On generating correlated random variables with a given valid or invalid Correlation matrix

SK Mishra

North-Eastern Hill University, Shillong (India)

2. August 2004

On generating Correlated Random Variables with a given Valid or Invalid Correlation Matrix

SK Mishra
Dept. of Economics
NEHU, Shillong, India

1. Introduction

The objective of this paper is to provide an algorithm that generates real $X(n,m)$ with a desired feasible intercorrelation matrix, $R(m,m)$, where n stands for the number of observations (or sample size) and $m < n$ is the number of variates (or variables). In simulation-oriented works, it is often required to generate a matrix of sample variates, $X(n,m)$, that characterizes a desired (feasible) intercorrelation matrix, $R(m,m)$. If each column (variate) of $X(n,m)$ has zero mean and unit standard deviation then the intercorrelation matrix $R(m,m) = n^{-1}X'X$. It may well be viewed as the dispersion matrix of the standardized variables. One may begin with a valid intercorrelation matrix, R (positive semidefinite), or an invalid matrix, Q (negative definite), whose main diagonal elements are unity and the rest of the elements are between –1 and 1.

Being the quadratic form (see Theil, 1971, pp. 22-29), a valid *product moment* intercorrelation matrix, R, is necessarily a positive semidefinite matrix. All the successive principal minors of R are non-negative (see Takayama, 1974, pp. 118-121, pp. 383-385) or stated differently all the eigenvalues of R are non-negative. Each element $r_{ij} \in R$ is the cosine of angle θ_{ij} between the vectors x_i *and* x_j. In practice, however, no x_i is a linear combination of x_j ; $i \neq j$; $j = 1,2,...,m$ (that is, $x_i = \sum_{j=1;\, j\neq i}^{m} x_j a_j$ is *not true* for any non-null real vector $a = (a_1 \;\; a_2 \;\; ... \;\; a_m)'$). In case, one requires such a vector, it can be obtained by a linear combination such as $x_i = \sum_{j=1;\, j\neq i}^{m} x_j a_j$. This case being very specific and trivial (and so set apart in practice), one requires to generate R, which is a positive definite matrix.

2. Methods to generate random numbers

First, let us look into the procedure that may be used to generate a single variate. Generally, the exercise begins with the uniformly distributed random numbers generated by some procedure such as the *power residue method* or the *mid-square-bit method* (see Krishnamurthy and Sen, 1976, pp. 302-304). Uniformly distributed random numbers may be transformed into $x \sim$ N(0,1); $x = \sqrt{\{-2\ln(u_1)\}}\{Cos(2\pi u_2)\}$ where u_1 and u_2 are uniformly distributed independent random numbers lying between (0,1) and x is the standard normal variate (see Knuth (1969), Texas Instruments Inc (1979), p. 54). Alternatively, one may generate N(0, 1) from uniformly distributed $U(0, 1)$ numbers, by using the Central Limit Theorem (see Gillett (1979, p. 519). However, this method is less accurate and time consuming than Knuth's method. Normally distributed variate, x, may be used to generate Gamma distributed variate, g, since, if x is a standard normal variate,

then g $= x^2/2$ is a Gamma variate with parameter $1/2$. Due to the additive property of Gamma variates, if x_i (i=1,2,…,n) are n independent normal variates with means m_i and standard deviations σ_i then g = $\frac{1}{2}\sum_{i=1}^{n}\frac{(x_i - m_i)^2}{\sigma_i^2}$ is a Gamma variate with parameter $2^{-1}n$.

From two independent normally distributed variates x_1 and x_2 we may obtain a Cauchy distributed variate, since, if x_1 and x_2 are independent normal variates with means m_1 and m_2 and variances σ_1^2 and σ_2^2 then the variate $c = (x_1 - m_1)/(x_2 - m_2)$ is Cauchy distributed. In particular, the quotient of two independent standard normal variates is Cauchy distributed. From two independent Gamma variates, g_1 and g_2 with parameters l and m respectively, we may obtain $v_1 = g_1/(g_1 + g_2)$, which is a $\beta_1(l,m)$ distributed variate, and $v_2 = g_1/g_2$, which is a $\beta_2(l,m)$ distributed variate. For these relations see Kapur and Saxena, 1982, pp. 292, 386, 288-289 and 427. In general, starting from uniformly distributed variates, we may obtain a variate with almost any kind of distribution by a sequence of suitable transformations.

Generation of multivariate distributions with desired parameters began with Hoffman (1924) who proposed a method to generate two variables that satisfy a given bivariate correlation (coefficient). However, his method cannot be applied to generate $m > 2$ variables that satisfy a given correlation matrix. Kaiser and Dichman (1962) generalized Hoffman's method for $m \geq 2$ variables. The Kaiser-Dichman method is based on factorization of $R(m,m)$. It presumes that $R(m,m)$ is a positive definite matrix (that has all its eigenvalues positive). Moreover, it generates variables that have a multinormal distribution.

Fleishman (1978) introduced an algorithm to generate normal or non-normal random numbers satisfying the first four moments (mean, variance, skewness and kurtosis). His method does not depend on factorization of the desired R matrix. Tadikamalla (1980) proposed several methods to generate non-normally distributed random numbers.

Vale and Maurelli (1983) proposed a method for generating multivariate (normally as well as non-normally distributed) variates with desired first four moments of each variate satisfying the specified intercorrelation matrix. Headrick and Sawilowski (1999) introduced a method that generates multivariate non-normal distributions with average values of intercorrelations approximating the population intercorrelations. This method, unlike that of Vale and Maurelli, performs well even if the distributions are heavily skewed or thick tailed. Moreover, being based on Fleishman's procedure, it does not require factorization of $R(m,m)$.

3. The case of negative definite invalid intercorrelation matrices

It may be noted that arbitrary *real* symmetric matrices, say Q, that have elements $q_{ii} = 1 \ \forall \ i = 1,2,...,m$ and $-1 \leq q_{ij} \leq 1 \ \forall \ i, j = 1,2,...,m$; $i \neq j$ are *not* the genuine product moment intercorrelation matrices, R, obtainable from some *real* X although they

may appear to be so. For example, the following three matrices appear to be genuine (product moment) intercorrelation matrices while they are not.

$$Q_1 = \begin{bmatrix} 1.00 & 0.70 & 0.00 \\ 0.70 & 1.00 & 0.80 \\ 0.00 & 0.80 & 1.00 \end{bmatrix};\ Q_2 = \begin{bmatrix} 1.00 & 0.90 & 0.10 \\ 0.90 & 1.00 & 0.80 \\ 0.10 & 0.80 & 1.00 \end{bmatrix};\ Q_3 = \begin{bmatrix} 1.00 & 0.60 & 0.13 \\ 0.60 & 1.00 & 0.90 \\ 0.13 & 0.90 & 1.00 \end{bmatrix}$$

$\mathrm{Det}(Q_1) = -0.13$, $\mathrm{Det}(Q_2) = -0.316$ and $\mathrm{Det}(Q_3) = -0.0465$. One of the eigenvalues of each matrix is negative. Several such examples may be generated. We will name such matrices as the invalid or pseudo intercorrelation matrices or Q matrices against the R matrices that are necessarily positive semidefinite.

