CHAPTER

6

KRYLOV SUBSPACE METHODS PART I

The next two chapters explore a few methods which are considered currently to be among the most important iterative techniques available for solving large linear systems. These techniques are based on projection processes, both orthogonal and oblique, onto Krylov subspaces, which are subspaces spanned by vectors of the form $p(A)v$ where p is a polynomial. In short, these techniques approximate $A^{-1}b$ by $p(A)b$, where p is a "good" polynomial. This chapter covers methods derived from, or related to, the Arnoldi orthogonalization. The next chapter covers methods based on Lanczos biorthogonalization.

6.1 INTRODUCTION

Recall from the previous chapter that a general *projection method* for solving the linear system

$$Ax = b, \tag{6.1}$$

is a method which seeks an approximate solution x_m from an affine subspace $x_0 + \mathcal{K}_m$ of dimension m by imposing the Petrov-Galerkin condition

$$b - Ax_m \perp \mathcal{L}_m,$$

where $\mathcal{L}_m$ is another subspace of dimension m. Here, x_0 represents an arbitrary initial guess to the solution. A Krylov subspace method is a method for which the subspace $\mathcal{K}_m$ is the Krylov subspace

$$\mathcal{K}_m(A, r_0) = span\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\},$$

where $r_0 = b - Ax_0$. When there is no ambiguity, $\mathcal{K}_m(A, r_0)$ will be denoted by $\mathcal{K}_m$. The different versions of Krylov subspace methods arise from different choices of the subspace $\mathcal{L}_m$ and from the ways in which the system is *preconditioned*, a topic that will be covered in detail in later chapters.

Viewed from the angle of approximation theory, it is clear that the approximations obtained from a Krylov subspace method are of the form

$$A^{-1}b \approx x_m = x_0 + q_{m-1}(A)r_0,$$

in which q_{m-1} is a certain polynomial of degree $m-1$. In the simplest case where $x_0 = 0$, then

$$A^{-1}b \approx q_{m-1}(A)b.$$

In other words, $A^{-1}b$ is approximated by $q_{m-1}(A)b$.

Although all the techniques provide the same type of *polynomial* approximations, the choice of $\mathcal{L}_m$, i.e., the constraints used to build these approximations, will have an important effect on the iterative technique. Two broad choices for $\mathcal{L}_m$ give rise to the best-known techniques. The first is simply $\mathcal{L}_m = \mathcal{K}_m$ and the minimum-residual variation $\mathcal{L}_m = A\mathcal{K}_m$. A few of the numerous methods in this category will be described in this chapter. The second class of methods is based on defining $\mathcal{L}_m$ to be a Krylov subspace method associated with A^T, namely, $\mathcal{L}_m = \mathcal{K}_m(A^T, r_0)$. Methods of this class will be covered in the next chapter. There are also block extensions of each of these methods termed *block Krylov subspace methods*, which will be discussed only briefly. Note that a projection method may have several different implementations, giving rise to different algorithms which are all mathematically equivalent.

6.2 KRYLOV SUBSPACES

In this section we consider projection methods on *Krylov subspaces*, i.e., subspaces of the form

$$\mathcal{K}_m(A, v) \equiv \text{span}\,\{v, Av, A^2v, \ldots, A^{m-1}v\} \tag{6.2}$$

which will be denoted simply by $\mathcal{K}_m$ if there is no ambiguity. The dimension of the subspace of approximants increases by one at each step of the approximation process. A few elementary properties of Krylov subspaces can be established, many of which need no proof. A first property is that $\mathcal{K}_m$ is the subspace of all vectors in $\mathbb{R}^n$ which can be written as $x = p(A)v$, where p is a polynomial of degree not exceeding $m-1$. Recall that the minimal polynomial of a vector v is the nonzero monic polynomial p of lowest degree such that $p(A)v = 0$. The degree of the minimal polynomial of v with respect to A is often called the *grade of v with respect to A*, or simply the grade of v if there is no ambiguity. A consequence of the Cayley-Hamilton theorem is that the grade of v does not exceed n. The following proposition is easy to prove.

PROPOSITION 6.1 *Let μ be the grade of v. Then $\mathcal{K}_\mu$ is invariant under A and $\mathcal{K}_m = \mathcal{K}_\mu$ for all $m \geq \mu$.*

It was mentioned above that the dimension of $\mathcal{K}_m$ is nondecreasing. In fact, the following proposition determines the dimension of $\mathcal{K}_m$ in general.

PROPOSITION 6.2 *The Krylov subspace $\mathcal{K}_m$ is of dimension m if and only if the grade μ of v with respect to A is not less than m, i.e.,*

$$\dim(\mathcal{K}_m) = m \quad \leftrightarrow \quad \mathrm{grade}(v) \geq m.$$

Therefore,

$$\dim(\mathcal{K}_m) = \min\,\{m, \mathrm{grade}(v)\}.$$

Proof. The vectors $v, Av, \ldots, A^{m-1}v$ form a basis of $\mathcal{K}_m$ if and only if for any set of m scalars $\alpha_i, i = 0, \ldots, m-1$, where at least one α_i is nonzero, the linear combination $\sum_{i=0}^{m-1} \alpha_i A^i v$ is nonzero. This is equivalent to the condition that the only polynomial of degree $\leq m-1$ for which $p(A)v = 0$ is the zero polynomial. The second part of the proposition is a consequence of the previous proposition. ∎

PROPOSITION 6.3 *Let Q_m be any projector onto $\mathcal{K}_m$ and let A_m be the section of A to $\mathcal{K}_m$, that is, $A_m = Q_m A_{|\mathcal{K}_m}$. Then for any polynomial q of degree not exceeding $m-1$,*

$$q(A)v = q(A_m)v,$$

and for any polynomial of degree $\leq m$,

$$Q_m q(A)v = q(A_m)v.$$

Proof. First we prove that $q(A)v = q(A_m)v$ for any polynomial q of degree $\leq m-1$. It is sufficient to show the property for the monic polynomials $q_i(t) \equiv t^i,\ i = 0, \ldots, m-1$. The proof is by induction. The property is true for the polynomial $q_0(t) \equiv 1$. Assume that it is true for $q_i(t) \equiv t^i$:

$$q_i(A)v = q_i(A_m)v.$$

Multiplying the above equation by A on both sides yields

$$q_{i+1}(A)v = Aq_i(A_m)v.$$

If $i+1 \leq m-1$ the vector on the left-hand side belongs to $\mathcal{K}_m$, and therefore if the above equation is multiplied on both sides by Q_m, then

$$q_{i+1}(A)v = Q_m A q_i(A_m)v.$$

Looking at the right-hand side we observe that $q_i(A_m)v$ belongs to $\mathcal{K}_m$. Hence,

$$q_{i+1}(A)v = Q_m A_{|\mathcal{K}_m} q_i(A_m)v = q_{i+1}(A_m)v,$$

which proves that the property is true for $i+1$, provided $i+1 \leq m-1$. For the case $i+1 = m$, it only remains to show that $Q_m q_m(A)v = q_m(A_m)v$, which follows from

$q_{m-1}(A)v = q_{m-1}(A_m)v$ by simply multiplying both sides by $Q_m A$. ■

6.3 ARNOLDI'S METHOD

Arnoldi's method [9] is an orthogonal projection method onto $\mathcal{K}_m$ for general non-Hermitian matrices. The procedure was introduced in 1951 as a means of reducing a dense matrix into Hessenberg form. Arnoldi presented his method in this manner but hinted that the eigenvalues of the Hessenberg matrix obtained from a number of steps smaller than n could provide accurate approximations to some eigenvalues of the original matrix. It was later discovered that this strategy leads to an efficient technique for approximating eigenvalues of large sparse matrices. The method will first be described theoretically, i.e., assuming exact arithmetic, then implementation details will be addressed.

6.3.1 THE BASIC ALGORITHM

Arnoldi's procedure is an algorithm for building an orthogonal basis of the Krylov subspace $\mathcal{K}_m$. In exact arithmetic, one variant of the algorithm is as follows:

ALGORITHM 6.1: Arnoldi

1. *Choose a vector* v_1 *of norm 1*
2. *For* $j = 1, 2, \ldots, m$ *Do:*
3.   *Compute* $h_{ij} = (Av_j, v_i)$ *for* $i = 1, 2, \ldots, j$
4.   *Compute* $w_j := Av_j - \sum_{i=1}^{j} h_{ij} v_i$
5. $h_{j+1,j} = \|w_j\|_2$
6.   *If* $h_{j+1,j} = 0$ *then Stop*
7. $v_{j+1} = w_j / h_{j+1,j}$
8. *EndDo*

At each step, the algorithm multiplies the previous Arnoldi vector v_j by A and then orthonormalizes the resulting vector w_j against all previous v_i's by a standard Gram-Schmidt procedure. It will stop if the vector w_j computed in line 4 vanishes. This case will be examined shortly. Now a few simple properties of the algorithm are proved.

PROPOSITION 6.4 *Assume that Algorithm 6.1 does not stop before the* m*-th step. Then the vectors* $v_1, v_2, \ldots, v_m$ *form an orthonormal basis of the Krylov subspace*

$$\mathcal{K}_m = \mathrm{span}\{v_1, Av_1, \ldots, A^{m-1}v_1\}.$$

Proof. The vectors $v_j, j = 1, 2, \ldots, m$, are orthonormal by construction. That they span $\mathcal{K}_m$ follows from the fact that each vector v_j is of the form $q_{j-1}(A)v_1$ where q_{j-1} is a polynomial of degree $j-1$. This can be shown by induction on j as follows. The result is clearly true for $j = 1$, since $v_1 = q_0(A)v_1$ with $q_0(t) \equiv 1$. Assume that the result is true for all integers $\leq j$ and consider v_{j+1}. We have

$$h_{j+1}v_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij}v_i = Aq_{j-1}(A)v_1 - \sum_{i=1}^{j} h_{ij}q_{i-1}(A)v_1 \tag{6.3}$$

which shows that v_{j+1} can be expressed as $q_j(A)v_1$ where q_j is of degree j and completes the proof. ■

PROPOSITION 6.5 *Denote by V_m, the $n \times m$ matrix with column vectors $v_1, \ldots, v_m$, by $\bar{H}_m$, the $(m+1) \times m$ Hessenberg matrix whose nonzero entries h_{ij} are defined by Algorithm 6.1, and by H_m the matrix obtained from $\bar{H}_m$ by deleting its last row. Then the following relations hold:*

$$AV_m = V_mH_m + w_me_m^T \tag{6.4}$$

$$= V_{m+1}\bar{H}_m, \tag{6.5}$$

$$V_m^TAV_m = H_m. \tag{6.6}$$

Proof. The relation (6.5) follows from the following equality which is readily derived from lines 4, 5, and 7 of Algorithm 6.1,

$$Av_j = \sum_{i=1}^{j+1} h_{ij}v_i, \quad j = 1, 2, \ldots, m. \tag{6.7}$$

Relation (6.4) is a matrix reformulation of (6.7). Relation (6.6) follows by multiplying both sides of (6.4) by V_m^T and making use of the orthonormality of $\{v_1, \ldots, v_m\}$. ■

The result of the proposition is illustrated in Figure 6.1.

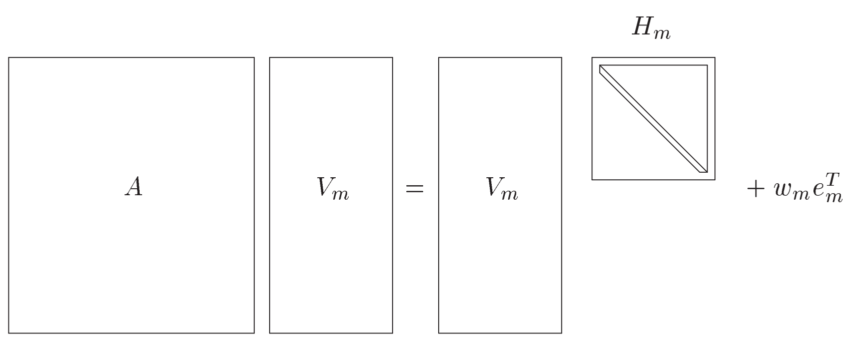

Figure 6.1 *The action of A on V_m gives V_mH_m plus a rank-one matrix.*

As was noted earlier, the algorithm may break down in case the norm of w_j vanishes at a certain step j. In this case, the vector v_{j+1} cannot be computed and the algorithm stops.

Still to be determined are the conditions under which this situation occurs.

PROPOSITION 6.6 *Arnoldi's algorithm breaks down at step j (i.e., $h_{j+1,j} = 0$ in line 5 of Algorithm 6.1), if and only if the minimal polynomial of v_1 is of degree j. Moreover, in this case the subspace $\mathcal{K}_j$ is invariant under A.*

Proof. If the degree of the minimal polynomial is j, then w_j must be equal to zero. Indeed, otherwise v_{j+1} can be defined and as a result $\mathcal{K}_{j+1}$ would be of dimension $j + 1$. Then Proposition 6.2 would imply that $\mu \geq j + 1$, which is a contradiction. To prove the converse, assume that $w_j = 0$. Then the degree μ of the minimal polynomial of v_1 is such that $\mu \leq j$. Moreover, it is impossible that $\mu < j$. Otherwise, by the first part of this proof, the vector w_μ would be zero and the algorithm would have stopped at the earlier step number μ. The rest of the result follows from Proposition 6.1. ∎

A corollary of the proposition is that a projection method onto the subspace $\mathcal{K}_j$ will be exact when a breakdown occurs at step j. This result follows from Proposition 5.6 seen in Chapter 5. It is for this reason that such breakdowns are often called *lucky breakdowns*.

6.3.2 PRACTICAL IMPLEMENTATIONS

In the previous description of the Arnoldi process, exact arithmetic was assumed, mainly for simplicity. In practice, much can be gained by using the Modified Gram-Schmidt or the Householder algorithm instead of the standard Gram-Schmidt algorithm. With the Modified Gram-Schmidt alternative the algorithm takes the following form:

ALGORITHM 6.2: Arnoldi-Modified Gram-Schmidt

1. *Choose a vector* v_1 *of norm 1*
2. *For* $j = 1, 2, \ldots, m$ *Do:*
3. *Compute* $w_j := Av_j$
4. *For* $i = 1, \ldots, j$ *Do:*
5. $h_{ij} = (w_j, v_i)$
6. $w_j := w_j - h_{ij}v_i$
7. *EndDo*
8. $h_{j+1,j} = \|w_j\|_2$. *If* $h_{j+1,j} = 0$ *Stop*
9. $v_{j+1} = w_j / h_{j+1,j}$
10. *EndDo*

In exact arithmetic, this algorithm and Algorithm 6.1 are mathematically equivalent. In the presence of round-off the above formulation is much more reliable. However, there are cases where cancellations are so severe in the orthogonalization steps that even the Modified Gram-Schmidt option is inadequate. In this case, two further improvements can be utilized.

The first improvement resorts to double orthogonalization. Whenever the final vector w_j obtained at the end of the main loop in the above algorithm has been computed, a test is performed to compare its norm with the norm of the initial w_j (which is $\|Av_j\|_2$). If the reduction falls below a certain threshold, indicating severe cancellation might have occurred, a second orthogonalization is made. It is known from a result by Kahan that additional orthogonalizations are superfluous (see, for example, Parlett [160]).

The second improvement is to use a different technique altogether. From the numerical point of view, one of the most reliable orthogonalization techniques is the Householder algorithm. Recall from Chapter 1 that the Householder orthogonalization uses reflection matrices of the form $P_k = I - 2w_k w_k^T$ to transform a matrix X into upper triangular form. In the Arnoldi algorithm, the column vectors of the matrix X to be orthonormalized are not available ahead of time. Instead, the next vector is obtained as Av_j, where v_j is the current basis vector. In the Householder algorithm an orthogonal column v_i is obtained as $P_1 P_2 \ldots P_i e_i$ where $P_1, \ldots, P_i$ are the previous Householder matrices. This vector is then multiplied by A and the previous Householder transforms are applied to it. Then, the next Householder transform is determined from the resulting vector. This procedure is described in the following algorithm, which was originally proposed by Walker [221].

ALGORITHM 6.3: Householder Arnoldi

1. *Select a nonzero vector* v; *Set* $z_1 = v$
2. *For* $j = 1, \ldots, m, m+1$ *Do:*
3.   *Compute the Householder unit vector* w_j *such that*
4. $(w_j)_i = 0, i = 1, \ldots, j-1$ *and*
5. $(P_j z_j)_i = 0, i = j+1, \ldots, n$, *where* $P_j = I - 2w_j w_j^T$
6. $h_{j-1} = P_j z_j$
7. $v_j = P_1 P_2 \ldots P_j e_j$
8.   *If* $j \le m$ *compute* $z_{j+1} := P_j P_{j-1} \ldots P_1 A v_j$
9. *EndDo*

For details regarding the determination of the Householder vector w_j in the third to fifth lines and on its use in the sixth to eight lines, see Chapter 1. Recall that the matrices P_j need not be formed explicitly. To obtain h_{j-1} from z_j in line 6, zero out all the components from position $j+1$ through n of the n-vector z_j and change its j-th component, leaving all others unchanged. Thus, the $n \times m$ matrix $[h_0, h_1, \ldots, h_m]$ will have the same structure as the matrix X_m of equation (1.22) in Chapter 1. By comparison with the Householder algorithm seen in Chapter 1, we can infer that the above process computes the QR factorization of the matrix $v, Av_1, Av_2, Av_3, \ldots, Av_m$. Define

$$Q_j = P_j P_{j-1} \ldots P_1. \tag{6.8}$$

The definition of z_{j+1} in line 8 of the algorithm yields the relation,

$$Q_j A v_j = z_{j+1}.$$

After the next Householder transformation P_{j+1} is applied in line 6, h_j satisfies the relation,

$$h_j = P_{j+1} z_{j+1} = P_{j+1} Q_j A v_j = Q_{j+1} A v_j. \tag{6.9}$$

Now observe that since the components $j+2,\ldots,n$ of h_j are zero, then $P_i h_j = h_j$ for any $i \geq j+2$. Hence,

$$h_j = P_m P_{m-1} \ldots P_{j+2} h_j = Q_m A v_j, \quad j = 1, \ldots, m.$$

This leads to the factorization,

$$Q_m[v, Av_1, Av_2, \ldots, Av_m] = [h_0, h_1, \ldots, h_m] \tag{6.10}$$

where the matrix $[h_0, \ldots, h_m]$ is $n \times (m+1)$ and is upper triangular and Q_m is unitary.

It is important to relate the vectors v_i and h_i defined in this algorithm with vectors of the standard Arnoldi process. Let $\bar{H}_m$ be the $(m+1) \times\ m$ matrix obtained from the first $m+1$ rows of the $n \times m$ matrix $[h_1, \ldots, h_m]$. Since Q_{j+1} is unitary we have $Q_{j+1}^{-1} = Q_{j+1}^T$ and hence, from the relation (6.9)

$$Av_j = Q_{j+1}^T \sum_{i=1}^{j+1} h_{ij} e_i = \sum_{i=1}^{j+1} h_{ij} Q_{j+1}^T e_i$$

where each e_i is the i-th column of the $n \times n$ identity matrix. Since $P_k e_i = e_i$ for $i < k$, it is not difficult to see that

$$Q_{j+1}^T e_i = P_1 \ldots P_{j+1} e_i = v_i, \text{ for } i \leq j+1. \tag{6.11}$$

This yields the relation $Av_j = \sum_{i=1}^{j+1} h_{ij} v_i$, for $j = 1, \ldots, m$, which can be written in matrix form as

$$AV_m = V_{m+1} \bar{H}_m.$$

This is identical with the relation (6.5) obtained with the Gram-Schmidt or Modified Gram-Schmidt implementation. The v_i's form an orthonormal basis of the Krylov subspace $\mathcal{K}_m$ and are identical with the v_i's defined by the Arnoldi process, apart from a possible sign difference.

Although the Householder algorithm is numerically more viable than the Gram-Schmidt or Modified Gram-Schmidt versions, it is also more expensive. The cost of each of the outer loops, corresponding to the j control variable, is dominated by lines 7 and 8. These apply the reflection matrices P_i for $i = 1, \ldots, j$ to a vector, perform the matrix-vector product Av_j, and then apply the matrices P_i for $i = j, j-1, \ldots, 1$ to a vector. The application of each P_i to a vector is performed as

$$(I - 2w_i w_i^T)v = v - \sigma w_i \quad \text{with} \quad \sigma = 2w_i^T v.$$

This is essentially the result of a dot-product of length $n - i + 1$ followed by a vector update of the same length, requiring a total of about $4(n-i+1)$ operations for each application of P_i. Neglecting the last step, the number of operations due to the Householder transformations alone approximately totals

$$\sum_{j=1}^{m} \sum_{i=1}^{j} 8(n-i+1) = 8 \sum_{j=1}^{m} \left(jn - \frac{j(j-1)}{2} \right) \approx 4m^2 n - \frac{4}{3} m^3.$$

The table below shows the costs of different orthogonalization procedures. GS stands for Gram-Schmidt, MGS for Modified Gram-Schmidt, MGSR for Modified Gram-Schmidt with reorthogonalization, and HO for Householder.

