Chapter 3

FAST STABLE SOLVERS FOR STRUCTURED LINEAR SYSTEMS

Ali H. Sayed

Shivkumar Chandrasekaran

3.1 INTRODUCTION

The derivation in Ch. 2 showed how to obtain a stable implementation of the generalized Schur algorithm for positive-definite structured matrices R and, also, how to solve $Rx = b$ when R is structured but still *positive definite.*

We now use this stable implementation of the Schur algorithm, and the same notation of Ch. 2, to solve in a stable manner linear systems of equations with possibly *indefinite* or even *nonsymmetric* structured coefficient matrices. In other words, we show how to use the stability results of the positive-definite case to derive stable solvers for the indefinite and nonsymmetric cases.

We shall focus on *shift-structured* coefficient matrices R, viz., those that satisfy displacement equations of the form

$$R - ZRZ^T = GB^T$$

for some (G, B). In fact, we shall denote the coefficient matrix throughout this chapter by T rather than R and write

$$T - ZTZ^T = GB^T. \tag{3.1.1}$$

The notation T is chosen for two reasons. First, this class of matrices includes as a special case Toeplitz matrices (which correspond to a special choice for G and B—see (2.3.5)). However, our results apply not only to Toeplitz coefficient matrices but, more generally, to shift-structured matrices T (obtained for other choices of G and B and also for multicolumn G and B). Second, the notation T is also chosen to avoid confusion with the notation R for the R factor in the QR factorization of a matrix. Such QR factorizations will be repeatedly invoked in this chapter.

The coefficient matrix T is not required to be symmetric or positive definite. It is required only to be a structured matrix in the sense defined by (3.1.1). It can thus

be indefinite or even nonsymmetric. Now given a linear system of equations $Tx = b$, one method for solving it fast is to compute the QR factorization of the coefficient matrix T rapidly. Several fast methods have been proposed for this purpose [BBH86], [CKL87], [Cyb83], [Cyb87], [Swe84] (see also Sec. 1.8.3) but none of them is numerically stable since the resulting Q matrix cannot be guaranteed to be orthogonal. In [CS98], however, the authors of this chapter showed how to circumvent this issue and derived an algorithm that is provably both fast *and* backward stable for solving $Tx = b$ for shift-structured matrices T that can be indefinite or even nonsymmetric.

The new algorithm relies on the observation that it is not really necessary to limit ourselves to only LDU or QR factorizations of the coefficient matrix T in order to solve the linear system of equations $Tx = b$. If other factorizations for T can be obtained in a fast and stable manner, and if they are also useful for solving the linear system of equations, then these factorizations should be pursued as well. In fact, the new algorithm, rather than returning Q, returns two matrices Δ and Q such that Δ is triangular and the product $\Delta^{-1}Q$ is "numerically orthogonal"; it provides a factorization for the coefficient matrix T that is of the form

$$T = \Delta(\Delta^{-1}Q)R.$$

This factorization is in terms of the three matrices $\{\Delta, Q, R\}$ and it is, of course, highly nonunique, since we can always replace Δ by any invertible matrix. The point, however, is that the new algorithm returns that particular Δ that allows us to compensate for the fact that Q is not numerically orthogonal. More important, these factors are then used to solve $Tx = b$ both fast and in a backward stable manner.

Our derivation is based on the following idea. We already know from the discussions in Ch. 2 how to develop a numerically reliable implementation of the generalized Schur algorithm for positive-definite structured matrices. This suggests that we should first transform a problem that involves a nonsymmetric or indefinite structured matrix T to an equivalent problem that involves sign-definite matrices only (either positive definite or negative definite). We achieve this by relying on a so-called embedding technique developed in [Chu89], [KC94] (see also Ch. 1). More specifically, we embed the square coefficient matrix T into a larger matrix of the form

$$M = \begin{bmatrix} T^T T & T^T \\ T & 0 \end{bmatrix}. \tag{3.1.2}$$

The product $T^T T$ is *never* formed explicitly. This definition for M is just for explanation purposes since the algorithm itself will end up working with a generator matrix for M and not with the entries of M, and this generator for M can be found from a generator for T, without the need to form $T^T T$ (see Sec. 3.4).

Now observe that the leading block of M is positive definite and the Schur complement of M with respect to the $(1,1)$ block is negative definite (and equal to $-I$). The matrix M also turns out to be structured (e.g., M is shift structured when T is quasi Toeplitz). In this case, and with proper modifications, the stability results of Ch. 2 can be applied.

3.2 OVERVIEW OF THE PROPOSED SOLUTION

Once the matrices $\{\Delta, Q, R\}$ are obtained, they are used to solve for x efficiently by using the expression

$$x = R^{-1}(Q^T\Delta^{-T})\Delta^{-1}b. \tag{3.2.1}$$

Note that in writing this expression we used the fact that $\Delta^{-1}Q$ is orthogonal and, hence, its inverse is the transpose matrix.

All computations in the above expression can be done in $O(n^2)$ operations, where n is the matrix dimension, and the algorithm turns out to be backward stable (cf. the definitions of stability in Ch. 4) in the sense that the computed solution $\widehat{x}$ is shown to satisfy an equation of the form

$$(T + \tilde{T})\widehat{x} = b,$$

where the norm of the error matrix $\tilde{T}$ satisfies

$$\|\tilde{T}\|_2 \leq O_n(\varepsilon)\, \|T\|_2 + O(\varepsilon^2),$$

where ε denotes machine precision.

The factorization for T is obtained as follows. We apply $2n$ steps of the generalized Schur algorithm to a generator of M in (3.1.2) and obtain its *computed* triangular factorization, which we partition in the form

$$\begin{bmatrix} \widehat{R}^T & 0 \\ \widehat{Q} & \Delta \end{bmatrix} \begin{bmatrix} \widehat{R} & \widehat{Q}^T \\ 0 & -\Delta^T \end{bmatrix},$$

where $\widehat{R}^T$ and Δ are $n \times n$ lower triangular matrices. The computed matrices $\{\widehat{R}, \widehat{Q}, \Delta\}$ are the quantities used in (3.2.1) to determine the computed solution $\widehat{x}$ in a backward stable manner.