Negative definite or pseudo intercorrelation matrices may enter into empirical investigation due to several reasons. First, the coefficients of correlation may not be computed by the Karl Pearson's (product moment) formula. They might have been obtained by Spearman's formula (of rank correlation) or they could be the polychoric coefficients of correlation. Secondly, some of them might have been computed from variables different in sample size (observations). Suppose $Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}$ such that Q_{11} is obtained from $X_1(n_1, m_1)$, Q_{22} is obtained from $X_2(n_2, m_2)$: $n_1 > n_2$, and $Q_{12} = Q_{21}'$ is obtained from $[X_1(n_2, m_1), X_2(n_2, m_2)]$, while $X = \left[X_1 \middle| \begin{matrix} X_2 \\ \varnothing \end{matrix} \right]$, $\varnothing$ standing for '*information not available*'. Then Q could fail to be positive semidefinite. Thirdly, when the off diagonal entries in Q are large (say 0.9 or still larger) in magnitude, but recorded with substantial error or approximation, Q may fail to be positive semidefinite. Fourthly, when the elements of near-singular matrices are rounded off (for reporting in research papers, etc.) without a due care taken to the possible effects of rounding off on the status of the matrices regarding the properties such as positive definiteness etc, the reported matrices may lose the properties that they originally have had. A telling example of this is the positive semidefinite matrix obtained by Higham (see Higham, 2002, p. 335 : the matrix was singular in the original). However, the reported matrix (rounded off at the fourth place after decimal) has its determinant = -2.441038E-05 (one of the eigenvalues being –1.343337484E-05, instead of zero). Surely, a negative value of the determinant is due to rounding off. Lastly, in simulation, especially when Q is an initial approximation to R large in dimension, the analyst has to arbitrarily fill in the values of q_{ij} ; $i \neq j\ \forall\ i, j = 1, 2, ..., m$. The only restraint obeyed by the analyst is that $q_{ii} = 1$ and $-1 \leq q_{ij} = q_{ji} \leq 1\ \forall\ i, j = 1, 2, ..., m.$ Such arbitrary Q may often fail to be positive semidefinite.

It is required, therefore, to obtain best possible R (positive semidefinite matrix) from Q (not positive semidefinite). Higham (2002) proposed a method to obtain $\hat{R}$ from Q such that $\left\| Q - \hat{R} \right\|_F$ is the least. Here $\| \ . \ \|_F$ is the Frobenius norm. The method is very general

and allows for weights to be assigned to different elements of the distance matrix as desired by the analyst according to the level of confidence put in to the accuracy or (rationally justified) most probable value of q_{ij}. In that case, the weighted norm of difference is minimized. However, for larger matrices, the method is time consuming due to the linear convergence of the algorithm used by Higham. Once $\hat{R}$ (the nearmost R to Q) is obtained, one may go in for obtaining the variates that satisfy the $\hat{R}$ matrix.

Al-Subaihi (2004) proposed a modification of Kaiser-Dichman procedure to generate normally distributed (correlated) variates from a given non-positive definite Q, which, in the process, is approximated by a positive definite R^* matrix. The resulting variates satisfy the R^* matrix. It appears that Al-Subaihi's method does not guarantee that R^* is sufficiently close to Q as in the Higham procedure.

We take an example from Al-Subaihi (2004, p. 11, the middle matrix). The values of $q_{ii} = 1 \ \forall \ i = 1,2,3,4,5.$ The value of $q_{12} = q_{21} = q_{13} = q_{31} = 0.5.$ Other elements in the first row (as well as the first column) are all zero. The values of the off-diagonal elements $q_{ij} = q_{ji} = 0.84$ for $i, j = 2,3,4,5 \ ; \ i \neq j.$

Al-Subaihi generated the first matrix (call it R^*, given below) as an approximation to Q, while we have simply perturbed R^* to obtain R^{**}. We find that the second matrix, R^{**}, approximates Q more accurately than the first matrix, R^*, generated by Al-Subaihi. Note that neither of the two matrices (R^* and R^{**}) is optimally nearmost to the given Q matrix.

	Al-Subaihi's generated R^* matrix					A relatively better R^{**} matrix				
	x_1	x_2	x_3	x_4	x_5	x_1	x_2	x_3	x_4	x_5
x_1	1.0000	0.4964	0.5008	**0.0011**	**0.0050**	1.0000	0.4964	0.5008	**0.0007**	**0.0010**
x_2	0.4964	1.0000	0.8819	0.7317	**0.7363**	0.4964	1.0000	0.8819	0.7317	**0.8400**
x_3	0.5008	0.8819	1.0000	0.7272	**0.7305**	0.5008	0.8819	1.0000	0.7272	**0.8200**
x_4	**0.0011**	0.7317	0.7272	1.0000	**0.8432**	**0.0007**	0.7317	0.7272	1.0000	**0.8400**
x_5	**0.0050**	**0.7363**	**0.7305**	**0.8432**	1.0000	**0.0010**	**0.8400**	**0.8200**	**0.8400**	1.0000

4. Proximity measured by the maximum norm of deviations

It is clear that instead of minimizing the Frobenius norm, one may opt for minimizing the maximum norm such that the $\max_{i,j} \left| q_{ij} - \hat{r}_{ij} \right|$ is minimum. This line of investigation may be useful since the minimization of the maximum norm allows for the least substitutability among the off-diagonal elements of the distance matrix $\Delta \ : \ \delta_{ij} \in \Delta \ ; \ \delta_{ij} = \left| q_{ij} - \hat{r}_{ij} \right| \ \forall \ i, j.$ We accomplish this task here and for the sake of comparison present some results. As an exercise we first take a matrix from Higham's (2002) paper. The results are as follows.

