CHAPTER 4

CANONICAL FORMS OBTAINED VIA ORTHOGONAL TRANSFORMATIONS

Topics covered

- Numerical Instabilities in obtaining the Jordan and Companion Matrices
- Hessenberg Reduction of a Matrix
- The Double-Shift Implicit QR Iteration for the Real Schur Form (RSF) of a Matrix
- Invariant Subspace Computation from the RSF
- The QZ Algorithm for the Generalized RSF of the Matrix Pencil $A - \lambda B$
- The SVD Computation

4.1 IMPORTANCE AND SIGNIFICANCE OF USING ORTHOGONAL TRANSFORMATIONS

The Jordan and companion matrices have special structures that can be conveniently exploited to solve many control problems. **Unfortunately, however, these forms in general, cannot be obtained in a numerically stable way.**

We examine this fact here in some detail below.

Suppose that X is a nonsingular matrix and consider the computation of $X^{-1}AX$ in floating point arithmetic. It can be shown that

$$fl(X^{-1}AX) = X^{-1}AX + E,$$

where $\|E\|_2 \approx \mu \text{Cond}(X)\|A\|_2$, μ is the machine precision.

Thus, when X is ill-conditioned, there will be large errors in computing $X^{-1}AX$.

For the Jordan canonical form (JCF), the transforming matrix X is highly ill-conditioned, whenever A has defective or nearly defective eigenvalue.

The reduction of a matrix A to an upper companion matrix C (**Section 2.4.5**) involves the following steps:

Step 1. A is transformed to an upper Hessenberg matrix $H_u = (h_{ij})$ by orthogonal similarity: $P^{\mathrm{T}}AP = H_u$.

Step 2. Assuming that H_u is unreduced, that is, $h_{i+1,i} \neq 0$, $i = 1, 2, \ldots, n-1$, then H_u is further reduced to the companion matrix C by similarity. Thus, if $Y = (e_1, H_u e_1, \ldots, H_u^{n-1} e_1)$, it is easy to see that $Y^{-1} H_u Y = C$.

A numerically stable algorithm to implement Step 1 is given in the next section; however, the matrix Y in Step 2 **can be highly ill-conditioned if H_u has small subdiagonal entries.**

(Note that Y is a lower triangular matrix with $1, h_{21}h_{32}, \ldots, h_{21}h_{32}\ldots h_{n,n-1}$ as the diagonal entries).

Thus, Step 2, in general, cannot be implemented in a numerically effective manner.

The above discussions clearly show that it is important from a numerical computation viewpoint to have canonical forms which can be achieved using only well-conditioned transforming matrices, such as orthogonal matrices.

Indeed, **if a matrix A is transformed to a matrix B using an orthogonal similarity transformation, then a perturbation in A will result in a perturbation in B of the same magnitude.** That is, if

$$B = U^{\mathrm{T}}AU \quad \text{and} \quad U^{\mathrm{T}}(A + \Delta A)U = B + \Delta B,$$

then $\|\Delta B\|_2 \approx \|\Delta A\|_2$.

In this chapter, we show that two very important canonical forms: the **Hessenberg** form and the **Real Schur Form** (RSF) of a matrix A, can be obtained using orthogonal similarity transformations. (Another important canonical form, known as the **generalized real Schur form**, can be obtained using orthogonal equivalence.)

We will see in the rest of the book that **these canonical forms form important tools in the development of numerically effective algorithms for control problems.**

Applications of **Hessenberg** and **real Schur** forms include:

1. Computation of frequency response matrix (**Chapter 5**)
2. Solutions of Lyapunov and Sylvester equations (**Chapter 8**), Algebraic Riccati equations (**Chapter 13**), Sylvester-observer equation (**Chapter 12**).
3. Solutions of eigenvalue assignment (**Chapter 11**), feedback stabilization problems (**Chapter 10**), stability and inertia computations (**Chapter 7**).

Applications of generalized real Schur form include:

1. Solutions of certain algebraic Riccati equations **(Chapter 13)**.
2. Solution of any descriptor control problem.
3. Computations of frequencies and modes of vibrating systems.

Besides these two forms, there are two other important canonical forms, namely, the **controller-Hessenberg** and **observer-Hessenberg** forms. These forms can also be obtained in a numerically effective way and will be used throughout the book. Methods for obtaining these two forms are described in **Chapter 6.**

4.2 HESSENBERG REDUCTION OF A MATRIX

Recall that a matrix $H = (h_{ij})$ is said to be an **upper Hessenberg** matrix if $h_{ij} = 0$ for $i > j + 1$.

An $n \times n$ matrix A can always be transformed to an upper Hessenberg matrix H_u by orthogonal similarity. That is, **given an $n \times n$ matrix A, there exists an orthogonal matrix P such that $PAP^{\mathrm{T}} = H_u$.**

Again, Householder and Givens matrices, being orthogonal, can be employed to obtain H_u from A.

We will discuss only Householder's method here.

Reduction to Hessenberg Form using Householder Matrices

The idea is to extend the QR factorization process using Householder matrices described in Chapter 3 to obtain P and H_u, such that $PAP^{\mathrm{T}} = H_u$ is an upper Hessenberg matrix and P is orthogonal.

The matrix P is constructed as the product of $(n-2)$ Householder matrices P_1 through P_{n-2}. The matrix P_1 is constructed to create zeros in the first column of A below the entry $(2, 1)$; P_2 is constructed to create zeros below the entry $(3, 2)$ of the second column of the matrix $P_1AP_1^{\mathrm{T}}$, and so on.

The process consists of $(n-2)$ steps. (Note that an $n \times n$ Hessenberg matrix contains at least $(n-2)(n-1)/2$ zeros.)

At the end of $(n-2)$th step, the matrix $A^{(n-2)}$ is an upper Hessenberg matrix H_u. The Hessenberg matrix H_u is orthogonally similar to A. This is seen as follows:

$$\begin{aligned} H_u = A^{(n-2)} &= P_{n-2}A^{(n-3)}P_{n-2}^{\mathrm{T}} = P_{n-2}(P_{n-3}A^{(n-4)}P_{n-3}^{\mathrm{T}})P_{n-2}^{\mathrm{T}} \\ &= \cdots = (P_{n-2}P_{n-3}\ldots P_1)A(P_1^{\mathrm{T}}P_2^{\mathrm{T}}\ldots P_{n-3}^{\mathrm{T}}P_{n-2}^{\mathrm{T}}). \end{aligned} \tag{4.2.1}$$

Set

$$P = P_{n-2}P_{n-3}\ldots P_1. \tag{4.2.2}$$

We then have $H_u = PAP^{\mathrm{T}}$. Since each Householder matrix P_i is orthogonal, the matrix P which is the product of $(n-2)$ Householder matrices, is also orthogonal.

