Chapter 5

GMRES and MINRES

5.1 GMRES

As we have seen in Section 3.2, the minimal residual approach leads to a small minimum least squares problem that has to be solved:

$$H_{i+1,i}y = ||r_0||_2\, e_1.$$

In GMRES [151] this is done efficiently with Givens rotations, that annihilate the subdiagonal elements in the upper Hessenberg matrix $H_{i+1,i}$. The resulting upper triangular matrix is denoted by $R_{i,i}$:

$$H_{i+1,i} = Q_{i+1,i}R_{i,i},$$

where $Q_{i+1,i}$ denotes the product of the successive Givens eliminations of the elements $h_{j+1,j}$, for $j = 1, .., i$.

Exercise 5.1 *Consider the* 2 *by* 2 *matrix*

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}.$$

Construct the 2 *by* 2 *matrix*

$$Q = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix},$$

so that $B = QA$ has a zero element in its lower left corner: $b_{2,1} = 0$. Q is called a Givens transformation. What is the value of θ? Verify that $Q^TQ = I$.
Show that Givens rotations can be used to reduce $H_{i+1,i}$ to upper triangular form. What do the Givens transformations look like?

After the Givens transformations the least squares solution y minimizes

$$\begin{aligned} \|H_{i+1,i}y - \|r_0\|_2 e_1\|_2 &= \|Q_{i+1,i}R_{i,i}y - \|r_0\|_2 e_1\|_2 \\ &= \|R_{i,i}y - Q_{i+1,i}^T\|r_0\|_2 e_1\|_2. \end{aligned} \tag{5.1}$$

The resulting least squares problem leads to the minimum norm solution

$$y = R_{i,i}^{-1} Q_{i+1,i}^T \|r_0\|_2 e_1.$$

The required approximation x_i is now computed as $x_i = V_i y$.

Exercise 5.2 *Show that $R_{i,i}$ can not be singular, unless x_{i-1} is equal to the solution of $Ax = b$*

Note that when A is Hermitian (but not necessarily positive definite), the upper Hessenberg matrix $H_{i+1,i}$ reduces to a tridiagonal system. This simplified structure can be exploited in order to avoid storage of all the basis vectors for the Krylov subspace, in a way similar as has been pointed out for CG. The resulting method is known as MINRES [137], see Section 5.2.

In order to avoid excessive storage requirements and computational costs for the orthogonalization, GMRES is usually restarted after each m iteration steps. This algorithm is referred to as GMRES(m); the not-restarted version is often called 'full' GMRES. There is no simple rule to determine a suitable value for m; the speed of convergence may vary drastically for nearby values of m. It may be the case that GMRES($m+1$) is much more expensive than GMRES(m), even in terms of numbers of iterations.

We present in Fig.5.1 the modified Gram-Schmidt version of GMRES(m) for the solution of the linear system $Ax = b$. The application to preconditioned systems, for instance, $K^{-1}Ax = K^{-1}b$ is straight-forward.

For complex valued systems, the scheme is as in Fig. 5.2. Note that the complex rotation is the only difference with respect to the real version.

Exercise 5.3 *For a real unsymmetric nondefective matrix A there exists a nonsingular real matrix Z so that*

$$Z^{-1}AZ = T,$$

where T is a real block diagonal matrix with 2 by 2 blocks along the diagonal. The eigenvalues of A are equal to those of T.
This can be used to construct test examples for GMRES. Would it be sufficient to test GMRES with diagonal matrices, as was the case for Conjugate Gradients? Why not?

$r = b - Ax_0$, for a given initial guess x_0
$x = x_0$
for $j = 1, 2, ...$
    $\beta = ||r||_2$, $v_1 = r/\beta$, $\widehat{b} = \beta e_1$
    for $i = 1, 2, ..., m$
        $w = Av_i$
        for $k = 1, ..., i$
            $h_{k,i} = v_k^T w$, $w = w - h_{k,i} v_k$
        $h_{i+1,i} = ||w||_2$, $v_{i+1} = w/h_{i+1,i}$
        $r_{1,i} = h_{1,i}$
        for $k = 2, .., i$
            $\gamma = c_{k-1} r_{k-1,i} + s_{k-1} h_{k,i}$
            $r_{k,i} = -s_{k-1} r_{k-1,i} + c_{k-1} h_{k,i}$
            $r_{k-1,i} = \gamma$
        $\delta = \sqrt{r_{i,i}^2 + h_{i+1,i}^2}$ $c_i = r_{i,i}/\delta$, $s_i = h_{i+1,i}/\delta$
        $r_{i,i} = c_i r_{i,i} + s_i h_{i+1,i}$
        $\widehat{b}_{i+1} = -s_i \widehat{b}_i$ $\widehat{b}_i = c_i \widehat{b}_i$
        $\rho = |\widehat{b}_{i+1}|$ $(= ||b - Ax_{(j-1)m+i}||_2))$
        if ρ is small enough **then**
            ($n_r = i$, goto *SOL*)
    $n_r = m$, $y_{n_r} = \widehat{b}_{n_r}/r_{n_r,n_r}$
SOL: **for** $k = n_r - 1, ..., 1$
        $y_k = (\widehat{b}_k - \sum_{i=k+1}^{n_r} r_{k,i} y_i)/r_{k,k}$
    $x = x + \sum_{i=1}^{n_r} y_i v_i$, **if** ρ small enough quit
    $r = b - Ax$

Figure 5.1: unpreconditioned GMRES(m) with modified Gram-Schmidt

Exercise 5.4 *The FOM method for real unsymmetric systems is based on the Galerkin relations:*

$$b - Ax_m \perp \mathcal{K}^m(A, x_0),$$

for $x_m \in \mathcal{K}^m(A, x_0)$.
Modify the scheme for (the real version of) GMRES(m) so that we obtain the restarted version of FOM: FOM(m).

$r = b - Ax_0$, for a given initial guess x_0
$x = x_0$
for $j = 1, 2, ...$
    $\beta = ||r||_2$, $v_1 = r/\beta$, $\widehat{b} = \beta e_1$
    for $i = 1, 2, ..., m$
        $w = Av_i$
        for $k = 1, ..., i$
            $h_{k,i} = v_k^* w$, $w = w - h_{k,i} v_k$
        $h_{i+1,i} = ||w||_2$, $v_{i+1} = w/h_{i+1,i}$
        $r_{1,i} = h_{1,i}$
        for $k = 2, .., i$
            $\gamma = c_{k-1} r_{k-1,i} + \bar{s}_{k-1} h_{k,i}$
            $r_{k,i} = -s_{k-1} r_{k-1,i} + c_{k-1} h_{k,i}$
            $r_{k-1,i} = \gamma$
        $\delta = \sqrt{|r_{i,i}|^2 + |h_{i+1,i}|^2}$
        if $|r_{i,i}| < |h_{i+1,i}|$
        then $\mu = r_{i,i}/h_{i+1,i}$, $\tau = \bar{\mu}/|\mu|$
        else $\mu = h_{i+1,i}/r_{i,i}$, $\tau = \mu/|\mu|$
        $c_i = |r_{i,i}|/\delta$, $s_i = |h_{i+1,i}|\tau/\delta$
        $r_{i,i} = c_i r_{i,i} + \bar{s}_i h_{i+1,i}$
        $\widehat{b}_{i+1} = -s_i \widehat{b}_i$ $\widehat{b}_i = c_i \widehat{b}_i$
        $\rho = |\widehat{b}_{i+1}|$ $(= ||b - Ax_{(j-1)m+i}||_2))$
        if ρ is small enough **then**
            $(n_r = i$, goto $SOL)$
    $n_r = m$, $y_{n_r} = \widehat{b}_{n_r}/r_{n_r,n_r}$
SOL: **for** $k = n_r - 1, ..., 1$
        $y_k = (\widehat{b}_k - \sum_{i=k+1}^{n_r} r_{k,i} y_i)/r_{k,k}$
    $x = x + \sum_{i=1}^{n_r} y_i v_i$, **if** ρ small enough quit
    $r = b - Ax$

Figure 5.2: unpreconditioned GMRES(m) for complex systems

Exercise 5.5 *Test the methods GMRES(m) and FOM(m). Run the tests with only one cycle of m steps. Construct matrices with complex conjugate eigenvalues with positive real part.*

Plot the log *of the norms of the first m residuals. Why do the norms for GMRES form a strictly monotonic decreasing curve? Should that also be the case for FOM? Do you observe situations where $\|r_i^{GMRES}\|_2 \approx \|r_i^{FOM}\|_2$ for some values of i? Could such situations be expected after inspection of the matrix $H_{i+1,i}$?*

Exercise 5.6 *Repeat the experiments for GMRES and FOM with real indefinite matrices (eigenvalues with positive real parts and eigenvalues with negative real parts). Can we modify a given test example so that FOM breaks down at step i? Hint: If $H_{i+1,i}$ is the reduced matrix for A, what is the reduced matrix for $A - \sigma I$? How can we expect a break-down step from inspection of $R_{i,i}$.*
What happens to GMRES at a break down step of FOM? Can FOM be continued after a break down? Modify FOM so that it computes only the approximated solutions at non break-down steps.

Another scheme for GMRES, based upon Householder orthogonalization instead of modified Gram-Schmidt has been proposed in [188]. For certain applications it seems attractive to invest in additional computational work in turn for improved numerical properties: the better orthogonality might save iteration steps.

The eigenvalues of $H_{i,i}$ are the Ritz values of A with respect to the Krylov subspace spanned by v_1, ..., v_i. They approximate eigenvalues of A increasingly well for increasing dimension i.

Exercise 5.7 *Construct some example systems for the testing of GMRES, such that the eigenvalues of the matrices are known. Plot the convergence curve:* $\log(\|r_i\|_2)$ *as function of i. Compute also the Ritz values for some values of i. Is there any relation between the Ritz values and the eigenvalues of the matrix of the linear system? Which eigenvalues are well approximated? Does that depend on the distribution of the eigenvalues of the matrix? Is there a relation between the convergence of Ritz values and effects in the convergence curve?*

There is an interesting and simple relation between the Ritz-Galerkin approach (FOM and CG) and the minimum residual approach (GMRES and MINRES). In GMRES the projected system matrix $H_{i+1,i}$ is transformed by Givens rotations to an upper triangular matrix (with last row equal to zero). So, in fact, the major difference between FOM and GMRES is that in

FOM the last (($i+1$)-th row is simply discarded, while in GMRES this row is rotated to a zero vector. Let us characterize the Givens rotation, acting on rows i and $i+1$, in order to zero the element in position $(i+1, i)$, by the sine s_i and the cosine c_i. Let us further denote the residuals for FOM with an superscript F and those for GMRES with superscript G. Then we have the following relation between FOM and GMRES:
If $c_k \neq 0$ then the FOM and the GMRES residuals are related by

$$\|r_k^F\|_2 = \frac{\|r_k^G\|_2}{\sqrt{1 - (\|r_k^G\|_2 / \|r_{k-1}^G\|_2)^2}}, \tag{5.2}$$

([45]: theorem 3.1). From this relation we see that when GMRES has a significant reduction at step k, in the norm of the residual (i.e., s_k is small, and $c_k \approx 1$), then FOM gives about the same result as GMRES. On the other hand when FOM has a breakdown ($c_k = 0$), then GMRES does not lead to an improvement in the same iteration step. Because of these relations we can link the convergence behaviour of GMRES with the convergence of Ritz values (the eigenvalues of the "FOM" part of the upper Hessenberg matrix). This has been exploited in [179], for the analysis and explanation of local effects in the convergence behaviour of GMRES.

There are various methods that are mathematically equivalent with FOM or GMRES. We will say that two methods are mathematically equivalent if they produce the same approximations $\{x_k\}$ in exact arithmetic. Among those that are equivalent to GMRES are: Orthomin [186], Orthodir [112], GENCR [70], and Axelsson's method [9]. These methods are often more expensive than GMRES per iteration step and they may also be less robust. Orthomin is still popular, since this variant can be easily truncated (Orthomin(s)), in contrast to GMRES. The truncated and restarted versions of these algorithms are not necessarily mathematically equivalent.
Methods that are mathematically equivalent to FOM are: Orthores [112] and GENCG [38, 193]. In these methods the approximate solutions are constructed such that they lead to orthogonal residuals (which form a basis for the Krylov subspace; analogously to the CG method).

The GMRES method and FOM are closely related to vector extrapolation methods, when the latter are applied to linearly generated vector sequences. For a discussion on this, as well as for implementations for these matrix free methods, see [153]. For an excellent overview of GMRES and related variants, such as FGMRES, see [150].

5.2 MINRES

When A is symmetric, then the matrix $H_{i+1,i}$ reduces to a tridiagonal matrix $T_{i+1,i}$. This property can be exploited to obtain short recurrence relations. As with GMRES, we look for an

$$x_i \in \{r_0, Ar_0, \ldots, A^{i-1}r_0\}, \quad x_i = R_i\bar{y}$$

$$\begin{aligned}\|Ax_i - b\|_2 &= \|AR_i\bar{y} - b\|_2 \\ &= \|R_{i+1}T_{i+1,i}y - b\|_2,\end{aligned}$$

such that this residual norm is minimal. Now we exploit the fact that $R_{i+1}D_{i+1}^{-1}$, with

$$D_{i+1} = diag(\|r_0\|_2, \|r_1\|_2, ..., \|r_i\|_2),$$

is an orthonormal transformation with respect to the current Krylov subspace:

$$\|Ax_i - b\|_2 = \|D_{i+1}T_{i+1,i}y - \|r_0\|_2e_1\|_2$$

and this final expression can simply be seen as a minimum norm least squares problem.
The element in the $(i+1, i)$ position of $T_{i+1,i}$ can be transformed to zero by a simple Givens rotation and the resulting upper tridiagonal system (the other subdiagonal elements being removed in previous iteration steps) can simply be solved, as we will show now.

The effect of the Givens transformations is that $T_{i+1,i}$ is decomposed in QR-form:

$$T_{i+1,i} = Q_{i+1,i}R_{i,i},$$

in which the orthogonal matrix $Q_{i+1,i}$ is the product of the Givens transformations and $R_{i,i}$ is an upper triangular matrix with three non-zero diagonals. We can exploit the banded structure of $R_{i,i}$, for the computation of x_i.
The solution of $T_{i+1,i}y = ||r_0||_2e_1$ can be written as

$$y = R_{i,i}^{-1}Q_{i+1,i}^T||r_0||_2e_1,$$

so that the solution x_i is obtained as

$$x_i = (V_iR_{i,i}^{-1})(Q_{i+1,i}^T\|r_0\|_2e_1).$$

We first compute the matrix $W_k = V_kR_k^{-1}$, and it is easy to see that the last column of W_k is obtained from the last 3 columns of V_k. The vector

$z_i \equiv Q_{i+1,i}^T \|r_0\|_2 e_1$ can also be updated by a short recurrence, since z_i follows from a simple Givens rotation on the last two coordinates of z_{i-1}.
These two coupled recurrence relations lead to MINRES [137].

