158526
P-34

NASA Contractor Report 191417

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

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-0001

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 eigen-solutions.

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.

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

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  I=1,N
DO 2  J=1,N
B(I) = B(I)+A(I,J)*x(J)
2 Continue
1 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  J = 1,N
DO 2  I = 1,N
B(I) = B(I) + A(I,J) * x(J)
2 Continue
1 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 = Old vector B + Constant * 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:

```
NUNROL = 2
DO 1  I = 1,N, NUNROL
DO 2  J = 1,N
B(I) = B(I) + A(I,J) * x(J)
B(I+1) = B(I+1) + A(I+1,J) * x(J)
2 Continue
1 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

```
NUNROL = 2
DO 1  J = 1,N, NUNROL
DO 2  I = 1,N
B(I) = B(I) + A(I,J) * x(J)
               + A(I,j+1) * x(J+1)
2 Continue
1 Continue
```

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
Parallel DO 1  J = 1,N, NUNROL
DO 2  I = 1,N
B(I) = B(I) + A(I,J) * x(J)
               + A(I,J+1) * 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 * A(J,I)

or

A new vector A(J, -) = old vector A(J, -) + (a constant) * 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 $[A]_{N x N}$ 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

Size N	"Original" CSI version (HR = Hessenberg Reduction Time; QR Time)	"New" version
100 x 100	0.41 sec 0.97 sec	0.39 sec 0.97 sec
200 x 200	2.210 sec 5.195 sec	2.22 sec 5.19 sec
400 x 400	16.9 33.9	14.00 33.93
600 x 600	55.48 94.20	51.0 94.2
800 x 800	161.6 *N/A*	119 *N/A*

Table 2: Parallel-Vector Performance on Cray-YMP (Reynolds) Using tsecnd ().

Size N	"Original" CSI version (HR = Hessenberg Reduction Time / QR Time) 1 Cray-YMP Processor	"New" version		
		1 Cray-YMP Processor	2 Cray-YMP Processors	3 Cray-YMP Processors
100 x 100	(0.02 sec / 0.07 sec)	(0.02 sec / 0.07 sec)	(0.03 sec / 0.07 sec)	(0.03 sec / 0.07 sec)
200 x 200	(0.12 / 0.42)	(0.11 / 0.42)	(0.08 / 0.41)	(0.07 / 0.41)
400 x 400	(1.19 / 3.15)	(0.72 / 3.19)	(0.41 / 3.18)	(0.27 / 3.19)
600 x 600	(2.90 / 7.12)	(2.28 / 7.12)	(1.22 / 7.08)	(0.70 / 7.12)
800 x 800	(14.34 / 33.25)	(5.17 / 33.31)	(2.69 / 33.27)	(1.45 / 33.43)

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 algorthm 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:

1. W. K. Belvin, P.G. Maghami, and D.T. Nguyen, "Efficient Use of High-Performance Computers for Integrated Controls and Structures Design," Proceedings of the Symposium on High-Performance Computing for Flight Vehicles, Washington, D.C., December 7-9, 1992.

2. K.J. Bathe, Finite Element Procedures In Engineering Analysis, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

3. G.H. Golub, and C.F.V. Loan, Matrix Computations, Baltimore: Johns Hopkins University Press (1983).

4. P.G. Maghami, S.M. Joshi, K.B. Elliot, and J.E. Walz, "Integrated Design of the CSI Evolutionary Structure: A verification of the design methodology," Proceedings of the Fifth NASA/DOD Controls-Structures Interaction Conference, Lake Tahoe, NV, March 1992.

5. W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, Numerical Recipes: The Art of Scientific Computing (Fortran Version), Cambridge University Press (1990).

APPENDIX

Parallel-Vector Hessenberg Reduction And Sequential QR Algorithm [5]

