The American Society of Mechanical Engineers , New York, 1996.

[53] R.W.C.P. Verstappen and A.E.P. Veldman. Spectro-consistent discretization of Navier-Stokes: a challenge to RANS and LES. *J. Engrg. Math.*, 34:163–179, 1998.

Samenvatting

In het dagelijkse leven hebben we veel te maken met stromingen van vloeistoffen en gassen. Men kan hierbij denken aan de stroming van lucht om een auto, vliegtuig of gebouw, de stroming van water in rivieren en oceanen en natuurlijk de stromingen in de atmosfeer. Een manier om inzicht te krijgen in het gedrag van stromingen is het uitvoeren van experimenten. Hierbij kan men denken aan het doen van metingen aan stromingen rond een auto of vliegtuig in een windtunnel. Dit soort experimenten is echter vaak duur, tijdrovend, of soms zelfs gevaarlijk (denk bijvoorbeeld aan onderzoek aan verbrandingsprocessen of explosies). Daarom worden stromingen tegenwoordig veelvuldig gesimuleerd met behulp van een computer; hiervoor zijn krachtige computers en snelle numerieke algoritmen nodig. De ontwikkeling van dergelijke algoritmen is nog steeds in volle gang en dit proefschrift hoopt hieraan een bijdrage te kunnen leveren.

De beweging van een vloeistof of gas wordt beschreven door de tijdsafhankelijke Navier-Stokes vergelijkingen. Deze vergelijkingen zijn reeds in de eerste helft van de negentiende eeuw opgesteld. Ze worden gevormd door de behoudswetten voor massa en impuls, die in het geval van een samendrukbaar gas aangevuld worden met de behoudswet voor energie. In dit proefschrift beperken we ons tot de onsamendrukbare vergelijkingen. Om de koppeling tussen de afzonderlijke behoudswetten goed weer te geven hebben we er voor gekozen om alle vergelijkingen van het stelsel tegelijk op te lossen, dit in tegenstelling tot een aanpak waarbij de vergelijkingen alternerend worden opgelost. Voor de tijdsintegratie gebruiken we een impliciete methode. Na discretisatie en linearisatie ontstaan hierdoor grote lineaire stelsels, waarvan (gelukkig) de meeste elementen gelijk zijn aan nul.

Het is derhalve belangrijk om snelle algoritmen voor het oplossen van lineaire stelsels $Ax = b$, met de matrix A groot en ijl, te ontwikkelen. Dergelijke stelsels kunnen met een directe of een iteratieve methode opgelost worden. Met een directe methode wordt door middel van Gauss eliminatie een onderdriehoeksmatrix L en een bovendriehoeksmatrix U geconstrueerd zodanig dat $A = LU$. Vervolgens wordt de oplossing van het lineaire stelsel verkregen door de twee stelsels $Ly = b$ en $Ux = y$ op te lossen. Het nadeel van deze methode is dat, alhoewel de matrix A ijl is, de matrices L en U dat niet hoeven te zijn, waardoor veel geheugencapaciteit nodig is. Verder is het maken van een LU ontbinding voor grote matrices erg duur. Hierdoor is het aantrekkelijker om de stelsels met een iteratieve methode op te lossen.

De convergentiesnelheid van iteratieve methoden hangt af van de eigenwaarden van de matrix A; het is met name ongunstig wanneer het spectrale conditiegetal, het quotiënt van de grootste en kleinste eigenwaarde, groot is. Om de convergentiesnelheid te verhogen kan gebruik worden gemaakt van preconditionering. In plaats van het oorspronkelijke stelsel wordt dan het stelsel $P^{-1}Ax = P^{-1}b$ opgelost. De preconditioner P moet een benadering zijn van A; hierdoor lijkt het gepreconditioneerde stelsel op een identiteitsmatrix, waardoor het spectrale conditiegetal veel gunstiger wordt. Verder moet het oplossen van $Py = r$, voor gegeven r, goedkoper zijn dan het oplossen van $Ay = r$.

