A Hessenberg–Schur Method for the Problem $AX+XB=C$

G. H. GOLUB, S. NASH, AND C. VAN LOAN

Abstract—**One of the most effective methods for solving the matrix equation $AX+XB=C$ is the Bartels–Stewart algorithm. Key to this technique is the orthogonal reduction of A and B to triangular form using the QR algorithm for eigenvalues. A new method is proposed which differs from the Bartels–Stewart algorithm in that A is only reduced to Hessenberg form. The resulting algorithm is between 30 and 70 percent faster depending upon the dimensions of the matrices A and B. The stability of the new method is demonstrated through a roundoff error analysis and supported by numerical tests. Finally, it is shown how the techniques described can be applied and generalized to other matrix equation problems.**

I. Introduction

Let $A\in R^{m\times m}$ and $B\in R^{n\times n}$ be given matrices and define the linear transformation ϕ: $R^{m\times n}\rightarrow R^{m\times n}$ by

$$\phi(X)=AX+XB. \tag{1.1}$$

This linear transformation is nonsingular if and only if A and $-B$ have no eigenvalues in common which we shall hereafter assume. Linear equations of the form

$$\phi(X)=AX+XB=C \tag{1.2}$$

arise in many problem areas and numerous algorithms have been proposed [4], [10]. Among them, the Bartels–Stewart algorithm [1] has enjoyed considerable success [2]. In this paper we discuss a modification of their technique which is just as accurate and considerably faster.

This new method is called the "Hessenberg–Schur algorithm" and like the Bartels–Stewart algorithm is an example of a "transformation method." Such methods are based upon the equivalence of the problems

$$AX+XB=C$$

and

$$(U^{-1}AU)(U^{-1}XV)+(U^{-1}XV)(V^{-1}BV)=U^{-1}CV$$

and generally involve the following four steps.

Step 1: Transform A and B into "simple" form via the similarity transformations $A_1=U^{-1}AU$ and $B_1=V^{-1}BV$.
Step 2: Solve $UF=CV$ for F.
Step 3: Solve the transformed system $A_1Y+YB_1=F$ for Y.
Step 4: Solve $XV=UY$ for X.

A brief look at the effect of roundoff error in Steps 2 and 4 serves as a nice introduction to both the Bartels–Stewart and Hessenberg–Schur algorithms.

In these steps linear systems are solved which involve the transformation matrices U and V. Suppose Gaussian elimination with partial pivoting is used for this purpose and that the computations are done on a computer whose floating-point numbers have t base β digits in the mantissa. Using the standard Wilkinson inverse error analysis [12], [13] it follows that relative errors of order $u[\kappa_2(U)+\kappa_2(V)]$ can be expected to contaminate the computed solution $\hat{X}$ where

$$u=\beta^{-t}$$

Manuscript received December 18, 1978; revised July 26, 1979. Paper recommended by A. J. Laub, Chairman of the Computational Methods and Discrete Systems Committee. The work of G. H. Golub was supported in part by DOE Contract EY-76-S-03-0326 PA#30.

G. H. Golub and S. Nash are with the Department of Computer Science, Stanford University, Stanford, CA 94305.

C. Van Loan is with the Department of Computer Science, Cornell University, Ithaca, NY 14853.

is the relative machine precision and $\kappa_2(\cdot)$ is defined by

$$\kappa_2(W)=\|W\|_2\|W^{-1}\|_2$$

$$\equiv \max_{x\neq 0}\sqrt{\frac{(Wx)^T(Wx)}{x^Tx}}\ast\max_{y\neq 0}\sqrt{\frac{y^Ty}{(Wy)^T(Wy)}}.$$

When $\kappa_2(W)$ is large with respect to u, then we say that W is "ill-conditioned."