This leads to the algorithmic form of MINRES in Fig. 5.3, which has been inspired by a MATLAB routine published in [75].

Compute $v_1 = b - Ax_0$ for some initial guess x_0
$\beta_1 = ||v_1||_2$; $\eta = \beta_1$;
$\gamma_1 = \gamma_0 = 1$; $\sigma_1 = \sigma_0 = 0$;
$v_0 = 0$; $w_0 = w_{-1} = 0$;
for $i = 1, 2, \ldots.$
The Lanczos recurrence:
$v_i = \frac{1}{\beta_i} v_i$; $\alpha_i = v_i^T A v_i$;
$v_{i+1} = Av_i - \alpha_i v_i - \beta_i v_{i-1}$
$\beta_{i+1} = ||v_{i+1}||_2$
QR part:
old Givens rot's on new column of T:
$\delta = \gamma_i \alpha_i - \gamma_{i-1} \sigma_i \beta_i$; $\rho_1 = \sqrt{\delta^2 + \beta_{i+1}^2}$
$\rho_2 = \sigma_i \alpha_i + \gamma_{i-1} \gamma_i \beta_i$; $\rho_3 = \sigma_{i-1} \beta_i$
New Givens rotation for subdiag elt:
$\gamma_{i+1} = \delta / \rho_1$; $\sigma_{i+1} = \beta_{i+1} / \rho_1$
Update of solution (with $W_i = V_i R_{i,i}^{-1}$)
$w_i = (v_i - \rho_3 w_{i-2} - \rho_2 w_{i-1}) / \rho_1$
$x_i = x_{i-1} + \gamma_{i+1} \eta w_i$
$||r_i||_2 = |\sigma_{i+1}| ||r_{i-1}||_2$
check convergence; continue if necessary
$\eta = -\sigma_{i+1} \eta$
end

Figure 5.3: The unpreconditioned MINRES algorithm

MINRES is attractive when the symmetric matrix A is symmetric and indefinite. In the positive definite case, CG is the preferred method.
For a symmetric positive definite preconditioner of the form LL^T, the MIN-

RES algorithm can be applied to the explicitly preconditioned system

$$L^{-1}AL^{-T}A\widetilde{x} = L^{-1}b, \quad \text{with} \quad x = L^{-1}\widetilde{x}.$$

Explicit inversions with L and L^T can be avoided, all one has to do is solve linear systems with these matrices.
We can not, without risk, apply MINRES to $K^{-1}Ax = K^{-1}b$, for symmetric K and A, when the matrix $K^{-1}A$ is unsymmetric. It does not help to replace the innerproduct by the bilinear form (x, Ky), with respect to which the matrix $K^{-1}A$ is symmetric, since this bilinear form does not define an innerproduct if K is not positive definite.
The construction of effective preconditioners for symmetric indefinite A is largely an open problem.

With respect to parallelism, or other implementation aspects, MINRES can be treated as CG. Note that most of the variables in MINRES may overwrite old ones that are obsolete.

The usage of the 3-term recurrence relation for the columns of W_i makes MINRES very vulnerable for rounding errors, as has been shown in [158]. It has been shown that rounding errors are propagated to the approximate solution with a factor proportional to the square of the condition number of A, whereas in GMRES these errors depend only on the condition number itself. Therefore, one should be careful with MINRES for ill-conditioned systems. If storage is no problem then GMRES should be preferred for ill-conditioned systems; if storage is a problem then one might consider the usage of SYMMLQ [137]. SYMMLQ, however, may converge a good deal slower than MINRES for ill-conditioned systems. For more details on this, see [158].

5.3 Rank-one updates for the Matrix Splitting

Iterative methods can be derived from a splitting of the matrix, and we have used the very simple splitting $A = I - R$, with $R = I - A$, in order to derive the projection type methods. In [68] it is suggested to update the matrix splitting with information obtained in the iteration process. We will give the flavour of this method here since it turns out that it has an interesting relation with GMRES. This relation is exploited in [180] for the construction of new classes of GMRES-like methods, that can be used as cheap alternatives for the increasingly expensive full GMRES method.

Assume that the matrix splitting in the k-th iteration step is given by $A = H_k^{-1} - R_k$, then we obtain the iteration formula

$$x_k = x_{k-1} + H_k r_{k-1} \quad \text{with} \quad r_k = b - Ax_k.$$

The idea is now to construct H_k by a suitable rank-one update to H_{k-1}:

$$H_k = H_{k-1} + u_{k-1} v_{k-1}^T,$$

which leads to

$$x_k = x_{k-1} + (H_{k-1} + u_{k-1} v_{k-1}^T) r_{k-1} \tag{5.3}$$

or

$$\begin{aligned} r_k &= r_{k-1} - A(H_{k-1} + u_{k-1} v_{k-1}^T) r_{k-1} \\ &= (I - AH_{k-1}) r_{k-1} - A u_{k-1} v_{k-1}^T r_{k-1} \\ &= (I - AH_{k-1}) r_{k-1} - \mu_{k-1} A u_{k-1}. \end{aligned} \tag{5.4}$$

The optimal choice for the update would have been to select u_{k-1} such that

$$\mu_{k-1} A u_{k-1} = (I - AH_{k-1}) r_{k-1},$$

or

$$\mu_{k-1} u_{k-1} = A^{-1} (I - AH_{k-1}) r_{k-1}.$$

However, A^{-1} is unknown and the best approximation we have for it is H_{k-1}. This leads to the choice

$$\bar{u}_{k-1} = H_{k-1} (I - AH_{k-1}) r_{k-1}. \tag{5.5}$$

The constant μ_{k-1} is chosen such that $\|r_k\|_2$ is minimal as a function of μ_{k-1}. This leads to

$$\mu_{k-1} = \frac{1}{\|A\bar{u}_{k-1}\|_2^2} (A\bar{u}_{k-1})^T (I - AH_{k-1}) r_{k-1}.$$

Since v_{k-1} has to be chosen such that $\mu_{k-1} = v_{k-1}^T r_{k-1}$, we have the following obvious choice for it

$$\bar{v}_{k-1} = \frac{1}{\|A\bar{u}_{k-1}\|_2^2} (I - AH_{k-1})^T A\bar{u}_{k-1} \tag{5.6}$$

(note that from the minimization property we have that $r_k \perp A\bar{u}_{k-1}$).

In principle the implementation of the method is quite straight forward, but note that the computation of r_{k-1}, $\bar{u}_{k-1}$ and $\bar{v}_{k-1}$ costs 4 matrix vector multiplications with A (and also some with H_{k-1}). This would make the method too expensive for being of practical interest. Also the updated splitting is most likely a dense matrix if we carry out the updates explicitly. We will now show, still following the lines set forth in [68], that there are orthogonality properties, following from the minimization step, by which the method can be implemented much more efficiently.
We define
1. $c_k = \frac{1}{\|A\bar{u}_k\|_2} A\bar{u}_k$ (note that $r_{k+1} \perp c_k$)
2. $E_k = I - AH_k$

From (5.4) we have that $r_k = E_k r_{k-1}$, and from (5.5):

$$A\bar{u}_k = AH_k E_k r_k = \alpha_k c_k$$

or

$$c_k = \frac{1}{\alpha_k}(I - E_k)E_k r_k = \frac{1}{\alpha_k}E_k(I - E_k)r_k. \tag{5.7}$$

Furthermore:

$$\begin{aligned} E_k &= I - AH_k = I - AH_{k-1} - A\bar{u}_{k-1}\bar{v}_{k-1}^T & (5.8)\\ (5.5) \Rightarrow \quad &= I - AH_{k-1} - A\bar{u}_{k-1}(A\bar{u}_{k-1})^T(I - AH_{k-1})\frac{1}{\|A\bar{u}_k\|_2^2} \\ &= (I - c_{k-1}c_{k-1}^T)E_{k-1} \\ &= \prod_{i=0}^{k-1}(I - c_i c_i^T)E_0. \end{aligned}$$

We see that the operator E_k has the following effect on a vector. The vector is multiplied by E_0 and then orthogonalized with respect to c_0, ..., c_{k-1}. Now we have from (5.7) that

$$c_k = \frac{1}{\alpha_k}E_k y_k,$$

and hence

$$c_k \perp c_0, ..., c_{k-1}. \tag{5.9}$$

A consequence from (5.9) is that

$$\prod_{j=0}^{k-1}(I - c_j c_j^T) = I - \sum_{j=0}^{k-1} c_j c_j^T = I - P_{k-1}$$

and therefore

$$P_k = \sum_{j=0}^{k} c_j c_j^T. \tag{5.10}$$

The actual implementation is based on the above properties. Given r_k we compute r_{k+1} as follows (and we update x_k in the corresponding way):

$$r_{k+1} = E_{k+1} r_k.$$

With $\xi^{(0)} = E_0 r_k$ we first compute (with the c_j from previous steps):

$$E_k r_k = \xi^{(k)} \equiv (I - \sum_{j=0}^{k-1} c_j c_j^T)\xi^{(0)} = \prod_{j=0}^{k-1}(I - c_j c_j^T)\xi^{(0)}.$$

The expression with $\sum$ leads to a Gram-Schmidt formulation, the expression with $\prod$ leads to the Modified Gram-Schmidt variant.
The computed updates $-c_j^T \xi^{(0)} c_j$ for r_{k+1} correspond to updates

$$c_j^T \xi^{(0)} A^{-1} c_j = c_j^T \xi^{(0)} u_j / \|Au_j\|_2$$

for x_{j+1}. These updates are in the scheme, given below, represented by η.

From (5.5) we know that

$$\bar{u}_k = H_k E_k r_k = H_k \xi^{(k)}.$$

Now we have to make $A\bar{u}_k \sim c_k$ orthogonal w.r.t. c_0, ..., c_{k-1}, and to update $\bar{u}_k$ accordingly. Once we have done that we can do the final update step to make H_{k+1}, and we can update both x_k and r_k by the corrections following from including c_k. The orthogonalization step can be carried out easily as follows. Define $c_k^{(k)} \equiv \alpha_k c_k = AH_k E_k r_k = (I - E_k)E_k r_k$ (see (5.7)) $= (I - E_0 + P_{k-1}E_0)\xi^{(k)}$ (see (5.8)) $= AH_0\xi^{(k)} + P_{k-1}(I - AH_0)\xi^{(k)} = c_k^{(0)} + P_{k-1}\xi^{(k)} - P_{k-1}c_k^{(0)}$. Note that the second term vanishes since $\xi^{(k)} \perp c_0, ..., c_{k-1}$.

The resulting scheme for the k-th iteration step becomes:

1. $\xi^{(0)} = (I - AH_0)r_k$; $\eta^{(0)} = H_0 r_k$;
   for $i = 0, ..., k-1$ do
   $\alpha_i = c_i^T \xi^{(i)}$; $\xi^{(i+1)} = \xi^{(i)} - \alpha_i c_i$; $\eta^{(i+1)} = \eta^{(i)} + \alpha_i u_i$;
2. $u_k^{(0)} = H_0 \xi^{(k)}$; $c_k^{(0)} = Au_k^{(0)}$;
   for $i = 0, ..., k-1$ do

$$\beta_i = -c_i^T c_k^{(i)};\ c_k^{(i+1)} = c_k^{(i)} + \beta_i c_i;\ u_k^{(i+1)} = u_k^{(i)} + \beta_i u_i;$$
$$c_k = c_k^{(k)}/\|c_k^{(k)}\|_2;\ u_k = u_k^{(k)}/\|c_k^{(k)}\|_2;$$
3. $$x_{k+1} = x_k + \eta^{(k)} + u_k c_k^T \xi^{(k)};$$
$$r_{k+1} = (I - c_k c_k^T)\xi^{(k)};$$

Remarks

1. The above scheme is a Modified Gram Schmidt variant, given in [180], of the original scheme in [68].

2. If we keep H_0 fixed, i.e., $H_0 = I$, then the method is not scaling invariant (the results for $\rho Ax = \rho b$ depend on ρ). In [180] a scaling invariant method is suggested.

3. Note that in the above implementation we have 'only' two matrix vector products per iteration step. In [180] it is shown that in many cases we may also expect comparable converge as for GMRES in half the number of iteration steps.

4. A different choice for $\bar{u}_{k-1}$ does not change the formulas for $\bar{v}_{k-1}$ and E_{k-1}. For each different choice we can derive similar schemes as the one above.

5. From (5.4) we have

$$r_k = r_{k-1} - AH_{k-1}r_{k-1} - \mu_{k-1}Au_{k-1}.$$

In view of the previous remark we might also make the different choice $\bar{u}_{k-1} = H_{k-1}r_{k-1}$. With this choice, we obtain a variant which is algebraically identical to GMRES (for a proof of this see [180]). This GMRES variant is obtained by the following changes in the previous scheme:
Take $H_0 = 0$ (note that in this case we have that $E_{k-1}r_{k-1} = r_{k-1}$, and hence we we may skip part 1 of the above algorithm), and set $\xi^{(k)} = r_k$, $\eta^{(k)} = 0$. In step 2 start with $u_k^{(0)} = \xi^{(k)}$.
The result is a different formulation of GMRES in which we can obtain explicit formulas for the updated preconditioner (i.e., the inverse of A is approximated increasingly well): The update for H_k is $\bar{u}_k c_k^T E_k$ and the sum of these updates gives an approximation for A^{-1}.

6. Also in this GMRES-variant we are still free to select u_k a little bit different. Remember that the leading factor H_{k-1} in (5.5) was introduced as an approximation for the actually desired A^{-1}. With $\bar{u}_{k-1} = A^{-1} r_{k-1}$, we would have that $r_k = E_{k-1} r_{k-1} - \mu_{k-1} r_{k-1} = 0$ for the minimizing μ_{k-1}. We could take other approximations for the inverse (with respect to the given residual r_{k-1}), e.g., the result vector y obtained by a few steps GMRES applied to $Ay = r_{k-1}$. This leads to the so-called GMRESR family of nested methods (for details see [180]). See also section 5.4. A similar algorithm, named FGMRES, has been derived independently by Saad [149]. In FGMRES the search directions are preconditioned, whereas in GMRESR the residuals are preconditioned. This gives GMRESR direct control over the reduction in norm of the residual. As a result GMRESR can be made robust while FGMRES may suffer from break-down. A further disadvantage of the FGMRES formulation is that this method cannot be truncated, or selectively orthogonalized, as GMRESR can.
   In [12] a generalized conjugate gradient method is proposed, a variant of which produces in exact arithmetic identical results as the proper variant of GMRESR, though at higher computational costs and with a classical Gram–Schmidt orthogonalization process instead of the modified process as in GMRESR.

5.4 GMRESR and GMRES$\star$

In [180] it has been shown that the GMRES-method can be effectively combined (or rather preconditioned) with other iterative schemes. The iteration steps of GMRES (or GCR) are called outer iteration steps, while the iteration steps of the preconditioning iterative method are referred to as inner iterations. The combined method is called GMRES$\star$, where $\star$ stands for any given iterative scheme; in the case of GMRES as the inner iteration method, the combined scheme is called GMRESR[180].
Similar schemes have been proposed recently. In FGMRES[149] the update directions for the approximate solution are preconditioned, whereas in GMRES$\star$ the residuals are preconditioned. The latter approach offers more control over the reduction in the residual, in particular break-down situations can be easily detected and remedied.
In exact arithmetic GMRES$\star$ is very close to the Generalized Conjugate Gradient method[12]; GMRES$\star$, however, leads to a more efficient computational scheme.

