# Global Optimization by Differential Evolution and Particle Swarm Methods
# Evaluation on Some Benchmark Functions

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

**I. A Brief History of Optimization Research**: The history of optimization of real-valued non-linear functions (including linear ones), unconstrained or constrained, goes back to Gottfried Leibniz, Isaac Newton, Leonhard Euler and Joseph Lagrange. However, those mathematicians often assumed differentiability of the optimand as well as constraint functions. Moreover, they often dealt with the equality constraints. Richard Valentine (1937) and William Karush (1939), however, were perhaps the first mathematicians to study optimization of nonlinear functions under inequality constraints. Leonid Kantorovich and George Dantzig are well known for developing and popularizing linear programming, which ushered a new era of 'operations research', a branch of mathematical science that specializes in optimization. The development of linear programming soon prompted the study of the optimization problem of nonlinear functions (often under linear or nonlinear constraints). The joint work of Harold Kuhn and Albert Tucker (1951) – that was backed up by the work of Karush – is a landmark in the history of optimization of nonlinear functions.

Initially, optimization of nonlinear functions was methodologically based on the Leibniz-Newton principles and therefore could not easily escape local optima. Hence, its development to deal with nonconvex (multimodal) functions stagnated until the mid 1950's. Stanislaw Ulam, John von Neumann and Nicolas Metropolis had in the late 1940's proposed the Monte Carlo method of simulation and it was gradually realized that the simulation approach could provide an alternative methodology to mathematical investigations in optimization. George Box (1957) was perhaps the first mathematician who exploited the idea and developed his evolutionary method of nonlinear optimization. Almost a decade later, MJ Box (1965) developed his complex method, which strews random numbers over the entire domain of the decision variables and therefore has a great potentiality to escape local optima and locate the global optimum of a nonlinear function. The simplex method of John Nelder and Roger Mead (1964) also incorporated the ability to learn from its earlier search experience and adapt itself to the topography of the surface of the optimand function.

**II. Global Optimization**: 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 genetic algorithm is a class of population-based adaptive stochastic optimization procedures, characterizing the presence of randomness in the optimization process. The randomness may be present as either noise in measurements or Monte Carlo randomness in the search procedure, or both. The basic idea behind the genetic algorithm is to mimic a simple picture of the Darwinian natural selection in order to find a good algorithm and involves the operations such as 'mutation', 'selection' and 'evaluation of

fitness' repeatedly. The genetic algorithms may claim to have ushered the new era of global optimization.

A little later, in 1978, Aimo Törn introduced his "Clustering Algorithm" of global optimization. The method improves upon the earlier local search algorithms that needed 'multiple start' from several points distributed over the whole optimization region. Multi-start is certainly one of the earliest global procedures used. It has even been used in local optimization for increasing the confidence in the obtained solution. However, one drawback of Multi-start is that when many starting points are used, the same minimum will eventually be determined several times. In order to improve the efficiency of Multi-start this should be avoided. The clustering method of Törn avoids this repeated determination of local minima. This is realized in three steps, which may be iteratively used. The three steps are: (i) sample points in the region of interest, (ii) transform the sample to obtain points grouped around the local minima, and (iii) use a clustering technique to recognize these groups (i.e. neigbourhoods of the local minima). If the procedure employing these steps is successful, then, starting a single local optimization from each cluster would determine the local minima and, thus, also the global minimum. The advantage in using this approach is that the work spared by computing each minimum just once can be spent on computations in (i) and (ii), which will increase the probability that the global minimum will be found.

While the clustering algorithm was optimizing on the multi-start requirements of local search algorithms and the genetic algorithm simulated the Darwinian struggle for the survival of the fittest, the simulated annealing method (Kirkpatrick et al., 1983; Cerny, 1985) proposed to mimic the annealing process in metallurgy. In an annealing process a metal in the molten state (at a very high temperature) is slowly cooled so that the system at any time is approximately in thermodynamic equilibrium. As cooling proceeds, the system becomes more ordered – the liquid freezes or the metal re-crystallizes – attaining the ground state at T=0. This process is simulated through the Monte Carlo experiment (Metropolis et al. 1953). If the initial temperature of the melt is too low or cooling is done unduly fast the metal may become 'quenched' due to being trapped in a local minimum energy state (meta-stable state) forming defects or freezing out. The simulated annealing method of optimization makes very few assumptions regarding the function to be optimized, and therefore, it is quite robust with respect to irregular surfaces. In this method, the mathematical system describing the problem mimics the thermodynamic system. The current solution to the problem mimics the current state of the thermodynamic system, the objective function mimics the energy equation for the thermodynamic system, and the global minimum mimics the ground state. However, nothing in the numerical optimization problem directly mimics the temperature, T, in the thermodynamic system underlying the metallurgical process of annealing. Therefore, a complex abstraction mimics it. An arbitrary choice of initial value of a variable called 'temperature', how many iterations are performed at each 'temperature', the step length at which the decision variables are adjusted, and the rate of fall of 'temperature' at each step as 'cooling' proceeds, together make an 'annealing schedule'. This schedule mimics the cooling process. At a high 'temperature' the step lengths at which the decision variables are adjusted are larger than those at a lower

