A PROGRAM TO SWAP DIAGONAL BLOCKS

by

B. N. Parlett

Memorandum No. UCB/ERL M77/66

3 November 1977

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720
A PROGRAM TO SWAP DIAGONAL BLOCKS

by

B.N. Parlett

Department of Mathematics
and
Computer Science Division
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California at Berkeley
Berkeley, CA 94720

*Research supported by Office of Naval Research Contract N00014-76-C-0013.
A PROGRAM TO SWAP DIAGONAL BLOCKS

Contents

1. Introduction ........................................
2. Ruhe's Trick ........................................
3. The General Case ...................................
4. The Algorithm for SWAP ............................
5. Solving $A_1X - XA_2 = \xi B$ ....................
6. Solving $CTC = W$ ...................................
7. Performing the Similarity Transformation .......
8. Gaussian Elimination for Solving $A_1X - XA_2 = B$ ....
9. Swapping Large Blocks ............................
10. Test Results ........................................
11. Program Listing ....................................
References .............................................

Acknowledgment

The author is grateful to Mr. Allan McCurdy for testing the programs and helping to improve them.
1. Introduction

A triangular matrix reveals its eigenvalues on the main diagonal. By Schur's lemma any square matrix is unitarily similar to an upper triangular matrix with the eigenvalues arranged in any desired order along the diagonal. In practice the QR algorithm in real arithmetic produces a block triangular matrix in which the eigenvalues are likely to be in monotone decreasing order by absolute value down the diagonal. However this monotonicity cannot be guaranteed and for some purposes the ordering by absolute value is not what is wanted.

The problem which we address here is to find some simple orthogonal similarity transformations which have the effect of exchanging two diagonal elements (or blocks) while preserving block triangular form. Actually we will show only how to swap adjacent blocks and so the exchange of distant blocks must be accomplished by a succession of adjacent swaps.

Although the cost of such a swap is small it is not negligible; in an $n \times n$ matrix $(p+q)^2 n$ multiplications are needed to swap adjacent diagonal blocks of orders $p$ and $q$.

2. Ruhe's Trick

For any real $\theta$ and $s = \sin \theta$, $c = \cos \theta$ the symmetric matrix

\[
\begin{pmatrix}
-s & c \\
c & s
\end{pmatrix}
\] is an orthogonal matrix representing a reflection of the plane. Observe that

\[
\begin{pmatrix}
-s & c \\
c & s
\end{pmatrix}
\begin{pmatrix}
\alpha_1 & \beta \\
0 & \alpha_2
\end{pmatrix}
\begin{pmatrix}
-s & c \\
c & s
\end{pmatrix} =
\begin{pmatrix}
\alpha_1 s^2 - \beta sc + \alpha_2 c^2, & -\alpha_1 sc - \beta s^2 + \alpha_2 sc \\
-\alpha_1 sc + \beta c^2 + \alpha_2 sc, & \alpha_1 c^2 + \beta sc + \alpha_2 s^2
\end{pmatrix}
\]

The new matrix is upper triangular if and only if

\[c[\beta c - (\alpha_1 - \alpha_2)s] = 0.\]
The choice \( c = 0 \) represents no change, the choice

\[
\tan \theta = s/c = \beta/\alpha_1 - \alpha_2
\]

results in an exchange of \( \alpha_1 \) and \( \alpha_2 \). The new \((1,2)\) element is

\[
-s[s + (\alpha_1 - \alpha_2)c] = -s[s + \beta c^2/s] = -\beta.
\]

Now suppose that \( \alpha_1 \) is the \((j,j)\) element of an \( n \times n \) upper triangular matrix. The plane reflection indicated above, effected in the \((j,j+1)\) coordinate plane, will swap \( \alpha_1 \) and \( \alpha_2 \). Postmultiplication affects columns \( j \) and \( j+1 \) while premultiplication affects rows \( j \) and \( j+1 \). This requires \( 4(n-2) \) multiplications. To keep the angle \( \theta \) in \((-\pi/2,\pi/2)\) we define

\[
d = \sqrt{(\alpha_1 - \alpha_2)^2 + \beta^2},
\]

\[
c = |\alpha_1 - \alpha_2|/d,
\]

\[
s = \beta \text{ sign}(\alpha_1 - \alpha_2)/d.
\]

Note that when \( \beta = 0 \) the transformation merely exchanges the two rows and the corresponding pair of columns.

3. The General Case

Consider the reduced matrix

\[
\begin{pmatrix}
A_1 & B \\
0 & A_2
\end{pmatrix}, \quad A_1 \text{ is } p \times p,
\]

\[
A_2 \text{ is } q \times q.
\]

We seek an orthogonal similarity transformation which swaps \( A_1 \) and \( A_2 \). In general this is not possible; fortunately we can achieve a form which is as useful as exchanging \( A_1 \) and \( A_2 \). We denote by \( Z^T \) the transpose
of any matrix $Z$. A partitioned matrix

\[
\begin{pmatrix}
-S_1^T & C_2 \\
C_1 & S_2
\end{pmatrix}, \quad C_1 \text{ is } p \times p, \quad C_2 \text{ is } q \times q,
\]

is orthogonal if, and only if, the following relations hold:

1. \[C_1 C_1^T + S_2 S_2^T = I_p = S_1^T S_1 + C_1^T C_1,\]
2. \[S_1^T S_1 + C_2 C_2^T = I_q = C_2^T C_2 + S_2 S_2^T,\]
3. \[-C_1 S_1 + S_2 C_2^T = 0_{p,q},\]
4. \[-S_1 C_2 + C_1^T S_2 = 0_{q,p}.\]

Note that if $C_1^T = C_1, \ C_2^T = C_2$ then we can take $S_1 = S_2$, however this is not always advantageous.

We seek an orthogonal matrix of the form shown above such that

\[
\begin{pmatrix}
-S_1^T & C_2 \\
C_1 & S_2
\end{pmatrix}
\begin{pmatrix}
A_1 & B \\
0 & A_2
\end{pmatrix}
= \begin{pmatrix}
\tilde{A}_2 & \tilde{B} \\
0 & \tilde{A}_1
\end{pmatrix}
\begin{pmatrix}
-S_1^T & C_2 \\
C_1 & S_2
\end{pmatrix},
\]

On equating the $(2,1)$ and $(2,2)$ blocks on each side of the equation we find

5. \[C_1 A_1 = \tilde{A}_1 C_1, \quad \text{(also } A_2 C_2^T = C_2^T \tilde{A}_2),\]
6. \[C_1 B + S_2 A_2 = \tilde{A}_1 S_2.\]