```
C-----   PARALLEL/VECTOR UNSYMMETRIC EIGENSOLVER by Qin & Nguyen,May 1992 **
c......This is a working version of "unsymmetrical" eigen-solver
c......on the sun386 work station. On the Cray-YMP (Reynold or Sabre),
c......this "exact" same version should offer good vector & parallel
c......speed (only for subroutine to perform Hessenberg reduction).
c......For SMALL problems, the improvements due to parallel-vector
c......Hessenberg is NOT MUCH. However, for LARGE problems, since the
c......Hessenberg reduction timing becomes more important (as compared to
c......the TOTAL eigen-solution time), the total time saving for the entire
c......eigen-solution process is also very significant.
c......Since this version was developed specifically for CSI applications
c......(according to Peiman's specifications/requirements),ALL EIGENVALUES
c......(and NONE of the corresponding EIGENVECTORS) of an N by N squared
c......unsymmatrical matrix are found.
c......"ARTIFICIAL" datas of varous sizes (N = 2 ----> 800) with ALL REAL
c......and MIXED REAL & COMPLEX eigenvalues have been verified (by comparing
c......the results obtained by the original unsym. eigen-sol. taken from
c......ORACLE and the modified version from the ODU team, and also by HAND
c......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(0=old version),nguyen(1=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(//,' INPUT PARAMETERS:',/,
     1 'N    = ',I5,'  -  Size  of System'//,
     1 'IMODE= ',I5,'  -  = 0 is old sequential'//)
       End Barrier
      Forcecall RESV(N,N,A,ER,EI,WK,IERR,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
      betam1=beta-1.0
      if((a+betam1)-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
70      geteps=eps
      return
      end
C**************************************************************
      Forcesub RESV(MAX,N,A,ER,EI,WK,IERR,EIG,IMODE,nguyen) of NP
     $ ident ME
      INTEGER MAX,N,IERR,IMODE
      Shared Integer LOW,IGH,NACC
C     F2.4
C ****
C   FUNCTION               - COMPUTES ALL THE EIGENVALUES AND SELECTED
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          - EIGENVALUES ARE STORED IN ASCENDING MAGNITUDE
C                              WITH COMPLEX CONJUGATES STORED WITH POSITIV
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 COMPLE
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,...
```

```
C                                N AND NO EIGENVECTORS ARE COMPUTED.
C   REQUIRED ROUTINES     - QXZ146,QXZ147,QXZ152
C ****
      REAL A(N,N),ER(N),EI(N),WK(N,2),EIG(1)
       End Declarations
C      DIMENSION A(MAX,N),ER(N),EI(N),V(MAX,*),WK(N,*)
c     LOGICAL LTESTV
c     EQUIVALENCE (TESTV,LTESTV)
CQIN  DATA TRUE,FALSE / '77777777777777777777'O, '00000000000000000000'O
CQIN     +/
c     DATA TRUE,FALSE / 7777777777777777777.0,0.0000000000000000000 /
C ****
C     PRELIMINARY REDUCTION
C ****
       Barrier
       DO 2 J=1,N
       DO 1 I=1,N
       if(i.lt.j) then
       a(i,j)=1.373737373737/(float(i+j))
       else
       A(i,j)=0.973197319731/(float(i+j+j/2))
       endif
1      continue
2      continue
       do 3 i=1,n
3      a(i,i)=float(i*i)
c......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)=-1.373737373737*10.0/(float(i+j))
       else
       A(i,j)=0.973197319731*10.0/(float(i+j+j/2))
       endif
19     continue
29     continue
       do 39 i=1,n
39     a(i,i)=float(i)
c-------------------------------------------
       a(1,1)=2.
       a(1,2)=-6.
       a(2,1)=8.
       a(2,2)=1.
       endif
c------------------------------------------------------------------
C *****  SAVE A FOR NORM CHECK  *****
        low=1
        igh=n
      TIME0=0.0
      End Barrier
      t00=TSECND()
c     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=TSECND()
      TIMEO=TIMEO+T22-T00
c     write(6,*)'** ME  CPU in QXZ146  =  ',ME, T11-T00
      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     COMPUTE ALL EIGENVALUES AND NO EIGENVECTORS
C ****
      Barrier
      t00=TSECND()
      if(imode.eq.0) then
      call HQR (MAX,N,LOW,IGH,A,ER,EI,IERR)
       else
      call qxz1521(max,n,low,igh,A,er,ei,ierr)
       endif
      t11=TSECND()
      write(*,*)' ** IMODE ,CPU time in QXZ152 = ',imode,t11-t00
       if(me.eq.1) then
      write(*,*)' *** Eigen value#,real ER(I), imaginary EI(I) ***'
      do 7 I=n-10,n
      write(*,*) I,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
C     ----------------------------------------------------------
      SUBROUTINE QXZ146(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)
      REAL C,F,G,R,S,B2,RADIX
      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     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 315-326(1971).
C
C     THIS SUBROUTINE BALANCES A REAL MATRIX AND ISOLATES
C     EIGENVALUES WHENEVER POSSIBLE.
C
C     ON INPUT
```

