CHAPTER 11

NUMERICAL METHODS AND CONDITIONING OF THE EIGENVALUE ASSIGNMENT PROBLEMS

Topics covered

- Numerical Methods for the Single-Input and Multi-Input Eigenvalue Assignment (EVA) Problems
- Sensitivity Analyses of the Feedback and EVA Problems and Conditioning of the Closed-Loop Eigenvalues
- Robust EVA

11.1 INTRODUCTION

We have introduced the eigenvalue assignment (EVA) problem **(pole-placement problem)** in Chapter 10 and given the results on existence and uniqueness of the solution.

In this chapter, we study numerical methods and the perturbation analysis for this problem.

There are many methods for the EVA problem. As stated in Chapter 10, some of the well-known theoretical formulas, such as the Ackermann formula, the Bass–Gura formula, etc., though important in their own rights, do not yield to computationally viable methods. The primary reason is that these methods are based on transformation of the controllable pair (A, B) to the controller-companion form, and the transforming matrix can be highly ill-conditioned in some cases. The computationally viable methods for EVA are based on transformation of the pair (A, B) to the controller-Hessenberg form or the matrix A to the real Schur form (RSF), which can be achieved using orthogonal transformations. Several methods of this

type have been developed in recent years and we have described a few of them in this chapter. The methods described here include:

- A single-input recursive algorithm (**Algorithm 11.2.1**) (Datta 1987) and its RQ implementation (**Algorithm 11.2.3**) (Arnold and Datta 1998).
- A multi-input generalization (**Algorithm 11.3.1**) of the single-input recursive algorithm (Arnold and Datta 1990).
- A multi-input explicit QR algorithm (**Section 11.3.2**) (Miminis and Paige 1988).
- A multi-input Schur algorithm (**Algorithm 11.3.3**) (Varga 1981).
- A Sylvester equation algorithm for **partial eigenvalue assignment** (PEVA) (**Algorithm 11.3.4**) (Datta and Sarkissian 2002).

Algorithms 11.2.1 and 11.3.1 are the fastest algorithms, respectively, for the single-input and the multi-input EVA problems.

Unfortunately, the numerical stability of these algorithms are not guaranteed. However, many numerical experiments performed by the author and others (e.g., Varga 1996; Arnold and Datta 1998; Calvetti *et al.* 1999) show that Algorithm 11.2.1 works extremely well in practice, even with ill-conditioned problems. Furthermore, there is an RQ implementation which is numerically stable (Arnold and Datta 1998). This stable version is described in **Algorithm 11.2.3.**

The **multi-input explicit QR** algorithm in **Section 11.3.2** is also numerically stable. **However, it might give a complex feedback in some cases.**

The **Schur algorithm (Algorithm 11.3.3)**, based on the real Schur decomposition of the matrix A, is most expensive, but it has a distinguished feature that it can be used as a partial-pole placement algorithm in the sense that it lets the user reassign only a part of the spectrum leaving the rest unchanged. The algorithm is also believed to be **numerically stable.**

Besides the above-mentioned algorithms, an algorithm (**Algorithm 11.6.1**) for **robust eigenvalue assignment** (REVA), which not only assigns a desired set of eigenvalues but also a set of well-conditioned eigenvectors as well, is also included in this chapter. The REVA is important because the conditioning of the closed-loop eigenvector matrix greatly influences the sensitivity of the closed-loop eigenvalues (**see Section 11.5**). **Algorithm 11.6.1** is due to Kautsky *et al.* (1985) and is popularly known as the **KNV algorithm.** The MATLAB function **place** has implemented this algorithm.

Sections 11.4 and 11.5 are devoted, respectively, to the **conditioning of the feedback matrix and that of the closed-loop eigenvalues.** The conditioning of the feedback matrix and the conditioning of the closed-loop eigenvalues are two different matters. *It is easy to construct examples for which the feedback matrix can be computed rather very accurately by using a backward stable algorithm, but the resulting closed-loop eigenvalues might still be significantly*

different from those to be assigned. These two problems are, therefore, treated separately.

The chapter concludes with a table of **comparison of different methods (Sections 11.7 and 11.8)** and **recommendations** are made based on this comparison (**Section 11.9**).

11.2 NUMERICAL METHODS FOR THE SINGLE-INPUT EIGENVALUE ASSIGNMENT PROBLEM

The constructive proof of Theorem 10.4.1 suggests the following method for finding the feedback vector f. Let (A, b) be controllable and S be the set of eigenvalues to be assigned.

Eigenvalue Assignment via Controller-Companion Form

Step 1. Find the coefficients $d_1, d_2, \ldots, d_n$ of the characteristic polynomial of the desired closed-loop matrix from the given set S.

Step 2. Transform (A, b) to the controller-companion form $(C, \tilde{b})$:

$$TAT^{-1} = C, \qquad Tb = \tilde{b},$$

where C is a **lower companion matrix** and $\tilde{b} = (0, 0, \ldots, 0, 1)^{\mathrm{T}}$.

Step 3. Compute $\hat{f}_i = d_i - a_i$, $i = 1, 2, \ldots, n$ where a_i, $i = 1, \ldots, n$ are the entries of the last row of C.

Step 4. Find $f^{\mathrm{T}} = \hat{f}^{\mathrm{T}} T$, where $\hat{f}^{\mathrm{T}} = (\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_n)$.

Because of the difficulty of the implementation of Step 1 for large problems and of the instability of the algorithm due to possible ill-condition of T for finding the controller-canonical form in Step 2, as discussed in Chapter 6, **the above method is clearly not numerically viable.** It is of theoretical interest only.

Similar remarks hold for Ackermann's formula. The Ackermann (1972) *formula, though important in its own right, is not numerically effective.* Recall that the Ackermann formula for computing f to assign the spectrum $S = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ is:

$$f^{\mathrm{T}} = e_n^{\mathrm{T}} C_{\mathrm{M}}^{-1} \phi(A),$$

where $C_{\mathrm{M}} = (b, Ab, \ldots, A^{n-1}b)$ and $\phi(x) = (x - \lambda_1)(x - \lambda_2) \cdots (x - \lambda_n)$. Thus, the implementation of Ackermann's formula requires: (i) computing $\phi(A)$ which involves computing various powers of A and (ii) computing the last row of the inverse of the controllability matrix. The **controllability matrix is usually ill-conditioned** (see the relevant comments in **Chapter 6**). The following example illustrates the point.

Example 11.2.1. Consider EVA using Ackermann's formula with

$$A = \begin{pmatrix} -4.0190 & 5.1200 & 0 & 0 & -2.0820 \\ -0.3460 & 0.9860 & 0 & 0 & -2.3400 \\ -7.9090 & 15.4070 & -4.0690 & 0 & -6.4500 \\ -21.8160 & 35.6060 & -0.3390 & -3.8700 & -17.8000 \\ -60.1960 & 98.1880 & -7.9070 & 0.3400 & -53.0080 \\ 0 & 0 & 0 & 0 & 94.0000 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12.8000 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0.8700 \\ 0 & 0 & 0 & 0.9700 \\ 0 & 0 & 0 & 2.6800 \\ 0 & 0 & 0 & 7.3900 \\ 0 & 0 & 0 & 20.4000 \\ -147.2000 & 0 & 53.2000 & 0 \\ 94.0000 & -147.20000 & 0 & 0 \\ 12.80000 & 0 & -31.6000 & 0 \\ 0 & 0 & 18.8000 & -31.6000 \end{pmatrix}$$

$$B = \begin{pmatrix} -0.1510 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad S = \begin{pmatrix} -1.0000 \\ -1.5000 \\ -2.0000 \\ -2.5000 \\ -3.0000 \\ -3.5000 \\ -4.0000 \\ -4.5000 \\ -5.0000 \end{pmatrix}$$

The closed-loop eigenvalues assigned by the **Ackermann's formula** are:

$$\begin{pmatrix} -0.8824 & - & 0.4891j \\ -0.8824 & + & 0.4891j \\ -2.2850 & - & 1.0806j \\ -2.2850 & + & 1.0806j \\ -3.0575 & & \\ -3.8532 & & \\ -4.2637 & - & 0.7289j \\ -4.2637 & + & 0.7289j \end{pmatrix}.$$

Thus, the desired eigenvalues in S are completely different from those assigned by the Ackermann's formula. The same problem is then solved using the MATLAB function **place**, which uses the KNV algorithm. The spectrum assigned by MATLAB function **place** is: {−4.9999, −4.5001, −4.0006, −3.4999 −3.0003, −2.4984, −2.0007, −1.5004, −0.9998}.

Accurate results were also obtained by the recursive Algorithm **(Algorithm 11.2.1)** (see **Example 11.2.3**).

A Template of Numerical Algorithms for EVA Problem

A practical numerically effective algorithm has to be based upon the reduction of the controllable pair (A, B) to a canonical form pair that uses a well-conditioned transformation. As we have seen in Chapter 6, the controller-Hessenberg form is one such.

Indeed, several numerically effective algorithms have been proposed both for the single-input and the multi-input problems in recent years, based on the reduction of (A, B) to a controller-Hessenberg pair. We will describe some such algorithms in the sequel.

Most of these algorithms have a common basic structure which can be described as follows. In the following and elsewhere in this chapter, $\Omega(A)$ denotes the spectrum of A.

Step 1. The controllable pair (A, B) is first transformed to a controller-Hessenberg pair $(H, \tilde{B})$, that is, an orthogonal matrix P is constructed such that

$$PAP^{\mathrm{T}} = H, \text{ an unreduced block upper Hessenberg matrix,}$$

$$PB = \tilde{B} = \begin{pmatrix} B_1 \\ 0 \end{pmatrix}, \text{ where } B_1 \text{ is upper triangular.}$$

Note: In the single-input case, the controller-Hessenberg pair is $(H, \tilde{b})$, where H is an unreduced upper Hessenberg matrix and $\tilde{b} = Pb = \beta e_1, \beta \neq 0$.

Step 2. The EVA problem is now solved for the pair $(H, \tilde{B})$, by exploiting the special forms of H and $\tilde{B}$. This step involves finding a matrix F such that

$$\Omega(H - \tilde{B}F) = \{\lambda_1, \ldots, \lambda_n\}.$$

Note: In the single-input case, this step amounts to finding a row vector f^{T} such that $\Omega(H - \beta e_1 f^{\mathrm{T}}) = \{\lambda_1, \ldots, \lambda_n\}$.

Step 3. A feedback matrix K of the original problem is retrieved from the feedback matrix F of the transformed Hessenberg problem by using an orthogonal matrix multiplication: $K = FP$. (Note that $\Omega(H - \tilde{B}F) = \Omega(PAP^{\mathrm{T}} - PBFPP^{\mathrm{T}}) = \Omega(P(A - BK)P^{\mathrm{T}}) = \Omega(A - BK)$.)

The different algorithms differ in the way Step 2 is implemented. In describing the algorithms below, we will assume that Step 1 has already been implemented using the numerically stable Staircase Algorithm described in Chapter 6 (**Section 6.8**).

11.2.1 A Recursive Algorithm for the Single-Input EVA Problem

In this subsection, we present a simple recursive scheme (Datta 1987) for the single-input Hessenberg EVA problem.

Let's first remind the readers of the statement of the **single-input Hessenberg EVA problem:**

> Given an unreduced upper Hessenberg matrix H, the number $\beta \neq 0$, and the set $S = \{\lambda_1, \ldots, \lambda_n\}$, closed under complex conjugation, find a row vector f^T such that
>
> $$\Omega(H - \beta e_1 f^T) = \{\lambda_1, \ldots, \lambda_n\}.$$

We will assume temporarily, without any loss of generality, that $\beta = 1$ (recall that $Pb = \tilde{b} = \beta e_1$.)

Formulation of the Algorithm

The single-input EVA problem will have a solution if the closed-loop matrix $(H - e_1 f^T)$ can be made similar to a matrix whose eigenvalues are the same as the ones to be assigned.

Thus, the basic idea here is to construct a nonsingular matrix X such that

$$X(H - e_1 f^T)X^{-1} = \Lambda, \tag{11.2.1}$$

where $\Omega(\Lambda) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$.

From (11.2.1), we have

$$XH - \Lambda X = X e_1 f^T. \tag{11.2.2}$$

Taking the transpose, the above equation becomes

$$H^T X^T - X^T \Lambda^T = f e_1^T X^T. \tag{11.2.3}$$

Setting $X^T = L$, Eq. (11.2.3) becomes

$$H^T L - L\Lambda^T = f e_1^T L. \tag{11.2.4}$$

The problem at hand now is to construct a nonsingular matrix L and a vector f such that Eq. (11.2.4) is satisfied. We show below how some special choices make it happen.

Let's choose

$$\Lambda^{\mathrm{T}} = \begin{pmatrix} \lambda_1 & & & & \\ * & \ddots & & 0 & \\ & \ddots & \ddots & & \\ & 0 & \ddots & \ddots & \\ & & & * & \lambda_n \end{pmatrix}$$

and let $e_1^{\mathrm{T}} L$ be chosen such that all but the last column of the matrix on the right-hand side of (11.2.4) are zero, that is, the matrix Eq. (11.2.4) becomes

$$H^{\mathrm{T}} L - L\Lambda^{\mathrm{T}} = (0, 0, \ldots, \alpha f), \quad \alpha \neq 0. \tag{11.2.5}$$

The simple form of the right-hand side of (11.2.5) allows us to compute recursively the second through nth column of $L = (l_1, l_2, \ldots, l_n)$, once the first column l_1 is known. The entries of the subdiagonal of Λ can be chosen as scaling factors for the computed columns of L. Once L is known, αf can be computed by equating the last column of both sides of (11.2.5):

$$\alpha f = (H^{\mathrm{T}} - \lambda_n I) l_n. \tag{11.2.6}$$

What now remains to be shown is that how to choose l_1 such that the resulting matrix L in (11.2.5) is nonsingular.

A theorem of K. Datta (1988) tells us that if l_1 is chosen such that (H^{T}, l_1) is controllable, then L satisfying (11.2.5) will be nonsingular. Since H^{T} is an unreduced upper Hessenberg matrix, the simplest choice is $l_1 = e_n = (0, 0, \ldots, 0, 1)^{\mathrm{T}}$. It is easy to see that this choice of l_1 will yield $\alpha = l_{1n}$, the first entry of l_n. Then equating the last column of both sides of (11.2.5), we have

$$f = \frac{(H^{\mathrm{T}} - \lambda_n I) l_n}{\alpha} = \frac{(H^{\mathrm{T}} - \lambda_n I) l_n}{l_{1n}}.$$

The above discussion immediately leads us to the following algorithm:

Algorithm 11.2.1. *The Recursive Algorithm for the Single-input Hessenberg EVA Problem*

Inputs. *H, an unreduced upper Hessenberg matrix, and $S = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, a set of n numbers, closed under complex conjugation.*

Output. *The feedback vector f such that $\Omega(H - e_1 f^{\mathrm{T}}) = S$.*

Step 1. *Set* $l_1 = e_n$, *the last column of the* $n \times n$ *identity matrix*

Step 2. *Construct a set of normalized vectors* $\{\ell_k\}$ *as follows:*

For $i = 1, 2, \ldots, n-1$ *do*

Compute $\hat{\ell}_{i+1} = (H^{\mathrm{T}} - \lambda_i I)\ell_i$

$$\ell_{i+1} = \frac{\hat{\ell}_{i+1}}{\|\hat{\ell}_{i+1}\|_2}$$

End

Step 3. *Compute* $\ell_{n+1} = (H^{\mathrm{T}} - \lambda_n I)\ell_n$.

Step 4. *Compute* $f = \dfrac{\ell_{n+1}}{\alpha}$, *where* α *is the first entry of* ℓ_n.

Theorem 11.2.1. *The vector* f *computed by Algorithm* 11.2.1 *is such that the eigenvalues of the closed-loop matrix* $(H - e_1 f^{\mathrm{T}})$ *are* $\lambda_1, \ldots, \lambda_n$.

Proof. Proof follows from the above discussions. ■

Flop-count: Since l_i contains only i nonzero entries and H is an unreduced Hessenberg matrix, computations of l_2 through l_n in Algorithm 11.2.1 takes about $n^3/3$ flops. Furthermore, with these operations, one gets the transforming matrix L that transforms the closed-loop matrix to Λ by similarity. Also, it takes about $\frac{10}{3}n^3$ flops for the single-input controller-Hessenberg reduction. So, the flop-count for the EVA problem for the pair (A, b) using Algorithm 11.2.1 is about $\frac{11}{3}n^3$ flops.

Avoiding complex arithmetic: When the eigenvalues to be assigned are complex, the use of complex arithmetic in Algorithm 11.2.1 can be avoided by setting Λ as a matrix in RSF, having a 2×2 block corresponding to each pair of complex conjugate eigenvalues to be assigned. Algorithm 11.2.1 needs to be modified accordingly **[Exercise 11.1]**.

MATCONTROL note: The modified algorithm that avoids the use of complex arithmetic has been implemented in MATCONTROL function **polercs**.

Example 11.2.2. Consider EVA using Algorithm 11.2.1 with

$$H = \begin{pmatrix} 9 & 4 & 7 \\ 3 & 1 & 2 \\ 0 & 9 & 6 \end{pmatrix}, \qquad S = \{9 \;\; 5 \;\; 1\}.$$

Step 1. $l_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

Step 2.
$i = 1$

$$\hat{l}_2 = \begin{pmatrix} 0 \\ 9 \\ -3 \end{pmatrix}, \qquad l_2 = \begin{pmatrix} 0 \\ 0.9487 \\ -0.3162 \end{pmatrix}.$$

$i = 2$

$$\hat{l}_3 = \begin{pmatrix} 2.8460 \\ -6.6408 \\ 1.5811 \end{pmatrix}, \qquad l_3 = \begin{pmatrix} 0.3848 \\ -0.8979 \\ 0.2138 \end{pmatrix}$$

Step 3.

$$l_4 = \begin{pmatrix} 0.3848 \\ 3.4633 \\ 1.9668 \end{pmatrix}.$$

Step 4.

$$f = \begin{pmatrix} 1.0000 \\ 9.0000 \\ 5.1111 \end{pmatrix}.$$

The closed-loop matrix:

$$H - e_1 f^{\mathrm{T}} = \begin{pmatrix} 8.0 & -5.0 & 1.8889 \\ 3.0 & 1.0 & 2.0 \\ 0 & 9.0 & 6.0 \end{pmatrix}.$$

Verify: The eigenvalues of the matrix $(H - e_1 f^{\mathrm{T}})$ are 9, 5, and 1.

Example 11.2.3. Let's apply Algorithm 11.2.1 to Example 11.2.1 again. The eigenvalues of the closed-loop matrix $H - e_1 f^{\mathrm{T}}$ with the vector f computed by Algorithm 11.2.1 are:

$-5.0001, -4.4996, -4.0009, -3.4981, -3.0034, -2.4958, -2.0031, -1.4988,$
$-1.0002.$

The computed closed-loop eigenvalues, thus, have the similar accuracy as those obtained by the MATLAB function **place**.