When $C_1$ is invertible (more on this below) then (6) can be rewritten as

\[B + C_1^{-1} S_2 A_2 = C_1^{-1} \tilde{A}_1 S_2 = A_1 C_1^{-1} S_2, \quad \text{by (5).}\]
We now let the $p \times q$ matrix $C_1^{-1}S_2 = X/\xi$, where $\xi$ is a positive constant at our disposal, and substitute into the equation above to get

$$A_1X - XA_2 = \xi B.$$  

In order to obtain $C_1$ from $X$ we pre- and post-multiply the first orthogonality relation (1) appropriately and invert to find

$$I_p + XX^T/\xi^2 = C_1^{-1}C_1^{-T}$$

or

$$(C_1/\xi)^T(C_1/\xi) = (I_p\xi^2 + XX^T)^{-1} = W_1.$$ 

Using (3) we find that $X/\xi$ also equals $S_1C_2^{-T}$ and by using (2) we obtain

$$(C_2/\xi)^T(C_2/\xi) = (I_q\xi^2 + X^TX)^{-1} = W_2.$$ 

It is well known that an $X$ satisfying (7) exists and is unique if and only if $A_1$ and $A_2$ have no eigenvalues in common. In practice only such cases interest us but we want the algorithm to be robust in the face of some perverse or extreme requests. Clearly if $A_1 = A_2$ we want the algorithm to do nothing rather than to fail. In such a case $C_1 = C_2 = 0$ which is far from invertible. By taking $\xi = 0$ and setting $C_1/\xi = C_2/\xi = X = I$ the algorithm will work. When the eigenvalues of $A_1$ and $A_2$ are close, in some sense, then $\xi$ will be chosen so that

$$\max\{\xi, \|X\|\} = 1$$ 

approximately.

There are infinitely many $C$'s satisfying (8) and (9) and any of them will do. In the absence of other constraints the symmetric solutions are the natural ones; if $C_1^T = C_1$, $C_2^T = C_2$ then $S_1 = S_2$, but this fact is not obvious. In this algorithm, however, we prefer to choose $C_1$ and $C_2$ so that $\tilde{A}_1$ and $\tilde{A}_2$ have a convenient form for most applications.
It is not necessary to compute $S_1$ and $S_2$ explicitly. Write $	ilde{C}_1 = C_1 / \xi$, the scaled version of $C_1$. Then

\begin{equation}
(10) \quad P = \begin{pmatrix} -S_1^T & C_2 \\ C_1 & S_2 \end{pmatrix} = \begin{pmatrix} \tilde{C}_2 & 0 \\ 0 & \tilde{C}_1 \end{pmatrix} \begin{pmatrix} -X^T & \xi I \\ \xi I & X \end{pmatrix}
\end{equation}

and $P$ is best applied in this factored form. In practice the orthogonality of $P$ is completely determined by the accuracy with which the $C$'s satisfy (8) and (9).

It is not necessary to compute $\tilde{A}_1$, $\tilde{A}_2$, or $\tilde{B}$ explicitly since they will emerge when the similarity transformation

\begin{equation}
(11) \quad P \begin{pmatrix} A_1 & B \\ 0 & A_2 \end{pmatrix} P^T
\end{equation}

is effected. For completeness we give the formulas

\begin{equation}
(12) \quad S_1 = X \tilde{C}_2^T, \quad S_2 = \tilde{C}_1 X, \\
\tilde{A}_1 = \tilde{C}_1 A_1 \tilde{C}_1^{-1}, \quad \tilde{A}_2 = \tilde{C}_2 A_2 \tilde{C}_2^{-1}, \\
\tilde{B} = \tilde{A}_2 C_2^{-T} S_2^{-T} - S_1^T C_1^{-1} \tilde{A}_1 = C_2^{-T} A_2 S_2^{-T} - S_1^T A_1 C_1^{-1}.
\end{equation}
4. The Algorithm for SWAP $A = \begin{bmatrix} A_1 & B \\ 0 & A_2 \end{bmatrix}$

1. Clear the (2,1) block of $A$.
2. Solve $A_1X - XA_2 = \xi B$ for $X$ and $\xi$ using subroutine TXMXT.
   $\xi$ is chosen so that $\|X\| \neq 1$
3. If $\xi = 0$ then exit.
4. Solve $\hat{c}_1^T \hat{c}_1 = (\xi^2 + XX^T)^{-1} \hat{c}_1$ for $\hat{c}_1$ using CTCEQW.
5. Solve $\hat{c}_2^T \hat{c}_2 = (\xi^2 + X^TX)^{-1} \hat{c}_2$ for $\hat{c}_2$ using CTCEQW.
6. Premultiply $A$ by $P$ using NEWCOL.
7. Postmultiply $PA$ by $P^T$ using NEWROW.
8. Update the matrix of orthogonal transformations using NEWROW.
9. Force the diagonal elements in the new blocks $\hat{A}_1$ and $\hat{A}_2$ to be equal.

<table>
<thead>
<tr>
<th>Name</th>
<th>Executable Statements</th>
<th>Count for 2x2 Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWAP</td>
<td>17</td>
<td>32n multiplications</td>
</tr>
<tr>
<td>TXMXT</td>
<td>52</td>
<td>42 multiplications</td>
</tr>
<tr>
<td>CTCEQW</td>
<td>17</td>
<td>32 multiplications, 4 square roots</td>
</tr>
<tr>
<td>NEWCOL</td>
<td>22</td>
<td>16 multiplications per column</td>
</tr>
<tr>
<td>NEWROW</td>
<td>22</td>
<td>16 multiplications per row</td>
</tr>
</tbody>
</table>

5. Solving $A_iX - XA_2 = B$

When $A_i = \begin{pmatrix} \alpha_i & \beta_i \\ \gamma_i & \alpha_i \end{pmatrix}$, $i = 1,2$, the linear equations which determine $X$ can be solved stably in closed form. Let $\delta = \alpha_1 - \alpha_2$, then the equations may be written as
(1) \[
\begin{pmatrix}
C & \beta_1 I_2 \\
\gamma_1 I_2 & C
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2
\end{pmatrix} = b; \quad
\begin{pmatrix}
x_1 \\
x_2
\end{pmatrix} = \begin{pmatrix} x_{11} \\
x_{12} \\
x_{21} \\
x_{22}
\end{pmatrix}, \quad
b = \begin{pmatrix} b_{11} \\
b_{12} \\
b_{21} \\
b_{22}
\end{pmatrix}
\]

where

(2) \[
C = \begin{pmatrix}
\delta & -\gamma_2 \\
-\beta_2 & \delta
\end{pmatrix}, \quad
C^2 = \begin{pmatrix}
\delta^2 + \beta_2 \gamma_2 & -2\delta \gamma_2 \\
-2\delta \beta_2 & \delta^2 + \beta_2 \gamma_2
\end{pmatrix}.
\]

Multiply (1) as indicated in order to make the coefficient matrix block diagonal,

(3) \[
\begin{pmatrix}
C^2 - \beta_1 \gamma_1 & 0 \\
0 & C^2 - \beta_1 \gamma_1
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2
\end{pmatrix} = \begin{pmatrix}
C & -\beta_1 \\
-\gamma_1 & C
\end{pmatrix}
\begin{pmatrix}
b_1 \\
b_2
\end{pmatrix}.
\]

