A HESSENBERG-SCHUR METHOD
FOR THE PROBLEM AX + XB = C

by

Gene Golub
Stephen Nash
Charles Van Loan

TR78-354

Department of Computer Science
Cornell University
Ithaca, NY 14853
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.

Authors' Addresses:

Professor Gene Golub
Department of Computer Science
Serra House, Stanford University
Stanford, California 94305

Mr. Stephen Nash
Department of Computer Science
Serra House, Stanford University
Stanford, California 94305

Professor Charles Van Loan
Department of Computer Science
Upson Hall, Cornell University
Ithaca, New York 14853
1. Introduction

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

$$\phi(X) = AX + XB$$

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

$$\phi(X) = AX + XB = C$$

arise in many problem areas and numerous algorithms have been proposed \cite{4,10}. Among them, the Bartels-Stewart algorithm \cite{1} has enjoyed considerable success \cite{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

$$(U^{-1}AU)(U^{-1}XV) + (U^{-1}XV)(U^{-1}HV) = U^{-1}CV$$
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 pivoting is used for this purpose and that the computations are performed on a computer whose floating point numbers have $t$ base 2 digits in the mantissa. Using the standard Wilkinson inverse error analysis [3,12] it follows that relative errors of order $u \left( r_2(U) + r_2(V) \right)$ can be expected to contaminate the computed solution $\hat{X}$ where

$$u = 2^{-t}$$

is the machine precision and $r_2(\cdot)$ is defined by
\[ \kappa_2(W) = \| W \|_2 \| W^{-1} \|_2 = \max_{x \neq 0} \sqrt{\frac{(Wx)^T(Wx)}{x^T x}} = \max_{y \neq 0} \sqrt{\frac{y^T}{(Wy)}} \]

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

\[
U^{-1}AU = \text{diag}(\alpha_1, \alpha_2, \ldots, \alpha_m) = A_1
\]

\[
V^{-1}BV = \text{diag}(\beta_1, \beta_2, \ldots, \beta_m) = B_1
\]

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

\[
A = \begin{bmatrix} 1.234567891 & 3.515985621 \\ 0 & 1.234078268 \end{bmatrix} \quad B = \begin{bmatrix} .3458968425 \\ .6521859685 \\ .3450509462 \end{bmatrix}
\]

\[
C = \begin{bmatrix} 5.748636323 \\ 2.232161079 \end{bmatrix} \quad D = \begin{bmatrix} 5.095604458 \\ 1.579129214 \end{bmatrix}
\]

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, \( u[ \kappa_2(U) + \kappa_2(V)] \approx 10^{-3} \) 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. (c.f. [6,9].)

This leads us to consider transformation methods which rely on orthogonal \( U \) and \( V \). (Recall that \( \bar{u}^T\bar{u} = I \) 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 3 and its roundoff properties are shown to be very desirable in Sections 4 and 5. Our claims regarding speed and accuracy are substantiated in Section 6 where we report on several numerical tests. Finally, we conclude by showing how the techniques in this paper can be extended to other matrix equation problems.
2. The Bartels-Stewart Algorithm

The crux of the Bartels-Stewart algorithm [1] is the computation of the real Schur decompositions

\[ R = U^T A U \quad R \text{ quasi-upper triangular, } U \text{ orthogonal} \]
\[ S = V^T B^T V \quad S \text{ quasi-upper triangular, } V \text{ orthogonal} \]

A matrix is quasi-upper triangular if it is upper triangular with the possible exception of 2x2 "bumps" on the diagonal, etc.

\[
\begin{bmatrix}
    x & x & x & x & x \\
    0 & x & x & x & 0 \\
    0 & x & x & x & 0 \\
    0 & 0 & 0 & x & 0 \\
    0 & 0 & 0 & 0 & x \\
\end{bmatrix}
\]

The 2x2 bumps, if they exist, correspond to complex conjugate eigenvalue pairs. (It is desirable to avoid complex arithmetic when solving a real \( AX + XB = C \) problem.) The above quasi-triangular forms may be found through application of the well-known QR algorithm [12].

From our remarks in Section 1, the reductions (2.1) lead to a system of the form

\[ RY + YS^T = F \quad (F = U^T C V, \ Y = U^T X V) \]

which can readily be solved for \( Y \). To see how, partition \( Y \) and
$P$ into their respective columns

$$Y = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix}, \quad P = \begin{bmatrix} f_1 & f_2 & \cdots & f_n \end{bmatrix}$$

and assume that we have reached a stage where $y_{k+1}, \ldots, y_n$ are known.

If $s_{k,k-1} = 0$, then by comparing the $k$-th columns in (2.2) we find

$$\begin{equation}
(R + s_{kk})y_k = f_k - \sum_{j=k+1}^{n} s_{kj} y_j
\end{equation}$$

This, $y_k$ is the solution of an $m \times m$ quasi-triangular system of equations. This system requires $m^2/2$ operations to evaluate once the right hand side is computed. (In all our operation counts, we tabulate only multiplicative operations and ignore low order terms as is traditional.)

If $s_{k,k-1}$ is nonzero, then by equating columns $k-1$ and $k$ in (2.2) we obtain

$$\begin{equation}
P[y_{k-1} \ y_k] + [y_{k-1} \ y_k] \begin{bmatrix}
    s_{k-1,k-1} & s_{k,k-1} \\
    s_{k-1,k} & s_{kk}
\end{bmatrix}
\begin{bmatrix}
    y_{k-1} \\
    y_k
\end{bmatrix} = \begin{bmatrix}
f_{k-1} \\
    f_k
\end{bmatrix} - \sum_{j=k+1}^{n} s_{k-1,j} y_j - s_{kj} y_j
\end{equation}$$

By expressing this linear system in standard matrix-vector form we can solve for the components of $y_{k-1}^T = (y_{1,k-1}, \ldots, y_{m,k-1})$ and $y_k^T = (y_{1,k}, \ldots, y_{m,k})$. In particular, we obtain the 2m-by-2m
system

\[
\begin{bmatrix}
  p_{11} & p_{12} & \ldots & p_{1m} \\
p_{21} & p_{22} & \ldots & p_{2m} \\
  \vdots & \vdots & \ddots & \vdots \\
p_{m1} & p_{m2} & \ldots & p_{mm}
\end{bmatrix}
\begin{bmatrix}
z_1 \\
z_2 \\
  \vdots \\
z_m
\end{bmatrix} =
\begin{bmatrix}
d_1 \\
d_2 \\
  \vdots \\
d_m
\end{bmatrix}
\]

(2.5)

where

\[
p_{ij} =
\begin{bmatrix}
  r_{ij} & 0 \\
  0 & r_{ij}
\end{bmatrix}
\quad (i \neq j)
\]

(2.6)

\[
\begin{bmatrix}
  r_{ii} + s_{k-1,k-1} & s_{k-1,k} \\
  s_{k,k-1} & r_{ii} + s_{kk}
\end{bmatrix}
\quad (i = j)
\]

and for \( i = 1, \ldots, m \)

\[
z_i = \begin{bmatrix}
y_{i,k-1} \\
y_{i,k}
\end{bmatrix}
\quad \text{and} \quad d_i = \begin{bmatrix}
q_i \\
w_i
\end{bmatrix}
\]

Since \( R = (r_{ij}) \) is quasi-upper triangular, the above 2m x 2m matrix is upper triangular with the possible exception of 4x4 bumps on the diagonal. Using Gaussian elimination with partial pivoting to solve a linear system with this structure and dimension requires \( m^2 \) operations once the right hand side is evaluated. Aside from a few scalar variables, no additional storage is required beyond what is needed in those “nice” situations when \( S \) is actually upper triangular and therefore free of 2x2 bumps.
In summary, the Bartels-Stewart algorithm involves the following steps and work counts:

1. Compute the real Schur decompositions:
   \[ R = U^T A U \] (save U) \hspace{1cm} 10 m^3
   \[ S^T = V^T B V \] (save V) \hspace{1cm} 10 n^3

2. Update the righthand side:
   \[ F = U^T C V \] \hspace{1cm} m^2 n + m n^2

3. Back substitute for \( Y \):
   \[ R Y + Y S^T = F \] \hspace{1cm} (m^2 n + m n^2)/2

4. Obtain solution:
   \[ X = U Y V^T \] \hspace{1cm} m^2 n + m n^2

\[ w_{BS}(m,n) = 10m^3 + 10n^3 + \frac{5}{2} (m^2 n + mn^2) \]

The work counts associated with the Schur forms are estimates based upon experience with the QR algorithm which normally requires between one and two iterations per eigenvalue [11,12].

