university of groningen

faculty of mathematics and natural sciences

Numerical Bifurcation Analysis of Large Scale Systems (Part 2)

Dr.ir. F.W. Wubs

Lecture Notes in Applied Mathematics

Academic year 2009-2010

Numerical Bifurcation Analysis of Large Scale Systems (Part 2)

Dr.ir. F.W. Wubs

Institute of Mathematics and Computing Science
P.O. Box 407
9700 AK Groningen
The Netherlands

Contents

Chapter 5

Iterative solution of LSs and EVPs

In this chapter we will introduce iterative methods to solve linear systems and methods to solve the eigenvalue problem. They are treated together since there is an strong relationship between them. It is convenient to introduce two classes of methods: stationary and non-stationary methods. In the former the iteration is the same in every step while it changes from step to step in non-stationary methods.

5.1 Stationary methods for linear systems

We consider the problem $A\mathbf{x} = \mathbf{b}$ and write $A = Q - H$, where Q is nonsingular. This is called a *splitting* of A. Now rewrite the system to

$$Q\mathbf{x} = \mathbf{H}\mathbf{x} + \mathbf{b},$$

and next to

$$\mathbf{x} = \mathbf{C}\mathbf{x} + \mathbf{Q^{-1}}\mathbf{b}, \tag{5.1}$$

From this, the general form of a stationary iteration is now given by

$$\mathbf{x^{(n+1)}} = \mathbf{C}\mathbf{x^{(n)}} + \mathbf{Q^{-1}}\mathbf{b}, \text{ for } \mathbf{n} = \mathbf{0}, \mathbf{1}, \mathbf{2}, ... \tag{5.2}$$

where $\mathbf{x^{(0)}}$ is given and $C = Q^{-1}H$ the iteration matrix. Hence, in a stationary method the $(n+1)$-th iterate depends on the n-th.
If for the iteration (5.2) it holds that $\|\mathbf{x^{(n)}} - \mathbf{x}\| \to \mathbf{0}$, in some norm then the method is *convergent*. If we define the *iteration error* by $\mathbf{v^{(n)}} = \mathbf{x} - \mathbf{x^{(n)}}$, then we find by subtracting (5.2) from (5.1)

$$\mathbf{v^{(n+1)}} = \mathbf{C}\mathbf{v^{(n)}} \tag{5.3}$$

This is a recursion and it is easily shown that

$$\mathbf{v^{(n)}} = \mathbf{C^n}\mathbf{v^{(0)}} \tag{5.4}$$

Exercise 5.1 *Show that* $\hat{v}^{(n)} = \mathbf{x^{(n)}} - \mathbf{x^{(n-1)}}$ *satisfies the same relations as* $\mathbf{v^{(n)}}$.

For the iterative method to be convergent we must require that the repetitive application of C to an arbitrary initial error $\mathbf{v^{(0)}}$ tends to zero.

Theorem 5.1 *The stationary iterative method is convergent if and only if the spectral radius of the iteration matrix is less than one ($\rho(C) < 1$).*

Proof. We only sketch the main line of the proof, details can be found in [48, 50].
If $\rho(C) \geq 1$, then there is an eigenvalue λ and an eigenvector $\mathbf{q}$ with $|\lambda| \geq 1$ and $C\mathbf{q} = \lambda\mathbf{q}$. Hence, $C^n\mathbf{q} = \lambda^{\mathbf{n}}\mathbf{q}$, and $\|C^n\mathbf{q}\| = |\lambda|^{\mathbf{n}}\|\mathbf{q}\|$. So if $\mathbf{v^{(0)}} = \mathbf{q}$ then the method will not converge.
Now suppose that $\rho(C) < 1$. There exists always a non-singular matrix P such that $P^{-1}CP$ is in *Jordan-normal form* J given by

$$J = \begin{bmatrix} J_1 & & & 0 \\ & J_2 & & \\ & & \ddots & \\ 0 & & & J_p \end{bmatrix}$$

in which every *Jordan block* J_i is of the form

$$J_i = \begin{bmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & & \\ \vdots & & \ddots & & \\ 0 & & & \lambda & 1 \\ 0 & . & . & . & \lambda \end{bmatrix} \tag{5.5}$$

where λ_i is an eigenvalue of C. In stead of C^n we can now consider J^n, more specifically J_i^n. A further study of the power of the Jordan block shows that it tends to zero for n to infinity, from which it follows that $\|C^n\| \to 0$. Now from

$$\|C^n\mathbf{v^{(0)}}\| \leq \|\mathbf{C^n}\|\|\mathbf{v^{(0)}}\|$$

the convergence of the iterative method follows. ∎

Theorem 5.2 *If J_i is a Jordan block of order p, as defined in (5.5), then it holds for $n \to \infty$*

$$\|J_i^n\| \approx cn^{p-1}|\lambda_i|^n \tag{5.6}$$

where c is a constant that depends on the choice of the norm.

From this theorem we observe that for sufficient big n the behavior of $\|J_i^n\|$ is determined by $|\lambda_i|$. Hence the behavior of $\|C^n\|$ is also of the form (5.6) with $|\lambda_i|$ replaced by $\rho(C)$ and for p the biggest occurring order Jordan block with $|\lambda_i| = \rho(C)$. For big values of p initially the behavior can be much different from the asymptotic behavior, i.e. the behavior for n tending to infinity.
¿From the above we conclude the best convergence occurs if $\rho(Q^{-1}H)$ is as small as possible, but in any case we must have that

$$\rho(Q^{-1}H) < 1 \tag{5.7}$$

5.1.1 The classical iterative methods

For this we split A as

$$A = D - L - R \tag{5.8}$$

where $D = \text{diag}(A)$, L the strict lower triangular part of A and R the upper triangular part. With this splitting we can describe the classical methods. We will also consider them for thee five-point stencil following from the discretization of the Laplace equation where a lexicographical ordering is used where both i and j increase (i the fastest).

The method of Jacobi

In this case $Q = D$ and $H = L + R$. Hence the iteration matrix assumes the form

$$B = D^{-1}(L + R) \tag{5.9}$$

and is called Jacobi iteration matrix.
This results in the following iteration for the five-point stencil

$$C_P U_{i,j}^{(n+1)} = C_W U_{i-1,j}^{(n)} + C_S U_{i,j-1}^{(n)} + C_E U_{i+1,j}^{(n)} + C_N U_{i,j+1}^{(n)} + f_P \tag{5.10}$$

In this case the ordering of the unknowns does not influence the result. This makes this method very fit for parallelization.

The Gauss-Seidel method

This is an improvement of Jacobi's method in that a result computed is used immediately to compute its neighbor. In this case $Q = D - L$, hence a lower triangular matrix, and $H = R$. Hence, the iteration matrix is

$$(D - L)^{-1} R \tag{5.11}$$

It can be shown that the convergence of the Gauss-Seidel method and the Jacobi method are related for an important class of matrices. For that class it holds that if Jacobi's method is converging then the method of Gauss-Seidel is converging faster.
For the Gauss-Seidel method the application to the five point stencil is as follows:

$$C_P U_{i,j}^{(n+1)} = C_W U_{i-1,j}^{(n+1)} + C_S U_{i,j-1}^{(n+1)} + C_E U_{i+1,j}^{(n)} + C_N U_{i,j+1}^{(n)} + f_P \tag{5.12}$$

In contrast to the Jacobi method we can do with one array for U now, because once a value is updated the old value is not used anymore.
For large problems the convergence may still be too slow, then in some cases one can speed up the method by over-relaxation, which we will treat next.

The successive over-relaxation method (*SOR*)

The idea of relaxation in general is to compute the correction one wants to add to the old value and premultiply it by a factor. Let us consider it for the the problem on the

five point stencil first. At a certain point (i, j) the new value proposed by Gauss-Seidel is

$$\hat{U}_{i,j} \;=\; [C_W U^{(n+1)}_{i-1,j} + C_S U^{(n+1)}_{i,j-1} + C_E U^{(n)}_{i+1,j} + C_N U^{(n)}_{i,j+1} + f_P]/C_P$$

The correction is now $\hat{U}_{i,j} - U^{(n)}_{i,j}$ and therefore

$$U^{(n+1)}_{i,j} = \; U^{(n)}_{i,j} + \omega(\hat{U}_{i,j} - U^{(n)}_{i,j}) = \; (1-\omega)U^{(n)}_{i,j} + \omega\hat{U}_{i,j}$$

In matrix form it is

$$(D - \omega L)\mathbf{x^{(n+1)}} = [(\mathbf{1}-\omega)\mathbf{D} + \omega\mathbf{R}]\mathbf{x^{(n)}} + \omega\mathbf{b}$$

Also this stationary iteration results from a splitting of A. In this case

$$Q = D/\omega - L \qquad \text{en} \;\; H = (1/\omega - 1)D + R$$

and the iteration matrix is

$$C_\omega = (D - \omega L)^{-1}[(1-\omega)D + \omega R] \tag{5.13}$$

Observe that the method is equal to the Gauss-Seidel method for $\omega = 1$. However, in many cases, among which the five-point discretization for the Poisson-equation, the convergence is faster than the Gauss-Seidel method for $1 < \omega < 2$, from which the prefix over in relaxation originates. Also in some cases where Gauss-Seidel does not converge the SOR method may converge for some $\omega < 1$, though in this case it would be better to speak of under relaxation.