The GMRES⋆ algorithm can be described by the computational scheme in Fig. 5.4.

x_0 is an initial guess; $r_0 = b - Ax_0$;
for $i = 0, 1, 2, 3, ...$
    Let $z^{(m)}$ be the approximate solution of $Az = r_i$
    obtained after m steps of an iterative method.
    $c = Az^{(m)}$ (often available from the iterative method)
    for $k = 0, ..., i-1$
        $\alpha = (c_k, c)$
        $c = c - \alpha c_k$
        $z^{(m)} = z^{(m)} - \alpha u_k$
    $c_i = c/\|c\|_2$; $u_i = z^{(m)}/\|c\|_2$
    $x_{i+1} = x_i + (c_i, r_i)u_i$
    $r_{i+1} = r_i - (c_i, r_i)c_i$
    if x_{i+1} is accurate enough then quit
end

Figure 5.4: The GMRES⋆ algorithm

A sufficient condition to avoid break-down in this method ($\|c\|_2 = 0$) is that the norm of the residual at the end of an inner iteration is smaller than the right-hand residual: $\|Az^{(m)} - r_i\|_2 < \|r_i\|_2$. This can easily be controlled during the inner iteration process. If stagnation occurs, i.e. no progress at all is made in the inner iteration, then it is suggested by Van der Vorst and Vuik[180] to do one (or more) steps of the LSQR method, which guarantees a reduction (but this reduction is often only small).

The idea behind this combined iteration scheme is that we explore parts of high-dimensional Krylov subspaces, hopefully localizing the same approximate solution that full GMRES would find over the entire subspace, but now at much lower computational costs. The alternatives for the inner iteration could be either one cycle of GMRES(m), since then we have also locally an optimal method, or some other iteration scheme, like for instance Bi-CGSTAB. As has been shown by Van der Vorst[178] there are a number of different situations where we may expect stagnation or slow convergence for GMRES(m). In such cases it does not seem wise to use this method.

On the other hand it may also seem questionable whether a method like Bi-CGSTAB should lead to success in the inner iteration. This method does not satisfy a useful global minimization property and large part of its effectiveness comes from the underlying Bi-CG algorithm, which is based on bi-orthogonality relations. This means that for each outer iteration the inner iteration process has to build a bi-orthogonality relation again. It has been shown for the related Conjugate Gradients method that the orthogonality relations are determined largely by the distribution of the weights at the lower end of the spectrum and on the isolated eigenvalues at the upper end of the spectrum[170]. By the nature of these kind of Krylov processes the largest eigenvalues and their corresponding eigenvector components quickly do enter the process after each restart, and hence it may be expected that much of the work is lost in rediscovering the same eigenvector components in the error over and over again, whereas these components may already be so small that further reduction in those directions in the outer iteration is waste of time, since it hardly contributes to a smaller norm of the residual. This heuristic way of reasoning may explain in part our rather disappointing experiences with Bi-CGSTAB as the inner iteration process for GMRES$\star$.

De Sturler and Fokkema[50] propose to prevent the outer search directions explicitly from being reinvestigated again in the inner process. This is done by keeping the Krylov subspace that is build in the inner iteration orthogonal with respect to the Krylov basis vectors generated in the outer iteration. The procedure works as follows.
In the outer iteration process the vectors c_0, ..., c_{i-1} build an orthogonal basis for the Krylov subspace. Let C_i be the n by i matrix with columns c_0, ..., c_{i-1}. Then the inner iteration process at outer iteration i is carried out with the operator A_i instead of A, and A_i is defined as

$$A_i = (I - C_i C_i^T)A. \tag{5.11}$$

It is easily verified that $A_i z \perp c_0, ..., c_{i-1}$ for all z, so that the inner iteration process takes place in a subspace orthogonal to these vectors. The additional costs, per iteration of the inner iteration process, are i inner products and i vector updates. In order to save on these costs, one should realize that it is not necessary to orthogonalize with respect to all previous c-vectors, and that "less effective" directions may be dropped, or combined with others. De Sturler and Fokkema[50] suggestions are made for such strategies. Of course, these strategies in cases where we see too little residual reducing effect in the inner iteration process in comparison with the outer iterations of GMRES$\star$.

Chapter 6

Bi-Conjugate Gradients

6.1 Derivation of the method

We may proceed in a similar way as in the symmetric case:

$$AV_i = V_{i+1}T_{i+1,i}, \tag{6.1}$$

but here we use the matrix $W_i = [w_1, w_2, ..., w_i]$ for the projection of the system

$$W_i^T(b - Ax_i) = 0,$$

or

$$W_i^T AV_i y - W_i^T b = 0.$$

Using (6.1), we find that y_i satisfies

$$T_{i,i}y = \|r_0\|_2 e_1,$$

and $x_i = V_i y$. The resulting method is known as the Bi-Lanczos method [119].

We have assumed that $d_i \neq 0$, that is $w_i^T v_i \neq 0$. The generation of the bi-orthogonal basis breaks down if for some i the value of $w_i^T v_i = 0$, this is referred to in literature as a *serious breakdown*. Likewise, when $w_i^T v_i \approx 0$, we have a near-breakdown. The way to get around this difficulty is the so-called Look-ahead strategy, which comes down to taking a number of successive basis vectors for the Krylov subspace together and to make them blockwise bi-orthogonal. This has been worked out in detail in [140, 79, 80, 81].

Another way to avoid breakdown is to restart as soon as a diagonal element gets small. Of course, this strategy looks surprisingly simple, but one should realise that at a restart the Krylov subspace, that has been built up so far, is thrown away, and this destroys the possibility of faster (i.e., superlinear) convergence.

We can try to construct an LU-decomposition, without pivoting, of $T_{i,i}$. If this decomposition exists, then, similar to CG, it can be updated from iteration to iteration and this leads to a recursive update of the solution vector, which avoids saving all intermediate r and w vectors. This variant of Bi-Lanczos is usually called Bi-Conjugate Gradients, or for short Bi-CG [76]. In Bi-CG, the d_i are chosen such that $v_i = r_{i-1}$, similarly to CG.

Of course one can in general not be certain that an LU decomposition (without pivoting) of the tridiagonal matrix $T_{i,i}$ exists, and this may lead also to breakdown (a breakdown of the *second kind*), of the Bi-CG algorithm. Note that this breakdown can be avoided in the Bi-Lanczos formulation of the iterative solution scheme, e.g., by making an LU-decomposition with 2 by 2 block diagonal elements [15]. It is also avoided in the QMR approach (see Section 6.3).

Note that for symmetric matrices Bi-Lanczos generates the same solution as Lanczos, provided that $w_1 = r_0$, and under the same condition Bi-CG delivers the same iterands as CG for positive definite matrices. However, the Bi-orthogonal variants do so at the cost of two matrix vector operations per iteration step.
For a computational scheme for Bi-CG, without provisions for breakdown, see [18].

The scheme in Figure 6.1 may be used for a computer implementation of the Bi-CG method. In the scheme the equation $Ax = b$ is solved with a suitable preconditioner K. The scheme has no provisions to prevent or cure break down.

As with conjugate gradients, the coefficients α_j and β_j, $j = 0, \ldots, i-1$ build the matrix T_i, as given in formula (4.10). This matrix is, for BiCG, in general not similar to a symmetric matrix. Its eigenvalues can be viewed as Petrov-Galerkin approximations, with respect to the spaces $\{\widetilde{r}_j\}$ and $\{r_j\}$, of eigenvalues of A. For increasing values of i they tend to converge to eigenvalues of A. The convergence patterns, however, may be much more complicated and irregular as in the symmetric case.

x_0 is an initial guess; $r_0 = b - Ax_0$
Choose $\widetilde{r}_0$ such that $(w_0, \widetilde{r}_0) \neq 0$;
usually one chooses $\widetilde{r}_0 = r_0$ or $\widetilde{r}_0 = w_0$
for $i = 1, 2, ...$
    Solve $Kw_{i-1} = r_{i-1}$
    Solve $K^T \widetilde{w}_{i-1} = \widetilde{r}_{i-1}$
    $\rho_{i-1} = w_{i-1}^T \widetilde{w}_{i-1}$
    if $\rho_{i-1} = 0$ **method fails**
    if $i = 1$
        $p_i = w_{i-1}$
        $\widetilde{p}_i = \widetilde{w}_{i-1}$
    else
        $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
        $p_i = w_{i-1} + \beta_{i-1} p_{i-1}$
        $\widetilde{p}_i = \widetilde{w}_{i-1} + \beta_{i-1} \widetilde{p}_{i-1}$
    endif
    $z_i = Ap_i$
    $\widetilde{z}_i = A^T \widetilde{p}_i$
    $\alpha_i = \rho_{i-1}/\widetilde{p}_i^T z_i)$
    $x_i = x_{i-1} + \alpha_i p_i$
    $r_i = r_{i-1} - \alpha_i z_i$
    $\widetilde{r}_i = \widetilde{r}_{i-1} - \alpha_i \widetilde{z}_i$
    if x_i is accurate enough **then quit**
end

Figure 6.1: Bi-CG algorithm

6.2 Another derivation of Bi-CG

An alternative way to derive Bi-CG comes from considering the following symmetric linear system:

$$\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} \widehat{x} \\ x \end{pmatrix} = \begin{pmatrix} b \\ \widehat{b} \end{pmatrix}, \quad \text{or} \quad B\widetilde{x} = \widetilde{b},$$

for some suitable vector $\widehat{b}$.
If we select $\widehat{b} = 0$ and apply the CG-scheme to this system, then we obtain LSQR again. However, if we select $\widehat{b} \neq 0$ and apply the CG scheme with the preconditioner

$$\begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix},$$

in the way as is shown in section 4.4.1, then we obtain right away the unpreconditioned Bi-CG scheme for the system $Ax = b$. Note that the CG-scheme can be applied since $K^{-1}B$ is symmetric (but not positive definite) with respect to the bilinear form

$$[p, q] \equiv (p, Kq),$$

which is not a proper innerproduct. Hence, this formulation clearly reveals the two principal weaknesses of Bi-CG (i.e., the causes for break-down situations). Note that if we restrict ourselves to vectors

$$p = \begin{pmatrix} p_1 \\ p_1 \end{pmatrix},$$

then $[p, q]$ defines a proper innerproduct. This situation arises for the Krylov subspace that is created for B and $\widetilde{b}$ if $A = A^T$ and $\widehat{b} = b$. If, in addition, A is positive definite then $K^{-1}B$ is positive definite symmetric with respect to the generated Krylov subspace, and we obtain the CG-scheme (as expected).
More generally, the choice

$$K = \begin{pmatrix} 0 & K_1 \\ K_1^T & 0 \end{pmatrix},$$

where K_1 is a suitable preconditioner for A, leads to the preconditioned version of the Bi-CG scheme, as given in section 5.4.

The above presentation of Bi-CG was inspired by a closely related presentation of BI-CG in [110]. The latter paper gives a rather untractable reference for the choice of the system $B\widetilde{x} = \widetilde{b}$ and the preconditioner

$$\begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix}$$

to [111].

6.3 QMR

The QMR method [81] relates to Bi-CG in a similar way as MINRES relates to CG. We start with the recurrence relations for the v_j:

$$AV_i = V_{i+1}T_{i+1,i}.$$

We would like to identify the x_i, with $x_i \in K^i(A; r_0)$, or $x_i = V_i y$, for which

$$\|b - Ax_i\|_2 = \|b - AV_i y\|_2 = \|b - V_{i+1}T_{i+1,i}y||_2$$

is minimal, but the problem is that V_{i+1} is not orthogonal. However, we pretend that the columns of V_{i+1} are orthogonal. Then

$$\|b - Ax_i\|_2 = \|V_{i+1}(\|r_0\|_2 e_1 - T_{i+1,i}y)\|_2 = \|(\|r_0\|_2 e_1 - T_{i+1,i}y)\|_2,$$

and in [81] it is suggested to solve the projected miniminum norm least squares problem $\|(\|r_0\|_2 e_1 - T_{i+1,i}y)\|_2$. The minimum value of this norm is called the quasi residual and will be denoted by $||r_i^Q||_2$.
Since, in general, the columns of V_{i+1} are not orthogonal, the computed $x_i = V_i y$ does not solve the minimum residual problem, and therefore this approach is referred to as a Quasi-minimum residual approach [81]. It can be shown that the norm of the residual r_i^{QMR} of QMR can be be bounded in terms of the quasi residual

$$||r_i^{QMR}||_2 \leq \sqrt{i+1}\,||r_i^Q||_2.$$

The sketched approach leads to the simplest form of the QMR method. A more general form arises if the least squares problem is replaced by a weighted least squares problem [81]. No strategies are yet known for optimal weights.

In [81] the QMR method is carried out on top of a look-ahead variant of the bi-orthogonal Lanczos method, which makes the method more robust. Experiments indicate that although QMR has a much smoother convergence behaviour than Bi-CG, it is not essentially faster than Bi-CG. This is confirmed explicitly by the following relation for the Bi-CG residual r_k^B and the quasi residual r_k^Q (in exact arithmetic):

$$\|r_k^B\|_2 = \frac{\|r_k^Q\|_2}{\sqrt{1 - (\|r_k^Q\|_2/\|r_{k-1}^Q\|_2)^2}},$$

see [45]: Theorem 4.1. This relation, which is similar to the relation for GMRES and FOM, shows that when QMR gives a significant reduction at step k, then Bi-CG and QMR have arrived at residuals of about the same norm (provided, of course, that the same set of starting vectors has been used).

It is tempting to compare QMR with GMRES, but this is difficult. GMRES really minimizes the 2-norm of the residual, but at the cost of increasing the work of keeping all residuals orthogonal and increasing demands for memory space. QMR does not minimize this norm, but often it has a convergence comparable to GMRES, at a cost of twice the amount of matrix vector products per iteration step. However, the generation of the basis vectors in QMR is relatively cheap and the memory requirements are limited and modest. In view of the relation between GMRES and FOM, it will be no surprise that there is a similar relation between QMR and Bi-CG, for details of this see [81]. This relation expresses that at a significant local error reduction of QMR, Bi-CG and QMR have arrived almost at the same residual vector (similar to GMRES and FOM). However, QMR is preferred to Bi-CG in all cases because of its much smoother convergence behaviour, and also because QMR removes one break-down condition (even when implemented without look-ahead). Several variants of QMR, or rather Bi-CG, have been proposed, which increase the effectiveness of this class of methods in certain circumstances.