For $n = 4$, schematically, we can represent the reduction as follow. Set $A^{(0)} \equiv A$. Then,

$$A \xrightarrow{P_1} P_1 A P_1^{\mathrm{T}} = \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix} = A^{(1)}.$$

$$A^{(1)} \xrightarrow{P_2} P_2 A^{(1)} P_2^{\mathrm{T}} = \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \end{pmatrix} = A^{(2)} = H_u.$$

Notes

1. Multiplication by P_i^{T} to the right does not destroy the zeros already present in $P_i A^{(i-1)}$.
2. The product $P_i A^{(i-1)} P_i^{\mathrm{T}}$ can be **implicitly** formed as shown in Chapter 3 **(Section 3.6.1)**.

Flop-count. The process requires $\frac{10}{3}n^3$ flops to compute H_u. This count does not include the explicit computation of P, which is stored in factored form. If P is computed explicitly, another $\frac{4}{3}n^3$ flops are required. However, when n is large, the storage required to form P is prohibitive.

Roundoff property. **The process is numerically stable.** It can be shown (Wilkinson (1965, p. 351) that the computed H_u is orthogonally similar to a nearby matrix $A + E$, where

$$\|E\|_F \leq cn^2 \mu \|A\|_{\mathrm{F}}.$$

Here c is a constant of order unity.

MATLAB note: The MATLAB Command $[P, H]$ = **hess** (A) computes an orthogonal matrix P and an upper Hessenberg matrix H such that $PAP^{\mathrm{T}} = H$.

4.2.1 Uniqueness in Hessenberg Reduction: The Implicit Q Theorem

We just described Householder's method for Hessenberg reduction. However, this form could also have been obtained using **Givens matrices** as well (see Datta (1995, pp. 163–165). The question, therefore, arises **how unique is the Hessenberg form?**

The question is answered in the following theorem, known as the **Implicit Q Theorem.** The proof can be found in Golub and Van Loan (1996, p. 347).

Theorem 4.2.1. *The Implicit Q Theorem. Let* $P = (p_1, p_2, \ldots, p_n)$ *and* $Q = (q_1, q_2, \cdots, q_n)$ *be orthogonal matrices such that* $P^{\mathrm{T}} A P = H_1$ *and*

*$Q^{\mathrm{T}}AQ = H_2$ are two **unreduced** upper Hessenberg matrices. Suppose that $p_1 = q_1$. Then H_1 and H_2 are essentially the same in the sense that $H_2 = D^{-1}H_1D$, where $D = \mathrm{diag}(\pm 1, \ldots, \pm 1)$. Furthermore, $p_i = \pm q_i, i = 2, \ldots, n$.*

4.3 THE REAL SCHUR FORM OF A: THE QR ITERATION METHOD

In this section, we describe how to obtain the RSF of a matrix. The RSF of a matrix A displays the eigenvalues of A. It is obtained by using the well-known QR iteration method. **This method is nowadays a standard method for computing the eigenvalues of a matrix.** First, we state a well-known classical result on this subject.

Theorem 4.3.1. *The Schur Triangularization Theorem. Let A be an $n \times n$ complex matrix, then there exists an $n \times n$ unitary matrix U such that*

$$U^*AU = T,$$

where T is an $n \times n$ upper triangular matrix and the diagonal entries of T are the eigenvalues of A.

Proof. See Datta (1995, pp. 433–439).

Since a real matrix can have complex eigenvalues (occurring in complex conjugate pairs), even for a real matrix A, U and T in the above theorem can be complex. However, we can choose U to be real orthogonal if T is replaced by a **quasi-triangular matrix** R, known as the RSF of A, as the following theorem shows. The proof can be found in Datta (1995, p. 434) or in Golub and Van Loan (1996, pp. 341–342). ■

Theorem 4.3.2. *The Real Schur Triangularization Theorem. Let A be an $n \times n$ real matrix. Then there exists an $n \times n$ orthogonal matrix Q such that*

$$Q^{\mathrm{T}}AQ = R = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1k} \\ 0 & R_{22} & \cdots & R_{2k} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & R_{kk} \end{pmatrix}, \tag{4.3.1}$$

where each R_{ii} is either a scalar or a 2×2 matrix. The scalars diagonal entries correspond to real eigenvalues, and each 2×2 matrix on the diagonal has a pair of complex conjugate eigenvalues.

Definition 4.3.1. *The matrix R in Theorem* 4.3.2 *is known as the RSF of A.*

Remarks

- The 2×2 matrices on the diagonal are usually referred to as **"Schur bumps."**
- The columns of Q are called the **Schur vectors. For each $k = 1, 2, \ldots, n$, the first k columns of Q form an orthonormal basis for the invariant subspace corresponding to the first k eigenvalues.**

We present below a method, known as the QR iteration method, for computing the RSF of A. **A properly implemented QR method is widely used nowadays for computing the eigenvalues of an arbitrary matrix.** As the name suggests, the method is based on the QR factorization and is iterative in nature. Since the roots of a polynomial equation of degree higher than four cannot be found in a finite number of steps, **any numerical method to compute the eigenvalues of a matrix of order higher than four has to be iterative in nature.** The QR iteration method was proposed in algorithmic form by J.G. Francis (1961), though its roots can be traced to a work of Rutishauser (1958). The method was also independently discovered by the Russian mathematician Kublanovskaya (1961).

For references of these papers, see Datta (1995) or Golub and Van Loan (1996).

4.3.1 The Basic QR Iteration

We first present the basic QR iteration method.

Set $A_0 \equiv A$.
Compute now a sequence of matrices $\{A_k\}$ as follows:
For $k = 1, 2, \ldots$ do
Find the QR factorization of A_{k-1}: $A_{k-1} = Q_k R_k$
Compute $A_k = R_k Q_k$.
End

The matrices in the sequence $\{A_k\}$ have a very interesting property: **Each matrix in the sequence is orthogonally similar to the previous one and is, therefore, orthogonally similar to the original matrix**. It is easy to see this. For example,

$$A_1 = R_1 Q_1 = Q_1^{\mathrm{T}} A_0 Q_1 \text{ (since } R_1 = Q_1^{\mathrm{T}} A_0),$$
$$A_2 = R_2 Q_2 = Q_2^{\mathrm{T}} A_1 Q_2 \text{ (since } R_2 = Q_2^{\mathrm{T}} A_1).$$

Thus, A_1 is orthogonally similar to A and A_2 is orthogonally similar to A_1. Therefore, A_2 is orthogonally similar to A, as the following computation shows:

$$A_2 = Q_2^{\mathrm{T}} A_1 Q_2 = Q_2^{\mathrm{T}} (Q_1^{\mathrm{T}} A_0 Q_1) Q_2 = (Q_1 Q_2)^{\mathrm{T}} A_0 (Q_1 Q_2).$$

Since each matrix A_k is orthogonally similar to the original matrix A, it has the same eigenvalues as A. It can then be shown (Wilkinson (1965, pp. 518–519) that under certain conditions, the sequence $\{A_k\}$ converges to the RSF or to the Schur form of A.