Now let \[
G = (C^2 - \beta_1 \gamma_1)^{-1} = \begin{pmatrix}
\tau & 2\delta \gamma_2 \\
2\delta \beta_2 & \tau
\end{pmatrix}/d
\]
where

(4) \[
\tau = \delta^2 + \beta_2 \gamma_2 - \beta_1 \gamma_1, \quad
d = \tau^2 - (2\delta \beta_2)(2\delta \gamma_2),
\]

and premultiply (3) diag(G, G) to find

(5) \[
\begin{pmatrix}
G : 0 \\
0 : G
\end{pmatrix}
\begin{pmatrix}
\begin{pmatrix}
C & -\beta_1 \\
-\gamma_1 & C
\end{pmatrix}
\begin{pmatrix}
b_1 \\
b_2
\end{pmatrix}
\end{pmatrix}
\begin{pmatrix}
\delta b_{11} - \gamma_2 b_{12} - \beta_1 b_{21} \\
-\beta_2 b_{11} + \delta b_{12} - \beta_1 b_{22} \\
-\gamma_1 b_{11} + \delta b_{21} - \gamma_2 b_{22} \\
-\gamma_1 b_{12} - \beta_2 b_{21} + \delta b_{22}
\end{pmatrix}.
\[
\begin{bmatrix}
\phi \delta b_{11} + (2\delta^2 - \tau) Y_2 b_{12} & \tau \beta_1 b_{21} & 2\delta Y_2 \beta_1 b_{22} \\
(2\delta^2 - \tau) \beta_2 b_{11} & \phi \delta b_{22} & -\tau \beta_1 b_{21} \\
-\tau Y_1 b_{11} & 2\delta Y_1 Y_2 b_{12} & \phi \delta b_{21} + (2\delta^2 - \tau) Y_2 b_{22} \\
-2\delta Y_1 \beta_2 b_{11} & \tau Y_1 b_{12} + (2\delta^2 - \tau) \beta_2 b_{21} & \phi \delta b_{22}
\end{bmatrix} / d,
\]

defining \( y \), where
\[
\phi = \tau - 2Y_2 \beta_2 = \delta^2 - (\beta_1 Y_1 + \beta_2 Y_2) > 0,
\]
\[
\psi = 2\delta^2 - \tau = \delta^2 + (\beta_1 Y_1 - \beta_2 Y_2).
\]

Inevitably (5) is Cramer's rule and \( d = \det(A_1 \otimes I - I \otimes A_2) \) so that \( d = 0 \) if and only if \( \alpha_1 = \alpha_2, \beta_1 Y_1 = \beta_2 Y_2 \).

Among all the coefficients in the linear combinations of the elements of \( B \) which are given above only \( \tau \) and \( \psi \) involve genuine subtractions and possible loss of information through cancellation. However by rewriting them in a more complicated form all unnecessary loss can be avoided. From (4) \( \tau = \delta^2 + \beta_2 Y_2 - \beta_1 Y_1 \) and if either of \( \delta^2 \) or \( -\beta_1 Y_1 \) is tiny compared with the other two terms we want to add it in last. Similarly for \( \psi \). Thus we use
\[
\psi = (\beta_1 Y_1 + \max(\delta^2, -\beta_2 Y_2)) + \min(\delta^2, -\beta_2 Y_2),
\]
\[
\tau = (\beta_2 Y_2 + \max(\delta^2, -\beta_1 Y_1)) + \min(\delta^2, -\beta_1 Y_1).
\]

Here is an example for a machine with a relative precision of 8 decimals, i.e. the floating point result \( \text{fl}(10^8 - 9) \) is \( 10^8 \) whereas \( \text{fl}(10^8 - 10) \) is \( 10^8 - 10 = 10(10^7 - 1) \). Let \( \delta^2 = 9, \beta_1 Y_1 = -(10^8 - 10), \beta_2 Y_2 = -10^8 \) then from (4), computing from the left, \( \tau = \text{fl}(\text{fl}(9 - 10^8) + 10^8 - 10) = -10 \), from (6), computing from the left, \( \tau = \text{fl}(\text{fl}(-10^8 + (10^8 - 10) + 9) = -1 \).
If we are given a matrix $M$ with eigenvalues near $\pm 10^3 i$ and are evaluating $\exp(10M)$ then values like the ones given above will occur.

**Normalization**

The important matrix in effecting the orthogonal transformations is

$$
\begin{pmatrix}
\xi_2 & 0 \\
0 & \xi_1
\end{pmatrix}
\begin{pmatrix}
-x^T \\
-\xi
\end{pmatrix}
\begin{pmatrix}
\xi & X
\end{pmatrix}
$$

and we want our formulas to be accurate right out to both extremes:

$$
\begin{pmatrix}
-I & 0 \\
0 & I
\end{pmatrix}
\quad \text{and} \quad
\begin{pmatrix}
0 & I \\
I & 0
\end{pmatrix}
$$

An appropriate way to achieve this is to choose $\xi$ so that

$$\max\{\xi, \|X\|\} \neq 1.$$

Equation (6) above yields $y$ so that the corresponding $2 \times 2$ matrix $Y$ satisfies

$$A_1 Y - YA_2 = dB$$

where $d$ is given in (4). To get $x$ and $\xi$ let $\eta = \|y\|_{\infty}$ then

**Case 1:** $\eta \leq d$, take $x = y/d$, $\xi = 1$.

**Case 2:** $\eta > d$, take $x = y/\eta$, $\xi = d/\eta$. 
The Algorithm for TXMXT

We solve $A_1^X - XA_2 = \xi B$ when $A_1$ and $A_2$ are $2 \times 2$ standardized matrices as follows:

$\delta = \alpha_1 - \alpha_2$, \quad \delta sq = \delta^2$,

$\pi_1 = \beta_1 \gamma_1$, \quad \pi_2 = \beta_2 \gamma_2$,

$e = \phi \delta = \delta (\delta sq - (\pi_1 + \pi_2))$

$f_1 = \psi = (\pi_1 + \max\{\delta sq, -\pi_2\}) + \min\{\delta sq, -\pi_2\}$,

$f_2 = \tau = (\pi_2 + \max\{\delta sq, -\pi_1\}) + \min\{\delta sq, -\pi_1\}$,

$g = 2 \delta \gamma_2$, \quad h = 2 \delta \gamma_2$,

$d = f_2 - gh$.

At this point $y$ can be evaluated from (5). Then

$\eta = \|y\|_\infty$,

$x = \xi y / \max(d,\eta)$,

new $\xi = \xi \cdot d / \max(d,\eta)$.
6. Solving \( C^T C = W \)

In some applications the \( A_i, \; i = 1, 2, \) have the special form

\[
A_i = \alpha_i I_2 + \begin{pmatrix}
0 & \beta_i \\
\gamma_i & 0
\end{pmatrix}, \quad \beta_i \gamma_i < 0,
\]

and we want \( \tilde{A}_i \) to have the same standardized form (equal diagonal elements). Because \( \tilde{A}_i \) is similar to \( A_i \) we must have

\[
\tilde{A}_i = \alpha_i I_2 + \begin{pmatrix}
0 & \tilde{\beta}_i \\
\tilde{\gamma}_i & 0
\end{pmatrix}, \quad \tilde{\beta}_i \tilde{\gamma}_i = \beta_i \gamma_i.
\]

This requirement fixes the matrices \( C_1 \) and \( C_2 \) of the previous section. A straightforward way to derive formulas for \( C_1 \) and \( C_2 \) is to obtain a particular solution to (8) via the Choleski decomposition and then to standardize the resulting diagonal blocks.

