Completing correlation matrices of arbitrary order by differential evolution method of global optimization: A Fortran program

Mishra, SK

North-Eastern Hill University, Shillong (India)

5 March 2007

Online at https://mpra.ub.uni-muenchen.de/31282/
MPRA Paper No. 31282, posted 06 Jun 2011 07:45 UTC
Completing Correlation Matrices of Arbitrary Order by Differential Evolution Method of Global Optimization: A Fortran Program

SK Mishra
Dept. of Economics
North-Eastern Hill University
Shillong (India)

Introduction: A product moment correlation matrix $R$ of order $n$ is a (square) symmetric positive semi-definite matrix such that $r_{ij} = r_{ji} \in R$ lies between $-1$ and $1$. Moreover, $r_{ii} = 1$. Each $r_{ij}$ is the cosine of angle $\theta$ between two variates, say $x_i$ and $x_j$, $i, j \in \{1, 2, ..., n\}$. Such matrices have many applications, particularly in marketing and financial economics as reflected in the works of Chesney and Scott (1989), Heston (1993), Schöbel and Zhu (1999), Tyagi and Das (1999), Xu and Evers (2003), etc. The need to forecast demand for a group of products in order to realize savings by properly managing inventories requires the use of correlation matrices (Budden et al. 2007).

In some cases, the matrix available to the analyst/decision-maker is complete, but it is an invalid (not positive semi-definite) correlation matrix. There could be many reasons that give rise to such invalid matrices (Mishra, 2004). In such cases, the problem is to obtain an approximate semi-definite correlation matrix, which, in some sense, is closest to the given invalid matrix. A number of methods have been developed to obtain such nearest correlation matrices. The works of Rebonato and Jäckel (1999), Higham (2002), Anjos et al. (2003), Pietersz and Groenen (2004), Grubisic and Pietersz (2004) and Mishra (2004) are some of them.

In many cases, however, due to paucity of data/information or dynamic nature of the problem at hand, it is not possible to obtain a complete correlation matrix. Some elements of $R$ are unknown. In such cases, the question of validity (semi-definiteness) or otherwise (of an incomplete correlation matrix) does not arise. Instead, the problem is to obtain a valid complete correlation matrix. In absence of sufficient side conditions that are often impracticable to specify, this problem cannot be solved uniquely.

Several methods have been suggested to complete a correlation matrix - that is to obtain a valid complete correlation matrix from an incomplete correlation matrix (some of whose elements are unknown). Works of Johnson (1980), Barett et al. (1989), Helton et al. (1989), Grone et al. (1984), Barett et al. (1998), Laurent (2001), Kahl and Jäckel (2005), Kahl and Günther (2005), etc are notable.

In view of non-unique solutions admissible to the problem of completing the correlation matrix, some authors have suggested numerical methods that provide ranges to different unknown elements. Stanley and Wang (1969), Glass and Collins (1970) and Olkin (1981) have suggested very efficient methods to find such ranges for the unknown elements of very small correlation matrices (of order $n < 4$). Budden (2007) suggests a method to obtain the ranges of missing values of elements of a 4×4 incomplete correlation matrix whose first row elements are known. With the known elements in the
first row, the method sets the range for \( r_{23} \) and one has to specify its value in that range. Once the value of \( r_{23} \) is chosen (within the specified range set for it), the method yields the range in which \( r_{24} \) would lie. One has to specify the value of \( r_{24} \) within the given range, which yields the range for \( r_{34} \). Thus the matrix is completed. In this procedure it is obvious that the ranges on latter elements are contingent upon the choice of values of former elements. Further, Budden’s method is limited to a 4×4 correlation matrix.

Objective of the Present Paper: Our objective in this paper is to suggest a method (and provide a Fortran program) that completes a given incomplete correlation matrix of an arbitrary order. The resulting complete matrices are many in number, but all of them are valid (positive semi-definite – with all non-negative eigenvalues). Additionally, the suggested method does not require any pre-assigned pattern as in case of Budden’s method. It allows for holes (unknown elements) in any row and any column. The program that works out such complete matrices does not require any interaction with the user either.

The Method: The method proposed here is based on the Differential Evolution (DE) procedure of global optimization (Storn and Price, 1995). It generates a random population of elements that fit the holes (\( m \) in number) in the given incomplete correlation matrix, yielding valid correlation matrices whose eigenvalues are all non-negative summing up to the order of the matrix, which is also the trace of the matrix.

The differential Evolution method is perhaps the fastest evolutionary computational procedure yielding most accurate solutions to continuous global optimization problems. It consists of three basic steps: (i) generation of (large enough) population with individuals in the m-dimensional space, randomly distributed over the entire domain of the function in question and evaluation of the individuals of the so generated by finding \( f(x) \), where \( x \) is the decision variable; (ii) replacement of this current population by a better fit new population, and (iii) repetition of this replacement until satisfactory results are obtained or certain criteria of termination are met.

The strength of DE lays on replacement of the current population by a new population that is better fit. Here the meaning of ‘better’ is in the Pareto improvement sense. A set \( S_a \) is better than another set \( S_b \) \textbf{iff} : (i) \( \text{no } x_i \in S_a \text{ is inferior to the corresponding member of } x_i \in S_b \); and (ii) \( \text{at least one } x_k \in S_a \text{ is better than the corresponding member } x_k \in S_b \). Thus, every new population is an improvement over the earlier one. To accomplish this, the DE method generates a candidate individual to replace each current individual in the population. A crossover of the current individual and three other randomly selected individuals obtains the candidate individual from the current population. The crossover itself is probabilistic in nature. Further, if the candidate individual is better fit than the current individual, it takes the place of the current individual else the current individual passes into the next iteration (Mishra, 2006).

