Parallel-Vector Unsymmetric Eigen-Solver on High Performance Computers

Duc T. Nguyen and Qin Jiangning

Old Dominion University Research Foundation
Norfolk, Virginia

Contract NAS1-18584, Task 122
February 1993

(NASA-CR-191417) PARALLEL-VECTOR UNSYMMETRIC EIGEN-SOLVER ON HIGH PERFORMANCE COMPUTERS Final Report (Old Dominion Univ.) 34 p
PARALLEL-VECTOR UNSYMMETRIC EIGEN-SOLVER ON HIGH PERFORMANCE COMPUTERS

Duc T. Nguyen
Qin Jiangning
Center for Multidisciplinary Parallel-Vector Computation
Civil Engineering Department
Old Dominion University

Abstract

The popular QR algorithm for solving all eigenvalues of an unsymmetric matrix is reviewed. Among the basic components in the QR algorithm, it has been concluded from this study, that the reduction of an unsymmetric matrix to a Hessenberg form (before applying the QR algorithm itself) can be done effectively by exploiting the vector speed and multiple processors offered by modern high-performance computers.

Numerical examples of several test cases have indicated that the proposed parallel-vector algorithm for converting a given unsymmetric matrix to a Hessenberg form offers computational advantages over the existing algorithm. The time saving obtained by the proposed method is increased as the problem size increased.

I. Introduction

The algorithms for symmetric matrices [1-3] are highly satisfactory in practice. By contrast, it is impossible to design equally satisfactory algorithms for the nonsymmetric cases, which is needed in Controls-Structures Interaction (CSI) applications [1,4]. There are two reasons for this. First, the eigenvalues of a nonsymmetric matrix can be very sensitive to small changes in the matrix elements. Second, the matrix itself can be defective, so that there is no complete set of eigenvectors.

There are several basic building blocks in the QR algorithm, which is generally regarded as the most effective algorithm, for solving all eigenvalues of a real, unsymmetric matrix. These basic components of the QR algorithm are reviewed in Section II. Basic techniques to exploit the vector speed and multiple processors offered by modern high-performance computers are explained in Section III. An analysis of the Hessenberg reduction component in the QR algorithm is given in Section IV where both vector and parallel techniques are incorporated into the Hessenberg reduction component. Numerical examples are provided in Section V to evaluate the performance of the proposed method over the existing one. Conclusions and recommendations are given in Section VI. Finally, a listing of the Hessenberg reduction algorithm (in the form of Fortran coding) is provided in the appendix.

II. Basic Components of the QR Algorithm [3,5]

2.1 Balancing:
The idea of balancing is to use similarity transformations to make corresponding rows and columns of the matrix have comparable norms, thus reducing the overall norm of the matrix while leaving the eigenvalues unchanged.

The time taken by the balanced procedure is insignificant as compared to the total time required to find the eigenvalues. For this reason, it is strongly recommended that a nonsymmetric matrix need to be balanced before even attempting to solve for eigensolutions.

2.2 Reduction to Hessenberg form:

The strategy for finding the eigensolution of an unsymmetric matrix is similar to that of the symmetric case. First we reduce the matrix to a simpler Hessenberg form, and then we perform an iterative procedure on the Hessenberg matrix. An upper Hessenberg matrix has zeros everywhere below the diagonal except for the first subdiagonal. For example, in the 6 x 6 case, the nonzero elements are:

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

Thus, a procedure analogous to Gaussian elimination can be used to convert a general unsymmetric matrix to an upper Hessenberg matrix. The detailed coding of the Hessenberg reduction procedure is listed in subroutine OELMHS of the appendix.

Once the unsymmetric matrix has already been converted into the Hessenberg form, the QR algorithm [3,5] itself can be applied on the Hessenberg matrix to find all the real and complex eigenvalues. For completeness, detailed coding of the QR algorithm on the Hessenberg matrix is listed in subroutine HQR of the appendix.

III. Basic Techniques For Vector and Parallel Speeds

In this section, a simple example of matrix times vector is used to explain some basic vector and parallel techniques which are useful for Hessenberg reduction algorithm.

Given a 3x3 Matrix \( A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \) and a vector \( x = \{1,0,0\}^T \)

Here, the dimension of the system is \( N=3 \). The objectives are to develop efficient parallel - vector matrix times vector subroutines.
3.1 Row-by-Row conventional approach:

\[
DO 1 \quad I=1,N
\]
\[
DO 2 \quad J=1,N
\]
\[
B(I) = B(I) + A(I,J) \times x(J)
\]
\[2 \quad \text{Continue}\]
\[1 \quad \text{Continue}\]

It should be emphasized here that in this approach, the value of \(B(I)\) corresponds to the final answer.

3.2 Column-by-Column conventional approach:

\[
DO 1 \quad J = 1,N
\]
\[
DO 2 \quad I = 1,N
\]
\[
B(I) = B(I) + A(I,J) \times x(J)
\]
\[2 \quad \text{Continue}\]
\[1 \quad \text{Continue}\]

It should be emphasized here that in this approach, the value of \(B(I)\) does NOT correspond to the final answer. \(B(I)\) only gives the partial (or incomplete) answer and it will give the final answer only if all values of \(J\) have been executed. It is also observed that \(x(J)\) is a constant (with respect to loop 2), thus the operations involved in loop 2 can be stated generally as: A new vector \(B = \text{Old vector } B + \text{Constant } \times \text{another vector } A\).

3.3 Row-by-Row "vector unrolling" approach:

Assuming the dimension \(N\) of the system is large, say \(N = 600\), then the algorithm in Section 3.1 can be modified to improve the vector speed as following:

\[
\text{NUNROL} = 2
\]
\[
DO 1 \quad I = 1,N, \text{ NUNROL}
\]
\[
DO 2 \quad J = 1,N
\]
\[
B(I) = B(I) + A(I,J) \times x(J)
\]
\[
B(I+1) = B(I+1) + A(I+1,J) \times x(J)
\]
\[2 \quad \text{Continue}\]
\[1 \quad \text{Continue}\]

The operations involved inside loop 2 is referred to as "dot product" operations.

3.4 Column-by-Column "loop-unrolling" approach

The algorithm in Section 3.2 can be modified to improve the vector speed performance
The operations involved inside loop 2 is referred to as "saxpy" operations.

3.5 Parallel-vector loop-unrolling approach:

For multiple processors, the algorithm in Section 3.4 can be modified to take advantage of parallel speed (in addition to vector speed)

\[ NUNROL = 2 \]
\[ DO 1 \quad J = 1, N, \quad NUNROL \]
\[ DO 2 \quad I = 1, N \]
\[ B(I) = B(I) + A(I, J) \times x(J) \]
\[ \quad + A(I, j+1) \times x(J+1) \]

2 Continue
1 Continue

In this algorithm, each value of the index J (of loop 1) is assigned to different processors for parallel computation.

IV. An Analysis of the Hessenberg Reduction Algorithm

A careful look into the Hessenberg reduction algorithm of Section 2.2 and subroutine OELMHS of the appendix will reveal that the most intensive computations of Subroutine OELMHS occur in loops 140 and 150 of the code. Furthermore, the Fortran statement inside loop 150 can be generally expressed as:

\[ A(J, M) = A(J, M) + Y \times A(J, I) \]

or

\[ A \text{ new vector } A(J, -) = \text{ old vector } A(J, -) + (\text{a constant}) \times \text{ another vector } A(J, *) \]

Thus, one can immediately see the similarity between loops 160 & 150 of Subroutine OELMHS and loops 1 & 2 of the matrix times vector algorithm presented in Section 3.2. From the experience we have had in section 3.5, we can therefore similarly apply the parallel computations in loop 160 and loop-unrolling (here NUNROL = 8 is used) for vector computations in loop 150 of subroutine OELMHS.