‘temperature’. Whether the system is trapped into local minima (quenching takes place) or it attains the global minimum (faultless crystallization) is dependent on the said annealing schedule. A wrong choice of the initial ‘temperature’, or the rate of fall in the ‘temperature’ leads to quenching or entrapment of the solution in the local minima. The method does not provide any clear guideline as to the choice of the ‘annealing schedule’ and often requires judgment or trial and error. If the schedule is properly chosen, the process attains the global minimum. It is said that using this method is an art and requires a lot of experience and judgment.

A little later, Fred Glover (1986) introduced his ‘Tabu Search’ method. This method economizes on repeated visits to the already visited points and in some sense is close to the clustering algorithms. Glover attributes it's origin to about 1977. The basic concept of Tabu Search as described by Glover is "a meta-heuristic superimposed on another heuristic. The overall approach is to avoid entrainment in cycles by forbidding or penalizing moves which take the solution, in the next iteration, to points in the solution space previously visited ( hence "tabu"). The Tabu method was partly motivated by the observation that human behavior appears to operate with a random element that leads to inconsistent behavior given similar circumstances. As Glover points out, the resulting tendency to deviate from a charted course, might be regretted as a source of error but can also prove to be source of gain. The Tabu method operates in this way with the exception that new courses are not chosen randomly. Instead the Tabu search proceeds according to the supposition that there is no point in accepting a new (poor) solution unless it is to avoid a path already investigated. This insures new regions of a problems solution space will be investigated in with the goal of avoiding local minima and ultimately finding the desired solution.

The pace of research in global optimization (GO) by stochastic process accelerated considerably in the 1990’s. Marco Dorigo in his Ph.D. thesis (1992) introduced his “Ant Colony” method of global optimization. It studies artificial systems that take inspiration from the behaviour of real ant colonies. Ants use pheromones that guide other fellow ants to identify the path that leads to a success. The chemical properties of pheromones and the ability of ants to gather information and use them are simulated in the Ant Colony method to reach at the global optimum. This method is well suited to combinatorial (discrete) optimization problems.

A couple of years later, James Kennedy and Russell Eberhart (1995) introduced their “Particle Swarm” method of global optimization. In the animal world we observe that a swarm of birds or insects or a school of fish searches for food, protection, etc. in a very typical manner. If one of the members of the swarm sees a desirable path to go, the rest of the swarm will follow quickly. The Particle Swarm method mimics this behaviour. Every individual of the swarm is considered as a particle in a multidimensional space that has a position and a velocity. These particles fly through hyperspace and remember the best position that they have seen. Members of a swarm communicate good positions to each other and adjust their own position and velocity based on these good positions. There are two main ways this communication is done: (i) “swarm best” that is known to all (ii) “local bests” are known in neighborhoods of particles. Updating of the position

and velocity are done at each iteration such that the solution often converges to the global optimum of the function. Interestingly, this method has a very sound and well-documented philosophical literature behind it (the British empiricist philosophy, the American pragmatism and others like those of Friedrich Hayek, Herbert Simon, etc.).

The method of Differential Evolution (DE) grew out of Kenneth Price's attempts to solve the Chebychev Polynomial fitting Problem that had been posed to him by Rainer Storn. A breakthrough happened (1996), when Price came up with the idea of using vector differences for perturbing the vector population. 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.

**III. The Characteristic Features of Population-Based GO Methods**: All population-based 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.

**IV. 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, 70 test functions have been chosen (for their algebraic specifications, see Mishra, 2006-c, d and e). Among these test functions, some are new while others are well known in the literature; some are unimodal, the others multi-modal; some are small in dimension (no. of variables, x in f(x)), while the others are large in

dimension; some are algebraic polynomial equations, while the other are transcendental, etc. In all, they together represent a wide variety of surfaces.

**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, $\in [0,1]$
- $\omega$ 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
- $\alpha$, $\beta$, $\gamma$ 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. However, in some cases moderately wide search (nstep=9, say; see program) works better.

