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

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 θ 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 α_1 and α_2. The new (1,2) element is

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

Now suppose that α_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 α_1 and α_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 θ in $(-\pi/2,\pi/2)$ we define

$$\begin{aligned} d &= \sqrt{(\alpha_1-\alpha_2)^2+\beta^2} \ , \\ c &= |\alpha_1-\alpha_2|/d \ , \\ s &= \beta \ \mathrm{sign}(\alpha_1-\alpha_2)/d \ . \end{aligned}$$

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

$$\begin{pmatrix} A_1 & B \\ 0 & A_2 \end{pmatrix}, \quad \begin{matrix} A_1 \text{ is } p \times p \ , \\ A_2 \text{ is } q \times q \ . \end{matrix}$$

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

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

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

$$(1) \qquad C_1C_1^T + S_2S_2^T = I_p = S_1S_1^T + C_1^TC_1 \ ,$$

$$(2) \qquad S_1^TS_1 + C_2C_2^T = I_q = C_2^TC_2 + S_2^TS_2 \ ,$$

$$(3) \qquad -\ C_1S_1 + S_2C_2^T = 0_{p,q} \ ,$$

$$(4) \qquad -\ S_1C_2 + C_1^TS_2 = 0_{q,p} \ .$$

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

$$(5) \qquad C_1A_1 = \tilde{A}_1C_1 \qquad (\text{also } A_2C_2^T = C_2^T\tilde{A}_2) \ ,$$

$$(6) \qquad C_1B + S_2A_2 = \tilde{A}_1S_2 \ .$$

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

$$\begin{aligned} B + C_1^{-1}S_2A_2 &= C_1^{-1}\tilde{A}_1S_2 \\ &= A_1C_1^{-1}S_2 \ , \quad \text{by (5)} \ . \end{aligned}$$

We now let the $p \times q$ matrix $C_1^{-1}S_2 = X/\xi$, where ξ is a positive constant at our disposal, and substitute into the equation above to get

(7) $$A_1X - XA_2 = \xi B \; .$$

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

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

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

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

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 ξ 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 $\mathring{C}_1 = C_1/\xi$, the scaled version of C_1. Then

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

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

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

is effected. For completeness we give the formulas

$$(12) \qquad \begin{aligned} S_1 &= X\mathring{C}_2^T \quad , \quad S_2 = \mathring{C}_1 X \quad , \\ \tilde{A}_1 &= \mathring{C}_1 A_1 \mathring{C}_1^{-1} \quad , \quad \tilde{A}_2^T = \mathring{C}_2 A_2^T \mathring{C}_2^{-1} \quad , \\ \tilde{B} &= \tilde{A}_2 C_2^{-T} S_2^T - S_1^T C_1^{-1} \tilde{A}_1 = C_2^{-T} A_2 S_2^T - S_1^T A_1 C_1^{-1} \ . \end{aligned}$$

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 ξ using subroutine TXMXT. ξ is chosen so that $\|X\| \doteq 1$
3. If $\xi = 0$ then exit.
4. Solve $\mathring{C}_1^T\mathring{C}_1 = (\xi^2+XX^T)^{-1} \equiv W_1$ for $\mathring{C}_1$ using CTCEQW.
5. Solve $\mathring{C}_2^T\mathring{C}_2 = (\xi^2+X^TX)^{-1} \equiv W_2$ for $\mathring{C}_2$ 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.

Name	Executable Statements	Count for 2×2 Case
SWAP	17	32n multiplications
TXMXT	52	42 multiplications
CTCEQW	17	32 multiplications, 4 square roots
NEWCOL	22	16 multiplications per column
NEWROW	22	16 multiplications per row

5. Solving $A_1X - XA_2 = B$

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

$$(1)\quad \begin{pmatrix} C & \beta_1 I_2 \\ \gamma_1 I_2 & C \end{pmatrix} \underset{\sim}{x} = \underset{\sim}{b} \; ; \quad \underset{\sim}{x} = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \end{pmatrix} , \quad \underset{\sim}{b} = \begin{pmatrix} b_{11} \\ b_{12} \\ b_{21} \\ b_{22} \end{pmatrix}$$

where

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

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

$$(3)\qquad \begin{pmatrix} C^2-\beta_1\gamma_1 & 0 \\ 0 & C^2-\beta_1\gamma_1 \end{pmatrix} \underset{\sim}{x} = \begin{pmatrix} C & -\beta_1 \\ -\gamma_1 & C \end{pmatrix} \underset{\sim}{b} .$$

