Performance of Differential Evolution and Particle Swarm Methods on Some Relatively Harder Multi-modal Benchmark Functions

SK Mishra

13. October 2006

Performance of Differential Evolution and Particle Swarm Methods on Some Relatively Harder Multi-modal Benchmark Functions

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

I. Introduction: Global optimization (GO) is concerned with finding the optimum point(s) of a non-convex (multi-modal) function in an m-dimensional space. Although some work was done in the pre-1970's to develop appropriate methods to find the optimal point(s) of a multi-modal function, the 1970's evidenced a great fillip in simulation-based optimization research due to the invention of the 'genetic algorithm' by John Holland (1975). A number of other methods of global optimization were soon developed. Among them, the 'Clustering Method" of Aimo Törn (1978), the "Simulated Annealing Method " of Kirkpatrick and others (1983) and Cerny (1985), "Tabu Search Method" of Fred Glover (1986), the "Ant Colony Algorithm" of Dorigo (1992), the "Particle Swarm Method" of Kennedy and Eberhart (1995) and the "Differential Evolution Method" of Price and Storn (1996) are quite effective and popular. All these methods use the one or the other stochastic process to search the global optima.

II. The Characteristic Features of Stochastic GO Methods: All stochastic search methods of global optimization partake of the probabilistic nature inherent to them. As a result, one cannot obtain certainty in their results, unless they are permitted to go in for indefinitely large search attempts. Larger is the number of attempts, greater is the probability that they would find out the global optimum, but even then it would not reach at the certainty. Secondly, all of them adapt themselves to the surface on which they find the global optimum. The scheme of adaptation is largely based on some guesswork since nobody knows as to the true nature of the problem (environment or surface) and the most suitable scheme of adaptation to fit the given environment. Surfaces may be varied and different for different functions. A particular type of surface may be suited to a particular method while a search in another type of surface may be a difficult proposition for it. Further, each of these methods operates with a number of parameters that may be changed at choice to make it more effective. This choice is often problem oriented and for obvious reasons. A particular choice may be extremely effective in a few cases, but it might be ineffective (or counterproductive) in certain other cases. Additionally, there is a relation of trade-off among those parameters. These features make all these methods a subject of trial and error exercises.

III. The Objectives: Our objective in this paper is to compare the performance of the Differential Evolution (DE) and the Repulsive Particle Swarm (RPS) methods of global optimization. To this end, some relatively difficult test functions have been chosen. Among these test functions, some are new while others are well known in the literature.

IV. Details of the Test Functions used in this Study: The following test (benchmark) functions have been used in this study.

(1) Perm function #1: In the domain x ∈ [-4, 4], the function has f_{min} =0 for x =(1, 2, 3, 4) . It is specified as

$$f(x)=\sum_{k=1}^{4}\left[\sum_{i=1}^{4}(i^k+\beta)\{(x_i/i)^k-1\}\right]^2$$

The value of β (=50) introduces difficulty to optimization. Smaller values of beta raise this difficulty further.

(2) Perm function #2: In the domain x ∈ [-1, 1], and for a given β (=10), this m-variable function has $f_{\min}=0 \;\; for\; x_i=(i)^{-1}:\; i=1,2,...,m$. It is specified as

$$\sum_{k=1}^{4}\left[\sum_{i=1}^{4}(i+\beta)\{(x_i)^k-(i)^{-k}\}\right]^2$$

Smaller values of beta raise difficulty in optimization.

(3) Power-sum function: Defined on four variables in the domain x ∈ [0, 4], this function has f_{min}=0 for any permutation of x = (1, 2, 2, 3). The function is defined as

$$f(x)=\sum_{k=1}^{4}\left[b_k-\sum_{i=1}^{4}x_i^k\right]^2;\; b_k=(8,\;18,\;44,\;114)\;\; for\; k=(1,\;2,\;3,\;4)$$ respectively.

(4) Bukin's functions: Bukin's functions are almost fractal (with fine seesaw edges) in the surroundings of their minimal points. Due to this property, they are extremely difficult to optimize by any method of global (or local) optimization and find correct values of decision variables (i.e. x_i for i=1,2). In the search domain $x_1\in[-15,\;-5], x_2\in[-3,\;3]$ the 6th Bukin's function is defined as follows.

$$f_6(x)=100\sqrt{\left|x_2-0.01x_1^2\right|}+0.01\left|x_1+10\right| \quad ;\; f_{\min}(-10,\;1)=0$$

(5) Zero-sum function: Defined in the domain x ∈ [-10, 10] this function (in m ≥ 2) has f(x) =0 if $\sum_{i=1}^{m}x_i=0$. Otherwise $f(x)=1+\left(10000\left|\sum_{i=1}^{m}x_i\right|\right)^{0.5}$. This function has innumerably many minima but it is extremely difficult to obtain any of them. Larger is the value of m (dimension), it becomes more difficult to optimize the function.

(6) Hougen function: Hougen function is typical complex test for classical non-linear regression problems. The Hougen-Watson model for reaction kinetics is an example of such non-linear regression problem. The form of the model is

$$rate=\frac{\beta_1x_2-x_3/\beta_5}{1+\beta_2x_1+\beta_3x_2+\beta_4x_3}$$

where the betas are the unknown parameters, $x=(x_1,\;x_2,\;x_3)$ are the explanatory variables and 'rate' is the dependent variable. The parameters are estimated via the least squares criterion. That is, the parameters are such that the sum of the squared differences between the observed responses and their fitted values of rate is minimized. The input data given alongside are used.

x_1	x_2	x_3	rate
470	300	10	8.55
285	80	10	3.79
470	300	120	4.82
470	80	120	0.02
470	80	10	2.75
100	190	10	14.39
100	80	65	2.54
470	190	65	4.35
100	300	54	13.00
100	300	120	8.50
100	80	120	0.05
285	300	10	11.32
285	190	120	3.13

The best results are obtained by the Rosenbrock-Quasi-Newton method: $\hat{\beta}_1 = 1.253031;\ \hat{\beta}_2 = 1.190943;\ \hat{\beta}_3 = 0.062798;\ \hat{\beta}_4 = 0.040063;\ \hat{\beta}_5 = 0.112453$. The sum of squares of deviations (S^2) is = 0.298900994 and the coefficient of correlation (R) between observed rate and expected rate is =0.99945.

(7) Giunta function: In the search domain $x_1,\ x_2 \in [-1,\ 1]$ this function is defined as follows and has $f_{\min}(0.45834282,\ 0.45834282) \simeq 0.0602472184$.

$$f(x) = 0.6 + \sum_{i=1}^{2}[\sin(\tfrac{16}{15}x_i - 1) + \sin^2(\tfrac{16}{15}x_i - 1) + \tfrac{1}{50}\sin(4(\tfrac{16}{15}x_i - 1))].$$

(8) DCS function: The generalized deflected corrugated spring function is an m-variable function with $f_{\min}(c_1, c_2, ..., c_m) = -1$. In case all c_i=α, then $f_{\min}(\alpha, \alpha, ..., \alpha) = -1$. For larger dimension it is a difficult function to optimize. In particular, one of its local minimum at f(x) = -0.84334 is very attractive and most of the optimization algorithms are attracted to and trapped by that local optimum. The DCS function is defined as:

$$f(x) = \left[\frac{1}{10}\right]\sum_{i=1}^{m}(x_i - c_i)^2 - \cos\left[k\left[\sum_{i=1}^{m}(x_i - c_i)^2\right]^{0.5}\right];\ x \in [-20,\ 20]$$

(9) Kowalik or Yao-Liu #15 function: It is a 4-variable function in the domain x ∈ [-5, 5], that has the global minimum $f_{\min}(0.19, 0.19, 0.12, 0.14) = 0.3075.$ This function is given as:

$f(x) = 1000\sum_{i=1}^{11}\left(a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right)^2$; where $b = \left(\frac{1}{0.25},\ \frac{1}{0.5},\ \frac{1}{1},\ \frac{1}{2},\ \frac{1}{4},\ \frac{1}{6},\ \frac{1}{8},\ \frac{1}{10},\ \frac{1}{12},\ \frac{1}{14},\ \frac{1}{16}\right)$ and

a = (0.1957, 0.1947, 0.1735, 0.1600, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.0235, 0.0246).

(10) Yao-Liu #7 function: It is an m-variable function in the domain x ∈ [-1.28, 1.28], that has the global minimum $f_{\min}(0,\ 0, ...,\ 0) = 0$. This function is given as:

$$f(x) = rand[0,1] + \sum_{i=1}^{m} i(x_i^4)$$

(11) Fletcher-Powell function: This is an m-variable function with $f_{\min}(c_1,\ c_2, ...,\ c_m) = 0$ given as follows:

$$f(x) = \sum_{i=1}^{m}(A_i - B_i)^2,$$

where $A_i = \sum_{j=1}^{m}\left[u_{ij}\sin(c_j) + v_{ij}\cos(c_j)\right];\quad B_i = \sum_{j=1}^{m}\left[u_{ij}\sin(x_j) + v_{ij}\cos(x_j)\right]\quad ;u_{ij},\ v_{ij} = rand[-100,\ 100]$

and $c_i \in [-\pi,\ \pi]$. Moreover, c could be stochastic or fixed. When c is fixed, it is easier to optimize, but for stochastic c optimization is quite difficult. One may visualize c as a vector that has two parts; c_1 (fixed) and r (random). Either of them could be zero or both of them could be non-zero.