From a numerical point of view, the above steps differ in a crucial way from the embeddings suggested in [Chu89], [KC94] and turn out to mark the difference between a numerically stable and a numerically unstable implementation. The discussion in [Chu89, pp. 37, 50, 52] and [KC94] is mainly concerned with fast procedures for the QR factorization of Toeplitz block and block Toeplitz matrices. It employs an embedding of the form

$$M = \begin{bmatrix} T^T T & T^T \\ T & I \end{bmatrix}, \tag{3.2.2}$$

where the identity matrix I in (3.2.2) replaces the zero matrix in our embedding (3.1.2). The derivation in [Chu89], [KC94] suggests applying n (rather than $2n$) steps of the generalized Schur algorithm to a generator of (3.2.2) and then using the resulting $\widehat{R}$ and $\widehat{Q}$ as the QR factors of T. However, numerical issues were not studied in [Chu89], [KC94], and it turns out that the above procedure *does not* guarantee a numerically orthogonal matrix $\widehat{Q}$ and therefore cannot be used to implement a stable solver for a linear system of equations $Tx = b$.

For this reason, we instead proposed in [CS98] to proceed with the previous embedding (3.1.2) since it seems difficult to obtain a stable algorithm that is based solely on the alternative embedding (3.2.2). We also apply $2n$ steps (rather than just n steps) of the generalized Schur algorithm to a generator of (3.1.2). This allows us to incorporate a correction procedure into the algorithm (by computing Δ), which is shown later to ensure backward stability when coupled with the other modifications that we discussed in Ch. 2 for stabilizing the generalized Schur algorithm.

3.3 THE GENERALIZED SCHUR ALGORITHM FOR INDEFINITE MATRICES

We described in Sec. 2.4 the array form of the Schur algorithm for symmetric positive definite structured matrices with displacement rank 2. Now, in view of the structure of

M, we shall need to apply the algorithm to a symmetric but possibly indefinite matrix with displacement rank larger than 2 with respect to a strictly lower triangular F, say,

$$M - FMF^T = GJG^T, \quad J = (I_p \oplus -I_q). \tag{3.3.1}$$

For this reason, and for ease of reference, we include here a statement of the algorithm for this particular case (this form is a special case of Alg. 1.7.1 of Ch. 1).

For displacement ranks larger than 2, we shall say that a generator matrix G is in *proper* form if its first nonzero row has a single nonzero entry, say, in the first column

$$G = \begin{bmatrix} x & 0 & 0 & 0 & 0 \\ x & x & x & x & x \\ x & x & x & x & x \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x & x & x & x & x \end{bmatrix} \tag{3.3.2}$$

or in the last column

$$G = \begin{bmatrix} 0 & 0 & 0 & 0 & x \\ x & x & x & x & x \\ x & x & x & x & x \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x & x & x & x & x \end{bmatrix}. \tag{3.3.3}$$

We note that in the statement below we are denoting the triangular factorization of a symmetric matrix M by $M = LDL^T$, where D is taken as a signature matrix (rather than $M = LD^{-1}L^T$ as in (1.6.1), where D is a diagonal matrix).

Algorithm 3.3.1 (Schur Algorithm for Indefinite Matrices). *Consider a symmetric strongly regular matrix $M \in \mathbb{R}^{n\times n}$ satisfying* (3.3.1).

- Input: *An $n \times n$ strictly lower triangular matrix F, an $n \times r$ generator $G_0 = G$, and $J = (I_p \oplus -I_q)$.*
- Output: *A lower triangular factor L and a signature matrix D such that $M = LDL^T$, where M is the solution of* (3.3.1) *(assumed $n \times n$).*
- Computation: *Start with $G_0 = G$, $F_0 = F$, and repeat for $i = 0, 1, \ldots, n-1$:*
  1. *Let g_i denote the top row of G_i.*
  2. *If $g_i J g_i^T > 0$ (we refer to this case as a positive step):*
     - ◇ *Choose a J-unitary rotation Θ_i that converts g_i to proper form with respect to the* first *column, i.e.,*

$$g_i\Theta_i = \begin{bmatrix} x & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.3.4}$$

     Let $\bar{G}_i = G_i\Theta_i$ (i.e., apply Θ_i to G_i).
     - ◇ *The ith column of L, denoted by l_i, is obtained by appending i zero entries to the first column of $\bar{G}_i$,*

$$l_i = \begin{bmatrix} 0 \\ \bar{G}_i \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix}. \tag{3.3.5}$$

     The ith signature is $d_i = 1$.

◇ *Keep the last columns of $\bar{G}_i$ unchanged and multiply the first column by F_i, where F_i denotes the submatrix obtained by deleting the first i rows and columns of F. This provides a new matrix whose first row is zero (since F_i is strictly lower triangular) and whose last rows are the rows of the next generator matrix G_{i+1}, i.e.,*

$$\begin{bmatrix} 0 \\ G_{i+1} \end{bmatrix} = \begin{bmatrix} F_i \bar{G}_i \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \bar{G}_i \begin{bmatrix} 0 & \\ & I \end{bmatrix} \end{bmatrix}. \tag{3.3.6}$$

3. *If $g_i J g_i^T < 0$ (we refer to this case as a negative step):*

◇ *Choose a J-unitary rotation Θ_i that converts g_i to proper form with respect to the* last *column, i.e.,*

$$g_i \Theta_i = \begin{bmatrix} 0 & 0 & 0 & 0 & x \end{bmatrix}. \tag{3.3.7}$$

Let $\bar{G}_i = G_i \Theta_i$ (i.e., apply Θ_i to G_i).

◇ *The ith column of L, denoted by l_i, is obtained by appending i zero entries to the last column of $\bar{G}_i$,*

$$l_i = \begin{bmatrix} 0 \\ \bar{G}_i \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix}. \tag{3.3.8}$$

The ith signature is $d_i = -1$.

◇ *Keep the first columns of $\bar{G}_i$ unchanged and multiply the last column by F_i. This provides a new matrix whose first row is zero (since F_i is strictly lower triangular) and whose last rows are the rows of the next generator matrix G_{i+1}, i.e.,*

$$\begin{bmatrix} 0 \\ G_{i+1} \end{bmatrix} = \begin{bmatrix} \bar{G}_i \begin{bmatrix} I & \\ & 0 \end{bmatrix} & F_i \bar{G}_i \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix}. \tag{3.3.9}$$

4. *The case $g_i J g_i^T = 0$ is ruled out by the strong regularity of M.*

◇

Again, after n steps, the algorithm provides the triangular decomposition

$$M = \sum_{i=0}^{n-1} d_i l_i l_i^T \tag{3.3.10}$$

at $O(rn^2)$ computational cost (see Ch. 1). Moreover, the successive matrices G_i that are obtained via the algorithm can be interpreted as follows (recall Remark 2 after Alg. 1.6.2). Let M_i denote the Schur complement of M with respect to its leading $i \times i$ submatrix; then

$$M_i - F_i M_i F_i^T = G_i J G_i^T. \tag{3.3.11}$$

Hence, G_i constitutes a generator matrix for the ith Schur complement M_i, which is therefore structured. Note further that $\bar{G}_i$ is also a generator matrix for the same Schur complement M_i since, due to the J-unitarity of Θ_i, we have $\bar{G}_i J \bar{G}_i^T = G_i \Theta_i J \Theta_i^T G_i^T = G_i J G_i^T$. We now address the main issues of this chapter.