```
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRIX.
C
C        A CONTAINS THE INPUT MATRIX TO BE BALANCED.
C
C     ON OUTPUT
C
C        A CONTAINS THE BALANCED MATRIX.
C
C        LOW AND IGH ARE TWO INTEGERS SUCH THAT A(I,J)
C          IS EQUAL TO ZERO IF
C           (1) I IS GREATER THAN J AND
C           (2) J=1,...,LOW-1 OR I=IGH+1,...,N.
C
C        SCALE CONTAINS INFORMATION DETERMINING THE
C           PERMUTATIONS AND SCALING FACTORS USED.
C
C     SUPPOSE THAT THE PRINCIPAL SUBMATRIX IN ROWS LOW THROUGH IGH
C     HAS BEEN BALANCED, THAT P(J) DENOTES THE INDEX INTERCHANGED
C     WITH J DURING THE PERMUTATION STEP, AND THAT THE ELEMENTS
C     OF THE DIAGONAL MATRIX USED ARE DENOTED BY D(I,J).  THEN
C        SCALE(J) = P(J),    FOR J = 1,...,LOW-1
C                 = D(J,J),      J = LOW,...,IGH
C                 = P(J)         J = IGH+1,...,N.
C     THE ORDER IN WHICH THE INTERCHANGES ARE MADE IS N TO IGH+1,
C     THEN 1 TO LOW-1.
C
C     NOTE THAT 1 IS RETURNED FOR IGH IF IGH IS ZERO FORMALLY.
C
C     THE ALGOL PROCEDURE EXC CONTAINED IN BALANCE APPEARS IN
C     QXZ146  IN LINE.  (NOTE THAT THE ALGOL ROLES OF IDENTIFIERS
C     K,L HAVE BEEN REVERSED.)
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     BASED ON THE EISPACK VERSION 3 ROUTINE BALANC, AS MODIFIED
C     BY COMPUTER SCIENCES CORPORATION, MAY 1984.
C
C     ------------------------------------------------------------------
C
      RADIX = 16.00
C
      B2 = RADIX * RADIX
      K = 1
      L = N
      GO TO 100
C     .......... IN-LINE PROCEDURE FOR ROW AND
C                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
C
         DO 110 I = 1, L
            IF (I .EQ. J) GO TO 110
            IF (A(J,I) .NE. 0.D0) GO TO 120
  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
C
         DO 150 I = K, L
            IF (I .EQ. J) GO TO 150
            IF (A(I,J) .NE. 0.D0) 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.D0
C     .......... ITERATIVE LOOP 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(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
C     ********** LAST CARD OF QXZ146 **********
      END
C --- SUBPROGRAM QXZ147 --- FORMERLY KNOWN AS ROUTINE ELMHES ---
C
C     ------------------------------------------------------------
      Forcesub QXZ147(NM,N,LOW,IGH,A,INT,temy) of NP ident ME
C
      INTEGER N,NM,IGH,LOW,INT(1)
      REAL A(N,N),temy(1)
      Shared INTEGER LA,KP1,MM1,MP1,IAM
      Shared Logical ilock
      Shared REAL X,Y,XMUL,XMUL1
```

```
      Private Real tema(1000)
      End Declarations
C     ----------------------------------------------------------------
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
  130    CONTINUE
         ENDIF
         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
```