In the present application of DE, the ‘complete correlation problem’ is cast into a minimization problem. It may be noted that the problem has innumerably many minima
and we need multiple solutions. Such problems cannot be solved satisfactorily by conventional optimization procedures. A stochastic population method such as DE or PSO (Particle Swarm Optimization) may, therefore, be a suitable choice. In the scheme of DE, a population of $N$ individuals (each represented by a $m$-dimensional vector, of which each element lies between $-1$ and $1$) is generated by using uniformly distributed random numbers whose each vector provides the candidate values filling in the $m$ number of holes (unknown elements) of the given incomplete matrix. The eigenvalues of the resulting matrices are computed and positive penalties are set if any of them is negative. Minimization of this formulation results into zero penalty, and the solution so obtained yields a valid correlation matrix. Since each individual in the population has gravitational pull to the global optimum, it corresponds to a valid correlation matrix. Thus, we obtain $N$ number of valid correlation matrices.

The Structure of Computer Program and Hints on its Use: The main program (in Fortran) to complete a correlation matrix has eight subroutines. The main program reads the input matrix from a file specified by the user. This file stores the main diagonal and upper diagonal elements of the given matrix. Thus the first row has $n$ elements beginning with 1.0; the second row has $n-1$ elements beginning with 1.0 and so on such that the last ($n^{th}$) row has only one element (=1.0). In making the input matrix file one has to indicate the known and the unknown elements differently. While the known elements naturally lie between $-1$ and $1$ they are put as they really are. However, a number lying beyond the range $[-1, 1]$ represents an unknown element. The value could be any number such as 2, -3, 1.5, etc that cannot be a correlation coefficient. For example, if $r_{ij}$ is known to be 0.73, say, it will be put as 0.73, but if $r_{ij}$ is unknown it may be represented by a number, say 2.0 or $-1.9$ and so on. A number outside the range [-1,1] indicates that it is a hole or an unknown correlation coefficient. When the program runs, it asks for the order of input matrix (morder) and the name of input data file in which the input matrix is already stored. The user has to specify them. The program also asks to name the output file in which the final results (valid correlation matrices) would be stored. The user should specify it. Then the program asks for a random number seed. Any 4-digit odd number (say 1271) can be fed as a seed. Subsequently, the program asks for the number of unknown elements (m) in the input matrix. This also has to be given by the user. The main program calls subroutine DE (differential Evolution optimizer). It asks for inputs from the user; the population size (N) and the number of iteration to be performed. The population size determines the number of valid matrices to be obtained as output. It should be normally 100 or so, but for larger problems, this number should be larger. The number of iterations should be specified at 1000 or larger. Then the program needs another random number seed that could be any 4-digit odd number. Once these inputs are given, DE starts running.

Other subroutines in the program are: Normal (generates normally distributed random numbers), Random (generates uniformly distributed random numbers between 0 and 1), Fselect (chooses a function), Func (organizes function calls), Eigen (computes eigenvalues and vectors), Concor (constructs correlation matrices for optimization) and Ncorx (constructs valid correlation matrices and stores them in the output file specified by the user). The output file may be opened in notepad or by any editor program (edit or
Microsoft Word of Microsoft Windows) to obtain the results. The source codes (Fortran programs) are appended here. Directly usable source codes that may be cut and pasted in an editor may be downloaded from http://www1.webng.com/economics/complete-cormat.txt or http://www.freewebs.com/nehu_economics/complete-cormat.txt. A Fortran compiler may be obtained from http://www.thefreecountry.com/compilers/fortran.shtml or http://www.download.com/Force/3000-2069_4-10233344.html freely. The source codes may be pasted in the Force editor directly. Presently, the dimensions in the program are set to deal with the matrices of order 10 or less. If needed, they may be increased suitably for larger matrices.

**An Example:** An incomplete matrix of order 7 (=morder =n) given in table-1 is used as an example to illustrate an application of the method and program given in this paper. It has 12 (=m) holes or unknown elements (colored red). They have been assigned an invalid number (5), outside the permissible range [-1, 1]. Other numbers in the range [-1, 1] are known elements of the matrix. The program is run for population size N=100 and it gives N valid correlation matrices. Two sample matrices from the output are given in table-2 and table-3. The program also gives the eigenvectors for each valid correlation matrix, but they are not presented here.

