SANDIA REPORT SAND81-1376 • Unlimited Release
Printed June 1982

An Iterative Algorithm to Produce a Positive Definite Correlation Matrix from an "Approximate Correlation Matrix" (With a Program User's Guide)

Ronald L. Iman, James M. Davenport

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

SF2900Q(8-81)

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors or subcontractors.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND81-1376

Unlimited Release
Printed June 1982

An Iterative Algorithm to Produce a Positive Definite Correlation Matrix from an "Approximate Correlation Matrix" (With a Program User's Guide)

James M. Davenport
Department of Mathematics
Texas Tech University
Lubbock, Texas 79409

Ronald L. Iman
Statistics and Computing Division 1223
Sandia National Laboratories
Albuquerque, New Mexico 87185

ABSTRACT

This report contains an explanation of an algorithm that, when executed, will operate on any symmetric "approximate correlation matrix" by iteratively adjusting the eigenvalues of this matrix. The objective of this algorithm is to produce a valid, positive definite, correlation matrix. Also a description of a program (called POSDEF) which implements the algorithm is given.

THIS PAGE LEFT BLANK INTENTIONALLY

Table of Contents

		Page
I.	Introduction	1
II.	Description of the Algorithm	6
III.	Important Variations of the Algorithm	11
IV.	Program Description	19
	A. Limitations on the Program	19
	B. Input-Parameter Cards	20
	C. Explanation of Output	23
	D. An Example with Selected Portions of the Output	24
V.	Deck Setup for POSDEF	34
VI.	Discussion	37
Bibliography		41
Appendix A - Listing of the Fortran Source Deck for POSDEF		42

THIS PAGE LEFT BLANK INTENTIONALLY

I. Introduction

Many modeling situations exist where the assumption of independence among variables may not be appropriate. Indeed, "real world problems" exist where dependences or correlations do exist among the variables, and the design of an experiment investigating these variables should make use of this "correlation structure."

This dependence structure among the variables is appropriately summarized in a rank correlation matrix. Iman and Davenport (1980) discuss in some detail the advantages of using correlations, specifically rank correlations, as a method of inducing dependences among variables. Therefore, within this report, we assume that the dependence structure among the variables is adequately summarized in a rank correlation matrix.

In general, when we speak of correlations in this report, we will mean rank correlations, since it is not always appropriate to "talk about" a raw correlation. However, it should be pointed out that, if it is appropriate to define a Pearsonian correlation coefficient between two variables, then the methods discussed here apply without loss of generality.

It is extremely important that existing correlation structure be specified as accurately as possible. If the modeler is fortunate, he can estimate this correlation matrix from data, and hence, the resulting estimated correlation matrix is either positive definite or positive

semi-definite. However, the modeler may find that no data are available, and he is faced with the task of assigning values to these correlation coefficients by whatever methods he can assemble. Under such conditions, it is not unusual for the modeler to use a "sophisticated guess" for the value of a correlation coefficient. Furthermore, the modeler seldom can "estimate" the entire correlation structure simultaneously and is compelled to provide a collection of pair-wise correlation coefficients. That is, the modeler specifies this correlation matrix by setting the diagonal elements equal to +1.0 and each off diagonal element equal to some number between -1.0 and +1.0, that he feels best describes the dependence between those two variables.

Such a collection of pair-wise correlation coefficients does <u>not</u> a "correlation matrix" make! Indeed, Mother Nature does not always have a charitable character, and it is not too surprising that such "correlation matrices" are not positive definite. Therefore, the modeler must deal with the problem that he cannot assign values indiscriminately to the off-diagonal elements without regard for the total correlation structure.

Such an example is the case of the multivariate k-dimensional normal distribution. Let $\underset{\sim}{\Sigma}_\rho$ be the correlation matrix of this distribution such that all off-diagonal elements are equal to ρ (the diagonal elements are, of course, equal to +1.0). Then it can be shown that

$$-\frac{1}{k-1} < \rho < 1.$$

That is, as k becomes large, then there can be no large negative correlations.

The intent of this report is to present an algorithm (and a program) that will take the collection of pair-wise rank correlations (in the sequel, called the "approximate correlation matrix") and iteratively adjust this matrix until it is positive definite (note that the approximate correlation matrix is symmetric).

The algorithm given in this report is based on the premise that an approximate correlation matrix that is constructed by the method described above will have the following structure. Some of the eigenvalues will be positive. In fact, if the modeler has been careful in the selection of the pairwise correlations, taking into consideration the total correlation structure, then all of these eigenvalues will be positive; and hence the correlation matrix is positive definite. However, it may be possible for some of the eigenvalues to be zero or negative. If there are negative eigenvalues present and if one or more of those negative eigenvalues are relatively large in absolute value (say a value 1, 2 or 3) , then we say that this matrix is illconditioned with respect to this algorithm.

We have investigated several examples of this type of approximate correlation matrix, and have noted the following additional patterns. The number of negative eigenvalues was relatively small as compared to the dimension of the matrix. Also the values of these negative eigenvalues were very near zero. In addition, some of the positive eigenvalues were very near zero.

We feel that this structure that we have observed in these examples

is related to the multicollinearity problem that is common in regression analysis. That is, as the dimension of the matrix (the number of variables) increases, then the associated correlation matrix is more likely to be near singular. The present problem is made more complicated by the fact that some negative eigenvalues are also present.

If the eigenvalues that are negative are few in number and small in absolute value, then it should be possible to adjust them so that the initial matrix is transformed into a positive definite matrix (all eigenvalues positive), subject to the constraint that the values of the correlation coefficients in the new matrix are very close to the values in the initial approximate correlation matrix. While we present no theorems in this report indicating the conditions necessary and/or sufficient under which this algorithm will be valid, our experience in working with this type of matrix indicates that it will converge (i.e., we have not encountered any examples where it failed to converge.)

Likewise, our investigations of the several examples that we considered indicate that this iterative adjustment scheme produces small changes in the values of the correlation coefficients in the initial approximate correlation matrix. This is due to the fact that, for this type of matrix, small changes in the eigenvalues produce small changes in the entries of the matrix.

In Section II, we describe the algorithm that is used in the program POSDEF in detail. However, there are some very important variations to this algorithm that merit our attention. These are discussed in section

III. A description of the program POSDEF is given in section IV, and an example of the deck setup for using POSDEF is in section V. The final section contains a discussion.

II. Description of the Algorithm

Let $\underset{\sim}{C}$ be an n x n symmetric, non-singular matrix such that the elements of $\underset{\sim}{C}$ satisfy the following:

$$c_{ii} = 1 \qquad i = 1, 2, \ldots n$$

and $-1 < c_{ij} < 1$ for $i \neq j$.

Then it is well known that there exists a matrix $\underset{\sim}{Z}$ and a diagonal matrix $\underset{\sim}{D}$, each of dimension n x n such that (see Graybill, 1961)

$$\underset{\sim}{C}\underset{\sim}{Z} = \underset{\sim}{Z}\underset{\sim}{D}.$$

The columns of $\underset{\sim}{Z}$ are the respective eigenvectors of the matrix $\underset{\sim}{C}$ and the elements of $\underset{\sim}{D}$ are the eigenvalues. That is,

$$\underset{\sim}{D} = \begin{bmatrix} \lambda_1 & & & & & \\ & \lambda_2 & & & & \\ & & \lambda_3 & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & & \cdot \\ & & & & & & \lambda_n \end{bmatrix}$$

and we assume, without loss of generality, that $\lambda_1 < \lambda_2 < \ldots < \lambda_n$. Since Z is an orthogonal matrix, we have the following result.

$$\underset{\sim}{C}\underset{\sim}{Z}\underset{\sim}{Z}' = \underset{\sim}{Z}\underset{\sim}{D}\underset{\sim}{Z}'$$

$$\underset{\sim}{C} = \underset{\sim}{Z}\underset{\sim}{D}\underset{\sim}{Z}'$$

$$\underset{\sim}{C} = \sum_{i=1}^{n} \lambda_i \underset{\sim}{Z}_i \underset{\sim}{Z}_i'$$

Where $\underset{\sim}{Z}_i$ is the ith column of $\underset{\sim}{Z}$.