3.4 FAST QR FACTORIZATION OF SHIFT-STRUCTURED MATRICES

As mentioned in (3.1.1), we shall focus on matrices $T \in \mathbb{R}^{n\times n}$ that are shift structured and therefore satisfy a displacement equation of the form

$$T - ZTZ^T = GB^T \tag{3.4.1}$$

for some generator pair (G, B). Consider, for now, the following alternative definition of a $3n \times 3n$ augmented matrix:

$$M = \begin{bmatrix} -I & T & 0 \\ T^T & 0 & T^T \\ 0 & T & 0 \end{bmatrix}. \tag{3.4.2}$$

The matrix M is also structured (as shown below) with respect to $Z_n \oplus Z_n \oplus Z_n$, where Z_n denotes the $n \times n$ lower shift triangular matrix (denoted earlier by Z—here we include the subscript n in order to explicitly indicate the size of Z).

It can be easily verified that $M - (Z_n \oplus Z_n \oplus Z_n)M(Z_n \oplus Z_n \oplus Z_n)^T$ is low rank since

$$M - (Z_n \oplus Z_n \oplus Z_n)M(Z_n \oplus Z_n \oplus Z_n)^T = \begin{bmatrix} -e_1e_1^T & GB^T & 0 \\ BG^T & 0 & BG^T \\ 0 & GB^T & 0 \end{bmatrix},$$

where $e_1 = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}^T$ is a basis vector of appropriate dimension. A generator matrix for M, with $3n$ rows and $(2r+1)$ columns, can be seen to be

$$\mathcal{G} = \frac{1}{\sqrt{2}} \begin{bmatrix} G & -G & e_1 \\ B & B & 0 \\ G & -G & 0 \end{bmatrix}, \quad \mathcal{J} = \begin{bmatrix} I_r & & \\ & -I_{r+1} & \end{bmatrix}. \tag{3.4.3}$$

That is,

$$M - \mathcal{F}M\mathcal{F}^T = \mathcal{G}\mathcal{J}\mathcal{G}^T,$$

where $\mathcal{F} = (Z_n \oplus Z_n \oplus Z_n)$ and $(\mathcal{G}, \mathcal{J})$ are as above.

The $n \times n$ leading submatrix of M is negative definite (in fact, equal to $-I$). Therefore, the first n steps of the generalized Schur algorithm applied to $(\mathcal{F}, \mathcal{G}, \mathcal{J})$ will be negative steps (cf. step 3 of Alg. 3.3.1). These first n steps lead to a generator matrix, denoted by $\mathcal{G}_n$ (with $2n$ rows), for the Schur complement of M with respect to its leading $n \times n$ leading submatrix, viz.,

$$M_n - (Z_n \oplus Z_n)M_n(Z_n \oplus Z_n)^T = \mathcal{G}_n\mathcal{J}\mathcal{G}_n^T, \tag{3.4.4}$$

where M_n is $2n \times 2n$ and equal to (what we denoted earlier in (3.2.2) by M)

$$M_n = \begin{bmatrix} T^TT & T^T \\ T & 0 \end{bmatrix}. \tag{3.4.5}$$

Clearly, M and its Schur complement M_n are related via the Schur complement relation:

$$M = \begin{bmatrix} I \\ -T^T \\ 0 \end{bmatrix} (-I) \begin{bmatrix} I & -T^T & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & T^TT & T^T \\ 0 & T & 0 \end{bmatrix}.$$

Therefore, $(\mathcal{G}_n, \mathcal{J})$ is a generator for M_n with respect to $(Z_n \oplus Z_n)$, as shown by (3.4.4).

The leading $n \times n$ submatrix of M_n is now positive definite (equal to $T^T T$). Therefore, the next n steps of the generalized Schur algorithm applied to $(Z_n \oplus Z_n, \mathcal{G}_n, \mathcal{J})$ will be positive steps (cf. step 2 of Alg. 3.3.1). These steps lead to a generator matrix, denoted by $\mathcal{G}_{2n}$ (with n rows), for the Schur complement of M with respect to its leading $2n \times 2n$ leading submatrix, viz.,

$$M_{2n} - Z_n M_{2n} Z_n^T = \mathcal{G}_{2n} \mathcal{J} \mathcal{G}_{2n}^T,$$

where M_{2n} is now $n \times n$ and equal to $-I$.

Again, M_n and M_{2n} are related via a (block) Schur complementation step, written as

$$\begin{bmatrix} T^T T & T^T \\ T & 0 \end{bmatrix} = M_n = \begin{bmatrix} R^T \\ Q \end{bmatrix} (I) \begin{bmatrix} R & Q^T \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -I \end{bmatrix}, \tag{3.4.6}$$

where we have denoted the first n columns of the triangular factor of M_n by

$$\begin{bmatrix} R^T \\ Q \end{bmatrix}$$

with R an $n \times n$ upper triangular matrix and Q an $n \times n$ matrix. The R and Q matrices are thus obtained by splitting the first n columns of the triangular factor of M_n into a leading lower triangular block followed by a full matrix Q.

By equating terms on both sides of (3.4.6) we can explicitly identify R and Q as follows:

$$T^T T = R^T R, \quad T = QR, \quad QQ^T - I = 0.$$

These relations show that Q and R define the QR factors of the matrix T.

In summary, the above discussion shows the following: given a shift-structured matrix T as in (3.4.1), its QR factorization can be computed efficiently by applying $2n$ steps of the generalized Schur algorithm to the matrices $(\mathcal{F}, \mathcal{G}, \mathcal{J})$ defined in (3.4.3). The factors Q and R can be obtained from the triangular factors $\{l_i\}$ for $i = n, n+1, \ldots, 2n-1$.

Alternatively, if a generator matrix is directly available for M_n in (3.4.5) (see the discussion of the Toeplitz case below), then we need only apply n Schur steps to this generator matrix and read the factors Q and R from the resulting n columns of the triangular factor.

In what follows we shall establish, for convenience of exposition, the numerical stability of a fast solver for $Tx = b$ that starts with a generator matrix for the embedding (3.4.5) rather than the embedding (3.4.2). It will become clear, however, that the same conclusions will hold if we instead start with a generator matrix for the embedding (3.4.2).

