PROGRAMS TO SWAP
DIAGONAL BLOCKS

BY
K.C. Ng\(^1\) and B.N. Parlett\(^2\)
May 21, 1987

1. Introduction
2. The Software Economizer
3. General Theory
4. Implementations details
   4.1. Standardized Real Schur Form
   4.2. Solving \( A_1X -XA_2 = B \)
   4.3. The Choleski Factor
   4.4. Representing \( H \) as a product of two reflectors
   4.5. Special treatment for the diagonal blocks.
5. Numerical Tests
6. Conclusion
7. References
Appendix A. Solving \( A_1X -XA_2 = B \)
Appendix B. Representing Reflector(s) in Factor Form
Appendix C. Listing of Fortran Subroutines

ABSTRACT

The real Schur form of a real square matrix is block upper triangular. We study techniques for performing orthogonal similarity transformations that preserve block triangular form but alter the order of the eigenvalues along the (block) diagonal.

\(^1\) Sun Microsystems, Inc.
\(^2\) Center for Pure and Applied Mathematics, Univ. of California, Berkeley, CA 94720.

This work was partially supported by Office of Naval Research Contract No. N00014-85-K-0180
1. Introduction

A triangular matrix reveals its eigenvalues along the main diagonal. By Schur's lemma any square complex matrix is unitarily similar to an upper triangular matrix with the eigenvalues arranged in any desired order along the main diagonal. It follows that any real square matrix is orthogonally similar to a real block upper triangular matrix in which each 2x2 block on the diagonal corresponds to a pair of complex conjugate eigenvalues. The Householder-QR algorithm is a stable, efficient algorithm that produces a Schur form. However the ordering of the eigenvalues that the QR algorithm produces may not be suitable for certain purposes, such as computing the exponential of the original matrix. There are programs in the library EISPACK that compute this real Schur form. See section 2.3.6 of [EIS, 1976].

This investigation presents and compares all the attractive methods we can think of for performing orthogonal similarity transformations that preserve block triangular form but rearrange the eigenvalues. This is a fairly straightforward task but it is always a challenge to try and keep down three conflicting costs: round off error, execution time, and program length.

We give some attention to the task of swapping adjacent diagonal blocks of orders p and q but our main concern is with the case p = q = 2. We use capital letters to denote matrices. Fortran programs are given at the end.

Before plunging into details we describe the methods in brief general terms. **Algorithm 0** (G.W. Stewart): Swap adjacent blocks using one or two QR steps with a pre-determined shift to force the ordering of the eigenvalues of the new blocks. **Algorithm 1**: Swap adjacent blocks as needed using an explicit orthogonal similarity transformation. At most 4 rows and columns will be modified at each swap. **Algorithm 2**: Swap adjacent blocks using Householder transformations. For swapping a pair of 2x2 blocks two Householder transformations are needed.

The table below compares the algorithms for code length and running time. The remainder of that paper is concerned with accuracy.
<table>
<thead>
<tr>
<th>Algorithm number</th>
<th>Fortran line count</th>
<th>Speed ratio (The ratio was determined by runs on 9×9 matrices)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>165</td>
<td>1.27</td>
</tr>
<tr>
<td>1</td>
<td>386</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>301</td>
<td>1.15</td>
</tr>
</tbody>
</table>

Table 1

We were encouraged to present our programs and results by Dr. Sven Hammarling (of NAG, Inc., Oxford) who showed us work on swapping that he had begun in cooperation with Dr. J. Dongarra (Argonne National Laboratory) and the late Prof. J. H. Wilkinson.

2. The Software Economizer (EXCHNG)

Consider a submatrix of the form

\[
\begin{bmatrix}
A_1 & B \\
0 & A_2
\end{bmatrix}
\]

where \(A_1\) and \(A_2\) are 2×2 diagonal blocks.

Algorithm 0 (called EXCHNG in [Ste.,1976]).

1. An implicit double shift is determined from \(A_1\).
2. An arbitrary QR step is performed to destroy the triangular form and put the matrix into Hessenberg form.
3. A sequence of double QR steps using the shift from step 1. The eigenvalues of the first block will emerge in the lower part of the array occupied by both blocks, usually in one or two steps.

Remark 1. The algorithm discards the information that there are two pairs of conjugate complex eigenvalues. Stewart modifies the standard QR program so that a supplied initial shift may overwrite the usual Francis shift at the first step. Such an algorithm would converge in one or two steps.
3. General Theory

Consider the block upper triangular matrix

\[
\begin{bmatrix}
A_1 & B \\
0 & A_2 \\
\end{bmatrix}
\]

\[A_1 \text{ is } p \times p,
\]

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

(1)

Throughout this paper we assume that \(A_1\) and \(A_2\) have no eigenvalue in common. It follows that there exists a unique \(p \times q\) matrix \(X\) such that

\[A_1X - XA_2 = B.\]

(2)

This is called Sylvester’s equation. It follows that

\[
\begin{bmatrix}
A_1 & B \\
0 & A_2 \\
\end{bmatrix} = \begin{bmatrix}
I_p & -X \\
0 & I_q \\
\end{bmatrix} \cdot \begin{bmatrix}
A_1 & 0 \\
0 & A_2 \\
\end{bmatrix} \cdot \begin{bmatrix}
I_p & X \\
0 & I_q \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
I_p & 0 \\
I_q & 0 \\
\end{bmatrix} \cdot \begin{bmatrix}
A_2 & 0 \\
0 & A_1 \\
\end{bmatrix} \cdot \begin{bmatrix}
I_p & X \\
0 & I_q \\
\end{bmatrix}
\]

(3)

**DEFINITION.** An orthogonal \((p+q)\times(p+q)\) matrix \(H\) is said to swap \(A_1\) and \(A_2\) if

\[H \cdot \begin{bmatrix}
A_1 & B \\
0 & A_2 \\
\end{bmatrix} \cdot H^T = \begin{bmatrix}
\tilde{A}_2 & \tilde{B} \\
0 & \tilde{A}_1 \\
\end{bmatrix}
\]

(4)

where \(\tilde{A}_i\) is similar to \(A_i\), \(i=1,2\).

**Lemma 1.** An orthogonal \((p+q)\times(p+q)\) matrix \(H\) swaps \(A_1\) and \(A_2\) if, and only if,

\[H \cdot \begin{bmatrix}
-X \\
I_q \\
\end{bmatrix} = \begin{bmatrix}
M_2 \\
0 \\
\end{bmatrix}
\]

for some invertible \(q \times q\) \(M_2\) where \(X\) is defined in (2).

Note that, since \(H\) is invertible,

\[
\text{rank} \begin{bmatrix}
M_2 \\
0 \\
\end{bmatrix} = \text{rank} \begin{bmatrix}
-X \\
I_q \\
\end{bmatrix} = q.
\]

Consequently \(M_2\) is \(q \times q\) and must be invertible.
Proof. If $H$ satisfies (5) then for some $q \times p$ $W$ and $p \times p$ $M_1$, 
\[
  H \cdot \begin{bmatrix} -X & I_p \\ I_q & 0 \end{bmatrix} = \begin{bmatrix} M_2 & W \\ 0 & M_1 \end{bmatrix}
\]
and, since both matrices on the left are invertible so are $M_1$ and $M_2$. Thus

\[
  H \cdot \begin{bmatrix} A_1 & B \\ 0 & A_2 \end{bmatrix} \cdot H^T = H \cdot \begin{bmatrix} -X & I_p \\ I_q & 0 \end{bmatrix} \cdot \begin{bmatrix} A_2 & 0 \\ 0 & A_1 \end{bmatrix} \cdot I_p x \cdot H^T,
\]

\[
  = \begin{bmatrix} M_2 & W \\ 0 & M_1 \end{bmatrix} \cdot \begin{bmatrix} A_2 & 0 \\ 0 & A_1 \end{bmatrix} \cdot \begin{bmatrix} M_2^{-1} & -M_2^{-1} \cdot W \cdot M_1^{-1} \\ 0 & M_1^{-1} \end{bmatrix},
\]

\[
  = \begin{bmatrix} \tilde{A}_2 & \tilde{B} \\ 0 & \tilde{A}_1 \end{bmatrix},
\]

where

\[
  \tilde{A}_i = M_i \cdot A_i \cdot M_i^{-1}, i=1,2; \quad \tilde{B} = (W A_1 - \tilde{A}_2 W) \cdot M_1^{-1}.
\]

Conversely, if $H$ swaps $A_1$ and $A_2$ then there exist $M_1$, $M_2$, and $W$ such that

\[
  \begin{bmatrix} \tilde{A}_2 & \tilde{B} \\ 0 & \tilde{A}_1 \end{bmatrix} = M_2 W \cdot A_2 0 \cdot M_1 \cdot A_1 \cdot 0 I_q \cdot M_1^{-1},
\]

\[
  = H \cdot \begin{bmatrix} -X & I_p \\ I_q & 0 \end{bmatrix} \cdot \begin{bmatrix} A_2 & 0 \\ 0 & A_1 \end{bmatrix} \cdot I_p x \cdot H^T.
\]

It follows that

\[
  D = \begin{bmatrix} M_2 & W \\ 0 & M_1 \end{bmatrix}^{-1} \cdot H \cdot \begin{bmatrix} -X & I_p \\ I_q & 0 \end{bmatrix}
\]

commutes with $\begin{bmatrix} A_2 & 0 \\ 0 & A_1 \end{bmatrix}$.