Let \( R_1 \) and \( R_2 \) be upper triangular and satisfy

\[
R_1^T R_1 = W_1 \equiv (\xi^2 I_2 + xx^T)^{-1}
\]

\[
R_2^T R_2 = W_2 \equiv (\xi^2 I_2 + x^T x)^{-1},
\]

where \( x \) solves (7), \( A_1 x - x A_2 = \xi B \). Next define

\[
\tilde{A}_1 \equiv R_1 A_1 R_1^{-1}, \quad \tilde{A}_2 \equiv R_2^{-T} A_2 R_2.
\]

Now let \( J_1 \) and \( J_2 \) be the unique plane rotation matrices which standardize \( \tilde{A}_1 \) and \( \tilde{A}_2 \), i.e. both

\[
\tilde{A}_1 \equiv J_1 \tilde{A}_1 J_1^T \quad \text{and} \quad \tilde{A}_2 \equiv J_2 \tilde{A}_2 J_2^T.
\]

have equal diagonal elements. The appropriate \( C_1 \) and \( C_2 \) are therefore

\[
\hat{C}_1 = C_1 / \xi = J_1 R_1, \quad \hat{C}_2 = C_2^T / \xi = J_2 R_2.
\]
Let us drop the subscript and dot from $C_1$ and $A_1$. The condition 

$$C^TC = W \equiv (\xi^2I_2 + XX^T)^{-1}$$

imposes three quadratic relations on the four elements of $C$. If 

$$A = \begin{pmatrix} a & b \\ \gamma & \alpha \end{pmatrix}$$

then the requirement that $CAC^{-1}$ have equal diagonal elements (both $\alpha$) imposes another quadratic constraint, namely

$$BC_{11}C_{21} = \gamma C_{12}C_{22},$$

which suffices to determine $C$. However the direct solution of these nonlinear equations is far from obvious. Instead we shall derive the solution in a straightforward but lengthy manner via the Choleski factorization of $W$. The final algorithm is however very compact. Let

$$d^2 = \det(\xi^2I_2 + XX^T) = \xi^4 + \xi^2(\sum |x_{ij}|^2) + (\det X)^2.$$

Then define $M$ by

$$W = M/d^2 = \frac{1}{d^2} \begin{bmatrix} \xi^2 + x_{21}^2 + x_{22}^2 & -(x_{11}x_{21} + x_{12}x_{22}) \\ -(x_{11}x_{21} + x_{12}x_{22}) & \xi^2 + x_{11}^2 + x_{12}^2 \end{bmatrix}$$

and note that

$$R = \frac{1}{d} \begin{bmatrix} \sqrt{m_{11}} & m_{12}/\sqrt{m_{11}} \\ 0 & d/\sqrt{m_{11}} \end{bmatrix}$$

is the Choleski factor of $W$. Note that $\det M = d^2$. The next step is to form
\[ \hat{A} = \text{RAR}^{-1} \]

\[
= \begin{bmatrix}
  r_{11} & r_{12} \\
  0 & r_{22}
\end{bmatrix}
\begin{bmatrix}
  0 & -r_{12} \\
  r_{22} & 0
\end{bmatrix}
\begin{bmatrix}
  r_{11} & 0 \\
  0 & r_{22}
\end{bmatrix}^{-1}
+ \alpha I_2, \\
= \begin{bmatrix}
  \gamma r_{12} & (\beta r_{11} - \gamma r_{12})/r_{22} \\
  \gamma r_{22} & -\gamma r_{12}
\end{bmatrix}^{-1} + \alpha I_2,
\]

\[
= \begin{bmatrix}
  \delta & \hat{a}_{12} \\
  \hat{a}_{21} & -\delta
\end{bmatrix} + \alpha I_2.
\]

Now let \( J \) be the plane rotation which standardizes \( \hat{A} \).

\[
\tilde{A} = J\hat{A}J^T = \begin{bmatrix}
  c & -s \\
  s & c
\end{bmatrix}
\begin{bmatrix}
  \delta & \hat{a}_{12} \\
  \hat{a}_{21} & -\delta
\end{bmatrix}
\begin{bmatrix}
  c & s \\
  s & c
\end{bmatrix} + \alpha I_2,
\]

\[
= \begin{bmatrix}
  \delta(c^2-s^2) - 2\delta sc & \delta(c^2+s^2) - \hat{a}_{12}c^2 - \hat{a}_{21}s^2 \\
  2\delta sc - \hat{a}_{12}s^2 + \hat{a}_{21}c^2 & -\delta(c^2-s^2) + 2\delta sc
\end{bmatrix} + \alpha I_2,
\]

where

\[ \hat{a} = (\hat{a}_{12} + \hat{a}_{21})/2. \]

The proper choice of \( c = \cos \theta \) is therefore given by

\[ \tan 2\theta = 2sc/(c^2-s^2) = \delta/\hat{a}. \]

So

\[ 2c^2 = 1 + \cos 2\theta = 1 + \hat{a}/v, \]

\[ v = \sqrt{\delta^2 + \hat{a}^2}, \]

\[ c = \sqrt{(1 + |\hat{a}|/v)/2}, \] to keep \(|\theta| < \pi/2\),

\[ s = \sin 2\theta/2 \cos \theta = \delta \text{ sign}(\hat{a})/2cv. \]
Finally

\[
C = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix} = \begin{bmatrix} cr_{11} & cr_{12} - sr_{22} \\ sr_{11} & sr_{12} + cr_{22} \end{bmatrix}.
\]

Our object now is to get rid of the intermediate quantities and express \( C \) in terms of \( d \) and \( M \). So

\[
\hat{a} = (\beta r_{11}^2 - \gamma (r_{12}^2 - r_{22}^2))/2r_{11}r_{22},
\]

\[
= \gamma [d^2 - (m_{12}^2 - \beta m_{11}^2)/\gamma]/2m_{11}d,
\]

\[\equiv \gamma \zeta/m_{11}, \text{ defining } \zeta,\]

\[
\delta = \gamma r_{12}/r_{11} = \gamma m_{12}/m_{11},
\]

\[\nu = |\gamma| \phi/m_{11} \text{ where } \phi \equiv \sqrt{\xi^2 + m_{12}^2}.
\]

Since \( \beta \gamma < 0 \) the expression

\[\omega^2 \equiv m_{12}^2 - \beta m_{11}^2/\gamma\]

is positive.