Zhou and Walker [196] have shown that the Quasi-Minimum Residual approach can be followed for other methods, such as CGS and Bi-CGSTAB, as well. The main idea is that in these methods the approximate solution is updated as

$$x_{i+1} = x_i + \alpha_i p_i,$$

and the corresponding residual is updated as

$$r_{i+1} = r_i - \alpha_i A p_i.$$

This means that $AP_i = W_i R_{i+1}$, with W_i a lower bidiagonal matrix. The x_i are combinations of the p_i, so that we can try to find the combination $P_i y_i$ for which $\|b - AP_i y_i\|_2$ is minimal. If we insert the expression for AP_i, and ignore the fact that the r_i are not orthogonal, then we can minimize the norm of the residual in a quasi-minimum least squares sense, similar to QMR.

Exercise 6.1 *Bi-CG and QMR are suitable for general unsymmetric nonsingular linear systems. Construct testcases with known spectra. How to* *avoid to work with dense matrices? Try to get an impression of systems for which Bi-CG and QMR work well. Is it possible to create linear systems for which Bi-CG nearly breaks down at step i? Hint: compute the Ritz values at step i and consider a shift for the matrix of the system.*

Exercise 6.2 *In Bi-CG and QMR, the iteration coefficients build a tridiagonal matrix T. Bi-CG is based on the i by i part, QMR works with the $i+1$ by i part. The difference is thus only in one element. Could that explain effects in the convergence behaviour of both methods? Have the eigenvalues of $T_{i,i}$ any relation with those of the system matrix? Is there an explanation for such a relation?*

6.4 CGS

For the derivation of the CGS method we exploit the fact that the vectors belong to some Krylov subspace and, in particular, we write these vectors in polynomial form. Our derivation is based on the Bi-CG algorithm. Note that the bi-conjugate gradient residual vector can be written as r_j $(= \rho_j v_j)$ $=$ $P_j(A)r_0$, and, likewise, the so-called shadow residual $\widetilde{r}_j$ $(= \rho_j w_j)$ can be written as $\widetilde{r}_j = P_j(A^T)\widehat{r}_0$. Because of the bi-orthogonality relation we have that

$$\begin{aligned}(r_j, \widetilde{r}_i) &= (P_j(A)r_0, P_i(A^T)\widetilde{r}_0)\\ &= (P_i(A)P_j(A)r_0, \widetilde{r}_0) = 0,\end{aligned}$$

for $i < j$. The iteration parameters for bi-conjugate gradients are computed from innerproducts like the above. Sonneveld [160] observed that we can also construct the vectors $\widehat{r}_j = P_j^2(A)r_0$, using only the latter form of the innerproduct for recovering the bi-conjugate gradients parameters (which implicitly define the polynomial P_j). By doing so, the computation of the vectors $\widetilde{r}_j$ can be avoided and so can the multiplication by the matrix A^T.

We will make this more precise. In Bi-CG (Fig. 6.1) we concentrate on the recursions that are necessary to compute the new residual vector r_i (for simplicity we consider the unpreconditioned algorithm: $K = I$):

$$\begin{aligned} p_i &= r_{i-1} + \beta_{i-1}p_{i-1}\\ r_i &= r_{i-1} - \alpha_i A p_i. \end{aligned}$$

Now we express these vectors explicitly as members of the Krylov subspace, by using the polynomial expressions

$$r_i = P_i(A)r_0 \quad \text{and} \quad p_i = Q_{i-1}(A)r_0. \tag{6.2}$$

In these expressions the index denotes the degree of the polynomials involved.

Exercise 6.3 *Show that the vectors r_i and p_i can be expressed as in (6.2).*

The recursion for r_i leads to the following recursion for the polynomials:

$$P_i(A) = P_{i-1}(A) - \alpha_i A Q_{i-1}(A), \tag{6.3}$$

and from the recursion for p_i we obtain

$$Q_{i-1}(A) = P_{i-1}(A) + \beta_{i-1} Q_{i-2}(A). \tag{6.4}$$

We are interested in the computation of the vector $\widehat{r}_i = P_i(A)^2 r_0$, and therefore we square the expression in (6.3):

$$P_i(A)^2 = P_{i-1}(A)^2 + \alpha_i^2 A^2 Q_{i-1}(A)^2 - 2\alpha_i A P_{i-1}(A) Q_{i-1}(A). \tag{6.5}$$

In order to make this a computable recursion, we need also expressions for $Q_{i-1}(A)^2$ and for $P_{i-1}(A)Q_{i-1}(A)$. Squaring (6.4) leads to

$$Q_{i-1}(A)^2 = P_{i-1}(A)^2 + \beta_{i-1} Q_{i-2}(A)^2 + 2\beta_{i-1} P_{i-1}(A) Q_{i-2}(A). \tag{6.6}$$

The expression for $P_{i-1}(A)Q_{i-1}(A)$ is obtained from multiplying (6.4) with $P_{i-1}(A)$:

$$P_{i-1}(A)Q_{i-1}(A) = P_{i-1}(A)^2 + \beta_{i-1} P_{i-1}(A) Q_{i-2}(A). \tag{6.7}$$

For (6.6) and (6.7) we need $P_{i-1}(A)Q_{i-2}(A)$ and this is obtained from the multiplication of the expression in (6.3) with $Q_{i-2}(A)$:

$$P_{i-1}(A)Q_{i-2}(A) = P_{i-2}(A)Q_{i-2}(A) - \alpha_{i-1} A Q_{i-2}(A)^2. \tag{6.8}$$

In the actual computations, all these polynomial expressions have to operate on r_0. This leads to a set of vector recursions for the vectors

$$\begin{aligned} r_i &\equiv P_i(A)^2 r_0 \\ p_i &\equiv Q_{i-1}(A)^2 r_0 \\ u_i &\equiv P_{i-1}(A) Q_{i-1}(A) \\ q_i &\equiv P_i(A) Q_{i-1}(A). \end{aligned}$$

Exercise 6.4 *Show that the above relations lead to the algorithm given in Fig. 6.2.*

In Bi-CG, the coefficient β_{i-1} is computed from the ratio of two successive ρ's. Note that we have now that

$$\begin{aligned} \rho_{i-1} &= \widetilde{r}_{i-1}^T r_{i-1} \\ &= (P_{i-1}(A^T)\widetilde{r}_0)^T P_{i-1}(A) r_0 \\ &= \widetilde{r}_0^T P_{i-1}(A)^2 r_0 \\ &= \widetilde{r}_0^T r_{i-1}, \end{aligned} \tag{6.9}$$

which shows that the Bi-CG recursion coefficient can be recovered without forming the shadow residuals $\widetilde{r}_j$ as in Bi-CG.

Exercise 6.5 *Show that α_i can also be computed from the above defined vectors, that is, without forming the Bi-CG shadow residuals.*

In the CGS algorithm we have so avoided to work with A^T, which may be an advantage in some situations. However, the main attraction is that we are able to generate residuals for which

$$r_i = P_i(A)^2 r_0.$$

Note that it is easy to find the update formula for x_i.

Exercise 6.6 *Compare the computational costs for one iteration of Bi-CG with the costs for one iteration of CGS.*

In order to see the possible advantage of CGS, we have to realize that the operator $P_i(A)$ transforms the starting residual r_0 into a much smaller residual $r_i = P_i(A)r_0$, if Bi-CG converges well. One might hope that applying the operator once again gives another reduction in the residual. Indeed, the resulting CGS [160] method works in general very well for many unsymmetric linear problems. It converges often much faster than BI-CG (about twice as fast in some cases, because of the squaring effect of the operator $P_i(A)$). The surprising observation is that we obtain this faster convergence for about the same amount of work per iteration as in Bi-CG.

Bi-CG and CGS have the advantage that fewer vectors are stored than in GMRES. These three methods have been compared in many studies (see, e.g., [144, 30, 142, 130]).
CGS, however, usually shows a very irregular convergence behaviour. This behaviour can even lead to cancellation and a "spoiled" solution [177]; see also Section 7. Freund [82] suggested a squared variant of QMR, which was

called TFQMR. His experiments show that TFQMR is not necessarily faster than CGS, but it has certainly a much smoother convergence behavior.

The scheme in Fig. 6.2 represents the CGS process for the solution of $Ax = b$, with a given preconditioner K.

x_0 is an initial guess, $r_0 = b - Ax_0$
$\widetilde{r}$ is an arbitrary vector, such that
$\widetilde{r}^T r_0 \neq 0$,
e.g., $\widetilde{r} = r_0$
for $i = 1, 2, ...$
  $\rho_{i-1} = \widetilde{r}^T r_0$
  if $\rho_{i-1} = 0$ **method fails**
  if $i = 1$
    $u_1 = r_0$
    $p_1 = u_1$
  else
    $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
    $u_i = r_{i-1} + \beta_{i-1} q_{i-1}$
    $p_i = u_i + \beta_{i-1}(q_{i-1} + \beta_{i-1} p_{i-1})$
  endif
  solve $\widehat{p}$ **from** $K\widehat{p} = p_i$
  $\widehat{v} = A\widehat{p}$
  $\alpha_i = \frac{\rho_{i-1}}{\widetilde{r}^T \widehat{v}}$
  $q_i = u_i - \alpha_i \widehat{v}$
  solve $\widehat{u}$ **from** $K\widehat{u} = u_i + q_i$
  $x_i = x_{i-1} + \alpha_i \widehat{u}$
  $r_i = r_{i-1} - \alpha_i A\widehat{u}$
  check convergence; continue if necessary
end

Figure 6.2: CGS algorithm

In exact arithmetic, the α_j and β_j are the same constants as those generated by Bi-CG. Therefore, they can be used to compute the Petrov-Galerkin approximations for eigenvalues of A.

Exercise 6.7 *Construct real unsymmetric systems with known eigenvalues. Test CGS and Bi-CG for problems where all eigenvalues have positive real part (and zero or small imaginary parts). Compare the convergence plots. Do the same for spectra with larger imaginary parts and for indefinite problems.*

Exercise 6.8 *Compare CGS and GMRES for some test problems. CGS may be faster than GMRES in terms of iterations, but can it be faster in terms of matrix vector products? Can CGS be faster in terms of CPU-time? In what sort of situations?*

Chapter 7

How serious is irregular convergence?

Bi-CG and methods derived from Bi-CG can display rather irregular convergence behaviour. By irregular convergence we refer to the situation where successive residual vectors in the iterative process differ in orders of magnitude in norm, and some of these residuals may be even much bigger in norm than the starting residual. In particular the CGS method suffers from this phenomenon. We will show why this is a point of concern, even if eventually the (updated) residual satisfies a given tolerance.

In the Bi-CG algorithms, as well as in CG, we see in the algorithm typically a statement for the update of x_i, like

$$x_{i+1} = x_i + w_i \tag{7.1}$$

and a statement for the update of r_i, of the form

$$r_{i+1} = r_i - Aw_i. \tag{7.2}$$

We see that, in exact arithmetic, the relation $r_{i+1} = b - Ax_{i+1}$ holds, just as expected. A further inspection of these algorithms reveals that x_i is not used at other places in the basic algorithm, whereas the r_i is also used for the computation of the search direction and for iteration parameters. The important consequence of this is that rounding errors introduced by the actual evaluation of r_{i+1} through equation (7.2) will influence the further iteration process, but rounding errors in the evaluation of x_{i+1} by (7.1) will have no effect on the iteration. This would not be much of a problem if the rounding error

$$\delta r_{i+1} \equiv fl(r_i - Aw_i) - (r_i - Aw_i)$$

would match the rounding error

$$\delta x_{i+1} \equiv fl(x_i + w_i) - (x_i + w_i),$$

in the sense that $\delta r_{i+1} = -A\delta x_{i+1}$, since that would keep the desired relation $r_{i+1} = b - Ax_{i+1}$ intact. However, it will be obvious that this is idle hope, and the question remains how serious a possible deviation between r_j and $b - Ax_j$ can be.

Of course, we make rounding errors in (7.1) and (7.2) through the vector addition, but usually these errors will be small in comparison with the rounding errors introduced in the multiplication of w_i with A. Therefore, we will here consider only the effect of these errors. In this case, we can write the computed r_{i+1} as

$$r_{i+1} = r_j - Aw_i - \Delta_{Ai} w_i, \tag{7.3}$$

where Δ_A is an $n \times n$ matrix for which $|\Delta_{Ai}| \preceq n_A \,\overline{\xi}\, |A|$: n_A is the maximum number of non-zero matrix entries per row of A, $|B| \equiv (|b_{ij}|)$ if $B = (b_{ij})$, $\overline{\xi}$ is the relative machine precision, the inequality $\preceq$ refers to element-wise $\leq$.

It then simply follows that

$$r_k - (b - Ax_k) = \sum_{j=1}^{k} \Delta_{Aj} w_j = \sum_{j=1}^{k} \Delta_{Aj}(e_{j-1} - e_j), \tag{7.4}$$

e_j is the approximation error in the jth approximation: $e_j \equiv x - x_j$. Hence,

$$\begin{aligned} \big|\|r_k\| - \|b - Ax_k\|\big| &\leq 2k\, n_A \,\overline{\xi}\, \||A|\| \max_j \|e_j\| \\ &\leq 2k\, n_A \,\overline{\xi}\, \||A|\| \left\|A^{-1}\right\| \max_j \|r_j\|. \end{aligned} \tag{7.5}$$

Except for the factor k, the first upper–bound appears to be rather sharp. We see that an approximation with a large approximation error (and hence a large residual) may lead to inaccurate results in the remaining iteration process. Such large local approximation errors are typical for CGS, and Van der Vorst[177] describes an example of the resulting numerical inaccuracy. If there are a number of approximations with comparable large approximation errors, then their multiplicity may replace the factor k, otherwise it will be only the largest approximation error that makes up virtually all of the bound for the deviation.

For more details we refer to Sleijpen et al [156, 159].

Exercise 7.1 *Show that similar update expressions can also be formulated for Bi-CGSTAB. Is the difference essential for the above discussion on the influence of rounding errors? Derive an upperbound for the deviation between* $\|r_k\|_2$ *and* $\|b - Ax_k\|_2$ *for Bi-CGSTAB, again under the assumption that only the multiplication with* A *leads to rounding errors.*

7.1 Reliable updating

It is of course important to maintain a reasonable correspondence between r_k and $b-Ax_k$, and the easiest way to do this would be to replace the vector r_k by $b-Ax_k$. However, the vectors r_k steer the entire iterative process and their relation defines the projected matrix $T_{i,i}$. If we replace these vectors then we ignore the rounding errors to these vectors and it will be clear that the iteration process cannot compensate for these rounding errors. These rounding errors may be significant at iteration steps where the update to r_j is relatively large and the above sketched naive replacement strategy may then not be expected to work well. Indeed, if we replace r_{i+1} in CGS by $b - Ax_i$, instead of updating it from r_i, then we observe stagnation in the convergence in many important situations. This means that we have to be more careful.

Neumaier (see [156] and references therein) suggested to replace r_j by $b-Ax_j$ in CGS only at places where $\|r_j\|_2$ is smaller than the smallest norm of the residual in the previous iteration history and to carry out a groupwise update for the iterates in between. Schematically, the groupwise update and residual replacement strategy of Neumaier can be described as in Fig 7.1.