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

$$(4)\qquad \tau = \delta^2 + \beta_2\gamma_2 - \beta_1\gamma_1 , \qquad d = \tau^2 - (2\delta\beta_2)(2\delta\gamma_2) ,$$

and premultiply (3) $\mathrm{diag}(G,G)$ to find

$$x = \begin{pmatrix} G & 0 \\ 0 & G \end{pmatrix} \begin{pmatrix} C & -\beta_1 \\ -\gamma_1 & C \end{pmatrix} \underset{\sim}{b}$$

$$(5)\qquad = \left(\begin{array}{c|c} G & 0 \\ \hline 0 & G \end{array}\right) \begin{pmatrix} \delta b_{11} - \gamma_2 b_{12} - \beta_1 b_{21} \\ -\beta_2 b_{11} + \delta b_{12} - \beta_1 b_{22} \\ -\gamma_1 b_{11} + \delta b_{21} - \gamma_2 b_{22} \\ -\gamma_1 b_{12} - \beta_2 b_{21} + \delta b_{22} \end{pmatrix} ,$$

$$= \begin{bmatrix} \phi\delta b_{11} + (2\delta^2-\tau)\gamma_2 b_{12} - & \tau\beta_1 b_{21} - & 2\delta\gamma_2\beta_1 b_{22} \\ (2\delta^2-\tau)\beta_2 b_{11} + & \phi\delta b_{12} - & 2\delta\beta_1\beta_2 b_{21} - & \tau\beta_1 b_{22} \\ -\tau\gamma_1 b_{11} - & 2\delta\gamma_1\gamma_2 b_{12} + & \phi\delta b_{21} + (2\delta^2-\tau)\gamma_2 b_{22} \\ -2\delta\gamma_1\beta_2 b_{11} - & \tau\gamma_1 b_{12} + (2\delta^2-\tau)\beta_2 b_{21} + & \phi\delta b_{22} \end{bmatrix} / d ,$$

$$\equiv \underset{\sim}{y}/d$$

defining $\underset{\sim}{y}$, where

$$\phi = \tau - 2\gamma_2\beta_2 = \delta^2 - (\beta_1\gamma_1+\beta_2\gamma_2) > 0 ,$$
$$\psi = 2\delta^2 - \tau = \delta^2 + (\beta_1\gamma_1-\beta_2\gamma_2) .$$

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 $\alpha_1 = \alpha_2$, $\beta_1\gamma_1 = \beta_2\gamma_2$.

Among all the coefficients in the linear combinations of the elements of B which are given above only τ and ψ 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) $\tau = \delta^2 + \beta_2\gamma_2 - \beta_1\gamma_1$ and if either of δ^2 or $-\beta_1\gamma_1$ is tiny compared with the other two terms we want to add it in last. Similarly for ψ. Thus we use

$$(6) \qquad \begin{aligned} \psi &= (\beta_1\gamma_1 + \max\{\delta^2, -\beta_2\gamma_2\}) + \min\{\delta^2, -\beta_2\gamma_2\} , \\ \tau &= (\beta_2\gamma_2 + \max\{\delta^2, -\beta_1\gamma_1\}) + \min\{\delta^2, -\beta_1\gamma_1\} . \end{aligned}$$

Here is an example for a machine with a relative precision of 8 decimals, i.e. the floating point result $fl(10^8-9)$ is 10^8 whereas $fl(10^8-10)$ is $10^8 - 10 = 10(10^7-1)$. Let $\delta^2 = 9$, $\beta_1\gamma_1 = -(10^8-10)$, $\beta_2\gamma_2 = -10^8$ then

from (4), computing from the left, $\tau = fl(fl(9-10^8) + 10^8 - 10) = -10$,

from (6), computing from the left, $\tau = fl(fl(-10^8 + (10^8-10) + 9) = -1$.

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

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

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 ξ so that

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

Equation (6) above yields $\underset{\sim}{y}$ so that the corresponding 2×2 matrix Y satisfies

$$A_1 Y - Y A_2 = dB$$

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

Case 1: $\eta \leq d$, take $\underset{\sim}{x} = \underset{\sim}{y}/d$, $\xi = 1$.

Case 2: $\eta > d$, take $\underset{\sim}{x} = \underset{\sim}{y}/\eta$, $\xi = d/\eta$.