At the cost of an extra square root the important quantity \( \zeta \) can be written in a form which is attractive for finite precision computation

\[\zeta \equiv [d^2 - (m_{12}^2 - \beta m_{11}^2/\gamma)]/2d = (d-\omega)(d+\omega)/2d.
\]

Having computed \( d, M, \xi, \) and \( \phi \) we obtain the desired formulas:

\[\sigma = m_{11}/2d^2,\]

\[c_{11} = cr_{11} = \sqrt{\sigma(1 + |\xi|/\phi)},\]

\[c_{21} = sr_{11} = r_{11}^2 \delta \text{ sign}(\hat{a})/2\nu(cr_{11}) = \text{sign}(\xi)s_{m_{12}/\phi c_{11}},\]

\[c_{12} = cr_{12} - sr_{22} = (c_{11}m_{12} - c_{21}d)/m_{11},\]

\[c_{22} = sr_{12} + cr_{22} = (c_{21}m_{12} + c_{11}d)/m_{11}.\]
For completeness we note that

\[
\begin{pmatrix}
0 & \tilde{\beta} \\
\tilde{\gamma} & 0
\end{pmatrix} = 
\begin{pmatrix}
c_{11} & c_{12} \\
c_{21} & c_{22}
\end{pmatrix}
\begin{pmatrix}
0 & \beta \\
\gamma & 0
\end{pmatrix}
\begin{pmatrix}
c_{22} & -c_{12} \\
-c_{21} & c_{11}
\end{pmatrix} d
\]

so that

\[
\tilde{\beta} = (\beta c_{11}^2 - \gamma c_{12}^2) d,
\]

\[
\tilde{\gamma} = \beta \gamma / \tilde{\beta}.
\]

The matrix \( C \) is computed by the subprogram named CTCEQW (i.e. \( C^T C = W \)).

**Computation of \( C_2 \)**

The subprogram which computes \( C_1 \) from \( d, X, \beta, \gamma \) can also be used to compute \( C_2 \). Recall from (9) that

\[
C_2^T C_2 = (I + X^T X)^{-1}.
\]

By symmetry \( d^2 = \det(I + X^T X) = \det(I + XX^T) \). Moreover, from (12)

\[
A_2^T = C_2 A_2 C_2^{-1}.
\]

By transposing the data we can use the same formulas as given above for \( C_1 \). The data is \( d, X^T, \gamma_2, \beta_2 \) and the output will be \( C_2, \tilde{\gamma}_2, \tilde{\beta}_2 \). In other words it is only the interpretation of the parameters which distinguishes the computation of \( C_2 \) from that of \( C_1 \).

**7. Performing the Similarity Transformations**

In practice \( A_1 \) and \( A_2 \) will be contiguous submatrices on the diagonal of some big block triangular matrix. The similarity transformation determined by \( P \) affects elements in the same row or column as those of \( A_1 \) and \( A_2 \) as indicated in the figure.
Figure 1
Let those elements in a typical column which are altered by the premultiplication by $P$ be partitioned conformably with $P$ as $\begin{pmatrix} u \\ v \end{pmatrix}$. They will be transformed into

$$
P \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \xi_2 & 0 \\ 0 & \xi_1 \end{pmatrix} \begin{pmatrix} -x^T & \xi I \\ \xi I & X \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \xi_2 (\xi v - x^T u) \\ \xi_1 (\xi u + x v) \end{pmatrix}.
$$

Notice that the number of multiplications required to effect this is $pq$ for each of $x^T u$ and $x v$ plus $q^2$ and $p^2$ for the application of $C_2$ and $C_1$. This is the same as for multiplication by the full, non-factored version of $P$ except for the $(p+q)$ multiplications involving $\xi$.

There is a surprising difficulty in writing a program to effect this. The program must work for any values of $p$ and $q$ and this condition prevents us from supplying the input data as values; they must be names or references since the number of them, $p+q$, is not known at compile time. In other words the subprogram is informed that elements $m+1$ through $m+p+q$ of an array $Y$ are to be transformed.

The disadvantage of this constraint is that the same code cannot be used for effecting the postmultiplication by $P^T$. More precisely, the price of using the same code for both cases is a loss in elegance and efficiency. The difficulty can be seen clearly by looking at the listings of the subprograms NEWCOL and NEWROW. They differ only where a variable $Y[i,k]$ in NEWCOL corresponds to a variable $Y[k,i]$ in NEWROW.
8. Gaussian Elimination for Solving $A_1X - XA_2 = B$

The linear equations defining $X$ can also be solved by block Gaussian elimination in about half the time required by the algorithm just described. Three different factorizations are appropriate (i.e. stable).

**Case 1:** $\delta^2 \gg \max(-\beta_1\gamma_1, -\beta_2\gamma_2)$

$$
\begin{pmatrix}
I_2 & 0 \\
\gamma_1C^{-1} & I_2
\end{pmatrix}
\begin{pmatrix}
C & \beta_1I_2 \\
0 & C - \beta_1\gamma_1C^{-1}
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2
\end{pmatrix},
\begin{pmatrix}
b_1 \\
b_2
\end{pmatrix}
= 
\begin{pmatrix}
b_{11} \\
b_{12}
\end{pmatrix},
\begin{pmatrix}
b_2 \\
b_{22}
\end{pmatrix}
= 
\begin{pmatrix}
b_{21} \\
b_{22}
\end{pmatrix}
$$

**Case 2:** $|\gamma| > |\beta_1| \gg \max(\delta^2, -\beta_2\gamma_2)$

$$
\begin{pmatrix}
I_2 & 0 \\
\gamma_1I_2 & 0
\end{pmatrix}
\begin{pmatrix}
C \\
-(C^2 - \beta_1\gamma_1)
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2
\end{pmatrix}
$$

**Case 3:** $|\gamma_2| > |\beta_2| \gg \max(\delta^2, -\beta_1\gamma_1)$

$$
\begin{pmatrix}
I_2 & 0 \\
-\gamma_2 & C
\end{pmatrix}
\begin{pmatrix}
\gamma_2^{-1} & 0 \\
\gamma_2^{-1}
\end{pmatrix}
\begin{pmatrix}
\hat{x}_1 \\
\hat{x}_2
\end{pmatrix}
= 
\begin{pmatrix}
\hat{b}_2 \\
\hat{b}_1
\end{pmatrix},
\begin{pmatrix}
\hat{x}_1 \\
\hat{x}_2
\end{pmatrix}
= 
\begin{pmatrix}
x_{11} \\
x_{12}
\end{pmatrix},
\begin{pmatrix}
\hat{b}_1 \\
\hat{b}_2
\end{pmatrix}
= 
\begin{pmatrix}
b_{11} \\
b_{21}
\end{pmatrix},
\begin{pmatrix}
\hat{b}_2 \\
b_{22}
\end{pmatrix}
= 
\begin{pmatrix}
b_{12} \\
b_{22}
\end{pmatrix}
$$

In each case $X$ can be found with 16 multiplications and 4 divisions. Further rearrangements should be made when $|\beta_1| > |\gamma_1|$ in Case 2, $|\beta_2| > |\gamma_2|$ in Case 3.

