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

Mishra, SK

North-Eastern Hill University, Shillong (India)

30 December 2008

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 $Z = \Re(Y) : Y = Xw;\ \max \sum_{j=1}^{m} r^2(Y, x_j)$, where the transformation $\Re(Y)$ 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, $Z = \Re(Y) : Y = Xw;\ \max \sum_{j=1}^{m} r^2(Y, x_j)$ does not necessarily ensure the optimality of Z with respect to the constituent variables. If $Z = \Re(Y) : Y = Xw;\ \max \sum_{j=1}^{m} r^2(Y, x_j)$ then oftentimes Z turns out to be only a suboptimal representation of X.

The procedure of obtaining the ranking scores $Z = \Re(Y) : Y = Xw;\ \max \sum_{j=1}^{m} r^2(Y, x_j)$ 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 $r_{ij} \in R;\ r_{ij} = \mathrm{cov}(x_i, x_j) / \sqrt{\mathrm{var}(x_i)\,\mathrm{var}(x_j)}\ \forall\ i, j$, (ii) finding the normed eigenvector, w, of R associated with its largest eigenvalue, λ_1 and (ii) using w to obtain $Y = X^* w$, where $x^*_{ij} = (x_{ij} - \bar{x}_j) / \sqrt{\mathrm{var}(x_j)}$. Next, $Z = \Re(Y)$ is obtained. It is well known that this procedure maximizes $\sum_{j=1}^{m} r^2(Y, x_j)$. The direct maximization of $\sum_{j=1}^{m} r^2(Y, x_j)$ 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 $Z = \Re(Y)$, 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, …, 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 (F_1=4.287558; S_1= 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

Table-1.3: Inter-Correlation Matrix, Weights and Factor Loadings of Rank Order Optimized Overall Ranking Scores for the Dataset in Example-1 (F_2=4.287902; S2=4.287902)								
	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

Table-2.1: Dataset Relating to Example-2 Showing Sub-optimality of PC-based Rank-ordering of Objects											
	Ranking Scores of 30 candidates awarded by Seven Evaluators							Composite Score (Y) Optimized Results		Rank-Order (Z_2) Optimized Results	
Sl No.	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

	Ranking Scores of 30 candidates awarded by Seven Evaluators							Composite Score (Y) Optimized Results		Rank-Order (Z_2) Optimized Results	
Sl No.	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

```
C     MAIN PROGRAM : PROVIDES TO USE DIFFERENTIAL EVOLUTION METHOD TO
C                    COMPUTE COMPOSITE INDEX INDICES
C     BY MAXIMIZING SUM OF (SQUARES, OR ABSOLUTES, OR MINIMUM) OF
C     CORRELATION OF THE INDEX WITH THE CONSTITUENT VARIABLES. THE MAX
C     SUM OF SQUARES IS THE PRINCIPAL COMPONENT INDEX. IT ALSO PRIVIDES
C     TO OBTAIN MAXIMUM ENTROPY ABSOLUTE CORRELATION INDICES.
C     PRODUCT MOMENT AS WELL AS ABSOLUTE CORRELATION (BRADLEY, 1985) MAY
C     BE USED. PROGRAM BY SK MISHRA, DEPT. OF ECONOMICS, NORTH-EASTERN
C     HILL UNIVERSITY, SHILLONG (INDIA)
C     -----------------------------------------------------------------
C         ADJUST THE PARAMETERS SUITABLY IN SUBROUTINES DE
C     WHEN THE PROGRAM ASKS FOR PARAMETERS, FEED THEM SUITABLY
C     -----------------------------------------------------------------
      PROGRAM RANKOPT
      PARAMETER(NOB=30,MVAR=7)!CHANGE THE PARAMETERS HERE AS NEEDED.
C     -----------------------------------------------------------------
C     NOB=NO. OF CASES AND MVAR=NO. OF VARIABLES
C     TO BE ADJUSTED IN SUBROUTINE CORD(M,X,F) ALSO: STATEMENT 931
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /KFF/KF,NFCALL,FTIT ! FUNCTION CODE, NO. OF CALLS & TITLE
      CHARACTER *30  METHOD(1)
      CHARACTER *70 FTIT
      CHARACTER *40 INFILE,OUTFILE
      COMMON /CORDAT/CDAT(NOB,MVAR),QIND(NOB),R(MVAR),ENTROPY,NORM,NCOR
      COMMON /XBASE/XBAS
      COMMON /RNDM/IU,IV ! RANDOM NUMBER GENERATION (IU = 4-DIGIT SEED)
      COMMON /GETRANK/MRNK
      INTEGER IU,IV
      DIMENSION XX(3,50),KKF(3),MM(3),FMINN(3),XBAS(1000,50)
      DIMENSION ZDAT(NOB,MVAR+1),FRANK(NOB),RMAT(MVAR+1,MVAR+1)
      DIMENSION X(50)! X IS THE DECISION VARIABLE X IN F(X) TO MINIMIZE
C     M IS THE DIMENSION OF THE PROBLEM, KF IS TEST FUNCTION CODE AND
C     FMIN IS THE MIN VALUE OF F(X) OBTAINED FROM DE OR RPS
      WRITE(*,*)'====================     WARNING    =============== '
      WRITE(*,*)'ADJUST PARAMETERS IN SUBROUTINES DE IF NEEDED '
      NOPT=1 ! OPTIMIZATION BY DE METHOD
      WRITE(*,*)'=================================================== '
      METHOD(1)=' : DIFFERENTIAL EVOLUTION'
C     INITIALIZATION. THIS XBAS WILL BE USED TO
C     INITIALIZE THE POPULATION.