The Algorithm for TXMXT

We solve $A_1X - XA_2 = \xi B$ when A_1 and A_2 are 2×2 standardized matrices as follows:

$$
\begin{aligned}
\delta &= \alpha_1 - \alpha_2 , \qquad \delta sq = \delta^2 ,\\
\pi_1 &= \beta_1\gamma_1 \quad , \qquad \pi_2 = \beta_2\gamma_2 ,\\
e &= \phi\delta = \delta(\delta sq - (\pi_1+\pi_2))\\
f_1 &= \psi = (\pi_1 + \max\{\delta sq,-\pi_2\}) + \min\{\delta sq,-\pi_2\} ,\\
f_2 &= \tau = (\pi_2 + \max\{\delta sq,-\pi_1\}) + \min\{\delta sq,-\pi_1\} ,\\
g &= 2\delta\gamma_2 \quad , \qquad h = 2\delta\beta_2 ,\\
d &= f_2 - gh .
\end{aligned}
$$

At this point $\underset{\sim}{y}$ can be evaluated from (5). Then

$$
\begin{aligned}
\eta &= \|\underset{\sim}{y}\|_\infty ,\\
\underset{\sim}{x} &= \xi\underset{\sim}{y}/\max(d,\eta) ,\\
\text{new } \xi &= \xi\cdot d/\max(d,\eta) .
\end{aligned}
$$

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), $A_1 X - XA_2 = \xi B$. Next define

$$\hat{A}_1 \equiv R_1 A_1 R_1^{-1} , \qquad \hat{A}_2 \equiv R_2^{-T} A_2 R_2^T .$$

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

$$\tilde{A}_1 \equiv J_1 \hat{A}_1 J_1^T \quad \text{and} \quad \tilde{A}_2 \equiv J_2 \hat{A}_2 J_2^T$$

have equal diagonal elements. The appropriate C_1 and C_2 are therefore $\mathring{C}_1 = C_1/\xi = J_1 R_1$, $\mathring{C}_2 = C_2^T/\xi = J_2 R_2$.

Let us drop the subscript and dot from C_1 and A_1. The condition

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

imposes three quadratic relations on the four elements of C. If $A = \begin{pmatrix} \alpha & \beta \\ \gamma & \alpha \end{pmatrix}$ then the requirement that CAC^{-1} have equal diagonal elements (both α) imposes another quadratic constraint, namely

$$\beta c_{11} c_{21} = \gamma c_{12} c_{22} \ ,$$

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

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

Then define M by

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

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

$$\hat{A} = RAR^{-1}$$

$$= \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix} \begin{bmatrix} 0 & \beta \\ \gamma & 0 \end{bmatrix} \begin{bmatrix} r_{22} & -r_{12} \\ 0 & r_{11} \end{bmatrix} r_{11}^{-1} r_{22}^{-1} + \alpha I_2 ,$$

$$= \begin{bmatrix} \gamma r_{12} & (\beta r_{11}^2 - \gamma r_{12}^2)/r_{22} \\ \gamma r_{22} & -\gamma r_{12} \end{bmatrix} r_{11}^{-1} + \alpha I_2 ,$$

$$\equiv \begin{bmatrix} \delta & \hat{a}_{12} \\ \hat{a}_{21} & -\delta \end{bmatrix} + \alpha I_2 .$$

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

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

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

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

$$2c^2 = 1 + \cos 2\theta = 1 + \hat{a}/\nu ,$$

$$\nu = \sqrt{\delta^2 + \hat{a}^2} ,$$

$$c = \sqrt{(1 + |\hat{a}|/\nu)/2} , \quad \text{to keep } |\theta| < \pi/2 ,$$

$$s = \sin 2\theta / 2\cos\theta = \delta \operatorname{sign}(\hat{a})/2c\nu .$$

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

$$\begin{aligned} \hat{a} &= (\beta r_{11}^2 - \gamma[r_{12}^2 - r_{22}^2])/2r_{11}r_{22} , \\ &= \gamma[d^2 - (m_{12}^2 - \beta m_{11}^2/\gamma)]/2m_{11}d , \\ &\equiv \gamma\zeta/m_{11} , \quad \text{defining } \zeta, \\ \delta &= \gamma r_{12}/r_{11} = \gamma m_{12}/m_{11} , \\ \nu &= |\gamma|\phi/m_{11} \quad \text{where } \phi \equiv \sqrt{\xi^2 + m_{12}^2} . \end{aligned}$$

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 ζ 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, ξ, and ϕ we obtain the desired formulas:

$$\begin{aligned} \sigma &= m_{11}/2d^2 , \\ c_{11} &= cr_{11} = \sqrt{\sigma(1 + |\zeta|/\phi)} , \\ c_{21} &= sr_{11} = r_{11}^2\,\delta\,\mathrm{sign}(\hat{a})/2\nu(cr_{11}) = \mathrm{sign}(\xi)\sigma m_{12}/\phi c_{11} , \\ c_{12} &= cr_{12} - sr_{22} = (c_{11}m_{12} - c_{21}d)/m_{11} , \\ c_{22} &= sr_{12} + cr_{22} = (c_{21}m_{12} + c_{11}d)/m_{11} . \end{aligned}$$

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

$$\tilde\beta = (\beta c_{11}^2 - \gamma c_{12}^2) d \quad ,$$
$$\tilde\gamma = \beta\gamma/\tilde\beta \quad .$$

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, β, γ can also be used to compute C_2. Recall from (9) that

$$C_2^T C_2 = (I_2 + X^T X)^{-1} \ .$$

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

$$\tilde A_2^T = C_2 A_2^T C_2^{-1} \ .$$

By transposing the data we can use the same formulas as given above for C_1. The data is d, X^T, γ_2, β_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.

