Solution of the Sylvester matrix equation AXB + CXD = E

Article *in* ACM Transactions on Mathematical Software · June 1992

DOI: 10.1145/146847.146929

4 authors, including:

Cleve Moler
The MathWorks, Inc
133 PUBLICATIONS **12,977** CITATIONS

Solution of the Sylvester Matrix Equation $AXB^T + CXD^T = E$

JUDITH D. GARDINER and ALAN J. LAUB
University of California, Santa Barbara
and
JAMES J. AMATO and CLEVE B. MOLER
University of New Mexico

A software package has been developed to solve efficiently the Sylvester-type matrix equation $AXB^T + CXD^T = E$. A transformation method is used which employs the QZ algorithm to structure the equation in such a way that it can be solved columnwise by a back substitution technique. The algorithm is an extension of the Bartels-Stewart method and the Hessenberg-Schur method. The numerical performance of the algorithms and software is demonstrated by application to near-singular systems.

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*linear systems (direct and iterative methods), conditioning, Sylvester equations*; G.4 [**Mathematics of Computing**]: Mathematical Software—*algorithm analysis, efficiency, reliability and robustness*

General Terms: Algorithms, Performance

1. INTRODUCTION

A matrix equation of interest in control theory is the Sylvester-type equation

$$AXB^T + CXD^T = E \tag{1}$$

in which all matrices are real (the extension to the complex-valued case is trivial), A and C are of dimension $m \times m$, B and D are $n \times n$, and E is $m \times n$. The desired solution X is $m \times n$. Equation (1) is a special case of the

J. D. Gardiner and A. J. Laub were supported in part by the National Science Foundation and AFOSR under grant ECS87-18897.

Authors' addresses: J. D. Gardiner, Dept. of Computer and Information Science, The Ohio State University, Columbus, OH 43210; A. J. Laub, Dept. of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106; J. J. Amato, University of New Mexico, Albuquerque, NM 87131; C. B. Moler, The Mathworks, 325 Linfield Place, Menlo Park, CA 94025.

general linear equation

$$\sum_{i=1}^{N} A_i X B_i = E$$

first studied by Sylvester [15].

Equation (1) has a unique solution if and only if the matrix pencils $A - \lambda C$ and $D - \lambda B$ are regular and the spectrum of one is disjoint from the negative of the spectrum of the other [4]. Thus any one of the matrices A, B, C, and D may be singular without causing the linear operator to be singular. With some restrictions, a second of these matrices may also be singular. Section 6 gives an example with both A and C singular. Our algorithms are capable of handling all these situations.

The brute force way to approach the solution of Eq. (1) is to rewrite it as a linear matrix-vector system in standard form. The system becomes

$$Gy = h \tag{2}$$

in which

$$G = B \otimes A + D \otimes C \tag{3}$$

$$y = (x_{1,1}, x_{2,1}, \ldots, x_{m,1}, x_{1,2}, \ldots, x_{m,n})^T$$

$$h = (e_{1,1}, e_{2,1}, \ldots, e_{m,1}, e_{1,2}, \ldots, e_{m,n})^T$$

The Kronecker product matrix $M \otimes N$ is the block matrix whose (i, j) block is $m_{ij}N$. Equation (2) can be solved by Gaussian elimination. Unfortunately, the coefficient matrix G has dimension $mn \times mn$, making this approach impractical except for small systems.

A second alternative is to cast Eq. (1) in a form for which effective algorithms already exist. It can be put in the form $FX + XG = H$ and solved by means of "standard" algorithms [2, 7] if, say, B and C of Eq. (1) are nonsingular. If we premultiply both sides by C^{-1} and postmultiply by B^{-T} we obtain

$$C^{-1}AX + XD^TB^{-T} = C^{-1}EB^{-T}.$$

A two-equation form could also be used [12]. For example, the pair $AR - LD^T = E$ and $CR + LB^T$ is equivalent to Eq. (1), with $XB^T = R$ and $CX = -L$. Our conditions for solvability of Eq. (1) imply that at least one of the matrices B and C is nonsingular, so X can, in principle, be recovered from L and R. These methods are not generally satisfactory, however, if either B or C is singular or ill-conditioned with respect to inversion. A more general method is desirable.