Since $A_1$ and $A_2$ have no eigenvalues in common $D$ must be a polynomial in $\text{diag}(A_2, A_1)$. See [Gant. vol. 1, p.222].

\[
  H \cdot \begin{bmatrix} -X & I_p \\ I_q & 0 \end{bmatrix} = \begin{bmatrix} M_2 & W \\ 0 & M_1 \end{bmatrix} \cdot D
\]

must be block upper triangular. This establishes the converse.

QED
Lemma 2. An orthogonal $H$ that swaps $A_1$ and $A_2$ must have the form

$$H = \begin{bmatrix} C_2^{-1} & 0 \\ 0 & C_1^{-1} \end{bmatrix} \cdot \begin{bmatrix} -X^T & I_q \\ I_p & X \end{bmatrix}$$

(6)

where

$$C_2 \cdot C_2^T = I + X^T \cdot X,$$

$$C_1 \cdot C_1^T = I + X \cdot X^T.$$  

(7)

Proof. Write $C_2^T$ for $M_2$, multiply (5) by $H^T$ and use the orthogonality of $H$ to find

$$H^T \cdot \begin{bmatrix} -X \\ I_q \end{bmatrix} = H^T \cdot \begin{bmatrix} C_2^T \\ 0 \end{bmatrix} = H^T \cdot \begin{bmatrix} I_q \\ 0 \end{bmatrix} \cdot C_2^T.$$  

Transposing reveals the first row of $H$. The second row follows by orthogonality.

QED

Remark 1. There are infinitely many choices for $C_1$ and $C_2$ that satisfy (7). See Section 4.3 for more details.

Remark 2. One of our implementations uses the form in (6) explicitly. The block rows are orthogonal by their form, so it is the accuracy with which (7) is fulfilled that determines the orthogonality of the computed $H$. An alternative implementation starts from (5) and seeks $H$ as a product of elementary reflectors (also known as Householder matrices).

The key blocks of the transformed matrix can be found explicitly. Using (3), (6), and (7) it is not difficult to see that

$$\tilde{A}_2 = C_2^T \cdot A_2 \cdot C_2^{-T}, \quad \tilde{A}_1 = C_1^{-1} \cdot A_1 \cdot C_1,$$

$$\tilde{B} = C_2^T \cdot A_2 \cdot X^T \cdot C_1^T - C_2^{-1} \cdot X^T \cdot A_1 \cdot C_1.$$  

(8)
4. Implementation details

4.1. Standardized Real Schur Form

The Schur form of a matrix is not unique and the real Schur form of a real matrix offers even more freedom. We urge the adoption of the following conventions.

i) $2 \times 2$ diagonal blocks are used exclusively for complex conjugate pairs of eigenvalues, not for distinct real eigenvalues.

ii) The diagonal elements of $2 \times 2$ diagonal blocks are made equal. This value is the real part of each eigenvalue.

Consequently we advocate the form

$$\begin{bmatrix}
\alpha & \beta \\
\gamma & \alpha
\end{bmatrix}, \quad \beta \gamma < 0.$$  

The off diagonal elements of the $2 \times 2$ diagonal blocks cannot always be made equal in absolute value but they must be opposite in sign. To guarant uniqueness one may require $\beta$ and $\gamma$ to satisfy $0 < \gamma \leq -\beta$, but that is not essential. Note that the eigenvalues are $\alpha \pm \sqrt{\beta \gamma}$.

The use of a standard real Schur form facilitates the swapping of diagonal blocks as well as ensuring that the real parts of all eigenvalues are held on the diagonal of the real Schur form.

If a given real Schur form does not have its eigenvalues ordered appropriately down the diagonal then some swapping of diagonal blocks will be needed. However the task is considerably simplified by the fact that no block has order exceeding 2. Any configuration of eigenvalues can be reached by swapping adjacent diagonal blocks and this is the task we consider below.

Here is a method (cf section 4.5) to put $2 \times 2$ diagonal blocks into standard form. Let

$$A = \begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix}.$$  

Define a reflection $P$ by

$$P^T = P = \begin{bmatrix}
-\cos(\theta) & \sin(\theta) \\
\sin(\theta) & \cos(\theta)
\end{bmatrix}, \quad \theta = \frac{1}{2} \tan^{-1} \left( \frac{a_{11} - a_{22}}{a_{12} + a_{21}} \right).$$
Write $c = \cos(\theta)$ and $s = \sin(\theta)$. It is not difficult to see that $PAP$ transforms $A$ to a standard form:

$$PAP = \begin{bmatrix}
\frac{a_{11}+a_{22}}{2} & \frac{a_{21} - c \cdot a_{11}-a_{22}}{s} \\
\frac{a_{21} - c \cdot a_{11}-a_{22}}{s} & \frac{a_{11}+a_{22}}{2}
\end{bmatrix} \quad (9)$$

4.2. Solving $A_1X - XA_2 = B$

Considerable attention has been paid to the general case of this equation, now known as Sylvester's equation. See [B&.S,1972] and [G,N,&vL,1979]. When $A_1$ and $A_2$ are either $1 \times 1$ or standardized $2 \times 2$ matrices the solution can be given explicitly using stable formulae.

In an earlier unpublished report [Pa,1977] we advocated scaling $X$, i.e., we solved

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

with $\xi$ chosen so that $\|X\| \leq 1$. Further analysis shows that this caution is unnecessary. There is no danger in working with $X$ of large norm provided that $\|X\|^2$ does not overflow. Moreover if $\|X\|^2$ does overflow then the blocks should not be swapped because a tiny perturbation will give the new $A_1$ and $A_2$ at least one common eigenvalue.

Our algorithm for solving the Sylvester equation is called $TXMXT$ (for $TX-XT$) and is described in Appendix A, see also the program Appendix C.

4.3. The Choleski Factor

If the explicit orthogonal $H$ described in Section 3 is to be used then it is necessary to solve the equations (7) for $C_1$ and $C_2$. We can see no reason to avoid the Choleski factorization. The formulae are given below. When presenting the algorithm in detail we write $x_{ij}$ for $X(i,j)$. Recall equation (7):

$$C_2 \cdot C_2^T = I + X^T \cdot X$$

$$C_1 \cdot C_1^T = I + X \cdot X^T$$
Algorithm 1.
An H is found explicitly in the form of (5) to swap A₁ and A₂. The choice for C₁ and C₂ in (7) are the Choleski factors.

Case 1. X is 1×1, then \( C₂(1,1) = C₁(1,1) = \sqrt{1+x₁₁^2} \).

Case 2. X is 1×2, then \( C₁(1,1) = \sqrt{1+x₁₁^2+x₁₂^2} \), and

\[
C₂ = \begin{bmatrix}
y & 0 \\
x₁₁ \cdot x₁₂ & \sqrt{1+(x₁₂/y)^2}
\end{bmatrix}, \quad y = \sqrt{1+x₁₁^2}.
\]

Case 3. X is 2×1, then \( C₂(1,1) = \sqrt{1+x₁₁^2+x₂₁^2} \), and

\[
C₁ = \begin{bmatrix}
y & 0 \\
x₁₁ \cdot x₂₁ & \sqrt{1+(x₂₁/y)^2}
\end{bmatrix}, \quad y = \sqrt{1+x₁₁^2}.
\]

Case 4. X is 2×2, let \( \delta = x₁₁ \cdot x₂₂ - x₁₂ \cdot x₂₁ \), \( y = \sqrt{1+x₁₁^2+x₁₂^2} \), and \( k = \sqrt{1+x₁₁^2+x₂₁^2} \); we have

\[
C₁ = \begin{bmatrix}
y & 0 \\
x₁₁ \cdot x₂₁ + x₁₂ \cdot x₂₂ & \sqrt{1 + x₂₁^2 \cdot x₂₂^2 + \delta^2/y^2}
\end{bmatrix},
\]

and

\[
C₂ = \begin{bmatrix}
k & 0 \\
x₁₁ \cdot x₂₁ + x₁₂ \cdot x₂₂ & \sqrt{1 + x₁₂^2 \cdot x₂₁^2 + \delta^2/k^2}
\end{bmatrix}.
\]

4.4. Representing H as a product of two reflectors

The explicit form of H in Section 3 is not mandatory. W. Kahan suggested using two reflections instead. Here are the details. First some notation: A n×n reflection (or Householder matrix) can be represented as \( I - uu^T/d \), where I is the n×n identity matrix, u is a
n-vector, and \( d = \frac{1}{2} \| u \|^2 \). We use the fact that if \( u = x + y \) and \( \| x \| = \| y \| \), then \((I - uu^T/(\| u \|^2)) \cdot x = -y\).