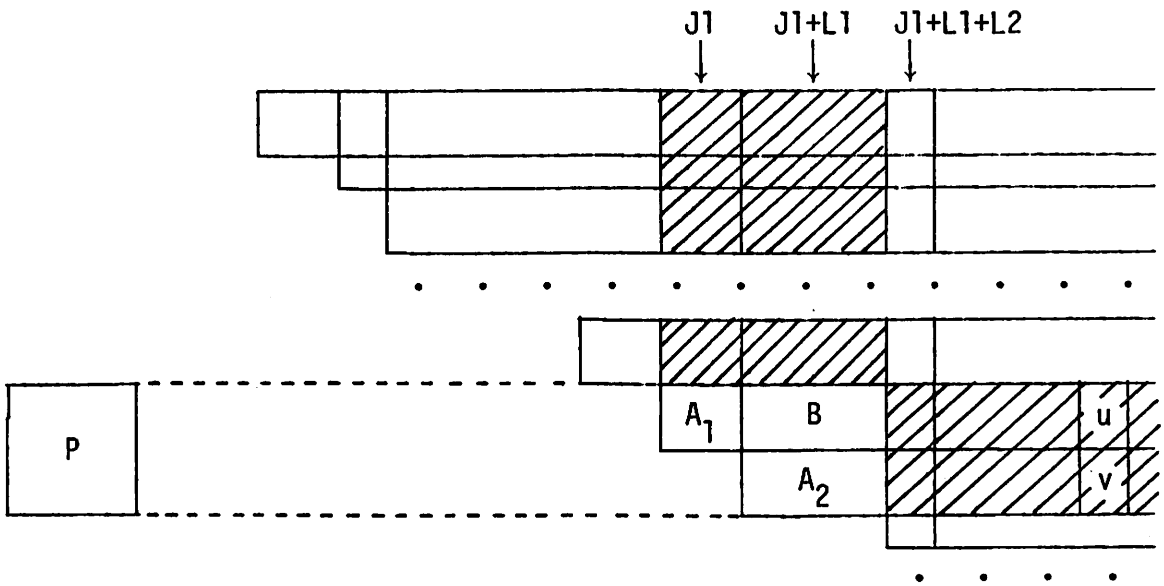

Figure 1

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

$$P\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \mathring{C}_2 & 0 \\ 0 & \mathring{C}_1 \end{bmatrix}\begin{bmatrix} -X^T & \xi I \\ \xi I & X \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

$$= \begin{bmatrix} \mathring{C}_2(\xi v - X^T u) \\ \mathring{C}_1(\xi u + Xv) \end{bmatrix} .$$

Notice that the number of multiplications required to effect this is pq for each of $X^T u$ and Xv 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 ξ.

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

$$\begin{pmatrix} I_2 & 0 \\ \gamma_1 C^{-1} & I_2 \end{pmatrix} \begin{pmatrix} C & \beta_1 I_2 \\ 0 & C-\beta_1\gamma_1 C^{-1} \end{pmatrix} \begin{pmatrix} \underset{\sim}{x}_1 \\ \underset{\sim}{x}_2 \end{pmatrix} = \begin{pmatrix} \underset{\sim}{b}_1 \\ \underset{\sim}{b}_2 \end{pmatrix}, \quad \underset{\sim}{b}_1 = \begin{pmatrix} b_{11} \\ b_{12} \end{pmatrix}, \quad b_2 = \begin{pmatrix} b_{21} \\ b_{22} \end{pmatrix}$$

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

$$\begin{pmatrix} I_2 & 0 \\ C\gamma_1^{-1} & \gamma_1^{-1} \end{pmatrix} \begin{pmatrix} \gamma_1 I_2 & C \\ 0 & -(C^2-\beta_1\gamma_1) \end{pmatrix} \begin{pmatrix} \underset{\sim}{x}_1 \\ \underset{\sim}{x}_2 \end{pmatrix} = \begin{pmatrix} \underset{\sim}{b}_2 \\ \underset{\sim}{b}_1 \end{pmatrix}$$