      WRITE(*,*)' '
      WRITE(*,*)'FEED RANDOM NUMBER SEED,NORM,ENTROPY,NCOR'
      WRITE(*,*)'SEED[ANY 4-DIGIT NUMBER]; NORM[1,2,3]; ENTROPY[0,1]; &
     &NCOR[0,1]'
      WRITE(*,*)' '
      WRITE(*,*)'NORM(1)=ABSOLUTE;NORM(2)=PCA-EUCLIDEAN;NORM(3)=MAXIMIN'
      WRITE(*,*)'ENTROPY(0)=MAXIMIZES NORM;ENTROPY(1)=MAXIMIZES ENTROPY'
      WRITE(*,*)'NCOR(0)=PRODUCT MOMENT; NCOR(1)=ABSOLUTE CORRELATION'
      READ(*,*) IU,NORM,ENTROPY,NCOR
      WRITE(*,*)'WANT RANK SCORE OPTIMIZATION? YES(1); NO(OTHER THAN 1)'
      READ(*,*) MRNK
      WRITE(*,*)'INPUT FILE TO READ DATA:YOUR DATA MUST BE IN THIS FILE'
      WRITE(*,*)'CASES (NOB) IN ROWS ; VARIABLES (MVAR) IN COLUMNS'
      READ(*,*) INFILE
      WRITE(*,*)'SPECIFY THE OUTPUT FILE TO STORE THE RESULTS'
      READ(*,*) OUTFILE
      OPEN(9, FILE=OUTFILE)
      OPEN(7,FILE=INFILE)
      DO I=1,NOB
      READ(7,*),CDA,(CDAT(I,J),J=1,MVAR)
      ENDDO
      CLOSE(7)
      DO I=1,NOB
      DO J=1,MVAR
      ZDAT(I,J+1)=CDAT(I,J)
      ENDDO
      ENDDO
```

```
      WRITE(*,*)'DATA HAS BEEN READ. WOULD YOU UNITIZE VARIABLES? [YES=1
     & ELSE NO UNITIZATION]'
      WRITE(*,*)'UNITIZE MEANS TRANSFORMATION FROM X(I,J) TO UNITIZED X'
      WRITE(*,*)'[X(I,J)-MIN(X(.,J))]/[MAX(X(.,J))-MIN(X(.,J))]'
      READ(*,*) NUN
      IF(NUN.EQ.1) THEN
      DO J=1,MVAR
      CMIN=CDAT(1,J)
      CMAX=CDAT(1,J)
      DO I=2,NOB
      IF(CMIN.GT.CDAT(I,J)) CMIN=CDAT(I,J)
      IF(CMAX.LT.CDAT(I,J)) CMAX=CDAT(I,J)
      ENDDO
      DO I=1,NOB
      CDAT(I,J)=(CDAT(I,J)-CMIN)/(CMAX-CMIN)
      ENDDO
      ENDDO
      ENDIF
C     -----------------------------------------------------
      WRITE(*,*)' '
      WRITE(*,*)'FEED RANDOM NUMBER SEED [4-DIGIT ODD INTEGER] TO BEGIN'
      READ(*,*) IU
C     THIS XBAS WILL BE USED AS INITIAL X
      DO I=1,1000
      DO J=1,50
      CALL RANDOM(RAND)
      XBAS(I,J)=RAND ! RANDOM NUMBER BETWEEN (0, 1)
      ENDDO
      ENDDO
C     -----------------------------------------------------------------
C     HOWEVER, THE FIRST ROW OF, THAT IS, XBAS(1,J),J=1,MVAR) MAY BE
C     SPECIFIED HERE IF THE USER KNOWS IT TO BE OPTIMAL OR NEAR-OPTIMAL
C     DATA (XBAS(1,J),J=1,MVAR) /DATA1, DATA2, ............, DATAMVAR/
C     -----------------------------------------------------------------
      WRITE(*,*)' ****************************************************'
C     -----------------------------------------------------------------
      DO I=1,NOPT
      IF(I.EQ.1) THEN
      WRITE(*,*)'==== WELCOME TO DE/RPS PROGRAM FOR INDEX CONSTRUCTION'
      CALL DE(M,X,FMINDE,Q0,Q1) !CALLS DE AND RETURNS OPTIMAL X AND FMIN
      IF(KF.EQ.1) THEN
      WRITE(9,*)'DIFFERENTIAL EVALUATION OPTIMIZATION RESULTS'
      RSUM1=0.D0
      RSUM2=0.D0
      DO J=1,MVAR
      RSUM1=RSUM1+DABS(R(J))
      RSUM2=RSUM2+DABS(R(J))**2
      ENDDO
      WRITE(9,*)'CORRELATION OF INDEX WITH CONSTITUENT VARIABLES'
      WRITE(9,*)(R(J),J=1,MVAR)
      WRITE(9,*)'SUM OF ABS (R)=',RSUM1,';   SUM OF SQUARE(R)=',RSUM2
      WRITE(9,*)'THE INDEX OR SCORE OF DIFFERENT CASES'
      DO II=1,NOB
      WRITE(9,*)QIND(II)
      FRANK(II)=QIND(II)
      ENDDO
      ENDIF
      FMIN=FMINDE
      ENDIF
C     -----------------------------------------------------------------
      DO J=1,M
      XX(I,J)=X(J)
      ENDDO
      KKF(I)=KF
      MM(I)=M
      FMINN(I)=FMIN
      ENDDO
```

```fortran