**Algorithm 2.**

An \( H \) is found implicitly in the form of either a reflector or a product of two reflectors to swap \( A_1 \) and \( A_2 \). The reflector(s) are determined as follows:

**Case 1.** \( X \) is \( 1 \times 1 \). Let \( sx = \text{sign}(x_{11}) \cdot \sqrt{1 + x_{11}^2} \). We seek a reflection \( H \) so that

\[
H \cdot \begin{bmatrix} -x_{11} \\ 1 \end{bmatrix} = \begin{bmatrix} -sx \\ 0 \end{bmatrix}.
\]

The special form of \( H \) leads to

- If \( x_{11}/sx < 0.5 \), then \( u_1 = sx - x_{11} \); else \( u_1 = 1/(sx + x_{11}) \)
- \( u_2 = 1 \)
- \( d = u_1 \cdot sx \)

**Case 2.** \( X \) is \( 1 \times 2 \). Let \( sx = -\text{sign}(x_{12}) \cdot \sqrt{1 + x_{11}^2 + x_{12}^2} \). We observe that if a reflector \( H \) satisfies

\[
H \cdot \begin{bmatrix} 1 \\ x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -sx \end{bmatrix},
\]

then it satisfies (5). The proof is left to the reader. The special form of \( H \) leads to

- \( u_1 = 1 \)
- \( u_2 = x_{11} \)
- If \( -x_{12}/sx < 0.5 \), then \( u_3 = x_{12} + sx \); else \( u_3 = (1 + x_{11}^2)/(sx - x_{12}) \)
- \( d = u_3 \cdot sx \)

**Case 3.** \( X \) is \( 2 \times 1 \). Let \( sx = \text{sign}(x_{11}) \cdot \sqrt{1 + x_{11}^2 + x_{21}^2} \). From (5) we seek a reflection \( H \) so that

\[
H \cdot \begin{bmatrix} -x_{11} \\ -x_{21} \\ 1 \end{bmatrix} = \begin{bmatrix} -sx \\ 0 \\ 0 \end{bmatrix}.
\]
The special form of $H$ leads to

If $x_{11}/sx \leq 0.5$, then $u_1 = sx - x_{11}$; else $u_1 = (1+x_{21}^2)/(sx+x_{11})$

$u_2 = -x_{21}$

$u_3 = 1$

$d = u_1 \cdot sx$

Case 4. $X$ is $2 \times 2$. Two reflections $H_1$ and $H_2$ are required. In (5) let $M_2$ be upper triangular and $H = H_2 \cdot H_1$. First define

$$sx = \text{sign}(x) \cdot \sqrt{1+x_{11}^2+x_{21}^2}.$$  

We seek a reflection $H_1 = I - uu^T/d$ so that

$$H_1 = \begin{bmatrix} -x_{11} & 0 \\ -x_{21} & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}.$$  

From the special form of $H$ we have

If $x_{11}/sx \leq 0.5$, then $u_1 = sx - x_{11}$; else $u_1 = (1+x_{21}^2)/(sx+x_{11})$

$u_2 = -x_{21}$

$u_3 = 1$

$u_4 = 0$

$d = u_1 \cdot sx$.

Next, define an intermediate vector $y$ by

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = H_1 \begin{bmatrix} -x_{12} \\ -x_{22} \\ 0 \\ 1 \end{bmatrix}.$$  

One can verify that

$y_1 = -(x_{11} \cdot x_{12} + x_{21} \cdot x_{22})/sx,$

$y_2 = -x_{22} - x_{21} \cdot (x_{12} \cdot u_1 - x_{21} \cdot x_{22})/d,$

$y_3 = (x_{12} \cdot u_1 - x_{21} \cdot x_{22})/d,$

$y_4 = 1.$

Note that $y_2 = -x_{22} - x_{21} \cdot y_3$. Let $sy = \text{sign}(y_2) \cdot \sqrt{1+y_2^2+y_3^2}$

We seek the second reflection $H_2 = I - vv^T/g$ so that
\[
\begin{bmatrix}
y_1 \\
y_2 \\
y_3 \\
y_4
\end{bmatrix}
= \begin{bmatrix}
y_1 \\
-sy \\
0 \\
0
\end{bmatrix}.
\]

There is no need to change the top row. Proceed as before to obtain
\[v_1 = 0\]
if \(-y_2/sy \leq 0.5\), then \(v_2 = sy + y_2\); else \(v_2 = (1+y_3^2)/(sy-y_2)\)
\[v_3 = y_3\]
\[v_4 = 1\]
\[g = v_2 \cdot sy.\]

Remark. Since each \(H\) that swaps \(A_1\) and \(A_2\) can be represented in the form of (6) (lemma 2), it is worthwhile to see what the reflection, or product of reflections, looks like in this form. In fact we make use of it in Section 4.5. We compute the corresponding \(C_1\) and \(C_2\) in appendix B. They are obtained by noting that
\[
H \cdot \begin{bmatrix}
-X \\
I \\
q
\end{bmatrix} = \begin{bmatrix}
C_T^1 \\
C_T^2
\end{bmatrix} \quad \text{and} \quad \begin{bmatrix}
I_p & X
\end{bmatrix} H^T = \begin{bmatrix}
0 & C_1
\end{bmatrix}.
\]

4.5. Special treatment for the diagonal blocks.

From (8) the new diagonal block \(\tilde{A}\) is equal to \(W \cdot A \cdot W^{-1}\) for some \(W\). Given \(A,W\) we use a special subroutine (called EQUDI, see Appendix C) to put 2x2 diagonal block \(\tilde{A} = W \cdot A \cdot W^{-1}\) into standard form and effect the associated changes in the corresponding rows and columns of the Schur form. For better accuracy we derive here the analytic formulae for the transformation, based on \(W\) and \(A\). Recall that \(a_{11} = a_{22}\).

\[d = \det(W),\]
\[z = a_{21} \cdot w_{12} \cdot w_{22} - a_{12} \cdot w_{11} \cdot w_{21};\]
\[
WAW^{-1} = \begin{bmatrix}
\frac{a_{11} + z/d}{d} & \frac{a_{12} \cdot w_{11}^2 - a_{21} \cdot w_{12}^2}{d} \\
\frac{a_{21} \cdot w_{22}^2 - a_{12} \cdot w_{21}^2}{d} & \frac{a_{11} - z/d}{d}
\end{bmatrix}
\]  

(10)

Apply (9) in Section 4.1 to \(A=WAW^{-1}\) above to find

\[
u = a_{12} \cdot (w_{11}-w_{21}) \cdot (w_{11}+w_{21}) + a_{21} \cdot (w_{22}-w_{12}) \cdot (w_{22}+w_{12})
\]

\[
\theta = \frac{1}{2} \cdot \tan^{-1}(2z/u)
\]

\[
s = \sin(\theta), \quad c = \cos(\theta), \quad \text{and}
\]

\[
P\tilde{A}P = \begin{bmatrix}
a_{11} & v_1 \\
v_2 & a_{11}
\end{bmatrix}
\]

(11)

where

\[
P = \begin{bmatrix}
-c & s \\
s & c
\end{bmatrix}
\]

\[
v_1 = \frac{a_{21} \cdot w_{22} \cdot (w_{22}-w_{12} \cdot c/s) - a_{12} \cdot w_{21} \cdot (w_{21}-w_{11} \cdot c/s)}{d},
\]

\[
v_2 = \frac{-a_{21} \cdot w_{22} \cdot (w_{22}+w_{12} \cdot s/c) + a_{12} \cdot w_{21} \cdot (w_{21}+w_{11} \cdot s/c)}{d}
\]

Since eigenvalues are preserved under similar transformation, we must have \(v_1 \cdot v_2 = a_{12} \cdot a_{21}\); thus we may recompute \(v_1\) from \(v_2\) or vice versa, depending on which one is smaller in magnitude.

5. Numerical Tests

We have done extensive testing on matrices with various mixtures of block size. All 3 algorithms perform well in most cases. To investigate more closely the accuracy of Algorithms 0, 1, and 2 under extreme conditions, we tested them on three sets of matrices: one with huge \(B\), one with a choice of \(B\) so that \(|\det(X)| \ll \|X\|^2\), and finally one with fairly close eigenvalues. These tests perform 2x2 block swaps.

How to measure the “correctness” of the computed output is not so easy. Let \(\tilde{A}\) be the output matrix \(P^TAP\) where \(P\) is the orthogonal matrix that accumulates all the transformations that are applied to \(A\). We believe that the only sensible measures for the accuracy of \(P\) and \(\tilde{A}\) are 1) how close is \(P \cdot P^T\) to the identity matrix? and 2) how close is \(P\tilde{A}P^T\) to the original matrix \(A\).
Thus our measuring parameters are:

1. $E_P = \| I - PP^T \|$,
\[ (12) \]

2. $E_A = \| A - P\tilde{A}P^T \| / \| A \|$. 
\[ (13) \]

$E_P$ is the orthogonality error in $P$; $E_A$ is the norm relative error in $P^T\tilde{A}P$. Out of curiosity we also computed

3. $\epsilon = \text{Max}\{|e_{i,j}|, A(i,j)\neq 0\}$ 
\[ (14) \]

where $e_{i,j} = \| (A - P\tilde{A}P^T)(i,j)/A(i,j) \|$. This is the worst relative error among the elements of $P\tilde{A}P^T$.

