Chapter 9
Krylov Subspace Methods

9.1 Introduction

We now turn to methods for large matrices. If a matrix is really large, the computation of its complete spectrum is out of the question. Fortunately it often suffices to compute just a few of the eigenvalues. For example, if stability is at issue, one might just want to know the few eigenvalues with maximum real part or with maximum absolute value. If a certain frequency range is of interest, one might like to calculate the few eigenvalues that are closest to some specified target value τ. Eigenvalue methods for large problems are designed to perform tasks like these, and they do it by computing the invariant subspace associated with the desired eigenvalues.

Very large matrices that arise in applications are almost always sparse. That is, the vast majority of their entries are zero. Were this not so, it might not even be possible to store such a matrix, much less find its eigenvalues. For example, consider a matrix of order $n = 10^6$. If we store all of its entries in the conventional way, we have to store 10^{12} numbers. If we are storing, say, double-precision real numbers, that takes 8×10^{12} bytes, i.e. 8000 gigabytes. However, it might well happen that the matrix only has about ten nonzero entries per row, and then we can save the situation by using a special data structure that stores only the nonzero entries of the matrix. Then we only have to store about 10^7 numbers or 8×10^7 bytes. Of course there is also a bit of overhead involved: For each matrix entry that we store, we have to store its row and column number as well, so that we know where it belongs in the matrix. Multiplying by two to take the overhead into account we see that we will need about 16×10^7 bytes or 160 megabytes. This is a manageable file.

The methods that we have discussed so far all use similarity transformations. That approach is not possible for large sparse matrices, as each similarity transformation causes fill-in, the introduction of nonzeros in positions that previously contained zeros. After just a few similarity transformations, the matrix becomes completely full, hence unstorable.

If we can't do similarity transformations, then what can we do? The one thing we can do very well with a large, sparse matrix is multiply it by a vector. Indeed, this operation can be done relatively quickly; the amount of work is proportional to the number of nonzeros in the matrix. If we multiply A by x to get Ax, we can then multiply A by that vector to get

A^2x, and so on, so it is easy to build up a *Krylov sequence*

$$x,\ Ax,\ A^2x,\ A^3x,\ \ldots.$$

In this chapter we investigate *Krylov subspace methods* which build up Krylov subspaces

$$\mathcal{K}_j(A, x) = \operatorname{span}\{x, Ax, A^2x, \ldots, A^{j-1}x\}$$

and look for good approximations to eigenvectors and invariant subspaces within the Krylov spaces.

In many applications the matrix has structure that allows it to be stored even more compactly than we have indicated above. For example, the matrix may have submatrices with repeating patterns that do not need to be stored multiple times. In fact, it may not be necessary to store the matrix at all. If a subroutine that can compute Ax, given any input x, is provided, Krylov subspace methods can be used.

If we want to build a Krylov subspace, we need a starting vector x. What would be a good starting vector? If we could find one that was in the invariant subspace that we are looking for, then the Krylov spaces that it generates would lie entirely within that subspace. That's a bit much to ask for, but if we can get a vector that is close to the desired invariant subspace, perhaps we can get some good approximations. That also is a lot to ask for, at least initially. At the outset we might have no idea where the desired invariant subspace lies. In that case we might as well pick an x at random and get started. As we shall see, we will be able to replace it by a better x later.

After $j - 1$ steps of a Krylov subspace process, we will have built up a j-dimensional Krylov subspace $\mathcal{K}_j(A, x)$. The larger j is, the better is our chance that the space contains good approximations to desired eigenvectors. But now we have to make an important observation. Storing a j-dimensional space means storing j vectors, and the vectors take up a lot of space. Consider again the case $n = 10^6$. Then each vector consists of 10^6 numbers and requires 8 megabytes of storage. If we store a lot of vectors, we will quickly fill up our memory. Thus there is a limit to how big we can take j. In typical applications j is kept under 100, or at most a few hundreds.

Given the limitation on subspace size, we ordinarily resort to restarts. Say we are looking for an invariant subspace of some modest dimension $\widehat{m}$. We pick m at least as big as $\widehat{m}$ and preferably a bit bigger, e.g. $m = \widehat{m} + 2$. We begin by generating a Krylov subspace $\mathcal{K}_k(A, x)$ of dimension k, where k is somewhat bigger than m, e.g. $k = 2m$ or $k = 5m$. Then we do an analysis that allows us to pick out a promising m-dimensional subspace of $\mathcal{K}_k(A, x)$, we discard all but m vectors, and we restart the process at step m. The details will be given in the coming sections. The restart is tantamount to picking a new, improved, x and building up a new, improved, Krylov space from that x. Since the process is organized so that we can start from step m instead of step 0, it is called an *implicit restart*. This gives improved efficiency and accuracy, compared with an explicit restart from step 0.

When the procedure works as intended (as it usually does), each restart gives us a better approximation to the desired invariant subspace. Repeated restarts lead to convergence.

Approximating eigenvectors from Krylov subspaces

We now consider the question of whether a given Krylov subspace $\mathcal{K}_j(A, x)$ contains good approximants to eigenvectors of A. We suppose $j \ll n$, as is usually the case in practical situations. We begin by establishing a simple characterization of $\mathcal{K}_j(A, x)$ in terms of polynomials.

$\mathcal{K}_j(A, x)$ is the set of all linear combinations

$$a_0 x + a_1 A x + a_2 A^2 x + \cdots + a_{j-1} A^{j-1} x. \tag{9.1.1}$$

Given any coefficients $a_0, \ldots, a_{j-1}$, we can build a polynomial $q(z) = a_0 + a_1 z + a_2 z^2 + \cdots + a_{j-1} z^{j-1}$ of degree $j - 1$ or less. Then the linear combination in (9.1.1) can be written more compactly as $q(A)x$. Thus we have the following simple characterization of the Krylov subspace.

Proposition 9.1.1. *Let $\mathcal{P}_{j-1}$ denote the set of all polynomials of degree less than j. Then*

$$\mathcal{K}_j(A, x) = \left\{ q(A)x \mid q \in \mathcal{P}_{j-1} \right\}.$$

The question of whether $\mathcal{K}_j(A, x)$ contains good approximants to a given eigenvector v is therefore that of whether there are polynomials $q \in \mathcal{P}_{j-1}$ such the $q(A)x \approx v$. To keep the discussion simple, let us suppose that A is a semisimple matrix and has linearly independent eigenvectors $v_1, v_2, \ldots, v_n$ with associated eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. Then

$$x = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$$

for some unknown coefficients $c_1, \ldots, c_n$. For any polynomial q we have

$$q(A)x = c_1 q(\lambda_1) v_1 + c_2 q(\lambda_2) v_2 \cdots + c_n q(\lambda_n) v_n.$$

If we want to find a good approximant to, say, v_1, we should look for a polynomial q such that $|q(\lambda_1)|$ is large relative to $|q(\lambda_2)|, \ldots, |q(\lambda_n)|$. This is most easily achieved if λ_1 is well separated from the rest of the spectrum. Picture a situation in which all of the eigenvalues are in a cluster in the complex plane, except for λ_1, which is off at a distance. Then there exist polynomials of the desired form: Just build a q of degree j that has all of its zeros in the cluster; for example, take $q(z) = (z - z_0)^j$, where z_0 is somewhere in the "middle" of the cluster. Then $q(\lambda_1)$ will be much larger than $q(\lambda_2), \ldots, q(\lambda_n)$, even if j is quite small. Thus $\mathcal{K}_j(A, x)$ contains vectors that approximate v_1 well, unless c_1 happens to be unusually small.

The closer λ_1 is to the other eigenvalues, the harder it is to get a polynomial q with the desired properties. Now suppose λ_1 is surrounded by other eigenvalues. Then it is normally impossible to build a q of low degree j such that $q(\lambda_1)$ is large and $q(\lambda_2), \ldots, q(\lambda_n)$ are small. Thus $\mathcal{K}_j(A, x)$ will not contain good approximants to v_1. We conclude that Krylov subspaces are best at approximating eigenvectors associated with eigenvalues on the periphery of the spectrum.

Rigorous results can be obtained with the help of Chebyshev polynomials, which are discussed in Exercises 9.1.2 and 9.1.4.[1] For more details see [100], [185], and the references therein, especially [127], [161], and [183].

[1]Analyses of this type are most meaningful in the case when A is normal. Then the eigenvectors $v_1, \ldots, v_n$ can be taken to be orthonormal (Theorem 2.3.5), and it is natural to work in the coordinate system defined by them. If A is not normal, then $v_1, \ldots, v_n$ cannot be taken to be orthonormal, and the coordinate system they define is distorted. The distortion is measured by a factor $\kappa_2(V)$, the condition number of the matrix of eigenvectors, which appears in the rigorous bounds. Bounds can be obtained for defective matrices by considering the Jordan canonical form (Theorem 2.4.11), but again the coordinate system is distorted.

The function of restarts is to improve the coefficients $c_1, \ldots, c_n$. Let us suppose that $v_1, \ldots, v_m$ are the eigenvectors that we wish to find. Then, by restarting, we hope to replace x by a new starting vector

$$\widehat{x} = \widehat{c}_1 v_1 + \widehat{c}_2 v_2 + \cdots + \widehat{c}_n v_n$$

such that $\widehat{c}_1, \ldots, \widehat{c}_m$ are relatively larger and $\widehat{c}_{m+1}, \ldots, \widehat{c}_n$ are relatively smaller.

The shift-and-invert strategy

As we have seen, Krylov subspace methods are good at producing the eigenvalues on the periphery of the spectrum. If we want to find eigenvalues in the interior, special action must be taken. Suppose we want to find the eigenvalues that are nearest to some target value $\tau \in \mathbb{C}$. If we shift by τ and invert, we get the matrix $(A - \tau I)^{-1}$, which has eigenpairs $((\lambda - \tau)^{-1}, v)$ corresponding to eigenpairs (λ, v) of A. Thus the eigenvalues of A that are closest to τ correspond to the eigenvalues of $(A - \tau I)^{-1}$ of greatest modulus. These are peripheral eigenvalues. If we apply a Krylov subspace method to $B = (A - \tau I)^{-1}$, we will get the desired eigenvalues and eigenvectors.

Care must be exercised in implementing this *shift-and-invert* strategy. We cannot afford to form the matrix $(A - \tau I)^{-1}$ explicitly, as it is not sparse. Fortunately we do not need $(A - \tau I)^{-1}$; all that's needed is way of making the transformation $x \mapsto (A - \tau I)^{-1} x$ for any x. This can be achieved by solving $(A - \tau I)y = x$ for y, as then $y = (A - \tau I)^{-1} x$. If we compute an LU factorization with pivoting: $A = PLU$, where P is a permutation matrix, L is unit lower triangular, and U is upper triangular, we can use this factorization to backsolve for y, given any x. The factorization only needs to be done once, then it can be used repeatedly. For sparse A it is typically the case that the factors L and U are much less sparse than A is, but they are (typically) still fairly sparse and can be stored compactly in a sparse data structure. However, they will (typically) take up much more memory than A does. The shift-and-invert strategy is a viable option if and only if there is enough available memory to store the L and U factors, When it is a viable option, it generally works very well.

Exercises

Exercise 9.1.1. Let $A \in \mathbb{C}^{n \times n}$, and let $\tau \in \mathbb{C}$ be a number that is not an eigenvalue of A. Let $\mathcal{S}$ be a subspace of $\mathbb{C}^n$.

(a) Show that $\mathcal{S}$ is invariant under $(A - \tau I)^{-1}$ if and only if $\mathcal{S}$ is invariant under A.

(b) Show that (λ, v) is an eigenpair of A if and only if $((\lambda - \tau)^{-1}, v)$ is an eigenpair of $(A - \tau I)^{-1}$.

Exercise 9.1.2. In this exercise we introduce and study the famous and very useful Chebyshev polynomials. A substantial reference on Chebyshev polynomials is [177]. We begin by studying the analytic map

$$x = \phi(z) = \frac{z + z^{-1}}{2}, \tag{9.1.2}$$

which has poles at zero and infinity and maps the rest of the complex plane onto the complex plane.

(a) Show that if z satisfies (9.1.2) for a given value of x, then z satisfies a certain quadratic equation. Solve that equation and deduce that

$$z = x \pm \sqrt{x^2 - 1}. \tag{9.1.3}$$

Thus each x is the image of two z values, which are distinct if $x \neq \pm 1$. Show that if $z = x + \sqrt{x^2 - 1}$, then $z^{-1} = x - \sqrt{x^2 - 1}$. The points ± 1 are images only of themselves. They are fixed points of ϕ. (Remark: ϕ is the map for Newton's method to compute $\sqrt{1}$.)

(b) For $k = 0, 1, 2, 3, \ldots$the kth *Chebyshev Polynomial* $T_k(x)$ is defined by

$$T_k(x) = \frac{z^k + z^{-k}}{2}, \quad \text{where} \quad x = \frac{z + z^{-1}}{2}, \tag{9.1.4}$$

that is,

$$T_k(x) = \phi(z^k), \quad \text{where} \quad x = \phi(z).$$

Use the definition (9.1.4) and some simple algebra to show that for $k = 1, 2, 3, \ldots$,

$$2xT_k(x) = T_{k+1}(x) + T_{k-1}(x).$$

This establishes the recurrence

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x), \qquad k = 1,\ 2,\ 3,\ \ldots \tag{9.1.5}$$

(c) This gives us a means of generating T_{k+1} from T_k and T_{k-1}. Show that the definition (9.1.4) gives $T_0(x) = 1$ and $T_1(x) = x$. Then use (9.1.5) to calculate T_2, T_3, T_4, and T_5.

(d) Prove by induction on k that for $k = 0, 1, 2, 3, \ldots$, T_k is a polynomial of degree k. Moreover, its leading coefficient is 2^{k-1}, except when $k = 0$. Show further that T_k is an even (odd) polynomial when k is even (odd).

(d) Substitute $z = e^{i\beta}$ in (9.1.4) and deduce that the Chebyshev polynomials satisfy

$$T_k(x) = \cos k\beta, \quad \text{where} \quad x = \cos\beta \tag{9.1.6}$$

for x satisfying $-1 \leq x \leq 1$. In the following discussion, restrict β to the interval $[0, \pi]$, which the cosine function maps 1-1 onto $[-1, 1]$.

(e) Graph the function $\cos\beta$, $\cos 2\beta$, and $\cos 3\beta$ for $0 \leq \beta \leq \pi$.

(f) Show that $T_k(x)$ has k real zeros in $(-1, 1)$. Since a kth degree polynomial can have no more than k zeros, we have now shown that all of the zeros of T_k are real, and they all lie in $(-1, 1)$.

(g) Show that $\sup_{-1\le x\le 1} |T_k(x)| = 1$. Show that there are $k+1$ points in $[-1, 1]$ at which $|T_k(x)| = 1$.

(h) We now know that $|T_k|$ is bounded by 1 in $[-1, 1]$, and all of its zeros lie in that interval. It follows that T_k must grow rapidly as x moves away from $[-1, 1]$. This is already evident from definition (9.1.4): Show that when x is real and large, $x \approx z/2$ and $T_k(x) \approx z^k/2 \approx 2^{k-1}x$.

(i) More importantly, for a fixed $x > 1$, $T_k(x)$ grows exponentially as a function of k. To prove this we introduce another parametrization. Substitute $z = e^\alpha$ $(\alpha \ge 0)$ into the definition (9.1.4) to obtain

$$T_k(x) = \cosh k\alpha, \quad \text{where} \quad x = \cosh\alpha \tag{9.1.7}$$

for all x satisfying $1 \le x < \infty$. For a given $x > 1$, there is a unique $\alpha > 0$ such that $x = \cosh\alpha$. Let $\rho = e^\alpha > 1$. Use (9.1.7) or (9.1.4) to show that $T_k(x) \ge \frac{1}{2}\rho^k$. Thus $T_k(x)$ grows exponentially as a function of k.

Exercise 9.1.3. In this exercise we investigate how well eigenvectors associated with peripheral eigenvalues are approximated by Krylov subspaces in the case when the matrix has real eigenvalues. Let A be a semisimple matrix with (real) eigenvalues $\lambda_1 > \lambda_2 \ge \cdots \ge \lambda_n$. For example, A could be Hermitian. Assume also that $\lambda_2 > \lambda_n$.

(a) Show that the transformation

$$w = 1 - 2\frac{x - \lambda_2}{\lambda_n - \lambda_2}$$

maps $[\lambda_n, \lambda_2]$ onto $[-1, 1]$

(b) For $k = 0, 1, 2, \ldots$, define polynomials $q_k \in \mathcal{P}_k$ by

$$q_k(x) = T_k(w) = T_k\left(1 - 2\frac{x - \lambda_2}{\lambda_n - \lambda_2}\right),$$

where T_k is the Chebyshev polynomial defined in Exercise 9.1.2. Show that $|q_k(\lambda_i)| \le 1$ for $i = 2, \ldots, n$, and

$$|q_k(\lambda_1)| \ge \frac{1}{2}\rho^m,$$

where

$$\rho \ge 1 + 2\frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n} > 1. \tag{9.1.8}$$

Use part (i) of Exercise 9.1.2.

(c) Let $v_1, \ldots, v_n$ be linearly independent eigenvectors associated with eigenvalues $\lambda_1, \ldots, \lambda_n$, respectively. Let x be a starting vector for a Krylov process satisfying

$$x = c_1 v_1 + \cdots + c_n v_n$$

with $c_1 \neq 0$. Define $\widetilde{q}_k \in \mathcal{P}_k$ by $\widetilde{q}_k(x) = q_k(x)/(c_1 q_k(\lambda_1))$. Let $w_k = \widetilde{q}_{k-1}(A)x \in \mathcal{K}_k(A, x)$. Show that there is a constant C such that

$$\| w_k - v_1 \|_2 \leq C\rho^{-k}, \qquad k = 0,\ 1,\ 2,\ \ldots,$$

where ρ is as in (9.1.8). Thus, for large enough k, the Krylov subspace $\mathcal{K}_k(A, x)$ contains a vector that is close to the eigenvector v_1.

(d) Modify the arguments given above to show that $\mathcal{K}_k(A, x)$ contains vectors that are close to v_n under suitable assumptions.

(e) Suppose $\lambda_1 > \lambda_2 > \lambda_3 > \lambda_n$. For $k = 1, 2, 3, \ldots$define a polynomial $\widehat{q}_k \in \mathcal{P}_k$ by

$$\widehat{q}_k(x) = (x - \lambda_1)T_{k-1}\left(1 - 2\frac{x - \lambda_3}{\lambda_n - \lambda_3}\right).$$

Use $\widehat{q}_k$ to show that for sufficiently large k, the Krylov subspace $\mathcal{K}_k(A, x)$ contains vectors that are close to v_2, assuming $c_2 \neq 0$.

(f) Suppose $\lambda_2 > \lambda_3 > \lambda_4 > \lambda_n$. Show that for sufficiently large k, $\mathcal{K}_k(A, x)$ contains vectors that approximate v_3 well, assuming $c_3 \neq 0$.

Exercise 9.1.4. In Exercise 9.1.2 we introduced the Chebyshev polynomials and studied their properties on the real line. We used these properties in Exercise 9.1.3 to investigate approximation properties of Krylov subspaces for semisimple matrices with real eigenvalues. If we want to obtain results for matrices with complex eigenvalues, we must investigate the properties of Chebyshev polynomials in the complex plane.

(a) Substitute $z = e^{\alpha + i\beta}$ into (9.1.4), where $\alpha > 0$ and $\beta \in (-\pi, \pi]$. Show that

$$x = \cosh\alpha\cos\beta + i\sinh\alpha\sin\beta.$$

and

$$T_k(x) = \cosh k\alpha\cos k\beta + i\sinh k\alpha\sin k\beta.$$

This generalizes the results of parts (d) and (i) of Exercise 9.1.2.

(b) Let $u = \Re(x)$ and $v = \Im(x)$, so that $x = u + iv$. Suppose α is held fixed and β is allowed to run from $-\pi$ to π. Show that x sweeps out an ellipse E_α with major semiaxis $u = \cosh\alpha$ and minor semiaxis $v = \sinh\alpha$, and meanwhile $T_k(x)$ sweeps out (k times) and ellipse $E_{k\alpha}$ with semiaxes $\cosh k\alpha$ and $\sinh k\alpha$. Notice that as $\alpha \to 0^+$, E_α becomes increasingly eccentric. In the limiting case $\alpha = 0$, E_α becomes the interval $[-1, 1]$ on the real axis.

(c) Show that T_k maps E_α (and its interior) to $E_{k\alpha}$ (and its interior). Thus

$$\max_{x \in E_\alpha} | T_k(x) | \leq \cosh k\alpha.$$

(d) These results can be used to get error bounds for Krylov subspace approximation, in case the spectrum is not real, as follows. Suppose all of the eigenvalues but one lie in the ellipse E_{α_2}, except that there is a positive eigenvalue $\lambda_1 = \cosh\alpha_1$ to the right of E_{α_2}. That is $\alpha_1 > \alpha_2 > 0$. Show that for $\lambda \in E_{\alpha_1}$,

$$\frac{|\,T_k(\lambda)\,|}{|\,T_k(\lambda_1)\,|} \le \frac{\cosh k\alpha_2}{\cosh k\alpha_1} \le 2\left(e^{\alpha_2-\alpha_1}\right)^k \to 0.$$

Deduce that the Krylov subspace $\mathcal{K}_k(A, x)$ contains good approximations to v_1 if k is large enough, provided that $c_1 \neq 0$. This result assumes that λ_1 is positive and real. However, similar bounds can be obtained for peripheral eigenvalues that are neither positive nor real since the polynomials can be shifted, scaled, and rotated. All we need is to find an ellipse that contains all eigenvalues but λ_1, such that λ_1 lies on the major semiaxis of the ellipse. Then we can get a convergence result.[2]

Exercise 9.1.5. Using the definition (9.1.4) and inverse map (9.1.3), we immediately obtain the expression

$$T_k(x) = \frac{(x+\sqrt{x^2-1})^k + (x-\sqrt{x^2-1})^k}{2}, \tag{9.1.9}$$

(a) Use (9.1.9) to compute T_0, T_1, T_2, and T_3.

(b) Simplifying (9.1.9) by taking $w = \sqrt{x^2-1}$, we have

$$T_k(x) = \frac{(x+w)^k + (x-w)^k}{2},$$

where $x^2 - w^2 = 1$. Apply the Binomial Theorem to this expression, and observe that the odd powers of w cancel out. Conclude that

$$T_k(x) = \sum_{m=0}^{\lfloor k/2\rfloor} \binom{k}{2m} x^{k-2m} w^{2m} = \sum_{m=0}^{\lfloor k/2\rfloor} \binom{k}{2m} x^{k-2m}(x^2-1)^m.$$

This shows clearly that T_k is a polynomial of degree k.

(c) Show further that

$$T_k(x) = \sum_{j=0}^{\lfloor k/2\rfloor} c_j x^{k-2j},$$

where

$$c_j = (-1)^j \sum_{m=j}^{\lfloor k/2\rfloor} \binom{k}{2m}\binom{m}{j}.$$

Exercise 9.1.6. Here we establish an orthogonality property for Chebyshev polynomials.

[2]It must be acknowledged, however, that these results are much weaker than one obtains in the real eigenvalue case, because the function $\cosh\alpha$ is flat at $\alpha = 0$ and not at $\alpha > 0$.

(a) Show that $\int_0^\pi \cos k\beta \cos j\beta \, d\beta = \delta_{kj}\frac{\pi}{2}$.

(b) Using (9.1.6) show that $\int_{-1}^1 T_k(x)T_j(x)(1-x^2)^{-1/2}dx = \delta_{kj}\frac{\pi}{2}$.

9.2 The Generic Krylov Process

If the vectors

$$x, Ax, A^2x, \ldots, A^{k-1}x$$

are linearly independent, they form a basis for the Krylov subspace $\mathcal{K}_k(A, x)$. From a numerical standpoint this is not a very good basis. As we know from our study of the power method, the vectors A^jx (usually) point more and more in the direction of the dominant eigenvector as j increases. This means that for large (or even moderate) k, most of the vectors in the Krylov basis will point in about the same direction. Thus the basis is (usually) ill conditioned. Let us consider how we might build a well-conditioned basis. Start with $u_1 = c_1x$, where c_1 is a scale factor. For example we might take $c_1 = 1$, or we might choose c_1 so that $\| u_1 \| = 1$. To get another basis vector we can calculate Au_1, then we can improve the basis by setting

$$u_2 = Au_1 - u_1h_{11},$$

where h_{11} is chosen with an eye to making $\{u_1, u_2\}$ a better basis than $\{u_1, Au_1\}$ in some sense. For example, we could make u_2 orthogonal to u_1. This popular choice leads to the Arnoldi process, which will be described in § 9.4. It is the best choice from the standpoint of numerical stability. Since we like to be able to control the norm of u_2, we set, more generally,

$$u_2h_{21} = Au_1 - u_1h_{11},$$

where h_{21} is a scale factor. We cannot afford to keep a lot of extra vectors lying around, so we keep u_2 and discard Au_1. When we build our next basis vector, we will start by forming Au_2 instead of the unavailable $A^2u_1 = A(Au_1)$.

Now, proceeding by induction, let us suppose that we have generated u_1, …, u_j, a basis for $\mathcal{K}_j(A, x)$. Suppose further that

$$\mathcal{K}_i(A, x) = \text{span}\{u_1, \ldots, u_i\}, \quad i = 1, \ldots, j.$$

We now wish to find a vector u_{j+1} such that $\mathcal{K}_{j+1}(A, x) = \text{span}\{u_1, \ldots, u_{j+1}\}$. Since A^ju_1 is unavailable, we use instead the vector $u_j \in \mathcal{K}_j(A, x) \setminus \mathcal{K}_{j-1}(A, x)$. We first compute Au_j and then form u_{j+1} by some normalization process:

$$u_{j+1}h_{j+1,j} = Au_j - \sum_{i=1}^{j} u_ih_{ij}. \tag{9.2.1}$$

We will admit only processes with the following characteristic: If it happens that $Au_j \in \text{span}\{u_1, \ldots, u_j\}$, the h_{ij} should be chosen so that

$$Au_j = \sum_{i=1}^{j} u_ih_{ij}.$$

Then $h_{j+1,j} = 0$, u_{j+1} is indeterminate, and the process halts. We shall call such a procedure a *Krylov process*.

There are two widely used Krylov processes: the Arnoldi process (§ 9.4) and the unsymmetric Lanczos process (§ 9.6). The Arnoldi process simply chooses the h_{ij} so that the basis is orthonormal. We defer the description of the unsymmetric Lanczos process, which is more complicated. There is also a symmetric Lanczos process, which is a special case of both the Arnoldi and unsymmetric Lanczos processes. In Sections 9.8 and 9.9 we will develop special Krylov processes for Hamiltonian and symplectic matrices.

The two following propositions, which are easily proved (Exercises 9.2.1 and 9.2.2), state some basic facts about all Krylov processes.

Proposition 9.2.1. *If x, Ax, A^2x, …, $A^{k-1}x$ are linearly independent, then every Krylov process starting with $u_1 = c_1 x$ produces a basis of $\mathcal{K}_k(A, x)$ such that*

$$\operatorname{span}\{u_1, \ldots, u_i\} = \mathcal{K}_i(A, x) \quad \textit{for} \quad i = 1, \ldots, k.$$

Proposition 9.2.2. *If x, Ax, A^2x, …, $A^{k-1}x$ are linearly independent, but $A^k x \in \mathcal{K}_k(A, x)$, then $\mathcal{K}_k(A, x)$ is invariant under A. Every Krylov process starting with $u_1 = c_1 x$ produces a u_k such that $Au_k \in \mathcal{K}_k(A, x)$, causing termination at this step.*

The process (9.2.1) can be summarized neatly as a matrix equation. If we rewrite (9.2.1) as

$$Au_j = \sum_{i=1}^{j+1} u_i h_{ij},$$

we easily see that the following proposition holds.