Now suppose that $\lambda_1 < \lambda_2 < \ldots. < \lambda_k$, $k < n$ are all negative, but small in absolute value. The first step in the algorithm is to change each of these negative eigenvalues to a small positive number that we will denote by ε. The idea is to make all of the eigenvalues positive, and hence, the new matrix to be calculated should be positive definite, or at least have fewer negative eigenvalues than it had previously. However, we do not wish to change the values of the individual elements of $\underset{\sim}{C}$ anymore than we have to. Therefore we choose a value for ε that is very small, namely $\varepsilon = 0.001$. Likewise, we choose not to change the eigenvectors, to minimize the effect on the matrix $\underset{\sim}{C}$.

The second step in the algorithm is to examine the eigenvalues $\lambda_{k+1} < \lambda_{k+2} < \ldots. < \lambda_{2k}$ and change these values, if necessary, by the following procedure.

$$\lambda_i^* = \begin{cases} \varepsilon \text{ if } \lambda_i < \varepsilon \\ \\ \lambda_i \text{ if } \lambda_i > \varepsilon \end{cases} \qquad \text{for } i = k+1, \ldots., 2k.$$

That is, if λ_i where $k + 1 \leq i \leq 2k$ is smaller than $\varepsilon = 0.001$, then we increase its value to .001. If it is larger than ε, then we leave λ_i unchanged. This portion of the algorithm improves the operating characteristics of the overall iteration scheme in the examples that we considered. This is discussed in more detail in Section III.

Now a new matrix can be calculated using the new eigenvalues discussed above and is defined as follows:

$$\underset{\sim}{C}^* = \sum_{i=1}^{k} \varepsilon \underset{\sim}{Z}_i \underset{\sim}{Z}_i' + \sum_{i=k+1}^{2k} \lambda_i^* \underset{\sim}{Z}_i \underset{\sim}{Z}_i' + \sum_{i=2k+1}^{n} \lambda_i \underset{\sim}{Z}_i \underset{\sim}{Z}_i'.$$

Recall that the original matrix $\underset{\sim}{C}$ had diagonal elements equal to +1. This will not be true of the matrix $\underset{\sim}{C}^*$. In general, the diagonal elements of $\underset{\sim}{C}^*$ will be slightly greater than one, and in fact, some of the off-diagonal elements of $\underset{\sim}{C}^*$ may be greater than one, depending upon the initial values within the matrix $\underset{\sim}{C}$.

Our objective is to produce a positive definite correlation matrix; therefore, some scaling of the elements of $\underset{\sim}{C}^*$ must be done in order to make it conform to the definition of a correlation matrix. The procedure used here is to simply replace each of the diagonal elements of $\underset{\sim}{C}^*$ by +1 and leave the off-diagonal elements unchanged. The result is a symmetric matrix with diagonal elements equal to positive one and off-diagonal elements approximately equal to those in $\underset{\sim}{C}$, the original matrix. We denote this new matrix by $\underset{\sim}{C}_1$ (indicating the end product of one cycle in the iteration process).

If $\underset{\sim}{C}_1$ is positive definite then the process is complete and no further adjustment is needed. If the matrix $\underset{\sim}{C}_1$ is not positive definite, then the entire process is repeated using the new matrix $\underset{\sim}{C}_1$ in place of the original matrix $\underset{\sim}{C}$. The end product of the second cycle of the iteration process is denoted by $\underset{\sim}{C}_2$. If $\underset{\sim}{C}_2$ is positive definite, then the process is complete. If $\underset{\sim}{C}_2$ is not positive definite then the entire process is repeated with a third cycle. This iteration process is executed as many times as necessary, until the end product is a positive definite matrix.

A disclaimer is in order. We do not know whether or not this algorithm will always converge to a positive definite matrix. As stated in a previous section, we have no theorems that indicate when it will converge and/or when it will fail. However, we have used this algorithm on many approximate correlation matrices and it has never failed to converge to a positive definite matrix.

Given that the algorithm does converge to a positive definite matrix (i.e., all of the eigenvalues are positive), then it would appear that there is a possibility of one or more of the off-diagonal elements having an absolute value larger than 1. The following theorem demonstrates that this is impossible.

<u>Theorem</u>. Given that the algorithm converges to a positive definite matrix A (of dimension nxn) then $|\rho_{ij}| < 1$ for all $i \neq j$ and $i, j = 1,2,\ldots,n$ where ρ_{ij} is the $(ij)^{th}$ element of A.

Proof: Assume ρ_{ij} is an arbitrary but fixed element of A such that $i \neq j$. Furthermore, assume $|\rho_{ij}| \geq 1$. The matrix A can be transformed

into a matrix B by elementary row and column interchanges so that $b_{12} = b_{21} = \rho_{ij} = \rho_{ji}$. That is

$$E_j\, E_i\, A\, E_i'\, E_j' = B = \begin{bmatrix} 1 & \rho_{ji} & \cdots & & b_{1n} \\ \rho_{ij} & 1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ b_{n1} & & & & 1 \end{bmatrix}$$

where E_i and E_j are the appropriately defined nonsingular matrices that accomplish the row and column interchanges. Note that the matrices A and B have identical eigenvalues (Graybill, 1961, Theorem 1.28, p.4). Further note that the matrix B is not positive definite, since $(1 - \rho_{ij}^2) < 0$ (Graybill, 1961, Theorem 1.23, p. 3-4). Therefore, at least one of the eigenvalues of B is zero or negative. But this contradicts the fact that all of the eigenvalues of the matrix A are positive. Therefore, $|\rho_{ij}| < 1$ for all $i \neq j$.

Hence, if the algorithm converges, then the matrix produced will indeed be a positive definite correlation matrix. The user should note that, in general, this matrix still will be very nearly singular. Likewise, our experience has shown that the changes in the values of the elements of the original matrix will be small (the reader is referred to Table 1 for some examples). In conclusion, the algorithm meets the objectives that we set at the outset.

III. Important Variations of the Algorithm

There are essentially two important steps in this adjusting algorithm. They are:

(1) adjust the values of the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{2k}$ associated with the matrix $\underset{\sim}{C}_i$, $i = 0,1,2,\ldots$

(2) scale the diagonal and off-diagonal elements of the matrix $\underset{\sim}{C}_i^*$, $i = 0,1,2,\ldots$

See Section II for the respective definitions of $\underset{\sim}{C}_i$ and $\underset{\sim}{C}_i^*$.

Several different schemes for accomplishing (1) were investigated before deciding to use the particular scheme described in section II. All of these were motivated by the following discussion.

Recall that we are assuming $\lambda_1, \lambda_2, \ldots, \lambda_k$ are all negative and small in absolute value. Any such adjustment scheme must change these to positive values. However, we do wish to minimize the effect on the elements of the original matrix, $\underset{\sim}{C}_i$. Therefore, it seems reasonable to adjust $\lambda_{k+1} < \lambda_{k+2} < \ldots < \lambda_{2k}$ accordingly to compensate for changing $\lambda_1 < \lambda_2 < \ldots < \lambda_k$. The objective is to minimize the net effect on the original matrix $\underset{\sim}{C}_i$.