	GS	MGS	MGSR	HO
Flops	$2m^2n$	$2m^2n$	$4m^2n$	$4m^2n - \frac{4}{3}m^3$
Storage	$(m+1)n$	$(m+1)n$	$(m+1)n$	$(m+1)n - \frac{1}{2}m^2$

The number of operations shown for MGSR corresponds to the worst case scenario when a second orthogonalization is performed each time. In practice, the number of operations is usually closer to that of the standard MGS. Regarding storage, the vectors $v_i, i = 1, \ldots, m$ need not be saved. In the algorithms for solving linear systems, these vectors are needed at the end of the process. This issue will be covered with the Householder implementations of these algorithms. For now, assume that only the w_i's are saved. The small gain in memory usage in the Householder version can be explained by the diminishing lengths of the vectors required at each step of the Householder transformation. However, this difference is negligible relative to the whole storage requirement of the algorithm, because $m \ll n$, typically.

The Householder orthogonalization may be a reasonable choice when developing general purpose, reliable software packages where robustness is a critical criterion. This is especially true for solving eigenvalue problems since the cost of orthogonalization is then amortized over several eigenvalue/eigenvector calculations. When solving linear systems, the Modified Gram-Schmidt orthogonalization, with a reorthogonalization strategy based on a measure of the level of cancellation, is more than adequate in most cases.

6.4 ARNOLDI'S METHOD FOR LINEAR SYSTEMS (FOM)

Given an initial guess x_0 to the original linear system $Ax = b$, we now consider an orthogonal *projection method* as defined in the previous chapter, which takes $\mathcal{L} = \mathcal{K} = \mathcal{K}_m(A, r_0)$, with

$$\mathcal{K}_m(A, r_0) = span\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\}, \tag{6.12}$$

in which $r_0 = b - Ax_0$. This method seeks an approximate solution x_m from the affine subspace $x_0 + \mathcal{K}_m$ of dimension m by imposing the Galerkin condition

$$b - Ax_m \perp \mathcal{K}_m. \tag{6.13}$$

If $v_1 = r_0/\|r_0\|_2$ in Arnoldi's method, and set $\beta = \|r_0\|_2$, then

$$V_m^T A V_m = H_m$$

by (6.6) and

$$V_m^T r_0 = V_m^T(\beta v_1) = \beta e_1.$$

As a result, the approximate solution using the above m-dimensional subspaces is given by

$$x_m = x_0 + V_m y_m, \tag{6.14}$$

$$y_m = H_m^{-1}(\beta e_1). \tag{6.15}$$

A method based on this approach and called the Full Orthogonalization Method (FOM) is described next. Modified Gram-Schmidt is used in the Arnoldi step.

ALGORITHM 6.4: Full Orthogonalization Method (FOM)

1. *Compute* $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, *and* $v_1 := r_0/\beta$
2. *Define the* $m \times m$ *matrix* $H_m = \{h_{ij}\}_{i,j=1,\ldots,m}$*; Set* $H_m = 0$
3. *For* $j = 1, 2, \ldots, m$ *Do:*
4. *Compute* $w_j := Av_j$
5. *For* $i = 1, \ldots, j$ *Do:*
6. $h_{ij} = (w_j, v_i)$
7. $w_j := w_j - h_{ij}v_i$
8. *EndDo*
9. *Compute* $h_{j+1,j} = \|w_j\|_2$*. If* $h_{j+1,j} = 0$ *set* $m := j$ *and Goto 12*
10. *Compute* $v_{j+1} = w_j/h_{j+1,j}$.
11. *EndDo*
12. *Compute* $y_m = H_m^{-1}(\beta e_1)$ *and* $x_m = x_0 + V_m y_m$

The above algorithm depends on a parameter m which is the dimension of the Krylov subspace. In practice it is desirable to select m in a dynamic fashion. This would be possible if the residual norm of the solution x_m is available inexpensively (without having to compute x_m itself). Then the algorithm can be stopped at the appropriate step using this information. The following proposition gives a result in this direction.

PROPOSITION 6.7 *The residual vector of the approximate solution* x_m *computed by the FOM Algorithm is such that*

$$b - Ax_m = -h_{m+1,m} e_m^T y_m v_{m+1}$$

and, therefore,

$$\|b - Ax_m\|_2 = h_{m+1,m} |e_m^T y_m|. \tag{6.16}$$

Proof. We have the relations,

$$\begin{aligned} b - Ax_m &= b - A(x_0 + V_m y_m) \\ &= r_0 - AV_m y_m \\ &= \beta v_1 - V_m H_m y_m - h_{m+1,m} e_m^T y_m v_{m+1}. \end{aligned}$$

By the definition of y_m, $H_m y_m = \beta e_1$, and so $\beta v_1 - V_m H_m y_m = 0$ from which the result follows immediately. ■

A rough estimate of the cost of each step of the algorithm is determined as follows. If $Nz(A)$ is the number of nonzero elements of A, then m steps of the Arnoldi procedure will require m matrix-vector products at the cost of $2m \times Nz(A)$. Each of the Gram-Schmidt steps costs approximately $4 \times j \times n$ operations, which brings the total over the m steps to approximately $2m^2n$. Thus, on the average, a step of FOM costs approximately

$$2Nz(A) + 2mn.$$

Regarding storage, m vectors of length n are required to save the basis V_m. Additional vectors must be used to keep the current solution and right-hand side, and a scratch vector for the matrix-vector product. In addition, the Hessenberg matrix H_m must be saved. The total is therefore roughly

$$(m+3)n + \frac{m^2}{2}.$$

In most situations m is small relative to n, so this cost is dominated by the first term.

6.4.1 VARIATION 1: RESTARTED FOM

Consider now the algorithm from a practical viewpoint. As m increases, the computational cost increases at least as $O(m^2)n$ because of the Gram-Schmidt orthogonalization. The memory cost increases as $O(mn)$. For large n this limits the largest value of m that can be used. There are two remedies. The first is to restart the algorithm periodically and the second is to "truncate" the orthogonalization in the Arnoldi algorithm. In this section we consider the first of these two options, which is described below.

ALGORITHM 6.5: Restarted FOM (FOM(m))

1. *Compute* $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, *and* $v_1 = r_0/\beta$.
2. *Generate the Arnoldi basis and the matrix* H_m *using the Arnoldi algorithm*
3. *starting with* v_1.
4. *Compute* $y_m = H_m^{-1}\beta e_1$ *and* $x_m = x_0 + V_m y_m$. *If satisfied then Stop.*
5. *Set* $x_0 := x_m$ *and go to 1.*

There are many possible variations to this basic scheme. One that is generally more economical in practice is based on the observation that sometimes a small m is sufficient for convergence and sometimes the largest possible m is necessary. Hence, the idea of averaging over different values of m. Start the algorithm with $m = 1$ and increment m by one in line 5 until a certain m_{max} is reached, after which m is reset to one, or kept the same. These variations will not be considered here.

Example 6.1 Table 6.1 shows the results of applying the FOM algorithm with no preconditioning to three of the test problems described in Section 3.7.

Matrix	Iters	Kflops	Residual	Error
F2DA	109	4442	0.36E-03	0.67E-04
F3D	66	11664	0.87E-03	0.35E-03
ORS	300	13558	0.26E+00	0.71E-04

Table 6.1 *A test run of FOM with no preconditioning.*

The column labeled *Iters* shows the total actual number of matrix-vector multiplications (matvecs) required to converge. The stopping criterion used is that the 2-norm of the residual be reduced by a factor of 10^7 relative to the 2-norm of the initial residual. A maximum of 300 matvecs are allowed. *Kflops* is the total number of floating point operations performed, in thousands. *Residual* and *Error* represent the two-norm of the residual and error vectors, respectively. In this test, m was taken to be 10. Note that the method did not succeed in solving the third problem.

6.4.2 VARIATION 2: IOM AND DIOM

A second alternative to FOM is to truncate the Arnoldi recurrence. Specifically, an integer k is selected and the following "incomplete" orthogonalization is performed.

ALGORITHM 6.6: Incomplete Orthogonalization Process

```
For j = 1, 2, ..., m Do:
   Compute w := A v_j
   For i = max{1, j - k + 1}, ..., j Do:
      h_{i,j} = (w, v_i)
      w := w - h_{ij} v_i
   EndDo
   Compute h_{j+1,j} = ||w||_2 and v_{j+1} = w / h_{j+1,j}
EndDo
```

The number of directions k against which to orthogonalize may be dictated by memory limitations. The Incomplete Orthogonalization Method (IOM) consists of performing the above incomplete orthogonalization procedure and computing an approximate solution using the same formulas (6.14) and (6.15).

ALGORITHM 6.7: IOM Algorithm

Run a modification of Algorithm 6.4 in which the Arnoldi process in lines 3 to 11 is replaced by the Incomplete Orthogonalization process and every other computation remains unchanged.

It is now necessary to keep only the k previous v_i vectors. The others are not needed in the above process and may be discarded. However, the difficulty remains that when the solution is computed by formula (6.14), all the vectors v_i for $i = 1, 2, \ldots, m$ are required. One option is to recompute them at the end, but essentially this doubles the cost of the algorithm. Fortunately, a formula can be developed whereby the current approximate solution x_m can be updated from the previous approximation x_{m-1} and a small number of vectors that are also updated at each step. This *progressive* formulation of the solution leads to an algorithm termed *Direct IOM* (DIOM) which we now derive.

The Hessenberg matrix H_m obtained from the incomplete orthogonalization process has a band structure with a bandwidth of $k+1$. For example, when $k=3$ and $m=5$, it is of the form

$$H_m = \begin{pmatrix} h_{11} & h_{12} & h_{13} & & \\ h_{21} & h_{22} & h_{23} & h_{24} & \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \end{pmatrix}. \tag{6.17}$$

The *Direct* version of IOM is derived from exploiting the special structure of the LU factorization, $H_m = L_m U_m$, of the matrix H_m. Assuming no pivoting is used, the matrix L_m is unit lower bidiagonal and U_m is banded upper triangular, with k diagonals. Thus, the above matrix has a factorization of the form

$$H_m = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ & l_{32} & 1 & & \\ & & l_{43} & 1 & \\ & & & l_{54} & 1 \end{pmatrix} \times \begin{pmatrix} u_{11} & u_{12} & u_{13} & & \\ & u_{22} & u_{23} & u_{24} & \\ & & u_{33} & u_{34} & u_{35} \\ & & & u_{44} & u_{45} \\ & & & & u_{55} \end{pmatrix}.$$

The approximate solution is then given by

$$x_m = x_0 + V_m U_m^{-1} L_m^{-1}(\beta e_1).$$

Defining

$$P_m \equiv V_m U_m^{-1}$$

and

$$z_m = L_m^{-1}(\beta e_1),$$

the approximate solution is given by

$$x_m = x_0 + P_m z_m. \tag{6.18}$$

Because of the structure of U_m, P_m can be updated easily. Indeed, equating the last columns of the matrix relation $P_m U_m = V_m$ yields

$$\sum_{i=m-k+1}^{m} u_{im} p_i = v_m,$$

which allows the vector p_m to be computed from the previous p_i's and v_m, with the help of the relation,

$$p_m = \frac{1}{u_{mm}} \left[v_m - \sum_{i=m-k+1}^{m-1} u_{im} p_i \right].$$

In addition, because of the structure of L_m, we have the relation

$$z_m = \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix}$$

in which

$$\zeta_m = -l_{m,m-1}\zeta_{m-1}.$$

From (6.18),

$$x_m = x_0 + [P_{m-1}, p_m]\begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix} = x_0 + P_{m-1}z_{m-1} + \zeta_m p_m.$$

Noting that $x_0 + P_{m-1}z_{m-1} = x_{m-1}$, it follows that the approximation x_m can be updated at each step by the relation,

$$x_m = x_{m-1} + \zeta_m p_m$$

where p_m is defined above. This gives the following algorithm, called the Direct Incomplete Orthogonalization Method (DIOM).

ALGORITHM 6.8: DIOM

1. *Choose* x_0 *and compute* $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, $v_1 := r_0/\beta$.
2. *For* $m = 1, 2, \ldots,$ *until convergence Do:*
3. *Compute* h_{im}, $i = \max\{1, m-k+1\}, \ldots, m$ *and* v_{m+1} *as in*
4. *lines 2-7 of Algorithm (6.6).*
5. *Update the LU factorization of* H_m*, i.e., obtain the last column*
6. *of* U_m *using the previous* k *pivots. If* $u_{mm} = 0$ *Stop.*
7. $\zeta_m = \{$ *if* $m = 1$ *then* β*, else* $-l_{m,m-1}\,\zeta_{m-1}\}$
8. $p_m = u_{mm}^{-1}\left(v_m - \sum_{i=m-k+1}^{m-1} u_{im}p_i\right)$ *(for* $i \le 0$ *set* $u_{im}p_i \equiv 0$*)*
9. $x_m = x_{m-1} + \zeta_m p_m$
10. *EndDo*

Note that the above algorithm is based implicitly on Gaussian elimination without pivoting for the solution of the Hessenberg system $H_m y_m = \beta e_1$. This may cause a premature termination in line 6. Fortunately, there is an implementation based on Gaussian elimination with partial pivoting. The details of this variant can be found in [174]. DIOM can also be derived by imposing the properties that are satisfied by the residual vector and the conjugate directions, i.e., the p_i's.

Observe that (6.4) is still valid and as a consequence, Proposition 6.7, which is based on it, still holds. That is because the orthogonality properties were not used to derive the two relations therein. Since the residual vector is a scalar multiple of v_{m+1} and since the v_i's are no longer orthogonal, IOM and DIOM are not orthogonal projection techniques. They can, however, be viewed as oblique projection techniques onto $\mathcal{K}_m$ and orthogonal to an artificially constructed subspace.

PROPOSITION 6.8 *IOM and DIOM are mathematically equivalent to a projection process onto* $\mathcal{K}_m$ *and orthogonally to*

$$\mathcal{L}_m = \mathrm{span}\{z_1, z_2, \ldots, z_m\}$$

where

$$z_i = v_i - (v_i, v_{m+1})v_{m+1}, \quad i = 1, \ldots, m.$$

Proof. The proof is an immediate consequence of the fact that r_m is a multiple of v_{m+1} and by construction, v_{m+1} is orthogonal to all z_i's defined in the proposition. ■

The following simple properties can be shown:

- The residual vectors r_i, $i = 1, \ldots, m$, are "locally" orthogonal,
$$(r_j, r_i) = 0, \quad \text{for} \quad |i - j| \leq k, \;\; i \neq j.$$
- The p_j's are locally A-orthogonal to the Arnoldi vectors, i.e.,
$$(Ap_j, v_i) = 0 \quad \text{for} \quad j - k + 1 < i < j.$$
- For the case $k = \infty$ (full orthogonalization) the p_j's are semi-conjugate, i.e.,
$$(Ap_j, p_i) = 0 \quad \text{for} \quad i < j.$$

6.5 GMRES

The Generalized Minimum Residual Method (GMRES) is a projection method based on taking $\mathcal{K} = \mathcal{K}_m$ and $\mathcal{L} = A\mathcal{K}_m$, in which $\mathcal{K}_m$ is the m-th Krylov subspace with $v_1 = r_0/\|r_0\|_2$. As seen in Chapter 5, such a technique minimizes the residual norm over all vectors in $x_0 + \mathcal{K}_m$. The implementation of an algorithm based on this approach is similar to that of the FOM algorithm. We first describe the basic idea and then discuss a few practical variations.

6.5.1 THE BASIC GMRES ALGORITHM

There are two ways to derive the algorithm. The first way exploits the optimality property and the relation (6.5). Any vector x in $x_0 + \mathcal{K}_m$ can be written as

$$x = x_0 + V_m y, \tag{6.19}$$

where y is an m-vector. Defining

$$J(y) = \|b - Ax\|_2 = \|b - A\left(x_0 + V_m y\right)\|_2, \tag{6.20}$$

the relation (6.5) results in

$$\begin{aligned} b - Ax &= b - A\left(x_0 + V_m y\right) \\ &= r_0 - AV_m y \\ &= \beta v_1 - V_{m+1}\bar{H}_m y \\ &= V_{m+1}\left(\beta e_1 - \bar{H}_m y\right). \end{aligned} \tag{6.21}$$

Since the column-vectors of V_{m+1} are orthonormal, then

$$J(y) \equiv \|b - A\left(x_0 + V_m y\right)\|_2 = \|\beta e_1 - \bar{H}_m y\|_2. \tag{6.22}$$

The GMRES approximation is the unique vector of $x_0 + \mathcal{K}_m$ which minimizes (6.20). By (6.19) and (6.22), this approximation can be obtained quite simply as $x_m = x_0 + V_m y_m$ where y_m minimizes the function $J(y) = \|\beta e_1 - \bar{H}_m y\|_2$, i.e.,

$$x_m = x_0 + V_m y_m, \quad \text{where} \tag{6.23}$$

$$y_m = \text{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2. \tag{6.24}$$

The minimizer y_m is inexpensive to compute since it requires the solution of an $(m+1)\times m$ least-squares problem where m is typically small. This gives the following algorithm.

ALGORITHM 6.9: GMRES

1. *Compute* $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, *and* $v_1 := r_0/\beta$
2. *Define the* $(m+1) \times m$ *matrix* $\bar{H}_m = \{h_{ij}\}_{1\le i\le m+1, 1\le j\le m}$. *Set* $\bar{H}_m = 0$.
3. *For* $j = 1, 2, \ldots, m$ *Do:*
4.   *Compute* $w_j := Av_j$
5.   *For* $i = 1, \ldots, j$ *Do:*
6. $h_{ij} := (w_j, v_i)$
7. $w_j := w_j - h_{ij} v_i$
8.   *EndDo*
9. $h_{j+1,j} = \|w_j\|_2$. *If* $h_{j+1,j} = 0$ *set* $m := j$ *and go to 12*
10. $v_{j+1} = w_j / h_{j+1,j}$
11. *EndDo*
12. *Compute* y_m *the minimizer of* $\|\beta e_1 - \bar{H}_m y\|_2$ *and* $x_m = x_0 + V_m y_m$.

The second way to derive the GMRES algorithm is to use the equations (5.7) with $W_m = AV_m$. This is the subject of Exercise 4.

6.5.2 THE HOUSEHOLDER VERSION

The previous algorithm utilizes the Modified Gram-Schmidt orthogonalization in the Arnoldi process. Section 6.3.2 described a Householder variant of the Arnoldi process which is numerically more robust than Gram-Schmidt. Here, we focus on a modification of GMRES which retrofits the Householder orthogonalization. Section 6.3.2 explained how to get the v_j and the columns of $\bar{H}_{m+1}$ at each step, from the Householder-Arnoldi algorithm. Since V_m and $\bar{H}_m$ are the only items needed to extract the approximate solution at the end of the GMRES process, the modification seems rather straightforward. However, this is only true if the v_i's are stored. In this case, line 12 would remain the same and the modification to the algorithm would be in lines 3-11 which are to be replaced by the Householder variant of the Arnoldi process. It was mentioned in Section 6.3.2 that it is preferable not to store the v_i's because this would double the storage requirement. In this case, a formula must be found to generate the approximate solution in line 12, using only the w_i's, i.e., the P_i's. Let

$$y_m = (\eta_1, \eta_2, \cdots, \eta_m)^T,$$

so that the solution is of the form $x_m = x_0 + \eta_1 v_1 + \cdots + \eta_m v_m$. Recall that in the Householder variant of the Arnoldi process, each v_j is defined by

$$v_j = P_1 P_2 \dots P_j e_j.$$

Using a Horner-like scheme, we obtain

$$\begin{aligned} x_m &= x_0 + \eta_1 P_1 e_1 + \eta_2 P_1 P_2 e_2 + \dots + \eta_m P_1 P_2 \dots P_m e_m \\ &= x_0 + P_1 \left(\eta_1 e_1 + P_2 \left(\eta_2 e_2 + \dots + P_{m-1} \left(\eta_{m-1} e_{m-1} + P_m \eta_m e_m\right)\right)\right). \end{aligned}$$

Therefore, when Householder orthogonalization is used, then line 12 of the GMRES algorithm should be replaced by a step of the form

$$z := 0 \tag{6.25}$$

$$z := P_j \left(\eta_j e_j + z\right), j = m, m-1, \dots, 1 \tag{6.26}$$

$$x_m = x_0 + z. \tag{6.27}$$

The above step requires roughly as many operations as computing the last Arnoldi vector v_m. Therefore, its cost is negligible relative to the cost of the Arnoldi loop.