This scheme was further analysed and refined, in particular with a flying restart strategy, in [156]. Note that the errors in the evaluation of w_j itself are not so important: it is the different treatment of w_j in the updating of x_j and of r_j that causes the two vectors to loose their wanted relation. In this respect we may consider the vectors w_j as exact quantities.

At a replacement step we perturb the recurrence relation for the basis vectors of the Krylov subspace and we want these errors to be as small as possible. The updates w_j usually vary widely in norm in various stages of the iteration process, for instance in an early phase these norms may be larger than $\|r_0\|_2$, whereas they are small in the final phase of the iteration process. Specially in a phase between two successive smallest values of $\|r_j\|_2$, the norms of the updates may be a good deal larger than in the next interval between two smallest residual norms. Grouping the updates in groups of updates avoids that rounding errors within one group spoil the

Groupwise Solution Update:
$z = x_0$, $\widehat{x} = 0$, $r_{min} = \|r_0\|_2$
...
for $j = 0, 2, \cdots$ until convergence
    ...
    $\widehat{x} = \widehat{x} + w_j$ (instead of update of x_i)
    if $\|r_j\|_2 < r_{min}$ (i.e. group update)
        $z = z + \widehat{x}$
        $\widehat{x} = 0$
        $r_j = b - Az$
        $r_{min} = \|r_j\|_2$
    end if
end

Figure 7.1: Neumaiers' update strategy

result for another group. More specifically, if we denote the sum of w_j's for the groups by S_i, and the total sum of updates by S, then groupwise updating leads to errors of the magnitude of $\xi|S_i|$, which can be much smaller than $\xi|S|$.

Now we have to determine how much we can perturb the recurrence relations for the Lanczos vectors r_j. This has been studied in much detail in [167]. It has been observed by many authors that the driving recurrences $r_j = r_{j-1} - \alpha_{j-1}Aq_{j-1}$ and $q_j = r_j + \beta_{j-1}q_{j-1}$ are locally satisfied almost to machine precision and this is one of the main properties behind the convergence of the computed residuals [92, 167, 156]. Tong and Ye [167] observed that these convergence is maintained even when we perturb the recurrence relations with perturbations that are significantly greater than machine precision, say of the order of the square root of the machine precision ξ, in a relative sense.

The idea, presented in [181], is to compute an upper bound for the deviation in r_j, with respect to $b - Ax_j$, in finite precision, and to replace r_j by $b - Ax_j$ as soon as this upper bound reaches the relative level of $\sqrt{\xi}$. This upper bound is denoted by d_j and it is computed from the recurrence

$$d_j = d_{j-1} + \xi N\|A\|\|\widehat{x}_j\| + \xi\|r_j\|,$$

with N the maximal number of nonzero entries per row of A.

The replacement strategy for reliable updating is then implemented schematically as in Fig. 7.2.

Reliable Updating Strategy:
Input: an initial approximation $x = x_0$
  a residual replacement threshold $\epsilon = \sqrt{xi}$, an estimate of $N\|A\|$
$r_0 = b - Ax_0$ $\widehat{x}_0 = 0$, $d_{init} = d_0 = \xi(\|r_0\| + N\|A\|\|x_0\|)$,
for $j = 1, 2, \cdots$, until convergence
  Generate a correction vector q_j by the Iterative Method
  $\widehat{x}_j = \widehat{x}_{j-1} + q_j$
  $r_j = r_{j-1} - Aq_j$
  $d_j = d_{j-1} + \xi N\|A\|\,\|\widehat{x}_j\| + \xi\|r_j\|$
  if $d_{j-1} \leq \epsilon\|r_{j-1}\|$, $d_j > \epsilon\|r_j\|$ and $d_j > 1.1 d_{init}$
    $z = z + \widehat{x}_j$
    $\widehat{x}_j = 0$
    $r_j = b - Az$
    $d_{init} = d_j = \xi(\|r_j\| + N\|A\|\|z\|)$
  end if
end for
$z = z + \widehat{x}_j$

Figure 7.2: Ye-van der Vorst update strategy

Remark: For this reliable implementation, we need to put a value for N (the maximal number of nonzero entries per row of A) and $\|A\|$. The number of nonzero entries may, in applications, vary from row to row, and selecting the maximum number may not be very realistic. In our experience with sparse matrices, the simple choice $N = 1$ still leads to a practical estimate d_n for $\|\delta_n\|$. For $\|A\|$, we suggest to take simply $\|A\|_\infty$.
In any case, we note that precise values are not essential, because the replacement threshold ϵ can be adjusted. We also need to choose this ϵ. Extensive numerical testing (see [181]) suggests that $\epsilon \sim \sqrt{\xi}$ is a practical criterion. However, there are examples where this choice leads to stagnating residuals at some unacceptable level. In such cases, choosing a smaller ϵ will regain the convergence to $O(\xi)$.

The presented implementation requires one extra matrix-vector multiplication when an replacement is carried out. Since only a few steps with replacement are required, this extra cost is marginal relative to the other costs. However, some savings can be made by selecting a slightly smaller ϵ and carrying out residual replacement at the step next to the one for which the residual replacement criterion is satisfied (cf [156]). It also requires one extra vector storage for the groupwise solution update (for z) and computation of a vector norm $\|\widehat{x}_n\|$ for the update of d_n ($\|r_n\|$ is usually computed in the algorithm for stopping criteria).

7.2 Rounding errors and discretization errors

When a linear system $Ax = b$ is solved in rounded finite precison floating point arithmetic, then the obtained solution $\widehat{x}$ is most likely contaminated by rounding errors. In the absence of precise information about these errors, we have to work with bounds for their effects. A well-known rigorous bound for the relative error in the solution of the perturbed linear system

$$(A + \delta A)y = b + \delta b$$

is given by [90, Ch.2.7.3]:

$$\frac{\|y - x\|}{\|x\|} \leq \frac{2\epsilon}{1 - r}\kappa(A), \tag{7.6}$$

under the conditions that

$$\|\delta A\| \leq \epsilon\|A\|, \quad \|\delta b\| \leq \epsilon\|b\|, \quad \epsilon\kappa(A) = r < 1,$$

and $\kappa(A)$ denotes the condition number of A:

$$\kappa(A) \equiv \|A^{-1}\| \, \|A\|.$$

The upperbound is quite realistic in the sense that there exist perturbations that lead to errors that are quite close to the upperbound.

However, this does not imply that we may expect the multiplying effect of the condition number for perturbations that have a specific structure, like for instance discretisation errors. In particular, if $Ax = b$ comes from a standard finite difference or finite element discretisation of an elliptic partial differential equation, then a common situation is that the exact solution $\bar{x}$ of the PDE, restricted to the gridpoints, satisfies

$$A\bar{x} = b + \tau,$$

with $\tau_i = \mathcal{O}(h^2)$.
In such situations, the solution x of $Ax = b$ typically satisfies

$$\|x - \bar{x}\| = \mathcal{O}(h^2).$$

See, for instance, [103, Ch.4.5] or [143, Ch.6.2.1] for more detailed information on this.

This means that a structured error in b may have a very well understood effect on the solution, in contrast to unstructured errors. When the discretised system $Ax = b$ is represented by machine numbers, then we introduce an additional relative error in the solution that is bounded by (7.6), with ϵ replaced by the relative machine precision ξ.

A major source of errors in the actually computed solution is in the algorithm by which the solution is generated. Gaussian elimination may lead to large additional errors, even with partial pivoting [90, Ch.3.3.2]. The errors introduced by the elimination process can be largely removed by iterative refinement, under the condition that the residual $b - A\widehat{x}$ can be computed with some significant digits [90, Ch.3.5.3]. It is tempting to believe that this is automatically the case in iterative solution methods, but this is only the case if the residual is computed explicitly, at least from time to time, as in the reliable updating technique. In particular, for an explicitly computed residual $r_i = b - Ax_i$, we have with (7.6) that

$$\frac{\|x_i - x\|}{\|x\|} \leq \frac{2}{1-r}\kappa(A)\frac{\|r_i\|}{\|b\|},$$

under the condition that $\kappa(A)\frac{\|r_i\|}{\|b\|} = r < 1$.

7.3 Effects of rounding errors to Krylov processes

In this section we will study the possible effects of inexact computations on Krylov subspace methods. We have seen how the effects of large updates (or better large residual vector corrections) can be reduced by appropriate update techniques. Now we will focus on other parts of the algorithms.

Essentially, the Krylov subspace methods are composed with the following ingredients

(a) The construction of a (bi-)orthogonal basis,

(b) The solution of a small reduced linear system,

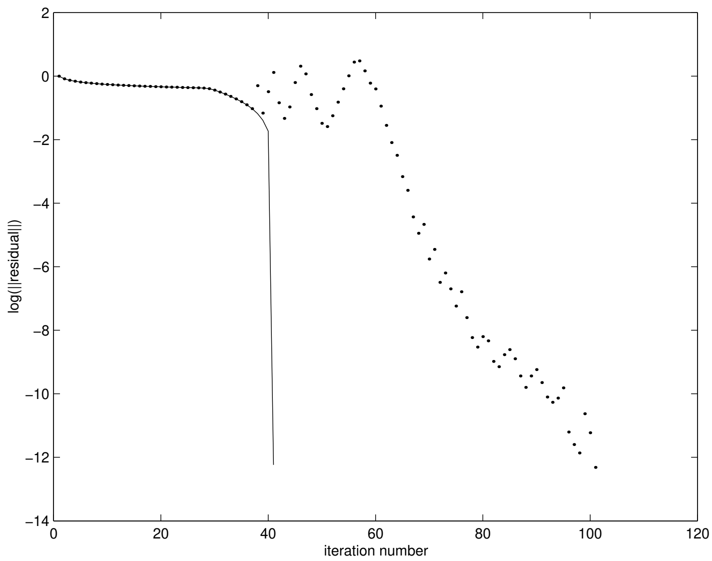

Figure 7.3: CG (...) and FOM (—)

(c) The construction of an approximate solution as a combination of the basis vectors.

Each of these three elements is a source for rounding errors and the fact that these elements are usually mixed in efficient implementations makes the analysis complicated. As a result, rigid upper bounds for actual methods are usually pessimistic and do not describe the actually observed behaviour well. For an example of state of the art work with respect to GMRES, see [60].

We will follow a slightly different approach in order to get more insight in the main sources of rounding errors. We will restrict ourselves to symmetric matrices. In that case, a basis for the Krylov subspace is generated by the Lanczos method. This remarkable three-term recurrence relation does many things simultaneously: it provides a reduced system that can be used to generate an approximate solution and it contains useful spectral information on the matrix A. Even more remarkable is its behaviour with respect to rounding errors. In finite arithmetic the iteration coefficients may differ more than 100% from those in exact arithmetic, after a number of iteration steps. Nevertheless, the reduced system may still lead to accurate approximate solutions and to accurate spectral information.

In the early 1950s, the Lanczos method was regarded as a method for the reduction of a square matrix A to tridiagonal form, and after the full n steps one would expect a similarity transformation of A to tridiagonal form, if there is no early termination encountered. In the latter case one would have identified an invariant subspace, which is just as useful information. It was soon recognized that rounding errors could change the process dramatically. Engeli *et al* [71] published an example for which the desired spectral information was obtained only after many more than n iterations. A decade later, Ginsburg [85] made an algorithmic description of the Conjugate Gradients for the famous Handbook of Automatic Computation [194]. As an illustrative example, Ginsburg also used a discretized bi-harmonic equation for a one-dimensional beam problem, which led to the 40 by 40 matrix:

$$A = \begin{bmatrix} 5 & -4 & 1 & & & & & & & & \\ -4 & 6 & -4 & 1 & & & & & & & \\ 1 & -4 & 6 & -4 & 1 & & & & \emptyset & & \\ & 1 & -4 & 6 & -4 & 1 & & & & & \\ & & \cdot & \cdot & \cdot & \cdot & \cdot & & & & \\ & & & \cdot & \cdot & \cdot & \cdot & \cdot & & & \\ & & & & \cdot & \cdot & \cdot & \cdot & \cdot & & \\ & & & & & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & & & & & 1 & -4 & 6 & -4 & 1 & \\ & \emptyset & & & & & 1 & -4 & 6 & -4 & 1 \\ & & & & & & & 1 & -4 & 6 & -4 \\ & & & & & & & & 1 & -4 & 5 \end{bmatrix}.$$

Then CG was used to solve the linear system $Ax = b$, where b is the first unit vector (of length 40). The convergence history is shown in Fig.7.3. Clearly, the CG method requires about 90 iterations to obtain a fairly accurate solution, and the approximate solution after 40 iterations is quite far from what it should have been in exact computation.

In the same figure, we have also plotted the convergence history for FOM, where we have solved the reduced system with Givens transformations in order to have optimal numerical stability. In theory, FOM and CG should produce the same iterates and we see that this is more or less the case until the 35th iteration step. The main difference between FOM and CG is that in FOM we orthogonalize the new basis vector for the Krylov subspace against all previous basis vectors. Apparently, the loss of orthogonality, among the basis vectors generated with the 3-term recurrence relations, leads to a significant delay of the convergence. This was not understood in the early 1970s so that it is no surprise that CG was not very popular around that time. The CG subroutine, published in [194], was one of the very few that did not make it to the influential LINPACK and EISPACK software collections. It was only in the late 1990s that the conjugate gradients method entered Matlab.

7.3.1 The Lanczos recurrence in finite precision

In our attempts to understand the main effects of rounding errors we follow initially Greenbaum's analysis [93, Chapter4]. This analysis starts with the famous results of Paige [134], published in 1976. Paige gave a rigorous analysis of the behavior of the Lanczos algorithm in finite precision. Before we are in the position to quote an important theorem from [134], we have to introduce some notations.

Let ξ denote the floating point relative machine precision. With m we denote the maximum number of nonzero elements per row of the n by n matrix A.
We repeat the algebraic formulation, in exact arithmetic, of the Lanczos reduction

$$AV_k = V_{k+1}T_{k+1,k}, \tag{7.7}$$

with $T_{k+1,k}$ a $k+1$ by k tridiagonal matrix and V_j denotes the n by j matrix with the orthonormal Lanczos vectors v_1, $v_2, \ldots, v_j$ as its columns. This relation can equivalently be expressed as

$$AV_k = V_kT_k + t_{k+1,k}v_{k+1}^Te_k^T,$$

where e_k denotes the kth unit vector of length k.

We know that the Lanczos vectors are, in exact arithmetic, generated with Arnoldi's algorithm, given in Fig. 2.1. The k by k leading part of the reduced matrix $H_{k+1,k}$ is symmetric, expressed by the fact that we have replace H by T in (7.7). When working in finite precision, we have to be more careful in how we ensure that T is tridiagonal. First, we avoid, of course, the computation of the elements outside the tridiagonal part. Second, we ensure that the matrix T is symmetric, by avoiding to compute the lower diagonal elements of T again. In fact, Paige [134] proved his results for the implementation shown in Fig. 7.4 of the algorithm in Fig. 2.1.