	Higham's original matrix			Higham's $\hat{R}_F$ matrix			Min(max norm) $\hat{R}_m$ matrix		
	x_1	x_2	x_3	x_1	x_2	x_3	x_1	x_2	x_3
x_1	1.0000	1.0000	0.0000	1.00000	0.76069	0.15731	1.00000	0.78077	0.21922
x_2	1.0000	1.0000	1.0000	0.76069	1.00000	0.76069	0.78077	1.00000	0.78077
x_3	0.0000	1.0000	1.0000	0.15731	0.76069	1.00000	0.21922	0.78077	1.00000

The $\max_{i,j}(\delta_{ij}) = \max_{i,j}\left|q_{ij} - \hat{r}_{Fij}\right|$ produced by Higham's estimated $\hat{R}_F$ is 0.23931 and $\sum_{i,j}\delta_{ij}$ is 1.27186. On the other hand, the $\max_{i,j}(\delta_{ij}) = \max_{i,j}\left|q_{ij} - \hat{r}_{mij}\right|$ produced by min(max norm) estimated $\hat{R}_m$ is 0.21922 and $\sum_{i,j}\delta_{ij}$ is 1.31536.

The determinants of the three matrices are : -1.0, 9.6469465821302971000096417818E-06 and 1.2815879075985762762989893198 11E-05 approximately. The eigenvalues of the three matrices are given below.

Eigenvalues of Higham's original matrix, his estimated $\hat{R}_F$ matrix and Min(max norm) $\hat{R}_m$ matrix			
Eigenvalues	λ_1	λ_2	λ_3
Higham's original matrix	2.4142135623731	1.0000	-4.1421356237309E-01
Higham's estimated $\hat{R}$ matrix	2.1573046934710	8.4269E-01	5.3065290382026E-06
Min(max norm) $\hat{R}$ matrix	2.2192126035928	0.78078	7.3964071641482E-06
Note: Higham's estimated matrix (see Higham, 2002, p. 335) has turned negative definite. We perturbed it slightly on the fifth place after decimal to make it a positive definite matrix.			

Then we take a matrix from Al-Subaihi's (2004) paper. The values of $q_{ii} = 1 \ \forall \ i = 1,2,3,4,5$. The value of $q_{12} = q_{21} = q_{13} = q_{31} = 0.5$. Other elements in the first row (as well as the first column) are all zero. The values of the off-diagonal elements $q_{ij} = q_{ji} = 0.84$ for $i, j = 2,3,4,5$; $i \neq j$. The results are presented below. The first of the two matrices presented below is obtained by Al-Subaihi, while the second is obtained by us by minimizing the maximum norm of $\hat{\Delta}$.

	Al-Subaihi's generated R^* matrix					Ours min(max norm) $\hat{R}_m$ matrix				
	x_1	x_2	x_3	x_4	x_5	x_1	x_2	x_3	x_4	x_5
x_1	1.0000	0.4964	0.5008	0.0011	0.0050	1.000000	0.477630	0.477630	0.018118	0.018118
x_2	0.4964	1.0000	0.8819	0.7317	0.7363	0.477630	1.000000	0.862370	0.817630	0.817630
x_3	0.5008	0.8819	1.0000	0.7272	0.7305	0.477630	0.862370	1.000000	0.817630	0.817630
x_4	0.0011	0.7317	0.7272	1.0000	0.8432	0.018118	0.817630	0.817630	1.000000	0.862370
x_5	0.0050	0.7363	0.7305	0.8432	1.0000	0.018118	0.817630	0.817630	0.862370	1.000000

The Erhardt-Schmidt (or Frobenius) norm $\left\|\Delta^*\right\|_F$ of Δ^* : $\delta_{ij}^* \in \Delta^*$; $\delta_{ij}^* = \left|q_{ij} - r_{ij}^*\right|$ $\forall$ i, j, where $r_{ij}^* \in R^*$ (Al-Subaihi's generated positive semidefinite matrix) and $q_{ij} \in Q$ (the negative definite matrix from which R^* is generated) is 0.313057 against 0.096539, which is the $\left\|\hat{\Delta}\right\|_F$ of $\hat{\Delta}$: $\hat{\delta}_{ij} \in \hat{\Delta}$; $\hat{\delta}_{ij} = \left|q_{ij} - \hat{r}_{ij}\right|$ $\forall$ i, j, while $\hat{r}_{ij} \in \hat{R}$, the positive semidefinite matrix nearmost to Q in the min(max norm) sense. The corresponding maximum norms $\left\|\Delta^*\right\|_m$ and $\left\|\hat{\Delta}\right\|_m$ are 0.564 and 0.11185 respectively.

Δ matrix from Al-Subaihi's R^* matrix					Δ matrix from min(max norm) $\hat{R}_m$ matrix				
0	0.0036	0.0008	0.0011	0.0050	0	0.022370	0.022370	0.018118	0.018118
0.0036	0	0.0419	0.1083	0.1037	0.022370	0	0.022370	0.022370	0.022370
0.0008	0.0419	0	0.1128	0.1095	0.022370	0.022370	0	0.022370	0.022370
0.0011	0.1083	0.1128	0	0.0032	0.018118	0.022370	0.022370	0	0.022370
0.0050	0.1037	0.1095	0.0032	0	0.018118	0.022370	0.022370	0.022370	0

The $\max_{i,j}(\delta_{ij}) = \max_{i,j}\left|q_{ij} - r_{ij}^*\right|$ produced by Al-Subaihi's R^* is 0.1128 and $\sum_{i,j}\delta_{ij}$ is 0.9798. The $\max_{i,j}(\delta_{ij}) = \max_{i,j}\left|q_{ij} - \hat{r}_{mij}\right|$ produced by $\hat{R}_m$ is 0.02237 and $\sum_{i,j}\delta_{ij}$ is 0.430392. Thus, $\hat{R}_m$ is an undubitably better approximation than R^*. This shows that the R^* matrix generated from Q by Al-Subaihi is only sub-optimally close to Q.

Thus we have two alternative methods to obtain the nearmost positive semidefinite matrices from the given negative definite matrix, Q, the one proposed by Higham that minimizes $\left\|\hat{\Delta}\right\|_F$ and the other proposed by us in this paper that minimizes $\left\|\hat{\Delta}\right\|_m$. Use of either norm has its own justification. The min(max norm) does not allow any element $\hat{r}_{ij\ \ i\neq j} \in \hat{R}$ to deviate too much from its corresponding q_{ij}, while the min(Frobenius norm) may permit excessive deviation of a few elements if so required to bring other element of $\hat{R}$ closer to their counterpart elements (of Q). However, to disallow any element $\hat{r}_{ij\ \ i\neq j} \in \hat{R}$ to deviate too much from its corresponding q_{ij} amounts to place a high degree of confidence on the elements of Q.

5. The Algorithms

I. The first algorithm that generates the nearmost positive definite intercorrelation matrix from a given (fed by the user) negative definite invalid symmetric intercorrelation matrix, Q, runs as follows:

1. Let Q_0 be the given invalid intercorrelation matrix. Set $Q = Q_0$.
2. Find all eigenvalues (L) and and eigenvectors (V) from Q. Each column of V has unit Euclidean length.
3. Replace all negative values in L (a diagonal matrix) by zero.