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. The crossover scheme (called exponential crossover, as suggested by Kenneth Price in his personal letter to the author) is given below. This is coded for ncross =>1 in the program.

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, NCROSS<=0 that works better in case of a few functions.

**VII. Specification of Adjustable Parameters**: As it has been mentioned before, these methods need certain parameters to be defined before they work. In case of the DE, we have fixed: max number of iterations allowed, Iter = 10000, population size, N = 10 times of the dimension of the function or 100 whichever maximum; scheme of crossover, ncross =1 (defined in the program); crossover probability, 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. 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. In case of the RPS: 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-04. Meanings of these parameters are explained in the programs (appended).

**VIII. Results and Discussion**: Among 70 functions, a few have been run for small as well as large dimensions. In total, 73 optimization exercises have been done. DE has succeeded in 65 cases while RPS has succeeded in 55 cases. In almost all cases, DE has converged faster and given much more accurate results. The convergence of RPS is much slower even for lesser stringency on accuracy. The summary of results is presented in table-1. Some test functions have been hard for both the methods. These are: Zero-Sum (30D), Perm#1, Perm#2, Power and Bukin-6 functions.

From what we see in table 1, one cannot yet reach at the definite conclusion that the DE performs better (or worse) than the RPS. None could assure a supremacy over the other. Each one faltered in some case; each one succeeded in some others. However, *DE is unquestionably faster, more accurate and more frequently successful than the RPS*. It may be argued, nevertheless, that alternative choice of adjustable parameters could have yielded better results in either method's case. The protagonists of either method could

suggest that. Our purpose is not to join with the one or the other. We simply want to highlight that in certain cases they both succeed, in certain other case they both fail and each one has some selective preference over some particular type of surfaces. What is needed is to identify such structures and surfaces that suit a particular method most.

It is needed that we find out some criteria to classify the problems that suit (or does not suit) a particular method. This classification will highlight the comparative advantages of using a particular method for dealing with a particular class of problems.

**Table 1: A Summary of Success or Failure of DE and RPS Methods in Optimization of Test Functions**

| Sl No | Test Function | Dim (M) | Success | | Sl No | Test Function | Dim (M) | Success | |
|---|---|---|---|---|---|---|---|---|---|
| | | | DE | RPS | | | | DE | RPS |
| 1 | New Fn #1 | 2 | yes | yes | 38 | Linear Programming | 3 | yes | yes |
| 2 | New Fn #2 | 2 | yes | yes | 39 | Hougen Fn | 5 | no | no |
| 3 | New Fn #3 | 2 | yes | yes | 40 | Giunta Fn | 2 | yes | yes |
| 4 | New Fn #4 | 2 | yes | yes | 41 | Egg-Holder Fn | 2 | yes | yes |
| 5 | New Fn #8 | 2 | yes | yes | 42 | Trid Fn | 30 | yes | yes |
| 6 | Quintic Fn | 4 | yes | no | 43 | Greiwank Fn | 30 | yes | yes |
| 7 | Quintic Fn | 30 | yes | no | 44 | Weierstrass Fn | 20 | yes | no |
| 8 | Needle-eye Fn | 10 | yes | no | 45 | Levy#3 Fn | 2 | yes | yes |
| 9 | Needle-eye Fn | 30 | yes | no | 46 | Levy#5 Fn | 2 | yes | yes |
| 10 | Zero-sum Fn | 10 | yes | no | 47 | Levy#8 Fn | 3 | yes | yes |
| 11 | Zero-sum Fn | 30 | no | no | 48 | Rastrigin Fn | 30 | yes | no |
| 12 | Corana Fn | 4 | yes | yes | 49 | Ackley Fn | 30 | yes | yes |
| 13 | Mod RCos Fn | 2 | yes | yes | 50 | Michalewicz Fn | 20 | yes | no |
| 14 | Freud-Roth Fn | 2 | yes | yes | 51 | Schwefel Fn | 30 | yes | no |
| 15 | Anns XOR Fn | 9 | yes | no | 52 | Shubert Fn | 2 | yes | yes |
| 16 | Perm #1 Fn | 4 | no | no | 53 | Dixon-Price Fn | 10 | yes | no |
| 17 | Perm #2 Fn | 5 | no | no | 54 | Shekel Fn | 4 | yes | yes |
| 18 | Power-Sum Fn | 4 | no | no | 55 | Paviani Fn | 10 | yes | yes |
| 19 | Goldstein-Price Fn | 2 | yes | yes | 56 | Branin#1 Fn | 2 | yes | yes |
| 20 | Bukin-6 Fn | 2 | no | no | 57 | Branin#2 Fn | 2 | yes | yes |
| 21 | DCS Fn | 4 | yes | yes | 58 | Bohachevsky#1 Fn | 2 | yes | yes |
| 22 | New Factorial Fn | 4 | yes | yes | 59 | Bohachevsky#2 Fn | 2 | yes | yes |
| 23 | New Decanomial Fn | 2 | yes | yes | 60 | Bohachevsky#3 Fn | 2 | yes | yes |
| 24 | Judge Fn | 2 | yes | yes | 61 | Easom Fn | 2 | yes | yes |
| 25 | New Dodecal Fn | 3 | yes | yes | 62 | Rosenbrock Fn | 10 | yes | yes |
| 26 | New sum=prod Fn | 2 | yes | yes | 63 | Crosslegged Table Fn | 2 | yes | no |
| 27 | New AM=GM Fn | 10 | yes | yes | 64 | Cross Fn | 2 | yes | yes |
| 28 | Yao-Liu#2 Fn | 30 | yes | yes | 65 | Cross-in-Tray Fn | 2 | yes | yes |
| 29 | Yao-Liu#3 Fn | 30 | yes | yes | 66 | Crowned Cross Fn | 2 | yes | yes |
| 30 | Yao-Liu#4 Fn | 30 | yes | yes | 67 | TT-Holder Fn | 2 | yes | yes |
| 31 | Yao-Liu#6 Fn | 30 | yes | yes | 68 | Holder Table Fn | 2 | yes | yes |
| 32 | Yao-Liu#7 Fn | 30 | no | yes | 69 | Carrom Table Fn | 2 | yes | yes |
| 33 | Yao-Liu#12 Fn | 30 | yes | yes | 70 | Pen-Holder Fn | 2 | yes | yes |
| 34 | Yao-Liu#13 Fn | 30 | yes | yes | 71 | Bird Fn | 2 | yes | yes |
| 35 | Yao-Liu#14 Fn | 2 | yes | yes | 72 | Chichinadze Fn | 2 | yes | yes |
| 36 | Yao-Liu#15 Fn | 4 | no | yes | 73 | McCormick Fn | 2 | yes | yes |
| 37 | Linear Programming | 2 | yes | yes | **Failed in no. of cases (in 73 trials)** | | | **8** | **18** |