Since λ_1 is the smallest eigenvalue, we arbitrarily pair λ_1 with λ_{2k}, the largest eigenvalue among the eigenvalues considered. Likewise pair λ_2 with λ_{2k-1} etc., and finally pair λ_k with λ_{k+1}. Consider the pair $(\lambda_1, \lambda_{2k})$. Recall that we will replace λ_1

with ε and now wish to change λ_{2k} to a value, say u, such that

$$\lambda_1 \underset{\sim}{Z}_1 \underset{\sim}{Z}_1' + \lambda_{2k} \underset{\sim}{Z}_{2k} \underset{\sim}{Z}_{2k}' = \varepsilon \underset{\sim}{Z}_1 \underset{\sim}{Z}_1' + u \underset{\sim}{Z}_{2k} \underset{\sim}{Z}_{2k}'$$

However, in general, this is impossible as no such value of u exists that satisfies all of these equations. So to obtain some "overall effect," we can sum these equations to form one equation in one unknown. Pre-multiplying the above equation by $\underset{\sim}{J}'$ and post-multiplying by $\underset{\sim}{J}$, where $\underset{\sim}{J}$ is a vector of all ones, yields

$$\lambda_1(\underset{\sim}{J}' \underset{\sim}{Z}_1)^2 + \lambda_{2k}(\underset{\sim}{J}' \underset{\sim}{Z}_{2k})^2 = \varepsilon(\underset{\sim}{J}' \underset{\sim}{Z}_1)^2 + u(\underset{\sim}{J}' \underset{\sim}{Z}_{2k})^2$$

Solving for u, we have

$$u = (\lambda_1 - \varepsilon)R + \lambda_{2k} \tag{3.1}$$

where

$$R = (\underset{\sim}{J}' \underset{\sim}{Z}_1)^2/(\underset{\sim}{J}' \underset{\sim}{Z}_{2k})^2 \tag{3.2}$$

Examination of equation 3.1 indicates that the essence of the adjustment to λ_{2k} is to decrease its value by some appropriate amount. Therefore, rather than using only equation 3.2 for the definition of R, we investigated many different ways of finding a suitable value of R including several obvious values such as zero and one.

Of course, it should be obvious that the total scheme of adjusting these eigenvalues would be accomplished on each pair of eigenvalues, $(\lambda_1, \lambda_{2k})$, $(\lambda_2, \lambda_{2k-1})$, ..., $(\lambda_k, \lambda_{k+1})$.

Note that any choice of $R \neq 0$ may produce a negative value for u (see equation 3.1), and hence this scheme would replace a positive eigenvalue with a negative number. This is obviously unacceptable, so we incorporated the following check into the algorithm.

```
IF(u . LT . ε )  u = ε                                    (3.3)
```

Hence, if $R = 0$ and statement 3.3 is included in the algorithm, the net effect is to replace those eigenvalues, among the set $(\lambda_{k+1}, \lambda_{k+2}, \ldots, \lambda_{2k})$, that are less than ε with the value of ε. On the surface, this may seem strange to leave statement 3.3 in the program when $R = 0$. Especially, since the eigenvalues $\lambda_{k+1}, \ldots, \lambda_{2k}$ should be left unchanged, whenever $R = 0$ (according to the equation 3.1). But empirical investigations indicated that leaving statement 3.3 in the algorithm whenever $R = 0$ slightly improved the convergence characteristics of this iterative algorithm. That is, the algorithm converged in slightly fewer iterations whenever statement 3.3 remained in the algorithm.

At the same time we investigated the effects of statement 3.3 being included or excluded from the program, we were investigating many non-zero values of R. Also we investigated several techniques of "estimating"

R from the eigenvectors $\underset{\sim}{Z}_i$, i = 1,2,...,2k. However, none of these produced any significant improvement in the algorithm over the choice R = 0.

In conclusion, the algorithm as stated in Section II amounts to setting R = 0 and including statement 3.3 in the program.

As stated at the beginning of this section, the second important step of this algorithm is to scale the elements of the matrix $\underset{\sim}{C}_i$*, i = 0,1,2, ... We tried five different methods of scaling this matrix and outline these in the following paragraphs.

Method A

The diagonal elements of the matrix are set equal to 1. No other changes are made.

Method B

The diagonal elements of the matrix are set equal to 1. Since some of the off-diagonal elements may be greater than or equal to one or less than or equal to minus one, it would seem that these elements should be scaled also. These off-diagonal elements are set equal to +.9999 and -.9999 respectively.

Method C

Each element in a row is divided by the square root of the diagonal element of that row. Likewise, each element in a column is divided by the square root of the diagonal element of that column. That is,

$$r(i,j) \text{ replaced by } \frac{r(i,j)}{\sqrt{r(i,i)} \cdot \sqrt{r(j,j)}}$$

where $r(i,j)$ is the $(i,j)^{th}$ element of the matrix under consideration. Note that this method guarantees that the diagonal elements will all be equal to 1. However, this does not insure that the off-diagonal elements will be between minus one and plus one.

Method D

Essentially, this method is the same as method C with the following exception. If a row (column) does not have any off-diagonal elements outside the interval (-1,1), then the division process described in method C is not accomplished. However, some of the diagonal elements may still be greater than 1 or less than 1. Therefore, we follow the needed divisions described above, by setting all of the diagonal elements equal to 1.

Method E

This method is the same as method D with the additional modification that whenever you do divide a row (column) by the square root the diagonal element of that row (column), you subtract 0.05 from the diagonal element before you take its square root. For example, suppose $r(4,4) = 1.0785$ and the division process is called for. Then every element in the fourth row and fourth column is divided by $\sqrt{1.0285}$. Then, as in method D, every diagonal element is set equal to 1.

These different methods of scaling the matrix produced some interesting results. For example, method C always converged in exactly one iteration on every example that we tested. However, method C also produced the greatest change in the "correlations" in the original matrix $\underset{\sim}{C}_i$ (see Table 1). For this reason, we decided not to use method C as the scaling scheme.

Several examples of approximate correlation matrices were tested using the algorithm with all combinations of the eigenvalue adjustments, the five scaling schemes, and with or without statement 3.3. One such matrix is given as an example in Figure 1. Within this matrix we selected two elements, namely ρ_{12} = .9000 an $\rho_{15,14}$ = .9999 and have summarized the effect of the iteration procedure on these two elements in Table 1.

As stated earlier, we investigated many techniques for selecting a value for R, but could find no value of R that offered any significant improvement over the choice R = 0. In Table 1, we present the two values R = 0 and R = 1.

Further investigation of the examples, whenever R = 0, indicated that the scaling scheme A seems to produce the smallest changes in the values within the original matrix. This is illustrated with the two cases given in Table 1.

Finally our investigations indicate that, whenever R = 0, the algorithm converged in fewer iterations with statement 3.3 included. Again, this is illustrated in Table 1.

Our recommendation to use R = 0, include statement 3.3 in the algorithm and use method A for the scaling scheme. This is the algorithm described in section II.

TABLE 1

COMPARISON OF DIFFERENT ADJUSTMENT AND SCALING SCHEMES

Value of R	Statement 3.3 *	Scaling Scheme	Value of Correlation Coef. After Convergence: Target Value = .9999	Value of Correlation Coef. After Convergence: Target Value = .9000	Number of Iterations to Convergence
0	w	A	.9708	.9011	11
0	w/o	A	.9711	.8975	12
0	w	B	.9689	.9011	11
0	w/o	B	.9692	.8975	12
0	w	C	.9389	.8698	1
0	w/o	C	.9389	.8698	1
0	w	D	.9425	.8996	11
0	w/o	D	.9421	.8990	11
0	w	E	.9647	.9011	11
0	w/o	E	.9635	.8942	12
1	w	A	.9712	.8790	12
1	w/o	A	.9703	.8747	11
1	w	B	.9712	.8790	12
1	w/o	B	.9703	.8747	11
1	w	C	.9421	.9133	1
1	w/o	C	.9421	.9133	1
1	w	D	.9712	.8790	12
1	w/o	D	.9703	.8747	11
1	w	E	.9712	.8790	12
1	w/o	E	.9703	.8747	11

* - w indicates that statement 3.3 is included, w/o indicates that it was excluded.

IV. Program Description

The description contained herein details how to use the program, called POSDEF, that implements the algorithm described in section II. This program has been developed at Sandia Laboratories and a complete listing of the Fortran computer code is given in Appendix A.

The procedure for the program is basically very simple. The "approximate correlation matrix" is read into the program. This matrix is adjusted iteratively until it converges to a positive definite matrix or until a fixed number of iterations have been executed. In the following, we discussed the limitations on the program, input parameter card requirements, an explanation of the output, and an example with selected portions of the output.