The third parameters make sense only when $A(i,j) \neq 0$, since fill-in (zero elements become non-zero) is unavoidable in recovering $A$ from $P$ and $\tilde{A}$. We should point out that $\epsilon$ is too exigent a measure for the accuracy of $P$ and $\tilde{A}$. It is unreasonable to demand high relative accuracy for tiny elements in $P\tilde{A}P^T$. Nevertheless we found $\epsilon$ helpful in showing subtle differences between good swapping programs. The following results were obtained on a SUN 3/50. $P$ and $\tilde{A}$ are computed solely in single precision arithmetic. However, the error measures are computed in double precision. Only three digits are displayed for the error measures in order to keep the display clean.

We have also run our program on a VAX/750 with similar results.

The Fortran program for Algorithm 0 is Stewart's EXCHNG, the Fortran program for Algorithm 1 and 2 are written according to section 4.3 and 4.4 with special formula for the diagonal blocks described in section 4.5. See the listing in Appendix C.

The roundoff unit is $2^{-23} \approx 1.192E-7$ in the following numerical results.

Test matrix I with parameter $\tau$ (large B)

\[
A(\tau) = \begin{bmatrix}
2 & -87 & -20000\tau & 10000\tau \\
5 & 2 & -20000\tau & -10000\tau \\
0 & 0 & 1 & -11 \\
0 & 0 & 37 & 1
\end{bmatrix}
\]
<table>
<thead>
<tr>
<th>$\tau$</th>
<th>Algorithm</th>
<th>$\epsilon$</th>
<th>$E_A$</th>
<th>$E_P$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>see (14)</td>
<td>see (13)</td>
<td>see (12)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2.11e-3</td>
<td>1.45e-6</td>
<td>1.66e-6</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2.56e-6</td>
<td>1.62e-7</td>
<td>1.96e-7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.74e-6</td>
<td>3.57e-7</td>
<td>3.41e-7</td>
</tr>
<tr>
<td>$2^{-3}$</td>
<td>0</td>
<td>1.25e-4</td>
<td>4.89e-7</td>
<td>1.05e-6</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.66e-6</td>
<td>1.27e-7</td>
<td>1.62e-7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>8.70e-6</td>
<td>4.15e-7</td>
<td>4.79e-7</td>
</tr>
<tr>
<td>$2^{-6}$</td>
<td>0</td>
<td>5.18e-5</td>
<td>4.69e-7</td>
<td>6.97e-7</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.90e-6</td>
<td>2.19e-7</td>
<td>2.32e-7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>8.34e-7</td>
<td>1.93e-7</td>
<td>2.14e-7</td>
</tr>
</tbody>
</table>

$P$ and $\bar{A} = P^TAP$ from algorithm 2 when $A = A(1)$.

$$X = \begin{bmatrix} -57387.61 & 6294.046 \\ -3106.521 & -7298.501 \end{bmatrix},$$


$$\bar{A} = \begin{bmatrix} 1.000000 & -85.98243 & 20011.92 & -10194.38 \\ 4.733524 & 1.000000 & 19985.38 & 9807.223 \\ 0.000000 & 0.000000 & 2.000000 & -11.01783 \\ 0.000000 & 0.000000 & 39.48143 & 2.000000 \end{bmatrix}.$$  

Test matrix II with parameter $\tau$ ($\mid \det(X) \mid \ll \|X\|^2$)

$$A(\tau) = \begin{bmatrix} -3 & -87 & 3576\tau & 4888\tau \\ 5 & -3 & -88\tau & -1440\tau \\ 0 & 0 & 17 & -45 \\ 0 & 0 & 37 & 17 \end{bmatrix}.$$
Test matrix III with parameter $\tau$ (close eigenvalues)

$$A(\tau) = \begin{bmatrix}
7.001 & -87 & 394\tau & 22.2\tau \\
5 & 7.001 & -12.2\tau & -36\tau \\
0 & 0 & 7.01 & -11.7567 \\
0 & 0 & 37 & 7.01
\end{bmatrix}$$
<table>
<thead>
<tr>
<th>$\tau$</th>
<th>Algorithm</th>
<th>$\varepsilon$</th>
<th>$E_A$</th>
<th>$E_P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5.85e-6</td>
<td>8.73e-7</td>
<td>8.34e-7</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.50e-7</td>
<td>2.91e-7</td>
<td>2.67e-7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4.69e-7</td>
<td>1.19e-7</td>
<td>2.48e-7</td>
</tr>
<tr>
<td>$2^{-3}$</td>
<td>0</td>
<td>6.12e-6</td>
<td>7.73e-7</td>
<td>1.30e-6</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>6.12e-7</td>
<td>2.78e-7</td>
<td>2.36e-7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6.18e-7</td>
<td>2.68e-7</td>
<td>8.03e-7</td>
</tr>
<tr>
<td>$2^{-6}$</td>
<td>0</td>
<td>4.61e-5</td>
<td>6.49e-7</td>
<td>7.89e-7</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.20e-6</td>
<td>4.45e-7</td>
<td>4.05e-7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.83e-6</td>
<td>3.45e-7</td>
<td>5.01e-7</td>
</tr>
</tbody>
</table>

$P$ and $\bar{A} = P^T A P$ from algorithm 2 when $A = A(1)$.

$$X = \begin{bmatrix}
12581.73 & -2869.060 \\
1218.421 & 1770.267 \\
\end{bmatrix},$$

$$P = \begin{bmatrix}
-1.0000000E+0 & -1.293420E-4 & 6.833661E-5 & -4.892822E-5 \\
1.293048E-4 & -9.999997E-1 & 1.150727E-4 & 5.055802E-4 \\
6.830178E-5 & 1.152873E-4 & 1.000000E+0 & 4.074871E-4 \\
-4.902146E-5 & 5.055270E-4 & -4.076063E-4 & 9.999997E-1 \\
\end{bmatrix},$$

$$\bar{A} = \begin{bmatrix}
7.010000 & -87.01575 & -39.38432 & -221.7753 \\
4.999070 & 7.010000 & 12.19859 & 36.000998 \\
0.000000 & 0.000000 & 7.000999 & -11.75932 \\
0.000000 & 0.000000 & 36.99190 & 7.000999 \\
\end{bmatrix}.$$  

### 6. Conclusion

The test results in section 5 reveal that all three algorithms are acceptable since Norm error measures $E_A$ are tiny. Algorithm 1 and 2 have the advantage of keeping the real eigenvalues on the diagonals.
at all times. The finer measure e indicates that Algorithm 0 and Algorithm 1 in certain cases are inferior to Algorithm 2 but in other tests cases the roles of Algorithm 1 and 2 are reversed.

We find no reason to reject any of the methods and can give no preference.

7. References


Appendix A. Solving $A_1 X - XA_2 = B$

When $A_1$ and $A_2$ are in standard form the inverse of the coefficient matrix can be expressed quite succinctly and safely. Let

$$A_i = \begin{bmatrix} \alpha_i & \beta_i \\ \gamma_i & \alpha_i \end{bmatrix}, \quad i=1,2, \quad \gamma_i \cdot \beta_i < 0.$$

Let $\delta = \alpha_1 - \alpha_2$, then the equations for solving $X$ may be written as

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

where

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

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

$$\left(3\right) \begin{bmatrix} C^2 - \beta_1 \gamma_1 & 0 \\ 0 & C^2 - \beta_1 \gamma_1 \end{bmatrix} x = \begin{bmatrix} C & -\beta_1 I_2 \\ -\gamma_1 I_2 & C \end{bmatrix} b.$$

Now let

$$G = \left(C^2 - \beta_1 \gamma_1\right)^{-1} = \begin{bmatrix} \tau & 2\delta \gamma_2 \\ 2\delta \gamma_2 & \tau \end{bmatrix} / d$$

where

$$\left(4\right) \tau = \delta^2 + \beta_2 \gamma_2 - \beta_1 \gamma_1, \quad d = \tau^2 - (2\delta \beta_2)(2\delta \gamma_2) > 0,$$

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

$$x = \begin{bmatrix} G & 0 \\ 0 & G \end{bmatrix} \begin{bmatrix} C & -\beta_1 I_2 \\ -\gamma_1 I_2 & C \end{bmatrix} b.$$
where

\[ \eta = \tau - 2\gamma_2 \beta_2 = \delta^2 - (\beta_1 \gamma_1 + \beta_2 \gamma_2) > 0, \]

\[ \psi = 2\delta^2 - \tau = \delta^2 + (\beta_1 \gamma_1 - \beta_2 \gamma_2). \]

Inevitably (5) is Cramer's rule and \( d = \det(A_1 \bullet I - I \bullet A_2) \) so that \( d=0 \) if and only \( \alpha_1 = \alpha_2, \beta_1 \gamma_1 = \beta_2 \gamma_2. \)

Remark. One step of iteration refinement may be needed if the structure matrix is ill-conditioned. We form the residual matrix \( R = B - (A_1 X - X A_2) \). If \( R \) is large relative to \( B \), then using (5) again to solve for the correction matrix \( E_x \) from \( A_1 E_x - E_x A_2 = R \) and refine \( X \) by subtracting \( E_x \) from \( X \).
Appendix B. Representing reflectors in form (6) of section 3

From (6) in section 3 we have

\[ H \begin{bmatrix} -X \\ I_q \end{bmatrix} = \begin{bmatrix} C_2^T \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} I_p & X \end{bmatrix} H^T = \begin{bmatrix} 0 & C_1 \end{bmatrix}. \]

Thus to represent the reflector(s) in 4.4 we need only to compute \( C_1 \) and \( C_2 \) using the formulae above. We skip the details of algebraic manipulations and give the results below.