The two special cases of Eq. (1) which are of particular interest in control theory are the continuous-time symmetric Sylvester equation

$$AXE^T + EXA^T + C = 0 \tag{4}$$

and its discrete-time counterpart

$$AXA^T - EXE^T + C = 0, \tag{5}$$

where C (and hence X) is symmetric. (A and E are not, in general, symmetric.) An example of the need to solve these equations is presented by Arnold and Laub [1], where they arise in connection with iterative improvement for generalized Riccati equations. The symmetry of these equations and of the solution matrices allows them to be solved more efficiently than the general equation.

2. A BARTELS–STEWART METHOD

The Bartels–Stewart method [2] (see also Chu [4] and Kågström and Westin [12]) is a transformation method developed for solving equations of the form $AX + XB = C$. It can be extended to our more general case by rewriting Eq. (1) as

$$(Q_1AZ_1)(Z_1^TXZ_2)(Z_2^TB^TQ_2^T) + (Q_1CZ_1)(Z_1^TXZ_2)(Z_2^TD^TQ_2^T) = Q_1EQ_2^T$$

in which Q_1, Z_1, Q_2, and Z_2, are orthogonal matrices. The QZ algorithm of Moler and Stewart [14] allows us to choose these matrices so that $Q_1AZ_1 = P$ is quasi-upper-triangular, $Q_1CZ_1 = S$ is upper triangular, $Q_2DZ_2 = T$ is quasi-upper-triangular, and $Q_2BZ_2 = R$ is upper triangular.

The QZ algorithm is applied to each of the matrix pairs (A, C) and (D, B). The first matrix of the pair is reduced to quasi-upper-triangular form while the second is made upper triangular. A quasitriangular matrix is block triangular with 1×1 and 2×2 diagonal blocks, the 1×1 blocks corresponding to real generalized eigenvalues of the pencil and the 2×2 blocks to complex generalized eigenvalues.

If we define $Z_1^TXZ_2 = Y$ and $Q_1EQ_2^T = F$, the transformed system becomes

$$PYR^T + SYT^T = F. \tag{6}$$

Because of its special triangular structure this system of equations can be solved by a back substitution technique. Let a_k denote the kth column of the matrix A. If we take the kth column of each side of Eq. (6), we get

$$P\sum_{j=k}^{n} r_{kj}y_j + S\sum_{j=k-1}^{n} t_{kj}y_j = f_k$$

where the summation ranges reflect the upper triangular nature of R and T. This can be rewritten as

$$(r_{kk}P + t_{kk}S)y_k + t_{k,k-1}Sy_{k-1} = f_k - \sum_{j=k+1}^{n}(r_{kj}P + t_{kj}S)y_j$$

or, in a form more suitable for efficient computation,

$$(r_{kk}P + t_{kk}S)y_k + t_{k,k-1}Sy_{k-1} = f_k^{(n-k)}$$

where $f_k^{(j)}$ is defined by the recursion

$$f_k^{(0)} = f_k,$$
$$f_k^{(j)} = f_k^{(j-1)} - r_{kj}(Py_j) - t_{kj}(Sy_j).$$

When y_j has been computed, Py_j and Sy_j are formed and $f_k^{(j)}$ is calculated for $k = 1, 2, \ldots, j-1$.