Unfortunately, several of the possible reductions in Step 1 can lead to ill-conditioned U and V matrices. For example, if A and B are diagonalizable, then there exist U and V so that

$$U^{-1}AU=\text{diag}(\alpha_1,\alpha_2,\cdots,\alpha_m)=A_1$$
$$V^{-1}BV=\text{diag}(\beta_1,\beta_2,\cdots,\beta_m)=B_1.$$

The matrix $Y=(y_{ij})$ in Step 3 is then prescribed by the simple formulas $y_{ij}=f_{ij}/(\alpha_i+\beta_j)$. If we apply this algorithm to the problem

$$A=\begin{bmatrix}1.234567891 & 3.515985621\\ 0 & 1.234078268\end{bmatrix}$$

$$B=\begin{bmatrix}0.3458968425 & 0\\ 0.6521859685 & 0.3450509462\end{bmatrix}$$

$$C=\begin{bmatrix}5.748636323 & 5.095604458\\ 2.232161079 & 1.579129214\end{bmatrix}$$

and use HP-67 arithmetic ($u=10^{-10}$), we find

$$\hat{X}=\begin{bmatrix}1.003948200 & 0.999995000\\ 0.999997700 & 1.000000000\end{bmatrix}.$$

Now in this example, $u[\kappa_2(U)+\kappa_2(V)]=10^{-3}$ and so we should not be surprised to learn that to full working precision

$$X=\begin{bmatrix}1.000000000 & 1.000000000\\ 1.000000000 & 1.000000000\end{bmatrix}.$$

Conclusion: We should avoid ill-conditioned transformation matrices. Methods which involve the computation of Jordan or companion forms in Step 1 do not do this (cf. [6], [9]).

This leads us to consider transformation methods which rely on orthogonal U and V. (Recall that $U^TU=I$ implies $\kappa_2(U)=1$.) In the next two sections we describe two such techniques: one old and one new. The first of these is the Bartels–Stewart algorithm. This method involves the orthogonal reduction of A and B to triangular form using the QR algorithm. The main point of this paper is to show how this algorithm can be streamlined by only reducing A to Hessenberg form. The resulting algorithm is described in Section III and its roundoff properties are shown to be very desirable in Sections IV and V. The superior efficiency of the new method for a large class of problems is substantiated in Section VI where we report on several numerical tests. Finally, we conclude by showing how the techniques developed can be extended to other matrix problems.

II. The Bartels–Stewart Algorithm

The crux of the Bartels–Stewart algorithm [1] is to use the QR algorithm to compute the real Schur decompositions

$$U^TAU=R \qquad V^TB^TV=S \tag{2.1}$$

where R and S are upper quasi-triangular and U and V are orthogonal. (A quasi-triangular matrix is triangular with possible nonzero 2×2 blocks along the diagonal.)

From our remarks in Section I, the reductions (2.1) lead to the transformed system

$$RY+YS^T=F \qquad (F=U^TCV,\ Y=U^TXV). \tag{2.2}$$

Assuming $s_{k,k-1}$ is zero, then it follows that

$$(R+s_{kk}I)y_k=f_k-\sum_{j=k+1}^{n} s_{kj}y_j \tag{2.3}$$

where $Y=[y_1 \mid y_2 \mid \cdots \mid y_n]$ and $F=[f_1 \mid f_2 \mid \cdots \mid f_n]$. Thus, y_k can be found from $y_{k+1},\cdots,y_n$ by solving an upper quasi-triangular system, a very easy computation requiring $m^2/2$ operations once the right-hand side is known. If $s_{k,k-1}$ is nonzero, then y_k and y_{k-1} are "simultaneously" found in an analogous fashion.

If we assume that the Schur decompositions in (2.1) require $10m^3$ and $10n^3$ operations, respectively, then the overall workcount for the Bartels–Stewart algorithm is given by

$$W_{BS}(m,n)=10m^3+10n^3+2.5[m^2n+mn^2].$$

The technique requires $2m^2+2n^2+mn$ storage assuming the data are overwritten.

III. The Hessenberg–Schur Algorithm

In this section we describe a new algorithm, called the Hessenberg–Schur algorithm, which differs from the Bartels–Stewart method in that the decompositions (2.1) are replaced by

$$\begin{aligned} &H=U^TAU && H \text{ upper Hessenberg, } U \text{ orthogonal}\\ &S=V^TB^TV && S \text{ quasi-upper triangular, } V \text{ orthogonal.}\end{aligned} \tag{3.1}$$

A matrix $H=(h_{ij})$ is upper Hessenberg if $h_{ij}=0$ for all $i>j+1$. The orthogonal reduction of A to upper Hessenberg form can be accomplished with Householder matrices in $\frac{5}{3}m^3$ operations. See [12, p. 347] for a description of this algorithm. The reductions (3.1) lead to a system of the form

$$HY+YS^T=F \tag{3.2}$$

which may be solved in a manner similar to what is done in the Bartels–Stewart algorithm. In particular, assume that $y_{k+1},\cdots,y_n$ have been computed.