Example 11.2.4. *Eigenvalue Assignment with Ill-Conditioned Eigenvalues.* Since the matrix H and the closed-loop matrix $H - e_1 f^{\mathrm{T}}$ differ only by the first row, Algorithm 11.2.1 amounts to finding a vector f such that, when the first row of the given Hessenberg matrix H is replaced by f^{T}, the resulting new Hessenberg matrix

has the prescribed spectrum. Let H be the well-known Wilkinson bidiagonal matrix (see Datta (1995), Wilkinson (1965)) with highly ill-conditioned eigenvalues:

$$H = \begin{pmatrix} 20 & & & & & \\ & 19 & & & & \\ 20 & & \ddots & & 0 & \\ & 20 & & \ddots & & \\ 0 & & \ddots & & \ddots & \\ & & & 20 & & 1 \end{pmatrix}.$$

First, the first row of H is replaced by the zero row vector and then Algorithm 11.2.1 is run on this new H with $S = \{20, 19, 18, \ldots, 1\}$, and f is computed. Since the eigenvalues of the original matrix H is the set S; in theory, f^{T} should be the same as the first row of H; namely, $f^{\mathrm{T}} = (20, 0, \ldots, 0)$. Indeed, the vector f^{T} computed by the algorithm is found to be $f^{\mathrm{T}} = (20, 0, \ldots, 0)$ and the eigenvalues of the closed-loop matrix with this computed f are $1, 2, 3, \ldots, 20$.

A closed-form solution of the feedback vector in the single-input EVA problem: We now show that Algorithm 11.2.1 yields an explicit closed-form solution for the single-input problem. The recursion in Step 2 of the algorithm yields

$$\gamma\, l_{i+1} = (H^{\mathrm{T}} - \lambda_1 I)(H^{\mathrm{T}} - \lambda_2 I) \cdots (H^{\mathrm{T}} - \lambda_i I) l_1, \tag{11.2.7}$$

for some (nonzero) scalar γ. Including Steps 1 and 4, (11.2.6) becomes

$$\alpha f = (H^{\mathrm{T}} - \lambda_1 I)(H^{\mathrm{T}} - \lambda_2 I) \cdots (H^{\mathrm{T}} - \lambda_n I) e_n, \tag{11.2.8}$$

where $\alpha = (h_{21} h_{32} \cdots h_{n,n-1})^{-1}$. If $\phi(x) = (x - \lambda_1)(x - \lambda_2) \cdots (x - \lambda_n)$, then this can be written as

$$f^{\mathrm{T}} = \alpha e_n^{\mathrm{T}} \phi(H). \tag{11.2.9}$$

Since this solution is unique, it **represents *the* Hessenberg representation of the Ackermann formula for the single-input EVA problem.**

11.2.2 An Error Analysis of the Recursive Single-Input Method

Knowing the above explicit expression for the feedback vector f, we can now carry out a forward error analysis of Algorithm 11.2.1. This is presented below.

By duality of (11.2.4), we see that the method computes a matrix L and a vector f such that

$$HL - L\Lambda = \alpha f e_1^{\mathrm{T}} L.$$

A careful look at the iteration reveals that the forward error has a special form. Define the polynomials $\phi_{j,k}$ for $j \leq k$ by

$$\phi_{j,k}(x) = (x - \lambda_j)(x - \lambda_{j+1}) \cdots (x - \lambda_k).$$

Let $\bar{l}_i$ be the computed value of the ith column of L. Define ϵ_i by

$$\bar{l}_{i+1} = (H - \lambda_i I)\bar{l}_i + \epsilon_i. \tag{11.2.10}$$

Then we have the following **forward error** formula, due to Arnold (1993) (See also Arnold and Datta (1998)).

Theorem 11.2.2. *Let $\bar{\alpha}\bar{f}$ be the* computed *feedback vector of the single-input EVA problem for (H, e_1) using Algorithm* 11.2.1. *If αf is the exact solution, then*

$$\bar{\alpha}\bar{f} - \alpha f = \sum_{j=1}^{n} \phi_{j,n}(H)\epsilon_j,$$

where ϵ_js are defined above.

Unfortunately, not much can be said about backward stability from a result like this. It is, however, possible to shed some light on the stability of this method by looking at ϵ_j in a different way. See Arnold and Datta (1998) for details.

Theorem 11.2.3. *Let $E = [\epsilon_1, \epsilon_2, \ldots, \epsilon_n]$ and let $\bar{L} = [\bar{\ell}_1, \bar{\ell}_2, \ldots, \bar{\ell}_n]$. Then $\bar{\alpha}\bar{f}$ solves (exactly) the single-input EVA problem for the perturbed system $(H - E\bar{L}^{-1}, \beta e_1, S)$, where the ϵ_i are the same as in the previous theorem, that is, the computed vector $\bar{f}$ is such that*

$$\Omega[(H - E\bar{L}^{-1}) - \beta e_1 \bar{f}^{\mathrm{T}}] = \{\lambda_1, \ldots, \lambda_n\}.$$

Proof. Notice that as defined, $\bar{L}$ satisfies the matrix equation:

$$H\bar{L} - \bar{L}\Lambda = E + \bar{\alpha}\bar{f}e_n^{\mathrm{T}},$$

where $\Lambda = \mathrm{diag}(\lambda_i)$. Since $\bar{L}$ is nonsingular by construction, we can solve the perturbed equation:

$$(H + \Delta H)\bar{L} - \bar{L}\Lambda = E + \bar{\alpha}\bar{f}e_n^{\mathrm{T}}$$

for ΔH. This yields $-\Delta H = E\bar{L}^{-1}$, and $\bar{\alpha}\bar{f}$ solves the EVA problem for $(H + \Delta H, \beta e_1, S)$. ■

Remarks on Stability and Reliability

- From the above result it cannot be said that the method is backward stable. The result simply provides an upper bound on the size of the ball around the initial data, inside which there exist $(H + \Delta H, \beta + \delta\beta)$ for which the computed solution is exact. If $\|\Delta H\|$ could be bounded above by a small quantity that was relatively independent of the initial data, then the method would be backward stable. However, Theorem 11.2.3 does allow one to say precisely when the results from the method are suspect. It is clear that $\|E\|$ is always small if the iterates are normalized every few steps, so that all of the backward error information is contained in $\bar{L}^{-1}$. **Thus, the stability of the algorithm can be monitored by monitoring the condition number of $\bar{L}$. Furthermore, since $\bar{L}$ is triangular, it is possible to estimate $\|\bar{L}^{-1}\|$ rather cheaply, even as the iteration proceeds.** (See Higham 1996).

  The matrix L gives us even more information about the EVA problem at hand. In case the eigenvalues to be assigned are distinct, an upper bound in the condition number of the matrix of eigenvectors that diagonalizes the closed loop matrix can be obtained from the condition number of the matrix L.

  This is important because, as said in the introduction, the condition number of the matrix of eigenvectors of the closed-loop matrix is an important factor in the accuracy of the assigned eigenvalues (see Section 11.5 and the Example therein).

  Specifically, if X diagonalizes Λ, then it can be shown (Arnold and Datta 1998), that $P = (\bar{L})^{-1}X$ diagonalizes the closed-loop matrix $H - e_1 f^{\mathrm{T}}$, furthermore,

$$\mathrm{Cond}_2(P) \leq \mathrm{Cond}_2(X)\mathrm{Cond}_2(L).$$

- **Computational experience has shown that if L is ill-conditioned, then so are the closed-loop eigenvalues.**

11.2.3 The QR and RQ Implementations of Algorithm 11.2.1

Algorithm 11.2.1 is an extremely efficient way to solve the Hessenberg single-input EVA problem, but as we have just seen, the backward stability of this algorithm cannot be guaranteed. It, however, turns out that there is a **numerically stable implementation** of this algorithm via QR iterations. We will discuss this below.

The QR Implementation of Algorithm 11.2.1

The idea of using QR iterations in implementing Algorithm 11.2.1 comes from the fact that the matrix $\phi(H)$ in the explicit expression of f in (11.2.9) can be written

as (Stewart 1973, p. 353):

$$\phi(H) = (H - \lambda_2 I)(H - \lambda_2 I)\cdots(H - \lambda_n I) = Q_1 Q_2 \cdots Q_n R_n R_{n-1} \cdots R_1,$$

where Q_i and R_i are generated by n steps of QR iterations as follows:

$H_1 = H$
For $i = 1, 2, \ldots, n$ do
    $Q_i R_i = H_i - \lambda_i I$
    $H_{i+1} = R_i Q_i + \lambda_i I$
End.

Remark

- Note that since H_i is Hessenberg, so is H_{i+1}, for each i (see **Chapter 4**).

MATCONTROL note: The QR version of Algorithm 11.2.1 has been implemented in MATCONTROL function **poleqrs**.

The RQ Implementation of Algorithm 11.2.1

The difficulty of implementing the QR strategy is that the R_i need to be accumulated; the process is both expensive and unstable.

We now show how the method can be made computationally efficient by using **RQ factorizations** instead of QR factorizations, as follows:
Set $H_1 = H$
For $i = 1, 2, \ldots, n$ compute the RQ step

$$R_i Q_i = H_i - \lambda_i I$$
$$H_{i+1} = Q_i R_i + \lambda_i I$$

This time

$$\phi(H) = R_1 R_2 \cdots R_n Q_n Q_{n-1} \cdots Q_1, \tag{11.2.11}$$

and by setting $Q = Q_n Q_{n-1} \cdots Q_1$ and $R = R_1 R_2 \cdots R_n$, we have from (11.2.9)

$$f^{\mathrm{T}} = \alpha e_n^{\mathrm{T}} R Q = \alpha \rho e_n^{\mathrm{T}} Q. \tag{11.2.12}$$

Here $\rho = \Pi_{i=1}^{n} r_{nn}^{(i)}$, where $r_{nn}^{(i)}$ denotes the (n, n)th entry of R_i. *This is a much nicer situation!* Thus, a straightforward RQ implementation of Algorithm 11.2.1 will be as follows:

Algorithm 11.2.2. *An RQ Implementation of Algorithm* 11.2.1
Inputs. *Same as in Algorithm* 11.2.1.
Output. *Same as in Algorithm* 11.2.1.
Step 0. *Set* $H_1 = H$.

Step 1. *For* $i = 1, 2, \ldots, n$ *do*
$R_i Q_i = H_i - \lambda_i I$
$H_{i+1} = Q_i R_i + \lambda_i I$
End

Step 2. *Compute* $f = \alpha \rho e_n^{\mathrm{T}} Q_n Q_{n-1} \cdots Q_1$, *where* $\rho = \Pi_{i=1}^{n} r_{nn}^{(i)}$, $r_{nn}^{(i)}$ *denotes the* (n, n)*th entry of* R_i.

Algorithm 11.2.2 may be made storage-efficient by observing that it is possible to **deflate** the problem at each RQ step, as follows:

$$H_{i+1} = Q_i R_i + \lambda_i I = \begin{pmatrix} * & * \\ 0 & \widetilde{H}_{i+1} \end{pmatrix}.$$

The matrix $\widetilde{H}_{i+1}$ can now be set as H_{i+1} and the iteration can be continued with $H_{i+1} \equiv \widetilde{H}_{i+1}$ after updating Q_i and ρ appropriately. Thus, algorithmically we have the **following storage-efficient version of Algorithm 11.2.2, which is recommended to be used in practice.**

Algorithm 11.2.3. *A Storage-Efficient Version of Algorithm 11.2.2*
Inputs. *Same as in Algorithm* 11.2.1
Output. *Same as in Algorithm* 11.2.1.

Step 0. *Set* $H_1 = H$.

Step 1. *Compute the RQ factorization of* $H_1 - \lambda_1 I$; *that is, compute* Q_1 *and* R_1 *such that* $(H - \lambda_1 I) Q_1^{\mathrm{T}} = R_1$. *Compute* $H_2 = Q_1 R_1 + \lambda_1 I$. *Set* $Q = Q_1$ *and* $\rho = r_{nn}^{(1)}$, *where* $R_1 = (r_{ij}^{(1)})$.

Step 2. *For* $i = 2, 3, \ldots, n$ *do*
Compute the RQ factorization of $H_i - \lambda_i I : (H_i - \lambda_i I) Q_i^{\mathrm{T}} = R_i$. *Compute* H_{i+1}, *where* $Q_i R_i + \lambda_i I = \begin{pmatrix} * & * \\ 0 & H_{i+1} \end{pmatrix}$. *Update* $Q \equiv \begin{pmatrix} I & \\ & Q_i \end{pmatrix} Q$, *where* I *is a matrix consisting of the first* $(i-2)$ *rows and columns of the identity matrix. Update* $\rho \equiv \rho\, r_{n+2-i,n+2-i}^{(i)}$ ($r_{n+2-i,n+2-i}^{(i)}$ is the last element of R_i).
End

Step 3. *Compute* $f^{\mathrm{T}} = \alpha' \rho e_n^{\mathrm{T}} Q$, *where* $\alpha' = 1/(h_{21} \cdots h_{n,n-1})$.

Flop-count and numerical stability: Algorithm 11.2.3 requires about $\frac{5}{3}n^3$ flops. Since reduction to the controller-Hessenberg form requires $\frac{10}{3}n^3$ flops, the total count for EVA of the pair (A, b) using Algorithm 11.2.3 is about $5n^3$.

The algorithm is *numerically stable* (see Arnold and Datta 1998). Specifically, the method computes, given a controllable pair (H, e_1), a vector $\bar{f}$ such that it solves exactly the EVA problem for the system with the matrix $H + \Delta H$, where

$$\|\Delta H\|_F \leq \mu g(n)\|H\|_F,$$

in which μ is the machine precision and $g(n)$ is a modest function of n.

Remark

- It can be shown (Arnold 1993) that if an EVA algorithm for the single-input Hessenberg problem is backward stable, then the algorithm is also backward stable for the original problem.

Thus, the RQ implementation of Algorithm 11.2.1 is backward stable for the original problem. That is, the feedback $\bar{k}$, computed by Algorithm 11.2.3, for the problem (A, b) is exact for a nearby problem: $\bar{k}$ exactly solves the EVA problem for $(A + \Delta A,\ b + \delta b)$, where ΔA and δb are small. For a proof of this backward stability result, see Arnold and Datta (1998).

Example 11.2.5. Consider Example 11.2.2 again

Step 0.

$$H_1 = H = \begin{pmatrix} 9 & 4 & 7 \\ 3 & 1 & 2 \\ 0 & 9 & 6 \end{pmatrix}, \qquad S = \{\lambda_1, \lambda_2, \lambda_3\} = \{9, 5, 1\}.$$

Step1. Compute R_1 and Q_1 such that $H_1 - \lambda_1 I = R_1 Q_1$: $[R_1, Q_1] = \boldsymbol{rq}\,(H_1 - \lambda_1 I)$.

$$\text{Compute } H_2 = Q_1 R_1 + \lambda_1 I = \begin{pmatrix} 10.5957 & -0.6123 & -6.5885 \\ -7.5693 & 7.2043 & 0.2269 \\ 0 & 2.9086 & -1.8000 \end{pmatrix}.$$

$$Q \equiv Q_1 = \begin{pmatrix} -0.2063 & 0.3094 & 0.9283 \\ 0.9785 & -0.0652 & -0.1957 \\ 0 & 0.9487 & -0.3162 \end{pmatrix}, \qquad \rho = 9.4868$$

Step 2. $i = 2$

Compute R_2 and Q_2 such that $H_2 - \lambda_2 I = R_2 Q_2$: $[R_2, Q_2] = \boldsymbol{rq}\ (H_2 - \lambda_2 I)$.

$$H_3 = \begin{pmatrix} 12.9533 & 4.6561 \\ -3.0909 & -1.5411 \end{pmatrix}, \quad \text{Update } Q\text{: } Q \equiv \begin{pmatrix} -0.8109 & -0.2183 & 0.5430 \\ -0.4409 & -0.3823 & -0.8121 \\ 0.3848 & -0.8479 & 0.2138 \end{pmatrix}.$$

$$\text{Update } \rho\text{: } \rho \equiv 70.1641$$

$i = 3$

Compute R_3 and Q_3 such that $H_3 - \lambda_3 I = R_3 Q_3$: $[R_3, Q_3] = rq(H_3 - \lambda_3 I)$.

$$R_3 = \begin{pmatrix} -3.9945 & -12.1904 \\ 0 & 4.0014 \end{pmatrix}, \qquad Q_3 = \begin{pmatrix} -0.6351 & 0.7725 \\ -0.7725 & -0.6351 \end{pmatrix}.$$

$H_4 = 7.8755$,

$$\text{Update } Q\text{: } Q \equiv \begin{pmatrix} -0.8109 & -0.2183 & 0.5430 \\ 0.5772 & -0.4508 & 0.6809 \\ 0.0962 & 0.8655 & 0.4915 \end{pmatrix}, \qquad \text{update } \rho \equiv 280.7526.$$

Step 3.

$$f = \begin{pmatrix} 1.0000 \\ 9.0000 \\ 5.1111 \end{pmatrix}.$$

Verify:

$$H - e_1 f^{\mathrm{T}} = \begin{pmatrix} 8.0000 & -5.0000 & 1.8889 \\ 3.0000 & 1.0000 & 2.0000 \\ 0 & 9.0000 & 6.0000 \end{pmatrix}.$$

The eigenvalues of $H - e_1 f^{\mathrm{T}}$ are $\{5, 1, 9\}$.

MATCONTROL note: Algorithm 11.2.3 has been implemented in MATCONTROL function **polerqs**.

11.2.4 Explicit and Implicit RQ Algorithms

We have just seen the QR and RQ versions of Algorithm 11.2.1. At least two more QR type methods were proposed earlier: An **explicit QR algorithm** by Miminis and Paige (1982) and an **implicit QR** algorithm by Patel and Misra (1984).

The explicit QR algorithm, proposed by Miminis and Paige (1982), constructs an orthogonal matrix Q such that

$$Q^{\mathrm{T}}(H - \beta e_1 f^{\mathrm{T}})Q = R,$$

where R is an upper triangular matrix with the desired eigenvalues $\lambda_1, \ldots, \lambda_n$ on the diagonal. The algorithm has a forward sweep and a backward sweep. The forward sweep determines Q and the backward sweep finds f and R, if needed.

The algorithm explicitly uses the eigenvalues to be assigned as shifts and that is why it is called an **explicit QR algorithm**.

It should come as no surprise that **an implicit** RQ step is possible, and in order to handle **complex pairs of eigenvalues with real arithmetic**, an implicit double step is needed. One such method has been proposed by Patel and Misra (1984). The Patel–Misra method is similar to the Miminis–Paige method, but it includes an alternative to the "backward sweep." There now exist RQ formulations of both these algorithms (Arnold 1993; Arnold and Datta 1998). *These RQ formulations are easier to describe, understand, and implement.*

It should be mentioned here that there now exists a generalization of the implicit QR algorithm due to Varga (1996) that performs an implicit multistep in place of a double-step. The Varga algorithm is slightly more efficient than the Patel–Misra algorithm and like the latter, is believed to be numerically stable.

Methods Not Discussed

Besides the methods discussed above, there are many other methods for the single-input problem. These include the methods based on solutions of independent linear systems (Datta and Datta 1986; Bru *et al.* 1994a); the eigenvector method by Petkov *et al.* (1984), etc., parallel algorithms by Coutinho *et al.* (1995), and by Bru *et al.* (1994c), etc.; and the multishift algorithm by Varga (1996). See **Exercises 11.2–11.4 and 11.8** for statements of some of these methods.

11.3 NUMERICAL METHODS FOR THE MULTI-INPUT EIGENVALUE ASSIGNMENT PROBLEM

Some of the single-input algorithms described in the last section have been generalized in a straightforward fashion to the multi-input case or similar algorithms have been developed for the latter.