The augmentation (3.4.2) was used in [Say92], [SK95b], and it is based on embedding ideas originally pursued in [Chu89], [KC94] (see other augmentations below).

3.4.1 The Toeplitz Case

In some cases it is possible to find an explicit generator matrix for M_n. This saves the first n steps of the generalized Schur algorithm.

For example, consider the case when T is a Toeplitz matrix (which is a special case of (3.4.1) whose first column is $[t_0, t_1, \ldots, t_{n-1}]^T$ and whose first row is $[t_0, t_{-1}, \ldots, t_{-n+1}]$. Define the vectors

$$\begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} \triangleq \frac{Te_1}{\|Te_1\|_2}, \qquad \begin{bmatrix} s_0 \\ \vdots \\ s_{n-1} \end{bmatrix} \triangleq T^T \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix}.$$

It can be verified that a generator matrix for M_n in (3.4.5) is the following [Chu89]:

$$M_n - (Z_n \oplus Z_n) M_n (Z_n \oplus Z_n)^T = \mathcal{G}_n \mathcal{J} \mathcal{G}_n^T,$$

where $\mathcal{J}$ is 5×5,

$$\mathcal{J} = \text{diag}[1, 1, -1, -1, -1],$$

and $\mathcal{G}_n$ is $2n \times 5$,

$$\mathcal{G}_n = \begin{bmatrix} s_0 & 0 & 0 & 0 & 0 \\ s_1 & t_{-1} & s_1 & t_{n-1} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ s_{n-1} & t_{-n+1} & s_{n-1} & t_1 & 0 \\ c_0 & 1 & c_0 & 0 & 1 \\ c_1 & 0 & c_1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ c_{n-1} & 0 & c_{n-1} & 0 & 0 \end{bmatrix}.$$

3.4.2 Other Augmentations

It is possible to compute the QR factors of a structured matrix T satisfying (3.4.1) by using augmented matrices other than (3.4.2). For example, consider the $3n \times 3n$ augmented matrix

$$M = \begin{bmatrix} -I & T & 0 \\ T^T & 0 & T^T \\ 0 & T & I \end{bmatrix}, \tag{3.4.7}$$

where an identity matrix replaces the zero matrix in the $(3,3)$ block entry of the matrix in (3.4.2). A generator matrix for M, with $3n$ rows and $(2r+2)$ columns, is now

$$\mathcal{G} = \frac{1}{\sqrt{2}} \begin{bmatrix} G & 0 & -G & e_1 \\ B & 0 & B & 0 \\ G & e_1 & -G & 0 \end{bmatrix}, \quad \mathcal{J} = \begin{bmatrix} I_{r+1} & \\ & -I_{r+1} \end{bmatrix}.$$

If T is Toeplitz, as discussed above, then the rank of $\mathcal{G}$ can be shown to reduce to $2r = 4$ [Chu89]. (This is in contrast to the displacement rank 5 that follows from the earlier embedding (3.4.2).)

After $2n$ steps of the generalized Schur algorithm applied to the above $(\mathcal{G}, \mathcal{J})$, we obtain the following factorization (since now $M_{2n} = 0$),

$$M = \begin{bmatrix} I & 0 \\ -T^T & R^T \\ 0 & Q \end{bmatrix} \begin{bmatrix} -I & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ -T^T & R^T \\ 0 & Q \end{bmatrix}^T,$$

from which we can again read the QR factors of T from the triangular factors $\{l_i\}$ for $i = n, \ldots, 2n-1$. This augmentation was suggested in [Chu89, p. 37] and [KC94].

However, from a numerical point of view, computing the QR factors of a structured matrix T using the generalized Schur algorithm on the augmented matrices M in (3.4.2) or (3.4.7) is not stable. The problem is that the computed Q matrix is not necessarily orthogonal. This is also true for other procedures for fast QR factorization [BBH86], [Cyb83], [Cyb87], [Swe84].

In the next section we show how to overcome this difficulty and develop a fast and stable algorithm for solving linear systems of equations with shift-structured coefficient

matrices T. For this purpose, we proceed with the embedding suggested earlier in (3.4.2) since it seems difficult to obtain a stable algorithm that is based solely on the alternative embedding (3.4.7). The reason is that the embedding (3.4.2) allows us to incorporate a correction procedure into the algorithm in order to ensure stability.

We first derive a stable algorithm for a well-conditioned coefficient matrix and then modify it for the case when the coefficient matrix is ill-conditioned. The interested reader may consult the summary of the final algorithm in Sec. 3.7.

3.5 WELL-CONDITIONED COEFFICIENT MATRICES

In this section we develop a stable algorithm for the case of well-conditioned matrices T. A definition of what we mean by a well-conditioned matrix is given later (see (3.5.9)). Essentially this refers to matrices whose condition number is less than the reciprocal of the square root of the machine precision. Modifications to handle the ill-conditioned case are introduced later.

We start with an $n \times n$ (possibly nonsymmetric) shift-structured matrix T with displacement rank r,

$$T - Z_n T Z_n^T = GB^T, \tag{3.5.1}$$

and assume we have available a generator matrix $\mathcal{G}$ for the $2n \times 2n$ augmented matrix

$$M = \begin{bmatrix} T^T T & T^T \\ T & 0 \end{bmatrix}; \tag{3.5.2}$$

that is,

$$M - \mathcal{F} M \mathcal{F}^T = \mathcal{G} \mathcal{J} \mathcal{G}^T, \tag{3.5.3}$$

where $\mathcal{F} = (Z_n \oplus Z_n)$. Note that, for ease of exposition, we have modified our notation. While we earlier denoted the above matrix M by M_n and its generator by $\mathcal{G}_n$ and used $\mathcal{F}$ to denote $(Z_n \oplus Z_n \oplus Z_n)$, we are now dropping the subscript n from $(M_n, \mathcal{G}_n)$ and are using $\mathcal{F}$ to denote the $2n \times 2n$ matrix $(Z_n \oplus Z_n)$.

We discussed in Sec. 3.4.1 an example where we showed a particular generator matrix $\mathcal{G}$ for the above M when T is Toeplitz. (We repeat that the error analysis of later sections will still apply if we instead start with the $3n \times 3n$ embedding (3.4.2) and its generator matrix (3.4.3).)