4. Generate m uniformly distributed random numbers $U(0,1)$ and add them to the diagonal elements of L matrix. Normalize L such that its trace is equal to m.
5. By random walk method of optimization find the best possible L that characterizes trace $= m$, positive determinant and therefore positive definite $\hat{R} = V'LV$ closest to Q_0. Closeness is defined in terms of the maximum norm $\left\|\hat{\Delta}\right\|_m = \left\|Q_0 - \hat{R}\right\|_m$.
6. Check if all $\hat{r}_{ii} \in \hat{R}$ are approximately unity. It would depend on tolerance level chosen. If not, replace them by unity. Consider it as Q and go to step 2, else stop.

II. The second algorithm that generates $X(n,m)$ from a valid (positive definite) inter-correlation matrix runs in the following steps:

1. Generate $Y(n,m)$ from a random number generator that yields $Y \sim U(0,1)$.
2. Standardize Y such that its each column has zero mean and unit standard deviation. Call this standardized Y by a new name, say Y^*.
3. Compute intercorrelation matrix S from Y^*.
4. Compute all eigenvalues (D) and the associated eigenvectors (V) of S. Here D is a diagonal matrix and V is an orthogonal matrix. Moreover, each column of V has a unit length (Euclidean norm).
5. Compute $Z = (Y^*)V$. Now $Z(n,m)$ is column-wise orthogonal.
6. Standardize $Z(n,m)$ such that each one of its columns has zero mean and unit standard deviation. This $Z(n,m)$ will be used at step 10.
7. Choose an intercorrelation matrix, $R(m,m)$. This is the intercorrelation matrix that is induced into Z. In choosing R one must be cautious to see that it should not violate the properties of an intercorrelation matrix described earlier. None of its eigenvalues should be negative. This is done in the next step.
8. Compute all eigenvalues (say, L) of R and the associated vectors (say E). If any of the eigenvalues are negative, change the R matrix since no intercorrelation matrix, by necessity, can have negative eigenvalues (if X is real). In that case, go to step 7.
9. Standardize E to obtain W such that each of its column has a squared (Euclidean) norm equal to the eigenvalue associated with it. Let $k_j = \{(\sum_{i=1}^{m} e_{ij}^2)/L_j\}^{1/2}$ then $w_{ij} = e_{ij}/k_j \;\; ; \; i, j = 1,2,...,m.$ This guarantees that $\sum_{i=1}^{m} w_{ij}^2 = L_j \;\; \forall \;\; j = 1,2,...,m.$
10. Compute $X = ZW$.
11. Standardize X such that each of its column has zero mean and unit standard deviation.

6. FORTRAN Computer Programs

We provide here the *source codes* of the computer programs that implement the algorithms given above. The first main program ($PROG_1$) checks if the Q matrix fed by the user is not a negative definite matrix. If Q is not a positive definite matrix, it is best approximated by a positive definite matrix, $\hat{R}$. It is stored in a file named by the user. $PROG_1$ invokes two

subroutine and one *function* subprograms. The second main program ($PROG_2$) reads a valid intercorrelation matrix (may be the output of $PROG_1$) and generates $X(n,m)$. $PROG_2$ invokes three *subroutine* and one *function* subprograms. The function that generates random numbers and the subroutine that finds eigenvalues are common to both ($PROG_1$ and $PROG_2$). While compiling $PROG_2$ it should be linked to the subroutine EIGEN and function RAND. Some procedures in the computer program (especially, the one that computes eigenvalues and eigenvectors) have been adapted from Krishnamurthy and Sen (1976), pp. 242-247. These source codes may easily be translated into any other computer language such as Pascal, C^{++} or even BASIC, if needed. Some languages may not have a provision to perform double precision arithmetic. In that case, single precision arithmetic may be used. The results would be sufficiently accurate for the desired purpose. In its present FORTRAN codes, the programs may be compiled by any suitable FORTRAN compiler. We have compiled the programs by Microsoft FORTRAN Compiler.

7. Inputs to the Computer Programs

When these programs are run, they ask for the following parameters (and inputs). Although they have been sufficiently explained in the program queries, they are explained here.

$PROG_1$: Before running the program, the Q matrix should be stored in some file. This can be done by some text editor such as EDIT.COM (a DOS program of MICROSOFT). The name of this file is, say ***inputfile.*** When the program runs, it asks for the value of m (order of the matrix) and the inputfile name (in which Q is stored). The file name should be in single quotes '***inputfile***'. Then it asks for the seed to generate random number: With this seed the uniformly distributed random numbers lying between (0, 1) = U(n,m) are generated. This number should lie between –32767 and 32767, zero excluded. This is a suitable number for most personal computers.

The program runs and if Q is not negative definite, it terminates. If so, the *inputfile* and the *outputfile* of $PROG_1$ are identical. If Q is negative definite, the program obtains $\hat{R}$ and asks for the outputfile name to store it. The file name should be in single quotes '*out**putfile***'. This *outputfile* then is used by $PROG_2$ as its *inputfile*.

$PROG_1$ should be run once more on its own output file to ensure that the resulting matrix is positive semidefinite. This is required because the output file stores correlation matrix with rounded off elements. Since the output matrix is almost always near-singular, rounding off may often make it negative definite. Note that a negative definite correlation matrix, Q, is a problematic and pathological case. It has to be handled with care and patience.

$PROG_2$: If the original Q fed by the user was already valid, the *inputfile* of $PROG_1$ is also the *inputfile* of $PROG_2$. Otherwise, the *outputfile* of $PROG_1$ is the *inputfile* of $PROG_2$. When $PROG_2$ runs, it asks for the following inputs.

1. Have you stored the intercorrelation matrix, etc. Yes is the answer.
2. What are N and M ?
3. Feed non-zero scalar, etc : Feed 1 or any other non-zero number.

4. Seed to generate random number: With this seed the uniformly distributed random numbers lying between (0, 1) = U(n,m) are generated. This number should lie between –32767 and 32767, zero excluded.
5. File in which correlation matrix is stored : As explained before, if the original Q fed by the user was already valid, the *inputfile* of $PROG_1$ is also the *inputfile* here. Otherwise, the *outputfile* of $PROG_1$ is the *inputfile* here.
6. Output file in which the generated X(n,m) characterizing intercorrelation matrix R will be stored : the output file name in single quotes ‘ ***outputfile***‘ is fed.
7. On termination the program stores the results X(n,m) in the ***outputfile***. It also stores the computed R matrix there, which may be different from the desired matrix R only slightly (may be at the 9^{th} or the 10^{th} place onwards after decimal).

Presently, in the codes given here, maximum N is 100 (=NL) and the maximum M is 10 (=ML). These parameters can be increased. Accordingly, dimensions in the program may be changed before compilation.