      WRITE(*,*)' '
      WRITE(*,*)' '
      WRITE(*,*)'---------------------- FINAL RESULTS=================='
      DO I=1,NOPT
      WRITE(*,*)'FUNCT CODE=',KKF(I),'  FMIN=',FMINN(I),' : DIM=',MM(I)
      WRITE(*,*)'OPTIMAL DECISION VARIABLES : ',METHOD(I)
      WRITE(*,*) 'WEIGHTS ARE AS FOLLOWS --------------'
      WRITE(9,*) 'WEIGHTS ARE AS FOLLOWS --------------'
      WRITE(9,*)(XX(I,J),J=1,M)
      WRITE(*,*)(XX(I,J),J=1,M)
      WRITE(*,*)'//////////////////////////////////////////////////////'
      ENDDO
      WRITE(*,*)'OPTIMIZATION PROGRAM ENDED'
      WRITE(*,*)'*******************************************************'
      WRITE(*,*)'MEASURE OF EQUALITY/INEQUALITY'
      WRITE(*,*)'DE: BEFORE AND AFTER OPTIMIZATION = ',Q0,Q1
      WRITE(*,*)' '
      WRITE(*,*)'RESULTS STORED IN FILE= ',OUTFILE
      WRITE(*,*)'OPEN BY MSWORD OR EDIT OR ANY OTHER EDITOR'
      WRITE(*,*)' '
      WRITE(*,*)'NOTE:VECTORS OF CORRELATIONS & INDEX(BOTH TOGETHER) ARE
     & IDETERMINATE FOR SIGN &   MAY BE MULTIPLED BY (-1) IF NEEDED'
      WRITE(*,*)'THAT IS IF R(J) IS TRANSFORMED TO -R(J) FOR ALL J THEN
     &THE INDEX(I) TOO IS         TRANSFORMED TO -INDEX(I) FOR ALL I'
      WRITE(9,*)' '
      WRITE(9,*)'NOTE: VECTORS OF CORRELATIONS AND INDEX (BOTH TOGETHER)
     & ARE IDETERMINATE FOR SIGN AND MAY BE MULTIPLED BY (-1) IF NEEDED'
      WRITE(9,*)'THAT IS IF R(J) IS TRANSFORMED TO -R(J) FOR ALL J THEN
     &THE INDEX(I) TOO IS TRANSFORMED TO -INDEX(I) FOR ALL I'
      CALL DORANK(FRANK,NOB)
      DO I=1,NOB
      ZDAT(I,1)=FRANK(I)
      ENDDO
      CALL CORREL(ZDAT,NOB,MVAR+1,RMAT)
      WRITE(9,*)'-------------------- CORRELATION MATRIX --------------'
      WRITE(*,*)'-------------------- CORRELATION MATRIX --------------'
      DO I=1,MVAR+1
      WRITE(9,1)(RMAT(I,J),J=1,MVAR+1)
      WRITE(*,1)(RMAT(I,J),J=1,MVAR+1)
      ENDDO
    1 FORMAT(8F10.6)
      WRITE(9,*)'====================================================  '
      WRITE(*,*)'====================================================  '
      WRITE(9,*)'VARIABLES: 1ST IS THE INDEX AND 2ND THE RANK OF INDEX'
      WRITE(*,*)'VARIABLES: 1ST IS THE INDEX AND 2ND THE RANK OF INDEX'
      WRITE(9,*)'====================================================  '
      WRITE(*,*)'====================================================  '
      DO I=1,NOB
      IF(MRNK.EQ.1) THEN
      QIND(I)=0.D0
      DO J=1,MVAR
      QIND(I)=QIND(I)+ZDAT(I,J+1)*XX(NOPT,J)
      ENDDO
      ENDIF
      WRITE(9,2)I,QIND(I),(ZDAT(I,J),J=1,MVAR+1)
      WRITE(*,2)I,QIND(I),(ZDAT(I,J),J=1,MVAR+1)
      ENDDO
    2 FORMAT(I3,F12.6,13F5.0)
      SR2=0.D0
      DO J=2,MVAR+1
      SR2=SR2+RMAT(1,J)**2
      ENDDO
      WRITE(9,*)'SUM OF SQUARE OF CORRELATION R(RANK(INDEX),VAR)=',SR2
      WRITE(*,*)'SUM OF SQUARE OF CORRELATION R(RANK(INDEX),VAR)=',SR2
      CLOSE(9)
      WRITE(*,*) 'THE JOB IS OVER'
      END
```

```fortran
C     -----------------------------------------------------------------
      SUBROUTINE DE(M,A,FBEST,G0,G1)
C     PROGRAM: "DIFFERENTIAL EVOLUTION ALGORITHM" OF GLOBAL OPTIMIZATION
C     THIS METHOD WAS PROPOSED BY R. STORN AND K. PRICE IN 1995. REF --
C     "DIFFERENTIAL EVOLUTION - A SIMPLE AND EFFICIENT ADAPTIVE SCHEME
C     FOR GLOBAL OPTIMIZATION OVER CONTINUOUS SPACES" : TECHNICAL REPORT
C     INTERNATIONAL COMPUTER SCIENCE INSTITUTE, BERKLEY, 1995.