Block variants

In all the above methods the point can be generalized to a line, a plane, or a group of unknowns. Doing this we get block variants, in which in every step the unknowns associated to a certain diagonal block are solved together. For example we have SLOR where the L stands for line.

```
== External Links ==
* http://en.wikipedia.org/wiki/Jacobi_method
* http://en.wikipedia.org/wiki/Gauss_seidel
* http://en.wikipedia.org/wiki/Successive_over_relaxation
```

5.1.2 The multigrid method

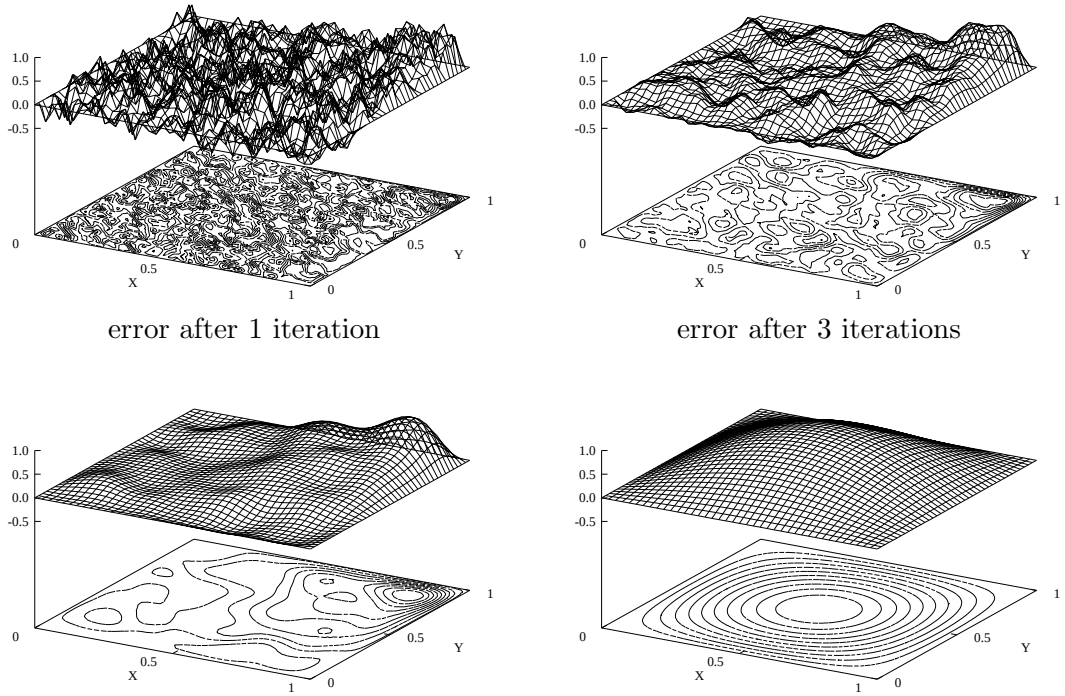
Consider the Poisson equation $-\Delta u = f$ defined on a rectangular domain $(-L_x, L_x) \times (-L_y, L_y)$. On this we define a uniform grid with mesh widths $h = L_x/m$ and $k = L_y/n$ in x and y direction respectively. The grid points are given by

$$\{(ih, jk) \mid i = -m, -m+1, \ldots, m;\; j = -n, -n+1, \ldots, n\} \tag{5.14}$$

After finite difference discretization in the standard way we obtain the difference equation

$$U_{i,j} = \frac{1}{2(h^2+k^2)}\left\{k^2(U_{i-1,j}+U_{i+1,j}) + h^2(U_{i,j-1}+U_{i,j+1}) + h^2k^2 f_{i,j}\right\} \qquad (5.15)$$

If we apply the Gauss-Seidel method to this equation then the error $\mathbf{v^{(n)}}$ satisfies $\mathbf{v^{(n+1)}} = \mathbf{C}\mathbf{v^{(n)}}$, with C the Gauss-Seidel iteration matrix. Fig. 5.1 show how this errors look like after a specified amount of iterations starting from a random initial error. The grid size is 40×40. These plots show something interesting. It shows that



error after 1 iteration

error after 3 iterations

error after 10 iterations

error after 400 iterations

Figure 5.1: Field view of the error after 1,3,10 and 400 Gauss-Seidel iterations

the error is quite smooth after a few iterations (here after 3 to 10 iterations), but after 400 iterations it still has a hard time with a smooth component in the error. It can also be shown analytically that the Gauss-Seidel iteration is good at removing high frequency components from the error and less good in handling low frequency components. Here high frequencies are the highest frequencies that can be represented on the grid. Now for something smooth we could do with less grid points to represent it. The crux of the multigrid method is to get rid of this error on a coarser grid. And the interesting thing is that on a coarser grid the error contains frequencies that are closer to the highest frequency that can be represented on the coarse grid. Hence, on this coarse grid the Gauss-Seidel iteration will do a good job to get rid of these

```
function x =solve(A, b)
    if(coarsest grid is not reached yet) then
      x = 0
      smooth r(≡ b − Ax)
      restrict r → r_c
      dx_c= solve(A_c, r_c)
      dx= prolongate(dx_c)
      x = x + dx
      smooth r
    else
      solve exactly Ax = b
    end
    return x
end
```

Figure 5.2: One multigrid cycle, written as a recursive function

frequencies. The argument can be repeated leading to a nested series of iterations which is displayed in the algorithm in Fig. 5.2. Here smooth is a smoothing iteration like the Gauss-Seidel iteration which updates x, prolongate is an interpolating function to add function values at the grid points not present on the coarser grid, A_c is the approximation of A on a coarser grid and r_c is the restriction of r on a coarser grid.
In order to demonstrate the typical behavior of the multigrid method we show a comparison between multigrid and SLOR applied to a Poisson problem, increasing the number of grid points. The latter method is a variant of SOR where the L stands for line, hence instead of progressing point by point we progress line by line, which means that per line a system for the unknowns on a line has to be solved. In Table 5.1 the number of iterations needed to bring down the residual to a value indicated in the first column for a 16×16, 32×32 and 64×64 grid. In Table 5.2 the amount of time needed to solve the problem is given.
With SLOR we see that if N becomes $4\times$ as big, the needed number of iterations becomes twice as big, whereas for the multigrid method this remains constant. Hence the convergence of the multigrid method is independent of the grid size or grid independent. Both behaviors can be proved for the Poisson equation. With respect to the time we see that that of multigrid increases with the number of grid points, which is due to the fact that the amount of work is just the number of iterations times the work per step. Since the former is nearly constant and the latter depends linearly on the number of unknowns the product is also linear in the number of unknowns. This behavior is optimal, since for an update we have to visit all unknowns. This behavior is very attractive for large problems. Using a similar reasoning for SLOR we find that upon refinement with a factor 2 in both directions the time increases by a factor 8 instead of the optimal value 4. In vertical direction we observe in the columns always a more or less fixed increase. This is typical for stationary methods. One such an

	MGRD			SLOR		
$\|r\|_2$	16×16	32×32	64×64	16×16	32×32	64×64
10^{-3}	6	5	6	10	21	41
10^{-6}	10	9	11	17	40	81
10^{-9}	14	13	17	24	58	121

Table 5.1: Number of iterations

	MGRD			SLOR		
$\|r\|_2$	16×16	32×32	64×64	16×16	32×32	64×64
10^{-3}	0.3	1.1	4.7	0.3	2.4	18.0
10^{-6}	0.6	1.9	8.6	0.5	4.6	35.5
10^{-9}	0.8	2.8	13.2	0.7	6.7	53.0

Table 5.2: Needed CPU-time

increment (say m) shows the number of iterations needed to gain 3 digits. From this, one could also estimate the spectral radius of the iteration matrix of the methods since we have the relation $\rho^m \approx 0.001$. For $m = 4$ this is about 0.17.

5.2 The Power Method to compute the dominant eigenvalue

Unlike the situation for linear systems solving, there are no truly direct methods for the solution of the eigenproblem, in the sense that in general one cannot compute the eigenvalues (or eigenvectors) exactly in a finite number of floating point operations. We will consider iterative methods that detect an invariant subspace. This second class of methods is explicitly based on the Power Method. The *QR method* is the most prominent member of this class; it converges so fast that the complete eigensystem of a dense matrix can be computed in a modest (but matrix dependent) factor times n^3 floating point operations. Note that this order of operations is equal to that of direct methods for solving dense linear systems. The QR method is in general to expensive and an overkill if we only want to find a few eigenvalues, e.g. only the ones closest to

the origin or the imaginary axis. In this case one uses iterative subspace projection techniques. In these methods the QR method is used to find eigenvalues of the small projected matrices which arise. The iterative subspace projection techniques attempt to detect partial eigeninformation in much less than $\mathcal{O}(n^3)$ work.
Because the Power Method is important as a driving mechanism in the subspace projection methods, we will discuss it in some more detail. The reader should keep in mind, however, that the Power Method is seldom competitive as a stand-alone method; we need it here for a better understanding of the more superior techniques to come.
The Power Method is based on the observation that if we multiply a given vector v by the matrix A, then each eigenvector component in v is multiplied by the corresponding eigenvalue of A.
Assume that A is real symmetric, then it has real eigenvalues and a complete set of orthonormal eigenvectors

$$Ax_k = \lambda_k x_k \quad , \quad \|x_k\|_2 = 1 \quad (k = 1, 2, \cdots, n).$$

We further assume that the largest eigenvalue in modulus is single and that

$$|\lambda_1| > |\lambda_2| \geq \cdots .$$

Now suppose we are given a vector v_1, which can be expressed in terms of the eigenvectors as $v_1 = \sum_i \gamma_i x_i$, and we assume that $\gamma_1 \neq 0$ (that means that v_1 has a nonzero component in the direction of the eigenvector corresponding λ_1).