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

$$\begin{pmatrix} I_2 & 0 \\ -\hat{C}\gamma_2^{-1} & \gamma_2^{-1} \end{pmatrix} \begin{pmatrix} -\gamma_2 & \hat{C} \\ 0 & \hat{C}^2-\beta_2\gamma_2 \end{pmatrix} \begin{pmatrix} \hat{\underset{\sim}{x}}_1 \\ \hat{x}_2 \end{pmatrix} = \begin{pmatrix} \hat{\underset{\sim}{b}}_2 \\ \hat{b}_1 \end{pmatrix};$$

$$\hat{\underset{\sim}{x}}_1 = \begin{pmatrix} x_{11} \\ x_{12} \end{pmatrix}, \quad \hat{\underset{\sim}{x}}_2 = \begin{pmatrix} x_{12} \\ x_{22} \end{pmatrix}, \quad \hat{\underset{\sim}{b}}_1 = \begin{pmatrix} b_{11} \\ b_{21} \end{pmatrix}, \quad \hat{\underset{\sim}{b}}_2 = \begin{pmatrix} b_{12} \\ b_{22} \end{pmatrix}$$

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. $A_1X - XA_2 = B$ 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

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

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

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

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

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 1×1's and 2×2'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 × 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 × 1 case

$$\begin{bmatrix} 6.0000 & -4.2583 & -4.6036 & 9.6667 & 11.328 & 12.990 \\ 2.2\times10^{-14} & 2.0000 & 3.2930 & -3.8212 & -4.3070 & -4.7929 \\ & -0.91103 & 2.0000 & -1.3978 & -1.4569 & -1.5161 \\ & & & 8.0000 & 9.0000 & 10.0000 \\ & & & & 12.0000 & 11.0000 \\ & & & & -1.0000 & 12.0000 \end{bmatrix}$$

3. Swap 3rd and 4th blocks, 1 × 2 case

$$\begin{bmatrix} 6.0000 & -4.2583 & -4.6036 & 13.192 & -13.265 & 6.3656 \\ 2.2\times10^{-14} & 2.0000 & 3.2930 & -4.8713 & 5.1785 & -2.3615 \\ & -0.91103 & 2.0000 & -1.5437 & 1.8492 & -0.75651 \\ & & & 12.0000 & 0.69449 & 5.5837 \\ & & & -15.839 & 12.000 & -4.5244 \\ & & & 4.2\times10^{-14} & -1.3\times10^{-14} & 8.0000 \end{bmatrix}$$

*Computations performed on 14 digit machine, results rounded to 5 figures for display.

4. Swap 2nd and 3rd blocks, 2×2 case

$$\begin{bmatrix} 6.0000 & 13.402 & -14.062 & -1.3625 & -3.1781 & 6.3656 \\ 2.2\times10^{-14} & 12.000 & 0.63866 & -3.6006 & 0.39991 & 5.3584 \\ & -17.224 & 12.000 & -0.21201 & 0.40812 & -5.1223 \\ & -1.0\times10^{-13} & 4.6\times10^{-14} & 2.0000 & 2.7985 & -1.6498 \\ & -4.7\times10^{-14} & 2.5\times10^{-14} & -1.0720 & 2.0000 & -0.35352 \\ & & & 4.2\times10^{-14} & -1.3\times10^{-14} & 8.0000 \end{bmatrix}$$

(b) 6×6 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, 2×1 case

$$\begin{bmatrix} 6.0000 & 0.99984 & -4.9995 & -5.9992 & -6.9990 & -7.9989 \\ & 6.0000 & 4.0006 & 5.0010 & 6.0011 & 7.0013 \\ & -2.4996\times10^{-5} & 6.0000 & 7.0000 & 8.0000 & 9.0000 \\ & & & 6.0001 & 9.0000 & 10.000 \\ & & & & 6.0001 & 10^{-4} \\ & & & & -1.0000 & 6.0001 \end{bmatrix}$$

*Computations performed on 14 digit machine, results rounded to 5 figures for display.