ALGORITHM 6.10: GMRES with Householder orthogonalization

1. *Compute* $r_0 = b - Ax_0$, $z := r_0$.
2. *For* $j = 1, \dots, m, m+1$ *Do:*
3.   *Compute the Householder unit vector* w_j *such that*
4. $(w_j)_i = 0, i = 1, \dots, j-1$ *and*
5. $(P_j z)_i = 0, i = j+1, \dots, n$ *where* $P_j = I - 2w_j w_j^T$;
6. $h_{j-1} := P_j z$; *If* $j = 1$ *then let* $\beta := e_1^T h_0$.
7. $v := P_1 P_2 \dots P_j e_j$.
8.   *If* $j \le m$ *compute* $z := P_j P_{j-1} \dots P_1 A v$,
9. *EndDo*
10. *Define* $\bar{H}_m$ *= the* $(m+1) \times m$ *upper part of the matrix* $[h_1, \dots, h_m]$.
11. *Compute* $y_m = \text{Argmin}_y \|\beta e_1 - \bar{H}_m y\|_2$. *Let* $y_m = (\eta_1, \eta_2, \dots, \eta_m)^T$.
12. *z := 0*
13. *For* $j = m, m-1, \dots, 1$ *Do:*
14. $z := P_j \left(\eta_j e_j + z\right)$,
15. *EndDo*
16. *Compute* $x_m = x_0 + z$

Note that now only the set of w_j vectors needs to be saved. The scalar β defined in line 6 is equal to $\pm\|r_0\|_2$. This is because $P_1 z = \beta e_1$ where β is defined by the equations (1.21) seen in Chapter 1, which define the first Householder transformation. As was observed earlier the Householder factorization actually obtains the QR factorization (6.10) with $v = r_0$. We can also formulate GMRES directly from this factorization. Indeed, if $x = x_0 + V_m y_m$, then according to this factorization, the corresponding residual norm is equal to

$$\|h_0 - \eta_1 h_1 - \eta_2 h_2 - \dots - \eta_m h_m\|_2$$

whose minimizer is the same as the one defined by the algorithm.

The details of implementation of the solution of the least-squares problem as well as the estimate of the residual norm are identical with those of the Gram-Schmidt versions and are discussed next.

6.5.3 PRACTICAL IMPLEMENTATION ISSUES

A clear difficulty with Algorithm 6.9 is that it does not provide the approximate solution x_m explicitly at each step. As a result, it is not easy to determine when to stop. One remedy is to compute the approximation solution x_m at regular intervals and check for convergence by a test on the residual, for example. However, there is a more elegant solution which is related to the way in which the least-squares problem (6.24) is solved.

In order to solve the least-squares problem $\min \|\beta e_1 - \bar{H}_m y\|$, it is natural to transform the Hessenberg matrix into upper triangular form by using plane rotations. Define the rotation matrices

$$\begin{array}{cc} \Omega_i = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c_i & s_i & & & \\ & & & -s_i & c_i & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix} & \begin{array}{l} \\ \\ \\ \leftarrow \text{row } i \\ \leftarrow \text{row } i+1 \\ \\ \\ \\ \end{array} \end{array} \tag{6.28}$$

with $c_i^2 + s_i^2 = 1$. If m steps of the GMRES iteration are performed then these matrices have dimension $(m+1) \times (m+1)$.

Multiply the Hessenberg matrix $\bar{H}_m$ and the corresponding right-hand side $\bar{g}_0 \equiv \beta e_1$ by a sequence of such matrices from the left. The coefficients s_i, c_i are selected to eliminate $h_{i+1,i}$ at each time. Thus, if $m = 5$ we would have

$$\bar{H}_5 = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \quad \bar{g}_0 = \begin{pmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Then premultiply $\bar{H}_5$ by

$$\Omega_1 = \begin{pmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}$$

with

$$s_1 = \frac{h_{21}}{\sqrt{h_{11}^2 + h_{21}^2}}, \quad c_1 = \frac{h_{11}}{\sqrt{h_{11}^2 + h_{21}^2}}$$

to obtain the matrix and right-hand side

$$\bar{H}_5^{(1)} = \begin{pmatrix} h_{11}^{(1)} & h_{12}^{(1)} & h_{13}^{(1)} & h_{14}^{(1)} & h_{15}^{(1)} \\ & h_{22}^{(1)} & h_{23}^{(1)} & h_{24}^{(1)} & h_{25}^{(1)} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \quad \bar{g}_1 = \begin{pmatrix} c_1\beta \\ -s_1\beta \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{6.29}$$

We can now premultiply the above matrix and right-hand side again by a rotation matrix Ω_2 to eliminate h_{32}. This is achieved by taking

$$s_2 = \frac{h_{32}}{\sqrt{(h_{22}^{(1)})^2 + h_{32}^2}}, \quad c_2 = \frac{h_{22}^{(1)}}{\sqrt{(h_{22}^{(1)})^2 + h_{32}^2}}.$$

This elimination process is continued until the m-th rotation is applied, which transforms the problem into one involving the matrix and right-hand side,

$$\bar{H}_5^{(5)} = \begin{pmatrix} h_{11}^{(5)} & h_{12}^{(5)} & h_{13}^{(5)} & h_{14}^{(5)} & h_{15}^{(5)} \\ & h_{22}^{(5)} & h_{23}^{(5)} & h_{24}^{(5)} & h_{25}^{(5)} \\ & & h_{33}^{(5)} & h_{34}^{(5)} & h_{35}^{(5)} \\ & & & h_{44}^{(5)} & h_{45}^{(5)} \\ & & & & h_{55}^{(5)} \\ & & & & 0 \end{pmatrix}, \quad \bar{g}_5 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ . \\ . \\ \gamma_6 \end{pmatrix}. \tag{6.30}$$

Generally, the scalars c_i and s_i of the i^{th} rotation Ω_i are defined as

$$s_i = \frac{h_{i+1,i}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}}, \quad c_i = \frac{h_{ii}^{(i-1)}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}}. \tag{6.31}$$

Define Q_m the product of matrices Ω_i,

$$Q_m = \Omega_m \Omega_{m-1} \dots \Omega_1 \tag{6.32}$$

and

$$\bar{R}_m = \bar{H}_m^{(m)} = Q_m \bar{H}_m, \tag{6.33}$$

$$\bar{g}_m = Q_m(\beta e_1) = (\gamma_1, \dots, \gamma_{m+1})^T. \tag{6.34}$$

Since Q_m is unitary,

$$\min \|\beta e_1 - \bar{H}_m y\|_2 = \min \|\bar{g}_m - \bar{R}_m y\|_2.$$

The solution to the above least-squares problem is obtained by simply solving the triangular system resulting from deleting the last row of the matrix $\bar{R}_m$ and right-hand side $\bar{g}_m$ in (6.30). In addition, it is clear that for the solution y_*, the "residual" $\|\beta e_1 - \bar{H}_m y_*\|$ is nothing but the last element of the right-hand side, i.e., the term γ_6 in the above illustration.

PROPOSITION 6.9 *Let $\Omega_i, i = 1, \dots, m$ be the rotation matrices used to transform $\bar{H}_m$ into an upper triangular form and $\overline{R}_m$, $\bar{g}_m = (\gamma_1, \dots, \gamma_{m+1})^T$ the resulting matrix and right-hand side, as defined by (6.33), (6.34). Denote by R_m the $m \times m$ upper triangular* *matrix obtained from $\bar{R}_m$ by deleting its last row and by g_m the m-dimensional vector obtained from $\bar{g}_m$ by deleting its last component. Then,*

1. *The rank of AV_m is equal to the rank of R_m. In particular, if $r_{mm} = 0$ then A must be singular.*
2. *The vector y_m which minimizes $\|\beta e_1 - \bar{H}_m y\|_2$ is given by*
$$y_m = R_m^{-1} g_m.$$
3. *The residual vector at step m satisfies*
$$b - Ax_m = V_{m+1}\left(\beta e_1 - \bar{H}_m y_m\right) = V_{m+1} Q_m^T (\gamma_{m+1} e_{m+1}) \tag{6.35}$$
and, as a result,
$$\|b - Ax_m\|_2 = |\gamma_{m+1}|. \tag{6.36}$$

Proof. To prove first part (1), use (6.5), to obtain the relation

$$\begin{aligned} AV_m &= V_{m+1}\bar{H}_m \\ &= V_{m+1} Q_m^T Q_m \bar{H}_m \\ &= V_{m+1} Q_m^T \bar{R}_m. \end{aligned}$$

Since $V_{m+1}Q_m^T$ is unitary, the rank of AV_m is that of $\bar{R}_m$, which equals the rank of R_m since these two matrices differ only by a zero row (the last row of $\bar{R}_m$). If $r_{mm} = 0$ then R_m is of rank $\leq m-1$ and as a result AV_m is also of rank $\leq m-1$. Since V_m is of full rank, this means that A must be singular.

The second part (2), was essentially proved before the proposition. For any vector y,

$$\begin{aligned} \|\beta e_1 - \bar{H}_m y\|_2^2 &= \|Q_m(\beta e_1 - \bar{H}_m y)\|_2^2 \\ &= \|\bar{g}_m - \bar{R}_m y\|_2^2 \\ &= |\gamma_{m+1}|^2 + \|g_m - R_m y\|_2^2 \end{aligned} \tag{6.37}$$

The minimum of the left-hand side is reached when the second term in the right-hand side of (6.37) is zero. Since R_m is nonsingular, this is achieved when $y = R_m^{-1} g_m$.

To prove the third part (3), we start with the definitions used for GMRES and the relation (6.21). For any $x = x_0 + V_m y$,

$$\begin{aligned} b - Ax &= V_{m+1}\left(\beta e_1 - \bar{H}_m y\right) \\ &= V_{m+1} Q_m^T\, Q_m \left(\beta e_1 - \bar{H}_m y\right) \\ &= V_{m+1} Q_m^T \left(\bar{g}_m - \bar{R}_m y\right). \end{aligned}$$

As was seen in the proof of the second part above, the 2-norm of $\bar{g}_m - \bar{R}_m y$ is minimized when y annihilates all components of the right-hand side $\bar{g}_m$ except the last one, which is equal to γ_{m+1}. As a result,

$$b - Ax_m = V_{m+1} Q_m^T (\gamma_{m+1} e_{m+1})$$

which is (6.35). The result (6.36) follows from the orthonormality of the column-vectors of $V_{m+1}Q_m^T$. ∎

So far we have only described a process for computing the least-squares solution y_m

of (6.24). Note that this approach with plane rotations can also be used to solve the linear system (6.15) for the FOM method. The only difference is that the last rotation Ω_m must be omitted. In particular, a single program can be written to implement both algorithms using a switch for selecting the FOM or GMRES options.

It is possible to implement the above process in a progressive manner, i.e., at each step of the GMRES algorithm. This approach will allow one to obtain the residual norm at every step, with virtually no additional arithmetic operations. To illustrate this, start with (6.30), i.e., assume that the first m rotations have already been applied. Now the residual norm is available for x_5 and the stopping criterion can be applied. Assume that the test dictates that further steps be taken. One more step of the Arnoldi algorithm must be executed to get Av_6 and the 6-th column of $\bar{H}_6$. This column is appended to $\bar{R}_5$ which has been augmented by a zero row to match the dimension. Then the previous rotations $\Omega_1, \Omega_2, \ldots, \Omega_5$ are applied to this last column. After this is done the following matrix and right-hand side are obtained:

$$H_6^{(5)} = \begin{pmatrix} h_{11}^{(5)} & h_{12}^{(5)} & h_{13}^{(5)} & h_{14}^{(5)} & h_{15}^{(5)} & h_{16}^{(5)} \\ & h_{22}^{(5)} & h_{23}^{(5)} & h_{24}^{(5)} & h_{25}^{(5)} & h_{26}^{(5)} \\ & & h_{33}^{(5)} & h_{34}^{(5)} & h_{35}^{(5)} & h_{36}^{(5)} \\ & & & h_{44}^{(5)} & h_{45}^{(5)} & h_{46}^{(5)} \\ & & & & h_{55}^{(5)} & h_{56}^{(5)} \\ & & & & 0 & h_{66}^{(5)} \\ & & & & 0 & h_{76} \end{pmatrix}, \; g_6^{(5)} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ . \\ . \\ \gamma_6 \\ 0 \end{pmatrix}. \tag{6.38}$$

The algorithm now continues in the same way as before. We need to premultiply the matrix by a rotation matrix Ω_6 (now of size 7×7) with

$$s_6 = \frac{h_{76}}{\sqrt{(h_{66}^{(5)})^2 + h_{76}^2}}, \quad c_6 = \frac{h_{66}^{(5)}}{\sqrt{(h_{66}^{(5)})^2 + h_{76}^2}}$$

to get the matrix and right-hand side,

$$\bar{R}_6 = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} & r_{16} \\ & r_{22} & r_{23} & r_{24} & r_{25} & r_{26} \\ & & r_{33} & r_{34} & r_{35} & r_{36} \\ & & & r_{44} & r_{45} & r_{46} \\ & & & & r_{55} & r_{56} \\ & & & & & r_{66} \\ & & & & & 0 \end{pmatrix}, \; \bar{g}_6 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ . \\ . \\ c_6\gamma_6 \\ -s_6\gamma_6 \end{pmatrix}. \tag{6.39}$$

If the residual norm as given by $|\gamma_{m+1}|$ is small enough, the process must be stopped. The last rows of $\bar{R}_m$ and $\bar{g}_m$ are deleted and the resulting upper triangular system is solved to obtain y_m. Then the approximate solution $x_m = x_0 + V_m y_m$ is computed.

Note from (6.39) that the following useful relation for γ_{j+1} results

$$\gamma_{j+1} = -s_j \gamma_j. \tag{6.40}$$

In particular, if $s_j = 0$ then the residual norm must be equal to zero which means that the solution is exact at step j.

6.5.4 BREAKDOWN OF GMRES

If Algorithm 6.9 is examined carefully, we observe that the only possibilities of breakdown in GMRES are in the Arnoldi loop, when $\hat{v}_{j+1} = 0$, i.e., when $h_{j+1,j} = 0$ at a given step j. In this situation, the algorithm stops because the next Arnoldi vector cannot be generated. However, in this situation, the residual vector is zero, i.e., the algorithm will deliver the exact solution at this step. In fact, the converse is also true: If the algorithm stops at step j with $b - Ax_j = 0$, then $h_{j+1,j} = 0$.

PROPOSITION 6.10 *Let A be a nonsingular matrix. Then, the GMRES algorithm breaks down at step j, i.e., $h_{j+1,j} = 0$, if and only if the approximate solution x_j is exact.*

Proof. To show the necessary condition, observe that if $h_{j+1,j} = 0$, then $s_j = 0$. Indeed, since A is nonsingular, then $r_{jj} = h_{jj}^{(j-1)}$ is nonzero by the first part of Proposition 6.9 and (6.31) implies $s_j = 0$. Then, the relations (6.36) and (6.40) imply that $r_j = 0$.

To show the sufficient condition, we use (6.40) again. Since the solution is exact at step j and not at step $j-1$, then $s_j = 0$. From the formula (6.31), this implies that $h_{j+1,j} = 0$. ∎

6.5.5 RELATIONS BETWEEN FOM AND GMRES

If the last row of the least-squares system in (6.38) is deleted, instead of the one in (6.39), i.e., before the last rotation Ω_6 is applied, the same approximate solution as FOM would result. As a practical consequence a single subroutine can be written to handle both cases. This observation can also be helpful in understanding the relationships between the two algorithms.

We begin by establishing an interesting relation between the FOM and GMRES iterates, which will be exploited in the next chapter. A general lemma is first shown regarding the solutions of the triangular systems

$$R_m y_m = g_m$$

obtained from applying successive rotations to the Hessenberg matrices $\bar{H}_m$. As was stated before, the only difference between the y_m vectors obtained in GMRES and Arnoldi is that the last rotation Ω_m is omitted in FOM. In other words, the R_m matrix for the two methods differs only in its (m,m) entry while the right-hand sides differ only in their last components.

LEMMA 6.1 *Let $\tilde{R}_m$ be the $m \times m$ upper part of the matrix $Q_{m-1}\bar{H}_m$ and, as before, let R_m be the $m \times m$ upper part of the matrix $Q_m \bar{H}_m$. Similarly, let $\tilde{g}_m$ be the vector of the first m components of $Q_{m-1}(\beta e_1)$ and let g_m be the vector of the first m components of $Q_m(\beta e_1)$. Define*

$$\tilde{y}_m = \tilde{R}_m^{-1}\tilde{g}_m, \quad y_m = R_m^{-1} g_m$$

the y vectors obtained for an m-dimensional FOM and GMRES methods, respectively.

Then

$$y_m - \begin{pmatrix} y_{m-1} \\ 0 \end{pmatrix} = c_m^2 \left(\tilde{y}_m - \begin{pmatrix} y_{m-1} \\ 0 \end{pmatrix} \right) \tag{6.41}$$

in which c_m is the cosine used in the m-th rotation Ω_m, as defined by (6.31).

Proof. The following relation holds:

$$R_m = \begin{pmatrix} R_{m-1} & z_m \\ 0 & \xi_m \end{pmatrix}, \quad \tilde{R}_m = \begin{pmatrix} R_{m-1} & z_m \\ 0 & \tilde{\xi}_m \end{pmatrix}.$$

Similarly, for the right-hand sides,

$$g_m = \begin{pmatrix} g_{m-1} \\ \gamma_m \end{pmatrix}, \quad \tilde{g}_m = \begin{pmatrix} g_{m-1} \\ \tilde{\gamma}_m \end{pmatrix}$$

with

$$\gamma_m = c_m \tilde{\gamma}_m. \tag{6.42}$$

Denoting by λ the scalar $\sqrt{\tilde{\xi}_m^2 + h_{m+1,m}^2}$, and using the definitions of s_m and c_m, we obtain

$$\xi_m = c_m \tilde{\xi}_m + s_m h_{m+1,m} = \frac{\tilde{\xi}_m^2}{\lambda} + \frac{h_{m+1,m}^2}{\lambda} = \lambda = \frac{\tilde{\xi}_m}{c_m}. \tag{6.43}$$

Now,

$$y_m = R_m^{-1} g_m = \begin{pmatrix} R_{m-1}^{-1} & -\frac{1}{\xi_m} R_{m-1}^{-1} z_m \\ 0 & \frac{1}{\xi_m} \end{pmatrix} \begin{pmatrix} g_{m-1} \\ \gamma_m \end{pmatrix} \tag{6.44}$$

which, upon observing that $R_{m-1}^{-1} g_{m-1} = y_{m-1}$, yields,

$$y_m - \begin{pmatrix} y_{m-1} \\ 0 \end{pmatrix} = \frac{\gamma_m}{\xi_m} \begin{pmatrix} -R_{m-1}^{-1} z_m \\ 1 \end{pmatrix}. \tag{6.45}$$

Replacing y_m, ξ_m, γ_m by $\tilde{y}_m, \tilde{\xi}_m, \tilde{\gamma}_m$, respectively, in (6.44), a relation similar to (6.45) would result except that γ_m / ξ_m is replaced by $\tilde{\gamma}_m / \tilde{\xi}_m$ which, by (6.42) and (6.43), satisfies the relation

$$\frac{\gamma_m}{\xi_m} = c_m^2 \frac{\tilde{\gamma}_m}{\tilde{\xi}_m}.$$

The result follows immediately. ■

If the FOM and GMRES iterates are denoted by the superscripts F and G, respectively, then the relation (6.41) implies that

$$x_m^G - x_{m-1}^G = c_m^2 \left(x_m^F - x_{m-1}^G \right),$$

or,

$$x_m^G = s_m^2 x_{m-1}^G + c_m^2 x_m^F. \tag{6.46}$$

This leads to the following relation for the residual vectors obtained by the two methods,

$$r_m^G = s_m^2 r_{m-1}^G + c_m^2 r_m^F \tag{6.47}$$

which indicates that, in general, the two residual vectors will evolve hand in hand. In particular, if $c_m = 0$, then GMRES will not progress at step m, a phenomenon known as stagnation. However, in this situation, according to the definitions (6.31) of the rotations, $h_{mm}^{(m-1)} = 0$ which implies that H_m is singular and, therefore, x_m^F is not defined. In fact, the reverse of this is also true, a result due to Brown [43], which is stated without proof in the following proposition.

PROPOSITION 6.11 *If at any given step m, the GMRES iterates make no progress, i.e., if $x_m^G = x_{m-1}^G$ then H_m is singular and x_m^F is not defined. Conversely, if H_m is singular at step m, i.e., if FOM breaks down at step m, and A is nonsingular, then $x_m^G = x_{m-1}^G$.*

Note also that the use of the above lemma is not restricted to the GMRES-FOM pair. Some of the iterative methods defined in this chapter and the next involve a least-squares problem of the form (6.24). In such cases, the iterates of the least-squares method and those of the orthogonal residual (Galerkin) method will be related by the same equation.