4.3.2 The Hessenberg QR Iteration and Shift of Origin

The QR iteration method as presented above is not practical if the matrix A is full and dense. This is because, as we have seen before, the QR factorization of a matrix A requires $O(n^3)$ flops and thus n iterations will consume $O(n^4)$ flops, making the method impractical.

Fortunately, something simple can be done:

> **Reduce the matrix A to a Hessenberg matrix by orthogonal similarity before starting the QR iterations.** An interesting practical consequence of this is that if $A = A_0$ is initially reduced to an upper Hessenberg matrix H and is assumed to be unreduced, then each member of the matrix sequence$\{H_k\}$ obtained by applying QR iteration to H is also upper Hessenberg. Since the QR factorization of a Hessenberg matrix requires $O(n^2)$ flops, the whole iteration process then becomes $O(n^3)$ method.

However, the convergence of the subdiagonal entries of H, in the presence of two or more nearly equal (in magnitude) eigenvalues, can be painfully slow.

Fortunately, the rate of convergence can be significantly improved by using a suitable shift.

The idea is to apply the QR iteration to the shifted matrix $\hat{H} = H - \hat{\lambda}_i I$, where $\hat{\lambda}_i$ is an approximate eigenvalue. This is known as the **single shift QR iteration.**

However, since the complex eigenvalues of a real matrix occur in conjugate pairs, in practice, the QR iteration is applied to the matrix H with double shifts. The process then is called the **double shift QR iteration** method.

4.3.3 The Double Shift QR Iteration

The Hessenberg double shift QR iteration scheme can be written as follows:

For $i = 1, 2, \ldots$ do
  Choose the two shifts k_1 and k_2
  Find the QR Factorization: $H - k_1 I = Q_1 R_1$
  Form: $H_1 = R_1 Q_1 + k_1 I$
  Find the QR factorization: $H_1 - k_2 I = Q_2 R_2$
  Form: $H_2 = R_2 Q_2 + k_2 I$
  End

The shifts k_1 and k_2 at each iteration are chosen as the eigenvalues of the 2×2 trailing principal submatrix at that iteration. The process is called the **explicit double-shift QR iteration** process.

The above explicit scheme requires complex arithmetic (since k_1 and k_2 are complex) to implement, and furthermore, the matrices $H - k_1 I$ and $H_1 - k_2 I$ need to be formed explicitly. In practice, an equivalent implicit version, known as the **double shift implicit QR iteration scheme**, is used. We state one step of this process in the following.

The Double Shift Implicit QR Step

1. Compute the first column n_1 of the matrix $N = (H - k_1 I)(H - k_2 I) = H^2 - (k_1 + k_2)H + k_1 k_2 I$.
2. Find a Householder matrix P_0 such that $P_0 n_1$ is a multiple of e_1.
3. Find Householder matrices P_1 through P_{n-2} such that $H_2 = (P_{n-2}^{\mathrm{T}} \ldots P_1^{\mathrm{T}} P_0^{\mathrm{T}}) H (P_0 P_1 \ldots P_{n-2})$ is an upper Hessenberg matrix.

It can be shown by using the **Implicit Q Theorem** (Theorem 4.2.1) that **the upper Hessenberg matrix H_2 obtained by the double shift implicit QR step is essentially the same as H_2 obtained by one step of the explicit scheme.** Furthermore, the first column n_1 of N can be computed without explicitly computing the matrix N and, the computation of H_2 from H can be done only in $O(n^2)$ flops. For details see Datta (1995, pp. 444–447).

4.3.4 Obtaining the Real Schur Form A

1. Transform the matrix A to Hessenberg form.
2. Iterate with the double shift implicit QR step.

Typically, after two to three iteration steps of the double shift implicit QR method, one or two (and sometimes more) subdiagonal entries from the bottom of the Hessenberg matrix converge to zero. This then will give us a real or a pair of complex conjugate eigenvalues.

Once a real or a pair of complex conjugate eigenvalues is computed, the last row and the last column in the first case, or the last two rows and the last two columns in the second case, are deleted and the computation of the other eigenvalues is continued with the submatrix.

This process is known as **deflation**.

Note that the eigenvalues of the deflated submatrix are also the eigenvalues of the original matrix. For, suppose, immediately before deflation, the matrix has the form:

$$H_k = \begin{pmatrix} A' & C' \\ 0 & B' \end{pmatrix},$$

where B' is the 2×2 trailing submatrix or a 1×1 matrix. Then the characteristic polynomial of H_k is: $\det(\lambda I - H_k) = \det(\lambda I - A') \det(\lambda I - B')$. Thus, the eigenvalues of H_k are the eigenvalues of A' together with those of B'. But H_k is orthogonally similar to the original matrix A and therefore has the same eigenvalues as A.

Example 4.3.1. Find the RSF of

$$H = \begin{pmatrix} 0.2190 & -0.0756 & 0.6787 & -0.6391 \\ -0.9615 & 0.9032 & -0.4571 & 0.8804 \\ 0 & -0.3822 & 0.4526 & -0.0641 \\ 0 & 0 & -0.1069 & -0.0252 \end{pmatrix}.$$

Iteration	h_{21}	h_{32}	h_{43}
1	0.3860	−0.5084	−0.0084
2	−0.0672	−0.3773	0.0001
3	0.0089	−0.3673	0
4	−0.0011	−0.3590	0
5	0.0001	−0.3905	0
...			

The computed RSF is

$$H = \begin{pmatrix} 1.4095 & 0.7632 & -0.1996 & 0.8394 \\ 0.0001 & 0.1922 & 0.5792 & 0.0494 \\ 0 & -0.3905 & 0.0243 & -0.4089 \\ 0 & 0 & 0 & -0.0763 \end{pmatrix}.$$

The eigenvalues of $\begin{pmatrix} 0.1922 & 0.5792 \\ -0.3905 & 0.0243 \end{pmatrix}$ are $0.1082 \pm 0.4681j$.

Balancing

It is advisable to balance the entries of the original matrix A, if they vary widely, before starting the QR process.

The balancing is equivalent to transforming the matrix A to $D^{-1}AD$, where the diagonal matrix D is chosen so that the transformed matrix has approximately equal row and column norms.

In general, preprocessing the matrix by balancing improves the accuracy of the QR iteration method. **Note that no round-off error is involved in this computation and it takes only $O(n^2)$ flops.**

MATLAB note: The MATLAB command $[T, B] =$ **balance**(A) finds a diagonal matrix T such that $B = T^{-1}AT$ has approximately the equal row and column norms. See MATLAB User's Guide (1992).

