



Munich Personal RePEc Archive

**A note on the sub-optimality of rank
ordering of objects on the basis of the
leading principal component factor scores**

SK Mishra

North-Eastern Hill University, Shillong (India)

30. December 2008

Online at <http://mpra.ub.uni-muenchen.de/12419/>

MPRA Paper No. 12419, posted 31. December 2008 06:34 UTC

A note on the sub-optimality of rank ordering of objects on the basis of the leading principal component factor scores

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

1. The Problem: Suppose we intend to optimally rank order n objects (candidates) each of which has m attributes or rank scores awarded by m evaluators. More explicitly, let m evaluators award ranking scores to n ($n > m$) candidates and let the array of ranking scores be denoted by X in n rows and m columns. The objective is to summarize X by a single n -element column vector of ranking scores, Z , such that, in some sense, Z is the optimal representation of X . Among many possible criteria of representation, the one could follow the principle that the sum of squares of the product moment coefficients of correlation between Z and $x_j \in X$ is maximum or, stated symbolically, $\sum_{j=1}^m r^2(Z, x_j)$ is maximum.

Conventionally, this problem is solved by the principal component analysis (Kendall and Stuart, 1968). It consists of obtaining $Z = \mathfrak{R}(Y) : Y = Xw; \max \sum_{j=1}^m r^2(Y, x_j)$, where the transformation $\mathfrak{R}(Y)$ assigns ranking scores to the elements of Y by any one of the ranking rules such as (a) standard competition ranking or the 1-2-2-4 rule, (b) modified competition ranking or 1-3-3-4 rule, (c) dense ranking or 1-2-2-3 rule, (d) ordinal ranking or 1-2-3-4 rule, (e) fractional ranking or 1-2.5-2.5-4 rule, etc (Wikipedia, 2008). The crux of the problem is, however, that optimality of Y does not entail optimality of Z or, stated differently, $Z = \mathfrak{R}(Y) : Y = Xw; \max \sum_{j=1}^m r^2(Y, x_j)$ does not necessarily ensure the optimality of Z with respect to the constituent variables. If $Z = \mathfrak{R}(Y) : Y = Xw; \max \sum_{j=1}^m r^2(Y, x_j)$ then oftentimes Z turns out to be only a suboptimal representation of X .

The procedure of obtaining the ranking scores $Z = \mathfrak{R}(Y) : Y = Xw; \max \sum_{j=1}^m r^2(Y, x_j)$ follows two straightforward steps. First, the Principal component analysis is run on X to obtain Y , which is the best linear combination, Xw , of X . It lies in (i) computing the inter-correlation matrix, $R(m, m)$, from X such that $r_{ij} \in R; r_{ij} = \text{cov}(x_i, x_j) / \sqrt{\text{var}(x_i) \text{var}(x_j)} \forall i, j$, (ii) finding the normed eigenvector, w , of R associated with its largest eigenvalue, λ_1 and (iii) using w to obtain $Y = X^*w$, where $x_{ij}^* = (x_{ij} - \bar{x}_j) / \sqrt{\text{var}(x_j)}$. Next, $Z = \mathfrak{R}(Y)$ is obtained. It is well known that this procedure maximizes $\sum_{j=1}^m r^2(Y, x_j)$. The direct maximization of $\sum_{j=1}^m r^2(Z, x_j)$ is a rare practice.

However, what we seek is not the Z that maximizes $\sum_{j=1}^m r^2(Y, x_j)$ but, instead, the Z that maximizes $\sum_{j=1}^m r^2(Z, x_j)$. This is not guaranteed by the conventional Principal Component Analysis. This is what we demonstrate in this paper.

2. The Objective: Our objective in this paper is to show, by means of a few numerical examples, that the rank ordering of objects (candidates) by $Z = \mathfrak{R}(Y)$, where $Y = Xw$ obtained by

maximizing $\sum_{j=1}^m r^2(Y, x_j)$, is suboptimal in the sense that it may not maximize $\sum_{j=1}^m r^2(Z, x_j)$. The rank ordering of objects by direct maximization of $\sum_{j=1}^m r^2(Z, x_j)$ is a better alternative. However, the conventional procedure of Principal Component analysis does not provide any scope for the same. There is no ready-to-use software available to maximize $\sum_{j=1}^m r^2(Z, x_j)$.

3. Materials and Methods: We simulate ranking score awarded to 30 candidates (objects) by 7 evaluators. The dataset, X , may also be viewed as the ranking scores obtained by 30 objects on each of their 7 attributes. First, X is subjected to the Principal Component analysis and the composite score, $Y = Xw$, is obtained. This composite score relates to the first principal component associated with the largest eigenvalue of the correlation matrix of X . Therefore, Y maximizes $\sum_{j=1}^m r^2(Y, x_j)$. Then Y is rank ordered according to the 1234 ordinal ranking rule to obtain Z_1 , the overall ranking scores for the candidates (objects). The inter-correlation matrix, R_1 , is computed for $[Z_1 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7]$ and, consequently, the measure of representativeness of Z_1 , namely $F_1 = \sum_{j=1}^7 r^2(Z_1, x_j)$, is computed. It may be noted that in this scheme $\sum_{j=1}^m r^2(Y, x_j)$ rather than $F_1 = \sum_{j=1}^7 r^2(Z_1, x_j)$ is maximized.

In the second scheme, $F_2 = \sum_{j=1}^7 r^2(Z_2, x_j)$, an alternative measure of representativeness, is directly maximized, where $Z_2 = \Re(Y')$: $Y' = Xv$. In finding $Z_2 = \Re(Y')$ the 1234 ordinal ranking rule is applied. As before, the inter-correlation matrix, R_2 , is computed for $[Z_2 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7]$. The method of Differential Evolution has been used for maximization of these alternative measures of representativeness, F_1 and F_2 .

It would be pertinent to provide an introduction to the Differential Evolution (DE) method of optimization. The DE is one of the most recently invented methods of global optimization that has been very successful in optimization of extremely difficult multimodal functions. The DE is a population-based stochastic search method of optimization grown out of the Genetic algorithms. The crucial idea behind DE is a scheme for generating trial parameter vectors. Initially, a population of points (p in d -dimensional space) is generated and evaluated (i.e. $f(p)$ is obtained) for their fitness. Then for each point (p_i) three different points (p_a , p_b and p_c) are randomly chosen from the population. A new point (p_z) is constructed from those three points by adding the weighted difference between two points ($w(p_b - p_c)$) to the third point (p_a). Then this new point (p_z) is subjected to a crossover with the current point (p_i) with a probability of crossover (c_r), yielding a candidate point, say p_u . This point, p_u , is evaluated and if found better than p_i then it replaces p_i else p_i remains. Thus we obtain a new vector in which all points are either better than or as good as the current points. This new vector is used for the next iteration. This process makes the differential evaluation scheme completely self-organizing. Operationally, this method consists of three basic steps: (i) generation of (large enough) population with N individuals [$u = (u_1, u_2, \dots, u_m)$] in the m -dimensional space, randomly distributed over the entire domain of the function in question and evaluation of the individuals of the so generated by finding $f(u)$; (ii) replacement of this current population by a better fit new population, and (iii) repetition of this replacement until satisfactory results are obtained or certain criteria of termination are met. The crux of the problem lays in replacement of the

current population by a new population that is better fit. In this context, the meaning of ‘better’ is in the Pareto improvement sense. A set S_a is better than another set S_b iff : (i) no $u_i \in S_a$ is inferior to the corresponding member of $u_i \in S_b$; and (ii) at least one member $u_k \in S_a$ is better than the corresponding member $u_k \in S_b$. Thus, every new population is an improvement over the earlier one. To accomplish this, the DE procedure generates a candidate individual to replace each current individual in the population. The candidate individual is obtained by a crossover of the current individual and three other randomly selected individuals from the current population. The crossover itself is probabilistic in nature. Further, if the candidate individual is better fit than the current individual, it takes the place of the current individual, else the current individual stays and passes into the next iteration (Mishra, 2006).

It may further be noted that we have maximized $\sum_{j=1}^m r^2(Y, x_j)$ as well as $\sum_{j=1}^7 r^2(Z_2, x_j)$ by DE. We have compared our results of maximization of $\sum_{j=1}^m r^2(Y, x_j)$ with those obtained from STATISTICA (which has a built in facility to find the principal component-based factor scores and other related measures such as the eigenvalues, factor loadings, etc). Our results are essentially identical to those obtained from STATISTICA. This comparison ensures that our efforts in direct optimization have been perfectly successful. However, STATISTICA does not have a provision to maximize $\sum_{j=1}^7 r^2(Z_2, x_j)$. But since in the examples shown below F_2 is larger than F_1 , we conclude that rank ordering of objects by Z_1 is suboptimal.