If $s_{k-1,k}=0$, then y_k can be determined by solving the $m\times m$ Hessenberg system

$$(H+s_{kk}I)y_k=f_k-\sum_{j=k+1}^{n} s_{kj}y_j. \tag{3.3}$$

When Gaussian elimination with partial pivoting is used for this purpose, m^2 operations are needed once the right-hand side is known.

If $s_{k,k-1}$ is nonzero, then by equating columns $k-1$ and k in (3.2) we find

$$H[y_{k-1}|y_k]+[y_{k-1}|y_k]\begin{bmatrix} s_{k-1,k-1} & s_{k,k-1}\\ s_{k-1,k} & s_{kk}\end{bmatrix}=[f_{k-1}|f_k] - \sum_{j=k+1}^{n}[s_{k-1,j}y_j \mid s_{kj}y_j]\equiv[g \mid w]. \tag{3.4}$$

This is a $2m$-by-$2m$ linear system for y_k and y_{k-1}. By suitably ordering the variables, the matrix of coefficients is upper triangular with two nonzero subdiagonals. Using Gaussian elimination with partial pivoting, this system can be solved in $6m^2$ operations once the right-hand side is formed. Unfortunately, a $2m^2$ workspace is required to carry out the calculations.

Part of this increase in storage is compensated for by the fact that the orthogonal matrix U can be stored in factored form below the diagonal H [12, p. 350]. This implies that we do not need an $m\times m$ array for U as in the Bartels–Stewart algorithm. Summarizing the Hessenberg–Schur algorithm and the associated work counts we have the following.

1) Reduce A to upper Hessenberg and B^T to quasi-upper triangular:

$H=U^TAU$	(store U in factored form)	$\frac{5}{3}m^3$
$S=V^TBV$	(save V)	$10n^3$

2) Update the right-hand side:

$F=U^TCV$	m^2n+mn^2

3) Back substitute for Y:

$HY+YS^T=F$	$3m^2n+\frac{1}{2}mn^2$

4) Obtain solution:

$X=UYV^T$	m^2n+mn^2

$$w_{HS}(m,n)=\frac{5}{3}m^3+10n^3+5m^2n+\frac{5}{2}mn^2.$$

To obtain the operation count associated with the determination of Y, we assumed that S has $n/2$ 2×2 bumps along its diagonal. (This is the "worst" case.)

Unlike the work count for the Bartels–Stewart algorithm, $w_{HS}(m,n)$ is not symmetric in m and n. Indeed, scrutiny of $w_{HS}(m,n)$ reveals that it clearly pays to have $m>n$. This can always be assured, for if $m<n$, we merely apply the Hessenberg–Schur algorithm to the transposed problem

$$B^TX^T+X^TA^T=C^T.$$

Comparing $w_{BS}(m,n)$ and $w_{HS}(m,n)$ we find

$$\frac{w_{HS}(m,n)}{w_{BS}(m,n)}=\frac{1+3(n/m)+\frac{3}{2}(n/m)^2+6(n/m)^3}{6+\frac{3}{2}(n/m)+\frac{3}{2}(n/m)^2+6(n/m)^3} \tag{3.5}$$

which indicates that substantial savings accrue when the Hessenberg–Schur method is favored. For example, if $m=4n$, then $w_{HS}(m,n)=0.30w_{BS}(m,n)$.

The storage requirements of the new method are a little greater than those for the Bartels–Stewart algorithm:

$A(m\times m)$	for the original A and subsequently H and U
$B(n\times n)$	for the original B and subsequently S
$V(n\times n)$	for the orthogonal matrix V
$C(m\times n)$	for the original C and subsequently Y and X
$W(2m^2)$	for handling the possible system (3.4).

IV. A Perturbation Analysis

In the next section we shall assess the effect of rounding errors on the Hessenberg–Schur algorithm. The assessment will largely be in the form of a bound on the relative error in the computed solution $\hat{X}$. To ensure a correct interpretation of our results, it is first necessary to investigate the amount of error which we can expect *any* algorithm to generate given finite precision arithmetic.