For completeness, the entire parallel-vector version of the Hessenberg reduction, and the original QR algorithms are listed in the Appendix.
V. Numerical Examples

In order to evaluate the numerical accuracy and the performance of the new parallel-vector Hessenberg Reduction portion of the QR algorithm, the following numerical tests are performed.

Example 1:
Find all eigenvalues of the following 2 x 2 unsymmetric matrix

\[ A = \begin{bmatrix} 2 & -6 \\ 8 & 1 \end{bmatrix} \]

The analytical eigen-value solution for this problem is:

\[ \lambda_1 = 1.5 + 6.91 \hat{i} \]
\[ \lambda_2 = 1.5 - 6.91 \hat{i} \]

which also matches with the computer solution.

Example 2:
In this example, the unsymmetric matrix \([A]_{N \times N}\) is automatically generated for any dimension \(N\) of the matrix \([A]\) (please refer to the code given in the Appendix). The accuracy and the performance of the new parallel-vector Hessenberg reduction algorithm is compared to the original subroutine. Since the QR algorithm itself is highly sequential, no attempts to parallelize and vectorize the QR algorithm have been made. However, the total solution time of the complete unsymmetric eigensolution process (\(=\) Hessenberg Reduction Time and QR Time) are also presented in Tables 1 and 2.
Table 1: Vector Performance on the Alliant Using etime (t), fortran -DAS -O -alt -l -OM where:
- l option will tell which loop does not vectorize
- OM option will not print warning messages

<table>
<thead>
<tr>
<th>Size N</th>
<th>&quot;Original&quot; CSI version (HR - Hessenberg Reduction Time)</th>
<th>&quot;New&quot; version</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 x 100</td>
<td>(0.41 sec, 0.97 sec)</td>
<td>(0.39 sec, 0.97 sec)</td>
</tr>
<tr>
<td>200 x 200</td>
<td>(2.210 sec, 5.195 sec)</td>
<td>(2.22 sec, 5.19 sec)</td>
</tr>
<tr>
<td>400 x 400</td>
<td>(16.9, 33.9)</td>
<td>(14.00, 33.93)</td>
</tr>
<tr>
<td>600 x 600</td>
<td>(55.48, 94.20)</td>
<td>(51.0, 94.2)</td>
</tr>
<tr>
<td>800 x 800</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Table 2: Parallel-Vector Performance on Cray-YMP (Reynolds) Using tsecnd.

<table>
<thead>
<tr>
<th>Size N</th>
<th>&quot;Original&quot; CSI version</th>
<th>&quot;New&quot; version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(HR = Hessenberg)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reduction Time</td>
<td>1 Cray-YMP Processor</td>
</tr>
<tr>
<td></td>
<td>QR Time</td>
<td>Processor</td>
</tr>
<tr>
<td>100 x 100</td>
<td>0.02 sec</td>
<td>0.02 sec</td>
</tr>
<tr>
<td></td>
<td>0.07 sec</td>
<td>0.07 sec</td>
</tr>
<tr>
<td>200 x 200</td>
<td>0.12</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>0.42</td>
<td>0.42</td>
</tr>
<tr>
<td>400 x 400</td>
<td>1.19</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>3.15</td>
<td>3.19</td>
</tr>
<tr>
<td>600 x 600</td>
<td>2.90</td>
<td>2.28</td>
</tr>
<tr>
<td></td>
<td>7.12</td>
<td>7.12</td>
</tr>
<tr>
<td>800 x 800</td>
<td>14.34</td>
<td>5.17</td>
</tr>
<tr>
<td></td>
<td>33.25</td>
<td>33.31</td>
</tr>
</tbody>
</table>

VI. Conclusions and Recommendations:

The most popular and effective procedure to solve all eigenvalues of an unsymmetric matrix involved 2 major tasks, namely Hessenberg reduction form and QR algorithm on the Hessenberg matrix. In general, QR algorithm requires between 2 to 3 times more computational effort than the Hessenberg reduction algorithm.

In this study, the parallel and vector speeds of the Hessenberg reduction algorithm has been developed and implemented on the Alliant and Cray-YMP (Reynolds) computers. Numerical results have indicated that the proposed parallel-vector Hessenberg reduction algorithm does offer computational advantages (without losing its accuracy) as compared to the existing algorithm. The time saving is more significant as the problem size increased. Further research work is critically needed to improve the unsymmetric eigensolution procedure (using the QR, or another better, new parallel algorithm).

Acknowledgments:

This research work was supported by a NASA Task NAS 1-18584-122, and Mr. Joseph Walz was the NASA technical monitor.
References:


APPENDIX

Parallel-Vector Hessenberg Reduction And Sequential QR Algorithm [5]
FILE: UNSEIG FRC AI OLD DOMINION UNIVERSITY

C------ PARALLEL/VECTOR UNSYMMETRIC EIGENSLVER by Qin & Nguyen, May 1992
...... This is a working version of "unsymmetrical" eigen-solver
...... on the sun 386 work station. On the Cray-YMP (Reynold or Sabre),
...... this "exact" same version should offer good vector & parallel
...... speed (only for subroutine to perform Hessenberg reduction).
...... For SMALL problems, the improvements due to parallel-vector
...... Hessenberg is NOT MUCH. However, for LARGE problems, since the
...... Hessenberg reduction timing becomes more important (as compared to
...... the TOTAL eigen-solution time), the total time saving for the entire
...... eigen-solution process is also very significant.
...... Since this version was developed specifically for CSI applications
...... (according to Peiman's specifications/requirements), ALL EIGENVALUES
...... (and NONE of the corresponding EIGENVECTORS) of an N by N squared
...... unsymmetric matrix are found.
...... "ARTIFICIAL" datas of various sizes (N = 2 ----> 800) with ALL REAL
...... and MIXED REAL & COMPLEX eigenvalues have been verified (by comparing
...... the results obtained by the original unsym. eigen-sol. taken from
...... ORACLE and the modified version from the ODU team, and also by HAND
...... CALCULATION for the size N = 2)
Force PVQR of NP ident ME
Shared REAL A(1000000), WK(1000, 2)
Shared REAL ER(1000), EI(1000), EIG(1000)
Shared REAL EPS, ERRCK
Shared INTEGER N, NM, NMM, NMAX, NST, MQ, IMODE, IERR, nguyen
End Declarations