3. Swap 3rd and 4th blocks, 1×2 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, 2×2 case

$$\begin{bmatrix}
6.0000 & -4.9997 & -0.99972 & -5.9991 & -7.9995 & 6.9983 \\
 & 6.0001 & 2.4995 \times 10^{-5} & 7.0000 & 9.0008 & -7.9992 \\
 & -4.0008 & 6.0001 & -5.0011 & -7.0020 & 6.0006 \\
 & -2.1 \times 10^{-23} & -6.6 \times 10^{-24} & 6.0000 & 10.001 & -8.9989 \\
 & 8.9 \times 10^{-24} & -3.8 \times 10^{-25} & -9.9994 \times 10^{-6} & 6.0000 & 1.0000 \\
 & & & 3.9 \times 10^{-18} & -4.3 \times 10^{-19} & 6.0001
\end{bmatrix}$$

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(2,6)
      EQUIVALENCE (X(1,1),Y(1,1)),(C1(1,1),Y(1,3)),(C2(1,1),Y(1,5))
C
C  EXCHANGE ADJACENT DIAGONAL BLOCKS T1 AND T2 BEGINNING IN ROW J1 BY
C  ORTHOGONAL SIMILARITY TRANSFORMATIONS, RECORDED IN P, PRESERVING THE
C  TRIANGULAR FORM OF T.  BLOCK T1 IS L1 BY L1, T2 IS L2 BY L2.
      J3=J1+L1
      J2=J3-1
      J4=J2+L2
      Z = 1.0
C
C**************CLEAR THE (2,1) BLOCK.
      DO 5 J=1,L1
      DO 5 I=1,L2
 5      T(J2+I,J1+J-1)=0.
C
C**************SOLVE FOR X IN T1*X-X*T2=Z*T12, WHERE T12 IS L1 BY L2.
      CALL TXMXT(NM,N,T,J1,J3,L1,L2,T,Z,X)
      IF (Z.EQ.0.) RETURN
C
C**************COMPUTE C1 WHERE C1T*C1 = (ZSQ*I + X*XT)**-1
C**************AND C2 WHERE C2T*C2 = (ZSQ*I + XT*X)**-1.
      CALL CTCEQW(Z,X(1,1),X(2,1),X(1,2),X(2,2),T(J1,J2),T(J2,J1),C1,L1)
      CALL CTCEQW(Z,X(1,1),X(1,2),X(2,1),X(2,2),T(J4,J3),T(J3,J4),C2,L2)
C
C**************PERFORM TRANSFORMATION ON COLUMNS AND ROWS OF T,
C**************UPDATE P.
      CALL NEWVEC (Z,L1,L2,Y,T,NM,N,J1,1)
      CALL NEWVEC (Z,L1,L2,Y,T,NM,J4,J1,NM)
      CALL NEWVEC (Z,L1,L2,Y,P,NM,N,J1,NM)
C
C**************PRESERVE EQUALITY OF DIAGONAL ELEMENTS IN BLOCKS.
      IF (L2 .EQ. 2)  T(J1,J1)=T(J1+1,J1+1)=(T(J1,J1)+T(J1+1,J1+1))/2.
      IF (L1 .EQ. 2)  T(J4-1,J4-1)=T(J4,J4)=(T(J4-1,J4-1)+T(J4,J4))/2.
      RETURN
      END
```

```
      SUBROUTINE CTCEQW(Z,X11,X21,X12,X22,BETA,GAM,C,L)
      DIMENSION C(2,2)
C
C FIND AN APPROPRIATE SOLUTION C TO CT*C = W = (ZSQ*I + X*XT)**-1
      ZSQ=Z*Z
      IF (L .GT. 1)  GO TO 10
      C(1,1)=1./SQRT(ZSQ+X11*X11+X21*X21+X12*X12)
      RETURN
 10   EM11= ZSQ + X21**2 + X22**2
      EM12=-(X11*X21+X12*X22)
      D=EM11*(ZSQ+X11**2+X12**2)-EM12**2
      RTD=SQRT(D)
      EGA = SQRT(EM12**2 - BETA*EM11**2/GAM)
      ZETA = (RTD - EGA)*(RTD + EGA)/(2.0*RTD)
      PHI=SQRT(ZETA**2+EM12**2)
      FAC=EM11/(2.0*D)
      C(1,1)=SQRT(FAC*(1.0+ABS(ZETA)/PHI))
      C(2,1)=SIGN(1.0,ZETA)*EM12*FAC/(PHI*C(1,1))
      C(1,2)=(C(1,1)*EM12-C(2,1)*RTD)/EM11
      C(2,2)=(C(2,1)*EM12+C(1,1)*RTD)/EM11
      RETURN
      END
```

