LAPACK–Style Algorithms and Software for Solving the Generalized Sylvester Equation and Estimating the Separation between Regular Matrix Pairs

Bo Kågström and Peter Poromaa*

December, 1993
Revised April, 1994

Abstract

Level 3 algorithms for solving the generalized Sylvester equation $(AR - LB, DR - LE) = (C, F)$ and a transposed analogue $(A^T U + D^T V, -UB^T - VE^T) = (C, F)$ are presented. These blocked algorithms permit reuse of data in complex memory hierarchies of current advanced computer architectures. The separation of two regular matrix pairs (A, D) and (B, E), $\mathrm{Dif}[(A, D), (B, E)]$, is defined in terms of the generalized Sylvester operator $(AR - LB, DR - LE)$. Robust, efficient and reliable Dif–estimators are presented. The basic problem is to find a lower bound on Dif^{-1}, which can be done by solving generalized Sylvester equations in triangular form. Frobenius norm–based and one–norm–based Dif–estimators are described and evaluated. These estimates lead to computable error bounds for the generalized Sylvester equation. The one–norm–based estimator makes the condition estimation uniform with LAPACK. Fortran 77 software that implements our algorithms for solving generalized Sylvester equations, and for computing error bounds and Dif–estimators are presented. Computational experiments that illustrate the accuracy, efficiency and reliability of our software are also described.

Key words. Generalized Sylvester equation, block algorithms, Dif–estimators, error bounds, LAPACK–style software.

AMS(MOS) subject classifications. Primary 65F05, 65G05.

*Institute of Information Processing, University of Umeå, S-901 87 Umeå, Sweden, Electronic addresses: `bokg@cs.umu.se` and `peterp@cs.umu.se`

1 Introduction

The generalized Sylvester equation

$$\begin{aligned} AR - LB &= C, \\ DR - LE &= F, \end{aligned} \tag{1.1}$$

where L and R are unknown $m \times n$ matrices, $(A, D), (B, E)$ and (C, F) are given pairs of $m \times m, n \times n,$ and $m \times n$ matrices, respectively, with real (or complex) entries has several applications relating to the problem of computing stable eigendecompositions of matrix pencils [23, 6]. For example, it can be formulated in terms of a block-diagonalizing equivalence transformation $P^{-1}(S - \lambda T)Q$, where

$$S - \lambda T \equiv \begin{bmatrix} A & -C \\ 0 & B \end{bmatrix} - \lambda \begin{bmatrix} D & -F \\ 0 & E \end{bmatrix}, \tag{1.2}$$

and

$$P^{-1} = \begin{bmatrix} I_m & -L \\ 0 & I_n \end{bmatrix}, \quad Q = \begin{bmatrix} I_m & R \\ 0 & I_n \end{bmatrix}. \tag{1.3}$$

We want to find (L, R) such that

$$P^{-1}(S - \lambda T)Q = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix} - \lambda \begin{bmatrix} D & 0 \\ 0 & E \end{bmatrix}. \tag{1.4}$$

The first m columns of P^{-1} and Q, respectively, span a pair of eigenspaces (deflating subspaces) associated with $\lambda(A, D)$ [24]. By solving for (L, R) in (1.1) we also get a pair of complementary eigenspaces (deflating subspaces associated with $\lambda(B, E)$) from the last n columns of P^{-1} and Q, respectively. One can show that (1.1) has a unique solution if and only if the *regular* pencils $A - \lambda D$ and $B - \lambda E$ have disjoint spectra [23]. If these pencils have common spectra or are *singular* (i.e., $\det(A - \lambda D) \equiv 0$ or $\det(B - \lambda E) \equiv 0$ for each λ), the generalized Sylvester equation will not in general be consistent. An important quantity that measures the sensitivity of these eigenspaces is the *separation of the matrix pairs* (A, D) and (B, E) [23], [24],

$$\mathrm{Dif}[(A, D), (B, E)] = \inf_{\|(L,R)\|_F = 1} \|(AR - LB, DR - LE)\|_F. \tag{1.5}$$

The relationship with the generalized Sylvester equation is that $\mathrm{Dif}[(A, D), (B, E)] > 0$ (Dif for short) if and only if (1.1) has a unique solution.

If we choose D and E to be the identity matrices and F as the zero matrix then (1.1) reduces to the (standard) Sylvester equation $AR - RB = C$, and the discussion above will then concern invariant subspaces of S, with $P = Q$ [17, 2].

Using Kronecker products the matrix equation (1.1) can be written as a $2mn \times 2mn$ linear system of equations [6]

$$Zx = b, \tag{1.6}$$

where

$$Z = \begin{bmatrix} I_n \otimes A & -B^T \otimes I_m \\ I_n \otimes D & -E^T \otimes I_m \end{bmatrix}, \quad x = \begin{bmatrix} \mathrm{col}(R) \\ \mathrm{col}(L) \end{bmatrix}, \quad b = \begin{bmatrix} \mathrm{col}(C) \\ \mathrm{col}(F) \end{bmatrix}. \tag{1.7}$$

The column vector col(X) denotes an ordered stack of the columns of a matrix X from left to right starting with the first column. Already for moderate m and n, the $2mn \times 2mn$ Z in (1.7) is a quite large matrix. Accordingly, this equivalent formulation is mainly of interest for theoretical purposes. For example, it can be shown that

$$\mathrm{Dif}^{-1}[(A, D), (B, E)] = \|Z^{-1}\|_2 = \sigma_{\min}(Z)^{-1}, \tag{1.8}$$

where $\sigma_{\min}(Z)$ is the smallest singular value of Z [6].

To compute $\sigma_{\min}(Z)$ is an $O(m^3n^3)$ operation, so there is a need for efficient and reliable Dif^{-1}–estimators [20]. In this context, besides solving $Zx = b$ we also need to be able to solve a transposed system $Z^T y = b$, where

$$Z^T = \begin{bmatrix} I_n \otimes A^T & I_n \otimes D^T \\ -B \otimes I_m & -E \otimes I_m \end{bmatrix}, \quad y = \begin{bmatrix} \mathrm{col}(U) \\ \mathrm{col}(V) \end{bmatrix}. \tag{1.9}$$

To solve for y in $Z^T y = b$ is equivalent to solve for (U, V) in the matrix equation

$$\begin{aligned} A^T U + D^T V &= C, \\ -UB^T - VE^T &= F. \end{aligned} \tag{1.10}$$

Notice that by transposing the left hand sides of (1.1) we obtain a transposed generalized Sylvester equation different from (1.10).

Other applications of (1.1) include a new direct method for reordering eigenvalues in the generalized real Schur form of a regular matrix pair [16], and an algorithm for computing an additive decomposition of a generalized transfer matrix [19]. The latter comprises both a reordering of eigenvalues and a block–diagonalization as described above. An alternative form of a generalized Sylvester equation with applications in control theory is

$$AXB^T + CXD^T = E, \tag{1.11}$$

where A and C are $m \times m$, B and D are $n \times n$, and E and the desired solution X are $m \times n$ [9]. The matrix equation (1.11) has a unique solution if and only if (A, C) and $(-D, B)$ are regular matrix pairs with disjoint spectra [4]. By introducing $R = XB^T$ and $L = CX$, (1.11) can be recast in the form (1.1):

$$\begin{aligned} AR + LD^T &= E, \\ CR - LB^T &= 0. \end{aligned}$$

The solvability condition for (1.11) implies that at least one of B and C must be nonsingular, so X can, at least in theory, be resolved from L or R. However, if both B and C are ill-conditioned with respect to inversion (or one of them is singular and the other ill-conditioned) it is recommended to solve (1.11) directly [9, 8].

The rest of the paper is outlined as follows. In Section 1.1 we collect our notation. In Section 2 we present block algorithms for solving $Zx = b$ and $Z^T y = b$, respectively. Section 3 describes Dif^{-1}–estimators based on the Frobenius norm and the one–norm (or infinity–norm), respectively. In Section 4 we discuss LAPACK–style error bounds for the generalized Sylvester equation. Section 5 presents our Fortran 77 software that implement algorithms for solving generalized Sylvester equations, and for computing error bounds and Dif–estimators. Some computational experiments that illustrate the accuracy, efficiency and reliability of our software are presented in Section 6. Finally, some conclusions are summarized in Section 7.

1.1 Notation

The following notation is used in the paper. I_n denotes an identity matrix of size n-by-n. $\lambda(A,B)$ denotes the spectrum of a regular matrix pair (A,B) or pencil $A-\lambda B$. $\|A\|_2$ denotes the spectral norm (2-norm) of a matrix A induced by the Euclidean vector norm. $\|A\|_F$ denotes the Frobenius (or Euclidean) matrix norm. $\|A\|_M = \max_{i,j}|a_{ij}|$, i.e. the maximum of the absolute values of the matrix entires. $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ denote the largest and smallest singular values of A, respectively. For a square matrix A we have that $\|A\|_2 = \sigma_{\max}(A)$ and $\|A^{-1}\|_2 = \sigma_{\min}(A)^{-1}$. $A \otimes B$ denotes the Kronecker product of two matrices A and B whose (i,j)-th block element is $a_{ij}B$. A^T denotes the transpose of A. A^H denotes the conjugate transpose of A. $\overline{A}$ denotes the conjugate of A. $|A|$ and $|x|$ denote the matrix and the vector whose elements are $|a_{ij}|$ and $|x_i|$, respectively. Inequalities such as $|A| \le |B|, |x| \le |y|$ are interpreted componentwise. $D = \mathrm{diag}(x)$ denotes a diagonal matrix with $d_{ii} = x_i$.

2 LAPACK–Style Algorithms for Solving Generalized Sylvester Equations

Algorithms used in LAPACK [1] are designed to be efficient on a wide range of modern high performance computers. To achieve high performance on these advanced architectures several well-known elementwise algorithms have been restructured in terms of Level 2 (matrix–vector) and Level 3 (matrix–matrix) operations. These blocked algorithms permit reuse of data in complex memory hierarchies. In the following, we derive similar block algorithms for solving the generalized Sylvester equation (1.1) and the transposed variant (1.10).

We start by recalling the algorithms in [20] for solving (1.1). These are generalizations of the Schur method [3] and the Hessenberg–Schur method [10] for solving $AR - RB = C$. Both methods are based on orthogonal equivalence transformations (unitary transformations if the matrix entries are complex) and involve the following four steps:

1. Transform (A,D) and (B,E) to simpler form:
$$\begin{aligned}(A_1, D_1) &:= (P^TAQ, P^TDQ),\\ (B_1, E_1) &:= (U^TBV, U^TEV).\end{aligned}$$

2. Modify the right hand sides (C,F):
$$C_1 := P^TCV, \quad F_1 := P^TFV.$$

3. Solve the transformed system for (L_1, R_1):
$$\begin{aligned} A_1R_1 - L_1B_1 &= C_1,\\ D_1R_1 - L_1E_1 &= F_1.\end{aligned} \tag{2.1}$$

4. Transform the solution back to the original system:
$$L := PL_1U^T, \quad R := QR_1V^T.$$

We have chosen to recast the generalized Schur method [20] in a blocked analogue. Using _GEGS, the LAPACK implementation of the QZ–algorithm [21] in Step 1, (A, D) and (B, E) are transformed to generalized real Schur form with A_1 and B_1 (upper) quasi triangular and D_1 and E_1 (upper) triangular. A quasi triangular matrix is block triangular with 1×1 and 2×2 diagonal blocks. The 2×2 blocks correspond to pairs of complex conjugate eigenvalues of the associated matrix pencil and the ratios of the 1×1 diagonal blocks (a_{ii}/b_{ii}) are the real eigenvalues. (In the generalized complex Schur form A_1 and B_1 will be (upper) triangular too, which simplifies the discussion below.)

Step 2 and Step 4 are performed with calls to _GEMM, the general matrix–multiply–and–add operation of the Level 3 BLAS [7]. It only remains to deal with Step 3, i.e. to solve a generalized Sylvester equation in quasi triangular form.