```
c         temy(i)=0.0
c         else
c         temy(i)=a(i,mm1)/x
c         a(i,mm1)=temy(i)
c         endif
1301      continue
C
c         DO 160 I = MP1, IGH
c            Y = A(I,MM1)
c            IF (Y .EQ. 0.00) GO TO 160
c            Y = Y / X
c            A(I,MM1) = Y
C
          ENDIF
             do 1399 j=1,igh
1399         tema(j)=0.0
             jend=((igh-m)/8)*8
          Barrier
          iam=iam+1
          if(IAM.GT.NP) IAM=1
          End Barrier
          Barrier
          End Barrier
c            do 1400 jj=m+1,m+jend,8
             Presched DO 1400 jj=m+1,m+jend,8
CDIR$ IVDEP
             do 1401 j=1,m-1
c            a(j,m)=a(j,m)+temy(jj)*a(j,jj)+temy(jj+1)*a(j,jj+1)
             tema(j)=tema(j)+temy(jj)*a(j,jj)+temy(jj+1)*a(j,jj+1)
     1                    +temy(jj+2)*a(j,jj+2)+temy(jj+3)*a(j,jj+3)
     2                    +temy(jj+4)*a(j,jj+4)+temy(jj+5)*a(j,jj+5)
     3                    +temy(jj+6)*a(j,jj+6)+temy(jj+7)*a(j,jj+7)
1401         continue
1400         End Presched DO
             Barrier
             End Barrier
             Presched DO 1402 jj=jend+1+m,igh
CDIR$ IVDEP
             do 1403 j=1,m-1
1403         tema(j)=tema(j)+temy(jj)*a(j,jj)
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
              xmul=temy(kk)
              xmull=xmul*a(m,kk)
c              write(*,*) a(kk,m)
c             a(kk,m)=a(kk,m)-temy(kk)*a(m,m)
              a(kk,m)=a(kk,m)-xmul*a(m,m)
CDIR$ IVDEP
              do 1608 ik=m,kk
1608          a(ik,m)=a(ik,m)+xmul*a(ik,kk)
CDIR$ IVDEP
              do 1609 ik=kk+1,igh
1609          a(ik,m)=a(ik,m)+xmul*a(ik,kk)+xmull*temy(ik)
c              write(*,*) a(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             DO 150 J = 1, IGH
c 150         A(J,M) = A(J,M) + Y * A(J,I)
C
c 160    CONTINUE
C
c801    write(*,*)' M-th step A(i,mm1-n) igh,jend =',M,igh,jend
c       do 1500 II=1,N
c       write(*,1501) (a(ii,jj),jj=mm1,n)
c500   continue
c501   format(1x,10(e9.3,1x))
      Barrier
      IAM=IAM+1
      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)
      REAL C,F,G,R,S,B2,RADIX
      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     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 315-326(1971).
C
C     THIS SUBROUTINE BALANCES A REAL MATRIX AND ISOLATES
C     EIGENVALUES WHENEVER POSSIBLE.
C
```

```
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRIX.
C
C        A CONTAINS THE INPUT MATRIX TO BE BALANCED.
C
C     ON OUTPUT
C
C        A CONTAINS THE BALANCED MATRIX.
C
C        LOW AND IGH ARE TWO INTEGERS SUCH THAT A(I,J)
C          IS EQUAL TO ZERO IF
C           (1) I IS GREATER THAN J AND
C           (2) J=1,...,LOW-1 OR I=IGH+1,...,N.
C
C        SCALE CONTAINS INFORMATION DETERMINING THE
C           PERMUTATIONS AND SCALING FACTORS USED.
C
C     SUPPOSE THAT THE PRINCIPAL SUBMATRIX IN ROWS LOW THROUGH IGH
C     HAS BEEN BALANCED, THAT P(J) DENOTES THE INDEX INTERCHANGED
C     WITH J DURING THE PERMUTATION STEP, AND THAT THE ELEMENTS
C     OF THE DIAGONAL MATRIX USED ARE DENOTED BY D(I,J).  THEN
C        SCALE(J) = P(J),    FOR J = 1,...,LOW-1
C                 = D(J,J),      J = LOW,...,IGH
C                 = P(J)         J = IGH+1,...,N.
C     THE ORDER IN WHICH THE INTERCHANGES ARE MADE IS N TO IGH+1,
C     THEN 1 TO LOW-1.
C
C     NOTE THAT 1 IS RETURNED FOR IGH IF IGH IS ZERO FORMALLY.
C
C     THE ALGOL PROCEDURE EXC CONTAINED IN BALANCE APPEARS IN
C     BALANC  IN LINE.  (NOTE THAT THE ALGOL ROLES OF IDENTIFIERS
C     K,L HAVE BEEN REVERSED.)
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
C
      RADIX = 16.0E0
C
      B2 = RADIX * RADIX
      K = 1
      L = N
      GO TO 100
C     .......... IN-LINE PROCEDURE FOR ROW AND
C                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
C
         DO 110 I = 1, L
            IF (I .EQ. J) GO TO 110
            IF (A(J,I) .NE. 0.0E0) GO TO 120
  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
C
         DO 150 I = K, 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 = 0.0E0
            R = 0.0E0
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. 0.0E0 .OR. R .EQ. 0.0E0) GO TO 270
            G = R / RADIX
            F = 1.0E0
            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. 0.95E0 * S) GO TO 270
            G = 1.0E0 / 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)
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
      LOGICAL NOTLAS
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE HQR,
C     NUM. MATH. 14, 219-231(1970) BY MARTIN, PETERS, AND WILKINSON.
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 359-371(1971).
C
C     THIS SUBROUTINE FINDS THE EIGENVALUES OF A REAL
C     UPPER HESSENBERG MATRIX BY THE QR METHOD.
```