We indicated earlier (at the end of Sec. 3.4) that by applying n steps of the generalized Schur algorithm to the matrix M in (3.5.2) we can obtain the QR factorization of T from the resulting n columns of the triangular factors of M. But this procedure is not numerically stable since the resulting Q is not guaranteed to be unitary. To fix this problem, we propose some modifications. The most relevant modification we introduce now is to run the Schur algorithm for $2n$ steps on M rather than just n steps. As suggested in Ch. 2, we also need to be careful in the application of the hyperbolic rotations. In particular, we assume that the hyperbolic rotations are applied using one of the methods suggested in that chapter such as mixed downdating, the OD method, or the H procedure.

The matrix T is required only to be invertible. In this case, the leading submatrix of M in (3.5.2) is positive definite, and therefore the first n steps of the generalized Schur algorithm will be positive steps. Hence, the hyperbolic rotations needed for the first n steps will perform transformations of the form (3.3.4), where generators are transformed into proper form with respect to their first column. Likewise, the Schur complement of M with respect to its leading submatrix $T^T T$ is equal to $-I$, which is negative

definite. This means that the last n steps of the generalized Schur algorithm will be negative steps. Hence, the hyperbolic rotations needed for the last n steps will perform transformations of the form (3.3.7), where generators are transformed into proper form with respect to their last column.

During a positive step (a similar discussion holds for a negative step), a generator matrix G_i will be reduced to proper form by implementing the hyperbolic transformation Θ_i as a sequence of orthogonal transformations followed by a 2×2 hyperbolic rotation (see also [SD97b]). The 2×2 rotation is implemented along the lines of [CS96] or Ch. 2, e.g., via mixed downdating [BBDH87], or the OD method, or the H procedure. Details are given below.

3.5.1 Implementation of the Hyperbolic Rotation

When the generalized Schur algorithm is applied to $(\mathcal{G}, \mathcal{F})$ in (3.5.3), we proceed through a sequence of generator matrices $(\mathcal{G}, \mathcal{G}_1, \mathcal{G}_2, \ldots)$ of decreasing number of rows $(2n, 2n-1, 2n-2, \ldots)$. Let g_i denote the top row of the generator matrix $\mathcal{G}_i$ at step i. In a positive step, it needs to be reduced to the form (3.3.4) via an $(I_p \oplus -I_q)$-unitary rotation Θ_i. We propose to perform this transformation as follows:

1. Apply a *unitary* (orthogonal) rotation (e.g., Householder—see App. B) to the first p columns of $\mathcal{G}_i$ so as to reduce the top row of these p columns into proper form,

$$g_i = \begin{bmatrix} x & x & x & x & x & x \end{bmatrix} \overset{unitary\ \Theta_{i,1}}{\longrightarrow} \begin{bmatrix} x & 0 & 0 & x & x & x \end{bmatrix} = g_{i,1},$$

with a nonzero entry in the first column. Let

$$\mathcal{G}_{i,1} = \mathcal{G}_i \begin{bmatrix} \Theta_{i,1} & 0 \\ 0 & I \end{bmatrix} \tag{3.5.4}$$

denote the modified generator matrix. Its last q columns coincide with those of $\mathcal{G}_i$.

2. Apply another *unitary* (orthogonal) rotation (e.g., Householder) to the last q columns of $\mathcal{G}_{i,1}$ so as to reduce the top row of these last q columns into proper form with respect to their last column,

$$g_{i,1} = \begin{bmatrix} x & 0 & 0 & x & x & x \end{bmatrix} \overset{unitary\ \Theta_{i,2}}{\longrightarrow} \begin{bmatrix} x & 0 & 0 & 0 & 0 & x \end{bmatrix} = g_{i,2},$$

with a nonzero entry in the last column. Let

$$\mathcal{G}_{i,2} = \mathcal{G}_{i,1} \begin{bmatrix} I & 0 \\ 0 & \Theta_{i,2} \end{bmatrix} \tag{3.5.5}$$

denote the modified generator matrix. Its first p columns coincide with those of $\mathcal{G}_{i,1}$.

3. Employ an elementary rotation Θ_i acting on the first and last columns (in mixed-downdating form or according to the OD or the H methods of Ch. 2) in order to annihilate the nonzero entry in the last column,

$$g_{i,2} = \begin{bmatrix} x & 0 & 0 & 0 & x \end{bmatrix} \overset{hyperbolic\ \Theta_{i,3}}{\longrightarrow} \begin{bmatrix} x & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

4. The combined effect of the above steps is to reduce g_i to the proper form (3.3.4) and, hence,

$$\bar{\mathcal{G}}_i = \mathcal{G}_i \begin{bmatrix} \Theta_{i,1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \Theta_{i,2} \end{bmatrix} \Theta_{i,3}. \tag{3.5.6}$$

Expression (3.5.6) shows that, in infinite precision, the generator matrices $\mathcal{G}_i$ and $\bar{\mathcal{G}}_i$ must satisfy the fundamental requirement

$$\mathcal{G}_i \mathcal{J} \mathcal{G}_i^T = \bar{\mathcal{G}}_i \mathcal{J} \bar{\mathcal{G}}_i^T. \tag{3.5.7}$$

Obviously, this condition cannot be guaranteed in finite precision. But, as discussed in Ch. 2 and in [CS96], with the above implementation of the transformation (3.5.6) (as a sequence of two orthogonal transformations and a hyperbolic rotation in mixed, OD, or H forms), equality (3.5.7) can be guaranteed to within a "small" error, viz.,

$$\|\hat{\bar{\mathcal{G}}}_i \mathcal{J} \hat{\bar{\mathcal{G}}}_i^T - \mathcal{G}_i \mathcal{J} \mathcal{G}_i^T\|_2 \leq O_n(\varepsilon) \left(\|\hat{\bar{\mathcal{G}}}_i\|_2^2 + \|\mathcal{G}_i\|_2^2 \right). \tag{3.5.8}$$

A similar analysis holds for a negative step, where the rotation Θ_i is again implemented as a sequence of two unitary rotations and one elementary hyperbolic rotation in order to guarantee the transformation (3.3.7). We forgo the details here.

We finally remark that in the algorithm, the incoming generator matrix $\mathcal{G}_i$ will in fact be the computed version, which we denote by $\hat{\mathcal{G}}_i$. This explains why in the error analysis of the next section we replace $\mathcal{G}_i$ by $\hat{\mathcal{G}}_i$ in the error bound (3.5.8).

Note also that we are implicitly assuming that the required hyperbolic rotation $\Theta_{i,3}$ exists. While that can be guaranteed in infinite precision, it is possible that in finite precision we can experience breakdowns.