C     PROGRAM BY SK MISHRA, DEPT. OF ECONOMICS, NEHU, SHILLONG (INDIA)
C     -----------------------------------------------------------------
C     PROGRAM DE
      IMPLICIT DOUBLE PRECISION (A-H, O-Z) ! TYPE DECLARATION
      PARAMETER(NMAX=500,MMAX=50) ! MAXIMUM DIMENSION PARAMETERS
      PARAMETER (RX1=1.0D0, RX2=1.0D0) ! TO BE ADJUSTED SUITABLY, IF NEEDED
C     RX1 AND RX2 CONTROL THE SCHEME OF CROSSOVER. (0 <= RX1 <= RX2) <=1
C     RX1 DETERMINES THE UPPER LIMIT OF SCHEME 1 (AND LOWER LIMIT OF
C     SCHEME 2; RX2 IS THE UPPER LIMIT OF SCHEME 2 AND LOWER LIMIT OF
C     SCHEME 3. THUS RX1 = .2 AND RX2 = .8 MEANS 0-20% SCHEME1, 20 TO 80
C     PERCENT SCHEME 2 AND THE REST (80 TO 100 %) SCHEME 3.
C     PARAMETER(NCROSS=2) ! CROSS-OVER SCHEME (NCROSS <=0 OR =1 OR =>2)
      PARAMETER(IPRINT=100,EPS=1.D-10)!FOR WATCHING INTERMEDIATE RESULTS
C     IT PRINTS THE INTERMEDIATE RESULTS AFTER EACH IPRINT ITERATION AND
C     EPS DETERMINES ACCURACY FOR TERMINATION. IF EPS= 0, ALL ITERATIONS
C     WOULD BE UNDERGONE EVEN IF NO IMPROVEMENT IN RESULTS IS THERE.
C     ULTIMATELY "DID NOT CONVERGE" IS REOPORTED.
      COMMON /RNDM/IU,IV ! RANDOM NUMBER GENERATION (IU = 4-DIGIT SEED)
      INTEGER IU,IV      ! FOR RANDOM NUMBER GENERATION
      COMMON /KFF/KF,NFCALL,FTIT ! FUNCTION CODE, NO. OF CALLS * TITLE
      COMMON /XBASE/XBAS
      CHARACTER *70 FTIT ! TITLE OF THE FUNCTION
C     -----------------------------------------------------------------
C     THE PROGRAM REQUIRES INPUTS FROM THE USER ON THE FOLLOWING ------
C     (1) FUNCTION CODE (KF), (2) NO. OF VARIABLES IN THE FUNCTION (M);
C     (3) N=POPULATION SIZE (SUGGESTED 10 TIMES OF NO. OF VARIABLES, M,
C         FOR SMALLER PROBLEMS N=100 WORKS VERY WELL);
C     (4) PCROS = PROB. OF CROSS-OVER (SUGGESTED : ABOUT 0.85 TO .99);
C     (5) FACT = SCALE (SUGGESTED 0.5 TO .95 OR  1, ETC);
C     (6) ITER = MAXIMUM NUMBER OF ITERATIONS PERMITTED (5000 OR MORE)
C     (7) RANDOM NUMBER SEED (4 DIGITS INTEGER)
C     ----------------------------------------------------------------
      DIMENSION X(NMAX,MMAX),Y(NMAX,MMAX),A(MMAX),FV(NMAX)
      DIMENSION IR(3),XBAS(1000,50)
C     ----------------------------------------------------------------
C     ------- SELECT THE FUNCTION TO MINIMIZE AND ITS DIMENSION -------
      CALL FSELECT(KF,M,FTIT)
C     SPECIFY OTHER PARAMETERS ---------------------------------------
      WRITE(*,*)'POPULATION SIZE [N] AND NO. OF ITERATIONS [ITER] ?'
      WRITE(*,*)'SUGGESTED : N => 100 OR =>10.M; ITER 10000 OR SO'
      READ(*,*) N,ITER
      WRITE(*,*)'CROSSOVER PROBABILITY [PCROS] AND SCALE [FACT] ?'
      WRITE(*,*)'SUGGESTED : PCROS ABOUT 0.9; FACT=.5 OR LARGER BUT <=1'
      READ(*,*) PCROS,FACT
      WRITE(*,*)'RANDOM NUMBER SEED ?'