In terms of storage, the algorithm needs five arrays for the following purposes:

- \( A \) (m×m) for the original \( A \) and subsequently \( R \)
- \( U \) (m×m) for the orthogonal matrix \( U \)
- \( B \) (n×n) for the original \( B \) and subsequently \( S \)
- \( V \) (n×n) for the orthogonal matrix \( V \)
- \( C \) (r×n) for the original \( C \) and subsequently \( Y \) and \( X \)
3. 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

\[ H = U^T A U \quad \text{H upper Hessenberg, } U \text{ orthogonal} \]
\[ S = V^T B V \quad \text{S quasi-upper triangular, } V \text{ orthogonal} \]

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 \( \frac{4}{3} n^3 \) operations.
See [12, p. 347] for a description of this algorithm. The reductions
(3.1) lead to a system of the form

\[ HY + YS^T = F \]

which may be solved in a manner similar to what is done in the Bar-
tels-Stewart algorithm. In particular, assume that \( y_{k+1}, \ldots, y_n \)
have been computed.

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

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

When Gaussian elimination with partial pivoting is used for this
purpose, \( m^2 \) operations are needed once the righthand side is known.
If $s_{k,k-1}$ is nonzero, then by equating columns $k-1$ and $k$ in (3.2) we find

$$H[y_{k-1} | y_k] + [y_{k-1} | y_k] \begin{bmatrix} s_{k-1,k-1} & s_{k,k-1} \\ s_{k-1,k} & s_{kk} \end{bmatrix}$$

$$= \{f_{k-1} | f_k\} - \frac{1}{n} \sum_{j=k+1}^n [s_{k-1,j} y_j | s_{kj} y_j] \cong \{g | w\}$$

This leads to the $2m \times 2m$ system (2.5) where the $P_{ij}$ are defined by

$$\begin{bmatrix} h_{ij} & 0 \\ 0 & h_{ij} \end{bmatrix} \quad (i \neq j)$$

$$(i = j)$$

$$P_{ij} = \begin{bmatrix} h_{ii} + s_{k-1,k-1} & s_{k-1,k} \\ s_{k,k-1} & h_{ii} + s_{kk} \end{bmatrix}$$

Since $H$ is upper Hessenberg, it follows that (2.5) is an upper triangular system with two nonzero subdiagonals. Using Gaussian elimination with partial pivoting, this system can be solved in $6m^2$ operations once the righthand side is known. Unfortunately, a $2m^2$ workspace is required to carry out the computations.
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 of $H$ [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 counts we have:

1. Reduce $A$ to upper Hessenberg and $H^T$ to quasi-upper triangular:
   
   $$ H = U^T A U \quad \text{(store } U \text{ in factored form)}$$
   
   $$ S = V^T B V \quad \text{(save } V)$$

   \[\frac{5}{3} m^3 \quad \frac{10}{3} n^3\]

2. Update the righthand side:

   $$ F = U^T C V$$

   \[m^2 n + mn^2\]

3. Back substitute for $Y$:

   $$ H Y + Y S^T = F$$

   \[3m^2 n + \frac{1}{2} mn^2\]

4. Obtain solution:

   $$ X = U Y V^T$$

   \[m^2 n + mn^2\]

\[\omega_{HS}(m,n) = \frac{5}{3} m^3 + 10n^3 + 5m^2 n + \frac{5}{2} mn^2\]

To obtain the operation count associated with the determination of $Y$, we assumed that $S$ has $\frac{n}{2}$ 2x2 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

$$ B^T x^T + x^T A^T = C^T $$

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

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

which indicates that substantial savings accrue when the Hessenberg-Schur method is favored. For example, if $m = 4n$, then $w_{HS}(m,n) = .30 w_{BS}(m,n)$.

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 $(n \times n)$ for the original $C$ and subsequently $Y$ and $X$
- W $(2m^2)$ for handling the possible system (2.5), (3.4)
4. 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 insure 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

\[
(4.1) \quad P x = c
\]

where

\[
(4.2) \quad P = (I_n \otimes A) + (B^T \otimes I_m)
\]

and

\[
x = \text{vec}(X) = (x_{11}, x_{21}, \ldots, x_{m1}, x_{12}, x_{22}, \ldots, x_{m2}, \ldots, x_{1n}, \ldots, x_{mn})^T
\]
\[
c = \text{vec}(C) = (c_{11}, c_{21}, \ldots, c_{m1}, c_{12}, c_{22}, \ldots, c_{m2}, \ldots, c_{1n}, \ldots, c_{mn})^T
\]

Here, the Kronecker product \( W \otimes Z \) of two matrices \( W \) and \( Z \) is the block 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 $\phi$, we need to define a norm on the space of linear transformations from $\mathbb{R}^{m \times n}$ to $\mathbb{R}^{m \times n}$:

$$
f: \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}, \quad \| f \|_2 = \max_{X \in \mathbb{R}^{m \times n}} \frac{\| f(X) \|_F}{\| X \|_F}
$$

Here, the Frobenius norm $\| \cdot \|_F$ is defined by $\| W \|_F^2 = \sum_{i,j} |w_{ij}|^2$.

Notice that for the linear transformation $\phi$ defined by (1.1) we have

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

where $P$ is defined by (4.2). If $\phi$ is nonsingular, then

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

Now consider solving $AX + XB = C$ on a computer having machine precision $u$. In general, rounding errors of order $u \| A \|_F \| B \|_F$ 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
(4.3) \[(A + E)\hat{x} + \hat{x}(B + F) = (C + G)\]

where

(4.4) \[\| E \|_F \leq u \| A \|_F\]

(4.5) \[\| F \|_F \leq u \| B \|_F\]

(4.6) \[\| G \|_F \leq u \| C \|_F\]

How accurate would such an \(\hat{x}\) be? To answer this question, we first establish a simple result concerning perturbed linear systems.

**Lemma**

Let \(M\) and \(\Delta M\) be \(k \times k\) matrices and let \(x\), \(\Delta x\), \(d\), and \(\Delta d\), be \(k\)-vectors. Assume that \(M\) is nonsingular and \(d\) nonzero. If

(4.7) \[Mx = d\]

(4.8) \[(M + \Delta M)(x + \Delta x) = (d + \Delta d)\]

and

(4.9) \[\| M^{-1} \|_2 \| \Delta M \|_2 \leq 1/2\]

then

(4.10) \[\frac{\| \Delta x \|_2}{\| x \|_2} \leq 2 \| M^{-1} \|_2 \left( \| \Delta M \|_2 + \frac{\| \Delta d \|_2}{\| x \|_2} \right)\]
Proof

Applying $M^{-1}$ to (4.8), rearranging, and taking norms we find

$$\|\Delta x\|_2 \leq \|M^{-1}\|_2 \cdot \|\Delta M\|_2 \left(\|x\|_2 + \|\Delta x\|_2\right) + \|M^{-1}\|_2 \cdot \|\Delta d\|_2$$

The Lemma now follows by using (4.9).

