7

Eigenvalue Problems

Small signal stability is the ability of a system to maintain stability when subjected to small disturbances. Small signal analysis provides valuable information about the inherent dynamic characteristics of the system and assists in its design, operation, and control. Time domain simulation and eigenanalysis are the two main approaches to study system stability.

Eigenanalysis methods are widely used to perform small signal stability studies. The dynamic behavior of a system in response to small perturbations can be determined by computing the eigenvalues and eigenvectors of the system matrix. The locations of the eigenvalues can be used to investigate the system's performance. In addition, eigenvectors can be used to estimate the relative participation of the respective states in the corresponding disturbance modes.

A scalar λ is an eigenvalue of an $n \times n$ matrix A if there exists a nonzero $n \times 1$ vector v such that

$$Av = \lambda v \tag{7.1}$$

where v is the corresponding right eigenvector. If there exists a nonzero vector w such that

$$w^T A = \lambda w^T \tag{7.2}$$

then w is a left eigenvector. The set of all eigenvalues of A is called the spectrum of A. Normally the term "eigenvector" refers to the right eigenvector unless denoted otherwise. The eigenvalue problem in Equation (7.1) is called the standard eigenvalue problem. Equation (7.1) can be written as

$$(A - \lambda I)\, v = 0 \tag{7.3}$$

and thus is a homogeneous system of equations for x. This system has a nontrivial solution only if the determinant

$$det\,(A - \lambda I) = 0$$

The determinant equation is also called the characteristic equation for A and is an nth degree polynomial in λ. The eigenvalues of an $n \times n$ matrix A are the roots of the characteristic equation

$$\lambda^n + c_{n-1}\lambda^{n-1} + c_{n-1}\lambda^{n-1} + \ldots + c_0 = 0 \tag{7.4}$$

Therefore, there are n roots (possibly real or complex) of the characteristic equation. Each one of these roots is also an eigenvalue of A.

7.1 The Power Method

The power method is one of the most common methods of finding the *dominant* eigenvalue of the $n \times n$ matrix A. The dominant eigenvalue is the largest eigenvalue in absolute value. Therefore, if $\lambda_1, \lambda_2, \ldots, \lambda_n$ are eigenvalues of A, then λ_1 is the dominant eigenvalue of A if

$$|\lambda_1| > |\lambda_i| \tag{7.5}$$

for all $i = 2, \ldots, n$.

The power method is actually an approach to finding the eigenvector v_1 corresponding to the dominant eigenvalue of the matrix A. Once the eigenvector is obtained, the eigenvalue can be extracted from the *Rayleigh quotient*:

$$\lambda = \frac{\langle Av, v \rangle}{\langle v, v \rangle} \tag{7.6}$$

The approach to finding the eigenvector v_1 is an iterative approach. Therefore, from an initial guess vector v^0, a sequence of approximations v^k is constructed which hopefully converges as k goes to ∞. The iterative algorithm for the power method is straightforward:

The Power Method

1. Let $k = 0$ and choose v^0 to be a nonzero $n \times 1$ vector.
2. $w^{k+1} = Av^k$
3. $\alpha_{k+1} = \|w^{k+1}\|$
4. $v^{k+1} = \frac{w^{k+1}}{\alpha^{k+1}}$
5. If $\|v^{k+1} - v^k\| < \varepsilon$, then done. Else, $k = k + 1$, go to Step 2.

The division by the norm of the vector in Step 4 is not a necessary step, but it keeps the size of the values of the eigenvector close to 1. Recall that a scalar times an eigenvector of A is still an eigenvector of A; therefore, scaling has no adverse consequence. However, without Step 4 and $\alpha^k = 1$ for all k, the values of the updated vector may increase or decrease to the extent that the computer accuracy is affected.

Example 7.1

Use the power method to find the eigenvector corresponding to the dominant eigenvalue of the following matrix:

$$A = \begin{bmatrix} 6 & -2 \\ -8 & 3 \end{bmatrix}$$

Solution 7.1 Start with the initial guess $v^0 = [1 \;\; 1]^T$. Then

$$w^1 = Av^0 = \begin{bmatrix} 4 \\ -5 \end{bmatrix}$$
$$\alpha^1 = \|w^1\| = 6.4031$$
$$v^1 = \begin{bmatrix} 0.6247 \\ -0.7809 \end{bmatrix}$$

The second iteration follows:

$$w^2 = Av^1 = \begin{bmatrix} 5.3099 \\ -7.3402 \end{bmatrix}$$
$$\alpha^2 = \|w^2\| = 9.0594$$
$$v^2 = \begin{bmatrix} 0.5861 \\ -0.8102 \end{bmatrix}$$

Continuing to convergence yields the eigenvector:

$$v^* = \begin{bmatrix} 0.5851 \\ -0.8110 \end{bmatrix}$$

From the eigenvector, the corresponding eigenvalue is calculated

$$\lambda = \frac{\begin{bmatrix} 0.5851 & -0.8110 \end{bmatrix} \begin{bmatrix} 6 & -2 \\ -8 & 3 \end{bmatrix} \begin{bmatrix} 0.5851 \\ -0.8110 \end{bmatrix}}{\begin{bmatrix} 0.5851 & -0.8110 \end{bmatrix} \begin{bmatrix} 0.5851 \\ -0.8110 \end{bmatrix}} = \frac{8.7720}{1} = 8.7720$$

which is the largest eigenvalue of A (the smaller eigenvalue is 0.2280). ■

To see why the power method converges to the dominant eigenvector, let the initial guess vector v^0 be expressed as the linear combination

$$v^0 = \sum_{i=1}^{n} \beta_i v_i \tag{7.7}$$

where v_i are the actual eigenvectors of A and the coefficients β_i are chosen to make Equation (7.7) hold. Then, applying the power method (without loss of generality it can be assumed that $\alpha^k = 1$ for all k) yields

$$\begin{aligned} v^{k+1} &= Av^k = A^2 v^{k-1} = \ldots = A^{k+1} v^0 \\ &= \sum_{i=1}^{n} \lambda_i^{k+1} \beta_i v_i = \lambda_1^{k+1} \left(\beta_1 v_1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1} \right)^{k+1} \beta_i v_i \right) \end{aligned}$$

Because

$$\left|\frac{\lambda_i}{\lambda_1}\right| < 1 \quad i = 2, \ldots, n$$

then, as k successively increases, these terms go to zero, and only the component corresponding to v_1 is left.

The power method will fail if there are two largest eigenvalues of the same absolute magnitude. Recall that the eigenvalues of a real matrix A are in general complex and occur in conjugate pairs (which necessarily have the same absolute value). Therefore, if the largest eigenvalue of A is not real, then the power method will certainly fail to converge. For this reason, it is sensible to apply the power method only to matrices whose eigenvalues are known to be real. One class of real eigenvalue matrices is symmetric matrices.

There are also cases in which the power method may not converge to the eigenvector corresponding to the dominant eigenvalue. This would occur in the case in which $\beta_1 = 0$. This implies that the initial guess v^0 contains no component of the eigenvector v_1. In this case, the method will converge to the eigenvector contained in the decomposition of v^0 of the next largest eigenvalue.

The rate of convergence of the power method is determined by the ratio $|\frac{\lambda_2}{\lambda_1}|$. Thus, if $|\lambda_2|$ is only slightly smaller than $|\lambda_1|$, then the power method will converge slowly and a large number of iterations will be required to meet the required accuracy.

There are several extensions to the power method. For example, if, instead of the dominant eigenvalue, the smallest eigenvalue was desired, then the power method can be applied to A^{-1}. Since the eigenvalues of A^{-1} are $\frac{1}{\lambda_n}, \ldots, \frac{1}{\lambda_1}$, the inverse power method should converge to $\frac{1}{\lambda_n}$.

Another extension is the spectral shift. This approach uses the fact that the eigenvalues of $A - aI$ are $\lambda_1 - a, \ldots, \lambda_n - a$. Thus, having computed the first eigenvalue λ_1, the power method can be reapplied using the shifted matrix $A - \lambda_1 I$. This reduces the first eigenvalue to zero, and the power method now converges to the largest in absolute value of $\lambda_2 - \lambda_1, \ldots, \lambda_n - \lambda_1$.

7.2 The QR Algorithm

Many methods for solving the eigenvalue problem are based on a sequence of similarity transformations with orthogonal matrices. Thus, if P is any nonsingular matrix, then the matrices A and PAP^{-1} have the same eigenvalues. Furthermore, if v is an eigenvector of A, then Pv is an eigenvector of PAP^{-1}. If the matrix P is orthogonal, then the condition of the eigenproblem is not affected. This is the basis for the similarity transformation methods.

The QR method [21], [57], [58] is one of the most widely used decomposition methods for calculating eigenvalues of matrices. It uses a sequence of orthogonal similarity transformations [14] [31] such that $A = A_0, A_1, A_2, \ldots$ is computed by

$$A_i = Q_i R_i, \quad R_i Q_i = A_{i+1}, \quad i = 0, 1, 2, \ldots,$$

Similar to the LU factorization, the matrix A can also be factored into two matrices such that

$$A = QR \tag{7.8}$$

where Q is a unitary matrix and R is an upper triangular matrix. The matrix Q is *unitary* if

$$QQ^* = Q^*Q = I \tag{7.9}$$

where $(*)$ denotes a complex conjugate transpose.

Examples of unitary matrices are

$$Q_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Q_2 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

It also follows that the inverse of a unitary matrix is also its conjugate transpose, i.e.,

$$Q^{-1} = Q^*$$

This decomposition yields the column vectors $[a_1, a_2, \ldots, a_n]$ of A and column vectors $[q_1, q_2, \ldots, q_n]$ of Q such that

$$a_k = \sum_{i=1}^{k} r_{ik} q_i, \quad k = 1, \ldots, n \tag{7.10}$$

The column vectors $a_1, a_2, \ldots, a_n$ must be orthonormalized from left to right into an orthonormal basis $q_1, q_2, \ldots, q_n$.

In the implementation of the QR algorithm, it is common practice to transform A into a Hessenberg matrix H having the same eigenvalues and then apply the QR matrix to H. In the end, the matrix becomes upper triangular and the eigenvalues can be read off the diagonal. A Hessenberg matrix is essentially an upper triangular matrix with one extra set of nonzero elements directly below the diagonal. The reason for reducing A to a Hessenberg matrix is that this greatly reduces the total number of operations required for

the QR algorithm. A Hessenberg matrix has the form

$$H = \begin{bmatrix} * & * & * & * & \cdots & * & * & * & * \\ * & * & * & * & \cdots & * & * & * & * \\ 0 & * & * & * & \cdots & * & * & * & * \\ 0 & 0 & * & * & \cdots & * & * & * & * \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ddots & * & * & * & * \\ 0 & 0 & 0 & 0 & \cdots & * & * & * & * \\ 0 & 0 & 0 & 0 & \cdots & 0 & * & * & * \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & * & * \end{bmatrix}$$

where $*$ indicates a nonzero entry.

