

**A Note on Solution of the Nearest Correlation Matrix Problem  
by von Neumann Matrix Divergence**

SK Mishra  
Dept. of Economics  
NEHU, Shillong (India)

Brian Culis (2005) in his paper makes an attempt to solve the nearest correlation matrix problem by using the Bregman matrix divergence and the von Neumann matrix divergence. Presently, we are concerned with the method (suggested by Culis) that uses the von Neumann divergence.

Given a matrix  $Q$  ( $-1 \leq q_{ij} \leq 1$ ;  $q_{ii} = 1$ ;  $q_{ij} \in Q$ ), we desire to obtain another matrix  $R$  ( $-1 \leq r_{ij} \leq 1$ ;  $r_{ii} = 1$ ;  $r_{ij} \in R$ ), such that  $R$  is nearest to  $Q$  in the von Neumann divergence sense. Culis suggests an iterative algorithm given as follows:

Set  $R^{(k)} = Q / \text{Trace}(Q)$ ;  $D^{(k)} = I$ ;  $k = 0$  ( $I$  is an identity matrix)

Repeat until convergence ( $\max_i |R_{ii}^{(k)} - R_{ii}^{(k+1)}| \leq \varepsilon$  pre-assigned) {

$$d_{ii}^{(k+1)} = \log \left( \sqrt{(1 - R_{ii}^{(k)}) / R_{ii}^{(k)}} \right) \quad \forall i ;$$

$$R^{(k+1)} = \exp(\log(R^{(k)}) + D^{(k+1)}) / \text{trace}[\exp(\log(R^{(k)}) + D^{(k+1)})];$$

$k \leftarrow k + 1$ ; }

$R^{(final)} = nR^{(last)}$ ;  $R^{(last)}$  is the  $R$  obtained finally through the above iterative procedure.

The computational issues are a little involved in obtaining the logarithm of  $R^{(k)}$  and the exponential of  $[R^{(k)} + D^{(k+1)}]$  matrices. The well-known Cayley-Hamilton theorem suggests that  $f(A) = V f(\Lambda) V'$  where  $V$  is the matrix of eigenvectors and  $\Lambda$  is (the diagonal) matrix of eigenvalues of  $A$ . Accordingly,  $\log(R^{(k)}) = V \log(\Lambda) V'$  where  $V$  is the matrix of eigenvectors and  $\Lambda$  is the matrix of eigenvalues of  $R^{(k)}$  and similarly  $\exp(R^{(k)} + D^{(k+1)}) = V \exp(\Lambda) V'$  where  $V$  is the matrix of eigenvectors and  $\Lambda$  is the matrix of eigenvalues of  $R^{(k)} + D^{(k+1)}$  (see Krisnamurthy & Sen, 1976).

What if one (or more) of the eigenvalues of  $R^{(k)}$  is either zero or negative?. The nearest correlation matrix problem is often associated with finding the nearest (positive semi-definite) matrix  $R$  from a given  $Q$ , which is a non-positive-definite matrix (see Rebonato & Jäckel, 1999; Higham, 2002; Anjos et al., 2003; Grubisic & Pietersz, 2004; Pietersz, & Groenen, 2004; Mishra, 2004, 2007). By being a non-positive-definite matrix,  $Q$  will obviously have (at least) one of its eigenvalues negative. In that case, it would not be possible to obtain the logarithm of that (negative) eigenvalue and consequently  $\log(R^{(k)})$ . So, the algorithm breaks down.

So that the algorithm given by Culis functions (somehow), one may set  $\log(\lambda_m) = -9999999999$  (or some negative number large in magnitude) for a non-positive eigenvalue,  $\lambda_m$  of  $R^{(k)}$ . A similar procedure was adopted by Rebonato & Jäckel (1999), who set negative eigenvalues to zero.