(12) New function 1: This function (with $f_{\min}(1,\ 1, ...,\ 1) = 2$) may be defined as follows:

$$f(x) = (1 + x_m)^{x_m};\ x_m = m - \sum_{i=1}^{m-1} x_i;\ x \in [0,\ 1]\ \forall\ i = 1, 2, ..., m$$

This function is not very difficult to optimize. However, its modification that gives us a new function (#2) is considerably difficult.

(13) New function 2: This function (with $f_{\min}(1,\ 1,...,\ 1) = 2$) may be defined as follows:

$$f(x) = (1 + x_m)^{x_m};\ \ x_m = m - \sum_{i=1}^{m-1}(x_i + x_{i+1})/2;\ \ x \in [0,\ 1]\ \forall\ i = 1, 2, ..., m$$

Unlike the new function #1, here x_m of the prior iteration indirectly enters into the posterior iteration. As a result, this function is extremely difficult to optimize.

V. Some Details of the Particle Swarm Methods used Here: In this exercise we have used (modified) Repulsive Particle Swarm method. The Repulsive Particle Swarm method of optimization is a variant of the classical Particle Swarm method (see Wikipedia, http://en.wikipedia.org/wiki/RPSO). It is particularly effective in finding out the global optimum in very complex search spaces (although it may be slower on certain types of optimization problems).

In the traditional RPS the future velocity, v_{i+1} of a particle at position with a recent velocity, v_i, and the position of the particle are calculated by:

$$v_{i+1} = \omega v_i + \alpha r_1(\hat{x}_i - x_i) + \omega\beta r_2(\hat{x}_{hi} - x_i) + \omega\gamma r_3 z$$

$$x_{i+1} = x_i + v_{i+1}$$

where,

- x is the position and v is the velocity of the individual particle. The subscripts i and $i+1$ stand for the recent and the next (future) iterations, respectively.
- r_1, r_2 r_3 are random numbers, $\in [0,1]$
- ω is inertia weight, $\in [0.01, 0.7]$
- $\hat{x}$ is the best position of a particle
- x_h is best position of a randomly chosen other particle from within the swarm
- z is a random velocity vector
- α, β, γ are constants

Occasionally, when the process is caught in a local optimum, some *chaotic* perturbation in position as well as velocity of some particle(s) may be needed.

The traditional RPS gives little scope of local search to the particles. They are guided by their past experience and the communication received from the others in the swarm. We have modified the traditional RPS method by endowing stronger (wider) local search ability to each particle. Each particle flies in its local surrounding and searches for a better solution. The domain of its search is controlled by a new parameter (*nstep*). This local search has no preference to gradients in any direction and resembles closely to tunneling. This added exploration capability of the particles brings the RPS method closer to what we observe in real life.

Each particle learns from its 'chosen' inmates in the swarm. At the one extreme is to learn from the best performer in the entire swarm. This is how the particles in the original PS method learn. However, such learning is not natural. How can we expect the

individuals to know as to the best performer and interact with all others in the swarm? We believe in limited interaction and limited knowledge that any individual can possess and acquire. So, our particles do not know the 'best' in the swarm. Nevertheless, they interact with some chosen inmates that belong to the swarm. Now, the issue is: how does the particle choose its inmates? One of the possibilities is that it chooses the inmates closer (at lesser distance) to it. But, since our particle explores the locality by itself, it is likely that it would not benefit much from the inmates closer to it. Other relevant topologies are : (the celebrated) *ring topology*, ring topology hybridized with random topology, star topology, von Neumann topology, etc.

Now, let us visualize the possibilities of choosing (a predetermined number of) inmates randomly from among the members of the swarm. This is much closer to reality in the human world. When we are exposed to the mass media, we experience this. Alternatively, we may visualize our particles visiting a public place (e.g. railway platform, church, etc) where it (he) meets people coming from different places. Here, geographical distance of an individual from the others is not important. Important is how the experiences of others are communicated to us. There are large many sources of such information, each one being selective in what it broadcasts and each of us selective in what we attend to and, therefore, receive. This selectiveness at both ends transcends the geographical boundaries and each one of us is practically exposed to randomized information. Of course, two individuals may have a few common sources of information. We have used these arguments in the scheme of dissemination of others' experiences to each individual particle. Presently, we have assumed that each particle chooses a pre-assigned number of inmates (randomly) from among the members of the swarm. However, this number may be randomized to lie between two pre-assigned limits.

VI. Some Details of the Differential Evolution Methods used Here: The differential Evolution method consists of three basic steps: (i) generation of (large enough) population with N individuals [$x = (x_1, x_2, \ldots, x_m)$] 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(x); (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. Here the meaning of 'better' is in the Pareto improvement sense. A set S_a is better than another set S_b ***iff :*** (i) **no** $x_i \in S_a$ is inferior to the corresponding member of $x_i \in S_b$; ***and*** (ii) **at least one** member $x_k \in S_a$ is better than the corresponding member $x_k \in S_b$. Thus, every new population is an improvement over the earlier one. To accomplish this, the DE method generates a candidate individual to replace each current individual in the population. 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.

Algorithmically stated, 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.

The crossover scheme (called exponential crossover, as suggested by Kenneth Price in his personal letter to the author) is given below.

The mutant vector is **v**i,g = **x**r1,g + F*(**x**r2,g - **x**r3,g) and the target vector is **x**i,g and the trial vector is **u**i,g. The indices r1, r2 and r3 are randomly but different from each other. Uj(0,1) is a uniformly distributed random number between 0 and 1 that is chosen anew for each parameter as needed.

Step 1: Randomly pick a parameter index j = jrand.

Step 2: The trial vector inherits the jth parameter (initially = jrand) from the mutant vector, i.e., uj,i,g = vj,i,g.

Step 3: Increment j; if j = D then reset j = 0.

Step 4: If j = jrand end crossover; else goto Step 5.

Step 5: If Cr <= Uj(0,1), then goto Step 2; else goto Step 6

Step 6: The trial vector inherits the jth parameter from the target vector, i.e., uj,i,g = xj,i,g.

Step 7: Increment j; if j = D then reset j = 0.

Step 8: If j = jrand end crossover; else goto Step 6.

There could be other schemes (as many as 10 in number) of crossover, including no crossover (only probabilistic replacement only, which works better in case of a few functions).

VII. Specification of Adjustable Parameters: The RPS as well as the DE method needs some parameters to be specified by the user. In case of the RPS we have fixed the parameters as follows:

Population size, N=100; neighbour population, NN=50; steps for local search, NSTEP=11; Max no. of iterations permitted, ITRN=10000; chaotic perturbation allowed, NSIGMA=1; selection of neighbour : random, ITOP=3; A1=A2=0.5; A3=5.e-04; W=.5; SIGMA=1.e-03; EPSI=1.d-08. Meanings of these parameters are explained in the programs (appended).

In case of the DE, we have used *two alternatives*: first, the exponential crossover (ncross=1) as suggested by Price, and the second, only probabilistic replacement (but no crossover, ncross=0). We have fixed other parameters as: max number of iterations allowed, Iter = 10000, population size, N = 10 times of the dimension of the function or 100 whichever maximum; pcros = 0.9; scale factor, fact = 0.5, random number seed, iu = 1111 and all random numbers are uniformly distributed between -1000 and 1000; accuracy needed, eps =1.0e-08.

In case of either method, if x in f(x) violates the boundary then it is forcibly brought within the specified limits through replacing it by a random number lying in the given limits of the function concerned.

VIII. Findings and Discussion: Our findings are summarized in tables #1 through #3. The first table presents the results of the DE method when used with the exponential crossover scheme, while the table #2 presents the results of DE with no crossover (only probabilistic replacement). Table #3 presents the results of the RPS method.

A perusal of table #1 suggests that DE (with the exponential crossover scheme) mostly fails to find the optimum. Of course, it succeeds in case of some functions (perm#2, zero-sum) for very small dimension (m), but begins to falter as soon as the dimension is increased. In case of DCS function, it works well up to m (dimension) = 5.

Table-1: Differential Evolution (with exponential crossover: Ncross=1, Cr = 0.9, F = 0.5)

Function	M=2	M=4	M=5	M=10	M=20	M=30	M=50	M=100
Bukin-6	0.01545	-	-	-	-	-	-	-
Giunta	0.06447	-	-	-	-	-	-	-
Perm #1	-	2.45035	-	-	-	-	-	-
Power-Sum	-	0.00752	-	-	-	-	-	-
Kowalik	-	0.36724	-	-	-	-	-	-
Hougen	-	-	0.34228	-	-	-	-	-
New Fn #2	2.05349	2.03479	2.12106	2.57714				
Fletcher	0.02288	0.72118	3.13058	12027.6				
Perm #2	0	0.00549	0.07510	0.01005	0.15991			
Yao-Liu#7	0.02924	0.02244	0.03338	0.04643	0.19073	0.27375	0.51610	
Zero-Sum	0	1.73194	1.70313	1.26159	1.13372	1.74603	1.70230	1.22196
DCS	-1	-0.99994	-0.99994	-0.84334	-0.84334	0.40992	0.40992	1.50650

Table-2: Differential Evolution (without crossover: Ncross = 0, Cr = 0.9, F = 0.5)

Function	M=2	M=4	M=5	M=10	M=20	M=30	M=50	M=100
Bukin-6	0.02132	-	-	-	-	-	-	-
Giunta	0.06447	-	-	-	-	-	-	-
Perm #1	-	0.00000	-	-	-	-	-	-
Power-Sum	-	0.00000	-	-	-	-	-	-
Kowalik	-	0.30745	-	-	-	-	-	-
Hougen	-	-	0.29890	-	-	-	-	-
New Fn #2	2.03031	2.11842	2.11040	2.57009				
Fletcher	0.00862	2.84237	7.09278	261.377				
Perm #2	0	0.00009	0.00011	0.07219	13.355			
Yao-Liu#7	0.02984	0.00668	0.02467	0.02396	0.21920	0.22435	0.31691	
Zero-Sum	0	0	0	0	0	0	1.32197	1.30459
DCS	-1	-1	-0.84334	-0.84334	-0.84334	-0.37336	-0.37302	-0.37302

When we use no crossover (only probabilistic replacement) we obtain better results in case of several of the functions under study. Thus, overall, table #2 presents better results than what table #1 does. In case of Perm#1, Perm#2, Zero-sum, Kowalik, Hougen and Power-sum functions the advantage is clear.