Equation (6) can be solved columnwise by starting at column n and working backwards to column 1. The columns of Y are computed either one or two at a time, depending on whether or not the subdiagonal element $t_{k,k-1}$ is zero. If $t_{k,k-1}$ is zero, then we can obtain column k by solving the $m \times m$ quasitriangular system

$$(r_{kk}P + t_{kk}S)\,y_k = f_k^{(n-k)}. \tag{7}$$

If the subdiagonal element $t_{k,k-1}$ is not zero, then we utilize the fact that a quasi-upper-triangular matrix cannot have two consecutive nonzero elements along its subdiagonal. This implies that $t_{k-1,k-2}$ is zero, and we can obtain columns $k-1$ and k together by solving the $2m \times 2m$ block structured system

$$\begin{pmatrix} r_{k-1,k-1}P + t_{k-1,k-1}S & r_{k-1,k}P + t_{k-1,k}S \\ t_{k,k-1}S & r_{kk}P + t_{kk}S \end{pmatrix} \begin{pmatrix} y_{k-1} \\ y_k \end{pmatrix} = \begin{pmatrix} f_{k-1}^{(n-k)} \\ f_k^{(n-k)} \end{pmatrix} \tag{8}$$

The rows and columns of this system can be permuted so that in each case the natural order $(1, 2, \ldots, m, m+1, m+2, \ldots, 2m)$ is replaced by the order $(1, m+1, 2, m+2, 3, m+3, \ldots, 2m)$. This restructures the coefficient matrix so that it is block upper triangular, with 2×2 and 4×4 diagonal blocks.

These linear systems of equations can be solved using Gaussian elimination with partial pivoting, yielding the columns of Y. Finally, the desired solution matrix X is obtained from the relation $X = Z_1 Y Z_2^T$.

3. A HESSENBERG–SCHUR METHOD

An alternative transformation method, introduced by Golub et al. [7] (see also Kågström and Westin [12]) has been found effective for the problem $AX + XB = C$. The same technique may be generalized to apply to our problem. It is a Hessenberg–Schur method, so called because the larger of the matrices A and D in Eq. (1), say A, is reduced via the QZ algorithm only to upper Hessenberg form, while D is again reduced to quasi-upper-triangular (real Schur) form. As in the Bartels–Stewart method, the matrices C and B are both reduced to upper triangular form.

More specifically, on the first call to the set of subroutines which implement the QZ algorithm [14] (hereafter referred to as the "QZ package"), in which A and C are reduced, only the first step of the algorithm is applied. That is, C is reduced to upper triangular form but A is reduced only to upper Hessenberg form. On the second call, which reduces D and B, the algorithm runs to completion as before. The remainder of the analysis carries over in the same way as for the Bartels–Stewart method, with the only difference being the number of nonzero subdiagonal elements in the matrix blocks on the left-hand sides of Eqs. 7 and 8. Equation 7 becomes an upper Hessenberg system, while Eq. (8) in its permuted form is upper triangular plus two nonzero subdiagonals.

The resulting savings in computation time on the first QZ call is shown in the next section to more than offset the subsequent complication of the back substitution step.

In the symmetric Eqs. 4 and 5, there is only one pair of matrices, A and E, to be reduced by the QZ algorithm, so the Hessenberg–Schur method offers no advantage over Bartels–Stewart. The back substitution process can be streamlined in this case by computing only the upper triangle of the solution matrix, the rest being known by symmetry.

4. COMPARISON OF THE TWO METHODS

In this section we give a rough work count of the floating-point operations involved in the two methods for solving the general Eq. (1). The symmetric solution method requires about half the work of the general Bartels–Stewart method. Hereafter we shall use the notation BS to refer to the Bartels–Stewart method and HS for Hessenberg–Schur. We will also use WBS and WHS to denote the respective work counts for the two methods. We can divide the computation into five steps.

The first call to the QZ package reduces the matrices A and C. We use the work count estimates of Golub and Van Loan [8]. The HS method reduces A only to upper Hessenberg form, while the BS method carries the reduction through to quasi-upper-triangular form. The transformation matrices Q_1 and Z_1 are accumulated. We obtain WHS = $8.8m^3$ and WBS = $33m^3$. As above, we are assuming without loss of generality that $m \geq n$. If this is not the case, then the algorithm can simply be applied to the transposed problem.