Suppose the transformed matrix equation (2.1) is partitioned according to the diagonal block structure of A_1 and B_1. Let A_{ii} of size $a \times a$, $a = 1, 2$ and B_{jj} of size $b \times b$, $b = 1, 2$ denote the diagonal blocks of A_1 and B_1, respectively, and let p, q be the number of diagonal blocks of A_1 and B_1. Then (2.1) is solved by the GS–algorithm that compactly can be written as

$$\begin{aligned} A_{ii}R_{ij} - L_{ij}B_{jj} &= C_{ij} - \left(\sum_{k=i+1}^{p} A_{ik}R_{kj} - \sum_{k=1}^{j-1} L_{ik}B_{kj}\right) \equiv G_{ij}, \\ D_{ii}R_{ij} - L_{ij}E_{jj} &= F_{ij} - \left(\sum_{k=i+1}^{p} D_{ik}R_{kj} - \sum_{k=1}^{j-1} L_{ik}E_{kj}\right) \equiv H_{ij}, \end{aligned} \tag{2.2}$$

for $j = 1, 2, \ldots, q$ and $i = p, p-1, \ldots, 1$ [20]. In total we solve $p \cdot q$ small subsystems (2.2). Each of them can be written as a linear system

$$\begin{bmatrix} I_b \otimes A_{ii} & -B_{jj}^T \otimes I_a \\ I_b \otimes D_{ii} & -E_{jj}^T \otimes I_a \end{bmatrix} \begin{bmatrix} \mathrm{col}(R_{ij}) \\ \mathrm{col}(L_{ij}) \end{bmatrix} = \begin{bmatrix} \mathrm{col}(G_{ij}) \\ \mathrm{col}(H_{ij}) \end{bmatrix}, \tag{2.3}$$

of size 2, 4 or 8. Note that even if A_{ii} and B_{jj} are upper triangular, the subsystems (2.3) cannot in general be transformed to upper triangular form only by permutations. For example, in the complex case all subsystems (2.3) will be 2-by-2 and dense. This is in contrast to the Schur method for $AR - RB = C$. The solution of the subsystems (2.3) will be discussed later.

In the following two subsections we discuss block algorithms to solve (quasi) triangular generalized Sylvester equations. Without loss of generality, we assume that (A, D) and (B, E) are in generalized Schur form.

2.1 A Block Algorithm for Solving $Zx = b$

Indeed, the GS–algorithm described above is a block method with block sizes one and two (only one in complex arithmetic). However, these block sizes are in general too small in order to efficiently utilize a complex memory hierarchy. Appropriate block sizes for this purpose (i.e. the granularities of the data items we access) are dictated from machine–dependent parameters such as different cache sizes (e.g., on–chip and local or global cache memories), vector register length, and memory bandwidth. Let M and N be the block sizes of (A, D) and (B, E), respectively, chosen with respect to certain machine–parameters of a target architecture. To simplify notation we assume that the problem sizes m and n are multiples of M and N, respectively. Moreover, let A_{ij} of size $M \times M$ and B_{ij} of size $N \times N$

denote blocks (subarrays) of A and B, respectively, and $nbA = n/N, nbB = m/M$ are the number of diagonal blocks of A and B, respectively.

To minimize data traffic and obtain as much data reuse as possible we also have to take the access patterns of the blocks (subarrays) into account. We have chosen a block–column Level 3 variant, which solves for blocks L_{ij} and R_{ij} of (L, R) starting at the south–west corner working upwards (north) and then rightwards (east) block–column by block–column. The Level 3 method is outlined in Algorithm (I).

$$
\begin{array}{l}
\textbf{for } j = 1 : nbB \\
\quad \textbf{for } i = nbA : -1 : 1 \\
\quad\quad \{ \text{ Solve the } (i,j)\text{–subsystem } \} \\
\quad\quad A_{ii}R_{ij} - L_{ij}B_{jj} = C_{ij} \\
\quad\quad D_{ii}R_{ij} - L_{ij}E_{jj} = F_{ij} \\
\quad\quad \textbf{for } k = 1 : i-1 \\
\quad\quad\quad \{ \text{ Update block column } j \text{ of } C \text{ and } F \ \} \\
\quad\quad\quad C_{kj} = C_{kj} - A_{ik}R_{ij} \\
\quad\quad\quad F_{kj} = F_{kj} - D_{ik}R_{ij} \\
\quad\quad \textbf{for } k = j+1 : nbB \\
\quad\quad\quad \{ \text{ Update block row } i \text{ of } C \text{ and } F \ \} \\
\quad\quad\quad C_{ik} = C_{ik} + L_{ij}B_{jk} \\
\quad\quad\quad F_{ik} = F_{ik} + L_{ij}E_{jk}
\end{array}
\tag{I}
$$

Each (i,j)–subsystem of size $2MN \times 2MN$ can be solved using Algorithm (I) with M and N equal to 1 or 2. The values of M and N are defined by the eigenvalues (1 for a real eigenvalue and 2 for a complex conjugate pair). The subsystems of this Level 2 variant are of type (2.3).

2.2 A Block Algorithm for Solving $Z^T y = b$

With the same blocking as used for solving $Zx = b$ it is possible to formulate a blocked variant to solve $Z^T y = b$, with similar arithmetic complexity as Algorithm (I). We solve for blocks U_{ij} and V_{ij} of (U, V) in a block–rowwise fashion, starting at the north–east corner working west and then downwards (south) block–row by block–row. The Level 3 method is outlined in Algorithm (II).

Also here each (i,j)–subsystem of size $2MN \times 2MN$ can be solved using Algorithm (II) with M and N equal to 1 or 2. A subsystem of the Level 2 algorithm looks like

$$
\begin{bmatrix} I_b \otimes A_{ii}^T & I_b \otimes D_{ii}^T \\ -B_{jj} \otimes I_a & -E_{jj} \otimes I_a \end{bmatrix} \begin{bmatrix} \mathrm{col}(U_{ij}) \\ \mathrm{col}(V_{ij}) \end{bmatrix} = \begin{bmatrix} \mathrm{col}(G_{ij}) \\ \mathrm{col}(H_{ij}) \end{bmatrix}, \tag{2.4}
$$

where $a, b = 1, 2$ and the right hand sides G_{ij} and H_{ij} are defined similarly as in (2.3). We use the same method to solve subsytems of type (2.3) and (2.4).

Notice, in the complex case, we want to solve for y in $Z^H y = b$, which corresponds to replace all transposed matrices (or subarrays) in (1.10), (2.4) and Algorithm (II) with the conjugate transposed analogues.

$$
\begin{array}{l}
\textbf{for } i = 1 : nbA \\
\quad \textbf{for } j = nbB : -1 : 1 \\
\quad\quad \{ \text{ Solve the } (i,j)\text{–subsystem } \} \\
\quad\quad \;\; A_{ii}^T U_{ij} + D_{ii}^T V_{ij} = C_{ij} \\
\quad\quad -U_{ij} B_{jj}^T - V_{ij} E_{jj}^T = F_{ij} \\
\quad\quad \textbf{for } k = 1 : j-1 \\
\quad\quad\quad \{ \text{ Update block row } i \text{ of } F \} \\
\quad\quad\quad F_{ik} = F_{ik} + U_{ij} B_{kj}^T + V_{ij} E_{kj}^T \\
\quad\quad \textbf{for } k = i+1 : nbA \\
\quad\quad\quad \{ \text{ Update block column } j \text{ of } C \} \\
\quad\quad\quad C_{kj} = C_{kj} - A_{ik}^T U_{ij} - D_{ik}^T V_{ij}
\end{array}
\tag{II}
$$

3 Estimating the Separation between Two Matrix Pairs

The basic problem is to find a lower bound on $\text{Dif}^{-1}[(A,D),(B,E)] \equiv \|Z^{-1}\|_2$, where Z is the matrix representation (1.7) of the generalized Sylvester operator. It is possible to compute lower bounds on Dif^{-1} by solving generalized Sylvester equations in triangular form. Frobenius norm–based estimators are presented in [20]. Here we describe modifications of two of these estimators and apply a one–norm–based estimator to estimate Dif^{-1}. All condition estimation in LAPACK for linear systems and the standard eigenvalue problem is based on a one–norm–based estimator [13], which earlier has been applied to estimate the separation between two matrices [17, 2]. Our contribution makes it possible to conform condition estimation and error bounds for the generalized eigenvalue problem with LAPACK [18].

By knowing a lower bound `DIFINV` on $\|Z^{-1}\|_2$ we also have an upper bound `DIF` = 1/`DIFINV` on the separation between two regular matrix pairs (1.5). Since we use our block algorithms to solve the generalized Sylvester equations involved in computing `DIFINV`, our estimators will mainly execute Level 3 operations.

3.1 Frobenius Norm-Based Estimators

From the $Zx = b$ representation (1.6) of the generalized Sylvester equation we get a lower bound on Dif^{-1}:

$$\|(L,R)\|_F / \|(C,F)\|_F = \|x\|_2 / \|b\|_2 \leq \|Z^{-1}\|_2. \tag{3.1}$$

To get an improved estimate we want to choose right hand sides (C^*, F^*) such that the associated solution (L^*, R^*) has as large norm as possible. Then the quantity

$$\phi_F \equiv \|(L^*, R^*)\|_F / \|(C^*, F^*)\|_F, \tag{3.2}$$

is our lower bound on $\|Z^{-1}\|_2$. The work to compute ϕ_F is comparable to solve a generalized Sylvester equation, which costs $O(m^3 + m^2 n + mn^2 + n^3)$ flops (only $2m^2 n + 2mn^2$ if the matrix pairs are in generalized Schur form) [20]. This is a very modest cost compared to compute the exact value of $\sigma_{\min}(Z)$, which requires $O(m^3 n^3)$ flops.

Conceptually, we use the GS–algorithm to compute ϕ_F, which means that we get a contribution from each subsystem (2.3) to the final (L^*, R^*). To simplify notation, we write a subsystem (2.3) as

$$Z_{ij}u = v(\equiv h - f), \tag{3.3}$$

where f denotes contributions from other subsystems and h is chosen to maximize $\|u\|$ while keeping $\|v\|$ small. We have made modifications to the estimators $\phi_F(\texttt{BSOLVE})$ and $\phi_F(\texttt{BSOLVD})$ in [20]. First, we replace partial pivoting with complete pivoting in the LU factorization of Z_{ij}, i.e. $VLUW = Z_{ij}$ (V, W permutation matrices). Secondly, we make the look ahead strategy in $\phi_F(\texttt{BSOLVE})$ more robust and the approximate nullvector of Z_{ij} used in $\phi_F(\texttt{BSOLVD})$ is now obtained by using the one–norm–based condition estimator of LAPACK. In summary, we have

- `BSOLVE`: Solve for w in $VLw = h - f$ using a look ahead strategy similar to the Linpack estimator [5], except that our right hand sides are loaded with contributions from earlier computations (f in (3.3)). (The entries of h are chosen to $+1$ or -1 and the sign of h_i is determined by an algorithm dependent local look ahead strategy.) Then, we solve for x_+ and x_- in $UWx_+ = w_+$ and $UWx_- = w_-$, where w_+ and w_- are the solutions resulting from choosing h_n as $+1$ and -1 in $VLw = h = y - f$, respectively. Finally, x is chosen as the solution (x_+ or x_-) with the largest 2–norm. For details see [20, 22].

- `BSOLVD`: Use `_GECON` [1] to compute an approximate null–vector e of Z_{ij}^T and normalize: $h = e/\|e\|_2$. Finally, solve for x in $Z_{ij}x = \pm h - f$ with the sign chosen to give the greater value of $\|x\|_2$.

The local choice of right hand sides gives the global (C_E^*, F_E^*) with entries ± 1 for $\phi_F(\texttt{BSOLVE})$, and for $\phi_F(\texttt{BSOLVD})$ all $p \cdot q$ blocks in the corresponding right hand sides (C_D^*, F_D^*) (where p and q are the number of diagonal blocks in the generalized Schur forms of (A, D) and (B, E), respectively) will have Frobenius norm 1, yielding

$$\phi_F(\texttt{BSOLVE}) = \|(L_E^*, R_E^*)\|_F / \sqrt{2mn}, \tag{3.4}$$

and

$$\phi_F(\texttt{BSOLVD}) = \|(L_D^*, R_D^*)\|_F / \sqrt{pq}. \tag{3.5}$$

3.2 An One–Norm–Based Estimator

From the relationship

$$\frac{1}{\sqrt{2mn}}\|Z^{-1}\|_1 \le \|Z^{-1}\|_2 \le \sqrt{2mn}\|Z^{-1}\|_1, \tag{3.6}$$

we know that $\|Z^{-1}\|_1$ can never differ more than a factor $\sqrt{2mn}$ from $\|Z^{-1}\|_2$. So it makes sense to compute an one–norm–based estimator of Dif^{-1}.