      WRITE(*,*)'A FOUR-DIGIT POSITIVE ODD INTEGER, SAY, 1171'
      READ(*,*) IU
      NFCALL=0 ! INITIALIZE COUNTER FOR FUNCTION CALLS
      GBEST=1.D30 ! TO BE USED FOR TERMINATION CRITERION
C     INITIALIZATION : GENERATE X(N,M) RANDOMLY
      DO I=1,N
      DO J=1,M
C      CALL RANDOM(RAND) ! GENERATES INITION X WITHIN
C      X(I,J)=(RAND-.5D00)*2000 ! GENERATES INITION X WITHIN
C     RANDOM NUMBERS BETWEEN -RRANGE AND +RRANGE (BOTH EXCLUSIVE)
      X(I,J)=XBAS(I,J)! TAKES THESE NUMBERS FROM THE MAIN PROGRAM
      ENDDO
      ENDDO
      WRITE(*,*)'COMPUTING --- PLEASE WAIT '
      IPCOUNT=0
```

```fortran
      DO 100 ITR=1,ITER   ! ITERATION BEGINS
C     ----------------------------------------------------------------
C     EVALUATE ALL X FOR THE GIVEN FUNCTION
      DO I=1,N
      DO J=1,M
      A(J)=X(I,J)
      ENDDO
      CALL FUNC(A,M,F)
C     STORE FUNCTION VALUES IN FV VECTOR
      FV(I)=F
      ENDDO
      IF(ITR.EQ.1) CALL GINI(FV,N,G0)
C     ---------------------------------------------------------------
C     FIND THE FITTEST (BEST) INDIVIDUAL AT THIS ITERATION
                FBEST=FV(1)
                KB=1
                DO IB=2,N
                     IF(FV(IB).LT.FBEST) THEN
                     FBEST=FV(IB)
                     KB=IB
                     ENDIF
                ENDDO
C     BEST FITNESS VALUE = FBEST : INDIVIDUAL X(KB)
C     ----------------------------------------------------------------
C     GENERATE OFFSPRINGS
      DO I=1,N       ! I LOOP BEGINS
C     INITIALIZE CHILDREN IDENTICAL TO PARENTS; THEY WILL CHANGE LATER
           DO J=1,M
           Y(I,J)=X(I,J)
           ENDDO
C     SELECT RANDOMLY THREE OTHER INDIVIDUALS
   20     DO IRI=1,3   ! IRI LOOP BEGINS
          IR(IRI)=0

          CALL RANDOM(RAND)
           IRJ=INT(RAND*N)+1
C     CHECK THAT THESE THREE INDIVIDUALS ARE DISTICT AND OTHER THAN I
           IF(IRI.EQ.1.AND.IRJ.NE.I) THEN
           IR(IRI)=IRJ
           ENDIF
           IF(IRI.EQ.2.AND.IRJ.NE.I.AND.IRJ.NE.IR(1)) THEN
           IR(IRI)=IRJ
           ENDIF
       IF(IRI.EQ.3.AND.IRJ.NE.I.AND.IRJ.NE.IR(1).AND.IRJ.NE.IR(2)) THEN
            IR(IRI)=IRJ
            ENDIF
            ENDDO   ! IRI LOOP ENDS
C     CHECK IF ALL THE THREE IR ARE POSITIVE (INTEGERS)
          DO IX=1,3
          IF(IR(IX).LE.0) THEN
          GOTO 20   ! IF NOT THEN REGENERATE
          ENDIF
          ENDDO
C     THREE RANDOMLY CHOSEN INDIVIDUALS DIFFERENT FROM I AND DIFFERENT
C     FROM EACH OTHER ARE IR(1),IR(2) AND IR(3)
C     ===================== RANDOMIZATION OF NCROSS ===================
C     RANDOMIZES NCROSS
      NCROSS=0
      CALL RANDOM(RAND)
      IF(RAND.GT.RX1) NCROSS=1 ! IF RX1=>1, SCHEME 2 NEVER IMPLEMENTED
      IF(RAND.GT.RX2) NCROSS=2 ! IF RX2=>1, SCHEME 3 NEVER IMPLEMENTED

C     ---------------------- SCHEME 1 ---------------------------------
C      NO CROSS OVER, ONLY REPLACEMENT THAT IS PROBABILISTIC
          IF(NCROSS.LE.0) THEN
          DO J=1,M       ! J LOOP BEGINS
          CALL RANDOM(RAND)
```

```
            IF(RAND.LE.PCROS) THEN ! REPLACE IF RAND < PCROS
            A(J)=X(IR(1),J)+(X(IR(2),J)-X(IR(3),J))*FACT ! CANDIDATE CHILD
            ENDIF
            ENDDO  ! J LOOP ENDS
            ENDIF
C     ----------------------- SCHEME 2 --------------------------------
C     THE STANDARD CROSSOVER SCHEME
C     CROSSOVER SCHEME (EXPONENTIAL) SUGGESTED BY KENNETH PRICE IN HIS
C     PERSONAL LETTER TO THE AUTHOR (DATED SEPTEMBER 29, 2006)
      IF(NCROSS.EQ.1) THEN
        CALL RANDOM(RAND)
        JR=INT(RAND*M)+1
        J=JR
    2   A(J)=X(IR(1),J)+FACT*(X(IR(2),J)-X(IR(3),J))
        J=J+1
        IF(J.GT.M) J=1
        IF(J.EQ.JR) GOTO 10
        CALL RANDOM(RAND)
        IF(PCROS.LE.RAND) GOTO 2
    6   A(J)=X(I,J)
        J=J+1
        IF(J.GT.M) J=1
        IF (J.EQ.JR) GOTO 10
        GOTO 6
   10   CONTINUE
      ENDIF
C     ------------------------ SCHEME 3 --------------------------------
C     ESPECIALLY SUITABLE TO NON-DECOMPOSABLE (NON-SEPERABLE) FUNCTIONS
C     CROSSOVER SCHEME (NEW) SUGGESTED BY KENNETH PRICE IN HIS
C     PERSONAL LETTER TO THE AUTHOR (DATED OCTOBER 18, 2006)
      IF(NCROSS.GE.2) THEN
          CALL RANDOM(RAND)
          IF(RAND.LE.PCROS) THEN
             CALL NORMAL(RN)
             DO J=1,M
             A(J)=X(I,J)+(X(IR(1),J)+ X(IR(2),J)-2*X(I,J))*RN
             ENDDO
           ELSE
            DO J=1,M
            A(J)=X(I,J)+(X(IR(1),J)- X(IR(2),J))! FACT ASSUMED TO BE 1
            ENDDO
           ENDIF
      ENDIF
C     -----------------------------------------------------------------
          CALL FUNC(A,M,F) ! EVALUATE THE OFFSPRING
          IF(F.LT.FV(I)) THEN ! IF BETTER, REPLACE PARENTS BY THE CHILD