<table>
<thead>
<tr>
<th>Table-1. Input Correlation Matrix with Some Unknown Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
</tr>
<tr>
<td>1.00</td>
</tr>
<tr>
<td>1.00</td>
</tr>
<tr>
<td>1.00</td>
</tr>
<tr>
<td>1.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table-2. Sample Output Correlation Matrix and its Eigenvalues</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00000000-0.50000000 0.50000000-0.50000000 0.56000000 0.21000000 0.34000000</td>
</tr>
<tr>
<td>-0.50000000 1.00000000-0.0285722 0.1840863-0.0967958 0.30000000 0.16000000</td>
</tr>
<tr>
<td>0.50000000-0.0285722 1.00000000-0.0249011 0.3674891 0.1476330 0.89000000</td>
</tr>
<tr>
<td>-0.50000000-0.1840863-0.0249011 1.00000000 0.0894851-0.0430459-0.0959958</td>
</tr>
<tr>
<td>0.56000000-0.0967958 0.3674891 0.0894851 1.00000000 0.2641564 0.2404028</td>
</tr>
<tr>
<td>-0.50000000 0.1840863-0.0249011 1.00000000-0.0030405 0.0628572 0.0727079</td>
</tr>
<tr>
<td>0.56000000-0.0285722 1.00000000-0.1237942 0.1573758 0.0478572 0.89000000</td>
</tr>
<tr>
<td>-0.50000000-0.162682-0.1237942 1.00000000-0.0030405 0.0628510-0.0986661</td>
</tr>
<tr>
<td>0.56000000-0.3235212 0.1573758-0.0030405 1.00000000 0.0528261 0.0727079</td>
</tr>
<tr>
<td>-0.21000000 0.30000000 0.0478572 0.0628510 0.0528261 1.00000000-0.0683791</td>
</tr>
<tr>
<td>0.34000000-0.16000000 0.89000000-0.0986661 0.0727079-0.0683791 1.00000000</td>
</tr>
<tr>
<td>EIGENVALUES, SUM AND PRODUCT OF EIGENVALUES</td>
</tr>
<tr>
<td>2.6161856 1.5874846 1.1577154 1.0120181 0.468504 0.0975220 0.060439</td>
</tr>
<tr>
<td>1.0000000 0.134378</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table-3. Sample Output Correlation Matrix and its Eigenvalues</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0000000-0.50000000 0.50000000-0.50000000 0.56000000 0.21000000 0.34000000</td>
</tr>
<tr>
<td>-0.50000000 1.00000000 0.0784965-0.0162682-0.3235212 0.30000000 0.16000000</td>
</tr>
<tr>
<td>0.50000000 0.0784965 1.00000000-0.1237942 0.1573758 0.0478572 0.89000000</td>
</tr>
<tr>
<td>-0.50000000-0.162682-0.1237942 1.00000000-0.0030405 0.0628510-0.0986661</td>
</tr>
<tr>
<td>0.56000000-0.3235212 0.1573758-0.0030405 1.00000000 0.0528261 0.0727079</td>
</tr>
<tr>
<td>-0.21000000 0.30000000 0.0478572 0.0628510 0.0528261 1.00000000-0.0683791</td>
</tr>
<tr>
<td>0.34000000-0.16000000 0.89000000-0.0986661 0.0727079-0.0683791 1.00000000</td>
</tr>
<tr>
<td>EIGENVALUES, SUM AND PRODUCT OF EIGENVALUES</td>
</tr>
<tr>
<td>2.4707041 1.609468 1.1648713 1.0422329 0.5538390 0.0926969 0.0147091</td>
</tr>
<tr>
<td>7.0000000 0.0037623</td>
</tr>
</tbody>
</table>
Conclusion: The method (and the program) given here has an advantage over other algorithms due to its ability to present a scenario of valid correlation matrices that might be obtained from a given incomplete matrix of an arbitrary order. The analyst may choose some particular matrices, most suitable to his purpose, from among those output matrices. Further, unlike other methods, it has no restriction on the distribution of holes over the entire matrix, nor the analyst has to interactively feed elements of the matrix sequentially (as in Budden’s scheme) which might be quite inconvenient for larger matrices. It is flexible and by merely choosing larger population size (N) one might obtain a more exhaustive scenario of valid matrices. As the number of holes increases, the program takes longer time no doubt, but for smaller number of holes it takes a small time even if the input matrix is quite large. This is a special advantage of this method.

References

complete-cormat

C MAIN PROGRAM : GENERATE A SEMIPOSITIVE CORRELATION MATRIX FROM
C A GIVEN CORRELATION MATRIX WITH SOME KNOWN ELEMENTS
C -------------------------------------------------------------------------------------------------
C METHOD:DIFFERENTIAL EVOLUTION
C -------------------------------------------------------------------------------------------------
C ADJUST THE PARAMETERS SUITABLY IN SUBRoutines DE
C WHEN THE PROGRAM ASKS FOR PARAMETERS, FEED THEM SUITABLY
C ============== MAIN PROGRAM ==============

IMPLICIT DOUBLE PRECISION (A-H, O-Z)
CHARACTER *70 INFILE,OUTFIL
COMMON /XBASE/XBAS
COMMON /RDNMIU,IV ! RANDOM NUMBER GENERATION (IU = 4-DIGIT SEED)
COMMON /NEAREST/Z,MORDER
DIMENSION Z(10,10) ! THE INPUT CORRELATION MATRIX
INTEGER IU,IV
DIMENSION XBAS(500,50)
DIMENSION X(50) ! X IS THE DECISION VARIABLE X IN F(X) TO MINIMIZE
M IS THE DIMENSION OF THE PROBLEM, KF IS TEST FUNCTION CODE AND
FMIN IS THE MIN VALUE OF F(X) OBTAINED FROM DE
C
WRITE(*,*)'==================== CONSTRUCTION OF VALID CORRELATION MATRIX ========'
WRITE(*,*)'================ OPTIMIZATION BY DIFFERENTIAL EVOLUTION ==========
C
WRITE(*,*)'ORDER OF INPUT MATRIX (MORDER)& NAME OF INPUT FILE ?'
READ(*,*) MORDER,INFILE
WRITE(*,*) 'SPECIFY THE OUTPUT FILE TO STORE VALID OUTPUT MATRICES'
READ(*,*) OUTFIL
C
READ THE GIVEN CORRELATION MATRIX (UPPER DIAGONAL ONLY)
OPEN(7,FILE=INFILE) ! OPEN INPUT FILE AND READ THE MATRIX
DO I=1,MORDER
READ(7,*)Z(I,J),J=1,MORDER
C
WRITE(*,*)'Z(I,J),J=1,MORDER)
ENDDO
CLOSE(7)
C
WRITE(*,*)'-------------------------------- WARNING --------------------------------
WRITE(*,*)'ADJUST PARAMETERS IN SUBRoutines DE SUBROUTINE'
WRITE(*,*)'-------------------------------- WARNING --------------------------------
C
INITIALIZATION. THIS XBAS WILL BE USED IN PROGRAMS TO
INITIALIZE THE POPULATION.
WRITE(*,*)' ' WRITE(*,*)'FEED RANDOM NUMBER SEED [4-DIGIT ODD INTEGER] TO BEGIN'
READ(*,*) IU
C
THIS XBAS WILL BE USED IN ALL THE THREE METHODS AS INITIAL X
DO I=1,500
DO J=1,500
CALL RANDOM(RAND)
XBAS(I,J)=(RAND-0.5D00)*2 ! RANDOM NUMBER BETWEEN (-1, 1)
ENDDO
ENDDO
WRITE(*,*)'***********************************************
C
WRITE(*,*)'TO PROCEED TYPE ANY CHARACTER AND STRIKE ENTER'
C
READ(*,*) PROCEED
CALL DE(M,X,FMINDE) ! CALLS DE AND RETURNS OPTIMAL X AND FMIN
C
CALL NCORX(X,M,OUTFIL)
WRITE(*,*)'PROGRAM ENDED, FOR RESULTS OPEN OUTPUT FILE ',OUTFIL
WRITE(*,*)'***********************************************
END
SUBROUTINE DE(M,A,FBEST)