**case 1x1:** \( C_1 = (sx) \), \( C_2 = (-sx) \), where \( sx = \text{sign}(x11) \cdot \sqrt{1+x11^2} \).

**case 1x2:** \( C_1 = (sx) \), and

\[
C_2 = \begin{bmatrix}
-x11 & 1 \\
-x12 - \frac{1}{u3} & -x11 \\
\frac{1}{u3} & \frac{x11}{u3}
\end{bmatrix}, \quad \det(C_2) = sx,
\]

where

\[
sx = -\text{sign}(x12) \cdot \sqrt{1+x11^2+x12^2},
\]

if \(-x12/sx \leq 0.5\), then \( u3 = x12 + sx \); else \( u3 = (1+x11^2)/(sx-x12) \).

**case 2x1:** \( C_2 = (sx) \), and

\[
C_1 = \begin{bmatrix}
\frac{x21}{u1} & x11 - \frac{1}{u1} \\
u1 & 1 \\
x21 & x21
\end{bmatrix}, \quad \det(C_1) = sx,
\]

where

\[
sx = \text{sign}(x11) \cdot \sqrt{1+x11^2+x21^2},
\]

if \( x11/sx \leq 0.5\), then \( u1 = sx - x11 \); else \( u1 = (1+x21^2)/(sx+x11) \).

**case 2x2:**

\[
C_1 = \begin{bmatrix}
x11 - \frac{sy-x22}{u1 \cdot v2} & x12 - \frac{x21}{u1 \cdot v2} \\
x21 - \frac{v3}{v2} & x22 - \frac{1}{v2}
\end{bmatrix}, \quad \det(C_1) = \det(C_2) = sx \cdot sy,
\]

\[
C_2 = \begin{bmatrix}
-sx & 0 \\
y1 & -sy
\end{bmatrix}, \quad \det(C_1) = \det(C_2) = sx \cdot sy,
\]
where \( sx, u_1, y_1, y_2, sy, v_2 \) are defined by

\[
\begin{align*}
sx &= \text{sign}(x_{11}) \cdot \sqrt{1 + x_{11}^2 + x_{21}^2}, \\
\text{If } x_{11}/sx \leq 0.5, \text{ then } u_1 &= sx - x_{11}; \text{ else } u_1 = (1 + x_{21}^2)/(sx + x_{11}), \\
y_1 &= -(x_{11} \cdot x_{12} + x_{21} \cdot x_{22})/sx, \\
y_3 &= (x_{12} \cdot u_1 - x_{21} \cdot x_{22})/(u_1 \cdot sx), \\
y_2 &= -x_{22} - x_{21} \cdot y_3, \\
sy &= -\text{sign}(y_2) \cdot \sqrt{1 + y_2^2 + y_3^2}, \\
\text{if } -y_2/sy \leq 0.5, \text{ then } v_2 &= sy + y_2; \text{ else } v_2 = (1 + y_3^2)/(sy - y_2).
\end{align*}
\]
Appendix C. Listing of Fortran Subroutines

Subroutine SWAPB (Algorithm 1)
Subroutine SWAPB (Algorithm 2)
Subroutine HOUSE (used in Algorithm 2's SWAPB)
Subroutine EQUID
Subroutine TXIXT
SUBROUTINE SWAPB(T,P,N,J1,N1,N2,MT,NP)
REAL T(MT,N),P(MT,N)
INTEGER NP,MT,N,J1,N1,N2
C GIVEN T IN SCHUR FORM SWAP SWAPS ADJACENT DIAGONAL BLOCKS T1
C AND T2 IN MATRIX T BEGINNING IN ROW J1 BY ORTHOGONAL SIMILARITY
C TRANSFORMATIONS THAT PRESERVES THE SCHUR FORM OF T. THE
C DIMENSION OF BLOCK T1 IS N1 BY N1 AND T2 IS N2 BY N2. THE
C PARAMETERS IN THE CALLING SEQUENCE ARE (STARRED PARAMETERS ARE
C ALTERED BY THE SUBROUTINE)
C *T  THE MATRIX Whose BLOCKS ARE BEING SWAPPED.
C **P  THE ARRAY INTO WHICH THE TRANSFORMATIONS
C       ARE TO BE ACCUMULATED.
C N  THE ORDER OF THE MATRIX T.
C J1  THE POSITION OF THE BLOCKS.
C N1  SIZE OF THE FIRST BLOCK.
C N2  SIZE OF THE SECOND BLOCK.
C MT  THE FIRST DIMENSION OF THE ARRAY T.
C NP  THE FIRST DIMENSION OF THE ARRAY P.
C METHOD:
C   ALGORITHM I OF "PROGRAMS TO SWAP DIAGONAL BLOCKS" WITH
C   SPECIAL FORMULA FOR THE DIAGONAL BLOCKS
C SUBPROGRAMS:
C   T00XT, EQUID
C INTERNAL VARIABLES:
C REAL D,R,S,Y,Z,01,02,03,04,01,02,03,04,05,06,07,08
C REAL T11,T12,T33
C REAL X(1,1),X11,X12,X21,X22
C REAL W(1,1),W11,W12,W21,W22
C REAL V(1,1),V11,V12,V21,V22
C REAL A(1,1),A2(1,2)
C EQUIVALENCE (X(1,1),X11), (X(1,2),X12), (X(2,1),X21), (X(2,2),X22)
C EQUIVALENCE (W(1,1),W11), (W(1,2),W12), (W(2,1),W21), (W(2,2),W22)
C EQUIVALENCE (V(1,1),V11), (V(1,2),V12), (V(2,1),V21), (V(2,2),V22)
C INTEGER IZ,K,J1,J2,J3,J4
C SOLVE X FOR \begin{bmatrix} 1 & -X \\ 0 & 1 \end{bmatrix} \begin{bmatrix} A1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \end{bmatrix} = \begin{bmatrix} A1 & T12 \end{bmatrix} \text{ BY CALLING T00XT}
C\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} A2 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix}
C CALL T00XT(T,M,N1,N2,J1,X,IZ,MT)
C IF IZ=0, A1 AND A2 ARE TOO CLOSE TO SWAP
C IF(IZ.EQ.0) GOTO 50
K=H1+H2+2
J2 = J1+1
J3 = J1+H1
J4 = J3+1
IF(N1.EQ.2) THEN
  A(1,1)=T(J1,J1)
  A(1,2)=T(J1,J2)
  A(2,1)=T(J2,J1)
  A(2,2)=T(J2,J2)
IF(N1.EQ.2) THEN
  A(1,1)=T(J1,J1)
  A(1,2)=T(J1,J2)
  A(2,1)=T(J2,J1)
  A(2,2)=T(J2,J2)
ENDIF
IF(M2.EQ.2) THEN
  R2(1,1)=T(J3,J3)
  R2(1,2)=T(J3,J4)
  R2(2,1)=T(J4,J3)
  R2(2,2)=T(J4,J4)
ENDIF
GOTO (10,20,30,40), K
C N1=1, N2=1 : H = [ S 0 ] [ -X11 1 ] , S = 1.0/SQRT(1+X11**2)
C
10 S=1.0/SQRT(1.0+X11*X11)
C
C PERFORM HT=HT
C
  T11 = T(J1,J1)
  T22 = T(J2,J2)
  DO 12 J=1,N
    W1 = T(J1,J1)
    W2 = T(J2,J2)
    Y1 = S*(-X11*W1 + W2)
    Y2 = S*(W1 + X11*W2)
    T(J1,J1) = Y1
    T(J2,J2) = Y2
12 CONTINUE
C
C PERFORM P=HT
C
  DO 14 J=1,N
    W1 = P(J1,J1)
    W2 = P(J2,J2)
    Y1 = S*(-X11*W1 + W2)
    Y2 = S*(W1 + X11*W2)
    P(J1,J1) = Y1
    P(J2,J2) = Y2
14 CONTINUE
C
C SWAP DIAGONAL ELEMENTS
C
  T(J2,J2)=T11
  T(J1,J1)=T22
GOTO 50
C
C
20 S = 1+X11*X11
Y = X12*X12
U1 = SORT(S)
U2 = SORT(1.0+Y/S)
U2 = X12*X12/U1
U1 = 1.0/(SORT(S+Y))
U1 = 1.0/U1
U2 = -<X11*X12*U3>/S
T11 = T(J1,J1)
T3 = J2+1
C
C PERFORM H*T*H
C DO 22 I=J1,N
   W1 = T(J1,1)
   W2 = T(J2,1)
   W3 = T(J3,1)
   V1 = -X11*W1 + W2
   V2 = -X12*W1 + W3
   V3 = W1 + X11*W2 + X12*W3
   T(J1,1) = U1*V1
   T(J2,1) = U2*V1+U3*V2
   T(J3,1) = U1*V3
22 CONTINUE
C
C PERFORM P*H
C DO 24 I=1,J3
   W1 = T(1,J1)
   W2 = T(1,J2)
   W3 = T(1,J3)
   V1 = -X11*W1 + W2
   V2 = -X12*W1 + W3
   V3 = W1 + X11*W2 + X12*W3
   T(1,J1) = U1*V1
   T(1,J2) = U2*V1+U3*V2
   T(1,J3) = U1*V3
24 CONTINUE
C
C T(J3,J3) = T11; CALL EQU1 WITH A2(2,2), U(2,2) TO GET A2
C T(J3,J3) = T11
C CALL EQU1(T,P,N,J1,A2,U11*U22,M1,M2
GOTO 50
C
C N1=2, N2=1 : H = [ 0 0 1 1 0 X11 ]
C [ 0 U2 U3 ] [ 0 1 X21 ]
C
C 30 S = 1 + X11 * X11
    Y = X21 * X21
    W11 = SQRT(S)
    W22 = SQRT((1 + Y) / S)
    W12 = 0
    W21 = X11 * (X21 / W11)
    U1 = 1.0 / (SQRT(S + Y))
    U1 = 1.0 / W11
    U3 = 1.0 / W22
    U2 = -(X11 * W21) / S
    W21 = -W21
    D = W11
    W11 = W22
    W22 = 0
    T33 = T(J3, J3)
C PERFORM HPT*HT
C
C DO 32 I=J1,N
    W1 = T(J1, J1)
    W2 = T(J2, J1)
    W3 = T(J3, J1)
    V1 = -X11 * W1 - X21 * W2 + W3
    V2 = W1 + X11 * W3
    V3 = W2 + X21 * W3
    T(J1, J1) = U1 * V1
    T(J2, J1) = U1 * V2
    T(J3, J1) = U2 * V2 + U3 * V3
    CONTINUE
C DO 34 I=1,J3
    W1 = T(I, J1)
    W2 = T(I, J2)
    W3 = T(I, J3)
    V1 = -X11 * W1 - X21 * W2 + W3
    V2 = W1 + X11 * W3
    V3 = W2 + X21 * W3
    T(I, J1) = U1 * V1
    T(I, J2) = U1 * V2
    T(I, J3) = U2 * V2 + U3 * V3
    CONTINUE
C C PERFORM P*HT
C
C DO 36 I=1,N
    W1 = P(I, J1)
    W2 = P(I, J2)
    W3 = P(I, J3)
    V1 = -X11 * W1 - X21 * W2 + W3
    V2 = W1 + X11 * W3
    V3 = W2 + X21 * W3
    P(I, J1) = U1 * V1
    P(I, J2) = U1 * V2
    P(I, J3) = U2 * V2 + U3 * V3
36 CONTINUE
C
C T(J1, J1) = T33; CALL EQUD1 WITH A1(2, 2), W(2, 2) TO GET A1'
C
T(J1, J1) = T33
CALL EQUD1(T, P, N, J2, A1, W, W11, W22, MT, NP)
GOTO 50
C
C M1=2, M2=2 : H = [ 0 0 U1 0 ] [ 1 0 X11 X12 ]
C
C [ 0 0 U2 U3 ] [ 0 1 X21 X22 ]
C
C
C 40 CONTINUE
D = X11*X22-X12*X21
S = 1+X11*X11
D = X22*X22+D*S
Z = X12*X12
R = X21*X21
Y = S+Z
W11 = SQRT(Y)
W22 = SQRT((1.0+D+R)/Y)
W12 = 0.0
W21 = (X11*X21+X12*X22)/W11
Y = S+R
U11 = SQRT(Y)
U22 = SQRT((1.0+D+Z)/Y)
U21 = 0.0
U12 = (X11*X12+X21*X22)/U11
U1 = 1.0/U11
U3 = 1.0/U22
U2 = -U12/(U11*U22)
V1 = 1.0/U11
V3 = 1.0/U22
V2 = -U21/(U11*U22)
W21 = -U21
W11 = H11
W12 = H22
W22 = Y