Example-1: The simulated dataset (X) on ranking scores of 30 candidates awarded by 7 evaluators, the results obtained by running the principal component algorithm and the overall rankings based on the same (Y and Z_1) and the results of rank order optimization exercise based on our method (Y' and Z_2) are presented in Table-1.1. In table-1.2 are presented the inter-correlation matrix, R_1 , for the variables [Z_1 x_1 x_2 x_3 x_4 x_5 x_6 x_7]. The last two rows of Table-1.2 are the weight (w) vector used to obtain $Y = Xw$ and factor loadings, that is, $r(Y, x_j)$. The sum of squares of factor loadings (S_1) = 4.352171. the measure of representativeness of Z_1 that is $F_1 = \sum_{j=1}^7 r^2(Z_1, x_j) = 4.287558$.

In Table-1.3 we have presented the inter-correlation matrix, R_2 , for variables [Z_2 x_1 x_2 x_3 x_4 x_5 x_6 x_7], weights and the factor loadings when the same dataset (as mentioned above) is subjected to the direct maximization of $\sum_{j=1}^7 r^2(Z_2, x_j)$. The weights and the factor loadings relate to $Y' = Xv$ and $r(Z_2, x_j)$. The sum of square of factor loadings (S_2)= 4.287902 and the measure of representativeness of Z_2 that is $F_2 = \sum_{j=1}^7 r^2(Z_2, x_j)$ also is 4.287902. Since $F_2 > F_1$, the sub-optimality of the PC-based F_1 for this dataset is demonstrated. Notably, the candidates #8, #20, #21 and #26 are rank ordered differently by the two methods.

Example-2: The simulated data and Y , Y' , Z_1 and Z_2 for this dataset are presented in Table-2.1. The inter-correlation matrices, R_1 and R_2 and the associated weights and factor loadings also are presented in Tables-2.2 and 2.3. The values of F_1 and F_2 for this dataset are 3.120505 and

3.124149 respectively. This also shows the sub-optimality of the PC-based F_1 . The candidates #8, #16, #18, #21, #23, #27, #28 and #29 are rank ordered differently by the two methods.

Example-3: One more simulated dataset and Y , Y' , Z_1 and Z_2 for this dataset are presented in Table-3.1. The inter-correlation matrices, R_1 and R_2 and the associated weights and factor loadings also are presented in Tables-3.2 and 3.3. The values of F_1 and F_2 for this dataset are 2.424195 and 2.426101 respectively. Once again, it is demonstrated that the PC-based F_1 is sub-optimal. The candidates #8, #13, #15, #19, and #26 are rank ordered differently by the two methods.

4. Conclusion: The three numerical examples suffice to demonstrate that the Principal Component based rankings scores, Z_1 , obtained by maximization of $\sum_{j=1}^m r^2(Y, x_j) : Y = Xw$ are not the best rank order scores and do not optimally represent the dataset from which they have been derived. Its alternative is to obtain Z_2 by the direct optimization of $\sum_{j=1}^m r^2(Z_2, x_j) = \sum_{j=1}^m r^2(\Re(Y'), x_j) : Y' = Xv$. Although this direct optimization problem is extremely nonlinear and complicated, it can be accomplished by the use of advanced methods such as the Differential Evaluation method of global optimization.

References

Kendall, M.G. and Stuart, A. (1968): *The Advanced Theory of Statistics*, vol. 3, Charles Griffin & Co. London.

Mishra, S.K. (2006) "Global Optimization by Differential Evolution and Particle Swarm Methods: Evaluation on Some Benchmark Functions", SSRN: <http://ssrn.com/abstract=933827>

Wikipedia (2008) "Ranking" at Wikipedia, http://en.wikipedia.org/wiki/Rank_order

Note: The Fortran Computer program for this exercise may also be obtained from mishrasknehu@yahoo.com

Table-1.1: Dataset Relating to Example-1 Showing Sub-optimality of PC-based Rank-ordering of Objects

Sl. No.	Ranking Scores of 30 candidates awarded by Seven Evaluators							Composite Score (Y) Optimized Results		Rank-Order (Z ₂) Optimized Results	
	X ₁	X ₂	X ₃	X ₄	X ₅	X ₆	X ₇	Y	Z ₁	Y'	Z ₂
1	1	10	3	1	1	6	8	11.22414	3	11.31748	3
2	4	9	12	14	11	5	1	21.21812	5	20.12333	5
3	28	18	20	25	27	15	30	61.89764	26	61.72889	26
4	23	29	15	18	30	17	29	60.44481	25	60.21783	25
5	11	19	18	26	20	23	26	54.18963	22	52.19204	22
6	26	27	28	24	29	28	18	67.29609	28	64.74165	28
7	18	25	30	21	16	18	24	57.60993	24	57.02551	24
8	8	16	9	15	15	27	12	37.83836	12	34.81273	11
9	5	21	26	23	23	9	15	46.22697	19	44.92322	19
10	16	17	11	16	14	20	19	42.55673	16	41.42991	16
11	22	15	21	20	17	19	13	47.86480	20	46.33942	20
12	25	12	22	22	19	30	21	56.74565	23	54.07614	23
13	15	23	16	27	10	8	14	43.60476	17	43.97609	17
14	21	4	25	9	22	16	16	42.06427	15	40.35769	15
15	24	26	27	28	13	29	25	65.25874	27	63.76723	27
16	29	24	29	30	21	24	28	70.25956	30	69.24598	30
17	3	8	13	8	3	13	17	24.69261	7	23.98144	7
18	12	30	14	12	12	14	11	39.33102	14	39.07950	14
19	14	1	5	3	2	1	9	13.52068	4	14.54720	4
20	17	5	7	17	8	26	20	37.82410	11	35.84812	12
21	2	3	1	2	4	12	4	10.12451	2	8.71359	1
22	20	28	8	13	25	21	23	51.38784	21	50.56218	21
23	10	7	6	7	9	22	5	24.16558	6	21.65172	6
24	9	14	17	5	18	2	2	24.73667	8	24.30123	8
25	30	20	23	19	6	11	6	43.86090	18	44.38925	18
26	6	2	2	4	7	3	3	10.08933	1	9.77266	2
27	13	13	10	11	28	7	22	38.94057	13	38.59856	13
28	19	6	19	6	5	10	7	27.10184	10	26.86406	10
29	27	22	24	29	26	25	27	68.06344	29	66.42601	29
30	7	11	4	10	24	4	10	26.02995	9	25.28209	9

Table-1.2: Inter-Correlation Matrix, Weights and Factor Loadings of Composite Score Optimized Overall Ranking Scores for the Dataset in Example-1 (F₁=4.287558; S₁= 4.352171)

	Z ₁	X ₁	X ₂	X ₃	X ₄	X ₅	X ₆	X ₇
Z ₁	1.000000	0.805117	0.781980	0.801112	0.891880	0.690768	0.658287	0.824694
X ₁	0.805117	1.000000	0.474082	0.666741	0.645384	0.451390	0.537709	0.597330
X ₂	0.781980	0.474082	1.000000	0.554616	0.688543	0.552392	0.409121	0.569299
X ₃	0.801112	0.666741	0.554616	1.000000	0.731257	0.438487	0.426919	0.491880
X ₄	0.891880	0.645384	0.688543	0.731257	1.000000	0.526140	0.606229	0.708120
X ₅	0.690768	0.451390	0.552392	0.438487	0.526140	1.000000	0.324583	0.630256
X ₆	0.658287	0.537709	0.409121	0.426919	0.606229	0.324583	1.000000	0.608009
X ₇	0.824694	0.597330	0.569299	0.491880	0.708120	0.630256	0.608009	1.000000
Weights		0.381563	0.369979	0.377237	0.430743	0.337585	0.337890	0.401926
Loadings		0.796012	0.771845	0.786986	0.898611	0.704264	0.704901	0.838493

	Z ₂	X ₁	X ₂	X ₃	X ₄	X ₅	X ₆	X ₇
Z ₂	1.000000	0.810901	0.776641	0.800667	0.893660	0.688988	0.653838	0.827809
X ₁	0.810901	1.000000	0.474082	0.666741	0.645384	0.451390	0.537709	0.597330
X ₂	0.776641	0.474082	1.000000	0.554616	0.688543	0.552392	0.409121	0.569299
X ₃	0.800667	0.666741	0.554616	1.000000	0.731257	0.438487	0.426919	0.491880
X ₄	0.893660	0.645384	0.688543	0.731257	1.000000	0.526140	0.606229	0.708120
X ₅	0.688988	0.451390	0.552392	0.438487	0.526140	1.000000	0.324583	0.630256
X ₆	0.653838	0.537709	0.409121	0.426919	0.606229	0.324583	1.000000	0.608009
X ₇	0.827809	0.597330	0.569299	0.491880	0.708120	0.630256	0.608009	1.000000
Weights		0.441363	0.418945	0.360646	0.411243	0.274724	0.203625	0.462126
Loadings		0.810901	0.776641	0.800667	0.89366	0.688988	0.653838	0.827809