The Householder method is one method used to reduce A to a Hessenberg matrix. For each $n \times n$ matrix A, there exist $n-2$ Householder matrices $H_1, H_2, \ldots, H_{n-2}$, such that for

$$Q = H_{n-2} \ldots H_2 H_1$$

the matrix

$$P = Q^* A Q$$

is a Hessenberg matrix [29]. A matrix H is a Householder matrix if

$$H = I - 2\frac{vv^*}{v^*v}$$

Note that Householder matrices are also unitary matrices. The vector v is chosen to satisfy

$$v_i = a_i \pm e_i \left\| a_i \right\|_2 \tag{7.11}$$

where the choice of sign is based upon the requirement that $\|v\|_2$ should not be too small, e_i is the ith column of I, and a_i is the ith column of A.

Example 7.2

Find the QR decomposition of the matrix A

$$A = \begin{bmatrix} 1 & 3 & 4 & 8 \\ 2 & 1 & 2 & 3 \\ 4 & 3 & 5 & 8 \\ 9 & 2 & 7 & 4 \end{bmatrix}$$

Solution 7.2 The first transformation will be applied to zero out the first column of A below the subdiagonal; thus

$$v_1 = a_1 + e_1 \left\| a_1 \right\|_2$$

$$= \begin{bmatrix} 1 \\ 2 \\ 4 \\ 9 \end{bmatrix} + 10.0995 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 11.0995 \\ 2.0000 \\ 4.0000 \\ 9.0000 \end{bmatrix}$$

leading to

$$H_1 = I - 2\frac{v_1 v_1^*}{(v_1^* v_1)}$$

$$= \begin{bmatrix} 1\ 0\ 0\ 0 \\ 0\ 1\ 0\ 0 \\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 1 \end{bmatrix} - \frac{2}{224.1990} \begin{bmatrix} 11.0995 \\ 2.0000 \\ 4.0000 \\ 9.0000 \end{bmatrix} \begin{bmatrix} 11.0995\ 2.0000\ 4.0000\ 9.0000 \end{bmatrix}$$

$$= \begin{bmatrix} -0.0990 & -0.1980 & -0.3961 & -0.8911 \\ -0.1980 & 0.9643 & -0.0714 & -0.1606 \\ -0.3961 & -0.0714 & 0.8573 & -0.3211 \\ -0.8911 & -0.1606 & -0.3211 & 0.2774 \end{bmatrix}$$

and

$$H_1 A = \begin{bmatrix} -10.0995 & -3.4655 & -9.0103 & -8.1192 \\ 0 & -0.1650 & -0.3443 & 0.0955 \\ 0 & 0.6700 & 0.3114 & 2.1910 \\ 0 & -3.2425 & -3.5494 & -9.0702 \end{bmatrix}$$

The second iteration will operate on the part of the transformed matrix that excludes the first column and row. Therefore,

$$v_2 = a_2 + e_2 \|a_2\|_2$$

$$= \begin{bmatrix} -0.1650 \\ 0.6700 \\ -3.2425 \end{bmatrix} + 3.3151 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 3.1501 \\ 0.6700 \\ -3.2425 \end{bmatrix}$$

which results in

$$H_2 = I - 2\frac{v_2 v_2^*}{(v_2^* v_2)}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.0498 & -0.2021 & 0.9781 \\ 0 & -0.2021 & 0.9570 & 0.2080 \\ 0 & 0.9781 & 0.2080 & -0.0068 \end{bmatrix}$$

and

$$H_2H_1A = \begin{bmatrix} -10.0995 & -3.4655 & -9.0103 & -8.1192 \\ 0 & -3.3151 & -3.5517 & -9.3096 \\ 0 & 0 & -0.3708 & 0.1907 \\ 0 & 0 & -0.2479 & 0.6108 \end{bmatrix}$$

Continuing the process yields

$$v_3 = \begin{bmatrix} 0.0752 \\ -0.2479 \end{bmatrix}$$

$$H_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.8313 & 0.5558 \\ 0 & 0 & 0.5558 & -0.8313 \end{bmatrix}$$

which results in

$$R = H_3H_2H_1A = \begin{bmatrix} -10.0995 & -3.4655 & -9.0103 & -8.1192 \\ 0 & -3.3151 & -3.5517 & -9.3096 \\ 0 & 0 & -0.4460 & 0.4980 \\ 0 & 0 & 0 & -0.4018 \end{bmatrix}$$

and

$$Q = H_1H_2H_3 = \begin{bmatrix} -0.0990 & -0.8014 & -0.5860 & -0.0670 \\ -0.1980 & -0.0946 & 0.2700 & -0.9375 \\ -0.3961 & -0.4909 & 0.7000 & 0.3348 \\ -0.8911 & 0.3283 & -0.3060 & 0.0670 \end{bmatrix}$$

It can be verified that $A = QR$ and further that $Q^* = Q^{-1}$. ■

Elimination by QR decomposition can be considered as an alternative to Gaussian elimination. However, the number of multiplications and divisions required is more than twice the number required for Gaussian elimination. Therefore, QR decomposition is seldom used for the solution of linear systems, but it does play an important role in the calculation of eigenvalues.

Although the eigenvalue problem gives rise to a simple set of algebraic equations to determine the solution to

$$det\,(A - \lambda I) = 0$$

the practical problem of solving this equation is difficult. Computing the roots of the characteristic equation or the nullspace of a matrix is a process that is not well suited for computers. In fact, no generalized direct process exists for solving the eigenvalue problem in a finite number of steps. Therefore, iterative methods for calculation must be relied upon to produce a series of successively improved approximations to the eigenvalues of a matrix.

The QR method is commonly used to calculate the eigenvalues and eigenvectors of full matrices. As developed by Francis [14], the QR method produces

a series of similarity transformations

$$A_k = Q_k^* A_{k-1} Q_k \quad Q_k^* Q_k = I \tag{7.12}$$

where the matrix A_k is similar to A. The QR decomposition is repeatedly performed and applied to A as the subdiagonal elements are iteratively driven to zero. At convergence, the eigenvalues of A in descending order by magnitude appear on the diagonal of A_k.

The next step is to find the eigenvectors associated with each eigenvalue. Recall that

$$Av_i = \lambda_i v_i \tag{7.13}$$

for each eigenvalue and corresponding eigenvector $i = 1, \ldots, n$. Equation (7.13) may also be written as

$$Av_i - \lambda_i v_i = 0$$

In other words, the matrix defined by $A - \lambda_i I$ is singular; thus only three of its rows (or columns) are independent. This fact can be used to determine the eigenvectors once the eigenvalues are known. Since $A - \lambda_i I$ is not of full rank, one of the elements of the eigenvector v_i can be chosen arbitrarily. To start, partition $A - \lambda_i I$ as

$$A - \lambda_i I = \begin{bmatrix} a_{11} & a_{1,2n} \\ a_{2n,1} & a_{2n,2n} \end{bmatrix}$$

where a_{11} is a scalar, $a_{1,2n}$ is a $1 \times (n-1)$ vector, $a_{2n,1}$ is a $(n-1) \times 1$ vector, and $a_{2n,2n}$ is an $(n-1) \times (n-1)$ matrix of rank $(n-1)$. Then let $v_i(1) = 1$ and solve for the remaining portion of the eigenvector as

$$\begin{bmatrix} v_i(2) \\ v_i(3) \\ \vdots \\ v_i(n) \end{bmatrix} = -a_{2n,2n}^{-1} a_{2n,1} v_i(1) \tag{7.14}$$

Now update $v_i(1)$ from

$$v_i(1) = -\frac{1}{a_{11}} a_{1,2n} * \begin{bmatrix} v_i(2) \\ v_i(3) \\ \vdots \\ v_i(n) \end{bmatrix}$$

Then the eigenvector corresponding to λ_i is

$$v_i = \begin{bmatrix} v_i(1) \\ v_i(2) \\ \vdots \\ v_i(n) \end{bmatrix}$$

The last step is to normalize the eigenvector; therefore,

$$v_i = \frac{v_i}{\|v_i\|}$$

Example 7.3
Find the eigenvalues and eigenvectors of the matrix of Example 7.2.

Solution 7.3 The first objective is to find the eigenvalues of the matrix A using the QR method. From Example 7.2, the first QR factorization yields the Q_0 matrix

$$Q_0 = \begin{bmatrix} -0.0990 & -0.8014 & -0.5860 & -0.0670 \\ -0.1980 & -0.0946 & 0.2700 & -0.9375 \\ -0.3961 & -0.4909 & 0.7000 & 0.3348 \\ -0.8911 & 0.3283 & -0.3060 & 0.0670 \end{bmatrix}$$

Using the given A matrix as A_0, the first update A_1 is found by

$$\begin{aligned} A_1 &= Q_0^* A_0 Q_0 \\ &= \begin{bmatrix} 12.4902 & 10.1801 & 1.1599 & 0.3647 \\ 10.3593 & -0.9987 & -0.5326 & 1.2954 \\ -0.2672 & 0.3824 & -0.4646 & -0.1160 \\ 0.3580 & -0.1319 & 0.1230 & -0.0269 \end{bmatrix} \end{aligned}$$

The QR factorization of A_1 yields

$$Q_1 = \begin{bmatrix} -0.7694 & -0.6379 & -0.0324 & -0.0006 \\ -0.6382 & 0.7660 & 0.0733 & -0.0252 \\ 0.0165 & -0.0687 & 0.9570 & 0.2812 \\ -0.0221 & 0.0398 & -0.2786 & 0.9593 \end{bmatrix}$$

and the A_2 matrix becomes

$$\begin{aligned} A_2 &= Q_1^* A_1 Q_1 \\ &= \begin{bmatrix} 17.0913 & 4.8455 & -0.2315 & -1.0310 \\ 4.6173 & -5.4778 & -1.8116 & 0.6064 \\ -0.0087 & 0.0373 & -0.5260 & -0.1757 \\ 0.0020 & -0.0036 & 0.0254 & -0.0875 \end{bmatrix} \end{aligned}$$

Note that the elements below the diagonals are slowly decreasing to zero. This process is carried out until the final A matrix is obtained:

$$A_* = \begin{bmatrix} 18.0425 & 0.2133 & -0.5180 & -0.9293 \\ 0 & -6.4172 & -1.8164 & 0.6903 \\ 0 & 0 & -0.5269 & -0.1972 \\ 0 & 0 & 0 & -0.0983 \end{bmatrix} \tag{7.15}$$

The eigenvalues are on the diagonals of A_* and are in decreasing order by magnitude. Thus the eigenvalues are

$$\lambda_{1,\ldots,4} = \begin{bmatrix} 18.0425 \\ -6.4172 \\ -0.5269 \\ -0.0983 \end{bmatrix}$$

The corresponding eigenvectors are:

$$\begin{bmatrix} 0.4698 \\ 0.2329 \\ 0.5800 \\ 0.6234 \end{bmatrix}, \begin{bmatrix} 0.6158 \\ 0.0539 \\ 0.2837 \\ -0.7330 \end{bmatrix}, \begin{bmatrix} 0.3673 \\ -0.5644 \\ -0.5949 \\ 0.4390 \end{bmatrix}, \begin{bmatrix} 0.0932 \\ 0.9344 \\ -0.2463 \\ -0.2400 \end{bmatrix}$$

■

7.2.1 Deflation

Once the elements of the last row are reduced to zero, the last row and column of the matrix may be neglected. This implies that the smallest eigenvalue is "deflated" by removing the last row and column. The procedure can then be repeated on the remaining $(n-1) \times (n-1)$ matrix.