C
C PERFORM H*T*H
C
DO 42 I=J1, N
   W1 = T(J1, I)
   W2 = T(J2, I)
   W3 = T(J3, I)
   W4 = T(J4, I)
   Y1 = -X11*W1 - X21*W2 + W3
   Y2 = -X12*W1 - X22*W2 + W4
   Y3 = W1 + X11*W3 + X12*W4
   Y4 = W2 + X21*W3 + X22*W4
   T(J1, I) = U1*Y1
   T(J2, I) = U2*Y1 + U3*Y2
   T(J3, I) = U1*Y3
   T(J4, I) = U2*Y3 + U3*Y4
42 CONTINUE

DO 44 I=1, J4
W_1 = T(I,J_1)
W_2 = T(I,J_2)
W_3 = T(I,J_3)
W_4 = T(I,J_4)
V_1 = -X_11*W_1 - X_21*W_2 + W_3
V_2 = -X_12*W_1 - X_22*W_2 + W_4
V_3 = W_1 + X_11*W_3 + X_12*W_4
V_4 = W_2 + X_21*W_3 + X_22*W_4
T(I,J_1) = U_1*V_1
T(I,J_2) = U_2*V_1 + U_3*V_2
T(I,J_3) = U_1*V_3
T(I,J_4) = U_2*V_3 + U_3*V_4

44   CONTINUE
C
C PERFORM P*HT
C
DO 46 I = 1, N
   W_1 = P(I,J_1)
   W_2 = P(I,J_2)
   W_3 = P(I,J_3)
   W_4 = P(I,J_4)
   V_1 = -X_11*W_1 - X_21*W_2 + W_3
   V_2 = -X_12*W_1 - X_22*W_2 + W_4
   V_3 = W_1 + X_11*W_3 + X_12*W_4
   V_4 = W_2 + X_21*W_3 + X_22*W_4
   P(I,J_1) = U_1*V_1
   P(I,J_2) = U_2*V_1 + U_3*V_2
   P(I,J_3) = U_1*V_3
   P(I,J_4) = U_2*V_3 + U_3*V_4

46   CONTINUE
C
C CALL EQUDI WITH A1, W TO GET A1', A2, V TO GET A2'
C
   CALL EQUDI(T,P,N,J_1,A_2,W,V_1,U_2,V_2,U_3,V_3,U_4,V_4)
   CALL EQUDI(T,P,N,J_3,A_1,W,W_1,U_2,W_2,U_3,W_3,U_4,W_4)
50   RETURN
END
SUBROUTINE SWAPB(T,P,M,J1,N1,N2,MT,MP)
REAL T(N,T),P(N,P)
INTEGER MP,MT,M,J1,N1,N2
C GIVEN T IN SCHUR FORM SWAPB SWAPS ADJACENT DIAGONAL BLOCKS T1
C AND T2 IN MATRIX T BEGINNING IN ROW J1 BY ORTHOGONAL SIMILIARITY
C TRANSFORMATIONS THAT PRESERVES THE SCHUR FORM OF T.  THE
C DIMENSION OF BLOCK T1 IS N1 BY N1 AND T2 IS N2 BY N2. THE
C PARAMETERS IN THE CALLING SEQUENCE ARE (STARRED PARAMETERS ARE
C ALTERED BY THE SUBROUTINE)
C *T  THE MATRIX WHOSE BLOCKS ARE BEING SWAPPED.
C **T  THE ARRAY INTO WHICH THE TRANSFORMATIONS
C  ARE TO BE ACCUMULATED.
C N  THE ORDER OF THE MATRIX T.
C J1  THE POSITION OF THE BLOCKS.
C N1  SIZE OF THE FIRST BLOCK.
C N2  SIZE OF THE SECOND BLOCK.
C MT  THE FIRST DIMENSION OF THE ARRAY T.
C MP  THE FIRST DIMENSION OF THE ARRAY P.