A. Limitations on the Program

As the program presently exists, there are several limitations. However, these are not absolute limitations and can be changed by the user to suit his/her needs.

The first limitation concerns the upper limit on the size of the correlation matrix that can be read into the program. At present, a matrix of dimensions less than or equal to 20 x 20 can be accommodated. In fact, the program expects this matrix to be in symmetric storage notation. This is explained in the next subsection.

The second limitation concerns the value used for ε. Within the program, this is designated by the variable EIG. The value used for this

quantity is 0.001. This value can be changed without adversely affecting the program, but the user should realize that if the value of EIG is increased, this will increase the number of iterations required for convergence and cause greater perturbations in the individual elements of the original matrix.

The third limitation concerns the upper limit on the number of iterations allowed. The program presently allows a maximum of 20 iterations. If the algorithm does not converge after 20 iterations, the user should examine the original input "approximate correlation matrix" very carefully for unlikely configurations of correlation structure. This is discussed in more detail in section VI.

The fourth limitation concerns the need for the IMSL subroutine EIGRS. The program POSDEF begins by decomposing the input matrix into its unique eigenvalues and eigenvectors. The IMSL subroutine will provide these eigenvalues and eigenvectors for an input matrix that is assumed to be in symmetric storage notation. If the EIGRS subroutine is not available, then some other subroutine that computes eigenvalues and eigenvectors should be used in its place.

B. Input - Parameter Cards.

All of the following parameter cards are required and must be in the order given. Also each card must conform to the format given. An explanation and illustration of each parameter card follows.

1. MATRIX SIZE AND OUTPUT CONTROL CARD - Format (2I5).

This card has two integer valued arguments specified as shown below:

NP	IDIAG
1... 5	6 ... 10

where

NP is the dimension of the "approximate correlation matrix" that is to be read in (NP - a maximum of 20).

IDIAG is an indicator variable that controls the printing of some of the output. After the last iteration the user may wish to see an analysis each p x p prime diagonal submatrix. See the next subsection for further discussion.

= 0 then this analysis will not be done.

= 1 then this analysis will be done and printed.

2. VARIABLE IDENTIFICATION CARD - Format (16I5)

This card reads in integer identification numbers that are associated with the variables that are represented by the "approximate correlation matrix." For example, suppose we wish to correlate the third, fourth and seventh variable out of ten variables being investigated. Then we would wish to label these variables as "3", "4", and "7" respectively. The following

example illustrates this and note that a maximum of 20 labels may be read in.

NK(17) 1 ... 5	. . .	NK(20) 16 ... 20		

NK(1) 1 ... 5	NK(2) 6 ... 10	NK(3) 11 ... 15	. . .	NK(16) 76 ... 80

The only purpose of these identification numbers is to label the variables in the output. The algorithm does not depend on these numbers in any way.

3. "APPROXIMATE CORRELATION MATRIX" CARD - Format (8G10.4)

The program is formated such that this matrix must be in symmetric storage notation. That is, the symmetric matrix $\underset{\sim}{C}$ is read in as a vector where the first element of the vector is c(1,1), the second is c(2,1), the third c(2,2), the fourth c(3,1), etc. As many cards as are necessary may be used as long as each card is in an 8G10.4 format. An example follows.

. . .	CORX(NKX)		

CORX(1) 1 ... 10	CORX(2) 11 ... 20	CORX(3) 21 ... 30	. . .

where NKX = NP(NP+1)/2. This is the last input-parameter card.

C. An Explanation of the Output.

A complete set of the following information is printed for each and every iteration.

1. The iteration number.
2. The input "approximate correlation matrix."
3. The eigenvalues and eigenvectors that are computed by the IMSL subroutine EIGRS. Note that the first row printed is actually the first column of $\underset{\sim}{Z}$ (see section II). That is, the first row printed is $\underset{\sim}{Z}_1'$. In general, the ith row printed is $\underset{\sim}{Z}_i'$, the ith column of $\underset{\sim}{Z}$.
4. The eigenvalues after the adjustment scheme has been executed (see sections II and III).
5. After the eigenvectors and the adjusted eigenvalues are multiplied appropriately to reconstruct the "new approximate correlation matrix," $\underset{\sim}{C}^*$, this matrix is printed.
6. This is followed by two statements indicating that the diagonal elements of $\underset{\sim}{C}^*$ are set equal to one. This completes a single iteration.

This information is printed for each iteration. If the algorithm converges on the mth iteration; then iteration m + 1 will be begun, but the program will detect that all the eigenvalues are positive. Then the program will indicate that all the eigenvalues were positive on iteration m + 1. This is followed by the final positive definite correlation matrix.

If IDIAG = 0 (see input - parameter card #1) then the following information will not be printed. If IDIAG = 1, then each p x p main diagonal submatrix p = 2, ..., NP is examined via its eigenvalues, eigenvectors and determinant. This completes the explanation of the output.

D. An Example with Selected Portions of the Output

In the following example, 16 of 64 different variables were to be correlated. The modeler provided a collection of pairwise rank correlations, that when taken as a "correlation matrix" was not positive definite. The labels for these 16 variables are 4, 8, 12, ..., 64. The deck set-up for this example is given in section V and the remainder of this example is best illustrated by the example output that follows.

Figure 1 gives the input "approximate correlation matrix." This is also the first page of the output of the program.

Figure 2 gives the eigenvalues, eigenvectors and determinant of this matrix. The eigenvalues after adjustment are also given. This is the second page of output.

Figure 3 gives the reconstructed "approximate correlation matrix" using the adjusted eigenvalues. Note that the diagonal elements and at least one of the off-diagonal elements is greater than one. This is also the third page of output.

Figure 4 gives the resulting "approximate correlation matrix" that is the end product of the first iteration. This matrix is also the

beginning matrix of the second iteration. This is the fourth page of output from the program.

Figure 5 presents the input matrix to the 12th iteration. Figure 6 presents the eigenvalues and eigenvectors of this matrix; as you can see, all the eigenvalues are positive and the determinant is positve. Therefore, the algorithm converged to this matrix after eleven iterations.

Figures 7 and 8 give respectively the eigenvalue/eigenvector analysis of the 2 x 2 and 3 x 3 prime diagonal submatrices of the matrix given in Figure 5. This is done for each prime diagonal submatrix as p = 2, ..., 16.

This completes the explanation of this example and illustrated output.

ITERATION NUMBER = 1

INPUT "RANK CORRELATION STRUCTURE" FOR THE RANDOM VARIABLES BEING TRANSFORMED

VARIABLE NUMBER

	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64
4	1.0000															
8	.9000	1.0000														
12	.6000	.5000	1.0000													
16	.7000	.7000	.5000	1.0000												
20	.8000	.8000	.5000	.6000	1.0000											
24	.5000	.5000	.3000	.6000	.6000	1.0000										
28	-.9000	-.8000	-.5000	-.4000	-.7000	-.4000	1.0000									
32	-.7000	-.6000	-.4000	-.6000	-.7000	-.6000	.8000	1.0000								
36	-.5000	-.7000	-.2000	-.8000	-.8000	-.8000	.7000	.9000	1.0000							
40	.2000	.1000	-.1000	.1000	.1000	.1000	-.1000	-.3000	-.5000	1.0000						
44	.1000	.1000	.1000	.1000	.1000	.1000	-.3000	-.3000	-.3000	.6000	1.0000					
48	-.7000	-.6000	-.4000	-.4000	-.3000	-.3000	.6000	.6000	.6000	-.1000	-.8000	1.0000				
52	-.8000	-.8000	-.6000	-.8000	-.8000	-.8000	.8000	.9000	.9000	-.6000	-.5000	.6000	1.0000			
56	.6000	.6000	.4000	.8000	.7000	.7000	-.8000	-.8000	-.9000	.6000	.6000	-.4000	-.9000	1.0000		
60	.6000	.6000	.2000	.9000	.7000	.7000	-.8000	-.7000	-.9000	.5000	.5000	-.4000	-.9000	.9999	1.0000	
64	-.7000	-.6000	-.2000	-.6000	-.6000	-.6000	.7000	.9000	.8000	-.5000	-.5000	.8000	.7000	-.7000	-.6000	1.0000