Given this v_1, we compute $Av_1, A(Av_1), ...$, and it follows for the Rayleigh quotient of these vectors that

$$\lim_{j\to\infty} \frac{w_j^T A w_j}{w_j^T w_j} = \lambda_1,$$

where $w_j \equiv A^{j-1}v_1$. Note that this expression is equal to (5.4), the expression we found for the iteration error when solving a linear system but then A is the iteration matrix.
The Power Method can be represented by the template given in Fig. 5.3. The sequence $\theta^{(i)}$ converges, under the above assumptions, to the dominant (in absolute value) eigenvalue A. The scaling of the iteration vectors is necessary in order to prevent over- or underflow. This scaling can be done a little bit cheaper by taking the maximum element of the vector instead of the norm. In that case the maximum element of v_i converges to the largest (in absolute value) eigenvalue of A.
It is not hard to see why the *Rayleigh quotients* converge to the dominant eigenvalue. We first write w_j in terms of the eigenvectors of A:

$$\begin{aligned} w_j &= \sum_{i\geq 1} \gamma_i \lambda_i^{j-1} x_i \\ &= \lambda_1^{j-1} \left\{ \gamma_1 x_1 + \sum_{i\geq 2} \gamma_i \left(\frac{\lambda_i}{\lambda_1}\right)^{j-1} x_i \right\} \quad . \end{aligned}$$

$v = v_1/||v_1||_2$
for $i = 1, 2, ...$ **until convergence**
    $v_{i+1} = Av$
    $\theta^{(i)} = v^T v_{i+1}$
    $v = v_{i+1}/||v_{i+1}||_2$
end

Figure 5.3: The Power Method for symmetric A

Hence

$$\frac{w_j^T A w_j}{w_j^T w_j} = \lambda_1 \frac{\gamma_1^2 + \sum_{i\geq 2} \gamma_i^2 \left(\frac{\lambda_i}{\lambda_1}\right)^{2j-1}}{\gamma_1^2 + \sum_{i\geq 2} \gamma_i^2 \left(\frac{\lambda_i}{\lambda_1}\right)^{2j-2}}$$

For non-symmetric matrices the situation is slightly more complicated, since A does not necessarily have an orthonormal set of eigenvectors. Let $A = XJX^{-1}$ denote the reduction to Jordan form, then $A^j v = XJ^j X^{-1} v$. If the largest eigenvalue, in modulus, is real and simple, then we see that this value will dominate and by similar arguments as above, we see convergence in the direction of the corresponding column of X. If the largest eigenvalue is complex, and if A is real, then there is a conjugate eigenpair. If there is only one eigenpair of the same maximal modulus, then the vector $A^j v$ will ultimately have an oscillatory behavior and it can be shown, see WILKINSON [49, p. 579], that a combination of two successive vectors in the sequence $A^j v$ will converge to a subspace spanned by the two conjugate eigenvectors. The two eigenvectors can then be recovered by a least squares solution approach. For matrices with a dominant Jordan block one can also show that the Power Method will converge, although very slowly, to the eigenvector associated with the Jordan block. The behavior is essential given by (5.6).

Exercise 5.2 *Suppose we replace the matrix A in the power method by a shifted variant $A - \sigma I$. Assume that A is real. If you would already know the eigenvalues of A, how would you choose σ in order to let the power method converge fastest to the eigenvector associated to the largest eigenvalue of A? And how would you choose it for the eigenvector corresponding to the eigenvalue which is smallest?*

Inverse power method If systems with the shifted matrix, like

$$(A - \sigma I)x = b$$

can be solved efficiently, then it is very profitable to use the inverse power Method with shift. If one has a good guess σ for the eigenvalue that one is interested in, then one can

apply the Power Method with $(A-\sigma I)^{-1}$. Note that it is not necessary to compute this inverted matrix explicitly. In the computation of the next vector $v_{j+1} = (A-\sigma I)^{-1}v_j$ the vector v_{j+1} can be solved from

$$(A-\sigma I)v_{j+1} = v_j.$$

Assume that the largest eigenvalue (in absolute value) is λ_1, and the second largest is λ_2, and that σ is close to λ_1. The speed of convergence now depends on the ratio

$$\frac{|\lambda_1 - \sigma|}{|\lambda_2 - \sigma|},$$

and this ratio may be a good deal smaller than $|\lambda_2|/|\lambda_1|$. Even when the solution of a shifted linear system is significantly more expensive than a matrix vector multiplication with A, the much faster convergence may easily pay off for these additional costs.

Rayleigh quotient iteration In the inverse power method we may update the shift as better approximations become available during the iteration process. That is, when we apply the Algorithm in Fig.5.3 with $(A-\sigma I)^{-1}$ at step i, then $\theta^{(i)}$ is an approximation for $1/(\lambda_1 - \sigma)$. This means that the approximation for λ_1 becomes $\sigma + 1/\theta^{(i)}$ and we can use this value as the shift for iteration $i+1$. This technique is known as *Rayleigh Quotient iteration*. Its convergence is ultimately cubic for symmetric matrices and quadratic for non-symmetric systems.

5.3 Simultaneous Iteration

We have already seen that for complex conjugate pairs it is necessary to work with three successive vectors in the Power iteration. This suggests that it may be a good idea to start with a block of vectors right from the start. So let us assume that we start with a set of independent vectors $U_k^{(0)} = [u_1, u_2, \ldots, u_k]$, and that we carry out the Power Method with $U_k^{(0)}$, which leads to the computation of

$$U_k^{(i)} = AU_k^{(i-1)}$$

per iteration. If we do this in a straightforward manner, then this will lead to unsatisfactory results because each of the columns of $U_k^{(0)}$ is effectively used as a starting vector for a single vector Power Method, and all these single vector processes will tend to converge towards the dominant eigenvector(s). This will make the columns of $U_k^{(i)}$ highly dependent in the course of the iteration. It is therefore a better idea to try to maintain better numerical independence between these columns and the most common technique for this is to make them orthonormal after each multiplication with A. This leads to the *Orthogonal Iteration* Method, as represented in Fig. 5.4.
The columns of $U_k^{(i)}$ converge to a basis of an invariant subspace of dimension k, under the assumption that the largest k (in absolute value) eigenvalues (counted according to multiplicity) are separated from the remainder of the spectrum. This can easily be

start with orthonormal $U_k^{(1)}$
for $i = 1, ..,$ **until convergence**
    $V_k = AU_k^{(i)}$
    orthonormalize the columns of V_k:
        $V_k = Q_k R_k$
    $U_k^{(i+1)} = Q_k$
end

Figure 5.4: The Orthogonal Iteration Method

seen from the same arguments as for the Power Method. If the eigenvalues are real, and the matrix is real, then the eigenvalues appear along the diagonal of R. For a real matrix the complex eigenvalues are the eigenvalues of 2x2 blocks that appear on the diagonal.
So far we treated stationary methods for linear systems and the eigenvalue problem. In the next part we will consider the Krylov subspace methods which are of non-stationary type.

5.4 Krylov subspaces

5.4.1 Construction of a basis

With the Power method (see 5.2), we have generated the spanning vectors of a *Krylov Subspace*

$$\mathcal{K}^m(A; v_1) \equiv \text{span}\{v_1, Av_1, \ldots, A^{m-1}v_1\}.$$

However, note that the Power method exploits only the two most recently computed vectors for the computation of an approximating eigenvalue. The result is that the speed of convergence depends on the ratio λ_2/λ_1. Here we have assumed that the eigenvalues have been ordered as $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$. Of course, one may try to influence the convergence of the Power Method by working with shifts, either to separate the wanted eigenvalue more from the remaining spectrum, or to suppress eigenvector components of nearby eigenvalues. For this one needs, in fact, good guesses for these nearby eigenvalues, and, as we will see, the methods of Lanczos and Arnoldi do this automatically. From the definition of a Krylov subspace, it is clear that

$$\mathcal{K}^m(\alpha A + \beta I; v_1) = \mathcal{K}^m(A; v_1) \quad \text{for} \quad \alpha \neq 0. \tag{5.16}$$

This says that the Krylov subspace is spanned by the same basis if A is scaled and/or shifted. The implication is that Krylov subspace methods for the spectrum of the matrix A are invariant under translations for A.

Krylov subspaces play a central role in iterative methods for eigenvalue computations. In order to identify better solutions in the Krylov subspace we need a suitable basis for this subspace, one that can be extended inductively for subspaces of increasing dimension. The obvious basis v_1, Av_1, ..., $A^{m-1}v_1$ for $K^m(A; v_1)$ is not very attractive from a numerical point of view, since the vectors $A^j v_1$ point more and more in the direction of the dominant eigenvector for increasing j (the power method!), and hence the basis vectors become dependent in finite precision arithmetic.
Lanczos [24] proposes to generate an orthogonal basis for the Krylov subspace, and shows that this could be done in a very economic way for symmetric A. Arnoldi [1] describes a procedure for the computation of an orthonormal basis for non-symmetric matrices and for ease of presentation, we start with this one. We will see that Lanczos' algorithm follows as a special case.
If we have already an orthonormal basis v_1, ..., v_j for $\mathcal{K}^j(A; v_1)$, then this basis is expanded by computing $\widetilde{v} = Av_j$, and by orthonormalizing this vector $\widetilde{v}$ with respect to v_1, ..., v_j. In principle the orthonormalization process can be carried out in different ways, but the most commonly used approach is to do this by a modified Gram-Schmidt procedure (Golub and Van Loan [16]).
This leads to the algorithm for the creation of an orthonormal basis for $\mathcal{K}^m(A; v)$, as given in Fig. 5.5.

```
v is a convenient starting vector
v_1 = v/||v||_2
for j = 1, .., m − 1
    t = Av_j
    for i = 1, ..., j
        h_{i,j} = v_i^* t
        t = t − h_{i,j} v_i
    end
    h_{j+1,j} = ||t||_2
    v_{j+1} = t/h_{j+1,j}
end
```

Figure 5.5: The Arnoldi algorithm with modified Gram-Schmidt to construct the reduced matrix H of A

The orthogonalization leads to relations between the v_j, that can be formulated in a compact algebraic form. Let V_j denote the matrix with columns v_1 up to v_j, $V_j \equiv [v_1 \mid v_2 \mid \cdots \mid v_j]$, then it follows that

$$AV_{m-1} = V_m H_{m,m-1}. \tag{5.17}$$

The m by $m-1$ matrix $H_{m,m-1}$ is upper Hessenberg, and its elements $h_{i,j}$ are defined

by the Arnoldi orthogonalization algorithm. We will refer to this matrix $H_{m,m-1}$ as the *reduced matrix* A, or more precisely, the matrix A reduced (or projected) to the current Krylov subspace. From a computational point of view, this construction is composed from three basic elements: a matrix vector product with A, inner products, and updates. We see that this orthogonalization becomes increasingly expensive for increasing dimension of the subspace, since the computation of each $h_{i,j}$ requires an inner product and a vector update.