Whether crossover or no crossover, DE falters when the optimand function has some element of randomness. This is indicated by the functions: Yao-Liu#7, Fletcher-Powell, and "New function#2". DE has no problems in optimizing the "New function #1". But the "New function #2" proves to be a hard nut. However, RPS performs much better for such stochastic functions. When the Fletcher-Powell function is optimized with non-stochastic c vector, DE works fine. But as soon as c is stochastic, it becomes unstable. Thus, it may be observed that *an introduction of stochasticity into the decision variables (or simply added to the function as in Yao-Liu#7) interferes with the fundamentals of DE, which works through attainment of better and better (in the sense of Pareto improvement) population at each successive iteration*.

Table-3: Repulsive Particle Swarm

Function	M=2	M=4	M=5	M=10	M=20	M=30	M=50	M=100
Bukin-6	0.75649	-	-	-	-	-	-	-
Giunta	0.06447	-	-	-	-	-	-	-
Perm #1	-	0.00000	-	-	-	-	-	-
Power-Sum	-	0.00000	-	-	-	-	-	-
Kowalik	-	0.30749	-	-	-	-	-	-
Hougen	-	-	0.42753	-	-	-	-	-
New Fn #2	2.00000	2.00021	2.00115	2.00644				
Fletcher	0.00003	0.09582	0.92413	40.70962				
Perm #2	0	0.00006	0.00011	0.00008	1.17420			
Yao-Liu#7	0.00000	0.00002	0.00000	0.00003	0.00007	0.00050	0.00060	
Zero-Sum	1.12312	1.14119	1.19259	1.21672	1.44780	1.02749	1.18303	1.56457
DCS	-1	-1	-1	-0.84334	-0.84334	-0.37336	-0.37336	11.68769

IX. Conclusion: Ours is a very small sample of test functions that we have used to compare DE (with and without crossover) and RPS methods. So the limitations of our conclusions are obvious. However, if any indication is obtained, those are: (1) for different types of problems, different schemes of crossover (including none) may be suitable or unsuitable, (2) Stochasticity entering into the optimand function may make DE unstable, but RPS may function well.

Bibliography

- Bauer, J.M.: "Harnessing the Swarm: Communication Policy in an Era of Ubiquitous Networks and Disruptive Technologies", *Communications and Strategies*, 45, 2002.
- Box, M.J.: "A new method of constrained optimization and a comparison with other methods". *Comp. J.* 8, pp. 42-52, 1965.
- Bukin, A. D.: *New Minimization Strategy For Non-Smooth Functions*, Budker Institute of Nuclear Physics preprint BUDKER-INP-1997-79, Novosibirsk 1997.
- Cerny, V.: "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm", *J. Opt. Theory Appl.*, 45, 1, 41-51, 1985.
- Eberhart R.C. and Kennedy J.: "A New Optimizer using Particle Swarm Theory", *Proceedings Sixth Symposium on Micro Machine and Human Science*, pp. 39–43. IEEE Service Center, Piscataway, NJ, 1995.
- Fleischer, M.: "Foundations of Swarm Intelligence: From Principles to Practice", Swarming Network Enabled C4ISR, arXiv:nlin.AO/0502003 v1 2 Feb 2005.
- G.E.P. Box, "Evolutionary operation: A method for increasing industrial productivity", *Applied Statistics*, 6 , pp. 81-101, 1957.
- Glover F.," Future paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research*, 5:533-549, 1986.
- Hayek, F.A.: *The Road to Serfdom*, Univ. of Chicago Press, Chicago, 1944.
- Holland, J.: *Adaptation in Natural and Artificial Systems,* Univ. of Michigan Press, Ann Arbor, 1975.
- Karush, W. *Minima of Functions of Several Variables with Inequalities as Side Constraints*. M.Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago, Chicago, Illinois, 1939.
- Kirkpatrick, S., Gelatt, C.D. Jr., and Vecchi, M.P.: "Optimization by Simulated Annealing", *Science*, 220, 4598, 671-680, 1983.
- Kuhn, H.W. and Tucker, A.W.: "Nonlinear Programming", in Neymann, J. (ed) *Proceedings of Second Berkeley Symposium on Mathematical Statistics and Probability,* Univ. of California Press, Berkrley, Calif. pp. 481-492, 1951.
- Metropolis, N. The Beginning of the Monte Carlo Method. *Los Alamos Science*, No. 15, Special Issue, pp. 125-130, 1987.
- Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E.: "Equation of State Calculations by Fast Computing Machines", *J. Chem. Phys.*,21, 6, 1087-1092, 1953.
- Mishra, S.K.: "Some Experiments on Fitting of Gielis Curves by Simulated Annealing and Particle Swarm Methods of Global Optimization", *Social Science Research Network* (SSRN): http://ssrn.com/abstract=913667, Working Papers Series, 2006 (a).
- Mishra, S.K.: "Least Squares Fitting of Chacón-Gielis Curves by the Particle Swarm Method of Optimization", *Social Science Research Network* (SSRN), Working Papers Series, http://ssrn.com/abstract=917762 , 2006 (b).
- Mishra, S.K.: "Performance of Repulsive Particle Swarm Method in Global Optimization of Some Important Test Functions: A Fortran Program" , *Social Science Research Network* (SSRN), Working Papers Series, http://ssrn.com/abstract=924339 , 2006 (c).
- Mishra, S.K.: "Some New Test Functions for Global Optimization and Performance of Repulsive Particle Swarm Method", *Social Science Research Network* (SSRN) Working Papers Series, http://ssrn.com/abstract=927134, 2006 (d).
- Mishra, S.K.: "Repulsive Particle Swarm Method on Some Difficult Test Problems of Global Optimization" ,SSRN: http://ssrn.com/abstract=928538 , 2006 (e).
- Nagendra, S.: *Catalogue of Test Problems for Optimization Algorithm Verification*, Technical Report 97-CRD-110, General Electric Company, 1997.
- Nelder, J.A. and Mead, R.: "A Simplex method for function minimization" *Computer Journal*, 7: pp. 308-313, 1964.