Variable Identification Numbers

Input "Approximate Correlation Matrix"

Off-diagonal element very near one

Figure 1

ANALYSIS OF THE 16 BY 16 PRIME DIAGONAL SUBMATRIX

PERFORMANCE INDEX = .102 } Information from Subroutine EIGRS - See the IMSL Manual

EIGENVALUES = Five negative eigenvalues

-.4119 -.2904 -.1623 -.0654 -.0006 .0756 .1844 .2670 .3596 .5073 .5312 .7244 .8397 1.4550 2.0274 9.9588

VECT. NO. EIGENVECTORS

1	.303	2.277E-02	8.596E-02	-.381	-.227	-.239	.377	.119	-.321	-.372	6.266E-02	.251	← First eigenvector		
	-.240	.133	.222	-.241											
2	-.264	-2.630E-02	-2.801E-02	-.213	.109	-.116	.183	7.864E-02	.326	1.198E-02	-.499	-.466	← second		
	-.234	.353	.242	-8.712E-02											
3	-3.744E-02	-.243	-.244	.227	3.605E-02	-.121	-.254	.314	-5.678E-02	-.169	4.633E-03	5.058E-02	.		
	-.565	.166	-.509	-9.580E-02									.		
4	.396	-8.400E-02	-.150	-5.291E-02	-6.404E-03	3.348E-02	.200	-.151	-.184	2.033E-02	-3.500E-02	-.211	etc.		
	.246	.535	-.334	.457											
5	.113	-.443	.224	-5.708E-02	6.499E-02	-1.874E-02	-.213	.434	-.497	.183	-.249	-.253			
	.192	-.184	.148	3.099E-03											
6	.274	-.351	-.215	3.801E-02	.159	-8.296E-02	.145	-.314	8.548E-02	-3.953E-02	3.917E-02	-.139			
	-.404	-.451	.252	.377											
7	.299	-.376	-4.489E-02	.106	.500	5.201E-02	.164	.153	.382	-8.389E-02	.186	.131			
	.284	.134	6.187E-02	-.381											
8	-.401	.125	9.053E-02	6.182E-02	.582	-.439	.274	-.112	-.364	-7.145E-02	.145	-4.858E-02			
	6.428E-02	-5.238E-02	-.145	5.743E-02											
9	-4.279E-02	-.356	.163	.379	-.319	-.455	-.111	-.481	2.294E-04	-4.980E-02	-.217	9.742E-02			
	.157	.145	4.327E-02	-.204											
10	-.296	-.314	9.815E-02	-.446	.137	.176	-.455	-.226	-8.730E-03	-.384	.192	.114			
	3.180E-02	.221	.125	.189											
11	-1.776E-02	-.219	.412	-.293	5.873E-02	.313	.247	-.315	-1.510E-02	.301	-.155	9.761E-02			
	-.213	-.119	-.455	-.226											
12	.261	.181	-.193	-.385	.234	-.426	-.377	-6.917E-02	8.171E-02	.410	-.216	.328			
	-5.136E-03	1.177E-02	-2.499E-02	-4.007E-03											
13	-1.475E-02	4.042E-02	-.603	-.147	1.301E-02	.179	-4.532E-02	-.262	-.251	-.241	-.224	-.209			
	.219	-.229	-.129	-.422											
14	.219	.106	.257	-.227	-.125	-.324	-.196	-1.750E-02	.234	-.137	.359	-.574			
	5.589E-02	-.188	-.266	-.151											
15	-.257	-.265	-.336	-.147	-.235	-7.555E-02	.123	-9.039E-03	-9.776E-02	.533	.524	-.108			
	-4.170E-02	.169	.135	-.158											
16	-.266	-.259	-.159	-.253	-.258	-.227	.269	.281	.293	-.127	-.140	.214			
	.314	-.293	-.284	.267											

DETERMINANT = -4.586E-09

Adjusted eigenvalues

Note that none of these five eigenvalues are less than .001. Therefore they are unchanged.

EIGENVALUES AFTER ADJUSTMENT =

.0010 .0010 .0010 .0010 .0010 .0756 .1844 .2670 .3596 .5073 .5312 .7244 .8397 1.4550 2.0274 9.9588

Figure 2

RESULTING PRODUCT MATRIX AFTER THE FIRST 10 EIGENVALUES HAVE BEEN TESTED AND ADJUSTED WHERE NECESSARY.
MINIMUM EIGENVALUE USED = .1000E-02 ← EIG • ≡ • ε

1.0688	.9041	.6105	.6660	.7629	.4807	-.8602	-.6970	-.5698	.1542	.1452	-.6386	-.8021	.6025	.6034	-.7108
.9041	1.0109	.5114	.6894	.7956	.5033	-.7887	-.6114	-.7019	.1029	.1046	-.5947	-.7795	.5891	.6222	-.6003
.6105	.5114	1.0146	.4797	.4897	.2969	-.4800	-.4073	-.2101	-.1067	.1064	-.3873	-.5865	.3898	.2296	-.2086
.6660	.6894	.4797	1.0816	.6302	.6402	-.4807	-.6115	-.7713	.1513	.1214	-.4080	-.7695	.7615	.8324	-.5618
.7629	.7956	.4897	.6302	1.0249	.6180	-.7310	-.7067	-.7599	.1342	.0783	-.3380	-.7883	.6995	.6841	-.5810
.4807	.5033	.2969	.6402	.6180	1.0300	-.4380	-.6210	-.7787	.1396	.1105	-.3105	-.7567	.6728	.6792	-.5703
-.8602	-.7887	-.4800	-.4807	-.7310	-.4380	1.0817	.8076	.6676	-.1500	-.3174	.6094	.7768	-.7602	-.7358	.6679
-.6970	-.6114	-.4073	-.6115	-.7067	-.6210	.8076	1.0256	.8903	-.3267	-.3079	.6062	.8515	-.7823	-.7062	.8766
-.5698	-.7019	-.2101	-.7713	-.7599	-.7787	.6676	.8903	1.0766	-.4485	-.3551	.5247	.9117	-.8920	-.8976	.8189
.1542	.1029	-.1067	.1513	.1342	.1396	-.1500	-.3267	-.4485	1.0618	.5884	-.1419	-.5480	.5769	.4804	-.4600
.1452	.1046	.1064	.1214	.0783	.1105	-.3174	-.3079	-.3551	.5884	1.0742	-.7251	-.4734	.5511	.4709	-.4947
-.6386	-.5947	-.3873	-.4080	-.3380	-.3105	.6094	.6062	.5247	-.1419	-.7251	1.0929	.5986	-.4402	-.4095	.7796
-.8021	-.7795	-.5865	-.7695	-.7883	-.7567	.7768	.8515	.9117	-.5480	-.4734	.5986	1.0959	-.9439	-.8970	.7462
.6025	.5891	.3898	.7615	.6995	.6728	-.7602	-.7823	-.8920	.5769	.5511	-.4402	-.9439	1.0672	1.0113	-.7086
.6034	.6222	.2296	.8324	.6841	.6792	-.7358	-.7062	-.8976	.4804	.4709	-.4095	-.8970	1.0113	1.0872	-.6304
-.7108	-.6003	-.2086	-.5618	-.5810	-.5703	.6679	.8766	.8189	-.4600	-.4947	.7796	.7462	-.7086	-.6304	1.0416

Note diagonal elements greater than one.

AFTER THE ABOVE PRODUCT MATRIX IS FORMED, THE DIAGONAL ELEMENTS ARE SET EQUAL TO 1.00 .

THE RESULTING "RANK CORRELATION MATRIX" IS GIVEN AT THE BEGINNING OF THE NEXT ITERATION.

New Approximate Correlation Matrix.

Note this off-diagonal element greater than one.