We now apply this result to the perturbed linear system given in (4.3).

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, $C$ nonzero, and

$$u [\|A\|_F + \|B\|_F] \cdot \|\phi^{-1}\| \leq 1/2,$$

then

$$\|X - \hat{X}\|_F \leq 4 u [\|A\|_F + \|B\|_F] \cdot \|\phi^{-1}\|\|X\|_F$$

Proof

Defining $x = \text{vec}(X)$, $\hat{x} = \text{vec}(\hat{X})$, $c = \text{vec}(C)$, $g = \text{vec}(G)$, $P = (I_n \otimes A) + (B^T \otimes I_m)$, and $\Delta P = (I_n \otimes E) + (F^T \otimes I_m)$, we find
\( P x = c \)

\[(P + LI) \hat{x} = c + g\]

\[\| \hat{x} \|_2 \leq \| P \|_2 + \| A \|_2 + \| B \|_2 < \| A \|_F + \| B \|_F\]

and

\[\| LP \|_2 < u \{ \| A \|_F + \| B \|_F \}\]

Since \( \| \phi^{-1} \| = \| P^{-1} \|_2 \) we have

\[\| P^{-1} \|_2 \| AP \|_2 < \| \phi^{-1} \|_{\| u \{ \| A \|_F + \| B \|_F \} \|} = 1/2\]

The above lemma can now be applied and with a little manipulation we get

\[\frac{\| x - \hat{x} \|_F}{\| x \|_F} = \frac{\| x - \hat{x} \|_2}{\| x \|_2} < 2 \| \phi^{-1} \| \{ u(\| A \|_F + \| B \|_F)\} \frac{\| x \|_2}{2}\]

The theorem follows since

\[\| g \|_2 < u \| C \|_F < u \{ \| A \|_F + \| B \|_F \} \| x \|_2\]

For the 2x2 example given in Section 1, the upper bound in (4.12) 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 (4.12) can roughly be attained. Setting

\[ A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 6-1 & 0 \\ 1 & 3 \end{bmatrix}, \quad C = \begin{bmatrix} 2+\delta & 5 \\ 1+\delta & 4 \end{bmatrix} \]

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

\[ X = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}. \]

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

\[ AX + XB = C + \begin{bmatrix} u & 0 \\ 0 & 0 \end{bmatrix} \]

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 \( u \cdot \delta^{-1} \) can be expected to contaminate the computed solution no matter what algorithm we employ to solve \( AX + XB = C \).
5. 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 computation \( H = U^T A U \), \( S = V^T B^T V \), \( F = U^T C V \), and \( X = U Y V^T \) while his well-known analysis of Gaussian elimination and back-substitution can be 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:

\[
\begin{align*}
(5.1) \quad & \hat{U} = U_1 + E_u \quad U_1^T U_1 = I, \quad \| E_u \|_F \leq \varepsilon \\
(5.2) \quad & \hat{V} = V_1 + E_v \quad V_1^T V_1 = I, \quad \| E_v \|_F \leq \varepsilon \\
(5.3) \quad & \hat{H} = U_1^T (A + E_1) U_1 \quad \| E_1 \|_F \leq \varepsilon \quad \| A \|_F \\
(5.4) \quad & \hat{S} = V_1^T (B + E_2) V_1 \quad \| E_2 \|_F \leq \varepsilon \quad \| B \|_F \\
(5.5) \quad & \hat{F} = U_1^T (C + E_3) V_1 \quad \| E_3 \|_F \leq \varepsilon \quad \| C \|_F \\
(5.6) \quad & (\hat{T} + E_4) \hat{y} = \hat{t} \quad \| E_4 \|_2 \leq \varepsilon \quad \| \hat{T} \|_2 \\
(5.7) \quad & \hat{X} = U_1 (\hat{Y} + E_5) V_1^T \quad \| E_5 \|_F \leq \varepsilon \quad \| \hat{Y} \|_F \\
\end{align*}
\]

where
\[(5.8) \quad \hat{T} = (I_n \otimes H) + (S \otimes I_m) \]
\[(5.9) \quad \hat{y} = \text{vec}(\hat{y}) \]
\[(5.10) \quad \hat{f} = \text{vec}(\hat{f}) \]

and \(c\) is a small multiple of the machine precision \(u\). (We have used the 2-norm in (5.6) for convenience.) We now proceed to bound the relative error in \(\hat{X}\).

Defining \(W = U_1^T X V_1\), it follows from (5.07) and the orthogonal invariance of the Frobenius norm that

\[(5.11) \quad \frac{\| X - \hat{X} \|_F}{\| X \|_F} \leq \frac{\| W - \hat{Y} \|_F}{\| W \|_F} + c \frac{\| \hat{Y} \|_F}{\| W \|_F} \]

Since \(W\) solves \((U_1^T A U_1) Z + Z (V_1^T B V_1) = U_1^T C V_1\), we have

\[(5.12) \quad T W = f \]

where \(w = \text{vec}(W)\), \(f = \text{vec}(U_1^T C V_1)\) and \(T = (I_n \otimes U_1^T A U_1) + (V_1^T B V_1 \otimes I_m)\)

Notice from (5.6), (5.8), (5.9) and (5.10) that \(\hat{y}\) satisfies a perturbed version of this system. In particular,

\[(5.13) \quad (T + \Delta T) \hat{y} = f + \Delta f \]

where \(\Delta T = (\hat{T} - T) + E_4\) and \(\Delta f = \text{vec}(U_1^T E_3 V_1)\). Manipulation with (5.3), (5.4) and (5.6) shows
\[ \| \Delta T \|_2 \leq c (2 + c) \| A \|_F + \| B \|_F \]

and with (5.5) and (5.10) that

\[ \| \Delta \|_2 \leq c \| C \|_F \leq c (\| A \|_F + \| B \|_F) \| W \|_F \]

Now \( \| \phi^{-1} \| = \| T^{-1} \|_2 \) and if we make the assumption

\[ (5.14) \quad \| \phi^{-1} \| \leq c (2 + c) \| A \|_2 + \| B \|_2 \]

then the Lemma is applicable and we find

\[ \frac{\| W - \hat{y} \|_F}{\| W \|_F} = \frac{\| W - \hat{y} \|_2}{\| W \|_2} \leq (6c + 2c^2) \| \phi^{-1} \| \| A \|_F + \| B \|_F \]

Furthermore, (5.14) can also be used to show

\[ \| \hat{y} \|_F \leq 3 \| W \|_F \]

and therefore from (5.11) we have

\[ (5.15) \quad \frac{\| X - \hat{X} \|_F}{\| X \|_F} \leq (9c + 2c^2) \| \phi^{-1} \| \| A \|_F + \| B \|_F \]
Inequality (5.15) indicates that errors no worse in magnitude than $O\left( \| \delta^{-1} \| \epsilon \right)$ will contaminate the computed $\hat{X}$. Since $\epsilon$ is a small multiple of the machine precision $u$, we see that (5.15) is essentially the same result as (4.12) which was established under the "ideal" assumptions (4.3)-(4.6). Likewise, assumption (5.14) corresponds to assumption (4.11). 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

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

where $R$ 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)-(5.10) and (5.14) hold, then

$$\tag{5.16} \\| A\hat{X} + \hat{X}B - C \|_F < (10\epsilon + 3\epsilon^2) (\| A \|_F + \| B \|_F ) \| X \|_F$$

Notice that the bound does not involve $\| \delta^{-1} \|$.
6. 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.