It is well-known (Krisnamurthy & Sen, 1976) that any real symmetric matrix  $A$  may be decomposed as  $A = A_1 + A_2 + \dots + A_n$  where  $A_j = u_j \lambda_j v_j$  while  $u_j$  is the solution of the characteristic equation  $Au_j = \lambda_j u_j$  and  $v_j$  is the transpose of  $u_j$ . Now if some (say,  $n-m > 0$ ) of the eigenvalues of  $R$  are negative and they are set to zero (or a very small positive number), it amounts to approximating  $A = \sum_{j=1}^n A_j$  by  $\hat{A} = \sum_{j=1}^m A_j$ . The

resulting  $E = A - \sum_{j=1}^m A_j$  or  $E = \sum_{j=m+1}^n A_j$  is the error matrix, assuming that the eigenvalues are arranged in a descending order,  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_m \geq \dots \geq \lambda_n$ . This is what was proposed by Rebonato & Jäckel (1999). The resulting A-E is a positive definite matrix. However,  $\hat{A} = \sum_{j=1}^m A_j$  may be devoid of the basic properties of a correlation matrix: first that every  $\hat{a}_{ii} = 1$  and secondly, that  $\sum_{i=1}^n \hat{a}_{ii} = n$  (the order of the matrix). This violation needs adjustments in the values of  $\hat{a}_{ij}$ .

An illustrative example will make the issue more clear. We take the matrix from Higham (2002). This matrix is a non-positive-definite matrix of order 3;  $q_{13} = q_{31} = 0$ ; other elements are unity. This Q can be decomposed into:

$$Q = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}; Q_1 = \begin{bmatrix} 0.603553 & 0.853553 & 0.603553 \\ 0.853553 & 1.207107 & 0.853553 \\ 0.603553 & 0.853553 & 0.603553 \end{bmatrix}; Q_2 = \begin{bmatrix} 0.5 & 0.0 & -0.5 \\ 0.0 & 0.0 & 0.0 \\ -0.5 & 0.0 & 0.5 \end{bmatrix}; Q_3 = \begin{bmatrix} -0.103553 & 0.146447 & -0.103553 \\ 0.146447 & -0.207107 & 0.146447 \\ -0.103553 & 0.146447 & -0.103553 \end{bmatrix}$$

$$Q_1 = \begin{bmatrix} 0.603553 & 0.853553 & 0.603553 \\ 0.853553 & 1.207107 & 0.853553 \\ 0.603553 & 0.853553 & 0.603553 \end{bmatrix}; Q_1 + Q_2 = \begin{bmatrix} 1.103553 & 0.853553 & 0.103553 \\ 0.853553 & 1.207107 & 0.853553 \\ 0.103553 & 0.853553 & 1.103553 \end{bmatrix}; Q_1 + Q_2 + Q_3 = \begin{bmatrix} 1.000000 & 1.000000 & 0.000000 \\ 1.000000 & 1.000000 & 1.000000 \\ 0.000000 & 1.000000 & 1.000000 \end{bmatrix}$$

It may be seen that neither  $Q_1$  nor  $Q_1 + Q_2$  have unitary diagonals; but  $Q_1 + Q_2 + Q_3$  matrix has this property. The reason is that the third eigenvalue is negative which plays the role. Addition of  $Q_3$  to  $Q_1 + Q_2$  brings all diagonals of  $Q_1 + Q_2 + Q_3$  matrix to make unity.

It is needed, therefore, that the elements of the  $Q_1 + Q_2$  matrix are adjusted such that its diagonals are all unity (and they sum up to the order, 3). A similar job is done by the von Neumann divergence suggested by Culis (with an adjustment, suggested by us, that for negative eigenvalues we set  $\log(\lambda_m) = -9999999999$ ). If we subject the original Q matrix of Higham as well as the  $[[Q_1] + [Q_2]]$  matrix (obtained from Q) to the