Note: For differently set adjustable parameters, either method may perform better or worse than reported here.

## 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.

```
C     MAIN PROGRAM : PROVIDES TO USE REPULSIVE PARTICLE SWARM METHOD
C     (SUBROUTINE RPS) AND DIFFERENTIAL WVOLUTION METHOD (DE)
c     ---------------------------------------------------------------
c     Adjust the parameters suitably in subroutines DE and RPS
c     When the program asks for parameters, feed them suitably
c     ---------------------------------------------------------------
      PROGRAM DERPS
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      COMMON /KFF/KF,NFCALL ! FUNCTION CODE AND NO. OF FUNCTION CALLS
      CHARACTER *30  METHOD(2)
      CHARACTER *1 PROCEED
      DIMENSION XX(2,50),KKF(2),MM(2),FMINN(2)
      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(*,*)'Adjust the parameters suitably in subroutines DE & RPS'
      WRITE(*,*)'====================     WARNING    ================ '
      METHOD(1)=' : DIFFERENTIAL EVALUATION'
      METHOD(2)=' : REPULSIVE PARTICLE SWARM'
      DO I=1,2

      IF(I.EQ.1) THEN
      WRITE(*,*)'============ DIFFERENTIAL EVOLUTION PROGRAM ==========='
      WRITE(*,*)'TO PROCEED TYPE ANY CHARACTER AND STRIKE ENTER'
      READ(*,*) PROCEED
      CALL DE(M,X,FMINDE) ! CALLS DE AND RETURNS OPTIMAL X AND FMIN
      FMIN=FMINDE
      ELSE
      WRITE(*,*)' '
      WRITE(*,*)' '
      WRITE(*,*)'==========REPULSIVE PARTICLE SWARM PROGRAM ==========='
      WRITE(*,*)'TO PROCEED TYPE ANY CHARACTER AND STRIKE ENTER'
      READ(*,*) PROCEED
      CALL RPS(M,X,FMINRPS) ! CALLS RPS AND RETURNS OPTIMAL X AND FMIN
      FMIN=FMINRPS
      ENDIF
      DO J=1,M
      XX(I,J)=X(J)
      ENDDO
      KKF(I)=KF
      MM(I)=M
      FMINN(I)=FMIN
      ENDDO
      WRITE(*,*)' '
      WRITE(*,*)' '
      WRITE(*,*)'---------------------- FINAL RESULTS=================='
      DO I=1,2
      WRITE(*,*)'FUNCT CODE=',KKF(I),'  FMIN=',FMINN(I),' : DIM=',MM(I)
      WRITE(*,*)'OPTIMAL DECISION VARIABLES : ',METHOD(I)
      WRITE(*,*)(XX(I,J),J=1,M)
      WRITE(*,*)'////////////////////////////////////////////////////'
      ENDDO
      WRITE(*,*)'PROGRAM ENDED'
      END
C     ---------------------------------------------------------------
      SUBROUTINE DE(M,A,FBEST)
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 EVOLDIF
      IMPLICIT DOUBLE PRECISION (A-H, O-Z) ! TYPE DECLARATION
      PARAMETER(NMAX=500,MMAX=50) ! 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 : 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
      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
```

```
135:                ENDDO
136: C     BEST FITNESS VALUE = FBEST : INDIVIDUAL X(KB)
137: C     ---------------------------------------------------------------
138: C     GENERATE OFFSPRINGS
139:       DO I=1,N     ! I LOOP BEGINS
140: C     INITIALIZE CHILDREN IDENTICAL TO PARENTS; THEY WILL CHANGE LATER
141:           DO J=1,M
142:           Y(I,J)=X(I,J)