PROGRAM: "DIFFERENTIAL EVOLUTION ALGORITHM" OF GLOBAL OPTIMIZATION
THIS METHOD WAS PROPOSED BY R. STORN AND K. PRICE IN 1995. REF --
"DIFFERENTIAL EVOLUTION - A SIMPLE AND EFFICIENT ADAPTIVE SCHEME
FOR GLOBAL OPTIMIZATION OVER CONTINUOUS SPACES" : TECHNICAL REPORT
PROGRAM BY SK MISHRA, DEPT. OF ECONOMICS, NEHU, SHILLONG (INDIA)

PROGRAM DE

IMPLICIT DOUBLE PRECISION (A-H, O-Z) ! TYPE DECLARATION
PARAMETER(NMAX=500,MMAX=50) ! MAXIMUM DIMENSION PARAMETERS
PARAMETER(RX1=1.d0, RX2=0.d0) ! TO BE ADJUSTED SUITABLY, IF NEEDED
RX1 AND RX2 CONTROL THE SCHEME OF CROSSOVER. (0 <= RX1 <= RX2) <=1
RX1 DETERMINES THE UPPER LIMIT OF SCHEME 1 (AND LOWER LIMIT OF
SCHEME 2; RX2 IS THE UPPER LIMIT OF SCHEME 2 AND LOWER LIMIT OF
SCHEME 3. THUS RX1 = .2 AND RX2 = .8 MEANS 0-20% SCHEME1, 20 TO 80
PERCENT SCHEME 2 AND THE REST (80 TO 100 %) SCHEME 3.
PARAMETER(NCROSS=2) ! CROSS-OVER SCHEME (NCROSS <=0 OR =1 OR =>2)
PARAMETER(IPRINT=500, EPS=1.d-08) FOR WATCHING INTERMEDIATE RESULTS
IT PRINTS THE INTERMEDIATE RESULTS AFTER EACH IPRINT ITERATION AND
EPS DETERMINES ACCURACY FOR TERMINATION. IF EPS= 0, ALL ITERATIONS
WOULD BE UNDERGONE EVEN IF NO IMPROVEMENT IN RESULTS IS THERE.
ULTIMATELY "DID NOT CONVERGE" IS REPORTED.
COMMON /RNDFU/IV ! RANDOM NUMBER GENERATION (IU = 4-DIGIT SEED)
CHARACTER*70 FTIT ! TITLE OF THE FUNCTION
CHARACTER*15 CFIL ! OUTPUT FILE

THE PROGRAM REQUIRES INPUTS FROM THE USER ON THE FOLLOWING -------
(1) FUNCTION CODE (KF), (2) NO. OF VARIABLES IN THE FUNCTION (M);
(3) N=POPULATION SIZE (SUGGESTED 10 TIMES OF NO. OF VARIABLES, M,
FOR SMALLER PROBLEMS N=100 WORKS VERY WELL);
(4) PCROS = PROB. OF CROSS-OVER (SUGGESTED : ABOUT 0.85 TO .99);
(5) FACT = SCALE (SUGGESTED 0.5 TO .95 OR 1, ETC);
(6) ITER = MAXIMUM NUMBER OF ITERATIONS PERMITTED (5000 OR MORE)
(7) RANDOM NUMBER SEED (4 DIGITS INTEGER)

DIMENSION X(NMAX,MMAX),Y(NMAX,MMAX),A(MMAX),FV(NMAX)
DIMENSION IR(3),XBASE(500,50)

SELECT THE FUNCTION TO MINIMIZE AND ITS DIMENSION -------
CALL FSELECT(KF,M,FTIT)
CFIL='CORRESULTS'! IT IS AN INTERMEDIATE FILE

SPECIFY OTHER PARAMETERS ---------------------------------------------
WRITE(*,*)'POPULATION SIZE [N] AND NO. OF ITERATIONS [ITER] ?'
WRITE(*,*)'SUGGESTED: N=>100 OR =>10.M; ITERATION 500 OR LARGER'
READ(*,*) N, ITER
WRITE(*,*)'CROSSOVER PROBABILITY [PCROS] AND SCALE [FACT] ?'
WRITE(*,*)'SUGGESTED : PCROS ABOUT 0.9; FACT=.5 OR LARGER BUT <=1'
READ(*, *) PCROS, FACT
PCROS=0.9d0
FACT=0.5d0
WRITE(*,*)'RANDOM NUMBER SEED ?'
WRITE(*,*)'A FOUR-DIGIT POSITIVE ODD INTEGER, SAY, 1171'
READ(*,*) IU