(modified) von Neumann divergence algorithm, we get the results very close to each other. Also note that the eigen structure of the two are close (but not identical)

$$\hat{Q} = [\hat{Q}_1 + Q_2] = \begin{bmatrix} 1.0000 & 0.7193 & 0.0348 \\ 0.7193 & 1.0000 & 0.7193 \\ 0.0348 & 0.7193 & 1.0000 \end{bmatrix}; \tilde{Q} = \begin{bmatrix} 1.0000 & 0.7103 & 0.0091 \\ 0.7103 & 1.0000 & 0.7103 \\ 0.0091 & 0.7103 & 1.0000 \end{bmatrix}; \tilde{\Lambda}(Q_1 + Q_2) = \begin{bmatrix} 2.0348 \\ 0.9652 \\ 0.0000 \end{bmatrix}; \tilde{\Lambda}(Q) = \begin{bmatrix} 2.0091 \\ 0.9909 \\ 0.0000 \end{bmatrix}$$

Which of the two would be a better procedure to obtain the nearest correlation from Q? The norm =  $\left[ \sum_{i=1}^n \sum_{j=1}^n (q_{ij} - \hat{q}_{ij})^2 \right]^{0.5} = 0.563541$  obtained from  $Q_1 + Q_2$  is smaller than the norm =  $\left[ \sum_{i=1}^n \sum_{j=1}^n (q_{ij} - \tilde{q}_{ij})^2 \right]^{0.5} = 0.579487$  obtained from Q directly. It appears, therefore, that when Q is a non-positive-definite matrix, it is better to decompose it, free it from the influences of negative eigenvalues and then use the von Neumann divergence algorithm for adjustment.

In any case, the question remains: are these approximate (nearest?) correlation matrices **actually nearest** to Q? We compare it with the nearest correlation matrix of Q obtained by Higham (2002) and Mishra (2007). Mishra and Higham obtained the nearest correlation matrices,  $\hat{Q}_M$  and  $\hat{Q}_H$  (to Higham's Q) as follows (approximated at the 4<sup>th</sup> place after decimal):

$$\hat{Q}_H = \begin{bmatrix} 1.0000 & 0.7607 & 0.1573 \\ 0.7607 & 1.0000 & 0.7607 \\ 0.1573 & 0.7607 & 1.0000 \end{bmatrix}; \Lambda = \begin{bmatrix} 2.1573 \\ 0.8427 \\ 0.0000 \end{bmatrix}; \hat{Q}_M = \begin{bmatrix} 1.0000 & 0.7607 & 0.1573 \\ 0.7607 & 1.0000 & 0.7607 \\ 0.1573 & 0.7607 & 1.0000 \end{bmatrix}; \Lambda = \begin{bmatrix} 2.1573 \\ 0.8427 \\ 0.0000 \end{bmatrix}$$

$$\text{The norm} = \left[ \sum_{i=1}^n \sum_{j=1}^n (q_{ij} - \hat{q}_{ij})^2 \right]_{\text{Mishra}}^{0.5} = 0.527797 \text{ and norm} = \left[ \sum_{i=1}^n \sum_{j=1}^n (q_{ij} - \hat{q}_{ij})^2 \right]_{\text{Higham}}^{0.5} = 0.527790 .$$

These norms are smaller than those obtained through the (modified) von Neumann divergence procedure. Clearly, the procedure gives sub-optimal solution to the nearest correlation matrix problem.

We may conclude therefore that the proposed von Neumann approximation of a non-positive-definite correlation matrix is either infeasible or sub-optimal. First, if a given matrix is already positive semi-definite, there is no need to obtain any other positive semi-definite matrix closest to it. It is in itself its own closest matrix. Secondly, when the given matrix is non-positive-definite, then only we seek a positive semi-definite matrix closest to it. Then the proposed procedure fails as we cannot find  $\log(Q)$ . In that case, if we replace negative eigenvalues of Q by a zero/near-zero values, we obtain a positive semi-definite matrix, but it is not nearest to the Q matrix; there are indeed other procedures to obtain better approximation (see the algorithm in the appendix).