3.5.2 Avoiding Breakdown

To avoid breakdown we need to guarantee that during the first n steps of the algorithm, the $\mathcal{J}$-unitary rotations Θ_i are well defined. This requires that the leading submatrices of the first n successive Schur complements remain positive definite, a condition that can be guaranteed by requiring T to be sufficiently well-conditioned [CS98], viz., by requiring that

$$\sigma_{\min}^2(T) \; > \; 2\, O_n(\varepsilon) \sum_{j=0}^{n-1} \|\hat{\mathcal{G}}_j\|_2^2 \,. \tag{3.5.9}$$

We refer to a matrix T that satisfies the above requirement as being well conditioned. (The scalar multiple 2 is made explicit for convenience in later discussion.)

Now, after the first n steps of the generalized Schur algorithm applied to $(\mathcal{F}, \mathcal{G})$ in (3.5.3), we let

$$\begin{bmatrix} \hat{R}^T \\ \hat{Q} \end{bmatrix}$$

denote the computed factors that correspond to expression (3.4.6). We further define the matrix S_n that solves the displacement equation

$$S_n - Z_n S_n Z_n^T = \hat{\mathcal{G}}_n \mathcal{J} \hat{\mathcal{G}}_n^T. \tag{3.5.10}$$

Note that S_n is an $n \times n$ matrix, which in infinite precision would have been equal to the Schur complement—I (cf. (3.4.6)). We can now define

$$\widehat{M} = \begin{bmatrix} \hat{R}^T \\ \hat{Q} \end{bmatrix} \begin{bmatrix} \hat{R} & \hat{Q}^T \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & S_n \end{bmatrix}. \tag{3.5.11}$$

We showed in [CS98] that the following error bound holds.

Theorem 3.5.1 (Error Bound). *The first n steps of the generalized Schur algorithm applied to $(\mathcal{F}, \mathcal{G})$ in* (3.5.3)*, for a matrix T satisfying* (3.5.9)*, and with the rotations Θ_i implemented as discussed in Sec.* 3.5.1*, guarantees the following error bound on the matrix $(M - \widehat{M})$ (with $\widehat{M}$ defined in* (3.5.11)*):*

$$\|M - \widehat{M}\|_2 \leq O_n(\varepsilon) \sum_{j=0}^{n-1} \|\widehat{\mathcal{G}}_j\|_2^2. \tag{3.5.12}$$

$\Diamond$

The natural question then is, How big can the norm of the generator matrices be? The following remark is motivated by an observation in [SD97b] that for matrices of the form $T^T T$, with T Toeplitz, there is no appreciable generator growth. Indeed, we showed in [CS98] that a first-order bound for the sum of the norms of the generators in (3.5.12) is given by

$$\sum_{i=0}^{n-1} \|\mathcal{G}_i\|_2^2 \leq 16n(1+n^2)(1+\|T\|_2+\|T^2\|_2) + O(\varepsilon^2). \tag{3.5.13}$$

3.5.3 Error Analysis of the Last Steps

We now study the last n steps. Assume T satisfies the following normalization:

$$\|T\|_2 \leq \frac{1}{5}, \tag{3.5.14}$$

which can always be guaranteed by proper scaling at the begining of the algorithm. We showed in [CS98] that under the well-conditioned assumption (3.5.9), the matrix S_n is guaranteed to be negative definite and well-conditioned. In particular, its condition number is at most 15.

Hence, we can proceed with the last n steps of the generalized Schur algorithm applied to $\widehat{\mathcal{G}}_n$, since $\widehat{\mathcal{G}}_n$ is a generator matrix for S_n:

$$S_n - Z_n S_n Z_n^T = \widehat{\mathcal{G}}_n \mathcal{J} \widehat{\mathcal{G}}_n^T.$$

All steps will now be negative steps. Hence, the discussion in Sec. 3.5.1 on the implementation of the hyperbolic rotations applies. The only difference will be that we make the generator proper with respect to its last column. In other words, the third step of that implementation should be modified as follows:

$$g_{i,2} = \begin{bmatrix} x & 0 & 0 & 0 & x \end{bmatrix} \overset{hyperbolic\ \Theta_{i,3}}{\longrightarrow} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & x \end{bmatrix}. \tag{3.5.15}$$

Let $-\Delta\Delta^T$ be the computed triangular factorization of S_n. It can be shown that

$$\|S_n - (-\Delta\Delta^T)\|_2 \leq O_n(\varepsilon) \sum_{i=n}^{2n-1} \|\widehat{\mathcal{G}}_i\|_2^2, \tag{3.5.16}$$

where the norm of the generators $\{\widehat{\mathcal{G}}_i\}$ appearing in the above error expression can be shown to be bounded [CS98].

3.5.4 Summary

We have shown so far that if we apply $2n$ steps of the generalized Schur algorithm to the matrices $(\mathcal{F}, \mathcal{G})$ in (3.5.3), with proper implementation of the $\mathcal{J}$-unitary rotations (as explained in Sec. 3.5.1), then the error in the computed factorization of M is bounded as follows:

$$\left\| M - \begin{bmatrix} \widehat{R}^T & 0 \\ \widehat{Q} & \Delta \end{bmatrix} \begin{bmatrix} \widehat{R} & \widehat{Q}^T \\ 0 & -\Delta^T \end{bmatrix} \right\|_2 \leq O_n(\varepsilon) \sum_{i=0}^{2n-1} \|\widehat{\mathcal{G}}_i\|_2^2. \tag{3.5.17}$$

We have also established (at least in infinite precision) that the norm of the generators is bounded. Therefore, the computed factorization is (at least asymptotically) backward stable with respect to M.

3.5.5 Solving the Linear System of Equations

We now return to the problem of solving the linear system of equations $Tx = b$, where T is a well-conditioned nonsymmetric shift-structured matrix (e.g., Toeplitz, quasi Toeplitz, product of two Toeplitz matrices).

We showed in [CS98] that $\Delta^{-1}\widehat{Q}$ is numerically orthogonal, viz.,

$$\|(\Delta^{-1}\widehat{Q})(\Delta^{-1}\widehat{Q})^T - I\|_2 \leq O_n(\varepsilon) \sum_{i=0}^{2n-1} \|\widehat{\mathcal{G}}_i\|^2,$$

and that

$$\|T - \widehat{Q}\widehat{R}\|_2 \leq O_n(\varepsilon) \sum_{i=0}^{2n-1} \|\widehat{\mathcal{G}}_i\|_2^2.$$

This shows that we can compute x by solving the nearby linear system

$$\Delta\Delta^{-1}\widehat{Q}\widehat{R}x = b$$

in $O(n^2)$ flops by exploiting the fact that $\Delta^{-1}\widehat{Q}$ is numerically orthogonal and Δ is triangular as follows:

$$\widehat{x} = \widehat{R}^{-1}(\widehat{Q}^T\Delta^{-T})\Delta^{-1}b\,. \tag{3.5.18}$$

The fact that this scheme for computing x is backward stable will be established in the next section (see the remark after expression (3.6.6)).