Flop-count of the QR iteration method: Since the QR iteration method is an iterative method, it is hard to give an exact flop-count for this method. However, empirical observations have established that it takes about two QR iterations per eigenvalue. Thus, it will require about $12n^3$ flops to compute all the eigenvalues. If the transforming matrix Q and the final quasitriangular matrix T are also needed, then the cost will be about $26n^3$ flops.

Numerical stability property of the QR iteration process: The QR iteration method is quite stable. An analysis of the round-off property of the algorithm shows that the computed RSF $\hat{T}$ is orthogonally similar to a nearby matrix $A + E$. Specifically,

$$Q^{\mathrm{T}}(A + E)Q = \hat{\mathrm{T}}, \quad \text{where} \|E\|_{\mathrm{F}} \leq \phi(n)\mu\|A\|_{\mathrm{F}},$$

where $\phi(n)$ is a slowly growing function of n and μ is the machine precision. The computed orthogonal matrix Q can also be shown to be nearly orthogonal.

MATLAB notes: The MATLAB function **schur** in the following format: $[U, T]$ = **schur**(A) produces a Schur matrix T and an unitary matrix U such that $A = UTU^*$.

By itself, **schur**(A) returns T. If A is real, the RSF is returned.

The RSF has the real eigenvalues on the diagonal and the complex eigenvalues in 2×2 blocks on the diagonal.

4.3.5 The Real Schur Form and Invariant Subspaces

The RSF of A displays information on the invariant subspaces.

Basis of an Invariant Subspace from RSF

Let

$$Q^{\mathrm{T}}AQ = R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

and let's assume that R_{11} and R_{22} do not have eigenvalues in common. **Then the first p columns of Q, where p is the order of R_{11}, form a basis for the invariant subspace associated with the eigenvalues of R_{11}.**

In many applications, such as in the **solution of algebraic Riccati equations** (see Chapter 13), in constructing a reduced-order model, etc., one needs to compute an orthonormal basis of an invariant subspace associated with a selected number of eigenvalues. Unfortunately, the RSF obtained by QR iteration will not, in general, give the eigenvalues in some desired order. Thus, if the eigenvalues are not in a desired order, one wonders if some extra work can be done to bring them into that order. That this can indeed be done, is seen from the following simple discussion. Let A be 2×2.

Let

$$Q_1^{\mathrm{T}}AQ_1 = \begin{pmatrix} \lambda_1 & r_{12} \\ 0 & \lambda_2 \end{pmatrix}, \quad \lambda_1 \neq \lambda_2.$$

If λ_1 and λ_2 are not in right order, all we need to do to reverse the order is to form a Givens rotation $J(1, 2, \theta)$ such that

$$J(1, 2, \theta) \begin{pmatrix} r_{12} \\ \lambda_2 - \lambda_1 \end{pmatrix} = \begin{pmatrix} * \\ 0 \end{pmatrix}.$$

Then $Q = Q_1 J(1, 2, \theta)^T$ is such that

$$Q^T A Q = \begin{pmatrix} \lambda_2 & r_{12} \\ 0 & \lambda_1 \end{pmatrix}.$$

The above simple process can be easily extended to achieve any desired ordering of the eigenvalues in the RSF. For a Fortran program, see Stewart (1976).

Example 4.3.1.

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix},$$

$$Q_1 = \begin{pmatrix} 0.8507 & 0.5257 \\ -0.5257 & 0.8507 \end{pmatrix},$$

$$Q_1^T A Q_1 = \begin{pmatrix} -0.2361 & 0.0000 \\ 0.0000 & 4.2361 \end{pmatrix}.$$

Suppose we now want to reverse the orders of -0.2361, and 4.2361.

Form: $J(1, 2, \theta) = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$.

Then, $J(1, 2, \theta) \begin{pmatrix} 0 \\ 4.4722 \end{pmatrix} = \begin{pmatrix} 4.4722 \\ 0 \end{pmatrix}$.

Form: $Q = Q_1 J(1, 2, \theta)^T = \begin{pmatrix} -0.5257 & -0.8507 \\ -0.8507 & 0.5257 \end{pmatrix}$.

Then, $Q^T A Q = \begin{pmatrix} 4.2361 & 0.00 \\ 0.00 & -0.2361 \end{pmatrix}$.

Flop-count and numerical stability. **The process is quite inexpensive.** It requires only $k(12n)$ flops, where k is the number of interchanges required to achieve the desired order. **The process is also numerically stable.**

MATCONTROL note: The routine **ordersch** in MATCONTROL can be used to order the eigenvalues in the RSF of the matrix.

Fortran Routine: The Fortran routine STRSYL in LAPACK (Anderson *et al.* (1999)) reorders the Schur decomposition of a matrix in order to find an orthonormal basis of a right invariant subspace corresponding to selected eigenvalues.

Invariant Subspace Sensitivity: Sep-Function

Let $Q^*AQ = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$ be the Schur decomposition of A. Let $Q = (Q_1, Q_2)$. Define

$$\text{sep}(T_{11}, T_{22}) = \min_{X \neq 0} \frac{\|T_{11}X - XT_{22}\|_F}{\|X\|_F}.$$

Then it can be shown (Golub and Van Loan (1996, pp. 325)) that **the reciprocal of** $\text{sep}(T_{11}, T_{22})$ **is a good measure of the sensitivity of the invariant subspace spanned by the columns of** Q**.**

4.3.6 Inverse Iteration

The inverse iteration is a commonly used procedure to compute a selected number of eigenvectors of a matrix.

Since A is initially reduced to a Hessenberg matrix H for the QR iteration process, then it is natural to take advantage of the structure of the Hessenberg matrix H in the process of inverse iteration. The **Hessenberg inverse iteration** can then be stated as follows:

Step 1. Reduce the matrix A to an upper Hessenberg matrix $H : PAP^T = H$.

Step 2. Compute an eigenvalue λ, whose eigenvector x is sought, using the implicit QR iteration method described in the previous section.

Step 3. Choose a unit-length vector $y_0 \in \mathbb{C}^n$.

For $k = 1, 2, \ldots$ do until convergence
  Solve for $z^{(k)} : (H - \lambda I)z^{(k)} = y^{(k-1)}$
  Compute $y^{(k)} = z^{(k)}/\|z^{(k)}\|$
End

Step 4. Recover the eigenvector x of the matrix $A : x = P^T y^{(k)}$, where $y^{(k)}$ is an approximation of the eigenvector y obtained at the end of Step 3.

Note: If y is an eigenvector of H, then $x = P^T y$ is the corresponding eigenvector of A.

Convergence and efficiency: The Hessenberg inverse iteration is very inexpensive. Once an eigenvalue is computed, the whole process requires only $O(n^2)$ flops. It typically requires only 1 to 2 iterations to obtain an approximate acceptable eigenvector.