To do this we need to make some observations about the sensitivity of the underlying problem $\phi(X)=C$. This system of equations can be written in the form

$$Px=c \tag{4.1}$$

where

$$P=(I_n\otimes A)+(B^T\otimes I_m) \tag{4.2}$$

and

$$x=\text{vec}(X)=(x_{11},x_{21},\cdots,x_{m1},x_{12},x_{22},\cdots,x_{m2},\cdots,x_{1n},\cdots,x_{mn})^T$$

$$c=\text{vec}(C)=(c_{11},c_{21},\cdots,c_{m1},c_{12},c_{22},\cdots,c_{m2},\cdots,c_{1n},\cdots,c_{mn})^T.$$

ᐧlere, the Kronecker product $W\otimes Z$ of two matrices W and Z is the ock matrix whose (i,j) block is $w_{ij}Z$.

Based on our knowledge of linear system sensitivity, we know that if P is ill-conditioned, then small changes in A, B, and/or C can induce relatively large changes in the solution. To relate this to the transformation ϕ, we need to define a norm on the space of linear transformations from $R^{m\times n}$ to $R^{m\times n}$:

$$f: R^{m\times n}\to R^{m\times n} \quad \|f\| = \max_{X\in R^{m\times n}} \frac{\|f(X)\|_F}{\|X\|_F}.$$

Here, the Frobenius norm $\|\cdot\|_F$ is defined by $\|W\|_F^2=\Sigma_{i,j}|w_{ij}|^2$. Notice that for the linear transformation ϕ defined by (1.1) we have

$$\|\phi\|=\|P\|_2<\|A\|_2+\|B\|_2$$

where P is defined by (4.2). If ϕ is nonsingular, then

$$\|\phi^{-1}\|=\left[\min_{X\in R^{m\times n}} \frac{\|\phi(X)\|_F}{\|X\|_F}\right]^{-1}=\|P^{-1}\|_2.$$

Now consider solving $AX+XB=C$ on a computer having machine precision u. In general, rounding errors of order $u\|A\|_F$, $u\|B\|_F$, and $u\|C\|_F$ will normally be present in A, B, and C, respectively, *before* any algorithm even begins execution. Hence, the "best" thing we can say about a computed solution $\hat{X}$ is that it satisfies

$$(A+E)\hat{X}+\hat{X}(B+F)=(C+G) \tag{4.3}$$

where

$$\|E\|_F<u\|A\|_F \tag{4.4}$$
$$\|F\|_F<u\|B\|_F \tag{4.5}$$
$$\|G\|_F<u\|C\|_F. \tag{4.6}$$

How accurate would such an $\hat{X}$ be? By applying standard linear system perturbation analysis [3], it is possible to establish the following result.

Theorem: Assume that $AX+XB=C$, $(A+E)\hat{X}+\hat{X}(B+F)=(C+G)$, and (4.4), (4.5), and (4.6) hold. If $\phi(Z)=AZ+ZB$ is nonsingular, C is nonzero, and

$$u[\|A\|_F+\|B\|_F]\|\phi^{-1}\|<1/2, \tag{4.7}$$

then

$$\frac{\|X-\hat{X}\|_F}{\|X\|_F}<4u[\|A\|_F+\|B\|_F]\|\phi^{-1}\|. \tag{4.8}$$

For the 2×2 example given in Section I, the upper bound in (4.8) has a value of about 10^{-9}. This indicates that an $AX+XB=C$ problem can be very well-conditioned even if the eigenvector matrices for A and B are poorly conditioned.

We conclude this section with the remark that the bound in (4.8) can roughly be attained. Setting

$$A=\begin{bmatrix}1&2\\0&1\end{bmatrix}\quad B=\begin{bmatrix}\delta-1&0\\1&3\end{bmatrix}\quad C=\begin{bmatrix}3+\delta&6\\1+\delta&4\end{bmatrix}$$

it is easy to verify that $\kappa(\phi)=\|\phi\|\|\phi^{-1}\|=0(1/\delta)$ and

$$X=\begin{bmatrix}1&1\\1&1\end{bmatrix}.$$

(Think of δ as a small positive constant.) Now if

$$A\hat{X}+\hat{X}B=C+\begin{bmatrix}u&0\\0&0\end{bmatrix}$$

it is easy to show

$$\hat{X}=X+\begin{bmatrix}u/\delta&0\\0&0\end{bmatrix}.$$

Thus, if $\|\phi^{-1}\|$ is large, then small changes in A, B, or C can induce relatively large changes in X.