However, the proposed method of obtaining the nearest correlation matrix from a non-positive-definite pseudo-correlation matrix (Q) is not devoid of all merits. It is, perhaps, one of the fastest (although sub-optimal) method of obtaining positive semi-definite matrices and works well for large matrices. But, choosing a right perturbation

parameter (that is used in place of non-positive eigenvalues) is a matter of trial and error. Vis-à-vis this, many other methods are quite slow. Especially, the method based on differential evolution (Mishra, 2007) is very slow, but since it is amenable to parallelization, it may work for larger matrices as well.

## References

- Anjos, MF, Higham, NJ, Takouda, PL and Wolkowicz, H (2003) “A Semidefinite Programming Approach for the Nearest Correlation Matrix Problem”, *Preliminary Research Report*, Dept. of Combinatorics & Optimization, Waterloo, Ontario.
- Culis, Brian (2005). “The Nearest Correlation Matrix”, [Uses Bregman von Neumann divergences] available at <http://www.cs.utexas.edu/users/kulis/cs383c/project.pdf>
- Grubisic, I and Pietersz, R (2004) “Efficient Rank Reduction of Correlation Matrices”, *Working Paper Series*, SSRN, <http://ssrn.com/abstract=518563>
- Higham, NJ (2002). “Computing the Nearest Correlation Matrix – A Problem from Finance”, *IMA Journal of Numerical Analysis*, 22, pp. 329-343.
- Krishnamurthy, EV and Sen, SK (1976) *Computer-Based Numerical Algorithms*, Affiliated East West Press Pvt. Ltd, New Delhi.
- Mishra, SK (2004) “Optimal Solution of the Nearest Correlation Matrix Problem by Minimization of the Maximum Norm”. <http://ssrn.com/abstract=573241>
- Mishra, SK (2007) “The Nearest Correlation Matrix Problem: Solution by Differential Evolution Method of Global Optimization”, <http://ideas.repec.org/p/pramprapa/2760.html>
- Pietersz, R and Groenen, PJF (2004) “Rank Reduction of Correlation Matrices by Majorization”, *Econometric Institute Report EI 2004-11*, Erasmus Univ. Rotterdam.
- Rebonato, R and Jäckel, P (1999) “The Most General Methodology to Create a Valid Correlation Matrix for Risk Management and Option Pricing Purposes”, *Quantitative Research Centre, NatWest Group*, <http://www.rebonato.com/CorrelationMatrix.pdf>