We describe here:

- A multi-input generalization of the single-input recursive algorithm (Arnold and Datta 1990).
- An explicit QR algorithm (Miminis and Paige 1988).
- A Schur method (Varga 1981).
- A Sylvester equation algorithm for PEVA algorithm (Datta and Sarkissian 2002).

There are many more algorithms for this problem that are not described here. Some of them are: a multi-input generalization of the single-input eigenvector algorithm by Petkov *et al.* (1986), a multi-input generalization of the

single-input algorithm using solutions of linear systems by Bru *et al.* (1994b) **(Exercise 11.5)**, a matrix equation algorithm by Bhattacharyya and DeSouza (1982) **(Exercise 11.14)**, and a multi-input version of the single-input implicit QR algorithm by Patel and Misra (1984), algorithms by Tsui (1986) and Shafai and Bhattacharyya (1988), and parallel algorithms by Baksi *et al.* (1994), Datta (1989), etc.

11.3.1 A Recursive Multi-Input Eigenvalue Assignment Algorithm

The following algorithm is a generalization of the single-input recursive algorithm (**Algorithm 11.2.1**) to the multi-input case.

The development of this algorithm is along the same line as Algorithm 11.2.1.

The version of the algorithm presented here is a little different than that originally proposed in Arnold and Datta (1990).

Given a controller-Hessenberg pair $(H, \tilde{B})$ and the set $S = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, the algorithm, like its single-input version, constructs a nonsingular matrix L recursively from where the feedback matrix F can be easily computed. Since in the multi-input case the matrix H of the controller-Hessenberg form is a block-Hessenberg matrix, by taking advantage of the block form of H, this time the matrix L can be computed in blocks. The matrix L can be computed either block column-wise or block row-wise. We compute L block row-wise here starting with the last block row.

Thus, setting

$$\Lambda = \begin{pmatrix} \Lambda_{11} & & & \\ \Lambda_{21} & \Lambda_{22} & & 0 \\ & \ddots & \ddots & \\ 0 & & \Lambda_{k,k-1} & \Lambda_{kk} \end{pmatrix},$$

where the eigenvalues $\lambda_1, \ldots, \lambda_n$ are contained in the diagonal blocks of Λ, and considering the equation:

$$LH - \Lambda L = L \begin{pmatrix} R \\ 0 \end{pmatrix} F,$$

it is easily seen that the matrices L and R can be found without knowing the matrix F. Indeed, the matrix

$$L = \begin{pmatrix} L_1 \\ L_2 \\ \vdots \\ L_k \end{pmatrix}$$

can be computed recursively block row-wise starting with L_k and if L_k is chosen as $L_k = (0, 0, \ldots 0, I_{n_k})$, then L will be nonsingular. Equating the corresponding block-rows of the equation:

$$LH - \Lambda H = \begin{pmatrix} R \\ 0 \end{pmatrix} F,$$

it is easy to see that

$$\Lambda_{i+1,i} L_i = L_{i+1} H - \Lambda_{i+1,i+1} L_{i+1} = \tilde{L}_i, \quad i = k-1, k-2, \ldots, 2, 1,$$

from where the matrices $\Lambda_{i+1,i}$ and L_i can be computed by using the QR factorization of $\tilde{L}_i$. Once L and R are found, the matrix F can be computed from the above equation by solving a block linear system. Overall, we have the following algorithm.

Algorithm 11.3.1. *The Recursive Algorithm for the Multi-Input EVA Problem*
Inputs.
A—The $n \times n$ state matrix.
B—The $n \times m$ input matrix ($m \leq n$).
S—The set of numbers $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, closed under complex conjugation.
Assumption. *(A, B) is controllable.*
Output. *A feedback matrix K such that $\Omega(A - BK) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$.*

Step 1. *Using the Staircase Algorithm in Section 6.7, reduce the pair (A, B) to the controller-Hessenberg pair $(H, \tilde{B})$, that is, find an orthogonal matrix P such that $PAP^{\mathrm{T}} = H$, an unreduced block upper Hessenberg matrix with k diagonal blocks and*

$$PB = \tilde{B} = \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R \text{ is upper triangular and has full rank.}$$

Step 2. *Partition S in such a way that $S = \cup\Omega(\Lambda_{ii})$, where each Λ_{ii} is an $n_i \times n_i$ diagonal matrix (Recall that n_is are defined by the dimensions of the blocks in $H = (H_{ij})$; $H_{ij} \in \mathbb{R}^{n_i \times n_j}$).*

Step 3. *Set $L_k = (0, \ldots, 0, I_{n_k})$.*
Step 4. *For $i = k-1, \ldots, 1$ do*
4.1. *Compute $\tilde{L}_i \equiv L_{i+1} H - \Lambda_{i+1,i+1} L_{i+1}$*
4.2. *Compute the QR decomposition of $\tilde{L}_i^{\mathrm{T}} : \tilde{L}_i^{\mathrm{T}} = QR$*
4.3. *Define $L_i = Q_1^{\mathrm{T}}$, where Q_1 are the first n_i columns of the matrix $Q = (Q_1, Q_2)$*
End

Step 5. *Solve the linear system* $(L_{11}R)F = L_1H - \Lambda_{11}L_1$ *for* F, *where* L_{11} *is the matrix of the first* n_1 *columns of* L_1.

Step 6. *Compute the feedback matrix* K *of the original problem:* $K \equiv FP$.

Theorem 11.3.1. *The feedback matrix* K *constructed by the above algorithm is such that*

$$\Omega(A - BK) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}.$$

Proof. Proof follows from the discussion preceding the algorithm. ■

Flop-count: Approximately $\frac{19}{3}n^3 + \frac{15}{2}n^2m$ flops are required to implement the algorithm.

It may be worth noting a few points regarding the complexity of this algorithm. First, the given operations count includes assigning complex eigenvalues using real arithmetic. Second, almost 95% of the total flops required for this method are in the reduction to the controller-Hessenberg form in Step 1. Finally, within the above operations count (but with some obvious additional storage requirements), the matrix L that transforms the reduced closed-loop system to the block bidiagonal matrix Δ by similarly, can be obtained.

Avoiding complex arithmetic: In order to assign a pair of complex conjugate eigenvalues using only real arithmetic, we set 2×2 "**Schur bumps**" on the otherwise diagonal Λ_{ii}. For example, if we want to assign the eigenvalues $x \pm iy$ to $A - BK$, we might set

$$\Lambda_3 = \begin{bmatrix} x & -y \\ y & x \end{bmatrix}.$$

However, the algorithm might give a complex feedback matrix if all the complex conjugate pairs cannot be distributed as above along the diagonal blocks Λ_{ii}. Some modifications of the algorithm in that case will be necessary. A *block algorithm that avoids complex feedback has been recently proposed by Carvalho and Datta* (2001).

MATCONTROL note: The modified version of Algorithm 11.3.1, proposed in Carvalho and Datta (2001), that avoids the use of complex arithmetic and is guaranteed to give a real feedback matrix has been implemented in MATCONTROL function **polercx**, while Algorithm 11.3.1 as it appears here has been implemented in MATCONTROL function **polercm**.

Example 11.3.1. Consider EVA using Algorithm 11.3.1 with

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

$$B = \tilde{B} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} R \\ 0 \end{pmatrix}, \qquad S = \{1, 2, 3, 4, 5\}.$$

Here $k = 3$, $n_1 = 3$, $n_2 = 1$, and $n_3 = 1$.

Step 1. The pair $(H, \tilde{B})$ is already in controller-Hessenberg form
Step 2. $\Lambda_{11} = \text{diag}(1, 2, 3)$, $\Lambda_{22} = 4$, $\Lambda_{33} = 5$.
Step 3. $L_3 = (0, 0, 0, 0, 1)$.
Step 4. $i = 2$
4.1: $\tilde{L}_2 = (0 \;\; 0 \;\; 0 \;\; 1 \;\; -4)$
4.2: (not shown)
4.3: $L_2 = (0 \;\; 0 \;\; 0 \;\; -0.2425 \;\; 0.9701)$
$i = 1$
4.1: $\tilde{L}_1 = (0 \;\; 0 \;\; -0.2425 \;\; 1.6977 \;\; -3.3955)$
4.2: (not shown)
4.3:

$$L_1 = \begin{pmatrix} 0 & 0 & 0.0638 & -0.4463 & 0.8926 \\ 0 & 1.0000 & 0 & 0 & 0 \\ 0.0638 & 0 & 0.9959 & 0.0285 & -0.0569 \end{pmatrix}.$$

Step 5.

$$F = \begin{pmatrix} -2.3333 & 3.3333 & 78.2161 & -212.4740 & 217.0333 \\ -0.3333 & -1.6667 & 5.6667 & -9.0000 & 9.6667 \\ 0.6667 & 0.3333 & -2.3333 & 5.0000 & -4.3333 \end{pmatrix}.$$

Verify: The eigenvalues of $H - \tilde{B}F$ are: {1.0000 2.0000 3.0000 4.0000 5.0000}.

11.3.2 The Explicit *QR* Algorithm for the Multi-Input EVA Problem

The following multi-input QR algorithm due to Miminis and Paige (1988) also follows the same "template" as that of the preceding algorithm. The algorithm consists of the following three major steps.

Step 1. The controllable pair (A, B) is transformed to the controller-Hessenberg pair $(H, \tilde{B})$:

$$PAP^{\mathrm{T}} = H = \begin{pmatrix} H_{11} & & & \\ H_{21} & \ddots & & H_{ij} \\ & \ddots & \ddots & \\ 0 & & H_{k,k-1} & H_{kk} \end{pmatrix} \quad \text{and} \quad PBU = \tilde{B} = \begin{pmatrix} B_{11} \\ 0 \end{pmatrix}.$$

The matrix B_{11} and the subdiagonal blocks in H are of the form $(0, R)$, where R is a nonsingular and upper triangular matrix.

Step 2. An orthogonal matrix Q and a feedback matrix F are constructed such that $\Omega(Q^{\mathrm{T}}(H - \tilde{B}F)Q) = \{\lambda_1, \ldots, \lambda_n\}$.

Step 3. A feedback matrix K of the original problem is recovered from the feedback matrix F of the Hessenberg problem in Step 2 as follows:

$$K = UFP.$$

Step 1 can be implemented using the Staircase Algorithm for the controller-Hessenberg form described in Chapter 6.

We therefore concentrate on Step 2, assuming that Step 1 has already been performed.

Let $n_1 =$ dimensions of H_{11} and $n_i = \operatorname{rank}(H_{i,i-1})$, $i = 2, 3, \ldots, k$. Assume also that B_{11} has n_1 columns.

We consider two cases. The algorithm comprises of implementing these two cases as the situations warrant. The feedback matrix F is obtained by accumulating feedback matrices from the individual cases.

Case 1. If $m_1 = n_1 - n_2 > 0$, that is, if $n_1 > n_2$, we can immediately allocate $m_1 = n_1 - n_2$ eigenvalues as follows:

Write

$$\begin{pmatrix} H_{11} \\ H_{21} \end{pmatrix} = \left(\begin{array}{c|c} H_{10} & * \\ \hline 0 & R_2 \end{array}\right), \qquad B_{11} = \begin{pmatrix} \hat{B}_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix}.$$

Then, we have

$$H - \tilde{B}F = \left(\begin{array}{c:c} H_{10} & G_1 \\ \hdashline 0 & H_2 \end{array}\right) - \left(\begin{array}{c|c} \hat{B}_{11} & B_{12} \\ \hline 0 & B_{22} \\ \hline \multicolumn{2}{c}{0} \end{array}\right) \left(\begin{array}{c|c} F_{11} & \begin{matrix} H_1 \\ \hline F_2 \end{matrix} \end{array}\right).$$

That is, a feedback matrix F_{11} for this allocation can be immediately found by solving

$$H_{10} - B_{11}F_{11} = \begin{pmatrix} \operatorname{diag}(\lambda_1, \ldots, \lambda_{m_1}) \\ 0 \end{pmatrix}.$$

Because of the last equation, we have

$$H - \tilde{B}F = \left(\begin{array}{c|c} \operatorname{diag}(\lambda_1, \ldots, \lambda_m) & \begin{matrix} G_1 - \hat{B}_{11}H_1 \\ -B_{12}F_2 \end{matrix} \\ \hline 0 & H_2 - B_2F_2 \end{array}\right),$$

where $B_2 = \begin{pmatrix} B_{22} \\ 0 \end{pmatrix}$.

Since B_{22} is a nonsingular upper triangular matrix and H_2 is still an unreduced upper Hessenberg matrix having the same form as H, (H_2, B_2) is a controllable pair.

We then solve the problem of finding F_2 such that $H_2 - B_2F_2$ has the remaining eigenvalues.

However, this time note that the first two blocks on the diagonal are $n_2 \times n_2$, thus, no more immediate assignment of eigenvalues is possible. The other eigenvalues have to be assigned using a different approach. If $n_2 = 1$, we then have a single-input problem to solve. Otherwise, we solve the multi-input problem with $n_1 = n_2$, using the approach below.

Case 2. Let $n_1 = n_2 = \cdots = n_r > n_{r+1} \cdots \geq n_k > 0$, for $1 < r \leq k$.

Suppose we want to assign an eigenvalue λ_1 to $H - \tilde{B}F$.

Then the idea is to find a unitary matrix Q_1 such that

$$Q_1^*(H - \tilde{B}F)Q_1 = \left(\begin{array}{c|c} \lambda_1 & * \\ \hline 0 & H_2 - B_2F_2 \end{array}\right).$$

The unitary matrix Q_1 can be found as the product of the Givens rotations such that

$$(H - \lambda_1 I)Q_1 e_1 = \begin{pmatrix} a_1 \\ 0 \end{pmatrix}.$$

For example, if $n = 4$, $m = 2$, $k = 2$, and $n_1 = n_2 = 2$, then $r = 2$.

$$H - \lambda_1 I = \left(\begin{array}{cc|cc} * & * & * & * \\ * & * & * & * \\ \hline \circledast & * & * & * \\ 0 & \circledast & * & * \end{array}\right).$$

Then Q_1 is the product of two Givens rotations Q_{11} and Q_{21}, where Q_{11} annihilates the entry h_{42} and Q_{21} annihilates the entry h_{31}. Thus,

$$(H - \lambda_1 I)Q_{11}Q_{21} = (H - \lambda I)Q_1 = \left(\begin{array}{c|ccc} * & * & * & * \\ * & * & * & * \\ \hline 0 & * & * & * \\ 0 & 0 & * & * \end{array}\right).$$

Once Q_1 is found, F can be obtained by solving $B_{11}f_1 = a_1$, where $FQ_1 = (f_1, F_2)$, and $\tilde{B} = \begin{pmatrix} B_{11} \\ 0 \end{pmatrix}$. Note that B_{11} is nonsingular.

It can now be shown that (H_2, B_2) is controllable and has the original form that we started with. The process can be continued to allocate the remaining eigenvalues with the pair (H_2, B_2).

To summarize, the allocation of eigenvalues is done using unitary transformations when $n_1 = n_2$ or without unitary transformations when $n_1 > n_2$.

(Note that the Case 1 ($n_1 > n_2$) is a special case of Case 2 with $Q_1 \equiv 1$, and $r = 1$).

Eventually, the process will end up with a single-input system which can be handled with a single-input algorithm described before.

For details of the process, see Miminis and Paige (1988).

Example 11.3.2. Let's consider Example 11.3.1 again. The eigenvalues to be assigned are: $\lambda_1 = 1, \lambda_2 = 2, \lambda_3 = 3, \lambda_4 = 4,$ and $\lambda_5 = 5$.
Then,

$$H_{11} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 1 \\ 2 & 1 & 1 \end{pmatrix}, \qquad H_{21} = (0\ 0\ 0\ 1), \qquad n_1 = 3, \qquad n_2 = 1.$$

Since $m_1 = n_1 - n_2 = 2$, the two eigenvalues 1 and 2, can be assigned immediately as in Case 1.

$$H_{10} = \begin{pmatrix} 1 & 2 \\ 1 & 1 \\ 2 & 1 \end{pmatrix}, \qquad B_{11} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \end{pmatrix}.$$

Solving for F_{11} from

$$H_{10} - B_{11}F_{11} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 0 & 0 \end{pmatrix},$$

we have $F_{11} = \begin{pmatrix} -0.3333 & 3.3333 \\ -0.3333 & -1.6667 \\ 0.6667 & 0.3333 \end{pmatrix}$.

Deflation

$$H = \left(\begin{array}{cc|ccc} 1 & 2 & 3 & 4 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ \hline 2 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right) = \left(\begin{array}{c|c} H_{10} & G_1 \\ \hline 0 & H_2 \end{array}\right),$$

$$\tilde{B} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \left(\begin{array}{cc|c} 1 & 1 & 1 \\ 0 & 1 & 2 \\ \hline 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right) = \left(\begin{array}{c|c} \hat{B}_{11} & B_{12} \\ 0 & B_{22} \end{array}\right).$$

Then, $H_2 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 0 & 1 & 1 \end{pmatrix}$, $\quad B_2 \equiv B_{22} = \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix}$.

(H_2, B_2) is controllable.

Now we find F_2 such that $H_2 - B_2F_2$ has the eigenvalues $(3, 4, 5)$.

This is a **single-input** problem. Using any of the single-input algorithms discussed before, we obtain

$$F_2 = (-3,\ 9.6667,\ -13.6667).$$

So, the required feedback matrix F is given by

$$F = \left(\begin{array}{c|c} F_{11} & 0 \\ \hline & F_2 \end{array} \right) = \left(\begin{array}{cc|ccc} -0.3333 & 3.3333 & 0 & 0 & 0 \\ -0.3333 & -1.6667 & 0 & 0 & 0 \\ 0.6667 & 0.33333 & -3 & 9.6667 & -13.6667 \end{array} \right).$$

Verify: The eigenvalues of $H - \tilde{B}F$ are 1, 2, 5, 4, and 3.

Flop-count: The solution of the Hessenberg multi-input problem, using the above-described method requires about $\frac{23}{7}n^3$ flops.

When combined with about $6n^3$ flops required for the multi-input controller-Hessenberg reduction, the total count is about $9n^3$ flops.

Stability: The round-off error analysis performed by Miminis and Paige (1988) shows that the algorithm is **numerically backward stable.** Specifically, it can be shown that the computed feedback matrix K is such that

$$\Omega((A + \triangle A) - (B + \triangle B)K) = \Omega(L),$$

where $\| \triangle A \|$ and $\| \triangle B \|$ are small, and L is the matrix with eigenvalues $\lambda_i + \delta\lambda_i, i = 1, \cdots, n$; where $|\delta \lambda_i| \leq |\lambda_i|\mu$, μ is the machine precision.

Avoiding complex arithmetic: The method as described above might give a complex feedback matrix because it is an explicit shift algorithm. To avoid complex arithmetic to assign complex conjugate pairs, the idea of implicit shift and the double step needs to be used.

MATCONTROL note: The explicit QR algorithm described in this section has been implemented in MATCONTROL function **poleqrm**.

11.3.3 The Schur Method for the Multi-Input Eigenvalue Assignment Problem

As the title suggests, the following algorithm due to A. Varga (1981) for the multi-input EVA is based on the reduction of the matrix A to the RSF. So, unlike the other two methods just described, the Schur method does not follow the "Template."

Let

$$R = QAQ^{\mathrm{T}} = \begin{pmatrix} A_1 & A_3 \\ 0 & A_2 \end{pmatrix}$$

be the RSF of A and let $\hat{B} = QB$ be the transformed control matrix.

Let's partition $\hat{B}$ as $\hat{B} = \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}$. Then, since (A, B) is controllable, so is (A_2, B_2).