7.2.2 Shifted QR

The speed of convergence of the QR method for calculating eigenvalues depends greatly on the location of the eigenvalues with respect to one another. The matrix $A - \sigma I$ has the eigenvalues $\lambda_i - \sigma$ for $i = 1, \ldots, n$. If σ is chosen as an approximate value of the smallest eigenvalue λ_n, then $\lambda_n - \sigma$ becomes small. This will speed up the convergence in the last row of the matrix, since

$$\frac{|\lambda_n - \sigma|}{|\lambda_{n-1} - \sigma|} \ll 1$$

The QR iterations can converge very slowly in many instances. However, if some information about one or more of the eigenvalues is known a priori, then a variety of techniques can be applied to speed up convergence of the iterations. One such technique is the *shifted* QR method, in which a shift σ is introduced at each iteration such that the QR factorization at the kth iteration is performed on

$$A_k - \sigma I = Q_k R_k \tag{7.16}$$

and

$$A_{k+1} = Q_k^* \left(A_k - \sigma I \right) Q_k + \sigma I \tag{7.17}$$

If σ is a good estimate of an eigenvalue, then the $(n, n-1)$ entry of A_k will converge rapidly to zero, and the (n, n) entry of A_{k+1} will converge to the eigenvalue closest to σ_k. The nth row and column can be removed (deflated), and an alternate shift can be applied.

Using a shift and deflation in combination can significantly improve convergence. Additionally, if only one eigenvalue is desired of a particular magnitude, this eigenvalue can be isolated via the shift method. After the last row has been driven to zero, the eigenvalue can be obtained and the remainder of the QR iterations abandoned.

Example 7.4
Find the largest eigenvalue of Example 7.3 using shifts and deflation.

Solution 7.4 Start with using a shift of $\sigma = 15$. This is near the 18.0425 eigenvalue, so convergence to that particular eigenvalue should be rapid. Starting with the original A matrix as A_0, the QR factorization of $A_0 - \sigma I$ yields

$$Q_0 = \begin{bmatrix} 0.8124 & -0.0764 & -0.2230 & -0.5334 \\ -0.1161 & 0.9417 & 0.0098 & -0.3158 \\ -0.2321 & -0.2427 & 0.7122 & -0.6164 \\ -0.5222 & -0.2203 & -0.6655 & -0.4856 \end{bmatrix}$$

and the update $A_1 = Q_0^* (A_0 - \sigma I) Q_0 + \sigma I$

$$A_1 = \begin{bmatrix} -4.9024 & 0.8831 & -1.6174 & 2.5476 \\ -0.2869 & 0.0780 & -0.1823 & 1.7775 \\ -2.9457 & 0.5894 & -1.5086 & 2.3300 \\ 2.5090 & 1.0584 & 3.1975 & 17.3330 \end{bmatrix}$$

The eigenvalue of interest ($\lambda = 18.0425$) will now appear in the lower right corner since, as the iterations progress, $A_{k+1}(n, n) - \sigma$ will be the smallest diagonal in magnitude. Recall that the eigenvalues are ordered on the diagonal from largest to smallest, and, since the largest eigenvalue is "shifted" by σ, it will now have the smallest magnitude. The convergence can be further increased by updating σ at each iteration, such that $\sigma_{k+1} = A_{k+1}(n, n)$. The iterations proceed as in Example 7.3. ■

7.2.3 Double Shifted QR

One problem that arises is if the eigenvalues sought are complex. As eigenvalues come in conjugate pairs, the QR may appear to fail since no dominant eigenvalue exists. However, instead of generating a single eigenvalue estimation, QR produces a 2×2 matrix "containing" the conjugate pair. By using deflation, the 2×2 matrix can be extracted and the eigenvalues easily computed. The remaining eigenvalues can then be computed as before.

Another approach is that, since complex eigenvalues always appear in conjugate pairs, it is natural to search for the pair simultaneously. This can be accomplished by collapsing two shifted QR steps in one double step with the two shifts being complex conjugates of each other.

Let σ_1 and σ_2 be two eigenvalues of a real matrix, where $\sigma_1^* = \sigma_2$. Performing two QR steps using Equations (7.16) and (7.17) yields

$$A_0 = Q_0 R_0 + \sigma_1 I \tag{7.18}$$
$$A_1 = Q_0^* \left(A_0 - \sigma_1 I\right) Q_0 + \sigma_1 I \tag{7.19}$$
$$= R_0 Q_0 + \sigma_1 I \tag{7.20}$$

and

$$A_1 = Q_1 R_1 + \sigma_2 I \tag{7.21}$$
$$A_2 = Q_1^* \left(A_1 - \sigma_2 I\right) Q_1 + \sigma_2 I \tag{7.22}$$
$$\tag{7.23}$$

Combining Equations (7.20) and (7.21) yields

$$R_0 Q_0 + \left(\sigma_1 - \sigma_2\right) I = Q_1 R_1 \tag{7.24}$$

Multiplying by Q_0 on the left and R_0 on the right results in

$$Q_0 R_0 Q_0 R_0 + Q_0 \left(\sigma_1 - \sigma_2\right) R_0 = Q_0 Q_1 R_1 R_0 \tag{7.25}$$

Thus

$$Q_0 Q_1 R_1 R_0 = Q_0 R_0 \left(Q_0 R_0 + \left(\sigma_1 - \sigma_2\right) I\right) \tag{7.26}$$
$$= \left(A_0 - \sigma_1 I\right)\left(A_0 - \sigma_2 I\right) \tag{7.27}$$
$$= A_0^2 - 2\mathrm{Re}(\sigma) A_0 + |\sigma|^2 I \tag{7.28}$$

The right-hand side of Equation (7.28) is a real matrix; thus $\left(Q_0 Q_1\right)\left(R_1 R_0\right)$ is the QR factorization of this real matrix:

$$A_2 = \left(Q_0 Q_1\right)^* A_0 \left(Q_0 Q_1\right) \tag{7.29}$$

Example 7.5

Find the eigenvalues of

$$A = \begin{bmatrix} 9 & 3 & 13 & 7 & 17 \\ 4 & 6 & 15 & 9 & 1 \\ 19 & 9 & 5 & 11 & 19 \\ 20 & 12 & 3 & 2 & 15 \\ 9 & 6 & 6 & 6 & 10 \end{bmatrix}$$

Solution 7.5 Applying the QR as in Example 7.3 yields the following updated A_{k+1} after 12 iterations:

$$A_{13} = \left[\begin{array}{c|cc|cc} 46.6670 & 5.4606 & 8.0469 & -13.9039 & -7.0580 \\ \hline 0.0000 & -5.8266 & 1.6519 & 2.1452 & 0.8102 \\ 0.0000 & -10.4370 & -12.3090 & 5.6634 & 4.1548 \\ \hline 0.0000 & 0.0000 & -0.0000 & -0.8085 & -1.3722 \\ 0.0000 & 0.0000 & -0.0000 & 9.3775 & 4.2771 \end{array}\right] \tag{7.30}$$

This matrix is now in upper triangular block form. The eigenvalues of the two 2×2 matrices can be computed. The eigenvalues of

$$\begin{bmatrix} -5.8266 & 1.6519 \\ -10.4370 & -12.3090 \end{bmatrix}$$

are $-9.0678 \pm j2.5953$ and the eigenvalues of

$$\begin{bmatrix} -0.8085 & -1.3722 \\ 9.3775 & 4.27710 \end{bmatrix}$$

are $1.7343 \pm j2.5303$. The real eigenvalue 46.6670 completes the set of eigenvalues of A. ■

7.3 Arnoldi Methods

In large interconnected systems, it is either impractical or intractable to find all of the eigenvalues of the system state matrix due to restrictions on computer memory and computational speed. The Arnoldi method has been developed as an algorithm that iteratively computes k eigenvalues of an $n \times n$ matrix A, where k is typically much smaller than n. This method therefore bypasses many of the constraints imposed by large matrix manipulation required by methods such as the QR decomposition. If the k eigenvalues are chosen selectively, they can yield rich information about the system under consideration, even without the full set of eigenvalues. The Arnoldi method was first developed in [3], but suffered from poor numerical properties such as loss of orthogonality and slow convergence. Several modifications to the Arnoldi method have overcome these shortcomings. The Modified Arnoldi Method (MAM) has been used frequently in solving eigenvalue problems in power system applications [32], [56]. This approach introduced preconditioning and explicit restart techniques to retain orthogonality. Unfortunately, however, an explicit restart will often discard useful information. The restart problem was solved by using implicitly shifted QR steps [49] in the Implicitly Restarted Arnoldi (IRA) method. Several commercial software packages have

been developed around the IRA method, including the well-known ARPACK and the MATLAB `speig` routines.

The basic approach of the Arnoldi method is to iteratively update a low-order matrix H whose eigenvalues successively approximate the selected eigenvalues of the larger A matrix, such that

$$AV = VH; \quad V^*V = I \tag{7.31}$$

where V is an $n \times k$ matrix and H is a $k \times k$ Hessenberg matrix. As the method progresses, the eigenvalues of A are approximated by the diagonal entries of H, yielding

$$HV_i = V_i D \tag{7.32}$$

where V_i is a $k \times k$ matrix whose columns are the eigenvalues of H (approximating the eigenvectors of A) and D is a $k \times k$ matrix whose diagonal entries are the eigenvalues of H (approximating the eigenvalues of A). The Arnoldi method is an orthogonal projection method onto a *Krylov* subspace.

The Arnoldi procedure is an algorithm for building an orthogonal basis of the Krylov subspace. One approach is given as:

The k-step Arnoldi Factorization

Starting with a vector v_1 of unity norm, for $j = 1, \ldots, k$ compute:

1. $H(i,j) = v_i^T A v_j$ for $i = 1, \ldots, j$
2. $w_j = Av_j - \sum_{i=1}^{j} H(i,j) v_i$
3. $H(j+1, j) = \|w_j\|_2$
4. If $H(j+1, j) = 0$, then stop
5. $v_{j+1} = \frac{w_j}{H(j+1,j)}$

At each step, the algorithm multiplies the previous Arnoldi vector v_j by A and then orthonormalizes the resulting vector w_j against all previous v_i's. The k-step Arnoldi factorization is shown in Figure 7.1 and is given by

$$AV_k = V_k H_k + w_k e_k^T \tag{7.33}$$

The columns $V = [v_1,\ v_2, \ldots, v_k]$ form an orthonormal basis for the Krylov subspace and H is the orthogonal projection of A onto this space. It is desirable for $\|w_k\|$ to become small because this indicates that the eigenvalues of H are accurate approximations to the eigenvalues of A. However, this "convergence" often comes at the price of numerical orthogonality in V. Therefore, the k-step Arnoldi factorization is "restarted" to preserve orthogonality.