In general, given the random nature of roundoff error, we conclude from the analysis in this section that errors of order $u\|\phi^{-1}\|$ can be expected to contaminate the computed solution no matter what algorithm we employ to solve $AX+XB=C$. An estimate of $\|\phi^{-1}\|$ is therefore something to be valued in practice. An expensive though reliable method for doing this would be to compute the singular value decomposition of $P=(I_n\otimes A)+(B^T\otimes I_m)$ using EISPACK [11]. A far cheaper alternative and one which we are currently investigating involves the condition estimator developed in [14].

V. Roundoff Error Analysis of the Hessenberg–Schur Algorithm

We now take a detailed look at the roundoff errors associated with the Hessenberg–Schur algorithm. This amounts to applying some standard results from Wilkinson [12]. His inverse error analysis pertaining to orthogonal matrices can be applied to the computations $H=U^TAU$, $S=V^TB^TV$, $F=U^TCV$, and $X=UYV^T$ while his well-known analysis of Gaussian elimination and back-substitution can used in connection with the determination of Y. (Y is essentially obtained by using Gaussian elimination and back substitution to solve the system $[(I_n\otimes H)+(S\otimes I_m)]\text{vec}(Y)=\text{vec}(F)$.) Denoting computed quantities with the "hat" notation, we can account for the rounding errors in the Hessenberg–Schur algorithm in the following way:

$$\hat{U}=U_1+E_u \qquad U_1^TU_1=I, \quad \|E_u\|_F<\epsilon \tag{5.1}$$
$$\hat{V}=V_1+E_v \qquad V_1^TV_1=I, \quad \|E_v\|_F<\epsilon \tag{5.2}$$
$$\hat{H}=U_1^T(A+E_1)U_1 \qquad \|E_1\|_F<\epsilon\|A\|_F \tag{5.3}$$
$$\hat{S}=V_1^T(B+E_2)^TV_1 \qquad \|E_2\|_F<\epsilon\|B\|_F \tag{5.4}$$
$$\hat{F}=U_1^T(C+E_3)V_1 \qquad \|E_3\|_F<\epsilon\|C\|_F \tag{5.5}$$
$$(\hat{T}+E_4)\hat{y}=\hat{f} \qquad \|E_4\|_2<\epsilon\|\hat{T}\|_2 \tag{5.6}$$
$$\hat{X}=U_1(\hat{Y}+E_5)V_1^T \qquad \|E_5\|_F<\epsilon\|\hat{Y}\|_F \tag{5.7}$$

where

$$\hat{T}=(I_n\otimes H)+(S\otimes I_m) \tag{5.8}$$
$$\hat{y}=\text{vec}(\hat{Y}) \tag{5.9}$$
$$\hat{f}=\text{vec}(\hat{F}) \tag{5.10}$$

and ϵ is a small multiple of the machine precision u. (We have used the 2-norm for convenience. A straightforward error analysis shows that if

$$\|\phi^{-1}\|\epsilon(2+\epsilon)[\|A\|_2+\|B\|_2]<1/2, \tag{5.11}$$

then

$$\frac{\|X-\hat{X}\|_F}{\|X\|_F}<(9\epsilon+2\epsilon^2)\|\phi^{-1}\|[\|A\|_F+\|B\|_F]. \tag{5.12}$$

Inequality (5.12) indicates that errors no worse in magnitude than $0(\|\phi^{-1}\|\epsilon)$ will contaminate the computed $\hat{X}$. Since ϵ is a small multiple of the machine precision u, we see that (5.12) is essentially the same result as (4.8) which was established under the "ideal" assumptions (4.3)–(4.6). Likewise, assumption (5.11) corresponds to assumption (4.7). Conclusion: the roundoff properties of the Hessenberg–Schur algorithm are as good as can be expected from any algorithm designed to solve $AX+XB=C$.