Suppose that the RSF R of A has been **ordered** in such a way that A_1 contains the "**good**" eigenvalues and A_2 contains the "**bad**" ones. The "**good**" eigenvalues are the ones we want to retain and the "**bad**" ones are those we want to reassign.

It is, therefore, natural to ask how the feedback matrix F can be determined such that after the application of feedback, the eigenvalues of A_1 will remain unchanged, while those in A_2 will be changed to "desired" ones by feedback.

The answer to this question is simple. If the feedback matrix F is taken in the form $F = (0, F_2)$, then after the application of the feedback matrix to the pair $(R, \hat{B})$ we have

$$R - \hat{B}F = \begin{pmatrix} A_1 & A_3 - B_1F_2 \\ 0 & A_2 - B_2F_2 \end{pmatrix}.$$

This shows that the eigenvalues of $R - \hat{B}F$ are the union of the eigenvalues of A_1 and of $A_2 - B_2F_2$.

The problem thus reduces to finding F_2 such that $A_2 - B_2F_2$ has a desired spectrum.

The special structure of A_2 can be exploited now.

Since the diagonal blocks of A_2 are either scalars (1×1) or 2×2 matrices, all we need is a procedure to assign eigenvalues to a $p \times p$ matrix where $p = 1$ or 2.

The following is a simple procedure to do this.

Algorithm 11.3.2. *An Algorithm to Assign p ($p = 1$ or 2) Eigenvalues*

Inputs.

M—The state matrix of order p.

G—The control matrix of order $p \times m$.

Γ_p—The set of p complex numbers, closed under complex conjugation.

r—Rank of G.

Output.

F_p—The feedback matrix such that $(M - GF_p)$ has the spectrum Γ_p.

Assumption. *(M, G) is controllable.*

Step 1. *Find the SVD of G, that is, find U and V such that $G = U(\hat{G}, 0)V^{\mathrm{T}}$, where $\hat{G}$ is $r \times r$.*

Step 2. *Update M: $\hat{M} = U^{\mathrm{T}}MU$.*

Step 3. *If $r = p$, compute $\hat{F}_p = (\hat{G})^{-1}(\hat{M} - J)$, where J is $p \times p$ and the eigenvalues of J are the set Γ_p. Go to Step 6.*

Step 4. *Let* $\Gamma_2 = \{\lambda_1, \lambda_2\}$ *and*

$$\hat{M} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}, \qquad \hat{G} = \begin{pmatrix} \beta \\ 0 \end{pmatrix}.$$

Step 5. *Compute* $\hat{F}_p = (\hat{F}_{p_1}, \hat{F}_{p_2})$ *as follows:*

$$\hat{F}_{p_1} = (m_{11} + m_{22} - \lambda_1 - \lambda_2)/\beta,$$

$$\hat{F}_{p_2} = \left(\frac{m_{22}}{m_{21}}\right) \hat{F}_{p1} - (m_{11}m_{22} - m_{12}m_{21} - \lambda_1\lambda_2)/(m_{21}\beta).$$

Step 6. *Compute* $F_p = V \begin{bmatrix} \hat{F}_p \\ 0 \end{bmatrix} U^{\mathrm{T}}$.

Algorithm 11.3.2 can now be used in an iterative fashion to assign all the eigenvalues of A_2, by shifting only 1 or 2 eigenvalues at a time.

The process starts with the last $p \times p$ block of A_2 and then after the assignment with this block is completed using the algorithm above, a new $p \times p$ diagonal block is moved, using orthogonal similarity, in the last diagonal position, and the assignment procedure is repeated on this new block.

The required feedback matrix is the sum of component feedback matrices, each of which assigns 1 or 2 eigenvalues.

The overall procedure then can be summarized as follows:

Algorithm 11.3.3. *The Schur Algorithm for the Multi-Input EVA Problem*

Inputs.

A—The $n \times n$ *state matrix.*

B—The $n \times m$ *input matrix.*

S—The set of numbers to be assigned, closed under complex conjugation.]

Output. *K—The feedback matrix such that the numbers in the set* Γ *belong to the spectrum of* $A - BK$.

Assumption. (A, B) *is controllable.*

Step 1. *Transform A to the* **ordered RSF**:

$$A \equiv QAQ^{\mathrm{T}} = \begin{bmatrix} A_1 & A_3 \\ 0 & A_2 \end{bmatrix},$$

where A_1 *is* $r \times r$, A_2 *is* $(n-r) \times (n-r)$; A_1 *contains the "***good***" eigenvalues and* A_2 *contains the "***bad***" eigenvalues.*

Update $B \equiv QB$ *and set* $\hat{Q} = Q$.

Step 2. *Set* $K \equiv 0$ *(zero matrix), and* $i = r + 1$.

Step 3. *If* $i > n$, *stop.*

Step 4. *Set M equal to the last block in A of order p* ($p = 1$ *or* 2) *and set G equal to the last p rows of B.*

Step 5. *Compute F_p using Algorithm* 11.3.2 *to shift p eigenvalues from the set S.*

Step 6. *Update K and A:* $K \equiv K - (0, F_p)\hat{Q}$, $A \equiv A - B(0, F_p)$.

Step 7. *Move the last block of A in position* (i, i) *accumulating the transformations in Q, and update* $B \equiv QB$, *and* $\hat{Q} = Q\hat{Q}$.

Step 8. *Set* $i \equiv i + p$ *and go to Step* 3.

Remarks

- The ordering of the RSF in Step 1 has to be done according to the procedure described in Chapter 4.
- It has been tacitly assumed that "the complex numbers in S are chosen and ordered so that the ordering agrees with the diagonal structure of the matrix A_2." If this requirement is not satisfied, some interchange of the blocks of A_2 need to be done so that the required condition is satisfied, using an appropriate orthogonal similarity (**Exercise 11.9**).
- The final matrix K at the end of this algorithm is the sum of the component feedback matrices, each of them assigning 1 or 2 eigenvalues.
- The algorithm has the additional flexibility to solve a "**PEVA**," which concerns reassigning only the "bad" eigenvalues, leaving the "good" ones unchanged.

Example 11.3.3. Let's apply Algorithm 11.3.3 with data from Example 11.3.1.
Step 1.

$$A = QAQ^{\mathrm{T}} = \begin{pmatrix} -0.4543 & 1.0893 & -0.2555 & -0.7487 & -0.5053 \\ -0.7717 & -1.6068 & 0.3332 & -1.2007 & 2.6840 \\ -0.0000 & -0.0000 & 0.2805 & -0.2065 & 0.2397 \\ -0.0000 & -0.0000 & -0.0000 & 1.8369 & -3.1302 \\ 0.0000 & 0.0000 & 0.0000 & -0.0000 & 4.9437 \end{pmatrix},$$

$$Q = \begin{pmatrix} -0.2128 & 0.0287 & 0.6509 & -0.6606 & 0.3064 \\ 0.8231 & -0.1628 & -0.2533 & -0.3926 & 0.2786 \\ 0.1612 & -0.8203 & 0.4288 & 0.2094 & -0.2708 \\ -0.4129 & -0.4850 & -0.3749 & 0.0539 & 0.6714 \\ 0.2841 & 0.2539 & 0.4332 & 0.6023 & 0.5517 \end{pmatrix}.$$

Let the desired closed-loop eigenvalues be the same as in Example 11.3.1: $S = \{5\ 4\ 3\ 2\ 1\}$.

Update $B \equiv QB$:

$$B = \begin{pmatrix} -0.2128 & -0.1841 & 1.7974 \\ 0.8231 & 0.6603 & -0.2625 \\ 0.1612 & -0.6591 & -0.1929 \\ -0.4129 & -0.8979 & -2.5077 \\ 0.2841 & 0.5380 & 2.0916 \end{pmatrix}.$$

Step 2.

$$K = 0, \qquad i = 1.$$

Step 3. $i = 1 < n = 5$. Continue

Step 4.

$$p = 1,$$

$$M = (4.9437),$$

$G = (0.2841,\ 0.5380,\ 2.0916)$, $\Gamma_p = 5$, the eigenvalue to be shifted.

Step 5.

$$F_p = \begin{pmatrix} -0.0034 \\ -0.0064 \\ -0.0248 \end{pmatrix}.$$

Step 6. Updated K and A are:

$$K = \begin{pmatrix} -0.0010 & -0.0009 & -0.0015 & -0.0020 & -0.0019 \\ -0.0018 & -0.0016 & -0.0028 & -0.0038 & -0.0035 \\ -0.0071 & -0.0063 & -0.0108 & -0.0150 & -0.0137 \end{pmatrix},$$

$$A = \begin{pmatrix} -0.4543 & 1.0893 & -0.2555 & -0.7487 & -0.4626 \\ -0.7717 & -1.6068 & 0.3332 & -1.2007 & 2.6845 \\ -0.0000 & -0.0000 & 0.2805 & -0.2065 & 0.2313 \\ -0.0000 & -0.0000 & -0.0000 & 1.8369 & -3.1996 \\ 0.0000 & 0.0000 & 0.0000 & -0.0000 & 5.0000 \end{pmatrix}.$$

Step 7. Reorder A and update $\hat{Q}$ and B:

$$A = \begin{pmatrix} 5.0000 & 3.2232 & -0.3674 & -1.1307 & -2.4841 \\ 0 & 1.8369 & 0.1125 & -1.2452 & -0.4711 \\ 0 & 0 & 0.2805 & -0.0392 & -0.4337 \\ 0 & 0 & 0 & -0.5524 & -1.2822 \\ 0 & 0 & 0 & 0.5749 & -1.5087 \end{pmatrix},$$

$$\hat{Q} = \begin{pmatrix} 0.7474 & 0.3745 & 0.5264 & 0.1504 & 0.0376 \\ 0.0568 & -0.0551 & -0.2840 & 0.6816 & 0.6696 \\ 0.2416 & -0.8972 & 0.2528 & 0.1953 & -0.1859 \\ 0.3829 & 0.1005 & -0.6528 & 0.2773 & -0.5833 \\ 0.4828 & -0.2040 & -0.3902 & -0.6307 & 0.4187 \end{pmatrix},$$

$$B = \begin{pmatrix} 0.7474 & 1.1219 & 3.0756 \\ 0.0568 & 0.0017 & -0.9056 \\ 0.2416 & -0.6556 & -0.7945 \\ 0.3829 & 0.4834 & -1.3744 \\ 0.4828 & 0.2789 & -1.0956 \end{pmatrix}.$$

Step 8. $i = 2$ and return to Step 3.

$$p = 2,$$

Step 3. $i = 2 < n = 5$. Continue.
Step 4.

$$M = \begin{pmatrix} 0.5524 & -1.2822 \\ 0.5749 & -1.5087 \end{pmatrix}, \qquad G = \begin{pmatrix} 0.3829 & 0.4834 & -1.3744 \\ 0.4828 & 0.2789 & -1.0956 \end{pmatrix},$$

$$\Gamma_p = \{2, 1\}$$

Step 5.

$$F_p = \begin{pmatrix} 7.0705 & -6.3576 \\ -5.1373 & 3.4356 \\ 1.7303 & 0.7260 \end{pmatrix}.$$

Step 6.

$$K = \begin{pmatrix} -0.3630 & 2.0061 & -2.1362 & 5.9678 & -6.7883 \\ -0.3103 & -1.2184 & 2.0102 & -3.5949 & 4.4318 \\ 1.0061 & 0.0194 & -1.4235 & 0.0069 & -0.7191 \end{pmatrix},$$

$$A = \begin{pmatrix} 5.0000 & 3.2232 & -0.3674 & -5.9733 & -3.8198 \\ 0 & 1.8369 & 0.1125 & -0.0712 & 0.5417 \\ 0 & 0 & 0.2805 & -3.7408 & 3.9316 \\ 0 & 0 & 0 & 1.6016 & 0.4896 \\ 0 & 0 & 0 & 0.4896 & 1.3984 \end{pmatrix}.$$

Step 7. Reorder A and update $\hat{Q}$ and B (Recorded A and updated B are shown below):

$$A = \begin{pmatrix} 5.0000 & 3.2232 & -0.3674 & -5.9733 & -3.8198 \\ 0 & 1.0000 & 0.3087 & 0.1527 & -5.2208 \\ 0 & 0 & 2.0000 & 0.2457 & 1.2394 \\ 0 & 0.0000 & 0.0000 & 1.8369 & -0.8889 \\ 0 & 0 & 0 & 0 & 0.2805 \end{pmatrix},$$

$$B = \begin{pmatrix} 0.7474 & 1.1219 & 3.0756 \\ -0.2402 & 0.6483 & 0.5859 \\ 0.4240 & 0.1944 & -1.8404 \\ 0.4443 & 0.5319 & -0.8823 \\ 0.0804 & -0.0193 & 0.1787 \end{pmatrix}.$$

Step 8. $i = 4$ and return to Step 3.
Step 3. $i = 4 < n = 5$. Continue.

Step 4.

$$p = 1, \qquad M = 0.2805, \qquad G = (0.0804 \ -0.0193 \ 0.1787), \qquad \Gamma_p = 4.$$

Step 5.

$$F_p = \begin{pmatrix} -7.7091 \\ 1.8532 \\ -17.1422 \end{pmatrix}.$$

Step 6.

$$K = \begin{pmatrix} -0.9827 & 2.7748 & -2.9014 & 11.1937 & -12.3165 \\ -0.1613 & -1.4032 & 2.1942 & -4.8511 & 5.7607 \\ -0.3719 & 1.7286 & -3.1250 & 11.6272 & -13.0117 \end{pmatrix},$$

$$A = \begin{pmatrix} 5.0000 & 3.2232 & -0.3674 & -5.9733 & 52.5851 \\ 0 & 1.0000 & 0.3087 & 0.1527 & 1.7698 \\ 0 & 0 & 2.0000 & 0.2457 & -27.4012 \\ 0 & 0.0000 & 0.0000 & 1.8369 & -13.5735 \\ 0 & 0 & 0 & 0 & 4.0000 \end{pmatrix}.$$

Step 7. Reorder A and update $\hat{Q}$ and B (Recorded A and updated B are shown below):

$$A = \begin{pmatrix} 5.0000 & 3.2232 & -0.3674 & -5.9733 & 52.5851 \\ 0 & 1.0000 & 0.3087 & 0.1527 & 1.7698 \\ 0 & 0 & 2.0000 & 0.2457 & -27.4012 \\ 0 & 0.0000 & 0.0000 & 4.0000 & -13.5735 \\ 0 & 0 & 0 & 0 & 1.8369 \end{pmatrix},$$

$$B = \begin{pmatrix} 0.7474 & 1.1219 & 3.0756 \\ -0.2402 & 0.6483 & 0.5859 \\ 0.4240 & 0.1944 & -1.8404 \\ 0.4261 & 0.5283 & -0.8994 \\ 0.1493 & 0.0646 & 0.0377 \end{pmatrix}.$$

Step 8. $i = 5$. Return to Step 3.
Step 3. $i = n = 5$. Continue
Step 4.

$$p = 1, \qquad M = 1.8369, \qquad G = (0.1493 \ \ 0.0646 \ \ 0.0377), \qquad \Gamma_p = 3.$$

Step 5.

$$F_p = \begin{pmatrix} -6.2271 \\ -2.6950 \\ -1.5710 \end{pmatrix}.$$

Step 6.

$$K = \begin{pmatrix} -1.9124 & 3.3022 & -3.0213 & 15.9029 & -16.2462 \\ -0.5637 & -1.1750 & 2.1423 & -2.8130 & 4.0599 \\ -0.6064 & 1.8617 & -3.1553 & 12.8153 & -14.0031 \end{pmatrix},$$

$$A = \begin{pmatrix} 5.0000 & 3.2232 & -0.3674 & -5.9733 & 65.0948 \\ 0 & 1.0000 & 0.3087 & 0.1527 & 2.9415 \\ 0 & 0 & 2.0000 & 0.2457 & -27.1285 \\ 0 & 0.0000 & 0.0000 & 4.0000 & -10.9093 \\ 0 & 0 & 0 & 0 & 3.0000 \end{pmatrix}.$$

All the eigenvalues are assigned, and the iteration terminates.

Verification: The eigenvalues of $A - BK$ are: {5.00000000000024, 3.99999999999960, 1.00000000000000, 3.00000000000018, 1.99999999999999}.

Flop-count and stability. The algorithm requires about $30n^3$ flops, most of which is consumed in the reduction of A to the RSF and ordering of this RSF.

The algorithm is believed to be numerically stable (note that it is based on all numerically stable operations). **However, no formal round-off error analysis of the algorithm has been performed yet.**

MATCONTROL function: Algorithm 11.3.3 has been implemented in MATCONTROL function **polesch**.

11.3.4 Partial Eigenvalue Assignment Problem

The PEVA problem is the one of reassigning by feedback only a few eigenvalues, say $\lambda_1, \ldots, \lambda_p (p < n)$, of an $n \times n$ matrix A leaving the other eigenvalues $\lambda_{p+1}, \ldots, \lambda_n$ unchanged.

Formally, PEVA is defined as follows:

Given $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, a part of the spectrum $\{\lambda_1, \ldots, \lambda_p\}$ of A, and a set of self-conjugate numbers $\{\mu_1, \ldots, \mu_p\}$, find a feedback matrix F such that the spectrum of $A - BF$ is the set $\{\mu_1, \ldots, \mu_p;\ \lambda_{p+1}, \ldots, \lambda_n\}$.

A projection algorithm for this problem was proposed by Saad (1988). Here we describe a simple parametric algorithm via Sylvester equation. Note that Varga's algorithm described in the last section can be used for solving PEVA; however, that will require full knowledge of the eigenvalues of A and is thus not suitable for large problems. *The algorithm described below requires the knowledge of only those small number of eigenvalues (and the corresponding eigenvectors) that are required to be reassigned.* Furthermore, the algorithm is parametric in nature which can be exploited to devise a robust EVA algorithm (see **Section 11.6**). The discussion here has been taken from Datta and Sarkissian (2002). This paper also contains a result on the existence and uniqueness of the solution for PEVA in the multi-input case.

Theorem 11.3.2. *(Parametric Solution to PEVA Problem). Assume that (i) B has full rank, (ii) the sets $\{\lambda_1, \ldots, \lambda_p\}$ and $\{\mu_1, \ldots, \mu_p\}$ are closed under complex conjunction and disjoint, and (iii) let the pair (A, B) be partially controllable with respect to $\{\lambda_1, \ldots, \lambda_p\}$. Assume further that the closed-loop matrix has a complete set of eigenvectors. Let $\Gamma = (\gamma_1, \ldots, \gamma_p)$*

be a matrix such that

$$\gamma_j = \overline{\gamma_k}, \text{ whenever } \mu_j = \overline{\mu_k},$$

Let Y_1 *be the matrix of left eigenvectors associated with* $\{\lambda_1, \ldots, \lambda_p\}$. *Set* $\Lambda_1 = \text{diag}(\lambda_1, \ldots, \lambda_p)$ *and* $\Lambda_{c1} = \text{diag}(\mu_1, \ldots, \mu_p)$. *Let* Z_1 *be a unique nonsingular solution of the Sylvester equation*

$$\Lambda_1 Z_1 - Z_1 \Lambda_{c1} = Y_1^H B \Gamma.$$

Let Φ be defined by $\Phi Z_1 = \Gamma$, then

$$F = \Phi Y_1^H,$$

solves the partial eigenvalue assignment problem for the pair (A, B).