Implicit restarting provides a means to extract rich information from very large Krylov subspaces while avoiding the storage and poor numerical properties associated with the standard approach. This is accomplished by continually compressing the information into a fixed size k-dimensional subspace, by

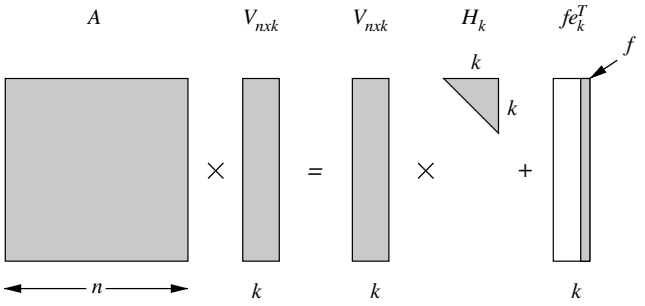

FIGURE 7.1
A k-step Arnoldi factorization

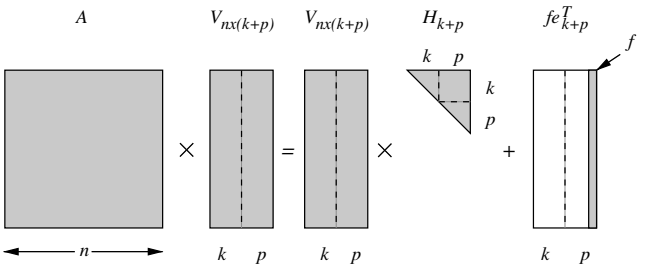

FIGURE 7.2
A $(k+p)$-step Arnoldi factorization

using a shifted QR mechanism. A $(k+p)$-step Arnoldi factorization

$$AV_{k+p} = V_{k+p}H_{k+p} + w_{k+p}e_{k+p}^T \tag{7.34}$$

is compressed to a factorization of length k that retains the eigen-information of interest. This is accomplished using QR steps to apply p shifts to yield

$$A\hat{V}_{k+p} = \hat{V}_{k+p}\hat{H}_{k+p} + \hat{w}_{k+p} \tag{7.35}$$

where $\hat{V}_{k+p} = V_{k+p}Q$, $\hat{H}_{k+p} = Q^*H_{k+p}Q$, and $\hat{w}_{k+p} = w_{k+p}e_{k+p}^TQ$. It may be shown that the first $k-1$ entries of the vector e_{k+p}^TQ are zero [50]. Equating the first k columns on both sides yields an updated k-step Arnoldi factorization. This now provides the "restart" vectors for extending the k-step Arnoldi factorization to the $(k+p)$-step Arnoldi factorization, shown in Figure 7.2.

The implicitly restarted Arnoldi algorithm consists of three main steps: initialization, iteration/refinement, and final calculation of the eigenvalues

and eigenvectors.

Implicitly Restarted Arnoldi Algorithm

1. Initialization

   Using the vector v_1 as a starting vector, generate a k-step Arnoldi factorization. At each step k of the factorization, the vector V_k is augmented by a vector v_k satisfying Equation (7.33). Note that H_k is a Hessenberg matrix. The shaded regions in Figure 7.1 represent nonzero entries. The unshaded region of $w_k e_k^T$ is a zero matrix of $(k-1)$ columns. The Arnoldi factorization is entirely dependent on the choice of initial vector v_1.

2. Iteration/Refinement

   (a) Extend the k-step factorization by p steps.

   Each of the p additions represents an eigenvalue/eigenvector that can be discarded at the end of the iteration if it does not meet the chosen criteria. In general, the choice of p is a trade-off between the length of factorization that may be tolerated and the rate of convergence. For most problems, the size of p is determined experimentally. The only requirement is that $1 \leq p \leq n-k$.

   (b) Calculate eigenvalues of H_{k+p}

   After the p-step extension has been completed, the eigenvalues of H_{k+p} are calculated by the QR method and sorted according to a predetermined sort criterion S and ordered from best to worst. The p worst eigenvalues $(\sigma_1, \sigma_2, \ldots, \sigma_p)$ are used as shifts to perform p shifted QR factorizations. Since the matrix H_{k+p} in the Arnoldi factorization

   $$AV_{k+p} = V_{k+p}H_{k+p} + w_{k+p}e_{k+p}^T \tag{7.36}$$

   is relatively small, the shifted QR factorization can be used efficiently to calculate the eigenvalues of H.

   (c) Update the Arnoldi matrices

   $$\hat{V}_{k+p} = V_{k+p}Q$$
   $$\hat{H}_{k+p} = Q^* H_{k+p} Q$$
   $$\hat{w}_{k+p} = w_{k+p}e_{k+p}^T Q$$

   Note that the updated matrix $\hat{V}_{k+p}$ has orthonormal columns since it is the product of V and an orthogonal matrix Q.

   (d) Obtain a new k-step Arnoldi factorization by equating the first k columns on each side of Equation (7.35) and discarding the last p equations:

   $$A\hat{V}_k = \hat{V}_k \hat{H}_k + \hat{w}_k e_k^T$$

The vector $\hat{w}$ is the new residual vector that is being driven to zero.

(e) If

$$\|AV_k - V_k H_k\| \leq \varepsilon$$

where ε is the preselected convergence tolerance, then the iteration/refinement terminates. Otherwise the process is repeated until tolerance is achieved.

3. Eigenvalue/Eigenvector Calculation

The last step in the Arnoldi method is to compute the eigenvalues and eigenvectors of the reduced matrix H_k from

$$H_k V_k + V_h D_k \tag{7.37}$$

The eigenvectors of A are then calculated as

$$V_k = V_k V_h \tag{7.38}$$

and the desired eigenvalues of A may be obtained from the diagonal entries of D_k:

$$AV_k = V_k D_k \tag{7.39}$$

Example 7.6

Using a three-step Arnoldi factorization, find the two smallest (in magnitude) eigenvalues and corresponding eigenvectors of the matrix of Example 7.2.

Solution 7.6 Since the two smallest eigenvalues are desired, the value of k is two. After the initialization step, the two-step Arnoldi method will be extended up to three steps; therefore, p is one. Thus, at each step, three eigenvalues will be calculated and the worst eigenvalue will be discarded.

The factorization can be initialized with an arbitrary nonzero vector. In many software implementations, the starting vector is chosen randomly such that all of the entries have absolute value less than 0.5. The starting vector for this example will be

$$v_0 = \begin{bmatrix} 0.2500 \\ 0.2500 \\ 0.2500 \\ 0.2500 \end{bmatrix}$$

To satisfy the requirement that the initial vector have unity norm, the starting vector is normalized to yield

$$v_1 = \frac{Av_0}{\|Av_0\|} = \begin{bmatrix} 0.4611 \\ 0.2306 \\ 0.5764 \\ 0.6340 \end{bmatrix}$$

After the initial vector has been chosen, the Arnoldi factorization is applied for k steps; thus

$$h_{2,1}v_2 = Av_1 - h_{1,1}v_1 \tag{7.40}$$

where v_2 produces the second column of the matrix V_k and $h_{1,1}$ is chosen such that

$$h_{1,1} = \langle v_1, Av_1 \rangle = v_1^T Av_1 \tag{7.41}$$

where $\langle \cdot \rangle$ denotes inner product. Thus, solving Equation (7.41) yields $h_{1,1} = 18.0399$. Applying the Arnoldi factorization for w_1 yields

$$\begin{aligned} w_1 = h_{2,1}v_2 &= Av_1 - h_{1,1}v_1 \\ &= \begin{bmatrix} 1 & 3 & 4 & 8 \\ 2 & 1 & 2 & 3 \\ 4 & 3 & 5 & 8 \\ 9 & 2 & 7 & 4 \end{bmatrix} \begin{bmatrix} 0.4611 \\ 0.2306 \\ 0.5764 \\ 0.6340 \end{bmatrix} - (18.0399) \begin{bmatrix} 0.4611 \\ 0.2306 \\ 0.5764 \\ 0.6340 \end{bmatrix} \\ &= \begin{bmatrix} 0.2122 \\ 0.0484 \\ 0.0923 \\ -0.2558 \end{bmatrix} \end{aligned}$$

The factor $h_{2,1}$ is chosen to normalize v_2 to unity; thus $h_{2,1} = 0.3483$, and

$$v_2 = \begin{bmatrix} 0.6091 \\ 0.1391 \\ 0.2650 \\ -0.7345 \end{bmatrix}$$

Calculating the remaining values of the Hessenberg matrix yields

$$\begin{aligned} h_{1,2} &= v_1^* A v_2 = 0.1671 \\ h_{2,2} &= v_2^* A v_2 = -6.2370 \end{aligned}$$

and

$$w_2 = h_{3,2}v_2 = Av_2 - h_{1,2}v_1 - h_{2,2}v_2 = \begin{bmatrix} -0.0674 \\ 0.5128 \\ -0.1407 \\ -0.0095 \end{bmatrix}$$

These values can be checked to verify that they satisfy Equation (7.33) for $i = 2$:

$$AV_2 = V_2 H_2 + w_2 \begin{bmatrix} 0 & 1 \end{bmatrix}$$

where

$$V_2 = [v_1 \;\; v_2] = \begin{bmatrix} 0.4611 & 0.6091 \\ 0.2306 & 0.1391 \\ 0.5764 & 0.2650 \\ 0.6340 & -0.7345 \end{bmatrix}$$

and

$$H_2 = \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} = \begin{bmatrix} 18.0399 & 0.1671 \\ 0.3483 & -6.2370 \end{bmatrix}$$

This completes the initialization stage.

After the initial k-step Arnoldi sequence has been generated, it can be extended to $k+p$ steps. In this example $p = 1$, so only one more extension is required. From the initialization, $w_2 = h_{3,2}v_2$ from which $h_{3,2}$ and v_2 can be extracted (recalling that $\|v_2\| = 1.0$) to yield $h_{3,2} = 0.5361$ and

$$v_3 = \begin{bmatrix} -0.1257 \\ 0.9565 \\ -0.2625 \\ -0.0178 \end{bmatrix}$$

The Hessenberg matrix H_3 becomes

$$H_3 = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ h_{2,1} & h_{2,2} & h_{2,3} \\ 0 & h_{3,2} & h_{3,3} \end{bmatrix} = \begin{bmatrix} 18.0399 & 0.1671 & 0.5560 \\ 0.3483 & -6.2370 & 2.0320 \\ 0 & 0.5361 & -0.2931 \end{bmatrix}$$

where

$$\begin{aligned} h_{1,3} &= v_1^T A v_3 \\ h_{2,3} &= v_2^T A v_3 \\ h_{3,3} &= v_3^T A v_3 \end{aligned}$$

and

$$w_3 = Av_3 - h_{1,3}v_1 - h_{2,3}v_2 - h_{3,3}v_3 = \begin{bmatrix} 0.0207 \\ -0.0037 \\ -0.0238 \\ 0.0079 \end{bmatrix}$$