Exercise 7.2 *Show that the algorithm in Fig. 7.4 leads in exact arithmetic*

Let v_1 be given with $v_1^T v_1 = 1$
$u = Av_1$
for $j = 1, 2, \cdots, k$
  $\alpha_j = v_j^T u$
  $w = u - \alpha v_j$
  $\beta_j = \|w\|_2$, **if** $\beta_j = 0$ STOP
  $v_{j+1} = w/\beta_j$
  $u = Av_{j+1} - \beta_j v_j$
end for

Figure 7.4: The Paige-style Lanczos algorithm

to the same vectors v_j as the algorithm in Fig. 2.1. How do the α_j and β_j define the matrix $T_{k+1,k}$?

This sets the stage for the following results.

Theorem 1 Paige [134]. *Let A be an n by n real symmetric matrix with at most m non-zero elements in any row. If the Lanczos algorithm described in Fig. 7.4 is executed in floating point arithmetic with relative machine precision ξ, then α_j, β_j, v_j will be computed such that*

$$\begin{aligned} AV_k &= V_kT_k + \beta_k v_{k+1} e_k^T + \delta V_k && (7.8)\\ with V_k &\equiv [v_1, \ldots, v_k] \\ \delta_k &\equiv [\delta v_1, \ldots, \delta v_k] \\ |v_j^T v_j - 1| &\leq (n+4)\xi && (7.9)\\ \|\delta v_j\|_2 &\leq \|A\|_2 (7 + m\| \, |A| \, \|_2 / \|A\|_2)\xi. && (7.10) \end{aligned}$$

Note that from now on, in this chapter, V_k and T_k will denote the computed quantities. We will now follow Greenbaum's way of reasoning and study what will happen if we use the computed V_k and T_K in a straight forward manner. We will consider two different cases:

Case A *The Ritz-Galerkin approach:* In this case the approximated solution is defined as

$$x_k^R = V_k y_k^R \quad \text{with} \quad y_k^R = \|r_0\|_2 T_{k,k}^{-1} e_1, \qquad (7.11)$$

and we will use this definition for the computation. In fact, this definition is, in exact arithmetic, equivalent to the CG method, but then the x_k is computed from different formulas.

Case B *The Minimum Residual approach:* Here the approximated solution is defined by

$$x_k^M = V_k y_k^M, \tag{7.12}$$

where y_k^M is defined by

$$\min_y \| \, \|r_0\|_2 e_1 - T_{k+1,k} y\|_2 \, (= \|b - Ax_k^M\|_2).$$

In exact arithmetic, these definitions lead to an x_k^M that is the same as the one computed by MINRES, but, again, in MINRES other formulas are used.

We will now assume that the defining formulas (7.11) and (7.12) hold exactly for the computed V_k and $T_{k+1,k}$.

Analysis for Case A The residual for x_k^R is:

$$\begin{aligned} r_k^R &= r_0 - AV_k y_k^R \\ &= r_0 - (V_k T_{k,k} + \beta_k v_{k+1} e_k^T + \delta V_k) y_k^R \\ &= -\beta_k v_{k+1} e_k^T y_k^R - \delta V_k y_k^R & (7.13) \\ &= -\beta_k \|r_0\|_2 v_{k+1} e_k^T T_{k,k}^{-1} e_1 - \delta V_k y_k^R. & (7.14) \end{aligned}$$

Note that we have treated y_k^R differently in the last expression. The reason is that in the first term at the right-hand side of (7.14) it is only the last term of the solution that matters and by using $T_{k,k}$ this can be nicely exploited. We will deviate slightly from Greenbaum's analysis by a different interpretation of the formulas.

From (7.14) it follows that the norm of $\|r_k^R\|_2$ can be bounded as

$$\|r_k^R\|_2 \le |\beta_k| \, \|r_0\|_2 \|v_{k+1}\|_2 \, |e_k^T T_{k,k}^{-1} e_1| + \|\delta V_k\|_2 \, \|y_k^R\|_2. \tag{7.15}$$

Exercise 7.3 *Consider the Lanczos process for the system*

$$T_{k+1,k+1} y = e_1. \tag{7.16}$$

Show that j steps of the process, starting with $v_1 = e_1$, generates the Lanczos basis $v_i = e_i$, for $i = 1, \ldots, j \le k$ and that the reduced matrix $\widetilde{T}$ is given by

$$\widetilde{T}_{j+1,j} = Tj + 1, j.$$

Exercise 7.4 *Show, with the result of exercise 7.3, that $j \leq k$ steps with CG for the system (7.16 with starting vector $y_0 = 0$ lead to an approximate solution for which the norm of the residual satisfies:*

$$\|r_j\|_2 = \beta_j \, |e_j^T T_{j,j}^{-1} e_1|. \tag{7.17}$$

This last exercise shows that the first term in the right hand side of (7.15) can be interpreted as $\|r_0\|_2$ times the norm of a residual of a CG process. Remains to link this CG process with a CG process for A. In exact arithmetic, the Lanczos algorithm leads to a transformation of A to tridiagonal form (or to a partial reduction with respect to an invariant subspace):

$$AV_n = V_n T_{n,n}.$$

Paige [135] has shown that in floating point arithmetic a tridiagonal matrix is generated that has eigenvalues θ_j that are in the interval

$$[\min \lambda A - \tau_1, \max \lambda(A) + \tau_2], \tag{7.18}$$

where τ_1, tau_2 are mild functions of j times the machine precision ξ. We can now view the matrix $T_{k+1,k}$ as a section of a much larger matrix T that satisfies Paige's eigenvalue bounds. This means that the residuals $\beta_k |e_k^T T_{k,k}^{-1} e_1|$ (see equation (7.17)) can be bounded as in CG (4.23, which leads to an upperbound that converges with a factor

$$\frac{\sqrt{\widetilde{\kappa}} - 1}{\sqrt{\widetilde{\kappa}} + 1},$$

where $\widetilde{\kappa}$ is the condition number based on the spectrum in (7.18).

This means that the first term in the right-hand side of (7.15) can be bounded by a function that converges to zero almost the same as the usual upper bound for the CG residuals for A (note that $\|v_{k+1}\|_2$ can be bounded essentially by 1 apart from a very small correction in the order of ξ. Eventually, the second term will dominate and this second term can be bounded as (using (7.10) and (7.11))

$$\| \, \|\delta V_k\|_2 \, \|y_k^R\|_2 \|_2 \leq \sqrt{k} \|A\|_2 (7 + m\| \, |A| \, \|_2 / \|A\|_2) \xi \, \|r_0\|_2 \|T_{k,k}^{-1}\|_2. \tag{7.19}$$

According to Paige's results, $\|T_{k,k}^{-1}\|_2$ is apart from a very small correction equal to $\|A^{-1}\|_2$. This leads to an upper bound that eventually behaves like a function that is proportional to $\sqrt{k}\kappa\xi$, which is rather satisfactory. The final conclusion is that the errors in the Lanczos process do not seriously affect the converging properties of the Ritz-Galerkin approach.

Analysis for Case B For the minimum residual case, we can follow exactly the same arguments as for case A. Now we have for the residual for x_k^M (again following Greenbaum's analysis [93, Chpater 4.3]:

$$\begin{aligned} r_k^M &= r_0 - AV_k y_k^M \\ &= r_0 - (V_{k+1}T_{k+1,k} + \delta V_k) y_k^M \\ &= V_{k+1}(\|r_0\|_2 - T_{k+1,k} y_k^M - \delta V_k y_k^M. \end{aligned} \tag{7.20}$$

From (7.20) it follows that the norm of $\|r_k^M\|_2$ can be bounded as

$$\|r_k^M\|_2 \leq \|V_{k+1}\|_2 \, \| \, \|r_0\|_2 e_1 - T_{k+1,k} y_k^M \|_2 + \|\delta V_k\|_2 \, \|y_k^M\|_2. \tag{7.21}$$

The first term at the right-hand side goes to zero on the account of the relation between Ritz-Galerkin (in exact arithmetic: FOM) and Minimum Residual (in exact arithmetic: MINRES) residuals, see (5.2). This implies that eventually the second term in the error bound (7.21) dominates and this is the same term as for the Ritz-Galerkin approach.

From this we conclude that the loss of accuracy in the Lanczos process has no serious effects on the overall convergence of the Ritz-Galerkin and the Minimum Residual iterative approaches. By no serious effects we mean that the global upper bound, based on the condition number of A, is almost the same in exact and in finite precision floating point arithmetic. One may see effects though. We have seen that, depending on the spectral distribution of A, superlinear convergence may occur for, for instance CG. This increased speed of convergence happens when extreme eigenvalues have been approximated sufficiently well in the underlying Lanczos process. It is well-known that loss of orthogonality in the Lanczos process goes hand in hand with the occurrence of (almost) multiple eigenvalues of $T_{k,k}$ (see, for instance, [139, Chapter 13.6], [134]). These so-called multiplets have nothing to do with possibly multiple eigenvalues of A. Because of rounding errors, the eigenvalues of A, to which eigenvalues of $T_{k,k}$ have converged, have their effect again on future iterations and they may reduce (part of) the effects of the so-called superlinear convergence behaviour. This is visible in a delay of the convergence with respect to the convergence that one would have observed in exact arithmetic (or that one might have expected from gaps in the spectrum of A).

7.3.2 Effects of rounding errors on implementations

In the previous section we have discussed the effects of finite precision computations in the Lanczos process and we have seen that these effects are limited. In actual implementations there may be bigger effects because of a specific implementation. This has been analysed to some extent in [158]. We will follow this discussion for two possible implementations of the minimum residual approach for symmetric linear systems.

We start again from (7.12). We see that a small overdetermined linear system

$$T_{k+1,k}y_k^M = \|r_0\|_2 e_1$$

has to be solved, and a good and stable way of doing this is via a reduction of $T_{k+1,k}$ with Givens rotations to upper triangular form

$$T_{k+1,k} = Q_{k+1,k}R_k,$$

in which R_k is a k by k upper triangular band matrix with bandwidth 3 and $Q_{k+1,k}$ is a $k+1$ by k matrix with orthonormal columns (that is the product of the k Givens rotations).

This leads to the approximated solution

$$x_k^M = \|r_0\|_2\, V_k R_k^{-1} Q_{k+1,k}^T e_1. \tag{7.22}$$

With the given Lanczos basis, this approximated solution can be computed in a straight forward manner by first evaluating the vector

$$z_k = Q_{k+1,k}^T\, \|r_0\|_2 e_1 \tag{7.23}$$

and then evaluating

$$\widetilde{z}_k = R_k^{-1} z_k \quad \text{and} \quad x_k = V_k \widetilde{z}_k. \tag{7.24}$$

We will denote this way of computing by rewriting expression (7.22) with parentheses that indicate the order of computation (in a way that is different from other orders of computation to come):

$$x_k^M = V_k\, (R_k^{-1} Q_{k+1,k}^T\, \|r_0\|_2 e_1). \tag{7.25}$$

In fact, this is the way of computation that is followed in GMRES for unsymmetric linear systems and therefore we will denote this implementation of the minimum residual approach on the basis of the three-term Lanczos relation as $MINRES_{GMRES}$.

The disadvantage of $MINRES_{GMRES}$ would be that we have to store all Lanczos vectors and that is avoided in actual implementations. Indeed, the computations in the formula (7.22) can be grouped alternatively as

$$x_k^M = (V_k R_k^{-1}) z_k \equiv W_k z_k, \tag{7.26}$$

with z_k as in (7.23). Because of the banded structure of R_k, it is easy to see that the last column of W_k can be computed from the last two columns of W_{k-1} and v_k. This interpretation makes it possible to generate x_k^M with a short recurrence, since z_k itself follows from the kth Givens rotation applied to $(z_{k-1}^T, 0)^T$. This approach is the basis for actual MINRES implementations and therefore we will refer to this implementation simply as $MINRES$.

Note that z_k is evaluated in the same way for $MINRES_{GMRES}$ and $MINRES$. In fact, the essential difference between the evaluations in (7.24) and (7.26) is in the action of R_k^{-1}. In $MINRES_{GMRES}$ the operator R_k^{-1} acts on z_k, whereas in $MINRES$ it acts on V_k. The analysis in [158] concentrates on this difference.

We start with $MINRES_{GMRES}$. With the given computed R_k and z_k, we compute in floating point finite precision arithmetic the vector $\widetilde{z}_k$ and we denote the actually computed result as $\widehat{z}_k$. This results satisfies (cf. [90, p.89]):

$$(R_k + \Delta_R)\widehat{z}_k, \quad \text{with} \quad |\Delta_R| \leq 3\xi|R_k| + \mathcal{O}(\xi^2),$$

where ξ denotes the machine precision.
Because $\widehat{z}_k = (I + R_k^{-1}\Delta_R)^{-1}R_k^{-1}z_k$, we have for the difference between the computed and the exact vector

$$\Delta_1 \equiv \widehat{z}_k - \widetilde{z}_k = -R_k^{-1}\Delta_R R_k^{-1}z_k.$$

Then we have to multiply V_k with $\widehat{z}_k$ in order to obtain the computed $\widehat{x}_k$, and this introduces also an error (see [107, p.78]):

$$\widehat{x}_k = V_k\widehat{z}_k + \Delta_2, \quad \text{with} \quad |\Delta_2| \leq k\xi|V_k||\widetilde{z}_k| + \mathcal{O}(\xi^2).$$

These two errors Δ_1 and Δ_2 lead to a contribution Δx_k in the computed solution x_k:

$$\Delta x_k = V_k\Delta_1 + \Delta_2.$$

When we evaluate $b - A\widehat{x}_k$ then this contribution leads to an extra deviation Δr_k^G (in addition to other error sources) with respect to the exact $b - Ax_k$, and after some formula manipulation (see [158, p.732]) we obtain

$$\frac{\|\Delta r_k^G\|_2}{\|b\|_2} \leq (3\sqrt{3}\|V_{k+1}\|_2 + k\sqrt{k})\xi\kappa_2(A), \tag{7.27}$$

where $\kappa_2(A)$ denotes the condition number $\|A^{-1}\|_2\|A\|_2$.

In actual situations $\|V_{k+1}\|_2$ is a modest number, much smaller than $\sqrt{k+1}$, because there is at least local orthogonality in the columns of V_{k+1}.

The dominating factor is the condition number, but note that we may expect relative errors in the order of $\xi\kappa_2(A)$ already because of rounding errors in the representation of A and b. This leads to the conclusion that the computation of $\widehat{x}_k$ with R_k and the evaluation of $V_k\widehat{z}_k$ is rather stable in $MINRES_{GMRES}$.