C *** THIS IS THE PROGRAM CALL UNSYMMETRIC EIGENSLVER **********
Barrier
WRITE(*,*) 'N, IMODE (O=old version), nguyen (l=duc-s data) = '
READ (5, *) N, imode, nguyen
WRITE(*,*) 'N, IMODE NGUYEN = ', N, IMODE, nguyen
ERRCK = 0.0000001
eps = GETEPS(ibeta, it, irnd)
write(*,*) 'EPS = ', eps
write(*,101) N, imode
101 FORMAT(1x, 'INPUT PARAMETERS:', 1x, 'N = ', i5, ' - Size of System' )
   1 'IMODE = ', i5, ' - 0 is old sequential')
End Barrier
Forcecall RESV(N, N, A, ER, EI, WK, EERR, EIG, IMODE, nguyen)
Join
END
FUNCTION GETEPS(IBETA, IT, IRND)
a = 1.0
10 a = a + a
   if (((a+1.0) - a) - 1.0.eq.0.00) go to 10
   b = 1.0
20 b = b + b
   if ((a+b) - a.eq.0.00) go to 20
   qina = (a+b) - a
   ibeta = int(qina)
   beta = float(ibeta)
   it = 0
   b = 1.0
30 it = it + 1
   b = b * beta
if (((b+1.0)-b)-1.0.eq.0.00) go to 30
irnd=0
betaml=beta-1.0
if((a+betaml)-a.ne.0.00) irnd=1
betain=1.0/beta
a=1.0
do 40 i=1,it+3
   a=a*betain 
40 continue
50 if((1.0+a)-1.0.ne.0.00) go to 60
   a=a*beta
   go to 50
60 eps=a
   if((ibeta.eq.2) .or. (irnd.eq.0)) go to 70
   a=(a*(1.0+a))/(1.0+1.0)
   if((1.0+a)-1.0.ne.0.00) eps=a
   geteps=eps
   return
end

C***************************************************************

FORCESUB RESV(MAX,N,A,ER,E1,WK,IERR,EIG,IMODE,nguyen) of NP

$ ident ME
INTEGER MAX,N,IERR,IMODE
Shared Integer LOW,IGH,NACC
C
C ****
C FUNCTION RESV
C PARAMETERS MAX - MAXIMUM ROW DIMENSION OF A
C N - ORDER OF A
C A(MAX,N) - INPUT MATRIX (DESTROYED)
C ER(N) - CONTAINS REAL PART OF THE EIGENVALUES
C EI(N) - CONTAINS IMAGINARY PART OF THE EIGENVALUES
C WK(-) - WORKING STORAGE OF FOLLOWING DIMENSION
C DIMENSION 3*N IF ISV+ILV = 0
C DIMENSION N*(N+7) OTHERWISE
C IERR - INTEGER ERROR CODE
C = 0 NORMAL RETURN
C = -J J-TH EIGENVECTOR DID NOT CONVERGE.
C VECTOR SET TO ZERO. IF FAILURE OCCURS
C MORE THAN ONCE, INDEX FOR LAST
C OCCURRENCE IN IERR.
C = J J-TH EIGENVALUE HAS NOT BEEN
C DETERMINED AFTER 30 ITERATIONS
C OUTPUT FORMAT
C EIGENVALUES ARE STORED IN ASCENDING MAGNITUDE
C WITH COMPLEX CONJUGATES STORED WITH POSITIVE
C IMAGINARY PARTS FIRST. THE EIGENVECTORS ARE
C PACKED AND STORED IN V IN THE SAME ORDER AS
C THEIR EIGENVALUES APPEAR IN ER AND EI.
C ONLY ONE EIGENVECTOR IS COMPUTED FOR CONJUGATE
C CONJUGATES (FOR CONJUGATE WITH POSITIVE
C IMAGINARY PART). UPON ERROR EXIT -J, EIGEN-
C VALUES ARE CORRECT AND EIGENVECTORS
C ARE CORRECT FOR ALL NON-ZERO VECTORS.
C UPON ERROR EXIT J, EIGENVALUES ARE CORRECT
C BUT UNORDERED FOR INDICES IERR+1,IERR+2,...
N AND NO EIGENVECTORS ARE COMPUTED.

**REAL A(N,N),ER(N),EI(N),WK(N,2),EIG(1)**

End Declarations

**DIMENSION A(MAX,N),ER(N),EI(N),V(MAX,*),WK(N,*)**

**LOGICAL LTESTV**

**EQUIVALENCE (TESTV,LTESTV)**

**CQIN DATA TRUE,FALSE / '77777777777777777777'0, '00000000000000000000'0
CQIN +/-**

**DATA TRUE,FALSE / 77777777777777777777.0,0.00000000000000000000 /**

**PRELIMINARY REDUCTION**

**Barrier**

DO 2 J=1,N
DO 1 I=1,N
if (i.lt.j) then
  a(i,j)=1.37373737373710.0/(float(i+j))
else
  A(i,j)=0.97319731973110.0/(float(i+j+j/2))
endif
1 continue
2 continue
3 DO 3 i=1,n
3 a(i,i)=float(i)

**Duc T. Nguyen added this portion to test "complex" eigen-solution**

if (nguyen.eq.1) then
  DO 29 J=1,N
  DO 19 I=1,N
  if (i.lt.j) then
    a(i,j)=-1.37373737373710.0/(float(i+j))
  else
    A(i,j)=0.97319731973110.0/(float(i+j+j/2))
  endif
19 continue
29 continue
39 DO 39 i=1,n
39 a(i,i)=float(i)

a(1,1)=2.
a(1,2)=-6.
a(2,1)=8.
a(2,2)=1.

**SAVE A FOR NORM CHECK *****

low=1
igh=n
TIME0=O.O
End Barrier
t00=TSECND()

CALL QXZ146 (MAX,N,A,LOW,IGH,WK)
t11=TSECND()
if (imode.ne.0) then
Forcecall QXZ147 (MAX,N,LOW,IGH,A,WK(1,2),eig)
else
Forcecall OELMHS(MAX,N,LOW,IGH,A,WK(1,2))
endif
T22=TSECNĐ()
TIMEO=TIMEO+T22-TOO
write(6,*)'** ME CPU in QXZ146 = ',ME,T11-TOO
write(6,*)'** ME CPU in QXZ147 (OELMHS) = ',ME,T22-T11
if(me.eq.1) then
write(*,*)'*** --- A --- ***'
do 1122 i=n-10,n
write(*,*)'A(',i,',n) = ',a(i,n)
1122 continue
endif
C
C ****
C COMPUTE ALL EIGENVALUES AND NO EIGENVECTORS
C ****
Barrier
t00=TSECNĐ()
if(imode.eq.O) then
call HQR (MAX,N,LOW,IGH,A,ER,EL,IERR)
else
call qxz1521(max,n,low,igh,A,er,el,ierr)
endif
t11=TSECNĐ()
write(*)'** IMODE ,CPU time in QXZ152 = ',imode,t11-t00
if(me.eq.1) then
write(*)'*** Eigen value#,real ER(1), imaginary EL(1) ***'
do 7 =n-10,n
write(*)'1,er(i),ei(i)
7 continue
C...... rearrange eigenvalues according to ascending order (of real part)
call ascend(n,er,ei,wk)
endif
End Barrier
RETURN
END
C --- SUBPROGRAM QXZ146 --- FORMERLY KNOWN AS ROUTINE BALANC ---
C
-----------------------------------------------
SUBROUTINE QXZ146(NM,N,A,LOW,IGH,SCALE)
C
INTEGER I,J,K,L,M,N,JJ,NM,IGH,LOW,EXC
REAL A(N,N),SCALE(N)
LOGICAL NOCONV
C
C THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE BALANCE,
C NUM. MATH. 13, 293-304(1969) BY PARLETT AND REINSCH.
C
C THIS SUBROUTINE BALANCES A REAL MATRIX AND ISOLATES
C EIGENVALUES WHENEVER POSSIBLE.
C
C ON INPUT
NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM DIMENSION STATEMENT.

N IS THE ORDER OF THE MATRIX.

A CONTAINS THE INPUT MATRIX TO BE BALANCED.

ON OUTPUT

A CONTAINS THE BALANCED MATRIX.