The next step is to compute (using QR factorization) and sort the eigenvalues (and eigenvectors) of the small matrix H_3. The eigenvalues of H_3 are

$$\sigma = \begin{bmatrix} 18.0425 \\ -6.4166 \\ -0.1161 \end{bmatrix}$$

Since the smallest two eigenvalues are desired, the eigenvalues are sorted such that the desired eigenvalues are at the bottom (which they already are). The undesired eigenvalue estimate is $\sigma_1 = 18.0425$. Applying the shifted QR factorization to $H_3 - \sigma_1 I$ yields

$$H_3 - \sigma_1 I = \begin{bmatrix} -0.0026 & 0.1671 & 0.5560 \\ 0.3483 & -24.2795 & 2.0320 \\ 0 & 0.5361 & -18.3356 \end{bmatrix}$$

and

$$QR = \begin{bmatrix} -0.0076 & -0.0311 & 0.9995 \\ 1.0000 & -0.0002 & 0.0076 \\ 0 & 0.9995 & 0.0311 \end{bmatrix} \begin{bmatrix} 0.3483 & -24.2801 & 2.0277 \\ 0 & 0.5364 & -18.3445 \\ 0 & 0 & 0 \end{bmatrix}$$

From Q, the update $\hat{H}$ can be found:

$$\begin{aligned} \hat{H}_3 &= Q^* H_3 Q \\ &= \begin{bmatrix} -6.2395 & 2.0216 & 0.2276 \\ 0.5264 & -0.2932 & -0.5673 \\ 0 & 0 & 18.0425 \end{bmatrix} \end{aligned}$$

Note that the $\hat{H}(3,2)$ element is now zero. Continuing the algorithm yields the update for $\hat{V}$:

$$\hat{V} = V_3 Q = \begin{bmatrix} 0.6056 & -0.1401 & 0.4616 \\ 0.1373 & 0.9489 & 0.2613 \\ 0.2606 & -0.2804 & 0.5699 \\ -0.7392 & -0.0374 & 0.6276 \end{bmatrix}$$

where $V_3 = [v_1\ v_2\ v_3]$, and

$$\hat{w}_3 e^T = w_3 e_3^T Q = \begin{bmatrix} 0 & 0.0207 & 0.0006 \\ 0 & -0.0037 & -0.0001 \\ 0 & -0.0238 & -0.0007 \\ 0 & 0.0079 & 0.0002 \end{bmatrix}$$

Note that the first column of $\hat{w}e^T$ is zeros, so that a new k-step Arnoldi factorization can be obtained by equating the first k columns on each side such that

$$A\hat{V}_2 = \hat{V}_2 \hat{H}_2 + \hat{w}_2 e_2^T \tag{7.42}$$

The third columns of $\hat{V}$ and $\hat{H}$ are discarded.

This iteration/refinement procedure is continued until

$$\|AV - VH\| = \left\|we^T\right\| < \varepsilon$$

at which time the calculated eigenvalues will be obtained within order of ε accuracy. ■

7.4 Singular Value Decomposition

Singular value decomposition (SVD) produces three matrices whose product is the (possibly rectangular) matrix A. In matrix form, the SVD is

$$A = U\Sigma V^T \tag{7.43}$$

where U satisfies $U^T U = I$ and the columns of U are the orthonormal eigenvectors of AA^T, V satisfies $V^T V = I$ and the columns of V are the orthonormal eigenvectors of $A^T A$, and Σ is a diagonal matrix containing the square roots of the eigenvalues corresponding to U (or V) in descending order.

The SVD decomposition can be found by applying either the QR or Arnoldi method to the matrices $A^T A$ and AA^T to compute the eigenvalues and eigenvectors. Once the eigenvalues are found, the singular values are the square roots. The condition number of a matrix is a measure of the "invertibility" of a matrix and is defined as the ratio of the largest singular value to the smallest singular value. A large condition number indicates a nearly singular matrix.

Example 7.7

Find the singular value decomposition of

$$A = \begin{bmatrix} 1 & 2 & 4 & 9 & 3 \\ 3 & 1 & 3 & 2 & 6 \\ 4 & 2 & 5 & 7 & 7 \\ 8 & 3 & 8 & 4 & 10 \end{bmatrix}$$

Solution 7.7 The matrix A is 4×5; therefore U will be a 4×4 matrix, Σ will be a 4×5 matrix with the four singular values on the diagonal followed by a column of zeros, and V will be a 5×5 matrix.

Starting with

$$\hat{A} = A^T A$$

the QR method for finding the eigenvalues and eigenvectors of $\hat{A}$ yields

$$\hat{A} = \begin{bmatrix} 90 & 37 & 97 & 75 & 129 \\ 37 & 18 & 45 & 46 & 56 \\ 97 & 45 & 114 & 109 & 145 \\ 75 & 46 & 109 & 150 & 128 \\ 129 & 56 & 145 & 128 & 194 \end{bmatrix}$$

$$D = \begin{bmatrix} 507.6670 & 0 & 0 & 0 & 0 \\ 0 & 55.1644 & 0 & 0 & 0 \\ 0 & 0 & 3.0171 & 0 & 0 \\ 0 & 0 & 0 & 0.1516 & 0 \\ 0 & 0 & 0 & 0 & 0.0000 \end{bmatrix}$$

$$V^T = \begin{bmatrix} -0.3970 & 0.4140 & -0.3861 & -0.7189 & -0.0707 \\ -0.1865 & -0.0387 & -0.2838 & 0.3200 & -0.8836 \\ -0.4716 & 0.0610 & -0.5273 & 0.5336 & 0.4595 \\ -0.4676 & -0.8404 & 0.0688 & -0.2645 & 0.0177 \\ -0.6054 & 0.3421 & 0.6983 & 0.1615 & -0.0530 \end{bmatrix}$$

The matrix Σ has the squares of the singular values on the diagonals and must be the same dimension as A; thus

$$\Sigma = \begin{bmatrix} 22.5315 & 0 & 0 & 0 & 0 \\ 0 & 7.4273 & 0 & 0 & 0 \\ 0 & 0 & 1.7370 & 0 & 0 \\ 0 & 0 & 0 & 0.3893 & 0 \end{bmatrix}$$

To find U, repeat with $\hat{A} = AA^T$ to obtain

$$U = \begin{bmatrix} -0.3853 & 0.8021 & -0.2009 & 0.4097 \\ -0.3267 & -0.2367 & 0.7502 & 0.5239 \\ -0.5251 & 0.2161 & 0.3574 & -0.7415 \\ -0.6850 & -0.5039 & -0.5188 & 0.0881 \end{bmatrix}$$

■

In addition to condition number, another common use of the SVD is to calculate the pseudoinverse A^+ of a non-square matrix A. The most commonly encountered pseudoinverse is the Moore–Penrose matrix inverse, which is a special case of a general type of pseudoinverse known as a matrix 1-inverse. The pseudoinverse is commonly used to solve the least-squares problem $Ax = b$ when A is nonsingular or nonsquare. From Section 6.1, the least-squares problem solution is given by

$$\begin{aligned} x &= \left(A^T A\right)^{-1} Ab \\ &= A^+ b \end{aligned}$$

The matrix A^+ can be found through LU factorization, but a much more common approach is to use SVD. In this case, the pseudoinverse is given by

$$A^+ = V\Sigma^+ U^T \tag{7.44}$$

where Σ^+ is a matrix of the same dimension as A^T with the reciprocal of the singular values on the diagonal.

Example 7.8
Repeat Example 6.1 using a pseudoinverse.

Solution 7.8 The system of equations is repeated here for convenience:

$$\begin{bmatrix} 4.27 \\ -1.71 \\ 3.47 \\ 2.50 \end{bmatrix} = \begin{bmatrix} 0.4593 & -0.0593 \\ 0.0593 & -0.4593 \\ 0.3111 & 0.0889 \\ 0.0889 & 0.3111 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \tag{7.45}$$

Use the pseudoinverse to solve for

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

Applying the SVD

$$U = \begin{bmatrix} -0.5000 & -0.6500 & -0.5505 & -0.1567 \\ 0.5000 & -0.6500 & 0.1566 & 0.5505 \\ -0.5000 & -0.2785 & 0.8132 & -0.1059 \\ -0.5000 & 0.2785 & -0.1061 & 0.8131 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 0.5657 & 0 \\ 0 & 0.5642 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.7071 & -0.7071 \\ -0.7071 & 0.7071 \end{bmatrix}$$

The matrix Σ^+ is

$$\Sigma^+ = \begin{bmatrix} 1.7677 & 0 & 0 & 0 \\ 0 & 1.7724 & 0 & 0 \end{bmatrix}$$

leading to A^+

$$A^+ = \begin{bmatrix} 1.4396 & 0.1896 & 0.9740 & 0.2760 \\ -0.1896 & -1.4396 & 0.2760 & 0.9740 \end{bmatrix}$$

and

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 1.4396 & 0.1896 & 0.9740 & 0.2760 \\ -0.1896 & -1.4396 & 0.2760 & 0.9740 \end{bmatrix} \begin{bmatrix} 4.27 \\ -1.71 \\ 3.47 \\ 2.50 \end{bmatrix}$$

$$= \begin{bmatrix} 9.8927 \\ 5.0448 \end{bmatrix}$$

which is the same solution as Example 6.1. ■

7.5 Modal Identification

Although many systems are inherently nonlinear, in some instances they may respond to well-tuned linear controls. In order to implement linear feedback control, the system designer must have an accurate model of sufficiently low

order from which to design the control. Several approaches to developing such lower-order models have included dynamic equivalencing, eigenanalysis, and pole/zero cancellation. Frequently, however, the original system is too complex or the parameters are not known with enough accuracy to produce an adequate reduced-order model. In practice, the system may have parameters that drift with time or operating condition, which compromises the accuracy of the mathematical model. In these cases, it is desirable to extract the modal information directly from the system response to a perturbation. Using this approach, it may be possible to replace the actual dynamic model with an estimated linear model that is derived from the system output waveform. The time-varying dynamic response of a power system to a disturbance may be composed of numerous modes that must be identified. Several methods have been proposed to extract the pertinent modal information from time-varying responses. An appropriate method must consider the inclusion of nonlinearities, the size of the model that can be effectively utilized, and the reliability of the results.

Methods that are applied directly to the nonlinear system simulation or field measurements include the effects of nonlinearities. In full-state eigenvalue analysis, the size of the system model is typically limited to several hundred states with present computing capabilities. This means that a typical system containing several thousand nodes must be reduced using dynamic equivalencing. Modal analysis techniques that operate directly on system output are not limited by system size. This means that standard time-domain-analysis results are directly usable. This eliminates the possibility of losing some of system modal content due to reduction. The estimated linear model may then be used for control design applications or other linear analysis techniques. The estimated model may be chosen to be of lower order than the original model, but still retain the dominant modal characteristics.

This problem may be posed such that, given a set of measurements that vary with time, it is desired to fit a time-varying waveform of prespecified form to the actual waveform (i.e., minimize the error between the actual measured waveform and the proposed waveform). The coefficients of the prespecified waveform yield the dominant modal characteristics of the underlying linear system. Consider the following linear system:

$$\dot{x}(t) = Ax(t) \quad x\left(t_0\right) = x_0 \tag{7.46}$$

where

$$x_i(t) = \sum_{k=1}^{n} a_k e^{(b_k t)} \cos\left(\omega_k t + \theta_k\right) \tag{7.47}$$

is one of the n states. The parameters a_k and θ_k are derived from the influence of the initial conditions, whereas the parameters b_k and ω_k are derived from the eigenvalues of A. The estimation of these responses yields modal information about the system that can be used to predict possible unstable

behavior, controller design, parametric summaries for damping studies, and modal interaction information.