4.4 COMPUTING THE SINGULAR VALUE DECOMPOSITION (SVD)

The following algorithm known as the **Golub–Kahan–Reinsch algorithm** is nowadays a standard computational algorithm for computing the SVD. The algorithm comes in two stages:

Stage I. The $m \times n$ matrix A $(m \geq n)$ is transformed to an upper $m \times n$ bidiagonal matrix by orthogonal equivalence:

$$U_0^{\mathrm{T}} A\, V_0 = \begin{pmatrix} B \\ 0 \end{pmatrix}, \tag{4.4.1}$$

where B is the $n \times n$ upper bidiagonal matrix given by

$$B = \begin{pmatrix} b_1 & * & & 0 \\ & \ddots & * & \\ & & \ddots & \ddots \\ & & & * \\ 0 & & & b_n \end{pmatrix}$$

Stage II. The transformed bidiagonal matrix B is further reduced by orthogonal equivalence to a diagonal matrix Σ using the QR iteration method; that is, orthogonal matrices U_1 and V_1 are constructed such that

$$U_1^{\mathrm{T}} B V_1 = \Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n). \tag{4.4.2}$$

The matrix Σ is the matrix of singular values. The singular vector matrices U and V are given by $U = U_0 U_1$, $V = V_0 V_1$.

We will briefly describe Stage I here. For a description of Stage II, see Golub and Van Loan (1996, pp. 452–457).

Reduction to Bidiagonal Form

We show how Householder matrices can be employed to construct U_0 and V_0 in Stage I.

The matrices U_0 and V_0 are constructed as the product of Householder matrices as follows: $U_0 = U_1 U_2 \ldots U_n$, and $V_0 = V_1 V_2 \ldots V_{n-2}$. Let's illustrate construction of U_1, V_1 and U_2, V_2, and their role in the bidiagonalization process with $m = 5$ and $n = 4$.

First, a Householder matrix U_1 is constructed such that

$$A^{(1)} = U_1 A = \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix}.$$

Next, a Householder matrix V_1 is constructed such that

$$A^{(2)} = A^{(1)} V_1 = \begin{pmatrix} * & * & 0 & 0 \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix} = \left(\begin{array}{c|ccc} * & * & 0 & 0 \\ \hline 0 & & & \\ 0 & & A' & \\ 0 & & & \\ 0 & & & \end{array}\right).$$

The process is now repeated with $A^{(2)}$; that is, Householder matrices U_2 and V_2 are constructed so that

$$U_2 A^{(2)} V_2 = \begin{pmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{pmatrix}.$$

Of course, in this step, we will work with the 4×3 matrix A' rather than the matrix $A^{(2)}$. Thus, first the orthogonal matrices U_2' and V_2' will be constructed such that

$$U_2' A' V_2' = \begin{pmatrix} * & * & 0 \\ 0 & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix},$$

then U_2 and V_2 will be constructed from U_2' and V_2' in the usual way, that is, by embedding them in identity matrices of appropriate orders. The process is continued until the bidiagonal matrix B is obtained.

Example 4.4.1. Let

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix}.$$

Step 1.

$$U_1 = \begin{pmatrix} -0.1474 & -0.4423 & -0.8847 \\ -0.4423 & 0.8295 & -0.3410 \\ -0.8847 & -0.3410 & 0.3180, \end{pmatrix},$$

$$A^{(1)} = U_1 A = \begin{pmatrix} -6.7823 & -8.2567 & -9.7312 \\ 0 & 0.0461 & 0.0923 \\ 0 & -0.9077 & -1.8154 \end{pmatrix}.$$

Step 2.

$$V_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.6470 & 0.7625 \\ 0 & -0.5571 & 0.6470 \end{pmatrix},$$

$$A^{(2)} = A^{(1)} V_1 = \begin{pmatrix} -6.7823 & 12.7620 & 0 \\ 0 & -1.0002 & 0.0245 \\ 0 & 1.9716 & -0.4824 \end{pmatrix}.$$

Step 3.

$$U_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.0508 & 0.9987 \\ 0 & 0.9987 & 0.0508 \end{pmatrix}$$

$$B = U_2 A^{(2)} = U_2 A^{(1)} V_1 = U_2 U_1 A V_1 = \begin{pmatrix} -6.7823 & 12.7620 & 0 \\ 0 & -1.0081 & -1.8178 \\ 0 & 0 & 0 \end{pmatrix}$$

Note that from the above expression of B, it immediately follows that zero is a singular value of A.

Flop-count: The Householder bidiagonalization algorithm requires $4mn^2 - 4n^3/3$ flops.

Stage II, that is, the process of iterative reduction of the bidiagonal matrix to a diagonal matrix containing the singular values requires $30n$ flops and $2n$ square roots. The matrices U and V can be accumulated with $6mn$ and $6n^2$ flops, respectively.

Stability: The Golub–Kahan–Reinsch algorithm is **numerically stable**. It can be shown that the process will yield orthogonal matrices U and V and a diagonal matrix Σ such that $U^{\mathrm{T}} A V = \Sigma + E$, where $\|E\|_2 \approx \mu \|A\|_2$.

4.5 THE GENERALIZED REAL SCHUR FORM: THE QZ ALGORITHM

In this section, we describe two canonical forms for a pair of matrices (A, B): The **Hessenberg-triangular** and the **Generalized RSF**. The Generalized RSF displays the eigenvalues of the matrix pencil $A - \lambda B$, as the RSF does for the matrix A.

Given $n \times n$ matrices A and B, a scalar λ and a nonzero vector x satisfying

$$Ax = \lambda Bx$$

are respectively called an **eigenvalue** and **eigenvector** for the pencil $A - \lambda B$. The eigenvalue problem itself is called **generalized eigenvalue problem**. The eigenvalues and eigenvectors of the generalized eigenvalue problem are often called **generalized eigenvalues** and **generalized eigenvectors**. The matrix pencil $A - \lambda B$ is often conveniently denoted by the pair (A, B).

The pair (A, B) is called **regular** if $\det(A - \lambda B)$ is not identically zero. Otherwise, it is **singular. We will consider only regular pencil here.** If B is nonsingular, then the eigenvalues **of the regular** pair (A, B) are finite and are the same as those of AB^{-1} or $B^{-1}A$.

If B is singular, and if the degree of $\det(A - \lambda B)$ is $r (< n)$, then $n - r$ eigenvalues of (A, B) are ∞, and the remaining ones are the zeros of $\det(A - \lambda B)$.

As we will see later, the generalized RSF is an important tool in the numerical solutions of the **discrete algebraic Riccati equation** and **the Riccati equations with singular and ill-conditioned control weighting matrices** (Chapter 13).