METHOD:
ALGORITHM 2 OF "PROGRAMS TO SWAP DIAGONAL BLOCKS" WITH
SPECIAL FORMULA FOR THE DIAGONAL BLOCKS

SUBPROGRAMS:
TXDXT, EQUI1

INTERNAL VARIABLES:
REAL X(2,2),U(4),UU(4),D,G,X11,X22,X12,X21,HALF,Y1,Y2,Y3
REAL W(2,2),H11,W12,W21,W22
REAL V(2,2),V11,V12,V21,V22,TEMP,T11,T22,T33
REAL A(2,2),B(2,2)
EQUIVALENCE (X(1,1),X11),(X(1,2),X12),(X(2,1),X21),(X(2,2),X22)
EQUIVALENCE (W(1,1),W11),(W(1,2),W12),(W(2,1),W21),(W(2,2),W22)
EQUIVALENCE (V(1,1),V11),(V(1,2),V12),(V(2,1),V21),(V(2,2),V22)
INTEGER K
HALF = 0.5
C SOLVE X FOR [1 -X] [T1 T2] [1 X] = [T1 0] BY CALLING TXDXT
C [ T0 1 ] [0 T2] [1 T1] [0 T1]
C CALL TXDXT(T,M,J1,N1,N2,X,IZ,NT)
C IF IZ=0, A1 AND A2 ARE TOO CLOSE TO SWAP
C IF(IZ.EQ.0) GOTO 50
K=M1+N1+N2-2
J2 = J1+1
J3 = J1+N1
J4 = J3+1
IF(N1.EQ.2) THEN
   A1(1,1)=T(J1,J1)
   A1(1,2)=T(J1,J2)
   A1(2,1)=T(J2,J1)
   A1(2,2)=T(J2,J2)
ENDIF
IF(M2.EQ.2) THEN
  A2(1, 1)-T(J3,J3)
  A2(1, 2)-T(J3,J4)
  A2(2, 1)-T(J4,J3)
  A2(2, 2)-T(J4,J4)
ENDIF

GOTO (10,20,30,40), K

C 1,1 : H=1-UU*O, H [ X11 ] = [ S ], S=SIGN(X11)*SQRT(1+X11^2)

C

10 S= SIGN(SQRT(1.0+X11*XI1),X11)
  T11 = T(J1,J1)
  T22 = T(J2,J2)
  U(1) = S - X11
  IF((X11/S).GT.HALF) U(1) = 1.0/(S+X11)
  U(2) = 1
  D = U(1)*S
  CALL HOUSE(T,P,H,J1,U,2,D,NT,MP)

C SWAP DIAGONAL ELEMENTS

C

T(J1,J1) = T22
T(J2,J2) = T11
GOTO 50

C 1,2 : [ X11 X12 ] H = [ 0 0 S ], S=-SIGN(X12)*SQRT(1+X11^2+X12^2)

C

20 Y = 1.0*X11*XI1
  S = SIGN(SQRT(Y+X12*X12),-X12)
  U(1) = 1
  U(2) = X11
  U(3) = X12 + S
  IF((-X12/S).GT.HALF) U(3) = Y/(S - X12)
  D = U(3)*S
  V11 = -X11
  V22 = -X11/U(3)
  V21 = 1.0
  V12 = -X12-1.0/U(3)
  T11 = T(J1,J1)
  CALL HOUSE(T,P,H,J1,U,3,D,NT,MP)

C T(J3,J3) = T11; CALL EQUA WITH R2(2,2), U<2,2> TO GET R2'

C

T(J3,J3) = T11
  CALL EQUA(T,P,H,J1,R2,U,S,NT,MP)
GOTO 50

C 2,1 : H [-X11] = [ S ], S = SIGN(X11)*SQRT(1+X11*X11+X21*X21)

C

30 T33 = T(J3,J3)
  Y = 1.0*X21*X21
  S = SIGN(SQRT(Y+X11*X11),X11)
  U(1) = S - X11
  IF((X11/S).GT.HALF) U(1) = Y/(S+X11)
U(2) = -x21
U(3) = 1
D = U(1)*S
W22 = X21/U(1)
W11 = X21
W12 = -X11+1.0/U(1)
W21 = -1.0
CALL HOUSE(T,P,N,J1,U,3,D,NT,MP)

C T(J1,J1) = T33; CALL EQU1 WITH A1(2,2), W(2,2) TO GET A1'
C
T(J1,J1) = T33
CALL EQU1(T,P,N,J2,A1,W,S,NT,MP)
GOTO 50
C
2,2 : H1 [-X11] = [ S1],
[1 ] = [ 0]
[0 ] = [ 0]

Y = 1.0*X21*X21
SX = SIGN(SORT(Y*X11*X11),X11)
U(1) = SX - X11
IF((X11/SX).GT.HALF) U(1) = Y/(SX*X11)
U(2) = -x21
U(3) = 1
D = U(1)*S
CALL HOUSE(T,P,N,J1,U,3,D,NT,MP)
TEMP = (T(J4,J1)*U(1)+T(J4,J2)*U(2)+T(J4,J3)*U(3))/D
T(J4,J1) = T(J4,J1) - TEMP*U(1)
T(J4,J2) = T(J4,J2) - TEMP*U(2)
T(J4,J3) = T(J4,J3) - TEMP*U(3)
Y1 = -(X11*X12*X21*X22)/SX
Y3 = (X12*U(1)-X11*X22)/0
Y2 = -X22+X11*Y3

C H2 [Y1] = [Y1], WHERE Y = H1*[-X12], S = -SIGN(Y2)*SORT(1+Y2**2+Y3**2)
C
[Y2] = [ S]
[ 0]
[Y3] = [ 0]
[Y4] = [ 1]

Y = 1.0*Y3*Y3
SY = SIGN(SORT(Y2*Y2+Y),-Y2)
UU(2) = SY+Y2
IF (ABS(Y2/SY).GT.HALF) UU(2) = Y/(SY-Y2)
UU(3) = Y3
UU(4) = 1
G = UU(2)*SY
CALL HOUSE(T,P,N,J2,UU(2),3,G,NT,MP)
TEMP = (UU(2)*T(J2,J1)+UU(3)*T(J3,J1)+UU(4)*T(J4,J1))/G
T(J2,J1) = T(J2,J1) - TEMP*UU(2)
T(J3,J1) = T(J3,J1) - TEMP*UU(3)
T(J4,J1) = T(J4,J1) - TEMP*UU(4)
W11 = X22 - 1.0/UU(2)
W22 = X11 - (SY-X22)/(UU(1)*UU(2))
W12 = -X12*X21/(UU(1)*UU(2))
W21 = -X22+Y3/UU(2)
W11 = -SX
CALL EQUID WITH A(2,2), W(2,2) TO GET A1', R2, V TO GET R2'
       CALL EQUID(T,P,M,J1,R2,U,Y,NT,MP)
       CALL EQUID(T,P,M,J3,A1,W,Y,NT,MP)
   50    RETURN
     END
SUBROUTINE HOUSE(T,P,M,J1,U,K,D,NT,MP)
     REAL T(NT,N),P(NT,N),U(K),D
     INTEGER K,J1,N,NT,MP

     THIS SUBROUTINE PERFORMS HOUSEHOLDER TRANSFORMATION ON T AND ACCUMULATE
     THE TRANSFORMATION IN P:

     T = (1-UU/D)T(1-UU/D)
     P = P(1-UU/D), WHERE U STANDS FOR THE TRANSPOSE OF U.

     THE TRANSFORMATION BEGINS AT T(J1,J1). THE LENGTH OF U IS K.

     INTERNAL VARIABLES:

     REAL S,ZERO
     INTEGER I,J
     ZERO=0.0

     IF(D.EQ.ZERO) GOTO 100
     DO 30 J=J1,N
          S=0.0
           DO 10 I=1,K
               S=S+U(I)*T(J1+I-1,J)
               S=S/D
          10    DO 20 I=J1,J1+K-1
               T(I,J)=T(I,J)-S*(I-J+1)
          20    CONTINUE
     30    CONTINUE

     COLUMN MODIFICATION
     DO 50 I=1,J1+K-1
          S=0.0
          DO 40 J=1,K
               S=S+U(J)*T(I,J+J-1)
               S=S/D
          40    DO 50 J=J1,J1+K-1
               T(I,J)=T(I,J)-S*(J-J1+1)
      50    CONTINUE

     ACCUMULATION
     DO 70 I=1,N
          S=0.0
           DO 60 J=1,K
               S=S+U(J)*P(I,J+J-1)
       60    CONTINUE
     70    CONTINUE
S=S/D