          FV(I)=F
          DO J=1,M
          Y(I,J)=A(J)
          ENDDO
          ENDIF
      ENDDO   ! I LOOP ENDS
      DO I=1,N
      DO J=1,M
      X(I,J)=Y(I,J) ! NEW GENERATION IS A MIX OF BETTER PARENTS AND
C                     BETTER CHILDREN
      ENDDO
      ENDDO
      DO J=1,M
      A(J)=X(KB,J)
      ENDDO
      IPCOUNT=IPCOUNT+1
      IF(IPCOUNT.EQ.IPRINT) THEN

      WRITE(*,*)(X(KB,J),J=1,M),'  FBEST UPTO NOW = ',FBEST
      WRITE(*,*)'TOTAL NUMBER OF FUNCTION CALLS =',NFCALL
           IF(DABS(FBEST-GBEST).LT.EPS) THEN
```

```
            WRITE(*,*) FTIT
            WRITE(*,*)'COMPUTATION OVER'
            GOTO 999
            ELSE
            GBEST=FBEST
            ENDIF
       IPCOUNT=0
       ENDIF
C      -----------------------------------------------------------------
  100 ENDDO   ! ITERATION ENDS : GO FOR NEXT ITERATION, IF APPLICABLE
C      -----------------------------------------------------------------
       WRITE(*,*)'DID NOT CONVERGE. REDUCE EPS OR RAISE ITER OR DO BOTH'
       WRITE(*,*)'INCREASE N, PCROS, OR SCALE FACTOR (FACT)'
  999 CALL FUNC(A,M,FBEST)
       CALL GINI(FV,N,G1)
       RETURN
       END
C      ------------------------------------------------------------------
       SUBROUTINE NORMAL(R)
C      PROGRAM TO GENERATE N(0,1) FROM RECTANGULAR RANDOM NUMBERS
C      IT USES BOX-MULLER VARIATE TRANSFORMATION FOR THIS PURPOSE.
C      ------------------------------------------------------------------
C      ----- BOX-MULLER METHOD BY GEP BOX AND ME MULLER (1958) ---------
C      BOX, G. E. P. AND MULLER, M. E. "A NOTE ON THE GENERATION OF
C      RANDOM NORMAL DEVIATES." ANN. MATH. STAT. 29, 610-611, 1958.
C      IF U1 AND U2 ARE UNIFORMLY DISTRIBUTED RANDOM NUMBERS (0,1),
C      THEN X=[(-2*LN(U1))**.5]*(COS(2*PI*U2) IS N(0,1)
C      ALSO,  X=[(-2*LN(U1))**.5]*(SIN(2*PI*U2) IS N(0,1)
C      PI = 4*ARCTAN(1.0)= 3.1415926535897932384626433832795
C      2*PI = 6.283185307179586476925286766559
C      ------------------------------------------------------------------
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       COMMON /RNDM/IU,IV
       INTEGER IU,IV
C      ------------------------------------------------------------------
       CALL RANDOM(RAND) ! INVOKES RANDOM TO GENERATE UNIFORM RAND [0, 1]
       U1=RAND ! U1 IS UNIFORMLY DISTRIBUTED [0, 1]
       CALL RANDOM(RAND) ! INVOKES RANDOM TO GENERATE UNIFORM RAND [0, 1]
       U2=RAND ! U1 IS UNIFORMLY DISTRIBUTED [0, 1]
       R=DSQRT(-2.D0*DLOG(U1))
       R=R*DCOS(U2*6.283185307179586476925286766559D00)
C      R=R*DCOS(U2*6.28318530718D00)
       RETURN
       END
C      ------------------------------------------------------------------
C      RANDOM NUMBER GENERATOR (UNIFORM BETWEEN 0 AND 1 - BOTH EXCLUSIVE)
       SUBROUTINE RANDOM(RAND1)
        DOUBLE PRECISION  RAND1
        COMMON /RNDM/IU,IV
       INTEGER IU,IV
        IV=IU*65539
        IF(IV.LT.0) THEN
        IV=IV+2147483647+1
        ENDIF
        RAND=IV
        IU=IV
        RAND=RAND*0.4656613E-09
        RAND1= DBLE(RAND)
        RETURN
        END
C      ------------------------------------------------------------------
       SUBROUTINE GINI(F,N,G)
       PARAMETER (K=1) !K=1 GINI COEFFICENT; K=2 COEFFICIENT OF VARIATION
C      THIS PROGRAM COMPUTES MEASURE OF INEQUALITY
C      IF K =1 GET THE GINI COEFFICIENT. IF K=2 GET COEFF OF VARIATIONE
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       DIMENSION F(*)
```

```fortran
      S=0.D0
      DO I=1,N
      S=S+F(I)
      ENDDO
      S=S/N
      H=0.D00
      DO I=1,N-1
      DO J=I+1,N
      H=H+(DABS(F(I)-F(J)))**K
      ENDDO
      ENDDO
      H=(H/(N**2))**(1.D0/K)! FOR K=1 H IS MEAN DEVIATION;
C                            FOR K=2 H IS STANDARD DEVIATION
      WRITE(*,*)'MEASURES OF DISPERSION AND CENTRAL TENDENCY = ',G,S
      G=DEXP(-H)! G IS THE MEASURE OF EQUALITY (NOT GINI OR CV)
C     G=H/DABS(S) !IF S NOT ZERO, K=1 THEN G=GINI, K=2 G=COEFF VARIATION
      RETURN
      END
C     -----------------------------------------------------------------
      SUBROUTINE FSELECT(KF,M,FTIT)
C     THE PROGRAM REQUIRES INPUTS FROM THE USER ON THE FOLLOWING ------
C     (1) FUNCTION CODE (KF), (2) NO. OF VARIABLES IN THE FUNCTION (M);
      CHARACTER *70 TIT(100),FTIT
      NFN=1
      KF=1
      WRITE(*,*)'----------------------------------------------------'
      DATA TIT(1)/'CONSTRUCTION OF INDEX FROM M VARIABLES '/
C     -----------------------------------------------------------------
      DO I=1,NFN
      WRITE(*,*)TIT(I)
      ENDDO
      WRITE(*,*)'----------------------------------------------------'
      WRITE(*,*)'SPECIFY NO. OF VARIABLES [MVAR] HERE ALSO ?'