The LAPACK routine `_LACON` implements a method for estimating the one–norm of a square matrix, using reverse communication for evaluating matrix–vector products [12, 13]. By applying this method to $\|Z^{-1}\|_1$ we have to provide the solution vectors x and y of

$Zx = z$ (1.6) and $Z^T y = z$ (1.9), where z is determined by _LACON. In each step we only solve one of these generalized Sylvester equations. For this purpose we use the block algorithms described in Section 2, giving us a Level 3 method for computing a lower bound of $\|Z^{-1}\|_1$. The cost for computing this bound is roughly equal to the number of steps in the reverse communication times the cost for one generalized Sylvester solve. Notice, $\|Z^{-1}\|_\infty$ also satisfy (3.6), i.e. can never differ more than a factor $\sqrt{2mn}$ from $\|Z^{-1}\|_2$. Moreover, since $\|B\|_\infty = \|B^T\|_1$ the same method can be used to compute an infinity–norm–based estimate of Dif^{-1}.

4 Error Bounds for the Generalized Sylvester Equation

Recently, a perturbation analysis of the generalized Sylvester equation was presented [15], that generalizes and extends results for the standard Sylvester equation [14]. By taking full account to the structure of the generalized Sylvester equation, expressions for the backward error of an approximate solution $(\hat{L}, \hat{R})$ and condition numbers that measure the sensitivity of a solution to perturbations in the data are derived. In the following we review some of the results in [15].

One important result shows that small values of the residuals

$$\begin{aligned} R_1 &\equiv C - (A\hat{R} - \hat{L}B), \\ R_2 &\equiv F - (D\hat{R} - \hat{L}E), \end{aligned} \tag{4.1}$$

do not necessarily yield a small backward error, unlike for a standard linear system. Let $(\hat{L}, \hat{R})$ denote an approximate solution of the generalized Sylvester equation (1.1). The normwise backward error of $(\hat{L}, \hat{R})$ is defined by

$$\begin{aligned} \eta(\hat{L}, \hat{R}) \equiv \min\{\epsilon : (A + \Delta A)\hat{R} - \hat{L}(B + \Delta B) = C + \Delta C, \\ (D + \Delta D)\hat{R} - \hat{L}(E + \Delta E) = F + \Delta F, \\ \|(\Delta A, \Delta D)\|_F \le \epsilon\alpha, \|(\Delta B, \Delta E)\|_F \le \epsilon\beta, \|(\Delta C, \Delta F)\|_F \le \epsilon\gamma\}. \end{aligned} \tag{4.2}$$

We see that $\eta(\hat{L}, \hat{R})$ is a measure of the distance to the closest perturbed generalized Sylvester equation that has $(\hat{L}, \hat{R})$ as the exact solution. By choosing $\alpha = \|(A, D)\|_F, \beta = \|(B, E)\|_F, \gamma = \|(C, F)\|_F$, $\eta(\hat{L}, \hat{R})$ corresponds to the normwise relative backward error with respect to the Frobenius norm.

Moreover, we can bound $\eta(\hat{L}, \hat{R})$ from below as

$$\frac{\|(R_1, R_2)\|_F}{(\alpha + \beta)\|(\hat{L}, \hat{R})\|_F + \gamma} \le \frac{\|(R_1, R_2)\|_F}{\alpha\|\hat{R}\|_F + \beta\|\hat{L}\|_F + \gamma} \le \eta(\hat{L}, \hat{R}), \tag{4.3}$$

and from above as

$$\eta(\hat{L}, \hat{R}) \le \mu(\hat{L}, \hat{R}) \frac{\|(R_1, R_2)\|_F}{(\alpha + \beta)\|(\hat{L}, \hat{R})\|_F + \gamma}, \tag{4.4}$$

where

$$\mu(\hat{L}, \hat{R}) = \frac{(\alpha + \beta)\|(\hat{L}, \hat{R})\|_F + \gamma}{(\alpha^2 \sigma_n(\hat{R})^2 + \beta^2 \sigma_m(\hat{L})^2 + \gamma^2)^{\frac{1}{2}}} \ge 1, \tag{4.5}$$

is a *growth factor* that measures by how much the backward error $\eta(\hat{L},\hat{R})$, at most, can be greater than the relative residual as defined in (4.4).

In the following, we discuss forward error bounds expressed in terms of the residuals (4.1). The first is a normwise error bound for a computed solution [15]:

$$\frac{\|(L,R)-(\hat{L},\hat{R})\|_F}{\|(L,R)\|_F} \leq \|Z^{-1}\|_2 \frac{\|(R_1,R_2)\|_F}{\|(L,R)\|_F}. \tag{4.6}$$

The normwise error bound (4.6) holds in general but can be weaker than one based on componentwise errors [15]. The rounding errors in the computed residuals can be expressed as

$$\begin{aligned} \hat{R}_1 &= \mathrm{fl}(C-(A\hat{R}-\hat{L}B)) = R_1+\Delta R_1, \\ \hat{R}_2 &= \mathrm{fl}(F-(D\hat{R}-\hat{L}E)) = R_2+\Delta R_2, \end{aligned}$$

where

$$\begin{aligned} |\Delta R_1| &\leq u\left(3|C|+(m+3)|A||\hat{R}|+(n+3)|\hat{L}||B|\right) \equiv {R_1}^u, \\ |\Delta R_2| &\leq u\left(3|F|+(m+3)|D||\hat{R}|+(n+3)|\hat{L}||E|\right) \equiv {R_2}^u, \end{aligned}$$

and u is the unit roundoff. Introducing

$$g = |\mathrm{col}(\begin{bmatrix} \hat{R}_1 & \hat{R}_2 \end{bmatrix})| + \mathrm{col}(\begin{bmatrix} {R_1}^u & {R_2}^u \end{bmatrix}),$$

and $G = \mathrm{diag}(g)$, we get the bound

$$\frac{\|(L,R)-(\hat{L},\hat{R})\|_M}{\|(\hat{L},\hat{R})\|_M} \leq \frac{\|\,|Z^{-1}|g\|_M}{\|(\hat{L},\hat{R})\|_M} = \frac{\|Z^{-1}G\|_M}{\|(\hat{L},\hat{R})\|_M}. \tag{4.7}$$

From (4.7) we see that large elements in the kth column of $|Z^{-1}|$ can be offset by a small kth element of g. This situation can never be reflected in the bound (4.6) which in these cases is a weaker bound. Accordingly, (4.7) has better scaling properties than (4.6) but none of the bounds are invariant under diagonal scalings of the (generalized) Sylvester equation [14, 15]. The factor $\|Z^{-1}G\|_M$ in (4.7) can be estimated by using the techniques of the one–norm–based estimator for $\|Z^{-1}\|_2$.

5 LAPACK–Style Software

Following the LAPACK conventions and standards [1], most computations described in sections 2, 3 and 4 have been implemented in Fortran 77 routines. In the following, we describe the top–level computational routines in some detail, while the auxiliary routines are just mentioned briefly. We also discuss some implementation details concerning the robustness of our computational kernels.

5.1 Subroutine Organization

The subroutine hierarchy of our software is described in Figure 5.1. LAPACK routines are used by other routines to compute machine dependent thresholds, generalized Schur forms of matrix pairs, matrix norms, and to copy matrices, perform column– and row–swapping and so on. BLAS routines are used to perform basic linear algebra operations such as matrix–matrix (Level 3), matrix–vector (Level 2) and vector (Level 1) operations.

_GGSYX is a driver routine that solves a generalized Sylvester equation

$$\begin{aligned} AR - LB &= sC, \\ DR - LE &= sF, \end{aligned} \tag{5.1}$$

where $(A, D), (B, E)$ and (C, F) are given general pairs of $m \times m$, $n \times n$, and $m \times n$ matrices, respectively, and $0 \leq s \leq 1$ is an output scale factor (named SCALE in the parameter list). SCALE is set to avoid overflow in (L, R) during the computations. Optionally, _GGSYX computes an estimate of $\mathrm{Dif}[(A, D), (B, E)]$. On output DIF holds the value of the reciprocal of the one–norm–based lower bound on $\|Z^{-1}\|_2$. Similarly, the componentwise forward error bound (4.7) is computed optionally (named FERR in the parameter list). By knowing an estimate on $\mathrm{Dif}[(A, D), (B, E)]$ it is also possible for the user to evaluate the conventional normwise error bound (4.6), which is not explicitly done by our software. Moreover, if at least one of DIF and FERR is asked for, then _GGSYX also delivers the relative residual $\|(R_1, R_2)\|_F / ((\alpha + \beta)\|(\hat{L}, \hat{R})\|_F + \gamma)$ in RELRES, which is a lower bound on the normwise relative backward error with respect to the Frobenius norm (see (4.3)). The choice between different options is specified by the input parameter SENSE. Notice that FERR and RELRES are computed with respect to the scaling factor SCALE. The calling sequence and the leading comment lines of DGGSYX are listed in Appendix A.

_TGSYL is a driver routine that solves standard generalized Sylvester equations (5.1) or transposed systems (conjugate transposed systems in the complex case)

$$\begin{aligned} A^T U + D^T V &= sC, \\ -UB^T - VE^T &= sF, \end{aligned} \tag{5.2}$$

where the matrix pairs (A, D) and (B, E) are in generalized Schur form. To solve these (quasi) triangular generalized Sylvester equations, _TGSYL uses the Level 3 algorithms presented in Section 2. Also here, SCALE is set to avoid overflow during the computations of (L, R) or (U, V). The choice between solving a "triangular" system (5.1) or (5.2) is specified by the input parameter TRANS. _TGSYL optionally computes a Frobenius norm–based estimate of $\mathrm{Dif}[(A, D), (B, E)]$. The choice of estimator and the functionality to be performed is specified by the input parameter IJOB (DIF = 1 / ϕ_F(BSOLVD) or 1 / ϕ_F(BSOLVE)). The calling sequence and the leading comment lines of DTGSYL are listed in Appendix B.

_GGSYX calls _TGSYL, but both can be used as driver routines. _TGSYL is a level 3 routine that calls _TGSY2, which solves for subsystems in Algorithm (I) and Algorithm (II) using Level 2 and Level 1 operations. Optionally, _TGSY2 contributes to the computation of a Frobenius norm-based upper bound on the separation of two matrix pairs. _GELUF is used to LU–factorize subsystems (2.3) or (2.4) at the inner–most level. This is done with complete pivoting (i.e. $Z_{ij} = VLUW$) and singularity of Z_{ij} is checked for and dealt with (see Section 5.2). _GELUS performs the forward and backward substitutions to solve for x in $Z_{ij}x = b$. Here is where the scaling is done if overflow in x is likely to appear. _TDIFE computes a Frobenius norm–based Dif–estimator as $1/\phi_F$(BSOLVE) (3.4). _TDIFD computes a Frobenius norm–based Dif–estimator as $1/\phi_F$(BSOLVD) (3.5).

Following the LAPACK conventions for naming, _ in _YYZZZ stands for S(ingle), D(ouble), C(omplex) or Z (Double complex). All four variants of the routines are available. The calling sequences for the real and complex routines are almost the same, except that an extra array RWORK is needed in the expert driver CGGSYX. Scalar arguments are always real. For one of the auxiliary routines, CTGSY2, the calling sequence also differs somewhat, but this does not affect the user of CGGSYX or CTGSYL.

5.2 Solving Subsystems and Dealing with Singularity

At the inner–most level of Algorithm (I) and Algorithm (II) subsystems (2.3) and (2.4), respectively, of size 2, 4 or 8 are solved. Here, we denote one subsystem of either type by $Z_{ij}x = b$. Gaussian elimination with complete pivoting and tests for overflow are used to solve these subsystems. This will hopefully prohibit a large growth factor (in the residual bounds [15]), despite the fact that examples have been found with growth factors greater than the problem size [11].

If a diagonal entry u_{kk} of U in the LU factorization of the coefficient matrix Z_{ij} is less than some SMIN, this entry is set to SMIN and we solve a slightly perturbed system. Notice that the input matrices A, B, D and E, whose entries are involved in different subsystems, are not changed. SMIN is computed as the the maximum of SMLNUM and $\mathtt{EPS} \cdot \|Z\|_M$, where EPS is the relative machine precision and SMLNUM is the safe minimum of the machine (i.e. its reciprocal does not overflow) divided by EPS. If this happens INFO is set to k in _TGSYL telling us that "the matrix pairs (A, D) and (B, E) have common or very close eigenvalues". This fact is also signaled by a small value of DIF (of size $O(\mathtt{EPS} \cdot \|Z\|_M)$).