Another important observation from (6.40) is that if ρ_i is the residual norm $\|b - Ax_i\|_2$ obtained at step i, then

$$\rho_m^G = |s_m| \rho_{m-1}^G.$$

The superscripts G and F are used again to distinguish between GMRES and FOM quantities. A consequence of this is that,

$$\rho_m^G = |s_1 s_2 \ \dots \ s_m| \beta. \tag{6.48}$$

Now consider the FOM iterates, assuming that x_m is defined, i.e., that H_m is nonsingular. An equation similar to (6.48) for FOM can be derived. Using the same notation as in the proof of the lemma, and recalling that

$$\rho_m^F = h_{m+1,m} |e_m^T H_m^{-1}(\beta e_1)|,$$

note that

$$e_m^T H_m^{-1}(\beta e_1) = \frac{\tilde{\gamma}_m}{\tilde{\xi}_m}.$$

Clearly,

$$|\tilde{\gamma}_m| = |s_{m-1}\gamma_{m-1}| = \cdots = |s_1 s_2 \dots s_{m-1}\beta|$$

and therefore,

$$\rho_m^F = \frac{h_{m+1,m}}{|\tilde{\xi}_m|} |s_1 s_2 \dots s_{m-1}\beta|.$$

Using (6.31), observe that $h_{m+1,m}/|\tilde{\xi}_m|$ is the tangent of the angle defining the m-th rotation, and therefore,

$$\rho_m^F = \frac{|s_m|\sqrt{\tilde{\xi}_m^2 + h_{m+1,m}^2}}{|\tilde{\xi}_m|} |s_1 s_2 \dots s_{m-1}\beta|$$

which, by a comparison with (6.48), yields a revealing relation between the residuals of the FOM and GMRES algorithms, namely,

$$\rho_m^F = \frac{1}{c_m}\rho_m^G \ = \rho_m^G \ \sqrt{1 + \frac{h_{m+1,m}^2}{\tilde{\xi}_m^2}}.$$

Another way to prove the above expression is to exploit the relation (6.47); see Exercise 12. These results are summarized in the following proposition (Brown [43]).

PROPOSITION 6.12 *Assume that m steps of the Arnoldi process have been taken and that H_m is nonsingular. Let $\xi \equiv (Q_{m-1}\bar{H}_m)_{mm}$ and $h \equiv h_{m+1,m}$. Then the residual norms produced by the FOM and the GMRES algorithms are related by the equality*

$$\rho_m^F = \frac{1}{c_m}\rho_m^G = \rho_m^G \ \sqrt{1 + \frac{h^2}{\xi^2}}. \tag{6.49}$$

6.5.6 VARIATION 1: RESTARTING

Similar to the FOM algorithm of the previous section, the GMRES algorithm becomes impractical when m is large because of the growth of memory and computational requirements as m increases. These requirements are identical with those of FOM. As with FOM, there are two remedies. One is based on restarting and the other on truncating the Arnoldi orthogonalization. The straightforward restarting option is described here.

ALGORITHM 6.11: Restarted GMRES

1. *Compute $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, and $v_1 = r_0/\beta$*
2. *Generate the Arnoldi basis and the matrix $\bar{H}_m$ using the Arnoldi algorithm*
3. *starting with v_1*
4. *Compute y_m which minimizes $\|\beta e_1 - \bar{H}_m y\|_2$ and $x_m = x_0 + V_m y_m$*
5. *If satisfied then Stop, else set $x_0 := x_m$ and GoTo 1*

Note that the implementation tricks discussed in the previous section can be applied, providing the residual norm at each sub-step j without computing the approximation x_j. This enables the program to exit as soon as this norm is small enough.

A well known difficulty with the restarted GMRES algorithm is that it can *stagnate* when the matrix is not positive definite. The full GMRES algorithm is guaranteed to converge in at most n steps, but this would be impractical if there were many steps required for convergence. Obviously, a preconditioner for the linear system can be used to reduce the number of steps, or a better preconditioner if one is already in use. This issue will be covered later along with preconditioning techniques.

Example 6.2 Table 6.2 shows the results of applying the GMRES algorithm with no preconditioning to three of the test problems described in Section 3.7.

Matrix	Iters	Kflops	Residual	Error
F2DA	95	3841	0.32E-02	0.11E-03
F3D	67	11862	0.37E-03	0.28E-03
ORS	205	9221	0.33E+00	0.68E-04

Table 6.2 *A test run of GMRES with no preconditioning.*

See Example 6.1 for the meaning of the column headers in the table. In this test, the dimension of the Krylov subspace is $m = 10$. Observe that the problem ORS, which could not be solved by FOM(10), is now solved in 205 steps.

6.5.7 VARIATION 2: TRUNCATED GMRES VERSIONS

It is possible to derive an Incomplete version of the GMRES algorithm. This algorithm is called Quasi-GMRES (QGMRES) for the sake of notational uniformity with other algorithms developed in the literature (some of which will be seen in the next chapter). A direct version called DQGMRES using exactly the same arguments as in Section 6.4.2 for DIOM can also be derived. We begin by defining the QGMRES algorithm, in simple terms, by replacing the Arnoldi Algorithm with Algorithm 6.6, the Incomplete Orthogonalization procedure.

ALGORITHM 6.12: Quasi-GMRES

Run a modification of Algorithm 6.9 in which the Arnoldi process in lines 3 to 11 is replaced by the Incomplete Orthogonalization process and all other computations remain unchanged.

Similar to IOM, only the k previous v_i vectors must be kept at any given step. However, this version of GMRES will potentially save computations but not storage. This is because computing the solution by formula (6.23) requires the vectors v_i for $i = 1, \ldots, m$ to be accessed. Fortunately, the approximate solution can be updated in a progressive manner, as in DIOM.

The implementation of this progressive version is quite similar to DIOM. First, note that if $\bar{H}_m$ is banded, as for example, when $m = 5, k = 2$,

$$\bar{H}_5 = \begin{pmatrix} h_{11} & h_{12} & & & \\ h_{21} & h_{22} & h_{23} & & \\ & h_{32} & h_{33} & h_{34} & \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{pmatrix}, \quad g = \begin{pmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{6.50}$$

then the premultiplications by the rotation matrices Ω_i as described in the previous section will only introduce an additional diagonal. For the above case, the resulting least-squares system is $\bar{R}_5 y = \bar{g}_5$ with:

$$\bar{R}_5 = \begin{pmatrix} r_{11} & r_{12} & r_{13} & & \\ & r_{22} & r_{23} & r_{24} & \\ & & r_{33} & r_{34} & r_{35} \\ & & & r_{44} & r_{45} \\ & & & & r_{55} \\ & & & & 0 \end{pmatrix}, \qquad \bar{g}_5 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ . \\ . \\ \gamma_6 \end{pmatrix}. \tag{6.51}$$

The approximate solution is given by

$$x_m = x_0 + V_m R_m^{-1} g_m$$

where R_m and g_m are obtained by removing the last row of $\bar{R}_m$ and $\bar{g}_m$, respectively. Defining P_m as in DIOM,

$$P_m \equiv V_m R_m^{-1}$$

then,

$$x_m = x_0 + P_m g_m.$$

Also note that similarly to DIOM,

$$g_m = \begin{bmatrix} g_{m-1} \\ \gamma_m \end{bmatrix}$$

in which

$$\gamma_m = c_m \gamma_m^{(m-1)},$$

where $\gamma_m^{(m-1)}$ is the last component of the vector $\bar{g}_{m-1}$, i.e., the right-hand side before the m-th rotation is applied. Thus, x_m can be updated at each step, via the relation

$$x_m = x_{m-1} + \gamma_m p_m.$$

ALGORITHM 6.13: DQGMRES

Compute $r_0 = b - Ax_0$, $\gamma_1 := \|r_0\|_2$, *and* $v_1 := r_0/\gamma_1$
For $m = 1, 2, \ldots,$ *until convergence Do:*
  Compute h_{im}, $i = \max\{1, m-k+1\}, \ldots, m$ *and* v_{m+1}
    as in lines 2 to 6 of Algorithm 6.6
  Update the QR factorization of $\bar{H}_m$, *i.e.,*
    Apply Ω_i, $i = m-k, \ldots, m-1$ *to the* m*-th column of* $\bar{H}_m$
    Compute the rotation coefficients c_m, s_m *by (6.31)*
  Apply Ω_m *to* $\bar{H}_m$ *and* $\bar{g}_m$, *i.e., Compute:*
    $\gamma_{m+1} := -s_m \gamma_m$
    $\gamma_m := c_m \gamma_m$
    $h_{mm} := c_m h_{mm} + s_m h_{m+1,m} \quad \left(= \sqrt{h_{m+1,m}^2 + h_{mm}^2}\right)$
  $p_m = \left(v_m - \sum_{i=m-k}^{m-1} h_{im} p_i\right) / h_{mm}$
  $x_m = x_{m-1} + \gamma_m p_m$
  If $|\gamma_{m+1}|$ *is small enough then Stop*
EndDo

The above algorithm does not minimize the norm of the residual vector over $x_0+\mathcal{K}_m$. Rather, it attempts to perform an approximate minimization. The formula (6.35) is still valid since orthogonality is not used to derive it. Therefore,

$$b - Ax_m = V_{m+1}Q_m^T(\gamma_{m+1}e_{m+1}). \tag{6.52}$$

If the v_i's were orthogonal to each other, then this is equivalent to GMRES and the residual norm is minimized over all vectors of the form $x_0 + V_m y$. Since only an incomplete orthogonalization is used then the v_i's are only locally orthogonal and, as a result, only an approximate minimization may be obtained. In addition, (6.36) is no longer valid. This equality had been derived from the above equation by exploiting the orthogonality of the v_i's. It turns out that in practice, $|\gamma_{m+1}|$ remains a reasonably good estimate of the actual residual norm because the v_i's are nearly orthogonal. The following inequality provides an actual upper bound of the residual norm in terms of computable quantities:

$$\|b - Ax_m\| \le \sqrt{m-k+1}\,|\gamma_{m+1}|. \tag{6.53}$$

Here, k is to be replaced by m when $m \le k$. The proof of this inequality is a consequence of (6.52). If the unit vector $q \equiv Q_m^T e_{m+1}$ has components $\eta_1, \eta_2, \ldots, \eta_{m+1}$, then

$$\begin{aligned}
\|b - Ax_m\|_2 &= |\gamma_{m+1}|\ \|V_{m+1}q\|_2 \\
&\le |\gamma_{m+1}| \left(\left\| \sum_{i=1}^{k+1} \eta_i v_i \right\|_2 + \left\| \sum_{i=k+2}^{m+1} \eta_i v_i \right\|_2 \right) \\
&\le |\gamma_{m+1}| \left(\left[\sum_{i=1}^{k+1} \eta_i^2 \right]^{1/2} + \sum_{i=k+2}^{m+1} |\eta_i|\ \|v_i\|_2 \right) \\
&\le |\gamma_{m+1}| \left(\left[\sum_{i=1}^{k+1} \eta_i^2 \right]^{1/2} + \sqrt{m-k} \left[\sum_{i=k+2}^{m+1} \eta_i^2 \right]^{1/2} \right)
\end{aligned}$$

Here, the orthogonality of the first $k+1$ vectors v_i was used and the last term comes from using the Cauchy-Schwartz inequality. The desired inequality follows from using the Cauchy-Schwartz inequality again in the form

$$1\,.\,a + \sqrt{m-k}\,.\,b \le \sqrt{m-k+1}\,\sqrt{a^2+b^2}$$

and from the fact that the vector q is of norm unity. Thus, using $|\gamma_{m+1}|$ as a residual estimate, we would make an error of a factor of $\sqrt{m-k+1}$ at most. In general, this is an overestimate and $|\gamma_{m+1}|$ tends to give an adequate estimate for the residual norm.

It is also interesting to observe that with a little bit more arithmetic, it is possible to actually compute the exact residual vector and norm. This is based on the observation that, according to (6.52), the residual vector is γ_{m+1} times the vector z_{m+1} which is the last column of the matrix

$$Z_{m+1} \equiv V_{m+1}Q_m^T. \tag{6.54}$$

It is an easy exercise to see that this last column can be updated from v_{m+1} and z_m. Indeed,

$$Z_{m+1} = [V_m, v_{m+1}]Q_{m-1}^T\Omega_m$$

$$= [V_m Q_{m-1}^T, v_{m+1}]\Omega_m$$
$$= [Z_m, v_{m+1}]\Omega_m$$

where all the matrices related to the rotation are of size $(m+1) \times (m+1)$. The result is that

$$z_{m+1} = -s_m z_m + c_m v_{m+1}. \tag{6.55}$$

The z_i's can be updated at the cost of one extra vector in memory and $4n$ operations at each step. The norm of z_{m+1} can be computed at the cost of $2n$ operations and the exact residual norm for the current approximate solution can then be obtained by multiplying this norm by $|\gamma_{m+1}|$.

Because this is a little expensive, it may be preferred to just "correct" the estimate provided by γ_{m+1} by exploiting the above recurrence relation,

$$\|z_{m+1}\|_2 \leq |s_m| \|z_m\|_2 + |c_m|.$$

If $\zeta_m \equiv \|z_m\|_2$, then the following recurrence relation holds,

$$\zeta_{m+1} \leq |s_m|\zeta_m + |c_m|. \tag{6.56}$$

The above relation is inexpensive to update, yet provides an upper bound that is sharper than (6.53); see Exercise 20.

An interesting consequence of (6.55) is a relation between two successive residual vectors:

$$\begin{aligned} r_m &= \gamma_{m+1} z_{m+1} \\ &= \gamma_{m+1}[-s_m z_m + c_m v_{m+1}] \\ &= s_m^2 r_{m-1} + c_m \gamma_{m+1} v_{m+1}. \end{aligned} \tag{6.57}$$

This exploits the fact that $\gamma_{m+1} = -s_m \gamma_m$ and $r_j = \gamma_{j+1} z_{j+1}$.

Example 6.3 Table 6.3 shows the results of applying the DQGMRES algorithm with no preconditioning to three of the test problems described in Section 3.7.

Matrix	Iters	Kflops	Residual	Error
F2DA	98	7216	0.36E-02	0.13E-03
F3D	75	22798	0.64E-03	0.32E-03
ORS	300	24138	0.13E+02	0.25E-02

Table 6.3 *A test run of DQGMRES with no preconditioning.*

See Example 6.1 for the meaning of the column headers in the table. In this test the number k of directions in the recurrence is $k = 10$.

It is possible to relate the quasi-minimal residual norm to the actual minimal residual norm provided by GMRES. The following result was proved by Nachtigal (1991) [152] for the QMR algorithm to be seen in the next chapter.

THEOREM 6.1 *Assume that V_{m+1}, the Arnoldi basis associated with DQGMRES, is of full rank. Let r_m^Q and r_m^G be the residual norms obtained after m steps of the DQGMRES and GMRES algorithms, respectively. Then*

$$\|r_m^Q\|_2 \leq \kappa_2(V_{m+1})\|r_m^G\|_2. \tag{6.58}$$

Proof. Consider the subset of $\mathcal{K}_{m+1}$ defined by

$$\mathcal{R} = \{r : r = V_{m+1}t;\ t = \beta e_1 - \bar{H}_m y;\ y \in \mathbb{C}^m\}.$$

Denote by y_m the minimizer of $\|\beta e_1 - \bar{H}_m y\|_2$ over y and $t_m = \beta e_1 - \bar{H}_m y_m$, $r_m = V_{m+1}t_m \equiv r_m^Q$. By assumption, V_{m+1} is of full rank and there is an $(m+1) \times (m+1)$ nonsingular matrix S such that $W_{m+1} = V_{m+1}S$ is unitary. Then, for any member of $\mathcal{R}$,

$$r = W_{m+1}S^{-1}t, \quad t = SW_{m+1}^H r$$

and, in particular,

$$\|r_m\|_2 \leq \|S^{-1}\|_2 \|t_m\|_2. \tag{6.59}$$

Now $\|t_m\|_2$ is the minimum of the 2-norm of $\beta e_1 - \bar{H}_m y$ over all y's and therefore,

$$\begin{aligned} \|t_m\|_2 = \|SW_{m+1}^H r_m\| &\leq \|SW_{m+1}^H r\|_2 \quad \forall r \in \mathcal{R} \\ &\leq \|S\|_2 \|r\|_2 \quad \forall r \in \mathcal{R} \\ &\leq \|S\|_2 \|r^G\|_2. \end{aligned} \tag{6.60}$$

The result follows from (6.59), (6.60), and the fact that $\kappa_2(V_{m+1}) = \kappa_2(S)$. ■

6.6 THE SYMMETRIC LANCZOS ALGORITHM

The symmetric Lanczos algorithm can be viewed as a simplification of Arnoldi's method for the particular case when the matrix is symmetric. When A is symmetric, then the Hessenberg matrix H_m becomes symmetric tridiagonal. This leads to a three-term recurrence in the Arnoldi process and short-term recurrences for solution algorithms such as FOM and GMRES. On the theoretical side, there is also much more to be said on the resulting approximation in the symmetric case.

6.6.1 THE ALGORITHM

To introduce the Lanczos algorithm we begin by making the observation stated in the following theorem.

THEOREM 6.2 *Assume that Arnoldi's method is applied to a real symmetric matrix A. Then the coefficients h_{ij} generated by the algorithm are such that*

$$h_{ij} = 0, \quad \text{for} \quad 1 \le i < j-1, \tag{6.61}$$

$$h_{j,j+1} = h_{j+1,j},\ j = 1, 2, \ldots, m. \tag{6.62}$$

In other words, the matrix H_m obtained from the Arnoldi process is tridiagonal and symmetric.

Proof. The proof is an immediate consequence of the fact that $H_m = V_m^T A V_m$ is a symmetric matrix which is also a Hessenberg matrix by construction. Therefore, H_m must be a symmetric tridiagonal matrix. ∎

The standard notation used to describe the Lanczos algorithm is obtained by setting

$$\alpha_j \equiv h_{jj}, \qquad \beta_j \equiv h_{j-1,j},$$

and if T_m denotes the resulting H_m matrix, it is of the form,

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & . & . & . & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{pmatrix}. \tag{6.63}$$

This leads to the following form of the Modified Gram-Schmidt variant of Arnoldi's method, namely, Algorithm 6.2.

ALGORITHM 6.14: The Lanczos Algorithm

1. *Choose an initial vector* v_1 *of norm unity. Set* $\beta_1 \equiv 0, v_0 \equiv 0$
2. *For* $j = 1, 2, \ldots, m$ *Do:*
3. $\quad w_j := Av_j - \beta_j v_{j-1}$
4. $\quad \alpha_j := (w_j, v_j)$
5. $\quad w_j := w_j - \alpha_j v_j$
6. $\quad \beta_{j+1} := \|w_j\|_2$. *If* $\beta_{j+1} = 0$ *then Stop*
7. $\quad v_{j+1} := w_j / \beta_{j+1}$
8. *EndDo*

It is rather surprising that the above simple algorithm guarantees, at least in exact arithmetic, that the vectors $v_i, i = 1, 2, \ldots,$ are orthogonal. In reality, exact orthogonality of these vectors is only observed at the beginning of the process. At some point the v_i's start losing their global orthogonality rapidly. There has been much research devoted to finding ways to either recover the orthogonality, or to at least diminish its effects by *partial* or *selective* orthogonalization; see Parlett [160].

The major practical differences with Arnoldi's method are that the matrix H_m is tridiagonal and, more importantly, that only three vectors must be saved, unless some form of reorthogonalization is employed.

6.6.2 RELATION WITH ORTHOGONAL POLYNOMIALS

In exact arithmetic, the core of Algorithm 6.14 is a relation of the form

$$\beta_{j+1}v_{j+1} = Av_j - \alpha_j v_j - \beta_{j-1}v_{j-1}.$$

This three-term recurrence relation is reminiscent of the standard three-term recurrence relation of orthogonal polynomials. In fact, there is indeed a strong relationship between the Lanczos algorithm and orthogonal polynomials. To begin, recall that if the grade of v_1 is $\geq m$, then the subspace $\mathcal{K}_m$ is of dimension m and consists of all vectors of the form $q(A)v_1$, where q is a polynomial with $\mathrm{degree}(q) \leq m-1$. In this case there is even an isomorphism between $\mathcal{K}_m$ and $\mathbb{P}_{m-1}$, the space of polynomials of degree $\leq m-1$, which is defined by

$$q \in \mathbb{P}_{m-1} \rightarrow x = q(A)v_1 \in \mathcal{K}_m.$$

Moreover, we can consider that the subspace $\mathbb{P}_{m-1}$ is provided with the inner product

$$< p, q >_{v_1} = (p(A)v_1, q(A)v_1). \tag{6.64}$$

This is indeed a nondegenerate bilinear form under the assumption that m does not exceed μ, the grade of v_1. Now observe that the vectors v_i are of the form

$$v_i = q_{i-1}(A)v_1$$

and the orthogonality of the v_i's translates into the orthogonality of the polynomials with respect to the inner product (6.64). It is known that real orthogonal polynomials satisfy a three-term recurrence. Moreover, the Lanczos procedure is nothing but the Stieltjes algorithm; (see, for example, Gautschi [102]) for computing a sequence of orthogonal polynomials with respect to the inner product (6.64). It is known [180] that the characteristic polynomial of the tridiagonal matrix produced by the Lanczos algorithm minimizes the norm $\|.\|_{v_1}$ over the monic polynomials. The recurrence relation between the characteristic polynomials of tridiagonal matrices also shows that the Lanczos recurrence computes the sequence of vectors $p_{T_m}(A)v_1$, where p_{T_m} is the characteristic polynomial of T_m.