In dit proefschrift worden preconditioners gebaseerd op een incomplete LU ontbinding van de matrix A beschouwd: $A = LU + R$, waarbij de matrix R de fout in de ontbinding representeert. De incomplete ontbinding wordt door middel van Gauss eliminatie geconstrueerd: er wordt een LU ontbinding van A gemaakt waarbij tijdens het eliminatieproces matrixelementen weggegooid worden. De ordening van de onbekenden en de weggooistrategie van elementen is van grote invloed op de kwaliteit van de preconditioner. Het verkrijgen van meer inzicht in deze afhankelijkheid vormt een belangrijk onderdeel van het onderzoek dat in dit proefschrift wordt beschreven.

In hoofdstuk 2 worden preconditioners beschouwd waarbij de onbekenden genummerd zijn volgens een herhaalde schaakbordvolgorde. Tijdens het eliminatieproces worden elementen weggegooid op basis van hun positie. Een aantrekkelijk punt van deze preconditioner is dat hij eenvoudig te construeren is. Doordat het criterium om elementen weg te gooien niet is gebaseerd op de grootte van de elementen, functioneert de preconditioner niet goed als de elementen sterk van grootte verschillen (bijvoorbeeld als gevolg van een discretisatie op een sterk gerekt rooster).

Om de kwaliteit van de preconditioners te verklaren is in hoofdstuk 3 een theoretische analyse uitgevoerd waarbij we een verzameling afleiden waarin de eigenwaarden van het gepreconditioneerde systeem bevat zijn. Vervolgens is met behulp van Fourier analyse voor verschillende testproblemen een afschatting gemaakt van deze verzameling. Het blijkt dat we op deze manier het convergentiegedrag redelijk kunnen verklaren.

In hoofdstuk 4 wordt een meer geavanceerde preconditioner, de MRILU (matrix renumbering ILU) preconditioner, beschouwd. In deze preconditioner wordt de nummering van de onbekenden bepaald tijdens de factorisatie. Verder wordt de beslissing om een element weg te gooien gebaseerd op zijn verhouding tot de diagonaal van de matrix en op de som van de elementen die reeds weggegooid zijn. Hierdoor ontstaat een accurate factorisatie. Deze preconditioner is in een aantal toepassingen getest en geeft goede resultaten.

Zoals reeds vermeld worden stromingen van vloeistoffen en gassen beschreven door de Navier-Stokes vergelijkingen. In deze vergelijkingen bevindt zich een parameter, het Reynolds getal. Dit getal kan beschouwd worden als een maat voor de snelheid van de stroming; hoe sneller de stroming beweegt, hoe hoger het Reynolds getal is. Het uitvoeren van een bifurcatieanalyse, d.w.z. het berekenen van stromingen voor een reeks Reynolds getallen, de stabiliteit van die stromingen (d.w.z. of ze in de praktijk kunnen voorkomen) en de waarde van het Reynolds getal waarbij de stroming van karakter verandert (bifurcatiepunt), is zowel vanuit theoretisch als praktisch oogpunt interessant. Voor bijvoorbeeld de stroming om een object is het van belang of een stationaire stroming rond dit object kan veranderen in een periodieke stroming. In dat geval gaat de stroming een periodieke kracht uitoefenen op het object waardoor dit beschadigd kan raken (bijvoorbeeld door vermoeidheidsverschijnselen van het materiaal).

Een numerieke techniek waarmee een bifurcatieanalyse uitgevoerd kan worden is een continuatiemethode. Continuatiemethoden worden reeds veelvuldig gebruikt voor problemen met een klein aantal (circa 10) vrijheidsgraden. Pas recent worden ze ook toegepast in problemen met een groot aantal vrijheidsgraden (circa 10^5), waaronder problemen uit de stromingsleer. Door de ontwikkeling van snelle numerieke algoritmen en de toename van de geheugencapaciteit van computers zijn berekeningen aan dergelijke grote systemen mogelijk geworden.

Hoofdstuk 5 beschrijft een bifurcatieanalyse voor het Rayleigh-Bénard probleem: een rechthoekige bak is gevuld met een vloeistof, de temperatuur van de onderkant van de bak is hoger dan de temperatuur van de bovenkant. De relevante parameter in dit probleem is het Rayleigh getal. Dit getal is een maat voor het temperatuurverschil tussen de bovenkant en onderkant van de bak. Als het temperatuurverschil hoog genoeg is zal de vloeistof gaan bewegen. Afhankelijk van het temperatuurverschil kunnen verschillende stromingspatronen ontstaan. In dit specifieke probleem zijn de stromingspatronen niet afhankelijk van de tijd. Hierdoor wordt de stabiliteit van een stromingspatroon bepaald door de eigenwaarden van de Jacobiaan van het systeem. Deze eigenwaarden berekenen we met de Jacobi-Davidson QZ methode, waarbij we de MRILU ontbinding als preconditioner gebruiken. De MRILU preconditioner passen we tevens toe om de stromingspatronen zelf te berekenen.