The extra length of the code (100 statements versus 50) does not appear to warrant a saving of 16 multiplications.
9. Swapping Large Blocks

The algorithm we developed for swapping was quite general with $A_1 \ p \times p$ and $A_2 \ q \times q$. However the individual subroutines TXMXT, CTCEQW, NEWCOL, and NEWROW were specialized for $p \leq 2$, $q \leq 2$. Here we want to point out that general versions of these programs are readily produced.

1. $A_1 X - XA_2 = B$ can be solved for $X$ by the algorithm of Bartels and Stewart [B and S, 1971]. In our case $A_1$ and $A_2$ are already in real Schur form and $X$ can be partitioned to match $A_1$ and $A_2$. If the equations defining $X$ are taken in the proper order the system is triangular and can be solved by

$$A_{k\ell}^{(1)} X_{k\ell} - X_{k\ell} A_{\ell\ell}^{(2)} = B_{k\ell} - \sum_{j=k+1}^{p} A_{kj}^{(1)} X_{j\ell} + \sum_{i=1}^{q-1} X_{ki} A_{i\ell}^{(2)}.$$  

The proper order is $k = \bar{p}, \bar{p}-1, \ldots, 1; \ell = 1, 2, \ldots, \bar{q}$. Here $\bar{p}$ and $\bar{q}$ are the block orders of $A_1$ and $A_2$.

2. $C^T C = (\xi^2 + XX^T)^{-1}$. The positive definite matrix $\xi^2 + XX^T$ can be formed explicitly and its Choleski factorization $R^T R$ computed in a standard manner. Then $R^T$ can be overwritten with its inverse to give a solution $\hat{C}$.

3. The execution of the orthogonal similarity transformation, in factored form

$$\begin{pmatrix}
\hat{C}_1 & 0 \\
0 & \hat{C}_2
\end{pmatrix}
\begin{pmatrix}
-X^T & \xi \\
\xi & X
\end{pmatrix}$$

presents no difficulties.

We mention this possibility only to reject it. The rival method is simply to swap $A_1$ and $A_2$ subblock by subblock, using the programs which we have presented here, that is by swapping many $1 \times 1$'s and $2 \times 2$'s. The
operation count for each method is approximately \((p+q)^2n\) multiplications and additions but the general procedure sketched above would require significantly more program statements.

In the language of computer science we are recommending the recursive swapping of big blocks.
10. Test Results

(a) $6 \times 6$ Matrix (Separated Eigenvalues)*

1. Original Matrix

\[
\begin{bmatrix}
2.0000 & 3.0000 & 4.0000 & 5.0000 & 6.0000 & 7.0000 \\
-1.0000 & 2.0000 & 5.0000 & 6.0000 & 7.0000 & 8.0000 \\
6.0000 & 7.0000 & 8.0000 & 9.0000 \\
8.0000 & 9.0000 & 10.0000 \\
12.0000 & 11.0000 \\
-1.0000 & 12.0000
\end{bmatrix}
\]

2. Swap 1st and 2nd blocks, $2 \times 1$ case

\[
\begin{bmatrix}
2.2 \times 10^{-14} & 2.0000 & 3.2930 & -3.8212 & -4.3070 & -4.7929 \\
-0.91103 & 2.0000 & -1.3978 & -1.4569 & -1.5161 \\
8.0000 & 9.0000 & 10.0000 \\
12.0000 & 11.0000 \\
-1.0000 & 12.0000
\end{bmatrix}
\]

3. Swap 3rd and 4th blocks, $1 \times 2$ case

\[
\begin{bmatrix}
2.2 \times 10^{-14} & 2.0000 & 3.2930 & -4.8713 & 5.1785 & -2.3615 \\
-0.91103 & 2.0000 & -1.5437 & 1.8492 & -0.75651 \\
12.0000 & 0.69449 & 5.5837 \\
-15.839 & 12.000 & -4.5244 \\
4.2 \times 10^{-14} & -1.3 \times 10^{-14} & 8.0000
\end{bmatrix}
\]

*Computations performed on 14 digit machine, results rounded to 5 figures for display.
4. Swap 2nd and 3rd blocks, $2 \times 2$ case

\[
\begin{bmatrix}
2.2 \times 10^{-14} & 12.000 & 0.63866 & -3.6006 & 0.39991 & 5.3584 \\
-17.224 & 12.000 & -0.21201 & 0.40812 & -5.1223 \\
-1.0 \times 10^{-13} & 4.6 \times 10^{-14} & 2.0000 & 2.7985 & -1.6498 \\
-4.7 \times 10^{-14} & 2.5 \times 10^{-14} & -1.0720 & 2.0000 & -0.35352 \\
& & 4.2 \times 10^{-14} & -1.3 \times 10^{-14} & 8.0000
\end{bmatrix}
\]

(b) $6 \times 6$ Matrix (Close Eigenvalues)*

1. Original Matrix

\[
\begin{bmatrix}
6.0000 & 10^{-4} & 4.0000 & 5.0000 & 6.0000 & 7.0000 \\
-1.0000 & 6.0000 & 5.0000 & 6.0000 & 7.0000 & 8.0000 \\
& & 6.0000 & 7.0000 & 8.0000 & 9.0000 \\
& & & 6.0001 & 9.0000 & 10.000 \\
& & & & 6.0001 & 10^{-4} \\
& & & & -1.0000 & 6.0001
\end{bmatrix}
\]

2. Swap 1st and 2nd blocks, $2 \times 1$ case

\[
\begin{bmatrix}
6.0000 & 0.99984 & -4.9995 & -5.9992 & -6.9990 & -7.9989 \\
6.0000 & 4.0006 & 5.0010 & 6.0011 & 7.0013 \\
-2.4996 \times 10^{-5} & 6.0000 & 7.0000 & 8.0000 & 9.0000 \\
& & 6.0001 & 9.0000 & 10.000 \\
& & & 6.0001 & 10^{-4} \\
& & & -1.0000 & 6.0001
\end{bmatrix}
\]

*Computations performed on 14 digit machine, results rounded to 5 figures for display.
3. Swap 3rd and 4th blocks, 1x2 case

\[
\begin{bmatrix}
6.0000 & 0.99984 & -4.9995 & -7.9996 & 5.9992 & 6.9983 \\
6.0000 & 4.0006 & 7.0019 & -5.0010 & -6.0004 \\
-2.4996 \times 10^{-5} & 6.0000 & 9.0008 & -7.0000 & -7.9991 \\
 & 6.0001 & 9.9992 \times 10^{-6} & 0.99992 \\
 & -10.001 & 6.0001 & 8.9991 \\
 & 3.9 \times 10^{-18} & -4.3 \times 10^{-19} & 6.0001 \\
\end{bmatrix}
\]