**TRANSF** - This subroutine computes products of the form \( U^T CV \) and \( UYV^T \) where \( U \) and \( V \) are orthogonal.

**NSOLVE**

**HESOLV** - These routines combine to solve upper Hessenberg systems using Gaussian elimination with partial pivoting.

**BACKSB**

**N2SOLV**

**H2SOLV** - These routines combine to solve the 2m-by-2m block Hessenberg systems encountered whenever \( S \) has a 2-by-2 bump.

**BACKSB**

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 25x25 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 \( n/m \), many examples were run ranging in dimension from 10 to 50. The timing ratios were then averaged. The following table summarizes what we found:

<table>
<thead>
<tr>
<th>n/m</th>
<th>( \frac{w_{HS}(m,n)}{w_{BS}(m,n)} )</th>
<th>HS Execution time (average)</th>
<th>BS Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>.76</td>
<td>.84</td>
<td></td>
</tr>
<tr>
<td>.75</td>
<td>.63</td>
<td>.70</td>
<td></td>
</tr>
<tr>
<td>.50</td>
<td>.46</td>
<td>.54</td>
<td></td>
</tr>
<tr>
<td>.25</td>
<td>.30</td>
<td>.35</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 1. Timings**

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
\[ A = \text{diag}(1,2,3,\ldots,m) + N_m \]
\[ B = 2^{-t} I_n - \text{diag}(n,n-1,\ldots,1) + N_n^\top \]

where

\[
N_k = \begin{bmatrix}
0 \\
1 & 0 \\
1 & 1 & 0 \\
\vdots & \ddots & \ddots \\
1 & 1 & 1 & \ldots & 1 & 0
\end{bmatrix}_{k \times k}
\]

Notice that there is 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 an IBM 370/168 with extended precision \( u = 16^{-14} \), we obtained the following results for an \( m = 10 \), \( n = 4 \) problem:

<table>
<thead>
<tr>
<th>( t )</th>
<th>( | \phi^{-1} | )</th>
<th>( | X - \hat{X} |_F / | X |_F )</th>
<th>( | \hat{X} A \hat{X} + \hat{X} B - C |_F / | X |_F (| A |_F + | B |_F ) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 2.3 \times 10^1 )</td>
<td>( 2.1 \times 10^{-14} )</td>
<td>( 8.2 \times 10^{-16} )</td>
</tr>
<tr>
<td>10</td>
<td>( 8.3 \times 10^3 )</td>
<td>( 5.0 \times 10^{-12} )</td>
<td>( 6.7 \times 10^{-16} )</td>
</tr>
<tr>
<td>15</td>
<td>( 2.7 \times 10^5 )</td>
<td>( 1.4 \times 10^{-10} )</td>
<td>( 8.5 \times 10^{-16} )</td>
</tr>
<tr>
<td>20</td>
<td>( 8.3 \times 10^7 )</td>
<td>( 9.3 \times 10^{-9} )</td>
<td>( 9.3 \times 10^{-16} )</td>
</tr>
<tr>
<td>25</td>
<td>( 2.8 \times 10^8 )</td>
<td>( 1.6 \times 10^{-7} )</td>
<td>( 6.1 \times 10^{-16} )</td>
</tr>
<tr>
<td>30</td>
<td>( 9.0 \times 10^9 )</td>
<td>( 8.6 \times 10^{-6} )</td>
<td>( 8.1 \times 10^{-16} )</td>
</tr>
</tbody>
</table>

**Table 2. Errors and Residuals**
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 can be found by using the subroutine SVD which is in EISPACK.

The results of Table 2 affirm the key results (5.15) and (5.16). In particular, we see that small residuals are obtained regardless of the norm of $\phi^{-1}$. 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 $AX + XA^T = C$. 
7. 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

(7.1) \[ AXM + X = C \]

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

\[ U^T A U = H \quad U^T U = I, \quad H \text{ upper Hessenberg} \]

and

\[ V^T M^T V = S \quad V^T V = I, \quad S \text{ quasi-upper triangular} \]

and \( F = U^T C V \), then (7.1) transforms to

(7.2) \[ HYS^T + Y = F \]

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

\[ H(\sum_{j=k}^{n} s_{kj}y_j) + y_k = f_k \]

Hence, \( y_k \) can be found by solving

\[ (s_{kk}H + I)y_k = [f_k - H\sum_{j=k+1}^{n} s_{kj}y_j] \]
The presence of 2x2 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: $X \in \mathbb{R}^{m \times n}$ such that

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

where $A, L \in \mathbb{R}^{m \times m}$, $M, B \in \mathbb{R}^{n \times n}$, and $C \in \mathbb{R}^{m \times n}$. 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

$$Q^T A U = P \quad \text{(quasi-upper triangular)}$$
$$Q^T L U = R \quad \text{(upper triangular)}$$
$$Z^T B T V = S \quad \text{(quasi-upper triangular)}$$
$$Z^T M^T V = T \quad \text{(upper triangular)}$$
If $Y = U^T X V$ and $P = Q^T C Z$, then (7.3) transforms to

$$PY^T + RYS^T = P$$

Comparing $k$-th 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

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

This quasi-triangular system can then be solved for $y_k$ once the righthand side is known and under the assumption that the matrix $(t_{kk} P + s_{kk} R)$ is nonsingular. (Note that $T$, $P$, $C$, 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 $(A, Z)$, the QZ algorithm first computes orthogonal $Q$ and $U$ such that $Q^T A U = H$ is upper Hessenberg and $Q^T L U = R$ 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 $2 \times 2$ burps on the diagonal of $S$ pose no serious difficulties.
3. 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 may appear boring, they are critical to the development of reliable software for this important computational problem.
REFERENCES


SUBROUTINE AXRC(NDIM,N,MDIM,M,A,V,IPR(1),J,M,NDIM,1)
INTEGER I,ISERR,INC,IPR(1),J,M,NDIM
REAL*8 A(MDIM,M),P(NDIM,N),V(NDIM,\n  C(MDIM,N),ORT(1),EPS,REPS,D(1)

THIS ROUTINE SOLVES THE MATRIX EQUATION

    A * X = M * N MATRIX
    P * N * N MATRIX
    C * M * N MATRIX.

THE PARAMETERS FOR THIS SUBROUTINE ARE:
NDIM - DECLARED ROW DIMENSION OF THE
N - ROW DIMENSION OF B; COLUMN
MDIM - DECLARED ROW DIMENSION OF A
M - ROW DIMENSION OF A, C, AND D
A - M * M MATRIX (DOUBLE PRECISION)
B - N * N MATRIX (DOUBLE PRECISION)
V - N * N MATRIX (DOUBLE PRECISION)
C - M * N MATRIX WHICH HOLDS C
ORT - DOUBLE PRECISION VECTOR USE
D - DOUBLE PRECISION VECTOR FOR
L - INTEGER VECTOR FOR INTERNAL
EPS - ERROR TOLERANCE. EPS SHOULD
IERR - INTEGER VARIABLE USED TO SE
   IERR=0 => NORMAL RETURN
   IERR=J => J-TH EIGENVALUE CON
   IERR=-J => A SINGULAR MATRIX

METHOD:

FIRST OF ALL, A(TRANSPOSE) IS FORM
P THROUGHOUT THE PROGRAM. (IN THE
USED TO REPRESENT A(TRANSPOSE)).
B(I) IS TRANSFORMED TO REAL UPPER
HESSENBerg FORM USING ORTHOGONAL T
THE RIGHT HAND SIDE C IS MULTIPLIED
MATRICES AND THE SOLUTION OF THIS
COMPUTED. THIS SOLUTION IS THEN M
FORMATION MATRICES IN ORDER TO OBT
ORIGINAL PROBLEM.