- Parsopoulos, K.E. and Vrahatis, M.N., "Recent Approaches to Global Optimization Problems Through Particle Swarm Optimization", *Natural Computing*, 1 (2-3), pp. 235- 306, 2002.
- Prigogine, I. and Strengers, I.: *Order Out of Chaos: Man's New Dialogue with Nature*, Bantam Books, Inc. NY, 1984.
- Silagadge, Z.K.: "Finding Two-Dimensional Peaks", Working Paper, Budkar Insttute of Nuclear Physics, Novosibirsk, Russia, arXive:physics/0402085 V3 11 Mar 2004.
- Simon, H.A.: *Models of Bounded Rationality*, Cambridge Univ. Press, Cambridge, MA, 1982.
- Smith, A.: *The Theory of the Moral Sentiments*, The Adam Smith Institute (2001 e-version), 1759.
- Sumper, D.J.T.: "The Principles of Collective Animal Behaviour", *Phil. Trans. R. Soc. B*. 361, pp. 5-22, 2006.
- Törn, A.A and Viitanen, S.: "Topographical Global Optimization using Presampled Points", *J. of Global Optimization*, 5, pp. 267-276, 1994.
- Törn, A.A.: "A search Clustering Approach to Global Optimization" , in Dixon, LCW and Szegö, G.P. (Eds) *Towards Global Optimization – 2*, North Holland, Amsterdam, 1978.
- Tsallis, C. and Stariolo, D.A.: "Generalized Simulated Annealing", *ArXive condmat/9501047 v1 12 Jan,* 1995.
- Valentine, R.H.: *Travel Time Curves in Oblique Structures*, Ph.D. Dissertation, MIT, Mass, 1937.
- Veblen, T.B.: "Why is Economics Not an Evolutionary Science" *The Quarterly Journal of Economics*, 12, 1898.
- Veblen, T.B.: *The Theory of the Leisure Class*, The New American library, NY. (Reprint, 1953), 1899.
- Vesterstrøm, J. and Thomsen, R.: "A comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems", *Congress on Evolutionary Computation, 2004. CEC2004***,** 2, pp. 1980-1987, 2004.
- Whitley, D., Mathias, K., Rana, S. and Dzubera, J.: "Evaluating Evolutionary Algorithms", *Artificial Intelligence*, 85, pp. 245-276, 1996.
- Yao, X. and Liu, Y.: "Fast Evolutionary Programming", in Fogel, LJ, Angeline, PJ and Bäck, T (eds) *Proc. 5th Annual Conf. on Evolutionary programming*, pp. 451-460, MIT Press, Mass, 1996.

```
1: C     MAIN PROGRAM : PROVIDES TO USE REPULSIVE PARTICLE SWARM METHOD
2: C     (SUBROUTINE RPS) AND DIFFERENTIAL WVOLUTION METHOD (DE)
3: c     ------------------------------------------------------------------
4: c     Adjust the parameters suitably in subroutines DE and RPS
5: c     When the program asks for parameters, feed them suitably
6: c     ------------------------------------------------------------------
7:       PROGRAM DERPS
8:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
9:       COMMON /KFF/KF,NFCALL ! FUNCTION CODE AND NO. OF FUNCTION CALLS
10:      CHARACTER *30  METHOD(2)
11:      CHARACTER *1 PROCEED
12:      DIMENSION XX(2,100),KKF(2),MM(2),FMINN(2)
13:      DIMENSION X(100)! X IS THE DECISION VARIABLE X IN F(X) TO MINIMIZE
14: C     M IS THE DIMENSION OF THE PROBLEM, KF IS TEST FUNCTION CODE AND
15: C     FMIN IS THE MIN VALUE OF F(X) OBTAINED FROM DE OR RPS
16:
17:      WRITE(*,*)'Adjust the parameters suitably in subroutines DE & RPS'
18:      WRITE(*,*)'====================     WARNING    ================ '
19:      METHOD(1)=' : DIFFERENTIAL EVALUATION'
20:      METHOD(2)=' : REPULSIVE PARTICLE SWARM'
21:      DO I=1,2
22:
23:      IF(I.EQ.1) THEN
24:      WRITE(*,*)'============ DIFFERENTIAL EVOLUTION PROGRAM ==========='
25:      WRITE(*,*)'TO PROCEED TYPE ANY CHARACTER AND STRIKE ENTER'
26:      READ(*,*) PROCEED
27:      CALL DE(M,X,FMINDE) ! CALLS DE AND RETURNS OPTIMAL X AND FMIN
28:      FMIN=FMINDE
29:      ELSE
30:      WRITE(*,*)' '
31:      WRITE(*,*)' '
32:      WRITE(*,*)'==========REPULSIVE PARTICLE SWARM PROGRAM ==========='
33:      WRITE(*,*)'TO PROCEED TYPE ANY CHARACTER AND STRIKE ENTER'
34:      READ(*,*) PROCEED
35:      CALL RPS(M,X,FMINRPS) ! CALLS RPS AND RETURNS OPTIMAL X AND FMIN
36:      FMIN=FMINRPS
37:      ENDIF
38:      DO J=1,M
39:      XX(I,J)=X(J)
40:      ENDDO
41:      KKF(I)=KF
42:      MM(I)=M
43:      FMINN(I)=FMIN
44:      ENDDO
45:      WRITE(*,*)' '
46:      WRITE(*,*)' '
47:      WRITE(*,*)'---------------------- FINAL RESULTS=================='
48:      DO I=1,2
49:      WRITE(*,*)'FUNCT CODE=',KKF(I),'  FMIN=',FMINN(I),' : DIM=',MM(I)
50:      WRITE(*,*)'OPTIMAL DECISION VARIABLES : ',METHOD(I)
51:      WRITE(*,*)(XX(I,J),J=1,M)
52:      WRITE(*,*)'//////////////////////////////////////////////////////'
53:      ENDDO
54:      WRITE(*,*)'PROGRAM ENDED'
55:      END
56: C     ------------------------------------------------------------------
57:      SUBROUTINE DE(M,A,FBEST)
58: C     PROGRAM: "DIFFERENTIAL EVOLUTION ALGORITHM" OF GLOBAL OPTIMIZATION
59: C     THIS METHOD WAS PROPOSED BY R. STORN AND K. PRICE IN 1995. REF --
60: C     "DIFFERENTIAL EVOLUTION - A SIMPLE AND EFFICIENT ADAPTIVE SCHEME
61: C     FOR GLOBAL OPTIMIZATION OVER CONTINUOUS SPACES" : TECHNICAL REPORT
62: C     INTERNATIONAL COMPUTER SCIENCE INSTITUTE, BERKLEY, 1995.
63: C     PROGRAM BY SK MISHRA, DEPT. OF ECONOMICS, NEHU, SHILLONG (INDIA)
64: C     ------------------------------------------------------------------
65: C      PROGRAM EVOLDIF
66:      IMPLICIT DOUBLE PRECISION (A-H, O-Z) ! TYPE DECLARATION
67:      PARAMETER(NMAX=1000,MMAX=100) ! MAXIMUM DIMENSION PARAMETERS
```

```
      PARAMETER(NCROSS=1) ! CROSS-OVER SCHEME (NCROSS <=0  OR =>1)
      PARAMETER(IPRINT=500,EPS=1.D-08)!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 ! FUNCTION CODE AND NO. OF FUNCTION CALLS
      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 SO);
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)
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: MAX(100, 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
      CALL RANDOM(RAND)
      X(I,J)=(RAND-.5D00)*2000
C     RANDOM NUMBERS BETWEEN -RRANGE AND +RRANGE (BOTH EXCLUSIVE)
      ENDDO
      ENDDO
      WRITE(*,*)'COMPUTING --- PLEASE WAIT '
      IPCOUNT=0
      DO 100 ITR=1,ITER  ! ITERATION BEGINS

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

```
202:           CALL FUNC(A,M,F) ! EVALUATE THE OFFSPRING
203:           IF(F.LT.FV(I)) THEN ! IF BETTER, REPLACE PARENTS BY THE CHILD
204:           FV(I)=F
205:           DO J=1,M
206:           Y(I,J)=A(J)
207:           ENDDO
208:           ENDIF
209:       ENDDO    ! I LOOP ENDS
210:       DO I=1,N
211:       DO J=1,M
212:       X(I,J)=Y(I,J) ! NEW GENERATION IS A MIX OF BETTER PARENTS AND
213: C                    BETTER CHILDREN
214:       ENDDO
215:       ENDDO
216:       IPCOUNT=IPCOUNT+1
217:       IF(IPCOUNT.EQ.IPRINT) THEN
218:       DO J=1,M
219:       A(J)=X(KB,J)
220:       ENDDO
221:       WRITE(*,*)(X(KB,J),J=1,M),'  FBEST UPTO NOW = ',FBEST
222:       WRITE(*,*)'TOTAL NUMBER OF FUNCTION CALLS =',NFCALL
223:            IF(DABS(FBEST-GBEST).LT.EPS) THEN
224:            WRITE(*,*) FTIT
225:            WRITE(*,*)'COMPUTATION OVER'
226:            RETURN
227:            ELSE
228:            GBEST=FBEST
229:            ENDIF
230:       IPCOUNT=0
231:       ENDIF
232: C     ----------------------------------------------------------------
233:   100 ENDDO    ! ITERATION ENDS : GO FOR NEXT ITERATION, IF APPLICABLE
234: C     ----------------------------------------------------------------
235:       WRITE(*,*)'DID NOT CONVERGE. REDUCE EPS OR RAISE ITER OR DO BOTH'
236:       WRITE(*,*)'INCREASE N, PCROS, OR SCALE FACTOR (FACT)'
237:       RETURN
238:       END
239: C     ----------------------------------------------------------------
240: C     RANDOM NUMBER GENERATOR (UNIFORM BETWEEN 0 AND 1 - BOTH EXCLUSIVE)
241:       SUBROUTINE RANDOM(RAND1)
242:        DOUBLE PRECISION  RAND1
243:        COMMON /RNDM/IU,IV
244:       INTEGER IU,IV
245:        RAND=REAL(RAND1)
246:        IV=IU*65539
247:        IF(IV.LT.0) THEN
248:        IV=IV+2147483647+1
249:        ENDIF
250:        RAND=IV
251:        IU=IV
252:        RAND=RAND*0.4656613E-09
253:        RAND1= (RAND)
254:        RETURN
255:        END
256: C     ----------------------------------------------------------------
257:       SUBROUTINE FSELECT(KF,M,FTIT)
258: C     THE PROGRAM REQUIRES INPUTS FROM THE USER ON THE FOLLOWING ------
259: C     (1) FUNCTION CODE (KF), (2) NO. OF VARIABLES IN THE FUNCTION (M);
260:       CHARACTER *70 TIT(100),FTIT
261:       WRITE(*,*)'-------------------------------------------------'
262:       DATA TIT(1)/'KF=1 NEW ZERO-SUM FUNCTION (N#7) M-VARIABLES M=?'/
263:       DATA TIT(2)/'KF=2 PERM FUNCTION #1 (SET BETA) 4-VARIABLES M=4'/
264:       DATA TIT(3)/'KF=3 PERM FUNCTION #2 (SET BETA) M-VARIABLES M=?'/
265:       DATA TIT(4)/'KF=4 POWER-SUM FUNCTION 4-VARIABLES M=4'/
266:       DATA TIT(5)/'KF=5 BUKIN 6TH FUNCTION 2-VARIABLES M=2'/
267:       DATA TIT(6)/'KF=6 DEFL CORRUG SPRING FUNCTION M-VARIABLES M=?'/
268:       DATA TIT(7)/'KF=7 YAO-LIU FUNCTION#7 M-VARIABLES M=?'/
```