NFCALL=0 ! INITIALIZE COUNTER FOR FUNCTION CALLS
GBEST=1.d30 ! TO BE USED FOR TERMINATION CRITERION
INITIALIZATION : GENERATE X(N,M) RANDOMLY
DO I=1,N
complete-cormat
DO J=1,M
C CALL RANDOM(RAND) ! GENERATES INITIATION X WITHIN
C X(I,J)=(RAND-.5D00)*2000 ! GENERATES INITIATION X WITHIN
C RANDOM NUMBERS BETWEEN -RRANGE AND +RRANGE (BOTH EXCLUSIVE)
C X(I,J)=XBASE(I,J) ! TAKES THESE NUMBERS FROM THE MAIN PROGRAM
ENDO
ENDO
WRITE(*,*)'COMPUTING --- PLEASE WAIT '
IPCOUNT=0
DO 100 ITR=1,ITER ! ITERATION BEGINS
C ---------------------------------------------------------------
C EVALUATE ALL X FOR THE GIVEN FUNCTION
DO I=1,N
DO J=1,M
A(J)=X(I,J)
ENDO
CALL FUNC(A,M,F)
C STORE FUNCTION VALUES IN FV VECTOR
FV(I)=F
ENDO
C ---------------------------------------------------------------
C FIND THE FITTEST (BEST) INDIVIDUAL AT THIS ITERATION
FBEST=FV(1)
KB=1
DO IB=2,N
IF(FV(IB).LT.FBEST) THEN
FBEST=FV(IB)
KB=IB
ENDIF
ENDDO
C BEST FITNESS VALUE = FBEST : INDIVIDUAL X(KB)
C ---------------------------------------------------------------
C GENERATE OFFSPRINGS
DO I=1,N ! I LOOP BEGINS
C INITIALIZE CHILDREN IDENTICAL TO PARENTS; THEY WILL CHANGE LATER
DO J=1,M
Y(I,J)=X(I,J)
ENDO
C SELECT RANDOMLY THREE OTHER INDIVIDUALS
20 DO IRI=1,3 ! IRI LOOP BEGINS
IR(I)=0
CALL RANDOM(RAND)
IRJ=INT(RAND*N)+1
C CHECK THAT THESE THREE INDIVIDUALS ARE DISTINCT AND OTHER THAN I
IF(IR(I).EQ.1.AND.IRJ.NE.I) THEN
IR(I)=IRJ
ENDIF
IF(IR(I).EQ.2.AND.IRJ.NE.I.AND.IRJ.NE.IR(1)) THEN
IR(I)=IRJ
ENDIF
IF(IR(I).EQ.3.AND.IRJ.NE.I.AND.IRJ.NE.IR(1).AND.IRJ.NE.IR(2)) THEN
IR(I)=IRJ
ENDIF
ENDDO ! IRI LOOP ENDS
C CHECK IF ALL THE THREE IR ARE POSITIVE (INTGERS)
DO IX=1,3
IF(IR(IX).LE.0) THEN
GOTO 20 ! IF NOT THEN REGENERATE
ENDIF
ENDDO
C THREE RANDOMLY CHOSEN INDIVIDUALS DIFFERENT FROM I AND DIFFERENT
complete-cormat

FROM EACH OTHER ARE IR(1), IR(2) AND IR(3)

RANDOMIZES NCROSS
NCROSS = 0
CALL RANDOM(RAND)
IF( RAND .GT. RX1 ) NCROSS = 1 ! IF RX1 => 1, SCHEME 2 NEVER IMPLEMENTED
IF( RAND .GT. RX2 ) NCROSS = 2 ! IF RX2 => 1, SCHEME 3 NEVER IMPLEMENTED

---------------------------------------- SCHEME 1 ----------------------------------------

NO CROSS OVER, ONLY REPLACEMENT THE IS PROBABILISTIC
IF(NCROSS .LE. 0) THEN
  DO J=1,M ! J LOOP BEGINS
  CALL RANDOM(RAND)
  IF( RAND .LE. PCROS ) THEN ! REPLACE IF RAND < PCROS
    A(J) = X(IR(1),J) + (X(IR(2),J) - X(IR(3),J)) * FACT ! CANDIDATE CHILD
  ENDIF
  ENDDO ! J LOOP ENDS
ENDIF

---------------------------------------- SCHEME 2 ----------------------------------------

THE STANDARD CROSSOVER SCHEME

CROSSOVER SCHEME (EXPONENTIAL) SUGGESTED BY KENNETH PRICE IN HIS
PERSONAL LETTER TO THE AUTHOR (DATED SEPTEMBER 29, 2006)
IF(NCROSS .EQ. 1) THEN
  CALL RANDOM(RAND)
  JR = INT(RAND^M) + 1
  J = JR
  2 A(J) = X(IR(1), J) + FACT * (X(IR(2), J) - X(IR(3), J))
  J = J + 1
  IF(J .GT. M) J = 1
  IF(J .EQ. JR) GOTO 10
  CALL RANDOM(RAND)
  IF(PCROS .LE. RAND) GOTO 2
  6 A(J) = X(I, J)
  J = J + 1
  IF(J .GT. M) J = 1
  IF(J .EQ. JR) GOTO 10
  GOTO 6
  10 CONTINUE
ENDIF

---------------------------------------- SCHEME 3 ----------------------------------------

ESPECIALLY SUITABLE TO NON-DECOMPOSABLE (NON-SEPERABLE) FUNCTIONS

CROSSOVER SCHEME (NEW) SUGGESTED BY KENNETH PRICE IN HIS
PERSONAL LETTER TO THE AUTHOR (DATED OCTOBER 18, 2006)
IF(NCROSS .GE. 2) THEN
  CALL RANDOM(RAND)
  IF(RAND .LE. PCROS) THEN
    CALL NORMAL(RN)
    DO J = 1, M
      A(J) = X(I, J) + (X(IR(1), J) + X(IR(2), J) - 2*X(I, J)) * RN
      ENDDO
    ELSE
      DO J = 1, M
        A(J) = X(I, J) + (X(IR(1), J) - X(IR(2), J)) ! FACT ASSUMED TO BE 1
      ENDDO
  ENDIF
ENDIF