Proposition 9.2.3. *Suppose x, Ax, A^2x, …, $A^{k-1}x$ are linearly independent, and let u_1, …, u_k be generated by a Krylov process starting with $u_1 = c_1 x$. Let U_k denote the $n \times k$ matrix whose columns are u_1, …, u_k. Let $H_{k+1,k}$ denote the $(k+1) \times k$ upper Hessenberg matrix whose (i, j) entry h_{ij} is as in (9.2.1) if $i \le j+1$, and $h_{ij} = 0$ if $i > j+1$. Then*

$$AU_k = U_{k+1} H_{k+1,k}. \tag{9.2.2}$$

Let H_k denote the square upper Hessenberg matrix obtained by deleting the bottom row of $H_{k+1,k}$. Then

$$AU_k = U_k H_k + u_{k+1} h_{k+1,k} e_k^T, \tag{9.2.3}$$

where $e_k^T = [0\ 0\ \ldots\ 0\ 1] \in \mathbb{R}^k$.

If $\mathcal{K}_k(A, x)$ is invariant, then $h_{k+1,k} = 0$, and (9.2.3) becomes

$$AU_k = U_k H_k. \tag{9.2.4}$$

Finding an invariant subspace is a desired event. If $k \ll n$, as it is in practical situations, we can easily compute the eigenvalues and eigenvectors of H_k by, say, the QR algorithm.

Every eigenpair of H_k yields an eigenpair of A: Given $H_k y = y\lambda$, let $v = U_k y$. Then $Av = v\lambda$ (Proposition 2.1.11).

If we ignore temporarily the problem of storage space, we can think about carrying on a Krylov process indefinitely. But in our finite-dimensional setting, every Krylov process must halt eventually. If x, Ax, …, $A^{n-1}x$, are independent, then they span the space $\mathbb{C}^n$, and $A^n x$ must be a linear combination of them. The Krylov process will produce a basis u_1, …, u_n and halt. If an invariant subspace is discovered along the way, we halt even sooner, but the process can be continued. If $\operatorname{span}\{u_1, \ldots, u_j\}$ is invariant, we can take u_{j+1} to be an appropriate vector that is not in $\operatorname{span}\{u_1, \ldots, u_j\}$. If we proceed in this way, we can always, in principle, build a basis $u_1, \ldots, u_n$. Then (9.2.4) becomes

$$AU = UH, \tag{9.2.5}$$

where U is $n \times n$ and nonsingular. Since $A = UHU^{-1}$, we conclude that every Krylov process, if carried to completion, produces a similarity transformation to an H that is in upper Hessenberg form. If we have found any invariant subspaces along the way, some of the $h_{j+1,j}$ will be zero, so H will be block triangular.

In real applications it is never possible to carry a Krylov process to completion, but now that we have considered the possibility, we can see that a partially completed Krylov process performs a partial similarity transformation. Looking at (9.2.2) or (9.2.3), we see that U_k consists of the first k columns of the transforming matrix U in (9.2.5), and H_k is the upper-left-hand $k \times k$ submatrix of H. Thus (9.2.2) and (9.2.3) represent partial similarity transformations.

It is a useful fact that Proposition 9.2.3 has a converse: If the vectors u_1, …, u_{k+1} satisfy an equation of the form (9.2.3), they must span nested Krylov subspaces.

Proposition 9.2.4. *Suppose u_1, …, u_{k+1} are linearly independent vectors, U_k is the matrix whose columns are u_1, …, u_k, H_k is properly upper Hessenberg, $h_{k+1,k} \neq 0$, and*

$$AU_k = U_k H_k + u_{k+1} h_{k+1,k} e_k^T.$$

Then

$$\operatorname{span}\{u_1, \ldots, u_j\} = \mathcal{K}_j(A, u_1), \qquad j = 1, \ldots, k+1.$$

To prove this result, one simply notes that the matrix equation implies that (9.2.1) holds for $j = 1$, …, k. One then uses (9.2.1) to prove Proposition 9.2.4 by induction on j. See Exercise 9.2.4.

Quality of eigenpair approximants

After enough steps of a Krylov process, we expect the Krylov subspace $\mathcal{R}(U_k) = \mathcal{K}_k(A, x)$ to contain good approximants to eigenvectors associated with some of the peripheral eigenvalues of A. How do we locate these good approximants? We know that if $h_{k+1,k} = 0$, each eigenpair (μ, y) of H_k yields an eigenpair $(\mu, U_k y)$ of A. It is therefore reasonable to expect that if $h_{k+1,k}$ is near zero, then the eigenpairs of H_k will yield good approximants to eigenpairs of A. If k is not too large, then computing these quantities poses no problem.

The next simple result shows that even if $h_{k+1,k}$ is not close to zero, some of the $(\mu, U_k y)$ may be excellent approximations of eigenpairs.

Proposition 9.2.5. *Suppose we perform k steps of a Krylov process on A to generate U_{k+1} and $H_{k+1,k}$ related by (9.2.2) and (9.2.3). Let (μ, y) be an eigenpair of H_k, and let $w = H_k y$, so that (μ, w) is an approximate eigenpair of A. Then*

$$\| Aw - w\mu \| = \| u_{k+1} \| \, | h_{k+1,k} | \, | y_k |. \tag{9.2.6}$$

Here y_k is the last component of the eigenvector y.

This is easily proved by multiplying (9.2.3) by y and doing some obvious manipulations. The value of Proposition 9.2.5 is this: (μ, w) is an exact eigenpair of A if and only if the residual $Aw - w\mu$ is exactly zero. If the residual is tiny, we expect (μ, w) to be a good approximant of an eigenpair. The right-hand side of (9.2.6) shows how to compute the norm of the residual at virtually no cost. Practical Krylov processes always compute vectors satisfying $\| u_{k+1} \| \approx 1$, so we can ignore that factor. The factor $| h_{k+1,k} |$ is expected. The factor $| y_k |$ shows that even if $h_{k+1,k}$ is not small, the residual can nevertheless be tiny. If the bottom component of the eigenvector y is tiny, then the residual must be tiny. Practical experience has shown that this happens frequently. It often happens that peripheral eigenpairs are approximated excellently even though $h_{k+1,k}$ is not small. Of course we must remember that a tiny residual does not necessarily imply an excellent approximation. All it means is that (μ, w) is the exact eigenpair of some nearby matrix $A + E$ (Proposition 2.7.20). If the eigensystem of A is well conditioned, we can then infer that (μ, w) approximates an eigenpair of A well.

One further related problem merits discussion. If a Krylov subspace $\mathcal{R}(U_k)$ contains a vector that approximates an eigenvector v well, is it necessarily true that the eigenpair (λ, v) is well approximated by one of the eigenpairs $(\mu, U_k y)$ generated from the eigenpairs of H_k? Exercise 9.2.7 shows that the answer is usually but not always yes. Again (as in the previous paragraph) conditioning is an issue.

Notes and references

Around 1930 the engineer and scientist Alexei Nikolaevich Krylov used Krylov subspaces to compute the coefficients of a minimal polynomial, the roots of which are eigenvalues of a matrix [138]. Krylov's method is described in [83, § 42]. The modern era of Krylov subspace methods began with the Lanczos [143] and Arnoldi [11] processes, as well as the conjugate-gradient method [113] for solving positive definite linear systems, around 1950.

Exercises

Exercise 9.2.1. Prove Proposition 9.2.1 by induction on k:

(a) Show that the proposition holds for $k = 1$.

(b) Suppose $j < k$ and

$$\text{span}\{u_1, \ldots, u_i\} = \mathcal{K}_i(A, x) \quad \text{for} \quad i = 1, \ldots, j.$$

Show that $u_j \in \mathcal{K}_j(A,x) \setminus \mathcal{K}_{j-1}(A,x)$. Deduce that $Au_j \in \mathcal{K}_{j+1}(A,x) \setminus \mathcal{K}_j(A,x)$.

(c) Use (9.2.1) to show that $\operatorname{span}\{u_1,\ldots,u_{j+1}\} \subseteq \mathcal{K}_{j+1}(A,x)$.

(d) Show that $\operatorname{span}\{u_1,\ldots,u_{j+1}\} = \mathcal{K}_{j+1}(A,x)$. This completes the proof by induction.

Exercise 9.2.2. Prove Proposition 9.2.2.

Exercise 9.2.3. Prove Proposition 9.2.3.

Exercise 9.2.4. Show that (9.2.3) implies that (9.2.1) holds for $j = 1, \ldots, k$. Then use (9.2.1) to prove by induction on j that $\operatorname{span}\{u_1,\ldots,u_j\} \subseteq \mathcal{K}_j(A,u_1)$ for $j = 1. \ldots k+1$. Get equality of subspaces by a dimension argument. This proves Proposition 9.2.4.

Exercise 9.2.5. Consider the trivial Krylov process that takes $u_{j+1} = Au_j$, as long as the vectors remain linearly independent. Show that if this process is run until an invariant subspace $\mathcal{K}_i(A,x)$ is found, the resulting upper Hessenberg matrix H_i is a companion matrix.

Exercise 9.2.6. Suppose A, U, and B satisfy a relationship of the form

$$AU = UB + re_k^T$$

(cf. (9.2.3)), where U is $n \times k$ and has rank k, and r is some residual vector. Let $u_1, \ldots, u_k$ denote the columns of U. Show that B is upper Hessenberg if and only if

$$\operatorname{span}\{u_1,\ldots,u_j\} = \mathcal{K}_j(A,u_1), \qquad j = 1, \ \ldots, \ k.$$

Exercise 9.2.7. Proposition 9.2.3 summarizes the situation after k steps of a Krylov process. The eigenvalues of H_k are estimates of eigenvalues of A If (μ, y) is an eigenpair of H_k, then $(\mu, U_k y)$ is an estimate of an eigenpair of A. We know that if $h_{k+1,k} = 0$, then $\mathcal{R}(U_k)$ is invariant under A, and $(\mu, U_k y)$ is an exact eigenpair of A (Proposition 2.1.11). It is therefore natural to expect that if $\mathcal{R}(U_k)$ is close to invariant, then each $(\mu, U_k y)$ will be a good approximation to an eigenpair. In this exercise we investigate a more general question. If $\mathcal{R}(U_k)$ contains a good approximation to an eigenvector v of A, is there an eigenpair (μ, y) of H_k such that $(\mu, U_k y)$ approximates the eigenpair (λ, v) well? In particular, does one of $\mu_1, \ldots, \mu_k$, the eigenvalues of H_k approximate λ well?

Suppose (λ, v) is an eigenpair of A satisfying $\| v \| = 1$, and suppose there is a vector $w \in \mathcal{R}(U_k)$ such that $\| v - w \| = \epsilon \ll 1$. There is a unique $y \in \mathbb{R}^k$ such that $w = U_k y$.

(a) The case when the Krylov process produces orthonormal vectors $u_1, u_2, \ldots$ is easiest to analyze. Assume this to be the case for now. Show that under this assumption, $\| w \| = \| y \|$. Use (9.2.3) to show that $H_k = U_k^* A U_k$. Then show that $H_k y - y\lambda = U_k^*(Aw - w\lambda)$, and

$$\frac{\| H_k y - y\lambda \|}{\| y \|} \leq \frac{\| Aw - w\lambda \|}{\| w \|}.$$

(b) Show that $\| Aw - w\lambda \| \leq 2\| A \|\epsilon$. This result and the result of part (a) together show that the pair (λ, y) is an eigenpair of a matrix near H_k (Proposition 2.7.20). Now the question of whether (λ, y) really is close to an eigenpair of H_k is a question of conditioning. If all the eigenpairs of H_k are well conditioned then one of those pairs has to be close to (λ, y). Thus we cannot say for sure that (λ, y) will always be close to an eigenpair of H_k, but in the well-conditioned case it will. Therefore, in the well-conditioned case, one of $\mu_1, \ldots, \mu_k$, the eigenvalues of H_k, must be close to λ.

(c) Now consider the case in which the vectors produced by the Krylov process are not orthonormal. Then U_k does not have orthonormal columns, but it does have full rank. Thus U_k has a finite condition number $\kappa(U_k) = \sigma_1/\sigma_k$, the ratio of largest and smallest singular values. Show that

$$\frac{\| H_k y - y\lambda \|}{\| y \|} \leq \kappa(U_k) \frac{\| Aw - w\lambda \|}{\| w \|}.$$

Thus we draw the same conclusion as in parts (a) and (b), except that an extra factor $\kappa(U_k)$ has appeared. This is not a problem if $\kappa(U_k)$ is not too big.

9.3 Implicit Restarts

Suppose we have run a Krylov process k steps, so that we have $k+1$ vectors $u_1, \ldots, u_{k+1}$, and the equations (9.2.2) and (9.2.3) hold. Unless we were really lucky, we will not have found an invariant subspace, at least not on the first try. Now we would like to use the information we have obtained so far to try to find a better starting vector and restart the process, hoping to get closer to an invariant subspace on the next try.

As we have already remarked, Krylov subspace methods are good at approximating eigenvalues on the periphery of the spectrum. Thus they are good at computing the eigenvalues of largest modulus or of largest real part, or of smallest real part, for example. For this discussion let us suppose that we are looking for the m eigenvalues of largest modulus and the associated invariant subspace. Then we would like our new starting vector to be rich in its components in the direction of the associated eigenvectors.

We will describe two closely related implicit restart processes here. The second might be superior to the first, but even if it is, the first process is worth discussing because it yields important insights.

First method

The first method is modelled on Sorensen's implicitly restarted Arnoldi process [191], which is the basis of the popular ARPACK software [147]. Consider the equation

$$AU_k = U_k H_k + u_{k+1} h_{k+1,k} e_k^T \tag{9.3.1}$$

from Proposition 9.2.3. We know that if $h_{k+1,k} = 0$, then $\mathcal{R}(U_k)$ is invariant under A, and the eigenvalues of H_k are eigenvalues of A. Even if $h_{k+1,k} \neq 0$ (even if it not close to zero), some of the largest eigenvalues of H_k, may be good approximations to peripheral eigenvalues of A. Moreover, $\mathcal{R}(U_k)$ may contain good approximations to eigenvectors associated with peripheral eigenvalues. Since k is not too big, it is a simple matter to compute the eigenvalues of H_k by a GR algorithm. Let us call them $\mu_1, \mu_2, \ldots, \mu_k$ and order them so that $|\mu_1| \geq |\mu_2| \geq \cdots \geq |\mu_k|$.[3] The largest m values, $\mu_1, \ldots, \mu_m$, are the best approximations to the largest eigenvalues of A that we have so far.

[3]This assumes that we are looking for the eigenvalues of largest modulus. If we were looking for the eigenvalues with largest real part, for example, we would order them so that $\Re\mu_1 \geq \cdots \geq \Re\mu_k$.

Roughly speaking, our plan is to keep the portion of $\mathcal{R}(U_k)$ associated with $\mu_1, \ldots, \mu_m$ and discard the rest. To this end, we perform an iteration of an implicit GR algorithm of degree j on H_k, where $j = k - m$, using shifts $\nu_1, \ldots, \nu_j$ located in regions of the complex plane containing portions of the spectrum that we want to suppress. The most popular choice is $\nu_1 = \mu_{m+1}, \nu_2 = \mu_{m+2}, \ldots, \nu_j = \mu_k$. Thus we take as shifts the approximate eigenvalues that we wish to discard. We call this the *exact-shift* version of the implicit restart. In practice the GR iteration can be executed as a sequence of j iterations of degree 1 or $j/2$ iterations of degree 2, for example (Exercise 4.9.1 or Theorem 5.2.1). After the iterations, H_k will have been transformed to

$$\widehat{H}_k = G^{-1} H_k G, \tag{9.3.2}$$

where

$$p(H_k) = (H_k - \nu_1 I)(H_k - \nu_2 I) \cdots (H_k - \nu_j I) = GR. \tag{9.3.3}$$

If we now multiply (9.3.1) by G on the right and use (9.3.2) in the form $H_k G = G\widehat{H}_k$, we obtain

$$A\widehat{U}_k = \widehat{U}_k \widehat{H}_k + u_{k+1} h_{k+1,k} e_k^T G, \tag{9.3.4}$$

where

$$\widehat{U}_k = U_k G.$$

The restart is accomplished by discarding all but the first m columns of (9.3.4). Let $\widehat{U}_m$ denote the submatrix of $\widehat{U}_k$ consisting of the first m columns, and let $\widehat{H}_m$ denote the $m \times m$ principal submatrix of $\widehat{H}_k$ taken from the upper left-hand corner. Consider also the form of $e_k^T G$, which is the bottom row of G. It is a crucial fact that G is j-Hessenberg (Exercise 9.3.2). This implies that $e_k^T G$ has zeros in its first $k - j - 1 = m - 1$ positions. Keep in mind also that $\widehat{H}_k$ is upper Hessenberg. Let $\tilde{u}$ denote column $m + 1$ of $\widehat{U}_k$ and $\tilde{h}$ the $(m + 1, m)$ entry of $\widehat{H}_k$. Then, if we extract the first m columns of (9.3.4), we obtain

$$A\widehat{U}_m = \widehat{U}_m \widehat{H}_m + \tilde{u}\tilde{h} e_m^T + u_{k+1} h_{k+1,k} g_{km} e_m^T. \tag{9.3.5}$$

Now define a vector $\widehat{u}_{m+1}$ and scalar $\widehat{h}_{m+1,m}$ so that[4]

$$\widehat{u}_{m+1} \widehat{h}_{m+1,m} = \tilde{u}\tilde{h} + u_{k+1} h_{k+1,k} g_{km}.$$

Then (9.3.5) becomes

$$A\widehat{U}_m = \widehat{U}_m \widehat{H}_m + \widehat{u}_{m+1} \widehat{h}_{m+1,m} e_m^T. \tag{9.3.6}$$

In the unlikely event that $\widehat{h}_{m+1,m} = 0$, the columns of $\widehat{U}_m$ span an invariant subspace. If this is not the case, we start over, but not from scratch. Equation (9.3.6) is analogous to (9.3.1), except that it has only m columns. It could have been arrived at by m steps of a Krylov process, starting from a new starting vector $\widehat{x} = c\widehat{u}_1$. The implicit restart procedure does not start anew with $\widehat{x}$; it starts from (9.3.6), which can also be written as

$$A\widehat{U}_m = \widehat{U}_{m+1} \widehat{H}_{m+1,m},$$

in analogy with (9.2.2). The Krylov process is used to generate $\widehat{u}_{m+2}, \widehat{u}_{m+3}, \ldots, \widehat{u}_{k+1}$ satisfying

$$A\widehat{U}_k = \widehat{U}_{k+1} \widehat{H}_{k+1,k}.$$

[4]The vector $\widehat{u}_{m+1}$ and scalar $\widehat{h}_{m+1,m}$ are not uniquely determined. They would be if we were to insist that, for example, $\|\widehat{u}_{m+1}\|_2 = 1$ and $\widehat{h}_{m+1,m} > 0$.

Then another implicit restart can be done and a new Krylov subspace generated from a new starting vector. The restarts can be repeated until the desired invariant subspace is obtained.

Why the method works

The shifts $\nu_1, \ldots, \nu_j$ were used in the GR iteration that determined the transforming matrix G. Starting from (9.3.1), subtract $\nu_1 U_k$ from both sides to obtain

$$(A - \nu_1 I)U_k = U_k(H_k - \nu_1 I) + E_1, \tag{9.3.7}$$

where $E_1 = u_{k+1}h_{k+1,k}e_k^T$ is a matrix that is zero except in its kth and last column. A similar equation holds for shifts $\nu_2, \ldots, \nu_k$. Now multiply (9.3.7) on the left by $A - \nu_2 I$ and use the ν_2 analogue of (9.3.7) to further process the resulting term involving $(A - \nu_2 I)U_k$. The result is

$$(A - \nu_2 I)(A - \nu_1 I)U_k = U_k(H_k - \nu_2 I)(H_k - \nu_1 I) + E_2, \tag{9.3.8}$$

where $E_2 = E_1(H_k - \nu_1 I) + (A - \nu_2 I)E_1$ is a matrix that is zero except in its last two columns. Applying factors $A - \nu_3 I$, $A - \nu_4 I$, and so on, we easily find that

$$p(A)U_k = U_k p(H_k) + E_j, \tag{9.3.9}$$

Where E_j is zero except in the last j columns. In other words, the first m columns of E_j are zero. Here $p(z) = (z - \nu_1)\cdots(z - \nu_j)$, as before. Since $p(H_k) = GR$ and $U_k G = \widehat{U}_k$, we can rewrite (9.3.9) as

$$p(A)U_k = \widehat{U}_k R + E_j.$$

If we then extract the first m columns from this equation, bearing in mind that R is upper triangular, we obtain

$$p(A)U_m = \widehat{U}_m R_m, \tag{9.3.10}$$

where R_m is the $m \times m$ submatrix of R from the upper left-hand corner. This equation tells the whole story. If we focus on the first column, we find that

$$\widehat{x} = \alpha\, p(A)x$$

for some constant α. The new starting vector is essentially $p(A)$ times the old one. Thus the restart effects an iteration of the power method driven by $p(A)$. For ease of discussion, let us suppose A is semisimple with linearly independent eigenvectors $v_1, \ldots, v_n$ and eigenvalues $\lambda_1, \ldots, \lambda_n$. Then

$$x = c_1 v_1 + c_2 v_2 + \cdots c_n v_n.$$

For some (unknown) constants $c_1, \ldots, c_n$. With probability 1, all c_i are nonzero. The corresponding expansion of $\widehat{x}$ is

$$\alpha^{-1}\widehat{x} = c_1 p(\lambda_1)v_1 + c_2 p(\lambda_2)v_2 + \cdots + c_n p(\lambda_n)v_n.$$

The polynomial $p(z)$ is small in magnitude for z near the shifts $\nu_1, \ldots, \nu_j$ and large for z away from the shifts. Therefore, in the transformation from x to $\widehat{x}$, the components corresponding to eigenvalues far from the shifts become enriched relative to those that are near the shifts. The peripheral eigenvalues furthest from the shifts receive the most enrichment. Thus the restart process reinforces the natural tendency of the Krylov process to bring out the peripheral eigenvalues. After a number of restarts we will have arrived at a starting vector

$$x \approx b_1 v_1 + \cdots + b_m v_m,$$

and we will have found the invariant subspace associated with the m eigenvalues of largest magnitude.

The implicit restart process uses different shifts on each restart. This makes the method adaptive and is essential to its success, but it also makes the analysis of convergence more difficult than it would be for a stationary power method. However, there has been some progress toward a rigorous convergence theory [25, 26].

Equation (9.3.10) shows that the implicit restart doesn't just do simple power iterations. It does nested subspace iterations. Bearing in mind that R_m is upper triangular, we find that

$$p(A)\text{span}\{u_1, \ldots, u_i\} = \text{span}\{\widehat{u}_1, \ldots, \widehat{u}_i\}, \qquad i = 1, \ldots, m. \tag{9.3.11}$$

These relationships are actually an automatic consequence of the fact that the subspaces in question are Krylov subspaces, as Exercise 9.3.5 shows.

Exact shifts

If exact shifts $\nu_1 = \mu_{m+1}, \ldots, \nu_j = \mu_n$ are used in the process, then in $\widehat{H}_k$ we have $\widehat{h}_{m+1,m} = 0$ by Theorem 4.6.1. Letting $\widetilde{G}$ denote the submatrix of G consisting of the first m columns, we see that $\mathcal{R}(\widetilde{G})$ is the invariant subspace of H_k corresponding to eigenvalues $\mu_1, \ldots, \mu_m$. Note that $\widehat{U}_m = U_k\widetilde{G}$, so $\mathcal{R}(\widehat{U}_m)$ is the subspace of $\mathcal{R}(U_k)$ corresponding to the eigenvalue approximations $\mu_1, \ldots, \mu_m$.

Second method

Our second implicit restart method is modeled on the thick-restart Lanczos method of Wu and Simon [233] and Stewart's Krylov-Schur algorithm [198]. After k steps of a Krylov process we have

$$AU_k = U_k H_k + u_{k+1} h_{k+1,k} e_k^T. \tag{9.3.12}$$

Compute a similarity transformation $H_k = STS^{-1}$ with T block triangular:

$$T = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix},$$

where T_{11} has eigenvalues $\mu_1, \ldots, \mu_m$ (the approximate eigenvalues we want to keep) and T_{22} has eigenvalues $\mu_{m+1}, \ldots, \mu_k$ (the approximate eigenvalues we want to discard). This can be done by a GR algorithm in conjunction with a block swapping procedure as described in § 4.8 to get the eigenvalues sorted appropriately. Multiply (9.3.12) on the right by S to obtain

$$A\widetilde{U}_k = \widetilde{U}_k T + u_{k+1}\widetilde{z}^T, \tag{9.3.13}$$

where $\widetilde{U}_k = U_k S$ and $\widetilde{z}^T = h_{k+1,k} e_k^T S$. The restart is effected by retaining only the first m columns of (9.3.13):

$$A\widetilde{U}_m = \widetilde{U}_m T_{11} + u_{k+1} z^T, \tag{9.3.14}$$

where $\widetilde{U}_m$ consists of the first m columns of $\widetilde{U}_k$, and z consists of the first m entries of $\widetilde{z}$.

Let $\widetilde{S}$ denote the submatrix of S consisting of the first m columns. The similarity transformation $H_k S = ST$ implies that $\mathcal{R}(\widetilde{S})$ is the invariant subspace of H_k corresponding to eigenvalues $\mu_1, \ldots, \mu_m$. This is the same as the space $\mathcal{R}(\widetilde{G})$ in the exact-shift variant of the first method. Since $\widetilde{U}_m = U_k\widetilde{S}$, we see that $\mathcal{R}(\widetilde{U}_m)$ is the same as the space $\mathcal{R}(\widehat{U}_m)$ in the exact-shift variant of the first method. Thus the two methods retain the same space. This implies equivalence of the methods, as we will show at the end of the section.

In (9.3.14) we cannot claim that the first $m-1$ entries of z are zero as we could with $e_k^T G$ in the first method. In order to resume the Krylov process, we must return our configuration to a form like

$$A\widehat{U}_m = \widehat{U}_m\widehat{H}_m + \widehat{u}_{m+1}\widehat{h}_{m+1,m}e_m^T,$$

in which z^T has been transformed to a multiple of e_m^T.[5] Let V_1 be an elimination matrix such that $z^T V_1^{-1} = \alpha e_m^T$ for some α. If we multiply (9.3.14) on the right by V_1^{-1}, we transform it to $u_{k+1}\alpha e_m^T = \widehat{u}_{m+1}\widehat{h}_{m+1,m}e_m^T$, where $\widehat{u}_{m+1} = u_{k+1}$ and $\widehat{h}_{m+1,m} = \alpha$.

The right multiplication by V_1^{-1} also transforms T_{11} to $V_1 T_{11} V_1^{-1}$. This is a full matrix, and we must reduce it to upper Hessenberg form. We do so, not column by column from left to right, but row by row from bottom to top, as described in Section 3.4. Elimination matrices $V_2 = \operatorname{diag}\{\widetilde{V}_2, 1\}$, $V_3 = \operatorname{diag}\{\widetilde{V}_3, 1, 1\}$, $\ldots V_{m-1}$ introduce zeros in rows $m-1$, $m-2, \ldots, 3$, so that the matrix $\widehat{H}_m = V_{m-1}\cdots V_1 T_{11} V_1^{-1}\cdots V_{m-1}^{-1}$ is upper Hessenberg. Let $V = V_{m-1}V_{m-2}\cdots V_1$. Since $z^T V_1^{-1} = \alpha e_m$, and $e_m^T V_j^{-1} = e_m^T$ for $j = 2, \ldots, m-1$, we also have $z^T V^{-1} = \alpha e_m^T$. In other words, the last row of V is proportional to z^T. This is an instance of Theorem 3.4.1, in which the roles of A, x, and G are played by T_{11}, z, and V, respectively.