Sl No.	Ranking Scores of 30 candidates awarded by Seven Evaluators							Composite Score (Y) Optimized Results		Rank-Order (Z ₂) Optimized Results	
	X ₁	X ₂	X ₃	X ₄	X ₅	X ₆	X ₇	Y	Z ₁	Y'	Z ₂
1	10	5	10	7	5	18	18	27.96401	9	27.66608	9
2	23	25	30	13	25	24	22	61.09793	26	60.93414	26
3	11	4	15	3	3	12	3	20.05915	3	19.25318	3
4	29	7	24	30	29	28	21	65.03329	27	64.65080	27
5	12	13	2	4	6	23	14	27.23433	8	27.55713	8
6	3	1	6	11	11	15	1	18.47398	2	18.61630	2
7	17	27	9	18	1	14	25	39.62596	13	40.64115	13
8	20	10	18	28	17	21	10	47.14993	20	47.27035	19
9	24	19	19	12	23	6	13	43.75459	17	43.78827	17
10	22	21	26	19	30	27	30	66.20587	28	66.35746	28
11	13	12	12	16	24	30	12	44.98232	18	45.39889	18
12	21	24	7	23	10	19	29	48.43576	24	49.56098	24
13	5	2	16	9	15	2	16	25.42514	7	25.15835	7
14	27	18	21	29	18	16	15	54.18113	25	54.42185	25
15	15	23	11	1	4	7	8	24.68224	6	24.91456	6
16	16	15	28	6	20	20	4	41.87725	16	41.18156	15
17	4	16	1	2	2	5	6	11.99989	1	12.74124	1
18	6	26	25	21	14	13	26	47.83244	21	48.71014	22
19	25	9	4	5	8	11	23	32.41507	11	32.13677	11
20	18	11	3	10	21	10	19	34.62240	12	35.12062	12
21	7	30	20	15	27	9	24	47.94087	22	49.26167	23
22	30	22	29	20	26	29	28	69.86155	30	69.59827	30
23	26	8	14	14	9	17	20	41.57229	15	41.00634	14
24	9	20	8	26	13	1	5	29.02329	10	30.49082	10
25	1	6	13	25	12	4	2	23.40574	5	24.09491	5
26	28	17	23	27	28	26	27	66.86960	29	66.99722	29
27	14	29	27	22	19	8	9	46.79227	19	47.55844	20
28	8	28	17	24	7	25	7	41.44998	14	42.63972	16
29	19	14	22	17	16	22	17	48.27078	23	48.05095	21
30	2	3	5	8	22	3	11	20.66025	4	21.15313	4

Table-2.2: Inter-Correlation Matrix, Weights and Factor Loadings of Composite Score Optimized Overall Ranking Scores for the Dataset in Example-2 ($F_1=3.120505$; $S_1= 3.215147$)								
	Z_1	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Z_1	1.000000	0.726363	0.499889	0.739266	0.636040	0.696997	0.650723	0.694772
X_1	0.726363	1.000000	0.189766	0.419355	0.315684	0.393548	0.554616	0.568409
X_2	0.499889	0.189766	1.000000	0.388654	0.295217	0.125695	0.126585	0.353949
X_3	0.739266	0.419355	0.388654	1.000000	0.439377	0.617798	0.426919	0.276085
X_4	0.636040	0.315684	0.295217	0.439377	1.000000	0.417575	0.314794	0.245384
X_5	0.696997	0.393548	0.125695	0.617798	0.417575	1.000000	0.348165	0.370857
X_6	0.650723	0.554616	0.126585	0.426919	0.314794	0.348165	1.000000	0.409566
X_7	0.694772	0.568409	0.353949	0.276085	0.245384	0.370857	0.409566	1.000000
Weights		0.417835	0.262744	0.427355	0.351103	0.398451	0.384622	0.379179
Loadings		0.749214	0.471122	0.766283	0.629557	0.714456	0.689659	0.679899

Table-2.3: Inter-Correlation Matrix, Weights and Factor Loadings of Rank Order Optimized Overall Ranking Scores for the Dataset in Example-2 ($F_2=3.124149$; $S_2=3.124149$)								
	Z_2	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Z_2	1.000000	0.701001	0.535484	0.740156	0.646719	0.695217	0.640934	0.696997
X_1	0.701001	1.000000	0.189766	0.419355	0.315684	0.393548	0.554616	0.568409
X_2	0.535484	0.189766	1.000000	0.388654	0.295217	0.125695	0.126585	0.353949
X_3	0.740156	0.419355	0.388654	1.000000	0.439377	0.617798	0.426919	0.276085
X_4	0.646719	0.315684	0.295217	0.439377	1.000000	0.417575	0.314794	0.245384
X_5	0.695217	0.393548	0.125695	0.617798	0.417575	1.000000	0.348165	0.370857
X_6	0.640934	0.554616	0.126585	0.426919	0.314794	0.348165	1.000000	0.409566
X_7	0.696997	0.568409	0.353949	0.276085	0.245384	0.370857	0.409566	1.000000
Weights		0.383664	0.31504	0.378173	0.385789	0.414933	0.379452	0.381511
Loadings		0.701001	0.535484	0.740156	0.646719	0.695217	0.640934	0.696997

Table-3.1: Dataset Relating to Example-3 Showing Sub-optimality of PC-based Rank-ordering of Objects											
Sl No.	Ranking Scores of 30 candidates awarded by Seven Evaluators							Composite Score (Y) Optimized Results		Rank-Order (Z_2) Optimized Results	
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	Y	Z_1	Y'	Z_2
1	4	5	27	18	2	20	11	32.10776	10	31.59814	10
2	8	28	30	29	17	29	13	59.10295	26	58.52745	26
3	2	22	13	3	1	1	20	22.97097	3	21.15144	3
4	24	23	9	26	30	25	28	62.42732	28	61.62047	28
5	25	25	22	5	22	6	1	39.50937	18	40.81833	18
6	27	30	16	24	28	27	30	68.43141	30	67.46975	30
7	22	2	19	12	16	4	16	32.23785	11	32.42260	11
8	3	8	20	2	9	21	18	30.81457	6	30.37723	8
9	18	19	8	21	8	14	7	35.11395	14	34.68187	14
10	14	6	15	4	15	3	29	30.98154	7	29.98579	7
11	26	4	4	1	7	7	3	17.61105	1	18.44842	1
12	21	7	21	14	4	22	10	35.41392	15	35.50598	15
13	1	20	7	16	5	13	21	31.73604	9	29.76353	6

14	19	3	2	8	14	2	2	17.95374	2	18.75359	2
15	10	10	14	11	20	15	22	38.67293	17	38.17462	16
16	13	11	18	22	26	11	23	46.74022	20	46.21905	20
17	20	12	17	20	6	10	8	33.26055	13	33.02942	13
18	16	13	29	27	21	8	17	48.42594	21	48.12730	21
19	12	17	6	7	11	16	14	31.43870	8	30.92993	9
20	6	18	5	28	3	5	9	27.61333	5	26.12312	5
21	29	27	23	30	24	18	27	65.74003	29	64.76105	29
22	30	26	26	9	19	19	19	54.32872	25	54.40519	25
23	15	1	11	17	13	17	15	32.69124	12	32.54366	12
24	28	16	10	23	23	9	26	49.29651	22	48.46761	22
25	5	29	3	19	10	23	25	44.09388	19	41.91719	19
26	9	9	12	6	29	24	6	37.86774	16	39.22546	17
27	7	14	1	15	12	12	4	25.41235	4	25.27022	4
28	17	15	25	25	18	28	12	52.66056	24	52.81574	24
29	23	21	28	13	25	26	24	59.89362	27	59.87255	27
30	11	24	24	10	27	30	5	51.48540	23	52.63350	23

Table-3.2: Inter-Correlation Matrix, Weights and Factor Loadings of Composite Score Optimized Overall Ranking Scores for the Dataset in Example-3
($F_1=2.424195$; $S_1=2.521362$)

	Z ₁	X ₁	X ₂	X ₃	X ₄	X ₅	X ₆	X ₇
Z ₁	1.000000	0.473637	0.612013	0.566185	0.562180	0.741046	0.654283	0.459844
X ₁	0.473637	1.000000	0.108788	0.187542	0.132814	0.470078	-0.006007	0.119911
X ₂	0.612013	0.108788	1.000000	0.163070	0.360623	0.303226	0.402447	0.309010
X ₃	0.566185	0.187542	0.163070	1.000000	0.180868	0.296552	0.389544	0.126585
X ₄	0.562180	0.132814	0.360623	0.180868	1.000000	0.200000	0.284983	0.289433
X ₅	0.741046	0.470078	0.303226	0.296552	0.200000	1.000000	0.384650	0.315684
X ₆	0.654283	-0.006007	0.402447	0.389544	0.284983	0.384650	1.000000	0.156396
X ₇	0.459844	0.119911	0.309010	0.126585	0.289433	0.315684	0.156396	1.000000
Weights		0.271198	0.414408	0.345937	0.366429	0.458344	0.416812	0.341998
Loadings		0.430630	0.658031	0.549306	0.581846	0.727796	0.661847	0.543052

Table-3.3: Inter-Correlation Matrix, Weights and Factor Loadings of Rank Order Optimized Overall Ranking Scores for the Dataset in Example-3
($F_2=2.426101$; $S_2=2.426101$)