8. Fields of Application

In Monte Carlo experiments that evaluate performance of competing estimators of regression coefficients (or evaluates the efficacy of a method of estimation of parameters) under severe multicollinearity conditions, we require to generate X(n,m) that are highly multicollinear across the variables. The author (see Mishra, 2004) generated highly multicollinear X(n,m) variables to test the performance of Maximum Entropy Leuven estimators vis-à-vis the OLS estimator of β in the model $y = X\beta + u$. To generate X(n,m), a slightly different algorithm (than the one presented here) was used. Filzmoser and Croux (2002) generated highly multicollinear $Z = [X_1 \mid X_2]$ by first generating $X_1 \sim N(0, \Sigma)$ and then obtaining $X_2 = X_1 + \Delta$ where $\Delta \sim N(0,\ 0.001)$. This procedure yielded $Z = [X_1 \mid X_2]$ highly correlated across X_1 *and* X_2. In his paper Paris (2001) dealt with multicollinear regressors but he did not explain how multicollinearity was induced into X(n,m). He (see Paris, 2001, p. 4) wrote: “X was drawn from a uniform U[-1.7, 2.0] … each component of the disturbance vector u was drawn from a normal distribution N[1, 5].”

Sometimes two variables Y and Z are each *cointegrated* with another variable X, but Y and Z do not appear to be cointegrated with each other, although, intuitively, one would expect that they should be cointegrated with each other and the transitivity property would be exhibited. By carrying out a Monte Carlo simulation, Ferré (2004) showed that even though the two variables were in fact cointegrated, the test for cointegration was not able to pick this up due to the interplay of the error terms of the relationships between the variables. By using the algorithm presented here, several such examples may be generated for experiments and further investigation. We present here two intercorrelation matrices which can be used (as inputs to the program given here) to generate X(n,m) that would show intransitivity of cointegration.

In the matrix given below, $r(x_1, x_5)$ is zero while other elements are large enough to exhibit cointegration. If this matrix is used to generate X(n,m) for n howsoever large (say 500 or so), we will obtain an example to show a lack of transitivity relation in cointegration.

Another intercorrelation matrix with elements : $r_{11} = r_{22} = r_{33} = 1.00$, $r_{12} = r_{21} = 0.60$, $r_{13} = r_{31} = 0.00$, $r_{23} = r_{32} = 0.55$ will produce a similar instance. Many such examples may be generated.

Intercorrelation Matrix of X Showing Intransitivity of Cointegration					
Variables	X_1	X_2	X_3	X_4	X_5
X_1	1.00	0.61	0.52	0.58	0.00
X_2	0.61	1.00	0.62	0.65	0.50
X_3	0.52	0.62	1.00	0.64	0.61
X_4	0.58	0.65	0.64	1.00	0.76
X_5	0.00	0.50	0.61	0.76	1.00

Finally, experiments that directly or indirectly use multivariate analysis methods (such as Principal components analysis, Canonical correlation analysis, Factor analysis or Cluster analysis; see Kendall and Stuart, 1968) as a *procedure* may require X(n,m) with a desired R matrix. In such experiments our algorithms may be useful.

9. Limitations and possibilities of improvement

Although theoretically there are no snags in minimizing the maximum norm of deviation of $\hat{R}$ from Q, our algorithm has clearly two weaknesses, (1) it fails if at any stage of iteration the intermediate $\hat{R}$ turns out to be extremely near-singular, and, for some pathological cases of Q, PROG_1 may not converge; and (2) the random walk method is a very crude and slow method of optimization. It is easy to preclude extreme near-singularity of $\hat{R}$ at any intermediate stage. But it would be a further research work to replace the random walk method of optimization by some more efficient method such as the ***Genetic Algorithm*** (see Holland, 1975; Goldberg, 1989; Wright, 1991).

References

Al-Subaihi, AA (2004). "Simulating Correlated Multivariate Pseudorandom Numbers", At *www.jstatsoft.org/counter.php?id=85& url=v09/i04/paper.pdf&ct=1* Searched by http://www.google.com on 28th July, 2004.

Ferré, M (2004). "The Johansen Test and the Transitivity Property", *Economics Bulletin*, Vol. 3 (27), pp. 1-7.

Filzmoser, P and C Croux (2002). "A Projection Algorithm for Regression with Collinearity", in K Jajuga, A Sokolowski, and HH Bock (eds), *Classification, Clustering, and Data Analysis,* Springer-Verlag, Berlin, pp. 227-234.

Fleishman, A (1978). "A Method for Simulating Non-Normal Distributions", *Psychometrica*, 43(4), pp. 521-532.

Gillett, BE (1979). *Introduction to Operations Research*. Tata McGraw Hill, New Delhi.

Goldberg, DE (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Reading, Mass.

Headrick TC and SS Sawilowski (1999). 'Simulating Correlated Multivariate Non-normal Distributions extending the Fleishman Power Method", *Psychometrica*, 64(1), pp. 25-35.

Higham, NJ (2002). 'Computing the Nearest Correlation Matrix – A Problem from Finance", *IMA Journal of Numerical Analysis*, 22, pp. 329-343.

Hoffman, PJ (1924). 'Generating Variables with Arbitrary Parameters", *Psychometrica*, 24, pp. 265-267.

Holland, J (1975). *Adaptation in Natural and Artificial Systems,* Univ. of Michigan Press, Ann Arbor, USA.

Kaiser, HF and K Dichman (1962). 'Sample and Population Score Matrices and Sample Correlation Matrices from an Arbitrary Population Correlation Matrix", *Psychometrica*, 27(2), pp. 179-182.

Kapur, J N and H C Saxena (1982). *Mathematical Statistics*. S Chand & Co. New Delhi.

Kendall, MG and A Stuart (1968). *The Advanced Theory of Statistics*, Vol. 3. Charles Griffin & Co. London.

Knuth, DE (1969). *The Art of Computer Programming*. Addison Wesley, London.

Krishnamurthy, EV and SK Sen (1976). *Computer-Based Numerical Algorithms*, Affiliated East-West Press, New Delhi.

Mishra, SK (2004). "Multicollinearity and Modular Maximum Entropy Leuven Estimator", *Social Science Research Network* at http://ssrn.com/author=353253

Paris, Q (2001). "Multicollinearity and Maximum Entropy Estimators", *Economics Bulletin*, Vol. 3 (11), pp. 1-9.

Takayama, A (1974). *Mathematical Economics*, The Dryden Press, Illinois.

Texas Instruments Inc (1979). *TI Programmable 58C/59 Master Library*, Texas Instruments Inc. Texas.

Tadikamalla, PR (1980). 'On Simulating Non-normal Distributions", *Psychometrica*, 45(2), pp. 273-279.

Theil, H (1971). *Principles of Econometrics*, Wiley, New York.