Conversely, if there exists a real feedback matrix F *of the form that solves the PEVA problem for the pair* (A, B), *then the matrix* Φ *can be constructed satisfying* Steps 2–4 *of* **Algorithm 11.3.4**.

Proof. see Datta and Sarkissian (2002). ■

Algorithm 11.3.4. *Parametric Algorithm for PEVA Problem*
Inputs.

(i) *A—The* $n \times n$ *state matrix.*
(ii) *B—The* $n \times m$ *control matrix.*
(iii) *The set* $\{\mu_1, \ldots, \mu_p\}$, *closed under complex conjugation.*
(iv) *The self-conjugate subset* $\{\lambda_1, \ldots, \lambda_p\}$ *of the spectrum* $\{\lambda_1, \ldots, \lambda_n\}$ *of the matrix A and the associated right eigenvector set* $\{y_1, \ldots, y_p\}$.

Outputs. *The real feedback matrix F such that the spectrum of the closed-loop matrix* $A - BF$ *is* $\{\mu_1, \ldots, \mu_p; \lambda_{p+1}, \ldots, \lambda_n\}$.
Assumptions.

(i) *The matrix pair* (A, B) *is partially controllable with respect to the eigenvalues* $\lambda_1, \ldots, \lambda_p$.
(ii) *The sets* $\{\lambda_1, \ldots, \lambda_p\}$, $\{\lambda_{p+1}, \ldots, \lambda_n\}$, *and* $\{\mu_1, \ldots, \mu_p\}$ *are disjoint.*

Step 1. *Form*

$$\Lambda_1 = \text{diag}(\lambda_1, \ldots, \lambda_p), \quad Y_1 = (y_1, \ldots, y_p), \quad \text{and} \quad \Lambda_{c1} = \text{diag}(\mu_1, \ldots, \mu_p).$$

Step 2. *Choose arbitrary* $m \times 1$ *vectors* $\gamma_1, \ldots, \gamma_p$ *in such a way that* $\overline{\mu_j} = \mu_k$ *implies* $\overline{\gamma_j} = \gamma_k$ *and form* $\Gamma = (\gamma_1, \ldots, \gamma_p)$.

Step 3. *Find the unique solution Z_1 of the Sylvester equation:*

$$\Lambda_1 Z_1 - Z_1 \Lambda_{c1} = Y_1^H B \Gamma.$$

If Z_1 is ill-conditioned, then return to Step 2 and select different $\gamma_1, \ldots, \gamma_p$.

Step 4. *Solve $\Phi Z_1 = \Gamma$ for Φ.*

Step 5. *Form $F = \Phi Y_1^H$.*

A Numerical Example

In this section, we report results of our numerical experiments with Algorithm 11.3.4 on a 400×400 matrix obtained by discretization of the partial differential equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + 20\frac{\partial u}{\partial x} + 180u(x, y, t) + \sum_{i=1}^{2} F_i(x, y) g_i(t)$$

on the unit square $\Omega = (0, 1) \times (0, 1)$ with the Dirichlet boundary conditions:

$$u(x, y, t) = 0, \text{ for } (x, y) \in \partial\Omega \text{ and } t \geq 0$$

and some initial condition which is of no importance for the PEVA problem. This problem was earlier considered by Saad (1988). Using finite difference scheme of order $O(\|\Delta x\|^2, \|\Delta y\|^2)$, we discretize the equation in the region Ω with 20 interior points in both the x and y directions, thus obtaining a 400×400 matrix A. The 400×2 matrix B, whose ith column discretizes the function $F_i(x, y)$ is filled with random numbers between -1 and 1.

Using sparse MATLAB command **eigs**, the following 10 eigenvalues with the largest real parts are computed

$$\begin{array}{llll} \lambda_1 = 55.0660, & \lambda_2 = 29.2717, & \lambda_3 = 25.7324, & \lambda_4 = -0.0618, \\ \lambda_5 = -13.0780, & \lambda_6 = -22.4283, & \lambda_7 = -42.4115, & \\ \lambda_8 = -48.2225, & \lambda_9 = -71.0371, & \lambda_{10} = -88.3402. & \end{array}$$

The residual of each eigenpair $\|y^*(A - \lambda I)\| < 4 \cdot 10^{-12}$ and each left eigenvector is normalized. Algorithm 11.3.4 was used to reassign $\lambda_1, \lambda_2, \lambda_3$, and λ_4 to $-7, -8, -9$, and -10, respectively, obtaining the 2×400 feedback matrix F with $\|F\|_2 < 127$. Note that the $\|A\|_2 = 3.3 \cdot 10^3$. The 10 eigenvalues of the closed-loop matrix $A - BF$ with the largest real parts obtained by the algorithm are the following:

$$\begin{array}{llll} \mu_1 = -7.00000, & \mu_2 = -8.0000, & \mu_3 = -9.0000, & \mu_4 = -10.0000, \\ \lambda_5 = -13.0780, & \lambda_6 = -22.4283, & \lambda_7 = -42.4115, & \\ \lambda_8 = -48.2225, & \lambda_9 = -71.0371, & \lambda_{10} = -88.3402. & \end{array}$$

11.4 CONDITIONING OF THE FEEDBACK PROBLEM

In this section, we will discuss the sensitivity of the feedback problem, that is, **we are interested in determining a measure that describes how small perturbations in the data affect the computed feedback.** We discuss the **single-input case first.**

11.4.1 The Single-Input Case

Arnold (1993) first discussed the perturbation analysis of the single-input Hessenberg feedback problem in his Ph.D. dissertation. Based on his analysis, he derived two condition numbers for the problem and identified several condition number estimators. For details, we refer the readers to the above dissertation. We simply state here one of the condition number estimators which has worked well in several meaningful numerical experiments. Recall that the single-input Hessenberg feedback problem is defined by the data (H, β, S), where H is an unreduced upper Hessenberg matrix, $\beta = (\alpha, 0, \ldots, 0)^T$, $\alpha \neq 0$ and $S = \{\lambda_1, \ldots, \lambda_n\}$.

Estimating the Condition Numbers of the Feedback Problem

Theorem 11.4.1. *If $\nu(H, \beta)$ is the condition number of the single-input Hessenberg feedback problem, then an estimator of this number is given by*

$$\nu_{d\phi} = \frac{|\beta|\,\|\omega\| + \|H\|\,\|e_n^{\mathrm{T}}\phi'(H)\|}{\|e_n^{\mathrm{T}}\phi(H)\|} \tag{11.4.1}$$

where

$$\omega = \left(\frac{1}{\beta}, \frac{1}{h_{21}}, \ldots, \frac{1}{h_{n,n-1}}\right)^{\mathrm{T}},$$

and $\phi(H) = (H - \lambda_1 I)\ldots(H - \lambda_n I)$.

Defining the quantity digits off (as in Rice (1966)) by

$$\log_{10}\left[\left(\frac{\mu\ \nu_{\text{estimate}}}{\text{err}}\right)\right], \tag{11.4.2}$$

where μ is the machine precision ($\mu \approx 2 \times 10^{-16}$) and err stands for the error tolerance, it has be shown that the maximum digits off in estimating conditioning for 100 ill-conditioned problems are only 1.86, and minimum digits off are 0.51. **Thus, it never underestimated the error and overestimated the error by less than two digits.**

The computation of this condition estimator requires only $2n^3/3$ flops once the system is in controller-Hessenberg form. For details of these experiments, see Arnold (1993). **Note these bounds work only for the single-input Hessenberg feedback problem.**

11.4.2 The Multi-Input Case

Arnold (1993) considered a perturbation analysis of the single-input Hessenberg feedback problem by considering only the perturbation of the matrix H and the vector b.

Sun (1996) has studied perturbation analysis of both single-input and the multi-input problems by allowing perturbations of all the data, namely A, B, and $S = \{\lambda_1, \ldots, \lambda_n\}$. Below, we state his result for the multi-input problem, without proof. For more general results on perturbation analyses of feedback matrices as well as those of conditioning of the closed-loop eigenvalues, see the recent papers of Mehrmann and Xu (1996, 1997).

Let $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$

Let $\lambda_i \neq \lambda_j$, for all $i \neq j$. Let $K = (k_1, \ldots, k_m)^{\mathrm{T}}$ and $X = (x_1, \ldots, x_n)$ be such that

$$A + BK = X\Lambda X^{-1}, \tag{11.4.3}$$

where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$.

Also, let $y_1, \ldots, y_n$ be the normalized left eigenvectors of $A + BK$, that is, $Y = X^{-T} = (y_1, \ldots, y_n)$, which implies $y_i^{\mathrm{T}} x_j = \delta_{ij}$ for all i and j.

Suppose that the data matrices A, B, and Λ and the feedback matrix K are so perturbed that the resulting closed-loop matrix has also the distinct eigenvalues.

Let $B = (b_1, \ldots b_m)$. Define now

$$W_k = (S_1 X^{\mathrm{T}}, S_2 X^{\mathrm{T}}, \ldots, S_m X^{\mathrm{T}})_{n\times mn}, \tag{11.4.4}$$

where $S_j = \mathrm{diag}(y_1^{\mathrm{T}} b_j, \ldots, y_n^{\mathrm{T}} b_j)$, $j = 1, 2, \ldots, m$.

Also define

$$W_a = (D_1(X)X^{-1}, D_2(X)X^{-1}, \ldots, D_n(X)X^{-1})_{n\times n^2}, \tag{11.4.5}$$

$$W_b = \mathrm{diag}(T_1 X^{-1}, T_2 X^{-1}, \ldots, T_m X^{-1})_{n\times nm}, \tag{11.4.6}$$

and $W_\lambda = -I_n$, where

$$D_i(X) = \mathrm{diag}(x_{i1}, \ldots, x_{in}), \quad i = 1, \ldots, n \tag{11.4.7}$$

$$T_j = \mathrm{diag}(k_j^{\mathrm{T}} x_1, \ldots k_j^{\mathrm{T}} x_n), \quad j = 1, \ldots m, \tag{11.4.8}$$

and $x_i = (x_{i1}, \ldots, x_{in})$.

$$\text{Also, let } Z = W_k^{\dagger}, \quad \Phi = -ZW_a, \quad \text{and} \quad \Psi = -ZW_b. \tag{11.4.9}$$

Here $W_k^{\dagger}$ denotes the generalized inverse of W_k.

Theorem 11.4.2. *Perturbation Bound for a Multi-Input Feedback Matrix*

Suppose that the controllable pair (A, B) is slightly perturbed to another controllable pair $(\tilde{A}, \tilde{B})$, and that the self-conjugate set $S = \{\lambda_1, \ldots, \lambda_n\}$, $\lambda_i \neq \lambda_j, i \neq j$ is slightly perturbed to the set $\tilde{S} = \{\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n\}$, $\tilde{\lambda}_i \neq \tilde{\lambda}_j, i \neq j$.

Let K be the feedback matrix of the EVA problem with the data A, B, S. Then there is a solution $\tilde{K}$ to the problem with data $\tilde{A}$, $\tilde{B}$, and $\tilde{S}$ such that for any consistent norm $\|\|$, we have

$$\begin{aligned} \| \tilde{K} - K \| &\leq \delta_K + O\left(\left\| \begin{pmatrix} \tilde{a} \\ \tilde{b} \\ \tilde{\lambda} \end{pmatrix} - \begin{pmatrix} a \\ b \\ \lambda \end{pmatrix} \right\|^2 \right) \\ &\leq \Delta_K + O\left(\left\| \begin{pmatrix} \tilde{a} \\ \tilde{b} \\ \tilde{\lambda} \end{pmatrix} - \begin{pmatrix} a \\ b \\ \lambda \end{pmatrix} \right\|^2 \right), \end{aligned} \tag{11.4.10}$$

where

$$\begin{aligned} a &= \mathrm{vec}(A),\ \tilde{a} = \mathrm{vec}(\tilde{A}), \\ b &= \mathrm{vec}(B),\ \tilde{b} = \mathrm{vec}(\tilde{B}), \\ \lambda &= (\lambda_1, \ldots, \lambda_n)^{\mathrm{T}},\ \tilde{\lambda} = (\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_n)^{\mathrm{T}}, \\ \delta_K &= \|\Phi(\tilde{a} - a) + \Psi(\tilde{b} - b) + Z(\tilde{\lambda} - \lambda)\|, \\ \Delta_K &= \|\Phi\|\|\tilde{a} - a\| + \|\Psi\|\|\tilde{b} - b\| + \|Z\|\|\tilde{\lambda} - \lambda\|, \end{aligned} \tag{11.4.11}$$

Z, Ψ, and Φ are defined by (11.4.9).

11.4.3 Absolute and Relative Condition Numbers

The three groups of **absolute condition numbers** that reflect the three different types of sensitivities of K with respect to the data A, B, and S, respectively, have been obtained by Sun (1996). These are:

$$\kappa_A(K) = \|\Phi\|, \qquad \kappa_B(K) = \|\Psi\|, \qquad \text{and } \kappa_\lambda(K) = \|Z\|. \tag{11.4.12}$$

Furthermore, the scalar $\kappa(K)$ defined by

$$\kappa(K) = \sqrt{\kappa_A^2(K) + \kappa_B^2(K) + \kappa_\lambda^2(K)} \tag{11.4.13}$$

can be regarded as an **absolute condition** number of K.

If $\|\cdot\|_2$ is used, and if the matrix $X = (x_1, x_2, \ldots, x_n)$ is such that $\|x_j\|_2 = 1$ for all j, then

$$\kappa_A(K) = \|\Phi\|_2 \leq \|Z\|_2 \|X^{-1}\|_2, \tag{11.4.14}$$

$$\kappa_B(K) = \|\Psi\|_2 \leq \max_{1\leq j\leq n} \|k_j\|_2 \|Z\|_2 \|X^{-1}\|_2, \tag{11.4.15}$$

$$\kappa_\lambda(K) = \|Z\|_2. \tag{11.4.16}$$

Thus, using the Frobenius norm, the respective **relative condition numbers** are given by

$$\kappa_A^{(r)}(K) = \kappa_A(K) \frac{\|A\|_F}{\|K\|_F}, \tag{11.4.17}$$

$$\kappa_B^{(r)}(K) = \kappa_B(K) \frac{\|B\|_F}{\|K\|_F}, \tag{11.4.18}$$

and

$$\kappa_\lambda^{(r)}(K) = \kappa_\lambda(K) \frac{\|\lambda\|_2}{\|K\|_F}, \quad \lambda = (\lambda_1, \ldots, \lambda_n)^T. \tag{11.4.19}$$

Furthermore, the **relative condition number of** K is given by

$$\kappa_K^{(r)}(K) = \sqrt{(\kappa_A^{(r)}(K))^2 + (\kappa_B^{(r)}(K))^2 + (\kappa_\lambda^{(r)}(K))^2}, \tag{11.4.20}$$

where $\kappa_A^{(r)}(K)$, $\kappa_B^{(r)}(K)$, and $\kappa_\lambda^{(r)}(K)$ are evaluated using 2-norm.

Remark

- A variation of the above results appears in Mehrmann and Xu (1996, 1997), where it has been shown that the ill-conditioning of the feedback problem is also related to the ill-conditioning of the open-loop eigenvector matrix and the distance to uncontrollability (see next section for more on this).

Example 11.4.1. (Laub and Linnemann 1986; Sun 1996).

Consider the following single-input problem:

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

and

$$S = \{-2.9992, -0.8808, -2, -1, 7.0032 \times 10^{-14}\}.$$

Choose $\alpha = 0.0010$.

Then, $K = (3.12, -1.67, 7.45, -2.98, 0.37)$.

The feedback problem with the above data is expected to be ill-conditioned, as $\kappa_A^{(r)}(K) = 3.2969 \times 10^{12}$, $\kappa_B^{(r)}(K) = 1.01117 \times 10^{12}$, $\kappa_\lambda^{(r)}(K) = 2.3134 \times 10^{12}$, and $\kappa_K^{(r)}(K) = 4.1527 \times 10^{12}$.

Indeed, if only the 1st entry of S is changed to -3 and all other data remain unchanged, then the feedback vector for this perturbed problem becomes $\hat{K} =$ (3.1192, 0.0078, 7.8345, 0.0004, 0.3701).

11.5 CONDITIONING OF THE CLOSED-LOOP EIGENVALUES

Suppose that the feedback matrix K has been computed using a stable algorithm, that is, the computed feedback matrix $\hat{K}$ is the exact feedback matrix for a nearby EVA problem. The question now is: **How far are the eigenvalues of the computed closed-loop matrix $\hat{M}_c = A - B\hat{K}$ from the desired eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$?** Unfortunately, the answer to this question is: *Even though a feedback matrix has been computed using a numerically stable algorithm, there is no guarantee that the eigenvalues of the closed-loop matrix will be near those which are to be assigned.*

The following interrelated factors, either individually, or in combination, can contribute to the conditioning of the closed-loop eigenvalues:

- The conditioning of the problem of determining the feedback matrix K from the given data.
- The condition number (with respect to a p-norm) of the eigenvector matrix of the closed-loop system.
- The distance to uncontrollability, and the distance between the open-loop and closed-loop eigenvalues.
- The norm of the feedback matrix.

Regarding the first factor, we note that if the problem of computing the feedback matrix is ill-conditioned, then even with the use of a stable numerical algorithm, the computed feedback matrix cannot be guaranteed to be accurate, and, as a result, the computed closed-loop eigenvalues might differ significantly from those to be assigned. (Note that the eigenvalue problem of a nonsymmetric matrix can be very ill-conditioned.)

The relation of the other two factors to the conditioning of the closed-loop eigenvalues can be explained by the following analysis using the **Bauer–Fike Theorem (Chapter 3)**.

Let $M_c = A - BK$ and let $E = \hat{M}_c - M_c$, where $\hat{M}_c = A - B\hat{K}$, $\hat{K}$ being the computed value of K.

Let X be the transforming matrix that diagonalizes the matrix M_c, that is, $X^{-1}M_cX = \text{diag}(\lambda_1, \ldots, \lambda_n)$. Let μ be an eigenvalue of $\hat{M}_c$. Then, by the

Bauer–Fike Theorem (**Theorem 3.3.3**), we have

$$\min_{\lambda_i} |\lambda_i - \mu| \leq \text{Cond}_2(X)\|E\|_2.$$

Again, $E = A - B\hat{K} - (A - BK) = B(K - \hat{K}) = B\triangle K$.

Thus, we see that the product of the spectral condition number of X and the $\|B\triangle K\|_2$ influences the distance between the desired poles and those obtained with a computed $\hat{K}$.

This again is related to the factors: *norm of the computed feedback matrix $\hat{K}$, distance to the uncontrollability of the pair (A, B), and the distance between closed-loop poles and the eigenvalues of A, etc.* (See Mehrmann and Xu 1997.) Note that some of these observations also follow from the explicit formula of the feedback vector (11.2.9) in the single-input case, and the one in the multi-input case derived in Arnold (1993).

Example 11.5.1. Consider EVA with the following data:

$$A = \begin{pmatrix} -4 & 0 & 0 & 0 & 0 \\ 0.001 & -3 & 0 & 0 & 0 \\ 0 & 0.001 & -2 & 0 & 0 \\ 0 & 0 & 0.001 & -1 & 0 \\ 0 & 0 & 0 & 0.001 & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

$$S = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{10, 12, 24, 29, 30\}.$$

Then $K = (-115, 4.887 \times 10^6, -9.4578 \times 10^{10}, 8.1915 \times 10^{14}, -2.5056 \times 10^{18})$

The eigenvalue assignment problem with the above data is very ill-conditioned as the following computation shows.