ncor

```

PROGRAM NCORMAT ! -----
C (OBTAINS POSITIVE SEMI-DEFINITE CORRELATION MATRIX (R)
C FROM A NON-POSITIVE-SEMI-DEFINITE or PSEUDO-CORRELATION MATRX (Q)
C Non-Positive-Semi-definite matrix has at least one - BUT NOT ALL -
C of its eigenvalues negative
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C PARAMETER (NORM=2)
C DIMENSION A(200,200),V(200,200),W(200,200),P(200),R(200,200)
C DIMENSION D(200,200),Q(200,200),C(200,200),MM(200),QQ(200,200)
C CHARACTER *80 INFIL,OUTFIL
C COMMON /RNDM/IU,IV
C -----
C WRITE(*,*) 'THE ORDER OF Q MATRIX ?'
C READ(*,*) N
C WRITE(*,*) 'MATRIX Q TO BE READ OR GENERATED? TO BE READ FEED(0)'
C WRITE(*,*) 'IF Q TO BE GENERATED, THEN FEED(ANY NON-ZERO NUMBER)'
C READ(*,*) IGEN
C WRITE(*,*) 'OUTPUT FILE NAME (TO STORE THE RESULTS)?'
C READ(*,*) OUTFIL
C WRITE(*,*) 'NUMBER OF ITERATIONS(SAY 100) AND ACC (SAY 1.0D-10)?'
C WRITE(*,*) 'ACC IS A SMALL NUMBER FOR ACCURACY'
C READ(*,*) NIT,ACC
C WRITE(*,*) 'FEED A PERTURBATION VALUE TO REPLACE NONPOSITIVE EIGEN'
C WRITE(*,*) 'VALUE. IT WOULD BE A SMALL POSITIVE VALUE, SAY, 1.D-10'
C READ(*,*) EPS
C -----
C IF(IGEN.EQ.0) THEN
C WRITE(*,*) 'INPUT FILE NAME (TO READ THE Q MATRIX)?'
C READ(*,*) INFIL
C OPEN(7,FILE=INFIL) ! OPENING THE INUT FILE
C DO I=1,N
C READ(7,*)(Q(I,J),J=1,N)
C ENDDO
C CLOSE(7) ! CLOSING THE INUT FILE
C -----
C ELSE
C WRITE(*,*) 'FEED SEED'
C READ(*,*) IU
C DO I=1,N
C DO J=1,N
C CALL RANDOM(RAND)
C Q(I,J)=0.2D0+RAND*.7 ! OR BY SOME OTHER SPECIFICATION
C IF(I.EQ.J) Q(I,J)=1.D0
C ENDDO
C ENDDO
C ENDIF
C -----
C OPEN(7,FILE=OUTFIL) !OPENING THE OUTPUT FILE, WILL BE CLOSED LATER
C WRITE(7,*) 'THE ORIGINAL MATRIX (Q) IS GIVEN BELOW'
C DO I=1,N
C WRITE(7,1)(Q(I,J),J=1,N)
C ENDDO
C 1 FORMAT(10F8.4)
C ----- STORE Q IN A: WORK WITH A, PRESERVE Q -----
C DO I=1,N
C DO J=1,N
C A(I,J)=Q(I,J)
C ENDDO
C ENDDO
C CALL EIGEN(A,N,NN,V,W,P,MM) ! FIND EIGENVALUES AND VECTORS OF A
C WRITE(*,*) 'THE EIGENVALUES OF THE ORIGINAL MATRIX ARE AS FOLLOWS'

```

```

ncor
WRITE(7,*)'THE EIGENVALUES OF THE ORIGINAL MATRIX ARE AS FOLLOWS'
WRITE(*,*)(A(I,I),I=1,N)
WRITE(7,*)(A(I,I),I=1,N)
WRITE(*,*)'-----'
C  A IS DISTURBED DUE TO INVOKING EIGEN. RESTORE IT, INITIALIZE OTHERS
DO I=1,N
DO J=1,N
W(I,J)=0.D0 ! INITIALISE AND KEEP NONNEGATIVE EIGENVALUES IN W(I,I)
IF(A(I,J).GT.0.D0 .AND. I.EQ.J) W(I,J)=A(I,J)
IF(I.EQ.J.AND.W(I,J).EQ.0.D0) W(I,J)=EPS
A(I,J)=Q(I,J) ! RESTORE THE ORIGINAL MATRIX IN A
D(I,J)=0.D0 ! INITIALIZE D MATRIX BY ZERO
ENDDO
ENDDO
C  -----
S=1.0D30 ! NORM VERY LARGE TO BEGIN WITH

DO IT=1,NIT ! ITERATION BEGINS
DO I=1,N
DO J=1,N
R(I,J)=A(I,J)-D(I,J)
A(I,J)=R(I,J)
ENDDO
ENDDO

CALL EIGEN(A,N,NN,V,W,P,MM) ! FIND EIGENVALUES AND VECTORS OF A
DO I=1,N
DO J=1,N
W(I,J)=0.D0
IF(I.EQ.J .AND. A(I,J).GT.0.D0) W(I,J)=A(I,J)
IF(I.EQ.J.AND.W(I,J).EQ.0.D0) W(I,J)=EPS
ENDDO
ENDDO
C  CONSTRUCTING THE MATRIX FROM EIGENVECTORS AND NON-NEG EIGENVALUES
DO I=1,N
DO J=1,N
C(I,J)=0.D0
DO K=1,N
C(I,J)=C(I,J)+V(I,K)*W(K,J)
ENDDO
ENDDO
ENDDO
DO I=1,N
DO J=1,N
A(I,J)=0.D0
DO K=1,N
A(I,J)=A(I,J)+C(I,K)*V(J,K)
ENDDO
ENDDO
ENDDO
C  MATRIX HAS BEEN RECONSTRUCTED
DO I=1,N
DO J=1,N
D(I,J)=A(I,J)-R(I,J)
IF(I.EQ.J) A(I,J)=1.D0
ENDDO
ENDDO

C  COMPUTE NORM
SS=0.D0
DO I=1,N
DO J=1,N
SS=SS+DABS(A(I,J)-Q(I,J))**2