LOW AND IGH ARE TWO INTEGERS SUCH THAT A(I,J) IS EQUAL TO ZERO IF
(1) I IS GREATER THAN J AND
(2) J = 1, ..., LOW-1 OR I = IGH+1, ..., N.

SCALE CONTAINS INFORMATION DETERMINING THE PERMUTATIONS AND SCALING FACTORS USED.


\[
\text{SCALE}(J) = \begin{cases} 
  P(J), & \text{for } J = 1, \ldots, \text{LOW}-1 \\
  D(J,J), & \text{for } J = \text{LOW}, \ldots, \text{IGH} \\
  P(J), & \text{for } J = \text{IGH+1}, \ldots, \text{N}.
\end{cases}
\]

THE ORDER IN WHICH THE INTERCHANGES ARE MADE IS N TO IGH+1, THEN 1 TO LOW-1.

NOTE THAT 1 IS RETURNED FOR IGH IF IGH IS ZERO FORMALLY.

THE ALGOL PROCEDURE EXC CONTAINED IN BALANCE APPEARS IN QXZ146 IN LINE. (NOTE THAT THE ALGOL ROLES OF IDENTIFIERS K,L HAVE BEEN REVERSED.)

QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW, MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY

THIS VERSION DATED AUGUST 1983.

BASED ON THE EISPACK VERSION 3 ROUTINE BALANC, AS MODIFIED BY COMPUTER SCIENCES CORPORATION, MAY 1984.

\[
\text{RADIX} = 16.00 \quad B2 = \text{RADIX} \ast \text{RADIX} \quad K = 1 \quad L = \text{N}
\]

GO TO 100

\[
\begin{array}{c}
\text{......... IN-LINE PROCEDURE FOR ROW AND COLUMN EXCHANGE ...........}
\end{array}
\]
20 SCALE(M) = J
1F (J .EQ. M) GO TO 50
C
DO 30 I = 1, L
F = A(I,J)
A(I,J) = A(I,M)
A(I,M) = F
30 CONTINUE
C
DO 40 I = K, N
F = A(J,I)
A(J,I) = A(M,I)
A(M,I) = F
40 CONTINUE
C
50 GO TO (80,130), IEXC
C ............ SEARCH FOR ROWS ISOLATING AN EIGENVALUE
C AND PUSH THEM DOWN ............
80 IF (L .EQ. I) GO TO 280
L = L - 1
C ............ FOR J=L STEP -1 UNTIL 1 DO -- ............
100 DO 120 JJ = 1, L
J = L + 1 - JJ
C
DO 110 I = 1, L
IF (I .EQ. J) GO TO 110
IF (A(J,I) .NE. 0.00) GO TO 120
CONTINUE
M = L
IEXC = 1
GO TO 20
120 CONTINUE
C
GO TO 140
C ............ SEARCH FOR COLUMNS ISOLATING AN EIGENVALUE
C AND PUSH THEM LEFT ............
130 K = K + 1
C
140 DO 170 J = K, L
C
DO 150 I = K, L
IF (I .EQ. J) GO TO 150
IF (A(I,J) .NE. 0.00) GO TO 170
CONTINUE
M = K
IEXC = 2
GO TO 20
170 CONTINUE
C ............ NOW BALANCE THE SUBMATRIX IN ROWS K TO L ............
DO 180 I = K, L
180 SCALE(I) = 1.00
C ............ ITERATIVELOOP FOR NORM REDUCTION ............
190 NOCONV = .FALSE.
C DO 270 I = K, L  
    C = 0.00  
    R = 0.00  
C
   DO 200 J = K, L  
     IF (J .EQ. I) GO TO 200  
     C = C + ABS(A(J,I))  
     R = R + ABS(A(I,J))  
200   CONTINUE  
C  ******* GUARD AGAINST ZERO C OR R DUE TO UNDERFLOW *******  
   IF (C .EQ. 0.00 .OR. R .EQ. 0.00) GO TO 270  
   G = R / RADIX  
   F = 1.00  
   S = C + R  
210   IF (C .GE. G) GO TO 220  
   F = F * RADIX  
   C = C * B2  
   GO TO 210  
220   G = R * RADIX  
230   IF (C .LT. G) GO TO 240  
   F = F / RADIX  
   C = C / B2  
   GO TO 230  
C  ****** NOW BALANCE ******  
240   IF ((C + R) / F .GE. 0.950 * S) GO TO 270  
   G = 1.00 / F  
   SCALE(1) = SCALE(1) * F  
   NOCONV = .TRUE.  
C
   DO 250 J = K, N  
250   A(I,J) = A(I,J) * G  
C
   DO 260 J = 1, L  
260   A(J,I) = A(J,I) * F  
C
270 CONTINUE  
C  IF (NOCONV) GO TO 190  
C
280   LOW = K  
   IGH = L  
RETURN  
C
************ LAST CARD OF QXZ146 ************
END  
C ---- SUBPROGRAM QXZ147 ---- FORMERLY KNOWN AS ROUTINE ELMHES ----
C
Forcesub QXZ147(NM,N,LOW,IGH,A,INT,temy) of NP ident ME
Private Real tema(1000)
End Declarations

C

IAM=1
Barrier
LA = IGH - 1
KP1 = LOW + 1
IF (LA .LT. KP1) GO TO 200
C

End Barrier
DO 180 M = KP1, LA
Barrier
End Barrier
IF(ME.EQ.IAM) THEN
    MM1 = M - 1
    X = 0.00
    I = M
C
    DO 100 J = M, IGH
            IF (ABS(A(J,MM1)) .LE. ABS(X)) GO TO 100
            X = A(J,MM1)
            I = J
    100 CONTINUE
C
    INT(M) = I
    IF (I .EQ. M) GO TO 130
C
    .......... INTERCHANGE ROWS AND COLUMNS OF A ..........
    DO 110 J = MM1, N
            Y = A(I,J)
            A(I,J) = A(M,J)
            A(M,J) = Y
    110 CONTINUE
C
    DO 120 J = 1, IGH
            Y = A(J,I)
            A(J,I) = A(J,M)
            A(J,M) = Y
    120 CONTINUE
C
    .......... END INTERCHANGE ..........
C130 IF (X.EQ.0.00) GO TO 180
C
    130 CONTINUE
END IF
Barrier
End Barrier
Barrier
IAM=IAM+1
if(IAM.GT.NP) IAM=1
End Barrier
IF(X.EQ.0.00) GO TO 1800
IF(ME.EQ.IAM) THEN
    do 1301 I=M+1,IGH
            temy(i)=a(i,mm1)/x
            if(temy(i).NE.0.00) a(i,mm1)=temy(i)
    c
    if(a(i,mm1).EQ.0.00) then
temy(i) = 0.0
else
  temy(i) = a(i, mm1) / x
a(i, mm1) = temy(i)
endif
continue