NOTE:

IN ORDER TO GET OPTIMAL EFFICIENCY
BE <= M. IF THIS IS NOT TRUE FOR
EQUATION AND SOLVE THE RESULTING A
THE ROUTINE AX + XB = C WHERE

ARE:

THE MATRIX A
DIMENSION OF B AND C
C, AND V
V; COLUMN DIMENSION OF A AND V
ICN)
(ICI
ICN) USED TO STORE THE MATRIX
SCHUR FORM
ICN INPUT AND X ON OUTPUT
C BY SUBROUTINE RG

INTERNAL USE OF LENGTH

USE OF LENGTH AT LEAST 4N
= EQUAL THE SMALLEST NUMBER
> 1.

ERROR CODES.

IF N HAS NOT BEEN DETERMINED
ATIONS.
X WAS ENCOUNTERED WHEN
J-TH COLUMN OF X.

AND IS USED INSTEAD OF
FOLLOWING DISCUSSION, A(T) IS
SCHUR FORM AND A TO UPPER
TRANSFORMATIONS.
BY THESE TRANSFORMATION
TRANSFORMED SYSTEM IS
MULTIPLIED BY THE TRANS-
AIN THE SOLUTION TO THE

FROM THIS ROUTINE. N SHOULD
YOUR PROBLEM. TRANSPOSE THE
PROBLEM.
FORM R(TRANSPOSE) FOR INTERNAL USE

DO 5 I = 1,N
    DO 5 J = 1,N
        D(I) = A(I,I)
        P(I,J) = B(J,I)
        B(J,I) = D(I)
      CONTINUE

A AND R(T) WILL BE TRANSFORMED INTO THE MODIFICATIONS OF EISPACK ROUTINES (MODI)

WARNING: DO NOT CHANGE THE ORDER OF THE CALLS

CALL FG(NDIM,N,B,1,V,ORT,EPS,IERR)
CALL RG(MDIM,N,A,0,A,ORT,EPS,IERR)

NOW TRANSFORM THE RIGHT HAND SIDE.

CALL TRANSF(A,ORT,1,C,V,0,N,N,NDIM,NCIM)

NOW SOLVE THE SYSTEM OF EQUATIONS (ELOC) THE SYSTEM IS BLOCK UPPER HESSEMBERG WITH MATRICES ON THE DIAGONAL (A + B(I,I) * THE IDENTITY OFF THE DIAGONAL. TESTS ARE MADE TO SEE WHETHER A SINGLE MUST BE HANDLED AT THE CURRENT STAGE AND SUBROUTINES ARE CALLED.

SET RELATIVE ERROR TOLERANCE

REPS = EPS*N*N*N*N
IND = N - 1
IF (IND.EQ.0) GO TO 40
IF (CABS(S(IND + 1,IND)) .LE. REPS) GO TO 30
CALL NSOLV(A,B,C,NDIM,N,MDIM,M,IND,I)
IF (IEFF.NE.0) RETURN
GO TO 30
CALL NSOLVE(A,B,C,NDIM,N,MDIM,M,IND,I)
IF (IEFF.NE.0) RETURN
IF (IND.EQ.0) GO TO 40
IF (IND.GT.0) CALL NSOLVE(A,B,C,NDIM,N,MDIM,M,IND,I)
THE SOLUTION OF THE TRANSFORMED SYSTEM HAS NOW BEEN OBTAINED. IT IS TRANSFORM THE SOLUTION OF THE ORIGINAL SYSTEM CF

CALL TRANSF(A,ORT,0,C,V,1,N,N,NDIM,NCIM)
RETURN
ENC

SUPROUTINE PG(NDIM,N,A,MATZ,Z,ORT,EPS,IERR)
INTEGER N,NDIM,IEFF,MATZ
REAL*8 A(NDIM,N),Z(NDIM,N),ORT(N),EPS
APPROPRIATE FORMS USING FIE C BY S. NASH).

NEXT TWO STATEMENTS

0)

BACK SUBSTITUTION)
TH UPP ER HESS E NBERG
I) ANC MULTIPLES OF
OR A COULE E BLOCK
D APPROPRIATE

0 20
PR ,RPPS, IERR)
PR ,RPPS, IERR)
N ,V DIM, M, IND, IPR ,RPPS, IERR)
OF EQUATIONS
ED BACK INTO
EQUATIONS.

C)

ERR)
ORTHOGONAL TRANSFORMATIONS

A matrix A is upper-Hessenberg if its first subdiagonal is zero except for the element just above the subdiagonal. The eigenvectors of an upper-Hessenberg matrix A can be computed by transforming A to upper-triangular form and then applying the Schur form. If A is transformed to upper-triangular form, the eigenvalues of A are the same as those of the Schur form. However, if A is transformed to upper-triangular form, the eigenvectors are not the same as those of the original matrix. The details of the transformation and the Schur form are discussed in the literature.

For the ALGOL procedures, ORTHES, and WILKINSON, see the relevant publications for details.
THIS SUBROUTINE DEPENDS ON THE SIZE OF THE MATRIX MO. A similar subroutine, SCOTM, is not used here.

PARAMETERS:

- **MATZ** = INTEGER (INPUT) = MATZ=0 FOR TRANSFORMED (CN)
- **Z** = VECTOR (INPUT) = Z=0.0 FOR TRANSFORMED (CN)
- **A** = MATRIX (INPUT) = A
- **N** = INTEGER (INPUT) = N
- **ORT** = INTEGER (OUTPUT) = ORT
- **EPS** = INTEGER (OUTPUT) = EPS
- **IERR** = INTEGER (OUTPUT) = IERR

CALL COTRENDIM(N, A, ORT)
CALL COTRZ(N, A, ORT, Z)
CALL COTRNZ(N, A, Z, ORT, Z)
RETURN
END
A CONTAINS THE HESSNERG MATRIX
THE ORTHOGONAL TRANSFORMATION
IS STORED IN THE REMAINING TIME
HESSNERG MATRIX:

ORT CONTAINS FURTHER INFORMATION
ONLY ELEMENTS 1 THROUGH N ARE

QUESTIONS AND COMMENTS SHOULD BE
APPLIED MATHEMATICS DIVISION, ARG

---------------------------------

LA = N - 1
KP1 = 2
IF (LA .LT. KP1) GC TC 200

DO 180 M = KP1, LA
      F = 0.000
      ORT(M) = 0.000
      SCALE = 0.000

      SCALE COLUMN (ALCCL TOL THEN NOT

      DO 90 I = N, N
         SCALE = SCALE + DABS(A(I,M-1))

      GO
      C

      IF (SCALE .EQ. 0.000) GO TO 18
         MP = N + N

      C

      FOR I=N STEP -1 UNTIL M DO --

      DO 100 II = M, N
         I = MP - II
         ORT(I) = A(I,M-1) / SCALE
         H = H + ORT(I) * ORT(I)