      READ(*,*) M
      FTIT=TIT(KF) ! STORE THE NAME OF THE CHOSEN FUNCTION IN FTIT
      RETURN
      END
C     -----------------------------------------------------------------
      SUBROUTINE FUNC(X,M,F)
C     TEST FUNCTIONS FOR GLOBAL OPTIMIZATION PROGRAM
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      COMMON /KFF/KF,NFCALL,FTIT
      INTEGER IU,IV
      DIMENSION X(*)
      CHARACTER *70 FTIT
      NFCALL=NFCALL+1 ! INCREMENT TO NUMBER OF FUNCTION CALLS
C     KF IS THE CODE OF THE TEST FUNCTION
      IF(KF.EQ.1) THEN
      CALL CORD(M,X,F)
      RETURN
      ENDIF
C     =================================================================
      WRITE(*,*)'FUNCTION NOT DEFINED. PROGRAM ABORTED'
      STOP
      END
C     ------------------------------------------------------------------
      SUBROUTINE CORD(M,X,F)
      PARAMETER (NOB=30,MVAR=7)! CHANGE THE PARAMETERS HERE AS NEEDED.
C     ------------------------------------------------------------------
C     NOB=NO. OF OBSERVATIONS (CASES) & MVAR= NO. OF VARIABLES
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      COMMON /CORDAT/CDAT(NOB,MVAR),QIND(NOB),R(MVAR),ENTROPY,NORM,NCOR
      COMMON /GETRANK/MRNK
      INTEGER IU,IV
      DIMENSION X(*),Z(NOB,2)
```

```
      DO I=1,M
      IF(X(I).LT.-1.0D0.OR.X(I).GT.1.0D0) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-0.5D0)*2
      ENDIF
      ENDDO
      XNORM=0.D0
      DO J=1,M
      XNORM=XNORM+X(J)**2
      ENDDO
      XNORM=DSQRT(XNORM)
      DO J=1,NOB
      X(J)=X(J)/XNORM
      ENDDO
C     CONSTRUCT INDEX
      DO I=1,NOB
      QIND(I)=0.D0
      DO J=1,M
      QIND(I)=QIND(I)+CDAT(I,J)*X(J)
      ENDDO
      ENDDO
C     ----------------------------------------------------------------
      !FIND THE RANK OF QIND
      IF(MRNK.EQ.1) CALL DORANK(QIND,NOB)
C     ----------------------------------------------------------------
C     COMPUTE CORRELATIONS
      DO I=1,NOB
      Z(I,1)=QIND(I)
      ENDDO
      DO J=1,M
      DO I=1,NOB
      Z(I,2)=CDAT(I,J)
      ENDDO
      IF(NCOR.EQ.0) THEN
      CALL CORLN(Z,NOB,RHO)
      ELSE
      CALL CORA(Z,NOB,RHO)
      ENDIF
      R(J)=RHO
      ENDDO
      IF(ENTROPY.EQ.0.D0) THEN
C     ------------------ MAXIMIN SOLUTION ----------------------------
      IF(NORM.GT.2) THEN
       F=DABS(R(1))
       DO J=2,M
       IF(F.GT.DABS(R(J))) F= DABS(R(J))
       ENDDO
       ENDIF
C     ------------------ FOR  NORM =1 OR 2 ---------------------------
      IF(NORM.LE.2) THEN
      F=0.D0
      DO J=1,M
      F=F+DABS(R(J))**NORM
      ENDDO
      ENDIF
C     --------------------------------------------------------------
      ELSE
      IF(ENTROPY.NE.0.D0) THEN
C     ENTROPY MAXIMIZATION
      ENT=0.0D0
      DO J=1,M
      ENT=ENT+DABS(R(J))
      ENDDO
      F=ENT*DLOG(ENT)
      DO J=1,M
      FX=DABS(R(J))
      F=F+ (FX/ENT)*DLOG(FX/ENT)
```

```
      ENDDO
      ENDIF
      ENDIF
C     ----------------------------------------------------------------
      F=-F
      RETURN
      END
      SUBROUTINE CORLN(Z,NOB,RHO)
C     NOB = NO. OF CASES
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION Z(NOB,2),AV(2),SD(2)
      DO J=1,2
      AV(J)=0.D0
      SD(J)=0.D0
      DO I=1,NOB
      AV(J)=AV(J)+Z(I,J)
      SD(J)=SD(J)+Z(I,J)**2
      ENDDO
      ENDDO
      DO J=1,2
      AV(J)=AV(J)/NOB
      SD(J)=DSQRT(SD(J)/NOB-AV(J)**2)
      ENDDO
C      WRITE(*,*)'AV AND SD ', AV(1),AV(2),SD(1),SD(2)