143:           ENDDO
144: C     SELECT RANDOMLY THREE OTHER INDIVIDUALS
145:    20     DO IRI=1,3  ! IRI LOOP BEGINS
146:           IR(IRI)=0
147:
148:           CALL RANDOM(RAND)
149:            IRJ=INT(RAND*N)+1
150: C     CHECK THAT THESE THREE INDIVIDUALS ARE DISTICT AND OTHER THAN I
151:           IF(IRI.EQ.1.AND.IRJ.NE.I) THEN
152:           IR(IRI)=IRJ
153:           ENDIF
154:           IF(IRI.EQ.2.AND.IRJ.NE.I.AND.IRJ.NE.IR(1)) THEN
155:           IR(IRI)=IRJ
156:           ENDIF
157:        IF(IRI.EQ.3.AND.IRJ.NE.I.AND.IRJ.NE.IR(1).AND.IRJ.NE.IR(2)) THEN
158:            IR(IRI)=IRJ
159:            ENDIF
160:            ENDDO    ! IRI LOOP ENDS
161: C     CHECK IF ALL THE THREE IR ARE POSITIVE (INTEGERS)
162:           DO IX=1,3
163:           IF(IR(IX).LE.0) THEN
164:           GOTO 20  ! IF NOT THEN REGENERATE
165:           ENDIF
166:           ENDDO
167: C     THREE RANDOMLY CHOSEN INDIVIDUALS DIFFERENT FROM I AND DIFFERENT
168: C     FROM EACH OTHER ARE IR(1),IR(2) AND IR(3)
169: C     ---------------------------------------------------------------
170: C      NO CROSS OVER, ONLY REPLACEMENT THAT IS PROBABILISTIC
171:           IF(NCROSS.LE.0) THEN
172:           DO J=1,M       ! J LOOP BEGINS
173:           CALL RANDOM(RAND)
174:           IF(RAND.LE.PCROS) THEN ! REPLACE IF RAND < PCROS
175:           A(J)=X(IR(1),J)+(X(IR(2),J)-X(IR(3),J))*FACT ! CANDIDATE CHILD
176:           ENDIF
177:           ENDDO  ! J LOOP ENDS
178:           ENDIF
179:
180: C     ---------------------------------------------------------------
181: C     CROSSOVER SCHEME (EXPONENTIAL) SUGGESTED BY KENNETH PRICE IN HIS
182: C     PERSONAL LETTER TO THE AUTHOR (DATED SEPTEMBER 29, 2006)
183:       IF(NCROSS.GE.1) THEN
184:         CALL RANDOM(RAND)
185:     1   JR=INT(RAND*M)+1
186:         J=JR
187:     2   A(J)=X(IR(1),J)+FACT*(X(IR(2),J)-X(IR(3),J))
188:     3   J=J+1
189:         IF(J.GT.M) J=1
190:     4   IF(J.EQ.JR) GOTO 10
191:     5   CALL RANDOM(RAND)
192:         IF(PCROS.LE.RAND) GOTO 2
193:     6   A(J)=X(I,J)
194:     7   J=J+1
195:         IF(J.GT.M) J=1
196:     8   IF (J.EQ.JR) GOTO 10
197:     9   GOTO 6
198:    10   CONTINUE
199:       ENDIF
200: C     ---------------------------------------------------------------
201:           CALL FUNC(A,M,F) ! EVALUATE THE OFFSPRING
```

```
202:           IF(F.LT.FV(I)) THEN ! IF BETTER, REPLACE PARENTS BY THE CHILD
203:           FV(I)=F
204:           DO J=1,M
205:           Y(I,J)=A(J)
206:           ENDDO
207:           ENDIF
208:       ENDDO    ! I LOOP ENDS
209:       DO I=1,N
210:       DO J=1,M
211:       X(I,J)=Y(I,J) ! NEW GENERATION IS A MIX OF BETTER PARENTS AND
212: C                    BETTER CHILDREN
213:       ENDDO
214:       ENDDO
215:       IPCOUNT=IPCOUNT+1
216:       IF(IPCOUNT.EQ.IPRINT) THEN
217:       DO J=1,M
218:       A(J)=X(KB,J)
219:       ENDDO
220:       WRITE(*,*)(X(KB,J),J=1,M),'  FBEST UPTO NOW = ',FBEST
221:       WRITE(*,*)'TOTAL NUMBER OF FUNCTION CALLS =',NFCALL
222:            IF(DABS(FBEST-GBEST).LT.EPS) THEN