DO 160 I = MPI, IGH
  Y = A(I, MM1)
  IF (Y .EQ. 0.00) GO TO 160
  Y = Y / X
  A(I, MM1) = Y
ENDIF

do 1399 j = 1, igh
  tema(j) = 0.0
  jend = ((igh - m) / 8) * 8
  Barrier
  iam = iam + 1
  if (IAM .GT. NP) IAM = 1
  End Barrier
 Barrier
 End Barrier

do 1400 jj = m + 1, igh
  Presched DO 1400 jj = m + 1, igh

CDIR$ IVDEP

do 1401 j = 1, m - 1
  a(j, m) = a(j, m) + temy(jj) * a(j, j + 1) + temy(jj + 1) * a(j, j + 1)
  tema(j) = tema(j) + temy(jj) * a(j, j + 1) + temy(jj + 1) * a(j, j + 1)
  + temy(jj + 2) * a(j, j + 2) + temy(jj + 3) * a(j, j + 3)
  + temy(jj + 4) * a(j, j + 4) + temy(jj + 5) * a(j, j + 5)
  + temy(jj + 6) * a(j, j + 6) + temy(jj + 7) * a(j, j + 7)
  continue
1400
End Presched DO
Barrier
End Barrier

Presched DO 1402 jj = jend + 1 + m, igh
CDIR$ IVDEP

do 1403 j = 1, m - 1
  tema(j) = tema(j) + temy(jj) * a(j, j)
1402 End Presched DO
Barrier
End Barrier
Critical ilock
  do 14001 j = 1, m - 1
14001 a(J, M) = a(J, M) + tema(j)
End Critical
Barrier
End Barrier
Presched DO 1411 jj = m + 1, n
CDIR$ IVDEP

do 1412 ii = m + 1, igh
1412 a(ii, jj) = a(ii, jj) - a(m, jj) * temy(ii)
1411 End Presched DO
Barrier
End Barrier
IF(ME.EQ.IAM) THEN
do 1407 kk=m+1,igh
xmuy=temy(kk)
xmul=xmuy*a(m,kk)
c write(*,*) (kk,m)
a(kk,m)=a(kk,m)-temy(kk)*a(m,m)
a(kk,m)=a(kk,m)-xmul*a(m,m)
CDIR$ IVDEP

1608 do 1608 ik=m,kk
a(ik,m)=a(ik,m)+xmul*a(ik,kk)
CDIR$ IVDEP

1609 do 1609 ik=kk+1,igh
a(ik,m)=a(ik,m)+xmul*a(ik,kk)+xmuy*temy(ik)
c write(*,*) (kk,m),temy(kk),a(m,m)
1407 continue
ENDIF
1800 continue
C
DO 140 J=M,N
C 140 A(I,J) = A(I,J) - Y * A(M,J)
C
C 150 A(J,M) = A(J,M) + Y * A(J,I)
C
C 160 continue
C
C do 1500 II=I,N
C write(*,1501) (a(ii,jj),jj=m+1,n)
C500 continue
C501 format(Ix,10(e9.3,1x))
Barrier
IAM=IAM+I
IF(IAM.GT.NP) IAM=1
End Barrier
180 CONTINUE
C
200 RETURN
C
****** LAST CARD OF QXZ147 ******
C** THIS PROGRAM VALID ON FTN4 AND FTN5 **
END
SUBROUTINE BALANC(NM,N,A,LOW,IGH,SCALE)
C
INTEGER I,J,K,L,M,N,JJ,NM,IGH,LOW,IEXC
REAL A(N,N),SCALE(N)
LOGICAL NOCONV
C
THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE BALANCE,
NUM. MATH. 13, 293-304 (1969) BY PARLETT AND REINSCH.
C
THIS SUBROUTINE BALANCES A REAL MATRIX AND ISOLATES
EIGENVALUES WHENEVER POSSIBLE.
ON INPUT

NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
DIMENSION STATEMENT.

N IS THE ORDER OF THE MATRIX.

A CONTAINS THE INPUT MATRIX TO BE BALANCED.

ON OUTPUT

A CONTAINS THE BALANCED MATRIX.

LOW AND IGH ARE TWO INTEGERS SUCH THAT A(I,J)
IS EQUAL TO ZERO IF
(1) I IS GREATER THAN J AND
(2) J=1,...,LOW-1 OR I=IGH+1,...,N.

SCALE CONTAINS INFORMATION DETERMINING THE
PERMUTATIONS AND SCALING FACTORS USED.

SUPPOSE THAT THE PRINCIPAL SUBMATRIX IN ROWS LOW THROUGH IGH
HAS BEEN BALANCED, THAT P(J) DENOTES THE INDEX INTERCHANGED
WITH J DURING THE PERMUTATION STEP, AND THAT THE ELEMENTS
OF THE DIAGONAL MATRIX USED ARE DENOTED BY D(I,J). THEN
SCALE(J) = P(J), FOR J = 1,...,LOW-1
= D(J,J), J = LOW,...,IGH
= P(J) J = IGH+1,...,N.

THE ORDER IN WHICH THE INTERCHANGES ARE MADE IS N TO IGH+1,
THEN 1 TO LOW-1.

NOTE THAT 1 IS RETURNED FOR IGH IF IGH IS ZERO FORMALLY.

THE ALGOL PROCEDURE EXC CONTAINED IN BALANCE APPEARS IN
BALANCE IN LINE. (NOTE THAT THE ALGOL ROLES OF IDENTIFIERS
K,L HAVE BEEN REVERSED.)

QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARROW,
MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY

THIS VERSION DATED AUGUST 1983.

---

RADIX = 16.0EO

B2 = RADIX * RADIX
K = 1
L = N
GO TO 100

......... IN-LINE PROCEDURE FOR ROW AND
COLUMN EXCHANGE ..........

20 SCALE(M) = J
IF (J .EQ. M) GO TO 50
C
DO 30 I = 1, L
   F = A(I,J)
   A(I,J) = A(I,M)
   A(I,M) = F
30 CONTINUE
C
DO 40 I = K, N
   F = A(J,I)
   A(J,I) = A(M,I)
   A(M,I) = F
40 CONTINUE
C
50 GO TO (80,130), IEXC
C .......... SEARCH FOR ROWS ISOLATING AN EIGENVALUE
C AND PUSH THEM DOWN ...........
80 IF (L .EQ. 1) GO TO 280
   L = L - 1
C .......... FOR J=L STEP -1 UNTIL 1 DO -- ........
100 DO 120 JJ = 1, L
       J = L + 1 - JJ
   DO 110 I = J, L
       IF (I .EQ. J) GO TO 110
       IF (A(J,I) .NE. 0.0E0) GO TO 20
   110 CONTINUE
C
   M = L
   IEXC = 1
   GO TO 20
120 CONTINUE
C
GO TO 140
C .......... SEARCH FOR COLUMNS ISOLATING AN EIGENVALUE
C AND PUSH THEM LEFT ...........
130 K = K + 1
C
140 DO 170 J = K, L
   DO 150 I = J, L
       IF (I .EQ. J) GO TO 150
       IF (A(I,J) .NE. 0.0E0) GO TO 170
   150 CONTINUE
C
   M = K
   IEXC = 2
   GO TO 20
170 CONTINUE
C .......... NOW BALANCE THE SUBMATRIX IN ROWS K TO L .........
DO 180 I = K, L
180 SCALE(I) = 1.0E0
C .......... ITERATIVE LOOP FOR NORM REDUCTION ...........
190 NOCONV = .FALSE.
C
DO 270 I = K, L
C = 0.0EO
R = 0.0EO