	Z ₂	X ₁	X ₂	X ₃	X ₄	X ₅	X ₆	X ₇
Z ₂	1.000000	0.479867	0.599555	0.576418	0.543493	0.751279	0.666741	0.446941
X ₁	0.479867	1.000000	0.108788	0.187542	0.132814	0.470078	-0.006007	0.119911
X ₂	0.599555	0.108788	1.000000	0.163070	0.360623	0.303226	0.402447	0.309010
X ₃	0.576418	0.187542	0.163070	1.000000	0.180868	0.296552	0.389544	0.126585
X ₄	0.543493	0.132814	0.360623	0.180868	1.000000	0.200000	0.284983	0.289433
X ₅	0.751279	0.470078	0.303226	0.296552	0.200000	1.000000	0.384650	0.315684
X ₆	0.666741	-0.006007	0.402447	0.389544	0.284983	0.384650	1.000000	0.156396
X ₇	0.446941	0.119911	0.309010	0.126585	0.289433	0.315684	0.156396	1.000000
Weights		0.297447	0.386881	0.363218	0.338013	0.508040	0.430645	0.268531
Loadings		0.479867	0.599555	0.576418	0.543493	0.751279	0.666741	0.446941

```

1: C   MAIN PROGRAM : PROVIDES TO USE DIFFERENTIAL EVOLUTION METHOD TO
2: C   COMPUTE COMPOSITE INDEX INDICES
3: C   BY MAXIMIZING SUM OF (SQUARES, OR ABSOLUTES, OR MINIMUM) OF
4: C   CORRELATION OF THE INDEX WITH THE CONSTITUENT VARIABLES. THE MAX
5: C   SUM OF SQUARES IS THE PRINCIPAL COMPONENT INDEX. IT ALSO PROVIDES
6: C   TO OBTAIN MAXIMUM ENTROPY ABSOLUTE CORRELATION INDICES.
7: C   PRODUCT MOMENT AS WELL AS ABSOLUTE CORRELATION (BRADLEY, 1985) MAY
8: C   BE USED. PROGRAM BY SK MISHRA, DEPT. OF ECONOMICS, NORTH-EASTERN
9: C   HILL UNIVERSITY, SHILLONG (INDIA)
10: C   -----
11: C   ADJUST THE PARAMETERS SUITABLY IN SUBROUTINES DE
12: C   WHEN THE PROGRAM ASKS FOR PARAMETERS, FEED THEM SUITABLY
13: C   -----
14: C   PROGRAM RANKOPT
15: C   PARAMETER (NOB=30,MVAR=7)!CHANGE THE PARAMETERS HERE AS NEEDED.
16: C   -----
17: C   NOB=NO. OF CASES AND MVAR=NO. OF VARIABLES
18: C   TO BE ADJUSTED IN SUBROUTINE CORD(M,X,F) ALSO: STATEMENT 931
19: C   IMPLICIT DOUBLE PRECISION (A-H, O-Z)
20: C   COMMON /KFF/KF,NFCALL,FTIT ! FUNCTION CODE, NO. OF CALLS & TITLE
21: C   CHARACTER *30 METHOD(1)
22: C   CHARACTER *70 FTIT
23: C   CHARACTER *40 INFILE,OUTFILE
24: C   COMMON /CORDAT/CDAT (NOB,MVAR) , QIND (NOB) , R (MVAR) , ENTROPY,NORM,NCOR
25: C   COMMON /XBASE/XBAS
26: C   COMMON /RNDM/IU,IV ! RANDOM NUMBER GENERATION (IU = 4-DIGIT SEED)
27: C   COMMON /GETRANK/MRNK
28: C   INTEGER IU,IV
29: C   DIMENSION XX(3,50) , KKF(3) , MM(3) , FMINN(3) , XBAS(1000,50)
30: C   DIMENSION ZDAT (NOB,MVAR+1) , FRANK (NOB) , RMAT (MVAR+1,MVAR+1)
31: C   DIMENSION X(50)! X IS THE DECISION VARIABLE X IN F(X) TO MINIMIZE
32: C   M IS THE DIMENSION OF THE PROBLEM, KF IS TEST FUNCTION CODE AND
33: C   FMIN IS THE MIN VALUE OF F(X) OBTAINED FROM DE OR RPS
34: C   WRITE (*,*) '===== WARNING ===== '
35: C   WRITE (*,*) 'ADJUST PARAMETERS IN SUBROUTINES DE IF NEEDED '
36: C   NOPT=1 ! OPTIMIZATION BY DE METHOD
37: C   WRITE (*,*) '===== '
38: C   METHOD(1)=' : DIFFERENTIAL EVOLUTION'
39: C   INITIALIZATION. THIS XBAS WILL BE USED TO
40: C   INITIALIZE THE POPULATION.
41: C   WRITE (*,*) ' '
42: C   WRITE (*,*) 'FEED RANDOM NUMBER SEED,NORM,ENTROPY,NCOR'
43: C   WRITE (*,*) 'SEED[ANY 4-DIGIT NUMBER]; NORM[1,2,3]; ENTROPY[0,1]; &
44: C   &NCOR[0,1] '
45: C   WRITE (*,*) ' '
46: C   WRITE (*,*) 'NORM(1)=ABSOLUTE;NORM(2)=PCA-EUCLIDEAN;NORM(3)=MAXIMIN'
47: C   WRITE (*,*) 'ENTROPY(0)=MAXIMIZES NORM;ENTROPY(1)=MAXIMIZES ENTROPY'
48: C   WRITE (*,*) 'NCOR(0)=PRODUCT MOMENT; NCOR(1)=ABSOLUTE CORRELATION'
49: C   READ (*,*) IU,NORM,ENTROPY,NCOR
50: C   WRITE (*,*) 'WANT RANK SCORE OPTIMIZATION? YES(1); NO(OTHER THAN 1)'
51: C   READ (*,*) MRNK
52: C   WRITE (*,*) 'INPUT FILE TO READ DATA:YOUR DATA MUST BE IN THIS FILE'
53: C   WRITE (*,*) 'CASES (NOB) IN ROWS ; VARIABLES (MVAR) IN COLUMNS'
54: C   READ (*,*) INFILE
55: C   WRITE (*,*) 'SPECIFY THE OUTPUT FILE TO STORE THE RESULTS'
56: C   READ (*,*) OUTFILE
57: C   OPEN (9, FILE=OUTFILE)
58: C   OPEN (7,FILE=INFILE)
59: C   DO I=1,NOB
60: C   READ (7,*) ,CDA, (CDAT (I,J) ,J=1,MVAR)
61: C   ENDDO
62: C   CLOSE (7)
63: C   DO I=1,NOB
64: C   DO J=1,MVAR
65: C   ZDAT (I,J+1)=CDAT (I,J)
66: C   ENDDO
67: C   ENDDO

```

```

68:      WRITE(*,*) 'DATA HAS BEEN READ. WOULD YOU UNITIZE VARIABLES? [YES=1
69: & ELSE NO UNITIZATION] '
70:      WRITE(*,*) 'UNITIZE MEANS TRANSFORMATION FROM X(I,J) TO UNITIZED X'
71:      WRITE(*,*) '[X(I,J)-MIN(X(.,J))]/[MAX(X(.,J))-MIN(X(.,J))]'
72:      READ(*,*) NUN
73:      IF(NUN.EQ.1) THEN
74:      DO J=1,MVAR
75:      CMIN=CDAT(1,J)
76:      CMAX=CDAT(1,J)
77:      DO I=2,NOB
78:      IF(CMIN.GT.CDAT(I,J)) CMIN=CDAT(I,J)
79:      IF(CMAX.LT.CDAT(I,J)) CMAX=CDAT(I,J)
80:      ENDDO
81:      DO I=1,NOB
82:      CDAT(I,J)=(CDAT(I,J)-CMIN)/(CMAX-CMIN)
83:      ENDDO
84:      ENDDO
85:      ENDIF
86: C -----
87:      WRITE(*,*) ' '
88:      WRITE(*,*) 'FEED RANDOM NUMBER SEED [4-DIGIT ODD INTEGER] TO BEGIN'
89:      READ(*,*) IU
90: C THIS XBAS WILL BE USED AS INITIAL X
91:      DO I=1,1000
92:      DO J=1,50
93:      CALL RANDOM(RAND)
94:      XBAS(I,J)=RAND ! RANDOM NUMBER BETWEEN (0, 1)
95:      ENDDO
96:      ENDDO
97: C -----
98: C HOWEVER, THE FIRST ROW OF, THAT IS, XBAS(1,J),J=1,MVAR) MAY BE
99: C SPECIFIED HERE IF THE USER KNOWS IT TO BE OPTIMAL OR NEAR-OPTIMAL
100: C DATA (XBAS(1,J),J=1,MVAR) /DATA1, DATA2, ....., DATAMVAR/
101: C -----
102:      WRITE(*,*) ' *****'
103: C -----
104:      DO I=1,NOPT
105:      IF(I.EQ.1) THEN
106:      WRITE(*,*) '==== WELCOME TO DE/RPS PROGRAM FOR INDEX CONSTRUCTION'
107:      CALL DE(M,X,FMINDE,Q0,Q1) !CALLS DE AND RETURNS OPTIMAL X AND FMIN
108:      IF(KF.EQ.1) THEN
109:      WRITE(9,*) 'DIFFERENTIAL EVALUATION OPTIMIZATION RESULTS'
110:      RSUM1=0.D0
111:      RSUM2=0.D0
112:      DO J=1,MVAR
113:      RSUM1=RSUM1+DABS(R(J))
114:      RSUM2=RSUM2+DABS(R(J))**2
115:      ENDDO
116:      WRITE(9,*) 'CORRELATION OF INDEX WITH CONSTITUENT VARIABLES'
117:      WRITE(9,*) (R(J),J=1,MVAR)
118:      WRITE(9,*) 'SUM OF ABS (R)=' ,RSUM1, '; SUM OF SQUARE(R)=' ,RSUM2
119:      WRITE(9,*) 'THE INDEX OR SCORE OF DIFFERENT CASES'
120:      DO II=1,NOB
121:      WRITE(9,*) QIND(II)
122:      FRANK(II)=QIND(II)
123:      ENDDO
124:      ENDIF
125:      FMIN=FMINDE
126:      ENDIF
127: C -----
128:      DO J=1,M
129:      XX(I,J)=X(J)
130:      ENDDO
131:      KKF(I)=KF
132:      MM(I)=M
133:      FMINN(I)=FMIN
134:      ENDDO