      RHO=0.D0
      DO I=1,NOB
      RHO=RHO+(Z(I,1)-AV(1))*(Z(I,2)-AV(2))
      ENDDO
      RHO=(RHO/NOB)/(SD(1)*SD(2))
      RETURN
      END
C     ---------------------------------------------------------------
      SUBROUTINE CORA(Z,N,R)
C     COMPUTING ABSOLUTE CORRELATION MATRIX
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION Z(N,2),X(N),Y(N)
C     ----------------------------------------------------------------
C     PUT Z INTO X AND Y
      DO I=1,N
      X(I)=Z(I,1)
      Y(I)=Z(I,2)
      ENDDO
C     ARRANGE X ANY IN AN ASCENDING ORDER
      DO I=1,N-1
      DO II=I+1,N
      IF(X(I).GT.X(II)) THEN
      TEMP=X(I)
      X(I)=X(II)
      X(II)=TEMP
      ENDIF
      IF(Y(I).GT.Y(II)) THEN
      TEMP=Y(I)
      Y(I)=Y(II)
      Y(II)=TEMP
      ENDIF
      ENDDO
      ENDDO
C     FIND MEDIAN
      IF(INT(N/2).EQ.N/2.D0) THEN
      XMED=(X(N/2)+X(N/2+1))/2.D0
      YMED=(Y(N/2)+Y(N/2+1))/2.D0
      ENDIF
      IF(INT(N/2).NE.N/2.D0) THEN
      XMED=X(N/2+1)
      YMED=Y(N/2+1)
      ENDIF
C     SUBTRACT RESPECTIVE MEDIANS FROM X AND Y AND FIND ABS DEVIATIONS
```

```
      VX=0.D0
      VY=0.D0
      DO I=1,N
      X(I)=X(I)-XMED
      Y(I)=Y(I)-YMED
      VX=VX+DABS(X(I))
      VY=VY+DABS(Y(I))
      ENDDO
C     SCALE THE VARIABLES X AND Y SUCH THAT VX=VY
      IF(VX.EQ.0.D0.OR.VY.EQ.0.D0) THEN
      R=0.D0
      RETURN
      ENDIF
      DO I=1,N
      X(I)=X(I)*VY/VX
      ENDDO
C     COMPUTE CORRELATION COEFFICIENT
      VZ=0.D0
      R=0.D0
      DO I=1,N
      VZ=VZ+DABS(X(I))+DABS(Y(I))
      R=R+DABS(X(I)+Y(I))-DABS(X(I)-Y(I))
      ENDDO
      R=R/VZ
      RETURN
      END
C     -----------------------------------------------------------------
      SUBROUTINE DORANK(X,N)! N IS THE NUMBER OF OBSERVATIONS
      PARAMETER (MXD=1000)! MXD IS MAX DIMENSION FOR TEMPORARY VARIABLES
      ! THAT ARE LOCAL AND DO NOT GO TO THE INVOKING PROGRAM
      ! X IS THE VARIABLE TO BE SUBSTITUTED BY ITS RANK VALUES
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(N),ID(MXD)
      ! GENERATE ID
      DO I=1,N
      ID(I)=I
      ENDDO

      DO I=1,N-1
      DO II=I+1,N
      IF(X(I).GT.X(II)) THEN ! ARRANGE ACCORDING TO ASCENDING ORDER OF X
      T=X(I)
      X(I)=X(II)
      X(II)=T
      IT=ID(I)
      ID(I)=ID(II)
      ID(II)=IT
      ENDIF
      ENDDO
      ENDDO

      DO I=1,N
      X(I)=I+0.D0
      ENDDO

      DO I=1,N-1
      DO II=I+1,N
      IF(ID(I).GT.ID(II)) THEN!ARRANGE ACCORDING TO ASCENDNG ORDER OF ID
      T=X(I)
      X(I)=X(II)
      X(II)=T
      IT=ID(I)
      ID(I)=ID(II)
      ID(II)=IT
      ENDIF
      ENDDO
      ENDDO
```

```
      RETURN
      END
C     ----------------------------------------------------------------
      SUBROUTINE CORREL(X,N,M,RMAT)
      PARAMETER (NMX=30)!DO NOT CHANGE UNLESS NO. OF VARIABLES EXCEED 30
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(N,M),RMAT(M,M),AV(NMX),SD(NMX)
      DO J=1,M
      AV(J)=0.D0
      SD(J)=0.D0
      DO I=1,N
      AV(J)=AV(J)+X(I,J)
      SD(J)=SD(J)+X(I,J)**2
      ENDDO
      AV(J)=AV(J)/N
      SD(J)=DSQRT(SD(J)/N-AV(J)**2)
      ENDDO
      DO J=1,M
      DO JJ=1,M
      RMAT(J,JJ)=0.D0
      DO I=1,N
      RMAT(J,JJ)=RMAT(J,JJ)+X(I,J)*X(I,JJ)
      ENDDO
      ENDDO
      ENDDO
      DO J=1,M
      DO JJ=1,M
      RMAT(J,JJ)=RMAT(J,JJ)/N-AV(J)*AV(JJ)
      RMAT(J,JJ)=RMAT(J,JJ)/(SD(J)*SD(JJ))
      ENDDO
      ENDDO
      RETURN
      END
```