Any time-varying function can be fit to a series of complex exponential functions over a finite time interval. However, it is not practical to include a large number of terms in the fitting function. The problem then becomes one of minimizing the error between the actual time-varying function and the proposed function by estimating the magnitude, phase, and damping parameters of the fitting function. In the problem of estimating a nonlinear waveform by a series of functions, the minimization function is given by

$$\min f = \sum_{i=1}^{N} \left[\sum_{k=1}^{n} \left[a_k e^{(b_k t_i)} \cos\left(\omega_k t_i + \theta_k\right) \right] - y_i \right]^2 \tag{7.48}$$

where n is the number of desired modes of the approximating waveform, N is the number of data samples, y_i is the sampled waveform, and

$$[a_1\ b_1\ \omega_1\ \theta_1,\ \ldots, a_n\ b_n\ \omega_n\ \theta_n]^T$$

are the parameters to be estimated.

There are several approaches to estimating the modal content of a time-varying waveform. The Prony method is well known and widely used in power systems applications. The matrix pencil approach was introduced for extracting poles from antennas' electromagnetic transient responses. The Levenberg–Marquardt iteratively updates the modal parameters by an analytic optimization to minimize the error between the resulting waveform and the input data.

7.5.1 Prony Method

One approach to estimating the various parameters is the *Prony method* [23]. This method is designed to directly estimate the parameters for the exponential terms by fitting the function

$$\hat{y}(t) = \sum_{i=1}^{n} A_i e^{\sigma_i t} \cos\left(\omega_i t + \phi_i\right) \tag{7.49}$$

to an observed measurement for $y(t)$, where $y(t)$ consists of N samples

$$y\left(t_k\right) = y(k), \quad k = 0, 1, \ldots, N-1$$

that are evenly spaced by a time interval Δt. Since the measurement signal $y(t)$ may contain noise or dc offset, it may have to be conditioned before the fitting process in applied.

Note that Equation (7.49) can be recast in complex exponential form as

$$\hat{y}(t) = \sum_{i=1}^{n} B_i e^{\lambda_i t} \tag{7.50}$$

which can be translated to

$$\hat{y}(k) = \sum_{i=1}^{n} B_i z_i^k \tag{7.51}$$

where

$$z_i = e^{(\lambda_i \Delta t)} \tag{7.52}$$

The system eigenvalues λ can be found from the discrete modes by

$$\lambda_i = \frac{ln\,(z_i)}{\Delta t} \tag{7.53}$$

The z_i are the roots of the nth order polynomial

$$z^n - \left(a_1 z^{n-1} + a_2 z^{n-2} + \ldots + a_n z^0\right) = 0 \tag{7.54}$$

where the a_i coefficients are unknown and must be calculated from the measurement vector.

The basic Prony method is summarized as

Prony Method

1. Assemble selected elements of the record into a Toeplitz data matrix.
2. Fit the data with a discrete linear prediction model, such as a least squares solution.
3. Find the roots of the characteristic polynomial (7.54) associated with the model of Step 1.
4. Using the roots of Step 3 as the complex modal frequencies for the signal, determine the amplitude and initial phase for each mode.

These steps are performed in the z-domain, translating the eigenvalues to the s-domain as a final step.

The approach to the Toeplitz (or the closely related Hankel) matrix assembly of Step 1 has received the most attention in the literature. The problem can be formulated in many different ways. If the initial (i.e., $i < 0$) and post (i.e., $i > N$) conditions are assumed to be zero, then the subscript ranges X_1 through X_4 in Figure 7.3 represent four such formulations. The range X_1 is termed the covariance problem. Because the initial and post conditions are not used, no assumption is required on their value. The range X_4 is called the correlation problem; it incorporates both initial and post conditions. The remaining problems, X_2 and X_3, are termed the prewindowed and postwindowed methods.

In the majority of practical cases, the Toeplitz matrix is nonsquare with more rows than columns. A Toeplitz matrix is a matrix with a constant diagonal in which each descending diagonal from left to right is constant.

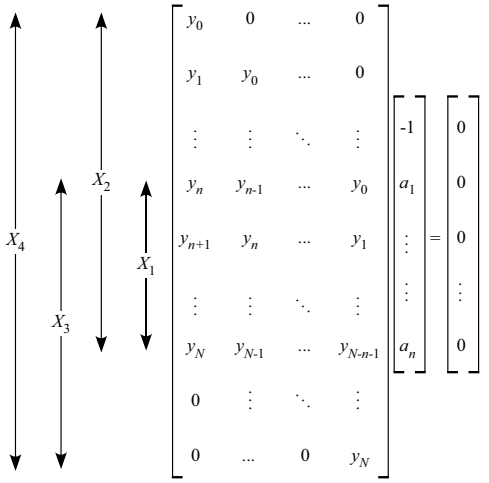

FIGURE 7.3
Toeplitz matrix

The system of equations requires a least-squares solution to find the factors a_1 through a_n. After the a_i coefficients are obtained, the n roots z_i of the polynomial in Equation (7.52) can be found by factoring.

Once z_i has been computed from the roots of Equation (7.54), then the eigenvalues λ_i can be calculated from Equation (7.53). The next step is to find the B_i that produces $\hat{y}(k) = y(k)$ for all k. This leads to the following relationship:

$$\begin{bmatrix} z_1^0 & z_2^0 & \dots & z_n^0 \\ z_1^1 & z_2^1 & \dots & z_n^1 \\ \vdots & \vdots & \vdots & \vdots \\ z_1^{N-1} & z_2^{N-1} & \dots & z_n^{N-1} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix} \tag{7.55}$$

which can be succinctly expressed as

$$ZB = Y \tag{7.56}$$

Note that the matrix B is $n \times N$; therefore, Equation (7.56) must also be solved by the least-squares method. The estimating waveform $\hat{y}(t)$ is then calculated from Equation (7.50). The reconstructed signal $\hat{y}(t)$ will usually not fit $y(t)$ exactly. An appropriate measure for the quality of this fit is a "signal to noise ratio (SNR)" given by

$$\text{SNR} = 20 \log \frac{\|\hat{y} - y\|}{\|y\|} \tag{7.57}$$

where the SNR is given in decibels (dB).

Since the fit for this method may be inexact, it is desirable to have control over the level of error between the fitting function and the original waveform. In this case, a nonlinear least squares can provide improved results.

7.5.2 The Matrix Pencil Method

The Prony method described in the previous section is a "polynomial" method in that it includes the process of finding the poles z_i of a characteristic polynomial. The matrix pencil (MP) method produces a matrix whose roots provide z_i. The poles are found as the solution of a generalized eigenvalue problem [24], [46]. The matrix pencil is given by

$$[Y_2] - \lambda [Y_1] = [Z_1] [B] \{[Z_0] - \lambda [I]\} [Z_2] \tag{7.58}$$

where

$$[Y] = \begin{bmatrix} y(0) & y(1) & \dots & y(L) \\ y(1) & y(2) & \dots & y(L+1) \\ \vdots & \vdots & & \vdots \\ y(N-L) & y(N-L+1) & \dots & y(N) \end{bmatrix} \tag{7.59}$$

$$[Z_0] = \operatorname{diag} [z_1, z_2, \dots, z_n] \tag{7.60}$$

$$[Z_1] = \begin{bmatrix} 1 & 1 & \dots & 1 \\ z_1 & z_2 & \dots & z_n \\ \vdots & \vdots & & \vdots \\ z_1^{(N-L-1)} & z_2^{(N-L-2)} & \dots & Z_n^{(N-L-1)} \end{bmatrix} \tag{7.61}$$

$$[Z_2] = \begin{bmatrix} 1 & z_1 & \dots & z_1^{L-1} \\ 1 & z_2 & \dots & z_2^{L-1} \\ \vdots & \vdots & & \vdots \\ 1 & Z_n & \dots & Z_n^{l-1} \end{bmatrix} \tag{7.62}$$

$[B]$ = matrix of residuals

$[I]$ = $n \times n$ identity matrix

n = desired number of eigenvalues

L = pencil parameter, such that $n \le L \le N - n$

Matrix Pencil Method

1. Choose L such that $n \le L \le N - n$.

2. Construct the matrix $[Y]$.

3. Perform a singular value decomposition of $[Y]$ to obtain

$$[Y] = [U][S][V]^T \tag{7.63}$$

where $[U]$ and $[V]$ are unitary matrices and contain the eigenvectors of $[Y][Y]^T$ and $[Y]^T[Y]$, respectively.

4. Construct the matrices $[V_1]$ and $[V_2]$ such that

$$V_1 = [v_1 \;\; v_2 \;\; v_3 \;\; \ldots \;\; v_{n-1}] \tag{7.64}$$
$$V_2 = [v_2 \;\; v_3 \;\; v_4 \;\; \ldots \;\; v_n] \tag{7.65}$$

where v_i is the ith right singular vector of V.

5. Construct $[Y_1]$ and $[Y_2]$:

$$[Y_1] = [V_1]^T[V_1]$$
$$[Y_2] = [V_2]^T[V_1]$$

6. The desired poles z_i may be found as the generalized eigenvalues of the matrix pair $\{[Y_2]\,;[Y_1]\}$.

From this point, the remainder of the algorithm follows that of the Prony method to calculate the eigenvalues λ and the residual matrix B.

If the pencil parameter L is chosen such that $L = N/2$, then the performance of the method is very close to the optimal bound [24].

It has been shown that, under noise, the statistical variance of the poles found from the matrix pencil method is always less than that of the Prony method [24].

7.5.3 The Levenberg–Marquardt Method

The nonlinear least squares for data fitting applications has the general form

$$\text{minimize } f(x) = \sum_{k=1}^{N} [\hat{y}(x, t_i) - y_i]^2 \tag{7.66}$$

where y_i is the output of the system at time t_i, and x is the vector of magnitudes, phases, and damping coefficients of Equation (7.49), which arise from the eigenvalues of the state matrix of the system.

To find the minimum of $f(x)$, the same procedure for developing the Newton–Raphson iteration is applied. The function $f(x)$ is expanded about some x_0 by the Taylor series

$$f(x) \approx f(x_0) + (x - x_0)^T f'(x_0) + \frac{1}{2}(x - x_0)^T f''(x_0)(x - x_0) + \ldots \tag{7.67}$$

where

$$f'(x) = \frac{\partial f}{\partial x_j} \text{ for } j = 1, \ldots, n$$
$$f''(x) = \frac{\partial^2 f}{\partial x_j \partial x_k} \text{ for } j, k = 1, \ldots, n$$

If the higher-order terms in the Taylor expansion are neglected, then minimizing the quadratic function on the right-hand side of Equation (7.67) yields

$$x_1 = x_0 - \left[f''(x_0)\right]^{-1} f'(x_0) \tag{7.68}$$

which yields an approximation for the minimum of the function $f(x)$. This is also one Newton–Raphson iteration update for solving the necessary minimization condition

$$f'(x) = 0$$

The Newton–Raphson Equation (7.68) may be rewritten as the iterative linear system

$$A(x_k)(x_{k+1} - x_k) = g(x_k) \tag{7.69}$$

where

$$g_j(x) = -\frac{\partial f}{\partial x_j}(x)$$
$$a_{jk}(x) = \frac{\partial^2 f}{\partial x_j \partial x_k}(x)$$

and the matrix A is the system Jacobian (or similar iterative matrix).