```
      SUBROUTINE TXMXT(NM,N,T,J1,J2,L1,L2,R,Z,X)
      DIMENSION T(NM,N),R(NM,N),X(2,2)
C
C SOLVE FOR L1 BY L2 MATRIX X IN T1*X - X*T2 = Z*R.
C T1 AND T2 BEGIN IN ROWS J1 AND J2, Z IS GIVEN ON ENTRY
C BUT ON EXIT Z IS CHANGED TO ENSURE NORM(X).LE.1.
      X(1,1) =X(2,1)=X(1,2)=X(2,2)=0.
      IF(Z.EQ.0.) RETURN
      DEL=T(J1,J1)-T(J2,J2)
      K=2*L1+L2-2
      IF (DEL .NE. 0. .OR. K .GT. 1) GO TO 5
C
C**************T1 AND T2 HAVE THE SAME EIGENVALUES. RETURN Z=0.
 4    X(1,1)=X(2,2)=1.
      Z=0.
      RETURN
C
C**************DETERMINE DIMENSIONS OF SOLUTION X.
 5    GO TO (10,20,30,40),K
C
C**************T1 IS 1 BY 1,  T2 IS 1 BY 1.
 10   XMAX=ABS(R(J1,J2))
      D=ABS(DEL)
      X(1,1)=R(J1,J2)*(Z/AMAX1(XMAX,D))*SIGN(1.,DEL)
      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(1,1)=(R(J1,J2)*DEL+R(J1,J2+1)*T(J2+1,J2))
      X(1,2)=(R(J1,J2)*T(J2,J2+1)+R(J1,J2+1)*DEL )
      XMAX=AMAX1(ABS(X(1,1)),ABS(X(1,2)))
      GO TO 50
C
C**************T1 IS 2 BY 2,  T2 IS 1 BY 1.
 30   D=DEL**2-T(J1,J1+1)*T(J1+1,J1)
      X(1,1)=(DEL*R(J1,J2)-T(J1,J1+1)*R(J1+1,J2))
      X(2,1)=(-T(J1+1,J1)*R(J1,J2)+DEL*R(J1+1,J2))
      XMAX=AMAX1(ABS(X(1,1)),ABS(X(2,1)))
      GO TO 50
C
C**************T1 IS 2 BY 2,  T2 IS 2 BY 2.
 40   BET1=T(J1,J1+1)
      GAM1=T(J1+1,J1)
      BET2=T(J2,J2+1)
      GAM2=T(J2+1,J2)
      P1=BET1*GAM1
      P2=BET2*GAM2
      DSQ=DEL**2
      E=DEL*(DSQ-(P1+P2))
      F1=(P1+AMAX1(DSQ,-P2))+AMIN1(DSQ,-P2)
      F2=(P2+AMAX1(DSQ,-P1))+AMIN1(DSQ,-P1)
      H=2.0*DEL*BET2
      G=2.0*DEL*GAM2
      D=F2**2-G*H
      IF (D .EQ. 0.)  GO TO 4
      R11=R(J1,J2)
      R12=R(J1,J2+1)
      R21=R(J1+1,J2)
      R22=R(J1+1,J2+1)
      X(1,1)=R11*E+R12*GAM2*F1-R21*BET1*F2-R22*BET1*G
      X(1,2)=R11*BET2*F1+R12*E-R21*BET1*H-R22*BET1*F2
      X(2,1)=-R11*GAM1*F2-R12*GAM1*G+R21*E+R22*GAM2*F1
      X(2,2)=-R11*GAM1*H-R12*GAM1*F2+R21*BET2*F1+R22*E
      XMAX=AMAX1(ABS(X(1,1)),ABS(X(1,2)),ABS(X(2,1)),ABS(X(2,2)))
C
C**************ENSURE NORM(X) ≤ 1.
 50   E=Z/AMAX1(XMAX,D)
      DO 60 J=1,L2
      DO 60 I=1,L1
 60     X(I,J)=X(I,J)*E
 70   IF (XMAX.GT.D) Z=Z*D/XMAX
      RETURN
      END
```