223:            WRITE(*,*) FTIT
224:            WRITE(*,*)'COMPUTATION OVER'
225:            RETURN
226:            ELSE
227:            GBEST=FBEST
228:            ENDIF
229:       IPCOUNT=0
230:       ENDIF
231: C     ---------------------------------------------------------------
232:   100 ENDDO    ! ITERATION ENDS : GO FOR NEXT ITERATION, IF APPLICABLE
233: C     ---------------------------------------------------------------
234:       WRITE(*,*)'DID NOT CONVERGE. REDUCE EPS OR RAISE ITER OR DO BOTH'
235:       WRITE(*,*)'INCREASE N, PCROS, OR SCALE FACTOR (FACT)'
236:       RETURN
237:       END
238: C     ---------------------------------------------------------------------
239: C     RANDOM NUMBER GENERATOR (UNIFORM BETWEEN 0 AND 1 - BOTH EXCLUSIVE)
240:       SUBROUTINE RANDOM(RAND1)
241:        DOUBLE PRECISION  RAND1
242:        COMMON /RNDM/IU,IV
243:       INTEGER IU,IV
244:        RAND=REAL(RAND1)
245:        IV=IU*65539
246:        IF(IV.LT.0) THEN
247:        IV=IV+2147483647+1
248:        ENDIF
249:        RAND=IV
250:        IU=IV
251:        RAND=RAND*0.4656613E-09
252:        RAND1= (RAND)
253:        RETURN
254:        END
255: C     ---------------------------------------------------------------
256:       SUBROUTINE FSELECT(KF,M,FTIT)
257: C     THE PROGRAM REQUIRES INPUTS FROM THE USER ON THE FOLLOWING ------
258: C     (1) FUNCTION CODE (KF), (2) NO. OF VARIABLES IN THE FUNCTION (M);
259:       CHARACTER *70 TIT(100),FTIT
260:       WRITE(*,*)'-----------------------------------------------------'
261:       DATA TIT(1)/'KF=1 NEW FUNCTION(N#1) 2-VARIABLES M=2'/
262:       DATA TIT(2)/'KF=2 NEW FUNCTION(N#2) 2-VARIABLES M=2'/
263:       DATA TIT(3)/'KF=3 NEW FUNCTION(N#3) 2-VARIABLES M=2'/
264:       DATA TIT(4)/'KF=4 NEW FUNCTION(N#4) 2-VARIABLES M=2'/
265:       DATA TIT(5)/'KF=5 NEW QUINTIC FUNCTION M-VARIABLES M=?'/
266:       DATA TIT(6)/'KF=6 NEW NEEDLE-EYE FUNCTION (N#6) M-VARIABLES M=?'/
267:       DATA TIT(7)/'KF=7 NEW ZERO-SUM FUNCTION (N#7) M-VARIABLES M=?'/
268:       DATA TIT(8)/'KF=8 CORANA FUNCTION 4-VARIABLES M=4'/
```

```
      DATA TIT(9)/'KF=9 MODIFIED RCOS FUNCTION 2-VARIABLES M=2'/
      DATA TIT(10)/'KF=10 FREUDENSTEIN ROTH FUNCTION 2-VARIABLES M=2'/
      DATA TIT(11)/'KF=11 ANNS XOR FUNCTION 9-VARIABLES M=9'/
      DATA TIT(12)/'KF=12 PERM FUNCTION #1 (SET BETA) 4-VARIABLES M=4'/
      DATA TIT(13)/'KF=13 PERM FUNCTION #2 (SET BETA) M-VARIABLES M=?'/
      DATA TIT(14)/'KF=14 POWER-SUM FUNCTION 4-VARIABLES M=4'/
      DATA TIT(15)/'KF=15 GOLDSTEIN PRICE FUNCTION 2-VARIABLES M=2'/
      DATA TIT(16)/'KF=16 BUKIN 6TH FUNCTION 2-VARIABLES M=2'/
      DATA TIT(17)/'KF=17 NEW FUNCTION (N#8) 2-VARIABLES M=2'/
      DATA TIT(18)/'KF=18 DEFL CORRUG SPRING FUNCTION M-VARIABLES M=?'/
      DATA TIT(19)/'KF=19 NEW FACTORIAL FUNCTION M-VARIABLES M=?'/
      DATA TIT(20)/'KF=20 NEW DECANOMIAL FUNCTION 2-VARIABLES M=2'/
      DATA TIT(21)/'KF=21 JUDGE FUNCTION 2-VARIABLES M=2'/
      DATA TIT(22)/'KF=22 NEW DODECAL FUNCTION 3-VARIABLES M=3'/