The derivatives of Equation (7.66) are

$$\frac{\partial f}{\partial x_j}(x) = 2\sum_{k=1}^{N} [\hat{y}_k - y_k] \frac{\partial \hat{y}_i}{\partial x_j}(x)$$

and

$$\frac{\partial^2 f}{\partial x_j \partial x_k}(x) = 2\sum_{k=1}^{N} \left\{ \frac{\partial \hat{y}_k}{\partial x_j}(x) \frac{\partial \hat{y}_i}{\partial x_k}(x) + [\hat{y}_k - y_k] \frac{\partial^2 \hat{y}_k}{\partial x_j \partial x_k}(x) \right\}$$

In this case, the matrix element a_{jk} contains second derivatives of the functions $\hat{y}_i$. These derivatives are multiplied by the factor $[\hat{y}_i(x) - y_i]$ and will become small during the minimization of f. Therefore, the argument can be made that these terms can be neglected during the minimization process. Note that, if the method converges, it will converge regardless of whether the exact Jacobian is used in the iteration. Therefore, the iterative matrix A can be simplified as

$$a_{jk} = 2\sum_{i=1}^{N} \frac{\partial \hat{y}_i}{\partial x_j}(x) \frac{\partial \hat{y}_i}{\partial x_k}(x) \tag{7.70}$$

and note that $a_{jj}(x) > 0$.

The Levenberg–Marquardt method modifies Equation (7.69) by introducing the matrix $\hat{A}$ with entries

$$\hat{a}_{jj} = (1+\gamma)\, a_{jj}$$
$$\hat{a}_{jk} = a_{jk} \quad j \neq k$$

where γ is some positive parameter. Equation (7.69) becomes

$$\hat{A}(x_0)(x_1 - x_0) = g \tag{7.71}$$

For large γ, the matrix $\hat{A}$ will become diagonally dominant. As γ approaches zero, Equation (7.71) will turn into the Newton–Raphson method. The Levenberg–Marquardt method has the basic feature of varying γ to select the optimal characteristics of the iteration. The basic Levenberg–Marquardt algorithm is summarized:

Levenberg–Marquardt Method

1. Set $k = 0$. Choose an initial guess x_0, γ, and a factor α.

2. Solve the linear system of Equation (7.71) to obtain x_{k+1}.

3. If $f(x_{k+1}) > f(x_k)$, reject x_{k+1} as the new approximation, replace γ by $\alpha\gamma$, and repeat Step 2.

4. If $f(x_{k+1}) < f(x_k)$, accept x_{k+1} as the new approximation, replace γ by γ/α, set $k = k + 1$, and repeat Step 2.

5. Terminate the iteration when

$$\|x_{k+1} - x_k\| < \varepsilon$$

In the problem of estimating a nonlinear waveform by a series of functions, the minimization function is given by

$$\text{minimize} f = \sum_{i=1}^{N} \left[\sum_{k=1}^{m} \left[a_k e^{(b_k t_i)} \cos\left(\omega_k t_i + \theta_k\right) \right] - y_i \right]^2 \tag{7.72}$$

where m is the number of desired modes of the approximating waveform, and $x = [a_1\ b_1\ \omega_1\ \theta_1\ \ldots a_m\ b_m\ \omega_m\ \theta_m]^T$.

As with all of the nonlinear iterative methods, the ability of the Levenberg–Marquardt method to converge to a solution depends on the choice of initial guess. In this case, it is wise to use the results of the matrix pencil or the Prony method to provide the initial values.

7.5.4 Eigensystem Realization Algorithm

The Eigensystem Realization Algorithm (ERA) is based on the singular value decomposition of the Hankel matrix H_0 associated with the linear ringdown of the system. A Hankel matrix is a square matrix with constant skew-diagonals. The Hankel matrices are typically assembled using all of the available data such that the top left-most element of H_0 is y_0 and the bottom right-most element of H_1 is y_N. The Hankel matrices are assembled such that

$$H_0 = \begin{bmatrix} y_0 & y_1 & \cdots & y_r \\ y_1 & y_2 & \cdots & y_{r+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_r & y_{r+1} & \cdots & y_{N-1} \end{bmatrix} \tag{7.73}$$

$$H_1 = \begin{bmatrix} y_1 & y_2 & \cdots & y_{r+1} \\ y_2 & y_3 & \cdots & y_{r+2} \\ \vdots & \vdots & \ddots & \vdots \\ y_{r+1} & y_{r+2} & \cdots & y_N \end{bmatrix} \tag{7.74}$$

where $r = \frac{N}{2} - 1$. This choice of r assumes that the number of data points is sufficient such that $r > n$.

The ERA formulation begins by separating the singular value decomposition of H_0 into two components according to the relative size of the singular values:

$$H_0 = U\Sigma V^T = [U_n \;\; U_z] \begin{bmatrix} \Sigma_n & 0 \\ 0 & \Sigma_z \end{bmatrix} \begin{bmatrix} V_n^T \\ V_z^T \end{bmatrix} \tag{7.75}$$

where Σ_n and Σ_z are diagonal matrices with their elements ordered by magnitude:

$$\Sigma_n = \text{diag} \;\; (\sigma_1, \sigma_2, \ldots, \sigma_n) \tag{7.76}$$

$$\Sigma_z = \text{diag} \;\; (\sigma_{n+1}, \sigma_{n+2}, \ldots, \sigma_N) \tag{7.77}$$

and the singular values are ordered by magnitude such that

$$\sigma_1 > \sigma_2 > \ldots > \sigma_n > \sigma_{n+1} > \sigma_{n+2} > \ldots > \sigma_N$$

The SVD is a useful tool for determining an appropriate value for n. The ratio of the singular values contained in Σ can determine the best approximation of n. The ratio of each singular value σ_i to the largest singular value $\sigma_{\max}$ is compared to a threshold value, where p is the number of significant decimal digits in the data:

$$\frac{\sigma_i}{\sigma_{\max}} \approx 10^{-p}$$

If p is set to 3, then any singular values with a ratio below 10^{-3} are assumed to be part of the noise and are not included in the reconstruction of the system. The value of n should be set to the number of singular values with a ratio

above the threshold 10^{-p}. It can be shown that, for a linear system of order n, the diagonal elements of Σ_z are zero (assuming that the impulse response is free of noise). The practical significance of this result is that the relative size of the singular values provides an indication of the identified system order. If the singular values exhibit a significant grouping such that $\sigma_n \geq \sigma_{n+1}$, then, from the partitioned representation, H_0 can be approximated by

$$H_0 \approx U_n \Sigma_n V_n^T \tag{7.78}$$

The method for obtaining the eigenvalue realization algorithm solution can be summarized as follows:

1. Assemble selected elements of the record into Hankel data matrices H_0 and H_1.

2. Perform the singular value decomposition of H_0 and estimate the system order n based on the magnitude of the singular values.

3. Computer the discrete system matrices as follows:

$$A = \Sigma_n^{-\frac{1}{2}} U_n^T H_1 V_n \Sigma_n^{-\frac{1}{2}} \tag{7.79}$$

$$B = \Sigma_n^{-\frac{1}{2}} V_n^T (1:n, 1:n) \tag{7.80}$$

$$C = U_n (1:N, 1:n) \Sigma_n^{-\frac{1}{2}} \tag{7.81}$$

$$D = y_0 \tag{7.82}$$

4. Calculate continuous system matrices A_c, B_c assuming a zero order hold and sampling interval Δt:

$$A_c = \ln\left(\frac{A}{\Delta t}\right) \tag{7.83}$$

$$B_c = \left[\int_0^{\Delta t} e^{A\tau} d\tau\right]^{-1} B \tag{7.84}$$

The reduced system response can then be computed from the continuous matrices.

7.5.5 Examples

The effectiveness of these methods will be illustrated with several examples ranging from a simple three-mode linear system to an actual power system oscillation.

Simple Example

The application of the methods will initially consider the waveform shown in

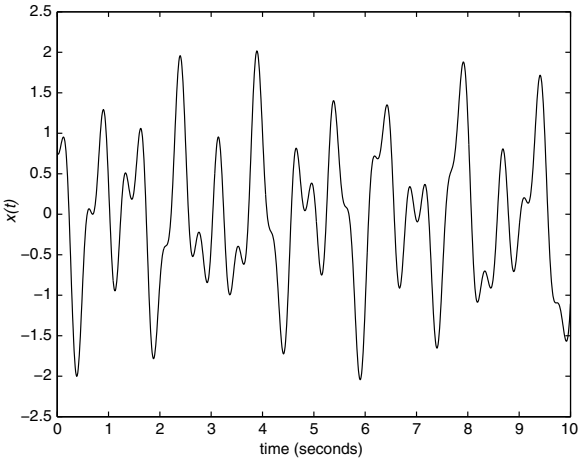

FIGURE 7.4
Three-mode waveform

Figure 7.4, which is generated from

$$x(t) = \sum_{i=1}^{3} a_i e^{b_i t} (\cos \omega_i t + \theta_i)$$

where

mode	a_i	b_i	ω_i	θ_i
1	1.0	–0.01	8.0	0.0
2	0.6	–0.03	17.0	π
3	0.5	0.04	4.7	$\pi/4$

Each of the methods described previously is used to estimate the signal parameters and to reconstruct the waveform.

Prony

mode	a_i	b_i	ω_i	θ_i
1	0.9927	–0.0098	7.9874	0.0617
2	0.6009	–0.0304	17.0000	3.1402
3	0.5511	0.0217	4.6600	0.9969

Matrix Pencil

mode	a_i	b_i	ω_i	θ_i
1	1.0121	–0.0130	8.0000	0.0008
2	0.6162	–0.0361	16.9988	3.1449
3	0.5092	0.0364	4.6953	0.7989

Levenberg–Marquardt

mode	a_i	b_i	ω_i	θ_i
1	1.0028	–0.0110	7.9998	0.0014
2	0.6010	–0.0305	16.9994	3.1426
3	0.5051	0.0378	4.6967	0.7989

The reconstruction error in each waveform is measured.

$$\text{error } = \sum_{i=1}^{N} \left[\sum_{k=1}^{m} \left[a_k e^{(b_k t_i)} \cos\left(\omega_k t_i + \theta_k\right) \right] - y_i \right]^2 \tag{7.85}$$

and the errors for each method are

Method	error
Matrix pencil	0.1411
Levenberg–Marquardt	0.0373
Prony	3.9749

Not surprisingly, the Levenberg–Marquardt yielded the best results since it is an iterative method, whereas the other estimation methods are linear noniterative methods.

Power System Example

In this example, the accuracy of the methods will be compared using the dynamic response of a time-domain simulation of a large Midwestern utility system, shown in Figure 7.5. This simulation contains several hundred states comprising a wide range of responses. The number of dominant modes is not known. The results of a fast Fourier transform (FFT) are shown in Figure 7.6. From this figure, it appears as if there are five dominant modes that contribute significantly to the original waveform, with several of the modes concentrated at low frequencies. Therefore, the estimation methods introduced earlier will be applied to extract five modes.