The QZ algorithm

Assume that B is nonsingular. Then the basic idea is to apply the QR iteration algorithm to the matrix $C = B^{-1}A$ (or to AB^{-1}), without explicitly forming the matrix C. For if B is nearly singular, then it is not desirable to form B^{-1}. In this case the entries of C will be much larger than those of A and B, and the eigenvalues of C will be computed inaccurately. (Note that the eigenvalues of $B^{-1}A$ are the same as those of AB^{-1}, because $AB^{-1} = B(B^{-1}A)B^{-1}$). If AB^{-1} or $B^{-1}A$ is not to be computed explicitly, then the next best alternative, of course, is to transform A and B simultaneously to some reduced forms such as the triangular forms and then extract the generalized eigenvalues from these reduced forms. The simultaneous reduction of A and B to triangular forms by equivalence is guaranteed by the following theorem:

Theorem 4.5.1. *The Generalized Real Schur Decomposition. Given two $n \times n$ real matrices A and B, there exist orthogonal matrices Q and Z such that $Q^{\mathrm{T}}AZ$ is an upper real Schur matrix and $Q^{\mathrm{T}}BZ$ is upper triangular:*

$$Q^{\mathrm{T}}AZ \equiv A', \text{ an upper real Schur matrix,}$$

$$Q^{\mathrm{T}}BZ \equiv B', \text{ an upper triangular matrix.}$$

The pair (A', B') is said to be in **generalized RSF**.

The reduction to the generalized RSF is achieved in two stages.

Stage I. The matrices A and B are reduced to an upper Hessenberg and an upper triangular matrix, respectively, by simultaneous orthogonal equivalence:

$$A \equiv Q^{\mathrm{T}} A Z, \text{ an upper Hessenberg matrix,}$$
$$B \equiv Q^{\mathrm{T}} B Z, \text{ an upper triangular matrix.}$$

Stage II. The Hessenberg-triangular pair (A, B) is further reduced to the **generalized RSF** by applying **implicit QR iteration** to AB^{-1}.

This process is known as the **QZ Algorithm**.

We will now briefly sketch these two stages in the sequel.

4.5.1 Reduction to Hessenberg-Triangular Form

Let A and B be two $n \times n$ matrices. Then,

Step 1. Find an orthogonal matrix U such that

$$B \equiv U^{\mathrm{T}} B$$

is an upper triangular matrix by finding the QR factorization of B.

Form

$$A \equiv U^{\mathrm{T}} A$$

(in general, A will be full).

Step 2. Reduce A to Hessenberg form while preserving the triangular structure of B.

Step 2 is achieved as follows:

To start with, we have

$$A \equiv U^{\mathrm{T}} A = \begin{pmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & & & \\ * & * & \cdots & * \\ * & * & \cdots & * \end{pmatrix},$$

$$B \equiv U^{\mathrm{T}} B = \begin{pmatrix} * & * & \cdots & \cdots & * \\ 0 & * & \cdots & \cdots & * \\ 0 & 0 & \ddots & \cdots & * \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots 0 & * \end{pmatrix}.$$

First, the $(n, 1)$th entry of A is made zero by applying a Givens rotation $Q_{n-1,n}$ in the $(n-1, n)$ plane:

$$A \equiv Q_{n-1,n}A = \begin{pmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & & & \\ * & * & \cdots & * \\ 0 & * & \cdots & * \end{pmatrix}.$$

This transformation, when applied to B from the left, will give a fill-in in the $(n, n-1)$ position:

$$B \equiv Q_{n-1,n}B = \begin{pmatrix} * & * & \cdots & \cdots & * \\ 0 & * & \cdots & \cdots & * \\ 0 & 0 & * & \cdots & * \\ \vdots & & & \ddots & \vdots \\ 0 & 0 \cdots & 0 & * & * \end{pmatrix}.$$

The Givens rotation $Z_{n-1,n} = J(n-1, n, \theta)$ is now applied to the right of B to make the $(n, n-1)$ entry of B zero. Fortunately, this rotation, when applied to the right of A, does not destroy the zero produced earlier. Schematically, we have

$$B \equiv BZ_{n-1,n} = \begin{pmatrix} * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ 0 & 0 & * & \cdots & * \\ \vdots & & \ddots & \ddots & \\ 0 & 0 & \cdots & 0 & * \end{pmatrix},$$

$$A \equiv AZ_{n-1,n} = \begin{pmatrix} * & * & * & \cdots & * \\ * & * & * & \cdots & * \\ * & * & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \end{pmatrix}.$$

The entries $(n-1, 1), (n-2, 1), \ldots, (3, 1)$ of A are now successively made zero, each time applying an appropriate rotation to the left of A, followed by another appropriate Givens rotation to the right of B to zero out the undesirable fill-in in B. At the end of the first step, the matrix A is Hessenberg in its first column, while

B remains upper triangular:

$$A = \begin{pmatrix} * & * & \cdots & * \\ * & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & * & \cdots & * \end{pmatrix}, \qquad B = \begin{pmatrix} * & * & \cdots & \cdots & * \\ * & * & \cdots & \cdots & * \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & * \\ 0 & \cdots & 0 & * & * \end{pmatrix}.$$

The zeros are now produced on the second column of A in the appropriate places while retaining the triangular structure of B in an analogous manner.

The process is continued until the matrix A is an upper Hessenberg matrix while keeping B in upper triangular form.

Example 4.5.1.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 4 \\ 1 & 3 & 3 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 2 \end{pmatrix}.$$

1. Form the Givens rotation Q_{23} to make a_{31} zero:

$$Q_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.7071 & 0.7071 \\ 0 & -0.7071 & 0.7071 \end{pmatrix},$$

$$A \equiv A^{(1)} = Q_{23}A = \begin{pmatrix} 1 & 2 & 3 \\ 1.4142 & 4.2426 & 4.9497 \\ 0 & 0 & -0.7071 \end{pmatrix}.$$

2. Update B:

$$B \equiv B^{(1)} = Q_{23}B = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0.7071 & 2.8284 \\ 0 & -0.7071 & 0 \end{pmatrix}.$$

3. Form the Givens rotation Z_{23} to make b_{32} zero:

$$Z_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix},$$

$$B \equiv B^{(1)}Z_{23} = Q_{23}BZ_{23} = \begin{pmatrix} 1 & 1 & -1 \\ 0 & 2.8284 & -0.7071 \\ 0 & 0 & 0.7071 \end{pmatrix}.$$

4. Update A:

$$A \equiv A^{(1)}Z_{23} = Q_{23}AZ_{23} = \begin{pmatrix} 1 & 3 & -2 \\ 1.4142 & 4.9497 & -4.2426 \\ 0 & -0.7071 & 0 \end{pmatrix}.$$

Now A is an upper Hessenberg and B is in upper triangular form.