      DATA TIT(23)/'KF=23 NEW SUM-EQ-PROD FUNCTION 2-VARIABLES M=2'/
      DATA TIT(24)/'KF=24 NEW AM-EQ-GM FUNCTION M-VARIABLES M=?'/
      DATA TIT(25)/'KF=25 YAO-LIU FUNCTION#2 M-VARIABLES M=?'/
      DATA TIT(26)/'KF=26 YAO-LIU FUNCTION#3 M-VARIABLES M=?'/
      DATA TIT(27)/'KF=27 YAO-LIU FUNCTION#4 M-VARIABLES M=?'/
      DATA TIT(28)/'KF=28 YAO-LIU FUNCTION#6 M-VARIABLES M=?'/
      DATA TIT(29)/'KF=29 YAO-LIU FUNCTION#7 M-VARIABLES M=?'/
      DATA TIT(30)/'KF=30 YAO-LIU FUNCTION#12 M-VARIABLES M=?'/
      DATA TIT(31)/'KF=31 YAO-LIU FUNCTION#13 M-VARIABLES M=?'/
      DATA TIT(32)/'KF=32 YAO-LIU FUNCTION#14 2-VARIABLES M=2'/
      DATA TIT(33)/'KF=33 YAO-LIU FUNCTION#15 4-VARIABLES M=4'/
      DATA TIT(34)/'KF=34 LINEAR PROGRAMMING-I : 2-VARIABLES M=2'/
      DATA TIT(35)/'KF=35 LINEAR PROGRAMMING-II : 3-VARIABLES M=3'/
      DATA TIT(36)/'KF=36 HOUGEN FUNCTION : 5-VARIABLES M=5'/
      DATA TIT(37)/'KF=37 GIUNTA FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(38)/'KF=38 EGGHOLDER FUNCTION : M-VARIABLES M=?'/
      DATA TIT(39)/'KF=39 TRID FUNCTION : M-VARIABLES M=?'/
      DATA TIT(40)/'KF=40 GRIEWANK FUNCTION : M-VARIABLES M=?'/
      DATA TIT(41)/'KF=41 WEIERSTRASS FUNCTION : M-VARIABLES M=?'/
      DATA TIT(42)/'KF=42 LEVY-3 FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(43)/'KF=43 LEVY-5 FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(44)/'KF=44 LEVY-8 FUNCTION : 3-VARIABLES M=3'/
      DATA TIT(45)/'KF=45 RASTRIGIN FUNCTION : M-VARIABLES M=?'/
      DATA TIT(46)/'KF=46 ACKLEY FUNCTION : M-VARIABLES M=?'/
      DATA TIT(47)/'KF=47 MICHALEWICZ FUNCTION : M-VARIABLES M=?'/
      DATA TIT(48)/'KF=48 SCHWEFEL FUNCTION : M-VARIABLES M=?'/
      DATA TIT(49)/'KF=49 SHUBERT FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(50)/'KF=50 DIXON-PRICE FUNCTION : M-VARIABLES M=?'/
      DATA TIT(51)/'KF=51 SHEKEL FUNCTION : 4-VARIABLES M=4'/
      DATA TIT(52)/'KF=52 PAVIANI FUNCTION : 10-VARIABLES M=10'/
      DATA TIT(53)/'KF=53 BRANIN FUNCTION#1 : 2-VARIABLES M=2'/
      DATA TIT(54)/'KF=54 BRANIN FUNCTION#2 : 2-VARIABLES M=2'/
      DATA TIT(55)/'KF=55 BOHACHEVSKY FUNCTION#1 : 2-VARIABLES M=2'/
      DATA TIT(56)/'KF=56 BOHACHEVSKY FUNCTION#2 : 2-VARIABLES M=2'/
      DATA TIT(57)/'KF=57 BOHACHEVSKY FUNCTION#3 : 2-VARIABLES M=2'/
      DATA TIT(58)/'KF=58 EASOM FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(59)/'KF=59 ROSENBROCK FUNCTION : M-VARIABLES M=?'/
      DATA TIT(60)/'KF=60 CROSS-LEGGED TABLE FUNCTION:2-VARIABLES M=2'/
      DATA TIT(61)/'KF=61 CROSS FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(62)/'KF=62 CROSS-IN-TRAY FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(63)/'KF=63 CROWNED CROSS FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(64)/'KF=64 TT-HOLDER FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(65)/'KF=65 HOLDER-TABLE FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(66)/'KF=66 CARROM-TABLE FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(67)/'KF=67 PENHOLDER FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(68)/'KF=68 BIRD FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(69)/'KF=69 CHICHINADZE FUNCTION : 2-VARIABLES M=2'/