6.7 THE CONJUGATE GRADIENT ALGORITHM

The Conjugate Gradient algorithm is one of the best known iterative techniques for solving sparse Symmetric Positive Definite linear systems. Described in one sentence, the method is a realization of an orthogonal projection technique onto the Krylov subspace $\mathcal{K}_m(r_0, A)$ where r_0 is the initial residual. It is therefore mathematically equivalent to FOM. However, because A is symmetric, some simplifications resulting from the three-term Lanczos recurrence will lead to more elegant algorithms.

6.7.1 DERIVATION AND THEORY

We first derive the analogue of FOM, or Arnoldi's method, for the case when A is symmetric. Given an initial guess x_0 to the linear system $Ax = b$ and the Lanczos vectors $v_i, i = 1, \ldots, m$ together with the tridiagonal matrix T_m, the approximate solution obtained from an orthogonal projection method onto $\mathcal{K}_m$, is given by

$$x_m = x_0 + V_m y_m, \quad y_m = T_m^{-1}(\beta e_1). \tag{6.65}$$

ALGORITHM 6.15: Lanczos Method for Linear Systems

1. *Compute* $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, *and* $v_1 := r_0/\beta$
2. *For* $j = 1, 2, \ldots, m$ *Do:*
3. $\quad w_j = Av_j - \beta_j v_{j-1}$ *(If* $j = 1$ *set* $\beta_1 v_0 \equiv 0$*)*
4. $\quad \alpha_j = (w_j, v_j)$
5. $\quad w_j := w_j - \alpha_j v_j$
6. $\quad \beta_{j+1} = \|w_j\|_2$. *If* $\beta_{j+1} = 0$ *set* $m := j$ *and go to 9*
7. $\quad v_{j+1} = w_j/\beta_{j+1}$
8. *EndDo*
9. *Set* $T_m = \mathrm{tridiag}\ (\beta_i, \alpha_i, \beta_{i+1})$, *and* $V_m = [v_1, \ldots, v_m]$.
10. *Compute* $y_m = T_m^{-1}(\beta e_1)$ *and* $x_m = x_0 + V_m y_m$

Many of the results obtained from Arnoldi's method for linear systems are still valid. For example, the residual vector of the approximate solution x_m is such that

$$b - Ax_m = -\beta_{m+1} e_m^T y_m v_{m+1}. \tag{6.66}$$

The Conjugate Gradient algorithm can be derived from the Lanczos algorithm in the same way DIOM was derived from IOM. In fact, the Conjugate Gradient algorithm can be viewed as a variation of DIOM(2) for the case when A is symmetric. We will follow the same steps as with DIOM, except that the notation will be simplified whenever possible.

First write the LU factorization of T_m as $T_m = L_m U_m$. The matrix L_m is unit lower bidiagonal and U_m is upper bidiagonal. Thus, the factorization of T_m is of the form

$$T_m = \begin{pmatrix} 1 & & & & \\ \lambda_2 & 1 & & & \\ & \lambda_3 & 1 & & \\ & & \lambda_4 & 1 & \\ & & & \lambda_5 & 1 \end{pmatrix} \times \begin{pmatrix} \eta_1 & \beta_2 & & & \\ & \eta_2 & \beta_3 & & \\ & & \eta_3 & \beta_4 & \\ & & & \eta_4 & \beta_5 \\ & & & & \eta_5 \end{pmatrix}.$$

The approximate solution is then given by,

$$x_m = x_0 + V_m U_m^{-1} L_m^{-1}(\beta e_1).$$

Letting

$$P_m \equiv V_m U_m^{-1}$$

and

$$z_m = L_m^{-1} \beta e_1,$$

then,

$$x_m = x_0 + P_m z_m.$$

As for DIOM, p_m, the last column of P_m, can be computed from the previous p_i's and v_m by the simple update

$$p_m = \eta_m^{-1}[v_m - \beta_m p_{m-1}].$$

Note that β_m is a scalar computed from the Lanczos algorithm, while η_m results from the m-th Gaussian elimination step on the tridiagonal matrix, i.e.,

$$\lambda_m = \frac{\beta_m}{\eta_{m-1}}, \tag{6.67}$$

$$\eta_m = \alpha_m - \lambda_m \beta_m. \tag{6.68}$$

In addition, following again what has been shown for DIOM,

$$z_m = \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix},$$

in which $\zeta_m = -\lambda_m \zeta_{m-1}$. As a result, x_m can be updated at each step as

$$x_m = x_{m-1} + \zeta_m p_m$$

where p_m is defined above.

This gives the following algorithm, which we call the direct version of the Lanczos algorithm for linear systems.

ALGORITHM 6.16: D-Lanczos

1. *Compute* $r_0 = b - Ax_0$, $\zeta_1 := \beta := \|r_0\|_2$, *and* $v_1 := r_0/\beta$
2. *Set* $\lambda_1 = \beta_1 = 0$, $p_0 = 0$
3. *For* $m = 1, 2, \ldots$, *until convergence Do:*
4. *Compute* $w := Av_m - \beta_m v_{m-1}$ *and* $\alpha_m = (w, v_m)$
5. *If* $m > 1$ *then compute* $\lambda_m = \frac{\beta_m}{\eta_{m-1}}$ *and* $\zeta_m = -\lambda_m \zeta_{m-1}$
6. $\eta_m = \alpha_m - \lambda_m \beta_m$
7. $p_m = \eta_m^{-1}(v_m - \beta_m p_{m-1})$
8. $x_m = x_{m-1} + \zeta_m p_m$
9. *If* x_m *has converged then Stop*
10. $w := w - \alpha_m v_m$
11. $\beta_{m+1} = \|w\|_2$, $v_{m+1} = w/\beta_{m+1}$
12. *EndDo*

This algorithm computes the solution of the tridiagonal system $T_m y_m = \beta e_1$ progressively by using Gaussian elimination without pivoting. However, as was explained for DIOM, partial pivoting can also be implemented at the cost of having to keep an extra vector. In fact, Gaussian elimination with partial pivoting is sufficient to ensure stability for tridiagonal systems. The more complex LQ factorization has also been exploited in this context and gave rise to an algorithm known as SYMMLQ [159].

The two algorithms 6.15 and 6.16 are mathematically equivalent, that is, they deliver the same approximate solution if they are both executable. However, since Gaussian elimination without pivoting is being used implicitly to solve the tridiagonal system $T_m y = \beta e_1$, the direct version may be more prone to breakdowns.

Observe that the residual vector for this algorithm is in the direction of v_{m+1} due to equation (6.66). Therefore, the residual vectors are orthogonal to each other as in FOM. Likewise, the vectors p_i are A-orthogonal, or *conjugate*. These results are established in the next proposition.

PROPOSITION 6.13 *Let $r_m = b - Ax_m$, $m = 0, 1, \ldots$, be the residual vectors produced by the Lanczos and the D-Lanczos algorithms (6.15 and 6.16) and p_m, $m = 0, 1, \ldots$, the auxiliary vectors produced by Algorithm 6.16. Then,*

1. *Each residual vector r_m is such that $r_m = \sigma_m v_{m+1}$ where σ_m is a certain scalar. As a result, the residual vectors are orthogonal to each other.*
2. *The auxiliary vectors p_i form an A-conjugate set, i.e., $(Ap_i, p_j) = 0$, for $i \neq j$.*

Proof. The first part of the proposition is an immediate consequence of the relation (6.66). For the second part, it must be proved that $P_m^T A P_m$ is a diagonal matrix, where $P_m = V_m U_m^{-1}$. This follows from

$$\begin{aligned} P_m^T A P_m &= U_m^{-T} V_m^T A V_m U_m^{-1} \\ &= U_m^{-T} T_m U_m^{-1} \\ &= U_m^{-T} L_m. \end{aligned}$$

Now observe that $U_m^{-T} L_m$ is a lower triangular which is also symmetric since it is equal to the symmetric matrix $P_m^T A P_m$. Therefore, it must be a diagonal matrix. ■

A consequence of the above proposition is that a version of the algorithm can be derived by imposing the orthogonality and conjugacy conditions. This gives the Conjugate Gradient algorithm which we now derive. The vector x_{j+1} can be expressed as

$$x_{j+1} = x_j + \alpha_j p_j. \tag{6.69}$$

Therefore, the residual vectors must satisfy the recurrence

$$r_{j+1} = r_j - \alpha_j A p_j. \tag{6.70}$$

If the r_j's are to be orthogonal, then it is necessary that $(r_j - \alpha_j A p_j, r_j) = 0$ and as a result

$$\alpha_j = \frac{(r_j, r_j)}{(Ap_j, r_j)}. \tag{6.71}$$

Also, it is known that the next search direction p_{j+1} is a linear combination of r_{j+1} and p_j, and after rescaling the p vectors appropriately, it follows that

$$p_{j+1} = r_{j+1} + \beta_j p_j. \tag{6.72}$$

Thus, a first consequence of the above relation is that

$$(Ap_j, r_j) = (Ap_j, p_j - \beta_{j-1} p_{j-1}) = (Ap_j, p_j)$$

because Ap_j is orthogonal to p_{j-1}. Then, (6.71) becomes $\alpha_j = (r_j, r_j)/(Ap_j, p_j)$. In addition, writing that p_{j+1} as defined by (6.72) is orthogonal to Ap_j yields

$$\beta_j = -\frac{(r_{j+1}, Ap_j)}{(p_j, Ap_j)}.$$

Note that from (6.70)

$$Ap_j = -\frac{1}{\alpha_j}(r_{j+1} - r_j) \tag{6.73}$$

and therefore,

$$\beta_j = \frac{1}{\alpha_j}\frac{(r_{j+1}, (r_{j+1} - r_j))}{(Ap_j, p_j)} = \frac{(r_{j+1}, r_{j+1})}{(r_j, r_j)}.$$

Putting these relations together gives the following algorithm.

ALGORITHM 6.17: Conjugate Gradient

1. *Compute* $r_0 := b - Ax_0$, $p_0 := r_0$.
2. *For* $j = 0, 1, \ldots$, *until convergence Do:*
3. $\quad\alpha_j := (r_j, r_j)/(Ap_j, p_j)$
4. $\quad x_{j+1} := x_j + \alpha_j p_j$
5. $\quad r_{j+1} := r_j - \alpha_j Ap_j$
6. $\quad\beta_j := (r_{j+1}, r_{j+1})/(r_j, r_j)$
7. $\quad p_{j+1} := r_{j+1} + \beta_j p_j$
8. *EndDo*

It is important to note that the scalars α_j, β_j in this algorithm are different from those of the Lanczos algorithm. The vectors p_j are multiples of the p_j's of Algorithm 6.16.

In terms of storage, in addition to the matrix A, four vectors (x, p, Ap, and r) must be saved in Algorithm 6.17, versus five vectors (v_m, v_{m-1}, w, p, and x) for Algorithm 6.16.

6.7.2 ALTERNATIVE FORMULATIONS

Algorithm 6.17 is the best known formulation of the Conjugate Gradient algorithm. There are, however, several alternative formulations. Here, only one such formulation is shown, which can be derived once more from the Lanczos algorithm.

The residual polynomial $r_m(t)$ associated with the m-th CG iterate must satisfy a three-term recurrence, implied by the three-term recurrence of the Lanczos vectors. Indeed, these vectors are just the scaled versions of the residual vectors. Therefore, we must seek a three-term recurrence of the form

$$r_{m+1}(t) = \rho_m(r_m(t) - \gamma_m t r_m(t)) + \delta_m r_{m-1}(t).$$

In addition, the consistency condition $r_m(0) = 1$ must be maintained for each m, leading to the recurrence,

$$r_{m+1}(t) = \rho_m(r_m(t) - \gamma_m t r_m(t)) + (1 - \rho_m) r_{m-1}(t). \tag{6.74}$$

Observe that if $r_m(0) = 1$ and $r_{m-1}(0) = 1$, then $r_{m+1}(0) = 1$, as desired. Translating the above relation into the sequence of residual vectors yields

$$r_{m+1} = \rho_m(r_m - \gamma_m A r_m) + (1 - \rho_m) r_{m-1}. \tag{6.75}$$

Recall that the vectors r_i's are multiples of the Lanczos vectors v_i's. As a result, γ_m should be the inverse of the scalar α_m of the Lanczos algorithm. In terms of the r-vectors this means

$$\gamma_m = \frac{(r_m, r_m)}{(A r_m, r_m)}.$$

Equating the inner products of both sides of (6.75) with r_{m-1}, and using the orthogonality of the r-vectors, gives the following expression for ρ_m, after some algebraic calculations,

$$\rho_m = \left[1 - \frac{\gamma_m}{\gamma_{m-1}} \frac{(r_m, r_m)}{(r_{m-1}, r_{m-1})} \frac{1}{\rho_{m-1}}\right]^{-1}. \tag{6.76}$$

The recurrence relation for the approximate solution vectors can be extracted from the recurrence relation for the residual vectors. This is found by starting from (6.74) and using the relation $r_m(t) = 1 - t s_{m-1}(t)$ between the solution polynomial $s_{m-1}(t)$ and the residual polynomial $r_m(t)$. Thus,

$$\begin{aligned} s_m(t) &= \frac{1 - r_{m+1}(t)}{t} \\ &= \rho_m \left(\frac{1 - r_m(t)}{t} - \gamma_m r_m(t)\right) + (1 - \rho_m) \frac{1 - r_{m-1}(t)}{t} \\ &= \rho_m \left(s_{m-1}(t) - \gamma_m r_m(t)\right) + (1 - \rho_m) s_{m-2}(t). \end{aligned}$$

This gives the recurrence,

$$x_{m+1} = \rho_m(x_m - \gamma_m r_m) + (1 - \rho_m) x_{m-1}. \tag{6.77}$$

All that is left for the recurrence to be determined completely is to define the first two iterates. The initial iterate x_0 is given. The first vector should be of the form

$$x_1 = x_0 - \gamma_0 r_0,$$

to ensure that r_1 is orthogonal to r_0. This means that the two-term recurrence can be started with $\rho_0 = 1$, and by setting $x_{-1} \equiv 0$. Putting these relations and definitions together gives the following algorithm.

ALGORITHM 6.18: CG – Three-Term Recurrence Variant

1. *Compute* $r_0 = b - Ax_0$. *Set* $x_{-1} \equiv 0$ *and* $\rho_0 = 1$.
2. *For* $j = 0, 1, \ldots$, *until convergence Do:*
3. *Compute* Ar_j *and* $\gamma_j = \frac{(r_j, r_j)}{(Ar_j, r_j)}$
4. *If* $(j > 0)$ *compute* $\rho_j = \left[1 - \frac{\gamma_j}{\gamma_{j-1}} \frac{(r_j, r_j)}{(r_{j-1}, r_{j-1})} \frac{1}{\rho_{j-1}}\right]^{-1}$
5. $x_{j+1} = \rho_j (x_j - \gamma_j r_j) + (1 - \rho_j) x_{j-1}$
6. *Compute* $r_{j+1} = \rho_j (r_j - \gamma_j A r_j) + (1 - \rho_j) r_{j-1}$
7. *EndDo*

The residual r_{j+1} could also be computed directly as $r_{j+1} = b - Ax_{j+1}$ in line 6 of the algorithm, but this would require an additional matrix-vector product.

6.7.3 EIGENVALUE ESTIMATES FROM THE CG COEFFICIENTS

Sometimes, it is useful to be able to obtain the tridiagonal matrix T_m related to the underlying Lanczos iteration from the coefficients of the Conjugate Gradient algorithm 6.17. This tridiagonal matrix can provide valuable eigenvalue information on the matrix A. For example, the largest and smallest eigenvalues of the tridiagonal matrix can approximate the smallest and largest eigenvalues of A. This could be used to compute an estimate of the condition number of A which in turn can help provide estimates of the error norm from the residual norm. Since the Greek letters α_i and β_i have been used in both algorithms, notations must be changed. Denote by

$$T_m = \text{tridiag}\,[\eta_j, \delta_j, \eta_{j+1}],$$

the tridiagonal matrix (6.63) associated with the m-th step of the Lanczos algorithm. We must seek expressions of the coefficients η_j, δ_j in terms of the coefficients α_j, β_j, obtained from the CG algorithm. The key information regarding the correspondence between the two pairs of coefficients resides in the correspondence between the vectors generated by the two algorithms. From (6.66) it is known that

$$r_j = \text{scalar} \times v_{j+1}. \tag{6.78}$$

As a result,

$$\delta_{j+1} = \frac{(Av_{j+1}, v_{j+1})}{(v_{j+1}, v_{j+1})} = \frac{(Ar_j, r_j)}{(r_j, r_j)}.$$

The denominator (r_j, r_j) is readily available from the coefficients of the CG algorithm, but the numerator (Ar_j, r_j) is not. The relation (6.72) can be exploited to obtain

$$r_j = p_j - \beta_{j-1} p_{j-1} \tag{6.79}$$

which is then substituted in (Ar_j, r_j) to get

$$(Ar_j, r_j) = (A(p_j - \beta_{j-1} p_{j-1}), p_j - \beta_{j-1} p_{j-1}).$$

Note that the terms $\beta_{j-1} p_{j-1}$ are defined to be zero when $j = 0$. Because the p vectors are A-orthogonal,

$$(Ar_j, r_j) = (Ap_j, p_j) + \beta_{j-1}^2 (Ap_{j-1}, p_{j-1}),$$

from which we finally obtain for $j > 0$,

$$\delta_{j+1} = \frac{(Ap_j, p_j)}{(r_j, r_j)} + \beta_{j-1}^2 \frac{(Ap_{j-1}, p_{j-1})}{(r_j, r_j)} = \frac{1}{\alpha_j} + \frac{\beta_{j-1}}{\alpha_{j-1}}. \tag{6.80}$$

The above expression is only valid for $j > 0$. For $j = 0$, the second term in the right-hand side should be omitted as was observed above. Therefore, the diagonal elements of T_m are given by

$$\delta_{j+1} = \begin{cases} \frac{1}{\alpha_j} & \text{for} \quad j = 0, \\ \frac{1}{\alpha_j} + \frac{\beta_{j-1}}{\alpha_{j-1}} & \text{for} \quad j > 0. \end{cases} \tag{6.81}$$

Now an expression for the co-diagonal elements η_{j+1} is needed. From the definitions in the Lanczos algorithm,

$$\eta_{j+1} = (Av_j, v_{j+1}) = \frac{|(Ar_{j-1}, r_j)|}{\|r_{j-1}\|_2 \|r_j\|_2}.$$

Using (6.79) again and the relation (6.73) as well as orthogonality properties of the CG algorithm, the following sequence of equalities results:

$$\begin{aligned} (Ar_{j-1}, r_j) &= (A(p_{j-1} - \beta_{j-2} p_{j-2}), r_j) \\ &= (Ap_{j-1}, r_j) - \beta_{j-2}(Ap_{j-2}, r_j) \\ &= \frac{-1}{\alpha_{j-1}}(r_j - r_{j-1}, r_j) + \frac{\beta_{j-2}}{\alpha_{j-2}}(r_{j-1} - r_{j-2}, r_j) \\ &= \frac{-1}{\alpha_{j-1}}(r_j, r_j). \end{aligned}$$

Therefore,

$$\eta_{j+1} = \frac{1}{\alpha_{j-1}} \frac{(r_j, r_j)}{\|r_{j-1}\|_2 \|r_j\|_2} = \frac{1}{\alpha_{j-1}} \frac{\|r_j\|_2}{\|r_{j-1}\|_2} = \frac{\sqrt{\beta_{j-1}}}{\alpha_{j-1}}.$$

This finally gives the general form of the m-dimensional Lanczos tridiagonal matrix in terms of the CG coefficients,

$$T_m = \begin{pmatrix} \frac{1}{\alpha_0} & \frac{\sqrt{\beta_0}}{\alpha_0} & & & \\ \frac{\sqrt{\beta_0}}{\alpha_0} & \frac{1}{\alpha_1} + \frac{\beta_0}{\alpha_0} & \frac{\sqrt{\beta_1}}{\alpha_1} & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \frac{\sqrt{\beta_{m-2}}}{\alpha_{m-2}} \\ & & & \frac{\sqrt{\beta_{m-2}}}{\alpha_{m-2}} & \frac{1}{\alpha_{m-1}} + \frac{\beta_{m-2}}{\alpha_{m-2}} \end{pmatrix}. \tag{6.82}$$

6.8 THE CONJUGATE RESIDUAL METHOD

In the previous section we derived the Conjugate Gradient algorithm as a special case of FOM for Symmetric Positive Definite matrices. Similarly, a new algorithm can be derived from GMRES for the particular case where A is Hermitian. In this case, the residual vectors should be A-orthogonal, i.e., conjugate. In addition, the vectors Ap_i's $i = 0, 1, \ldots,$ are orthogonal. When looking for an algorithm with the same structure as CG, but satisfying these conditions, we find the Conjugate Residual algorithm. Notice that the residual vectors are now conjugate to each other, hence, the name of the algorithm.