4. Swap 2nd and 3rd blocks, 2x2 case

\[
\begin{bmatrix}
6.0000 & -4.9997 & -0.99972 & -5.9991 & -7.995 & 6.9983 \\
6.0001 & 2.4995 \times 10^{-5} & 7.0000 & 9.0008 & -7.9992 \\
-4.0008 & 6.0001 & -5.0011 & -7.0020 & 6.0006 \\
-2.1 \times 10^{-23} & -6.6 \times 10^{-24} & 6.0000 & 10.001 & -8.9989 \\
8.9 \times 10^{-24} & -3.8 \times 10^{-25} & -9.9994 \times 10^{-6} & 6.0000 & 1.0000 \\
 & 3.9 \times 10^{-18} & -4.3 \times 10^{-19} & 6.0001 \\
\end{bmatrix}
\]
11. Program Listing

SUBROUTINE SWAP(NM,N,T,P,J1,L1,L2)
DIMENSION T(NM,N),P(NM,N)
REAL X(2,2),C1(2,2),C2(2,2),Y(2,2)
EQUIVALENCE (X(1,1),Y(1,1)),(C1(1,1),Y(1,1)),(C2(1,1),Y(1,1))
C EXCHANGE ADJACENT DIAGONAL BLOCKS T1 AND T2 BEGINNING IN ROW J1 BY
C ORTHOGONAL SIMILARITY TRANSFORMATIONS, RECORDERD IN P, PRESERVING THE
C TRIANGULAR FORM OF T. BLOCK T1 IS 1 BY L1, T2 IS L2 BY L2.
J3=J1+1
J2=J3-1
J4=J2+1
Z = 1.0
C******************************CLEAR THE (2,1) BLOCK.
DO 5 J=1,1
5 T(J2-1+J1+J-1)=0.
C******************************SOLVE FOR X IN T1*X-X2*T2=X1*T1, WHERE T12 IS L1 BY L2.
CALL TXMXT(NM,N,T,J1,J3,L1,L2,T,Y,X)
IF (Z EQ 0.0) RETURN
C******************************COMPUTE C1 WHERE: C1**T*C1 = (ZSQ**I + X**XT)**-1.
CALL CTFGW(Z,X(1,1),X(2,1),X(1,2),X(2,2),T(J1,J2),T(J2,J1),C1,L1)
CALL CTFGW(Z,X(1,1),X(1,2),X(2,1),X(2,2),T(J3,J4),T(J4,J3),C2,L2)
C******************************PERFORM TRANSFORMATION ON COLUMNS AND ROWS OF T.
C******************************UPDATE P.
CALL NEWVFEC(Z,L1,L2,Y,T,NM,N,J1,1)
CALL NEWVFEC(Z,L1,L2,Y,T,NM,N,J1,1)
C******************************PERFORM QUALITY OF DIAGONAL ELEMENTS IN BLOCKS.
IF (L2 EQ 0.0) T(J4-J1,J4-J1)=(T(J4,J4) + T(J4,J4))/2.
RETURN
END

SUBROUTINE CTFGW(Z,X11,X21,X12,X22,BETA,GAM,C,L)
DIMENSION C(2,2)
C FIND AN APPROPRIATE SOLUTION C TO CT*C = W = (ZSQ**I + X**XT)**-1.
ZSQ = Z**2
IF (L EQ 1) GO TO 10
C(1,1)=1.0/SQRT(ZSQ+X11*X11+X21*X21+X12*X12)
RETURN
10 EM11=ZSQ+X11*X11+X21*X21
EM12=-(X11*X21+X12*X22)
D=EM11*(ZSQ+X11*X11+X21*X21+X12*X12)-EM12*X22
RTD=SQRT(D)
EGA=SQRT(EM12**2-BETA*EM11**2/GAM)
ZETA=(RTD-EGA)*(RTD+EGA)/(2.*RTD)
PHI=SQRT(ZETA**2+EM12**2)
FAC=EM11/(2.*RTD)
C(1,1)=SORT(FAC*(1.0+BETA*ZETA/PHI))
C(2,1)=SORT((1.0+BETA*ZETA/PHI))
C(1,2)=C(1,1)*EM12*C(2,1)/EM11
C(2,2)=(C(2,1)*EM12+C(1,1)*RTD)/EM11
RETURN
END
SUBROUTINE TXMXT(NM,N,T,J1,J2,L1,L2,P,R,Z,X)
DIMENSION T(NM,N),P(NM,N),X(L1,L2)
C
C SOLVE FOR L1 BY L2 MATRX X IN T*X = X*T2 = #%#.
C T1 AND T2 BEGIN IN DOWS J1 AND J2, Z IS GIVEN ON ENTRY.
C BUT ON EXIT Z IS CHANGED TO ENSURE NORM(X) < 1.
X(J1,J2) = X(J1,J2) + X(J1,J2) = T(J1+1,J2)