In the second call to the QZ package, D and B are reduced. For this call, the reduction is carried to completion for both matrix pairs. The transformation matrices Q_2 and Z_2 are accumulated. Thus the work counts are the same for this step: WHS = WBS = $33n^3$.

In the third step the right-hand side matrix E is updated to $Q_1 E Q_2^T$. This update requires $m^2n + mn^2$ operations for either method.

In the back substitution step the transformed Eq. (6) is solved columnwise by solving a series of linear systems. These may be either of the $m \times m$ form in Eq. (7) or the $2m \times 2m$ form in Eq. (8). The work count depends on the fraction of nonzero elements in the first subdiagonal of the quasi-upper-triangular matrix T; we designate this fraction by t and note that $0 \leq t \leq 0.5$. There are tn systems of size $2m \times 2m$ to be solved and $(1 - 2t)n$ systems of size $m \times m$. A count for the HS method, retaining only third order terms, gives WHS = $(3 + 5.5t)m^2n + mn^2$. For the BS method, the work count is WBS = $m^2n + mn^2$.

In the last step we find the desired solution matrix $X = Z_1 Y Z_2^T$, which requires $m^2n + mn^2$ operations.

If we add together the work counts for the five steps, we obtain the following estimates for the two methods:

$$\text{WHS} = 8.8m^3 + 33n^3 + (5 + 5.5t)m^2n + 3mn^2,$$

$$\text{WBS} = 33m^3 + 33n^3 + 3m^2n + 3mn^2.$$

To give more insight into the above results, we consider the special case in which $m = n$ and $t = 0.5$. We obtain WHS $\approx 53m^3$ and WBS $\approx 72m^3$.

Comparing these values we note that the HS method enjoys a significant advantage because it does not carry the first QZ transformation to completion (i.e., the iterative step is not performed). This indicates that the algorithm becomes more efficient the more m exceeds n.

A series of timing runs was used to check the HS and BS work count estimates. We used matrices of various dimensions with randomly generated elements in the range -1 to 1. For each choice of m and n we used five different sets of randomly generated coefficient matrices. The matrix E was chosen so that the exact solution X would consist of a matrix with all unit elements.

The program was run using both the HS method and the BS method, and the respective execution times TBS and THS were recorded. The ratio THS/TBS was averaged over the five trials. The results are compared with the expected WHS/WBS, computed using the actual values of t, in Table I. As can be seen, the agreement is quite satisfactory.

5. NUMERICAL PERFORMANCE

The HS method can be expected to exhibit satisfactory numerical properties because it utilizes two algorithms which are normally numerically stable, Gaussian elimination with partial pivoting and the QZ algorithm. While it is true that there are certain pathological cases in which element growth may cause loss of stability for Gaussian elimination with partial pivoting, such situations are rarely encountered in practice [11, 16].

To test the numerical performance of the method, we ran some tests on ill-conditioned problems. If we examine Eq. 3, we note that if we choose B and C to be identity matrices of the proper dimensions, then the coefficient matrix G is the Kronecker sum [13, pp. 259-260] of A and D^T. It has eigenvalues that are simple sums of the eigenvalues of A and D. Therefore, a convenient way to approach singularity is to simultaneously force B and C toward identity matrices and to force the eigenvalues of D to approach the negatives of the eigenvalues of A.

We can do this with the following definitions. Let I_k be an identity matrix of dimension k and U_k be a square $k \times k$ matrix with unit entries below the diagonal and all other entries zero. Then we define

$$
\begin{aligned}
A &= \operatorname{diag}(1, 2, 3, \ldots, m) + U_m, \\
B &= I_n + 2^{-p} U_n^T, \\
C &= I_m + 2^{-p} U_m^T, \quad \text{and} \\
D &= 2^{-p} I_n - \operatorname{diag}(n, n-1, \ldots, 1) + U_n
\end{aligned}
$$

where p is a parameter. In addition, we define E to be the $m \times n$ matrix such that the true solution matrix X has all unit entries. It can easily be seen that this system approaches singularity as p increases.