Let $\widehat{U}_m = \widetilde{U}_m V^{-1}$, $\widehat{u}_{m+1} = u_{k+1}$, and $\widehat{h}_{m+1,m} = \alpha$. Multiplying (9.3.14) by V^{-1} on the right and using the fact that $z^T V^{-1} = \alpha e_m^T$, we obtain

$$A\widehat{U}_m = \widehat{U}_m\widehat{H}_m + \widehat{u}_{m+1}\widehat{h}_{m+1,m}e_m^T, \tag{9.3.15}$$

as desired. The columns of $\widehat{U}_m$ span the same subspace as do those of $\widetilde{U}_m$. This is the space corresponding to the eigenvalue estimates $\mu_1, \ldots, \mu_m$. Now that we have restored the Hessenberg form, the restart is complete. Now, starting from (9.3.15), which can also be written as

$$A\widehat{U}_m = \widehat{U}_{m+1}\widehat{H}_{m+1,m},$$

we can use the Krylov process to generate $\widehat{u}_{m+2}, \widehat{u}_{m+3}, \ldots, \widehat{u}_{k+1}$ satisfying

$$A\widehat{U}_k = \widehat{U}_{k+1}\widehat{H}_{k+1,k}.$$

From here another implicit restart can be done. The restarts can be repeated until the desired invariant subspace is obtained.

[5] Actually it is possible to defer this transformation, as is done in both [233] and [198]. However, there seems to be no advantage to putting it off. Nothing is gained in terms of efficiency, and for the purpose of understanding the algorithm it is surely better to do the transformation right away.

Convergence and locking

We still have to describe the process by which convergence is detected. A natural opportunity to check for convergence occurs at (9.3.14), which we rewrite as

$$A\widetilde{U}_m = \widetilde{U}_m B + u_{k+1} z^T, \tag{9.3.16}$$

replacing the symbol T_{11} by B. The matrix B is a submatrix of T, which was computed using some kind of GR algorithm. Therefore B will normally be quasitriangular. For the sake of ease of discussion, let us assume at first that it is upper triangular.

Convergence can be checked by examining the entries of the vector z. If the first j entries of z are zero (or within a prescribed tolerance of zero), we can partition (9.3.16) as

$$A\begin{bmatrix} \widetilde{U}_{m1} & \widetilde{U}_{m2} \end{bmatrix} = \begin{bmatrix} \widetilde{U}_{m1} & \widetilde{U}_{m2} \end{bmatrix}\begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix} + u_{k+1}\begin{bmatrix} 0 & z_2^T \end{bmatrix}, \tag{9.3.17}$$

where $\widetilde{U}_{m1}$ is $n \times j$ and B_{11} is $j \times j$. The partition of B is valid because B is upper triangular. Equation (9.3.17) implies that $A\widetilde{U}_{m1} = \widetilde{U}_{m1}B_{11}$, whence $\mathcal{R}(\widetilde{U}_{m1})$ is invariant under A, and the eigenvalues of B_{11} are eigenvalues of A. These are typically (though not necessarily) $\mu_1, \ldots, \mu_j$.

We have assumed that B is upper triangular, which guarantees that no matter what the value of j is, the resulting partition of B has the form

$$\begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix}.$$

Clearly we can make the same argument for quasitriangular or more general block triangular B, so long as we refrain from using values of j that split a block.

Suppose we have found an invariant subspace of dimension j, and we have the configuration (9.3.17). The next step in the process, as described above, is to build an elimination matrix V_1 such that $z^T V_1 = \alpha e_m^T$. But now, since the first j entries of z are already zero, we can take V_1 to have the form $V_1 = \operatorname{diag}\{I_j, \widehat{V}_1\}$. Then the operation $T_{11} = B \to BV_1$ does not touch the first j columns of B, nor does the operation $BV_1 \to V_1^{-1}BV_1 = M$ touch the first j rows. Thus the resulting matrix M has the form

$$M = \begin{bmatrix} B_{11} & M_{12} \\ 0 & M_{22} \end{bmatrix};$$

only the submatrix M_{22} is full and needs to be returned to Hessenberg form. This implies that the subsequent transformations $V_2, V_3, \ldots$, which return M to upper Hessenberg form, all have the form $\operatorname{diag}\{I_j, \widehat{V}_i\}$, and so does the product V. Thus the transformation $\widetilde{U}_m \to \widetilde{U}_m V = \widehat{U}_m$ does not touch the first j columns, the submatrix $\widetilde{U}_{m1}$. These columns are now locked in and will remain untouched for the remainder of the computation. In the equation

$$A\widehat{U}_m = \widehat{U}_m \widehat{H}_m + \widehat{u}_{m+1}\widehat{h}_{m+1,m} e_m^T,$$

the first j columns of $\widehat{U}_m$ are $\widetilde{U}_{m1}$, which span an invariant subspace. The $(j+1, j)$ entry of $\widehat{H}_m$ is zero, so $\widehat{H}_m$ is block triangular. Its upper left-hand $j \times j$ block is B_{11}, which houses j eigenvalues. B_{11} is also locked in and will remain unchanged for the rest of the computation.

Subsequent restarts will operate only on the "unconverged" parts of $\widehat{U}_m$ and $\widehat{H}_m$. Each restart has a chance to set more entries of z to zero, thereby enlarging the converged invariant subspace and causing more vectors to be locked in. Once a sufficiently large invariant subspace has been found, the computation is terminated.

Equivalence of the two methods

Here we show that the second method is equivalent to the exact-shift variant of the first method.[6] The key is the following simple Lemma, which is proved in Exercise 9.3.6.

Lemma 9.3.1. *Let $\mathcal{S} = \mathcal{K}_m(A, \widehat{u}) = \mathcal{K}_m(A, \check{u})$, and suppose $\mathcal{S}$ is not invariant under A. Then $\check{u} = \alpha\widehat{u}$ for some nonzero scalar α.*

Assume that no convergence or locking has taken place so far. In order to compare the two methods we rewrite (9.3.15) as

$$A\check{U}_m = \check{U}_m\check{H}_m + \check{u}_{m+1}\check{h}_{m+1,m}e_m^T. \tag{9.3.18}$$

We have observed that $\mathcal{R}(\check{U}_m) = \mathcal{R}(\widetilde{U}_m)$ and that $\mathcal{R}(\widetilde{U}_m) = \mathcal{R}(\widehat{U}_m)$ if exact shifts are used in the first method, where $\widehat{U}_m$ is as in (9.3.6). Since no convergence or locking has taken place, the matrix $\widehat{H}_m$ in (9.3.6) is properly upper Hessenberg, and $\widehat{h}_{m+1,m} \neq 0$. This implies by Proposition 9.2.4 that $\operatorname{span}\{\widehat{u}_1, \ldots, \widehat{u}_j\} = \mathcal{K}_j(A, \widehat{u}_1)$ for $j = 1, \ldots, m$. Moreover, none of these spaces is invariant. Applying the same reasoning to (9.3.18), we find that $\operatorname{span}\{\check{u}_1, \ldots, \check{u}_j\} = \mathcal{K}_j(A, \check{u}_1)$ for $j = 1, \ldots, m$, and none of these spaces are invariant. We thus have $\mathcal{K}_m(A, \widehat{u}_1) = \mathcal{R}(\widehat{U}_m) = \mathcal{R}(\check{U}_m) = \mathcal{K}_m(A, \check{u}_1)$, and this space is not invariant. Thus by Lemma 9.3.1 $\widehat{u}_1$ and $\check{u}_1$ are proportional, and

$$\operatorname{span}\{\widehat{u}_1, \ldots, \widehat{u}_i\} = \mathcal{K}_i(A, \widehat{u}_1) = \mathcal{K}_i(A, \check{u}_1) = \operatorname{span}\{\check{u}_1, \ldots, \check{u}_i\}, \qquad i = 1, \ldots, m.$$

Thus the second restart method produces exactly the same sequence of Krylov subspaces as the first method with exact shifts.

For this analysis we have assumed that no convergence or locking have taken place. A more careful analysis shows that the equivalence continues to hold even after some vectors have been locked in. See Exercise 9.3.7.

Notes and references

Explicit restarts of Krylov processes were advocated by Saad [184] in 1984. The first implicit restarting method was due to Sorensen [191] in 1992. It is the basis of our first restart method, and it is the algorithm that underlies ARPACK [147]. The thick restart Lanczos process was introduced by Wu and Simon [233] for the symmetric case. (Precursors were [157] and [192].) Subsequently Stewart [198] introduced the Krylov-Schur algorithm, which reduces to the thick restart method in the symmetric case. These algorithms are the basis of our second restart method.

[6] I thank my former student Roden David for helping me understand this equivalence better.

Exercises

Exercise 9.3.1.

(a) Verify equation (9.3.4).

(b) Verify equation (9.3.5).

Exercise 9.3.2. Recall that a matrix G is m-Hessenberg if $g_{ij} = 0$ whenever $i > j+m$. Suppose H is upper Hessenberg, and $p(H) = GR$, where p is a polynomial of degree j, and R is upper triangular.

(a) Under the assumption that R is nonsingular, prove that G is j-Hessenberg. (See Exercise 4.5.5.)

(b) Assuming R is singular, use Theorem 4.6.4 to prove that G is j-Hessenberg.

Exercise 9.3.3.

(a) Verify equation (9.3.8).

(b) Prove (9.3.9) by induction.

Exercise 9.3.4. Suppose $v_1, \ldots, v_m$ are linearly independent eigenvectors of A, and let $x = c_1v_1 + \ldots + c_mv_m$, where $c_1, \ldots c_m$ are all nonzero. Prove that $\mathcal{K}_m(A, x) = \text{span}\{v_1, \ldots, v_m\}$. Thus the Krylov subspace is invariant.

Exercise 9.3.5.. Show that if $\hat{x} = p(A)x$ for some polynomial p, then

$$p(A)\mathcal{K}_i(A, x) = \mathcal{K}_i(A, \hat{x})$$

for $i = 0, 1, \ldots, n$.

Exercise 9.3.6. This exercise works out the proof of Lemma 9.3.1.

(a) Under the hypotheses of Lemma 9.3.1, show that $\hat{u}$, $A\hat{u}$, $\ldots$, $A^{m-1}\hat{u}$ are linearly independent. Deduce that there are unique $c_1, \ldots, c_m$, not all zero, such that

$$\check{u} = c_1\hat{u} + c_2A\hat{u} + \cdots + c_mA^{m-1}\hat{u}. \tag{9.3.19}$$

(b) Show that $A^m\hat{u} \notin \mathcal{S}$.

(c) Let r be the largest integer such that $c_r \neq 0$ in (9.3.19). Note that if $r > 1$, then $A^{m-r+1}\check{u} \in \mathcal{S}$. But on the other hand,

$$A^{m-r+1}\check{u} = c_1A^{m-r+1}\hat{u} + c_2A^{m-r+2}\hat{u} + \cdots + c_rA^m\hat{u}.$$

Deduce that $A^m\hat{u} \in \mathcal{S}$, in contradiction to the result of part (b). Thus the assumption $r > 1$ must be false. Deduce that $r = 1$ and the lemma is true.

Exercise 9.3.7. Consider equation (9.3.18) in the case when exactly s converged vectors have been locked in. Then $\check{h}_{s+1,s} = 0$, and $\check{h}_{j+1,j} \neq 0$ for $j = s+1, \ldots, m$. The space $\text{span}\left\{\check{u}_1, \ldots, \check{u}_s\right\}$ is invariant.

(a) Show that under these conditions

$$\operatorname{span}\{\check{u}_1, \ldots, \check{u}_j\} = \operatorname{span}\{\check{u}_1, \ldots, \check{u}_s\} \oplus \mathcal{K}_{j-s}(A, \check{u}_{s+1})$$

for $j = s + 1, \ldots, m$.

(b) Show that $\operatorname{span}\{\check{u}_1, \ldots, \check{u}_m\}$ is not invariant under A.

(c) Prove the following extension of Lemma 9.3.1: Suppose the space

$$\operatorname{span}\{\check{u}_1, \ldots, \check{u}_s\} = \operatorname{span}\{\widehat{u}_1, \ldots, \widehat{u}_s\}$$

is invariant under A, the space

$$\begin{aligned}\operatorname{span}\{\check{u}_1, \ldots, \check{u}_m\} &= \operatorname{span}\{\check{u}_1, \ldots, \check{u}_s\} \oplus \mathcal{K}_{m-s}(A, \check{u}_{s+1}) \\ &= \operatorname{span}\{\widehat{u}_1, \ldots, \widehat{u}_s\} \oplus \mathcal{K}_{m-s}(A, \widehat{u}_{s+1}) \\ &= \operatorname{span}\{\widehat{u}_1, \ldots, \widehat{u}_m\}\end{aligned}$$

is not invariant under A. Then there is a nonzero constant α and a $v \in \operatorname{span}\{\check{u}_1, \ldots, \check{u}_s\}$ such that

$$\check{u}_{s+1} = \alpha \widehat{u}_{s+1} + v.$$

(d) Using (9.3.18) and (9.3.6) with $\check{h}_{s+1,s} = \widehat{h}_{s+1,s} = 0$ and $\check{u}_i = \widehat{u}_i$ for $i = 1, \ldots, s$ (locked-in vectors untouched on this iteration) and the fact that $\mathcal{R}(\check{U}_m) = \mathcal{R}(\widehat{U}_m)$, show that

$$\operatorname{span}\{\check{u}_1, \ldots, \check{u}_j\} = \operatorname{span}\{\widehat{u}_1, \ldots, \widehat{u}_j\}, \qquad j = s + 1, \ldots, m.$$

Thus the two restart methods produce the same sequence of subspaces.

9.4 The Arnoldi and Symmetric Lanczos Processes

The Arnoldi process [11] is by far the most widely used Krylov process. Given $A \in \mathbb{C}^{n\times n}$ and nonzero starting vector $x \in \mathbb{C}^n$, the Arnoldi process begins with $u_1 = cx$, where $c = 1/\| x \|$. Given orthonormal vectors $u_1, \ldots, u_j$, it produces u_{j+1} as follows.

$$h_{ij} = \langle Au_j, u_i \rangle = u_i^* A u_j, \qquad i = 1, \ldots, j, \tag{9.4.1}$$

$$\widehat{u}_{j+1} = Au_j - \sum_{i=1}^{j} u_i h_{ij}, \tag{9.4.2}$$

$$h_{j+1,j} = \| \widehat{u}_{j+1} \|, \tag{9.4.3}$$

and if $h_{j+1,j} \neq 0$,

$$u_{j+1} = \widehat{u}_{j+1} / h_{j+1,j}. \tag{9.4.4}$$

It is an easy matter to show that $\widehat{u}_{j+1}$ is orthogonal to $u_1, \ldots, u_j$ if and only if the coefficients $h_{1j}, \ldots, h_{jj}$ are defined as shown in (9.4.1) (See Exercise 9.4.7). If $Au_j \notin \operatorname{span}\{u_1, \ldots, u_j\}$, then certainly $\widehat{u}_{j+1} \neq 0$. Conversely, if $Au_j \in \operatorname{span}\{u_1, \ldots, u_j\}$, the choice of coefficients

(9.4.1) forces $\widehat{u}_{j+1} = 0$. In this case $\operatorname{span}\{u_1, \ldots, u_j\} = \mathcal{K}_j(A, x)$ is invariant under A, and the Arnoldi process terminates. Thus the Arnoldi process is a Krylov process as defined in Section 9.2.

Letting $U_j = \begin{bmatrix} u_1 & \cdots & u_j \end{bmatrix}$, we can write (9.4.1) and (9.4.2) in matrix-vector form as

$$h_{1:j,j} = U_j^* A u_j \tag{9.4.5}$$

and

$$\widehat{u}_{j+1} = A u_j - U_j h_{1:j,j}, \tag{9.4.6}$$

respectively. The Arnoldi process is an instance of the Gram-Schmidt process (Exercise 1.5.3), and as such it is vulnerable to roundoff errors [221, §3.4]. Over the course of several steps executed in floating point arithmetic, the orthogonality of the vectors will steadily deteriorate. A simple and practical remedy is to do the orthogonalization twice. One might even consider doing three or more orthogonalizations, but the unpublished advice of Kahan is that "twice is enough" [169]. This has been confirmed in practice and by theoretical analysis [95]. See also [2], [62], and [116]. After two orthogonalizations, the vector $\widehat{u}_{j+1}$ will be orthogonal to $u_1, \ldots, u_j$ to working precision. We do not mind the added cost of an additional orthogonalization because we never carry the Arnoldi run very far; we always keep the total number of vectors small. The following algorithm takes the need for reorthogonalization into account.

Arnoldi process with reorthogonalization.

```
$u_1 \leftarrow x/\|x\|$
$U_1 \leftarrow [u_1]$
for $j = 1, 2, 3, \ldots$
    $u_{j+1} \leftarrow A u_j$
    $h_{1:j,j} \leftarrow U_j^* u_{j+1}$
    $u_{j+1} \leftarrow u_{j+1} - U_j h_{1:j,j}$   (orthogonalize)
    $\delta_{1:j} \leftarrow U_j^* u_{j+1}$
    $u_{j+1} \leftarrow u_{j+1} - U_j \delta_{1:j}$   (reorthogonalize)          (9.4.7)
    $h_{1:j,j} \leftarrow h_{1:j,j} + \delta_{1:j}$
    $h_{j+1,j} \leftarrow \|u_{j+1}\|$
    if $h_{j+1,j} = 0$
        set flag (span$\{u_1, \ldots, u_j\}$ is invariant)
        exit
    $u_{j+1} \leftarrow u_{j+1}/h_{j+1,j}$
    $U_{j+1} \leftarrow \begin{bmatrix} U_j & u_{j+1} \end{bmatrix}$
```

Equations (9.4.2) and (9.4.4)together imply

$$A u_j = \sum_{i=1}^{j+1} u_i h_{ij},$$

and after k steps of the Arnoldi process, we have

$$A U_k = U_k H_k + u_{k+1} h_{k+1,k} e_k^T, \tag{9.4.8}$$

as explained in Proposition 9.2.3. The only thing we need to add to Proposition 9.2.3 in the Arnoldi case is that now the columns of U_k are orthonormal. Multiplying (9.4.8) by U_k^* on the left, we find that

$$H_k = U_k^* A U_k.$$

The eigenvalues of H_k are estimates of eigenvalues of A. In this case we call them *Ritz values* (Exercise 9.4.8).

Implicit Restarts

Since we normally do not have enough memory to store a great number of vectors, we normally have to do restarts. A second motive for restarting is that the computational cost of (9.4.1) and (9.4.2) increases linearly with j; each step takes more time than the previous one.

The two implicit restart methods described in Section 9.2 can both be applied to the Arnoldi process. The only caveat is that all transformations that are performed must be unitary in order to preserve the orthonormality of the vectors. Thus, for example, the filtering of approximate eigenvalues (now called Ritz values) must be done by the QR algorithm, not just any GR algorithm. In this context, the first restart method becomes the implicitly restarted Arnoldi method of Sorensen [147, 191], and the second method becomes the Krylov-Schur algorithm of Stewart [198].

The symmetric Lanczos process

Now assume A is Hermitian ($A = A^*$). In this case the Arnoldi process undergoes significant simplification in principle and is called the *symmetric Lanczos process* [143]. The upper-Hessenberg matrix $H_k = U_k^* A U_k$ is Hermitian because A is, so it is tridiagonal. The main-diagonal entries of H_k are, of course, real. Moreover the definition (9.4.3) implies that the subdiagonal entries are real as well. Therefore H_k is a real symmetric tridiagonal matrix.

The fact that $h_{ij} = 0$ for $i < j - 1$ implies that most of the terms in the sum in (9.4.2) are zero. If we introduce new notation $\alpha_j = h_{jj}$ and $\beta_j = h_{j+1,j} = h_{j,j+1}$, (9.4.2) becomes

$$u_{j+1}\beta_j = Au_j - u_j\alpha_j - u_{j-1}\beta_{j-1}, \tag{9.4.9}$$

with the understanding that $u_0\beta_0 = 0$. Based on this three-term recurrence we have the following algorithm.

Symmetric Lanczos process without reorthogonalization.

```
$u_1 \leftarrow x/\|x\|$
for $j = 1, 2, 3, \ldots$
    $u_{j+1} \leftarrow Au_j$
    $\alpha_j \leftarrow u_j^* u_{j+1}$
    $u_{j+1} \leftarrow u_{j+1} - u_j \alpha_j$
    if $j > 1$
        $u_{j+1} \leftarrow u_{j+1} - u_{j-1}\beta_{j-1}$
    $\beta_j \leftarrow \|u_{j+1}\|$
    if $\beta_j = 0$
        set flag (span$\{u_1, \ldots, u_j\}$ is invariant)
        exit
    $u_{j+1} \leftarrow u_{j+1}/\beta_j$
```
(9.4.10)

The appeal of this algorithm is that, if we are willing to forego reorthogonalization, we can run it to much higher values of j. If we are just after eigenvalues, we do not need to save all the vectors that we generate. To compute u_{j+1}, we just need u_j and u_{j-1}. Moreover, the computational cost of each step is small and does not grow with j. The most expensive aspect is the multiplication of A by u_j. At any given step, the eigenvalues of the tridiagonal matrix H_j are estimates of eigenvalues of A. These can be computed cheaply by the QR algorithm or a number of other techniques.

The drawback of the procedure is that in floating-point arithmetic the orthogonality is gradually lost. The consequences are interesting. Once an eigenvalue has been found, the algorithm "forgets" that it has found it and produces another one or more "ghost" copies. The use of the symmetric Lanczos algorithm without reorthogonalization was studied by Paige [161, 162, 163] and Cullum and Willoughby [59]. See also [154, 169]. Several partial reorthogonalization schemes, in which v_{j+1} is reorthogonalized only against selected v_i, have been proposed [169, 175, 189]. We will not pursue this idea.

There are interesting relationships between the symmetric Lanczos process, algorithms for generating orthogonal polynomials, and numerical integration formulas. See [215] and the references therein.

Preserving orthogonality

The surest way to preserve orthogonality in the symmetric Lanczos process is to save all the vectors and orthogonalize against them. The runs are kept short, and restarts are done frequently. In this mode the symmetric Lanczos process differs little from the Arnoldi process.

Symmetric Lanczos process with reorthogonalization.

```
$u_1 \leftarrow x/\|x\|$
$U_1 \leftarrow [u_1]$
for $j = 1, 2, 3, \ldots$
    $u_{j+1} \leftarrow Au_j$
    $\alpha_j \leftarrow u_j^* u_{j+1}$
    $u_{j+1} \leftarrow u_{j+1} - u_j \alpha_j$
    if $j > 1$
        $u_{j+1} \leftarrow u_{j+1} - u_{j-1}\beta_{j-1}$
    $\delta_{1:j} \leftarrow U_j^* u_{j+1}$                                  (9.4.11)
    $u_{j+1} \leftarrow u_{j+1} - U_j \delta_{1:j}$   (reorthogonalize)
    $\alpha_j \leftarrow \alpha_j + \delta_j$
    $\beta_j \leftarrow \|u_{j+1}\|$
    if $\beta_j = 0$
        set flag (span$\{u_1, \ldots, u_j\}$ is invariant)
        exit
    $u_{j+1} \leftarrow u_{j+1}/\beta_j$
    $U_{j+1} \leftarrow [\ U_j \quad u_{j+1}\ ]$
```

One might well ask why we should bother to use the symmetric Lanczos process in this mode. Why not just use the Arnoldi process? The answer is that the Lanczos process preserves the structure. The matrix

$$H_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix}$$

that is produced is truly symmetric. When a restart is done, it can be done by the symmetric QR algorithm, which keeps the structure intact. The eigenvalues that are computed are real; there is no danger of a repeated real eigenvalue being confused for a pair of complex eigenvalues because of roundoff errors. The eigenvectors that are computed are orthogonal to working precision. These are properties that the eigensystem of a Hermitian matrix ought to have, but they can sometimes be lost if the structure-ignoring Arnoldi process is used. By the way, we do not feel overburdened by having to keep all the vectors around, since we need them anyway if we want to compute eigenvectors.

In this chapter we will meet many examples of structures (e.g. skew symmetric (Exercise 9.4.11), unitary, Hamiltonian, symplectic) for which there exist Krylov processes that have short recurrences. In no case are we able to exploit the short recurrence if we insist on preserving orthogonality. The methods are worth using nevertheless, simply because they preserve the structure.

Unitary Matrices and Cayley Transforms

Let U be a unitary matrix. The eigenvalue problem for U can be transformed to a Hermitian eigenvalue problem via a Cayley transform. This is an excellent option if U is not too large for the shift-and-invert methodology to be used [63].

All of the eigenvalues of U lie on the unit circle. Let τ be a target value on the unit circle that is not an eigenvalue of U, and suppose we wish to find the eigenpairs of U associated with the eigenvalues that are closest to τ. The matrix

$$A = i(U + \tau I)(U - \tau I)^{-1} \tag{9.4.12}$$

is a Hermitian matrix whose largest eigenvalues correspond to the eigenvalues of U that are closest to τ (Exercise 9.4.10). Therefore the desired eigenpairs of U can be found quickly by applying the symmetric Lanczos process to A. Of course, we do not form the matrix A explicitly. We just need an LU factorization of $U - \tau I$, in order to effect the transformations $x \to Ax$. Thus this method is feasible whenever the shift-and-invert strategy is.

We refer to the transformation in (9.4.12) as a *Cayley transform*, and we call A a *Cayley transform* of U. We also refer to the inverse operation

$$U = \tau(A + iI)(A - iI)^{-1}$$

as a Cayley transform.

Exercises

Exercise 9.4.1.

(a) Using (9.4.7) as a guide, write MATLAB code that implements the Arnoldi process with reorthogonalization. (Alternatively, download the file `arnoldi.m` from the website.[7])

(b) MATLAB supplies a sparse demonstration matrix called `west0479`.) Try the following commands in MATLAB

```
>> load west0479
>> A = west0479;
>> issparse(A)
>> size(A)
>> nnz(A) % number of nonzeros
>> spy(A)
```

Since this matrix is really not all that large, you can compute its "exact" eigenvalues using MATLAB's `eig` function: `lam = eig(full(A));` and you can plot them using `plot(real(lam),imag(lam),'r+')`, for example. Notice that this matrix has some outlying eigenvalues.

(c) Run 30 steps of the Arnoldi process, starting with a random initial vector. Compute $\| AU_j - U_{j+1}H_{j+1,j} \|$ (with $j = 30$) to check the correctness of your code. Also check the orthonormality of the columns of U by computing $\| I_{j+1} - U_{j+1}^* U_{j+1} \|$. Both of these residuals should be tiny.

[7] www.siam.org/books/ot101

(d) The Ritz values are the eigenvalues of the $j \times j$ Hessenberg matrix H_j. Compute these using the `eig` command, and plot the Ritz values and the exact eigenvalues together:

```
>> plot(real(lam),imag(lam),'r+',real(ritz),imag(ritz),'bo')
```

As you can see, the outlying eigenvalues are approximated well by Ritz values, as we claimed in Section 9.1.

(e) If we had taken fewer Arnoldi steps, the Ritz values would not have been as good. Compute the Ritz values from 10-step and 20-step Arnoldi runs, and plot them together with the exact eigenvalues. For this it is not necessary to run the Arnoldi process again, since the desired Ritz values are eigenvalues of submatrices of the H that you computed on the first run.