Finite termination In exact arithmetic, this process must terminate after at most n steps, since then the n orthonormal vectors for the Krylov subspace span the whole space. In fact, the process terminates after k steps, if the starting vector has components only in the directions of eigenvectors corresponding to k *different* eigenvalues.

Exercise 5.3 *Write* $v = \sum_{l=1}^{k} \alpha_k x_k$ *and show that* $t = 0$ *after at most* k *steps.*

5.5 Arnoldi's Method

5.5.1 The computation of eigenpairs

As we have seen in Section 5.4, the construction of a basis with Arnoldi's algorithm for the Krylov subspace $\mathcal{K}^m(A; v_1)$ leads to an upper Hessenberg matrix that describes the relation between the basis vectors. If we increase the indices in (5.17) by one we obtain

$$AV_m = V_{m+1}H_{m+1,m}, \tag{5.18}$$

which we do here for convenience. If we premultiply this expression by V_m^T then because of orthogonality we have that

$$V_m^T A V_m = H_{m,m}. \tag{5.19}$$

The matrix $H_{m,m}$ is the projection of A onto $K^m(A; v_1)$.
Now, if (θ, s) is an eigenpair of $H_{m,m}$ then

$$H_{m,m}s = \theta s$$

$$V_m^T A V_m s - \theta V_m^T V_m s = 0$$

$$V_m^T (A V_m s - \theta V_m s) = 0$$

$$V_m^T (Ay - \theta y) = 0,$$

where $y = V_m s$.
Hence, the residual for the approximate eigenpair (θ, y) is orthogonal to the current Krylov subspace. Since the search space, i.e. the space containing the approximate eigenvalue, is equal to the test space, i.e. the space we need to make the problem solvable, this is the Galerkin approach to find the approximate eigenpairs. In the symmetric case it would have been called the Ritz approach. However, in both cases the value θ is called a *Ritz value* of A with respect to $K^m(A; v_1)$, and y is the corresponding *Ritz vector*.

The main problem with Arnoldi's method is that it becomes increasingly expensive per iteration step. In order to restrict memory storage, as well as computational work, restarts are necessary. However, at a restart we do not want to throw away useful information. By choosing a smart basis vector for the restart, one can keep important information.

5.5.2 Convergence behavior and exterior and interior eigenvalues

Due to the presence of powers of A in the Krylov subspace, it is obvious that we can use our knowledge of the convergence of the power method to say something about the convergence of the Ritz values. Firstly we consider a property of the Ritz values for a real symmetric matrix. In that case, it holds that

$$\max_x \frac{(x, Ax)}{(x, x)} = \max_{\lambda \in \sigma(A)} \lambda$$

and likewise for the minimum. This is just a special case of the Courant-Fisher min-max theorem [18]. Now we have that

$$\max_{\theta \in \sigma(H)} \theta = \max_s \frac{(s, Hs)}{(s, s)} = \max_s \frac{(s, V^T AVs)}{(s, V^T Vs)} = \max_{x \in V} \frac{(x, Ax)}{(x, x)} \leq \max_x \frac{(x, Ax)}{(x, x)} = \max_{\lambda \in \sigma(A)} \lambda$$

So if V is a Krylov subspace, then we know that the eigenvector related to the maximum eigenvalue is one of the first to become strongly present in the subspace. The above learns us that then also the largest Ritz value will converge faster to the largest eigenvalue of A than the Rayleigh quotient in the Power Method. But this is not all, due to the invariance of the Krylov subspace for scaling of A and shifting of A also the eigenvalues which can be made extreme by scaling and shifting can be found faster than by applying the power method to the corresponding scaled and shifted matrix. So in the real symmetric case also the minimum Ritz value will converge among the first to the mimimum eigenvalue. The above can be extended to normal matrices and to some extend it also holds for nonnormal matrices, depending on the so-called departure of normality.
This implies that for the Krylov subspace method the notion of largest (in absolute) value eigenvalue looses its special meaning as opposed to the Power Method. Since the Krylov methods are shift invariant, the position of the origin in the spectrum is not relevant and we rather make the distinction between *exterior* and *interior* eigenvalues.

Exercise 5.4 *What does Exercise 5.2 learn us about the convergence of the eigenvectors corresponding to the exterior eigenvalues in the Krylov subspace?*

The Krylov methods have no special preference for eigenvalues that are at about the same distance of the center of the spectrum, provided that these eigenvalues are about equally well separated from the others. In particular, when the real spectrum of a symmetric real matrix is symmetrically distributed with respect to $(\lambda_1+\lambda_n)/2$, λ_1 being the largest eigenvalue of A and λ_n the smallest one, then for a starting vector that has also a symmetric weight distribution with respect to the corresponding eigenvectors,

the convergence of the smallest Ritz value towards λ_n will be equally fast (or slow) as the convergence of the largest Ritz value to λ_1.
For complex spectra, one has to consider the smallest circle that encloses all eigenvalues. With proper assumptions about the starting vector, one may expect that the eigenvalues close to this circle will be approximated fastest and that the more interior eigenvalues will be approximated later in the Krylov process (that is, for larger values of m). For more general starting vectors this describes more or less the generic case, but with special starting vectors one can force convergence towards favored parts of the spectrum. This forms the basis for restart strategies.
For an excellent overview of subspace methods, see SAAD [35] or [37].

5.6 The Lanczos Method

Note that if A is symmetric and real, then so is

$$H_{m-1,m-1} = V_{m-1}^T A V_{m-1},$$

so that in this situation $H_{m-1,m-1}$ is tridiagonal:

$$AV_{m-1} = V_m T_{m,m-1}. \tag{5.20}$$

The matrix $T_{m,m-1}$ is an m by $m-1$ tridiagonal matrix, its leading $m-1$ by $m-1$ part is symmetric.
This means that in the orthogonalization process, each new vector has to be orthogonalized with respect to the previous two vectors only, since all other inner products vanish. The resulting three term recurrence relation for the basis vectors of $\mathcal{K}(A; v_1)$ is the kernel of the *Lanczos method* and some very elegant methods are derived from it. A template for the Lanczos algorithm to find the coefficients α_j and β_j that build T is given in Algorithm 5.6.

v is a convenient starting vector
$v_1 = v/||v||_2,\ v_0 = 0,\ \beta_0 = 0,$
for $j = 1, 2, ..., m-1$
    $t = Av_j - \beta_{j-1} v_{j-1}$
    $\alpha_j = v_j^T t$
    $t = t - \alpha_j v_j$
    $\beta_j = \|t\|_2$
    $v_{j+1} = t/\beta_j$
end

Figure 5.6: The Lanczos algorithm for the construction of the reduced matrix

N.B. The result of the Lanczos algorithm is that $\beta_j v_{j+1} = Av_j - \beta_{j-1}v_{j-1} - \alpha_j v_j$ or

$$Av_j = [v_{j-1}, v_j, v_{j+1}] \begin{bmatrix} \beta_{j-1} \\ \alpha_j \\ \beta_j \end{bmatrix}$$

which is just another way of writing (5.20).
In the symmetric case the orthogonalization process involves constant arithmetical costs per iteration step: one matrix vector product, two inner products, and two vector updates.

5.7 The Two-sided Lanczos Method

In this section we will discuss a generalization of the Lanczos method for non-symmetric matrices in order to avoid the storage and computational work needed in Arnoldi's method. This method is known as the Two-sided Lanczos method, or Bi-Lanczos method. As we will see this is in fact a Petrov-Galerkin approach to determine approximate eigenpairs.
For ease of notation, we will restrict ourselves to the real case. For an non-symmetric matrix $A \in \mathbb{R}^{n\times n}$ we will try to obtain a suitable non-orthogonal basis with a 3-term recurrence, by requiring that this basis is orthogonal with respect to some other basis. This leads to the *Bi-Lanczos Method* (Lanczos [24]).
We derive the two-sided Lanczos algorithm, following Wilkinson [49, Chapter 6.36] . We start from two Arnoldi like recursions for the basis vectors of $K^m(A; v_1)$ and $K^m(A^T; w_1)$:

$$\begin{aligned} h_{j+1,j}v_{j+1} &= Av_j - \sum_{i=1}^{j} h_{i,j}v_i, \\ g_{j+1,j}w_{j+1} &= A^T w_j - \sum_{i=1}^{j} g_{i,j}w_i, \end{aligned}$$

and require that v_{j+1} is orthogonal to all previous w_i and that w_{j+1} is orthogonal to all previous v_i. Clearly, this defines, apart from the constants $h_{j+1,j}$ and $g_{j+1,j}$, the vectors v_{j+1} and w_{j+1}, once the previous vectors are given. Then we have that

$$AV_j = V_{j+1}H_{j+1,j} \quad \text{and} \quad A^T W_j = W_{j+1}G_{j+1,j}.$$

Since each new v_{j+1} is only orthogonal with respect to the w_i, for $i < j$, and likewise for w_{j+1} with respect to the v_i, it follows that

$$W_j^T V_j = L_{j,j} \quad \text{and} \quad V_j^T W_j = K_{j,j},$$

where $L_{j,j}$ and $K_{j,j}$ are lower triangular. Clearly

$$K_{j,j}^T = L_{j,j},$$

so that both matrices are diagonal. Let us denote this matrix by D. Then, we have that

$$W_j^T A V_j = D_j H_{j,j}$$

and also

$$V_j^T A^T W_j = D_j G_{j,j}.$$

Hence, $D_j H_{j,j} = G_{j,j}^T D_j$. This shows that $H_{j,j}$ and $G_{j,j}$ must be tridiagonal and we write H as T from now on. So this means that only the last few columns of V and W needs to be retained to construct T.
There is still a freedom here which can be exploited: the entries of D. Here we choose it such that $D_j T_{j,j}$ is symmetric. This means that also $G_{j,j}^T D_j$ is symmetric hence $D_j H_{j,j} = G_{j,j}^T D_j = D_j G_{j,j}$ and thus $G = H = T$.

