Multi-level ILU preconditioners and continuation methods in fluid dynamics

Tiesinga, Geesien

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 $R^n \times R \to R^n$.

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

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

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

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

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 $\mu_i \quad (i = 1, \ldots, n)$ determine the stability of the stationary solution as follows:

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

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

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

and

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

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 $W^* AVQ = ZS$ and $W^* BVQ = ZT$. 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

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.

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 $128 \times \infty$ 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.

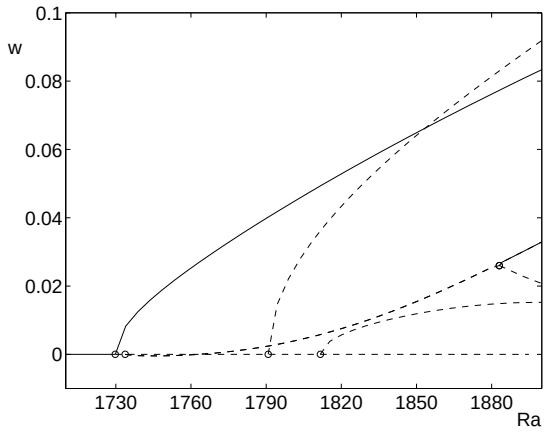

Figure 5.1: Bifurcation diagram for Pr=5.5, solid curves indicate stable states, dashed curves unstable states and circles bifurcation points.