4.5.2 Reduction to the Generalized Real Schur Form

At the beginning of this process, we have A and B as an upper Hessenberg and an upper triangular matrix, respectively, obtained from Stage 1. We can assume without loss of generality that the matrix A is an unreduced upper Hessenberg matrix. **The basic idea now is to apply an implicit QR step to AB^{-1} without ever forming this matrix explicitly.** We sketch just the basic ideas here. For details, see Datta (1995, pp. 500–504).

Thus a QZ step, analogous to an implicit QR step, will be as follows:

1. Compute the first column n_1 of $N = (C - \alpha_1 I)(C - \alpha_2 I)$, where $C = AB^{-1}$ and α_1 and α_2 are suitably chosen shifts, without explicitly forming the matrix AB^{-1}.
   (Note that n_1 has only three nonzero entries and the rest are zero).
2. Find a Householder matrix Q_1, such that $Q_1 n_1$ is a multiple of e_1.
3. Form $Q_1 A$ and $Q_1 B$.
4. Simultaneously transform $Q_1 A$ to an upper Hessenberg matrix A_1, and $Q_1 B$ to an upper triangular matrix B_1:

$$A_1 \equiv Q^{\mathrm{T}}(Q_1 A)Z : \text{ an upper Hessenberg;}$$
$$B_1 \equiv Q^{\mathrm{T}}(Q_1 B)Z : \text{ an upper triangular.}$$

Using the implicit Q theorem (Theorem 4.2.1) we can show that the matrix $A_1 B_1^{-1}$ is essentially the same as that would have been obtained by applying an implicit QR step directly to AB^{-1}.

Applications of a few QZ steps in sequence will then yield a quasi-triangular matrix $R = Q^{\mathrm{T}}AZ$ and an upper triangular $T = Q^{\mathrm{T}}BZ$, from which the generalized eigenvalues can be easily extracted.

Choosing the Shifts

The double shifts α_1 and α_2 at a QZ step can be taken as the eigenvalues of the lower 2×2 submatrix of $C = AB^{-1}$. **The 2×2 lower submatrix of C again can be computed without explicitly forming B^{-1}** (see Datta (1995, p. 501)).

Algorithm 4.5.1. *The Complete QZ Algorithm for Reduction to Generalized Schur Form*

Inputs: Real $n \times n$ matrices A and B.

Outputs: The pair (R, T) of the generalized RSF of the pencil $A - \lambda B$. The matrix R is Quasi-triangular and T is upper triangular.

1. Transform (A, B) to a Hessenberg-triangular pair by orthogonal equivalence:

$$A \equiv Q^{\mathrm{T}} A Z, \text{ an upper Hessenberg,}$$
$$B \equiv Q^{\mathrm{T}} B Z, \text{ an upper triangular.}$$

2. Apply a sequence of the QZ steps to the Hessenberg-triangular pair (A, B) to produce $\{A_k\}$ and $\{B_k\}$, with properly chosen shifts.
3. Monitor the convergence of the sequences $\{A_k\}$ and $\{B_k\}$:

$$\{A_k\} \longrightarrow R, \text{ quasi-triangular (in RSF),}$$
$$\{B_k\} \longrightarrow T, \text{ upper triangular.}$$

Flop-count: The implementation of (1)–(3) requires about $30n^3$ flops. The formation of Q and Z, if required, needs, respectively, another $16n^3$ and $20n^3$ flops (from experience it is known that about two QZ steps per eigenvalue are adequate).

Numerical Stability Properties: The QZ iteration algorithm is as **stable** as the QR iteration algorithm. It can be shown that the computed $\hat{R}$ and $\hat{S}$ satisfy

$$Q_0^{\mathrm{T}}(A + E)Z_0 = \hat{R}, \qquad Q_0^{\mathrm{T}}(B + F)Z_0 = \hat{S}.$$

Here Q_0 and Z_0 are orthogonal, $\|E\| \cong \mu\|A\|$and $\|F\| \cong \mu\|B\|$; μ is the machine precision.

4.6 COMPUTING OF THE EIGENVECTORS OF THE PENCIL $A - \lambda B$

Once an approximate generalized eigenvalue λ is computed, the corresponding eigenvector v of the pencil $A - \lambda B$ can be computed using the **generalized inverse iteration** as before.

Step 1. Choose an initial eigenvector v_0.

Step 2. For $k = 1, 2, \ldots$ do until convergence

$$\text{Solve } (A - \lambda B)\hat{v}_k = Bv_{k-1};$$
$$v_k = \hat{v}_k / \|\hat{v}_k\|_2. \tag{4.6.1}$$

A Remark on Solving $(A - \lambda B)\hat{v}_k = Bv_{k-1}$

In solving $(A - \lambda B)\hat{v}_k = Bv_{k-1}$, substantial savings can be made by exploiting the Hessenberg-triangular structure to which the pair (A, B) is reduced as a part

of the QZ algorithm. Note that in this case for a given λ, the matrix $A - \lambda B$ is also a Hessenberg matrix. Thus, at each iteration, only a Hessenberg system needs to be solved, which requires only $O(n^2)$ flops, compared to $O(n^3)$ flops required for a system with a full matrix.

Example 4.6.1.

$$A = 10^9 \begin{pmatrix} 3 & -1.5 & 0 \\ -1.53 & -1.5 & \\ 0 & -1.51.5 & \end{pmatrix}, \qquad B = 10^3 \begin{pmatrix} 200 \\ 030 \\ 004 \end{pmatrix}.$$

$$\lambda_1 = \text{a generalized eigenvalue of } (A - \lambda B) = 1950800.$$

$$v_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

$k = 1$: Solve for v_1:

$$\text{Solve: } (A - \lambda_1 B)\hat{v}_1 = Bv_0$$

$$\hat{v}_1 = \begin{pmatrix} 0.0170 \\ -0.0102 \\ 0.0024 \end{pmatrix}, \qquad v_1 = \hat{v}_1/\|\hat{v}_1\| = \begin{pmatrix} 0.8507 \\ -0.5114 \\ 0.1217 \end{pmatrix}.$$

MATLAB and MATCOM notes: The MATLAB function **qz** in the form: $[AA, BB, Q, Z, V] = qz(A, B)$ produces upper triangular matrices AA and BB, and the orthogonal matrices Q and Z such that $QAZ = AA$, $QBZ = BB$.

The matrix V contains the eigenvectors. The generalized eigenvalues are obtained by taking the ratios of the corresponding diagonal entries of AA and BB. The MATLAB function **eig** (A, B) gives only the generalized eigenvalues of the pencil $A - \lambda B$ from the generalized Schur decomposition. MATCOM functions HESSTRI and INVITRGN compute, respectively, the Hessenberg-triangular reduction of the pair (A, B) and the eigenvectors of the pencil $A - \lambda B$ using inverse iteration.