Exercise 5.5 *Show that any tridiagonal matrix can be made symmetric by premultiplication by an appropriate diagonal matrix.*

So in the non-symmetric Lanczos method (Lanczos [24]), we generate a dual bases $\{v_j\}$ and $\{w_j\}$ for the Krylov subspace $K^i(A; v_1)$ and its adjoint $K^i(A^T; w_1)$. The v_j are generated with a three term recurrence relation, with A:

$$\gamma_j v_{j+1} = A v_j - \alpha_j v_j - \beta_{j-1} v_{j-1},$$

and the w_j with a similar recurrence for A^T:

$$\gamma_j w_{j+1} = A^T w_j - \alpha_j w_j - \beta_{j-1} w_{j-1}.$$

In Figure 5.7, we show schematically an algorithm for the two-sided Lanczos, suitable for execution with an non-symmetric matrix A.
In this algorithm, we choose γ_i such that $\|v_i\|_2 = 1$, and use the same γ_i as a scaling for w_i.

For the computation of approximate eigenpairs, we may proceed in a similar way as in the symmetric case:

$$A V_j = V_{j+1} T_{j+1,j}, \tag{5.21}$$

but here we use the matrix $W_j = [w_1, w_2, ..., w_j]$ for orthogonality conditions on an approximate vector. Let $V_j s$ be the wanted eigenvector approximation in V_j, with corresponding eigenvalue approximation, then we impose the *Petrov-Galerkin* condition

$$W_j^T (A V_j s - \theta V_j s) = 0,$$

or

$$W_j^T A V_j s - \theta W_j^T V_j s = 0.$$

This shows that s and θ form an eigenpair of the generalized symmetric eigenproblem

$$D_j T_{j,j} s = \theta D_j s,$$

Select a normalized pair v_1, w_1 (for instance, $w_1 = v_1$)
such that $w_1^T v_1 = \delta_1 \neq 0$
$\beta_0 = 0$, $w_0 = v_0 = 0$
for $i = 1, 2, ...$
$p = Av_i - \beta_{i-1} v_{i-1}$
$\alpha_i = w_i^T p / \delta_i$
$p = p - \alpha_i v_i$
$\gamma_{i+1} = \|p\|_2$
$v_{i+1} = p / \gamma_{i+1}$
$w_{i+1} = (A^T w_i - \beta_{i-1} w_{i-1} - \alpha_i w_i) / \gamma_{i+1}$
$\delta_{i+1} = w_{i+1}^T v_{i+1}$
$\beta_i = \gamma_{i+1} \delta_{i+1} / \delta_i$
end;

Figure 5.7: The Two-sided Lanczos algorithm

or of the standard non-symmetric eigenproblem

$$T_{j,j} s = \theta s.$$

An interesting aspect of the two-sided Lanczos approach is that it also admits possibilities for the approximation of left eigenvectors. Let $p^* W_j^T$ denote an approximation for a left eigenvector corresponding to an approximate eigenvalue θ. Note that the eigenvectors of A and $T_{j,j}$ may be complex; this explains the $*$. Then the Petrov-Galerkin condition with respect to V_j leads to

$$p^* W_j^T A V_j - \theta p^* W_j^T V_j = 0, \tag{5.22}$$

which shows that the pair p, θ is a *left eigenpair* of the symmetric generalized eigenproblem

$$p^* D_j T_{j,j} = \theta p^* D_j.$$

Since $D_j T_{j,j}$ is real symmetric, the vector p is a right eigenvector of $T_{j,j}$. It is, because of the Petrov-Galerkin condition, natural to associate Petrov-Galerkin approximations for eigenvectors of A with the eigenpair s, θ of $T_{j,j}$. We will use the terminology of *Petrov value* for θ, *right Petrov vector* for $V_j s$, and *left Petrov vector* for $W_j s$.

5.8 The Jacobi-Davidson Method

As a sideline we treat in this section the Jacobi-Davidson method as proposed by Van der Vorst et al. [13] will be discussed. This method is not a Krylov-subspace method. It uses however a subspace to accelerate the convergence. The exposition in this section starts from the familiar Newton method.

5.8.1 Newton's method for the eigenvalue problem

Assume we solve the ordinary eigenvalue problem $(A-\lambda I)x = 0$. Say we have already computed the partial Schur form

$$AQ = QR,$$

where R is an upper-triangular matrix and the eigenvalues are on its diagonal. Furthermore $Q^*Q = I$. Now we want to compute the next q such that

$$A[Q\ q] \quad = \quad [Q\ q]\begin{bmatrix} R & s \\ 0 & \lambda \end{bmatrix}.$$

Working out this equation, we find that the new part is $Aq = Qs + \lambda q$. We add two equations to normalize q and the ensure that q is perpendicular to the current Q. We obtain the following set of equations

$$\begin{aligned} (A-\lambda I)q - Qs &= 0, \\ -(q^*q)/2 + 1/2 &= 0, \\ -Q^*q &= 0. \end{aligned}$$

Define $u \equiv (q, \lambda, s)$ and assume we have some good initial guess $u_0 = (q_0, \lambda_0, s_0)$ for u. Now we use Newton's method to solve these equations. First we solve $\Delta u = (\Delta q_i, \Delta\lambda_i, \Delta s_i)$ from the following equation

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

and then compute $u_{i+1} = u_i + \Delta u_i$.
In principle this algorithm will converge to one of the eigenpairs of A. In the following we will try to speed-up the method by using a subspace.
Exploiting the problem we adapt the recursion a bit. First we assume that q_i has length one and is perpendicular to Q. Hence the last two entries of the right-hand side become zero. Of course we have to make sure that q_{i+1} also has these properties which can be done by applying the Gram-Schmidt process. Moreover, observe that we could take $s_i = 0$ by computing immediately s_{i+1} instead of Δs_i.

5.8.2 Acceleration

Instead of using Δq_i in the update of u_i one can also add it to the subspace spanned by the hitherto found corrections Δq_j, $j = 1, ..., i-1$, and compute the Ritz values of A with respect to this subspace. For example to compute the k smallest eigenvalues of A one can apply the following algorithm.
We start off with some initial guess for the eigenvector t, and void spaces Q and V, the latter being the space spanned by the corrections.

1. Orthonormalize t with respect to current Q and V and extend V with it.

2. Construct the reduced matrix on the space spanned by V: $M = V^*AV$.

3. Compute the smallest eigenvalue of M, say θ, and the corresponding eigenvector s.

4. Compute the eigenpair residual: $r = Aq - \theta q$, where $q = Vs$.

5. If the residual is small enough extend Q by q. If, on top of that, the dimension of Q is still less than k generate a new initial t and return to 1, else STOP.

6. Solve the correction equation for a new t and return to 1.

In practice also a restart strategy is added in order to control the dimension of the subspace V. Moreover variants for the reduced eigenvalue problem are also considered in order to avoid difficulties in computing interior eigenvalues.
Finally, the correction equation can also be solved approximately. For example by applying an iterative method to solve the system. Only a few steps may suffice to get a useful correction to extend the space V.
The above method is an instance of the so-called Jacobi-Davidson method, reflecting that it is a combination of ideas of a method of Jacobi and one of Davidson. This combination was devised and analyzed by Van der Vorst and his co-workers and is treated extensively in his book on this subject.

5.9 Generalized Eigenproblems

In this section we will pay more attention to the possibilities for the iterative computation of (*right*) eigenpairs of the generalized non-Hermitian eigenvalue problem

$$Ax = \lambda Bx. \tag{5.23}$$

where A and B are general n by n matrices.
the QZ algorithm For the dense generalized eigenvalue problem the QZ algorithm has been devised, a generalization of the QR method. This is currently the most powerful method for dense problems, but it is only suitable for small to moderate size problems because the QZ method requires $\mathcal{O}(n^3)$ flops and $\mathcal{O}(n^2)$ memory locations.
A common approach for a large scale generalized eigenvalue problem is to reduce the problem (5.23) to a standard eigenvalue problem, and then apply an appropriate iterative method, for instance, Arnoldi's method, two-sided Lanczos, or the Jacobi-Davidson method. This reduction to standard form requires, for each iteration, the solution of a linear system with A, or B, or a combination of A and B.
Here we will discuss three approaches for the reduction to a standard eigenproblem.

1. **Inverse** B: If the matrix B is nonsingular and well conditioned, then we can do operations with the inverse of B. The problem (5.23) is equivalent to

$$(B^{-1}A)x = \lambda x. \tag{5.24}$$

For an iterative method, such as we have described in previous chapters, for the reduced standard eigenvalue problem (5.24) it is not necessary to evaluate the product $B^{-1}A$. This is important, since A and B are in most applications sparse, and forming the matrix $B^{-1}A$ (most likely a dense matrix) would have been very very unattractive. The computation of a vector $v = B^{-1}y$ is done via a (sparse) LU factorization of B and then solving v from $Bv = y$.