Exercise 9.4.2. Modify your Arnoldi code so that reorthogonalization is not done. Do 30 Arnoldi steps and then check the residual $\| I_{j+1} - U_{j+1}^* U_{j+1} \|$. Notice that the orthogonality has been significantly degraded. Repeat this with a run of 60 Arnoldi steps to see a complete loss of orthogonality.

Exercise 9.4.3. To find the eigenvalues of `west0479` that are closest to a target τ, modify your Arnoldi code so that it does shift-and-invert. Instead of multiplying a vector by A at each step, multiply the vector by $(A - \tau I)^{-1}$. Do not form the matrix $(A - \tau I)^{-1}$. Instead perform a sparse LU decomposition: `[L R P] = lu(A-tau*speye(n));` and use the factors L, R, and P. The factorization only has to be done once for a given run. Then the factors can be reused repeatedly. Try $\tau = 10$, $\tau = -10$, and other values. Complex values are okay as well.

Exercise 9.4.4.

(a) Using (9.4.11) as a guide, write MATLAB code that implements the symmetric Lanczos process with reorthogonalization. (Alternatively, download the file `symlan.m` from the website.)

(b) Generate a sparse discrete negative Laplacian matrix using the commands

```
>> A = delsq(numgrid('H',40))
>> perm = symamd(A); A = A(perm,perm);
```

How big is A? Is A symmetric? Some informative commands:

```
>> issparse(A)
>> size(A)
>> nnz(A)
>> spy(A)
>> norm(A-A',1)
>> help delsq, help numgrid, help symamd, ...
```

A is a symmetric, positive-definite matrix, so its eigenvalues are real and positive. The smallest ones are of greatest physical interest. Again this matrix is not really large, so you can compute its “true” eigenvalues by `lam = eig(full(A));`.

(c) Run 20 steps of the symmetric Lanczos process applied to A, starting with a random initial vector. Compute $\| AU_j - U_{j+1}H_{j+1,j} \|$ (with $j = 20$) to check the correctness of your code. Also check the orthonormality of the columns of U by computing $\| I_{j+1} - U_{j+1}^* U_{j+1} \|$.

(d) The Ritz values are the eigenvalues of the $j \times j$ tridiagonal matrix H_j. Compute these using the `eig` command. Notice that the smallest Ritz values are not very good approximations of the smallest eigenvalues of A. Try some longer Lanczos runs to get better approximations.

(e) Now run 20 steps of the symmetric Lanczos process applied to A^{-1}. This is the shift-and-invert strategy with zero shift. Do not form A^{-1} explicitly; do an LU or Cholesky decomposition and use it to perform the transformations $x \to A^{-1}x$ economically. The inverses of the largest Ritz values are estimates of the smallest eigenvalues of A. Notice that we now have really good approximations.

(f) Modify your Lanczos code so that it does not do the reorthogonalization step. Then repeat the previous run. Notice that orthogonality is completely lost and the approximations do not look nearly so good. On closer inspection you will see that this is mainly due to the fact that the smallest eigenvalue has been found twice. Do a longer run of, say, 60 steps to observe more extensive duplication of eigenvalues.

Exercise 9.4.5.

(a) Write a MATLAB program that does the symmetric Lanczos process with implicit restarts by the second restart method described in Section 9.2. Alternatively, download the file `relan.m` from the website.

(b) Apply your implicitly-restarted Lanczos code to the matrix A of Exercise 9.4.4 to compute the smallest 8 eigenvalues and corresponding eigenfunctions of A to as much accuracy as you can. Compute the residuals $\| Av - v\lambda \|$ to check the quality of your computed eigenpairs. Compare your computed eigenvalues with the true eigenvalues (from `eig(full(A))`) to make sure that they are correct.

(c) Apply your code to the matrix A^{-1}. Do not compute A^{-1} explicitly; use an LU or Cholesky decomposition to effect the transformations $x \to A^{-1}x$. Compute the smallest eight eigenvalues of A. As before, compute residuals and compare with the true eigenvalues. Notice that this iteration converged in many fewer iterations than the iteration with A did.

(d) In producing your test matrix A, you used the command `numgrid('H',40)` (and others). You can build a bigger test matrix by replacing the number 40 by a larger number. Build a matrix that is big enough that your computer runs out of memory when it tries to compute the eigenvalues by `eig(full(A))`. Repeat parts (b) and (c) using this larger matrix. Of course, you will not be able to compare with the "true" eigenvalues from `eig`, but you can compare the results of (b) and (c) with each other. In order to get convergence in (b), you may need to increase the dimension of the Krylov spaces that are being built (the parameter k, which is known as `nmax` in `relan.m`).

(e) Now make an even bigger matrix that is so big that you cannot use the shift-and-invert strategy because you do not have enough memory to store the factors of the LU or Cholesky decomposition. Compute the eight smallest eigenvalues by applying the implicitly-restarted Lanczos scheme to A. This will take a long time, but if you take enough iterations and large enough Krylov subspaces, you will eventually get the eigenpairs.

Exercise 9.4.6.

(a) Familiarize yourself with the built-in MATLAB function `eigs`, which computes eigenpairs by the implicitly-restarted Arnoldi process, as implemented in ARPACK [147].

(b) Repeat Exercise 9.4.5 using `eigs`.

Exercise 9.4.7. This exercise checks some basic properties of the Arnoldi process

(a) Show that the vector $\widehat{u}_{j+1}$ defined by (9.4.2) is orthogonal to u_k if and only if h_{kj} is defined as in (9.4.1).

(b) If $Au_j \in \operatorname{span}\{u_1, \ldots, u_j\}$, then $Au_j = \sum_{i=1}^{j} c_i u_i$ for some choice of coefficients c_1, …, c_j. Show that $c_k = u_k^* A u_j$ for $k = 1, \ldots, j$. Deduce that $\widehat{u}_{j+1} = 0$ if and only if $Au_j \in \operatorname{span}\{u_1, \ldots, u_j\}$.

(c) Verify (9.4.5) and (9.4.6)

(d) Show that $\widehat{u}_{j+1} = (I - P_j)Au_j$, where $P_j = U_j U_j^*$, the orthoprojector of $\mathbb{C}^n$ onto $\mathcal{R}(U_j)$. Thus $\widehat{u}_j$ is the orthogonal projection of Au_j onto $\mathcal{R}(U_j)^\perp$.

Exercise 9.4.8. Let $A \in \mathbb{C}^{n\times n}$, and let $\mathcal{U}$ be a subspace of $\mathbb{C}^n$. Then a vector $u \in \mathcal{U}$ is called a *Ritz vector* of A associated with $\mathcal{U}$, $\theta \in \mathbb{C}$ is called a *Ritz value*, and (θ, u) is called a *Ritz pair* if

$$Au - u\theta \perp \mathcal{U}.$$

Let U be a matrix with orthonormal columns such that $\mathcal{U} = \mathcal{R}(U)$, and let $B = U^*AU$. Then each $u \in \mathcal{U}$ can be expressed in the form $u = Ux$, where $x = U^*u$. Prove that (θ, u) is a Ritz pair of A if and only if (θ, x) is an eigenpair of B.

Exercise 9.4.9. In the symmetric Lanczos process, the Hessenberg matrices H_k are tridiagonal. Although this is a very easy result, it is nevertheless instructive to prove it by a second method. In this exercise we use Krylov subspaces to show that if $A = A^*$ in the Arnoldi process, then $h_{ij} = 0$ if $i < j - 1$. Recall that $\operatorname{span}\{u_1, \ldots, u_i\} = \mathcal{K}_i(A, x)$. Thus $u_k \perp \mathcal{K}_i(A, x)$ if $k > i$. Assume $A = A^*$.

(a) Show that $h_{ij} = \langle Au_j, u_i\rangle = \langle u_j, Au_i\rangle$.

(b) Show that $Au_i \in \mathcal{K}_{i+1}(A, x)$.

(c) Deduce that $h_{ij} = 0$ if $i < j - 1$.

Exercise 9.4.10. This exercise investigates Cayley transforms.

(a) Consider the transform $w \mapsto z$ of the extended complex plane $\mathbb{C} \cup \{\infty\}$ given by

$$z = \phi(w) = \frac{w+i}{w-i}.$$

This is the original Cayley transform, but we will refer to all of the other linear fractional transformations appearing in this exercise as Cayley transforms as well. Show that ϕ maps the extended real line one-to-one onto the unit circle, with $\phi(\infty) = 1$.

(b) Given a point τ on the unit circle, define

$$z = \phi_\tau(w) = \tau\,\frac{w+i}{w-i}.$$

Show that ϕ_τ maps the extended real line one-to-one onto the unit circle, with $\phi_\tau(\infty) = \tau$

(c) Show that the inverse of ϕ_τ is given by

$$w = \phi_\tau^{-1}(z) = i\,\frac{z+\tau}{z-\tau}.$$

This transformation maps the unit circle onto the extended real line, with $\tau \mapsto \infty$.

(d) Let U be a unitary matrix, and let τ be a point on the unit circle that is not an eigenvalue of U. Let

$$A = i(U+\tau I)(U-\tau I)^{-1}.$$

Show that for each eigenpair (λ, v) of U, A has an eigenpair $(i(\lambda+\tau)/(\lambda-\tau), v)$. In particular, all of the eigenvalues of A are real. Moreover, the eigenvalues of U that are closest to τ are mapped to eigenvalues of A that have the largest modulus.

(e) Show that A is normal. Deduce that A is Hermitian.

Exercise 9.4.11. What form does the Arnoldi process take when it is applied to a skew-symmetric matrix $A = -A^T \in \mathbb{R}^{n\times n}$?

(a) What form does the equation $AU_k = U_{k+1}H_{k+1,k}$ (cf. Proposition 9.2.3) take in this case? That is, what is the form of $H_{k+1,k}$?

(b) What is the form of the short recurrence, and how are the coefficients computed? This is the *skew-symmetric Lanczos process*.

(c) Show that $A^2U_k = U_{k+2}H_{k+2,k+1}H_{k+1,k}$. Compute $H_{k+2,k+1}H_{k+1,k}$ and deduce that the skew-symmetric Lanczos process is equivalent in principle to two independent Lanczos processes on the symmetric matrix A^2, one starting with u_1 and the other starting with $u_2 = Au_1/\|Au_1\|$.

9.5 The Unitary Arnoldi Process

In the previous section we discussed the unitary eigenvalue problem briefly, showing that a Cayley transform can be used to transform the problem to a Hermitian eigenvalue problem. In this section we consider an approach that works directly with the unitary matrix.

We have seen that if A is Hermitian, then the Arnoldi process admits significant simplification in principle. The same is true if A is unitary, although this is not so obvious. In this section we will develop the isometric Arnoldi process of Gragg [102, 104]. Here we will call it the unitary Arnoldi process, since all isometric matrices are unitary in the finite-dimensional case. Different but equivalent algorithms are developed in [40, 51, 84, 215, 218].[8]

Let $U \in \mathbb{C}^{n\times n}$ be a unitary matrix ($U^* = U^{-1}$). Then U^{-1} is as accessible as U is, so one could consider applying the Arnoldi process to U or U^{-1} with equal ease. It turns out that the significant simplification is achieved by doing both at once. If we apply the Arnoldi process to U with starting vector $v_1 = cx$, we produce an orthonormal sequence v_1, v_2, v_3, ..., such that

$$\operatorname{span}\{v_1, \ldots, v_j\} = \mathcal{K}_j(U, x), \qquad j = 1, 2, 3, \ldots .$$

The jth step of the process has the form

$$v_{j+1}h_{j+1,j} = Uv_j - \sum_{i=1}^{j} v_i h_{ij}, \tag{9.5.1}$$

where the h_{ij} are chosen so that $v_{j+1} \perp \operatorname{span}\{v_1, \ldots, v_j\}$.

Now suppose we apply the Arnoldi process to U^{-1} with the same starting vector $w_1 = v_1 = cx$. Then we generate an orthonormal sequence w_1, w_2, w_3, ..., such that

$$\operatorname{span}\{w_1, \ldots, w_j\} = \mathcal{K}_j(U^{-1}, x), \qquad j = 1, 2 3, \ldots .$$

The jth step of the process has the form

$$w_{j+1}k_{j+1,j} = U^{-1}w_j - \sum_{i=1}^{j} w_i k_{ij}, \tag{9.5.2}$$

where the k_{ij} are chosen so that $w_{j+1} \perp \operatorname{span}\{w_1, \ldots, w_j\}$.

Notice that

$$\operatorname{span}\{U^{j-1}w_j, \ldots, U^{j-1}w_1\} = U^{j-1}\operatorname{span}\{U^{-(j-1)}x, \ldots, U^{-1}x, x\} =$$
$$\operatorname{span}\{x, Ux, \ldots, U^{j-1}x\} = \mathcal{K}_j(U, x).$$

Thus v_1, ..., v_j and $U^{j-1}w_j$, ..., $U^{j-1}w_1$ are orthonormal bases of the same space. The latter basis is orthonormal because unitary matrices preserve orthonormality. Therefore we could make v_{j+1} orthogonal to $\operatorname{span}\{v_1, \ldots, v_j\}$ by orthogonalizing against $U^{j-1}w_j$,

[8]The isometric Arnoldi process is also equivalent to Szegö's recursion [201] for orthogonal polynomials on the unit circle, the lattice algorithm of speech processing, and the Levinson-Durbin algorithm [78, 148] for solving the Yule-Walker equations. The relationships are discussed in [218] and elsewhere in the literature.

…, $U^{j-1}w_1$ instead of v_1, …, v_j in (9.5.1). This is where we realize our economy. Since $\operatorname{span}\{U^{j-2}w_{j-1}, \ldots, U^{j-2}w_1\} = \operatorname{span}\{x, Ux, \ldots, U^{j-2}x\} = \operatorname{span}\{v_1, \ldots, v_{j-1}\}$, which is orthogonal to v_j, we see that $\operatorname{span}\{U^{j-1}w_{j-1}, \ldots, U^{j-1}w_1\}$ is orthogonal to Uv_j, again using the fact that unitary matrices preserve orthogonality. Thus $U^{j-1}w_j$ is the only member of the basis $U^{j-1}w_j$, …, $U^{j-1}w_1$ that is not already orthogonal to Uv_j, and we can replace (9.5.1) by

$$v_{j+1}h_{j+1,j} = Uv_j - U^{j-1}w_j\gamma_j, \tag{9.5.3}$$

where

$$\gamma_j = \langle Uv_j, U^{j-1}w_j\rangle. \tag{9.5.4}$$

Now it is natural to ask how one can economically produce the vector $U^{j-1}w_j$. Reversing the roles of U and U^{-1}, we find that the vectors $U^{-(j-1)}v_1$, …$U^{-(j-1)}v_j$ span the same space as w_j, …, w_1 and that all but $U^{-(j-1)}v_j$ are already orthogonal to $U^{-1}w_j$. Thus (9.5.2) can be replaced by

$$w_{j+1}k_{j+1,j} = U^{-1}w_j - U^{-(j-1)}v_j\delta_j, \tag{9.5.5}$$

where

$$\delta_j = \langle U^{-1}w_j, U^{-(j-1)}v_j\rangle. \tag{9.5.6}$$

One easily checks that $\delta_j = \overline{\gamma_j}$. Moreover $k_{j+1,j} = h_{j+1,j} = \sqrt{1 - |\gamma_j|^2}$ by the orthogonality properties of the vectors in (9.5.3) and (9.5.5). Let $\tilde{v}_j = U^{j-1}w_j$ for $j = 1, 2, \ldots$. Then, multiplying (9.5.5) by U^j, we obtain

$$\tilde{v}_{j+1}\sigma_j = \tilde{v}_j - Uv_j\overline{\gamma_j},$$

where σ_j is a new name for $k_{j+1,j}$. Combining this with (9.5.3), we obtain the pair of two-term recurrences

$$\begin{aligned} v_{j+1}\sigma_j &= Uv_j - \tilde{v}_j\gamma_j, \\ \tilde{v}_{j+1}\sigma_j &= \tilde{v}_j - Uv_j\overline{\gamma_j}, \end{aligned} \tag{9.5.7}$$

where

$$\gamma_j = \langle Uv_j, \tilde{v}_j\rangle \quad \text{and} \quad \sigma_j = \sqrt{1 - |\gamma_j|^2}. \tag{9.5.8}$$

This is the unitary Arnoldi process [102, 104]. Equations (9.5.7) and (9.5.8) involve U only; U^{-1} has disappeared.[9] Of course, we could have defined $\tilde{w}_j = U^{-(j-1)}v_j$ and written down a pair of equivalent recurrences in which U^{-1} appears and U does not. For recurrences involving both U and U^{-1} see [51, 215], for example.

In practical floating-point computations, the unitary Arnoldi process must be modified in several ways to make it robust. First of all, the formula (9.5.8) is not a good way to compute σ_j because it is inaccurate when $|\gamma_j| \approx 1$. A better method is to compute

$$\hat{v}_{j+1} = Uv_j - \tilde{v}_j\gamma_j,$$

[9]Since U^{-1} is not used in (9.5.7,9.5.8), the process is applicable to isometric operators ($\|Ux\| = \|x\|$ on an infinite-dimensional Hilbert space) that are not unitary. Therefore Gragg called it the isometric Arnoldi process.

then let $\sigma_j = \| \widehat{v}_{j+1} \|$ and $v_{j+1} = \sigma_j^{-1} \widehat{v}_{j+1}$. This is just what we do in the standard Arnoldi process. In practice the σ_j and γ_j so produced will not exactly satisfy $\sigma_j^2 + |\gamma_j|^2 = 1$, so we must enforce unitarity by the steps

$$\begin{aligned} \nu &\leftarrow \left(\sigma_j^2 + |\gamma_j^2| \right)^{1/2} \\ \gamma_j &\leftarrow \gamma_j / \nu \\ \sigma_j &\leftarrow \sigma_j / \nu \end{aligned}$$

We took this same precaution in the unitary QR algorithm (c.f. 8.10.4).

In principle the short recurrences (9.5.7) generate v_{j+1} from v_j and $\widetilde{v}_j$; the earlier vectors are not needed. In practice roundoff errors will cause the same gradual loss of orthogonality as in any Krylov process unless v_{j+1} is specifically orthogonalized against all of the v_i. As before, this is not a problem if we do short runs with frequent implicit restarts. Moreover, we do not mind keeping all of the v_i around, as we need them anyway if we want to compute eigenvectors.

A worse problem is what to do about the auxiliary vectors $\widetilde{v}_j$. The sequence $(w_j) = (U^{-j+1}\widetilde{v}_j)$ is orthonormal in principle, and we should enforce orthonormality of that sequence as well. However, the sequence that we have in hand is $(\widetilde{v}_j)$, not $(U^{-j+1}\widetilde{v}_j)$, so we cannot do the orthonormalization. We have found [63, 65] that under these circumstances the second formula of (9.5.7) is not very robust; the accuracy of the $\widetilde{v}_{j+1}$ is poor when σ_j is small. Fortunately there is another formula for $\widetilde{v}_{j+1}$ that works much better in practice. Solving the first equation of (9.5.7) for Uv_j, substituting this into the second equation, using the formula $\sigma_j^2 + |\gamma_j^2| = 1$, and dividing by σ_j, we obtain

$$\widetilde{v}_{j+1} = \widetilde{v}_j \sigma_j - v_{j+1} \overline{\gamma_j}. \tag{9.5.9}$$

There is no possibility of cancellation in this formula. It takes the two orthogonal unit vectors $\widetilde{v}_j$ and v_{j+1}, rescales them and combines them to form a third unit vector $\widetilde{v}_{j+1}$.

With these modifications we have the following algorithm.

Unitary Arnoldi process with reorthogonalization.

```
$v_1 \leftarrow x/\|x\|$
$V_1 \leftarrow [v_1]$
$\tilde{v} \leftarrow v_1$
for $j = 1, 2, 3, \ldots$
    $v_{j+1} \leftarrow U v_j$
    $\gamma_j \leftarrow \tilde{v}^* v_{j+1}$
    $v_{j+1} \leftarrow v_{j+1} - \tilde{v}\gamma_j$
    $\delta_{1:j} \leftarrow V_j^* v_{j+1}$
    $v_{j+1} \leftarrow v_{j+1} - V_j \delta_{1:j}$    (reorthogonalize)
    $\gamma_j \leftarrow \gamma_j + \delta_j$
    $\sigma_j \leftarrow \|v_{j+1}\|$                                   (9.5.10)
    if $\sigma_j = 0$
        set flag (span$\{v_1, \ldots, v_j\}$ is invariant)
        exit
    $v_{j+1} \leftarrow v_{j+1}/\sigma_j$
    $\nu \leftarrow \sqrt{\sigma_j^2 + |\gamma_j^2|}$
    $\sigma_j \leftarrow \sigma_j/\nu$
    $\gamma_j \leftarrow \gamma_j/\nu$
    $\tilde{v} \leftarrow \tilde{v}\sigma_j - v_{j+1}\overline{\gamma_j}$
    $V_{j+1} \leftarrow \begin{bmatrix} V_j & v_{j+1} \end{bmatrix}$
```

We have yet to discuss how the data produced by the unitary Arnoldi process can be used to yield information about the eigensystem of U. Just as in Section 9.4, we can rewrite (9.5.1) in the form

$$UV_j = V_{j+1}H_{j+1,j} = V_j H_j + v_{j+1}h_{j+1,j}e_j^T,$$

where the notation is analogous to that established in Section 9.4. If we want to use the unitary Arnoldi process to estimate eigenvalues, we need the entries h_{ij} of H_j or equivalent information. With (9.5.7) one gets instead the quantities γ_i and σ_i, $i = 1, 2, \ldots, j$. As we will show below, these are exactly the Schur parameters of H_j, as described in Section 8.10. Thus we can obtain H_j (or rather a closely related matrix) in product form and use an algorithm like the one described in Section 8.10 to compute its eigenvalues.

Comparing (9.5.1) with (9.5.7), it must be the case that

$$\tilde{v}_j\gamma_j = \sum_{i=1}^{j} v_i h_{ij}.$$

Since $v_1, \ldots, v_j$ are linearly independent, the h_{ij} are uniquely determined by this expression, and we can figure out what they are by establishing a recurrence for the $\tilde{v}_i$. To begin with, note that $\tilde{v}_1 = v_1$. We have already observed that

$$\tilde{v}_{j+1} = \tilde{v}_j\sigma_j - v_{j+1}\overline{\gamma_j}, \quad j = 1, 2, 3, \ldots. \tag{9.5.11}$$

Applying this result repeatedly, we can deduce that

$$\widetilde{v}_j = v_1 \prod_{k=1}^{j-1} \sigma_k + \sum_{i=2}^{j} v_i \left(-\overline{\gamma}_{i-1} \prod_{k=i}^{j-1} \sigma_k \right) \tag{9.5.12}$$

for $j = 1, 2, 3, \ldots$, with the usual conventions that an empty product equals 1 and an empty sum equals zero. Introducing the simplifying notation

$$\pi_{ij} = \prod_{k=i}^{j-1} \sigma_k \tag{9.5.13}$$

(with $\pi_{jj} = 1$), equation (9.5.12) becomes

$$\widetilde{v}_j = v_1 \pi_{1j} + \sum_{i=2}^{j} v_i \left(-\overline{\gamma}_{i-1} \pi_{ij} \right). \tag{9.5.14}$$

Rewriting the first equation in (9.5.7) as $U v_j = \widetilde{v}_j \gamma_j + v_{j+1} \sigma_j$, and inserting the result from (9.5.14), we obtain

$$U v_j = v_1 \pi_{1j} \gamma_j + \sum_{i=2}^{j} v_i (-\overline{\gamma}_{i-1} \pi_{ij} \gamma_j) + v_{j+1} \sigma_j. \tag{9.5.15}$$

Since

$$U v_j = \sum_{i=1}^{j+1} v_i h_{ij},$$

equation (9.5.15) tells us immediately what the entries of $H_{j+1,j}$ are:

$$h_{ij} = \begin{cases} \pi_{ij} \gamma_j & \text{if } i = 1 \\ -\overline{\gamma}_{i-1} \pi_{ij} \gamma_j & \text{if } 2 \le i \le j \\ \sigma_j & \text{if } i = j+1 \\ 0 & \text{if } i > j+1 \end{cases}$$

Thus

$$H_{j+1,j} = \begin{bmatrix} \gamma_1 & \sigma_1 \gamma_2 & \sigma_1 \sigma_2 \gamma_3 & \cdots & \sigma_1 \cdots \sigma_{j-1} \gamma_j \\ \sigma_1 & -\overline{\gamma}_1 \gamma_2 & -\overline{\gamma}_1 \sigma_2 \gamma_3 & \cdots & -\overline{\gamma}_1 \sigma_2 \cdots \sigma_{j-1} \gamma_j \\ 0 & \sigma_2 & -\overline{\gamma}_2 \gamma_3 & \cdots & -\overline{\gamma}_2 \sigma_3 \cdots \sigma_{j-1} \gamma_j \\ 0 & 0 & \sigma_3 & \cdots & -\overline{\gamma}_3 \sigma_4 \cdots \sigma_{j-1} \gamma_j \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \sigma_j \end{bmatrix}. \tag{9.5.16}$$

For $k = 1, \ldots, j$, define matrices $G_k \in \mathbb{C}^{j+1 \times j+1}$ by (cf. § 8.10)

$$G_k = \begin{bmatrix} I_{k-1} & & & \\ & \gamma_k & \sigma_k & \\ & \sigma_k & -\overline{\gamma}_k & \\ & & & I_{j-k} \end{bmatrix}. \tag{9.5.17}$$

Then one easily checks that

$$H_{j+1,j} = G_1 G_2 \cdots G_j I_{j+1,j}.$$

That is, $H_{j+1,j}$ consists of the first j columns of $G_1 \cdots G_j$.

Implicit Restarts

We describe a restart by the second (thick restart) method following [63, 65]. After k steps of the unitary Arnoldi process, we have

$$UV_k = V_{k+1} H_{k+1,k},$$

where $H_{k+1,k}$ is available in the form

$$H_{k+1,k} = G_1 G_2 \cdots G_k I_{k+1,k}.$$

To make an implicit restart we would normally compute the eigenvalues of H_k, the square matrix obtained by deleting the bottom row from $H_{k+1,k}$. Notice, however, that H_k is not unitary. The columns of $H_{k+1,k}$ are orthonormal because $H_{k+1,k}$ consists of the first k columns of the unitary matrix $G_1 \cdots G_k$. Thus the columns of H_k are orthogonal, but they are not orthonormal because the last column has norm less than 1 (unless $\sigma_k = 0$). We would like to have a unitary matrix to operate on so that we can manipulate it (almost) entirely in terms of its Schur parameters. To this end we make a modification of H_k. Let $\widetilde{\gamma}_k = \gamma_k/|\gamma_k|$.[10] Let $\widetilde{G}_1, \ldots, \widetilde{G}_{k-1} \in \mathbb{C}^{k\times k}$ be the submatrices of $G_1, \ldots, G_{k-1}$ obtain by deleting the last row and column. Define $\widetilde{G}_k = \operatorname{diag}\{1, \ldots, 1, \widetilde{\gamma}_k\} \in \mathbb{C}^{k\times k}$, and define

$$\widetilde{H}_k = \widetilde{G}_1 \ldots \widetilde{G}_{k-1} \widetilde{G}_k.$$

$\widetilde{H}_k$ is the same as H_k, except that its last column has been rescaled to have norm 1. Now $\widetilde{H}_k$ is unitary, and we can manipulate it in terms of its Schur parameters.

For our restart we will take the eigenvalues of $\widetilde{H}_k$ as estimates of eigenvalues of U. Note that the eigenvalues of $\widetilde{H}_k$ lie on the unit circle, while those of H_k lie inside but typically not on the unit circle. Once σ_k becomes small, as happens as we approach convergence after several restarts, the eigenvalues of H_k and $\widetilde{H}_k$ are nearly the same; those of H_k lie just inside the unit circle.