Having the LU factorization $VLUW = Z_{ij}$ computed by _GELUF, _GELUS solves for x in $Z_{ij}x = sb$, where s is a local scaling factor determined during the solution of the subsystem. Initially, the global SCALE and the local s are set to 1.0. Then we use the following approach [1]. Solve for y in $VLy = b$, followed by the test: if $\tau \cdot \mathtt{SMLNUM} \cdot \|y\|_\infty > |u_{nn}|$, s is set to $1/(\tau\|y\|_\infty)$ and y and SCALE are scaled with s. Finally, the solution x in $UWx = y$ is computed by back–substitution. We use the value 2.0 for the parameter τ.

6 Computational Experiments

As shown in the introduction the generalized Sylvester equation can be formulated in terms of a block–diagonalizing equivalence transformation of $S - \lambda T$ (1.2). In our computational experiments we keep $m + n$, the size of the square S and T constant, while varying m and n. This corresponds to vary the size of the eigenspaces associated with the $(1, 1)$–block of $S - \lambda T$.

Our computational experiments include timing and performance results of the Level 3 and Level 1-2 generalized Sylvester solvers, a comparison between different Dif–estimators including accuracy, performance and reliability results. Finally, we summarize some qualitative results from our test software.

6.1 Timing and Performance Results

All results presented in sections 6.1 and 6.2 are computed on an IBM RS6000/530 in double precision real arithmetic with unit roundoff $\approx$ 2.2D$-$16.

In Table 6.1 we display timing results (in *secs*) for the Level 1–2 (DTGSY2) and Level 3 (DTGSYL) block algorithms. The problems are generated as follows. First, (A, D) and (B, E) in upper triangular generalized Schur form are chosen as

$$a_{ij} = j, \quad d_{ij} = i, \quad i = 1, \ldots, m, \quad j = i, \ldots, m,$$

$$b_{ij} = e_{ij} = -i - j, \quad i = 1, \ldots, n, \quad j = i, \ldots, n.$$

Then, A and B are made quasi upper triangular with A having $m/2$ 2-by-2 diagonal blocks and every 16–th diagonal block of B is chosen 2-by-2. This is done by setting $a_{k+1,k+1} = a_{kk}$ and $a_{k+1,k} = -a_{k,k+1} \cdot \mathit{rand}$ for appropriate values of k, where *rand* is a random number chosen uniformly distributed in $(0, 1)$, and similarly for selected entries in B. The entries of L and R are chosen as random numbers with uniform distribution in $(0, 1)$, defining the right hand sides C and F in (1.1).

The results shown are for some sample m and n, where $m + n = 256, 512$ and 1024. DTGSY2 and DTGSYL perform the same number of flops and computed the same solutions within working accuracy. Here $M = 2$ and $N = 1$ (in Table 6.1) correspond to the Level 1–2 algorithm. Several combinations of block sizes M and N have given the same timing results. We have chosen to display the smallest $M \geq 2$ and $N \geq 2$ that give the best timing results. The speedup factor (Level $1 - 2$/Level 3 in Table 6.1) for this particular choice of block sizes is also shown. We notice a significant improvement in performance for the Level 3 algorithm (up to a factor nine for $m + n = 1024$).

m	n	M	N	*secs*	$\frac{\text{Level1}-2}{\text{Level3}}$	m	n	M	N	*secs*	$\frac{\text{Level1}-2}{\text{Level3}}$
2	254	2	1	.22D+0	1.00	16	240	2	1	.11D+1	1.00
2	254	2	32	.70D−1	3.14	16	240	8	16	.50D+0	2.20
8	248	2	1	.57D+0	1.00	128	128	2	1	.34D+1	1.00
8	248	8	16	.26D+0	2.19	128	128	8	8	.22D+1	1.55
2	510	2	1	.86D+0	1.00	16	496	2	1	.68D+1	1.00
2	510	2	32	.17D+0	5.05	16	496	4	16	.13D+1	5.23
8	504	2	1	.33D+1	1.00	128	384	2	1	.35D+2	1.00
8	504	8	8	.67D+0	4.92	128	384	16	16	.80D+1	4.38
2	1022	2	1	.37D+1	1.00	16	1008	2	1	.29D+2	1.00
2	1022	2	32	.46D+0	8.04	16	1008	16	16	.31D+2	9.35
8	1016	2	1	.14D+2	1.00	128	896	2	1	.19D+3	1.00
8	1016	8	16	.16D+1	8.75	128	896	16	16	.22D+2	8.64

Table 6.1: *Timing and performance results for Level 1–2 and Level 3 solvers*

In Table 6.2 we display similar results (as in Table 6.1) for each of the three Dif–estimators discussed in Section 3. We keep $m + n = 512$ and vary m and n and the block sizes M and N. The first four rows show results for DTDIFE, the next four rows for DTDIFD, and the last four rows show results for the one–norm–based Dif–estimator in DGGSYX. As before, we display the smallest block sizes $M \geq 2$ and $N \geq 2$ that give the best timing results. DTDIFE shows the best improvement in performance for the Level 3 algorithm (up to a factor five faster than the Level 1-2 algorithm). In DTDIFE we solve one generalized Sylvester equation, where we also choose the right hand sides during the computations, and the performance behaviour is quite similar to the Level 3 solver itself. The corresponding improvements for the other two estimators are up to a factor two and three times faster, respectively.

m	n	M	N	$secs$	$\frac{\text{Level1-2}}{\text{Level3}}$	m	n	M	N	$secs$	$\frac{\text{Level1-2}}{\text{Level3}}$
2	510	2	1	.91D+0	1.00	16	496	2	1	.77D+1	1.00
2	510	2	32	.20D+0	4.50	16	496	8	8	.15D+1	5.13
8	504	2	1	.40D+1	1.00	128	384	2	1	.40D+2	1.00
8	504	4	32	.75D+0	5.33	128	384	8	16	.90D+1	4.44
2	510	2	1	.15D+1	1.00	16	496	2	1	.12D+2	1.00
2	510	2	64	.79D+0	1.90	16	496	4	16	.61D+1	1.97
8	504	2	1	.62D+1	1.00	128	384	2	1	.68D+2	1.00
8	504	8	8	.29D+1	2.14	128	384	8	16	.38D+2	1.79
2	510	2	1	.29D+1	1.00	16	496	2	1	.36D+2	1.00
2	510	2	16	.16D+1	1.24	16	496	8	16	.15D+2	2.40
8	504	2	1	.11D+2	1.00	128	384	2	1	.17D+3	1.00
8	504	8	16	.49D+1	1.92	128	384	8	16	.60D+2	2.83

Table 6.2: *Timing and performance results for Level 1–2 and Level 3 Dif–estimators*

In Table 6.3 we present the relative performance of the three Dif–estimators. We display the ratios `DTDIFD/DTDIFE` and `DGGSYX/DTDIFE` of the timings in Table 6.2 for each m and n. As expected, the Frobenius norm–based estimator in `DTDIFE` is the least expensive Dif–estimator (roughly the same cost as solving one quasi triangular generalized Sylvester equation). The Frobenius norm–based estimator in `DTDIFD` and the one–norm–based estimator in `DGGSYX` both solve several quasi triangular generalized Sylvester equations using reverse communication.

m	n	DTDIFD / DTDIFE	DGGSYX / DTDIFE	m	n	DTDIFD / DTDIFE	DGGSYX / DTDIFE
2	510	1.65	3.19	16	496	1.56	4.68
2	510	3.95	8.00	16	496	4.07	10.0
8	504	1.55	2.75	128	384	1.70	4.25
8	504	3.87	6.53	128	384	4.22	6.67

Table 6.3: *Relative performance results for Level 1–2 and Level 3 Dif–estimators*

6.2 Accuracy and Reliability Results

We have developed two test programs `_CHK1` and `_CHK2` for testing and verification of the routines `_GGSYX` and `_TGSYL`, respectively. Their leading comment lines are listed in appendices C and D, respectively. The output from test runs on two different machines is also included. These programs typically verify that the relative residuals for a computed solution are small, that scaling and singularity are appropriately handled, and that Dif–estimates and forward error–estimates are within some (user–defined) tolerances. Moreover, the functionality of the target subroutine is verified. The test problems are chosen from five different types, which are generated as follows.

The size of a problem is chosen such that $m + n \leq 10$ (i.e. $m = 1, 2, \ldots, 9$, $n = 1, \ldots, 10 - m$), where the unknowns L and R are m-by-n. The scheme is to initialize the matrix pairs $(A, D), (B, E)$ and the solution (L, R), and thereby defining the right hand sides C and F in (1.1). Note that Type 1 corresponds to a standard eigenvalue problem and types 2–5 correspond to generalized eigenvalue problems $S - \lambda T$.

Type 1: Upper triangular problem adapted from [25]: $A = J_m(1, -1), D = I_m$ and $B = J_n(1 - \alpha, 1), E = I_n$, where $J_n(d, s)$ denotes a Jordan block of dimension n with d and s as diagonal and superdiagonal elements, respectively. In our tests we use $\alpha = 0.5, \sqrt{\mathtt{EPS}}$, and $1/\sqrt{\mathtt{EPS}}$. For $0 < \alpha < 1$, the size of Dif is $O(\alpha^{m+n-1})$. The entries in $L = R$ are chosen as $r_{ij} = 20(0.5 - \sin(i/j))$, where $i = 1, \ldots, m$ and $j = 1, \ldots, n$.

Type 2: Upper triangular problem with

$$a_{ij} = 2(0.5 - \sin(i)), \quad d_{ij} = 2(0.5 - \sin(i * j)), \quad i = 1, \ldots, m \quad j = i, \ldots, m.$$

$$b_{ij} = 2(0.5 - \sin(i + j)), \quad e_{ij} = 2(0.5 - \sin(j)), \quad i = 1, \ldots, n \quad j = i, \ldots, n.$$

$$l_{ij} = 20(0.5 - \sin(i + j)), \quad r_{ij} = 20(0.5 - \sin(i * j)), \quad i = 1, \ldots, m \quad j = 1, \ldots, n.$$

Type 3: Quasi upper triangular problem, where the entries in $(A, D), (B, E)$ and (L, R) at first are chosen as for Type 2. Then each second diagonal block in A and each third diagonal block in B are made 2-by-2 by setting $a_{k+1,k+1} = a_{kk}$ and $a_{k+1,k} = -\sin(a_{k,k+1})$, and similarly for the entries in B. In the corresponding diagonal blocks of D and E, the superdiagonal element is set to zero, giving a standardized, generalized real Schur form as defined in LAPACK. For the complex case Type 3 is similar to Type 2 except that a_{11} is multiplied by a factor twenty.

Type 4: Dense problem with

$$a_{ij} = 20(0.5 - \sin(i)), \quad d_{ij} = 2(0.5 - \sin(j)), \quad i, j = 1, \ldots, m.$$

$$b_{ij} = 20(0.5 - \sin(i + j)), e_{ij} = 2(0.5 - \sin(i * j)), \text{ where } i, j = 1, \cdots, n.$$

$$l_{ij} = 20(0.5 - \sin(i * j)), \quad r_{ij} = 2(0.5 - \sin(i/j)), \quad i = 1, \ldots, m \quad j = 1, \cdots, n.$$

Type 5: The matrix pairs (A, D) and (B, E) potentially have close or common eigenvalues, and large or very large–normed solutions: A is chosen as the m-by-m leading submatrix of

$$S = \begin{pmatrix} 1 & \beta & & & & & & & \\ -\beta & 1 & & & & & & & \\ & & 1+\delta & \beta & & & & & \\ & & -\beta & 1+\delta & & & & & \\ & & & & \delta & 1 & & & \\ & & & & -1 & \delta & & & \\ & & & & & & -\delta & 1 & \\ & & & & & & -1 & -\delta & \\ & & & & & & & & 1 \end{pmatrix},$$

and B is chosen as the n-by-n leading submatrix of

$$T = \begin{pmatrix} -1 & \beta & & & & & & & \\ -\beta & -1 & & & & & & & \\ & & 1-\delta & \beta & & & & & \\ & & -\beta & 1-\delta & & & & & \\ & & & & \delta & 1+\beta & & & \\ & & & & -1-\beta & \delta & & & \\ & & & & & & -\delta & 1+\beta & \\ & & & & & & -1-\beta & -\delta & \\ & & & & & & & & 1-\delta \end{pmatrix}.$$

Furthermore, we choose $D = I_m$ and $E = I_n$. All integer values $m, n \geq 0$ such that $m+n \leq 10$ are chosen. Values of $\beta = 20/\alpha$ and $\delta = -1.5/\alpha$ are used for three different α (= $0.5, 1/\sqrt{\texttt{EPS}}$ and $10/\texttt{SMLNUM}$). The entries in L and R are chosen as $l_{ij} = \alpha(0.5-\sin(i+j))/20$ and $r_{ij} = \alpha(0.5 - \sin(i * j))/20$, where $i = 1, \cdots, m$ and $j = 1, \cdots, n$.