Table I. Timings

m	n	THS/TBS	WHS/WBS
12	12	0.77	0.73
16	12	0.70	0.65
18	9	0.54	0.52
20	5	0.39	0.36

The software was run with the above input matrices for many different dimensions and for different values of p. In each case the normalized error NE and the normalized residual NR were calculated. These quantities are defined as

$$NE = (\| X - X^* \|) / \| X \|$$

$$NR = (\| AXB^T + CXD^T - E \|) / ((\| X \| (\| A \| \| B \| + \| C \| \| D \|)))$$

in which X is the computed solution, X^* is the true solution, and $\| X \|$ represents a norm of the matrix X. The infinity norm was used for convenience.

In addition, for the purpose of testing the algorithm only, the coefficient matrix G in Eq. (2) was actually formed, and the condition estimator of Cline et al. [5] as implemented in the LINPACK routine DGECO, was used on it. The quantity RCOND is an estimate of the inverse of the condition number of G, that is, RCOND = $1/\kappa(G)$, where $\kappa(G)$ is the condition number of G. The results are presented in Table II for dimensions $m = 10$ and $n = 4$.

As expected, regardless of the condition number of G the algorithm achieves a small normalized residual of the order of magnitude of the unit round-off error for double precision floating-point operations on the machine (VAX 11/780 with all codes in Fortran and run under the f77 compiler with 4.2 bsd Unix). On the other hand, the normalized error does depend on the condition number and is of the order of magnitude of the condition number times the normalized residual.

6. A SMALL EXAMPLE

A 2 × 1 example illustrates the application of the algorithm when two of the coefficient matrices are singular. The matrices are already in generalized Schur form.

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 2 \end{bmatrix}, \quad B = [2], \quad C = \begin{bmatrix} 3 & 4 \\ 0 & 0 \end{bmatrix} \quad D = [1], \quad E = \begin{bmatrix} 9 \\ 4 \end{bmatrix}.$$

Since X has only a single column in this example, just one 2 × 2 system of linear equations must be formed and solved: (2A + 1C)X = E.

$$2A + C = \begin{bmatrix} 0 & 2 \\ 0 & 4 \end{bmatrix} + \begin{bmatrix} 3 & 4 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 5 \\ 0 & 2 \end{bmatrix}.$$

Table II. Errors and Residuals

p	NE	NR	RCOND
0	3.8×10^{-14}	9.8×10^{-17}	8.6×10^{-4}
10	2.1×10^{-11}	5.4×10^{-16}	7.7×10^{-6}
20	1.1×10^{-8}	3.8×10^{-16}	7.4×10^{-9}
30	1.5×10^{-5}	2.6×10^{-16}	7.3×10^{-12}
40	1.2×10^{-2}	3.8×10^{-16}	7.1×10^{-15}

The solution, obtained by back substitution, is $X = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. It is easily seen that the singularity of individual coefficient matrices causes no problem as long as the conditions of Section 1 are satisfied.

7. CONDITION ESTIMATION

The software optionally provides an estimate of the condition number κ:

$$\kappa = \|G\| \|G^{-1}\| \approx (\|A\| \|B\| + \|C\| \|D\|)/\text{SEP}, \tag{9}$$

where $G = B \otimes A + D \otimes C$ and SEP is an estimate of the reciprocal of the norm of the inverse operator, or the "separation,"

$$\text{SEP} = \min_{Z \neq 0} \|AZB^T + CZD^T\| / \|Z\|. \tag{10}$$

If the norm used in Eq. (10) is the Frobenius norm, then SEP can be interpreted as the smallest singular value of G: $\text{SEP} = 1/\|G^{-1}\|_2$. The estimates in this package use the 1-norm.

The algorithm for estimating SEP is a straightforward generalization of the method given by Byers for the equation $AX - XB^T = C$ [3]. Other approaches to estimating the condition of Sylvester equations involve estimating the norm of the inverse of the Sylvester operator Φ defined by $\Phi(X) := AXB^T + CXD^T$. In this respect, Byers [3], Higham [10], Hewer and Kenney [9], and Kågström and Westin [12] can be consulted for details.