      100 CONTINUE

      G = -DSIGN(DSCRT(H),ORT(M))
      H = H - ORT(M) * G
      ORT(M) = ORT(N) - G

      C

      FORM (I-(I*UT))/H) * A

      DC 130 J = N, N
      F = 0.000

      C

      FOR I=N STEP -1 UNTIL M DO

      DC 110 II = N, N
         I = MP - II
         F = F + CRT(I) * A(I,J)

      110 CONTINUE

      C

      F = F / H

      C

      DO 120 I = N, N
         A(I,J) = A(I,J) - F * ORT(I)

      120 CONTINUE

      C

      CONTINUE
IX. INFORMATION ABOUT
TRANSFORMATIONS USED IN THE REDUCTION
TRIANGLE UNDER THE

DIRECTED TO R. S. GARROW,
CANDY NATIONAL LABORATORY

-------------

NEECOC
SUBROUTINE ORTRAN(NDIM,N,A,ORT,Z)

This subroutine is a translation of the INTEGRAL NORMALIZE subroutine (part of the ELLIPTICAL INTEGRAL subroutine) of the ELLIPTIC INTEGRAL subroutine. It transforms the input array A to its normalized form by ORTHOGONAL TRANSFORMATION. The transformation is used in the reduction of the HESSIAN matrix to its TRILINAR form. ORTHOGONALITY is preserved throughout the reduction.

NDIM must be set to the size of the orthogonal transformation. The order of the matrix is the only input parameter.

A contains the input transformation matrix. ORT contains the transformation parameters. Z contains the output transformation matrix.

END
ON OUTPUT:
7 CONTAINS THE TRANSFORMATION REDUCTION BY ORTHES;
ORL HAS BEEN ALTERED.

QUESTIONS AND COMMENTS SHOULD BE
APPLIED MATHEMATICS DIVISION, AR

---------------------------------

INITIALIZE Z TO IDENTITY MATRIX

DO 80 I = 1, N
    DO 60 J = 1, N
    Z(I, J) = 0.00D0
    Z(I, I) = 1.00D0
    CONTINUE

KL = N - 2
IF (KL .LT. 1) GOTO 200

FOR MP=N-1 STEP -1 UNTIL 2 DO ---

DO 140 MM = 1, KL
    MP = N - MM
    IF (A(MP, MP-1) .EQ. 0.00D0) GOTO 330
    MP1 = MP + 1

DO 100 I = MP1, N
    ORT(I) = A(I, MP-1)

DO 130 J = MP, N
    G = 0.00D0

DO 110 I = MP, N
    G = G + ORT(I) * Z(I, J)

DIVIDEP BELOW IS NEGATIVE DOUBLE DIVISION AVOIDS FCS

G = (G / ORT(MP)) / A(MP, N)

DO 120 I = MP, N
    Z(I, J) = Z(I, J) + G * ORT(I)

CONTINUE

CONTINUE

RETURN
END
MATRIX PRODUCED IN THE

DIRECTED TO B. S. GARROW,
GCOME NATIONAL LABORATORY

-------------------------

TC 140

OF E FORMED IN ORTHES,
SIELE UNDERFLOW

MP-1)

I)
FOR ARITHMETIC

---

HAS NOT BEEN


PRODUCED BY ORTRAN

IX:

THE CALLING PROGRAM

A MAXIMUM OF TWO DIAGONAL

SEEN BY MATRIX H.".

SAYS THAT IT ONLY


AND WILKINSON.

ALGOL PROCEDURE HQR2.

PLEX NUMBERS

REAL AND

DRM, EPS

DIM, NM.
**THE RELATIVE PRECISION OF FLOATING POINT EPS**

A machine dependent parameter specified determined after 30 iterations if the 20th eigenvalue zero is set to 7.2. CONTINUE the transformation matrix if has been destroyed.

In Output:

After the rectangular array output.

CONTINUE the transformation matrix.

N is the order of the matrix:

DIMENSION STATEMENT:

ARRAY PARAMETERS AS DECLARED IN THE STATEMENT.

REAL*8 CREAL, DREAL

COMPLEX*16 DCMPLEX

INTEGER*4 NATL

REAL*16 R(NMAT*NZ)

INTEGER I,J,K,L,N

REAL*8 EPS(NZ,NZ)

REAL*8 H(NZ,NZ)

REAL*8 X(NZ)

INTEGER*2 ENDR, IERF

SUBROUTINE H02 (NCIW,NH,JEPS,IERP)
C DO 50 I = 1, N
C
C DO 40 J = K, N
C NORM = NORM + DABS(H(I,J))
C
C K = I
C 50 CONTINUE
C
C EN = N
C T = 0.000
C
C SEARCH FOR NEXT EIGENVALUES
C
C 60, IF (EN .LT. 1) GO TO 340
C ITS = 0
C NA = EN - 1
C FNM2 = NA - 1
C
C LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C FOR L=EN STEP -1 UNTIL 1 DO --
C
C 70 DO 80 LL = 1, EN
C
C IF (L .EQ. 1) GC TO 100
C S = DABS(H(L-1,L-1)) + DABS(H(L,L))
C IF (S .LT. 0.000) S = NORM
C IF (DABS(H(L,L-1)) .LE. EPS * S) GC TO 80
C 80 CONTINUE
C
C FORM SHIFT
C
C 100 X = H(FN,EN)
C IF (L .EQ. EN) GO TO 270
C Y = H(NA,NA)
C W = H(FN,NA) * H(NA,EN)
C IF (L .EQ. NA) GC TO 280
C IF (ITS .LE. 30) GC TO 1000
C IF (ITS .LE. 10 .AND. ITS .GT. 20) GO TO 108
C
C FORM EXCEPTIONAL SHIFT
C
C 270 T = T + X
C
C DO 120 I = 1, EN
C
C H(I,I) = H(I,I) - X
C
C 280 S = DABS(H(FN,NA)) + DABS(H(NA,ENM2))
C X = 0.75D0 * S
C Y = X
C W = -0.4375D0 * S * S
C ITS = ITS + 1
C
C LOCK FOR TWO CONSECUTIVE SMALL
C SUP-DIAGONAL ELEMENTS.
C
C 300 FOR M=EN-2 STEP -1 UNTIL L DO --
C
C 310 DO 140 MM = L, ENM2
C XM = ENM2 + L - MM
C ZZ = H(M,M)
C R = X - ZZ
COLUMN MODIFICATION

DO 230 I = 1, J
P = X * H(I, K) + Y * H(I, K+1)
IF (NOT. NTLAS) GO TO 220
P = P + ZZ * H(I, K+2)
H(I, K+2) = H(I, K+2) - P * R
H(I, K+1) = H(I, K+1) - P * Q
H(I, K) = H(I, K) - P
230 CONTINUE

ACCUMULATE TRANSFORMATIONS

DO 250 I = 1, N
P = X * Z(I, K) + Y * Z(I, K+1)
IF (NOT. NTLAS) GO TO 240
P = P + ZZ * Z(I, K+2)
Z(I, K+2) = Z(I, K+2) - P * R
Z(I, K+1) = Z(I, K+1) - P * Q
Z(I, K) = Z(I, K) - P
240 CONTINUE

250 CONTINUE

260 CONTINUE

GO TO 70

ONE ROOT FOUND

270 H(EN, EN) = X + T
EN = NA
GO TO 60

TWO ROOTS FOUND

290 P = (Y - X) / 2.0, CCC
Q = P * P + W
ZZ = DSOFT(DABS(Q))
H(EN, FN) = X + T
X = H(EN, EN)
H(NA, NA) = Y + T
IF (G .LT. 0.000) GO TO 320

REAL PAIR

ZZ = P + DSIGN(ZZ, P)
X = H(FN, NA)
S = DABS(X) + DABS(ZZ)
P = X / S
Q = ZZ / S
R = DSOFT(P * P + Q * Q)
P = P / R
C = C / R

ROW MODIFICATION

DO 290 J = NA, N
ZZ = H(NA, J)
H(NA, J) = Q * ZZ + P * H(EN, J)
H(FN, J) = Q * H(EN, J) - P * ZZ
290
As for the subroutine
see subroutine arthes.