In tables 6.4–6.5 we display results for 18 examples from the five problem types, ranging from well–conditioned to very ill–conditioned problems. The column "Tag" in Table 6.4 is used to identify a particular problem. Problems 1–6 are of Type 1, 7–8 of Type 2, 9–10 of Type 3, 11–12 of Type 4, and finally, problems 13–18 are of Type 5. Most of the results displayed refer to quantities computed by DGGSYX and DTGSYL, respectively.

The columns Dif and $\|(L, R)\|_F$ in Table 6.4 display the "true" value of Dif (computed as $\sigma_{\min}(Z)$ by DGESVD in LAPACK) and the norm of the exact solution (L, R), respectively. We see that problems 3, 4, 12, 17 and 18 all have Dif at unit roundoff level, which means that the corresponding systems (1.6) are almost singular. For problems 3 and 4 any perturbation at roundoff level in the $(m, 1)$ and $(n, 1)$ elements of the Jordan blocks A and B, respectively, will make (A, D) and (B, E) to have eigenvalues that almost coalesce. Problems 5, 6, 15–18 have coefficient matrices of different size and illustrate problems with badly scaled data (especially, problems 17 and 18).

FERR/Ferr in Table 6.5 shows the ratio between the computed estimate of the relative forward error and the "true" computed forward error $\|(L, R) - (\tilde{L}, \tilde{R})\|_F / \|(L, R)\|_F$, where (L, R) and $(\tilde{L}, \tilde{R})$ are the exact and computed solutions, respectively. Furthermore, r_1 = Dif/DGGSYX shows the ratio between the "true" value of Dif and the one–norm–based Dif–estimator computed in DGGSYX. Similarly, r_{FE} = Dif/DTDIFE and r_{FD} = Dif/DTDIFD in Table 6.6 show the ratios between the "true" value of Dif and the two Frobenius norm–based estimators. The column DGECON in Table 6.6 displays an one–normed–based estimate of Dif = $\sigma_{\min}(Z)$ computed by the LAPACK general condition estimator DGECON. We called DGECON with the LU-decomposition computed by our routine DGELUF in order to make fair comparisons. Notice, $\sigma_{\min}(Z)$ and the estimate from DGECON can only be computed for rather small problems (both require $O(m^3n^3)$ flops). In order to be able to compare the ratios r_1, r_{FE} and r_{FD} we have to multiply r_1 by a factor $1/\sqrt{2mn}$ (the one–norm–based estimate times $\sqrt{2mn}$ is an upper bound on Dif (3.6)). Now, the reliability of the estimators is proved if $0 < r_{\mathrm{x}} \leq 1$ for x = 1, FE, FD and their accuracy is determined by how close the products are to one ($r_x = 1$ means that we have equality in one of the upper bounds for Dif = $\sigma_{\min}(Z)$). Overall, all three estimators show reliable and accurate results. For problems 3 and 4, which are singular to roundoff level, all estimators underestimate the true value of

Tag	m	n	α	$\|(A,D)\|_F$	$\|(B,E)\|_F$	$\|(C,F)\|_F$	Dif	$\|(L,R)\|_F$
1	2	3	.50D+00	.22D+01	.24D+01	.28D+02	.99D−02	.30D+02
2	5	4	.50D+00	.37D+01	.28D+01	.70D+02	.56D−04	.74D+02
3	2	3	.15D−07	.22D+01	.28D+01	.28D+02	.83D−17	.30D+02
4	5	4	.15D−07	.37D+01	.33D+01	.74D+02	.13D−17	.74D+02
5	2	3	.67D+08	.22D+01	.12D+09	.14D+10	.10D+01	.30D+02
6	5	4	.67D+08	.37D+01	.13D+09	.35D+10	.10D+01	.74D+02
7	2	3		.30D+01	.53D+01	.21D+03	.49D−01	.58D+02
8	5	4		.82D+01	.81D+01	.55D+03	.22D−03	.11D+03
9	2	3		.30D+01	.48D+01	.18D+03	.12D+00	.58D+02
10	5	4		.81D+01	.78D+01	.53D+03	.27D−02	.11D+03
11	2	3		.15D+02	.63D+02	.15D+04	.14D−02	.33D+02
12	5	4		.91D+02	.80D+02	.31D+04	.17D−17	.73D+02
13	2	3	.50D+00	.47D+01	.39D+02	.19D+01	.43D+00	.73D−01
14	5	4	.50D+00	.71D+02	.56D+02	.40D+01	.43D+00	.14D+00
15	2	3	.67D+08	.20D+01	.24D+01	.11D+08	.15D−06	.98D+07
16	5	4	.67D+08	.30D+01	.28D+01	.23D+08	.15D−06	.18D+08
17	2	3	.10+292	.20D+01	.24D+01	.16+291	.34D−16	.15+291
18	5	4	.10+292	.30D+01	.28D+01	.33+291	.66D−17	.27+291

Table 6.4: *Some characteristics of 18 sample problems*

Dif with quite a large factor, but signal the severe ill–conditioning correctly. Consequently, this fact is also reflected in the estimated forward error, which (naturally) is large for all problems that are singular to roundoff level. However, we can see that our software always compute a small relative residual even if the forward error is large.

DTGSYL returns an INFO greater than zero whenever one subsystem (2.3) or (2.4) is singular (problems 12, 17 and 18 in Table 6.6, see Section 5.2 for the value of INFO). Notice that problems 3 and 4 do not fulfill our condition to signal singularity. For that to happen (i.e. INFO is set greater than zero), α must be $<$ EPS. Similarly, DGGSYX returns INFO = 1 if DTGSYL returns an INFO greater than zero when solving (1.1). Notice that scaling is performed for problems 17 and 18.

Problems of Type 4 can be very ill–conditioned or even singular, because the matrices A and B may have almost multiple rows and D will have almost multiple columns. This will introduce close to zero elements on the diagonals in the generalized Schur forms of (A, D) and (B, E) when DGEGS (that implements the QZ algorithm in LAPACK) is applied to the matrix pairs. For example, the computed spectrum of problem 12 includes zero (i.e. $0/T_{ii}$), infinite (i.e. $S_{jj}/0$), and undefined (i.e. $0/0$) eigenvalues. An undefined eigenvalue shows that the pencil is singular. Anyhow, our software behaves robustly. All three Dif–estimators produce accurate results.

We also substituted the *sin*-function in all our problems and β and δ in problems of Type 5 with a uniformly random number $\in (0, 1)$. We run thousands of tests and the Dif-estimators, DTDIFE, DTDIFD and DGGSYX differed from the true value of Dif with more than a factor 100 only in 7 %, 1 % and 7 % of the cases, respectively. Our accuracy and reliability tests on small problems show that all three estimators behave well.

Tag	INFO	SCALE	RELRES	FERR	$\frac{\texttt{FERR}}{\text{Ferr}}$	DGGSYX	r_1
1	0	.10D+01	.00D+00	.16D−12	.27D+04	.67D−02	.15D+01
2	0	.10D+01	.93D−17	.38D−10	.42D+04	.37D−04	.15D+01
3	0	.10D+01	.19D−24	.26D+10	.26D+10	.82D−32	.10D+16
4	0	.10D+01	.11D−16	.31D+11	.31D+11	.35D−64	.37D+47
5	0	.10D+01	.11D−16	.22D−14	.27D+02	.10D+00	.10D+01
6	0	.10D+01	.12D−16	.28D−14	.44D+02	.10D+00	.10D+01
7	0	.10D+01	.19D−16	.13D−13	.30D+02	.57D−01	.85D+00
8	0	.10D+01	.22D−16	.13D−10	.62D+02	.14D−03	.16D+01
9	0	.10D+01	.61D−16	.14D−13	.30D+02	.91D−01	.13D+01
10	0	.10D+01	.35D−16	.19D−11	.13D+03	.13D−02	.20D+01
11	0	.10D+01	.15D−15	.35D−11	.73D+01	.14D−02	.10D+01
12	1	.10D+01	.15D−16	.12D+05	.12D+05	.31D−17	.56D+00
13	0	.10D+01	.50D−17	.24D−14	.10D+02	.38D+00	.11D+01
14	0	.10D+01	.21D−16	.31D−14	.16D+02	.38D+00	.11D+01
15	0	.10D+01	.19D−16	.73D−08	.29D+02	.14D−06	.11D+01
16	0	.10D+01	.25D−16	.12D−07	.53D+02	.13D−06	.11D+01
17	1	.15−289	.12D−16	.64D+01	.12D+02	.11D−15	.30D+00
18	1	.84−290	.11D−16	.88D+01	.13D+02	.11D−15	.60D−01

Table 6.5: *Some computed quantities for the verification of* DGGSYX

Tag	INFO	SCALE	DTDIFE	r_{FE}	DTDIFD	r_{FD}	DGECON
1	0	.10D+01	.15D−01	.64D+00	.15D−01	.64D+00	.61D−02
2	0	.10D+01	.18D−03	.36D+00	.18D−03	.36D+00	.32D−04
3	0	.10D+01	.20D−31	.41D+15	.20D−31	.41D+15	.67D−16
4	0	.10D+01	.16D−63	.83D+46	.16D−63	.83D+46	.17D−16
5	0	.10D+01	.14D+01	.71D+00	.10D+01	.10D+01	.10D+01
6	0	.10D+01	.14D+01	.71D+00	.10D+01	.10D+01	.10D+01
7	0	.10D+01	.11D+00	.43D+00	.10D+00	.46D+00	.29D−01
8	0	.10D+01	.10D−02	.22D+00	.95D−03	.23D+00	.11D−03
9	0	.10D+01	.27D+00	.44D+00	.15D+00	.79D+00	.66D−01
10	0	.10D+01	.92D−02	.27D+00	.10D−01	.26D+00	.15D−02
11	0	.10D+01	.32D−02	.45D+00	.26D−02	.55D+00	.11D−02
12	2	.10D+01	.80D−17	.21D+00	.39D−17	.53D+00	.31D−15
13	0	.10D+01	.92D+00	.46D+00	.58D+00	.74D+00	.25D+00
14	0	.10D+01	.13D+01	.34D+00	.82D+00	.52D+00	.25D+00
15	0	.10D+01	.26D−06	.58D+00	.21D−06	.71D+00	.14D−06
16	0	.10D+01	.30D−06	.50D+00	.33D−06	.45D+00	.13D−06
17	4	.11−289	.19D−15	.17D+00	.16D−15	.21D+00	.11D−15
18	8	.78−290	.18D−15	.38D−01	.19D−15	.34D−01	.11D−15

Table 6.6: *Some computed quantities for the verification of* DTGSYL

m	n	$\frac{\texttt{DTDIFE}}{\text{Dif}}$	$\frac{\texttt{DTDIFD}}{\text{Dif}}$	$\frac{\texttt{DGGSYX}}{\text{Dif}}$	Dif	$\sigma_{\min}(Y)$	RELRES	FERR
2	30	0.21	0.19	1.92	.12D−01	.12D−01	.22D−16	.78D−11
4	28	0.11	0.19	2.63	.15D−01	.17D−01	.37D−16	.16D−09
16	16	0.11	0.13	2.38	.40D−03	.42D−03	.52D−16	.29D−09
2	62	0.37	0.37	3.12	.10D+00	.15D+00	.22D−16	.25D−09
4	60	0.29	0.34	3.22	.38D−01	.65D−01	.36D−16	.12D−08
8	56	0.16	0.13	1.37	.24D−02	.35D−02	.38D−16	.18D−07
16	48	0.08	0.05	4.00	.17D−04	.18D−03	.50D−16	.31D−06
2	98	0.13	0.12	1.32	.34D−01	.36D−01	.20D−16	.12D−08
4	96	0.18	0.17	1.39	.69D−01	.80D−01	.25D−16	.15D−08
8	92	0.10	0.08	3.57	.14D−02	.41D−02	.36D−16	.11D−06
m	n	DTDIFE	DTDIFD	DGGSYX	Dif	$\sigma_{\min}(Y)$	RELRES	FERR
2	510	.64D−01	.57D−01	.53D−02	–	.27D+00	.25D−16	.16D−05
8	504	.28D−01	.26D−01	.25D−03	–	.80D−01	.38D−16	.75D−05
16	496	.39D−03	.30D−03	.52D−05	–	.20D−02	.53D−16	.53D−03
128	384	.53D−48	.47D−48	.11D−50	–	.84D−15	.23D−19	.59D+03

Table 6.7: *Comparing Dif–estimators for large problems.*

Finally, we try to compare our Dif–estimators for a larger problem (the same problem as used in Section 6.1). In Table 6.7, we display the true value of Dif and how it relates to the different estimates for some problems with $m+n \leq 100$ and $m+n = 512$. We also show $\sigma_{\min}(Y)$, where Y is the Kronecker matrix constructed from the pencils (A, D) and (B_1, E_1), where B_1 and E_1 are the leading 2-by-2 blocks of B and E, respectively. It is obvious that $\sigma_{\min}(Y) \geq \sigma_{\min}(Z) = \text{Dif}$. For the large problems, we only display the computed estimates and $\sigma_{\min}(Y)$. Note that for the very last problem we try to estimate $\sigma_{\min}(Z)$, where Z is a matrix of size 98304×98304. For $m + n \leq 100$, DTDIFE and DTDIFD produce similar results, overestimating Dif by a factor roughly between 3 and 10. The estimator in DGGSYX underestimates Dif by a similar factor for the same problems. The upper bound $\sigma_{\min}(Y)$ seems to get weaker as m grows. Likely, these trends hold even for larger problems, giving some support for our estimators when $m + n = 512$.