8. SOFTWARE PACKAGE

A companion paper [6] describes the actual software package implementing the Hessenberg-Schur algorithm for the general equation and the Bartels-Stewart method for the symmetric equations, with condition estimators for both.

REFERENCES

1. ARNOLD, W. F., AND LAUB, A. J. Generalized eigenproblem algorithms and software for algebraic Riccati equations. *Proc. IEEE 72*, 12 (Dec. 1984), 1746-1754.
2. BARTELS, R. H., AND STEWART, G. W. Solution of the equation $AX + XB = C$. *Commun. ACM 15*, 9 (Sept. 1972), 820-826.
3. BYERS, R. A LINPACK-style condition estimator for the equation $AX - XB^T = C$. *IEEE Trans. Aut. Contr. AC-29*, 10 (Oct. 1984), 926-928.

4. CHU, K.-W. E. The solution of the matrix equation $AXB - CXD = E$ and $(YA - DZ, YC - BZ) = (E, F)$. *Lin. Alg. Appl. 93* (Aug. 1987), 93-105.
5. CLINE, A. K., MOLER, C. B., STEWART, G. W., AND WILKINSON, J. H. An estimate for the condition number of a matrix *SIAM J. Numer. Anal. 16*, 2 (Apr. 1979), 368-375.
6. GARDINER, J. D., WETTE, M. R., LAUB, A. J., AMATO, J. J., AND MOLER, C. B. Algorithm ???: A FORTRAN-77 software package for solving the Sylvester matrix equation $AXB^T + CXD^T = E$. This issue, pp. 000-000.
7. GOLUB, G. H., NASH, S., AND VAN LOAN, C. F. A Hessenberg-Schur method for the problem $AX + XB = C$. *IEEE Trans. Aut. Control. AC-24*, 6 (Dec. 1979), 909-913.
8. GOLUB, G. H., AND VAN LOAN, C. F. *Matrix Computations*. Johns Hopkins Univ. Press, Baltimore, Second Edition, 1989.
9. HEWER, G., AND KENNEY, C. The sensitivity of the stable Lyapunov equation. *SIAM J. Control. Optim. 26*, 2 (Mar. 1988), 321-344.
10. HIGHAM, N. Algorithm 674: FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation. *ACM Trans. Math. Softw. 14*, 4 (Dec. 1988), 381-396.
11. HIGHAM, N. J., AND HIGHAM, D. J. Large growth factors in Gaussian elimination with pivoting. *SIAM J. Matrix Anal. Appl. 10*, 2 (Apr. 1989), 155-164.
12. KÅGSTRÖM, B., AND WESTIN, L. Generalized Schur methods with condition estimators for solving the generalized Sylvester equation. *IEEE Trans. Aut. Control. AC-34*, 7 (July 1989), 745-751.
13. LANCASTER, P. *Theory of Matrices*. Academic Press, New York, 1969.
14. MOLER, C. B., AND STEWART, G. W. An algorithm for generalized matrix eigenvalue problems. *SIAM J. Numer. Anal. 10*, 2 (Apr. 1973), 241-256.
15. SYLVESTER, J J. Sur la solution du cas le plus général des équations linéaires en quantités binaires, c'est-à-dire en quaternions ou en matrices du second ordres. Sur la résolution générale de l'équation linéaire en matrices d'un ordre quelconque. Sur l'équation linéaire trinôme en matrices d'un ordre quelconque. *Comptes Rendus Acad. Sci. 99* (1884), 117-118, 409-412, 432-436, 527-529.
16. TREFETHEN, L. N., AND SCHREIBER, R. S. Average-case stability of Gaussian elimination. *SIAM J. Matrix Anal. Appl. 11*, 3 (July 1990), 335-360.

Received April 1990; revised February 1991; accepted February 1991