Change the entry a_{51} of A to 10^{-6} and keep all other data unchanged. The eigenvalues of the closed-loop matrix then become: $\{1.5830 \times 10^6, -1.5829 \times 10^6, -3, -2, -1\}$.

The explanation of this drastic change in the closed-loop eigenvalues can be given in the light of the discussions we just had in the last section.

- **Ill-conditioning of the feedback vector:** Let $\hat{K}$ be obtained by changing the first entry of K to -114.999 and leaving the remaining entries unchanged. The eigenvalues of $(A - B\hat{K})$ are $\{29.5386 \pm 0.4856j, 23.9189, 12.0045, 9.9984\}$.

  So, **the problem of computing the feedback vector K is ill-conditioned.**
- All the subdiagonal entries of A **are small, indicating that the system is near an uncontrollable system.**
- **Distance between the open-loop and closed-loop eigenvalues:** The open-loop eigenvalues are $\{0, -1, -2, -3, -4\}$.

  Thus, **the open-loop eigenvalues are well-separated from those of the closed-loop eigenvalues.**

- **Ill-conditioning of the closed-loop eigenvector matrix:** $\text{Cond}_2(X) = 1.3511 \times 10^{24}$.
  Thus, **the spectral condition number of the closed-loop matrix is large.** The condition numbers of the individual eigenvalues are also large.
  Note: In Example 11.5.1, the feedback vector K was computed using Algorithm 11.2.3. The MATLAB function **place** cannot place the eigenvalues.

Concluding Remarks

We have identified several factors that contribute to the ill-conditioning of the closed-loop eigenvalues. In general, the problem of assigning eigenvalues is an intrinsically ill-conditioned problem. Indeed, in Mehrmann and Xu (1996), it has been shown that in the single-input case, the feedback vector K (which is unique) depends upon the solution of a linear system whose matrix is a **Cauchy matrix (Exercise 11.13)**, and a Cauchy matrix is well-known to be ill-conditioned for large order matrices. Thus, the **distribution of eigenvalues is also an important factor for conditioning of the EVA problem**, and the condition number of the problem can be reduced by choosing the eigenvalues judiciously in a prescribed compact set in the complex plane. For details, see (Mehrmann and Xu (1998)). See also Calvetti *et al.* (1999).

11.6 ROBUST EIGENVALUE ASSIGNMENT

In the last section we have discussed the aspect of the closed-loop eigenvalue sensitivity due to perturbations in the data A, B, and K.

The problem of finding a feedback matrix K such that the closed-loop eigenvalues are as insensitive as possible is called the **robust eigenvalue assignment (REVA)** problem.

Several factors affecting the closed-loop eigenvalue sensitivity were identified in the last section, the principal of those factors being the conditioning of the closed-loop eigenvector matrix.

In this section, we consider REVA with respect to minimizing the condition number of the eigenvector matrix of the closed-loop matrix. In the multi-input case, one can think of solving the problem by making use of the available freedom. One such method was proposed by Kautsky *et al.* (1985). For an excellent account of the REVA problem and discussion on this and other methods, see the paper by Byers and Nash (1989). See also Tits and Yang (1996).

11.6.1 Measures of Sensitivity

Let the matrix $M = A - BK$ be diagonalizable, that is, assume that there exists a nonsingular matrix X such that

$$X^{-1}(A - BK)X = \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n).$$

Recall from Chapter 3 (see also Wilkinson (1965), Datta (1995), etc.) that a measure of sensitivity c_j of an individual eigenvalue λ_j due to perturbations in the data A, B, and K is given by

$$c_j = \frac{1}{s_j} = \frac{\|y_j\|_2 \, \|x_j\|_2}{|y_j^{\mathrm{T}} x_j|},$$

where x_j and y_j are, respectively, the right and left eigenvectors of M corresponding to λ_j. Furthermore, the overall sensitivity of all the eigenvalues of the matrix M is given by $\mathrm{Cond}_2(X) = \|X\|_2 \, \|X^{-1}\|_2$. Note also that $\max_j c_j \leq \mathrm{Cond}_2(X)$.

Thus, two natural measures of sensitivity are:

$$\nu_1 = \|C\|_\infty, \quad \text{and} \quad \nu_2 = \mathrm{Cond}_2(X),$$

where $C = (c_1, c_2, \ldots, c_n)^{\mathrm{T}}$.

One could also take (see Kautsky *et al.* 1985)

$$\nu_3 = \|X^{-1}\|_F n^{1/2} = \|C\|_2 n^{1/2} \quad \text{and} \quad \nu_4 = \left(\sum_j \sin^2 \theta_j \right)^{1/2} \Big/ n^{1/2}$$

as other measures. Here θ_j are the angles between the eigenvectors x_j and certain corresponding orthonormal vectors $\hat{x}_j$, $j = 1, 2, \ldots, n$.

11.6.2 Statement and Existence of Solution of the Robust EigenValue Assignment Problem

In view of the above, the *REVA problem* with respect to minimizing the conditioning of the eigenvalue matrix X can be formulated as follows:

> Given $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ $(m \leq n)$, having full rank, and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, find a real matrix K and a nonsingular matrix X satisfying
>
> $$(A - BK)X = X\Lambda \tag{11.6.1}$$
>
> such that some measures ν of the sensitivity of the closed-loop eigenproblem is optimized.

The following result, due to Kautsky *et al.* (1985), gives conditions under which a given nonsingular X can be assigned to (11.6.1).

Theorem 11.6.1. *Given a nonsingular X, and Λ as above, there exists a matrix K satisfying* (11.6.1) *if and only if*

$$U_1^{\mathrm{T}}(X\Lambda - AX) = 0, \tag{11.6.2}$$

where U_1 is defined by

$$B = [U_0, U_1]\begin{bmatrix} Z \\ 0 \end{bmatrix}, \tag{11.6.3}$$

with $U = [U_0, U_1]$ orthogonal and Z nonsingular.

The matrix K is explicitly given by

$$K = Z^{-1}U_0^{\mathrm{T}}(A - X\Lambda X^{-1}). \tag{11.6.4}$$

Proof. Since B has full rank, the factorization of B given by (11.6.3) exists with Z nonsingular. Again, from (11.6.1), we have

$$BK = A - X\Lambda X^{-1}. \tag{11.6.5}$$

Multiplying (11.6.5) by U^{T}, we obtain

$$\begin{aligned} ZK &= U_0^{\mathrm{T}}(A - X\Lambda X^{-1}), \\ 0 &= U_1^{\mathrm{T}}(A - X\Lambda X^{-1}). \end{aligned} \tag{11.6.6}$$

Since X and Z are invertible, we immediately have (11.6.2) and (11.6.4). ■

11.6.3 A Solution Technique for the Robust Eigenvalue Assignment Problem

Theorem 11.6.1 suggests the following algorithm for a solution of the REVA problem.

Algorithm 11.6.1. *An REVA Algorithm (The KNV Algorithm)*

Input.

A—The $n \times n$ state matrix.

B–The $n \times m$ input matrix with full rank.

Λ—The diagonal matrix containing the eigenvalues $\lambda_1, \ldots, \lambda_n$.

Assumptions. *(A, B) is controllable and $\lambda_1, \ldots, \lambda_n$ is a self-conjugate set.*

Output. *K—The feedback matrix such that the spectrum of $A - BK$ is the set $\{\lambda_1, \ldots, \lambda_n\}$, and the condition number of the eigenvector matrix is as small as possible.*

Step 1. *Decompose the matrix B to determine U_0, U_1, and Z as in* (11.6.3).

Construct orthonormal bases, comprised of the columns of matrices S_j and $\hat{S}_j$ for the space $s_j = N\{U_1^{\mathrm{T}}(A - \lambda_j I)\}$ and its complement $\hat{s}_j$, for $\lambda_j \in s_j$, $j = 1, 2, \ldots, n$.

Step 2. *Select a set of n normalized vectors $x_1, \ldots, x_n$ from the space s_j such that $X = (x_1, \ldots, x_n)$ is well-conditioned.*

Step 3. *Compute $M = A - BK$ by solving the linear systems: $MX = X\Lambda$.*

Step 4. *Compute K : $K = Z^{-1}U_0^{\mathrm{T}}(A - M)$.*

Example 11.6.1. Consider the REVA with the following data:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \qquad B = \begin{pmatrix} 6 & 3 \\ 1 & 2 \\ 8 & 9 \end{pmatrix}, \qquad \Lambda = \operatorname{diag}(9, 5, 1).$$

Step 1.

$$U_0 = \begin{pmatrix} -0.5970 & 0.7720 \\ -0.0995 & -0.3410 \\ -0.7960 & -0.5364 \end{pmatrix}, \qquad U_1 = \begin{pmatrix} -0.2181 \\ -0.9348 \\ 0.2804 \end{pmatrix}$$

$$Z = \begin{pmatrix} -10.0499 & -9.1543 \\ 0 & -3.1934 \end{pmatrix}.$$

Step 2. $X = \begin{pmatrix} -0.0590 & 0.9859 & -0.0111 \\ -0.7475 & 0.0215 & -0.8993 \\ -0.6617 & -0.1657 & 0.4371 \end{pmatrix}$.

Step 3. $M = \begin{pmatrix} 5.0427 & 0.0786 & 0.2640 \\ 0.9987 & 3.7999 & 5.7856 \\ 0.1399 & 2.0051 & 6.1575 \end{pmatrix}$.

Step 4. $K = \begin{pmatrix} -1.8988 & 0.0269 & 0.5365 \\ 2.4501 & 0.5866 & -0.1611 \end{pmatrix}$.

Verify: The eigenvalues of $(A - BK)$ are 5, 9, 1.
$\mathrm{Cond}_2(K) = 6.3206$.

Some Implementational Details

Implementation of Step 1: Decomposition of B in Step 1 of the algorithm amounts to the QR factorization of B. Once this decomposition is performed, constructions of the bases can be done either by QR factorization of $(U_1^{\mathrm{T}}(A - \lambda_j I))^{\mathrm{T}}$ or by computing its SVD.

If QR factorization is used, then

$$(U_1^{\mathrm{T}}(A - \lambda_j I))^{\mathrm{T}} = (\hat{S}_j, S_j) \begin{pmatrix} R_j \\ 0 \end{pmatrix}$$

Thus, S_j and $\hat{S}_j$ are the matrices whose columns form the required orthonormal bases.

If SVD is used, then from

$$U_1^{\mathrm{T}}(A - \lambda_j I) = T_j\,(\Gamma_j, 0)(\hat{S}_j, S_j)^{\mathrm{T}},$$

we see that the columns of S_j and $\hat{S}_j$ form the required orthonormal bases. Here Γ_j is the diagonal matrix containing the singular values.

Note: The QR decomposition, as we have seen, is more efficient than the SVD approach.

Implementation of Step 2: **Step 2 is the key step in the solution process.** Kautsky *et al.* (1985) have proposed four methods to implement Step 2. Each of these four methods aims at minimizing a different measure ν of the sensitivity.

We present here only one (Method 0 in their paper), which is the simplest and most natural one. This method is designed to minimize the measure $\nu_2 = \mathrm{Cond}_2(X)$.

First, we note that $\mathrm{Cond}_2(X)$ will be minimized if each vector $x_j \in s_j$, $j = 1, 2, \ldots, n$ is chosen such that the angle between x_j and the space

$$t_j = < x_i, i \neq j >$$

is maximized for all j. The symbol $< x_k >$ denotes the space spanned by the vectors x_k.

This can be done in an iterative fashion. Starting with an **arbitrary** set of n independent vectors $X = (x_1, \ldots, x_n)$, $x_j \in s_j$, $j = 1, \ldots, n$, we replace each vector x_j by a new vector such that the angle to the current space t_j is maximized for each j. The QR method is again used to compute the new vectors as follows:

Find $\tilde{y}_j$ by computing the QR decomposition of

$$\begin{aligned} X_j &= (x_1, x_2, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n), \\ &= (\tilde{Q}_j, \tilde{y}_j)\begin{pmatrix} \tilde{R}_j \\ 0 \end{pmatrix}. \end{aligned}$$

and then compute the new vector

$$x_j = \frac{S_j S_j^{\mathrm{T}} \tilde{y}_j}{\| S_j^{\mathrm{T}} \tilde{y}_j \|_2}.$$

Note that with this choice of x_j, the condition $c_j = 1/|\tilde{y}_j^{\mathrm{T}} x_j|$ is minimized. The n steps of the process required to replace successively n vectors x_1 through x_n will constitute a sweep.

At the end of each sweep, $\mathrm{Cond}_2(X)$ is measured to see if it is acceptable; if not, a new iteration is started with the current X as the starting set. The iteration is continued until $\mathrm{Cond}_2(X)$, after a full sweep of the powers ($j = 1, 2, \ldots n$) is less than some positive tolerance.

Implementations of Step 3 and Step 4: Implementations of Step 3 and Step 4 are straightforward. M in Step 3 is computed by solving linear systems: $X^{\mathrm{T}}M^{\mathrm{T}} = \Lambda^{\mathrm{T}}X^{\mathrm{T}}$ using Gaussian elimination with partial pivoting.

K in Step 4 is computed by solving upper triangular systems:

$$ZK = U_0^{\mathrm{T}}(A - M).$$

Flop-count: Step 1: $O(n^3m)$ flops, Step 2: $O(n^3) + O(n^2m)$ flops per *sweep*, Step 3: $O(n^3)$ flops, and Step 4: $O(mn^2)$ flops.

MATCONTROL note: Algorithm 11.6.1 has been implemented in MATCONTROL function **polerob**. It computes both the feedback matrix K and the transforming matrix X.

Remarks on convergence of Algorithm 11.6.3 and the Tits–Yang Algorithm

- Each step of the above iteration amounts to rank-one updating of the matrix X such that the sensitivity of the eigenvalue λ_j is minimized. **However, this does not necessarily mean that the overall conditioning is improved at each step**. This is because the conditioning of the other eigenvalues $(\lambda_i, i \neq j)$ will be disturbed when the old vector x_j is replaced by the new vector.
- It was, thus, stated by Kautsky *et al.* (1985) that "the process does not necessarily converge to a fixed point." It, however, turned out to be the case of "slow convergence" only. Indeed, Tits and Yang (1996) later gave a proof of the convergence of the algorithm. Tits and Yang (1996) observed that this algorithm amounts to maximize, at each iteration, the determinant of the candidate eigenvector matrix X with respect to one of its column (subject to the constraints that it is still an eigenvector matrix of the closed-loop system). Based on this observation, Tits and Yang developed a more efficient algorithm by maximizing $\det(X)$ with respect to two columns concurrently. The **Tits–Yang algorithm** can easily be extended to assign the eigenvalues with complex conjugate pairs. For details of these algorithms, we refer the readers to the paper by Tits and Yang (1996). There also exists software called **robpole** based on the Tits–Yang algorithm.

MATLAB note: The MATLAB function **place** has implemented Algorithm 11.6.1.

Given a controllable pair (A, B) and a vector p containing the eigenvalues to be assigned, $K = \mathbf{place}(A, B, p)$ computes the feedback matrix K such that $(A - BK)$ has the desired eigenvalues. The software **robpole**, based on the Tits–Yang algorithm, is available in SLICOT (see Section 11.10).

Some Properties of the Closed-Loop System

The minimization of the condition number of the eigenvector matrix leads to some desirable robust properties of the closed-loop system. We state some of these properties below. The proofs can be found in Kautsky *et al.* (1985) or the readers can work out the proofs themselves.

Theorem 11.6.2.

(i) The gain matrix K obtained by Algorithm 11.6.1 *satisfies the inequality*

$$\|K\|_2 \leq (\|A\|_2 + \max_j |\lambda_j| \text{Cond}_2(X))/(\sigma_{\min}(B)) = k',$$

where $\sigma_{\min}(B)$ denotes the smallest singular value of B.

(ii) The transient response $x(t)$ satisfies

$$\|x(t)\|_2 \leq \text{Cond}_2(X) \max_j\{|e^{\lambda_j t}|\}.\|x_0\|_2,$$

where $x(0) = x_0$ or in the discrete case

$$\|x(k)\|_2 \leq \text{Cond}_2(X) \cdot \max_j\{|\lambda_j|^k \cdot \|x_0\|_2\}.$$

Example 11.6.2. For Example 11.6.1, we easily see that

$$\|K\|_2 = 3.2867, \qquad k' = 12.8829.$$

Thus, the result of part (i) of Theorem 11.6.2 is verified.

Theorem 11.6.3. *If the feedback matrix K assigns a set of stable eigenvalues λ_j, then the perturbed closed-loop matrix $A - BK + \Delta$ remains stable for all perturbations Δ that satisfy*

$$\|\Delta\|_2 \leq \min_{s=j\omega} \sigma_{\min}(sI - A + BK) = \delta(K).$$

Furthermore, $\|\delta(K)\| \leq \min_j \text{Re}(-\lambda_j)/\text{Cond}_2(X)$.

In the discrete-case, the closed-loop system remains stable for perturbations Δ such that

$$\|\Delta\|_2 \leq \min_{s=\exp(i\omega)} \{sI - A + BK\} = \Delta(F)$$

and $\Delta(F) \geq \min_j(1 - |\lambda_j|)/\text{Cond}_2(X)$.

Minimum-norm robust pole assignment: We stated in the previous section that the norm of the feedback matrix is another important factor that influences the sensitivity of closed-loop poles. Thus, it is important to consider this aspect of REVA as well.

The REVA with respect to minimizing the norm of the feedback matrix has been considered by Keel *et al.* (1985) and more recently by Varga (2000).

Both algorithms are Sylvester equation based (see **Exercise 11.11** for the statement of a Sylvester equation based EVA algorithm). The paper by Keel *et al.* (1985) addresses minimization of the performance index

$$I - \text{Trace}(K^{\mathrm{T}}K),$$

whereas Varga (2000) considers the minimization of the performance index

$$J = \frac{\alpha}{2}(\|X\|_F^2 + \|X^{-1}\|_F) + \frac{1-\alpha}{2}\|K\|_F^2$$

Note that minimizing J as above for $0 < \alpha < 1$ amounts to simultaneous minimization of the norm of the feedback matrix K and of the condition number of the eigenvector matrix X (with respect to the Frobenius norm).

For space limitations, we are not able to describe these algorithms here. The readers are referred to the papers by Keel *et al.* (1985) and Varga (2000). There also exists a software, based on the Varga algorithm, called "**sylvplace**" (available from Dr. Varga (E-mail: andras.varga@dlr.de).