```
C
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRIX.
C
C        LOW AND IGH ARE INTEGERS DETERMINED BY THE BALANCING
C          SUBROUTINE  BALANC.  IF  BALANC  HAS NOT BEEN USED,
C          SET LOW=1, IGH=N.
C
C        H CONTAINS THE UPPER HESSENBERG MATRIX.  INFORMATION ABOUT
C          THE TRANSFORMATIONS USED IN THE REDUCTION TO HESSENBERG
C          FORM BY  ELMHES  OR  ORTHES, IF PERFORMED, IS STORED
C          IN THE REMAINING TRIANGLE UNDER THE HESSENBERG MATRIX.
C
C     ON OUTPUT
C
C        H HAS BEEN DESTROYED.  THEREFORE, IT MUST BE SAVED
C          BEFORE CALLING  HQR  IF SUBSEQUENT CALCULATION AND
C          BACK TRANSFORMATION OF EIGENVECTORS IS TO BE PERFORMED.
C
C        WR AND WI CONTAIN THE REAL AND IMAGINARY PARTS,
C          RESPECTIVELY, OF THE EIGENVALUES.  THE EIGENVALUES
C          ARE UNORDERED EXCEPT THAT COMPLEX CONJUGATE PAIRS
C          OF VALUES APPEAR CONSECUTIVELY WITH THE EIGENVALUE
C          HAVING THE POSITIVE IMAGINARY PART FIRST.  IF AN
C          ERROR EXIT IS MADE, THE EIGENVALUES SHOULD BE CORRECT
C          FOR INDICES IERR+1,...,N.
C
C        IERR IS SET TO
C          ZERO       FOR NORMAL RETURN,
C          J          IF THE LIMIT OF 30*N ITERATIONS IS EXHAUSTED
C                     WHILE THE J-TH EIGENVALUE IS BEING SOUGHT.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
C
      IERR = 0
      NORM = 0.0E0
      K = 1
C     .......... STORE ROOTS ISOLATED BY BALANC
C                AND COMPUTE MATRIX NORM ..........
      DO 50 I = 1, N
C
         DO 40 J = K, N
   40    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) = 0.0E0
   50 CONTINUE
C
      EN = IGH
      T = 0.0E0
      ITN = 30*N
C     .......... SEARCH FOR NEXT EIGENVALUES ..........
   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
         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     .......... 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. 0) GO TO 1000
      IF (ITS .NE. 10 .AND. ITS .NE. 20) GO TO 130
C     .......... FORM EXCEPTIONAL SHIFT ..........
      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 = 0.75E0 * S
      Y = X
      W = -0.4375E0 * S * S
  130 ITS = ITS + 1
      ITN = 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(Q) + ABS(R))
         IF (TST2 .EQ. TST1) GO TO 150
  140 CONTINUE
C
  150 MP2 = M + 2
C
      DO 160 I = MP2, EN
         H(I,I-2) = 0.0E0
         IF (I .EQ. MP2) GO TO 160
         H(I,I-3) = 0.0E0
  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 = 0.0E0
         IF (NOTLAS) R = H(K+2,K-1)
         X = ABS(P) + ABS(Q) + ABS(R)
         IF (X .EQ. 0.0E0) GO TO 260
         P = P / X
         Q = Q / X
         R = R / X
  170    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
  180    IF (L .NE. M) H(K,K-1) = -H(K,K-1)
  190    P = P + S
         X = P / S
         Y = Q / S
         ZZ = R / S
         Q = Q / P
         R = R / P
         IF (NOTLAS) GO TO 225
C     .......... ROW MODIFICATION ..........
         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 = MIN0(EN,K+3)
C     .......... COLUMN MODIFICATION ..........
         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
  225    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
  230    CONTINUE
C
         J = MIN0(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+1) = H(I,K+1) - P * Q
            H(I,K+2) = H(I,K+2) - P * R
  240    CONTINUE
  255    CONTINUE
C
  260 CONTINUE
C
      GO TO 70
C     .......... ONE ROOT FOUND ..........
  270 WR(EN) = X + T
      WI(EN) = 0.0E0
      EN = NA
      GO TO 60
C     .......... TWO ROOTS FOUND ..........
  280 P = (Y - X) / 2.0E0
      Q = P * P + W
      ZZ = SQRT(ABS(Q))
      X = X + T
      IF (Q .LT. 0.0E0) GO TO 320
C     .......... REAL PAIR ..........
      ZZ = P + SIGN(ZZ,P)
      WR(NA) = X + ZZ
      WR(EN) = WR(NA)
      IF (ZZ .NE. 0.0E0) WR(EN) = X - W / ZZ
      WI(NA) = 0.0E0
      WI(EN) = 0.0E0
      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
C                CONVERGED AFTER 30*N ITERATIONS ..........
 1000 IERR = EN
 1001 RETURN
```