Since H_k and $\widetilde{H}_k$ differ only in their kth column, $H_k = \widetilde{H}_k + pe_k^T$ for some $p \in \mathbb{C}^k$. Thus the relationship $UV_k = V_{k+1}H_{k+1,k} = V_k H_k + v_{k+1}\sigma_k e_k^T$ can be rewritten as

$$UV_k = V_k \widetilde{H}_k + \widetilde{p} e_k^T, \tag{9.5.18}$$

where $\widetilde{p} = p + v_{k+1}\sigma_k$.

Since $\widetilde{H}_k$ is given in terms of its Schur parameters, the eigensystem of $\widetilde{H}_k$ can be computed by the version of the QR algorithm presented in Section 8.10 or any other stable method that works with the Schur parameters. Since U is normal, this yields a diagonalization

$$\widetilde{H}_k = S D_k S^*,$$

[10] In the rare event $\gamma_k = 0$, take $\widetilde{\gamma}_k = 1$ or $\widetilde{\gamma}_k = e^{i\theta}$, where θ is random, for example.

where S is unitary and D_k is diagonal. We can suppose that the eigenvalues of D_k are sorted so that the m that we wish to retain are in the upper left-hand corner of D_k. Multiplying (9.5.18) on the right by S, we obtain

$$U\widetilde{V}_k = \widetilde{V}_k D_k + \widetilde{p}\,\widetilde{z}^T, \tag{9.5.19}$$

where $\widetilde{V}_k = V_k S$ and $\widetilde{z}^T = e_k^T S$. To effect the restart we retain only the first m columns of (9.5.18):

$$U\widetilde{V}_m = \widetilde{V}_m D_m + \widetilde{p} z^T, \tag{9.5.20}$$

where D_m is the upper left-hand $m \times m$ submatrix of D_k, and z is the vector consisting of the first m components of $\widetilde{z}$. Now we have to make a transformation that maps z^T to a multiple of e_m^T and returns D_m to upper Hessenberg form. $\widetilde{H}_k$ was presented in terms of its Schur parameters, and so are D_m and D_k. We would also like the new Hessenberg matrix to be expressed in terms of Schur parameters. Thus we would like this final transformation to operate on the Schur parameters.

We begin by applying a unitary transformation on the right (rotator or reflector) that acts only on columns 1 and 2 and transforms the first entry of z^T to zero. When this same transformation is applied to D_m, it affects only the first two columns. When we complete the similarity transformation by a left multiplication, only the first two rows are affected. The resulting matrix has a nonzero entry in position $(2, 1)$. It is upper Hessenberg, so it can be represented in terms of Schur parameters. Next a transformation is applied to columns 2 and 3 to create a zero in the second position of the (transformed) z^T vector. When this same transformation is applied the (transformed) D_m matrix, it affects columns 2 and 3, creating a nonzero in the $(3, 2)$ position. The matrix is still upper Hessenberg. When the similarity transformation is completed by transforming rows 2 and 3, a nonzero entry is created in position $(3, 1)$. This is a bulge, and it needs to be eliminated. Fortunately it can be chased off the top of the matrix by the reverse (RQ variant) of the bulge chasing procedure for unitary Hessenberg matrices described in Section 8.10. This just consists of a unitary operation on columns 1 and 2 to get rid of the bulge, followed by an operation on rows 1 and 2 that completes the similarity transformation. Since this transformation acts only on columns 1 and 2, it does not destroy the two zeros that have been created in z^T.

The next step is a transformation on columns 3 and 4 that creates a zero in the third position of the z vector. The corresponding operations on the D_m matrix create a bulge in position $(4, 2)$. This too can be chased off of the top of the matrix. This requires operations in the $(2, 3)$ plane, followed by operations in the $(1, 2)$ plane. Because only the first three columns are touched, the zeros that have been created in z are left intact. Continuing in this manner, we transform z^T to a multiple of e_m^T and D_m to a unitary Hessenberg matrix $\widehat{H}_m$ represented in terms of its Schur parameters. Let Q denote the product of all of the transformations involved in this process, so that $z^T Q = \alpha e_m^T$ and $Q^* D_m Q = \widehat{H}_m$. Multiplying (9.5.20) on the right by Q and letting $\widehat{V}_m = \widetilde{V}_m Q$, we have

$$U\widehat{V}_m = \widehat{V}_m \widehat{H}_m + \widetilde{p}\,\alpha\, e_m^T. \tag{9.5.21}$$

This is still not an Arnoldi configuration, as $\widetilde{p}$ is not orthogonal to the columns of $\widehat{V}_m$.[11] However, we can obtain an Arnoldi configuration by dropping the mth column from (9.5.21).

[11]Moreover, $\widehat{H}_m$ is unitary, which is impossible in a unitary Arnoldi configuration unless $\mathcal{R}(\widehat{V}_m)$ is invariant under U.

This gives

$$U\widehat{V}_{m-1} = \widehat{V}_m \widehat{H}_{m,m-1} = \widehat{V}_{m-1}\widehat{H}_{m-1} + \widehat{h}_m \widehat{\sigma}_m e_m^T,$$

which is a configuration from which the Arnoldi process can be continued. This is exactly what would be obtained from $m-1$ steps of the Arnoldi process with starting vector $\widehat{v}_1$. We now recycle the notation, dropping the hats, and write

$$UV_{m-1} = V_{m-1}H_{m-1} + v_m \sigma_m e_m^T. \tag{9.5.22}$$

Recall that our unitary Arnoldi process uses the recurrences (9.5.7) and (9.5.9), so the first step after (9.5.22), to produce v_{m+1}, will be

$$\begin{aligned} v_{m+1}\sigma_m &= Uv_m - \widetilde{v}_m \gamma_m, \\ \widetilde{v}_{m+1} &= \widetilde{v}_m \sigma_m - v_{m+1}\overline{\gamma_m}, \end{aligned} \tag{9.5.23}$$

for which we need $\widetilde{v}_m$. This we generate as follows. Replacing m by $m-1$ in the first equation of (9.5.23) and solving for $\widetilde{v}_{m-1}$, we have

$$\widetilde{v}_{m-1} = (Uv_{m-1} - v_m\sigma_{m-1})/\gamma_{m-1}.$$

Using this to compute $\widetilde{v}_{m-1}$, we then can obtain $\widetilde{v}_m$ by

$$\widetilde{v}_m = \widetilde{v}_{m-1}\sigma_{m-1} - v_m\overline{\gamma}_{m-1}.$$

The numbers γ_{m-1} and σ_{m-1} are available; they are Schur parameters that form part of the description of H_m. Once we have $\widetilde{v}_m$, we can proceed with (9.5.23) and subsequent steps.

In Section 9.3 we justified implicit restarts by showing that they effect nested subspace iterations driven by $p(U)$, where p is a polynomial whose zeros are near parts of the spectrum that we want to suppress. The restart procedure outlined here is somewhat different because it involves a modification of the matrix H_k. However, the theory of Section 9.3 continues to be valid here since the modification to H_k affects only its last column. The key equation is (9.5.18), in which the error term $\tilde{p}e_k^T$ affects only the last column. More details are given in [63, 65].

Exercises

Exercise 9.5.1. This exercise fills in some of the details in the development of the unitary Arnoldi process.

(a) Show that $\operatorname{span}\{U^{-(j-1)}v_1, \ldots, U^{-(j-1)}v_j\} = \mathcal{K}_j(U^{-1}, x) = \operatorname{span}\{w_1, \ldots, w_j\}$.

(b) Show that $U^{-1}w_j$ is orthogonal to $U^{-(j-1)}v_1, \ldots U^{-(j-1)}v_{j-1}$.

(c) Using the fact that unitary matrices preserve the inner product, show that $\delta_j = \overline{\gamma_j}$, where γ_j and δ_j are as in (9.5.4) and (9.5.6), respectively.

(d) All of the vectors in (9.5.3) and (9.5.5) are unit vectors, and some of them are orthogonal to each other. Use the Pythagorean theorem to verify the $k_{j+1,j} = h_{j+1,j} = \sqrt{1-|\gamma_j|^2}$. Thus, letting $\sigma_j = k_{j+1,j} = h_{j+1,j}$, we have $\sigma_j^2 + |\gamma_j|^2 = 1$.

(e) Let $\widetilde{w}_j = U^{-(j-1)}v_j$. Write down a pair of recurrences for w_j and $\widetilde{w}_j$ that are equivalent to (9.5.7) but involve U^{-1} and not U.

Exercise 9.5.2. This exercise works out the details of the construction of the matrix $H_{j+1,j}$ from the parameters γ_i and σ_i.

(a) From (9.5.7) deduce that $Uv_j = \widetilde{v}_j\gamma_j + v_{j+1}\sigma_j$. Substitute this into the second equation of (9.5.7) to deduce that

$$\widetilde{v}_{j+1} = \widetilde{v}_j\sigma_j - v_{j+1}\overline{\gamma_j}, \quad j = 1, 2, \ldots.$$

Bear in mind that $\sigma_j^2 + |\gamma_j|^2 = 1$.

(b) Deduce that

$$\begin{aligned}\widetilde{v}_2 &= v_1\sigma_1 - v_2\overline{\gamma}_1\\ \widetilde{v}_3 &= v_1\sigma_1\sigma_2 - v_2\overline{\gamma}_1\sigma_2 - v_3\overline{\gamma}_2\\ \widetilde{v}_4 &= v_1\sigma_1\sigma_2\sigma_3 - v_2\overline{\gamma}_1\sigma_2\sigma_3 - v_3\overline{\gamma}_2\sigma_3 - v_4\overline{\gamma}_3.\end{aligned}$$

Prove by induction on j that

$$\widetilde{v}_j = v_1\prod_{k=1}^{j-1}\sigma_k + \sum_{i=2}^{j} v_i\left(-\overline{\gamma}_{i-1}\prod_{k=i}^{j-1}\sigma_k\right)$$

for $j = 1, 2, 3, \ldots$, which becomes

$$\widetilde{v}_j = v_1\pi_{1j} + \sum_{i=2}^{j} v_j\left(-\overline{\gamma}_{i-1}\pi_{ij}\right)$$

with the abbreviation (9.5.13).

(c) Deduce (9.5.15).

(d) Check that (9.5.16) is correct.

Exercise 9.5.3. Show that $\| H_j \| \le 1$. Deduce that the eigenvalues of H_j (which are Ritz values of U) satisfy $|\lambda| \le 1$. Thus they lie on or inside the unit circle.

9.6 The Unsymmetric Lanczos Process

As we have seen, the recurrence in the Arnoldi process becomes a short recurrence if A is Hermitian, in which case it is called the symmetric Lanczos process. The unsymmetric Lanczos process [143], which we will now introduce, produces short recurrences even for nonhermitian A by giving up orthogonality. In the Hermitian case, the unsymmetric Lanczos process reduces to the symmetric Lanczos process. This and the next three sections draw substantially from [222].

A pair of sequences of vectors $u_1, u_2, \ldots, u_j$ and $w_1, w_2, \ldots, w_j$ is called *biorthonormal* if

$$\langle u_i, w_k\rangle = w_k^* u_i = \begin{cases} 0 & \text{if } i \ne k\\ 1 & \text{if } i = k.\end{cases} \tag{9.6.1}$$

This is a much weaker property than orthonormality (Exercise 9.6.1). The unsymmetric Lanczos process builds a biorthonormal pair of sequences, each of which spans a Krylov subspace.

Let $A \in \mathbb{C}^{n\times n}$ be an arbitrary square matrix. The process begins with any two vectors $u_1, w_1 \in \mathbb{C}^n$ satisfying $\langle u_1, w_1\rangle = 1$. After $j-1$ steps, vectors $u_1, \ldots, u_j$ and $w_1, \ldots, w_j$ satisfying the biorthonormality property (9.6.1) will have been produced. They also satisfy

$$\operatorname{span}\{u_1, \ldots, u_k\} = \mathcal{K}_k(A, v_1), \quad k = 1, \ldots, j$$

and

$$\operatorname{span}\{w_1, \ldots, w_k\} = \mathcal{K}_k(A^*, w_1), \quad k = 1, \ldots, j.$$

The vectors u_{j+1} and w_{j+1} are then produced as follows.

$$h_{ij} = \langle Au_j, w_i\rangle, \quad g_{ij} = \langle A^* w_j, u_i\rangle, \quad i = 1, \ldots, j, \tag{9.6.2}$$

$$\widehat{u}_{j+1} = Au_j - \sum_{i=1}^{j} u_i h_{ij}, \tag{9.6.3}$$

$$\widehat{w}_{j+1} = A^* w_j - \sum_{i=1}^{j} w_i g_{ij}, \tag{9.6.4}$$

$$\tau_{j+1} = \langle \widehat{u}_{j+1}, \widehat{w}_{j+1}\rangle, \tag{9.6.5}$$

and if $\tau_{j+1} \neq 0$, choose $h_{j+1,j}$ and $g_{j+1,j}$ so that $h_{j+1,j}\overline{g_{j+1,j}} = \tau_{j+1}$, and then set

$$u_{j+1} = \widehat{u}_{j+1}/h_{j+1,j}, \quad w_{j+1} = \widehat{w}_{j+1}/g_{j+1,j}. \tag{9.6.6}$$

Some of the basic properties of the unsymmetric Lanczos process are worked out in Exercise 9.6.2. To begin with, $\widehat{u}_{j+1}$ is orthogonal to $w_1, \ldots, w_k$, and $\widehat{w}_{j+1}$ is orthogonal to $u_1, \ldots, u_k$. Indeed the definitions of h_{ij} and g_{ij} given by (9.6.2) are the unique ones for which this desired property is realized. Assuming τ_{j+1} defined by (9.6.5) is not zero, there are many ways to choose $h_{j+1,1}$ and $g_{j+1,j}$ so that $h_{j+1,j}\overline{g_{j+1,j}} = \tau_{j+1}$. For any such choice, the vectors u_{j+1} and w_{j+1} satisfy $\langle u_{j+1}, w_{j+1}\rangle = 1$. If $\tau_{j+1} = 0$, the method breaks down. The consequences of breakdown will be discussed below.

It is easy to prove by induction that the unsymmetric Lanczos process produces vectors satisfying $\operatorname{span}\{u_1, \ldots, u_j\} = \mathcal{K}_j(A, u_1)$ and $\operatorname{span}\{w_1, \ldots, w_j\} = \mathcal{K}_j(A^*, w_1)$, for all j. Moreover, if at some step $Au_j \in \operatorname{span}\{u_1, \ldots, u_j\}$, then $\widehat{u}_{j+1} = 0$ (Exercise 9.6.2). Thus the process is a Krylov process as defined in Section 9.2. It is of course also true that if $A^* w_j \in \operatorname{span}\{w_1, \ldots, w_j\}$, then $\widehat{w}_{j+1} = 0$. We can think of the unsymmetric Lanczos process as a Krylov process on A that produces the sequence (u_j) with the help of the auxiliary sequence (w_j), but it is equally well a Krylov process on A^* that produces the sequence (w_j) with the help of the auxiliary sequence (u_j). Better yet, we can think of it as a pair of mutually supporting Krylov process. The process is perfectly symmetric with respect to the roles of A and A^*.

There are several ways in which a breakdown ($\tau_{j+1} = 0$) can occur. First of all, it can happen that $Au_j \in \operatorname{span}\{u_1, \ldots, u_j\}$, causing $\widehat{u}_{j+1} = 0$. This happens only when $\operatorname{span}\{u_1, \ldots, u_j\}$ is invariant under A (Proposition 9.2.3). Since we can then extract j

eigenvalues and eigenvectors of A from $\operatorname{span}\{u_1, \ldots, u_j\}$, we call this a benign breakdown. Another good sort of breakdown occurs when $A^* w_j \in \operatorname{span}\{w_1, \ldots, w_j\}$, causing $\widehat{w}_{j+1} = 0$. In this case $\operatorname{span}\{w_1, \ldots, w_j\}$ is invariant under A^*, so we can extract j eigenvalues and eigenvectors of A^* (left eigenvectors of A) from $\operatorname{span}\{w_1, \ldots, w_j\}$. It can also happen that $\tau_{j+1} = \langle \widehat{u}_{j+1}, \widehat{w}_{j+1} \rangle = 0$ even though both $\widehat{u}_{j+1}$ and $\widehat{w}_{j+1}$ are nonzero. This is a serious breakdown. The surest remedy is to start over with different starting vectors. Perhaps more dangerous than a serious breakdown is a near breakdown. This happens when $\widehat{u}_{j+1}$ and $\widehat{w}_{j+1}$ are healthily far from zero but they are nearly orthogonal, so that τ_{j+1} is nearly zero. This does not cause the process to stop, but it signals potential instability and inaccuracy of the results. Because of the risk of serious breakdowns or near breakdowns, it is best to keep the Lanczos runs fairly short and do implicit restarts frequently.[12]

We claimed at the beginning that the unsymmetric Lanczos process has short recurrences, which means that most of the coefficients h_{ij} and g_{ij} in (9.6.2) are zero. To see that this is so, we introduce notation similar to that used in previous sections. Suppose we have taken k steps of the unsymmetric Lanczos process. Let $U_k = \begin{bmatrix} u_1 & \cdots & u_k \end{bmatrix}$ and $W_k = \begin{bmatrix} w_1 & \cdots & w_k \end{bmatrix}$, and let $H_{k+1,k}$, H_k, $G_{k+1,k}$, and G_k denote the upper-Hessenberg matrices built from the coefficients h_{ij} and g_{ij}. Then, reorganizing (9.6.3), (9.6.4), and (9.6.6), we find that

$$Au_j = \sum_{i=1}^{j+1} u_i h_{ij} \quad \text{and} \quad A^* w_j = \sum_{i=1}^{j+1} w_i g_{ij}, \qquad j = 1, \ldots, k,$$

and consequently

$$AU_k = U_{k+1} H_{k+1,k} \quad \text{and} \quad A^* W_k = W_{k+1} G_{k+1,k}, \tag{9.6.7}$$

which can be rewritten as

$$AU_k = U_k H_k + u_{k+1} h_{k+1,k} e_k^T \tag{9.6.8}$$

and

$$A^* W_k = W_k G_k + w_{k+1} g_{k+1,k} e_k^T. \tag{9.6.9}$$

Biorthonormality implies that $W_k^* U_k = I_k$ and $W_k^* u_{k+1} = 0$. Therefore we can deduce from (9.6.8) that $H_k = W_k^* A U_k$. Similarly, from (9.6.9), we have $G_k = U_k^* A^* W_k$, so $G_k = H_k^*$. Since G_k and H_k are both upper Hessenberg, we deduce that they must in fact be tridiagonal. Thus most of the h_{ij} and g_{ij} are indeed zero, and those that are not zero are related. Let $\alpha_j = h_{jj}$, $\beta_j = h_{j+1,j}$, and $\gamma_j = h_{j,j+1}$. Then $\overline{\alpha_j} = g_{jj}$, $\overline{\beta_j} = g_{j,j+1}$, and $\overline{\gamma_j} = g_{j+1,j}$, and the recurrences (9.6.3) and (9.6.4) become

$$u_{j+1}\beta_j = Au_j - u_j\alpha_j - u_{j-1}\gamma_{j-1} \tag{9.6.10}$$

and

$$w_{j+1}\overline{\gamma_j} = A^* w_j - w_j\overline{\alpha_j} - w_{j-1}\overline{\beta_{j-1}}, \tag{9.6.11}$$

respectively, where it is understood that $u_0\gamma_0 = 0 = w_0\beta_0$.

[12]An alternate approach is to circumvent the breakdown by a "look ahead" technique [90, 176].

These are elegant formulas. Unfortunately the process suffers from steady loss of biorthogonality, just as the Arnoldi process suffers from loss of orthogonality. If one wishes to keep biorthonormality intact, one must add a re-biorthogonalization at each step.

In the case $A^* = A$, the recurrences (9.6.10) and (9.6.11) are both driven by the same matrix A, (9.6.10) is identical in appearance to the symmetric Lanczos recurrence (9.4.9), and (9.6.11) has the same general look. If we start the unsymmetric Lanczos process with $w_1 = u_1$, then we get $w_{j+1} = u_{j+1}$ at every step, provided that we make the normalization (9.6.6) in such a way that $h_{j+1,j} = g_{j+1,j} > 0$, i.e. $\beta_j = \gamma_j > 0$. This is accomplished by taking $\beta_j = \gamma_j = \| \widehat{u}_{j+1} \| = \| \widehat{w}_{j+1} \|$. Then (9.6.10) and (9.6.11) become literally identical to each other and to the symmetric Lanczos recurrence (9.4.9). Thus the unsymmetric Lanczos process reduces to the symmetric Lanczos process in this case. The Arnoldi process and the unsymmetric Lanczos process are two very different extensions of the symmetric Lanczos process.

The real case

From this point on we will restrict our attention to real quantities. Thus we have $A \in \mathbb{R}^{n\times n}$ and $u_1, v_1 \in \mathbb{R}^n$. All quantities produced by the unsymmetric Lanczos process are then real, assuming that β_j and γ_j ($h_{j+1,j}$ and $g_{j+1,j}$) are always chosen to be real in (9.6.6). Let us take a closer look at the normalization step. If the real number $\tau_{j+1} = \langle \widehat{u}_{j+1}, \widehat{w}_{j+1} \rangle$ is not zero, then it is either positive or negative. In the former case we can satisfy $\beta_j \gamma_j = \tau_{j+1}$ by taking $\beta_j = \gamma_j = \sqrt{\tau_{j+1}}$; in the latter case we can take $\beta_j = \sqrt{-\tau_{j+1}} = -\gamma_j$. In either event we have $|\beta_j| = |\gamma_j|$, which implies that the matrix

$$H_k = \begin{bmatrix} \alpha_1 & \gamma_1 & & & \\ \beta_1 & \alpha_2 & \gamma_2 & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \gamma_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{bmatrix}$$

is pseudosymmetric (4.9.3). Thus it can be expressed as a product $H_k = T_k D_k$, where T_k is symmetric and tridiagonal and D_k is a signature matrix:

$$T_k = \begin{bmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & b_2 & a_3 & \ddots & \\ & & \ddots & \ddots & b_{k-1} \\ & & & b_{k-1} & a_k \end{bmatrix}, \quad D_k = \begin{bmatrix} d_1 & & & & \\ & d_2 & & & \\ & & d_3 & & \\ & & & \ddots & \\ & & & & d_k \end{bmatrix},$$

with $d_j = \pm 1$ for $j = 1, \ldots, k$. We also have $G_k = H_k^T = D_k T_k$.

Now let us see what the unsymmetric Lanczos process looks like in this new notation. The recurrences (9.6.10) and (9.6.11) become

$$u_{j+1} b_j d_j = A u_j - u_j a_j d_j - u_{j-1} b_{j-1} d_j \qquad (9.6.12)$$

and

$$w_{j+1}d_{j+1}b_j = A^T w_j - w_j d_j a_j - w_{j-1}d_{j-1}b_{j-1}, \tag{9.6.13}$$

respectively. At the beginning of the jth step we will already have computed the quantities u_{j-1}, u_j, w_{j-1}, w_j, d_j, d_{j-1}, and b_{j-1}. Taking the inner product of (9.6.12) with w_j or of (9.6.13) with u_j and invoking biorthogonality, we find that

$$a_j = d_j^{-1}\langle Au_j, w_j\rangle = d_j^{-1}\langle A^T w_j, u_j\rangle = d_j^{-1}w_j^T A u_j,$$

in agreement with (9.6.2). Using this formula to compute a_j, we now have enough information to compute the right-hand sides of (9.6.12) and (9.6.13). As before, let $\widehat{u}_{j+1}$ and $\widehat{w}_{j+1}$ denote the vectors that result from these two computations, and let $\tau_{j+1} = \langle \widehat{u}_{j+1}, \widehat{w}_{j+1}\rangle$. The condition $\langle u_{j+1}, w_{j+1}\rangle = 1$ implies that $\tau_{j+1} = b_j^2 d_j d_{j+1}$, which can be used to determine b_j and d_{j+1}. If $\tau_{j+1} > 0$, we take $b_j = +\sqrt{\tau_{j+1}}$ and $d_{j+1} = d_j$; if $\tau_{j+1} < 0$, we take $b_j = +\sqrt{-\tau_{j+1}}$ and $d_{j+1} = -d_j$. We can then compute $u_{j+1} = \widehat{u}_{j+1}/(b_j d_j)$ and $w_{j+1} = \widehat{w}_{j+1}/(b_j d_{j+1})$. These considerations give the following algorithm.

Unsymmetric Lanczos process without reorthogonalization
Start with initial vectors u_1 and w_1 satisfying $w_1^T u_1 = 1$.

```
$d_1 \leftarrow \pm 1$
for $j = 1, 2, 3, \ldots$
    $y \leftarrow Au_j$
    $a_j \leftarrow d_j w_j^T y$
    $u_{j+1} \leftarrow y - u_j a_j d_j$
    $w_{j+1} \leftarrow A^T w_j - w_j d_j a_j$
    if $j > 1$ then
        $u_{j+1} \leftarrow u_{j+1} - u_{j-1} b_{j-1} d_j$
        $w_{j+1} \leftarrow w_{j+1} - w_{j-1} d_{j-1} b_{j-1}$
    $\tau_{j+1} \leftarrow w_{j+1}^T u_{j+1}$                              (9.6.14)
    if $\tau_{j+1} = 0$ then
        stop (breakdown)
    if $\tau_{j+1} > 0$ then
        $b_j \leftarrow \sqrt{\tau_{j+1}}$, $d_{j+1} \leftarrow d_j$
    if $\tau_{j+1} < 0$ then
        $b_j \leftarrow \sqrt{-\tau_{j+1}}$, $d_{j+1} \leftarrow -d_j$
    $u_{j+1} \leftarrow u_{j+1}/(b_j d_j)$
    $w_{j+1} \leftarrow w_{j+1}/(d_{j+1} b_j)$
```

It does not matter whether d_1 is taken to be $+1$ or -1. Reversing d_1 has no essential effect on the algorithm: All computed quantities are the same as before, except that minus signs are introduced in a systematic way.

If we want to keep the two sequences biorthogonal in practice, we must include reorthogonalization. Instead of rewriting the algorithm completely, we will just indicate the changes that would need to be made in (9.6.14). Immediately before the line that computes

τ_{j+1}, insert the following commands.

$$\begin{aligned} s_{1:j} &\leftarrow W_j^T u_{j+1}, \quad t_{1:j} \leftarrow U_j^T w_{j+1} \\ u_{j+1} &\leftarrow u_{j+1} - U_j s_{1:j} \\ w_{j+1} &\leftarrow w_{j+1} - W_j t_{1:j} \\ a_j &\leftarrow a_j + s_j d_j \end{aligned}$$

and insert updates $U_{j+1} \leftarrow \begin{bmatrix} U_j & u_{j+1} \end{bmatrix}$ and $W_{j+1} \leftarrow \begin{bmatrix} W_j & w_{j+1} \end{bmatrix}$ in the appropriate places.

Implicit Restarts

Either of the two implicit restart methods described in Section 9.2 can be applied to the unsymmetric Lanczos process. In this case we want to preserve the pseudosymmetry of $H_k = T_k D_k$, so we use transformations that preserve that structure. Thus the HR algorithm is the right version of GR to use here [106].