6.3 A Summary of Results from the Test Programs

Below, we summarize the test results from DCHK1 and DCHK2, which tests DGGSYX and DTGSYL, respectively. The results for the single complex test runs were similar. We run both programs on two different target machines, both supporting IEEE standards. A Sun Sparc station using only non-optimized BLAS, and an IBM RS6000/530, where we used the ESSL library for BLAS calls. All programs were compiled with the -O option. The relative machine precision EPS is $\approx$ 2.22D−16 and SMLNUM is $\approx$ 1.0D−292 for both machines. In the following, we always start to list a number for Sun Sparc followed by the corresponding number (in parentheses) for IBM RS6000/530.

A total number of 900 test configurations (including different functionality tests for some problems) are generated in DCHK1. In the test programs only $\alpha = 0.5$ is used for problems of Type 1. The maximum and minimum values of SCALE were 1.0 (1.0) and 0.840D−290 (0.840D−290), respectively. The maximum relative residual `RELRES` was 0.258D−15 (0.149D−15). The maximum value of the true relative forward error was 1.55 (1.55) and 10.0 (10.0) for the corresponding estimate `FERR`. The minimum value of the true relative forward error was 0.00 (0.00) and 0.344D−14 (0.344D−14) for `FERR`. The maximum value of the true value of Dif was 1.41 (1.41) and 1.0 (1.0) for the corresponding estimate `DIF`. The minimum value of the true value of Dif was 0.00 (0.520D−48) and 0.111D−15 (0.111D−15) for `DIF`, but not for the same problem on different machines. The estimated forward error differed more than a factor 1000 from the true value in 28 (50) cases out of 450 (i.e. 6 % and 11 %, respectively). The (one–norm–based) Dif estimate differed more than a factor 100 in 7 (16) cases out of 450 (i.e. 2 % and 4 %, respectively). The relative residual `RELRES` was never larger than 10`EPS`. Singularity was reported by `DTGSYL` in 190 (190) cases.

A total number of 2250 test configurations (including different functionality tests for some problems) are generated in `DCHK2`. The maximum and minimum values of `SCALE` were 1.0 (1.0) and 0.395D−290 (0.395D−290), respectively. The maximum relative residual `RELRES` was 0.808D−16 (0.797D−16). The maximum of the true value of Dif was 1.41 (1.41) and the corresponding estimate was 1.41 (1.41) for both estimators `DTDIFE` and `DTDIFD`. The minimum of the true value of Dif was 0.00 (0.157D−18) and the corresponding estimates were 0.192D−15 (0.655D−16) and 0.192D−15 (0.277D−16) for `DTDIFE` and `DTDIFD`, respectively, but not for the same problems on different machines. The Dif estimate differed more than a factor 100 in 23 (16) cases out of 450 for `DTDIFE` (i.e. 5 % and 4 %, respectively). The Dif estimate differed more than a factor 100 in 23 (16) cases out of 450 for `DTDIFD` (i.e. 5 % and 4 %, respectively). The relative residual `RELRES` was never larger than 10`EPS`. Singularity was reported by `DTGSYL` in 502 (502) cases.

7 Some Conclusions

The computational experiments presented in Section 6 give us support to state the following conclusions about our algorithms and software.

- Accuracy and performance results comparing the Level 3 and Level 1-2 generalized Sylvester solvers show the advantage of the Level 3 algorithm on hierarchical memory architectures (e.g., RISC–based workstations).

- The qualitative comparison, including accuracy, performance and reliability results, between the different Dif–estimators (the Frobenius norm–based `DTDIFE`, `DTDIFD` in `DTGSYL`, and the one–norm–based estimator in `DGGSYX`) can be summarized as follows. All three estimators give very accurate results, typically, within a factor 2–10 and seldom worse than a factor 100 of the true value of $\mathrm{Dif}[(A, D), (B, E)]$, for the problems we have studied. `DTDIFD` and the estimator in `DGGSYX` are both based on `DLACON` in LAPACK and are, typically, a factor 3–10 times as expensive as `DTDIFE`. To include the one–norm–based estimator in `DGGSYX` makes the condition estimation uniform with LAPACK. To choose `DTDIFE` in `DTGSYL` gives the user an equally reliable, low–cost Dif–estimator.

- Qualitative results from our test software on both well–conditioned and ill–conditioned problems, including estimates of the forward error (`FERR`) and backward error (`RELRES`), and the accuracy of the Dif–estimators, show the reliability and robustness of the algorithms and software presented.

The application of this work into algorithms and software for computing eigenspaces with specified eigenvalues of a regular matrix pair (A, B) and condition estimation for the generalized eigenvalue problem is well on the way [18].

Acknowledgements

We are grateful to Zhaojun Bai and Jim Demmel for fruitful discussions on this work. A special thank to Charles Van Loan for constructive discussions concerning the interpretation of the transposed problem $Z^T y = b$ as a (generalized) matrix equation.
Financial support has been received from the Swedish National Board of Industrial and Technical Development under grant NUTEK 89-02578P.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[2] Z. Bai, J. Demmel, and McKenney A. On computing condition numbers for the nonsymmetric eigenproblem. *ACM Trans. Math. Software*, 19(2):202–223, 1993.

[3] R. H. Bartels and G. W. Stewart. Solution of the equation $AX + XB = C$. *Comm. Assoc. Comput. Mach.*, 15:820–826, 1972.

[4] K-W. E. Chu. The solution of the matrix equations $AXB - CXD = E$ and $(YA - DZ, YC - BZ) = (E, F)$. *Lin. Alg. Appl.*, 93:93–105, 1987.

[5] A. Cline, C. Moler, G.W. Stewart, and J. Wilkinson. An estimate of the condition number of a matrix. *SIAM J. Num. Anal.*, 16:368–375, 1979.

[6] J. Demmel and B. Kågström. Computing stable eigendecompositions of matrix pencils. *Lin. Alg. Appl.*, 88/89:139–186, April 1987.

[7] J. Dongarra, J. Du Croz, I. Duff, and S. Hammarling. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Software*, 16:1–17, 1990.

[8] J. D. Gardiner, A. L. Laub, J. A. Amato, and C. B. Moler. Algorithm 705: A software package for solving the Sylvester matrix equation $AXB^T + CXD^T = E$. *ACM Trans. Math. Software*, 18(2):232–238, 1992.

[9] J. D. Gardiner, A. L. Laub, J. A. Amato, and C. B. Moler. Solution of the Sylvester matrix equation $AXB^T + CXD^T = E$. *ACM Trans. Math. Software*, 18(2):223–231, 1992.

[10] G. Golub, S. Nash, and C. Van Loan. A Hessenberg-Schur method for the matrix problem $AX + XB = C$. *IEEE Trans. Autom. Contr.*, AC-24:909–913, 1979.

[11] N. Gould. On growth in Gaussian elimination with pivoting. *SIAM J. Matrix Anal. Appl.*, 12(2):354–361, 1990.

[12] W. W. Hager. Condition estimators. *SIAM J. Sci. Stat. Comput.*, 5:311–316, 1984.

[13] N. J. Higham. ALGORITHM 674: Fortran codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation. *ACM Trans. Math. Software*, 15(2):168, 1989.

[14] N. J. Higham. Perturbation theory and backward error analysis for $AX - XB = C$. *BIT*, 33(1):124–136, 1993.

[15] B. Kågström. A Perturbation Analysis of the Generalized Sylvester Equation. Report UMINF-92.17, Institute of Information Processing, University of Umeå, S-901 87 Umeå, Sweden, 1992. To appear in *SIAM J. on Matrix Anal. Appl.*

[16] B. Kågström. A Direct Method for Reordering Eigenvalues in the Generalized Real Schur Form of a Regular Matrix Pair (A, B). In M.S. Moonen, G.H. Golub, and B.L.R. De Moor, editors, *Linear Algebra for Large Scale and Real–Time Applications*, pages 195–218. Kluwer Academic Publishers, Amsterdam, 1993.

[17] B. Kågström and P. Poromaa. Distributed and shared memory block algorithms for the triangular Sylvester equation with Sep^{-1} estimators. *SIAM J. Matrix. Anal. Appl.*, 13(1):99–101, 1992.

[18] B. Kågström and P. Poromaa. Computing Eigenspaces with Specified Eigenvalues of a Regular Matrix Pair (A, B) and Condition Estimation: Theory, Algorithms and Software. Report UMINF-94.04, Institute of Information Processing, University of Umeå, S-901 87 Umeå, Sweden, 1994.

[19] B. Kågström and P. Van Dooren. A generalized state-space approach for the additive decomposition of a transfer matrix. *Int. J. Numerical Linear Algebra with Applications*, 1(2):165–181, 1992.

[20] B. Kågström and L. Westin. Generalized Schur methods with condition estimators for solving the generalized Sylvester equation. *IEEE Trans. Autom. Contr.*, 34(4):745–751, 1989.

[21] C. Moler and G. Stewart. An algorithm for the generalized matrix eigenvalue problem. *SIAM J. Numer. Anal.*, 10:241–256, 1973.

[22] P. Poromaa. On Efficient and Robust Estimators for the Separation between two Regular Matrix Pairs with Applications in Condition Estimation. Report UMINF-94.xx, Institute of Information Processing, University of Umeå, S-901 87 Umeå, Sweden, 1994. To appear.

[23] G. W. Stewart. Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM Review*, 15(4):727–764, Oct 1973.

[24] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory.* Academic Press, New York, 1990.

[25] J. Varah. On the separation of two matrices. *SIAM J. Numer. Anal.*, 16:216–222, 1979.

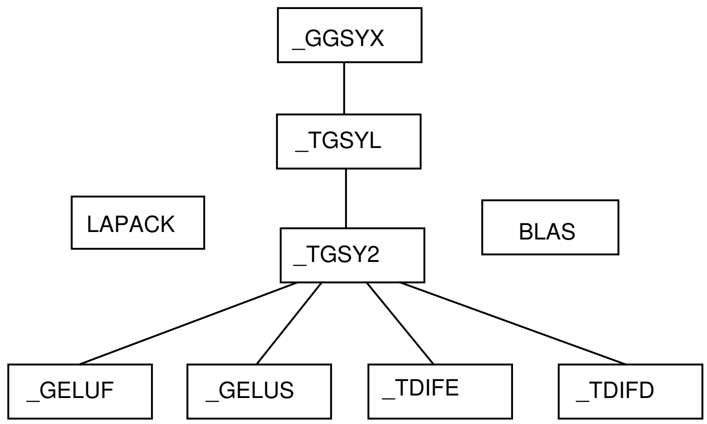

Figure 1: Subroutine hierarchy.