The transformation

Ed. If \( i = 1 \), then

end. If \( v(x) \) (transpose)
The integer variables

\( t(x,y), c(1:10), g \)

\( \omega \)

\( \phi \)

\( \theta \)
C
IF (M2 .LE. 0) GO TO 45
DO 40 KK = 1, M2
     K = M2 - KK + 1
     IF (IT1 .EQ. 1) K = KK
     K1 = K + 1
     IF (A(K1,K) .EQ. 0.0 .DO) GO TO 40
     D(K1) = ORT(K1)
     K2 = K + 2
     DO 10 I = K2, N
     D(I) = A(I,K)
10 CONTINUE
DC 30 J = 1, N
G = 0.0
DO 20 I = K1, N
     G = G + D(I) * C(I,J)
20 CONTINUE
DC 30 I = K1, N
C(I,J) = C(I,J) + G * D(I)
C
C
FORM C(NEW) = (U * C) * V
C
DO 60 I = 1, N
DO 50 J = 1, N
D(J) = 0.0
DO 50 K = 1, N
     IF (IT2 .EQ. 0) D(J) = D(J) + C(I,J)
     IF (IT2 .EQ. 1) D(J) = D(J) + C(I,J)
50 CONTINUE
DO 60 J = 1, N
C(I,J) = C(J)
60 CONTINUE
RETURN
END

SUBROUTINE NSCLVE(A, B, C, D, NCIM, N, M)
C
INTEGER I, N, IERR, IND, IPR(1), IROW(1, J), K
REAL*8 A(MDIM,M), E(NDIM,N), C(MDIM,N), S

THIS SUBROUTINE SETS UP AND SOLVES (WITH THE SUBROUTINE HESCLV) A SINGLE SYSTEM
C
\[ A * C + E(IND+1, IND+1) * I = C(IND+1) \]
C
\( (X(I), C(I)) - COLUMN I OF THE MATRICES \)
C
\( A AND STORES THE RESULT IN THE APPROPRIATE \)
C
\( MATRIX C. THE UPPER-HESSIAN MOUTH MATRIX \)
C
\( VECTOR D IN ORDER TO SAVE STORAGE (ENTIRE \)
C
\( DIAGONAL ARE IGNORED) \)
C
\( THE RIGHT HAND SIDE IS ALSO STORED IN \)
C
\( TO REDUCE THE NUMBER OF VECTORS USED \)
C
\( THE PARAMETERS ARE AS IN THE SUBROUTI..."
\[ c(i, k) * v(k, j) \]
\[ c(i, k) * v(j, k) \]
CRSC INTERCHANGE

RIGHT HAND SIDE (AN INPUT).

All Parameters (The Method Used)

The Vector D" The Method Used

SEVERAL SYSTEM OF EQUATIONS

\[ \text{MIN} \]

\[ j, i, k, n, l, m, n \]

Trace The Result In C.

Within The Vector D

\[ e \text{ necessary} \]

----------------------------------------
REAL (9,1) D, EPS, WNL, REP
INTEGER I, II, IER, IPR(1), IRMW(1),
SUBLANGUAGE HEAVY(O,1), IPR, M, EPS, IER
END
RETURN
IERP = -IND - 1
RETURN
IND = 1
CONTINUE
C (I,1)+1) = C(I,0)+1)
D(I,1) = D(I,1) + B(I,1)+1, I+1
IF (I.E.0.1) J = 1
J = 2
CONTINUE
D(M+1)+1, II+1) = A(I,1,2)
DO 10 J = II, N
IF (I.E.0.1) II = I - 1
II = I
M1 = IRMW(I,1)
FIN INDEX OF BEGINNING OF ROW I
DO 20 I = 1, M
MFIN = (M + I * I) / (2 + I)
RETURN
IF (IND+1, N = 1) CALL BACKSUB(C, B, EPS
PERFORM BLOCK BACK-SUBSTITUTION IF NECESSARY.
IEPR = ERRCC (IEPR=0 => NE,  
              (IEPR=-1 => S)

INITIALIZE INTERCHANGE VECTORS AND

IEPR = 0
MFIN = (M * (M + 1)) / 2 + M
D(J) 10 I = 1, M
IPR(I+1) = IFCW1(I,M)  
IPR(I) = I + MFIN
CONTINUE
M1 = M - 1

REDUCE THE MATRIX TO UPPER TRIANGULAR

IF (M.EQ.1) GO TO 35
DO 20 I = 1, M1
       IF (DABS(D(IPR(M+I+1))) .GT. DABS)

          INTERCHANGE ROWS IF NECESSARY

          K = IPR(M+I)
          IPR(M+I) = IPR(M+I+1)
          IPR(M+I+1) = K
          K = IPR(I)
          IPR(I) = IPR(I+1)
          IPR(I+1) = K

          CHECK FOR COMPLETLY DIAGONALLY SINGULAR

          IF (DABS(D(IPR(M+I+1))) .LT. REPS)
            IPR(M+I+1) = IPR(M+I+1) + 1

          ELIMINATE SUBDIAGONAL ELEMENTS

          MULT = D(IPR(M+I+1)) / C(IPR(M+I+1))
          D(IPR(I+1)) = D(IPR(I+1)) - MULT
          I1 = I + 1
          DO 20 J = I1, M
             D(IPR(M+I+1)+J-1) = D(IPR(M+I+1)+J-1) - MULT
          CONTINUE

          IF (DABS(D(IPR(M+M+1))) .LT. REPS)

            PERFORM BACK-SUBSTITUTION

            D(IPR(M)) = D(IPR(M)) / D(IPR(M+M))
            IF (M1.EQ.0) RETURN
            DO 50 I1 = 1, M1
                 I = M - I1
                 I2 = I + 1
                 MULT = 0.0
                 DO 40 J1 = I2, M
                  J = J1 - I2 + 2
                  MULT = MULT + D(IPR(J1))
                  CONTINUE
                  C(IPR(I)) = (D(IPR(I)) - MULT)
            CONTINUE

          30
          35
          40
          50
FINAL RETURN
SINGULAR MATRIX

PARAMETERS.

S(C(IPR(M+I+1)+1)) GO TO 20

SINGULAR FORM

S(C(IPR(M+I+1)+1)) GO TO 20

SINGULAR MATRIX

PS) GO TO 60

CF A

(M+I+1)+1)

ULT * D(IPR(I))

(M+I+1)+J-I) -

* C(IPR(M+I)+J+1-I)

GO TO 60

(M+V)+1)

* C(IPR(M+I)+J)

M'T) / D(IPR(M+I)+1)
AND THEN USING GAUSSIAN
10 INSURE THAT THE ERE

N \end{pmatrix} = \begin{pmatrix}
(1 + \kappa) (X (I D + \kappa)) (C (I D + \kappa)) \\
C (I D + \kappa) (X (I D + \kappa)) \end{pmatrix}

THE SET OF EQUATIONS.

\text{REFERENCES}

J. JI, J. W., K. L. W. M. D.

C (I D, I P R, I P S, I E R R)

\begin{pmatrix}
C (I D + \kappa) - C (I D) \\
C (I D)
\end{pmatrix} (I D, I P R, I P S, I E R R)

2

A T R I X.