The error introduced by this reduction to the standard form can be proportional to $\|A\|_2\|B^{-1}\|_2$. If B is ill-conditioned, then the approach is potentially suspect. In that situation, one may consider the usage of the shift-and-invert transformation for the reduction (see (5.26)), the Cayley Transform (see (5.27)), or the usage of the Jacobi-Davidson method.

It may be tempting to consider iterative solution methods for the approximate solution of systems like $Bz = u$, instead of the possibly expensive LU factorization. However, if one does so then one should realize that the approximate solution satisfies some nearby system, say with a matrix $\widetilde{B}$. This nearby matrix may differ significantly from iteration to iteration in forming the (Krylov) subspace for the eigenvalue method. This will most likely destroy the structure of the Krylov subspace. Therefore, an iterative solution technique is only recommended if it leads efficiently to high accuracy, comparable with a stable direct solution method. Likewise, if the direct solution method has made a compromise between stability and sparsity (to the advantage of the latter), for instance, by working with threshold pivoting (Duff, Erisman and Reid [11]), it is advisable to use iterative refinement in order to get more accurate solutions (Golub and Van Loan [16]).

2. **Symmetrized inverse** B: In some applications, B is Hermitian positive definite and well-conditioned. In this case, it is recommended to compute first a sparse Cholesky decomposition

$$B = LL^*,$$

with L a lower triangular matrix. The equivalent standard eigenvalue problem is

$$(L^{-1}AL^{*-1})y = \lambda y, \tag{5.25}$$

where $y = L^*x$. As in the 'Invert B' approach, matrices like $L^{-1}AL^{*-1}$ should never be evaluated explicitly, since they will in general not be sparse. For the application of an iterative method for the standard eigenvalue problem, we only need to provide the efficient evaluation of matrix-vector products, like

$$p = (L^{-1}AL^{*-1})q.$$

where q is a given vector.

Since L is a triangular matrix, the solutions of linear systems with L or L^* can be obtained by forward and backward substitutions. Sparsity can be exploited in a straightforward manner in all these steps.

3. **Shift-and-Invert**: The reductions to standard form (5.24) or (5.25) cannot be used when B is singular or ill-conditioned. An attractive and popular technique is to apply first the shift to the original problem and then carry out the reduction. This is the so-called *shift and invert spectral transformation*. Specifically, let σ be a user-selected shift such that the matrix $A - \sigma B$ is nonsingular, then the original problem (5.23) can be transformed to

$$Cx = \mu x, \tag{5.26}$$

where

$$C = (A - \sigma B)^{-1} B \quad \text{and} \quad \mu = \frac{1}{\lambda - \sigma},$$

We see that the eigenvalues λ of the problem (5.23) closest to the shift σ are mapped as the exterior eigenvalues of the reduced standard eigenvalue problem (5.26), that is to the eigenvalues of largest magnitude, and these are the eigenvalues that are first well approximated by most iterative methods.

In practice, an effective shift selection depends on the user's preferences and on knowledge of the underlying generalized eigenproblem. A good shift not only amplifies the desired eigenvalues, but it also leads to a well-conditioned matrix $A - \sigma B$. This makes the task of selecting good shifts often a challenging one.

For the application of an iterative method for the reduced standard eigenvalue problem (5.26), one needs to evaluate matrix-vector products $p = (A - \sigma B)^{-1} Bq$, for a given vector q. For an efficient evaluation, we make an LU factorization of $(A - \sigma B)$. Since $A - \sigma B$ is assumed to be nonsingular, the factors L and U are also nonsingular. The factorization should be chosen so that the corresponding linear systems of equations with L, L^* and/or U, U^* can be solved efficiently, and typically, sparse LU factorizations are used.

4. **Cayley Transform**: With Shift-and-Invert it is possible to emphasize the converge towards eigenvalues close to the shift σ. The Cayley Transform goes one step further. It is defined by the operator

$$(A - \sigma B)^{-1}(A - \tau B).$$

To be more specific, let σ and τ be a user-selected shifts such that the matrix $A - \sigma B$ is nonsingular, then the original problem (5.23) can be transformed to

$$Cx = \mu x, \tag{5.27}$$

where

$$C = (A - \sigma B)^{-1}(A - \tau B) \quad \text{and} \quad \mu = \frac{\lambda - \tau}{\lambda - \sigma}.$$

The efficient implementation of the Cayley Transform can be done along similar lines as the Shift-and-Invert transformation. With the Cayley Transform we can emphasize the convergence towards eigenvalues close to σ while suppressing the influence of eigenvalues close to τ. It is a special form of the more general polynomial preconditioners by which we can accelerate the convergence towards wanted parts of the spectrum. For more details and references, see Saad [35].

The shift and invert spectral transformation technique is a powerful tool in the treatment of the generalized eigenvalue problem (5.23). The major problem, which often becomes a bottleneck, is to find a convenient LU factorization of $A - \sigma B$ so that the associated linear systems of equations can be solved efficiently. If accurate solution of the linear systems with $A - \sigma B$ becomes too expensive, then one may consider the usage of the Jacobi-Davidson method, or the application of inexact Cayley transforms (Lehoucq and Meerbergen [25], Meerbergen [27], Bai *et al* [26]).

5.10 Krylov subspace methods for linear systems

In this section we will shortly introduce how the Arnoldi, Lanczos and Bi-Lanczos algorithm lead to Krylov subspace methods for linear systems.

5.10.1 The Conjugate Gradient method

First we make some general remarks on minimization and projection and then see how the conjugate gradient (CG) method can be derived.

Minimization and projection

Suppose we have found in one or the other way an orthonormal basis for a linear subspace of dimension m, denoted by the columns of the matrix P, for which it holds that the reduction of the SPD matrix A of order n van be written as a tridiagonal matrix: $T_m = P^T AP$. Then for any $x \in R^n$ it holds that

$$\min_{y \in P}(x - y, A(x - y))$$

gives that $y^* \in P$ for which $((x - y^*), Ay) = 0$ for all $y \in P$. Here y^* is the projection of x on the space spanned by the columns of p under the innerproduct (u, Av).
We can write any y in P as $y = P\hat{y}$. With that the condition to find the minimum turns into $(P^T Ax - P^T AP\hat{y}^*, \hat{y}) = 0$ for all $\hat{y} \in R^m$, or equivalently $(P^T Ax - T_m\hat{y}^*, \hat{y}) = 0$ for all $\hat{y} \in R^m$. From this it follows that

$$T_m\hat{y}^* = P^T Ax \tag{5.28}$$

The above basis can be made in a variety of ways, among which the Lanczos algorithm.

Ritz and the CG method

Suppose we want to solve $Ax = b$, wit A SPD. Then (5.28) turns into $T_m\hat{y}^* = P^T b$. Clearly, T_m is also SPD and for this kind of matrices it is known that we can solve the system without pivoting. Now, suppose $T_m = LU$, then the system to be solved turns into $LU\hat{y}^* = P^T b$. When we define $x_m = P\hat{y}^*$ we can also write this as

$$x_m = PU^{-1}L^{-1}P^T b = \hat{P}\hat{b}$$

where $\hat{P} = PU^{-1}$ and $\hat{b} = L^{-1}P^Tb$. Note that $\hat{P}$ follows from the system $\hat{P}U = P$ and $\hat{b}$ from $L\hat{b} = P^Tb$. Since U is behind the unknown, the columns of $\hat{P}$ are solved from left to right. The elements of $\hat{b}$ are solved from top to bottom.
Now, note that if P is extended by one column, then as a result T_m, the LU-factorization, $\hat{P}$, and $\hat{b}$ are extended. Only the product of the new column of $\hat{P}$ and $\hat{b}$ need to be added to x_m in order to get x_{m+1}. So this defines an iterative procedure.
Looking back we may say that the approximate solution x_m is found by

$$\min_{y \in P}(x - y, A(x - y))$$

where x is the exact solution of $Ax = b$. Or equivalently by $\min_{y \in P}(x, b) + (y, Ay) - 2(y, b)$ and since x and b are constants this is equivalent to

$$\min_{y \in P}(y, Ay) - 2(y, b)$$

The last expression is just the Ritz approach to solve the problem $Ax = b$ as best as possible on the search space P. If P is a Krylov space associated to A and b then this Ritz approach can be build into the conjugate gradients method along the lines given above.
There are a few important properties which this method has.

1. The method is a direct method since after N steps, N being the order of the matrix, P is R^N which contains the solution.

2. As soon as the solution is in the Krylov subspace it is found.

3. The number of steps is at maximum equal to the number of different eigenvalues.

4. The iteration applied to the system $\tilde{A}\tilde{x} = \tilde{b}$, with $\tilde{A} = Q^TAQ$, $\tilde{x} = Q^Tx$, $\tilde{b} = Q^Tb$ and Q some orthogonal matrix converges in exactly the same way as on $Ax = b$. So even if we would diagonalize A the convergence will be the same.