Deflating Subspace for the Pencil $A - \lambda B$

A k-dimensional subspace $S \in \mathbb{R}^n$ is a **deflating subspace** of the pencil $A - \lambda B$ if the subspace $\{Ax + By \mid x, y \in S\}$ has dimension k or less. It can be easily seen that the **columns of Z in the generalized Schur decomposition form a family of deflating subspaces.** Also, span$\{Az_1, \ldots, Az_k\}$ and span$\{Bz_1, \ldots, Bz_k\}$ belong to span$\{q_1, \ldots, q_k\}$, where z_i and q_i are, respectively, the columns of Z and Q.

Remark

- In solving algebraic Riccati equations, deflating subspaces with specified spectrum need to be computed. There exist Fortran routines for computing such deflating subspaces developed by Van Dooren (1982).

4.7 SUMMARY AND REVIEW

Numerical Instability in Obtaining Jordan and Companion Matrices

The JCF and a companion form of a matrix, because of their rich structures, are important theoretical tools. Using these two decompositions, many important results in control theory have been established (see Kailath 1980).

Unfortunately, however, these two forms cannot be obtained in a numerically stable way in general. Since it is necessary to use non-orthogonal transformations to achieve these forms, the transforming matrices can be highly ill-conditioned. Some discussions to this effect have been given in **Section 4.1**. Because of possible numerical instabilities in reduction of A to a companion matrix, and the fact that the zeros of a polynomial can be extremely sensitive to small perturbations, **it is not advisable to compute the eigenvalues of a matrix by finding the zeros of its characteristic polynomial.**

Hessenberg and Real Schur Forms

Both Hessenberg and RCFs can be obtained via orthogonal similarity transformations. These two forms, thus, are extremely valuable tools in numerical computations. In fact, many of the numerically effective algorithms for control problems described in this book, are based on these two forms.

Reduction to Hessenberg form. A Hessenberg form, via orthogonal similarity transformation, is obtained using either Householder or Givens transformations. The Householder method for Hessenberg reduction is described in Section 4.2. For a description of Givens Hessenberg reduction, see Datta (1995) or Golub and Van Loan (1996). The **implicit Q theorem (Theorem 4.2.1)** guarantees that the Hessenberg forms obtained by two different methods are essentially the same, provided that the transforming matrices have the same first column.

Real Schur form: Computing the eigenvalues, eigenvectors, and orthonormal bases for invariant subspaces. The RSF of a matrix is a **quasi-triangular matrix** whose diagonal entries are either scalars or 2×2 matrices. Every real matrix A can be transformed to RSF by an orthogonal similarity. Since the RSF of a matrix A displays the eigenvalues of A, any numerical method for obtaining the RSF of order higher than four X has to be iterative in nature. The standard method for obtaining the RSF is the QR iteration method with implicit double shift. This method is

described in some detail in Sections 4.3.1–4.3.4. **The double shift implicit QR iteration method is nowadays the standard method for finding the eigenvalues of a matrix.**

An orthonormal basis for the invariant subspace associated with a given set of eigenvalues can also be found by reordering the eigenvalues in RSF in a suitable way. This is discussed in **Section 4.3.5.**

Once the RSF is found, it can be employed to compute the eigenvectors of A. This is not discussed here. Interested readers are referred to Datta (1995, pp. 452–455). Instead, a commonly used procedure for computing selected eigenvectors, called the **inverse iteration method, is described in Section 4.3.**

Computing the SVD of a Matrix

The standard method for computing the SVD, called the **Golub–Kahan–Reinsch algorithm**, is described in Section 4.4. The method comes in two stages:

Stage I. Reduction of the matrix A to a bidiagonal form.

Stage II. Further reduction of the bidiagonal matrix obtained in Stage I to a diagonal matrix using implicit QR iteration.

The detailed discussion of Stage II is omitted here. The readers are referred to Golub and Van Loan (1996, pp. 452–456).

The Generalized Real Schur Form

The generalized RSF of a pair of matrices (A, B) is a matrix-pair (A', B'), where A' is an upper real Schur matrix and B' is an upper triangular matrix **(Theorem 4.5.1).**

The standard method for computing the general RSF is the **QZ iteration algorithm.** The QZ algorithm also comes in two stages:

Stage I. Reduction of (A, B) to Hessenberg-triangular form.

Stage II. Further reduction of the Hessenberg-triangular form obtained in Stage I to the generalized RSF.

Stage I is a finite procedure. Again, the Householder or Givens transformations can be used. The Householder procedure is described in **Section 4.5.1.** Stage II is an iterative procedure. Only a brief sketch of the procedure is presented here in **Section 4.5.2.** For details, readers are referred to Datta (1995, pp. 500–504).

The generalized RSF displays the eigenvalues (called generalized eigenvalues) of the linear pencil $A - \lambda B$. Once the eigenvalues are obtained, the selected eigenvectors can be computed using **generalized inverse iteration (Section 4.6).**

4.8 CHAPTER NOTES AND FURTHER READING

The material of this chapter has been taken from the recent book of the author (Datta 1995). For advanced readings of the topics dealt with in this chapter, consult the

book by Golub and Van Loan (1996) and Stewart (2001). For a description of the toolbox MATCOM and how to obtain it, see the section on Chapter Notes and Further Reading of Chapter 3 (**Section 3.11**). For MATLAB functions and LAPACK routines, see the respective user's guides; Anderson *et al.* (1995) and MATLAB User's Guide (1992)

References

Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., and Sorensen D. *LAPACK Users' Guide*, 2nd edn, SIAM, Philadelphia, 1999.

Datta B.N. *Numerical Linear Algebra and Applications*, Brooks/Cole Publishing Company, Pacific Grove, CA. 1995.

Golub G.H. and Van Loan C.F. *Matrix Computations*, 3rd edn, The Johns Hopkins University Press, Baltimore, MD, 1996.

Kailath T. *Linear Systems*, Prentice Hall, Englewood Cliffs, N.J, 1980.

MATLAB *User's Guide*, The Math Works, Inc., Natick, MA, 1992.

Stewart G.W. "Algorithm 406. HWR3 and EXCHNG: FORTRAN programs for calculating the eigensystems of a real upper Hessenberg matrix in a prescribed order," *ACM Trans. Math. Soft.* Vol. 2, pp. 275–280, 1976.

Stewart G.W. *Matrix Algorithms, Vol. II: Eigen Systems*, SIAM, Philadelphia, 2001.

Van Dooren P. "Algorithm 590–DSUBSP and EXCHQZ: Fortran subroutines for computing deflating subspaces with specified spectrum," *ACM Trans. Math. Soft.*, Vol. 8, pp. 376–382, 1982.

Wilkinson J.H. *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England, 1965.