DO 90 J=J1,J1+K-1
   P(I,J)=P(I,J)-S*U(J-J1+1)
90    CONTINUE
100   RETURN
      END
SUBROUTINE EQU1(T,P,N,J1,A,W,DETH,MT,MP)
REAL T(N,T),P(N,N),A(2,2),W(2,2),DETH
INTEGER J1,N,MT,MP
C THIS SUBROUTINE PERFORMS A UNITARY TRANSFORMATION (A REFLECTION)
C TO MAKE THE DIAGONAL ELEMENTS IN W**T**1 - 1 EQUAL AND PUT THE
C RESULT IN T.
C | -COS(Q)  SIN(Q) | | T11  T12  | | -COS(Q)  SIN(Q) |
C | SIN(Q)  COS(Q) | | T21  T22  | | SIN(Q)  COS(Q) |
C THE TRANSFORMATION IS ACCUMULATED IN (POSTMULTIPLIED TO) P. THE
C INPUT A MUST HAVE EQUAL DIAGONAL ELEMENT. HERE DETH IS THE
C DETERMINANT OF W.
C
F77 GENERIC FUNCTIONS: ATAN, COS, SIN
C
C INTERNAL VARIABLES
C
INTEGER I,J,J2
REAL Q,U,S,C,Z,V1,V2,TEMP,ZERO,ONE,HALF
REAL A11,A12,A21,A22,W11,W12,W21,W22
A11 = A(1,1)
A12 = A(1,2)
A21 = A(2,1)
A22 = A(2,2)
W11 = W(1,1)
W12 = W(1,2)
W21 = W(2,1)
W22 = W(2,2)
ONE = 1.0
HALF = 0.5
ZERO = 0.0
J2 = J1+1
C
DETERMINE THE ANGLE Q
C
Z = A21*W12*W22 - A12*W11*W21
IF(U+Z.EQ.Z) THEN
  Q = ATAN(1.0)
ELSE
  Q = HALF*ATAN((Z+Z)/U)
ENDIF
S = SIN(Q)
C = COS(Q)
C
NEW DIAGONAL BLOCK
C
Z = C/S
Z = S/C
V2 = (-A21*W22*(W22+W12*Z)+A12*W21*(W21+W11*Z))/DETH
Z = A12*A21
IF(ABS(V1).GT.ABS(V2)) THEN
  V2 = Z/V1
ELSE
  V1 = Z/V2
ENDIF
C ROW MODIFICATION
C
DO 10 J=J2+1,N
   TEMP = -C*T(J1,J)+S*T(J2,J)
   T(J2,J) = S*T(J1,J)+C*T(J2,J)
   T(J1,J) = TEMP
10 CONTINUE
C COLUMN MODIFICATION
C
DO 20 I=1,J1-1
   TEMP = T(I,J1)*(-C)+T(I,J2)*S
   T(I,J2) = T(I,J1)*S+T(I,J2)*C
   T(I,J1) = TEMP
20 CONTINUE
T(J1,J1) = A11
T(J1,J2) = U1
T(J2,J1) = U2
T(J2,J2) = A11
C ACCUMULATION
C
DO 30 I=1,N
   TEMP = P(I,J1)*(-C)+P(I,J2)*S
   P(I,J2) = P(I,J1)*S+P(I,J2)*C
   P(I,J1) = TEMP
30 CONTINUE
40 RETURN
END
SUBROUTINE TXT(T,N,J1,M1,N2,J2,N2)
REAL T(N,N),X(2,2)
INTEGER N,M,N1,M2,N2,J1,J2
C THIS SUBROUTINE SOLVES FOR N1 BY N2 MATRIX X IN T1*X - X*T2 = T12.
C T1 AND T2 BEGIN IN ROWS J1 AND J2 RESPECTIVELY, T12 IS THE UPPER
C TRIANGULAR PART BETWEEN T1 AND T2. THIS PROGRAM ASSUMES THE
C DIAGONALS OF T1 (T2) ARE EQUAL. THE PARAMETERS IN THE CALLING
C SEQUENCE ARE (STARTED PARAMETERS ARE ALTERED BY THE SUBROUTINE)
C
C *T  INPUT MATRIX
C N  THE ORDER OF THE MATRIX T
C J1  THE POSITION OF THE BLOCKS.
C M1  SIZE OF THE FIRST BLOCK.
C M2  SIZE OF THE SECOND BLOCK.
C *X  OUTPUT MATRIX
C *I12 OUTPUT INDICATOR: 1-X SOLVED SUCCESSFULLY,
C ZER0-OVERFLOW MAY OCCUR.
C NT  THE FIRST DIMENSION OF THE ARRAY T.
C
C INTERNAL VARIABLES:
C
REAL D, DEL, BET1, BET2, GAM1, GAM2, T1, T2, TAU, ETA, PHI, DSQ
REAL P1, P2, P3, P4, P5, P6, P7, P8, P9
INTEGER I, J, K
ZERO=0.0
I1= 1
C
C INITIALIZE
C
DO 1 I=1,2
DO 1 J=1,2
1 X(I,J)=ZERO
J2= J1+M1
DEL=T(J1,J1)-T(J2,J2)
B11=T(J1,J2)
IF(N1.EQ.2) THEN
J1P1 = J1+1
BET1 = T(J1,J1P1)
GAM1 = T(J1P1,J1)
B21 = T(J1P1,J2)
ENDIF
IF(N2.EQ.2) THEN
J2P1 = J2+1
BET2 = T(J2,J2P1)
GAM2 = T(J2P1,J2)
B12 = T(J1,J2P1)
IF(N1.EQ.2) B22=T(J1P1,J2P1)
ENDIF
K=M1+M1+M2-2
GOTO (10,20,30,40), K
C
C BY 1: APH1*X - X*APH2 = B11
C
10 IF(DEL.EQ.ZERO) THEN
I1 = 0
ELSE
X(1,1) = B11/DEL
END IF
GOTO 50
C
C 1 BY 2: APH1*(X11 X12) - [X11 X12]*[APH2 BET2) = [B11 B12]
C [GAM2 APH2]
C
20 D = DEL*DEL - BET2*GAM2
IF(D.EQ.ZERO) THEN
IZ = 0
ELSE
X(1,1) = (DEL *B11 + GAM2*B12)/D
X(1,2) = (BET2*B11 + DEL *B12)/D
END IF
GOTO 50
C
C 2 BY 1: [APH1 BET1]*[X11] - [X11]*[APH2 = [B11]
C [GAM1 APH1] [X21] [X21] [B21]
C
30 D = DEL*DEL - BET1*GAM1
IF(D.EQ.ZERO) THEN
IZ = 0
ELSE
X(1,1) = (DEL *B11 - BET1*B21)/D
X(2,1) = (-GAM1*B11 + DEL *B21)/D
END IF
GOTO 50
C
C 2 BY 1: [APH1 BET1]*[X11 X12) - [X11 X12]*[APH2 BET2) = [B11 B12]
C [GAM1 APH1] [X21 X22] [X21 X22] [GAM2 APH2] [B21 B22]
C
40 DSQ = DEL*DEL
T1 = BET1*GAM1
T2 = BET2*GAM2
TAU = DSQ + (T2 - T1)
D = TAU*TAU - 4*DSQ*T2
IF(D.EQ.ZERO) THEN
IZ = 0
ELSE
ETR = DSQ - (T1+T2)
PH1 = DSQ + (T1-T2)
T2 = -(DEL+DEL)
T1 = T2*BET1
T2 = T2*GAM1
P1 = ETR*DEL
P2 = PH1*GAM2
P3 = -TAU*BET1
P4 = T1*GAM2
P5 = PH1*BET2
P6 = T1*BET2
P7 = -TAU*GAM1
P8 = T2*GAM2
P9 = T2*BET2
X(1,1) = (P1*B11+P2*B12+P3*B21+P4*B22)/D
X(1,2) = (P5*B11+P1*B12+P6*B21+P3*B22)/D
X(2,1) = (P7*B11+P8*B12+P1*B21+P2*B22)/D
X(2,2) = (P9*B11+P7*B12+P5*B21+P1*B22)/D
CC
**COMPUTE RESIDUAL**

\[ R_1 = B_{11} - (\text{DEL} \cdot X(1,1) - \text{GAM2} \cdot X(1,2) \cdot \text{BET1} \cdot X(2,1)) \]
\[ R_2 = B_{12} - (-\text{BET2} \cdot X(1,1) \cdot \text{DEL} \cdot X(1,2) \cdot \text{BET1} \cdot X(2,2)) \]
\[ R_3 = B_{21} - (\text{GAM1} \cdot X(1,1) \cdot \text{DEL} \cdot X(2,1) - \text{GAM2} \cdot X(2,2)) \]
\[ R_4 = B_{22} - (\text{GAM1} \cdot X(1,2) \cdot \text{BET2} \cdot X(2,1) \cdot \text{DEL} \cdot X(2,2)) \]

**PERFORM ONE ITERATION IF R* IS NOT SMALL COMPARED TO B***

\[ T_1 = \text{ABS}(B_{11}) \cdot \text{ABS}(B_{12}) \cdot \text{ABS}(B_{13}) \cdot \text{ABS}(B_{14}) \]
\[ T_2 = 0.0005 \cdot (\text{ABS}(R_1) \cdot \text{ABS}(R_2) \cdot \text{ABS}(R_3) \cdot \text{ABS}(R_4)) \]
\[ \text{IF}(T_1 + T_2 \geq T_1) \text{ THEN} \]
\[ X(1,1) = X(1,1) + (P_1 \cdot R_1 + P_2 \cdot R_2 + P_3 \cdot R_3 + P_4 \cdot R_4) / D \]
\[ X(1,2) = X(1,2) + (P_5 \cdot R_1 + P_6 \cdot R_2 + P_7 \cdot R_3 + P_8 \cdot R_4) / D \]
\[ X(2,1) = X(2,1) + (P_9 \cdot R_1 + P_{10} \cdot R_2 + P_{11} \cdot R_3 + P_{12} \cdot R_4) / D \]
\[ X(2,2) = X(2,2) + (P_9 \cdot R_1 + P_{10} \cdot R_2 + P_{11} \cdot R_3 + P_{12} \cdot R_4) / D \]

**ENDIF**

**END**
The real Schur form of a real square matrix is block upper triangular. We study techniques for performing orthogonal similarity transformations that preserve block triangular form but alter the order of the eigenvalues along the (block) diagonal.
END
Filmed
5-89
DTIC