```
      END
      Forcesub OELMHS(NM,N,LOW,IGH,A,RINDEX) of NP ident ME
C
      REAL
     + A(N,N), RINDEX(IGH)
      INTEGER
     + IGH, LOW, N, NM
      REAL
     + X, Y
      Shared INTEGER KP1,LA,MM1,MP1
      End Declarations
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE ELMHES,
C     NUM. MATH. 12, 349-368(1968) BY MARTIN AND WILKINSON.
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 339-358(1971).
C
C     GIVEN A REAL GENERAL MATRIX, THIS SUBROUTINE
C     REDUCES A SUBMATRIX SITUATED IN ROWS AND COLUMNS
C     LOW THROUGH IGH TO UPPER HESSENBERG FORM BY
C     STABILIZED ELEMENTARY SIMILARITY TRANSFORMATIONS.
C
C     ON INPUT
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT.
C
C        N IS THE ORDER OF THE MATRIX.
C
C        LOW AND IGH ARE INTEGERS DETERMINED BY THE BALANCING
C          SUBROUTINE  BALANC.  IF  BALANC  HAS NOT BEEN USED,
C          SET LOW=1, IGH=N.
C
C        A CONTAINS THE INPUT MATRIX.
C
C     ON OUTPUT
C
C        A CONTAINS THE HESSENBERG MATRIX.  THE MULTIPLIERS
C          WHICH WERE USED IN THE REDUCTION ARE STORED IN THE
C          REMAINING TRIANGLE UNDER THE HESSENBERG MATRIX.
C
C        RINDEX CONTAINS INFORMATION ON THE ROWS AND COLUMNS
C          INTERCHANGED IN THE REDUCTION.
C          ONLY ELEMENTS LOW THROUGH IGH ARE USED.
C
C     QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO BURTON S. GARBOW,
C     MATHEMATICS AND COMPUTER SCIENCE DIV, ARGONNE NATIONAL LABORATORY
C
C     THIS VERSION DATED AUGUST 1983.
C
C     ------------------------------------------------------------------
C
      LA = IGH - 1
      KP1 = LOW + 1
      IF (LA .LT. KP1) RETURN
```