C (A T R I X) (I D, I P R, I P S, I E R R)

N C I M

A T R I X.

N C I M

N C I M

A T R I X.
ELIMINATION. IN ORDER TO TAKE ADVANTAGE OF ZEROES IN THIS MATRIX, THE MATRIX IS
BROUGHT INTO PRESSED FORM. (ENTRIES BELOW THE MAIN DIAGONAL ARE IGNORED.) THE RIGHT-HAND SIDE AND SOLUTIONS
THE PARAMETERS ARE AS IN THE SUBROUTINE

PERFORM BLOCK BACK-SUBSTITUTION IF N > 1

IF (IND.LT.N-1) CALL BACK2(C,B,IND)

SET UP THE SYSTEM OF EQUATIONS FOR M = 2 * M

M2 = 2 * M
MFIN = (M2 * (M2 + 1)) / 2 + 4 * N

DO 20 1 = 1, M

FIND BEGINNING AND LENGTH OF ROWS

M1 = IRNW2(2*I-1,M)
K = LRNW2(2*I-1,M)
II = I - 1
IF (I*EQ.1) II = 1

DO 10 J = II, M

J1 = 2 * (J - II + 1)
J2 = 1
IF (M1*EQ.0) J2 = 0
D(M1+J1-1) = A(I,J)
D(M1+J1) = 0.0
D(M1+K+J1-J2) = A(I,J)
D(M1+K+J1-1-J2) = 0.0

10 CONTINUE

CONTINUE

IF (I*EQ.1) J1 = 1
D(J1+M1) = D(J1+M1) + B(IND,INC)
D(J1+M1+1) = D(J1+M1+1) + B(IND+1)

IF (I*EQ.0) J1 = 2
D(J1+M1+K) = C(J1+M1+K) + B(IND+1)
D(J1+M1+K+1) = D(J1+M1+K+1) + E(IND)

STORE RIGHT-HAND SIDE

D(2*I+MFIN) = C(I,INC+1)
D(2*I-1+MFIN) = C(I,IND)

CONTINUE

20 CONTINUE

SOLVE THE SYSTEM OF EQUATIONS AND STORE

CALL H25CLV(D,IPR,M,REFS,IERR)
IF (IPR*EQ.0) GO TO 40

DO 30 I = 1, M

C(I,IND) = D(2*I-1)
C(I,IND+1) = C(2*I-1)

30 CONTINUE

INC = IND - 2
RETURN

40 IERR = -IND - 1
RETURN

END
STATE OF THE NUMBER OF
6 STORED IN THE VECTOR
SUE-ULR-DIAGONAL ARE
SOLUTION ARE ALSO STORED IN D.
DINE AXXBGC.

---------------------------------

NECESSARY
(N,M,MDIM,NDIM)
-2SOLV

2I-1 IN THE VECTOR D.

(IND+1)
(1,IND)
(IND+1,IND+1)

STORE THE RESULT BACK IN C.
Matrix

max (GC T0 20

* METERS

-------------------------------------------
| LAE MATRIX |
| RETURN |

RMEUTATIONS

IT HAND SIDE (ON INPUT).

N.

CTUR IN PACKED FORM.
SUE - E5 - DIAGONAL
* WM ECUATIONS

K. K. L. V42. N21. MFIN
CHECK FOR COMPUTATIONAL SINGULARITY

CONTINUE

L =
MAX = 0.485(CIPR(W2+1)+I)
IF (IAGS(0) IFR(M2+1)+I).LE.
0.20 J = J+1
MAX = MAJJ5(DIIP(W2+1)+I)

INTERCHANGE PUPS IF NECESSARY

L = 0

IF (I=60*I21) J = I
I = I+2
J = J+1

TRANSFORM MATRIX TC TRIANGULAR FORM

M21 = M2 - 1
CONTINUE

IAG = 0, W2 = 1
IPR = IFR(W2+1)
IPR = IFR(W2+1)
I = I21 = J+1
MAX = (W2+1) / 2 + 4
N = W2 + 2
I = 0

PARAMETERS:
The method used is Gaussian elimination where the matrix is stored as a
and vertical lines are traced along the columns.
The subroutine solves a system of 2n
REAL AUX(1), MAX, IAE
INTEGER I, I, IER, IFR(1), IAGS, J
SUBROUTINE H2SOLV(D,IPR,W,REPS,IER)

---

IER = ERROR CODE (IER = 0 = NORMAL)
IPR = ERROR TOLERANCE
M = SIZE OF SYSTEM
IPR = VEGET LSSD TO RECORD FOR CODE
AND SAVATION (CN JUETIL)
IPR = MATRIX CF COEFFICIENTS, AICH
IPR = MATRIX CF COEFFICIENTS, AICH
IPR(I+L) = IPR(I+L) + K
CONTINUE

ADJUST POINTERS TO BEGINNINGS
IPR(M2+I+1) = IPR(M2+I+1) + 1
IF (I+NF+M2) IPR(M2+I+2) = I + 1
IF1 = I + 1

ELIMINATE SUBDIAGONAL ELEMENT
DO 40 J = 1, I1
      MAX = D(IPR(N2+I+J)) / D(IPR(I+J))
      D(IPR(I+J)) = D(IPR(I+J))
      DO 40 K1 = IF1, M2
      K = K1 - I
      D(IPR(M2+I+J)+K) = D(IPR(M2+I+J))

40 CONTINUE
IF (DABS(D(IPR(M2+M2)+1)), LE. REP.

PERFORM BACK SUBSTITUTION

D(IPR(M2)) = D(IFN(M2)) / C(IPR(M2))
D0 60 I1 = 1, M2
      I = M2 - I1
      I2 = I + 1
      MAX = 0, D0
      DO 60 J1 = I2, M2
      J = J1 - I2 + 2
      MAX = MAX + D(IPR(J1)) * D(IPR(J1))
60 CONTINUE
D(IPR(I)) = (C(IPR(I)) - MAX)

CONTINUE
RETURN

SUBROUTINE BACKS2(C, B, IND, N, M, MDIM)
C INTEGER I, IND, IND1, IND2, J, M, MDIM
C REAL*8 B(IND1:N), C(MDIM:N)

BLOCK PACK - SUBSTITUTION FOR TWO PARAMETERS ARE AS IN SUBROUTINE

IND1 = IND + 1
IND2 = IND + 2
DO 10 I = IND2, N
     DO 10 J = 1, M
     C(J, IND1) = C(J, IND1) - E(J)
     C(J, IND) = C(J, IND) - B(IN)
10 CONTINUE
RETURN
CF ROWS

PR(M2+I+2) + 1

S IN THE MATRIX

PR(M2+I+1) - MAX * D(IPR(I))

R(M2+I+J)+K) =
MAX * D(IPR(M2+I+1)+1+K)

S) GC T1 80

M2+M2)+1)

(IPR(M2+I)+J)
/ C(IPR(M2+I)+1)

IM,NDIM)
N,NCIM

J 'FCSWS'
A*X*BC

INC1,I) * C(J,I)
D,I) * C(J,I)
INTEGER FUNCTION IFNW(I,M)

THIS FUNCTION FINDS THE INDEX (FJ)
BEFORE THE BEGINNING OF ROW I IN THE

RETURN
END

INTEGER FUNCTION LFCW2(I,M)

THIS FUNCTION FINDS THE LENGTH OF THE DOUBLE MATRIX WHEN IT IS STORED IN

RETURN
END