We finish this section with two remarks. First, the entire analysis is applicable to the Bartels–Stewart algorithm. We simply replace (5.3) with

$$\hat{R}=U_1^T(A+E_1)U_1 \qquad \|E_1\|_F<\epsilon\|A\|_F \tag{5.3'}$$

where $\hat{R}$ is now quasi-triangular instead of Hessenberg.

Our second remark concerns another standard by which the quality of $\hat{X}$ can be judged. In some applications, one may be more interested in the norm of the residual $\|A\hat{X}+\hat{X}B-C\|_F$ than the relative error. An analysis similar to that above reveals that if (5.1)–(5.11) hold, then

$$\|A\hat{X}+\hat{X}B-C\|_F \leq (10\epsilon+3\epsilon^2)(\|A\|_F+\|B\|_F)\|X\|_F. \quad (5.13)$$

Notice that the bound does not involve $\|\phi^{-1}\|$.

VI. The Fortran Codes and Their Performance

A collection of Fortran subroutines have been written which implement the Hessenberg–Schur algorithm. Here is a summary of what the chief routines in the package do.

AXXBC—This is the main calling subroutine and the only one which the user "sees." It assumes $m>n$.

ORTHES—This subroutine reduces a matrix to upper Hessenberg form using Householder matrices. All the information pertaining to the reduction is stored below the main diagonal of the reduced matrix.

ORTRAN—This subroutine is used to explicitly form the orthogonal matrix obtained by ORTHES.

HQR2—This subroutine reduces an upper Hessenberg matrix to upper quasi-triangular form using the QR algorithm. (It is an adaptation of the EISPACK routine having the same name [11].)

TRANSF—This subroutine computes products of the form U^TCV and UYV^T where U and V are orthogonal.

NSOLVE, HESOLV, BACKSB—These routines combine to solve upper Hessenberg systems using Gaussian elimination with partial pivoting.

N2SOLV, H2SOLV, BACKSB—These routines combine to solve the $2m$-by-$2m$ block Hessenberg systems encountered whenever S has a 2-by-2 bump.

The above codes are designed to handle double precision A, B, and C and require about 23 000 bytes of storage. This amount of memory is put into perspective with the remark that when a 25×25 problem is solved, the program itself accounts for one-half of the total storage.

To assess the effectiveness of our subroutines we ran two sets of tests. In the first set we compared the execution times for our method and the Bartels–Stewart algorithm. For a given value of n/m, approximately 20 randomly generated examples were run ranging in dimension from 10 to 50. The timing ratios were then averaged. Table I summarizes what we found. Although the predicted savings (second column) are a little greater than those actually obtained (third column), the results certainly confirm the superior efficiency of the Hessenberg–Schur algorithm.

We also compared the accuracy of the two methods on the same class of examples and found them indistinguishable. This is to be expected because the favorable error analysis in the previous section applies to both algorithms.

The second class of test problems was designed to examine the behavior of the algorithm on ill-conditioned $AX+XB=C$ examples. This was accomplished by letting A and B have the form

$$A=\operatorname{diag}(1,2,3,\cdots,m)+N_m$$
$$B=2^{-t}I_n-\operatorname{diag}(n,n-1,\cdots,1)+N_n^T$$

where

$$N_k=\begin{bmatrix} 0 & & & & \\ 1 & 0 & & \bigcirc & \\ 1 & 1 & \ddots & & \\ \vdots & \vdots & & & \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix} k\times k.$$

Notice that there is a coalescence among the eigenvalues of A and $-B$ as t gets large. This enables us to control the sensitivity of the transformation $\phi(X)=AX+XB$. (In particular, it is easy to show that $\|\phi^{-1}\|>2^t$.) To facilitate the checking of errors, C is chosen so that the solution X is the matrix whose entries are each one. Using the same computer and compiler as above ($u=16^{-13}$), we obtained the results for an $m=10$, $n=4$ problem, as shown in Table II.

TABLE I
TIMINGS

n/m	$\dfrac{w_{HS}(m,n)}{w_{BS}(m,n)}$	$\dfrac{\text{HS Execution Time}}{\text{BS Execution Time}}$ (Average)
1.00	.76	.84
.75	.63	.70
.50	.46	.54
.25	.30	.35

All computations were performed using long arithmetic on the IBM 370/168 with the Fortran H compiler, OPT=2.