3.6 ILL-CONDITIONED COEFFICIENT MATRICES

We now consider modifications to the algorithm when the inequality (3.5.9) is not satisfied by T. This essentially means that the condition number of T is larger than the square root of the reciprocal of the machine precision. We will refer to such matrices T as being ill-conditioned.

There are now several potential numerical problems, all of which have to be eliminated. First, the $(1,1)$ block of M can fail to factorize as it is not sufficiently positive definite. Second, even if the first n steps of the Schur algorithm are completed successfully, the Schur complement S_n of the $(2,2)$ block may no longer be negative definite, making the algorithm unstable. Third, the matrix Δ may no longer be well-conditioned, in which case it is not clear how one can solve the linear system $Tx = b$ in a stable manner. We now show how these problems can be resolved.

To resolve the first two problems we add small multiples of the identity matrix to the $(1,1)$ and $(2,2)$ blocks of M, separately:

$$M = \begin{bmatrix} T^T T + \alpha I & T \\ T^T & -\beta I \end{bmatrix}, \tag{3.6.1}$$

where α and β are positive numbers that will be specified later. (We continue to use M for the new matrix in (3.6.1) for convenience of notation.) This leads to an increase in the displacement rank of M. For Toeplitz matrices the rank increases only by one and the new generators are given as follows:

$$M - (Z_n \oplus Z_n) M (Z_n \oplus Z_n)^T = \mathcal{G} \mathcal{J} \mathcal{G}^T, \tag{3.6.2}$$

where $\mathcal{J}$ is 6×6,

$$\mathcal{J} = \mathrm{diag}[1, 1, 1, -1, -1, -1], \tag{3.6.3}$$

and $\mathcal{G}$ is $2n \times 6$,

$$\mathcal{G} = \begin{bmatrix} \sqrt{\alpha} & s_0 & 0 & 0 & 0 & 0 \\ 0 & s_1 & t_{-1} & s_1 & t_{n-1} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & s_{n-1} & t_{-n+1} & s_{n-1} & t_1 & 0 \\ 0 & c_0 & 1 & c_0 & 0 & \sqrt{1+\beta} \\ 0 & c_1 & 0 & c_1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & c_{n-1} & 0 & c_{n-1} & 0 & 0 \end{bmatrix}. \tag{3.6.4}$$

Had we started instead with the embedding (3.4.2) for more general shift-structured matrices, we would then modify the generators as explained later in the remark.

For suitably chosen α and β (see the statement of the algorithm in Sec. 3.7), it can be shown that in this case [CS98]

$$\left\| M - \begin{bmatrix} \widehat{R}^T & 0 \\ \widehat{Q} & \Delta \end{bmatrix} \begin{bmatrix} \widehat{R} & \widehat{Q}^T \\ 0 & -\Delta^T \end{bmatrix} \right\|_2 \leq \alpha + \beta + O_n(\varepsilon) \sum_{i=0}^{2n-1} \|\widehat{\mathcal{G}}_i\|_2^2 ,$$

where the norm of the generators is again bounded. Hence, we also obtain a backward-stable factorization of M.

Since Δ is no longer provably well-conditioned, we cannot argue that $\Delta^{-1}\widehat{Q}$ is numerically orthogonal. For this reason, we now discuss how to solve the linear system of equations $Tx = b$ in the ill-conditioned case.

3.6.1 Solving the Linear System of Equations

Note that if x solves $Tx = b$, then it also satisfies

$$\begin{bmatrix} T^T T & T^T \\ T & 0 \end{bmatrix} \begin{bmatrix} x \\ -b \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}.$$

Using the above backward-stable factorization for M we can solve the above linear system of equations to get

$$\left(\begin{bmatrix} T^T T & T^T \\ T & 0 \end{bmatrix} + \tilde{M} \right) \begin{bmatrix} \widehat{y} \\ \widehat{z} \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}, \tag{3.6.5}$$

where the error matrix $\tilde{M}$ satisfies

$$\|\tilde{M}\|_2 \leq \alpha + \beta + O_n(\varepsilon) \sum_{i=0}^{2n-1} \|\widehat{\mathcal{G}}_i\|_2^2 + O_n(\varepsilon) \left\| \begin{bmatrix} \widehat{R}^T & 0 \\ \widehat{Q} & \Delta \end{bmatrix} \right\|_2^2.$$

Note that $\widehat{y}$ is computed by the expression

$$R^{-1}\widehat{Q}^T\Delta^{-T}\Delta^{-1}b, \tag{3.6.6}$$

which is identical to the formula (3.5.18) we obtained earlier by assuming $\Delta^{-1}\widehat{Q}$ is numerically orthogonal! Therefore, the subsequent error bound holds equally well for the well-conditioned case. It can be shown that the computed solution $\widehat{y}$ satisfies

$$(T + \tilde{T})\widehat{y} = b,$$

where the norm of the error matrix is bounded by

$$\|\tilde{T}\|_2 \leq 2(\alpha + \beta) + O_n(\varepsilon)[1 + \|T\|_2] + O(\varepsilon^2) \leq O_n(\varepsilon)\, \|T\|_2 + O(\varepsilon^2). \tag{3.6.7}$$

3.6.2 Conditions on the Coefficient Matrix

For ease of reference, we list here the conditions imposed on the coefficient matrix T in order to guarantee a fast backward-stable solver of $Tx = b$:

1. $\|T\|_2$ is suitably normalized to guarantee $\|T\|_2 \approx 1$ (cf. (3.5.14)).
2. The condition number of T should essentially be less than the reciprocal of the machine precision.

Remark. Had we started instead with the embedding (3.4.2), we first perform n steps of the generalized Schur algorithm to get a generator matrix $\widehat{\mathcal{G}}_n$ for the computed version of the $2n \times 2n$ embedding (3.4.5). We then add two columns to $\widehat{\mathcal{G}}_n$ as follows:

$$\begin{bmatrix} \sqrt{\alpha} & & 0 \\ 0 & & 0 \\ 0 & & \sqrt{\beta} \\ \vdots & \widehat{\mathcal{G}}_{n+1} & \vdots \\ 0 & & 0 \\ 0 & & 0 \end{bmatrix},$$

where the entry $\sqrt{\beta}$ occurs in the nth row of the last column. The new first column has a positive signature and the new last column has a negative signature.