For the CG method it can be shown that the following estimation of the error holds

$$\|x - x_m\|_A \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^m \|x\|_A \tag{5.29}$$

where $\|x\|_A = \sqrt{(x, Ax)}$. Here κ is the condition number in the two norm

$$\kappa = ||A||_2||A^{-1}||_2$$

which is due to the symmetry and positive definiteness equal to the ratio of the largest and smallest eigenvalue of A

Exercise 5.6 *Show that the above κ is also equal to the ratio of the largest and smallest eigenvalue of A*

Next to the condition number κ also the relative position of the eigenvalues of A are important. This derives immediately from the convergence of the Krylov subspace discussed earlier. The exterior eigenvalues are converging first and the speed of that convergence is at least that of the ratio of the second largest divided by a largest eigenvalue after an optimal shift. If these eigenvalues are converged to some extend the observed convergence is that based on the remaining eigenvalues. Repeating this argument we see that the CG method will converge faster and faster during the iteration. So in contrast with the stationary methods which converge linearly, the CG method converges super linearly (see [43] for more details).
The eigenvalue distribution can be influenced by preconditioning. Say K is a preconditioner then in fact the system

$$K^{-1}Ax = K^{-1}b$$

is solved. So if K is equal to A then we would have convergence in one step. However, then we need in fact a factorization of A, which we wanted to avoid. So the goal is to find a preconditioner that is relatively cheaply to apply. This appears to be possible and made the conjugate gradient method very popular for systems with SPD matrices. The conjugate gradients method was proposed by Hestenes and Stiefel in 1952 [17], but for a long time it was seen as a direct method, which had no advantages over factorization. Later one recognized its strength as iterative methods when preconditioned is applied.
The CG algorithm with preconditioning is shown in Fig.5.8. Of course application of

- Choose $\mathbf{x^{(0)}}$ and compute $\mathbf{r^{(0)}} = \mathbf{b} - \mathbf{Ax^{(0)}}$ and $\mathbf{z^{(0)}} = \mathbf{K^{-1}r^{(0)}}$
- For $n = 0, 1, 2, ...$

(i) $\mathbf{x^{(n+1)}} = \mathbf{x^{(n)}} + \alpha_\mathbf{n}\mathbf{z^{(n)}}$ with $\alpha_n = \dfrac{(\mathbf{r^{(n)}}, \mathbf{K^{-1}r^{(n)}})}{(\mathbf{z^{(n)}}, \mathbf{Az^{(n)}})}$

(ii) $\mathbf{r^{(n+1)}} = \mathbf{r^{(n)}} - \alpha_\mathbf{n}\mathbf{Az^{(n)}}$

(iii) $\mathbf{z^{(n+1)}} = \mathbf{K^{-1}r^{(n+1)}} + \beta_\mathbf{n}\mathbf{z^{(n)}}$ with $\beta_n = \dfrac{(\mathbf{r^{(n+1)}}, \mathbf{K^{-1}r^{(n+1)}})}{(\mathbf{r^{(n)}}, \mathbf{K^{-1}r^{(n)}})}$

Figure 5.8: CG algorithm with preconditioning

K^{-1} means solving a system with K.

5.10.2 Galerkin and FOM

If A is not symmetric but still positive definite, so $x^TAx/x^Tx > 0$ for all $x \neq 0$, then we cannot look for a minimum but we can still satisfy the same condition which yields the minimum in the symmetric case. So we look for y^* in P such that $(b - Ay^*, y) = 0$ for all $y \in P$. We require now that P is such that $H = P^TAP$ is a Hessenberg matrix. Such a P can be constructed by Arnoldi's algorithm. Also this could be worked out in an iterative method, but the L matrix will be dense now and therefore we have

to store the whole Krylov space. This method is not common used because there are alternatives which also work for general non-singular matrices. This approach is implemented in the Full Orthogonalization Method (FOM).

Exercise 5.7 *Given the* 2×2 *diagonal matrix with a 1 and -1 on the diagonal. Determine the one-dimensional subspace for which the Galerkin approximation of this matrix, i.e. the matrix reduced to this subspace, is singular.*

In general it is possible to construct a subspace such that the Galerkin approximation of a non-positive definite matrix A on the subspace is singular.

5.10.3 Petrov-Galerkin, BiCG and BiCGstab

Suppose we have now constructed bases P and Q such that $Q^T AP$ is tridiagonal. This can for example be obtained by the Bi-Lanczos algorithm.
We now want to find $y^* \in P$ (search space) such that $(b - Ay^*, z) = 0$ for all $z \in Q$ (test space). This leads to the solution of the reduced system $T\hat{y}^* = Q^T b$. This can be put into an iterative method in very much the same way as the CG method, which leads to the BiCG method if we choose as start vector b. The convergence of this method can be quite irregular and therefore the BiCGstab method was developed by Van der Vorst [44]. Still however the method may break down because the matrix T from the Bi-Lanczos process becomes singular. Furthermore a generalization was made called BiCGstab(l) in which more vectors from the past are used leading to a more robust method [40].

5.10.4 Least squares on the search space and GMRES

Suppose we have a basis P. We minimize now the residual over the space spanned by P, hence we are looking for the minimum of $||b - AP\hat{y}||_2$. This means that we want to solve a least squares solution on P. The standard way to solve such a system is by making a QR factorization, in this case of AP. Hence $||b - AP\hat{y}||_2 = ||Q^T b - R\hat{y}||_2$, where Q is an orthogonal square matrix and R trapezoidal. From this one easily solves $\hat{y}^*$.
Now suppose P is generated by the Arnoldi algorithm, hence it holds that $AP_k = P_{k+1}\tilde{H}$ where $\tilde{H}$ is a $(k+1) \times k$ Hessenberg matrix. Using this we can write $||b - AP_k\hat{y}||_2 = ||b - P_{k+1}\tilde{H}\hat{y}|| = \sqrt{||P_{k+1}^T b - \tilde{H}\hat{y}||_2^2 + C^2}$, where C is the length of the part of b that is in the orthogonal complement of P_{k+1}. Finally, we transform $\tilde{H}$ to upper triangular form using a QR factorization. Hence if $\tilde{H} = QR$ (NB Q is a square Hessenberg matrix and R has the same size as $\tilde{H}$) then $||P_{k+1}^T b - \tilde{H}\hat{y}||_2 = ||Q^T P_{k+1}^T b - R\hat{y}||_2$. Since in our case $x_0 = 0$, it holds that the first vector in P is the normalized vector b and therefore $Q^T P_{k+1}^T b = ||b||Q^T e_1$. Furthermore, by this $C = 0$. The minimizing solution is now found by solving the first k equations. As error remains the last element of $||b||Q^T e_1$, which is precisely the norm of the residual. Note that the actual computation of $\hat{y}^*$ can be postponed till this last element is small enough. This

algorithm is the famous Generalized Minimal Residual method (GMRES) developed by Saad and Schulz [38].
This method can be applied to any non-singular matrix. The disadvantage is that the whole Krylov space needs to be retained. One has overcome this by performing a restart after a fixed number of iterations.
An alternative would be to solve instead of $Ax = b$ the normal equations $A^T Ax = A^T b$. By construction the matrix $A^T A$ is SPD and we can apply the CG method. However, the condition number of this matrix is usually much worse than that of A. In the worst case it can even be the square of that of A. This slows down the convergence drastically and therefore this approach is not used in practice.

Recent developments

An overview of methods and some more can be found in [3] and in [37]. It also contains information how to get the associated Fortran, Matlab and C++ codes.
Recently, again some progress is made. A method called Induced Dimension Reduction method which was already introduced in 1980 is put into block form and appears to be an attractive alternative for GMRES and BiCGstab [41].
There exist also variants which allow a preconditioning that varies from step to step, e.g. FGMRES [36] and GMRESR [45].

5.11 Preconditoning

In the foregoing we have seen that the convergence of Krylov subspace methods depends largely on the condition number of the matrix or more specifically on the distribution of the eigenvalues. This can be influenced by preconditioning. So if we have a system $Ax = b$ then we like to find a matrix K for which $K^{-1}A$ has a better eigenvalue distribution and for which the system $K\mathbf{x} = \mathbf{b}$ is much easier to solve than the original system. With this K we replace the system to solve by $K^{-1}Ax = K^{-1}b$, or by $(K^{-1/2}AK^{-1/2})(K^{1/2}x) = K^{-1/2}b$ (for SPD A and K) and apply to this the Krylov subspace method. In the following we mention briefly a few preconditioning approaches. More information can be found in text books on the iterative solution of linear systems, e.g. [37]

5.11.1 Incomplete LU factorizations

If one makes an LU factorization of the matrix then the factors L and U contain usually a large amount of small entries (with respect to the biggest element occurring in it). It is attractive to drop these small elements. In general one can write

$$A = LU + R$$

where R is the rest or residual matrix. The standard approach is to just drop the elements, this is usually called incomplete LU (ILU) factorization. For M matrices it was shown by Meijerink and Van der Vorst [28], that this always brings the condition number down. However, one can also require that the factorization is exact for certain

test or probe vectors. So if v is a test vector then $Av = LUv$ or $Rv = 0$. For that we have to compensate the dropped element on the remaining elements. This is usually called Modified Incomplete LU (MILU) factorization. Of course for symmetric matrices we can make incomplete Cholesky factorizations. We mention a few well known methods in this class.

(M)ILU(0)

In ILU(0) no fill is allowed outside the pattern of the original matrix. So this is a very cheap factorization. The downside of this is that the speed up it brings is already not spectacular for an M-matrix. Changing it to a modified ILU by requiring that the factorization is exact for the constant vector gives a significant improvement for the M-matrix case, but may worsen the situation for others.
There also exists (M)ILU(k), which allows more fill and gives in general better results than (M)ILU(0).

ILUTP

ILUTP contains a drop parameter ϵ, the threshold (explaining the T in the name). Here a multiplier is not added to the L-factor if it less than ϵ times the norm of the current row in the original matrix. Similarly an element is not added to the U-factor if it satisfies the same criterion. This parameter adds a certain robustness since for ϵ is zero we have simply an exact factorization. So for ϵ small enough the factorization will improve the convergence.
Usually also pivoting is included in this method (explaining the P in the name). As with the direct methods one likes to preclude pivoting, because it may destroy the structure which is in the matrix. So here a value, say γ is given which allows pivoting if γ times the biggest element below the diagonal is bigger than the diagonal element. For $\gamma = 1$ we have standard partial pivoting and for smaller γ we are tempering the pivoting. Usually a value of $\gamma = 0.1$ is chosen.