ALGORITHM 6.19: Conjugate Residual Algorithm

1. *Compute* $r_0 := b - Ax_0$, $p_0 := r_0$
2. *For* $j = 0, 1, \ldots,$ *until convergence Do:*
3. $\quad \alpha_j := (r_j, Ar_j)/(Ap_j, Ap_j)$
4. $\quad x_{j+1} := x_j + \alpha_j p_j$
5. $\quad r_{j+1} := r_j - \alpha_j Ap_j$
6. $\quad \beta_j := (r_{j+1}, Ar_{j+1})/(r_j, Ar_j)$
7. $\quad p_{j+1} := r_{j+1} + \beta_j p_j$
8. $\quad$ *Compute* $Ap_{j+1} = Ar_{j+1} + \beta_j Ap_j$
9. *EndDo*

The last line in the above algorithm computes Ap_{j+1} from Ar_{j+1} without an additional matrix-vector product. Five vectors of storage are needed in addition to the matrix A: x, p, Ap, r, Ar. The algorithm requires one more vector update, i.e., $2n$ more operations than the Conjugate Gradient method and one more vector of storage. Since the two methods exhibit typically a similar convergence behavior, the Conjugate Gradient method is often preferred over the Conjugate Residual algorithm.

6.9 GCR, ORTHOMIN, AND ORTHODIR

All algorithms developed in this chapter are strongly related to, as well as defined by, the choice of a basis of the Krylov subspace. The GMRES algorithm uses an orthogonal basis. In the Conjugate Gradient algorithm, the p's are A-orthogonal, i.e., conjugate. In the Conjugate Residual method just described, the Ap_i's are orthogonal, i.e., the p_i's are A^TA-orthogonal. A number of algorithms can be developed using a basis of this form in the nonsymmetric case as well. The main result that is exploited in all these algorithms is the following lemma.

LEMMA 6.2 *Let $p_0, p_1, \ldots, p_{m-1}$ be a basis of the Krylov subspace $\mathcal{K}_m(A, r_0)$ which is A^TA-orthogonal, i.e., such that*

$$(Ap_i, Ap_j) = 0, \quad \text{for } i \neq j.$$

Then the approximate solution x_m which has the smallest residual norm in the affine space $x_0 + \mathcal{K}_m(A, r_0)$ is given by

$$x_m = x_0 + \sum_{i=0}^{m-1} \frac{(r_0, Ap_i)}{(Ap_i, Ap_i)}\, p_i. \tag{6.83}$$

In addition, x_m can be computed from x_{m-1} by

$$x_m = x_{m-1} + \frac{(r_{m-1}, Ap_{m-1})}{(Ap_{m-1}, Ap_{m-1})} p_{m-1}. \tag{6.84}$$

Proof. The approximate solution and the associated residual vector can be written in the form

$$x_m = x_0 + \sum_{i=0}^{m-1} \alpha_i p_i, \quad r_m = r_0 - \sum_{i=0}^{m-1} \alpha_i A p_i. \tag{6.85}$$

According to the optimality result of Proposition 5.3, in order for $\|r_m\|_2$ to be minimum, the orthogonality relations

$$(r_m, Ap_i) = 0, \quad i = 0, \ldots, m-1$$

must be enforced. Using (6.85) and the orthogonality of the Ap_i's gives immediately,

$$\alpha_i = (r_0, Ap_i)/(Ap_i, Ap_i).$$

This proves the first part of the lemma. Assume now that x_{m-1} is known and that x_m must be determined. According to formula (6.83), $x_m = x_{m-1} + \alpha_{m-1} p_{m-1}$ with α_{m-1} defined above. Note that from the second part of (6.85),

$$r_{m-1} = r_0 - \sum_{j=0}^{m-2} \alpha_j A p_j$$

so that

$$(r_{m-1}, Ap_{m-1}) = (r_0, Ap_{m-1}) - \sum_{j=0}^{m-2} \alpha_j (Ap_j, Ap_{m-1}) = (r_0, Ap_{m-1})$$

exploiting, once more, the orthogonality of the vectors Ap_j, $j = 0, \ldots, m-1$. Thus,

$$\alpha_{m-1} = \frac{(r_{m-1}, Ap_{m-1})}{(Ap_{m-1}, Ap_{m-1})},$$

which proves the expression (6.84). ∎

This lemma opens up many different ways to obtain algorithms that are mathematically equivalent to the full GMRES. The simplest option computes the next basis vector p_{m+1} as a linear combination of the current residual r_m and all previous p_i's. The approximate solution is updated by using (6.84). This is called the Generalized Conjugate Residual (GCR) algorithm.

ALGORITHM 6.20: GCR

1. *Compute* $r_0 = b - Ax_0$. *Set* $p_0 = r_0$.
2. *For* $j = 0, 1, 2, \ldots,$ *until convergence Do:*
3. $\quad \alpha_j = \frac{(r_j, Ap_j)}{(Ap_j, Ap_j)}$
4. $\quad x_{j+1} = x_j + \alpha_j p_j$
5. $\quad r_{j+1} = r_j - \alpha_j A p_j$
6. $\quad$ *Compute* $\beta_{ij} = -\frac{(Ar_{j+1}, Ap_i)}{(Ap_i, Ap_i)}$, *for* $i = 0, 1, \ldots, j$
7. $\quad p_{j+1} = r_{j+1} + \sum_{i=0}^{j} \beta_{ij} p_i$
8. *EndDo*

To compute the scalars β_{ij} in the above algorithm, the vector Ar_j and the previous Ap_i's are required. In order to limit the number of matrix-vector products per step to one, we can proceed as follows. Follow line 5 by a computation of Ar_{j+1} and then compute Ap_{j+1} after line 7 from the relation

$$Ap_{j+1} = Ar_{j+1} + \sum_{i=0}^{j} \beta_{ij} Ap_i.$$

Both the set of p_i's and that of the Ap_i's need to be saved. This doubles the storage requirement compared with GMRES. The number of arithmetic operations per step is also roughly 50% higher than GMRES.

The above version of GCR suffers from the same practical limitations as GMRES and FOM. A restarted version called GCR(m) can be trivially defined. Also, a truncation of the orthogonalization of the Ap_i's, similar to IOM, leads to an algorithm known as ORTHOMIN(k). Specifically, lines 6 and 7 of Algorithm 6.20 are replaced by

6a. Compute $\beta_{ij} = -\frac{(Ar_{j+1}, Ap_i)}{(Ap_i, Ap_i)}$, for $i = j-k+1, \ldots, j$
7a. $p_{j+1} = r_{j+1} + \sum_{i=j-k+1}^{j} \beta_{ij} p_i$.

Another class of algorithms is defined by computing the next basis vector p_{j+1} as

$$p_{j+1} = Ap_j + \sum_{i=0}^{j} \beta_{ij} p_i \tag{6.86}$$

in which, as before, the β_{ij}'s are selected to make the Ap_i's orthogonal, i.e.,

$$\beta_{ij} = -\frac{(A^2 p_j, Ap_i)}{(Ap_i, Ap_i)}.$$

The resulting algorithm is called ORTHODIR [127]. Restarted and truncated versions of ORTHODIR can also be defined.

6.10 THE FABER-MANTEUFFEL THEOREM

As was seen in Section 6.6 when A is symmetric, the Arnoldi algorithm simplifies into the Lanczos procedure, which is defined through a three-term recurrence. As a consequence, FOM is mathematically equivalent to the Conjugate Gradient algorithm in this case. Similarly, the full GMRES algorithm gives rise to the Conjugate Residual algorithm. It is clear that the CG-type algorithms, i.e., algorithms defined through short-term recurrences, are more desirable than those algorithms which require storing entire sequences of vectors as in the GMRES process. These algorithms require less memory and operations per step.

Therefore, the question is: *Is it possible to define algorithms which are based on optimal Krylov subspace projection and which give rise to sequences involving short-term recurrences?* An optimal Krylov subspace projection means a technique which minimizes a certain norm of the error, or residual, on the Krylov subspace. Such methods can be defined from the Arnoldi process. If the Arnoldi process simplifies into an s-term recurrence, i.e., if $h_{ij} = 0$ for $i < j - s + 1$, then the conjugate directions p_i in DIOM are also defined from an s-term recurrence. Similarly, the full GMRES would also simplify into a DQGMRES algorithm involving a short recurrence. Therefore, for all purposes, it is sufficient to analyze what happens to the Arnoldi process (or FOM). We start by generalizing the CG result in a simple way, by considering the DIOM algorithm.

PROPOSITION 6.14 *Let A be a matrix such that*

$$A^T v \in \mathcal{K}_s(A, v)$$

for any vector v. Then, DIOM(s) is mathematically equivalent to the FOM algorithm.

Proof. The assumption is equivalent to the statement that, for any v, there is a polynomial q_v of degree $\leq s - 1$, such that $A^T v = q_v(A)v$. In the Arnoldi process, the scalars h_{ij} are defined by $h_{ij} = (Av_j, v_i)$ and therefore

$$h_{ij} = (Av_j, v_i) = (v_j, A^T v_i) = (v_j, q_{v_j}(A)v_i). \tag{6.87}$$

Since q_{v_j} is a polynomial of degree $\leq s - 1$, the vector $q_{v_j}(A)v_i$ is a linear combination of the vectors $v_i, v_{i+1}, \ldots, v_{i+s-1}$. As a result, if $i < j - s + 1$, then $h_{ij} = 0$. Therefore, DIOM(k) will give the same approximate solution as FOM. ■

In particular, if

$$A^T = q(A)$$

where q is a polynomial of degree $\leq s - 1$, then the result holds. However, the above relation implies that each eigenvector of A is also an eigenvector of A^T. According to Theorem 1.2, this can be true only if A is a normal matrix. As it turns out, the reverse is also true. That is, when A is normal, then there is a polynomial of degree $\leq n - 1$ such that $A^H = q(A)$. Proving this is easy because when $A = Q\Lambda Q^H$ where Q is unitary and Λ diagonal, then $q(A) = Qq(\Lambda)Q^H$. By choosing the polynomial q so that

$$q(\lambda_j) = \bar{\lambda}_j, \ j = 1, \ldots, n$$

we obtain $q(A) = Q\bar{\Lambda}Q^H = A^H$ which is the desired result.

Let $\nu(A)$ be the smallest degree of all polynomials q such that $A^H = q(A)$. Then the following lemma due to Faber and Manteuffel [85] states an interesting relation between s and $\nu(A)$.

LEMMA 6.3 *A nonsingular matrix A is such that*

$$A^H v \in \mathcal{K}_s(A, v)$$

for every vector v if and only if A is normal and $\nu(A) \leq s - 1$.

Proof. The sufficient condition is trivially true. To prove the necessary condition, assume that, for any vector v, $A^H v = q_v(A)v$ where q_v is a polynomial of degree $\leq s - 1$. Then it is easily seen that any eigenvector of A is also an eigenvector of A^H. Therefore, from Theorem 1.2, A is normal. Let μ be the degree of the minimal polynomial for A. Then, since A has μ distinct eigenvalues, there is a polynomial q of degree $\mu - 1$ such that

$q(\lambda_i) = \bar{\lambda}_i$ for $i = 1, \ldots, \mu$. According to the above argument, for this q, it holds $A^H = q(A)$ and therefore $\nu(A) \le \mu - 1$. Now it must be shown that $\mu \le s$. Let w be a (nonzero) vector whose grade is μ. By assumption, $A^H w \in \mathcal{K}_s(A, w)$. On the other hand, we also have $A^H w = q(A)w$. Since the vectors $w, Aw, \ldots, A^{\mu-1}w$ are linearly independent, $\mu-1$ must not exceed $s-1$. Otherwise, two different expressions for $A^H w$ with respect to the basis $w, Aw, \ldots, A^{\mu-1}w$ would result and this would imply that $A^H w = 0$. Since A is nonsingular, then $w = 0$, which is a contradiction. ■

Proposition 6.14 gives a sufficient condition for DIOM(s) to be equivalent to FOM. According to Lemma 6.3, this condition is equivalent to A being normal and $\nu(A) \le s-1$. Now consider the reverse result. Faber and Manteuffel define CG(s) to be the class of all matrices such that *for every* v_1*, it is true that* $(Av_j, v_i) = 0$ *for all* i, j *such that* $i + s \le j \le \mu(v_1) - 1$. The inner product can be different from the canonical Euclidean dot product. With this definition it is possible to show the following theorem [85] which is stated without proof.

THEOREM 6.3 $A \in CG(s)$*, if and only if* $\mu(A) \le s$ *or* A *is normal and* $\nu(A) \le s-1$.

It is interesting to consider the particular case where $\nu(A) \le 1$, which is the case of the Conjugate Gradient method. In fact, it is easy to show that in this case A either has a minimal degree ≤ 1, or is Hermitian, or is of the form

$$A = e^{i\theta}(\rho I + B)$$

where θ and ρ are real and B is skew-Hermitian, i.e., $B^H = -B$. Thus, the cases in which DIOM simplifies into an (optimal) algorithm defined from a three-term recurrence are already known. The first is the Conjugate Gradient method. The second is a version of the CG algorithm for skew-Hermitian matrices which can be derived from the Lanczos algorithm in the same way as CG. This algorithm will be seen in Chapter 9.

6.11 CONVERGENCE ANALYSIS

The convergence behavior of the different algorithms seen in this chapter can be analyzed by exploiting optimality properties whenever such properties exist. This is the case for the Conjugate Gradient and the GMRES algorithms. On the other hand, the non-optimal algorithms such as FOM, IOM, and QGMRES will be harder to analyze.

One of the main tools used in the analysis of these methods is Chebyshev polynomials. These polynomials are useful both in theory, when studying convergence, and in practice, as a means of accelerating single-vector iterations or projection processes. In the following, real and complex Chebyshev polynomials are discussed separately.

6.11.1 REAL CHEBYSHEV POLYNOMIALS

The Chebyshev polynomial of the first kind of degree k is defined by

$$C_k(t) = \cos[k \cos^{-1}(t)] \quad \text{for} \quad -1 \le t \le 1. \tag{6.88}$$

That this is a polynomial with respect to t can be shown easily by induction from the trigonometric relation

$$\cos[(k+1)\theta] + \cos[(k-1)\theta] = 2\cos\theta\cos k\theta,$$

and the fact that $C_1(t) = t, C_0(t) = 1$. Incidentally, this also shows the important three-term recurrence relation

$$C_{k+1}(t) = 2\,t\,C_k(t) - C_{k-1}(t).$$

The definition (6.88) can be extended to cases where $|t| > 1$ with the help of the following formula:

$$C_k(t) = \cosh\,[k \cosh^{-1}(t)], \quad |t| \ge 1. \tag{6.89}$$

This is readily seen by passing to complex variables and using the definition $\cos\theta = (e^{i\theta} + e^{-i\theta})/2$. As a result of (6.89) the following expression can be derived:

$$C_k(t) = \frac{1}{2}\left[\left(t + \sqrt{t^2-1}\right)^k + \left(t + \sqrt{t^2-1}\right)^{-k}\right], \tag{6.90}$$

which is valid for $|t| \ge 1$ but can also be extended to the case of $|t| < 1$. The following approximation, valid for large values of k, will be sometimes used:

$$C_k(t) \gtrapprox \frac{1}{2}\left(t + \sqrt{t^2-1}\right)^k \quad \text{for} \quad |t| \ge 1. \tag{6.91}$$

In what follows we denote by $\mathbb{P}_k$ the set of all polynomials of degree k. An important result from approximation theory is the following theorem.

THEOREM 6.4 *Let $[\alpha, \beta]$ be a non-empty interval in $\mathbb{R}$ and let γ be any real scalar outside the interval $[\alpha, \beta]$. Then the minimum*

$$\min_{p \in \mathbb{P}_k, p(\gamma)=1} \; \max_{t \in [\alpha,\beta]} |p(t)|$$

is reached by the polynomial

$$\hat{C}_k(t) \equiv \frac{C_k\left(1 + 2\frac{t-\beta}{\beta-\alpha}\right)}{C_k\left(1 + 2\frac{\gamma-\beta}{\beta-\alpha}\right)}. \tag{6.92}$$

For a proof, see Cheney [52]. The maximum of C_k for t in $[-1, 1]$ is 1 and a corollary of the above result is

$$\min_{p \in \mathbb{P}_k,\; p(\gamma)=1} \; \max_{t \in [\alpha,\beta]} |p(t)| = \frac{1}{|C_k(1 + 2\frac{\gamma-\beta}{\beta-\alpha})|} = \frac{1}{|C_k(2\frac{\gamma-\mu}{\beta-\alpha})|}$$

in which $\mu \equiv (\alpha + \beta)/2$ is the middle of the interval. The absolute values in the denominator are needed only when γ is to the left of the interval, i.e., when $\gamma \leq \alpha$. For this case, it may be more convenient to express the best polynomial as

$$\hat{C}_k(t) \equiv \frac{C_k\left(1 + 2\frac{\alpha - t}{\beta - \alpha}\right)}{C_k\left(1 + 2\frac{\alpha - \gamma}{\beta - \alpha}\right)}.$$

which is obtained by exchanging the roles of α and β in (6.92).

6.11.2 COMPLEX CHEBYSHEV POLYNOMIALS

The standard definition of real Chebyshev polynomials given by equation (6.88) extends without difficulty to complex variables. First, as was seen before, when t is real and $|t| > 1$, the alternative definition, $C_k(t) = \cosh[k \cosh^{-1}(t)]$, can be used. These definitions can be unified by switching to complex variables and writing

$$C_k(z) = \cosh(k\zeta), \quad \text{where} \quad \cosh(\zeta) = z\,.$$

Defining the variable $w = e^{\zeta}$, the above formula is equivalent to

$$C_k(z) = \frac{1}{2}[w^k + w^{-k}] \quad \text{where} \quad z = \frac{1}{2}[w + w^{-1}]. \tag{6.93}$$

The above definition for Chebyshev polynomials will be used in $\mathbb{C}$. Note that the equation $\frac{1}{2}(w + w^{-1}) = z$ has two solutions w which are inverse of each other. As a result, the value of $C_k(z)$ does not depend on which of these solutions is chosen. It can be verified directly that the C_k's defined by the above equations are indeed polynomials in the z variable and that they satisfy the three-term recurrence

$$\begin{aligned} &C_{k+1}(z) = 2\,zC_k(z) - C_{k-1}(z), \\ &C_0(z) \equiv 1, \qquad C_1(z) \equiv z. \end{aligned} \tag{6.94}$$

As is now explained, Chebyshev polynomials are intimately related to ellipses in the complex plane. Let C_ρ be the circle of radius ρ centered at the origin. Then the so-called Joukowski mapping

$$J(w) = \frac{1}{2}[w + w^{-1}]$$

transforms C_ρ into an ellipse of center the origin, foci $-1, 1$, major semi-axis $\frac{1}{2}[\rho + \rho^{-1}]$ and minor semi-axis $\frac{1}{2}|\rho - \rho^{-1}|$. This is illustrated in Figure 6.2.

There are two circles which have the same image by the mapping $J(w)$, one with the radius ρ and the other with the radius ρ^{-1}. So it is sufficient to consider only those circles with radius $\rho \geq 1$. Note that the case $\rho = 1$ is a degenerate case in which the ellipse $E(0, 1, -1)$ reduces to the interval $[-1, 1]$ traveled through twice.

An important question is whether or not a generalization of the min-max result of Theorem 6.4 holds for the complex case. Here, the maximum of $|p(z)|$ is taken over the ellipse boundary and γ is some point not enclosed by the ellipse. The answer to the question is no; Chebyshev polynomials are only optimal in some cases. However, Chebyshev polynomials are asymptotically optimal, which is all that is needed in practice.