DO 200 J = K, L
  IF (J .EQ. I) GO TO 200
  C = C + ABS(A(J,I))
  R = R + ABS(A(I,J))
200 CONTINUE

C .......... GUARD AGAINST ZERO C OR R DUE TO UNDERFLOW ..........
  IF (C .EQ. 0.0EO .OR. R .EQ. 0.0EO) GO TO 270
  G = R / RADIX
  F = 1.0EO
  S = C + R
210 IF (C .GE. G) GO TO 220
  F = F * RADIX
  C = C * B2
  GO TO 210
220 G = R * RADIX
230 IF (C .LT. G) GO TO 240
  F = F / RADIX
  C = C / B2
  GO TO 230
C .......... NOW BALANCE ..........
240 IF ((C + R) / F .GE. 0.95EO * S) GO TO 270
  G = 1.0EO / F
  SCALE(I) = SCALE(I) * F
  NOCONV = .TRUE.
C
DO 250 J = K, N
250 A(I,J) = A(I,J) * G
C
DO 260 J = 1, L
260 A(J,I) = A(J,I) * F
C
270 CONTINUE
C
IF (NOCONV) GO TO 190
C
280 LOW = K
IGH = L
RETURN
END

SUBROUTINE HQR(NM,N,LOW,IGH,H,WR,WI,IERR)

INTEGER I,J,K,L,M,N,EN,LL,MM,NA,NM,IGH,ITN,ITS,LOW,MP2,ENM2,IERR
REAL H(N,N),WR(N),WI(N)
REAL P,Q,R,S,T,W,X,Y,ZZ,NORM,TST1,TST2
LOGICAL NOTLAS
C
THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE HQR,
NUM. MATH. 14, 219-231(1970) BY MARTIN, PETERS, AND WILKINSON.
C
THIS SUBROUTINE FINDS THE EIGENVALUES OF A REAL
UPPER HESSENEBERG MATRIX BY THE QR METHOD.
ON INPUT

NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM DIMENSION STATEMENT.

N IS THE ORDER OF THE MATRIX.

LOW AND IGH ARE INTEGERS DETERMINED BY THE BALANCING SUBROUTINE BALANC. IF BALANC HAS NOT BEEN USED, SET LOW=1, IGH=N.

H CONTAINS THE UPPER HESSENBERG MATRIX. INFORMATION ABOUT THE TRANSFORMATIONS USED IN THE REDUCTION TO HESSENBERG FORM BY ELMHES OR ORTHES, IF PERFORMED, IS STORED IN THE REMAINING TRIANGLE UNDER THE HESSENBERG MATRIX.

ON OUTPUT

H HAS BEEN DESTROYED. THEREFORE, IT MUST BE SAVED BEFORE CALLING HQR IF SUBSEQUENT CALCULATION AND BACK TRANSFORMATION OF EIGENVECTORS IS TO BE PERFORMED.

WR AND WI CONTAIN THE REAL AND IMAGINARY PARTS, RESPECTIVELY, OF THE EIGENVALUES. THE EIGENVALUES ARE UNORDERED EXCEPT THAT COMPLEX CONJUGATE PAIRS OF VALUES APPEAR CONSECUTIVELY WITH THE EIGENVALUE HAVING THE POSITIVE IMAGINARY PART FIRST. IF AN ERROR EXIT IS MADE, THE EIGENVALUES SHOULD BE CORRECT FOR INDICES IERR+1,...,N.

IERR IS SET TO
ZERO FOR NORMAL RETURN,
J IF THE LIMIT OF 30*N ITERATIONS IS EXHAUSTED WHILE THE J-TH EIGENVALUE IS BEING SOUGHT.

QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW, MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY

THIS VERSION DATED AUGUST 1983.

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

IERR = 0
NORM = 0.000
K = 1

............... STORE ROOTS ISOLATED BY BALANC AND COMPUTE MATRIX NORM .........

DO 50 I = 1, N

DO 40 J = K, N

40 NORM = NORM + ABS(H(I,J))

K = 1
IF (I .GE. LOW .AND. I .LE. IGH) GO TO 50
WR(I) = H(I,1)
WI(I) = 0.0E0

50 CONTINUE
C
EN = IGH
T = 0.0E0
ITN = 30*AN
C
60 IF (EN .LT. LOW) GO TO 1001
ITS = 0
NA = EN - 1
ENM2 = NA - 1
C
DO 70 LL = LOW, EN
L = EN + LOW - LL
IF (L .EQ. LOW) GO TO 100
S = ABS(H(L-1,L-1)) + ABS(H(L,L))
IF (S .EQ. 0.0E0) S = NORM
TST1 = S
TST2 = TST1 + ABS(H(L,L-1))
IF (TST2 .EQ. TST1) GO TO 100
70 CONTINUE
C
DO 80 LL = LOW, EN
L = EN + LOW - LL
IF (L .EQ. LOW) GO TO 100
S = ABS(H(L-1,L-1)) + ABS(H(L,L))
IF (S .EQ. 0.0E0) S = NORM
TST1 = S
TST2 = TST1 + ABS(H(L,L-1))
IF (TST2 .EQ. TST1) GO TO 100
80 CONTINUE
C
100 X = H(EN,EN)
IF (L .EQ. EN) GO TO 270
Y = H(NA,NA)
W = H(EN,NA) * H(NA,EN)
IF (L .EQ. NA) GO TO 280
IF (ITN .EQ. 0) GO TO 1000
IF (ITS .NE. 10 .AND. ITS .NE. 20) GO TO 130
T = T + X
C
DO 120 I = LOW, EN
120 H(I,1) = H(I,1) - X
C
S = ABS(H(EN,NA)) + ABS(H(NA,ENM2))
X = 0.75E0 * S
Y = X
W = -0.4375E0 * S * S
130 ITS = ITS + 1
ITN = ITN - 1
C
DO 140 MM = L, ENM2
M = ENM2 + L - MM
ZZ = H(M,M)
R = X - ZZ
S = Y - ZZ
P = (R * S - W) / H(M+1,M) + H(M,M+1)
Q = H(M+1,M+1) - ZZ - R - S
R = H(M+2,M+1)

24
\[ S = \text{ABS}(P) + \text{ABS}(Q) + \text{ABS}(R) \]
\[ P = \frac{P}{S} \]
\[ Q = \frac{Q}{S} \]
\[ R = \frac{R}{S} \]
\[ \text{IF} (M \ .\ EQ. \ L) \text{ GO TO 150} \]
\[ \text{TST1} = \text{ABS}(P) \times (\text{ABS}(H(M+1,M-1)) + \text{ABS}(ZZ) + \text{ABS}(H(M+1,M-1))) \]
\[ \text{TST2} = \text{TST1} + \text{ABS}(H(M,M-1)) \times (\text{ABS}(Q) + \text{ABS}(R)) \]
\[ \text{IF} (\text{TST2} .\ EQ. \ TST1) \text{ GO TO 150} \]

140 CONTINUE

150 \( MP2 = M + 2 \)

DO 160 I = MP2, EN
\[ H(I,I-2) = 0.00 \]
\[ \text{IF} (I .\ EQ. \ MP2) \text{ GO TO 160} \]
\[ H(I,I-3) = 0.00 \]
160 CONTINUE

C .......... DOUBLE QR STEP INVOLVING ROWS L TO EN AND C COLUMNS M TO EN ...........