11.7 COMPARISON OF EFFICIENCY AND STABILITY: THE SINGLE-INPUT EVA PROBLEM

Table 11.1: Comparison of efficiency and stability of a single-input EVA problem

Method	**Efficiency: Flop-count (Approximate) This count includes transformation of (A, b) to the controller-Hessenberg form**	**Numerical stability (backward stability and other features)**
The Recursive Algorithm (**Algorithm 11.2.1**)	$\frac{11}{3}n^3$	**Stability is not guaranteed**, but the algorithm allows the users to monitor the stability. **Reliable**
The RQ implementations of the recursive algorithm (**Algorithms 11.2.2** and **11.2.3**)	$5n^3$	**Stable**
The explicit QR method (**Miminis and Paige (1982)**)	$5n^3$	**Stable**
The implicit QR method (**Patel and Misra (1984)**)	$5n^3$	**Stable**
The eigenvector method (Petkov *et al.* 1984; **Exercise 11.8**).	$\frac{20}{3}n^3$	**Stable**

11.8 COMPARISON OF EFFICIENCY AND STABILITY: THE MULTI-INPUT EVA PROBLEM

Table 11.2: Comparison of efficiency and stability: the multi-input EVA problem

Method	**Efficiency: Flop-count (approximate).** These counts include transformation of (A, B) to the controller-Hessenberg form	**Numerical stability (backward stability) and other features**
The recursive algorithm **(Algorithm 11.3.1)**	$\frac{19}{3}n^3$	No formal round-off error analysis available. The algorithm is believed to be **reliable**
The explicit QR algorithm **(Section 11.3.2).**	$9n^3$	**Stable**
The implicit QR algorithm (the multi-input version of the implicit single-input QR algorithm of Patel and Misra (1984))	$9n^3$	Stability not formally proven, but is **believed to be stable**
The Schur method **(Algorithm 11.3.3)**	$30n^3$	Stability not formally proven, but is **believed to be stable.** The algorithm has an attractive feature that it can also be used for partial pole placement in the sense that it allows one to reassign only the "bad" eigenvalues, leaving the "good" ones unchanged
The eigenvector method (Petkov *et al.* 1986; not described in the book)	$\frac{40}{3}n^3$	**Stable**

11.9 COMPARATIVE DISCUSSION OF VARIOUS METHODS AND RECOMMENDATION

For the single-input problem: The recursive algorithm (**Algorithm 11.2.1**) is the fastest one proposed so far. It is also extremely simple to implement. Unfortunately,

the numerical stability of the algorithm cannot be guaranteed in all cases. The algorithm, however, allows the users to monitor the stability. **Algorithm 11.2.1 is thus reliable.** In a variety of test examples, this algorithm has done remarkably well, even for some ill-conditioned problems (e.g., see **Example 11.2.3**). The RQ implementation of the recursive algorithm (**Algorithm 11.2.3**), the explicit and implicit QR algorithms all have the same efficiency, and are numerically **stable.** The eigenvector algorithm (**Exercise 11.8**) if properly implemented, is also stable, but it is the most expensive one of all the single-input algorithms.

One important thing to note here is that there exist RQ implementations of all the single-input algorithms mentioned in Table 11.1 (Arnold and Datta 1998). These RQ implementations are much easier to understand and implement on computers. **We strongly recommend the use of RQ implementations of these algorithms.**

For the multi-input problem: The recursive algorithm (**Algorithm 11.3.1**) is again the fastest algorithm; however, no round-off stability analysis of this algorithm has been done yet. **The explicit QR algorithm described in Section 11.3.2 is stable.** The properly implemented eigenvector algorithm due to Petkov *et al.* (1986), is also stable but is more expensive than the explicit QR algorithm. The Schur algorithm (**Algorithm 11.3.3**) is the most expensive one. It is believed to be numerically stable. An important feature of this algorithm is that it can be used for partial pole assignment in the sense that it offers a choice to the user to place only the "bad" eigenvalues, leaving the "good" ones unchanged.

The REVA algorithm (**Algorithm 11.6.1**) exploits the freedom offered by the problem to minimize the conditioning of the eigenvector matrix which is a major factor for the sensitivity of the closed-loop poles. However, when a well-conditioned eigenvector matrix does not exist, the algorithm may give inaccurate results. When the eigenvector matrix is ill-conditioned, it may be possible to obtain more accurate results using other methods.

Based on the above observations, it is recommended that for the single-input problem, the recursive algorithm (**Algorithm 11.2.1**) *be tried first. In case of possible ill-conditioning of the matrix L, its RQ formulation* (**Algorithm 11.2.2**) *should be used.*

For the multi-input problem, the multi-input version of the recursive algorithm (Algorithm 11.3.1) *should be tried first.* If the algorithm appears to be unstable (as indicated by the condition number of the matrix L), the explicit QR algorithm (**Section 11.3.1**) is to be used.

It should, however, be noted that Algorithm 11.3.1 and the explicit QR algorithm, as stated here, might give complex feedback matrix. There now exists a modified version of Algorithm 11.3.1 (Carvalho and Datta 2001) that avoids complex arithmetic and this modified version has been implemented in MATCONTROL function **polercm**.

For REVA, the choices are either Algorithm 11.6.1 *or the Tits–Yang Algorithm. For PEVA, the choices are either the Schur algorithm (Algorithm* 11.3.3*), or the Sylvester equation algorithm (Algorithm* 11.3.4*).* Algorithm 11.6.1 does not handle complex EVA as such, but its implementation in MATLAB function '**place**' does in an ad hoc fashion. Numerical experimental results suggest the Tits–Yang algorithm "typically produce more robust design" than that constructed by Algorithm 11.6.1. For partial pole placement, Algorithm 11.3.4 seems to be very efficient and not computationally intensive.

11.10 SOME SELECTED SOFTWARE

11.10.1 MATLAB Control System Toolbox

Classical design tools
acker SISO pole placement
place MIMO pole placement.

11.10.2 MATCONTROL

POLERCS	Single-input pole placement using the recursive algorithm
POLEQRS	Single-input pole placement using the QR version of the recursive algorithm
POLERQS	Single-input pole placement using RQ version of the recursive algorithm
POLERCM	Multi-input pole placement using the recursive algorithm
POLERCX	Multi-input pole placement using the modified recursive algorithm that avoids complex arithmetic and complex feedback
POLEQRM	Multi-input pole placement using the explicit QR algorithm
POLESCH	Multi-input pole placement using the Schur decomposition
POLEROB	Robust pole placement.

11.10.3 CSP-ANM

Pole assignment

- The recursive algorithm is implemented as `StateFeedbackGains` [*system, poles,* `Method` → `Recursive`].
- The explicit QR algorithm is implemented as `StateFeedbackGains` [*system, poles,* `Method` → `QRDecomposition`].
- The Schur method is implemented as `StateFeedbackGains` [*system, poles,* `Method` → `SchurDecomposition`].

- The RQ implementation of the recursive single-input algorithm is implemented as `StateFeedbackGains` [*system, poles*, `Method` → `RecursiveRQDecomposition`].
- The implicit single-input RQ algorithm is implemented as `StateFeedbackGains` [*system, poles*, `Method` → `ImplicitRQDecomposition`].

11.10.4 SLICOT

Eigenvalue/eigenvector assignment

SB01BD Pole assignment for a given matrix pair (A, B)
SB01DD Eigenstructure assignment for a controllable matrix pair (A, B) in orthogonal canonical form
SB01MD State feedback matrix of a time-invariant single-input system
ROBPOLE Robust Pole Assignment (Additional function added in 2003).

11.10.5 MATRIX$_X$

Purpose: Calculate state feedback gains via pole placement for single-input continuous-time or discrete-time systems.
Syntax: KC = POLEPLACE (A, B, POLES) ...controller design
KE = POLEPLACE (A′, B′, POLES) ...estimator design

11.10.6 POLEPACK

A collection of MATLAB programs for EVA, developed by G.S. Miminis (1991). Available on **NETLIB**.

11.11 SUMMARY AND REVIEW

Statement of the EVA Problem

Given a pair of matrices (A, B), and the set $S = \{\lambda_1, \ldots, \lambda_n\}$, closed under complex conjugation, find a matrix K such that $\Omega(A - BK) = S$.

Here $\Omega(M)$ denotes the spectrum of M.

In the single-input case, the problem reduces to that of finding a row vector f^{T} such that

$$\Omega(A - bf^{\mathrm{T}}) = S.$$

Existence and Uniqueness

The EVA problem has a solution if and only if (A, B) is controllable. In the single-input case, the feedback vector, when it exists, is unique. In the multi-input case, when there exists a feedback matrix, there are many. Therefore, the existing freedom can be exploited to improve the conditioning of the solution and of the closed-loop eigenvectors.

Numerical Methods

There are many methods for the EVA problem. Only a few have been described here. These include:

- Recursive algorithms (**Algorithm 11.2.1** for the single-input problem and **Algorithm 11.3.1** for the multi-input problem).
- QR-type algorithms (**Algorithms 11.2.2, 11.2.3, and those described in Miminis and Paige (1982), and Patel and Misra (1984)**. For the single-input problem and the explicit QR method described in **Section 11.3.2** for the multi-input problem).
- The Schur algorithm (**Algorithm 11.3.3**) for the multi-input problem.
- PEVA (**Algorithm 11.3.4**).

Efficiency and Numerical Stability

The recursive algorithms are the most efficient algorithms. The computer implementations of these algorithms are extremely simple. The algorithms, however, do not have guaranteed numerical stability, except for the RQ version of the single-input recursive algorithm, which has been proved to be numerically stable (Arnold and Datta 1998).

In the single-input case, it has been proved (see Arnold and Datta (1998)), by forward round-off error analysis, that the stability of the recursive algorithm (**Algorithm 11.2.1**) can be monitored and it is possible for the user to know exactly when the algorithm starts becoming problematic. **It is thus reliable.** Similar results are believed to hold for the multi-input recursive algorithm as well. But no formal analysis in the multi-input case has yet been done.

The QR-type algorithms for single-input problems all have the same efficiency and are numerically stable.

For the multi-input problem, the Schur Algorithm (**Algorithm 11.3.3**) is the most expensive one. However, it has an important feature, namely, it allows one to place only the "bad" eigenvalues, leaving the "good" ones unchanged.

The explicit QR algorithm (**Section 11.3.2**) and the multi-input version of the single-input implicit QR algorithm (not described in this book) have the same efficiency. The explicit QR algorithm has been proven to be stable and the implicit QR algorithm is believed to be stable as well.

Explicit Solutions

An explicit expression for the unique feedback vector for the single-input EVA problem has been given using the recursive algorithm (**Algorithm 11.2.1**). This formula is

$$f = \frac{1}{\alpha}(H^{\mathrm{T}} - \lambda_1 I)(H^{\mathrm{T}} - \lambda_2 I)\cdots(H^{\mathrm{T}} - \lambda_n I)e_n,$$

where $H = (h_{ij})$ is the Hessenberg matrix of the controller-Hessenberg form of the pair (A, b) and $\alpha = \prod_{i=1}^{n-1} h_{i+1,i}$. In the multi-input case, the expression is rather complicated (see Arnold (1993)).

Conditioning of the Feedback Problem: From the explicit expression of the feedback vector f of the single-input EVA problem, it is clear that the Hessenberg single-input feedback problem is essentially a polynomial evaluation $\phi(H)$ at an unreduced Hessenberg matrix, where $\phi(x) = (\lambda - \lambda_1)(\lambda - \lambda_2)\cdots(\lambda - \lambda_n)$ is the characteristic polynomial of the closed-loop matrix.

A result on the Frechet derivative $D\phi(H)$ is first given in Ph.D. dissertation of Arnold (1993) and the condition numbers for the feedback problem are then defined using this derivative. Next, a condition number estimator for the problem is stated. It worked well on test examples. This estimator never underestimated the error and overestimated the error by less than 2 digits, in all 100 test examples of sizes varying from 10 to 50, both for ill-conditioned and well-conditioned problems.

In the multi-output case, **Theorem 11.4.2** gives the perturbation bound for the feedback matrix from which the absolute and relative condition numbers are defined (**Section 11.4.3**).

Conditioning of the Closed-Loop Eigenvalues

The major factors responsible for the sensitivity of the closed-loop eigenvalues have been identified in **Section 11.5.** These factors are: **the condition number of the eigenvector matrix of the closed-loop system, the distance to uncontrollability and the distance between the open-loop and the closed-loop eigenvalues, the conditioning of the feedback problem, and the norm of the feedback matrix.** The most important of them is the condition number of the eigenvector matrix.

Robust Eigenvalue Assignment

Given the pair (A, B) and the matrix $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, the problem is to find a nonsingular matrix X, and a matrix K satisfying

$$(A - BK)X = X\Lambda$$

such that $\mathrm{Cond}_2(X)$ is minimum. In view of the last sentence of the preceding paragraph, the REVA problem with respect to minimizing the condition number

of the eigenvector matrix is a very important practical problem. An algorithm (**Algorithm 11.6.1**) due to Kautsky *et al.* (1985) is given in Section 11.6. The algorithm requires constructions of orthonormal bases for a certain space and for its complement. The QR factorization or the SVD can be used for this purpose. An analysis of convergence and a more improved version of this algorithm can be found in Tits and Yang (1996).

Algorithm for Minimizing Feedback Norm

Our discussions on the conditioning of the closed-loop eigenvalues (**Section 11.5**) show that it is also important to have algorithms that minimize the norm of the feedback matrix.

For such algorithms, see Keel *et al.* (1985) and Varga (2000).

11.12 CHAPTER NOTES AND FURTHER READING

Many algorithms have been developed for solving the EVA by state feedback. A good account of these algorithms can be found in the recent book by Xu (1998).

The earlier algorithms, based on reduction to controller-canonical forms, turn out to be numerically unstable. For a reference of some of these earlier algorithms, see Miminis and Paige (1982, 1988).

For a comprehensive reference of the Hessenberg or controller-Hessenberg based algorithms, which are more numerically reliable, see the recent paper of Arnold and Datta (1998). For a matrix equation based algorithm for EVA, see Bhattacharyya and DeSouza (1982). For robust eigenvalue and eigenstructure assignment algorithms, see Cavin and Bhattacharyya (1983), Kautsky *et al.* (1985), Byers and Nash (1989), and Tits and Yang (1996). For REVA by output feedback, see Chu, *et al.* (1984) and references therein. The other algorithms include there is Tsui (1986), Valasek and Olgac (1995a, 1995b).

For algorithms that minimize the norm of the feedback matrix, see Keel *et al.* (1985) and Varga (2000). The EVA problem by output feedback is a difficult problem and only a few algorithms are available. See Misra and Patel (1989) for output feedback algorithms. For the EVA and eigenstructure algorithms for descriptor systems (not discussed in this chapter), see Fletcher *et al.* (1986), Chu (1988), and Kautsky *et al.* (1989), Varga (2000). For partial pole-assignment algorithms, see Varga (1981), Saad (1988), and Datta and Saad (1991), Datta and Sarkissian (2000).

For round-off error analysis of various algorithms for EVA, see Cox and Moss (1989, 1992), Arnold and Datta (1998), and Miminis and Paige (1988).

The perturbation analysis for the single-input feedback problem was considered by Arnold (1993) and our discussion in the multi-input feedback problem has been taken from Sun (1996).

For discussions on conditioning of the EVA problem, see He *et al.* (1995), Mehrmann and Xu (1996, 1997, 1998), Konstantinov and Petkov (1993), Calvetti *et al.* (1999).

For an extension of the single-input recursive algorithm to assigning Jordan canonical form (JCF), companion and Hessenberg forms, etc., see Datta and Datta (1990). For parallel algorithms for the EVA problem, see Bru *et al.* (1994c), Coutinho *et al.* (1995), Datta and Datta (1986), Datta (1991), Baksi *et al.* (1994).

Now, there also exists a block algorithm (Carvalho and Datta 2001) for the multi-input EVA. **This block algorithm, besides being suitable for high-performance computing, is guaranteed to give a real feedback matrix.**

Exercises

11.1 Modify both the single-input (**Algorithm 11.2.1**) so that the use of complex arithmetic can be avoided (consult Carvalho and Datta (2001)).

11.2 *Single-input pole placement via linear systems.* Consider the following algorithm (Datta and Datta (1986)) for the single-input Hessenberg problem $(H, \hat{b})$, where H is an unreduced upper Hessenberg matrix and $\hat{b} = (\alpha, 0, \cdots, 0)^{\mathrm{T}}, \alpha \neq 0$. Let $\{\mu_i\}_{i=1}^{n}$ be the eigenvalues to be assigned.

Step 1. Solve the $n \times n$ Hessenberg systems:

$$(H - \mu_i I)t_i = \hat{b}, \quad i = 1, 2, \ldots, n$$

Step 2. Solve for d:

$$T^{\mathrm{T}}d = r,$$

where $T = (t_1, t_2, \ldots, t_n)$ and $r = (\alpha, \alpha, \ldots, \alpha)^{\mathrm{T}}$.

Step 3. Compute $f^{\mathrm{T}} = \dfrac{1}{\alpha}d^{\mathrm{T}}$.

(a) Give a proof of this algorithm, that is, prove that $\Omega(H - \hat{b}f^{\mathrm{T}}) = \{\mu_1, \ldots, \mu_n\}$; making necessary assumptions. Do an illustrative example.
(**Hint:** Take $\Lambda = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_n)$ in the proof of Algorithm 11.2.1 and follow the lines of the proof there.)

(b) Prove that T in Step 2 is nonsingular if the entries in the set $\{\mu_1, \mu_2, \ldots, \mu_n\}$ are pairwise distinct and none of them is an eigenvalue of H.

11.3 Consider the following modification of Algorithm in Exercise 11.2, proposed by Bru *et al.* (1994a):

Step 1. For $i = 1, 2, \ldots, n$ do

If μ_i is not in the spectrum of H, then solve the system

$$(H - \mu_i I)t_i = \hat{b}.$$

Else solve the system $(H - \mu_i I)t_i = 0$.

Step 2. Define the vector $u = (u_1, \ldots, u_n)^{\mathrm{T}}$ as follows:

$u_i = 1$, if μ_i is an eigenvalue of H,

$u_i = 0$, otherwise.

Step 3. Solve for f:

$$f^{\mathrm{T}}T = u^{\mathrm{T}},$$

where $T = (t_1, t_2, \ldots, t_n)$.

(a) Give a proof of this algorithm assuming that the pair $(H, \hat{b})$ is controllable and that the numbers in the set $\{\mu_1, \mu_2, \ldots, \mu_n\}$ are closed under complex conjugation and pairwise distinct. Do an illustrative example.

(b) Give an example to show that the assumption that $\mu_i, i = 1, \ldots, n$ are pairwise distinct, cannot be relaxed.

Note: Bru *et al.* (1994a) have given a more general algorithm which can assign multiple eigenvalues (algorithm III in that paper.

11.4 *Assigning canonical forms* (*Datta and Datta 1990*). Extend Algorithm 11.2.1 to the problems of assigning the following canonical forms: a companion matrix, an unreduced upper Hessenberg matrix, a Jordan matrix with no two Jordan blocks having the same eigenvalue. (**Hint:** Follow the line of the development of Algorithm 11.2.1 replacing Λ by the appropriate canonical form to be assigned). Do illustrative examples.

11.5 *Multi-input pole-placement via Linear systems* (*Datta 1989*). Develop a multi-input version of the Algorithm in Exercise 11.2, making necessary assumptions. (See Tsui (1986) and Bru *et al.* (1994b).)

11.6 Give a proof of Algorithm 11.2.3 (the RQ formulation of Algorithm 11.2.1). Consult Arnold and Datta (1998), if necessary.

11.7 Show that the explicit formula for the single-input pole assignment problem (Formula 11.2.9) is a Hessenberg-form of the Ackermann's formula.

11.8 *Eigenvector method for the pole-placement* (*Petkov et al. 1984*)

(a) Given the single-input controller-Hessenberg pair $(H, \hat{b})$, show that it is possible to find an eigenvector $\tilde{v}$, corresponding to an eigenvalue μ to be assigned, for the closed-loop matrix $H - bf^{\mathrm{T}}$, without knowing the feedback vector f.

(b) Let $\Lambda = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_n)$ be the matrix of the eigenvalues to be assigned, and V be the eigenvector matrix and $v^{(1)}$ be the first row of V. Assume that $\mu_i, i = 1, \ldots, n$ are all distinct. Prove that the feedback vector f can be computed from

$$f = \frac{1}{\alpha}(h_1 - v^{(1)}\Lambda V^{-1}),$$

where h_1 is the first row of H, and α is the 1st entry of $\hat{b}$.