Figure 3

ITERATION NUMBER = 2 Beginning of Second Iteration

INPUT "RANK CORRELATION STRUCTURE" FOR THE RANDOM VARIABLES BEING TRANSFORMED

VARIABLE NUMBER

	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64
4	1.0000															
8	.9041	1.0000														
12	.6105	.5114	1.0000													
16	.6660	.6894	.4797	1.0000												
20	.7629	.7956	.4897	.6302	1.0000											
24	.4807	.5033	.2969	.6402	.6180	1.0000										
28	-.8602	-.7887	-.4800	-.4807	-.7310	-.4380	1.0000									
32	-.6970	-.6114	-.4073	-.6115	-.7067	-.6210	.8076	1.0000								
36	-.5698	-.7019	-.2101	-.7713	-.7599	-.7787	.6676	.8903	1.0000							
40	.1542	.1029	-.1067	.1513	.1342	.1396	-.1500	-.3267	-.4485	1.0000						
44	.1452	.1046	.1064	.1214	.0783	.1105	-.3174	-.3079	-.3551	.5884	1.0000					
48	-.6386	-.5947	-.3873	-.4080	-.3380	-.3105	.6094	.6062	.5247	-.1419	-.7251	1.0000				
52	-.8021	-.7795	-.5865	-.7695	-.7883	-.7567	.7768	.8515	.9117	-.5480	-.4734	.5986	1.0000			
56	.6025	.5891	.3898	.7615	.6995	.6728	-.7602	-.7823	-.8920	.5769	.5511	-.4402	-.9439	1.0000		
60	.6034	.6222	.2296	.8324	.6841	.6792	-.7358	-.7062	-.8976	.4804	.4709	-.4095	-.8970	1.0113	1.0000	
64	-.7108	-.6003	-.2086	-.5618	-.5810	-.5703	.6679	.8766	.8189	-.4600	-.4947	.7796	.7462	-.7086	-.6304	1.0000

Diagonal elements has been set equal to one.

Off-diagonal element is still greater than one.

Figure 4

ITERATION NUMBER = 12 Beginning of the 12th Iteration

INPUT "RANK CORRELATION STRUCTURE" FOR THE RANDOM VARIABLES BEING TRANSFORMED

VARIABLE NUMBER

	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64
4	1.0000															
8	.9011	1.0000														
12	.6143	.5115	1.0000													
16	.6535	.6882	.4708	1.0000												
20	.7533	.7944	.4891	.6359	1.0000											
24	.4780	.5084	.3009	.6499	.6216	1.0000										
28	-.8445	-.7779	-.4783	-.5106	-.7381	-.4467	1.0000									
32	-.6972	-.6295	-.4061	-.6116	-.7073	-.6281	.7956	1.0000								
36	-.6042	-.6828	-.2230	-.7638	-.7508	-.7756	.6711	.8698	1.0000							
40	.1452	.1014	-.0973	.1610	.1427	.1529	-.1652	-.3285	-.4468	1.0000						
44	.1590	.1146	.1047	.1352	.0711	.1158	-.3238	-.3186	-.3653	.5807	1.0000					
48	-.6257	-.5849	-.3889	-.3993	-.3483	-.3111	.6066	.5997	.5136	-.1584	-.6857	1.0000				
52	-.7834	-.7650	-.5555	-.7703	-.7847	-.7276	.7759	.8343	.8881	-.5082	-.4656	.5895	1.0000			
56	.6151	.5954	.3742	.7497	.6982	.6689	-.7356	-.7839	-.8866	.5688	.5317	-.4647	-.9375	1.0000		
60	.5964	.6270	.2551	.8002	.6842	.6756	-.7124	-.7134	-.8902	.4855	.4540	-.4202	-.8830	.9708	1.0000	
64	-.7010	-.6030	-.2108	-.5570	-.5780	-.5635	.6667	.8717	.8125	-.4501	-.4960	.7690	.7698	-.6925	-.6457	1.0000

The off-diagonal element has been reduced from .9999 to .9708.

Figure 5

ANALYSIS OF THE 16 BY 16 PRIME DIAGONAL SUBMATRIX

PERFORMANCE INDEX = .105

EIGENVALUES =

.0000 .0005 .0005 .0006 .0008 .0009 .1072 .1981 .2962 .4042 .4493 .6395 .7649 1.3453 1.9416 9.8499 } All eigenvalues are positive.

VECT. NO.	EIGENVECTORS															
1	-5.483E-02	-.124	-.213	-3.408E-02	-4.010E-02	-.178	4.849E-02	6.806E-02	.101	-.231	-7.517E-02	-1.408E-03	-.902	-7.064E-02	-8.456E-02	-2.391E-02
2	2.315E-02	.127	-1.563E-02	-.248	-7.462E-02	-9.049E-02	.269	.113	2.966E-02	-.153	-.220	-.134	5.506E-02	.826	-.187	-.110
3	.124	-9.783E-02	.266	-.397	-.125	-.166	.192	.264	-.416	-.127	-.143	2.072E-02	-5.751E-03	-.183	.544	-.239
4	.330	-.318	-3.995E-02	.100	-3.926E-02	-4.225E-02	-.180	.244	-.637	2.035E-03	7.170E-02	8.613E-03	3.573E-03	5.504E-02	-.465	.231
5	.404	.202	-8.860E-02	-9.333E-02	-.122	-2.201E-02	.366	-.352	-2.876E-02	-.270	.463	.460	1.538E-03	-2.323E-03	-2.180E-02	7.605E-02
6	.320	-.248	-.150	-9.592E-02	.131	3.481E-03	.284	-.308	6.796E-02	.116	-.196	-.456	-8.601E-04	-2.147E-02	.239	.538
7	.394	-.515	-9.412E-02	.151	.370	4.889E-02	3.915E-02	.267	.378	-8.869E-02	.105	9.747E-02	.111	7.794E-02	8.728E-02	-.372
8	.360	-2.247E-02	-6.588E-02	-.116	-.660	.404	-.351	9.240E-02	.249	9.259E-02	-.153	7.396E-02	-6.115E-02	5.873E-02	.117	3.731E-02
9	3.380E-02	.325	-.102	-.409	.349	.496	.115	.455	2.634E-02	9.642E-02	.217	-9.317E-02	-.105	-.125	-9.931E-02	.156
10	-.250	-.195	-.110	-.271	.150	-3.416E-03	-.512	-6.332E-02	-8.090E-03	-.465	.259	5.852E-02	8.271E-02	.236	.288	.304
11	-.101	-.374	.397	-.461	.128	.279	-1.943E-02	-.441	5.639E-03	.147	-.103	.163	-.108	-1.890E-02	-.310	-.155
12	.256	.223	-.203	-.316	.259	-.488	-.335	-2.648E-02	7.869E-02	.417	-.206	.317	-4.852E-03	1.893E-02	3.290E-03	3.288E-02
13	-1.503E-02	-6.456E-02	.627	.164	-1.757E-02	-.145	7.128E-02	.260	.205	.256	.221	.205	-.194	.226	.131	.422
14	-.218	-.108	-.279	.226	.136	.351	.193	1.362E-02	-.235	.128	-.364	.556	-4.962E-02	.177	.252	.149
15	-.251	-.269	-.349	-.142	-.239	-7.282E-02	.118	-1.469E-02	-.101	.534	.515	-.105	-4.144E-02	.167	.133	-.166
16	-.266	-.261	-.160	-.252	-.260	-.229	.269	.283	.292	-.127	-.139	.213	.311	-.291	-.282	.268

DETERMINANT = 7.249E-23

Note, the determinant is positive.

Figure 6

ANALYSIS OF THE 2 BY 2 PRIME DIAGONAL SUBMATRIX

PERFORMANCE INDEX = .102

EIGENVALUES =

.0989 1.9011

VECT. NO.	EIGENVECTORS	
1	-.707	.707
2	-.707	-.707

DETERMINANT = .188

Figure 7

```
ANALYSIS OF THE  3 BY  3 PRIME DIAGONAL SUBMATRIX
PERFORMANCE INDEX =  .255
EIGENVALUES =
  .0895  .5446 2.3659
VECT. NO.                    EIGENVECTORS

    1 -.740      .660      .128
    2 -.256     -.453      .854
    3  .622      .599      .504
DETERMINANT =   .115
```

Figure 8

V. Deck Setup for POSDEF

Following we give an example of how to set up the cards for use in running POSDEF. The first 10 cards are the usual job card, account card, and other control cards. This is followed by the Fortran program which is followed by the data (a listing of the program is given in Appendix A).