```

ncor

```
ENDDO
ENDDO
IF(DABS(SS-S).LT.ACC) THEN
S=SS
GO TO 10
ELSE
S=SS
ENDIF
ENDDO ! ITERATIONS END
```

```
C -----
C FINAL RESULTS
10 WRITE(*,*)'NO. OF ITERATIONS PERFORMED = ',IT
```

```
WRITE(7,*)'FINAL RESULTS - THE OUTPUT MATRIX'
DO I=1,N
WRITE(7,1)(A(I,J),J=1,N)
DO J=1,N
R(I,J)=A(I,J)
ENDDO
ENDDO
WRITE(7,*)'-----THE FINAL EIGENVALUES ARE -----'
CALL EIGEN(R,N,NN,V,W,P,MM)
WRITE(7,*)(R(I,I),I=1,N)
WRITE(7,*)'NORM=',SS**(1.D0/NORM)
WRITE(7,*)'NO. OF ITERATIONS PERFORMED = ',IT
CLOSE(7)! CLOSE THE OUTPUT FILE
```

```
C -----
C WRITE(*,*)'RESULTS HAVE BEEN STORED IN THE OUTPUT FILE'
C WRITE(*,*)'END'
C END
```

```
C -----
C SUBROUTINE EIGEN(A,N,NN,V,W,P,MM)
C THIS SUBROUTINE IS ADAPTED FROM KRISHNAMURTHY AND SEN (1976)
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C DIMENSION A(200,200),V(200,200),W(200,200),P(200)
C DIMENSION MM(200)
```

```
C ----- INITIALISATION -----
```

```
NN=1
DO 50 I=1,N
DO 51 J=1,N
51 V(I,J)=0.0
W(I,J)=0.0
P(I)=0.0
50 CONTINUE
PMAX=0
EPLN=0
TAN=0
SIN=0
COS=0
AI=0
TT=0
EPLN=1.0D-310
C EPLN=1.0D-99
```

```
C -----
C IF(NN.NE.0) THEN
C DO 3 I=1,N
C DO 3 J=1,N
C V(I,J)=0.0
C IF(I.EQ.J) V(I,J)=1.0
3 CONTINUE
C ENDFIF
2 NR=0
```