Let's look at the second restart method in some detail. In our current situation, the equations (9.6.8) and (9.6.9), which become

$$AU_k = U_k T_k D_k + u_{k+1} b_k d_k e_k^T \tag{9.6.15}$$

and

$$A^T W_k = W_k D_k T_k + w_{k+1} d_{k+1} b_k e_k^T. \tag{9.6.16}$$

in our new notation, correspond to (9.3.12). We can use the HR algorithm, implemented as an HZ algorithm as described in Section 6.4, to diagonalize $T_k D_k$ and $D_k T_k$. Specifically, the HZ algorithm produces H, T, and D such that T is block diagonal, D is a signature matrix,

$$T = H^{-1} T_k H^{-T} \qquad \text{and} \qquad D = H^T D_k H.$$

Thus

$$TD = H^{-1}(T_k D_k) H \qquad \text{and} \qquad DT = H^T (D_k T_k) H^{-T}.$$

Both TD and DT are block diagonal with a 2×2 block for each complex conjugate pair of eigenvalues and a 1×1 block for each real eigenvalue. The m eigenvalues that we wish to retain should be at the top of TD and DT. Notice that if they are not at the top, the necessary swaps are easily made in this case, because the matrices are block diagonal (not merely triangular). We have $D_k = HDH^T$ and $T_k = HTH^T$. If we wish to swap the real eigenvalues in positions (i, i) and (j, j) of TD and DT, we simply swap the corresponding entries of T and of D while at the same time swapping the ith and jth columns of H. Swapping complex conjugate pairs is no harder, nor is swapping a complex pair with a real eigenvalue.

Multiply (9.6.15) on the right by H and (9.6.16) on the right by H^{-T} to obtain

$$A\tilde{U}_k = \tilde{U}_k TD + u_{k+1}\tilde{z}^T, \tag{9.6.17}$$

$$A^T \tilde{W}_k = \tilde{W}_k DT + w_{k+1}\tilde{y}^T, \tag{9.6.18}$$

where $\tilde{U}_k = U_k H$, $\tilde{W}_k H^{-T}$, $\tilde{z}^T = b_k d_k e_k^T H$, and $\tilde{y}^T = d_{k+1} b_k e_k^T H^{-T}$. Notice that this transformation preserves biorthonormality: $\tilde{W}_k^T \tilde{U}_k = I$, $\tilde{W}_k^T u_{k+1} = 0$, and $\tilde{U}_k^T w_{k+1} = 0$.

Moreover, $\widetilde{y}^T$ is closely related to $\widetilde{z}^T$. The equation $D_k = HDH^T$ implies that $H^{-T} = D_k HD$. Therefore $\widetilde{y}^T = d_{k+1}\widetilde{z}^T D$. Since $d_{k+1} = \pm 1$ and D is a signature matrix, this implies that the entries of $\widetilde{y}$ and $\widetilde{z}$ are the same up to signs. Thus if the first few entries of $\widetilde{z}$ are small enough to be set to zero, indicating convergence and locking of an invariant subspace as in (9.3.17), then so are the first few entries of $\widetilde{y}$. Thus, whenever we lock in a subspace that is invariant under A, we also lock in a subspace that is invariant under A^T. Locking was discussed in Section 9.2 and will not be revisited here.

We now restart by retaining only the first m columns of (9.6.17) and (9.6.18):

$$A\widetilde{U}_m = \widetilde{U}_m T_m D_m + u_{k+1} z^T, \tag{9.6.19}$$

$$A^T \widetilde{W}_m = \widetilde{W}_m D_m T_m + w_{k+1} y^T, \tag{9.6.20}$$

where $y^T = d_{k+1} z^T D_m$. In order to resume the nonsymmetric Lanczos process, we must transform these equations to a form like

$$A\widehat{U}_m = \widehat{U}_m \widehat{T}_m \widehat{D}_m + \widehat{u}_{m+1}\widehat{b}_m \widehat{d}_m e_m^T,$$

$$A^T \widehat{W}_m = \widehat{W}_m \widehat{D}_m \widehat{T}_m + \widehat{w}_{m+1}\widehat{d}_{m+1}\widehat{b}_m e_m^T,$$

in which z^T and y^T have been transformed to multiples of e_m^T. Let V_1 be an elimination matrix such that $z^T V_1 = \alpha e_m^T$. Assume further that V_1 has been constructed in such a way that $V_1^T D_m V_1 = \check{D}_m$ is another signature matrix. Such a V_1 can (almost always) be constructed as a product of hyperbolic and orthogonal transformations, essentially as described in the proof of Theorem 3.1.5. Since $V_1^{-T} = D_m V_1 \check{D}_m$, we also have $y^T V_1^{-T} = d_{k+1} z^T D_m^2 V_1 \check{D}_m = d_{k+1}\check{d}_m \alpha e_m^T$, so $y^T V_1^{-T} = \beta e_m^T$, where $\beta = d_{k+1}\check{d}_m \alpha$. Note that $|\beta| = |\alpha|$

Let $\check{U}_m = \widetilde{U}_m V_1$, $\check{W}_m = \widetilde{W}_m V_1^{-T}$, and $\check{T}_m = V_1^{-1} T_m V_1^{-T}$. Then, multiplying (9.6.19) on the right by V_1 and (9.6.20) on the right by V_1^{-T}, we obtain

$$A\check{U}_m = \check{U}_m \check{T}_m \check{D}_m + u_{k+1}\alpha e_m^T, \tag{9.6.21}$$

and

$$A^T \check{W}_m = \check{W}_m \check{D}_m \check{T}_m + w_{k+1}\beta e_m^T. \tag{9.6.22}$$

This is almost the form we want; the only problem is that $\check{T}_m$ is not tridiagonal, although it is symmetric. The final task of the restart is to return $\check{T}_m$ to tridiagonal form, row by row, from bottom to top, by a congruence transformation that preserves the signature-matrix form of $\check{D}_m$. Call the transforming matrix V_2, constructed so that $\widehat{T}_m = V_2^{-1}\check{T}_m V_2^{-T}$ is tridiagonal and $\widehat{D}_m = V_2^{-1}\check{D}_m V_2^{-T} = V_2^T \check{D}_m V_2$ is a signature matrix. V_2 can be constructed as a product of $m-2$ matrices essentially of the form described in Theorem 3.1.5. By construction V_2 has the form

$$V_2 = \begin{bmatrix} \widetilde{V}_2 & 0 \\ 0 & 1 \end{bmatrix},$$

so $e_m^T V_2 = e_m^T$ and $e_m^T V_2^{-T} = e_m^T$. Multiplying (9.6.21) and (9.6.22) on the right by V_2 and V_2^{-T}, respectively, we obtain

$$A\widehat{U}_m = \widehat{U}_m \widehat{T}_m \widehat{D}_m + \widehat{u}_{m+1}\widehat{b}_m \widehat{d}_m e_m^T, \tag{9.6.23}$$

$$A^T \widehat{W}_m = \widehat{W}_m \widehat{D}_m \widehat{T}_m + \widehat{w}_{m+1} \widehat{d}_{m+1} \widehat{b}_m e_m^T, \tag{9.6.24}$$

where $\widehat{U}_m = \check{U}_m V_2$, $\widehat{W}_m = \check{W}_m V_2^{-1}$, $\widehat{u}_{m+1} = u_{k+1}$, $\widehat{w}_{m+1} = w_{k+1}$, $\widehat{d}_m$ is the last diagonal entry in $\widehat{D}_m$, $\widehat{b}_m = \alpha/\widehat{d}_m$, and $\widehat{d}_{m+1} = \beta/\widehat{b}_m = \pm 1$. These computations are consistent because $|\beta| = |\alpha|$. Notice that biorthonormality has been preserved all along: $\widehat{W}_m^T \widehat{U}_m = I$, $\widehat{W}_m^T \widehat{u}_{m+1} = 0$, and $\widehat{U}_m^T \widehat{w}_{m+1} = 0$.

Equations (9.6.23) and (9.6.24) are now in a form where we can pick up the unsymmetric Lanczos process at step $m+1$. The implicit restart is complete.

An alternate way of generating the transformations V_1 and V_2 that saves a few flops and has a certain aesthetic appeal is sketched in Exercise 9.6.5.

We must not neglect to remind the reader that the unsymmetric Lanczos process can break down along the way. The implicit restart process can also break down, as it relies on hyperbolic transformations. It is therefore important to keep the Lanczos runs fairly short and restart frequently. This will guarantee that the probability of a breakdown on any given run is slight. Moreover, in the rare event of a breakdown, not much computational effort will have been invested in that particular short run. When a breakdown does occur, the process needs to be restarted somehow. One could start from scratch or think of some more clever way of continuing without losing everything. It is clear at least that any vectors that have converged and been locked in can be kept. We have not invested much thought in the question of emergency restarts because, in our experience with this and related algorithms, breakdowns are rare.

After a number of restarts, the Lanczos process will normally produce an invariant subspace of the desired dimension. Let's say that dimension is $\widehat{m} \le m$. The final configuration is

$$AU_{\widehat{m}} = U_{\widehat{m}} T_{\widehat{m}} D_{\widehat{m}}, \qquad A^T W_{\widehat{m}} = W_{\widehat{m}} D_{\widehat{m}} T_{\widehat{m}}.$$

$\mathcal{R}(U_{\widehat{m}})$ is invariant under A, and $\mathcal{R}(W_{\widehat{m}})$ is invariant under A^T. Moreover, the right and left eigenvectors of A associated with $\mathcal{R}(U_{\widehat{m}})$ are normally readily accessible in $U_{\widehat{m}}$ and $W_{\widehat{m}}$, respectively. This is so because T_m is normally block diagonal with a 1×1 block corresponding to each real eigenvalue of $T_{\widehat{m}} D_{\widehat{m}}$ and a 2×2 block corresponding to each complex conjugate pair. If the (i, i) position of $T_{\widehat{m}} D_{\widehat{m}}$ houses a real eigenvalue of A, then the ith column of $U_{\widehat{m}}$ is a right eigenvector of A associated with that eigenvalue, and the ith column of $W_{\widehat{m}}$ is a left eigenvector. If $T_{\widehat{m}} D_{\widehat{m}}$ has a 2×2 block in positions i and $i+1$, then columns i and $i+1$ of $U_{\widehat{m}}$ span the invariant subspace of A associated with that pair. Complex eigenvectors can be extracted easily. Similarly, complex left eigenvectors can be extracted from columns i and $i+1$ of $W_{\widehat{m}}$.

Since we have eigenvectors accessible, we can easily compute a residual for each eigenpair. This constitutes an *a posteriori* test of backward stability (Proposition 2.7.20). If the residual is not tiny, the results cannot be trusted. Since the Lanczos process is potentially unstable, this test is absolutely essential and should never be omitted.

Since the algorithm delivers both right and left eigenvectors, we can easily compute the condition number of each of the computed eigenvalues (2.7.4). If the residual is tiny and the condition number is not too big, the eigenvalue is guaranteed accurate.

Exercises

Exercise 9.6.1.

(a) Let $u_1, u_2, \ldots u_j$ and $w_1, w_2, \ldots, w_j$ be two sequences of vectors in $\mathbb{C}^n$, and let $U_j = \begin{bmatrix} u_1 & \cdots & u_j \end{bmatrix}$ and $W_j = \begin{bmatrix} w_1 & \cdots & w_j \end{bmatrix} \in \mathbb{C}^{n \times j}$. Show that the two sequences are biorthonormal if and only if $W_j^* U_j = I_j$.

(b) Given vectors $u_1, \ldots, u_n \in \mathbb{C}^n$, show that there exists a sequence $w_1, \ldots, w_n$ that is biorthonormal to $u_1, \ldots, u_n$ if and only if the matrix $U = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix}$ is nonsingular, that is, the vectors $u_1, \ldots, u_n$ are linearly independent.

(c) State and prove a generalization of the result in part (b) that is valid for a set of j vectors $u_1, \ldots, u_j$, where $j \leq n$.

Since linear independence is a much weaker property than orthonormality, these results show that biorthonormality is much weaker than orthonormality.

Exercise 9.6.2. This exercise works out some of the basic properties of the unsymmetric Lanczos process.

(a) If $\widehat{u}_{j+1}$ is given by (9.6.3), show that $\widehat{u}_{j+1}$ is orthogonal to w_k if and only if $h_{kj} = \langle Au_j, w_k \rangle$. Similarly, show that $\widehat{w}_{j+1}$ is orthogonal to u_k if and only if $g_{kj} = \langle A^* w_j, u_k \rangle$.

(b) Show that if $\tau_{j+1} \neq 0$ in (9.6.5), and $h_{j+1,j}$ and $g_{j+1,j}$ are chosen so that $h_{j+1,j}\overline{g_{j+1,j}} = \tau_{j+1}$, then the vectors defined by (9.6.6) satisfy $\langle u_{j+1}, w_{j+1} \rangle = 1$, and the sequences $u_1, \ldots, u_{j+1}$ and $w_1, \ldots, w_{j+1}$ are biorthonormal.

(c) Prove by induction on j that (if $\tau_{j+1} \neq 0$ at every step), the unsymmetric Lanczos process (9.6.2 – 9.6.6) produces vectors satisfying $\operatorname{span}\{u_1, \ldots, u_j\} = \mathcal{K}_j(A, u_1)$ and $\operatorname{span}\{w_1, \ldots, w_j\} = \mathcal{K}_j(A^*, w_1)$, for $j = 1, 2, 3, \ldots$.

(d) Show that if $Au_j \in \operatorname{span}\{u_1, \ldots, u_j\}$, then this fact and the relationships $\langle \widehat{u}_{j+1}, w_k \rangle = 0$, $k = 1, \ldots, j$ imply that $\widehat{u}_{j+1} = 0$.

(e) Show that if $Aw_j \in \operatorname{span}\{w_1, \ldots, w_j\}$, then $\widehat{w}_{j+1} = 0$.

Exercise 9.6.3.

(a) Prove (9.6.7), (9.6.8), and (9.6.9).

(b) Show that $W_k^* U_k = I_k$, $W_k^* u_{k+1} = 0$, and $U_k^* w_{k+1} = 0$.

(c) Use (9.6.8) and (9.6.9) to show that $G_k = H_k^*$.

Exercise 9.6.4. Verify that the unsymmetric Lanczos process reduces to the symmetric Lanczos process in the case $A^* = A$ if we start with $w_1 = u_1$ and take $\beta_j = \gamma_j = \| \widehat{u}_{j+1} \|$ at each step.

Exercise 9.6.5. This exercise sketches an economical way to build the transforming matrix $V_1 V_2$. The challenge is this. Given a diagonal matrix T_m, a signature matrix D_m and $z \in \mathbb{C}^n$, construct V so that $\widehat{T}_m = V^{-1} T_m V^{-T}$ is tridiagonal, $\widehat{D}_m = V^{-1} D_m V^{-T} = V^T D_m V$ is a signature matrix, and $z^T V = \alpha e_m^T$. We will produce V as a product of essentially 2×2 transformations, each of which is either orthogonal or hyperbolic. It is not claimed that the construction will always work; it will break down whenever a hyperbolic transformation that is needed does not exist. The idea is to create zeros in z one at a time, at the same time keeping the T matrix from filling in.

(a) Show that there is a matrix $\widehat{V}$, a product of $n-1$ essentially 2×2 matrices such that $z^T \widehat{V} = \alpha e_m^T$ and $\widehat{V}^T D_m \widehat{V}$ is a signature matrix. The first transformation transforms z_1 to zero. It is hyperbolic if $d_1 = -d_2$ and orthogonal if $d_1 = d_2$.

(b) The matrix $\widehat{V}^{-1} T_m \widehat{V}^{-T}$ is far from tridiagonal. We seek to intersperse additional factors among the factors of $\widehat{V}$ in such a way that the tridiagonal form is preserved. Show that after the first two factors of $\widehat{V}$ have been applied to T_m, the resulting matrix is not tridiagonal, as it has a "bulge" at positions $(3,1)$ and $(1,3)$. Show that that bulge can be chased away by a single transformation acting in the $(1,2)$ plane, which does not disturb the zeros that have been introduced into z.

(c) Now suppose we have chased that first "bulge" and now proceed to create one more zero in the z vector by applying the third factor of $\widehat{V}$. Show that the corresponding transformation of the T matrix make that matrix fail to be tridiagonal because of a bulge in positions $(4,2)$ and $(2,4)$. Show that that bulge can be removed from the matrix by an upward bulge chase involving transformations in the $(2,3)$ and $(1,2)$ planes. These transformations do not disturb the zeros that have been created in z.

(d) Now suppose the first j entries of z have been transformed to zero, the top $(j+1) \times (j+1)$ submatrix of the transformed T matrix is tridiagonal, and the bottom $(m-j-1) \times (m-j-1)$ submatrix of the transformed T is still diagonal. Show that the transformation that zeros out the $j+1$st entry of the z matrix also introduces a "bulge" in positions $(j+2, j)$ and $(j, j+2)$. Show that this bulge can be chased from the matrix by an upward bulge chase (an implicit HZ iteration in reverse) that does not disturb the zeros that have been introduced into z. The result is a transformed T matrix whose top $(j+2) \times (j+2)$ submatrix is tridiagonal and whose bottom $(m-j-2) \times (m-j-2)$ submatrix is still diagonal.

(e) What does the final step of the process look like?

Notice that this procedure is about the same as the procedure for returning a unitary matrix to Hessenberg form that was sketched in Section 9.5. The methodology is not specific to the unitary and tridiagonal cases; it can be applied to implicit restarts by the second method in any context.

9.7 Skew-Hamiltonian Krylov Processes

In this and the next few sections we draw heavily from [222].

Hamiltonian and related structures

In [9] and [153] we studied large eigenvalue problems in which the smallest few eigenvalues of a Hamiltonian matrix are needed. Since these are also the largest eigenvalues of H^{-1}, a Krylov subspace method can be applied to H^{-1} to find them, provided H^{-1} is accessible. Since H^{-1} inherits the Hamiltonian structure of H, we prefer to use a procedure that respects the Hamiltonian structure, in the interest of efficiency, stability, and accuracy. Eigenvalues of real Hamiltonian matrices occur in pairs $\{\lambda, -\lambda\}$ or quadruples $\{\lambda, -\lambda, \overline{\lambda}, -\overline{\lambda}\}$. A structure-preserving method will extract entire pairs and quadruples intact.

In situations where we have some prior information, we might prefer to shift before we invert. Specifically, if we know that the eigenvalues of interest lie near τ, we might prefer to work with $(H - \tau I)^{-1}$. Unfortunately, the shift destroys the structure, so we are lead to think of ways of effecting shifts without destroying the structure. One simple remedy is to work with the matrix

$$(H - \tau I)^{-1}(H + \tau I)^{-1},$$

which is not Hamiltonian but skew Hamiltonian. It makes perfect sense that the shifts τ and $-\tau$ should be used together in light of the symmetry of the spectrum of H. If τ is neither real nor purely imaginary, we prefer to work with the Hamiltonian matrix

$$(H - \tau I)^{-1}(H - \overline{\tau} I)^{-1}(H + \tau I)^{-1}(H + \overline{\tau} I)^{-1},$$

in order to stay within the real number system. If we want to apply either of these transformations, we would like to use a Krylov process that preserves skew-Hamiltonian structure.

If we wish to use a shift *and* keep the Hamiltonian structure, we can work with the Hamiltonian matrix

$$H^{-1}(H - \tau I)^{-1}(H + \tau I)^{-1}$$

or

$$H(H - \tau I)^{-1}(H + \tau I)^{-1},$$

for example.

Another possibility is to work with the Cayley transform

$$(H - \tau I)^{-1}(H + \tau I),$$

which is symplectic. To attack this one, we need a Krylov process that preserves symplectic structure.

In this and the next two sections we will develop Krylov processes that preserve these three related structures. We will begin with skew-Hamiltonian structure, which is easiest. Once we have developed a skew-Hamiltonian Lanczos process, we will find that we can derive Hamiltonian and symplectic Lanczos processes from it with amazingly little effort.

Recall from Exercise 2.2.17 that a similarity transformation preserves skew-Hamiltonian or Hamiltonian or symplectic structure if the transforming matrix is symplectic. As we observed in Section 9.2, Krylov processes effect partial similarity transformations. Therefore, a Krylov process can preserve any of the three structures if it produces vectors that are the columns of a symplectic matrix.

We ask therefore what special properties the columns of a symplectic matrix have. Recall that a matrix $S \in \mathbb{R}^{2n \times 2n}$ is symplectic if $S^T J S = J$, where J is given by

$$J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}.$$

Partitioning S as $S = \begin{bmatrix} S_1 & S_2 \end{bmatrix}$, where S_1 and S_2 are both $2n \times n$, we see immediately from the equation $S^T J S = J$ that $S_1^T J S_1 = 0$, $S_2^T J S_2 = 0$, and $S_1^T J S_2 = I$.

Recall that a subspace $\mathcal{S} \subseteq \mathbb{R}^{2n}$ is called *isotropic* if $x^T J y = 0$ for all $x, y \in \mathcal{S}$. A subspace $\mathcal{S} = \mathcal{R}(X)$ is isotropic if and only if $X^T J X = 0$. It follows that if $S \in \mathbb{R}^{2n \times 2n}$ be symplectic, then the space spanned by the first n columns of S is isotropic, and so is the space spanned by the last n columns. These subspaces are maximal, as the dimension of an isotropic subspace of $\mathbb{R}^{2n}$ cannot exceed n:

Proposition 9.7.1. *Let $\mathcal{S}$ be an isotropic subspace of $\mathbb{R}^{2n}$. Then* $\dim(\mathcal{S}) \le n$.

Proof. Let $\mathcal{S}$ be an isotropic subspace of $\mathbb{R}^{2n}$ of dimension k, and let $q_1, \ldots, q_k$ be an orthonormal basis of $\mathcal{S}$. Then $q_1, \ldots, q_k, Jq_1, \ldots, Jq_k$ is an orthonormal set in $\mathbb{R}^{2n}$, so $2k \le 2n$. Thus $k \le n$. $\square$

Now we come to the special property of skew-Hamiltonian matrices.

Proposition 9.7.2. *Let $B \in \mathbb{R}^{2n\times 2n}$ be skew Hamiltonian, and let $u \in \mathbb{R}^{2n}$ be an arbitrary nonzero vector. Then the Krylov subspace $\mathcal{K}_j(B, u)$ is isotropic for all j.*

Proof. Since B is skew Hamiltonian, one easily shows that all of its powers B^i, $i = 0, 1, 2, 3, \ldots$, are also skew Hamiltonian. This means that JB^i is skew symmetric. To establish isotropy of $\mathcal{K}_j(B, u)$, it suffices to prove that $(B^i u)^T JB^k u = 0$ for all $i \ge 0$ and $k \ge 0$. Since $B^T J = JB$, we have $(B^i u)^T JB^k u = u^T JB^{i+k} u$, which equals zero because JB^{i+k} is skew symmetric. $\square$

Proposition 9.7.2 shows that every Krylov process driven by a skew-Hamiltonian matrix automatically produces vectors that span isotropic subspaces. Thus we should have no trouble finding Krylov processes that preserve skew-Hamiltonian structure. Notice that this property guarantees that a skew-Hamiltonian Krylov process cannot be run to completion. Since an isotropic subspace cannot have dimension greater than n, we conclude that the Krylov space $\mathcal{K}_{n+1}(B, u_1)$ has dimension at most n. This implies that $\mathcal{K}_{n+1}(B, u_1) = \mathcal{K}_n(B, u_1)$ and $\mathcal{K}_n(B, u_1)$ is invariant under B. Thus a skew-Hamiltonian Krylov process must terminate with an invariant subspace in at most n iterations.

Skew-Hamiltonian Arnoldi process

When we run the Arnoldi process on a skew-Hamiltonian matrix, we find that it does indeed produce isotropic subspaces. However, over the course of the iterations the isotropy gradually deteriorates in just the same way as orthogonality is lost if reorthogonalization is not used. In order to preserve isotropy, we must enforce it explicitly, as shown in the following algorithm.

Skew-Hamiltonian Arnoldi process.

```
$u_1 \leftarrow x/\|x\|$
$U_1 \leftarrow [u_1]$
for $j = 1, 2, 3, \ldots$
    $u_{j+1} \leftarrow Bu_j$   ($B$ is skew Hamiltonian)
    $h_{1:j,j} \leftarrow U_j^T u_{j+1}$
    $u_{j+1} \leftarrow u_{j+1} - U_j h_{1:j,j}$   (orthogonalize)
    $\delta_{1:j} \leftarrow U_j^T u_{j+1}$
    $u_{j+1} \leftarrow u_{j+1} - U_j \delta_{1:j}$   (reorthogonalize)
    $h_{1:j,j} \leftarrow h_{1:j,j} + \delta_{1:j}$                    (9.7.1)
    $\gamma_{1:j} \leftarrow (JU_j)^T u_{j+1}$
    $u_{j+1} \leftarrow u_{j+1} - JU_j\gamma_{1:j}$   (enforce isotropy)
    $h_{j+1,j} \leftarrow \|u_{j+1}\|$
    if $h_{j+1,j} = 0$
        set flag (span$\{u_1, \ldots, u_j\}$ is invariant)
        exit
    $u_{j+1} \leftarrow u_{j+1}/h_{j+1,j}$
    $U_{j+1} \leftarrow \begin{bmatrix} U_j & u_{j+1} \end{bmatrix}$
```

This is just like the ordinary Arnoldi process,except that the two lines

$$\begin{aligned} \gamma_{1:j} &\leftarrow (JU_j)^T u_{j+1} \\ u_{j+1} &\leftarrow u_{j+1} - JU_j\gamma_{1:j} \end{aligned}$$

have been added. These lines ensure that u_{j+1} is orthogonal to $Ju_1, \ldots, Ju_j$, which is what isotropy means. In principle the numbers in $\gamma_{1:j}$ should all be zero, but in practice they are tiny numbers on the order of 10^{-15}. The tiny correction at each step ensures that the subspace stays isotropic to working precision. When this algorithm is run with restarts, it becomes the *skew-Hamiltonian implicitly-restarted Arnoldi* (SHIRA) method [153]. The enforcement of isotropy is crucial to the efficiency of the algorithm. If it is not done, the eigenvalues will show up in duplicate. For example, if eight eigenvalues are wanted, four distinct eigenvalues will be produced, each in duplicate. (Recall that all eigenvalues of a skew-Hamiltonian matrix have multiplicity two or greater.) If, on the other hand, isotropy is enforced, eight distinct eigenvalues are produced.

Skew-Hamiltonian Lanczos process

If the unsymmetric Lanczos process is applied to a skew-Hamiltonian matrix B, then the spaces $\mathcal{K}_j(B, u_1) = \text{span}\{u_1, \ldots, u_j\}$ are all isotropic, and so are the spaces $\mathcal{K}_j(B^T, w_1) = \text{span}\{w_1, \ldots, w_j\}$, because B^T is also skew Hamiltonian.

Suppose we are able to run the algorithm for n steps, terminating in invariant subspaces $\mathcal{K}_n(B, u_1) = \text{span}\{u_1, \ldots, u_n\} = \mathcal{R}(U_n)$ and $\mathcal{K}_n(B^T, w_1) = \text{span}\{w_1, \ldots, w_n\} = \mathcal{R}(W_n)$. Then $BU_n = U_nT_nD_n$ and $B^TW_n = W_nD_nT_n$, as the remainder terms are zero. Since we can go no further than this, let us make the following abbreviations: $U = U_n \in \mathbb{R}^{2n\times n}$, $W = W_n \in \mathbb{R}^{2n\times n}$, $T = T_n \in \mathbb{R}^{n\times n}$, and $D = D_n \in \mathbb{R}^{n\times n}$. Now we have

$$BU = U(TD), \tag{9.7.2}$$

$$B^T W = W(DT), \tag{9.7.3}$$

$$U^T W = I, \tag{9.7.4}$$

and by isotropy,

$$U^T JU = 0 \quad \text{and} \quad W^T JW = 0. \tag{9.7.5}$$

These equations represent only partial similarities. We can build full similarity transformations by exploiting the structure. Since $U^T W = I$, if we define a new matrix V by

$$V = -JW, \tag{9.7.6}$$

then we immediately have $U^T JV = I$. Since also $V^T JV = 0$ (isotropy condition inherited from W), the matrix $\begin{bmatrix} U & V \end{bmatrix}$ is symplectic.