3.7 SUMMARY OF THE ALGORITHM

For convenience we summarize the algorithm here for the case of nonsymmetric Toeplitz systems. We hasten to add, however, that the algorithm also applies to more general shift-structured matrices T (other than Toeplitz, such as quasi Toeplitz or with higher displacement ranks, as demonstrated by the analysis in the earlier sections). The only difference will be in the initial generator matrix $\mathcal{G}$ and signature matrix $\mathcal{J}$ for M in (3.6.1) and (3.6.2). The algorithm will also be essentially the same, apart from an additional n Schur steps, if we instead employ the embedding (3.4.2).

- Input: A nonsymmetric Toeplitz matrix $T \in \mathbb{R}^{n\times n}$ and column vector $b \in \mathbb{R}^{n\times 1}$. The entries of the first column of T are denoted by $[t_0, t_1, \ldots, t_{n-1}]^T$, while the entries of the first row of T are denoted by $[t_0, t_{-1}, \ldots, t_{-n+1}]$.

- Output: A backward-stable solution of $Tx = b$.

- Algorithm:

- Normalize T and b. Since the Frobenius norm of T is less than

$$\gamma = \sqrt{n \sum_{i=-n+1}^{n-1} t_i^2},$$

we can normalize T by setting t_i to be $t_i/(5\gamma)$ for all i. Similarly, divide the entries of b by 5γ. In what follows, T and b will refer to these normalized quantities.

- Define the vectors

$$\begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} = \frac{Te_1}{\|Te_1\|_2}, \qquad \begin{bmatrix} s_0 \\ \vdots \\ s_{n-1} \end{bmatrix} = T^T \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix}.$$

- Construct the 6×6 signature matrix $\mathcal{J} = \mathrm{diag}[1, 1, 1, -1, -1, -1]$, and the $2n \times 6$ generator matrix $\mathcal{G}$,

$$\mathcal{G} = \begin{bmatrix} \sqrt{\alpha} & s_0 & 0 & 0 & 0 & 0 \\ 0 & s_1 & t_{-1} & s_1 & t_{n-1} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & s_{n-1} & t_{-n+1} & s_{n-1} & t_1 & 0 \\ 0 & c_0 & 1 & c_0 & 0 & \sqrt{1+\beta} \\ 0 & c_1 & 0 & c_1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & c_{n-1} & 0 & c_{n-1} & 0 & 0 \end{bmatrix},$$

where the small positive numbers α and β are chosen as follows (by experimental tuning):

$$\alpha = n^{1/2}\varepsilon \, \|\mathcal{G}\|_2^2, \qquad \beta = 4(2n)^{1/4}\varepsilon.$$

(If T is well-conditioned (say, $\kappa(T) < 1/\sqrt{\varepsilon}$), then we can set $\beta = 0 = \alpha$ and delete the first columns of $\mathcal{G}$ and $\mathcal{J}$, which then become $2n \times 5$ and 5×5, respectively.)

- Apply n steps of the generalized Schur algorithm starting with $\mathcal{G}_0 = \mathcal{G}$ and $\mathcal{F} = (Z_n \oplus Z_n)$ and ending with $\mathcal{G}_n$ and $\mathcal{F} = Z_n$. These are positive steps according to the description of Alg. 3.3.1 (step 2), where the successive generators are reduced to proper form relative to their first column. Note that this must be performed with care for numerical stability as explained in Sec. 3.5.1.

- Apply n more steps of the generalized Schur algorithm starting with $\mathcal{G}_n$. These are negative steps according to the description of Alg. 3.3.1 (step 3), where the successive generators are reduced to proper form relative to their last column. This also has to be performed with care as explained prior to (3.5.15).

Table 3.1. Complexity analysis of the fast algorithm.

During each iteration of the algorithm	Count in flops
Compute two Householder transformations	$3r$
Apply the Householder transformations	$4 \cdot i \cdot r$
Compute the hyperbolic transformation	7
Apply the hyperbolic transformation using OD	$6 \cdot i$
Shift columns	i
Total for $i = 2n - 1$ down to 0	$(14 + 8r)n^2 + 10nr + 21n$
Cost of 3 back-substitution steps	$3n^2$
Cost of matrix-vector multiplication	$2n^2$
Start-up costs	$n(24 \log n + r + 52)$
Total cost of the algorithm	$(19 + 8r)n^2$ $+ n(24 \log n + 11r + 73)$

- Each of the above $2n$ steps provides a column of the triangular factorization of the matrix M in (3.6.1), as described in Alg. 3.3.1 (steps 2 and 3). The triangular factor of M is then partitioned to yield the matrices $\{\widehat{R}, \widehat{Q}, \Delta\}$,

$$\begin{bmatrix} \widehat{R}^T & 0 \\ \widehat{Q} & \Delta \end{bmatrix},$$

  where $\widehat{R}$ is upper triangular and Δ is lower triangular.

- The solution $\widehat{x}$ is obtained by evaluating the quantity

$$R^{-1}\widehat{Q}^T\Delta^{-T}\Delta^{-1}b$$

  via a sequence of back substitutions and matrix-vector multiplications. The computed solution is backward stable. It satisfies $(T + \tilde{T})\widehat{x} = b$, where the norm of the error matrix is bounded by (3.6.7).

Operation Count

The major computational cost is due to the application of the successive steps of the generalized Schur algorithm. The overhead operations that are required for the normalization of T and for the determination of the generator matrix $\mathcal{G}$ amount at most to $O(n \log n)$ flops. Table 3.1 shows the number of flops needed at each step of the algorithm. (i denotes the iteration number and runs from $i = 2n$ down to $i = 1$.) The operation count given in the table assumes that, for each iteration, two Householder transformations are used to implement the reduction to the proper form of Sec. 3.5.1, combined with an elementary hyperbolic rotation in OD form.

The specific costs of the algorithm for the special case of Toeplitz matrices are the following:

1. For a well-conditioned Toeplitz matrix, the cost is $O(59n^2 + n(24 \log n + 128))$ operations.

2. For an ill-conditioned Toeplitz matrix, the cost is $O(67n^2 + n(24 \log n + 139))$ operations.

Acknowledgments

The authors wish to thank Professor Thomas Kailath for comments and feedback on an earlier draft of this chapter. They also gratefully acknowledge the support of the National Science Foundation; the work of A. H. Sayed was partially supported by awards MIP-9796147 and CCR-9732376, and the work of S. Chandrasekaran was partially supported by award CCR-9734290.