DECK SETUP FOR POSDEF

```
YPO,T5, STSSZ.
USER,NAMEXXX,PASWORD.
CHARGE,0188000.
FTN,R=2.
PFGET,SUBZ,OLDLIB,AU=ELFROST.
PFGET,IMSL,FXIMSL,AU=MATHLIB.
LIBRARY,SUBZ,IMSL.
LDSET,PRESET=NGINDEF.
LGO,PL=10000.
EXIT.
EXIT.
(END OF RECORD -- MULTI-PUNCH 789 IN COL 1)

    PROGRAM POSDEF GOES HERE

(END OF RECORD -- MULTI-PUNCH 789 IN COL 1)
```

[DATA AND INPUT-PARAMETER CARDS GO HERE]

(END OF RECORD -- MULTI-PUNCH 789 IN COL 1)

(END OF INFORMATION -- MULTI-PUNCH 6789 IN COL 1)

This completes the information needed to run POSDEF. An example of the data that was used in the example in section IV is given in Figure 9.

Figure 9

Input-Parameter Cards

Card No.																
1	16	1														
2	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64
3		1.00		0.90		1.00		0.60		0.50		1.00		0.70		0.70
4		0.50		1.00		0.80		0.80		0.50		0.60		1.00		0.50
5		0.50		0.30		0.60		0.60		1.00		-0.90		-0.80		-0.50
6		-0.40		-0.70		-0.40		1.00		-0.70		-0.60		-0.40		-0.60
7		-0.70		-0.60		0.80		1.00		-0.50		-0.70		-0.20		-0.80
8		-0.80		-0.80		0.70		0.90		1.00		0.20		0.10		-0.10
9		0.10		0.10		0.10		-0.10		-0.30		-0.50		1.00		0.10
10		0.10		0.10		0.10		0.10		0.10		-0.30		-0.30		-0.30
11		0.60		1.00		-0.70		-0.60		-0.40		-0.40		-0.30		-0.30
12		0.60		0.60		0.60		-0.10		-0.80		1.00		-0.80		-0.80
13		-0.60		-0.80		-0.80		-0.80		0.80		0.90		0.90		-0.60
14		-0.50		0.60		1.00		0.60		0.60		0.40		0.80		0.70
15		0.70		-0.80		-0.80		-0.90		0.60		0.60		-0.40		-0.90
16		1.00		0.60		0.60		0.20		0.90		0.70		0.70		-0.80
17		-0.70		-0.90		0.50		0.50		-0.40		-0.90		.9999		1.00
18		-0.70		-0.60		-0.20		-0.60		-0.60		-0.60		0.70		0.90
19		0.80		-0.50		-0.50		0.80		0.70		-0.70		-0.60		1.00

VI. Discussion

The user should recall that the input "approximate correlation matrix" can be ill-conditioned. These types of matrices frequently occur when a modeler places positive ones on the diagonal elements and then assigns correlations to the pairwise configurations of variables. While it is impossible to construct a negative definite matrix in this way, this matrix will have only a few negative eigenvalues. These will be small in absolute value (very near zero), provided this matrix is not ill-conditioned (i.e. one or more negative eigenvalues that are relatively large in absolute value such as -1, -2 or -3).

Recall that we have no guarantee that this algorithm will converge in such ill-conditioned cases, but we have never failed to see it converge. However, the final correlation matrix may be quite different from the original "approximate correlation matrix." We illustrate some of these principles with the following example.

Figure 10 gives the input "approximate rank correlation matrix" for 19 variables, labeled as indicated. As can be seen, the modeler indicates that variable numbers 1 through 11 are independent of each other. Also variable numbers 22 and 23 are each correlated negatively with each of variables 1-11. The indicated value of this correlation coefficient is -0.7. Finally the modeler indicates that variables 22 and 23 are independent.

A careful examination of these relationships will show that such an arrangement is essentially impossible. Consider the three hypothetical variables X_1, X_2 and X_3. Suppose $(X_1, X_2) = -0.7$ and furthermore suppose $(X_1, X_3) = -.8$. Then it is obvious that $(X_2, X_3) > 0$; that is variables X_2 and X_3 will have a strong positive correlation.

This matrix is ill-conditioned. There is only one negative eigenvalue but its value is -2.3571. However, the algorithm did converge after eight iterations. But it is not surprising that the values of the original matrix have been altered quite a bit. For example, the correlation value -0.7 becomes -0.4033; the zero correlation between variables 22 and 23 becomes 0.6451; and the zero correlations assigned to variables 1-11 all become 0.1214 (see Figure 11). This may or may not be acceptable to the modeler, but does indicate that care should be taken in the construction of the original "approximate correlation matrix." This simply illustrates that a collection of pairwise correlation coefficients does not make a correlation matrix. A multivariate distribution is much more complicated than this.

ITERATION NUMBER = 1

INPUT "RANK CORRELATION STRUCTURE" FOR THE RANDOM VARIABLES BEING TRANSFORMED

VARIABLE NUMBER

	1	2	3	4	5	6	7	8	9	10	11	22	23	24	25	26	27	28	29
1	1.0000																		
2	0.0000	1.0000																	
3	0.0000	0.0000	1.0000																
4	0.0000	0.0000	0.0000	1.0000															
5	0.0000	0.0000	0.0000	0.0000	1.0000														
6	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000													
7	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000												
8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000											
9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000										
10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000									
11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000								
22	-.7000	-.7000	-.7000	-.7000	-.7000	-.7000	-.7000	-.7000	-.7000	-.7000	-.7000	1.0000							
23	-.7000	-.7000	-.7000	-.7000	-.7000	-.7000	-.7000	-.7000	-.7000	-.7000	-.7000	0.0000	1.0000						
24	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	.7000	0.0000	1.0000					
25	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	.7000	0.0000	1.0000				
26	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000			
27	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	.7000	1.0000		
28	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	
29	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	.7000	1.0000

Note the zero correlations.

Note this zero correlation coefficient.

Figure 10

ITERATION NUMBER = 9

Algorithm converged on the 9th iteration.

INPUT "RANK CORRELATION STRUCTURE" FOR THE RANDOM VARIABLES BEING TRANSFORMED

VARIABLE NUMBER

	1	2	3	4	5	6	7	8	9	10	11	22	23	24	25	26	27	28	29
1	1.0000																		
2	.1214	1.0000																	
3	.1214	.1214	1.0000																
4	.1214	.1214	.1214	1.0000															
5	.1214	.1214	.1214	.1214	1.0000														
6	.1214	.1214	.1214	.1214	.1214	1.0000													
7	.1214	.1214	.1214	.1214	.1214	.1214	1.0000												
8	.1214	.1214	.1214	.1214	.1214	.1214	.1214	1.0000											
9	.1214	.1214	.1214	.1214	.1214	.1214	.1214	.1214	1.0000										
10	.1214	.1214	.1214	.1214	.1214	.1214	.1214	.1214	.1214	1.0000									
11	.1214	.1214	.1214	.1214	.1214	.1214	.1214	.1214	.1214	.1214	1.0000								
22	-.4033	-.4033	-.4033	-.4033	-.4033	-.4033	-.4033	-.4033	-.4033	-.4033	-.4033	1.0000							
23	-.4033	-.4033	-.4033	-.4033	-.4033	-.4033	-.4033	-.4033	-.4033	-.4033	-.4033	.6451	1.0000						
24	-.0602	-.0602	-.0602	-.0602	-.0602	-.0602	-.0602	-.0602	-.0602	-.0602	-.0602	.5013	-.0952	1.0000					
25	-.0602	-.0602	-.0602	-.0602	-.0602	-.0602	-.0602	-.0602	-.0602	-.0602	-.0602	-.0952	.5013	-.0029	1.0000				
26	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000			
27	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	.7000	1.0000		
28	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	
29	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	.7000	1.0000

All of these values were initially zero.