CALL FUNC(A, M, F) ! EVALUATE THE OFFSPRING
IF(F .LT. FV(I)) THEN ! IF BETTER, REPLACE PARENTS BY THE CHILD
  FV(I) = F
  DO J = 1, M
    Y(I, J) = A(J)
  ENDDO
ENDDO
ENDIF
ENDDO ! I LOOP ENDS
DO I=1,N
DO J=1,M
X(I,J)=Y(I,J) ! NEW GENERATION IS A MIX OF BETTER PARENTS AND
BETTER CHILDREN
C ENDDO
IPCOUNT=IPCOUNT+1
IF(IPCOUNT.EQ.IPRINT) THEN
DO J=1,M
A(J)=X(KB,J)
ENDDO
WRITE(*,*)(X(KB,J),J=1,M)," FBEST UPTO NOW = ",FBEST
WRITE(*,*),'TOTAL NUMBER OF FUNCTION CALLS =',NFCALL
IF(DABS(FBEST-GBEST).LT.EPS) THEN
WRITE(*,*)'FITIT
WRITE(*,*)'COMPUTATION OVER'
GOTO 999
ELSE
GBEST=FBEST
ENDIF
IPCOUNT=0
ENDIF
--------------------------------------------------------------------------------
100 ENDDO ! ITERATION ENDS : GO FOR NEXT ITERATION, IF APPLICABLE
--------------------------------------------------------------------------------
WRITE(*,*)'DID NOT CONVERGE. REDUCE EPS OR RAISE ITER OR DO BOTH'
WRITE(*,*)'INCREASE N, PCROS, OR SCALE FACTOR (FACT)'
999 OPEN(7,FILE=CFIL)
WRITE(7,*) N
DO I=1,N
WRITE(7,*) (X(I,J),J=1,M),FV(I)
ENDDO
CLOSE(7)
RETURN
END
--------------------------------------------------------------------------------
SUBROUTINE NORMAL(R)
C PROGRAM TO GENERATE N(0,1) FROM RECTANGULAR RANDOM NUMBERS
C IT USES BOX-MULLER VARIEATE TRANSFORMATION FOR THIS PURPOSE.
--------------------------------------------------------------------------------
C ----- BOX-MULLER METHOD BY GEP BOX AND ME MULLER (1958) ------
C BOX, G. E. P. AND MULLER, M. E. "A NOTE ON THE GENERATION OF
C IF U1 AND U2 ARE UNIFORMLY DISTRIBUTED RANDOM NUMBERS (0,1),
C THEN X=[(-2*LN(U1))**0.5]*(COS(2*PI*U2)) IS N(0,1)
C ALSO, X=[(-2*LN(U1))**0.5]*(SIN(2*PI*U2)) IS N(0,1)
C PI = 4*ARCTAN(1.0)= 3.1415926535897932384626433832795
C 2*PI = 6.283185307179586476925286766559
C --------------------------------------------------------------------------------
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
COMMON /RNDM/IU,IV
INTEGER IU,IV
C -------------------------------------------------------------------------------
CALL RANDOM(RAND) ! INVOKES RANDOM TO GENERATE UNIFORM RAND [0, 1]
U1=RAND ! U1 IS UNIFORMLY DISTRIBUTED [0, 1]
CALL RANDOM(RAND) ! INVOKES RANDOM TO GENERATE UNIFORM RAND [0, 1]
U2=RAND ! U1 IS UNIFORMLY DISTRIBUTED [0, 1]
R=DQRT(-2.0*DLOG(U1))
R=R*DOSCOS(U2*6.283185307179586476925286766559D00)
C R=R*DOSCOS(U2*6.28318530718D00)
complete-cormat

RETURN
END

C RANDOM NUMBER GENERATOR (UNIFORM BETWEEN 0 AND 1 - BOTH EXCLUSIVE)
SUBROUTINE RANDOM RAND1
DOUBLE PRECISION RAND1
COMMON /RNDM/IU,IV
IV=IU*65539
IF(IV.LT.0) THEN
IV=IV+2147483647+1
ENDIF
RAND=IV
IU=IV
RAND=RAND*0.4656613E-09
RAND1= dble(RAND)
RETURN
END

C ---------------------------------------------------------------------

SUBROUTINE FSELECT(KF,M,FTIT)
PARAMETER (NFUNCT=1) ! NO. OF FUNCTIONS IN THE LIST
C THE PROGRAM REQUIRE INPUTS FROM THE USER ON THE FOLLOWING -------
C (1) FUNCTION CODE (KF), (2) NO. OF VARIABLES IN THE FUNCTION (M);
C CHARACTER "70 TIT(100),FTIT
C WRITE(*,'')!'------------------------------------------------------'
KF=1 ! NO. OF FUNCTIONS
DATA TIT(1)/KF=1 CONSTRUCT CORRELATION MATRIX: M-VARIABLES M=?/
---------------------------------------------------------------
C DO I=1,NFUNCT
C WRITE(*,'')TIT(I)
C ENDDO
C WRITE(*,'')!******************************************************
C WRITE(*,'')!NO. OF VARIABLES=UNKNOWN CORRELATION COEFFICIENTS [M]?
C READ(*,* M
FTIT=TIT(KF) ! STORE THE NAME OF THE CHOSEN FUNCTION IN FTIT
RETURN
END

C ---------------------------------------------------------------------

SUBROUTINE FUNC(X,M,F)
C TEST FUNCTIONS FOR GLOBAL OPTIMIZATION PROGRAM
C IMPLICIT DOUBLE PRECISION (A-H,0-Z)
COMMON /NFcall/ NFcall
DIMENSION X(*)
NFcall=NFcall+1 ! INCREMENT TO NUMBER OF FUNCTION CALLS
C KF IS THE CODE OF THE TEST FUNCTION
C
C CONSTRUCT CORRELATION MATRIX
C CALL CONCOR(M,X,F)
RETURN
C
C=======================================================================
END