TABLE II
ERROR AND RESIDUALS

t	$\|\phi^{-1}\|$	$\dfrac{\|X-\hat{X}\|_F}{\|X\|_F}$	$\dfrac{\|A\hat{X}+\hat{X}B-C\|_F}{\|X\|_F[\|A\|_F+\|B\|_F]}$
1	2.3×10^{1}	2.1×10^{-14}	8.2×10^{-16}
10	8.3×10^{3}	5.0×10^{-12}	6.7×10^{-16}
15	2.7×10^{5}	1.4×10^{-10}	8.5×10^{-16}
20	8.3×10^{7}	9.3×10^{-9}	9.3×10^{-16}
25	2.8×10^{8}	1.6×10^{-7}	6.1×10^{-16}
30	9.0×10^{9}	8.6×10^{-6}	8.1×10^{-16}

The quantity $\|\phi^{-1}\|$ is the reciprocal of the smallest singular value of the matrix $P=(I_4\otimes A)+(B^T\otimes I_{10})$ and for this modestly sized problem was found using the subroutine SVD in EISPACK [11].

The results of Table II affirm the key results (5.12) and (5.13). In particular, we see that small residuals are obtained regardless of the norm of ϕ^{-1}. In contrast, the accuracy of $\hat{X}$ deteriorates as $\|\phi^{-1}\|$ becomes large.

We conclude this section with the remark that the Hessenberg–Schur algorithm offers no advantage over the Bartels–Stewart method for the important case when $B=A^T$, i.e., the Lyapunov problem. This is because the latter algorithm requires only one Schur decomposition to solve $AX+XA^T=C$.

VII. Extensions to Other Matrix Equation Problems

In this final section we indicate how the Hessenberg–Schur "idea" can be applied to two other matrix equation problems. Consider first the problem

$$AXM+X=C \quad (7.1)$$

where $A\in R^{m\times m}$, $M\in R^{n\times n}$, $C\in R^{m\times n}$, and $X\in R^{m\times n}$. If

$$U^TAU=H \quad U^TU=I, \qquad H \text{ upper Hessenberg}$$

and

$$V^TM^TV=S \quad V^TV=I, \qquad S \text{ quasi-upper triangular}$$

and $F=U^TCV$, then (7.1) transforms to

$$HYS^T+Y=F \quad (7.2)$$

where $Y=U^TXV$. As in the Hessenberg–Schur algorithm, once $y_{k+1},\cdots,y_n$ are known, y_k can be found by solving a Hessenberg system. (Recall y_k is the kth column of Y.) To see how, assume $s_{k,k-1}=0$ and equate kth columns in (7.2):

$$H\left(\sum_{j=k}^{n} s_{kj}y_j\right)+y_k=f_k.$$

Hence, y_k can be found by solving

$$(s_{kk}H+I)y_k=\left[f_k-H\sum_{j=k+1}^{n}s_{kj}y_j\right].$$

The presence of 2×2 bumps on the diagonal of T can be handled in a fashion similar to what is done in the Hessenberg–Schur method.

This algorithm which we have sketched should be 30–70 percent faster than the Bartels–Stewart type technique in which both A and M are reduced to triangular form via the QR algorithm. (See [5].)

The second matrix equation problem we wish to consider involves finding $X\in R^{m\times n}$ such that

$$AXM+LXB=C \tag{7.3}$$

where $A,L\in R^{m\times m}$, $M,B\in R^{n\times n}$, and $C\in R^{m\times n}$. For a discussion of these and more general problems, see [7] and [13]. If M and L are nonsingular, then (7.3) can be put into "standard" $AX+XB=C$ form,

$$(L^{-1}A)X+X(BM^{-1})=L^{-1}CM^{-1}.$$

If M and/or L is poorly conditioned, it may make more numerical sense to apply the QZ algorithm of Moler and Stewart [8] to effect a stable transformation of (7.3). In particular, their techniques allow us to compute orthogonal U, V, Q, and Z such that

$$\begin{aligned}Q^TAU&=P \quad &&\text{(quasi-upper triangular)}\\ Q^TLU&=R &&\text{(upper triangular)}\\ Z^TB^TV&=S &&\text{(quasi-upper triangular)}\\ Z^TM^TV&=T &&\text{(upper triangular).}\end{aligned}$$