Since B^T is skew Hamiltonian, we have $JB^T = BJ$. Multiplying (9.7.3) by $-J$ on the left, we obtain

$$BV = V(DT), \tag{9.7.7}$$

Now we can combine (9.7.2) and (9.7.7) to obtain the full similarity transformation

$$B\begin{bmatrix} U & V \end{bmatrix} = \begin{bmatrix} U & V \end{bmatrix}\begin{bmatrix} TD & \\ & DT \end{bmatrix}. \tag{9.7.8}$$

Equation (9.7.8) holds only if we are able to reach the nth step with no breakdowns. Even if we do not get that far, we still obtain partial results. After k steps the Lanczos configuration is

$$BU_k = U_k T_k D_k + u_{k+1} b_k d_k e_k^T, \tag{9.7.9}$$

$$B^T W_k = W_k D_k T_k + w_{k+1} d_{k+1} b_k e_k^T. \tag{9.7.10}$$

If we multiply the second of these equations by $-J$ on the left, we obtain

$$BV_k = V_k D_k T_k + v_{k+1} d_{k+1} b_k e_k^T. \tag{9.7.11}$$

Equations (9.7.9) and (9.7.11) form the basis for our skew-Hamiltonian Lanczos process. The associated recurrences are

$$u_{j+1} b_j d_j = Bu_j - u_j a_j d_j - u_{j-1} b_{j-1} d_j, \tag{9.7.12}$$

$$v_{j+1} d_{j+1} d_j = Bv_j - v_j d_j a_j - v_{j-1} d_{j-1} b_{j-1}. \tag{9.7.13}$$

The coefficients are computed just as in the ordinary nonsymmetric Lanczos process (9.6.14), with w_j replaced by Jv_j. Since the unsymmetric Lanczos process starts with vectors satisfying $u_1^T w_1 = 1$, the recurrences (9.7.12) and (9.7.13) must be initialized with vectors satisfying $u_1^T Jv_1 = 1$. Then the vectors produced will automatically satisfy the properties of the columns of a symplectic matrix. One should always bear in mind, however, that roundoff errors will degrade the "symplecticness" of the vectors in practice, unless it is explicitly enforced at each step.

Exercises

Exercise 9.7.1. Let H be a Hamiltonian matrix. In each of the following tasks, assume that the indicated inverse matrices exist.

(a) Prove that H^{-1} is Hamiltonian.

(b) Prove that $(H - \tau I)^{-1}(H + \tau I)^{-1}$ is skew Hamiltonian.

(c) Prove that $H(H - \tau I)^{-1}(H + \tau I)^{-1}$ is Hamiltonian.

(d) Prove that $H^{-1}(H - \tau I)^{-1}(H + \tau I)^{-1}$ is Hamiltonian.

(e) Prove that the Cayley transform $(H - \tau I)^{-1}(H + \tau I)$ is symplectic.

Exercise 9.7.2. Suppose the skew-Hamiltonian Arnoldi process is run for n steps, culminating in an invariant subspace $\mathcal{K}_n(B, u_1) = \operatorname{span}\{u_1, \ldots, u_n\}$. The final Arnoldi configuration is

$$BU = UH,$$

where $U = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix}$, and $H \in \mathbb{R}^{n\times n}$ is upper Hessenberg.

(a) Let $S = \begin{bmatrix} U & -JU \end{bmatrix} \in \mathbb{R}^{2n\times 2n}$. Show that S is both orthogonal and symplectic.

(b) Let $C = S^{-1}BS$. Show that C has the form

$$C = \begin{bmatrix} H & N \\ 0 & H^T \end{bmatrix},$$

where $N^T = -N$.

Thus an orthogonal symplectic similarity transformation has been effected on B. If the Arnoldi process is carried only part way to completion, then a partial orthogonal, symplectic similarity transformation is carried out.

Exercise 9.7.3. With notation as in (9.7.2) through (9.7.5), define $X = JU$.

(a) Show that $\begin{bmatrix} W & X \end{bmatrix}$ is symplectic.

(b) Starting from the equation $BU = U(TD)$, show that $B^T X = X(TD)$. Consequently

$$B^T \begin{bmatrix} W & X \end{bmatrix} = \begin{bmatrix} W & X \end{bmatrix} \begin{bmatrix} DT & \\ & TD \end{bmatrix}.$$

This is another symplectic similarity transformation, which is easily seen to be equivalent to (9.7.8)

(c) Show that $\begin{bmatrix} W & X \end{bmatrix} = \begin{bmatrix} U & V \end{bmatrix}^{-T}$.

Exercise 9.7.4.

(a) Write a pseudocode algorithm like (9.6.14) for the skew-Hamiltonian Lanczos process (9.7.12), (9.7.13).

(b) Write down a version of the algorithm in which the "symplecticness" conditions $U_j^T J U_j = 0$, $V_j^T J V_j = 0$, and $U_j^T J V_j = I$ are explicitly enforced.

9.8 The Hamiltonian Lanczos Process

The Hamiltonian Lanczos process with restarts was first published in [28]. The version presented here is from [222].

Let $H \in \mathbb{R}^{2n\times 2n}$ be Hamiltonian. Then H^2 is skew Hamiltonian, and we will take the skew-Hamiltonian Lanczos process for H^2 as our point of departure. From (9.7.12) and (9.7.13) we have

$$u_{j+1}b_j d_j = H^2 u_j - u_j a_j d_j - u_{j-1}b_{j-1}d_j, \tag{9.8.1}$$

$$v_{j+1}d_{j+1}b_j = H^2 v_j - v_j d_j a_j - v_{j-1}d_{j-1}b_{j-1}, \tag{9.8.2}$$

where the recurrences are started with a pair of vectors satisfying $u_1^T J v_1 = 1$. Define new vectors $\widetilde{v}_j = Hu_j d_j$, $j = 1, 2, 3, \ldots$. Since $d_j = \pm 1$, we also have

$$\widetilde{v}_j d_j = Hu_j. \tag{9.8.3}$$

Multiply (9.8.1) on the left by H and by d_j to obtain

$$\widetilde{v}_{j+1}d_{j+1}b_j = H^2\widetilde{v}_j - \widetilde{v}_j d_j a_j - \widetilde{v}_{j-1}d_{j-1}b_{j-1}, \tag{9.8.4}$$

a recurrence that appears identical to (9.8.2). Now suppose we start (9.8.1) and (9.8.2) with vectors that satisfy both $u_1^T J v_1 = 1$ and $Hu_1 = v_1 d_1$. Then (9.8.2) and (9.8.4) have the same starting vector, and they are identical, so $\widetilde{v}_j = v_j$ for every j. This means that we do not have to run both of the recurrences (9.8.1) and (9.8.2). We can just run (9.8.1) along with the supplementary condition (9.8.3). Noting also that the term $H^2 u_j$ can be replaced by $Hv_j d_j$, we obtain the *Hamiltonian Lanczos process*

$$u_{j+1}b_j = Hv_j - u_j a_j - u_{j-1}b_{j-1}, \tag{9.8.5}$$

$$v_{j+1}d_{j+1} = Hu_{j+1}. \tag{9.8.6}$$

Now we just need to give some thought to how the coefficients are computed. Multiplying (9.8.5) on the left by $v_j^T J$, we find that

$$a_j = -v_j^T JHv_j.$$

The other coefficients are produced in the normalization process. In practice (9.8.5) and (9.8.6) take the form

$$\widehat{u}_{j+1} = Hv_j - u_j a_j - u_{j-1}b_{j-1}, \tag{9.8.7}$$

$$\widehat{v}_{j+1} = H\widehat{u}_{j+1}. \tag{9.8.8}$$

Then $\widehat{u}_{j+1} = u_{j+1}b_j$, and $\widehat{v}_{j+1} = Hu_{j+1}b_j = v_{j+1}d_{j+1}b_j$. We can now determine b_j and d_{j+1} uniquely from the condition $u_{j+1}^T Jv_{j+1} = 1$. Let $\pi = \widehat{u}_{j+1}^T J\widehat{v}_{j+1}$. Then we must have $\pi = d_{j+1}b_j^2$. If $\pi = 0$, we have a breakdown. If $\pi > 0$ we must have $d_{j+1} = 1$ and $b_j = \sqrt{\pi}$, and if $\pi < 0$ we must have $d_{j+1} = -1$ and $b_j = \sqrt{-\pi}$. We then have $u_{j+1} = \widehat{u}_{j+1}/b_j$ and $v_{j+1} = \widehat{v}_{j+1}/(d_{j+1}b_j)$. Now we have all we need for the next step.

Before we can write down the algorithm, we need to say a few words about picking starting vectors. u_1 and v_1 must satisfy both $v_1 d_1 = Hu_1$ and $u_1^T Jv_1 = 1$. We can obtain vectors satisfying these conditions by picking a more or less arbitrary starting vector $\widehat{u}_1$ and using the normalization process described in the previous paragraph. Thus we define $\widehat{v}_1 = H\widehat{u}_1$, $\pi = \widehat{u}_1^T J\widehat{v}_1$, and so on.

Now we are ready to write down the algorithm.

Hamiltonian Lanczos Process.
Start with initial vector u_1.

```
$b_0 \leftarrow 0,\ u_0 \leftarrow 0$
for $j = 0, 1, 2, \ldots,$
    if $j > 0$ then
        $y \leftarrow Hv_j$
        $a_j \leftarrow -v_j^T Jy$
        $u_{j+1} \leftarrow y - u_j a_j - u_{j-1}b_{j-1}$
    $v_{j+1} \leftarrow Hu_{j+1}$
    $\pi \leftarrow u_{j+1}^T J v_{j+1}$
    if $\pi = 0$ then
        stop (breakdown)
    if $\pi > 0$ then
        $s \leftarrow \sqrt{\pi},\ d_{j+1} \leftarrow 1$
    if $\pi < 0$ then
        $s \leftarrow \sqrt{-\pi},\ d_{j+1} \leftarrow -1$
    $u_{j+1} \leftarrow u_{j+1}/s$
    $v_{j+1} \leftarrow v_{j+1}/(d_{j+1}s)$
    if $j > 0$ then
        $b_j \leftarrow s$
```
(9.8.9)

This is the theoretical algorithm. In exact arithmetic it guarantees “symplecticness” of the vectors produced. In the real world of roundoff errors, the symplectic structure will gradually be lost unless it is explicitly enforced. This can be achieved by adding the lines

$$\begin{aligned} z &\leftarrow Ju_{j+1} \\ u_{j+1} &\leftarrow u_{j+1} - V_j(U_j^T z) + U_j(V_j^T z) \\ a_j &\leftarrow a_j + v_j^T z \end{aligned} \tag{9.8.10}$$

immediately after the line $u_{j+1} \leftarrow y - u_j a_j - u_{j-1}b_{j-1}$. Here U_j and V_j have their usual meanings. This step ensures that new vector $\widehat{u}_{j+1}$ satisfies $U_j^T J\widehat{u}_{j+1} = 0$ and $V_j^T J\widehat{u}_{j+1} = 0$ to working precision, so that we have $U_k^T JU_k = 0$ and $U_k^T JV_k = I_k$ at every step. Our experience has been that because of the very tight connection between the “u” and “v” vectors in this algorithm, the condition $V_k^T JV_k = 0$ takes care of itself.

Implicit restarts

Rearranging (9.8.5) and (9.8.6), we obtain

$$\begin{aligned} Hv_j &= u_{j-1}b_{j-1} + u_j a_j + u_{j+1}b_j, \\ Hu_{j+1} &= v_{j+1}d_{j+1}, \end{aligned}$$

which when packed into matrix equations gives

$$HV_k = U_k T_k + u_{k+1} b_k e_k^T, \qquad HU_k = V_k D_k \tag{9.8.11}$$

after $k - 1/2$ steps. Here D_k is the signature matrix $D_k = \operatorname{diag}\{d_1, \ldots, d_k\}$, and T_k is the symmetric, tridiagonal matrix

$$T_k = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & b_{k-1} \\ & & b_{k-1} & a_k \end{bmatrix}$$

as always. Equations (9.8.11) can also be written as the single equation

$$H \begin{bmatrix} U_k & V_k \end{bmatrix} = \begin{bmatrix} U_k & V_k \end{bmatrix} \begin{bmatrix} & T_k \\ D_k & \end{bmatrix} + u_{k+1} b_k e_{2k}^T.$$

The Hamiltonian matrix $\begin{bmatrix} & T_k \\ D_k & \end{bmatrix}$, which is familiar from Section 8.9, tells us how restarts should be done. Either of the restart methods that we have presented can be used. The filtering should be done by the product HR algorithm (= HZ algorithm), as explained in Section 8.9.

Extracting eigenvectors

After a number of restarts, the process will normally converge to an invariant subspace of some dimension m. The result has the form

$$H \begin{bmatrix} U_m & V_m \end{bmatrix} = \begin{bmatrix} U_m & V_m \end{bmatrix} \begin{bmatrix} & T_m \\ D_m & \end{bmatrix}. \tag{9.8.12}$$

where D_m is a signature matrix, and T_m is block diagonal. It is thus a simple matter to determine the eigenpairs of $\begin{bmatrix} & T_m \\ D_m & \end{bmatrix}$. Each eigenvector $\begin{bmatrix} x \\ y \end{bmatrix}$ of $\begin{bmatrix} & T_m \\ D_m & \end{bmatrix}$, corresponding to eigenvalue λ, yields an eigenvector $U_m x + V_m y$ of H corresponding to eigenvalue λ.

It is also a simple matter to get left eigenvectors. Let $W_m = JV_m$ and $X_m = -JU_m$ (consistent with notation used previously). Multiplying the equation (9.8.12) by J on the left and using the fact that H is Hamiltonian, we immediately find that

$$H^T \begin{bmatrix} X_m & W_m \end{bmatrix} = \begin{bmatrix} X_m & W_m \end{bmatrix} \begin{bmatrix} 0 & T_m \\ D_m & 0 \end{bmatrix}.$$

Thus left eigenvectors can be obtained by the same method as we use to compute right eigenvectors.

Playing it safe

Since we are able to get eigenvectors, we should always compute residuals as an *a posteriori* test of backward stability (Proposition 2.7.20). Because the Hamiltonian Lanczos process is potentially unstable, this test should never be skipped.

Since we have both left and right eigenvectors, we can also compute condition numbers of the computed eigenvalues (2.7.4). If the residual is tiny and the condition number is not too big, the eigenvalue is guaranteed accurate.

Exercises

Exercise 9.8.1. The file `hamlan.m` encodes the Hamiltonian Lanczos process. Download this file and the driver program `hamlango.m` from the website and play with them. Try adjusting the matrix size and the number of Lanczos steps and see how this affects the results.

Exercise 9.8.2. Download the files `rehamlan.m` and `rehamlango.m` and their auxiliary codes from the website. Figure out what they do and experiment with them (in Hamiltonian mode). For example, try adjusting `nmin`, `nmax`, and `nwanted`, and see how this affects the performance of the codes.

Exercise 9.8.3. Suppose $H \in \mathbb{R}^{2n\times 2n}$ is both Hamiltonian and skew symmetric.

(a) Show that H has the form

$$H = \begin{bmatrix} A & K \\ -K & A \end{bmatrix},$$

where $A^T = -A$ and $K^T = K$.

(b) Suppose the skew-symmetric Lanczos process (Exercise 9.4.11) is applied to H. Identify a pair of orthogonal isotropic Krylov subspaces that are generated by that algorithm.

(c) Compare and contrast the skew-symmetric Lanczos process with the Hamiltonian Lanczos process derived in this section.

9.9 The Symplectic Lanczos Process

The symplectic Lanczos process was discussed in [17], [29], and [30]. The version presented here is from [222].

If $S \in \mathbb{R}^{2n\times 2n}$ is symplectic, then $S + S^{-1}$ is skew Hamiltonian (Proposition 1.4.4). Applying the skew-Hamiltonian Lanczos process (9.7.12) and (9.7.13) to $S + S^{-1}$, we have

$$u_{j+1}b_jd_j = (S + S^{-1})u_j - u_ja_jd_j - u_{j-1}b_{j-1}d_j, \tag{9.9.1}$$

$$v_{j+1}d_{j+1}b_j = (S + S^{-1})v_j - v_jd_ja_j - v_{j-1}d_{j-1}b_{j-1}, \tag{9.9.2}$$

where the recurrences are started with vectors satisfying $u_1^TJv_1 = 1$. Define new vectors $\tilde{v}_j = S^{-1}u_jd_j$, $j = 1, 2, 3, \ldots$. Since $d_j = \pm 1$, we also have

$$\tilde{v}_jd_j = S^{-1}u_j. \tag{9.9.3}$$

Multiply (9.9.1) by S^{-1} and by d_j to obtain

$$\widetilde{v}_{j+1}d_{j+1}d_j = (S + S^{-1})\widetilde{v}_j - \widetilde{v}_j d_j a_j - \widetilde{v}_{j-1}d_{j-1}b_{j-1}, \tag{9.9.4}$$

which is identical to (9.9.2). Now suppose we start (9.9.1) and (9.9.2) with vectors that satisfy both $u_1^T J v_1 = 1$ and $S^{-1}u_1 = v_1 d_1$. Then (9.9.2) and (9.9.4) have the same starting vector, and they are identical, so $\widetilde{v}_j = v_j$ for every j. This means that we do not have to run both of the recurrences (9.9.1) and (9.9.2). We can just run (9.9.1) along with the supplementary condition (9.9.3). Noting also that the term $S^{-1}u_j$ can be replaced by $v_j d_j$, we obtain

$$u_{j+1}b_j = Su_jd_j + v_j - u_ja_j - u_{j-1}b_{j-1}, \tag{9.9.5}$$
$$v_{j+1}d_{j+1} = S^{-1}u_{j+1}. \tag{9.9.6}$$

We call this pair of recurrences the *symplectic Lanczos process*. The occurrence of S^{-1} is not a problem, as $S^{-1} = -JS^TJ$.[13]

Now let us consider how to compute the coefficients. Multiplying (9.9.5) on the left by $v_j^T J$, we find that

$$a_j = -d_j v_j^T JSu_j.$$

The other coefficients are produced in the normalization process. In practice (9.9.5) and (9.9.6) take the form

$$\widehat{u}_{j+1} = Su_jd_j + v_j - u_ja_j - u_{j-1}b_{j-1}, \tag{9.9.7}$$
$$\widehat{v}_{j+1} = S^{-1}\widehat{u}_{j+1}. \tag{9.9.8}$$

Then $\widehat{u}_{j+1} = u_{j+1}b_j$ and $\widehat{v}_{j+1} = S^{-1}u_{j+1}b_j = v_{j+1}d_{j+1}b_j$. This is exactly the same situation as we had in the Hamiltonian case, and we can determine b_j and d_{j+1} by exactly the same method: Let $\pi = \widehat{u}_{j+1}^T J\widehat{v}_{j+1}$. Then we must have $\pi = d_{j+1}b_j^2$. If $\pi = 0$, we have a breakdown. If $\pi > 0$ we must have $d_{j+1} = 1$ and $b_j = \sqrt{\pi}$, and if $\pi < 0$ we must have $d_{j+1} = -1$ and $b_j = \sqrt{-\pi}$. We then have $u_{j+1} = \widehat{u}_{j+1}/b_j$ and $v_{j+1} = \widehat{v}_{j+1}/(d_{j+1}b_j)$.

The method for picking starting vectors is also the same as for the Hamiltonian case: We pick a more or less arbitrary starting vector $\widehat{u}_1$ and use the normalization process described in the previous paragraph. Thus we define $\widehat{v}_1 = S^{-1}\widehat{u}_1$, $\pi = \widehat{u}_1^T J\widehat{v}_1$, and so on. We end up with starting vectors u_1 and v_1 that satisfy both $v_1d_1 = S^{-1}u_1$ and $u_1^T Jv_1 = 1$.

Putting these ideas together, we have the following algorithm.

Symplectic Lanczos Process.

[13]An equivalent set of recurrences is obtained by retaining (9.9.2), (9.9.3) and reorganizing them to get $v_{j+1}d_{j+1}b_j = Sv_j + u_jd_j - v_jd_ja_j - v_{j-1}d_{j-1}b_{j-1}$ and $u_{j+1}d_{j+1} = S^{-1}v_{j+1}$. These are the recurrences that were derived in [222].

Start with initial vector u_1.

```
$b_0 \leftarrow 0,\ d_0 \leftarrow 0,\ v_0 \leftarrow 0$
for $j = 0, 1, 2, \ldots$
    if $j > 0$ then
        $y \leftarrow Su_j$
        $a_j \leftarrow -d_j(v_j^T Jy)$
        $u_{j+1} \leftarrow yd_j + v_j - u_j a_j - u_{j-1}b_{j-1}$
    $v_{j+1} \leftarrow S^{-1}u_{j+1}$
    $\pi \leftarrow u_{j+1}^T J v_{j+1}$
    if $\pi = 0$ then
        stop (breakdown)
    if $\pi > 0$ then
        $s \leftarrow \sqrt{\pi},\ d_{j+1} \leftarrow 1$
    if $\pi < 0$ then
        $s \leftarrow \sqrt{-\pi},\ d_{j+1} \leftarrow -1$
    $u_{j+1} \leftarrow u_{j+1}/s$
    $v_{j+1} \leftarrow v_{j+1}/(d_{j+1}s)$
    if $j > 0$ then
        $b_j \leftarrow s$
                                                        (9.9.9)
```

Just as for the Hamiltonian algorithm, the "symplecticness" of the vectors produced will decline over time unless we explicitly enforce it. To do this, insert exactly the same lines (9.8.10) as in the Hamiltonian algorithm.

Implicit restarts

Rearranging (9.9.5) and (9.9.6), we obtain

$$Su_j = u_{j-1}b_{j-1}d_j + u_j a_j d_j + u_{j+1}b_j d_j - v_j d_j,$$

$$Sv_j = u_j d_j,$$

which when packed into matrix equations gives

$$SU_k = U_k(T_k D_k) + u_{k+1}b_{k+1}d_{k+1}e_k^T - V_k D_k, \qquad SV_k = U_k D_k \tag{9.9.10}$$

after $k - 1/2$ steps. Here D_k is the signature matrix $D_k = \operatorname{diag}\{d_1, \ldots, d_k\}$, and T_k is the same symmetric, tridiagonal matrix as always. Equations (9.9.10) can also be written as the single equation

$$S\begin{bmatrix} U_k & V_k \end{bmatrix} = \begin{bmatrix} U_k & V_k \end{bmatrix}\begin{bmatrix} T_k D_k & D_k \\ -D_k & 0 \end{bmatrix} + u_{k+1}b_{k+1}d_{k+1}e_k^T.$$

The symplectic matrix $\begin{bmatrix} T_k D_k & D_k \\ -D_k & 0 \end{bmatrix}$, which is familiar from Section 8.9, tells us how restarts should be done. Either of the restart methods that we have presented can be used. The filtering should be done by the product HR algorithm (= HZ algorithm), as explained in Section 8.9.

Extracting eigenvectors

Once the process has converged to an invariant subspace of dimension m, we have

$$S\begin{bmatrix} U_m & V_m \end{bmatrix} = \begin{bmatrix} U_m & V_m \end{bmatrix}\begin{bmatrix} T_m D_m & D_m \\ -D_m & 0 \end{bmatrix}, \tag{9.9.11}$$

where D_m is a signature matrix, and T_m is block diagonal. The eigenvectors of $\begin{bmatrix} T_m D_m & D_m \\ -D_m & 0 \end{bmatrix}$ are easy to compute, and each such eigenvector $\begin{bmatrix} x \\ y \end{bmatrix}$ yields an eigenvector $U_m x + V_m y$ of S.

Left eigenvectors are also easily obtained. Recalling that

$$\begin{bmatrix} T_m D_m & D_m \\ -D_m & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & -D_m \\ D_m & D_m T_m \end{bmatrix},$$

multiply (9.9.11) on the right by this matrix and on the left by S^{-1} to obtain

$$S^{-1}\begin{bmatrix} U_m & V_m \end{bmatrix} = \begin{bmatrix} U_m & V_m \end{bmatrix}\begin{bmatrix} 0 & -D_m \\ D_m & D_m T_m \end{bmatrix}. \tag{9.9.12}$$

Let $W_m = JV_m$ and $X_m = -JU_m$, as before. Then, multiplying (9.9.12) by J on the left and using the identity $JS^{-1} = S^T J$, we obtain

$$S^T\begin{bmatrix} -X_m & W_m \end{bmatrix} = \begin{bmatrix} -X_m & W_m \end{bmatrix}\begin{bmatrix} 0 & -D_m \\ D_m & D_m T_m \end{bmatrix}. \tag{9.9.13}$$

From this we can get the right eigenvectors of S^T, and hence the left eigenvectors of S.

The caveat about *a posteriori* testing that we gave for the Hamiltonian case is in equal force here. We should always compute residuals as a test of backward stability. Moreover, the presence of both right and left eigenvectors allows us to check the condition numbers of all the computed eigenvalues.

Exercises

Exercise 9.9.1. The file `symplan.m` encodes the symplectic Lanczos process. Download this file and the driver program `symplango.m` from the website and play with them. Try adjusting the matrix size and the number of Lanczos steps and see how this affects the results. Note: The symplectic matrix that is used in this example was obtained by taking a Cayley transform of Hamiltonian matrix:

$$S = (H - \tau I)^{-1}(H + \tau I).$$

You can vary the value of τ and see how this affects the results. The eigenvalues and approximations that are shown in the plots are those of the Hamiltonian matrix H, not of the symplectic matrix S.

Exercise 9.9.2. Download the files `rehamlan.m` and `rehamlango.m` and their auxiliary codes from the website. Figure out how to make them run in symplectic mode, then experiment with them. For example, try adjusting `nmin`, `nmax`, and `nwanted`, and see how this affects the performance of the codes. Note: Read the note to the previous exercise.

9.10 Product Krylov Processes

In this section we return to the theme of Chapter 8. Again we draw from Kressner [134]. Suppose we want to compute some eigenvalues of a large matrix presented as a product of k matrices: $A = A_k A_{k-1} \cdots A_1$. The developments in this section hold for arbitrary k, but we will consider the case $k = 3$ for illustration. This is done solely to keep the notation under control. Thus let $A = A_3 A_2 A_1$. As in Chapter 8 we find it convenient to work with the cyclic matrix

$$C = \begin{bmatrix} & & A_3 \\ A_1 & & \\ & A_2 & \end{bmatrix}. \tag{9.10.1}$$

Consider what happens when we apply the Arnoldi process to C with a special starting vector of the form

$$\begin{bmatrix} u_1 \\ 0 \\ 0 \end{bmatrix}.$$

Multiplying this vector by C, we obtain

$$\begin{bmatrix} 0 \\ A_1 u_1 \\ 0 \end{bmatrix},$$

which is already orthogonal to the first vector. Thus only a normalization needs to be done:

$$\begin{bmatrix} 0 \\ v_1 \\ 0 \end{bmatrix} s_{11} = \begin{bmatrix} & & A_3 \\ A_1 & & \\ & A_2 & \end{bmatrix} \begin{bmatrix} u_1 \\ 0 \\ 0 \end{bmatrix},$$

where $s_{11} = \| A_1 u_1 \|_2$. On the second step we again get a vector that is already orthogonal to the first two vectors, so only a normalization is necessary:

$$\begin{bmatrix} 0 \\ 0 \\ w_1 \end{bmatrix} t_{11} = \begin{bmatrix} & & A_3 \\ A_1 & & \\ & A_2 & \end{bmatrix} \begin{bmatrix} 0 \\ v_1 \\ 0 \end{bmatrix},$$

where $t_{11} = \| A_2 v_1 \|_2$. Finally, on the third step, we get a vector that has to be orthogonalized against one of the previous vectors:

$$\begin{bmatrix} u_2 \\ 0 \\ 0 \end{bmatrix} h_{21} = \begin{bmatrix} & & A_3 \\ A_1 & & \\ & A_2 & \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ w_1 \end{bmatrix} - \begin{bmatrix} u_1 \\ 0 \\ 0 \end{bmatrix} h_{11}.$$

This step can be expressed more concisely as

$$u_2 h_{21} = A_3 w_1 - u_1 h_{11}.$$

The computations would be done just as in any Arnoldi step: $\widehat{u}_2 \leftarrow A_3 w_1$, $h_{11} = u_1^* \widehat{u}_2$, $\widehat{u}_2 \leftarrow \widehat{u}_2 - u_1 h_{11}$, $h_{21} = \| \widehat{u}_2 \|_2$, and $u_2 = \widehat{u}_2 / h_{21}$. (In practice the orthogonalization would be done twice.)