Vale, CD and VA Maurelli (1983). 'Simulating Multivariate Nonnormal Distributions", *Psychometrica*, 48(3), pp. 465-471.

Wright, AH (1991). 'Genetic Algorithms for Real Parameter Optimization", in GJE Rawlings (ed) *Foundations of Genetic Algorithms*, Morgam Kauffmann Publishers, San Mateo, CA, pp. 205-218.

```
C     ------------- PROG1 to generate R from Q ------------
C     RANDOM WALK METHOD TO FIND Min(max norm) Nearest Positive
C     Semidefinite Marix from a given Non-positive definite matrix
C     ------------------------------------------------------------
      INTEGER *2 IU,IV
      DOUBLE PRECISION A(10),R(10),AR(10),XO(10,10),AA(10)
      DOUBLE PRECISION V(10,10),W(10,10),P(10),D,RH(10,10)
      DOUBLE PRECISION SUML,LAMBDA,EPS,F,VO,VR,VOO,RAND,CN
      DOUBLE PRECISION RNORM
      DIMENSION MM(10)
      CHARACTER *11 OFIL,IFIL
C     PARAMETERS ----- MAY BE CHANGED --------------------
C     EPS=ITERATIVE ACCURACY: EPSL=MAIN DIAGONAL ACCURACY
C     NITR=NO. OF TRIALS FOR RANDOM WALK SEARCH
C     ITMAX=MAX NO. OF ITERATION FOR CONVERGENCE
      EPS= 0.00001
      EPSL=0.00001
      NITR=10
      ITMAX=100
C     ------------------------------------------------
      WRITE(*,*)'FEED M and INPUT FILE NAME'
      WRITE(*,*)'(M is the order of the input square matrix'
      WRITE(*,*)'INPUT FILE NAME in single quotes)'
      READ(*,*) M,IFIL
      OPEN(7,FILE=IFIL)
      DO 1 I=1,M
      READ(7,*)(XO(I,J),J=1,M)
    1 CONTINUE
      CLOSE(7)
      WRITE(*,*)'FEED SEED TO GENERATE RANDOM NUMBERS'
      WRITE(*,*)'(SEED lies between -32767 AND 32767, avoid zero)'
      READ(*,*) IU
      WRITE(*,*)'OUTPUT FILE ? '
      READ(*,*) OFIL
      VOO=10.0**10
      LTRY=0
      ICO=1
      CALL CONS(XO,P,M,ICO)
      WRITE(*,*)'EIGEN VALUES OF THE ORIGINAL MATRIX XO ARE :'
      WRITE(*,*)(P(I),I=1,M)
      PAUSE 'STRIKE ENTER TO PROCEED'
      DO 70 I=1,M
      IF(P(I).LT.0.0) GOTO 90
   70 CONTINUE
      WRITE(*,*)'ALL EIGEN VALUES ARE NON-NEGATIVE'
      STOP
C     ============================================================
   90 WRITE(*,*)'SOME EIGEN VALUES ARE NEGATIVE'
      WRITE(*,*)'THE R MATRIX AT THIS STAGE IS : '
      OPEN(8,FILE=OFIL, STATUS='NEW')
      DO 789 I=1,M
      P(I)=1.0
  789 WRITE(8,89)(RH(I,J),J=1,M)
      CLOSE(8)
      DO 788 I=1,M
```

```
  788 WRITE(*,89)(RH(I,J),J=1,M)
      ITEST=0
C     PAUSE 'STRIKE ENTER TO PROCEED'
      F=0.0
      DO 71 I=1,M
      IF(P(I).LE.0.0) P(I)=RAND(IU,IV)
      F=F+P(I)
   71 CONTINUE
      DO 72 I=1,M
      P(I)=DABS(P(I)/F*M)
   72 CONTINUE
      WRITE(*,*)'EIGEN VALUES ARE FORCED TO BE ALL POSITIVE'
      SUML=0.0
      DO 78 I=1,M
      SUML=SUML+P(I)
   78 CONTINUE
C     WRITE(*,*)(P(I),I=1,M),' SUM = ',SUML
      DO 999 IIT=1,NITR
      LAMBDA=20.0
C     Initialisation of decision variables
      DO 7 I=1,M
    7 A(I)=DABS(P(I))
      ICO=2
      CALL CONS(RH,P,M,ICO)
C     --------------- FUNCTION EVALUATION ----------------------
      F=0.0
      DO 11 I=1,M
      DO 11 J=1,M
      D=DABS(XO(I,J)-RH(I,J))
      IF(D.GT.F) F=D
   11 CONTINUE
      VO=F
C     ----------------------------------------------------------
      IT=0
  200 IT=IT+1
      IF(IT.GT.1000) THEN
      WRITE(*,*)'NO CONVERGENCE IN 1000 ITERATIONS'
      GOTO 1000
      ENDIF
      LAMBDA=LAMBDA/2.0
      IMP=0
      DO 100 II=1,ITMAX
C     GENERATE M UNIFORMLY DISTRIBUTED RANDOM NUMBERS (-1,1)
  150 DO 2 I=1,M
      R(I)=2.0*(RAND(IU,IV)-0.5)
    2 CONTINUE
C     NORMALISE THE RANDOM NUMBERS
      RNORM=0.0
      DO 3 I=1,M
      RNORM=RNORM+R(I)**2
    3 CONTINUE
      RNORM=DSQRT(RNORM)
      IF(RNORM.GT.1.0) GOTO 150
      DO 4 I=1,M
      R(I)=R(I)/RNORM
    4 CONTINUE
C     ADD RANDOM NUMBERS TIMES LAMBDA TO A VECTOR
```

```
      DO 5 I=1,M
      AR(I)=A(I)+LAMBDA*R(I)
      AR(I)=DABS(AR(I))
    5 CONTINUE
C     --------------- FUNCTION EVALUATION ---------------------
      CN=0.0
      DO 73 I=1,M
      CN=CN+AR(I)
   73 CONTINUE
      SUML=0.0
      DO 74 I=1,M
      AR(I)=AR(I)/CN*M
      SUML=SUML+AR(I)
   74 CONTINUE
C     WRITE(*,*) 'SUML = ',SUML
      ICO=2
      CALL CONS(RH,AR,M,ICO)
      F=0.0
      DO 13 I=1,M
      DO 13 J=1,M
      D=DABS(XO(I,J)-RH(I,J))
      IF(D.GT.F) F=D
   13 CONTINUE
      VR=F
C     ------------------------------------------------------------
      IF(VR.LT.VO) THEN
      VO=VR
      DO 6 I=1,M
      A(I)=AR(I)
    6 CONTINUE
      IMP=1
      ENDIF
  100 CONTINUE
      IF((IMP.EQ.0).OR.(LAMBDA.GT.EPS)) GOTO 200