```
      Barrier
C
      DO 180 M = KP1, LA
         MM1 = M - 1
         X = 0.0D0
         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. 0.0D0) GO TO 180
         MP1 = M + 1
C
         DO 160 I = MP1, IGH
            Y = A(I,MM1)
            IF (Y .EQ. 0.0D0) GO TO 160
            Y = Y / X
            A(I,MM1) = Y
C
            DO 140 J = M, N
  140       A(I,J) = A(I,J) - Y * A(M,J)
C
            DO 150 J = 1, IGH
  150       A(J,M) = A(J,M) + Y * A(J,I)
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
```

```
      LOGICAL NOTLAS
C
      IERR = 0
      NORM = 0.00
      K = 1
C     .......... STORE ROOTS ISOLATED BY QXZ146
C                AND COMPUTE MATRIX NORM ..........
      DO 50 I = 1, N
C
         DO 40 J = K, N
   40    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
   50 CONTINUE
C
      EN = IGH
      T = 0.00
      ITN = 30*N
C     .......... SEARCH FOR NEXT EIGENVALUES ..........
   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
         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
   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. 0) 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
```

```
      W = -0.43750 * S * S
  130 ITS = ITS + 1
      ITN = 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(Q) + ABS(R))
         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
  170    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
  180    IF (L .NE. M) H(K,K-1) = -H(K,K-1)
  190    P = P + S
         X = P / S
         Y = Q / S
         ZZ = R / S
         Q = Q / P
```

```
         R = R / P
         IF (NOTLAS) GO TO 225
C     .......... ROW MODIFICATION ..........
         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 = MIN0(EN,K+3)
C     .......... COLUMN MODIFICATION ..........
         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
  225    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
  230    CONTINUE
C
         J = MIN0(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+1) = H(I,K+1) - P * Q
            H(I,K+2) = H(I,K+2) - P * R
  240    CONTINUE
  255    CONTINUE
C
c       write(*,*)'NOTLAS,K,H(K,K)=',NOTLAS,K,H(K,K)
  260 CONTINUE
C
      GO TO 70
C     .......... ONE ROOT FOUND ..........
  270 WR(EN) = X + T
      WI(EN) = 0.00
      EN = NA
      GO TO 60
C     .......... TWO ROOTS FOUND ..........
  280 P = (Y - X) / 2.00
      Q = P * P + W
      ZZ = SQRT(ABS(Q))
      X = X + T
c      *****  the following if is added by Qin  **
      IF( Q.LT.0.00) go to 320
C     .......... REAL PAIR ..........
      ZZ = P + SIGN(ZZ,P)
      WR(NA) = X + ZZ
```

```
      WR(EN) = WR(NA)
      IF (ZZ .NE. 0.D0) WR(EN) = X - W / ZZ
      WI(NA) = 0.D0
      WI(EN) = 0.D0
      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
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......
       do 1 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)
       wk(i,2)=ei(j)
       endif
 2     continue
       er(locate)=999999999.
 1     continue
c......
       do 21 i=1,n
       er(i)=wk(i,1)
 21    ei(i)=wk(i,2)
c......
       write(6,*) 'real & imaginary evalues in ascending order'
       do 11 i=n-10,n
       write(6,*) i,er(i),ei(i)
 11    continue
       return
       end
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
February 1993

3. REPORT TYPE AND DATES COVERED
Contractor Report

4. TITLE AND SUBTITLE
Parallel-Vector Unsymmetric Eigen-Solver on High Performance Computers

5. FUNDING NUMBERS
585-03-11-05
NAS1-18584

6. AUTHOR(S)
Duc T. Nguyen and Qin Jiangning

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Old Dominion University Research Foundation
P. O. Box 6369
Norfolk, VA 23508-0369

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA CR-191417

11. SUPPLEMENTARY NOTES
Final Report
Langley Technical Monitor: Joseph E. Walz

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 60

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

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.

14. SUBJECT TERMS Eigenvalues, Unsymmetric Matrices, QR Algorithm, Parallel-Vector, High Perfornance Computers

15. NUMBER OF PAGES
33

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		

Completed
5-28-93

ERRATA

NASA Contractor Report 191417

Parallel-Vector Unsymmetric Eigen-Solver
on High Performance Computers

Duc T. Nguyen and Qin Jiangning

February 1993

The word "Unsymmetic" 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.

Issued May 1993