MRILU

In this method also the freedom of ordering is exploited (hence the name Matrix Renumbering ILU). The aim is to order the matrix in a fill reducing way during the process such that the amount of dropped elements is decreased. These factorizations show near grid independent convergence for M-matrices. For more details see [6]. Another method in this class is ILUpack [5]

5.11.2 Sparse approximate inverse

Another idea is to not approximate the matrix but the inverse immediately. In general the inverse of an irreducible matrix is full. The game is to get a sparse approximation which is also a good approximation. It appears that one can quite easily construct approximate inverses using the Frobenius norm of a matrix which is just the square

root of the sum of all squares of the elements of the matrix. It holds that

$$||I - AM||_F^2 = \sum_{j=1}^{n} ||e_j - Am_j||_2^2$$

where e_j the elementary basis vector with a one at position j and zeros elsewhere and m_j is the j-th column of the matrix M. So if we want to find an M such that the Frobenius norm is less than a certain tolerance and moreover M is as sparse as possible, we can try to minimize the norms for the respective columns of M. One way to do this is by solving such a system by a few steps of the GMRES method.

5.11.3 Algebraic Multigrid

This is a big container of multilevel methods which build a multigrid like method using only the matrix. Hence, without information about the type of problem or the geometry. From a user stand point this is very attractive, but these methods are not easy to construct.

5.11.4 Preconditioner form of stationary methods

All the stationary methods mentioned in the beginning of this chapter can also be used as preconditioner. The recursion (5.2) can also be written as

$$x^{(n+1)} = (I - Q^{-1}A)x^{(n)} + Q^{-1}b = x^{(n)} + r^{(n)}$$

where $r^{(n)} = Q^{-1}(b - Ax^{(n)})$. Hence, an alternative formulation of the iteration, given $x^{(0)}$, is

$$\begin{aligned} r^{(n)} &= Q^{-1}(b - Ax^{(n)}) \\ x^{(n+1)} &= x^{(n)} + r^{(n)} \end{aligned}$$

It is clear that when $Q = A$ that $x^{(1)}$ will be the sought solution, hence the Q from (5.1) is the preconditioner here. In fact we can show that we iterate in the space $K^m(Q^{-1}A, r^{(0)})$ as long as m is less equal to the order of A and after that (we have reached the full space then) it will just continue to iterate in that space. In the case of Gauss-Seidel and SOR the associated preconditioners will give a non-symmetric matrix even if the original matrix is symmetric. This precludes the use of the CG method. However there exists a symmetrized variant of SOR (and hence of GS) called the SSOR where both upper and lower part of the matrix are used one after another such that the preconditioner becomes symmetric.

5.11.5 Vanka preconditioners

In some cases the matrix assumes the form

$$\begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix}$$

where A is positive definite but not necessarily symmetric and B a rectangular matrix with more rows than columns. Such a matrix is clearly not positive definite. One can even show that it has negative eigenvalues. Therefore it is said to be indefinite. It is also often called a saddle-point matrix. One could see this matrix as arising from a problem $Ax = b$ where a constraint is added $B^T x = 0$. From Section 2.3.2 we see that we arrive at the system matrix above. The idea of the Vanka preconditioner is to locally satisfy this constraint. For instance in the incompressible Navier-Stokes equation in the Finite Volume approach we find that the constraint means just that the numerical divergence on every mass control volume is zero. The Vanka preconditoner runs over all these control volumes and adapts the velocities on the boundaries such that divergence freedom is still satisfied in the control volume (in surrounding volumes it may be disturbed now). For details see [47].
Recently also a review paper on solution methods for Saddle Point problems has been published [4], which is an important starting point for anyone who wants to solve systems of equations in computational fluid dynamics.

```
== External links ==
* http://en.wikipedia.org/wiki/Preconditioning
```

Bibliography

[1] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenproblem. *Quart. Appl. Math.*, 9:17–29, 1951.

[2] K. E. Atkinson. *An Introduction to Numerical Analysis.* John Wiley and Sons, 1976.

[3] R. Barret, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R.Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, PA, 1994.

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

[5] M. Bollhöfer. A robust ILU with pivoting based on monitoring the growth of the inverse factors. *Lin. Alg. Appl*, 338:515–526, 2001.

[6] E.F.F. Botta and F.W. Wubs. MRILU: An effective algebraic multi-level ilu-preconditioner for sparse matrices. *SIAM J. on Matrix Anal. and Appl.*, pages 1007–1026, 1999.

[7] R.L. Burden and J.D. Faires. *Numerical Analysis.* Brooks/Cole, 2001.

[8] S. Chandrasekhar. *Hydrodynamic and Hydromagnetic Stability.* Clarendon Press, Oxford, U.K., 1961.

[9] H. A. Dijkstra. On the structure of cellular solutions in rayleigh-bénard-marangoni flows in small-aspect-ratio containers. *J. Fluid Mech.*, 243:73–102, 1992.

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

[11] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct methods for sparse matrices.* Oxford University Press, London, 1986.

[12] I.S. Duff, A.M. Erisman, and J.K. Reid. *Direct methods for sparse matrices.* Monographs on numerical analysis. Oxford science publications, 1986.

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

[14] A. Yu. Gelfgat. Different modes of Rayleigh-Benard instability in two- and three dimensional rectangular enclosures. *J. Comp. Physics*, 156:300–324, 1999.

[15] A. George. Nested dissection of a regular finite-element mesh. *SIAM J. Numer. Anal.*, 10:345–363, 1973.

[16] G.H. Golub and C.F. van Loan. *Matrix Computations*. Johns Hopkins, 3 edition, 1996.

[17] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.*, 49:409–436, 1954.

[18] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge Universiy Press, 1985.

[19] C. A. Katsman, M. J. Schmeits, and H. A. Dijkstra. Application of continuation methods in physical oceanography. In D. Henry and A. Bergeon, editors, *Continuation methods in Fluid Dynamics*, pages 155–166. Vieweg, 2000.

[20] H. B. Keller. Numerical solution of bifurcation and nonlinear eigenvalue problems. In P. H. Rabinowitz, editor, *Applications of Bifurcation Theory*. Academic Press, New York, U.S.A., 1977.

[21] E. L. Koschmieder. *Bénard Cells and Taylor Vortices*. Cambridge University Press, Cambridge, UK, 1993.

[22] E. L. Koschmieder and D. W. Switzer. The wavenumbers of supercritical surface-tension-driven Bénard convection. *J. Fluid Mech.*, 240:533–548, 1992.

[23] Y. A. Kuznetsov. *Elements of Applied Bifurcation Theory*. Springer Verlag, New York, U.S.A., 1995.

[24] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand*, 45:225–280, 1950.

[25] R. B. Lehoucq and K. Meerbergen. Using generalized Cayley transformations within an inexact rational Krylov sequence method. *SIAM J. Matrix Anal. Applic.*, 20:131–148, 1998.

[26] R. Lippert and A. Edelman. Nonlinear eigenvalue problems with orthogonality constraints. In Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems: a Practical Guide*. SIAM, Philadelphia, 2000. in press.

[27] K. Meerbergen. *Robust methods for the calculation of rightmost eigenvalues of nonsymmetric eigenvalue problems*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 1996.

[28] 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. Comput.*, 31:148–162, 1977.

[29] A. H. Nayfeh and B. Balachandran. *Applied Nonlinear Dynamics*. John Wiley, New York, U.S.A., 1995.

[30] D. A. Nield. Surface tension and buoyancy effects in cellular convection. *J. Fluid Mech.*, 19:341–352, 1964.

[31] R. Peyret and T.D. Taylor. *Computational methods for fluid flow*. Springer-Verlag, 1983.

[32] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Springer, 2007.

[33] R.D. Richtmyer and K.W. Morton. *Difference methods for initial value problems*. Interscience, 1967.

[34] P. Roache. *Computational Fluid Dynamics*. Hermosa Publishing, Albequerque, NM, U.S.A., 1976.

[35] Y. Saad. *Numerical methods for large eigenvalue problems*. Manchester University Press, Manchester, UK, 1992.

[36] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Statist. Comput.*, 14:461–469, 1993.

[37] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2 edition, 2003.

[38] Y. Saad and M.H. Schultz. A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[39] R. Seydel. *Practical Bifurcation and Stability Analysis: From Equilibrium to Chaos*. Springer-Verlag, New York, U.S.A., 1994.

[40] G.L.G. Sleijpen and D.R. Fokkema. Bicgstab(l) for linear equations involving unsymmetric matrices with complex spectrum. *Electron. Trans. Numer. Anal.*, pages 11–32, 1993.

[41] P. Sonneveld and M. van Gijzen. IDR(s): a family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM J. Sci. Comput.* to appear.

[42] W.F. Tinney and J.W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization. In *Proc. IEEE 55*, pages 1801–1809, 1967. Proceedings, Reading.

[43] A. van der Sluis and H.A. van der Vorst. The rate of convergence of conjugate gradients. *Numer. Math.*, 48:543–560, 1986.

[44] 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.

[45] H.A. van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods, 1992. to appear.

[46] J. J. Van Dorsselaer. Computing eigenvalues occurring in continuation methods with the Jacobi-Davidson QZ method. *J. Comp. Physics*, 138:714–733, 1997.

[47] S. Vanka. Block-implicit multigrid calculation of two-dimensional recirculating flows. *Comp. Meth. Appl. Mech. Eng.*, 59(1):29–48, 1986.

[48] R.S. Varga. *Matrix iterative analysis*. Prentice Hall, 1962.

[49] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, 1965.

[50] D.M. Young. *Iterative solution of large linear systems*. Academic Press, 1971.