```

```

135:      WRITE (*,*) ' '
136:      WRITE (*,*) ' '
137:      WRITE (*,*) '----- FINAL RESULTS-----'
138:      DO I=1,NOPT
139:      WRITE (*,*) 'FUNCT CODE=',KKF(I), ' FMIN=',FMINN(I), ' : DIM=',MM(I)
140:      WRITE (*,*) 'OPTIMAL DECISION VARIABLES : ',METHOD(I)
141:      WRITE (*,*) 'WEIGHTS ARE AS FOLLOWS -----'
142:      WRITE (9,*) 'WEIGHTS ARE AS FOLLOWS -----'
143:      WRITE (9,*) (XX(I,J),J=1,M)
144:      WRITE (*,*) (XX(I,J),J=1,M)
145:      WRITE (*,*) '/////////////////////////////////////'
146:      ENDDO
147:      WRITE (*,*) 'OPTIMIZATION PROGRAM ENDED'
148:      WRITE (*,*) '*****'
149:      WRITE (*,*) 'MEASURE OF EQUALITY/INEQUALITY'
150:      WRITE (*,*) 'DE: BEFORE AND AFTER OPTIMIZATION = ',Q0,Q1
151:      WRITE (*,*) ' '
152:      WRITE (*,*) 'RESULTS STORED IN FILE= ',OUTFILE
153:      WRITE (*,*) 'OPEN BY MSWORD OR EDIT OR ANY OTHER EDITOR'
154:      WRITE (*,*) ' '
155:      WRITE (*,*) 'NOTE:VECTORS OF CORRELATIONS & INDEX(BOTH TOGETHER) ARE
156: & IDETERMINATE FOR SIGN & MAY BE MULTIPLIED BY (-1) IF NEEDED'
157:      WRITE (*,*) 'THAT IS IF R(J) IS TRANSFORMED TO -R(J) FOR ALL J THEN
158: &THE INDEX(I) TOO IS TRANSFORMED TO -INDEX(I) FOR ALL I'
159:      WRITE (9,*) ' '
160:      WRITE (9,*) 'NOTE: VECTORS OF CORRELATIONS AND INDEX (BOTH TOGETHER)
161: & ARE IDETERMINATE FOR SIGN AND MAY BE MULTIPLIED BY (-1) IF NEEDED'
162:      WRITE (9,*) 'THAT IS IF R(J) IS TRANSFORMED TO -R(J) FOR ALL J THEN
163: &THE INDEX(I) TOO IS TRANSFORMED TO -INDEX(I) FOR ALL I'
164:      CALL DORANK(FRANK,NOB)
165:      DO I=1,NOB
166:      ZDAT(I,1)=FRANK(I)
167:      ENDDO
168:      CALL CORREL(ZDAT,NOB,MVAR+1,RMAT)
169:      WRITE (9,*) '----- CORRELATION MATRIX -----'
170:      WRITE (*,*) '----- CORRELATION MATRIX -----'
171:      DO I=1,MVAR+1
172:      WRITE (9,1) (RMAT(I,J),J=1,MVAR+1)
173:      WRITE (*,1) (RMAT(I,J),J=1,MVAR+1)
174:      ENDDO
175:      1 FORMAT(8F10.6)
176:      WRITE (9,*) '===== '
177:      WRITE (*,*) '===== '
178:      WRITE (9,*) 'VARIABLES: 1ST IS THE INDEX AND 2ND THE RANK OF INDEX'
179:      WRITE (*,*) 'VARIABLES: 1ST IS THE INDEX AND 2ND THE RANK OF INDEX'
180:      WRITE (9,*) '===== '
181:      WRITE (*,*) '===== '
182:      DO I=1,NOB
183:      IF (MRNK.EQ.1) THEN
184:      QIND(I)=0.D0
185:      DO J=1,MVAR
186:      QIND(I)=QIND(I)+ZDAT(I,J+1)*XX(NOPT,J)
187:      ENDDO
188:      ENDIF
189:      WRITE (9,2) I,QIND(I), (ZDAT(I,J),J=1,MVAR+1)
190:      WRITE (*,2) I,QIND(I), (ZDAT(I,J),J=1,MVAR+1)
191:      ENDDO
192:      2 FORMAT(I3,F12.6,13F5.0)
193:      SR2=0.D0
194:      DO J=2,MVAR+1
195:      SR2=SR2+RMAT(1,J)**2
196:      ENDDO
197:      WRITE (9,*) 'SUM OF SQUARE OF CORRELATION R(RANK(INDEX),VAR)=',SR2
198:      WRITE (*,*) 'SUM OF SQUARE OF CORRELATION R(RANK(INDEX),VAR)=',SR2
199:      CLOSE(9)
200:      WRITE (*,*) 'THE JOB IS OVER'
201:      END