Vervolgens wordt in hoofdstuk 6 een bifurcatieanalyse beschreven voor het driven-cavity probleem: de bovenkant van een vierkante bak wordt met constante snelheid naar rechts bewogen, waardoor de vloeistof in de bak gaat ronddraaien. De snelheid waarmee de bovenkant van de bak naar rechts wordt bewogen bepaalt hoe de stroming er uit zal zien; het Reynolds getal is een maat voor die snelheid en fungeert derhalve als continueringsparameter. Ondanks dat het driven-cavity probleem veel gebruikt wordt om nieuwe methoden voor het oplossen van de Navier-Stokes vergelijkingen te testen is er nog weinig bekend over zijn bifurcatiegedrag. Als de bovenkant harder gaat bewegen zullen de stromingen afhankelijk worden van de tijd: er ontstaan periodieke stromingen en uiteindelijk zelfs turbulente stromingen. Hierdoor is de methode van hoofdstuk 5 niet meer bruikbaar; er is een krachtigere methode nodig om het bifurcatiegedrag te kunnen bepalen. De bifurcatieanalyse wordt uitgevoerd met de Newton-Picard methode van Lust en Roose. Deze methode maakt gebruik van het feit dat, ondanks dat we te maken hebben met een hoog-dimensionaal systeem, de dynamica van het systeem laag-dimensionaal is. Als gevolg hiervan vormen de eigenvectoren behorende bij de eigenwaarden die verantwoordelijk zijn voor de bifurcaties en de instabiliteit een relatief laag-dimensionale ruimte. Door gebruik te maken van deze eigenschap kunnen de periodieke stromingen en hun stabiliteit efficiënt berekend worden.

Met dit proefschrift hopen we een bijdrage te hebben geleverd aan de ontwikkeling van goede preconditioners voor het oplossen van stelsels lineaire vergelijkingen. Doordat we de beschikking hebben over goede continuatiemethoden en snelle methoden voor het oplossen van lineaire stelsels zijn we in staat om bifurcatieanalyses uit te voeren. We hebben dit gedaan voor zowel tijdsonafhankelijke als tijdsafhankelijke problemen. Verder onderzoek is nodig om de kwaliteit van preconditioners verder te verbeteren; de strategie voor het ordenen van de onbekenden en het weggooien van elementen tijdens de factorisatie spelen hierbij een belangrijke rol.

Dankwoord

Het onderzoek dat ten grondslag ligt aan dit proefschrift is gesubsidieerd door het Prioriteitsprogramma "Niet Lineaire Systemen" van NWO. Op deze plek wil ik graag een aantal mensen bedanken die geholpen hebben bij het tot stand komen van dit proefschrift.

In het bijzonder wil ik Freddy Wubs en Arthur Veldman bedanken: Freddy heeft met zijn ideeën een grote bijdrage aan mijn onderzoek geleverd en Arthur heeft mij, door zijn enthousiasme en vertrouwen, door moeilijke perioden heen geholpen. De leden van mijn leescommissie, Prof.dr.ir. H.W. Hoogstraten, Prof.dr. H.A. van der Vorst and Prof.dr.ir. C.B. Vreugdenhil, wil ik bedanken voor het lezen van dit proefschrift. Verder wil ik Henk Dijkstra bedanken voor het beschikbaar stellen van zijn continueringscode, Eugen Botta voor het mogen gebruiken van zijn MRILU code en Doeke de Vries voor zijn hulp met betrekking tot die code.

De goede sfeer onder de wiskunde aio's en de vele gezamenlijke activiteiten hebben er voor gezorgd dat ik met plezier op de afgelopen jaren terugkijk. Met name Jaap, Jasper en mijn oud kamergenoten, Jan, Willem en Jeroen, wil ik bedanken; zij hebben elk op hun eigen wijze bijgedragen aan die goede sfeer.

Tenslotte wil ik mijn ouders, Eite en Gerard bedanken voor hun ondersteuning en het vertrouwen dat ze in me hebben.