Extracting five modes, the results are shown in Figure 7.7 and summarized:

Prony

mode	a_i	b_i	ω_i	θ_i
1	1.7406	–0.5020	3.7835	–1.4870
2	1.5723	–0.1143	4.8723	–1.1219
3	1.0504	–0.0156	6.2899	–0.0331
4	2.1710	–0.2455	7.7078	2.2011
5	0.9488	–0.3515	8.3854	–1.6184

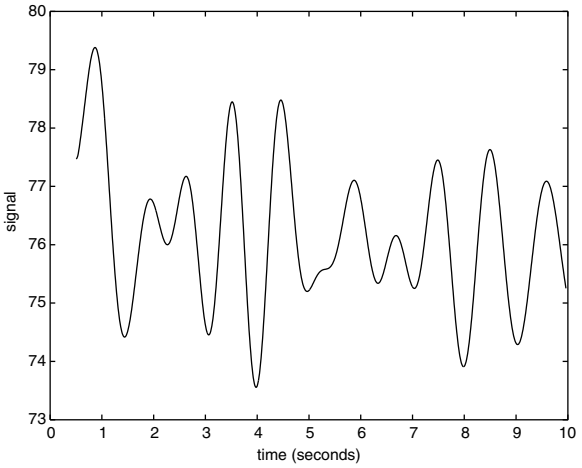

FIGURE 7.5
PSS/E waveform

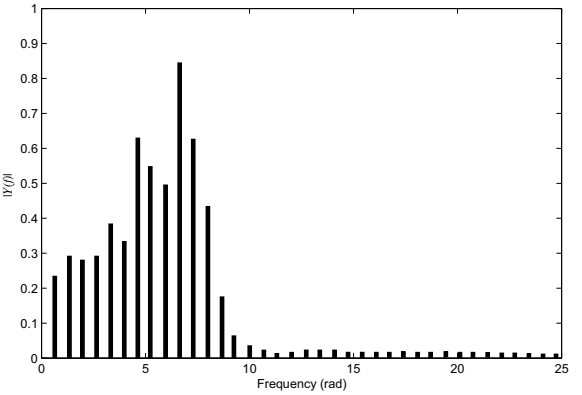

FIGURE 7.6
FFT of PSS/E waveform

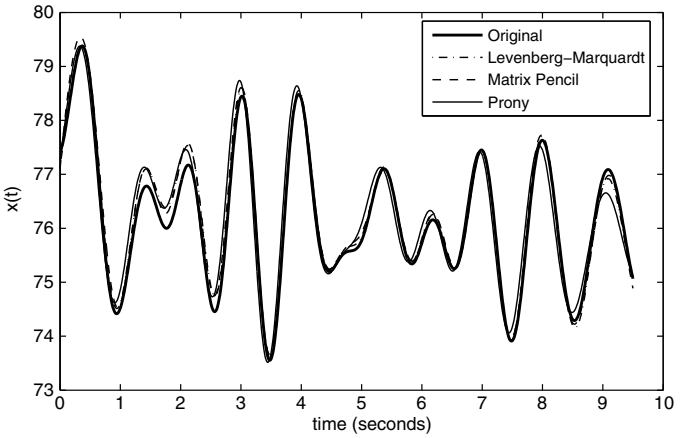

FIGURE 7.7
Reconstruction of the PSS/E waveform using various methods

Matrix Pencil

mode	a_i	b_i	ω_i	θ_i
1	2.0317	–0.5610	3.7357	–1.5158
2	1.3204	–0.0774	4.8860	–1.3607
3	0.7035	0.0527	6.3030	–0.3093
4	1.2935	–0.2400	7.4175	2.9957
5	0.6718	–0.0826	8.0117	0.3790

Levenberg–Marquardt

mode	a_i	b_i	ω_i	θ_i
1	1.8604	–0.5297	3.6774	–1.3042
2	1.1953	–0.0578	4.8771	–1.3405
3	0.8164	0.0242	6.2904	–0.2537
4	1.9255	–0.2285	7.6294	2.1993
5	0.6527	–0.2114	8.3617	–1.5187

The error in each method as determined by Equation (7.85) and the relative computation times are

method	CPU	Error
Prony	0.068	228.16
Matrix pencil	39.98	74.81
Levenberg–Marquardt	42.82*	59.74

* Depends on initial condition

Note that the Prony method is the most computationally efficient since it only requires two least-squares solutions. The matrix pencil method is more computationally expensive since it requires a singular value decomposition of a relatively large matrix. It also requires an eigensolution, but, since the matrix itself is relatively small (the size of the number of required modes), it is not overly burdensome. Not surprisingly, the Levenberg–Marquardt method is the most computationally expensive since it is an iterative method. Its computational burden is directly related to the initial guess: the better the initial guess, the faster the method converges. It is wise to choose the initial guess as the parameters obtained from either the Prony or the matrix pencil methods.

Similarly, the level of error in each method varies with the complexity of the method. The Levenberg–Marquardt method yields the best results, but with the greatest computational effort. The Prony has the largest error, but this is offset by the relative speed of computation.

7.6 Power System Applications

7.6.1 Participation Factors

In the analysis of large-scale power systems, it is sometimes desirable to have a measure of the impact that a particular state has on a selected system mode (or eigenvalue). In some cases, it is desirable to know whether a set of physical states has influence over an oscillatory mode such that control of that component may mitigate the oscillations. Another use is to identify which system components contribute to an unstable mode. One tool for identifying which states significantly participate in a selected mode is the method of *participation factors* [62]. In large-scale power systems, participation factors can also be used to identify interarea oscillations versus those that persist only within localized regions (intraarea oscillations).

Participation factors provide a measure of the influence each dynamic state has on a given mode or eigenvalue. Consider a linear system

$$\dot{x} = Ax \tag{7.86}$$

The participation factor p_{ki} is a sensitivity measure of the ith eigenvalue to the (k, k) diagonal entry of the system A matrix. This is defined as

$$p_{ki} \frac{\partial \lambda_i}{\partial a_{kk}} \tag{7.87}$$

where λ_i is the ith eigenvalue and a_{kk} is the kth diagonal entry of A. The participation factor p_{ki} relates the kth state variable to the ith eigenvalue. An

equivalent, but more common expression for the participation factor is also defined as

$$p_{ki} = \frac{w_{ki} v_{ik}}{w_i^T v_i} \tag{7.88}$$

where w_{ki} and v_{ki} are the kth entries of the left and right eigenvectors associated with λ_i. As with eigenvectors, participation factors are frequently normalized to unity, such that

$$\sum_{k=1}^{n} p_{ki} = 1 \tag{7.89}$$

When the participation factors are normalized, they provide a straightforward measure of the percent of impact each state has on a particular mode. Participation factors for complex eigenvalues (and eigenvectors) are defined in terms of magnitudes, rather than complex quantities. In the case of complex eigenvalues, the participation factors are defined as

$$p_{ki} = \frac{|v_{ik}||w_{ki}|}{\sum_{i=1}^{n} |v_{ik}||w_{ki}|} \tag{7.90}$$

In some applications, it may be preferred to retain the complex nature of the participation factors to yield both phase and magnitude information [32].

7.7 Problems

1. Find the eigenvalues and eigenvectors of the following matrices.

$$A_1 = \begin{bmatrix} 5 & 4 & 1 & 1 \\ 4 & 5 & 1 & 1 \\ 1 & 1 & 4 & 2 \\ 1 & 1 & 2 & 4 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 2 & 3 & 4 \\ 7 & -1 & 3 \\ 1 & -1 & 5 \end{bmatrix}$$

2. Find the complex eigenvalues of the follow matrix method.

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

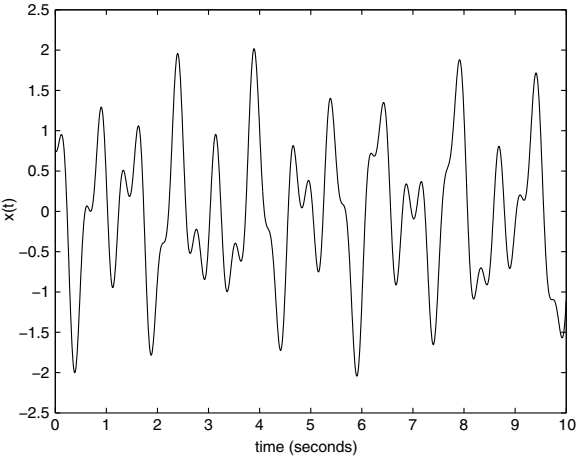

FIGURE 7.8
Waveform for Problem 5

3. Using the implicitly restarted Arnoldi method, find the two largest eigenvalues and the corresponding eigenvectors of

$$A = \begin{bmatrix} 8 & 3 & 4 & 4 & 10 \\ 3 & 6 & 6 & 5 & 4 \\ 4 & 6 & 10 & 7 & 8 \\ 4 & 5 & 7 & 10 & 4 \\ 10 & 4 & 8 & 4 & 2 \end{bmatrix}$$

4. Find the singular values and singular vectors of the matrix:

$$A0 = \begin{bmatrix} 13 & 4 & 7 & 6 & 20 \\ 6 & 3 & 20 & 9 & 7 \\ 9 & 8 & 19 & 11 & 15 \\ 1 & 4 & 2 & 19 & 14 \end{bmatrix}$$

5. Generate the waveform shown in Figure 7.8 on the interval $t \in [0, 10]$ with a time step of 0.01 seconds.

$$x(t) = \sum_{i=1}^{3} a_i e^{b_i t} (\cos c_i t + d_i)$$

where

mode	a_i	b_i	c_i	d_i
1	1.0	–0.01	8.0	0.0
2	0.6	–0.03	17.0	π
3	0.5	0.04	4.7	$\pi/4$

(a) Using 100 equidistant points on the interval $[0, 10]$, estimate the six system eigenvalues using Prony analysis. How do these compare with the actual eigenvalues?

(b) Using 100 equidistant points on the interval $[0, 10]$, estimate the six system eigenvalues using Levenberg–Marquardt. How do these eigenvalues compare with the actual eigenvalues? With those obtained from the Prony analysis?

(c) Using 100 equidistant points on the interval $[0, 10]$, estimate the six system eigenvalues using the matrix pencil method. How do these eigenvalues compare with the actual eigenvalues? With those obtained from the Prony analysis?

(d) Using all of the points, estimate the six system eigenvalues using Prony analysis. How do these compare with the actual eigenvalues?

(e) Using all of the points, estimate the six system eigenvalues using Levenberg–Marquardt. How do these compare with the actual eigenvalues?

(f) Using all of the points, estimate the six system eigenvalues using the matrix pencil method. How do these compare with the actual eigenvalues?

(g) Using all of the points, estimate the two dominant modes (two complex eigenvalue pairs) of the system response using the matrix pencil method. Substitute the estimated parameters into

$$x(t) = \sum_{i=1}^{2} a_i e^{b_i t} (\cos c_i t + d_i)$$

and plot this response versus the three-mode response. Discuss the differences and similarities.