Now we turn our attention to the implementation of $MINRES$. According to (7.26), we have to evaluate W_k. The jth row $w_{j,:}$ of W_k satisfies

$$w_j R_k = v_{j,:},$$

which in finite precision leads to a row $\widehat{w}_{j,:}$ that satisfies

$$\widehat{w}_{j,:}(R_k + \Delta_{R_j}) = v_{j,:} \quad \text{with} \quad |\Delta_{R_j}| \leq 3\xi|R_k| + \mathcal{O}(\xi^2). \tag{7.28}$$

After combining the relations for the k rows of the computed $\widehat{W}_k$, we have that $\widehat{W}_k$ satisfies

$$\widehat{W}_k = (V_k + \Delta_W)R_k^{-1} \quad \text{with} \quad |\Delta_W| \leq 3\xi|\widehat{W}_k|\,|R_k| + \mathcal{O}(\xi^2). \tag{7.29}$$

In this equation, we may replace $\widehat{W}_k$ in the expression for $|\Delta_W|$ by $V_k R_k^{-1}$, because this leads to additional errors of $\mathcal{O}(\xi^2)$ only. The computation of x_k with the computed $\widehat{W}_k$ leads to additional errors that can be expressed as

$$\widehat{x}_k = \widehat{W}_k z_k + \Delta_3 \quad \text{with} \quad |\Delta_3| \leq k\xi|W_k|\,|z_k| + \mathcal{O}(\xi^2).$$

Proceeding in a similar way as for $MINRES_{GMRES}$, we obtain errors Δr_k^M in the computed $b - A\widehat{x}_k$ that can be attributed to the computation of $\widehat{W})k$ and the assembly of $\widehat{x}_k$, that can be bounded as

$$\frac{\|\Delta r_k^M\|_2}{\|b\|_2} \leq 3\sqrt{3k}\xi\kappa_2(A)^2 + k\sqrt{k}\xi\kappa_2(A). \tag{7.30}$$

Comparing this with the expression (7.27) for $MINRES_{GMRES}$, we see that we have now an error bound for $MINRES$, due to the evaluation of $\widehat{W}_k$ and the assembly of $\widehat{x}_k$, that is proportional with $\kappa_2(A)^2$. Of course, one might wonder how pessimistic these upper bounds are, but for the example given in Figure 7.5 (taken from [158]) we see that we may really encounter differences in practice that may be explained by the condition numbers.

Our analysis does not imply that MINRES is always an unattractive method. Of course, when A is not ill-conditioned, the method is attractive because of economy in storage. But also, if A is ill-conditioned, then the ill-conditioning is felt in the solution, if that solution has components of about

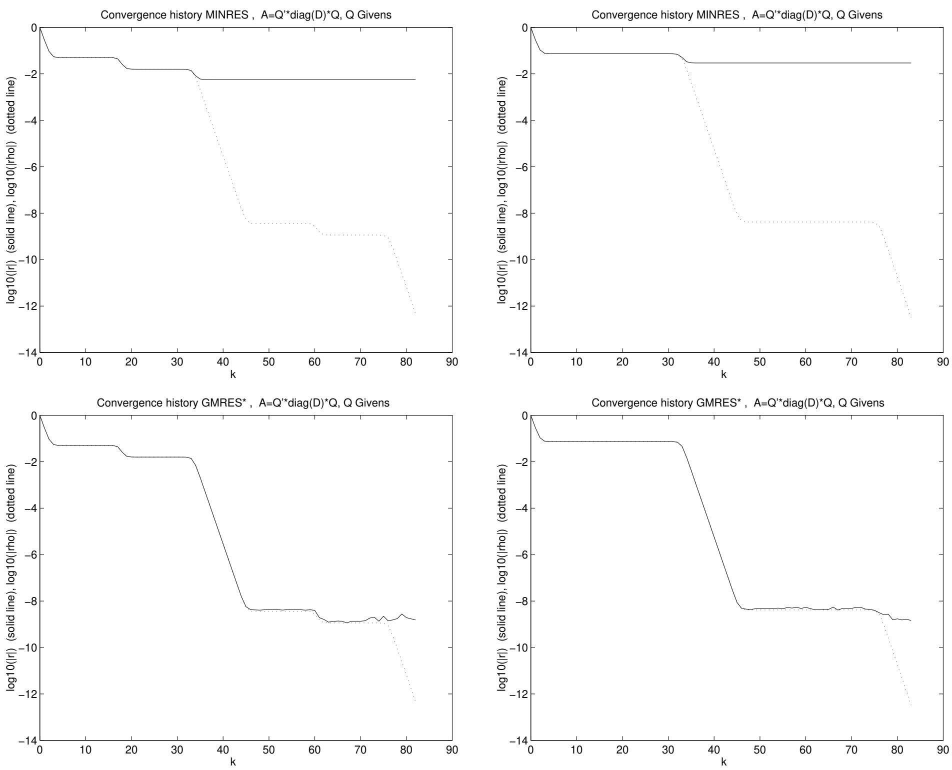

Figure 7.5: $MINRES$ (top) and $MINRES_{GMRES}$ (bottom): solid line (—) $\log_{10}$ of $\|b - A\widehat{x}_k\|_2/\|b\|_2$; dotted line $(\cdots)$ $\log_{10}$ of the estimated residual norm reduction ρ_k. The pictures show the results for a positive definite system (the left pictures) and for an indefinite system (the right pictures). For both examples $\kappa_2(A) = 3\,10^8$. To be more specific: at the left $A = GDG'$ with D diagonal, $D \equiv \mathrm{diag}(10^{-8}, 2\,10^{-8}, 2 : h : 3)$, $h = 1/789$, and G the Givens rotation in the $(1, 30)$-plane over an angle of $45°$; at the right $A = GDG'$ with D diagonal $D \equiv \mathrm{diag}(-10^{-8}, 10^{-8}, 2 : h : 3)$, $h = 1/389$, and G the same Givens rotation as for the left example; in both examples b is the vector with all coordinates equal to 1, $x_0 = 0$, and the relative machine precision $\xi = 1.1\,10^{-16}$.

the same magnitude in right singular vector directions corresponding to the smallest and the largest singular vectors. For some classes of problems, including problems from tomography, one wants to avoid components in the directions of the small singular vectors (regularization) and in such cases MINRES may be still attractive, because the Krylov methods, including

MINRES, tend to discover the small singular vectors after more iterations steps than the large singular vector directions. This may also be viewed as a sort of regularization, for an analysis of this phenomenon, see [170].

7.3.3 Some considerations for CG

The somewhat alarming aspect of our analysis of $MINRES$ and $MINRES_{GMRES}$ is that it gives the impression that we have to pay a high price for short recurrence relations, when we use these to transform the Krylov basis: the transformation from V_k to W_k in $MINRES_{GMRES}$. This may make us slightly nervous with respect to Conjugate Gradients, where the Krylov basis is also transformed, see 4.3: the transformation of the orthogonal R-basis to RU^{-1}. The discussion in [157, Section 4] indicates that the usual CG implementations, based on coupled two-term recurrence relations are very stable and lead to perturbations in the order of $\kappa(A)$ times machine precision at most. The crux is that in $MINRES$ we transform the basis by a three-term recurrence and these three-term recurrence relations may trigger parasytic components in the transformed vectors.

Chapter 8

Bi-CGSTAB

8.1 A more smoothly converging variant of CGS

We have mentioned that the residuals r_i in CGS satisfy the relation $r_i = P_i(A)^2 r_0$, in which $P_i(A)r_0$ just defines the residual $r_{bi-CG,i}$ in the bi-conjugate gradient method:

$$r_{bi-CG,i} = P_i(A)r_0.$$

By construction we have that $(P_i(A)r_0, P_j(A^T)\widehat{r}_0) = 0$ for $j < i$, which expresses the fact that $P_i(A)r_0$ is perpendicular to the subspace $K^i(A^T; \widehat{r}_0)$, spanned by the vectors

$$\widehat{r}_0, A^T\widehat{r}_0, ..., (A^T)^{i-1}\widehat{r}_0.$$

This implies that, in principle, we can also recover the bi-conjugate gradient iteration parameters by requiring that, e.g., r_i is perpendicular to $\widetilde{P}_j(A^T)\widehat{r}_0$, or, equivalently, $(\widetilde{P}_j(A)P_i(A)r_0, \widehat{r}_0) = 0$, for another suitable set of polynomials $\widetilde{P}_j$ of degree j. In bi-CG one takes $\widetilde{P}_j = P_j$, namely $\widehat{r}_j = P_j(A^T)\widehat{r}_0$. This is exploited in CG-S, as we have indicated before, since recursion relations for the vectors $P_j^2(A)r_0$ can be derived from those for $P_j(A)r_0$.

Of course, we can now construct iteration methods, by which the x_i are generated so that $r_i = \widetilde{P}_i(A)P_i(A)r_0$ with other i^{th} degree polynomials, like, e.g., Chebychev-polynomials, which might be more suitable. Unfortunately, the optimal parameters for the Chebychev polynomials are in general not easily obtainable and also the recurrence relations for the resulting method are more complicated than for CG-S. Another possibility is to take for $\widetilde{P}_j$ a polynomial of the form

$$Q_i(x) = (1 - \omega_1 x)(1 - \omega_2 x)...(1 - \omega_i x), \tag{8.1}$$

and to select suitable constants ω_j. This expression leads to an almost trivial recurrence relation for the Q_i.
An obvious possibility to select ω_j in the j^{th} iteration step is to minimize r_j, with respect to ω_j, for residuals that can be written as $r_j = Q_j(A)P_j(A)r_0$. This leads to the Bi-CGSTAB method [177].

We will now derive the (unpreconditioned) Bi-CGSTAB scheme. This will be done completely analogous to the derivation of CGS. We repeat the two important relations for the vectors r_i and p_i, extracted from the unpreconditioned version of Bi-CG, see Fig¿ 6.1 (take $K = I$):

$$\begin{aligned} p_i &= r_{i-1} + \beta_{i-1} p_{i-1} \\ r_i &= r_{i-1} - \alpha_i A p_i. \end{aligned}$$

We write r_i and p_i again in polynomial form:

$$r_i = P_i(A) r_0 \quad \text{and} \quad p_i = T_{i-1}(A) r_0,$$

in which $P_i(A)$ and $T_{i-1}(A)$ are polynomials in A of degree i, $i+1$, respectively. From the Bi-CG recurrence relations we obtain recurrence relations for these polynomials:

$$T_{i-1}(A) r_0 = (P_{i-1}(A) + \beta_{i-1} T_{i-2}(A)) r_0,$$

and

$$P_i(A) r_0 = (P_{i-1}(A) - \alpha_i A T_{i-1}(A)) r_0.$$

In the Bi-CGSTAB scheme we wish to have recurrence relations for

$$\widehat{r}_i = Q_i(A) P_i(A) r_0.$$

With Q_i as in (8.1) and the Bi-CG relation for the factors P_i and T_{i-1}, it then follows that

$$Q_i(A)P_i(A)r_0 = (1 - \omega_i A) Q_{i-1}(A)(P_{i-1}(A) - \alpha_i A T_{i-1}(A)) r_0$$

$$= \{Q_{i-1}(A)P_{i-1}(A) - \alpha_i A Q_{i-1}(A) T_{i-1}(A)\} r_0$$

$$-\omega_i A\{(Q_{i-1}(A)P_{i-1}(A) - \alpha_i A Q_{i-1}(A) T_{i-1}(A))\} r_0.$$

Clearly, we also need a relation for the product $Q_i(A)T_i(A)r_0$. This can also be obtained from the Bi-CG relations:

$$Q_i(A)T_i(A)r_0 = Q_i(A)(P_i(A) + \beta_i T_{i-1}(A))r_0$$

$$= Q_i(A)P_i(A)r_0 + \beta_i(1 - \omega_i A)Q_{i-1}(A)T_{i-1}(A)r_0$$

$$= Q_i(A)P_i(A)r_0 + \beta_i Q_{i-1}(A)T_{i-1}(A)r_0 - \beta_i\omega_i A Q_{i-1}(A)T_{i-1}(A)r_0.$$

Finally we have to recover the Bi-CG constants ρ_i, β_i, and α_i by inner-products in terms of the new vectors that we now have generated.
E.g., β_i can be computed as follows. First we compute

$$\widetilde{\rho}_i = (\widetilde{r}_0, Q_i(A)P_i(A)r_0) = (Q_i(A^T)\widetilde{r}_0, P_i(A)r_0).$$

By construction $P_i(A)r_0$ is orthogonal with respect to all vectors $U_{i-1}(A^T)\widetilde{r}_0$, where U_{i-1} is an arbitrary polynomial of degree $i-1$ at most. This means that we have to consider only the highest order term of $Q_i(A^T)$ when computing $\widetilde{\rho}_i$. This term is given by $(-1)^i\omega_1\omega_2\cdots\omega_i(A^T)^i$. We actually wish to compute

$$\rho_i = (P_i(A^T)\widehat{r}_0, P_i(A)r_0),$$

and since the highest order term of $P_i(A^T)$ is given by $(-1)^i\alpha_1\alpha_2\cdots\alpha_i(A^T)^i$, it follows that

$$\beta_i = (\widetilde{\rho}_i/\widetilde{\rho}_{i-1})(\alpha_{i-1}/\omega_{i-1}).$$

The other constants can be derived similarly.

Note that in our discussion we have focussed on the recurrence relations for the vectors r_i and p_i, while in fact our main goal is to determine x_i. As in all CG-type methods, x_i itself is not required for continuing the iteration, but it can easily be determined as a "sideproduct" by realizing that an update of the form $r_i = r_{i-1} - \gamma Ay$ corresponds to an update $x_i = x_{i-1} + \gamma y$ for the current approximated solution.

By writing r_i for $Q_i(A)P_i(A)r_0$ and p_i for $Q_{i-1}(A)T_{i-1}(A)r_0$, we obtain the following scheme for Bi-CGSTAB (we trust that, with the foregoing observations, the reader will now be able to verify the relations in Bi-CGSTAB). In this scheme we have computed the ω_i so that $r_i = Q_i(A)P_i(A)r_0$ is minimized in 2-norm as a function of ω_i. In Fig. 8.1, we have presented the algorithm, skipping the $\widehat{\ }$ notation for the Bi-CGSTAB iteration vectors.

In order to restrict on memory traffic, we have carried out both updates to the current solution x in one single step, while the updates to the residual

x_0 is an initial guess; $r_0 = b - Ax_0$
Choose $\widetilde{r}$, for example, $\widehat{r} = r_0$
for $i = 1, 2, ...$
    $\rho_{i-1} = \widetilde{r}^T r_{i-1}$
    if $\rho_{i-1} = 0$ method fails
    if $i = 1$
        $p_i = r_{i-1}$
    else
        $\beta_{i-1} = (\rho_{i-1}/\rho_{i-2})(\alpha_{i-1}/\omega_{i-1})$
        $p_i = r_{i-1} + \beta_{i-1}(p_{i-1} - \omega_{i-1}v_{i-1})$
    endif
    $v_i = Ap_i;$
    $\alpha_i = \rho_{i-1}/\widetilde{r}^T v_i$
    $s = r_{i-1} - \alpha_i v_i$
    check $\|s\|_2$, if small enough: $x_i = x_{i-1} + \alpha_i p_i$ and stop
    $t = As$, $\omega_i = t^T s / t^T t$
    $x_i = x_{i-1} + \alpha_i p_i + \omega_i s$
    $r_i = s - \omega_i t$
    check convergence; continue if necessary
    for continuation necessary that $\omega_i \neq 0$
end

Figure 8.1: The Bi-CGSTAB algorithm

r had to be done separately ($s = r_{i-1} - \alpha v_i$ and $r_i = s - \omega_i t$). So s represents the residual after a "Bi-CG step". If the norm of s is small enough then one might stop, but in that case, before stopping the algorithm, the current solution has to be updated appropriately as $x_i = x_{i-1} + \alpha p_i$ in order to be compatible with the current residual s (and the computation of t, ω_i, as well as the second update $\omega_i s$ should be skipped).