```
      SUBROUTINE NEWVEC (F,N1,N2,Z,Y,NM,N,M1,INC)
      DIMENSION Y(1),Z(2,6),W(4)
C
C COMPUTE C2*(F*V - XT*U) = NEWU ,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.
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), LB=M1, J=NM.
C WHEN INC=NM,
C PERFORM BLOCK EXCHANGE TRANSFORMATION ON 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-NM, LB=1, J=1.
      I=INC
      LB=J=NM-I+1
      IF (I .LT. J)  LB=M1
      NY=(M1-1)*NM+LB-I
      L=N1+2*N2-2
      GO TO (10,20,30,40),L
C
C**************N1=1, N2=1.
 10   DO 15 K=LB,N
         W(1)=Y(NY+2*I)*F-Z(1,1)*Y(NY+I)
         W(2)=Y(NY+I)*F+Z(1,1)*Y(NY+2*I)
         Y(NY+I)=Z(1,5)*W(1)
         Y(NY+2*I)=Z(1,3)*W(2)
 15      NY=NY+J
      RETURN
C
C**************N1=2, N2=1.
 20   DO 25 K=LB,N
         W(1)=Y(NY+3*I)*F-Z(1,1)*Y(NY+I)-Z(2,1)*Y(NY+2*I)
         W(2)=Y(NY+I)*F+Z(1,1)*Y(NY+3*I)
         W(3)=Y(NY+2*I)*F+Z(2,1)*Y(NY+3*I)
         Y(NY+I)=Z(1,5)*W(1)
         Y(NY+2*I)=Z(1,3)*W(2)+Z(1,4)*W(3)
         Y(NY+3*I)=Z(2,3)*W(2)+Z(2,4)*W(3)
 25      NY=NY+J
      RETURN
C
C**************N1=1, N2=2.
 30   DO 35 K=LB,N
         W(1)=Y(NY+2*I)*F-Z(1,1)*Y(NY+I)
         W(2)=Y(NY+3*I)*F-Z(1,2)*Y(NY+I)
         W(3)=Y(NY+I)*F+Z(1,1)*Y(NY+2*I)+Z(1,2)*Y(NY+3*I)
         Y(NY+I)=Z(1,5)*W(1)+Z(1,6)*W(2)
         Y(NY+2*I)=Z(2,5)*W(1)+Z(2,6)*W(2)
         Y(NY+3*I)=Z(1,3)*W(3)
 35      NY=NY+J
      RETURN
C
C**************N1=2, N2=2.
 40   DO 45 K=LB,N
         W(1)=Y(NY+3*I)*F-Z(1,1)*Y(NY+I)-Z(2,1)*Y(NY+2*I)
         W(2)=Y(NY+4*I)*F-Z(1,2)*Y(NY+I)-Z(2,2)*Y(NY+2*I)
         W(3)=Y(NY+I)*F+Z(1,1)*Y(NY+3*I)+Z(1,2)*Y(NY+4*I)
         W(4)=Y(NY+2*I)*F+Z(2,1)*Y(NY+3*I)+Z(2,2)*Y(NY+4*I)
         Y(NY+I)=Z(1,5)*W(1)+Z(1,6)*W(2)
         Y(NY+2*I)=Z(2,5)*W(1)+Z(2,6)*W(2)
         Y(NY+3*I)=Z(1,3)*W(3)+Z(1,4)*W(4)
         Y(NY+4*I)=Z(2,3)*W(3)+Z(2,4)*W(4)
 45      NY=NY+J
      RETURN
      END
```

Alternative, but less efficient, version of NEWVEC which better illustrates the column and row operations.

```
      SUBROUTINE NEWCOL(F,N1,N2,Z,Y,NM,N,M1)
      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 - XT*U) = NEWU ,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
      DO 10 J=1,N2
      W(J) = Y(M+N1+J,K)*F
      DO 10 L=1,N1
   10   W(J)=W(J)-Z(L,J)*Y(M+L,K)
      DO 20 J=1,N1
      W(N2+J) = Y(M+J,K)*F
      DO 20 L=1,N2
   20 W(N2+J)=W(N2+J)+Z(J,L)*Y(M+N1+L,K)
      DO 35 J=1,N2
      S=0.
      DO 30 L=1,N2
   30 S=S+Z(J,L+4)*W(L)
   35 Y(M+J,K)=S
      DO 45 J=1,N1
      S=0.
      DO 40 L=1,N1
   40 S=S+Z(J,L+2)*W(N2+L)
   45 Y(M+N2+J,K)=S
   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 I ROWS OF Y
C IN THE N1+N2 COLUMNS STARTING WITH COLUMN M1.
C COMPUTE C2*(F*V - XT*U) = NEWU ,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,I
      DO 10 J=1,N2
      W(J) = Y(K,M+N1+J)*F
      DO 10 L=1,N1
   10   W(J)=W(J)-Z(L,J)*Y(K,M+L)
      DO 20 J=1,N1
      W(N2+J) = Y(K,M+J)*F
      DO 20 L=1,N2
   20 W(N2+J)=W(N2+J)+Z(J,L)*Y(K,M+N1+L)
      DO 35 J=1,N2
      S=0.
      DO 30 L=1,N2
   30 S=S+Z(J,L+4)*W(L)
   35 Y(K,M+J)=S
      DO 45 J=1,N1
      S=0.
      DO 40 L=1,N1
   40 S=S+Z(J,L+2)*W(N2+L)
   45 Y(K,M+N2+J)=S
   50 CONTINUE
      RETURN
      END
```

References

R.H. Bartels and G.W. Stewart, Algorithm 432, Solution of the matrix equation $AX + XB = C$, Comm. ACM 15 (1972), 820-826.

B.N. Parlett, A recurrence among the elements of functions of triangular matrices, Linear Algebra and its Applications 14 (1976), 117-121.