A Calling sequence _GGSYX

Here we display the parameter list and the leading comment lines of the double precision routine DGGSYX.

```
      SUBROUTINE DGGSYX( SENSE, M, N, A, LDA, B, LDB, C, LDC,
     $                   D, LDD, E, LDE, F, LDF, FERR, RELRES, DIF,
     $                   SCALE, WORK, LWORK, IWORK, INFO )
      IMPLICIT NONE
*
*     --- (preliminary version) ---
*     Bo Kagstrom and Peter Poromaa, Institute of Information Processing,
*     Univ. of Umea, S-901 87 Sweden.
*     Nov 1993
*
*     .. Scalar Arguments ..
      DOUBLE PRECISION   FERR, RELRES, SCALE, DIF
      INTEGER            INFO, LDA, LDB, LDC, LDD, LDE, LDF, LWORK, M, N
      CHARACTER          SENSE
*     ..
*     .. Array Arguments ..
      DOUBLE PRECISION   A(LDA, *), B(LDB, *), C(LDC, *),
     $                   D(LDD, *), E(LDE, *), F(LDF, *), WORK( * )
      INTEGER            IWORK( * )
*     ..
*
*  Purpose
*  =======
*  DGGSYX solves the generalized Sylvester equation:
*
*        A * R - L * B = scale * C
*        D * R - L * E = scale * F
*
*  where R and L are unknown m-by-n matrices, (A, D), (B, E) and
*  (C, F) are given matrix pairs of size m-by-m, n-by-n and m-by-n, respectively,
*  with real entries.
*  The solution (R, L) overwrites (C, F). 0 <= SCALE <= 1 is an output scaling
*  factor chosen to avoid overflow.
*
*  References
*  ==========
*  [1] B. Kagstrom and P. Poromaa, LAPACK-Style Algorithms and Software for
*      Solving the Generalized Sylvester Equation and Estimating the Separation
*      between Regular Matrix Pairs, Report UMINF - 93.23, Inst. of Information
*      Processing, University of Umea, S-901 87 Umea, Sweden, November 1993.
*      (also published as LAPACK Working Note xx)
*
*  [2] B. Kagstrom, A Perturbation Analysis of the Generalized Sylvester Equation
*      (AR - LB, DR - LE ) = (C, F), Report UMINF - 92.17, Inst. of Information
*      Processing, University of Umea, S-901 87 Umea, Sweden, November 1992,
*      (accepted for publication in SIAM J. Matrix Anal. Appl.)
```

```
*
*  [3] N.J. Higham, Perturbation theory and backward error for AX - XB = C,
*      Numerical Analysis Report No. 211, University of Manchester, England,
*      April 1992; (published in BIT 33 (1) 124-136, 1993).
*
*  Arguments
*  =========
*
*  SENSE   (input) CHARACTER*1
*          Determines which of Dif([A,B], [D,E]) and the forward error bound is
*          computed.
*          = 'N': Neither is computed.
*          = 'F': Forward error bound is computed.
*          = 'D': Dif[(A,D), (B,E)] is computed.
*          = 'B': Both are computed.
*          If SENSE = 'F', 'D' or 'B', RELRES is computed.
*
*  M       (input) INTEGER
*          The number of rows and columns of the input matrices A and D. M >= 0.
*
*  N       (input) INTEGER
*          The number of rows and columns of the input matrices B and E. N >= 0.
*
*  A, D    (input/output) DOUBLE PRECISION arrays, dimension (LDx,M).
*          On input, A and D are dense m-by-m coefficient matrices.
*          On output, (A, D) has been overwritten by its generalized Schur form.
*
*  LDA     (input) INTEGER
*          The leading dimension of the array A.  LDA >= max(1,M).
*
*  LDD     (input) INTEGER
*          The leading dimension of the array D.  LDD >= max(1,M).
*
*  B, E    (input/output) DOUBLE PRECISION arrays, dimension (LDx,N).
*          On input, B and E are the n-by-n coefficient matrices.
*          On output, (B, E) has been overwritten by its generalized Schur form.
*
*  LDB     (input) INTEGER
*          The leading dimension of the array B.  LDB >= max(1,N).
*
*  LDE     (input) INTEGER
*          The leading dimension of the array E.  LDE >= max(1,N).
*
*  C, F    (input/output) DOUBLE PRECISION arrays, dimension (LDx,N).
*          On input, C and F are the m-by-n right-hand side coefficient matrices.
*          On output, C has been overwritten by the solution matrix R, and F has
*          been overwritten by the solution matrix L.
*
*  LDC     (input) INTEGER
*          The leading dimension of the array C.  LDC >= max(1,M).
*
```

```
*  LDF      (input) INTEGER
*           The leading dimension of the array F.  LDF >= max(1,M).
*
*  WORK     (workspace) DOUBLE PRECISION array, dimension (LWORK).
*
*  LWORK    (input) INTEGER
*           The dimension of the array WORK.
*           LWORK >= 4*(M*M + N*N) + 6*M*N + 4*max(M,N).
*           For good performance, LWORK must generally be larger.
*
*  IWORK    (workspace) INTEGER array, dimension (2*M*N).
*
*  FERR     (output) DOUBLE PRECISION
*           On exit, if SENSE = 'F' or 'B',
*           an estimated forward error bound for the solution (L, R).
*           If (LTRUE, RTRUE) is the true solution, FERR bounds the magnitude
*           of the largest entry in ((L, R) - (LTRUE, RTRUE)) divided by
*           the magnitude of the largest entry in (L, R).
*
*  RELRES   (output) DOUBLE PRECISION
*           On exit, if SENSE = 'F', 'D' or 'B',
*           the relative residual for the computed solution, measured
*           in the Frobenius norm,
*
*                   norm(AR-LB-scale*C, DR-LE-scale*F)
*           -------------------------------------------------------
*           ((norm(A, D) + norm(B, E))*norm(L, R) + scale*norm(C, F) .
*
*           Note: This is a lower bound for the normwise relative backward error.
*
*  DIF      (output) DOUBLE PRECISION
*           On exit, if SENSE = 'D' or 'B',
*           this is an one-norm-based estimate of Dif[(A,D), (B,E)], the
*           separation between to regular matrix pairs.
*           The estimate is computed with a condition estimator.
*           Dif[(A,D), (B,E)] >= 0 is the standard measure of how ill-conditioned
*           the generalized Sylvester equation is: the smaller Dif, the greater the
*           ill-conditioning. Dif = 0 (or small) corresponds to that the matrix
*           pairs (A,D) and (B,E) have common (or close) eigenvalues.
*           See [1, 2] for a full explanation.
*
*  SCALE    (output) DOUBLE PRECISION
*           The scale factor, is set 0 <= SCALE <= 1 to avoid overflow in (L, R).
*
*  INFO     (output) INTEGER
*             0: successful exit.
*             1: (A, D) and (B, E) have common or very close eigenvalues
*                (INFO > 0 from DTGSYL).
*             2: Problems when computing generalized Schur forms of (A, D) or (B, E)
*                in DGEGS, which is called by this routine.
*             3: Both the above problems.
```

```
*          < 0: if INFO = -k, the k-th argument had an illegal value.
*
* ===============================================================================
```

B Calling sequence _TGSYL

Here we display the parameter list and the leading comment lines of the double precision routine DTGSYL.

```
      SUBROUTINE DTGSYL(TRANS, IJOB, M, N, A, LDA, B, LDB, C, LDC,
     $                  D, LDD, E, LDE, F, LDF, SCALE, DIF, WORK, LWORK,
     $                  IWORK, INFO)

       IMPLICIT NONE
*   -- (preliminary version) --
*      Bo Kagstrom and Peter Poromaa, Institute of Information Processing,
*      Univ. of Umea, S-901 87 Sweden.
*      Nov 1993

*      .. Scalar Arguments ..
       CHARACTER        TRANS
       INTEGER          IJOB, M, N, LDA, LDB, LDC, LDD, LDE, LDF,
     $                  LWORK, INFO
       DOUBLE PRECISION SCALE, DIF

*       .. Array Arguments ..
       INTEGER          IWORK(*)
       DOUBLE PRECISION A(LDA, *), B(LDB, *), C(LDC, *), D(LDD, *),
     $                  E(LDE, *), F(LDF, *), WORK(*)

*
*   Purpose
*   =======
*
*       DTGSYL solves the generalized Sylvester equation:
*
*              A * R - L * B = scale * C                 (1)
*              D * R - L * E = scale * F
*
*       where R and L are unknown m-by-n matrices, (A, D), (B, E) and
*       (C, F) are given matrix pairs of size m-by-m, n-by-n and m-by-n,
*       respectively, with real entries. (A, D) and (B, E) must be in
*       generalized (real) Schur canonical form, i.e. A, B are upper quasi
*       triangular and D, E are upper triangular.
*       The solution (R, L) overwrites (C, F). 0 <= SCALE <= 1 is an output
*       scaling factor chosen to avoid overflow.
*       In matrix notation (1) is equivalent to solve
*       Zx = scale b, where Z is defined as
*
*             Z = [ kron(In, A)  -kron(B', Im) ]         (2)
```

```
*                 [ kron(In, D)  -kron(E', Im) ].
*
*       Here Ix is the identity matrix of size x and X' is the
*       transpose of X. kron(X, Y) is the Kronecker product between
*       the matrices X and Y. Dim(Ix) = m or n.
*
*       If TRANS = 'T', y in the transposed system Z'y = scale * b is solved for,
*       which is equivalent to solve for R and L in
*
*              A' * R  + D' * L   = scale *  C           (3)
*              R  * B' + L  * E'  = scale * -F
*
*       This case (TRANS = 'T') is used to compute an one-norm-based estimate
*       of Dif[(A,D), (B,E)], the separation between the matrix pairs (A,D) and
*       (B,E), using DLACON. See DGGSYX. See [1-2] for more information.
*
*       If IJOB >= 1, DTGSYL computes a Frobenius norm-based estimate
*       of Dif[(A,D),(B,E)]. That is, the reciprocal of a lower bound on the
*       reciprocal of the smallest singular value of Z. See [1-2] for more
*       information.
*
*       Level 3 and Level 1-2 algorithms for solving (1) are described in [1,3].
*
*   References
*   ==========
*  [1] B. Kagstrom and P. Poromaa, LAPACK-Style Algorithms and Software for
*      Solving the Generalized Sylvester Equation and Estimating the Separation
*      between Regular Matrix Pairs, Report UMINF - 93.23, Inst. of Information
*      Processing, University of Umea, S-901 87 Umea, Sweden, November 1993.
*      (also published as LAPACK Working Note xx)
*
*  [2] B. Kagstrom, A Perturbation Analysis of the Generalized Sylvester Equation
*      (AR - LB, DR - LE ) = (C, F), Report UMINF - 92.17, Inst. of Information
*      Processing, University of Umea, S-901 87 Umea, Sweden, November 1992,
*      (accepted for publication in SIAM J. Matrix Anal. Appl.)
*
*  [3] B. Kagstrom and L. Westin, Generalized Schur Methods with Condition
*      Estimators for Solving the Generalized Sylvester Equation, IEEE
*      Transactions on Automatic Control, Vol. 34, No. 7, July 1989, pp 745-751.
*
*   Arguments
*   =========
*   TRANS        'N': The generalized Sylvester equation (1) is solved.
*                     Other functionality is depending on the value of IJOB.
*                'T': The ''transposed'' system (3) is solved.
*                     The value of IJOB is ignored.
*
*   IJOB    (input) INTEGER
*              Specifies what kind of functionality to be performed.
*              0 :    Only (1) is solved. Scaling is used when solving sub-systems.
*                     DGELUF (complete pivoting) and DGELUS (scaling) is used.
```

```
*                      0. <= SCALE <= 1.0D0 will be returned.
*               1 :    The functionality of 0 and 3.
*               2 :    The functionality of 0 and 4.
*               3 :    Only an estimate of Dif[(A,D), (B,E)] is computed.
*                      DTDIFE (look ahead strategy) is used.
*               4 :    Only an estimate of Dif[(A,D), (B,E)] is computed.
*                      DTDIFD (DLACON on sub-systems) is used.
*               For more details on the Frobenius norm-based estimators see [1].
*
*   M            (input) INTEGER
*                On entry, M specifies the order of A and D, and the row
*                dimension of C, F, R and L.
*
*   N            (input) INTEGER
*                On entry, N specifies the order of B and E, and the column
*                dimension of C, F, R and L.
*
*   A            (input) DOUBLE PRECISION array, dimension (LDA, M)
*                On entry, A contains an upper quasi triangular matrix.
*
*   LDA          (input) INTEGER
*                The leading dimension of the matrix A. LDA >= max(1, M).
*
*   B            (input) DOUBLE PRECISION array, dimension (LDB, N)
*                On entry, B contains an upper quasi triangular matrix.
*
*   LDB          (input) INTEGER
*                The leading dimension of the matrix B. LDB >= max(1, N).
*
*   C            (input/ output) DOUBLE PRECISION array, dimension (LDC, N)
*                On entry, C contains the right-hand-side of the first matrix
*                equation in (1) or (3).
*                On exit, if IJOB = 0, 1 or 2, C has been overwritten by
*                the solution R. If IJOB = 3 or 4 and TRANS = 'N', C holds
*                R, the solution achieved during the computation of the Dif-estimate.
*
*   LDC          (input) INTEGER
*                The leading dimension of the matrix C. LDC >= max(1, M).
*
*   D            (input) DOUBLE PRECISION array, dimension (LDD, M)
*                On entry, D contains an upper triangular matrix.
*
*   LDD          (input) INTEGER
*                The leading dimension of the matrix D. LDD >= max(1, M).
*
*   E            (input) DOUBLE PRECISION array, dimension (LDE, N)
*                On entry, E contains an upper triangular matrix.
*
*   LDE          (input) INTEGER
*                The leading dimension of the matrix E. LDE >= max(1, N).
*
```

```
*   F            (input/ output) DOUBLE PRECISION array, dimension (LDF, N)
*                On entry, F contains the right-hand-side of the first matrix
*                equation in (1) or (3).
*                On exit, if IJOB = 0, 1 or 2, F has been overwritten by
*                the solution L. If IJOB = 3 or 4 and TRANS = 'N', F holds
*                L, the solution achieved during the computation of the Dif-estimate.
*
*   LDF          (input) INTEGER
*                The leading dimension of the matrix F. LDF >= max(1, M).
*
*   DIF          (output) DOUBLE PRECISION
*                On exit DIF is the reciprocal of a lower bound of the reciprocal
*                of the Dif-function, i.e. DIF is an upper bound of
*                    Dif[(A,D), (B,E)] = sigma-min(Z), where Z as in (2).
*                IF IJOB = 0 or TRANS = 'T', DIF is not touched.
*                For more information see [1].
*
*   SCALE        (output) DOUBLE PRECISION
*                On exit SCALE is the scaling factor in (1) or (3). If 0 < SCALE < 1,
*                C and F hold the solutions R and L, respectively, to a slightly
*                perturbed system but the input matrices A, B, D and E have not
*                been changed. If SCALE = 0, C and F hold the solutions R and L,
*                respectively, to the homogeneous system with C = F = 0.
*                Normally, SCALE = 1.
*
*   WORK        (workspace) DOUBLE PRECISION array, dimension (LWORK)
*
*   LWORK       (input) INTEGER
*                The dimension of the array WORK. LWORK > = 1.
*                If IJOB = 1 or 2 and TRANS = 'N', LWORK >= 2*M*N.
*
*   IWORK        (workspace) INTEGER array, dimension (M + N + 6)
*
*   INFO         (output) INTEGER
*                On exit, if INFO is set to
*                 0 : Normal return
*               -k : Input argument number k is illegal.
*               >0 : (A, D) and (B, E) have common or very close eigenvalues.
*
*  ===============================================================================
```