```

```

202: C -----
203: C SUBROUTINE DE(M,A,FBEST,G0,G1)
204: C PROGRAM: "DIFFERENTIAL EVOLUTION ALGORITHM" OF GLOBAL OPTIMIZATION
205: C THIS METHOD WAS PROPOSED BY R. STORN AND K. PRICE IN 1995. REF --
206: C "DIFFERENTIAL EVOLUTION - A SIMPLE AND EFFICIENT ADAPTIVE SCHEME
207: C FOR GLOBAL OPTIMIZATION OVER CONTINUOUS SPACES" : TECHNICAL REPORT
208: C INTERNATIONAL COMPUTER SCIENCE INSTITUTE, BERKLEY, 1995.
209: C PROGRAM BY SK MISHRA, DEPT. OF ECONOMICS, NEHU, SHILLONG (INDIA)
210: C -----
211: C PROGRAM DE
212: C IMPLICIT DOUBLE PRECISION (A-H, O-Z) ! TYPE DECLARATION
213: C PARAMETER (NMAX=500,MMAX=50) ! MAXIMUM DIMENSION PARAMETERS
214: C PARAMETER (RX1=1.0D0, RX2=1.0D0) ! TO BE ADJUSTED SUITABLY, IF NEEDED
215: C RX1 AND RX2 CONTROL THE SCHEME OF CROSSOVER. (0 <= RX1 <= RX2) <=1
216: C RX1 DETERMINES THE UPPER LIMIT OF SCHEME 1 (AND LOWER LIMIT OF
217: C SCHEME 2; RX2 IS THE UPPER LIMIT OF SCHEME 2 AND LOWER LIMIT OF
218: C SCHEME 3. THUS RX1 = .2 AND RX2 = .8 MEANS 0-20% SCHEME1, 20 TO 80
219: C PERCENT SCHEME 2 AND THE REST (80 TO 100 %) SCHEME 3.
220: C PARAMETER (NCROSS=2) ! CROSS-OVER SCHEME (NCROSS <=0 OR =1 OR =>2)
221: C PARAMETER (IPRINT=100,EPS=1.D-10) !FOR WATCHING INTERMEDIATE RESULTS
222: C IT PRINTS THE INTERMEDIATE RESULTS AFTER EACH IPRINT ITERATION AND
223: C EPS DETERMINES ACCURACY FOR TERMINATION. IF EPS= 0, ALL ITERATIONS
224: C WOULD BE UNDERGONE EVEN IF NO IMPROVEMENT IN RESULTS IS THERE.
225: C ULTIMATELY "DID NOT CONVERGE" IS REOORTED.
226: C COMMON /RNDM/IU,IV ! RANDOM NUMBER GENERATION (IU = 4-DIGIT SEED)
227: C INTEGER IU,IV ! FOR RANDOM NUMBER GENERATION
228: C COMMON /KFF/KF,NFCALL,FTIT ! FUNCTION CODE, NO. OF CALLS * TITLE
229: C COMMON /XBASE/XBAS
230: C CHARACTER *70 FTIT ! TITLE OF THE FUNCTION
231: C -----
232: C THE PROGRAM REQUIRES INPUTS FROM THE USER ON THE FOLLOWING -----
233: C (1) FUNCTION CODE (KF), (2) NO. OF VARIABLES IN THE FUNCTION (M);
234: C (3) N=POPULATION SIZE (SUGGESTED 10 TIMES OF NO. OF VARIABLES, M,
235: C FOR SMALLER PROBLEMS N=100 WORKS VERY WELL);
236: C (4) PCROS = PROB. OF CROSS-OVER (SUGGESTED : ABOUT 0.85 TO .99);
237: C (5) FACT = SCALE (SUGGESTED 0.5 TO .95 OR 1, ETC);
238: C (6) ITER = MAXIMUM NUMBER OF ITERATIONS PERMITTED (5000 OR MORE)
239: C (7) RANDOM NUMBER SEED (4 DIGITS INTEGER)
240: C -----
241: C DIMENSION X(NMAX,MMAX),Y(NMAX,MMAX),A(MMAX),FV(NMAX)
242: C DIMENSION IR(3),XBAS(1000,50)
243: C -----
244: C ----- SELECT THE FUNCTION TO MINIMIZE AND ITS DIMENSION -----
245: C CALL FSELECT(KF,M,FTIT)
246: C SPECIFY OTHER PARAMETERS -----
247: C WRITE (*,*) 'POPULATION SIZE [N] AND NO. OF ITERATIONS [ITER] ?'
248: C WRITE (*,*) 'SUGGESTED : N => 100 OR =>10.M; ITER 10000 OR SO'
249: C READ (*,*) N,ITER
250: C WRITE (*,*) 'CROSSOVER PROBABILITY [PCROS] AND SCALE [FACT] ?'
251: C WRITE (*,*) 'SUGGESTED : PCROS ABOUT 0.9; FACT=.5 OR LARGER BUT <=1'
252: C READ (*,*) PCROS,FACT
253: C WRITE (*,*) 'RANDOM NUMBER SEED ?'
254: C WRITE (*,*) 'A FOUR-DIGIT POSITIVE ODD INTEGER, SAY, 1171'
255: C READ (*,*) IU
256: C NFCALL=0 ! INITIALIZE COUNTER FOR FUNCTION CALLS
257: C GBEST=1.D30 ! TO BE USED FOR TERMINATION CRITERION
258: C INITIALIZATION : GENERATE X(N,M) RANDOMLY
259: C DO I=1,N
260: C DO J=1,M
261: C CALL RANDOM(RAND) ! GENERATES INITION X WITHIN
262: C X(I,J)=(RAND-.5D00)*2000 ! GENERATES INITION X WITHIN
263: C RANDOM NUMBERS BETWEEN -RRANGE AND +RRANGE (BOTH EXCLUSIVE)
264: C X(I,J)=XBAS(I,J) ! TAKES THESE NUMBERS FROM THE MAIN PROGRAM
265: C ENDDO
266: C ENDDO
267: C WRITE (*,*) 'COMPUTING --- PLEASE WAIT '
268: C IPCOUNT=0

```

```

269:      DO 100 ITR=1,ITER  ! ITERATION BEGINS
270:  C -----
271:  C  EVALUATE ALL X FOR THE GIVEN FUNCTION
272:      DO I=1,N
273:      DO J=1,M
274:      A(J)=X(I,J)
275:      ENDDO
276:      CALL FUNC(A,M,F)
277:  C  STORE FUNCTION VALUES IN FV VECTOR
278:      FV(I)=F
279:      ENDDO
280:      IF(ITR.EQ.1) CALL GINI(FV,N,G0)
281:  C -----
282:  C  FIND THE FITTEST (BEST) INDIVIDUAL AT THIS ITERATION
283:      FBEST=FV(1)
284:      KB=1
285:      DO IB=2,N
286:      IF(FV(IB).LT.FBEST) THEN
287:      FBEST=FV(IB)
288:      KB=IB
289:      ENDIF
290:      ENDDO
291:  C  BEST FITNESS VALUE = FBEST : INDIVIDUAL X(KB)
292:  C -----
293:  C  GENERATE OFFSPRINGS
294:      DO I=1,N  ! I LOOP BEGINS
295:  C  INITIALIZE CHILDREN IDENTICAL TO PARENTS; THEY WILL CHANGE LATER
296:      DO J=1,M
297:      Y(I,J)=X(I,J)
298:      ENDDO
299:  C  SELECT RANDOMLY THREE OTHER INDIVIDUALS
300:  20      DO IRI=1,3  ! IRI LOOP BEGINS
301:      IR(IRI)=0
302:
303:      CALL RANDOM(RAND)
304:      IRJ=INT(RAND*N)+1
305:  C  CHECK THAT THESE THREE INDIVIDUALS ARE DISTICT AND OTHER THAN I
306:      IF(IRI.EQ.1.AND.IRJ.NE.I) THEN
307:      IR(IRI)=IRJ
308:      ENDIF
309:      IF(IRI.EQ.2.AND.IRJ.NE.I.AND.IRJ.NE.IR(1)) THEN
310:      IR(IRI)=IRJ
311:      ENDIF
312:      IF(IRI.EQ.3.AND.IRJ.NE.I.AND.IRJ.NE.IR(1).AND.IRJ.NE.IR(2)) THEN
313:      IR(IRI)=IRJ
314:      ENDIF
315:      ENDDO  ! IRI LOOP ENDS
316:  C  CHECK IF ALL THE THREE IR ARE POSITIVE (INTEGERS)
317:      DO IX=1,3
318:      IF(IR(IX).LE.0) THEN
319:      GOTO 20  ! IF NOT THEN REGENERATE
320:      ENDIF
321:      ENDDO
322:  C  THREE RANDOMLY CHOSEN INDIVIDUALS DIFFERENT FROM I AND DIFFERENT
323:  C  FROM EACH OTHER ARE IR(1),IR(2) AND IR(3)
324:  C  ===== RANDOMIZATION OF NCROSS =====
325:  C  RANDOMIZES NCROSS
326:      NCROSS=0
327:      CALL RANDOM(RAND)
328:      IF(RAND.GT.RX1) NCROSS=1 ! IF RX1=>1, SCHEME 2 NEVER IMPLEMENTED
329:      IF(RAND.GT.RX2) NCROSS=2 ! IF RX2=>1, SCHEME 3 NEVER IMPLEMENTED
330:
331:  C ----- SCHEME 1 -----
332:  C  NO CROSS OVER, ONLY REPLACEMENT THAT IS PROBABILISTIC
333:      IF(NCROSS.LE.0) THEN
334:      DO J=1,M  ! J LOOP BEGINS
335:      CALL RANDOM(RAND)

```

```

336:         IF (RAND.LE.PCROS) THEN ! REPLACE IF RAND < PCROS
337:         A(J)=X(IR(1),J)+(X(IR(2),J)-X(IR(3),J))*FACT ! CANDIDATE CHILD
338:         ENDIF
339:         ENDDO ! J LOOP ENDS
340:         ENDIF
341: C ----- SCHEME 2 -----
342: C THE STANDARD CROSSOVER SCHEME
343: C CROSSOVER SCHEME (EXPONENTIAL) SUGGESTED BY KENNETH PRICE IN HIS
344: C PERSONAL LETTER TO THE AUTHOR (DATED SEPTEMBER 29, 2006)
345:         IF (NCROSS.EQ.1) THEN
346:             CALL RANDOM(RAND)
347:             JR=INT(RAND*M)+1
348:             J=JR
349:         2 A(J)=X(IR(1),J)+FACT*(X(IR(2),J)-X(IR(3),J))
350:             J=J+1
351:             IF (J.GT.M) J=1
352:             IF (J.EQ.JR) GOTO 10
353:             CALL RANDOM(RAND)
354:             IF (PCROS.LE.RAND) GOTO 2
355:         6 A(J)=X(I,J)
356:             J=J+1
357:             IF (J.GT.M) J=1
358:             IF (J.EQ.JR) GOTO 10
359:             GOTO 6
360:         10 CONTINUE
361:         ENDIF
362: C ----- SCHEME 3 -----
363: C ESPECIALLY SUITABLE TO NON-DECOMPOSABLE (NON-SEPERABLE) FUNCTIONS
364: C CROSSOVER SCHEME (NEW) SUGGESTED BY KENNETH PRICE IN HIS
365: C PERSONAL LETTER TO THE AUTHOR (DATED OCTOBER 18, 2006)
366:         IF (NCROSS.GE.2) THEN
367:             CALL RANDOM(RAND)
368:             IF (RAND.LE.PCROS) THEN
369:                 CALL NORMAL(RN)
370:                 DO J=1,M
371:                     A(J)=X(I,J)+(X(IR(1),J)+X(IR(2),J)-2*X(I,J))*RN
372:                 ENDDO
373:             ELSE
374:                 DO J=1,M
375:                     A(J)=X(I,J)+(X(IR(1),J)-X(IR(2),J)) ! FACT ASSUMED TO BE 1
376:                 ENDDO
377:             ENDIF
378:         ENDIF
379: C -----
380:         CALL FUNC(A,M,F) ! EVALUATE THE OFFSPRING
381:         IF (F.LT.FV(I)) THEN ! IF BETTER, REPLACE PARENTS BY THE CHILD
382:             FV(I)=F
383:             DO J=1,M
384:                 Y(I,J)=A(J)
385:             ENDDO
386:         ENDIF
387:         ENDDO ! I LOOP ENDS
388:         DO I=1,N
389:             DO J=1,M
390:                 X(I,J)=Y(I,J) ! NEW GENERATION IS A MIX OF BETTER PARENTS AND
391: C BETTER CHILDREN
392:             ENDDO
393:         ENDDO
394:         DO J=1,M
395:             A(J)=X(KB,J)
396:         ENDDO
397:         IPCOUNT=IPCOUNT+1
398:         IF (IPCOUNT.EQ.IPRINT) THEN
399:
400:         WRITE (*,*) (X(KB,J),J=1,M), ' FBEST UPTO NOW = ',FBEST
401:         WRITE (*,*) 'TOTAL NUMBER OF FUNCTION CALLS =',NFCALL
402:         IF (DABS (FBEST-GBEST) .LT.EPS) THEN