From the orthogonality property $(P_i(A)r_0, Q_j(A^T)\widehat{r}_0) = 0$, for $j < i$, it follows that Bi-CGSTAB is also a finite method, i.e., in exact arithmetic it will terminate after $m \leq n$ iteration steps. In this case we get $s = 0$ at iteration step m and ω_m is then not defined. This represents a lucky breakdown of the algorithm and the process should be terminated as indicated in the previous paragraph.

In the presented form Bi-CGSTAB requires, for the solution of an N by N system $Ax = b$, evaluation of 2 matrix vector products with A, $12N$ flops for vector updates and 4 inner products. This has to be compared with (unpreconditioned) CGS which requires also 2 matrix vector products with A, and $13N$ flops, but only 2 inner products. In practical situations, however, the two additional inner products lead to only a small increase in computational work per iteration step and this is readily undone by almost any reduction in the number of iteration steps (specially on vector computers on which inner products are usually fast operations).

Except for memory locations for x, b, r and A we need for Bi-CGSTAB memory space for 4 additional N-vectors $\widetilde{r}, p, v$, and t (note that r may be overwritten by s). This is the same as for CGS.

Of course, Bi-CGSTAB may suffer from the same break down problems as Bi-CG and CGS. These problems stem basically from the fact that for general matrices the bilinear form

$$[x, y] \equiv (P(A^T)x, P(A)y),$$

which is used to form the bi-orthogonality, does not define an inner product. In particular, it may occur that, by an unfavourable choice for $\widetilde{r}$, an iteration parameter ρ_i or $\widetilde{r}^T v_i$ is zero (or very small), without convergence having taken place. In an actual code one should test for such situations and take appropriate measures, e.g., restart with a different $\widetilde{r}$ or switch to another method (for example GMRES).

The preconditioned Bi-CGSTAB algorithm for solving the linear system $Ax = b$, with preconditioning K reads as in Fig. 8.2.

The matrix K in this scheme represents the preconditioning matrix and the way of preconditioning [177]. The above scheme in fact carries out the Bi-CGSTAB procedure for the explicitly postconditioned linear system

$$AK^{-1}y = b,$$

but the vectors y_i and the residual have been backtransformed to the vectors x_i and r_i corresponding to the original system $Ax = b$.

In exact arithmetic, the α_j and β_j have the same values as those generated by Bi-CG and CGS. Hence, they can be used to extract eigenvalue approximations for the eigenvalues of A (see Bi-CG).
Bi-CGSTAB can be viewed as the product of Bi-CG and GMRES(1). Of

x_0 is an initial guess; $r_0 = b - Ax_0$
Choose $\widetilde{r}$ (for example, $\widetilde{r} = r_0$;
for $i = 1, 2, ...$
    $\rho_{i-1} = \widetilde{r}^T r_{i-1})$
    if $\rho_{i-1} = 0$ **method fails**
    if $i = 1$
        $p_i = r_{i-1}$
    else
        $\beta_{i-1} = (\rho_{i-1}/\rho_{i-2})(\alpha_{i-1}/\omega_{i-1})$
        $p_i = r_{i-1} + \beta_{i-1}(p_{i-1} - \omega_{i-1}v_{i-1});$
    endif
    Solve $\widehat{p}$ from $K\widehat{p} = p_i$
    $v_i = A\widehat{p}$
    $\alpha_i = \rho_{i-1}/\widetilde{r}^T v_i$
    $s = r_{i-1} - \alpha_i v_i$
    if $\|s\|$ small enough **then**
        $x_i = x_{i-1} + \alpha_i \widehat{p}$, quit
    Solve $\widehat{s}$ from $K\widehat{s} = s$
    $t = A\widehat{s}$
    $\omega_i = t^T s / t^T t$
    $x_i = x_{i-1} + \alpha_i \widehat{p} + \omega_i \widehat{s}$
    if x_i is accurate enough **then** quit
    $r_i = s - \omega_i t$
    for continuation it is necessary that $\omega_i \neq 0$
end

Figure 8.2: The Bi-CGSTAB algorithm with preconditioning

course, other product methods can be formulated as well. Gutknecht [100] has proposed BiCGSTAB2, which is constructed as the product of Bi-CG and GMRES(2).

8.2 Bi-CGSTAB2 and variants

The residual $r_k = b - Ax_k$ in the Bi-Conjugate Gradient method, when applied to $Ax = b$ with start x_0 can be written formally as $P_k(A)r_0$, where P_k is a k-degree polynomial. These residuals are constructed with one operation with A and one with A^T per iteration step. It was pointed out in [160] that with about the same amount of computational effort one can construct residuals of the form $\widetilde{r}_k = P_k^2(A)r_0$, which is the basis for the CGS method. This can be achieved without any operation with A^T. The idea behind the improved efficiency of CGS is that if $P_k(A)$ is viewed as a reduction operator in BiCG, then one may hope that the square of this operator will be a twice as powerful reduction operator. Although this is not always observed in practice, one typically has that CGS converges faster than BiCG. This, together with the absence of operations with A^T, explains the success of the CGS method. A drawback of CGS is that its convergence behavior can look quite erratic, that is the norms of the resdiduals converge quite irregularly, and it may easily happen that $\|r_{k+1}\|_2$ is much larger than $\|r_k\|_2$ for certain k (for an explanation of this see [176]).
In [177] it was shown that by a similar approach as for CGS, one can construct methods for which r_k can be interpreted as $r_k = P_k(A)Q_k(A)r_0$, in which P_k is the polynomial associated with BiCG and Q_k can be selected free under the condition that $Q_k(0) = 1$. In [177] it was suggested to construct Q_k as the product of k linear factors $1 - \omega_j A$, where ω_j was taken to minimize locally a residual. This approach leads to the BiCGSTAB method. Because of the local minimization, BiCGSTAB displays a much smoother convergence behavior than CGS, and more surprisingly it often also converges (slightly) faster. One weak point in BiCGSTAB is that we get breakdown if an ω_j is equal to zero. One may equally expect negative effects when ω_j is small. In fact, BiCGSTAB can be viewed as the combined effect of BiCG and GCR(1), or GMRES(1), steps. As soon as the GCR(1) part of the algorithm (nearly) stagnates, then the BiCG part in the next iteration step cannot (or only poorly) be constructed.
Another dubious aspect of BiCGSTAB is that the factor Q_k has only real roots by construction. It is well-known that optimal reduction polynomials for matrices with complex eigenvalues may have complex roots as well. If, for instance, the matrix A is real skew-symmetric, then GCR(1) stagnates forever, whereas a method like GCR(2) (or GMRES(2)), in which we minimize over two combined successive search directions, may lead to convergence, and this is mainly due to the fact that then complex eigenvalue components in the error can be effectively reduced.

This point of view was taken in [100] for the construction of the BiCGSTAB2 method. In the odd-numbered iteration steps the Q-polynomial is expanded by a linear factor, as in BiCGSTAB, but in the even-numbered steps this linear factor is discarded, and the Q-polynomial from the previous even-numbered step is expanded by a quadratic $1 - \alpha_k A - \beta_k A^2$. For this construction the information from the odd-numbered step is required. It was anticipated that the introduction of quadratic factors in Q might help to improve convergence for systems with complex eigenvalues, and, indeed, some improvement was observed in practical situations (see also [141]).
However, our presentation suggests a possible weakness in the construction of BiCGSTAB2, namely in the odd-numbered steps the same problems may occur as in BiCGSTAB. Since the even-numbered steps rely on the results of the odd-numbered steps, this may equally lead to unnecessary break-downs or poor convergence. In [155] another, and even simpler approach was taken to arrive at the desired even-numbered steps, without the necessity of the construction of the intermediate BiCGSTAB-type step in the odd-numbered steps. Hence, in this approach the polynomial Q is constructed straight-away as a product of quadratic factors, without ever constructing a linear factor. As a result the new method BiCGSTAB(2) leads only to significant residuals in the even-numbered steps and the odd-numbered steps do not lead necessarily to useful approximations.
In fact, it is shown in [155] that the polynomial Q can also be constructed as the product of ℓ-degree factors, without the construction of the intermediate lower degree factors. The main idea is that ℓ successive BiCG steps are carried out, where for the sake of an A^T-free construction the already available part of Q is expanded by simple powers of A. This means that after the BICG part of the algorithm vectors from the Krylov subspace $s, As, A^2s, ..., A^\ell s$, with $s = P_k(A)Q_{k-\ell}(A)r_0$ are available, and it is then relatively easy to minimize the residual over that particular Krylov subspace. There are variants of this approach in which more stable bases for the Krylov subspaces are generated [159], but for low values of ℓ a standard basis satisfies, together with a minimum norm solution obtained through solving the associated normal equations (which requires the solution of an ℓ by ℓ system. In most cases BiCGSTAB(2) will already give nice results for problems where BiCGSTAB or BiCGSTAB2 may fail. Note, however, that, in exact arithmetic, if no break-down situation occurs, BiCGSTAB2 would produce exactly the same results as BiCGSTAB(2) at the even-numbered steps.

Bi-CGSTAB(2) can be represented by the scheme in Fig. 8.3.

For more general BiCGSTAB(ℓ) schemes see [155, 159].
Another advantage of BiCGSTAB(2) over BiCGSTAB2 is in its efficiency. The BiCGSTAB(2) algorithm requires 14 vector updates, 9 innerproducts and 4 matrix vector products per full cycle. This has to be compared with a combined odd-numbered and even-numbered step in BiCGSTAB2, which requires 22 vector updates, 11 innerproducts, and 4 matrix vector products, and with two steps of BiCGSTAB which require 4 matrix vector products, 8 innerproducts and 12 vector updates. The numbers for BiCGSTAB2 are based on an implementation described in [141].
Also with respect to memory requirements, BiCGSTAB(2) takes an intermediate position: it requires 2 n-vectors more than BiCGSTAB and 2 n-vectors less than BiCGSTAB2.

For distributed memory machines the innerproducts may cause communication overhead problems (see, e.g., [44]). We note that the BiCG steps are very similar to conjugate gradient iteration steps, so that we may consider all kind of tricks that have been suggested to reduce the number of synchronization points caused by the 4 innerproducts in the BiCG parts. For an overview of these approaches see [18]. If on a specific computer it is possible to overlap communication with communication, then the BiCG parts can be rescheduled as to create overlap possibillities: 1. the computation of ρ_1 in the even BiCG step may be done just before the update of u at the end of the GCR part.
2. The update of x_{i+2} may be delayed until after the computation of γ in the even BiCG step.
3. The computation of ρ_1 for the odd BiCG step can be done just before the update for x at the end of the even BiCG step.
4. The computation of γ in the odd BiCG step has already overlap possibillities with the update for u.
For the GCR(2) part we note that the 5 innerproducts can be taken together, in order to reduce start-up times for their global assembling. This gives the method BiCGSTAB(2) a (slight) advantage over BiCGSTAB. Furthermore we note that the updates in the GCR(2) may lead to more efficient code than for BiCGSTAB, since some of them can be combined.

	x_0 is an initial guess; $r_0 = b - Ax_0$;
	$\widehat{r}_0$ is an arbitrary vector, such that $(r, \widehat{r}_0) \neq 0$,
	e.g., $\widehat{r}_0 = r$;
	$\rho_0 = 1; u = 0; \alpha = 0; \omega_2 = 1$;
	for $i = 0, 2, 4, 6, ...$
	$\rho_0 = -\omega_2 \rho_0$
even BiCG step:	$\rho_1 = (\widehat{r}_0, r_i); \beta = \alpha\rho_1/\rho_0; \rho_0 = \rho_1$
	$u = r_i - \beta u$;
	$v = Au$
	$\gamma = (v, \widehat{r}_0); \alpha = \rho_0/\gamma$;
	$r = r_i - \alpha v$;
	$s = Ar$
	$x = x_i + \alpha u$;
odd BiCG step:	$\rho_1 = (\widehat{r}_0, s); \beta = \alpha\rho_1/\rho_0; \rho_0 = \rho_1$
	$v = s - \beta v$;
	$w = Av$
	$\gamma = (w, \widehat{r}_0); \alpha = \rho_0/\gamma$;
	$u = r - \beta u$
	$r = r - \alpha v$
	$s = s - \alpha w$
	$t = As$
GCR(2)-part:	$\omega_1 = (r, s); \mu = (s, s); \nu = (s, t); \tau = (t, t)$;
	$\omega_2 = (r, t); \tau = \tau - \nu^2/\mu; \omega_2 = (\omega_2 - \nu\omega_1/\mu)/\tau$;
	$\omega_1 = (\omega_1 - \nu\omega_2)/\mu$
	$x_{i+2} = x + \omega_1 r + \omega_2 s + \alpha u$
	$r_{i+2} = r - \omega_1 s - \omega_2 t$
	if x_{i+2} accurate enough **then** quit
	$u = u - \omega_1 v - \omega_2 w$
	end

Figure 8.3: The Bi-CGSTAB(2) algorithm

Chapter 9

Termination Criteria

An important point, when using iterative processes, is to decide when to terminate the process. Popular stopping criteria are based on the norm of the current residual, or on the norm of the update to the current approximation to the solution (or a combination of these norms). More sophisticated criteria have been discussed in litterature.
In [115] a practical termination criterion for the conjugate gradient method is considered. Suppose we want an approximation x_i for the solution x for which

$$\|x_i - x\|_2 / \|x\|_2 \leq \varepsilon,$$

where ε is a tolerance set by the user.
It is shown in [115] that such an approximation is obtained by CG as soon as

$$\|r_i\|_2 \leq \mu_1 \|x_i\|_2 \varepsilon / (1 + \varepsilon),$$

where μ_1 stands for the smallest eigenvalue of the positive definite symmetric (preconditioned) matrix A. Of course, in most applications the value for μ_1 will be unknown, but with the iteration coefficients of CG we can build the tridiagonal matrix T_i, and compute the smallest eigenvalue (Ritz value) $\mu_1^{(i)}$ of T_i, which is an approximation for μ_1. In [115] a simple algorithm for the computation of $\mu_1^{(i)}$, along with the CG algorithm, is described, and it is shown that a rather robust stopping criterion is formed by

$$\|r_i\|_2 \leq \mu_1^{(i)} \|x_i\|_2 \varepsilon / (1 + \varepsilon).$$

A similar criterion has also been suggested earlier in [104].

A quite different, but much more generally applicable approach has been suggested in [5]. In this approach the approximate solution of an iterative process is regarded as the exact solution of some (nearby) linear system, and computable bounds for the perturbations with respect to the given system are given. A nice overview of termination criteria has been presented in [18]: Section 4.2.