C Test program DCHK1

Here we display the leading comment lines of the double precision routine DCHK1 for testing DGGSYX. The output from test runs on a Sun Sparc and an IBM RS6000/530 is also included.

```
      PROGRAM DCHK1
      IMPLICIT NONE
*     Peter Poromaa,Institute of Information Processing
*     Univ. of Umea, S 901 87 Umea,Sweden
```

```
*     November 1993
*
*
*  Purpose
*  =======
*
*  DCHK1 is a test code for the expert driver DGGSYX that
*  solves the real generalized Sylvester equation:
*    A * R - L * B = SCALE* C
*    D * R - L * E = SCALE* F
*
* The test code verifies that:
*
* (1) 0 <= SCALE <= 1.
*
* (2) the relative residual RELRES is small (of order one), i.e.
*              norm(A*R-L*B-scale*C, D*R+L*E-scale*F)
*     -----------------------------------------------------------   < TOL1
*     EPS*(((norm(A, D) + norm(B, E))*norm(L, R) + scale*norm(C, F))
*
* (3)  the estimated forward error FERR does not differ from the true forward
*      error more than a factor TOL2. If the true value equals zero, FERR
*      must not differ from EPS more than a factor TOL2.
*
* (4) the estimate DIF of Dif[(A, D),(B, E)] does not differ from the true value
*     of Dif[.] more than a factor TOL3. If the true value equals zero DIF must
*     not differ from EPS more than a factor TOL3.
*
* (5) INFO >=  0 is returned by DGGSYX.
*
* (6) INFO =   0 is returned by DGESVD.
*
* (7) If INFO = 1 or 3 is returned by DGGSYX, the system is regarded as
*     numerically singular and we count those events.
*
* (8) If INFO = 2 or 3 is returned by DGGSYX, the routine DGEGS
*     had problems to compute the generalized real Schur forms and
*     we count those events.
*
*  DGESVD is used for computing the true value of Dif.
*  In the above, EPS is the relative machine precision. The variable FAIL(i)
*  will report the number of fails for test (i). If test (i) never fails
*  this will not be reported.
*
*  The parameters TOLi and NOUT may be changed by the user.
*  No other changes to the code should be done.
*  NOUT is the unit on which the test results are reported.
*
*  Ouput from this program on Sun Sparc and IBM RS6000/530:
*  Sun 4 SPARC :
****************** Start of test *****************
```

```
* Test expert driver DGGSYX:  900 no. of problems.
* Max and min values of SCALE:   0.100D+01   0.840-290
* Max value of relative residual RELRES   0.258D-15 for example no  550
* Max value of true forward error and corresponding estimate FERR
*  0.155D+01   0.100D+02 for example no.  736
* Min value of true forward error and corresponding estimate FERR
*  0.000D+00   0.344D-14 for example no.    2
* Max value of true Dif and corresponding estimate DIF:
*  0.141D+01   0.100D+01 for example no.  723
* Min value of true Dif and corresponding estimate DIF:
*  0.000D+00   0.111D-15 for example no.  732
* Estimated forward error differs more than a factor 0.100D+04
* from true value in   28 cases out of  450
* Estimated Dif differs more than a factor 0.100D+03
* from true value in    7 cases out of  450
*  190 no of problems were regarded as "numerically singular".
******************* End of test *******************
* IBM RS6000/530:
******************* Start of test ******************
* Test expert driver DGGSYX:        900  no. of problems.
* Max and min values of SCALE:    .100D+01    .840-290
* Max value of relative residual RELRES    .149D-15 for example no  550
* Max value of true forward error and corresponding estimate FERR
*   .155D+01    .100D+02 for example no.  736
* Min value of true forward error and corresponding estimate FERR
*   .000D+00    .344D-14 for example no.    2
* Max value of true Dif and corresponding estimate DIF:
*   .141D+01    .100D+01 for example no.  723
* Min value of true Dif and corresponding estimate DIF:
*   .520D-48    .111D-15 for example no.  772
* Estimated forward error differs more than a factor  .100D+04
* from true value in   50 cases out of  450
* Estimated Dif differs more than a factor  .100D+03
* from true value in   16 cases out of  450
* 190  no. of problems was regarded as "numerically singular".
******************* End of test *******************
```

D Test program DCHK2

Here we display the leading comment lines of the double precision routine DCHK2 for testing DTGSYL. The output from test runs on a Sun Sparc and an IBM RS6000/530 is also included.

```
      PROGRAM DCHK2
      IMPLICIT NONE
*
*     Peter Poromaa, Institute of Information Processing
*     Univ. of Umea, S-901 87 Umea, Sweden
*     November 1993
*
*  Purpose
```

```
*  =======
*
*  DCHK2 is a test code for the expert driver DTGSYL that
*  solves the real quasi triangular generalized Sylvester equation
*
*    A * R - L * B = SCALE * C
*    D * R - L * E = SCALE * F
*
*  and the "transposed" generalized system
*
*    A' * R +  D' * L  = SCALE *  C
*    R  * B' + L  * E' = SCALE * -F
*
* The test code verifies that:
*
* (1) 0 <= SCALE <= 1.
*
* (2) the relative residual RELRES is small (of order one), i.e.
*            norm(A*R-L*B-SCALE*C, D*R-L*E-SCALE*F)
*     ------------------------------------------------------------ < TOL1
*     EPS*((norm(A, D) + norm(B, E))*norm(L, R) + SCALE*norm(C, F))
*
* (3) or in the "transposed" case the relative residual is small
*            norm(A'*R+D'*L-SCALE*C, R*B'+L*E'+SCALE*F)
*     ------------------------------------------------------------ < TOL1
*     EPS*((norm(A, D) + norm(B, E))*norm(L, R) + SCALE*norm(C, F))
*
* (4) when the auxiliary routine DTDIFE is called, the estimate of Dif does
*     not differ from the true value of Dif more than a factor TOL2.
*     If the true value of Dif equals zero the estimate must not differ from
*     EPS more than a factor TOL2. Not actual for the "transposed" case.
*
* (5) when the auxulary routine DTDIFD is called the estimate of Dif does not
*     differ from the true value of Dif more than a factor TOL2.
*     If the true value of Dif equals zero the estimate must not differ from
*     EPS more than a factor TOL2. Not actual for the "transposed" case.
*
* (6) INFO  >= 0 is returned by DTGSYL.
*
* (7) If INFO > 0 is returned by DTGSYL the system is regarded as
*     numerically singular and we check if the true value of
*     Dif < (EPS*(norm(A, D) + norm(B, E))*norm(L, R) + SCALE*norm(C, F)).
*
* (8) INFO = 0 is returned by DGESVD.
*
* (9) INFO = 0 is returned by DGEGS .
*
*  DGEGS is used to compute generalized real Schur forms of matrix pairs.
*  DGESVD is used for computing the true value of Dif.
*  In the above, EPS is the relative machine precision. The variable FAIL(i)
*  will report the number of fails for test (i). If test (i) never fails
```

```
*  this will not be reported.
*  The parameters TOLi and NOUT may be changed by the user.
*  No other changes to the code should be done.
*  NOUT is the unit on which the test results are reported.
*
*  Ouput from this program on Sun Sparc and IBM RS6000/530:
* Sun SPARC 4
****************** Start of test ******************
* Test expert driver DTGSYL:  2250 no. of problems.
* Max and min value of SCALE:   0.100D+01   0.395-290
* Max value of relative residual RELRES   0.808D-16 for example no 1891
* Max value of true Dif and corresponding estimate for DTDIFE:
*  0.141D+01   0.141D+01 for example no.  902
* Min value of true Dif and corresponding estimate for DTDIFE:
*  0.000D+00   0.192D-15 for example no.  914
* Max value of true Dif and corresponding estimate for DTDIFD:
*  0.141D+01   0.141D+01 for example no.  903
* Min value of true Dif and corresponding estimate for DTDIFD:
*  0.000D+00   0.192D-15 for example no.  915
* Estimated Dif differs more than a factor 0.100D+03
* from true value in   23 cases out of  450 when DTDIFE is  called
* Estimated Dif differs more than a factor 0.100D+03
* from true value in   23 cases out of  450 when DTDIFD is  called
* Singularity reported correctly in  502
* cases and mabye not correctly reported in  0 cases (i.e. test 7 failed).
****************** End of test ******************
* IBM RS6000/530
****************** Start of test ******************
*Test expert driver DTGSYL:       2250  no. of problems.
* Max and min value of SCALE:    .100D+01    .395-290
* Max value of relative residual RELRES    .797D-16 for example no 1891
* Max value of true Dif and corresponding estimate for DTDIFE:
*   .141D+01    .141D+01 for example no.  902
* Min value of true Dif and corresponding estimate for DTDIFE:
*   .157D-18    .655D-16 for example no.  832
* Max value of true Dif and corresponding estimate for DTDIFD:
*   .141D+01    .141D+01 for example no.  903
* Min value of true Dif and corresponding estimate for DTDIFD:
*   .157D-18    .277D-16 for example no.  833
* Estimated Dif differs more than a factor  .100D+03
* from true value in   16 cases out of  450 when DTDIFE is  called
* Estimated Dif differs more than a factor  .100D+03
* from true value in   16 cases out of  450 when DTDIFD is  called
* Singularity reported correctly in 502
* cases and mabye not correctly reported in 0  cases (i.e. test 7 failed).
****************** End of test ******************
```