ncor

```
5      MI=N-1
      DO 6 I=1,MI
      P(I)=0.0
      MJ=I+1
      DO 6 J=MJ,N
      IF(P(I).GT.DABS(A(I,J))) GO TO 6
      P(I)=DABS(A(I,J))
      MM(I)=J
6      CONTINUE
7      DO 8 I=1,MI
      IF(I.LE.1) GOTO 10
      IF(PMAX.GT.P(I)) GOTO 8
10     PMAX=P(I)
      IP=I
      JP=MM(I)
8      CONTINUE
      IF (PMAX.LE.EPLN) THEN
      GO TO 12
      ENDIF
      NR=NR+1
13     TA=2.0*A(IP,JP)
      TB=(DABS(A(IP,IP)-A(JP,JP))+
* DSQRT((A(IP,IP)-A(JP,JP))**2+4.0*A(IP,JP)**2))
      TAN=TA/TB
      IF(A(IP,IP).LT.A(JP,JP)) TAN=-TAN
14     COS=1.0/DSQRT(1.0+TAN**2)
      SIN=TAN*COS
      AI=A(IP,IP)
      A(IP,IP)=(COS**2)*(AI+TAN*(2.0*A(IP,JP)+TAN*A(JP,JP)))
      A(JP,JP)=(COS**2)*(A(JP,JP)-TAN*(2.0*A(IP,JP)-TAN*AI))
      A(IP,JP)=0.0
      IF(A(IP,IP).GE.A(JP,JP)) GO TO 15
      TT=A(IP,IP)
      A(IP,IP)=A(JP,JP)
      A(JP,JP)=TT
      IF(SIN.GE.0) GO TO 16
      TT=COS
      GO TO 17
16     TT=-COS
17     COS=DABS(SIN)
      SIN=TT
15     DO 18 I=1,MI
      IF(I-IP) 19, 18, 20
      IF(I.EQ.JP)GO TO 18
20     IF(MM(I).EQ.IP) GO TO 21
19     IF(MM(I).NE.JP) GO TO 18
21     K=MM(I)
      TT=A(I,K)
      A(I,K)=0.0
      MJ=I+1
      P(I)=0.0
      DO 22 J=MJ,N
      IF(P(I).GT.DABS(A(I,J))) GO TO 22
      P(I)=DABS(A(I,J))
      MM(I)=J
22     CONTINUE
      A(I,K)=TT
18     CONTINUE
      P(IP)=0.0
      P(JP)=0.0
      DO 23 I=1,N
      IF(I-IP) 24, 23, 25
24     TT=A(I,IP)
```

ncor

```
A(I,IP)=COS*TT+SIN*A(I,JP)
IF(P(I).GE.DABS(A(I,IP))) GO TO 26
P(I)=DABS(A(I,IP))
MM(I)=IP
26 A(I,JP)=-SIN*TT+COS*A(I,JP)
IF(P(I).GE.DABS(A(I,JP))) GO TO 23
30 P(I)=DABS(A(I,JP))
MM(I)=JP
GO TO 23
25 IF(I.LT.JP) GO TO 27
IF(I.GT.JP) GO TO 28
IF(I.EQ.JP) GO TO 23
27 TT=A(IP,I)
A(IP,I)=COS*TT+SIN*A(I,JP)
IF(P(IP).GE.DABS(A(IP,I))) GO TO 29
P(IP)=DABS(A(IP,I))
MM(IP)=I
29 A(I,JP)=-TT*SIN+COS*A(I,JP)
IF(P(I).GE.DABS(A(I,JP))) GO TO 23
GO TO 30
28 TT=A(IP,I)
A(IP,I)=TT*COS+SIN*A(JP,I)
IF(P(IP).GE.DABS(A(IP,I))) GO TO 31
P(IP)=DABS(A(IP,I))
MM(IP)=I
31 A(JP,I)=-TT*SIN+COS*A(JP,I)
IF(P(JP).GE.DABS(A(JP,I))) GO TO 23
P(JP)=DABS(A(JP,I))
MM(JP)=I
23 CONTINUE
IF(NN.EQ.0) GOTO 7
DO 32 I=1,N
TT=V(I,IP)
V(I,IP)=TT*COS+SIN*V(I,JP)
V(I,JP)=-TT*SIN+COS*V(I,JP)
32 CONTINUE
GO TO 7
12 RETURN
END
C -----
C RANDOM NUMBER GENERATOR (UNIFORM BETWEEN 0 AND 1 - BOTH EXCLUSIVE)
SUBROUTINE RANDOM(RAND)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
COMMON /RNDM/IU,IV
IV=IU*65539
IF(IV.LT.0) THEN
IV=IV+2147483647+1
ENDIF
RAND=DBLE(IV+.0D0)
IU=IV
RAND=RAND*0.4656613E-09
RETURN
END
C -----
```