```

```

403:         WRITE(*,*) FTIT
404:         WRITE(*,*) 'COMPUTATION OVER'
405:         GOTO 999
406:         ELSE
407:         GBEST=FBEST
408:         ENDDIF
409:         IPCOUNT=0
410:         ENDDIF
411: C -----
412: 100 ENDDO ! ITERATION ENDS : GO FOR NEXT ITERATION, IF APPLICABLE
413: C -----
414:         WRITE(*,*) 'DID NOT CONVERGE. REDUCE EPS OR RAISE ITER OR DO BOTH'
415:         WRITE(*,*) 'INCREASE N, PCROS, OR SCALE FACTOR (FACT)'
416: 999 CALL FUNC(A,M,FBEST)
417:         CALL GINI(FV,N,G1)
418:         RETURN
419:         END
420: C -----
421:         SUBROUTINE NORMAL(R)
422: C PROGRAM TO GENERATE N(0,1) FROM RECTANGULAR RANDOM NUMBERS
423: C IT USES BOX-MULLER VARIATE TRANSFORMATION FOR THIS PURPOSE.
424: C -----
425: C ----- BOX-MULLER METHOD BY GEP BOX AND ME MULLER (1958) -----
426: C BOX, G. E. P. AND MULLER, M. E. "A NOTE ON THE GENERATION OF
427: C RANDOM NORMAL DEVIATES." ANN. MATH. STAT. 29, 610-611, 1958.
428: C IF U1 AND U2 ARE UNIFORMLY DISTRIBUTED RANDOM NUMBERS (0,1),
429: C THEN X=[(-2*LN(U1))**.5]*(COS(2*PI*U2) IS N(0,1)
430: C ALSO, X=[(-2*LN(U1))**.5]*(SIN(2*PI*U2) IS N(0,1)
431: C PI = 4*ARCTAN(1.0)= 3.1415926535897932384626433832795
432: C 2*PI = 6.283185307179586476925286766559
433: C -----
434:         IMPLICIT DOUBLE PRECISION (A-H,O-Z)
435:         COMMON /RNDM/IU,IV
436:         INTEGER IU,IV
437: C -----
438:         CALL RANDOM(RAND) ! INVOKES RANDOM TO GENERATE UNIFORM RAND [0, 1]
439:         U1=RAND ! U1 IS UNIFORMLY DISTRIBUTED [0, 1]
440:         CALL RANDOM(RAND) ! INVOKES RANDOM TO GENERATE UNIFORM RAND [0, 1]
441:         U2=RAND ! U2 IS UNIFORMLY DISTRIBUTED [0, 1]
442:         R=DSQRT(-2.0*DLOG(U1))
443:         R=R*DCOS(U2*6.283185307179586476925286766559D00)
444: C R=R*DCOS(U2*6.28318530718D00)
445:         RETURN
446:         END
447: C -----
448: C RANDOM NUMBER GENERATOR (UNIFORM BETWEEN 0 AND 1 - BOTH EXCLUSIVE)
449:         SUBROUTINE RANDOM(RAND1)
450:         DOUBLE PRECISION RAND1
451:         COMMON /RNDM/IU,IV
452:         INTEGER IU,IV
453:         IV=IU*65539
454:         IF (IV.LT.0) THEN
455:         IV=IV+2147483647+1
456:         ENDDIF
457:         RAND=IV
458:         IU=IV
459:         RAND=RAND*0.4656613E-09
460:         RAND1= DBLE(RAND)
461:         RETURN
462:         END
463: C -----
464:         SUBROUTINE GINI(F,N,G)
465:         PARAMETER (K=1) !K=1 GINI COEFFICIENT; K=2 COEFFICIENT OF VARIATION
466: C THIS PROGRAM COMPUTES MEASURE OF INEQUALITY
467: C IF K =1 GET THE GINI COEFFICIENT. IF K=2 GET COEFF OF VARIATIONE
468:         IMPLICIT DOUBLE PRECISION (A-H,O-Z)
469:         DIMENSION F(*)

```



```

470:      S=0.D0
471:      DO I=1,N
472:      S=S+F(I)
473:      ENDDO
474:      S=S/N
475:      H=0.D00
476:      DO I=1,N-1
477:      DO J=I+1,N
478:      H=H+(DABS(F(I)-F(J)))*K
479:      ENDDO
480:      ENDDO
481:      H=(H/(N**2))** (1.D0/K) ! FOR K=1 H IS MEAN DEVIATION;
482: C      FOR K=2 H IS STANDARD DEVIATION
483:      WRITE(*,*) 'MEASURES OF DISPERSION AND CENTRAL TENDENCY = ',G,S
484:      G=DEXP(-H) ! G IS THE MEASURE OF EQUALITY (NOT GINI OR CV)
485: C      G=H/DABS(S) !IF S NOT ZERO, K=1 THEN G=GINI, K=2 G=COEFF VARIATION
486:      RETURN
487:      END
488: C
489:      SUBROUTINE FSELECT(KF,M,FTIT)
490: C      THE PROGRAM REQUIRES INPUTS FROM THE USER ON THE FOLLOWING -----
491: C      (1) FUNCTION CODE (KF), (2) NO. OF VARIABLES IN THE FUNCTION (M);
492:      CHARACTER *70 TIT(100),FTIT
493:      NFN=1
494:      KF=1
495:      WRITE(*,*) '-----'
496:      DATA TIT(1) /'CONSTRUCTION OF INDEX FROM M VARIABLES '/
497: C
498:      DO I=1,NFN
499:      WRITE(*,*) TIT(I)
500:      ENDDO
501:      WRITE(*,*) '-----'
502:      WRITE(*,*) 'SPECIFY NO. OF VARIABLES [MVAR] HERE ALSO ?'
503:      READ(*,*) M
504:      FTIT=TIT(KF) ! STORE THE NAME OF THE CHOSEN FUNCTION IN FTIT
505:      RETURN
506:      END
507: C
508:      SUBROUTINE FUNC(X,M,F)
509: C      TEST FUNCTIONS FOR GLOBAL OPTIMIZATION PROGRAM
510:      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
511:      COMMON /RNDM/IU,IV
512:      COMMON /KFF/KF,NFCALL,FTIT
513:      INTEGER IU,IV
514:      DIMENSION X(*)
515:      CHARACTER *70 FTIT
516:      NFCALL=NFCALL+1 ! INCREMENT TO NUMBER OF FUNCTION CALLS
517: C      KF IS THE CODE OF THE TEST FUNCTION
518:      IF (KF.EQ.1) THEN
519:      CALL CORD(M,X,F)
520:      RETURN
521:      ENDIF
522: C
523:      WRITE(*,*) 'FUNCTION NOT DEFINED. PROGRAM ABORTED'
524:      STOP
525:      END
526: C
527:      SUBROUTINE CORD(M,X,F)
528:      PARAMETER (NOB=30,MVAR=7) ! CHANGE THE PARAMETERS HERE AS NEEDED.
529: C
530: C      NOB=NO. OF OBSERVATIONS (CASES) & MVAR= NO. OF VARIABLES
531:      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
532:      COMMON /RNDM/IU,IV
533:      COMMON /CORDAT/CDAT (NOB,MVAR), QIND (NOB), R (MVAR), ENTROPY, NORM, NCOR
534:      COMMON /GETRANK/MRNK
535:      INTEGER IU,IV
536:      DIMENSION X(*), Z (NOB,2)