```
      DATA TIT(8)/'KF=8 HOUGEN FUNCTION : 5-VARIABLES M=5'/
      DATA TIT(9)/'KF=9 GIUNTA FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(10)/'KF=10 KOWALIK FUNCTION : 4-VARIABLES M=4'/
      DATA TIT(11)/'KF=11 FLETCHER-POWELL FUNCTION : M-VARIABLES M=?'/
      DATA TIT(12)/'KF=12 NFUNCT#1 FUNCTION : M-VARIABLES M=?'/
      DATA TIT(13)/'KF=13 NFUNCT#2 FUNCTION : M-VARIABLES M=?'/
      DATA TIT(14)/'KF=14 WOOD FUNCTION : 4-VARIABLES M=4'/
      DATA TIT(15)/'KF=15 FENTON-EASON FUNCTION : 2-VARIABLES M=2'/
       DATA TIT(16)/'KF=16 TYPICAL NON-LINEAR FUNCTION:M-VARIABLES M=?'/
C     ------------------------------------------------------------------
      DO I=1,16
      WRITE(*,*)TIT(I)
      ENDDO
      WRITE(*,*)'-----------------------------------------------------'
      WRITE(*,*)'FUNCTION CODE [KF] AND NO. OF VARIABLES [M] ?'
      READ(*,*) KF,M
      FTIT=TIT(KF) ! STORE THE NAME OF THE CHOSEN FUNCTION IN FTIT
      RETURN
      END
C     ------------------------------------------------------------------
C     ================== REPULSIVE PARTICLE SWARM ====================
      SUBROUTINE RPS(M,BST,FMINIM)
C     PROGRAM TO FIND GLOBAL MINIMUM BY REPULSIVE PARTICLE SWARM METHOD
C     WRITTEN BY SK MISHRA, DEPT. OF ECONOMICS, NEHU, SHILLONG (INDIA)
C     ------------------------------------------------------------------
      PARAMETER(N=100,NN=50,MX=100,NSTEP=11,ITRN=100000,NSIGMA=1,ITOP=3)
      PARAMETER (NPRN=500) ! ECHOS RESULTS AT EVERY 500 TH ITERATION
C     PARAMETER(N=50,NN=25,MX=100,NSTEP=9,ITRN=10000,NSIGMA=1,ITOP=3)
C     PARAMETER (N=100,NN=15,MX=100,NSTEP=9,ITRN=10000,NSIGMA=1,ITOP=3)
C     IN CERTAIN CASES THE ONE OR THE OTHER SPECIFICATION WORKS BETTER
C     DIFFERENT SPECIFICATIONS OF PARAMETERS MAY SUIT DIFFERENT TYPES
C     OF FUNCTIONS OR DIMENSIONS - ONE HAS TO DO SOME TRIAL AND ERROR
C     ------------------------------------------------------------------
C     N = POPULATION SIZE. IN MOST OF THE CASES N=30 IS OK. ITS VALUE
C     MAY BE INCREASED TO 50 OR 100 TOO. THE PARAMETER NN IS THE SIZE OF
C     RANDOMLY CHOSEN NEIGHBOURS. 15 TO 25 (BUT SUFFICIENTLY LESS THAN
C     N) IS A GOOD CHOICE. MX IS THE MAXIMAL SIZE OF DECISION VARIABLES.
C     IN F(X1, X2,...,XM) M SHOULD BE LESS THAN OR EQUAL TO MX. ITRN IS
C     THE NO. OF ITERATIONS. IT MAY DEPEND ON THE PROBLEM. 200(AT LEAST)
C     TO 500 ITERATIONS MAY BE GOOD ENOUGH. BUT FOR FUNCTIONS LIKE
C     ROSENBROCKOR GRIEWANK OF LARGE SIZE (SAY M=30) IT IS NEEDED THAT
C     ITRN IS LARGE, SAY 5000 OR EVEN 10000.
C     SIGMA INTRODUCES PERTURBATION & HELPS THE SEARCH JUMP OUT OF LOCAL
C     OPTIMA. FOR EXAMPLE : RASTRIGIN FUNCTION OF DMENSION 3O OR LARGER
C     NSTEP DOES LOCAL SEARCH BY TUNNELLING AND WORKS WELL BETWEEN 5 AND
C     15, WHICH IS MUCH ON THE HIGHER SIDE.
C     ITOP <=1 (RING); ITOP=2 (RING AND RANDOM); ITOP=>3 (RANDOM)
C     NSIGMA=0 (NO CHAOTIC PERTURBATION);NSIGMA=1 (CHAOTIC PERTURBATION)
C     NOTE THAT NSIGMA=1 NEED NOT ALWAYS WORK BETTER (OR WORSE)
C     SUBROUTINE FUNC( ) DEFINES OR CALLS THE FUNCTION TO BE OPTIMIZED.
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      COMMON /KFF/KF,NFCALL
      INTEGER IU,IV
      CHARACTER *70 FTIT
      DIMENSION X(N,MX),V(N,MX),A(MX),VI(MX)
      DIMENSION XX(N,MX),F(N),V1(MX),V2(MX),V3(MX),V4(MX),BST(MX)
C     A1 A2 AND A3 ARE CONSTANTS AND W IS THE INERTIA WEIGHT.
C     OCCASIONALLY, TINKERING WITH THESE VALUES, ESPECIALLY A3, MAY BE
C     NEEDED.
      DATA A1,A2,A3,W,SIGMA,EPSI /.5D0,.5D0,5.D-04,.5D00,1.D-03,1.D-08/
C     ------------------------------------------------------------------
C     CALL SUBROUTINE FOR CHOOSING FUNCTION (KF) AND ITS DIMENSION (M)
      CALL FSELECT(KF,M,FTIT)
C     ------------------------------------------------------------------
      GGBEST=1.D30 ! TO BE USED FOR TERMINATION CRITERION
      LCOUNT=0
```

```
      NFCALL=0
      WRITE(*,*)'4-DIGITS SEED FOR RANDOM NUMBER GENERATION'
      WRITE(*,*)'A FOUR-DIGIT POSITIVE ODD INTEGER, SAY, 1171'
      READ(*,*) IU
      DATA FMIN /1.0E30/
C     GENERATE N-SIZE POPULATION OF M-TUPLE PARAMETERS X(I,J) RANDOMLY
      DO I=1,N
        DO J=1,M
        CALL RANDOM(RAND)
         X(I,J)=(RAND-0.5D00)*2000
C     WE GENERATE RANDOM(-5,5). HERE MULTIPLIER IS 10. TINKERING IN SOME
C     CASES MAY BE NEEDED
         ENDDO
        F(I)=1.0D30
      ENDDO
C     INITIALISE VELOCITIES V(I) FOR EACH INDIVIDUAL IN THE POPULATION
      DO I=1,N
      DO J=1,M
      CALL RANDOM(RAND)
       V(I,J)=(RAND-0.5D+00)
C       V(I,J)=RAND
      ENDDO
      ENDDO
      DO 100 ITER=1,ITRN
C     LET EACH INDIVIDUAL SEARCH FOR THE BEST IN ITS NEIGHBOURHOOD
        DO I=1,N
          DO J=1,M
          A(J)=X(I,J)
          VI(J)=V(I,J)
          ENDDO
          CALL LSRCH(A,M,VI,NSTEP,FI)
          IF(FI.LT.F(I)) THEN
           F(I)=FI
           DO IN=1,M
           BST(IN)=A(IN)
           ENDDO
C     F(I) CONTAINS THE LOCAL BEST VALUE OF FUNCTION FOR ITH INDIVIDUAL
C     XX(I,J) IS THE M-TUPLE VALUE OF X ASSOCIATED WITH LOCAL BEST F(I)
             DO J=1,M
             XX(I,J)=A(J)
             ENDDO
             ENDIF
        ENDDO
C      NOW LET EVERY INDIVIDUAL RANDOMLY COSULT NN(<<N) COLLEAGUES AND
C      FIND THE BEST AMONG THEM
      DO I=1,N
C     ---------------------------------------------------------------------
      IF(ITOP.GE.3) THEN
C     RANDOM TOPOLOGY ***************************************
C     CHOOSE NN COLLEAGUES RANDOMLY AND FIND THE BEST AMONG THEM
          BEST=1.0D30
           DO II=1,NN
                 CALL RANDOM(RAND)
               NF=INT(RAND*N)+1
                IF(BEST.GT.F(NF)) THEN
                 BEST=F(NF)
                NFBEST=NF
                 ENDIF
            ENDDO
      ENDIF
C---------------------------------------------------------------------------
      IF(ITOP.EQ.2) THEN
C     RING + RANDOM TOPOLOGY ********************************************
C     REQUIRES THAT THE SUBROUTINE NEIGHBOR IS TURNED ALIVE
       BEST=1.0D30
          CALL NEIGHBOR(I,N,I1,I3)
          DO II=1,NN
```