The pattern is clear. In each cycle of three Arnoldi steps a new "u", "v", and "w" vector will be produced, each of which has to be orthogonalized only against the previous "u", "v", or "w" vectors, respectively. For example, the "w" vector that is produced on the jth cycle is given by

$$\begin{bmatrix} 0 \\ 0 \\ w_j \end{bmatrix} t_{jj} = \begin{bmatrix} & & A_3 \\ A_1 & & \\ & A_2 & \end{bmatrix} \begin{bmatrix} 0 \\ v_j \\ 0 \end{bmatrix} - \sum_{i=1}^{j-1} \begin{bmatrix} 0 \\ 0 \\ w_i \end{bmatrix} t_{ij}$$

or, more concisely,

$$w_j t_{jj} = A_2 v_j - \sum_{i=1}^{j-1} w_i t_{ij}.$$

Similar formulas hold for v_j and w_{j+1}. Placed together, in order of execution, they are

$$v_j s_{jj} = A_1 u_j - \sum_{i=1}^{j-1} v_i s_{ij},$$

$$w_j t_{jj} = A_2 v_j - \sum_{i=1}^{j-1} w_i t_{ij},$$

$$u_{j+1} h_{j+1,j} = A_3 w_j - \sum_{i=1}^{j} u_i h_{ij}.$$

Rewrite these equations as

$$A_1 u_j = \sum_{i=1}^{j} v_i s_{ij}, \qquad A_2 v_j = \sum_{i=1}^{j} w_i t_{ij}, \qquad A_3 w_j = \sum_{i=1}^{j+1} u_i h_{ij},$$

then assemble them into matrix equations. Suppose we have gone through k complete cycles. Then we have

$$\begin{aligned} A_1 U_k &= V_k S_k, \\ A_2 V_k &= W_k T_k, \\ A_3 W_k &= U_k H_k + u_{k+1} h_{k+1,k} e_k^T, \end{aligned} \tag{9.10.2}$$

where $U_k = \begin{bmatrix} u_1 & \cdots & u_k \end{bmatrix}$, for example, and the notation is mostly self explanatory. Notice that S_k and T_k are upper-triangular matrices, but H_k is upper Hessenberg. Equations (9.10.2) can be combined into the single equation

$$\begin{bmatrix} & & A_3 \\ A_1 & & \\ & A_2 & \end{bmatrix} \begin{bmatrix} U_k & & \\ & V_k & \\ & & W_k \end{bmatrix} = \begin{bmatrix} U_k & & \\ & V_k & \\ & & W_k \end{bmatrix} \begin{bmatrix} & & H_k \\ S_k & & \\ & T_k & \end{bmatrix} + \begin{bmatrix} u_{k+1} \\ \\ \\ \end{bmatrix} h_{k+1,k} e_{3k}^T. \tag{9.10.3}$$

If we wanted to rewrite the matrix of vectors

$$\begin{bmatrix} U_k & & \\ & V_k & \\ & & W_k \end{bmatrix}$$

so that order of the vectors in the matrix was the same as the order in which they were computed, we would have to do a perfect shuffle of the columns. Then, to make (9.10.3) hold for the shuffled matrix, we would have to shuffle the rows and columns of the matrix of coefficients

$$\begin{bmatrix} & & H_k \\ S_k & & \\ & T_k & \end{bmatrix}.$$

This results in an upper Hessenberg matrix whose structure is illustrated by (8.3.4).

Combining the equations in (9.10.2) we get

$$(A_3A_2A_1)U_k = U_k(H_kT_kS_k) + u_{k+1}(h_{k+1,k}t_{kk}s_{kk})e_k^T. \tag{9.10.4}$$

Since $H_kT_kS_k$ is upper Hessenberg, this shows that our Arnoldi process on C is automatically effecting an Arnoldi process on the product $A = A_3A_2A_1$. If we rewrite the equation $A_3W_k = U_kH_k + u_{k+1}h_{k+1,k}$ in the style $A_3W_k = U_{k+1}H_{k+1,k}$, we can write (9.10.2) as

$$(A_3A_2A_1)U_k = U_{k+1}(H_{k+1,k}T_kS_k).$$

Moreover, if we recombine (9.10.2) in different ways, we obtain

$$(A_1A_3A_2)V_k = V_{k+1}(S_{k+1}H_{k+1,k}T_k)$$

and

$$(A_2A_1A_3)W_k = W_{k+1}(T_{k+1}S_{k+1}H_{k+1,k}),$$

showing that we are also doing Arnoldi processes on the cycled products $A_1A_3A_2$ and $A_2A_1A_3$.

So far we have restricted our attention to the Arnoldi process, but other product Krylov processes can be built in the same fashion. For all such processes, an equation of the form (9.10.3) will hold, but the vectors produced need not be orthonormal. As we shall see below, the Hamiltonian Lanczos process (§ 9.8) can be viewed as a product Krylov process.

Implicit restarts can be done by either of the two restart methods that we have presented. Whichever method is used, the filtering can be done by an appropriate product GR algorithm (Section 8.3) to preserve the product structure.

Symmetric Lanczos process for singular values

Consider the problem of finding the largest few singular values and associated singular vectors of B. This is equivalent to the eigenvalue problem for B^*B and BB^*, so consider the cyclic matrix

$$C = \begin{bmatrix} 0 & B^* \\ B & 0 \end{bmatrix}.$$

If we run the symmetric Lanczos (= Arnoldi) process on this matrix, we obtain, in analogy with (9.10.3),

$$\begin{bmatrix} & B^* \\ B & \end{bmatrix}\begin{bmatrix} V_k & \\ & U_k \end{bmatrix} = \begin{bmatrix} V_k & \\ & U_k \end{bmatrix}\begin{bmatrix} & H_k \\ T_k & \end{bmatrix} + \begin{bmatrix} v_{k+1} \\ \end{bmatrix} h_{k+1,k} e_{2k}^T. \quad (9.10.5)$$

By symmetry we have $H_k = T_k^*$, from which it follows immediately that both H_k and T_k are bidiagonal. (Moreover, they can be taken to be real. If we shuffle the rows and columns, we obtain a tridiagonal matrix with zeros on the main diagonal, like 8.4.3.) Letting

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ & \alpha_2 & \ddots & \\ & & \ddots & \beta_{k-1} \\ & & & \alpha_k \end{bmatrix},$$

equation (9.10.5) gives the short recurrences

$$u_j \alpha_j = B v_j - u_{j-1}\beta_{j-1}$$

and

$$v_{j+1}\beta_j = B^* u_j - v_j \alpha_j,$$

which appeared first in [97]. These automatically effect symmetric Lanczos processes on B^*B and BB^*.

They can be restarted by either of the methods of § 9.2 [38, 132, 12]. Consider the second method, for example. After $2k$ steps we have

$$\begin{bmatrix} & B^* \\ B & \end{bmatrix}\begin{bmatrix} V_k & \\ & U_k \end{bmatrix} = \begin{bmatrix} V_k & \\ & U_k \end{bmatrix}\begin{bmatrix} & T_k^T \\ T_k & \end{bmatrix} + \begin{bmatrix} v_{k+1} \\ 0 \end{bmatrix} \beta_k e_{2k}^T. \quad (9.10.6)$$

If we want to restart with $2m$ vectors, we compute the SVD of T_k: $T_k = P_k \Sigma_k Q_k^T$, where the singular values in Σ_k are ordered so that the m that we want to keep (the m biggest ones in this case) are at the top. Then

$$\begin{bmatrix} & T_k^T \\ T_k & \end{bmatrix} = \begin{bmatrix} Q_k & \\ & P_k \end{bmatrix}\begin{bmatrix} & \Sigma_k^T \\ \Sigma_k & \end{bmatrix}\begin{bmatrix} Q_k^T & \\ & P_k^T \end{bmatrix}.$$

Substituting this into (9.10.6) and multiplying on the right by $\operatorname{diag}\{Q_k, P_k\}$, we obtain

$$\begin{bmatrix} & B^* \\ B & \end{bmatrix}\begin{bmatrix} \widetilde{V}_k & \\ & \widetilde{U}_k \end{bmatrix} = \begin{bmatrix} \widetilde{V}_k & \\ & \widetilde{U}_k \end{bmatrix}\begin{bmatrix} & \Sigma_k^T \\ \Sigma_k & \end{bmatrix} + \begin{bmatrix} v_{k+1} \\ 0 \end{bmatrix} \beta_k \begin{bmatrix} 0 & \widetilde{z}^T \end{bmatrix}, \quad (9.10.7)$$

where $\widetilde{V}_k = V_k Q_k$, $\widetilde{U}_k = U_k P_k$, and $\widetilde{z}^T = e_k^T P_k$. If we do a perfect shuffle of the vectors in (9.10.7), it becomes an instance of (9.3.13). Now let $\widetilde{V}_m$ and $\widetilde{U}_m$ denote the first m columns of $\widetilde{V}_k$ and $\widetilde{U}_k$, respectively. Let z denote the first m entries of $\widetilde{z}$, and let Σ_m denote the upper left-hand $m \times m$ submatrix of Σ_k. Σ_m contains the singular values of T_k that we wish to retain. Keeping $2m$ columns of (9.10.7), we have

$$\begin{bmatrix} & B^* \\ B & \end{bmatrix}\begin{bmatrix} \widetilde{V}_m & \\ & \widetilde{U}_m \end{bmatrix} = \begin{bmatrix} \widetilde{V}_m & \\ & \widetilde{U}_m \end{bmatrix}\begin{bmatrix} & \Sigma_m^T \\ \Sigma_m & \end{bmatrix} + \begin{bmatrix} v_{k+1} \\ 0 \end{bmatrix} \beta_k \begin{bmatrix} 0 & z^T \end{bmatrix}, \quad (9.10.8)$$

which is (upon shuffling) an instance of (9.3.14). Now let X_m and Y_m be orthogonal matrices such that $z^T X_m^T = \gamma e_m^T$ (for some γ) and the matrix $\widehat{B}_m = X_m \Sigma_m Y_m^T$ is bidiagonal. This is achieved by starting with an orthogonal matrix X_{m1} such that $z^T X_{m1}^T = \alpha e_m^T$, then transforming $X_{m1}\Sigma_m$ to bidiagonal form from bottom to top. (Here is one viewpoint. Do a perfect shuffle of the rows and columns of

$$\begin{bmatrix} & (X_{m1}\Sigma_m)^T \\ X_{m1}\Sigma_m & \end{bmatrix}.$$

Then reduce the resulting matrix to upper Hessenberg form from bottom to top, as described in Section 3.4, using orthogonal transformations. The result will be tridiagonal by symmetry with zeros on the main diagonal. When we deshuffle this tridiagonal matrix, we get

$$\begin{bmatrix} & \widehat{B}_m^T \\ \widehat{B}_m & \end{bmatrix},$$

where $\widehat{B}_m$ is bidiagonal.)

Since $\widehat{B}_m = X_m \Sigma_m Y_m^T$, we have

$$\begin{bmatrix} & \Sigma_m^T \\ \Sigma_m & \end{bmatrix} = \begin{bmatrix} Y_m^T & \\ & X_m^T \end{bmatrix} \begin{bmatrix} & \widehat{B}_m^T \\ \widehat{B}_m & \end{bmatrix} \begin{bmatrix} Y_m & \\ & X_m \end{bmatrix}.$$

Substituting this into (9.10.8) and multiplying on the right by $\begin{bmatrix} Y_m^T & \\ & X_m^T \end{bmatrix}$, we obtain

$$\begin{bmatrix} & B^* \\ B & \end{bmatrix} \begin{bmatrix} \widehat{V}_m & \\ & \widehat{U}_m \end{bmatrix} = \begin{bmatrix} \widehat{V}_m & \\ & \widehat{U}_m \end{bmatrix} \begin{bmatrix} & \widehat{B}_m^T \\ \widehat{B}_m & \end{bmatrix} + \begin{bmatrix} \widehat{v}_{m+1} \\ 0 \end{bmatrix} \widehat{\beta}_m \begin{bmatrix} 0 & e_m^T \end{bmatrix}, \tag{9.10.9}$$

where $\widehat{V}_m = \widetilde{V}_m Y_m^T$, $\widehat{U}_m = \widetilde{U}_m X_m^T$, $\widehat{v}_{m+1} = v_{k+1}$, and $\widehat{\beta}_m = \beta_k \gamma$. Equation (9.10.9) is (upon shuffling) an instance of (9.3.15), and it is just what we need to restart the Lanczos process.

In the description here, everything is done twice. In an actual implementation, just the following needs to be done: First the SVD $T_k = P_k \Sigma_k Q_k^T$ is computed. The vector z^T is obtained as the first m entries of the last row of P_k, and Σ_m is obtained as a submatrix of Σ_k. Then the transformation $\widehat{B}_m = X_m \Sigma_m Y_m^T$ with $z^T X_m^T = \gamma e_m^T$ is done. $\widetilde{U}_m$ and $\widetilde{V}_m$ are obtained by computing the first m columns of the transformations $\widetilde{U}_k = U_k P_k$ and $\widetilde{V}_k = V_k Q_k$, respectively, then $\widehat{U}_m = \widetilde{U}_m X_m^T$ and $\widehat{V}_m = \widetilde{V}_m Y_m^T$ are computed.

The method we have just described is just the thick restart method of Wu and Simon [233], taking the cyclic structure of the matrix into account.

The Hamiltonian Lanczos process revisited

Since the Hamiltonian Lanczos process for H was derived from the skew-Hamiltonian Lanczos process for the product HH, we should not be surprised that this algorithm can also be derived as a product Krylov process. Such a process would satisfy an equation analogous to (9.10.3), which in this case would have the form

$$\begin{bmatrix} & H \\ H & \end{bmatrix} \begin{bmatrix} U_k & \\ & V_k \end{bmatrix} = \begin{bmatrix} U_k & \\ & V_k \end{bmatrix} \begin{bmatrix} & T_k \\ D_k & \end{bmatrix} + \begin{bmatrix} u_{k+1} \\ \end{bmatrix} \beta_k e_{2k}^T. \tag{9.10.10}$$

We use the symbols D_k and T_k here, because they will turn out to have the same form as the D_k and T_k from Sections 9.7, 9.8 and 9.9. However, all we know at this point is that D_k is upper triangular and T_k is upper Hessenberg. Equation (9.10.10) can be written more compactly as

$$HU_k = V_k D_k, \qquad HV_k = U_k T_k + u_{k+1}\beta_k e_k^T \tag{9.10.11}$$

or

$$H\begin{bmatrix} U_k & V_k \end{bmatrix} = \begin{bmatrix} U_k & V_k \end{bmatrix}\begin{bmatrix} & T_k \\ D_k & \end{bmatrix} + u_{k+1}t_{k+1,k}e_{2k}^T. \tag{9.10.12}$$

We can preserve Hamiltonian structure by building vectors $u_1, \ldots, u_k$ and $v_1, \ldots v_k$ that are columns of a symplectic matrix. If we rewrite the second equation of (9.10.11) in the style $HV_k = U_{k+1}T_{k+1,k}$, we have

$$H^2 U_k = U_{k+1}T_{k+1,k}D_k \quad \text{and} \quad H^2 V_k = V_{k+1}D_{k+1}T_{k+1,k},$$

so $\mathcal{R}(U_k) = \mathcal{K}_k(H^2, u_1)$ and $\mathcal{R}(V_k) = \mathcal{K}_k(H^2, v_1)$. Thus we have isotropy ($U_k^T J U_k = 0$ and $V_k^T J V_k = 0$) automatically by Proposition 9.7.2. The one other property we need is

$$U_k^T J V_k = I_k, \tag{9.10.13}$$

Equating jth columns in (9.10.11), we have

$$Hu_j = \sum_{i=1}^{j} v_i d_{ij}, \qquad Hv_j = \sum_{i=1}^{j+1} u_i t_{ij}, \tag{9.10.14}$$

which give the recurrences

$$u_{j+1}t_{j+1,j} = Hv_j - \sum_{i=1}^{j} u_i t_{ij}, \tag{9.10.15}$$

$$v_{j+1}d_{j+1,j+1} = Hu_{j+1} - \sum_{i=1}^{j} v_i d_{i,j+1}. \tag{9.10.16}$$

It is not hard to show that the coefficients t_{ij} and $d_{i,j+1}$ can be chosen in such a way that (9.10.13) holds. Moreover, it can be done in such a way that $d_{jj} = \pm 1$ for all j (assuming no breakdowns).

Once we have achieved this, the symplectic structure of $\begin{bmatrix} U_k & V_k \end{bmatrix}$ in (9.10.12) can be used to prove that the small matrix

$$\begin{bmatrix} & T_k \\ D_k & \end{bmatrix}$$

must be Hamiltonian. Therefore D_k and T_k are symmetric, D_k is diagonal (and even a signature matrix), and T_k is tridiagonal. Almost all of the coefficients in (9.10.15) and (9.10.16) are zero, and the process that we have derived is exactly the Hamiltonian Lanczos process of § 9.8. The details are left as an exercise.

9.11 Block Krylov Processes

Krylov processes can have difficulties with matrices that have multiple eigenvalues. For example, consider a Hermitian matrix $A \in \mathbb{C}^{n\times n}$ whose largest eigenvalue λ_1 has multiplicity greater than one. If we run the symmetric Lanczos process on A, our approximate eigenvalues are taken from a symmetric properly tridiagonal matrix T_k, which cannot have multiple eigenvalues (Exercise 9.11.2). Therefore we must find and lock out one eigenvector associated with λ_1 before we can even begin to search for the rest of the eigenspace associated with λ_1. In fact, in exact arithmetic we will never find the rest of the eigenspace (Exercise 9.11.3), so we are dependent on roundoff errors to help us find it. Clearly there is a danger of overlooking some wanted eigenvectors, particularly if the multiplicity is high.

A remedy for this problem is to use a block Krylov process. We will illustrate by discussing the block Arnoldi process. Throughout this book the lower-case symbol u has always stood for a vector. If we wanted to refer to a matrix with several columns, we always used an upper-case letter. In this section we will violate that convention in the interest of avoiding a notational conflict. Our block Arnoldi process begins with a u_1 that is not a vector but a matrix with b columns, where b is the *block size*. We can take the columns to be orthonormal; for example, we can get u_1 by picking an $n \times b$ matrix at random and orthonomalizing its columns by a QR decomposition.

After $j-1$ steps we will have $u_1, \ldots, u_j$, each consisting of b orthonormal columns such that the jb columns of $u_1, \ldots, u_j$ taken together form an orthonormal set. We now describe the jth step. In analogy with (9.4.1) we define coefficients h_{ij} by

$$h_{ij} = u_i^* A u_j, \qquad i = 1, \ldots, j. \tag{9.11.1}$$

Each h_{ij} is now a $b \times b$ matrix. In analogy with (9.4.2) we define

$$\widehat{u}_{j+1} = Au_j - \sum_{i=1}^{j} u_i h_{ij}, \tag{9.11.2}$$

where $\widehat{u}_{j+1}$ is a matrix with b columns. We cannot expect that the columns of $\widehat{u}_{j+1}$ will be orthogonal, but it is easy to show that they are orthogonal to all of the columns of $u_1, \ldots, u_j$. In practice one would do this orthonormalization step twice, as illustrated in Algorithm 9.4.7. The process of normalizing $\widehat{u}_{k+1}$ is more complicated than what is shown in (9.4.3) and (9.4.4) but hardly difficult. Instead of taking the norm and dividing, we do a condensed QR decomposition:

$$\widehat{u}_{j+1} = u_{j+1} h_{j+1,j}. \tag{9.11.3}$$

Here u_{j+1} is $n \times b$ with orthonormal columns, and $h_{j+1,j}$ is $b \times b$ and upper triangular. This completes the description of a block Arnoldi step.

Algorithm 9.4.7 serves as a description of the block Arnoldi process, except that the normalization step has to be changed to (9.11.3). Equations like

$$AU_k = U_k H_k + u_{k+1} h_{k+1,k} e_k^T$$

continue to hold if we reinterpret e_k to be the last b columns of the $kb \times kb$ identity matrix. H_k is now a block upper Hessenberg matrix. Since the subdiagonal blocks $h_{j+1,j}$ are upper triangular, it is in fact b-Hessenberg (Everything below the bth subdiagonal is zero). Since H_k is not Hessenberg in the scalar sense, it can have multiple eigenvalues, and eigenvalues of A with geometric multiplicity as large as b can be found and extracted together. Implicit restarts can be done in much the same way as they are for the scalar case.

We have described a block Arnoldi process. Other block Krylov processes can be developed in the same way.

Exercises

Exercise 9.11.1. The matrix generated by the MATLAB command

```
A = delsq(numgrid('S',40)),
```

a discrete negative Laplacian on a square, has numerous repeated eigenvalues. Download the block symmetric Lanczos code `blocklan.m` from the website. Use it to compute a few of the smallest eigenvalues of A by the block Lanczos process applied to A^{-1}. Try block sizes 1, 2, and 3. Notice that with block sizes 2 and greater, all of the small repeated eigenvalues are found, while with block size 1, some of the repeated eigenvalues are overlooked.

Exercise 9.11.2. Let $H \in \mathbb{C}^{k\times k}$ be a properly upper Hessenberg matrix.

(a) Show that for any $\mu \in \mathbb{C}$, $H - \mu I$ has rank at least $k - 1$.

(b) Show that every eigenvalue of H has geometric multiplicity 1. Deduce that H has one Jordan block associated with each eigenvalue.

(c) Now suppose H is also Hermitian, hence tridiagonal. Show that each eigenvalue has algebraic multiplicity 1. Therefore H has k distinct eigenvalues.

Exercise 9.11.3. Let $A \in \mathbb{C}^{n\times n}$ be a semisimple matrix with j distinct eigenvalues $\lambda_1, \ldots, \lambda_j$, where $j < n$. Let $\mathcal{S}_i$ denote the eigenspace associated with λ_i, $i = 1, \ldots, j$.

(a) Show that each $x \in \mathbb{C}^n$ can be expressed uniquely as

$$x = q_1 + q_2 + \cdots + q_j,$$

where $q_i \in \mathcal{S}_i$, $i = 1, \ldots, j$.

(b) Show that every Krylov space $\mathcal{K}_i(A, x)$ satisfies

$$\mathcal{K}_i(A, x) \subseteq \operatorname{span}\{q_1, \ldots, q_j\}.$$

Therefore any Krylov process starting with x sees only multiples of q_i from the eigenspace $\mathcal{S}_i$. If λ_i is a multiple eigenvalue, the complete eigenspace will never be found (in exact arithmetic).

(c) Show that this problem is not alleviated by implicit restarts.

Exercise 9.11.4. Think about how the results of Exercise 9.11.3 might be extended to defective matrices.

Exercise 9.11.5. Consider the block Arnoldi process applied to a matrix $A \in \mathbb{R}^{2n\times 2n}$ that is both skew symmetric ($A^T = -A$) and Hamiltonian ($(JA)^T = JA$).

(a) What simplifications occur in (9.11.2) due to the fact that A is skew symmetric?

(b) Show that $JA = AJ$, where $J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}$, as always.

(c) Now take block size $b = 2$ and consider the block Arnoldi process with starting vectors $u_1 = \begin{bmatrix} q_1 & -Jq_1 \end{bmatrix}$, where $q_1 \in \mathbb{R}^{2n}$ is a starting vector chosen so that $\| q_1 \| = 1$. Show that the columns of u_1 are orthonormal.

(d) Consider the first step $\widehat{u}_2 = Au_1 - u_1 h_{11}$. Show that $h_{11} = \begin{bmatrix} 0 & -\alpha_1 \\ \alpha_1 & 0 \end{bmatrix}$, where $\alpha_1 = q_1^T JAq_1$. Letting $\widehat{u}_2 = \begin{bmatrix} \widehat{q}_2 & v \end{bmatrix}$, determine formulas for $\widehat{q}_2$ and v, and show that $v = -J\widehat{q}_2$.

(e) In the QR decomposition $\widehat{u}_2 = u_2 h_{21}$, show that

$$u_2 = \begin{bmatrix} q_2 & -Jq_2 \end{bmatrix} \quad \text{and} \quad h_{21} = \begin{bmatrix} \beta_1 & 0 \\ 0 & \beta_1 \end{bmatrix},$$

where $\beta_1 = \| \widehat{q}_2 \|$ and $q_2 = \widehat{q}_2 / \beta_1$.

(f) Using the equations

$$u_{j+1} h_{j+1,j} = \widehat{u}_{j+1} = Au_j - u_j h_{jj} - u_{j-1} h_{j-1,j},$$

prove by induction on j that $\widehat{u}_j = \begin{bmatrix} \widehat{q}_j & -J\widehat{q}_j \end{bmatrix}$, $u_j = \begin{bmatrix} q_j & -Jq_j \end{bmatrix}$,

$$h_{jj} = \begin{bmatrix} 0 & -\alpha_j \\ \alpha_j & 0 \end{bmatrix}, \quad \text{and} \quad h_{j+1,j} = -h_{j,j+1} = \begin{bmatrix} \beta_j & 0 \\ 0 & \beta_j \end{bmatrix}$$

for all j, where $\alpha_j = q_j^T JAq_j$, $\beta_j = \| \widehat{q}_{j+1} \|$, and the vectors q_j are generated by the recurrence

$$q_{j+1}\beta_j = \widehat{q}_{j+1} = Aq_j + Jq_j\alpha_j + q_{j-1}\beta_{j-1}.$$

This is the *Hamiltonian skew-symmetric Lanczos process*.

(g) Let $Q_j = \begin{bmatrix} q_1 & \cdots & q_j \end{bmatrix}$ and $\mathcal{Q}_j = \mathcal{R}(Q_j)$. Show that $\mathcal{Q}_j$ is isotropic. Show that the $2n \times 2j$ matrix $U_j = \begin{bmatrix} Q_j & -JQ_j \end{bmatrix}$ is symplectic, in the sense that $U_j^T J U_j = J_{2j}$.

(h) Show that

$$AQ_j = Q_j B_j - JQ_j A_j + q_{j+1}\beta_j e_j^T,$$

where

$$B_j = \begin{bmatrix} 0 & -\beta_1 & & \\ \beta_1 & 0 & \ddots & \\ & \ddots & \ddots & -\beta_{j-1} \\ & & \beta_{j-1} & 0 \end{bmatrix} \quad \text{and} \quad A_j = \begin{bmatrix} \alpha_1 & 0 & & \\ 0 & \alpha_2 & \ddots & \\ & \ddots & \ddots & 0 \\ & & 0 & \alpha_j \end{bmatrix}.$$

(i) Show that

$$A\begin{bmatrix} Q_j & -JQ_j \end{bmatrix} = \begin{bmatrix} Q_j & -JQ_j \end{bmatrix}\begin{bmatrix} B_j & -A_j \\ A_j & B_j \end{bmatrix} + R_j,$$

where

$$R_j = q_{j+1}\beta_j\begin{bmatrix} e_j^T & 0 \end{bmatrix} - Jq_{j+1}\beta_j\begin{bmatrix} 0 & e_j^T \end{bmatrix}.$$