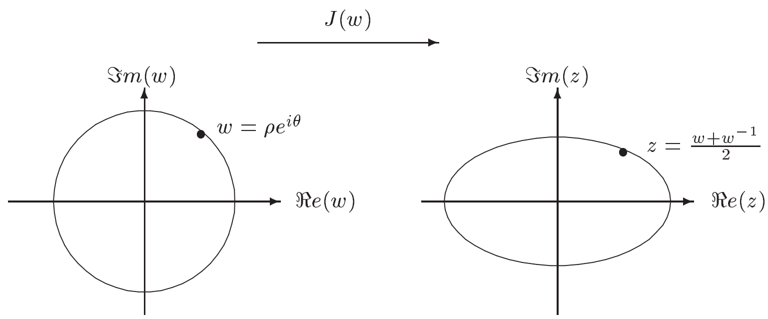

Figure 6.2 *The Joukowski mapping transforms a circle into an ellipse in the complex plane.*

To prove the asymptotic optimality, we begin with a lemma due to Zarantonello, which deals with the particular case where the ellipse reduces to a circle. This particular case is important in itself.

LEMMA 6.4 Zarantonello Let $C(0, \rho)$ be a circle of center the origin and radius ρ and let γ be a point of $\mathbb{C}$ not enclosed by $C(0, \rho)$. Then

$$\min_{p \in \mathbb{P}_k, \; p(\gamma)=1} \quad \max_{z \,\in\, C(0,\rho)} |p(z)| \;=\; \left(\frac{\rho}{|\gamma|}\right)^k, \tag{6.95}$$

the minimum being achieved for the polynomial $(z/\gamma)^k$.

Proof. See reference [168] for a proof. ∎

Note that by changing variables, shifting, and rescaling the polynomial, then for any circle centered at c and for any scalar γ such that $|\gamma| > \rho$, the following min-max result holds:

$$\min_{p \in \mathbb{P}_k \;\; p(\gamma)=1} \quad \max_{z \,\in\, C(c,\rho)} |p(z)| \;=\; \left(\frac{\rho}{|\gamma - c|}\right)^k.$$

Now consider the case of an ellipse centered at the origin, with foci $1, -1$ and semi-major axis a, which can be considered as mapped by J from the circle $C(0, \rho)$, with the convention that $\rho \geq 1$. Denote by E_ρ such an ellipse.

THEOREM 6.5 *Consider the ellipse E_ρ mapped from $C(0, \rho)$ by the mapping J and let γ be any point in the complex plane not enclosed by it. Then*

$$\frac{\rho^k}{|w_\gamma|^k} \leq \min_{p \in \mathbb{P}_k \; p(\gamma)=1} \quad \max_{z \,\in\, E_\rho} |p(z)| \leq \frac{\rho^k + \rho^{-k}}{|w_\gamma^k + w_\gamma^{-k}|} \tag{6.96}$$

in which w_γ is the dominant root of the equation $J(w) = \gamma$.

Proof. We start by showing the second inequality. Any polynomial p of degree k satisfying the constraint $p(\gamma) = 1$ can be written as

$$p(z) = \frac{\sum_{j=0}^{k} \xi_j z^j}{\sum_{j=0}^{k} \xi_j \gamma^j}.$$

A point z on the ellipse is transformed by J from a certain w in $C(0, \rho)$. Similarly, let w_γ be one of the two inverse transforms of γ by the mapping, namely, the one with largest modulus. Then, p can be rewritten as

$$p(z) = \frac{\sum_{j=0}^{k} \xi_j (w^j + w^{-j})}{\sum_{j=0}^{k} \xi_j (w_\gamma^j + w_\gamma^{-j})}. \tag{6.97}$$

Consider the particular polynomial obtained by setting $\xi_k = 1$ and $\xi_j = 0$ for $j \neq k$,

$$p^*(z) = \frac{w^k + w^{-k}}{w_\gamma^k + w_\gamma^{-k}}$$

which is a scaled Chebyshev polynomial of the first kind of degree k in the variable z. It is apparent that the maximum modulus of this polynomial is reached in particular when $w = \rho e^{i\theta}$ is real, i.e., when $w = \rho$. Thus,

$$\max_{z \in E_\rho} |p^*(z)| = \frac{\rho^k + \rho^{-k}}{|w_\gamma^k + w_\gamma^{-k}|}$$

which proves the second inequality.

To prove the left inequality, we rewrite (6.97) as

$$p(z) = \left(\frac{w^{-k}}{w_\gamma^{-k}}\right) \frac{\sum_{j=0}^{k} \xi_j (w^{k+j} + w^{k-j})}{\sum_{j=0}^{k} \xi_j (w_\gamma^{k+j} + w_\gamma^{k-j})}$$

and take the modulus of $p(z)$,

$$|p(z)| = \frac{\rho^{-k}}{|w_\gamma|^{-k}} \left| \frac{\sum_{j=0}^{k} \xi_j (w^{k+j} + w^{k-j})}{\sum_{j=0}^{k} \xi_j (w_\gamma^{k+j} + w_\gamma^{k-j})} \right|.$$

The polynomial in w of degree $2k$ inside the large modulus bars in the right-hand side is such that its value at w_γ is one. By Lemma 6.4, the modulus of this polynomial over the circle $C(0, \rho)$ is not less than $(\rho/|w_\gamma|)^{2k}$, i.e., for any polynomial, satisfying the constraint $p(\gamma) = 1$,

$$\max_{z \in E_\rho} |p(z)| \geq \frac{\rho^{-k}}{|w_\gamma|^{-k}} \frac{\rho^{2k}}{|w_\gamma|^{2k}} = \frac{\rho^k}{|w_\gamma|^k}.$$

This proves that the minimum over all such polynomials of the maximum modulus on the ellipse E_ρ is $\geq (\rho/|w_\gamma|)^k$. ∎

The difference between the left and right bounds in (6.96) tends to zero as k increases to infinity. Thus, the important point made by the theorem is that for large k, the Chebyshev polynomial

$$p^*(z) = \frac{w^k + w^{-k}}{w_\gamma^k + w_\gamma^{-k}}, \quad \text{where} \quad z = \frac{w + w^{-1}}{2}$$

is close to the optimal polynomial. More specifically, Chebyshev polynomials are *asymptotically* optimal.

For a more general ellipse $E(c, d, a)$ centered at c, and with focal distance d and semi-major axis a, a simple change of variables shows that the near-best polynomial is given by

$$\hat{C}_k(z) = \frac{C_k\left(\frac{c-z}{d}\right)}{C_k\left(\frac{c-\gamma}{d}\right)}. \tag{6.98}$$

In addition, by examining the expression $(w^k + w^{-k})/2$ for $w = \rho e^{i\theta}$ it is easily seen that the maximum modulus of $\hat{C}_k(z)$, i.e., the infinity norm of this polynomial over the ellipse, is reached at the point $c + a$ located on the real axis. From this we get,

$$\max_{z \in E(c,d,a)} |\hat{C}_k(z)| = \frac{C_k\left(\frac{a}{d}\right)}{|C_k\left(\frac{c-\gamma}{d}\right)|}$$

Here, we point out that d and a both can be purely imaginary [for an example, see part (B) of Figure 6.3]. In this case a/d is real and the numerator in the above expression is always real. Using the definition for C_k we obtain the following useful expression and approximation:

$$\frac{C_k\left(\frac{a}{d}\right)}{C_k\left(\frac{c-\gamma}{d}\right)} = \frac{\left(\frac{a}{d} + \sqrt{\left(\frac{a}{d}\right)^2 - 1}\right)^k + \left(\frac{a}{d} + \sqrt{\left(\frac{a}{d}\right)^2 - 1}\right)^{-k}}{\left(\frac{c-\gamma}{d} + \sqrt{\left(\frac{c-\gamma}{d}\right)^2 - 1}\right)^k + \left(\frac{c-\gamma}{d} + \sqrt{\left(\frac{c-\gamma}{d}\right)^2 - 1}\right)^{-k}} \tag{6.99}$$

$$\approx \left(\frac{a + \sqrt{a^2 - d^2}}{c - \gamma + \sqrt{(c-\gamma)^2 - d^2}}\right)^k \tag{6.100}$$

Finally, we note that an alternative and more detailed result has been proven by Fischer and Freund in [89].

6.11.3 CONVERGENCE OF THE CG ALGORITHM

As usual, $\|x\|_A$ denotes the norm defined by

$$\|x\|_A = (Ax, x)^{1/2}.$$

The following lemma characterizes the approximation obtained from the Conjugate Gradient algorithm.

LEMMA 6.5 *Let x_m be the approximate solution obtained from the m-th step of the CG algorithm, and let $d_m = x_* - x_m$ where x_* is the exact solution. Then, x_m is of the form*

$$x_m = x_0 + q_m(A) r_0$$

where q_m is a polynomial of degree $m - 1$ such that

$$\|(I - Aq_m(A))d_0\|_A = \min_{q \in \mathbb{P}_{m-1}} \|(I - Aq(A))d_0\|_A.$$

Proof. This is a consequence of the fact that x_m minimizes the A-norm of the error in the affine subspace $x_0 + \mathcal{K}_m$, a result of Proposition 5.2, and the fact that $\mathcal{K}_m$ is the set of all vectors of the form $x_0 + q(A)r_0$, where q is a polynomial of degree $\leq m-1$. ■

From this, the following theorem can be proved.

THEOREM 6.6 *Let x_m be the approximate solution obtained at the m-th step of the Conjugate Gradient algorithm, and x_* the exact solution and define*

$$\eta = \frac{\lambda_{min}}{\lambda_{max} - \lambda_{min}}. \tag{6.101}$$

Then,

$$\|x_* - x_m\|_A \leq \frac{\|x_* - x_0\|_A}{C_m(1+2\eta)}, \tag{6.102}$$

in which C_m is the Chebyshev polynomial of degree m of the first kind.

Proof. From the previous lemma, it is known that $\|x_* - x_m\|_A$ minimizes A-norm of the error over polynomials $r(t)$ which take the value one at 0, i.e.,

$$\|x_* - x_m\|_A = \min_{r \in \mathbb{P}_m,\ r(0)=1} \|r(A)d_0\|_A.$$

If $\lambda_i, i = 1, \ldots, n$ are the eigenvalues of A, and $\xi_i, i = 1, \ldots, n$ the components of the initial error d_0 in the eigenbasis, then

$$\begin{aligned} \|r(A)d_0\|_A^2 = \sum_{i=1}^{n} \lambda_i r(\lambda_i)^2 (\xi_i)^2 &\leq \max_i (r(\lambda_i))^2 \|d_0\|_A^2 \\ &\leq \max_{\lambda \in [\lambda_{min}, \lambda_{max}]} (r(\lambda))^2 \|d_0\|_A^2. \end{aligned}$$

Therefore,

$$\|x_* - x_m\|_A \leq \min_{r \in \mathbb{P}_m,\ r(0)=1} \ \max_{\lambda \in [\lambda_{min}, \lambda_{max}]} |r(\lambda)| \|d_0\|_A.$$

The result follows immediately by using the well known result of Theorem 6.4 from approximation theory. This gives the polynomial r which minimizes the right-hand side. ■

A slightly different formulation of inequality (6.102) can be derived. Using the relation,

$$\begin{aligned} C_m(t) &= \frac{1}{2}\left[\left(t + \sqrt{t^2-1}\right)^m + \left(t + \sqrt{t^2-1}\right)^{-m}\right] \\ &\geq \frac{1}{2}\left(t + \sqrt{t^2-1}\right)^m \end{aligned}$$

then

$$\begin{aligned} C_m(1+2\eta) &\geq \frac{1}{2}\left(1 + 2\eta + \sqrt{(1+2\eta)^2 - 1}\right)^m \\ &\geq \frac{1}{2}\left(1 + 2\eta + 2\sqrt{\eta(\eta+1)}\right)^m. \end{aligned}$$

Now notice that

$$1 + 2\eta + 2\sqrt{\eta(\eta+1)} = \left(\sqrt{\eta} + \sqrt{\eta+1}\right)^2 \tag{6.103}$$

$$= \frac{\left(\sqrt{\lambda_{min}} + \sqrt{\lambda_{max}}\right)^2}{\lambda_{max} - \lambda_{min}} \tag{6.104}$$

$$= \frac{\sqrt{\lambda_{max}} + \sqrt{\lambda_{min}}}{\sqrt{\lambda_{max}} - \sqrt{\lambda_{min}}} \tag{6.105}$$

$$= \frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1} \tag{6.106}$$

in which κ is the spectral condition number $\kappa = \lambda_{max}/\lambda_{min}$.

Substituting this in (6.102) yields,

$$\|x_* - x_m\|_A \leq 2 \left[\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right]^m \|x_* - x_0\|_A. \tag{6.107}$$

This bound is similar to that of the steepest descent algorithm except that the condition number of A is now replaced by its square root.

6.11.4 CONVERGENCE OF GMRES

We begin by stating a *global* convergence result. Recall that a matrix A is called positive definite if its symmetric part $(A + A^T)/2$ is Symmetric Positive Definite. This is equivalent to the property that $(Ax, x) > 0$ for all nonzero real vectors x.

THEOREM 6.7 *If A is a positive definite matrix, then GMRES(m) converges for any $m \geq 1$.*

Proof. This is true because the subspace $\mathcal{K}_m$ contains the initial residual vector at each restart. Since the algorithm minimizes the residual norm in the subspace $\mathcal{K}_m$, at each outer iteration, the residual norm will be reduced by as much as the result of one step of the Minimal Residual method seen in the previous chapter. Therefore, the inequality (5.18) is satisfied by residual vectors produced after each outer iteration and the method converges. ■

Next we wish to establish a result similar to the one for the Conjugate Gradient method, which would provide an upper bound on the convergence rate of the GMRES iterates. We begin with a lemma similar to Lemma 6.5.

LEMMA 6.6 *Let x_m be the approximate solution obtained from the m-th step of the GMRES algorithm, and let $r_m = b - Ax_m$. Then, x_m is of the form*

$$x_m = x_0 + q_m(A)r_0$$

and

$$\|r_m\|_2 = \|(I - Aq_m(A))r_0\|_2 = \min_{q \in \mathbb{P}_{m-1}} \|(I - Aq(A))r_0\|_2.$$

Proof. This is true because x_m minimizes the 2-norm of the residual in the affine subspace $x_0 + \mathcal{K}_m$, a result of Proposition 5.3, and the fact that $\mathcal{K}_m$ is the set of all vectors of the form $x_0 + q(A)r_0$, where q is a polynomial of degree $\leq m-1$. ■

Unfortunately, it not possible to prove a simple result such as Theorem 6.6 unless A is normal.

PROPOSITION 6.15 *Assume that A is a diagonalizable matrix and let $A = X\Lambda X^{-1}$ where $\Lambda = \mathrm{diag}\ \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ is the diagonal matrix of eigenvalues. Define,*

$$\epsilon^{(m)} = \min_{p \in \mathbb{P}_m, p(0)=1} \max_{i=1,\ldots,n} |p(\lambda_i)|.$$

Then, the residual norm achieved by the m-th step of GMRES satisfies the inequality

$$\|r_m\|_2 \leq \kappa_2(X)\epsilon^{(m)}\|r_0\|_2.$$

where $\kappa_2(X) \equiv \|X\|_2\|X^{-1}\|_2$.

Proof. Let p be any polynomial of degree $\leq m$ which satisfies the constraint $p(0) = 1$, and x the vector in $\mathcal{K}_m$ to which it is associated via $b - Ax = p(A)r_0$. Then,

$$\|b - Ax\|_2 = \|Xp(\Lambda)X^{-1}r_0\|_2 \leq \|X\|_2\|X^{-1}\|_2\|r_0\|_2\|p(\Lambda)\|_2$$

Since Λ is diagonal, observe that

$$\|p(\Lambda)\|_2 = \max_{i=1,\ldots,n} |p(\lambda_i)|.$$

Since x_m minimizes the residual norm over $x_0 + \mathcal{K}_m$, then for any consistent polynomial p,

$$\|b - Ax_m\| \leq \|b - Ax\|_2 \leq \|X\|_2\|X^{-1}\|_2\|r_0\|_2 \max_{i=1,\ldots,n} |p(\lambda_i)|.$$

Now the polynomial p which minimizes the right-hand side in the above inequality can be used. This yields the desired result,

$$\|b - Ax_m\| \leq \|b - Ax\|_2 \leq \|X\|_2\|X^{-1}\|_2\|r_0\|_2\epsilon^{(m)}.$$

■

The results of Section 6.11.2 on near-optimal Chebyshev polynomials in the complex plane can now be used to obtain an upper bound for $\epsilon^{(m)}$. Assume that the spectrum of A in contained in an ellipse $E(c, d, a)$ with center c, focal distance d, and major semi axis a. In addition it is required that the origin lie outside this ellipse. The two possible cases are shown in Figure 6.3. Case (B) corresponds to the situation when d is purely imaginary, i.e., the major semi-axis is aligned with the imaginary axis.

COROLLARY 6.1 *Let A be a diagonalizable matrix, i.e, let $A = X\Lambda X^{-1}$ where $\Lambda = \mathrm{diag}\ \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ is the diagonal matrix of eigenvalues. Assume that all the eigenvalues of A are located in the ellipse $E(c, d, a)$ which excludes the origin. Then, the* *residual norm achieved at the m-th step of GMRES satisfies the inequality,*

$$\|r_m\|_2 \le \kappa_2(X)\, \frac{C_m\left(\frac{a}{d}\right)}{\left|C_m\left(\frac{c}{d}\right)\right|}\, \|r_0\|_2.$$

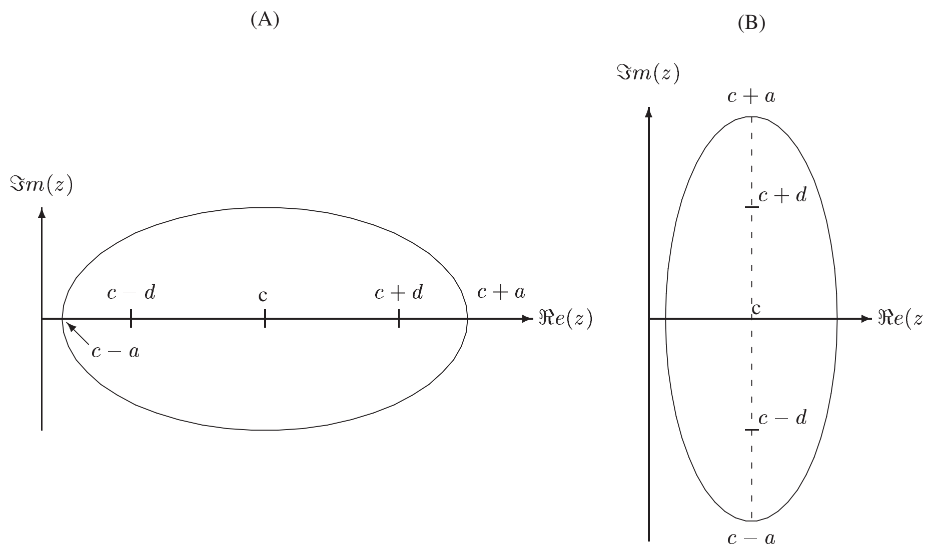

Figure 6.3 *Ellipses containing the spectrum of A. Case (A): real d; case (B): purely imaginary d.*

Proof. All that is needed is an upper bound for the scalar $\epsilon^{(m)}$ under the assumptions. By definition,

$$\begin{aligned}\epsilon^{(m)} &= \min_{p\in\mathbb{P}_m, p(0)=1} \ \max_{i=1,\ldots,n} |p(\lambda_i)| \\ &\le \min_{p\in\mathbb{P}_m, p(0)=1} \ \max_{\lambda\, \in\, E(c,d,a)} |p(\lambda)|.\end{aligned}$$

The second inequality is due to the fact that the maximum modulus of a complex analytical function is reached on the boundary of the domain. We can now use as a trial polynomial $\hat{C}_m$ defined by (6.98), with $\gamma = 0$:

$$\begin{aligned}\epsilon^{(m)} &\le \min_{p\in\mathbb{P}_m, p(0)=1} \ \max_{\lambda\, \in\, E(c,d,a)} |p(\lambda)| \\ &\le \max_{\lambda\, \in\, E(c,d,a)} |\hat{C}_m(\lambda)| = \frac{C_m\left(\frac{a}{d}\right)}{\left|C_m\left(\frac{c}{d}\right)\right|}.\end{aligned}$$

This completes the proof. ■

An explicit expression for the coefficient $C_m\left(\frac{a}{d}\right) \,/\, C_m\left(\frac{c}{d}\right)$ and an approximation are readily obtained from (6.99-6.100) by taking $\gamma = 0$:

$$\frac{C_m\left(\frac{a}{d}\right)}{C_m\left(\frac{c}{d}\right)} = \frac{\left(\frac{a}{d} + \sqrt{\left(\frac{a}{d}\right)^2 - 1}\right)^k + \left(\frac{a}{d} + \sqrt{\left(\frac{a}{d}\right)^2 - 1}\right)^{-k}}{\left(\frac{c}{d} + \sqrt{\left(\frac{c}{d}\right)^2 - 1}\right)^k + \left(\frac{c}{d} + \sqrt{\left(\frac{c}{d}\right)^2 - 1}\right)^{-k}}$$
$$\approx \left(\frac{a + \sqrt{a^2 - d^2}}{c + \sqrt{c^2 - d^2}}\right)^k .$$

Since the condition number $\kappa_2(X)$ of the matrix of eigenvectors X is typically not known and can be very large, results of the nature of the corollary are of limited practical interest. They can be useful only when it is known that the matrix is nearly normal, in which case, $\kappa_2(X) \approx 1$.