These were initially set equal to -0.7.

Was initially set equal to zero.

These variables are unchanged.

Figure 11

BIBLIOGRAPHY

Graybill, F. A. (1961). An Introduction to Linear Statistical Models, Volume 1, New York City, McGraw-Hill Book Company, Inc.

IMSL (1980). The IMSL, Volume 1, Ed. 8, Houston, Texas, International Mathematical and Statistical Libraries, Inc.

Iman, R. L. and Davenport, J. M. (1980). Rank Correlation Plots for Use with Correlated Input Variables in Simulation Studies. Tech Report SAND80-1903, Sandia Laboratories, Albuquerque, NM.

THIS PAGE LEFT BLANK INTENTIONALLY

APPENDIX A. Listing of the Fortran Source Deck for the POSDEF Program.

THIS PAGE LEFT BLANK INTENTIONALLY

```
      PROGRAM POSDEF(INPUT,OUTPUT,PUNCH,TAPE5)
      DIMENSION CORX(210), NK(20), D(20), Z(20,20), WK(210), CORZ(210)
C
C     NN = THE DIMENSIONS OF Z,  I.E.  Z(NN,NN).
      DATA NN/20/
C
C
C     EIG = THE VALUE THAT THE NEGATIVE EIGENVALUES ARE SET EQUAL TO.
      DATA EIG/0.001/
C
C
C     M = MAXIMUM NUMBER OF ITERATIONS ALLOWED.
      M=20
      M1=M+1
C
      ITEST=1
      READ 110, NP,IDIAG
      READ 120, (NK(I),I=1,NP)
      NKX=NP*(NP+1)/2
      READ 130, (CORX(I),I=1,NKX)
   10 PRINT 140
      IF (ITEST.GT.M) GO TO 20
      GO TO 30
   20 IF (IDIAG.EQ.0) GO TO 100
      PRINT 150
      GO TO 40
   30 PRINT 160, ITEST
   40 PRINT 170
      PRINT 180
      PRINT 190, (NK(I),I=1,NP)
      II=1
      IK=1
      DO 50 I=1,NP
      PRINT 200, NK(I),(CORX(J),J=II,IK)
      II=IK+1
      IK=IK+I+1
   50 CONTINUE
      IL=NP
      IF (ITEST.EQ.M1) IL=2
      DO 80 K=IL,NP
      CALL EIGRS (CORX,K,2,D,Z,NN,WK,IER)
      PRINT 210, K,K
      PRINT 220, WK(1)
      PRINT 230, (D(I),I=1,K)
      PRINT 240
      DO 60 J=1,K
   60 PRINT 250, J,(Z(I,J),I=1,K)
      DET=1.0
      DO 70 J=1,K
   70 DET=DET*D(J)
```

```
      PRINT 260, DET
   80 CONTINUE
      IF (ITEST.GE.M1) GO TO 100
      CALL FINDIT (NP,D,Z,CORZ,ITEST,M,EIG)
      ITEST=ITEST+1
      DO 90 I=1,NKX
   90 CORX(I)=CORZ(I)
      GO TO 10
  100 CONTINUE
      CALL EXIT
C
  110 FORMAT (2I5)
  120 FORMAT (16I5)
  130 FORMAT (8G10.4)
  140 FORMAT (1H1)
  150 FORMAT (76H0 AFTER THE LAST ITERATION, WE EXAMINE EACH P BY P PRIM
     1E DIAGONAL SUBMATRIX.)
  160 FORMAT (21H0 ITERATION NUMBER = ,I5)
  170 FORMAT (////,1X,77HINPUT "RANK CORRELATION STRUCTURE" FOR THE RAND
     1OM VARIABLES BEING TRANSFORMED,//)
  180 FORMAT (20X,15HVARIABLE NUMBER,/)
  190 FORMAT (6X,17I7,/,8X,17I7,/,8X,17I7)
  200 FORMAT (1H0,I5,17(F7.4),/,7X,17(F7.4),/,17(F7.4))
  210 FORMAT (16H1ANALYSIS OF THE,I3,3H BY,I3,25H PRIME DIAGONAL SUBMATR
     1IX)
  220 FORMAT (1H0,19HPERFORMANCE INDEX =,1PG10.3)
  230 FORMAT (14H0EIGENVALUES =,/,1H0,17(F7.4),/,17(F7.4))
  240 FORMAT (11H0VECT. NO. ,20X,12HEIGENVECTORS,/)
  250 FORMAT (1H0,I5,12(1PG10.3),/,6X,12(1PG10.3))
  260 FORMAT (15H0DETERMINANT = ,1PG10.3)
      END


      SUBROUTINE FINDIT (NP,D,Z,CORZ,ITEST,M,EIG)
      DIMENSION D(1), Z(20,1), CORZ(1), X(20,20), T(20,20)
      NEV=0
      DO 10 I=1,NP
      IF (D(I).LT.0.0) NEV=NEV+1
   10 CONTINUE
      IF (NEV.EQ.0) GO TO 40
C
C     EIGENVALUE ADJUSTMENT SCHEME
C
      DO 20 I=1,NEV
      D(I)=EIG
   20 CONTINUE
      L1=NEV+1
      L2=NEV+NEV
      DO 30 I=L1,L2
      IF (D(I).LT.EIG) D(I)=EIG
```

```
   30 CONTINUE
C
C     END OF SCHEME
C
      PRINT 140, (D(I),I=1,NP)
      GO TO 60
   40 PRINT 150, ITEST
      IF (ITEST.EQ.1) GO TO 60
   50 CONTINUE
      ITEST=M
      GO TO 130
   60 CONTINUE
      DO 70 J=1,NP
      DO 70 I=1,NP
   70 T(I,J)=Z(I,J)*D(J)
      DO 80 I=1,M
      DO 80 J=1,M
   80 X(I,J)=0.0
      DO 100 I=1,NP
      DO 100 J=1,NP
      DO 90 K=1,NP
   90 X(I,J)=X(I,J)+T(I,K)*Z(J,K)
  100 CONTINUE
      PRINT 170
      NE2=NEV*2
      PRINT 160, NE2,EIG
      DO 110 I=1,NP
  110 PRINT 180, (X(I,J),J=1,NP)
      CALL SCALEIT (NP,X)
      KI=0
      DO 120 I=1,NP
      DO 120 J=1,I
      KI=KI+1
  120 CORZ(KI)=X(I,J)
      IF (ITEST.EQ.1.AND.NEV.EQ.0) GO TO 50
  130 RETURN
C
  140 FORMAT (31HOEIGENVALUES AFTER ADJUSTMENT =,/,1HO,17(F7.4),/,17(F7.
     14))
  150 FORMAT (56H1 ALL THE EIGENVALUES WERE POSITIVE ON ITERATION NUMBER
     1 ,I5,2H .,/)
  160 FORMAT (42HORESULTING PRODUCT MATRIX AFTER THE FIRST ,I5,59H EIGEN
     1VALUES HAVE BEEN TESTED AND ADJUSTED WHERE NECESSARY.,/,1X,26HMINI
     2MUM EIGENVALUE USED = ,G12.4,/)
  170 FORMAT (1H1)
  180 FORMAT (1HO,17(F7.4),/,17(F7.4))
      END
```

```
      SUBROUTINE SCALEIT(NP,X)
      DIMENSION X(20,1)
      DO 10 I=1,NP
   10 X(I,I)=1.0
      PRINT 20
      PRINT 30
      RETURN
C
   20 FORMAT (//,1X,87HAFTER THE ABOVE PRODUCT MATRIX IS FORMED, THE DIA
     1GONAL ELEMENTS ARE SET EQUAL TO 1.00 .)
   30 FORMAT (/,1X,88HTHE RESULTING "RANK CORRELATION MATRIX" IS GIVEN A
     1T THE BEGINNING OF THE NEXT ITERATION.)
      END
```