DO 260 K = M, NA
\[ \text{NOTLAS} = K .\ NE. \ NA \]
\[ \text{IF} (K .\ EQ. \ M) \text{ GO TO 170} \]
\[ P = H(K,K-1) \]
\[ Q = H(K+1,K-1) \]
\[ R = 0.00 \]
\[ \text{IF} (\text{NOTLAS}) \ R = H(K+2,K-1) \]
\[ X = \text{ABS}(P) + \text{ABS}(Q) + \text{ABS}(R) \]
\[ \text{IF} (X .\ EQ. \ 0.00) \text{ GO TO 260} \]
\[ P = \frac{P}{X} \]
\[ Q = \frac{Q}{X} \]
\[ R = \frac{R}{X} \]
170 \[ S = \text{SIGN SQRT}(P^2+Q^2+R^2), P) \]
\[ \text{IF} (K .\ EQ. \ M) \text{ GO TO 180} \]
\[ H(K,K-1) = -S \times X \]
GO TO 190
180 \[ \text{IF} (L .\ NE. \ M) \ H(K,K-1) = -H(K,K-1) \]
190 \[ P = P + S \]
\[ X = \frac{P}{S} \]
\[ Y = Q / S \]
\[ ZZ = R / S \]
\[ Q = Q / P \]
\[ R = R / P \]
\[ \text{IF} (\text{NOTLAS}) \text{ GO TO 225} \]

C .......... ROW MODIFICATION ...........

DO 200 J = K, N
\[ P = H(K,J) + Q \times H(K+1,J) \]
\[ H(K,J) = H(K,J) - P \times X \]
\[ H(K+1,J) = H(K+1,J) - P \times Y \]
200 CONTINUE

C
\[ J = \text{MINO}(EN,K+3) \]

C .......... COLUMN MODIFICATION ...........

DO 210 I = 1, J
\[ P = X \times H(I,K) + Y \times H(I,K+1) \]
\[ H(I,K) = H(I,K) - P \]
H(I,K+1) = H(I,K+1) - P * Q

CONTINUE
GO TO 255

CONTINUE

C *********** ROW MODIFICATION ***********
DO 230 J = K, N
P = H(K,J) + Q * H(K+1,J) + R * H(K+2,J)
H(K,J) = H(K,J) - P * X
H(K+1,J) = H(K+1,J) - P * Y
H(K+2,J) = H(K+2,J) - P * ZZ
CONTINUE

C
J = MINO(EN,K+3)

C *********** COLUMN MODIFICATION ***********
DO 240 I = 1, J
P = X * H(I,K) + Y * H(I,K+1) + ZZ * H(I,K+2)
H(I,K) = H(I,K) - P
H(I,K+I) = H(I,K+I) - P * Q
H(I,K+2) = H(I,K+2) - P * R
CONTINUE
CONTINUE
GO TO 70

C *********** ONE ROOT FOUND ***********
270 WR(EN) = X + T
WI(EN) = 0.0EO
EN = NA
GO TO 60

C *********** TWO ROOTS FOUND ***********
280 P = (Y - X) / 2.0EO
Q = P * P + W
ZZ = SQRT(ABS(Q))
X = X + T
IF (Q .LT. 0.0EO) GO TO 320

C *********** REAL PAIR ***********
ZZ = P + SIGN(ZZ,P)
WR(NA) = X + ZZ
WR(EN) = WR(NA)
IF (ZZ .NE. 0.0EO) WR(EN) = X - W / ZZ
WI(NA) = 0.0EO
WI(EN) = 0.0EO
GO TO 330

C *********** COMPLEX PAIR ***********
320 WR(NA) = X + P
WR(EN) = X + P
WI(NA) = ZZ
WI(EN) = -ZZ
330 EN = ENM2
GO TO 60

C *********** SET ERROR -- ALL EIGENVALUES HAVE NOT CONVERGED AFTER 30AN ITERATIONS ***********
1000 IERR = EN
1001 RETURN
Real A(N,N), RINDEX(IGH)
Integer IGH, LOW, N, NM
Real X, Y

Shared Integer KPI, LA, MM1, MPI

End Declarations

This subroutine is a translation of the Algol procedure ELMHES, Num. Math. 12, 349-368 (1968) by Martin and Wilkinson.

Given a real general matrix, this subroutine reduces a submatrix situated in rows and columns LOW through IGH to upper Hessenberg form by stabilized elementary similarity transformations.

On input

NM must be set to the row dimension of two-dimensional array parameters as declared in the calling program dimension statement.

N is the order of the matrix.

LOW and IGH are integers determined by the balancing subroutine BALANC. If BALANC has not been used, set LOW=1, IGH=N.

A contains the input matrix.

On output

A contains the Hessenberg matrix. The multipliers which were used in the reduction are stored in the remaining triangle under the Hessenberg matrix.

RINDEX contains information on the rows and columns interchanged in the reduction.
Only elements LOW through IGH are used.

Questions and comments should be directed to Burton S. Garbow, Mathematics and Computer Science Div., Argonne National Laboratory

This version dated August 1983.

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

LA = IGH - 1
KPI = LOW + 1
If (LA .LT. KPI) RETURN
Barrier
C
DO 180 M = KPI, LA
   MM1 = M - 1
   X = O.000
   I = M
C
   DO 100 J = M, IGH
      IF (ABS(A(J,MM1)) .LE. ABS(X)) GO TO 100
      X = A(J,MM1)
      I = J
100 CONTINUE
C
   RINDEX(M) = REAL(I)
   IF (I .EQ. M) GO TO 130
C
   .......... INTERCHANGE ROWS AND COLUMNS OF A ..........
   DO 110 J = MM1, N
      Y = A(I,J)
      A(I,J) = A(M,J)
      A(M,J) = Y
110 CONTINUE
C
   DO 120 J = 1, IGH
      Y = A(J,I)
      A(J,I) = A(J,M)
      A(J,M) = Y
120 CONTINUE
C
   .......... END INTERCHANGE ..........
   130 IF (X .EQ. 0.000) GO TO 180
      MPI = M + 1
C
   DO 160 I = MPI, IGH
      Y = A(I,MM1)
      IF (Y .EQ. 0.000) GO TO 160
      Y = Y / X
      A(I,MM1) = Y
C
   DO 140 J = M, N
      A(I,J) = A(I,J) - Y * A(M,J)
140 CONTINUE
C
   DO 150 J = 1, IGH
      A(J,M) = A(J,M) + Y * A(J,I)
150 CONTINUE
C
   160 CONTINUE
C
   180 CONTINUE
C
End Barrier
RETURN
END
SUBROUTINE QXZ1521(NM,N,LOW,IGH,H,WR,WI,IERR)
C
INTEGER I,J,K,L,M,N,EN,LL,MM,NA,NM,IGH,ITN,ITS,LOW,MP2,ENM2,IERR
REAL H(N,N),WR(N),WI(N)
REAL P,Q,R,S,T,W,X,Y,ZZ,NORM,TST1,TST2