```
403:                 IF(II.EQ.1) NF=I1
404:                  IF(II.EQ.2) NF=I
405:                   IF(II.EQ.3) NF=I3
406:                       IF(II.GT.3) THEN
407:                      CALL RANDOM(RAND)
408:                       NF=INT(RAND*N)+1
409:                      ENDIF
410:                   IF(BEST.GT.F(NF)) THEN
411:                   BEST=F(NF)
412:                   NFBEST=NF
413:                  ENDIF
414:               ENDDO
415:        ENDIF
416: C-------------------------------------------------------------------------
417:       IF(ITOP.LE.1) THEN
418: C     RING TOPOLOGY ******************************************************
419: C     REQUIRES THAT THE SUBROUTINE NEIGHBOR IS TURNED ALIVE
420:         BEST=1.0D30
421:            CALL NEIGHBOR(I,N,I1,I3)
422:               DO II=1,3
423:               IF (II.NE.I) THEN
424:              IF(II.EQ.1) NF=I1
425:               IF(II.EQ.3) NF=I3
426:                   IF(BEST.GT.F(NF)) THEN
427:                    BEST=F(NF)
428:                    NFBEST=NF
429:                   ENDIF
430:                   ENDIF
431:             ENDDO
432:        ENDIF
433: C-------------------------------------------------------------------------
434: C     IN THE LIGHT OF HIS OWN AND HIS BEST COLLEAGUES EXPERIENCE, THE
435: C     INDIVIDUAL I WILL MODIFY HIS MOVE AS PER THE FOLLOWING CRITERION
436: C     FIRST, ADJUSTMENT BASED ON ONES OWN EXPERIENCE
437: C     AND OWN BEST EXPERIENCE IN THE PAST (XX(I))
438:            DO J=1,M
439:            CALL RANDOM(RAND)
440:            V1(J)=A1*RAND*(XX(I,J)-X(I,J))
441: C     THEN BASED ON THE OTHER COLLEAGUES BEST EXPERIENCE WITH WEIGHT W
442: C     HERE W IS CALLED AN INERTIA WEIGHT 0.01< W < 0.7
443: C     A2 IS THE CONSTANT NEAR BUT LESS THAN UNITY
444:            CALL RANDOM(RAND)
445:            V2(J)=V(I,J)
446:            IF(F(NFBEST).LT.F(I)) THEN
447:            V2(J)=A2*W*RAND*(XX(NFBEST,J)-X(I,J))
448:            ENDIF
449: C     THEN SOME RANDOMNESS AND A CONSTANT A3 CLOSE TO BUT LESS THAN UNITY
450:            CALL RANDOM(RAND)
451:            RND1=RAND
452:            CALL RANDOM(RAND)
453:             V3(J)=A3*RAND*W*RND1
454: C             V3(J)=A3*RAND*W
455: C     THEN ON PAST VELOCITY WITH INERTIA WEIGHT W
456:            V4(J)=W*V(I,J)
457: C     FINALLY A SUM OF THEM
458:            V(I,J)= V1(J)+V2(J)+V3(J)+V4(J)
459:            ENDDO
460:       ENDDO
461: C     CHANGE X
462:       DO I=1,N
463:       DO J=1,M
464:       RANDS=0.D00
465: C      ------------------------------------------------------------------
466:       IF(NSIGMA.EQ.1) THEN
467:        CALL RANDOM(RAND) ! FOR CHAOTIC PERTURBATION
468:        IF(DABS(RAND-.5D00).LT.SIGMA) RANDS=RAND-0.5D00
469: C     SIGMA CONDITIONED RANDS INTRODUCES CHAOTIC ELEMENT IN TO LOCATION
```

```
C     IN SOME CASES THIS PERTURBATION HAS WORKED VERY EFFECTIVELY WITH
C     PARAMETER (N=100,NN=15,MX=100,NSTEP=9,ITRN=100000,NSIGMA=1,ITOP=2)
      ENDIF
C     ----------------------------------------------------------------
      X(I,J)=X(I,J)+V(I,J)*(1.D00+RANDS)
      ENDDO
      ENDDO
       DO I=1,N
          IF(F(I).LT.FMIN) THEN
          FMIN=F(I)
          II=I
          DO J=1,M
          BST(J)=XX(II,J)
          ENDDO
          ENDIF
          ENDDO
      IF(LCOUNT.EQ.NPRN) THEN
      LCOUNT=0
      WRITE(*,*)'OPTIMAL SOLUTION UPTO THIS (FUNCTION CALLS=',NFCALL,')'
      WRITE(*,*)'X = ',(BST(J),J=1,M),' MIN F = ',FMIN
C      WRITE(*,*)'NO. OF FUNCTION CALLS = ',NFCALL
      IF(DABS(FMIN-GGBEST).LT.EPSI) THEN
           WRITE(*,*)'COMPUTATION OVER'
           FMINIM=FMIN
           RETURN
           ELSE
           GGBEST=FMIN
           ENDIF
      ENDIF
      LCOUNT=LCOUNT+1
  100 CONTINUE
      WRITE(*,*)'COMPUTATION OVER:',FTIT
      FMINIM=FMIN
      RETURN
      END
C     ----------------------------------------------------------------
      SUBROUTINE LSRCH(A,M,VI,NSTEP,FI)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /KFF/KF,NFCALL
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION A(*),B(100),VI(*)
      AMN=1.0D30
      DO J=1,NSTEP
         DO JJ=1,M
         B(JJ)=A(JJ)+(J-(NSTEP/2)-1)*VI(JJ)
         ENDDO
      CALL FUNC(B,M,FI)
        IF(FI.LT.AMN) THEN
        AMN=FI
        DO JJ=1,M
        A(JJ)=B(JJ)
        ENDDO
        ENDIF
      ENDDO
      FI=AMN
      RETURN
      END
C     ----------------------------------------------------------------
C     THIS SUBROUTINE IS NEEDED IF THE NEIGHBOURHOOD HAS RING TOPOLOGY
C     EITHER PURE OR HYBRIDIZED
       SUBROUTINE NEIGHBOR(I,N,J,K)
       IF(I-1.GE.1 .AND. I.LT.N) THEN
       J=I-1
       K=I+1
       ELSE
       IF(I-1.LT.1) THEN
```

```
      J=N-I+1
      K=I+1
      ENDIF
      IF(I.EQ.N) THEN
      J=I-1
      K=1
      ENDIF
      ENDIF
      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
      INTEGER IU,IV
      DIMENSION X(*)
      NFCALL=NFCALL+1 ! INCREMENT TO NUMBER OF FUNCTION CALLS
C     KF IS THE CODE OF THE TEST FUNCTION
C     ------------------------------------------------------------------
      IF(KF.EQ.1) THEN
C     ZERO SUM FUNCTION : MIN  = 0 AT SUM(X(I))=0
      CALL ZEROSUM(M,F,X)
      return
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.2) THEN
C     PERM FUNCTION #1 MIN = 0 AT (1, 2, 3, 4)
C     BETA => 0. CHANGE IF NEEDED. SMALLER BETA RAISES DIFFICULY
C     FOR BETA=0, EVERY PERMUTED SOLUTION IS A GLOBAL MINIMUM
      CALL PERM1(M,F,X)
      return
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.3) THEN
C     PERM FUNCTION #2 MIN = 0 AT (1/1, 1/2, 1/3, 1/4,..., 1/M)
C     BETA => 0. CHANGE IF NEEDED. SMALLER BETA RAISES DIFFICULY
C     FOR BETA=0, EVERY PERMUTED SOLUTION IS A GLOBAL MINIMUM
      CALL PERM2(M,F,X)
      return
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.4) THEN
C     POWER SUM FUNCTION; MIN = 0 AT PERM(1,2,2,3) FOR B=(8,18,44,114)
C     0 =< X <=4
      CALL POWERSUM(M,F,X)
      return
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.5) THEN
C     BUKIN'S 6TH FUNCTION  MIN = 0 FOR (-10, 1)
C     -15. LE. X(1) .LE. -5 AND -3 .LE. X(2) .LE. 3
      CALL BUKIN6(M,F,X)
      return
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.6) THEN
C     DEFLECTED CORRUGATED SPRING FUNCTION
C     MIN VALUE = -1 AT (5, 5, ..., 5) FOR ANY K AND ALPHA=5; M VARIABLE
      CALL DCS(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.7) THEN
C     M =>2
      CALL FUNCT7(M,F,X)
```

```
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.8) THEN
C     HOUGEN FUNCTION 5 VARIABLES : M =3
      CALL HOUGEN(X,M,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.9) THEN
C     GIUNTA FUNCTION 2 VARIABLES  :M =2
      CALL GIUNTA(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.10) THEN
C     FOR X(1)=0.1928D00; X(2)=0.1905D00; X(3)=0.1230D00; X(4)=0.1356D00
C     FMIN=0.3075D-03  ; -4 <= X(I) <= 4.
      CALL KOWALIK(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.11) THEN
C     FLETCHER & POWELL FUNCTION : F MIN f(0,0,...,0 )=0
      CALL FLETCHER(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.12) THEN
C     NEW FUNCTION 1 : F MIN f(0,0,...,0 )=0
      CALL NFUNCT1(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.13) THEN
C     NEW FUNCTION 2 : F MIN f(0,0,...,0 )=0
      CALL NFUNCT2(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.14) THEN
C     WOOD FUNCTION : F MIN  : M=4
      CALL WOOD(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.15) THEN
C     FENTON & EASON FUNCTION :   : M=2
      CALL FENTONEASON(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.16) THEN
      CALL TYPICAL(M,X,F)
      RETURN
      ENDIF
C     ==================================================================
      WRITE(*,*)'FUNCTION NOT DEFINED. PROGRAM ABORTED'
      STOP
      END
C     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      SUBROUTINE ZEROSUM(M,F,X)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*)
C     ZERO SUM FUNCTION : MIN  = 0 AT SUM(X(I))=0
```