IF (Z=0.) RETURN

D01 = T(J1,J1) - T(J2,J2)
K = 2*L1 + 2*J2 + 1
C
C **************************** T1 AND T2 HAVE THE SAME EIGENVALUES, RETURN Z=0.
C
Z = C.
C
RETURN
C
C **************************** DETERMINE DIMENSIONS OF SOLUTION X.
C
GO TO (1C, 2C, 3C, AC), K.
C
C **************************** T1 IS 1 BY 1, T2 IS 1 BY 1.
10 XMAX = ABS(R(J1,J2))
D = ABS(S/DL1)
X(J1,J1) = R(J1,J2)*ABS(X(J1,J1)), ABS(X(J1,J2))
GO TO 70
C
C **************************** T1 IS 1 BY 1, T2 IS 2 BY 2.
20 D = DEL**2 - T(J2,J2+1)*T(J2+1,J2)
X(J1,J1) = DEL*R(J1,J2+1)*DEL*R(J1,J2)*T(J2,J2+1)*T(J2+1,J2)
X(J1,J2) = DEL*R(J1,J2)*T(J2,J2+1)*T(J2+1,J2)
XMAX = MAX1(ABS(X(J1,J1)), ABS(X(J1,J2)))
GO TO 50
C
C **************************** T1 IS 2 BY 2, T2 IS 1 BY 1.
30 D = DL1**2 - T(J1,J1+1)*T(J1+1,J1)
X(J1,J1) = DL1*R(J1,J2)*R(J1,J2+1)*R(J1+1,J1+1)*T(J1+1,J1+1)
X(J1,J2) = DEL*R(J1,J2)*R(J1,J2+1)*DEL*R(J1+1,J1+1)*T(J1+1,J1+1)
XMAX = MAX1(ABS(X(J1,J1)), ABS(X(J1,J2)))
GO TO 50
C
C **************************** T1 IS 2 BY 2, T2 IS 2 BY 2.
40 BET1 = T(J1,J1+1)
BET2 = T(J2,J2+1)
GAM1 = T(J1,J1+1)
GAM2 = T(J2,J2+1)
P1 = BET1*GAM1
P2 = BET2*GAM2
DSQ = D**2
E = DL1**2 - (P1**2 + P2**2) + AMIN1(DSQ,-P1)
F2 = (P2**2 + AMIN1(DSQ,-P1)) + AMIN1(DSQ,-P2)
H = 2.0*DEL*BET2
G = 2.0*DEL*GAM2
D = 2.0*DEL*GAM2
IF (D < EPS) GC TO A
P1 = R(J1,J1+1)
R21 = R(J1+1,J1)
R22 = R(J1+1,J2+1)
X(J1,J1) = R11**2 + R12**2 + GAM2**2 + F1**2 + R21**2 + R22**2
X(J1,J2) = R11**2 + R12**2 + GAM2**2 + F1**2 + R21**2 + R22**2
X(J2,J2) = R11**2 + R12**2 + GAM2**2 + F1**2 + R21**2 + R22**2
XMAX = MAX1(ABS(X(J1,J1)), ABS(X(J1,J2)), ABS(X(J2,J2)))
GO TO 50
C
C **************************** ENSURE NORM(X) < 1.
50 E = 2.0*MAX1(XMAX, 0)
D0 50 T1,J1).
50 DO 60 J1 = 1, L1
60 IF (X(J1,J1) < EPS) GC TO 50
C
C END
SUBROUTINE NEWVEC (F, N1, N2, Y, NM, N, M1, INC)
DIMENSION Y(1), (1(1, 1), W(1))
C
C COMPUTE C2E (F = V - XT*U) = NEWV - C1E (F = U + X*V) = NEWV
C WHERE U IS N1 BY N1, C2 IS N2 BY N2, AND (U, V) = (Y(M1), Y(M1+1)), etc.
C THE MATRICES X, C1, C2 ARE STORED IN Z. F IS SCALING FACTOR FOR X.
C WHEN INC = 1,
C PERFORM BLOCK EXCHANGE TRANSFORMATION ON COLUMNS M1 TO N OF Y.
C IN THE N1+N2 ROWS STARTING WITH M1.
C NY = (M1-1)*M1 = M1 - 1 + (M1-1)*NM = (M1-1)*(NM+1), LR = M1, J = NM,
C WHEN INC = NM,
C PERFORM BLOCK EXCHANGE TRANSFORMATION IN FIRST N ROWS OF Y.
C IN THE N1+N2 COLUMNS STARTING WITH COLUMN M1.
C NY = (1, M1-1) = 1 + (M1-2)*NM = (M1-1)*NM+1 - NW, LR = 1, J = 1.
I = INC
LB = J = NM - I + 1
IF (I > L) RETURN
NY = (M1-1)*NM + L - 1
I = INC
GO TO (10, 20, 30, 40, 50)
C
C******************N1 = 1, N2 = 1
10 DC 15 K = LB, N
W(1) = Y(NY+2*X1) - Z(1, 1) * Y(NY+1)
W(2) = Y(NY+1) * Z(1, 1) * Y(NY+2)
W(3) = Z(1, 5) * W(1)
Y(NY+2) = Z(1, 3) * W(2)
NY = NY + 1
RETURN
C******************N1 = 2, N2 = 1
20 DC 28 K = LB, N
W(1) = Y(NY+3*X1) - Z(1, 1) * Y(NY+1)
W(2) = Y(NY+1) * Z(1, 2) * Y(NY+1)
W(3) = Z(1, 5) * W(1) + Z(1, 4) * W(2)
Y(NY+2) = Z(2, 3) * W(2) + Z(2, 4) * W(3)
NY = NY + 1
RETURN
C******************N1 = 1, N2 = 2
30 DC 35 K = LB, N
W(1) = Y(NY+2*X1) - Z(1, 1) * Y(NY+1)
W(2) = Y(NY+1) * Z(1, 2) * Y(NY+1)
W(3) = Z(1, 5) * W(1) + Z(1, 4) * W(2)
Y(NY+2) = Z(1, 3) * W(2)
NY = NY + 1
RETURN
C******************N1 = 2, N2 = 2
40 DC 45 K = LB, N
W(1) = Y(NY+3*X1) - Z(1, 1) * Y(NY+1) - Z(2, 1) * Y(NY+2)
W(2) = Y(NY+1) * Z(1, 2) * Y(NY+1) - Z(2, 2) * Y(NY+2)
W(3) = Z(1, 5) * W(1) + Z(1, 4) * W(2)
Y(NY+2) = Z(2, 3) * W(2) + Z(2, 4) * W(3)
NY = NY + 1
RETURN
END
Alternative, but less efficient, version of NEWVEC which better illustrates
the column and row operations.

SUBROUTINE NEWCCL(F,N1,N2,Z,Y,NM,N,M1,I)
DIMENSION Y(NM,N),Z(2,6),W(4)
C
C PERFORM BLOCK EXCHANGE TRANSFORMATION ON COLUMNS M1 TO N OF Y
C IN THE N1+N2 ROWS STARTING WITH M1.
C COMPUTE C2*(F*V - X*T*U) = NEWV ; C1*(F*U + X*V) = NEWV
C WHERE C1 IS N1 BY N1, C2 IS N2 BY N2, AND (U,V)=(Y(M1),Y(M1+1),...).
C THE MATRICES X, C1, C2 ARE STORED IN Z. F IS SCALING FACTOR FOR X.
M=M1-1
DO 50 K=M1,N
D0 10 J=1,N2
W(J) = Y(M+K+1,J)*F
10 DO 10 L=1,N1
W(J) = W(J) - Z(L,J)*Y(K+M+L)
DO 20 J=1,N1
W(N2+J) = W(N2+J) - Z(L,J)*Y(K+M+L)
20 DO 35 J=1,N2
S=0
35 W(J) = W(J) + Z(J,L)*Y(K+M+L)
DO 50 J=1,N2
S=0
50 CONTINUE
RETURN
END

SUBROUTINE NEWROW(F,N1,N2,Z,Y,NM,N,M1,I)
DIMENSION Y(NM,N),Z(2,6),W(4)
C
C PERFORM BLOCK EXCHANGE TRANSFORMATION ON FIRST N ROWS OF Y
C IN THE N1+N2 COLUMNS STARTING WITH COLUMN M1.
C COMPUTE C2*(F*V - X*T*U) = NEWV ; C1*(F*U + X*V) = NEWV
C WHERE C1 IS N1 BY N1, C2 IS N2 BY N2, AND (U,V)=(Y(M1),Y(M1+1),...).
C THE MATRICES X, C1, C2 ARE STORED IN Z. F IS SCALING FACTOR FOR X.
M=M1-1
DO 50 K=1,N
D0 10 J=1,N1
W(J) = Y(K+M+1,J)*F
10 DO 10 L=1,N1
W(J) = W(J) - Z(L,J)*Y(K+M+L)
DO 20 J=1,N1
W(N2+J) = W(N2+J) - Z(L,J)*Y(K+M+L)
20 DO 35 J=1,N2
S=0
35 W(J) = W(J) + Z(J,L)*Y(K+M+L)
DO 50 J=1,N2
S=0
50 CONTINUE
RETURN
END
References