 1000 CONTINUE
      IF(VOO.GT.VO) THEN
      VOO=VO
      DO 998 I=1,M
      AA(I)=A(I)
  998 CONTINUE
      ENDIF
  999 CONTINUE
      DO 997 I=1,M
      A(I)=AA(I)
  997 CONTINUE
      VO=VOO
      SUML=0.0
      DO 77 I=1,M
      SUML=SUML+A(I)
   77 CONTINUE
      WRITE(*,*)'SMALLEST MAX DEVIATE = ',VO
      WRITE(*,*)' '
      WRITE(*,*)' TRIAL NUMBER = ',LTRY
      WRITE(*,*)' '
      DO 152 I=1,M
      IF(DABS(RH(I,I)-1.00).GT.EPSL) THEN
      RH(I,I)=1.0
```

```
      ITEST=1
      ENDIF
  152 CONTINUE
      IF(ITEST.EQ.1) THEN
      CALL CONS(RH,AR,M,1)
      LTRY=LTRY+1
      VOO=10.0**10
      GOTO 90
      ENDIF
      Write(*,*)' ------------ Convergence achieved ---------'
      write(*,*)' '
      WRITE(*,*)' NAME THE OUTPUT FILE TO STORE THE RESULT'
      WRITE(*,*)' (OUTPUT FILE NAME IN SINGLE QUOTES)'
      WRITE(*,*)'ESTIMATED MATRIX'
      OPEN(8,FILE=OFIL, STATUS='NEW')
      DO 75 I=1,M
      RH(I,I)=1.00
      WRITE(*,89)(RH(I,J),J=1,M)
      WRITE(8,89)(RH(I,J),J=1,M)
   75 CONTINUE
      CLOSE(8)
   89 FORMAT(1X,6D13.5)
      WRITE(*,*)'RESULTING MATRIX STORED IN FILE = ',OFIL
      WRITE(*,*)'RUN THIS PROGRAM ONCE MORE ON ITS OWN OUTPUT FILE'
      WRITE(*,*)'UNTIL IT SAYS ALL EIGENVALUES ARE NON-NEGATIVE'
      END
C     ------------------------------------------------
      SUBROUTINE CONS(A,P,M,ICO)
C     Constructs Matrix from its eigenvectors and values
      DOUBLE PRECISION A(10,10),B(10,10),V(10,10),W(10,10),P(10),F
      DIMENSION MM(10)
      IF(ICO.GT.1) GOTO 100
      NN=1
 1000 NADJUST=0
      DO 10 I=1,M
      DO 10 J=1,M
      B(I,J)=A(I,J)
   10 CONTINUE
      CALL EIGEN(A,M,NN,V,W,P,MM)
C     ================================================
C     NORMALIZATION OF EIGEN VECTORS TO UNITY
      DO 50 I=1,M
      P(I)=A(I,I)
      F=0.0
      DO 51 J=1,M
   51 F=F+V(J,I)*V(J,I)
      F=DSQRT(F)
      DO 52 J=1,M
   52 V(J,I)=V(J,I)/F
   50 CONTINUE
      DO 11 I=1,M
      DO 11 J=1,M
      A(I,J)=B(I,J)
   11 CONTINUE
      RETURN
C     ================================================
  100 DO 34 J=1,M
```

```
      DO 341 JJ=1,M
  341 W(J,JJ)=0.0
      W(J,J)=P(J)
   34 CONTINUE
      DO 35 J=1,M
      DO 35 JJ=1,M
      A(J,JJ)=0.0
      DO 35 I=1,M
      A(J,JJ)=A(J,JJ)+V(J,I)*W(I,JJ)
   35 CONTINUE
      DO 36 J=1,M
      DO 36 JJ=1,M
      W(J,JJ)=0.0
      DO 36 I=1,M
      W(J,JJ)=W(J,JJ)+A(J,I)*V(JJ,I)
   36 CONTINUE
      DO 361 I=1,M
      DO 361 J=1,M
      A(I,J)=W(I,J)
  361 CONTINUE
C     WRITE(*,*)'NOW A IS V*L*VT MATRIX'
C     NORMALIZED V (EIGEN VECTORS) ARE UNDISTURBED
      RETURN
      END
C     ------------------------------------------------------
      SUBROUTINE EIGEN(A,N,NN,V,W,P,MM)
C     Computes eigenvalues and vectors of a real symmetrix matrix
      DOUBLE PRECISION A(10,10),V(10,10),W(10,10),P(10)
      DOUBLE PRECISION PMAX,EPLN,TAN,SIN,COS,AI,TT,TA,TB
      DIMENSION MM(10)
C     ------------ INITIALISATION -------------------------
      DO 50 I=1,N
      DO 51 J=1,N
      V(I,J)=0.0
   51 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     ------------------------------------------------------
      IF(NN.NE.0) THEN
        DO 3 I=1,N
        DO 3 J=1,N
        V(I,J)=0.0
        IF(I.EQ.J) V(I,J)=1.0
    3   CONTINUE
      ENDIF
    2 NR=0
    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))+
     1DSQRT((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
   20 IF(I.EQ.JP)GO TO 18
   19 IF(MM(I).EQ.IP) GO TO 21
      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)
      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     ---------------------------------------------------------------
      FUNCTION RAND(IU,IV)
C     Generates Rectangular (0,1) Random Numbers
      DOUBLE PRECISION RAND
      INTEGER *2 IU,IV
      IV=IU*259
      IF(IV.GE.0) GOTO 2
      IV=IV+32767+1
   2  RAND=IV
      IU=IV
      RAND=RAND*0.3051851E-04
      RETURN
      END
```

```
C     ----- PROG2 : Main Program to generate X from R ------
      COMMON ML,NL,FC,FCO
      DOUBLE PRECISION X(100,10),SCALE
      CHARACTER *15 FC,FCO
      INTEGER *2 IU,IV
C     ---------------------------------------------------------
C     NL AND ML ARE THE HIGHEST PERMISSIBLE DIMENSION LIMITS TO
C     X(NL,ML) MATRICES. OTHER MATRICES HAVE COMPATIBLE DIMENSIONS
C     CHANGE THEM IF REQUIRED AND PERMISSIBLE BY MEMORY LIMITS.
      NL=100
      ML=10
C     ---------------------------------------------------------