C SUBROUTINE EIGEN(A,N,V,W)
C COMPUTES EIGENVALUES AND VECTORS OF A REAL SYMMETRIC MATRIX
C A(N,N) = GIVEN REAL SYMMETRIC MATRIX WHOSE EIGENVALUES AND VECTORS
C ARE BE FOUND. ITS ORDER IS N X N
C W(N,N) CONTAINS EIGENVALUES IN ITS MAIN DIAGONAL. OTHER ELEMENTS=0
C V(N,N) CONTAINS EIGENVECTORS
C PROGRAM BY KRISNAMURTHY, EV & SEN (1976) COMPUTER-BASED NUMERICAL
C ALGORITHMS. AFFILIATED EAST-WEST PRESS, NEW DELHI
C
DOUBLE PRECISION A(10,10),V(10,10),W(10,10),P(10)
DOUBLE PRECISION Pmax, EPLN, TAN, SIN, COS, AI, TT, TA, TB

Page 6
DIMENSION MM(10)

C ------------------ INITIALISATION ----------------------------

DO I=1,N
DO J=1,N
V(I,J)=0.d0
W(I,J)=A(I,J)
ENDDO
P(I)=0.d0
ENDDO
PMAX=0.d0
EPLN=0.d0
TAN=0.d0
SIN=0.d0
COS=0.d0
AI=0.d0
TT=0
NN=1
EPLN=1.0d-100

C -------------------------------------------------------------

IF(NN.NE.0) THEN
DO I=1,N
DO J=1,N
V(I,J)=0.d0
IF(I.EQ.J) V(I,J)=1.d0
ENDDO
ENDDO
ENDIF
NR=0
MI=N-1
DO I=1,MI
P(I)=0.d0
M2=I+1
DO J=M1,J,1
IF(P(I).LE.DABS(A(I,J))) THEN
P(I)=DABS(A(I,J))
MM(I)=J
ENDIF
ENDDO
ENDDO
ENDDO

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

C EPLN=DABS(PMAX)*1.0D-09
IF (PMAX.LE.EPLN) THEN
C WRITE(*,*),'PMAX EPLN',PMAX, EPLN
C PAUSE 'CONVERGENCE CRITERION IS MET'
GO TO 12
ENDIF
NR[N]1
TA=2.d0*A(IP,JP)
TB=(DABS(A(IP,IP)-A(JP,JP))+
TAN=TA/TB
COS=1.d0/DSQRT(1.d0+TAN**2)
SIN=TAN*COS
AI=A(IP,JP)
A(IP,IP)=(COS**2)*(AI+TAN*(2.d0*A(IP,JP)+TAN*A(JP,JP)))
**complete-cormat**

\[ A(JP, JP) = (\cos^{*}2)(A(JP, JP) - \tan^{*}(2.d0* A(IP, JP) - \tan^{*} A)) \]

\[ A(IP, JP) = 0.0 \]

IF(A(IP, IP).GE.A(JP, JP)) GO TO 15

\[ T = A(IP, IP) \]

A(IP, IP) = A(JP, JP)

A(JP, JP) = T

IF(SIN.GE.0.0 d0) GO TO 16

\[ T = \cos \]

GO TO 17

\[ T = -\cos \]

COS = DABS(SIN)

\[ \sin = T \]

DO 18 I = 1, M

IF(I - IP) 19, 18, 20

IF(I.EQ.IP) GO TO 18

IF(MM(I).EQ.IP) GO TO 21

IF(MM(I).NE.IP) GO TO 18

K = MM(I)

\[ T = A(I, K) \]

A(I, K) = 0.0

M(J) = I + 1

P(I) = 0.0 d0

DO 22 J = M(J), N

IF(P(I).GT.DABS(A(I, J))) GO TO 22

P(I) = DABS(A(I, J))

MM(I) = J

CONTINUE

\[ A(I, K) = T \]

CONTINUE

P(IP) = 0.0 d0

P(JP) = 0.0 d0

DO 23 I = 1, N

IF(I - IP) 24, 23, 25

\[ T = A(I, IP) \]

A(I, IP) = \cos^{*}T + \sin^{*}A(I, JP)

IF(P(I).GE.DABS(A(I, IP))) GO TO 26

P(I) = DABS(A(I, IP))

\[ \text{MM}(I) = I \]

A(I, JP) = \sin^{*}T + \cos^{*}A(I, JP)

IF(P(I).GE.DABS(A(I, JP))) GO TO 23

\[ P(I) = DABS(A(I, JP)) \]

\[ \text{MM}(I) = JP \]

GO TO 23

IF(I.LT.JP) GO TO 27

IF(I.GT.JP) GO TO 28

IF(I.EQ.JP) GO TO 23

\[ T = A(IP, I) \]

A(IP, I) = \cos^{*}T + \sin^{*}A(IP, I)

IF(P(IP).GE.DABS(A(IP, I))) GO TO 29

P(IP) = DABS(A(IP, I))

\[ \text{MM}(I) = I \]

A(IP, I) = -T* \sin + \cos^{*}A(IP, I)

IF(P(IP).GE.DABS(A(IP, I))) GO TO 23

GO TO 30

\[ T = A(IP, I) \]

A(IP, I) = T* \cos + \sin^{*}A(IP, I)

IF(P(IP).GE.DABS(A(IP, I))) GO TO 31

P(IP) = DABS(A(IP, I))

\[ \text{MM}(I) = I \]

A(IP, I) = -T* \sin + \cos^{*}A(IP, I)

IF(P(IP).GE.DABS(A(IP, I))) GO TO 23

P(IP) = DABS(A(IP, I))

\[ \text{MM}(IP) = I \]