If $Y=U^TXV$ and $F=Q^TCZ$, then (7.3) transforms to

$$PYT^T+RYS^T=F.$$

Comparing kth columns and assuming $s_{k,k-1}=T_{k,k-1}=0$ we find

$$P\sum_{j=k}^{n}t_{kj}y_j+R\sum_{j=k}^{n}s_{kj}y_j=f_k$$

and so

$$(t_{kk}P+s_{kk}R)y_k=f_k-P\sum_{j=k+1}^{n}t_{kj}y_j-R\sum_{j=k+1}^{n}s_{kj}y_j \tag{7.4}$$

This quasi-triangular system can then be solved for y_k once the right-hand side is known and under the assumption that the matrix $(t_{kk}P+s_{kk}R)$ is nonsingular. (Note that T, P, S, and R can all be singular without $t_{kk}P+s_{kk}R$ being singular.)

Now, as in the Hessenberg–Schur algorithm, significant economies can be made if A is only reduced to Hessenberg form. This is easily accomplished for when applied to the matrix pair (A,L), the QZ algorithm first computes orthogonal Q and U such that $Q^TAU=H$ is upper Hessenberg and $Q^TLU=R$ is upper triangular. The systems in (7.4) are now Hessenberg form and can consequently be solved very quickly. Again, we leave it to the reader to verify that the presence of 2×2 bumps on the diagonal of S pose no serious difficulties.

VIII. Conclusions

We have presented a new algorithm for solving the matrix equation $AX+XB=C$. The technique relies upon orthogonal matrix transformations and is not only extremely stable, but considerably faster than its nearest competitor, the Bartels–Stewart algorithm. We have included perturbation and roundoff analyses for the purpose of justifying the favorable performance of our method. Although these analyses are quite tedious, they are critical to the development of reliable software for this important computational problem.

References

[1] R. H. Bartels and G. W. Stewart, "A solution of the equation $AX+XB=C$," *Commun. ACM*, vol. 15, pp. 820–826, 1972.
[2] P. R. Belanger and T. P. McGillivray, "Computational experience with the solution of the matrix Lyapunov equations," *IEEE Trans. Automat. Contr.*, vol. AC-21, pp. 799–800, 1976.
[3] G. E. Forsythe and C. B. Moler, *Computer Solution of Linear Algebraic Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1967.
[4] P. Hagander, "Numerical solution of $A^TS+SA+Q=0$," *Inform. Sci.*, vol. 4, pp. 35–40, 1972.
[5] G. Kitagawa, "An algorithm for solving the matrix equation $X=FXF^T+S$," *Int. J. Contr.*, vol. 25, pp. 745–753, 1977.
[6] G. Kreisselmeier, "A solution of the bilinear matrix equation $AY+YB=-Q$," *SIAM J. Appl. Math.*, vol. 23, pp. 334–338, 1973.
[7] P. Lancaster, "Explicit solutions of linear matrix equations," *SIAM Rev.*, vol. 12, pp. 544–566, 1970.
[8] C. B. Moler and G. W. Stewart, "An algorithm for generalized matrix eigenvalue problems," *SIAM J. Numer. Anal.*, vol. 10, pp. 241–256, 1973.
[9] B. P. Molinari, "Algebraic solution of matrix linear equations in control theory," *Proc. Inst. Elec. Eng.*, vol. 116, pp. 1748–1754, 1969.
[10] D. Rothschild and A. Jameson, "Comparison of four numerical algorithms for solving the Lyapunov matrix equation," *Int. J. Contr.*, vol. 11, pp. 181–198, 1970.
[11] B. T. Smith *et al.*, *Matrix Eigensystem Routines—EISPACK Guide* (Lecture Notes in Computer Science). New York: Springer-Verlag, 1970.
[12] J. H. Wilkinson, *The Algebraic Eigenvalue Problem.* Oxford, England: Oxford Univ. Press, 1965.
[13] H. Wimmer and A. D. Ziebur, "Solving the matrix equation $\sum_{p=1}^{r} f_p(A)Xg_p(B)=C^*$," *SIAM Rev.*, vol. 14, pp. 318–323, 1972.
[14] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson, "An estimate for the condition number of a matrix," *SIAM J. Numer. Anal.*, vol. 16, pp. 368–375, 1979.