      WRITE(*,*)'HAVE YOU STORED THE CORRELATION MATRIX, R ? IF NOT'
      WRITE(*,*)'STORE IT IN AN ASCII FILE. THEN RUN THE PROGRAM'
      WRITE(*,*)'IF NO THEN FEED ZERO (0).IF YES FEED ANY OTHER NUMBER'
      READ(*,*) NY
      IF(NY.EQ.0) THEN
      WRITE(*,*)'SO, STORE R MATRIX FIRST THEN RUN THE PROGRAM'
      STOP
      ENDIF
C     ---------------------------------------------------------
      WRITE(*,*)'WHAT ARE N AND M ? '
      WRITE(*,*)'(N = NO. OF OBSERVATIONS, M = NO. OF VARIABLES)'
      READ(*,*) N,M
      WRITE(*,*)'NON-ZERO SCALAR TO SCALE UP THE X VARIABLES ?'
      WRITE(*,*)'(NOT NECESSARY. FEED 1 OR ANY OTHER NON-ZERO NUMBER'
      READ(*,*) SCALE
      WRITE(*,*)'FEED A NON-ZERO SEED TO GENERATE RANDOM VARIABLE ?'
      WRITE(*,*)'(SEED MUST LIE BETWEEN -32767 AND 32767 AND NOT ZERO)'
      READ(*,*) IU
      WRITE(*,*)'FILE IN WHICH CORRELATION MATRIX IS STORED ?'
      WRITE(*,*)'FEED THE FILE NAME IN THE SINGLE QUOTES'
      READ(*,*) FC
      WRITE(*,*)'FILE IN WHICH OUTPUT X WILL BE STORED ?'
      WRITE(*,*)'FEED THE FILE NAME IN THE SINGLE QUOTES'
      READ(*,*) FCO
      OPEN(9,FILE=FCO,STATUS='NEW')
      CALL GENX(X,N,M,IU,IV,SCALE)
      CLOSE(9)
      END
C     ---------------------------------------------------------
C
      SUBROUTINE GENX(X,N,M,IU,IV,SCALE)
      COMMON ML,NL,FC,FCO
      DOUBLE PRECISION A(10,10),V(10,10),W(10,10),P(10),R(10)
      DOUBLE PRECISION X(NL,ML),Z(100,10),SUML,RAND,SCALE
      DIMENSION MM(10)
      CHARACTER *15 FC,FCO
      INTEGER *2 IU,IV
C     WRITE(*,*)'ENTERS GENX'
C     ----------------- PARAMETERS --------------------------------
      NN=1
      NSTD=1
C     ---------------------------------------------------------
      DO 1 I=1,N
      DO 1 J=1,M
      X(I,J)=RAND(IU,IV)*SCALE
```

```
    1 CONTINUE
      CALL CORR(X,N,M,A,NSTD)
      CALL EIGEN(A,M,NN,V,W,P,MM)
C      WRITE(*,*)'RETURNS FROM EIGEN'
C      PAUSE
      DO 301 I=1,N
      DO 301 J=1,M
      Z(I,J)=0.0
      DO 301 K=1,M
      Z(I,J)=Z(I,J)+X(I,K)*V(K,J)
  301 CONTINUE
C     ------------------------------------------------------
      CALL CORR(Z,N,M,A,NSTD)
      OPEN(8,FILE=FC)
      DO 302 I=1,M
      READ(8,*) (A(I,J),J=1,M)
  302 CONTINUE
      CLOSE(8)
      CALL EIGEN(A,M,NN,V,W,P,MM)
C      WRITE(*,*)'RETURNS FROM EIGEN'
C     ------------------------------------------------------
      DO 60 I=1,M
      R(I)=A(I,I)
   60 CONTINUE
      WRITE(*,*)'EIGENVALUES = ',(R(I),I=1,M)
      DO 64 I=1,M
      IF(R(I).LE.0.0) THEN
      WRITE(*,*)'SOME OF THE EIGENVALUES ARE NOT POSITIVE'
      WRITE(*,*)'PROGRAM TERMINATED. FEED DIFFERENT R MATRIX'
      STOP
      ENDIF
   64 CONTINUE
      DO 62 J=1,M
      P(J)=0.0
      DO 61 I=1,M
      P(J)=P(J)+V(I,J)**2
   61 CONTINUE
      P(J)=DSQRT(P(J)/R(J))
   62 CONTINUE
      DO 63 J=1,M
      DO 63 I=1,M
      W(I,J)=V(I,J)/P(J)
   63 CONTINUE
      DO 304 I=1,N
      DO 304 J=1,M
      X(I,J)=0.0
      DO 304 K=1,M
      X(I,J)=X(I,J)+Z(I,K)*W(J,K)
  304 CONTINUE
      WRITE(9,*)'GENERATED X(N,M) MATRIX'
      DO 305 I=1,N
      WRITE(9,310)(X(I,J),J=1,M)
  305 CONTINUE
      WRITE(9,*)'COMPUTED INTER-CORRELATION MATRIX'
      CALL CORR(X,N,M,A,NSTD)
      DO 306 I=1,M
      WRITE(9,310)(A(I,J),J=1,M)
```

```
  306 CONTINUE
  310 FORMAT(1X,5D15.7)
      RETURN
      END
C     ----------------------------------------------------------------
      SUBROUTINE CORR(X,N,M,A,NSTD)
      COMMON ML,NL,FC,FCO
      DOUBLE PRECISION X(NL,ML),A(10,10),AV(10),SD(10)
      CHARACTER *15 FC,FCO
      NSQR=N*N
      DO 1 J=1,M
      DO 2 JJ=J,M
      A(J,JJ)=0.0
      DO 2 I=1,N
    2 A(J,JJ)=A(J,JJ)+X(I,J)*X(I,JJ)
      DO 21 JJ=J,M
      A(JJ,J)=A(J,JJ)
   21 CONTINUE
    1 CONTINUE
      DO 3 J=1,M
      AV(J)=0.0
      DO 3 I=1,N
    3 AV(J)=AV(J)+X(I,J)
      DO 4 J=1,M
      DO 5 JJ=1,M
      A(J,JJ)=(N*A(J,JJ)-AV(J)*AV(JJ))/NSQR
    5 CONTINUE
    4 CONTINUE
      DO 6 J=1,M
      AV(J)=AV(J)/N
      SD(J)=DSQRT(A(J,J))
    6 CONTINUE
      DO 7 J=1,M
      DO 7 JJ=1,M
      A(J,JJ)=A(J,JJ)/(SD(J)*SD(JJ))
    7 CONTINUE
      IF(NSTD.NE.0) THEN
      DO 8 I=1,N
      DO 8 J=1,M
      X(I,J)=(X(I,J)-AV(J))/SD(J)
    8 CONTINUE
      ENDIF
   10 FORMAT(6D12.4)
      WRITE(*,*)'CORRELATION MATRIX -------------------------'
      DO 22 I=1,M
      WRITE(*,10)(A(I,J),J=1,M)
   22 CONTINUE
      WRITE(*,*)'-------------------------------------------'
C      PAUSE
      RETURN
      END
```