```
      F=0.D00
      DO I=1,M
      IF(DABS(X(I)).GT.10.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-0.5D00)*20
      ENDIF
      ENDDO
      SUM=0.D00
      DO I=1,M
      SUM=SUM+X(I)
      ENDDO
      IF(SUM.NE.0.D00) F=1.D00+(10000*DABS(SUM))**0.5
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE PERM1(M,F,X)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*)
C     PERM FUNCTION #1 MIN = 0 AT (1, 2, 3, 4)
C     BETA => 0. CHANGE IF NEEDED. SMALLER BETA RAISES DIFFICULY
C     FOR BETA=0, EVERY PERMUTED SOLUTION IS A GLOBAL MINIMUM
      BETA=50.D00
      F=0.D00
      DO I=1,M
      IF(DABS(X(I)).GT.M) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-0.5D00)*2*M
      ENDIF
      ENDDO
      DO K=1,M
      SUM=0.D00
      DO I=1,M
      SUM=SUM+(I**K+BETA)*((X(I)/I)**K-1.D00)
      ENDDO
      F=F+SUM**2
      ENDDO
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE PERM2(M,F,X)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*)
C     PERM FUNCTION #2 MIN = 0 AT (1/1, 1/2, 1/3, 1/4,..., 1/M)
C     BETA => 0. CHANGE IF NEEDED. SMALLER BETA RAISES DIFFICULY
C     FOR BETA=0, EVERY PERMUTED SOLUTION IS A GLOBAL MINIMUM
      BETA=10.D00
      DO I=1,M
      IF(DABS(X(I)).GT.1.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-.5D00)*2
      ENDIF
      SGN=X(I)/DABS(X(I))
      ENDDO
      F=0.D00
      DO K=1,M
      SUM=0.D00
      DO I=1,M
      SUM=SUM+(I+BETA)*(X(I)**K-(1.D00/I)**K)
      ENDDO
      F=F+SUM**2
      ENDDO
      RETURN
      END
```

```
C     ------------------------------------------------------------------
      SUBROUTINE POWERSUM(M,F,X)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*)
C     POWER SUM FUNCTION; MIN = 0 AT PERM(1,2,2,3) FOR B=(8,18,44,114)
C     0 =< X <=4
      F=0.D00
      DO I=1,M
C     ANY PERMUTATION OF (1,2,2,3) WILL GIVE MIN = ZERO
      IF(X(I).LT.0.D00 .OR. X(I).GT.4.D00) THEN
      CALL RANDOM(RAND)
      X(I)=RAND*4
      ENDIF
      ENDDO
      DO K=1,M
      SUM=0.D00
      DO I=1,M
      SUM=SUM+X(I)**K
      ENDDO
      IF(K.EQ.1) B=8.D00
      IF(K.EQ.2) B=18.D00
      IF(K.EQ.3) B=44.D00
      IF(K.EQ.4) B=114.D00
      F=F+(SUM-B)**2
      ENDDO
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE BUKIN6(M,F,X)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*)
C     BUKIN'S 6TH FUNCTION  MIN = 0 FOR (-10, 1)
C     -15. LE. X(1) .LE. -5 AND -3 .LE. X(2) .LE. 3
        IF(X(1).LT. -15.D00 .OR. X(1) .GT. -5.D00) THEN
        CALL RANDOM(RAND)
        X(1)=-(RAND*10+5.D00)
        ENDIF
        IF(DABS(X(2)).GT.3.D00) THEN
        CALL RANDOM(RAND)
        X(2)=(RAND-.5D00)*6
        ENDIF
      F=100.D0*DSQRT(DABS(X(2)-0.01D0*X(1)**2))+ 0.01D0*DABS(X(1)+10.D0)
       RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE DCS(M,F,X)
C     FOR DEFLECTED CORRUGATED SPRING FUNCTION
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*),C(100)
C     OPTIMAL VALUES OF (ALL) X ARE ALPHA , AND K IS ONLY FOR SCALING
C     OPTIMAL VALUE OF F IS -1. DIFFICULT TO OPTIMIZE FOR LARGER M.
      DATA K,ALPHA/5,5.D00/ ! K AND ALPHA COULD TAKE ON ANY OTHER VALUES
      do i=1, m
      if(dabs(x(i)).gt.20.d00) then
      call random(rand)
      x(i)=(rand-0.5d00)*40
      endif
      enddo
      R2=0.D00
      DO I=1,M
      C(I)=alpha
```

```
      R2=R2+(X(I)-C(I))**2
      ENDDO
      R=DSQRT(R2)
      F=-DCOS(K*R)+0.1D00*R2
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE FUNCT7(M,F,X)
C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
C     MIN F(0, 0, ..., 0) = 0
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*)
      F=0.D00
        DO I=1,M
        IF(DABS(X(I)).GT.1.28D00) THEN
        CALL RANDOM(RAND)
        X(I)=(RAND-0.5D00)*2.56D00
        ENDIF
        ENDDO
      DO I=1,M
      CALL RANDOM(RAND)
      F=F+(I*X(I)**4)
      ENDDO
       CALL RANDOM(RAND)
       F=F+RAND
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE HOUGEN(A,M,F)
      PARAMETER(N=13,K=3)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(N,K),RATE(N),A(*)
C     ------------------------------------------------------------------
C     HOUGEN FUNCTION (HOUGEN-WATSON MODEL FOR REACTION KINATICS)
C     NO. OF PARAMETERS (A) TO ESTIMATE = 5 = M
c     BEST RESULTS ARE: FMIN=0.298900994
C     A(1)=1.253031; A(2)=1.190943; A(3)=0.062798; A(4)=0.040063
C     A(5)=0.112453  ARE 2nd BEST ESTIMATES OBTAINED BY ROSENBROCK &
C     QUASI-NEWTON METHOD WITH SUM OF SQUARES OF DEVIATION =0.298900994
C     AND R=0.99945.
c     SECOND BEST RESULTS: fmin=0.298901 for X(1.25258611. 0.0627758222,
c     0.0400477567, 0.112414812, 1.19137715) given by DE with ncross=0.
C     THE NEXT BEST RESULTS GIVEN BY HOOKE-JEEVES & QUASI-NEWTON
C     A(1)=2.475221;A(2)=0.599177; A(3)=0.124172; A(4)=0.083517
C     A(5)=0.217886; SUM OF SQUARES OF DEVIATION = 0.318593458
C     R=0.99941
C     MOST OF THE OTHER METHODS DO NOT PERFORM WELL
C     ------------------------------------------------------------------
      DATA X(1,1),X(1,2),X(1,3),RATE(1) /470,300,10,8.55/
      DATA X(2,1),X(2,2),X(2,3),RATE(2) /285,80,10,3.79/
      DATA X(3,1),X(3,2),X(3,3),RATE(3) /470,300,120,4.82/
      DATA X(4,1),X(4,2),X(4,3),RATE(4) /470,80,120,0.02/
      DATA X(5,1),X(5,2),X(5,3),RATE(5) /470,80,10,2.75/
      DATA X(6,1),X(6,2),X(6,3),RATE(6) /100,190,10,14.39/
      DATA X(7,1),X(7,2),X(7,3),RATE(7) /100,80,65,2.54/
      DATA X(8,1),X(8,2),X(8,3),RATE(8) /470,190,65,4.35/
      DATA X(9,1),X(9,2),X(9,3),RATE(9) /100,300,54,13/
      DATA X(10,1),X(10,2),X(10,3),RATE(10) /100,300,120,8.5/
      DATA X(11,1),X(11,2),X(11,3),RATE(11) /100,80,120,0.05/
      DATA X(12,1),X(12,2),X(12,3),RATE(12) /285,300,10,11.32/
      DATA X(13,1),X(13,2),X(13,3),RATE(13) /285,190,120,3.13/
C     WRITE(*,1)((X(I,J),J=1,K),RATE(I),I=1,N)
C   1 FORMAT(4F8.2)
```