Page 8
23 CONTINUE
   IF(NN.EQ.0) GOTO 7
   DO 32 I=1,N
      TT=V(I,IP)
      V(I,IP)=TT*COS+SIGN(V(I,JP))
      V(I,JP)=-TT*SIN+COS*V(I,JP)
   32 CONTINUE
   GO TO 7
   DO I=1,N
      P(I)=A(I,I)
   ENDDO
   DO I=1,N
      DO J=1,N
         A(I,J)=W(I,J)
         WI,J=0.D0
      ENDDO
      WI,I=P(I)
   ENDDO
   RETURN
END

---

C SUBROUTINE CONCOR(M,X,F)
C CONSTRUCTING VALID CORRELATION MATRICES
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C COMMON /NEAREST/Z,MORDER
C COMMON /RNDM/IU,IV
C DIMENSION Z(10,10),A(10,10)
C DIMENSION X(*) ,V(10,10),W(10,10),P(10)
C
C CHECK THE NUMBER OF INVALID ELEMENTS
C MINVAL=0
   DO I=1,MORDER
      DO J=I,MORDER
         IF(DABS(Z(I,J)).GT.1.D0) MINVAL=MINVAL+1
      ENDDO
   ENDDO
   IF(M.NE.MINVAL) THEN
      WRITE(*,*)'??????????????????????????????????????????????'
      WRITE(*,*)'THE VALUE OF M SHOULD BE ',MINVAL
      WRITE(*,*)'RE-RUN THE PROGRAM WITH M = ',MINVAL
      STOP
   ENDIF
C
   DO I=1,M
      IF(X(I).LT.-1.D0 OR. X(I).GT.1.D0) THEN
         CALL RANDOM(RAND)
         X(I)=(RAND-.5D0)*2
      ENDIF
   ENDDO
C
   CONSTRUCT THE MATRIX(MM,MM)
   ICOUNT=0
   DO I=1,MORDER
      A(I,I)=1.D0
      DO J=I+1,MORDER
         IF(DABS(Z(I,J)).GT.1.D0) THEN
            ICOUNT=ICOUNT+1
            A(I,J)=X(ICOUNT)
         ELSE
            A(I,J)=0.D0
         END IF
      ENDDO
   ENDDO
   RETURN
END
A(I,J)=Z(I,J)
ENDIF
ENDDO
C FILLING THE LOWER DIAGONAL
DO I=1,MORDER
DO J=1,I
A(I,J)=A(J,I)
ENDDO
ENDDO
C FIND EIGENVALUES AND EIGENVECTORS OF MATRIX A
CALL EIGEN(A,MORDER,V,W)
C STORE EIGENVALUES (DIAGONAL OF RETURNING W) INTO P
F=0.D0
PSUM=0.D0 ! SUM OF MAGNITUDE OF EIGENVALUES
DO I=1,MORDER
P(I)=W(I,I)
IF(P(I).LT.0.D0) PSUM=PSUM+DABS(P(I))
ENDDO
PROD=1.D0
DO I=1,MORDER
IF(P(I).LT.0.D0) THEN
F=F+P(I)**2
PROD=PROD*P(I)
ENDIF
ENDDO
IF(PROD.LT.0.D0.OR.PROD.GT.1.D0) F=(F+PSUM+PROD**2)**2
RETURN
END
C SUBROUTINE NCORX(X,M,OUTFIL)
C NEAREST CORRELATION MATRIX PROBLEM
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
COMMON /NEAREST/Z,MORDER
DIMENSION Z(10,10),X("")
DIMENSION A(10,10),V(10,10),W(10,10)
CHARACTER *70 OUTFIL
CHARACTER *15 CFIL
C CFIL='CORRESULTS!' IT IS AN INTERMEDIATE FILE
C OPEN (7, FILE=CFIL) ! OPENS INTERMEDIATE FILE FOR INPUT
OPEN(8, FILE=OUTFIL) ! OPENS OUTPUT FILE TO STORE VALID MATRICES
C CONSTRUCT THE CORRELATION MATRIX
READ(7,"(I0)"
ICOUNT=0
DO I=1,MORDER
A(I,1)=1.D0
DO J=I+1,MORDER
IF(DABS(Z(I,J)).GT.1.D0) THEN
ICOUNT=ICOUNT+1
A(I,J)=X(ICOUNT)
ELSE
A(I,J)=Z(I,J)
ENDIF
ENDDO
ENDDO
C FILLING THE LOWER DIAGONAL
DO I=1,MORDER
DO J=1,I

Page 10
A(I,J)=A(J,I)
ENDDO
ENDDO
WRITE(*,*) ' '  
WRITE(8,*) '******************************************************************************'
WRITE(8,*) 'A VALID CORRELATION MATRIX'
DO I=1,MORDER
WRITE(8,1)(A(I,J),J=1,MORDER)
ENDDO
WRITE(*,*) ' '  
CALL EIGEN(A,MORDER,V,W)
MSIGN=0
SUMW=0.D0  
PROD=1.D0
DO I=1,MORDER
SUMW=SUMW+W(I,I)
PROD=PROD*W(I,I)
IF(W(I,I).LT.0) MSIGN=1
ENDDO
WRITE(8,*) 'EIGENVALUES, SUM AND PRODUCT OF EIGENVALUES'
WRITE(8,1)(W(I,I),I=1,MORDER),SUMW,PROD
WRITE(8,*) 'EIGENVECTORS '
DO I=1,MORDER
WRITE(8,1)(V(I,J),J=1,MORDER)
ENDDO
1 FORMAT(8F10.7)
IF(MSIGN.EQ.1) THEN
WRITE(8,*) 'FAILURE OF THE METHOD'
ELSE
WRITE(8,*) 'SUCCESS OF THE METHOD'
ENDIF
ENDDO
CLOSE(7)
CLOSE(8)
RETURN
END