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

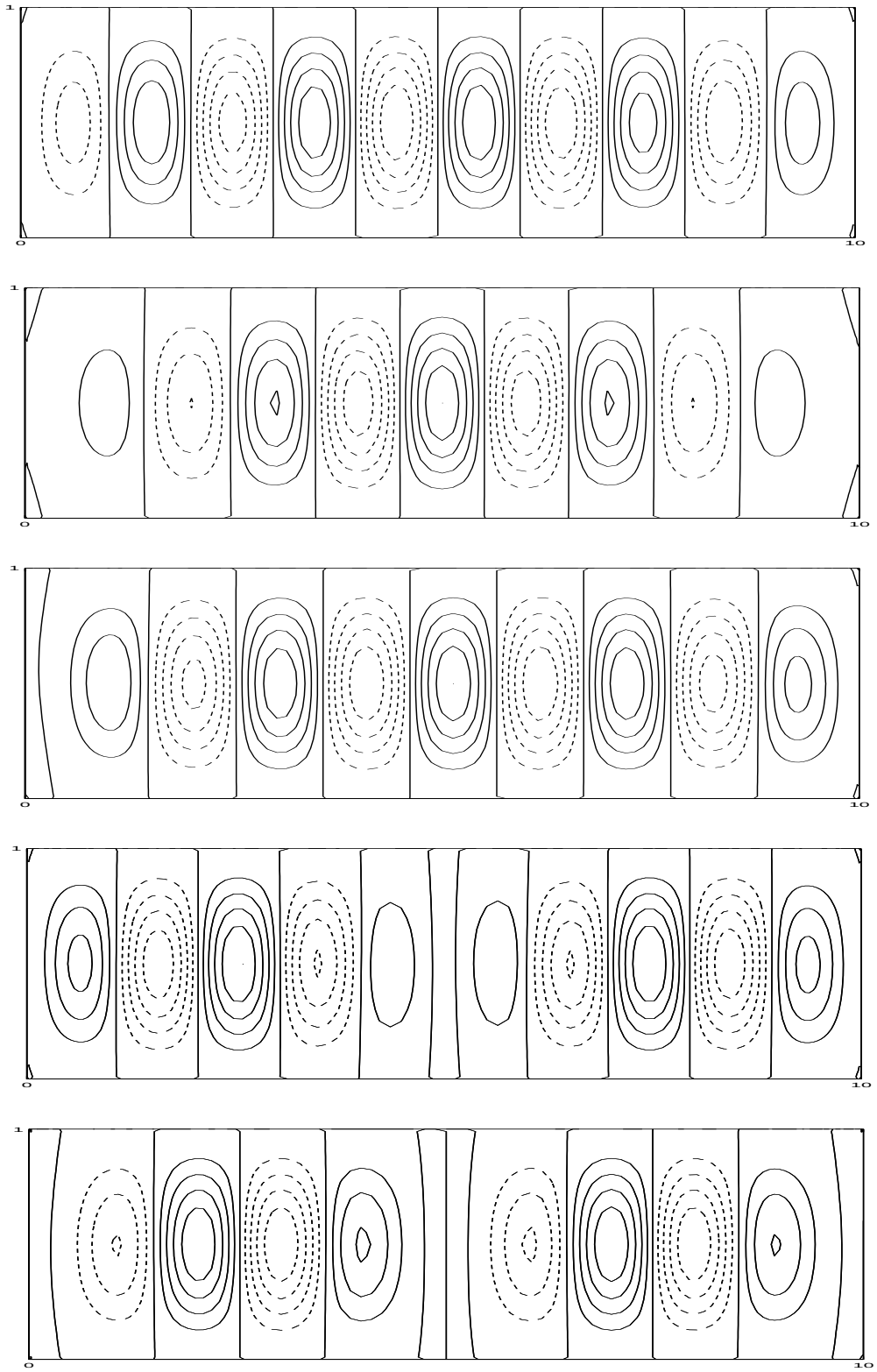

Figure 5.2: Various flow patterns, from top to bottom a stable 10-cell, an unstable 9-cell, an unstable asymmetric 10-cell, an unstable 11-cell and an unstable 12-cell solution.

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

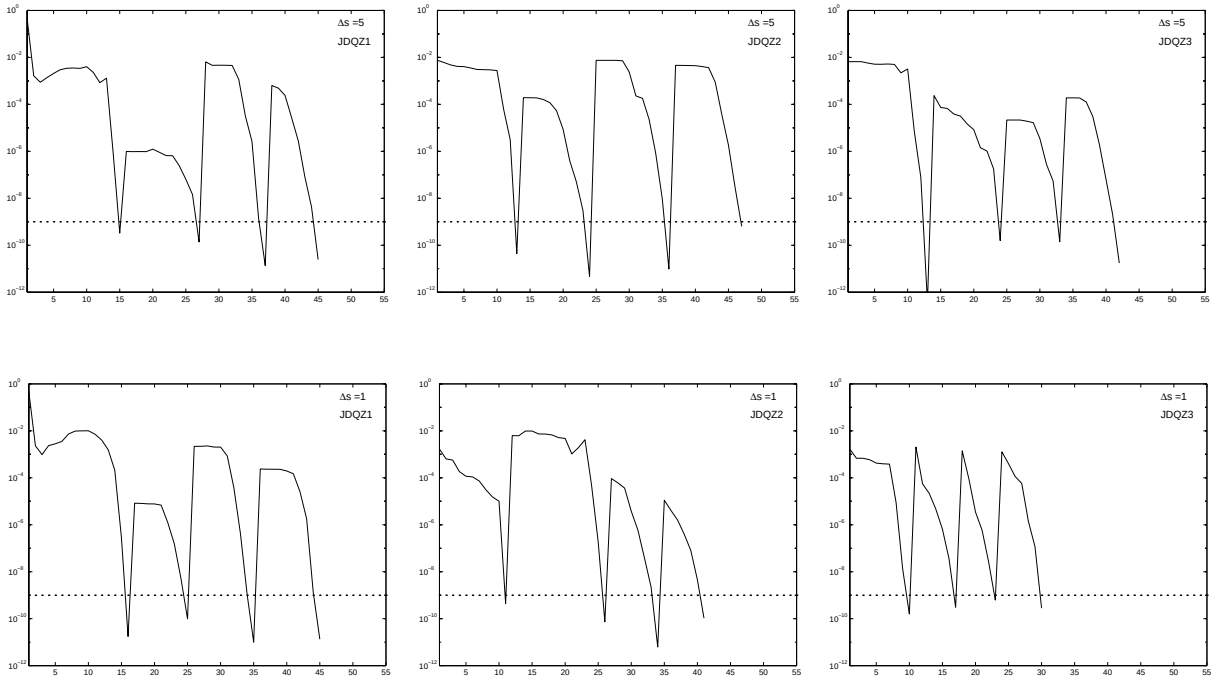

Figure 5.3: Convergence behaviour of the different variants of the JDQZ method for different step sizes in the continuation method.

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.