```
872:       DO J=1,M
873:       IF(DABS(A(J)).GT.5.D00) THEN
874:       CALL RANDOM(RAND)
875:       A(J)=(RAND-0.5D00)*10.D00
876:       ENDIF
877:       ENDDO
878:       F=0.D00
879:       DO I=1,N
880:       D=1.D00
881:          DO J=1,K
882:          D=D+A(J+1)*X(I,J)
883:          ENDDO
884:       FX=(A(1)*X(I,2)-X(I,3)/A(M))/D
885: C     FX=(A(1)*X(I,2)-X(I,3)/A(5))/(1.D00+A(2)*X(I,1)+A(3)*X(I,2)+
886: C     A(4)*X(I,3))
887:       F=F+(RATE(I)-FX)**2
888:       ENDDO
889:       RETURN
890:       END
891: C     ----------------------------------------------------------------
892:       SUBROUTINE GIUNTA(M,X,F)
893:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
894:       COMMON /RNDM/IU,IV
895:       INTEGER IU,IV
896:       DIMENSION X(*)
897: C     GIUNTA FUNCTION
898: C     X(I) = -1 TO 1;  M=2
899:         DO I=1,M
900:         IF(DABS(X(I)).GT.1.D00) THEN
901:         CALL RANDOM(RAND)
902:         X(I)=(RAND-0.5D00)*2.D00
903:         ENDIF
904:         ENDDO
905:       C=16.D00/15.D00
906:       F=DSIN(C*X(1)-1.D0)+DSIN(C*X(1)-1.D0)**2+DSIN(4*(C*X(1)-1.D0))/50+
907:      &DSIN(C*X(2)-1.D0)+DSIN(C*X(2)-1.D0)**2+DSIN(4*(C*X(2)-1.D0))/50+.6
908:       RETURN
909:       END
910: C     ----------------------------------------------------------------
911:       SUBROUTINE KOWALIK(M,X,F)
912:       PARAMETER(N=11)
913:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
914:       COMMON /RNDM/IU,IV
915:       INTEGER IU,IV
916:       DIMENSION X(*),A(N),BI(N)
917:       DATA (A(I),I=1,N) /0.1957D0,0.1947D0,0.1735D0,0.1600D0,0.0844D0,
918:      & 0.0627D0, 0.0456D0, 0.0342D0, 0.0323D0, 0.0235D0, 0.0246D0/
919:       DATA (BI(I),I=1,N)/0.25D0, 0.5D0, 1.D0, 2.D0, 4.D0, 6.D0, 8.D0,
920:      & 10.D0, 12.D0, 14.D0, 16.D0/
921: C     KOWAKIK, J & MORRISON, JF (1968) : ANALYSIS OF KINETIC DATA FOR
922: C     ALLOSTERIC ANZYME REACTION AS A NONLINEAR REGRESSIO PROBLEM
923: C     MATH. BIOSC. 2: 57-66.
924: C     NOTE: FOR LARGER L AND U LIMITS THIS FUNCTION MISGUIDES DE, BUT
925: C     RPS IS QUITE ROBUST. DE with ncross=1 FAILS TO FIND MINIMUM, but
926: C     DE with ncross=0 SUCCEEDS.  RPS SUCCEEDS.
927:         DO I=1,M
928:           IF(DABS(X(I)).GT.4.D00) THEN
929:           CALL RANDOM(RAND)
930:           X(I)=(RAND-0.5D00)*8
931:           ENDIF
932:        ENDDO
933: C     FOR X(1)=0.1928D00; X(2)=0.1905D00; X(3)=0.1230D00; X(4)=0.1356D00
934: C     FMIN=0.3075D-03  ; -4 <= X(I) <= 4.
935:       F=0.D00
936:       DO I=1,N
937:       F1=X(1)*(BI(I)**(-2)+X(2)/BI(I))
938:       F2=BI(I)**(-2)+X(3)/BI(I)+X(4)
```

```
      F=F+(A(I)-F1/F2)**2
      ENDDO
      f=f*1000.d00
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE WOOD(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*)
C     WOOD FUNCTION  : M=4
      DO I=1,M
      IF(DABS(X(I)).GT.5.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-0.5D00)*10
      ENDIF
      ENDDO
      F1=100*(X(2)+X(1)**2)**2 + (1.D0-X(1))**2 +90*(X(4)-X(3)**2)**2
      F2=(1.D0-X(3))**2 +10.1*((X(2)-1.D0)**2+(X(4)-1.D0)**2)
      F3=19.8*(X(2)-1.D0)*(X(4)-1.D0)
      F=F1+F2+F3
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE FENTONEASON(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
C     FENTON & EASON FUNCTION   M=2
      DO I=1,M
      IF(DABS(X(I)).GT.100.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-0.5D00)*200
      ENDIF
      ENDDO
      F=1.2D00+0.1*X(1)**2 +(0.1D00+0.1*X(2)**2)/X(1)**2+
     & (.1*X(1)**2*X(2)**2+10.D00)/((X(1)*X(2))**4)
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE FLETCHER(M,X,F)
C     FLETCHER-POWELL FUNCTION, M <= 10, ELSE IT IS VERY MUCH SLOW
C     SOLUTION: MIN F = 0 FOR X=(C1, C2, C3,...,CM)
      PARAMETER(N=10) ! FOR DIMENSION OF DIFFERENT MATRICES AND VECTORS
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /KFF/KF,NFCALL
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*),A(N,N),B(N,N),AA(N),BB(N),AL(N),C(N),C1(N)
      PI=4*DATAN(1.D00)
C     GENERATE A(I,J) AND B(I,J) BETWEEN (-100, 100) RANDOMLY.
C     C(I) = BETWEEN (-PI, PI) IS EITHER GIVEN OR RANDOMLY GENERATED.
C     DATA (C(I),I=1,10)/1,2,3,-3,-2,-1,0,1,2,3/ ! BETWEEN -PI AND PI
      DATA (C1(I),I=1,N)/-3,-3.02,-3.01,1,1.03,1.02,1.03,-.08,.001,3/
C     DATA (C1(I),I=1,N)/0,0,0,0,0,0,0,0,0,0/ ! another example c1 = 0
      NC=1  ! DEFINE NC HERE 0 OR 1 OR 2
C     IF NC=0, C1 FROM DATA IS USED (THAT IS FIXED C);
C     IF NC=1, C IS MADE FROM C1 BY ADDING RANDOM PART - THAT IS C=C1+r
C     IF NC=2 THEN C IS PURELY RANDOM THAT IS C= 0 + r
C     IN ANY CASE C LIES BETWEEN -PI AND PI.
C     ------------------------------------------------------------------
C     FIND THE MAX MAGNITUDE ELEMENT IN C1 VECTOR (UPTO M ELEMENTS)
       CMAX=DABS(C1(1))
       DO J=2,M
       IF(DABS(C1(J)).GT.CMAX) CMAX=DABS(C1(J))
       ENDDO
       RANGE=PI-CMAX
```

```
1006: C     ------------------------------------------------------------------
1007:
1008:       DO J=1,M
1009:       DO I=1,M
1010:       CALL RANDOM(RAND)
1011:       A(I,J)=(RAND-0.5D00)*200.D00
1012:       CALL RANDOM(RAND)
1013:       B(I,J)=(RAND-0.5D00)*200.D00
1014:       ENDDO
1015:       IF(NC.EQ.0) AL(J)=C1(J) ! FIXED OR NON-STOCHASTIC C
1016:       IF(NC.EQ.1) THEN
1017:       CALL RANDOM(RAND)
1018:       AL(J)=C1(J)+(RAND-0.5D0)*2*RANGE ! A PART FIXED, OTHER STOCHASTIC
1019:       ENDIF
1020:       IF(NC.EQ.2) THEN
1021:       CALL RANDOM(RAND)
1022:       AL(J)=(RAND-0.5D00)*2*PI ! PURELY STOCHASTIC
1023:       ENDIF
1024:       ENDDO
1025:
1026:       DO I=1,M
1027:       AA(I)=0.D00
1028:       DO J=1,M
1029:       AA(I)=AA(I)+A(I,J)*DSIN(AL(J))+B(I,J)*DCOS(AL(J))
1030:       ENDDO
1031:       ENDDO
1032: C     ------------------------------------------------------------------
1033:       DO I=1,M
1034:       IF(DABS(X(I)).GT.PI) THEN
1035:       CALL RANDOM(RAND)
1036:       X(I)=(RAND-0.5D00)*2*PI
1037:       ENDIF
1038:       ENDDO
1039:
1040:       DO I=1,M
1041:       BB(I)=0.D00
1042:       DO J=1,M
1043:       BB(I)=BB(I)+A(I,J)*DSIN(X(J))+B(I,J)*DCOS(X(J))
1044:       ENDDO
1045:       ENDDO
1046:
1047:       F=0.D00
1048:       DO I=1,M
1049:       F=F+(AA(I)-BB(I))**2
1050:       ENDDO
1051:       RETURN
1052:       END
1053: C     ------------------------------------------------------------------
1054:       SUBROUTINE NFUNCT1(M,X,F)
1055:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
1056:       COMMON /KFF/KF,NFCALL
1057:       COMMON /RNDM/IU,IV
1058:       INTEGER IU,IV
1059:       DIMENSION X(*)
1060: C     MIN F (1, 1, ..., 1) =2
1061:       DO I=1,M
1062:       IF(X(I).LT.0.D00 .OR. X(I).GT.1.D00) THEN
1063:       CALL RANDOM(RAND)
1064:       X(I)=RAND
1065:       ENDIF
1066:       ENDDO
1067:       S=0.D00
1068:       DO I=1,M-1
1069:       S=S+X(I)
1070:       ENDDO
1071:       X(M)=(M-S)
1072:       F=(1.D00+X(M))**X(M)
```

```
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE NFUNCT2(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /KFF/KF,NFCALL
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*)
C     MIN F (1, 1, ..., 1) =2
      DO I=1,M
      IF(X(I).LT.0.D00 .OR. X(I).GT.1.D00) THEN
      CALL RANDOM(RAND)
      X(I)=RAND
      ENDIF
      ENDDO
      S=0.D00
      DO I=1,M-1
      S=S+(X(I)+X(I+1))/2.D00
      ENDDO
      X(M)=(M-S)
      F=(1.D00+X(M))**X(M)
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE TYPICAL(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /KFF/KF,NFCALL
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*)
C     A TYPICAL NONLINEAR MULTI-MODAL FUNCTION   FMIN=0
      F=0.D00
      DO I=1,M
      IF(DABS(X(I)).GT.10.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-0.5D00)*20
      ENDIF
      ENDDO
      DO I=2,M
      F=F+DCOS(  DABS(X(I)-X(I-1)) / DABS(X(I-1)+X(I)) )
      ENDDO
      F=F+(M-1.D00)
C     IF 0.001*X(1) IS ADDED TO F, IT BECOMES UNIMODAL
C      F=F+0.001*X(1)
      RETURN
      END
```