6.12 BLOCK KRYLOV METHODS

In many circumstances, it is desirable to work with a block of vectors instead of a single vector. For example, out-of-core finite-element codes are more efficient when they are programmed to exploit the presence of a block of the matrix A in fast memory, as much as possible. This can be achieved by using block generalizations of Krylov subspace methods, for which A always operates on a group of vectors instead of a single vector. We begin by describing a block version of the Arnoldi algorithm.

ALGORITHM 6.21: Block Arnoldi

1. *Choose a unitary matrix* V_1 *of dimension* $n \times p$.
2. *For* $j = 1, 2, \ldots, m$ *Do:*
3. *Compute* $H_{ij} = V_i^T A V_j \quad i = 1, 2, \ldots, j$
4. *Compute* $W_j = AV_j - \sum_{i=1}^{j} V_i H_{ij}$
5. *Compute the Q-R factorization of* W_j: $W_j = V_{j+1} H_{j+1,j}$
6. *EndDo*

The above algorithm is a straightforward block analogue of Algorithm 6.1. By construction, the blocks generated by the algorithm are orthogonal blocks that are also orthogonal to each other. In the following we denote by I_k the $k \times k$ identity matrix and use the following notation:

$$U_m = [V_1, V_2, \ldots, V_m],$$
$$H_m = (H_{ij})_{1 \le i,j \le m}, \quad H_{ij} \equiv 0, \quad \text{for} \quad i > j+1,$$
$$E_m = \text{matrix of the last } p \text{ columns of } I_n.$$

Then, the following analogue of the relation (6.4) is easily proved:

$$AU_m = U_m H_m + V_{m+1} H_{m+1,m} E_m^T. \tag{6.108}$$

Here, the matrix H_m is no longer Hessenberg, but band-Hessenberg, meaning that it has p subdiagonals instead of only one. Note that the dimension of the subspace in which the solution is sought is not m but $m.p$.

A second version of the algorithm uses a modified block Gram-Schmidt procedure instead of the simple Gram-Schmidt procedure used above. This leads to a block generalization of Algorithm 6.2, the Modified Gram-Schmidt version of Arnoldi's method.

ALGORITHM 6.22: Block Arnoldi with Block MGS

1. *Choose a unitary matrix* V_1 *of size* $n \times p$
2. *For* $j = 1, 2, \ldots, m$ *Do:*
3. *Compute* $W_j := AV_j$
4. *For* $i = 1, 2, \ldots, j$ *do:*
5. $H_{ij} := V_i^T W_j$
6. $W_j := W_j - V_i H_{ij}$
7. *EndDo*
8. *Compute the Q-R decomposition* $W_j = V_{j+1} H_{j+1,j}$
9. *EndDo*

Again, in practice the above algorithm is more viable than its predecessor. Finally, a third version, developed by A. Ruhe [170] for the symmetric case (block Lanczos), yields a variant that is quite similar to the original Arnoldi algorithm. Assume that the initial block of p orthonormal vectors, $v_1, \ldots, v_p$ is available. The first step of the algorithm is to multiply v_1 by A and orthonormalize the resulting vector w against $v_1, \ldots, v_p$. The resulting vector is defined to be v_{p+1}. In the second step it is v_2 that is multiplied by A and orthonormalized against all available v_i's. Thus, the algorithm works similarly to Algorithm 6.2 except for a delay in the vector that is multiplied by A at each step.

ALGORITHM 6.23: Block Arnoldi–Ruhe's variant

1. *Choose* p *initial orthonormal vectors* $\{v_i\}_{i=1,\ldots,p}$.
2. *For* $j = p, p+1, \ldots, m$ *Do:*
3. *Set* $k := j - p + 1$;
4. *Compute* $w := Av_k$;
5. *For* $i = 1, 2, \ldots, j$ *Do:*
6. $h_{i,k} := (w, v_i)$
7. $w := w - h_{i,k} v_i$
8. *EndDo*
9. *Compute* $h_{j+1,k} := \|w\|_2$ *and* $v_{j+1} := w/h_{j+1,k}$.
10. *EndDo*

Observe that the particular case $p = 1$ coincides with the usual Arnoldi process. Also, the dimension m of the subspace of approximants, is no longer restricted to being a multiple of the block-size p as in the previous algorithms. The mathematical equivalence of Algorithms 6.22 and 6.23 when m is a multiple of p is straightforward to show. The advantage of the above formulation is its simplicity. A slight disadvantage is that it gives up some potential parallelism. In the original version, the columns of the matrix AV_j can be computed in parallel whereas in the new algorithm, they are computed in sequence. This can be remedied, however, by performing p matrix-by-vector products every p steps.

At the end of the loop consisting of lines 5 through 8 of Algorithm 6.23, the vector w satisfies the relation

$$w = Av_k - \sum_{i=1}^{j} h_{ik} v_i,$$

where k and j are related by $k = j - p + 1$. Line 9 gives $w = h_{j+1,k} v_{j+1}$ which results in

$$Av_k = \sum_{i=1}^{k+p} h_{ik} v_i.$$

As a consequence, the analogue of the relation (6.5) for Algorithm 6.23 is

$$AV_m = V_{m+p} \bar{H}_m. \tag{6.109}$$

As before, for any j the matrix V_j represents the $n \times j$ matrix with columns $v_1, \ldots v_j$. The matrix $\bar{H}_m$ is now of size $(m+p) \times m$.

Now the block generalizations of FOM and GMRES can be defined in a straightforward way. These block algorithms can solve linear systems with multiple right-hand sides,

$$Ax^{(i)} \;=\; b^{(i)}, \quad i = 1, \cdots, p, \tag{6.110}$$

or, in matrix form

$$AX = B, \tag{6.111}$$

where the columns of the $n \times p$ matrices B and X are the $b^{(i)}$'s and $x^{(i)}$'s, respectively. Given an initial block of initial guesses $x_0^{(i)}$ for $i = 1, \ldots, p$, we define R_0 the block of initial residuals

$$R_0 \equiv [r_0^{(1)}, r_0^{(2)}, \ldots, r_0^{(p)}],$$

where each column is $r_0^{(i)} = b^{(i)} - Ax_0^{(i)}$. It is preferable to use the unified notation derived from Algorithm 6.23. In this notation, m is not restricted to being a multiple of the block-size p and the same notation is used for the v_i's as in the scalar Arnoldi Algorithm. Thus, the first step of the block-FOM or block-GMRES algorithm is to compute the QR factorization of the block of initial residuals:

$$R_0 = [v_1, v_2, \ldots, v_p]\, R.$$

Here, the matrix $[v_1, \ldots, v_p]$ is unitary and R is $p \times p$ upper triangular. This factorization provides the first p vectors of the block-Arnoldi basis.

Each of the approximate solutions has the form

$$x^{(i)} = x_0^{(i)} + V_m y^{(i)}, \tag{6.112}$$

and, grouping these approximations $x^{(i)}$ in a block X and the $y^{(i)}$ in a block Y, we can write

$$X = X_0 + V_m Y. \tag{6.113}$$

It is now possible to imitate what was done for the standard FOM and GMRES algorithms. The only missing link is the vector βe_1 in (6.21) which now becomes a matrix. Let E_1 be the $(m+p) \times p$ matrix whose upper $p \times p$ principal block is an identity matrix. Then, the relation (6.109) results in

$$\begin{aligned} B - AX &= B - A\left(X_0 + V_m Y\right) \\ &= R_0 - AV_m Y \\ &= [v_1, \ldots, v_p]R - V_{m+p}\bar{H}_m Y \\ &= V_{m+p}\left(E_1 R - \bar{H}_m Y\right). \end{aligned} \tag{6.114}$$

The vector

$$\bar{g}^{(i)} \equiv E_1 R e_i$$

is a vector of length $m+p$ whose components are zero except those from 1 to i which are extracted from the i-th column of the upper triangular matrix R. The matrix $\bar{H}_m$ is an $(m+p) \times m$ matrix. The block-FOM approximation would consist of deleting the last p rows of $\bar{g}^{(i)}$ and $\bar{H}_m$ and solving the resulting system,

$$H_m y^{(i)} = g^{(i)}.$$

The approximate solution $x^{(i)}$ is then computed by (6.112).

The block-GMRES approximation $x^{(i)}$ is the unique vector of the form $x_0^{(i)} + V_m y^{(i)}$ which minimizes the 2-norm of the individual columns of the block-residual (6.114). Since the column-vectors of V_{m+p} are orthonormal, then from (6.114) we get,

$$\|b^{(i)} - Ax^{(i)}\|_2 = \|\bar{g}^{(i)} - \bar{H}_m y^{(i)}\|_2. \tag{6.115}$$

To minimize the residual norm, the function on the right hand-side must be minimized over $y^{(i)}$. The resulting least-squares problem is similar to the one encountered for GMRES. The only differences are in the right-hand side and the fact that the matrix is no longer Hessenberg, but band-Hessenberg. Rotations can be used in a way similar to the scalar case. However, p rotations are now needed at each new step instead of only one. Thus, if $m = 6$ and $p = 2$, the matrix $\bar{H}_6$ and block right-hand side would be as follows:

$$\bar{H}_8 = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} & h_{16} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} & h_{26} \\ h_{31} & h_{32} & h_{33} & h_{34} & h_{35} & h_{36} \\ & h_{42} & h_{43} & h_{44} & h_{45} & h_{46} \\ & & h_{53} & h_{54} & h_{55} & h_{56} \\ & & & h_{64} & h_{65} & h_{66} \\ & & & & h_{75} & h_{76} \\ & & & & & h_{86} \end{pmatrix} \quad \bar{G} = \begin{pmatrix} g_{11} & g_{12} \\ & g_{22} \\ & \\ & \\ & \\ & \\ & \\ & \end{pmatrix}.$$

For each new column generated in the block-Arnoldi process, p rotations are required to eliminate the elements $h_{k,j}$, for $k = j+p$ down to $k = j+1$. This backward order is important. In the above example, a rotation is applied to eliminate $h_{3,1}$ and then a second rotation is used to eliminate the resulting $h_{2,1}$, and similarly for the second, third step, etc.

This complicates programming slightly since two-dimensional arrays must now be used to save the rotations instead of one-dimensional arrays in the scalar case. After the first column of $\bar{H}_m$ is processed, the block of right-hand sides will have a diagonal added under the diagonal of the upper triangular matrix. Specifically, the above two matrices will have the structure,

$$\bar{H}_8 = \begin{pmatrix} \star & \star & \star & \star & \star & \star \\ & \star & \star & \star & \star & \star \\ & \star & \star & \star & \star & \star \\ & \star & \star & \star & \star & \star \\ & & \star & \star & \star & \star \\ & & & \star & \star & \star \\ & & & & \star & \star \\ & & & & & \star \end{pmatrix} \qquad \bar{G} = \begin{pmatrix} \star & \star \\ \star & \star \\ & \star \\ & \\ & \\ & \\ & \\ & \end{pmatrix},$$

where a $\star$ represents a nonzero element. After all columns are processed, the following least-squares system is obtained.

$$\bar{H}_8 = \left(\begin{array}{cccccc} \star & \star & \star & \star & \star & \star \\ & \star & \star & \star & \star & \star \\ & & \star & \star & \star & \star \\ & & & \star & \star & \star \\ & & & & \star & \star \\ & & & & & \star \\ \hline & & & & & \\ & & & & & \end{array}\right) \qquad \bar{G} = \left(\begin{array}{cc} \star & \star \\ \star & \star \\ \star & \star \\ \star & \star \\ \star & \star \\ \star & \star \\ \hline \star & \star \\ & \star \end{array}\right).$$

To obtain the least-squares solutions for each right-hand side, ignore anything below the horizontal lines in the above matrices and solve the resulting triangular systems. The residual norm of the i-th system for the original problem is the 2-norm of the vector consisting of the components $m+1$, through $m+i$ in the i-th column of the above block of right-hand sides.

Generally speaking, the block methods are of great practical value in applications involving linear systems with multiple right-hand sides. However, they are not as well studied from the theoretical point of view. Perhaps, one of the reasons is the lack of a convincing analogue for the relationship with orthogonal polynomials, established in subsection 6.6.2 for the single-vector Lanczos algorithm. The block version of the Lanczos algorithm has not been covered but the generalization is straightforward.

EXERCISES

1. In the Householder implementation of the Arnoldi algorithm, show the following points of detail:

a. Q_{j+1} is unitary and its inverse is Q_{j+1}^T.

b. $Q_{j+1}^T = P_1 P_2 \dots P_{j+1}$.

c. $Q_{j+1}^T e_i = v_i$ for $i < j$.

d. $Q_{j+1} A V_m = V_{m+1}[e_1, e_2, \ldots, e_{j+1}]\bar{H}_m$, where e_i is the i-th column of the $n \times n$ identity matrix.

e. The v_i's are orthonormal.

f. The vectors $v_1, \ldots, v_j$ are equal to the Arnoldi vectors produced by the Gram-Schmidt version, except possibly for a scaling factor.

2. Rewrite the Householder implementation of the Arnoldi algorithm with more detail. In particular, define precisely the Householder vector w_j used at step j (lines 3-5).

3. Consider the Householder implementation of the Arnoldi algorithm. Give a detailed operation count of the algorithm and compare it with the Gram-Schmidt and Modified Gram-Schmidt algorithm.

4. Derive the basic version of GMRES by using the standard formula (5.7) with $V = V_m$ and $W = AV_m$.

5. Derive a version of the DIOM algorithm which includes partial pivoting in the solution of the Hessenberg system.

6. Show how the GMRES and FOM methods will converge on the linear system $Ax = b$ when

$$A = \begin{pmatrix} & & & & 1 \\ 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

and with $x_0 = 0$.

7. Give a full proof of Proposition 6.11.

8. Let a matrix A have the form

$$A = \begin{pmatrix} I & Y \\ 0 & I \end{pmatrix}.$$

Assume that (full) GMRES is used to solve a linear system, with the coefficient matrix A. What is the maximum number of steps that GMRES would require to converge?

9. Let a matrix A have the form:

$$A = \begin{pmatrix} I & Y \\ 0 & S \end{pmatrix}.$$

Assume that (full) GMRES is used to solve a linear system with the coefficient matrix A. Let

$$r_0 = \begin{pmatrix} r_0^{(1)} \\ r_0^{(2)} \end{pmatrix}$$

be the initial residual vector. It is assumed that the degree of the minimal polynomial of $r_0^{(2)}$ with respect to S (i.e., its grade) is k. What is the maximum number of steps that GMRES would require to converge for this matrix? [Hint: Evaluate the sum $\sum_{i=0}^{k} \beta_i (A^{i+1} - A^i) r_0$ where $\sum_{i=0}^{k} \beta_i t^i$ is the minimal polynomial of $r_0^{(2)}$ with respect to S.]

10. Let

$$A = \begin{pmatrix} I & Y_2 & & & & \\ & I & Y_3 & & & \\ & & I & \ddots & & \\ & & & I & Y_{k-1} & \\ & & & & I & Y_k \\ & & & & & I \end{pmatrix}.$$

a. Show that $(I - A)^k = 0$.

b. Assume that (full) GMRES is used to solve a linear system with the coefficient matrix A. What is the maximum number of steps that GMRES would require to converge?

11. Show that if H_m is nonsingular, i.e., x_m^F is defined, and if $x_m^G = x_m^F$, then $r_m^G = r_m^F = 0$, i.e., both the GMRES and FOM solutions are exact. [Hint: use the relation (6.46) and Proposition 6.11 or Proposition 6.12.]

12. Derive the relation (6.49) from (6.47). [Hint: Use the fact that the vectors on the right-hand side of (6.47) are orthogonal.]

13. In the Householder-GMRES algorithm the approximate solution can be computed by formulas (6.25-6.27). What is the exact cost of this alternative (compare memory as well as arithmetic requirements)? How does it compare with the cost of keeping the v_i's?

14. An alternative to formulas (6.25-6.27) for accumulating the approximate solution in the Householder-GMRES algorithm without keeping the v_i's is to compute x_m as

$$x_m = x_0 + P_1 P_2 \dots P_m y$$

where y is a certain n-dimensional vector to be determined. (1) What is the vector y for the above formula in order to compute the correct approximate solution x_m? [Hint: Exploit (6.11).] (2) Write down an alternative to formulas (6.25-6.27) derived from this approach. (3) Compare the cost of this approach with the cost of using (6.25-6.27).

15. Obtain the formula (6.76) from (6.75).

16. Show that the determinant of the matrix T_m in (6.82) is given by

$$\det\ (T_m) = \frac{1}{\prod_{i=0}^{m-1}\ \alpha_i}$$

17. The Lanczos algorithm is more closely related to the implementation of Algorithm 6.18 of the Conjugate Gradient algorithm. As a result the Lanczos coefficients δ_{j+1} and η_{j+1} are easier to extract from this algorithm than from Algorithm 6.17. Obtain formulas for these coefficients from the coefficients generated by Algorithm 6.18, as was done in Section 6.7.3 for the standard CG algorithm.

18. Show that if the rotations generated in the course of the GMRES (and DQGMRES) algorithm are such that

$$|c_m| \geq c > 0,$$

then GMRES, DQGMRES, and FOM will all converge.

19. Show the exact expression of the residual vector in the basis $v_1, v_2, \dots, v_{m+1}$ for either GMRES or DQGMRES. [Hint: A starting point is (6.52).]

20. Prove that the inequality (6.56) is sharper than (6.53), in the sense that $\zeta_{m+1} \leq \sqrt{m - k + 11}$ (for $m \geq k$). [Hint: Use Cauchy-Schwarz inequality on (6.56).]

21. Denote by S_m the unit upper triangular matrix S in the proof of Theorem 6.1 which is obtained from the Gram-Schmidt process (exact arithmetic assumed) applied to the incomplete orthogonalization basis V_m. Show that the Hessenberg matrix $\bar{H}_m^Q$ obtained in the incomplete orthogonalization process is related to the Hessenberg matrix $\bar{H}_m^G$ obtained from the (complete) Arnoldi process by

$$\bar{H}_m^G = S_{m+1}^{-1} \bar{H}_m^Q S_m.$$

NOTES AND REFERENCES. Lemma 6.1 was proved by Roland Freund [95] in a slightly different form. Proposition 6.12 is due to Brown [43] who proved a number of other theoretical results, including Proposition 6.11. Recently, Cullum and Greenbaum [63] discussed further relationships between FOM and GMRES and other Krylov subspace methods.

The Conjugate Gradient method was developed independently and in different forms by Lanczos [142] and Hesteness and Stiefel [120]. The method was essentially viewed as a direct solution technique and was abandoned early on because it did not compare well with other existing techniques. For example, in inexact arithmetic, the method does not terminate in n steps as is predicted by the theory. This is caused by the severe loss of of orthogonality of vector quantities generated by the algorithm. As a result, research on Krylov-type methods remained dormant for over two decades thereafter. This changed in the early 1970s when several researchers discovered that this loss of orthogonality did not prevent convergence. The observations were made and explained for eigenvalue problems [158, 106] as well as linear systems [167]. The early to the middle 1980s saw the development of a new class of methods for solving nonsymmetric linear systems [13, 14, 127, 172, 173, 185, 218]. The works of Faber and Manteuffel [85] and Voevodin [219] showed that one could not find optimal methods which, like CG, are based on short-term recurrences. Many of the methods developed are mathematically equivalent, in the sense that they realize the same projection process, with different implementations.

The Householder version of GMRES is due to Walker [221]. The Quasi-GMRES algorithm described in Section 6.5.7 was initially described by Brown and Hindmarsh [44], although the direct version DQGMRES was only discussed recently in [187]. The proof of Theorem 6.1 can be found in [152] for the QMR algorithm.

The non-optimality of the Chebyshev polynomials on ellipses in the complex plane was established by Fischer and Freund [90]. Prior to this, a 1963 paper by Clayton [59] was believed to have established the optimality for the special case where the ellipse has real foci and γ is real.

Until recently, little attention has been given to block Krylov methods. In addition to their attraction for solving linear systems with several right-hand sides [177, 196], these techniques can also help reduce the effect of the sequential inner products in parallel environments and minimize I/O costs in out-of-core implementations. The block-GMRES algorithm is analyzed by Simoncini and Gallopoulos [197] and in [184]. Alternatives to GMRES which require fewer inner products have been proposed by Sadok [188] and Jbilou [125]. Sadok investigated a GMRES-like method based on the Hessenberg algorithm [227], while Jbilou proposed a multi-dimensional generalization of Gastinel's method seen in Exercise 2 of Chapter 5. ■