28
LOGICAL NOTLAS
C
IERR = 0
NORM = 0.00
K = 1
C
.......... STORE ROOTS ISOLATED BY QXZ146
C
AND COMPUTE MATRIX NORM ............
C
DO 50 I = 1, N
C
DO 40 J = K, N
C
NORM = NORM + ABS(H(I,J))
C
K = I
IF (I .GE. LOW .AND. I .LE. IGH) GO TO 50
WR(I) = H(I,I)
WI(I) = 0.00
C
50 CONTINUE
C
EN = IGH
T = 0.00
ITN = 30*N
C
............ SEARCH FOR NEXT EIGENVALUES ............
C
60 IF (EN .LT. LOW) GO TO 1001
ITS = 0
NA = EN - 1
ENM2 = NA - 1
C
............ LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C
FOR L=EN STEP -1 UNTIL LOW DO -- .........
70 DO 80 LL = LOW, EN
C
L = EN + LOW - LL
IF (L .EQ. LOW) GO TO 100
S = ABS(H(L-1,L-1)) + ABS(H(L,L))
IF (S .EQ. 0.00) S = NORM
TST1 = S
TST2 = TST1 + ABS(H(L,L-1))
IF (TST2 .EQ. TST1) GO TO 100
C
80 CONTINUE
C
............ FORM SHIFT ............
100 X = H(EN,EN)
IF (L .EQ. EN) GO TO 270
Y = H(NA,NA)
W = H(EN,NA) * H(NA,EN)
IF (L .EQ. NA) GO TO 280
IF (ITN .EQ. O) GO TO 1000
IF (ITS .NE. 10 .AND. ITS .NE. 20) GO TO 130
C
............ FORM EXCEPTIONAL SHIFT ............
WRITE(*,*)'** EN, T X =',EN,T,X
T = T + X
C
DO 120 I = LOW, EN
120 H(I,I) = H(I,I) - X
C
S = ABS(H(EN,NA)) + ABS(H(NA,ENM2))
X = 0.750 * S
Y = X
C
W = -0.43750 * S * S
130 ITS = ITS + 1
1TN = ITN - 1
C ............ LOOK FOR TWO CONSECUTIVE SMALL
C SUB-DIAGONAL ELEMENTS.
C FOR M=EN-2 STEP -1 UNTIL L DO -- ........
DO 140 MM = L, ENM2
     M = ENM2 + L - MM
     ZZ = H(M,M)
     R = X - ZZ
     S = Y - ZZ
     P = (R * S - W) / H(M+1,M) + H(M,M+1)
     Q = H(M+1,M+1) - ZZ - R - S
     R = H(M+2,M+1)
     S = ABS(P) + ABS(Q) + ABS(R)
     P = P / S
     Q = Q / S
     R = R / S
     IF (M .EQ. L) GO TO 150
     TST1 = ABS(P) + ABS(H(M-1,M-1)) + ABS(ZZ) + ABS(H(M+1,M+1))
     TST2 = TST1 + ABS(H(M,M-1)) + ABS(H(M+1,M+1)) + ABS(H(M,M+1))
     IF (TST2 .EQ. TST1) GO TO 150
140 CONTINUE
C
150 MP2 = M + 2
C
DO 160 I = MP2, EN
     H(I,I-2) = 0.00
     IF (I .EQ. MP2) GO TO 160
     H(I,I-3) = 0.00
160 CONTINUE
C ............ DOUBLE QR STEP INVOLVING ROWS L TO EN AND
C COLUMNS M TO EN ..........
DO 260 K = M, NA
     NOTLAS = K .NE. NA
     IF (K .EQ. M) GO TO 170
     P = H(K,K-1)
     Q = H(K+1,K-1)
     R = 0.00
     IF (NOTLAS) R = H(K+2,K-1)
     X = ABS(P) + ABS(Q) + ABS(R)
     IF (X .EQ. 0.00) GO TO 260
     P = P / X
     Q = Q / X
     R = R / X
     S = SIGN(SQRT(P*P+Q*Q+R*R),P)
     IF (K .EQ. M) GO TO 180
     H(K,K-1) = -S * X
     GO TO 190
170 IF (L .NE. M) H(K,K-1) = -H(K,K-1)
180 P = P + S
     X = P / S
     Y = Q / S
     ZZ = R / S
     Q = Q / P
R = R / P
IF (NOTLAS) GO TO 225
C
DO 200 J = K, N
  P = H(K,J) + Q * H(K+1, J)
  H(K,J) = H(K,J) - P * X
  H(K+1,J) = H(K+1,J) - P * Y
200 CONTINUE
C
J = MINO(EN,K+3)
C
DO 210 I = 1, J
  P = X * H(I,K) + Y * H(I,K+1)
  H(I,K) = H(I,K) - P
  H(I,K+1) = H(I,K+1) - P * Q
210 CONTINUE
GO TO 255
C
DO 230 J = K, N
  P = H(K,J) + Q * H(K+1,J) + R * H(K+2,J)
  H(K,J) = H(K,J) - P * X
  H(K+1,J) = H(K+1,J) - P * Y
  H(K+2,J) = H(K+2,J) - P * ZZ
230 CONTINUE
C
J = MINO(EN,K+3)
C
DO 240 I = 1, J
  P = X * H(I,K) + Y * H(I,K+1) + ZZ * H(I,K+2)
  H(I,K) = H(I,K) - P
  H(I,K+1) = H(I,K+1) - P * Q
  H(I,K+2) = H(I,K+2) - P * R
240 CONTINUE
C
WRITE(*,*) 'NOTLAS,K,H(K,K)=',NOTLAS,K,H(K,K)
260 CONTINUE
C
GO TO 70
C
DO 270 EN = X + T
  W(EN) = 0.00
  EN = NA
  GO TO 60
C
DO 280 P = (Y - X) / 2.00
  Q = P * P + W
  ZZ = SQRT(ABS(Q))
  X = X + T
C
WRITE(*,*) 'NOTLAS,K,H(K,K)=',NOTLAS,K,H(K,K)
290 CONTINUE
C
IF (Q.LT.0.00) go to 320
C
ZZ = P + SIGN(ZZ, P)
WR(NA) = X + ZZ
WR(EN) = WR(NA)
IF (ZZ .NE. 0.00) WR(EN) = X - W / ZZ
WI(NA) = 0.00
WI(EN) = 0.00
GO TO 330

C ........... COMPLEX PAIR ...........
320 WR(NA) = X + P
WR(EN) = X + P
WI(NA) = ZZ
WI(EN) = -ZZ
330 EN = EMN2
GO TO 60

C ........... SET ERROR -- ALL EIGENVALUES HAVE NOT
C CONVERGED AFTER 30*N ITERATIONS ...........
1000 IERR = EN
1001 RETURN
END

subroutine ascend(n,er,ei,wk)
C implicit real*8(a-h,o-z)
real er(1),ei(1),wk(n,1)
C........
doi i=1,n
small=999999999.
do 2 j=1,n
if( er(j).lt.small ) then
small=er(j)
locate=j
wk(i,1)=er(j)
w(i,2)=ei(j)
endif
2 continue
er(locate)=999999999.
1 continue
c........
do 21 i=1,n
er(i)=wk(i,1)
e(i)=wk(i,2)
c........
write(6,*) 'real & imaginary evals in ascending order'
do 11 i=n-10,n
write(6,*) i,er(i),ei(i)
11 continue
return
end

32
The popular QR algorithm for solving all eigenvalues of an unsymmetric matrix is reviewed. Among the basic components in the QR algorithm, it has been concluded from this study, that the reduction of an unsymmetric matrix to a Hessenberg form (before applying the QR algorithm itself) can be done effectively by exploiting the vector speed and multiple processors offered by modern high-performance computers.

Numerical examples of several test cases have indicated that the proposed parallel-vector algorithm for converting a given unsymmetric matrix to a Hessenberg form offers computational advantages over the existing algorithm. The time saving obtained by the proposed methods is increased as the problem size increased.

13. ABSTRACT (Maximum 200 words)
ERRATA

NASA Contractor Report 191417
Parallel-Vector Unsymmetric Eigen-Solver
on High Performance Computers
Duc T. Nguyen and Qin Jiangning
February 1993

The word "Unsymmetric" in the title on the cover should be "Unsymmetric." Task 122 has been added to the contract number on the cover.

A corrected cover is attached.