```

```

537:      DO I=1,M
538:      IF (X(I) .LT. -1.0D0 .OR. X(I) .GT. 1.0D0) THEN
539:      CALL RANDOM(RAND)
540:      X(I) = (RAND-0.5D0) * 2
541:      ENDF
542:      ENDDO
543:      XNORM=0.D0
544:      DO J=1,M
545:      XNORM=XNORM+X(J)**2
546:      ENDDO
547:      XNORM=DSQRT(XNORM)
548:      DO J=1,NOB
549:      X(J)=X(J)/XNORM
550:      ENDDO
551: C      CONSTRUCT INDEX
552:      DO I=1,NOB
553:      QIND(I)=0.D0
554:      DO J=1,M
555:      QIND(I)=QIND(I)+CDAT(I,J)*X(J)
556:      ENDDO
557:      ENDDO
558: C -----
559:      !FIND THE RANK OF QIND
560:      IF (MRNK.EQ.1) CALL DORANK(QIND,NOB)
561: C -----
562: C      COMPUTE CORRELATIONS
563:      DO I=1,NOB
564:      Z(I,1)=QIND(I)
565:      ENDDO
566:      DO J=1,M
567:      DO I=1,NOB
568:      Z(I,2)=CDAT(I,J)
569:      ENDDO
570:      IF (NCOR.EQ.0) THEN
571:      CALL CORLN(Z,NOB,RHO)
572:      ELSE
573:      CALL CORA(Z,NOB,RHO)
574:      ENDF
575:      R(J)=RHO
576:      ENDDO
577:      IF (ENTROPY.EQ.0.D0) THEN
578: C ----- MAXIMIN SOLUTION -----
579:      IF (NORM.GT.2) THEN
580:      F=DABS(R(1))
581:      DO J=2,M
582:      IF (F.GT.DABS(R(J))) F= DABS(R(J))
583:      ENDDO
584:      ENDF
585: C ----- FOR NORM =1 OR 2 -----
586:      IF (NORM.LE.2) THEN
587:      F=0.D0
588:      DO J=1,M
589:      F=F+DABS(R(J))**NORM
590:      ENDDO
591:      ENDF
592: C -----
593:      ELSE
594:      IF (ENTROPY.NE.0.D0) THEN
595: C      ENTROPY MAXIMIZATION
596:      ENT=0.0D0
597:      DO J=1,M
598:      ENT=ENT+DABS(R(J))
599:      ENDDO
600:      F=ENT*DLOG(ENT)
601:      DO J=1,M
602:      FX=DABS(R(J))
603:      F=F+ (FX/ENT)*DLOG(FX/ENT)

```

```

604:      ENDDO
605:      ENDIF
606:      ENDIF
607: C -----
608:      F=-F
609:      RETURN
610:      END
611:      SUBROUTINE CORLN(Z,NOB,RHO)
612: C      NOB = NO. OF CASES
613:      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
614:      DIMENSION Z(NO,2),AV(2),SD(2)
615:      DO J=1,2
616:      AV(J)=0.D0
617:      SD(J)=0.D0
618:      DO I=1,NOB
619:      AV(J)=AV(J)+Z(I,J)
620:      SD(J)=SD(J)+Z(I,J)**2
621:      ENDDO
622:      ENDDO
623:      DO J=1,2
624:      AV(J)=AV(J)/NOB
625:      SD(J)=DSQRT(SD(J)/NOB-AV(J)**2)
626:      ENDDO
627: C      WRITE(*,*)'AV AND SD ',AV(1),AV(2),SD(1),SD(2)
628:      RHO=0.D0
629:      DO I=1,NOB
630:      RHO=RHO+(Z(I,1)-AV(1))*(Z(I,2)-AV(2))
631:      ENDDO
632:      RHO=(RHO/NOB)/(SD(1)*SD(2))
633:      RETURN
634:      END
635: C -----
636:      SUBROUTINE CORA(Z,N,R)
637: C      COMPUTING ABSOLUTE CORRELATION MATRIX
638:      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
639:      DIMENSION Z(N,2),X(N),Y(N)
640: C -----
641: C      PUT Z INTO X AND Y
642:      DO I=1,N
643:      X(I)=Z(I,1)
644:      Y(I)=Z(I,2)
645:      ENDDO
646: C      ARRANGE X ANY IN AN ASCENDING ORDER
647:      DO I=1,N-1
648:      DO II=I+1,N
649:      IF(X(I).GT.X(II)) THEN
650:      TEMP=X(I)
651:      X(I)=X(II)
652:      X(II)=TEMP
653:      ENDIF
654:      IF(Y(I).GT.Y(II)) THEN
655:      TEMP=Y(I)
656:      Y(I)=Y(II)
657:      Y(II)=TEMP
658:      ENDIF
659:      ENDDO
660:      ENDDO
661: C      FIND MEDIAN
662:      IF(INT(N/2).EQ.N/2.D0) THEN
663:      XMED=(X(N/2)+X(N/2+1))/2.D0
664:      YMED=(Y(N/2)+Y(N/2+1))/2.D0
665:      ENDIF
666:      IF(INT(N/2).NE.N/2.D0) THEN
667:      XMED=X(N/2+1)
668:      YMED=Y(N/2+1)
669:      ENDIF
670: C      SUBTRACT RESPECTIVE MEDIANS FROM X AND Y AND FIND ABS DEVIATIONS

```

```

671:      VX=0.D0
672:      VY=0.D0
673:      DO I=1,N
674:      X(I)=X(I)-XMED
675:      Y(I)=Y(I)-YMED
676:      VX=VX+DABS(X(I))
677:      VY=VY+DABS(Y(I))
678:      ENDDO
679: C      SCALE THE VARIABLES X AND Y SUCH THAT VX=VY
680:      IF(VX.EQ.0.D0.OR.VY.EQ.0.D0) THEN
681:      R=0.D0
682:      RETURN
683:      ENDIF
684:      DO I=1,N
685:      X(I)=X(I)*VY/VX
686:      ENDDO
687: C      COMPUTE CORRELATION COEFFICIENT
688:      VZ=0.D0
689:      R=0.D0
690:      DO I=1,N
691:      VZ=VZ+DABS(X(I))+DABS(Y(I))
692:      R=R+DABS(X(I)+Y(I))-DABS(X(I)-Y(I))
693:      ENDDO
694:      R=R/VZ
695:      RETURN
696:      END
697: C      -----
698:      SUBROUTINE DORANK(X,N) ! N IS THE NUMBER OF OBSERVATIONS
699:      PARAMETER (MXD=1000) ! MXD IS MAX DIMENSION FOR TEMPORARY VARIABLES
700:      ! THAT ARE LOCAL AND DO NOT GO TO THE INVOKING PROGRAM
701:      ! X IS THE VARIABLE TO BE SUBSTITUTED BY ITS RANK VALUES
702:      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
703:      DIMENSION X(N), ID(MXD)
704:      ! GENERATE ID
705:      DO I=1,N
706:      ID(I)=I
707:      ENDDO
708:
709:      DO I=1,N-1
710:      DO II=I+1,N
711:      IF(X(I).GT.X(II)) THEN ! ARRANGE ACCORDING TO ASCENDING ORDER OF X
712:      T=X(I)
713:      X(I)=X(II)
714:      X(II)=T
715:      IT=ID(I)
716:      ID(I)=ID(II)
717:      ID(II)=IT
718:      ENDIF
719:      ENDDO
720:      ENDDO
721:
722:      DO I=1,N
723:      X(I)=I+0.D0
724:      ENDDO
725:
726:      DO I=1,N-1
727:      DO II=I+1,N
728:      IF(ID(I).GT.ID(II)) THEN !ARRANGE ACCORDING TO ASCENDNG ORDER OF ID
729:      T=X(I)
730:      X(I)=X(II)
731:      X(II)=T
732:      IT=ID(I)
733:      ID(I)=ID(II)
734:      ID(II)=IT
735:      ENDIF
736:      ENDDO
737:      ENDDO

```

```
738:      RETURN
739:      END
740: C -----
741:      SUBROUTINE CORREL(X,N,M,RMAT)
742:      PARAMETER (NMX=30) !DO NOT CHANGE UNLESS NO. OF VARIABLES EXCEED 30
743:      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
744:      DIMENSION X(N,M),RMAT(M,M),AV(NMX),SD(NMX)
745:      DO J=1,M
746:      AV(J)=0.D0
747:      SD(J)=0.D0
748:      DO I=1,N
749:      AV(J)=AV(J)+X(I,J)
750:      SD(J)=SD(J)+X(I,J)**2
751:      ENDDO
752:      AV(J)=AV(J)/N
753:      SD(J)=DSQRT(SD(J)/N-AV(J)**2)
754:      ENDDO
755:      DO J=1,M
756:      DO JJ=1,M
757:      RMAT(J,JJ)=0.D0
758:      DO I=1,N
759:      RMAT(J,JJ)=RMAT(J,JJ)+X(I,J)*X(I,JJ)
760:      ENDDO
761:      ENDDO
762:      ENDDO
763:      DO J=1,M
764:      DO JJ=1,M
765:      RMAT(J,JJ)=RMAT(J,JJ)/N-AV(J)*AV(JJ)
766:      RMAT(J,JJ)=RMAT(J,JJ)/(SD(J)*SD(JJ))
767:      ENDDO
768:      ENDDO
769:      RETURN
770:      END
771:
```