(c) What are the possible numerical difficulties of the above method for computing f? Give an example to illustrate these difficulties.

(d) Following the same procedure as in the RQ formulation of **Algorithm 11.2.1**, work out an RQ version of the above eigenvector method. (Consult Arnold and Datta (1998), if necessary.) Compare this RQ version with the above formulation with respect to flop-count and numerical effectiveness.

11.9 Modify the Schur algorithm (**Algorithm 11.3.3**) for the multi-input problem to handle the case when the complex numbers in the matrix Γ are not so ordered that the ordering agrees with the diagonal structure of the matrix A_2. Work out a simple example with this modified Schur method.

11.10 Write MATLAB codes to implement the Algorithms 11.2.1 and 11.2.3 and those in Exercises 11.2, 11.3 and 11.8, and then using these programs, make a comparative study with respect to CPU time, flop-count, the norm of the feedback vector and the largest error-norm between the closed-loop and open-loop eigenvalues. Use randomly generated matrices.

11.11 *EVA via Sylvester matrix equation.* The following algorithm by Bhattacharyya and DeSouza (1982) solves the multi-input EVA problem:

Step 1. Pick a matrix G arbitrarily.

Step 2. Solve the Sylvester equation $AX - X\tilde{A} = -BG$, where $\tilde{A}$ is a matrix having the spectrum $\{\lambda_1, \ldots, \lambda_n\}$ to be assigned, for a full-rank solution X.

If the solution matrix X does not have full rank, return to Step 1 and pick another G.

Step 3. Compute the feedback matrix F by solving $FX = G$.

Give a proof of the algorithm and construct an example to illustrate the algorithm. (For the conditions on the existence of full-rank solution of the Sylvester equation, see Chapter 12 and the paper by DeSouza and Bhattacharyya (1981)).

11.12 Construct an example to demonstrate that even if the feedback (vector) for the single-input problem is computed reasonably accurately, the closed-loop eigenvalues may still differ from those to be assigned.

11.13 *Sensitivity analysis of the single-input pole-placement problem via Cauchy matrix (Mehmann and Xu 1998; Calvetti et al. 1999).* Let $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ with $\lambda_i \neq \lambda_j$ for $i \neq j$. Define $e = (1, 1, \ldots, 1)^{\mathrm{T}}$. Let $S = \{\mu_1, \mu_2, \ldots, \mu_n\}$, the eigenvalue set to be assigned; μ_i's are distinct and none of them is in the spectrum of Λ.

(a) Prove that the vector f_e defined by $f_e = C_h^{-T} e$ is such that $\Omega(\Lambda - e f_e^{\mathrm{T}}) = S$, where $C_h = (c_{ij})$ is the Cauchy matrix: $c_{ij} = 1/(\lambda_i - \mu_j)$.

(b) Show that $\text{Cond}_2(C_h)$ is the spectral condition number of the closed-loop matrix $\Lambda - e f_e^{\mathrm{T}}$.

(c) Using the result in (a) find an expression for the feedback vector f such that $\Omega(A - b f^{\mathrm{T}}) = S$, assuming that A is diagonalizable.

(d) Give a bound of the condition number of the eigenvector matrix of the closed-loop matrix $A - b f^{\mathrm{T}}$ in terms of the condition number of C_h, the condition number of the eigenvector matrix X of A, and the minimum and maximum entries of the vector $X^{-1}b$.

(e) Give a bound of the feedback vector f in terms of the norm of f_e and the norm of the inverse of the matrix XR, where $R = \text{diag}(\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_n)^{\mathrm{T}}$, and $X^{-1}b = (\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_n)^{\mathrm{T}}$.

(f) From the bounds obtained in (d) and (e), verify the validity of some of the factors responsible for the ill-conditioning of the single-input EVA problem, established in Section 11.5.

(g) Work out an example to illustrate (a)–(f).

11.14 From the expression (11.6.4) of the feedback matrix K, prove that if the condition of the eigenvector matrix is minimized, then a bound of the norm of the feedback matrix K is also minimized.

Give an example to show that this does not necessarily mean that the resulting feedback matrix will have the minimal norm.

11.15 (a) Perform a numerical experiment to demonstrate the slow convergence of Algorithm 11.6.1.

(b) Using MATCONTROL function **polerob** and **robpole** (from SLICOT), make a comparative study between Algorithms 11.6.1, and the Tits–Yang algorithm with respect to number of iterations, $\text{Cond}_2(X)$, and $\|K\|_2$. Use data of Example 11.6.1 and randomly generated matrices.

11.16 *Deadbeat control.* Given the discrete-system:

$$x_{i+1} = Ax_i + Bu_i,$$

the problem of "deadbeat" control is the problem of finding a state feedback $u_i = -Kx_i + v_i$ such that the resulting system:

$$x_{i+1} = (A - BK)x_i + v_i$$

has the property that $(A - BK)^p = 0$ for some $p < n$ and $\text{rank}(B) > 1$.

The solution of the homogeneous part of the closed-loop system then "dies out" after p steps; and that is why the name **deadbeat control**.

A numerically reliable algorithm for the deadbeat control has been provided by Van Dooren (1984). The basic idea of the algorithm is as follows:
If

$$\left(H = (H_{ij}),\ \tilde{B} = \begin{pmatrix} B_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \right)$$

is the controller-Hessenberg pair of (A, B), then the solution of the problem is equivalent to finding a feedback matrix K such that

$$V^{\mathrm{T}}(H - \hat{B}K)V = \begin{pmatrix} 0 & H_{12} & \cdots & \cdots & \cdots & H_{1p} \\ 0 & & H_{23} & \cdots & \cdots & H_{2p} \\ \vdots & \ddots & & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \ddots & & H_{p-1,p} \\ 0 & & & & & 0 \end{pmatrix},$$

for some orthogonal matrix V. The form on the right-hand side is called the **"deadbeat" form.**

Note that in this case $(H - \hat{B}K)^p = 0$. Van Dooren's algorithm finds K recursively in p steps. At the end of the ith step, one obtains the matrices V_i^{T} and K_i such that

$$V_i^{\mathrm{T}}(H - \hat{B}K_i)V_i = \begin{pmatrix} H_d^i & * \\ 0 & H_h^i \end{pmatrix},$$

where the matrix H_d^i is in "deadbeat" form again, and the pair $(H_h^i,\ \hat{B}_h^i)$; $V_i^{\mathrm{T}}\hat{B} = \begin{pmatrix} \hat{B}_d^i \\ \hat{B}_h^i \end{pmatrix}$, is still in block-Hessenberg form.

Develop a scheme for obtaining V_i and hence complete the algorithm for "deadbeat" control problem. (Consult Van Dooren's paper as necessary).

Work out an illustrative example.

11.17 Using random matrices A of order $n = 5, 10, 15, 20$, and 30, and appropriate input matrices B, make a comparative study between Algorithm 11.3.1, one in Section 11.3.2, and those in Exercises 11.5 and 11.10 with respect to CPU time, flop-count, accuracy of the closed-loop eigenvalues, and norms of feedback matrices.

Research problems

11.1 Carry out a round-off error analysis of the implicit QR algorithm of Patel and Misra (1984) to establish the numerical stability of the algorithm.

11.2 Carry out a round-off error analysis of the recursive multi-input algorithm of Arnold and Datta (1990) (**Algorithm 11.3.1**). The algorithm is believed to be reliable in practice. Prove or disprove this using the results of your analysis.

11.3 An explicit expression for the family of feedback matrices for the multi-input EVA problem has been given in Arnold (1993). Use this expression to establish the fact that the sensitivities of the closed-loop eigenvalues depend upon the nearness of the system to an uncontrollable system, the separation of the open-loop and the closed-loop eigenvalues, and the ill-conditioning of the closed-loop eigenvector matrix.

11.4 Work out an RQ version of the recursive algorithm for the multi-input EVA problem by Arnold and Datta (1990) (**Algorithm 11.3.1**).

11.5 Carry out a round-off error analysis of the Schur algorithm of Varga (**Algorithm 11.3.3**) to establish the numerical stability of the algorithm.

11.6 In the QR algorithm of Miminis and Paige (1988) for the multi-input EVA problem, explicit shifting is used for the allocation of each eigenvalue. Work out an **implicit version** of this algorithm.

References

Ackermann J. "Der entwurf linear regelungssysteme im zustandsraum. *Regelungstechnik und prozessedatenverarbeitung*," Vol. 7, pp. 297–300, 1972.

Arnold M. *Algorithms and Conditioning for Eigenvalue Assignment*, Ph.D. dissertation, Northern Illinois University, DeKalb, May 1993.

Arnold M. and Datta B.N. "An algorithm for the multi input eigenvalue assignment problem," *IEEE Trans. Autom. Control*, Vol. 35(10), pp. 1149–1152, 1990.

Arnold M. and Datta B.N. "The single-input eigenvalue assignment algorithms: A close-look," *SIAM J. Matrix Anal. Appl.*, Vol. 19(2), pp. 444–467, 1998.

Baksi D., Datta K.B. and Roy G.D. "Parallel algorithms for pole assignment of multi-input systems," *IEEE Proc. Control Theory Appl.*, Vol. 141(6), pp. 367–372, 1994.

Bhattacharyya S.P. and DeSouza E. "Pole assignment via Sylvester's equation," *Syst. Control Letter.*, Vol. 1, pp. 261–283, 1982.

Bru R., Mas J. and Urbano A. "An Algorithm for the single input pole assignment problem," *SIAM J. Matrix Anal. Appl.*, Vol. 15, pp. 393–407, 1994a.

Bru R., Cerdan J. and Urbano A. "An algorithm for the multi-input pole assignment problem, *Lin. Alg. Appl.*," Vol. 199, pp. 427–444, 1994b.

Bru R., Cerdan J., Fernandez de Cordoba P. and Urbano A. "A parallel algorithm for the partial single-input pole assignment problem," *Appl. Math. Lett.*, Vol. 7, pp. 7–11, 1994c.

Byers R. and Nash S.G. "Approaches to robust pole assignment," *Int. J. Control*, Vol. 49(1) pp. 97–117, 1989.

Calvetti D., Lewis B. and Reichel L. "On the selection of poles in the single-input pole placement problem," *Lin. Alg. Appl.*, Vols. 302–303, pp. 331–345, 1999.

Carvalho J. and Datta B.N. "A block algorithm for the multi-input eigenvalue assignment problem," *Proc. IFAC/IEEE Sym. Syst., Struct. and Control*, Prague, 2001.

Cavin R.K. and Bhattacharyya S. P. "Robust and well conditioned eigenstructure assignment via Sylvester's equation," *Optim. Control Appl. Meth.*, Vol. 4, pp. 205–212, 1983.

Chu E.K., Nichols N.K. and Kautsky J. "Robust pole assignment for output feedback," *Proceedings of the fourth IMA Conference on Control Theory*, 1984.

Chu K.-W.E. "A controllability condensed form and a state feedback pole assignment algorithm for descriptor systems," *IEEE Trans. Autom., Control*, Vol. 33, pp. 366–370, 1988.

Coutinho M. G., Bhaya A. and Datta B. N. "Parallel algorithms for the eigenvalue assignment problem in linear systems," *Proc Int. Conf. Control and Inform. Hong Kong*, pp. 163–168, 1995.

Cox C.L. and Moss W.F. "Backward error analysis for a pole assignment algorithm," *SIAM J. Matrix Anal. Appl.*, Vol. 10(4), pp. 446–456, 1989.

Cox C.L. and Moss W.F. "Backward error analysis of a pole assignment algorithm II: The Complex Case," *SIAM J. Matrix Anal. Appl.*, Vol. 13, pp. 1159–1171, 1992.

Datta B.N. "Parallel and large-scale matrix computations in control: Some ideas," *Lin. Alg. Appl.*, Vol. 121, pp. 243–264, 1989.

Datta B.N. and Datta K. Efficient parallel algorithms for controllability and eigenvalue assignment problems, *Proc. IEEE Conf. Dec. Control*, Athens, Greece, pp. 1611–1616, 1986.

Datta B.N. "An algorithm to assign eigenvalues in a Hessenberg matrix: single input case," *IEEE Trans. Autom. Contr.*, Vol. AC-32(5), pp. 414–417, 1987.

Datta B.N. and Datta K. "On eigenvalue and canonical form assignments." *Lin. Alg. Appl.*, Vol. 131, pp. 161–182, 1990.

Datta B.N. Parallel algorithms in control theory, *Proc. IEEE Conf. on Dec. Control*, pp. 1700–1704, 1991.

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

Datta B.N. and Saad Y. "Arnoldi methods for large Sylvester-like matrix equations and an associated algorithm for partial spectrum assignment," *Lin. Alg. Appl.*, Vol. 154–156, pp. 225–244, 1991.

Datta B.N. and Sarkissian D.R. "Partial eigenvalue assignment in linear systems: Existence, uniqueness and numerical solution," *Proc. Math. Theory of Networks and Sys.*, (MTNS'02), Notre Dame, August, 2002.

Datta K. "The matrix equation $XA - BX = R$ and its applications," *Lin. Alg. Appl.*, Vol. 109, pp. 91–105, 1988.

DeSouza E. and Bhattacharyya S.P. "Controllability, observability and the solution of $AX - XB = C$," *Lin. Alg. Appl.*, Vol. 39, pp. 167–188, 1981.

Fletcher L.R., Kautsky J., and Nichols N.K. "Eigenstructure assignment in descriptor systems," *IEEE Trans. Autom. Control*, Vol. AC-31, pp. 1138–1141, 1986.

He C., Laub A.J. and Mehrmann V. *Placing plenty of poles is pretty preposterous, DFG-Forschergruppe Scientific Parallel Computing, Preprint 95–17*, Fak. f. Mathematik, TU Chemnitz-Zwickau, D-09107, Chemnitz, FRG, 1995.

Higham N.J. *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

Kautsky J., Nichols N.K. and Chu K.-W.E. "Robust pole assignment in singular control systems," *Lin. Alg. Appl.*, Vol. 121, pp. 9–37, 1989.

Kautsky J., Nichols N.K. and Van Dooren P. "Robust pole assignment in linear state feedback," *Int. J. Control*, Vol. 41(5), pp. 1129–1155, 1985.

Keel L.H., Fleming J.A. and Bhattacharyya S.P. Pole assignment via Sylvester's equation, in *Contemporary Mathematics*, (Brualdi R., *et al.*, eds.), Vol. 47, pp. 265, 1985, American Mathematical Society, Providence, RI.

Konstantinov M.M. and Petkov P. "Conditioning of linear state feedback," *Technical Report:* 93–61, Dept. of Engineering, Leicester University, 1993.

Laub A.J. and Linnemann A. "Hessenberg forms in linear systems theory," in *Computational and Combinatorial Methods in Systems Theory*, (Byrnes C.I. and Lindquist A., eds.), pp. 229–244, Elsevier Science publishers, North-Holland, 1986.

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

Mehrmann V. and Xu H. "An analysis of the pole placement problem I: The single-input case," *Electron. Trans. Numer. Anal.*, Vol. 4, pp. 89–105, 1996.

Mehrmann V. and Xu H. "An analysis of the pole placement problem II: The multi-input Case," *Electron. Trams. Numer. Anal.*, Vol. 5, pp. 77–97, 1997.

Mehrmann V. and Xu H. "Choosing the poles so that the single-input pole placement problem is well-conditioned," *SIAM J. Matrix Anal. Appl.*, 1998.

Miminis G.S. and Paige C.C. "An algorithm for pole assignment of time-invariant linear systems." *Int. J. Control*, Vol. 35(2), pp. 341–354, 1982.

Miminis G.S. and Paige C.C. "A direct algorithm for pole assignment of time-invariant multi-input linear systems using state feedback," *Automatica*, Vol. 24(3), pp. 343–356, 1988.

Miminis G.S. *Polepack*, A collection of MATLAB programs for eigenvalue assignment, available on *NETLIB* (www.netlib.org), 1991.

Misra P. and Patel R.V. "Numerical algorithms for eigenvalue assignment by constant and dynamic output feedback," *IEEE Trans. Autom. Control*, Vol. 34(6), pp. 579–580, 1989.

Patel R.V. and Misra P. "Numerical algorithms for eigenvalue assignment by state feedback," *Proc. IEEE*, Vol. 72(12), pp. 1755–1764, 1984.

Petkov P., Christov N.D. and Konstantinov M.M. "A computational algorithm for pole assignment of linear multi input systems," *IEEE Trans. Autom. Control*, Vol. AC-31(11), pp. 1044–1047, 1986.

Petkov P., Christov N.D. and Konstantinov M.M. "A computational algorithm for pole assignment of linear single-input systems," *IEEE Trans. Autom. Contr.*, Vol. AC-29(11), pp. 1045–1048, 1984.

Rice J. "Theory of Conditioning," *SIAM J. Numer. Anal.*, Vol. 3(2), pp. 287–311, 1966.

Saad Y. "Projection and deflation methods for partial pole assignment in linear state feedback," *IEEE Trans. Autom. Control*, Vol. 33, pp. 290–297, 1988.

Shafai B. and Bhattacharyya S.P. "An algorithm for pole placement in high-order multivariable systems," *IEEE Trans. Autom. Control*, Vol. 33, 9, pp. 870–876, 1988.

Stewart G.W. *Introduction to Matrix Computations*, Academic Press, New York, 1973.

J.-G. Sun "Perturbation analysis of the pole assignment problem," *SIAM J. Matrix Anal. Appl.*, Vol. 17, pp. 313–331, 1996.

Szidarovszky F. and Bahill A.T. *Linear Systems Theory*, CRC Press, Boca Raton, 1991.

Tits A.L. and Yang Y. Globally convergent algorithms for robust pole assignment by state feedback, *IEEE Trans. Autom. Control*, Vol. AC-41, pp. 1432–1452, 1996.

Tsui C.C. An algorithm for computing state feedback in multi input linear systems, *IEEE Trans. Autom. Control*, Vol. AC-31(3), pp. 243–246, 1986.

Valasek M. and Olgac N. "Efficient eigenvalue assignments for general linear MIMO systems," *Automatica*, Vol. 31, pp. 1605–1617, 1995a.

Valasek M. and Olgac N. "Efficient pole placement technique for linear time-variant SISO systems," *Proc. IEEE Control Theory Appl.*, Vol. 142, 451–458, 1995b.

Van Dooren P. "Deadbeat Control: A special inverse eigenvalue problem," *BIT*, Vol. 24, pp. 681–699, 1984.

Van Dooren P.M. and Verhaegen M. "On the use of unitary state-space transformations," *Contemporary Mathematics*, (Brualdi R. *et al.* eds.) Vol. 47, pp. 447–463, 1985. American Mathematical Society, Providence, RI.

Varga A. "A multishift Hessenberg method for pole assignment of single-input systems," *IEEE Trans. Autom. Control.*, Vol. 41, pp. 1795–1799, 1996.

Varga A. "Robust pole assignment for descriptor systems," *Proc. Math. Theory of Networks and Sys.* (MTNS '2000), 2000.

Varga A. "A Schur method for pole assignment," *IEEE Trans. Autom. Control*, Vol. AC-26(2), pp. 517–519, 1981.

Xu S.-F. *An Introduction to Inverse Algebraic Eigenvalue Problems*, Peking University Press, Peking, China, 1998.

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