      DATA TIT(70)/'KF=70 MCCORMICK FUNCTION : 2-VARIABLES M=2'/
C     ------------------------------------------------------------------
      DO I=1,70
      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     ------------------------------------------------------------------
      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(*)

      PI=4.D+00*DATAN(1.D+00)! DEFINING THE VALUE OF PI
      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     FUNCTION #1 MIN AT -0.18467 APPROX AT (-8.4666, -10) APPROX
      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
      F=DABS(DCOS(DSQRT(DABS(X(1)**2+X(2)))))**0.5 +0.01*X(1)+.01*X(2)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.2) THEN
C     FUNCTION #2 MIN = -0.199409 APPROX AT (-9.94112, -10) APPROX
      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
      F=DABS(DSIN(DSQRT(DABS(X(1)**2+X(2)))))**0.5 +0.01*X(1)+.01*X(2)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.3) THEN
C     FUNCTION #3 MIN = -1.01983 APPROX AT (-1.98682, -10.00000) APPROX
      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
      F1=DSIN(( DCOS(X(1))+DCOS(X(2)) )**2)**2
      F2=DCOS(( DSIN(X(1))+DSIN(X(2)) )**2)**2
      F=(F1+F2+X(1))**2 ! IS MULTIMODAL
      F=F+ 0.01*X(1)+0.1*X(2) ! MAKES UNIMODAL
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.4) THEN
C     FUNCTION  #4 MIN = -2.28395 APPROX AT (2.88631, 1.82326) APPROX
      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
      F1=DSIN((DCOS(X(1))+DCOS(X(2)))**2)**2
      F2=DCOS((DSIN(X(1))+DSIN(X(2)))**2)**2
      F3=-DLOG((F1-F2+X(1))**2 )
      F=F3+0.1D00*(X(1)-1.D00)**2+0.1D00*(X(2)-1.D00)**2
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.5) THEN
C     QUINTIC FUNCTION:GLOBAL MINIMA,EXTREMELY DIFFICULT TO OPTIMIZE
C     MIN VALUE = 0 AT PERMUTATION OF (2, 2,..., 2, -1, -1, ..., -1,
C     -0.402627941) GIVES MIN F = 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
      CALL QUINTIC(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.6) THEN
C     NEEDLE-EYE FUNCTION M=>1;
C     MIN = 1 IF ALL ABS(X) ARE SMALLER THAN THE EYE
C     SMALLER THE VALUE OF ZZ, MORE DIFFICULT TO ENTER THE EYE
C     LARGER THE VALUE OF M, MORE DIFFICULT TO FIND THE OPTIMUM
      F=0.D00
      EYE=0.000001D00
      FP=0.D00
      DO I=1,M
      IF(DABS(X(I)).GT.EYE) THEN
      FP=1.D00
       F=F+100.D00+DABS(X(I))
      ELSE
      F=F+1.D00
      ENDIF
      ENDDO
      IF(FP.EQ.0.D00) F=F/M
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.7) THEN
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
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.8) THEN
C     CORANA FUNCTION : MIN = 0 AT (0, 0, 0, 0) APPROX
      F=0.D00
      DO I=1,M
      IF(DABS(X(I)).GT.1000.D00) THEN
      CALL RANDOM(RAND)
```

```
      X(I)=(RAND-0.5D00)*2000
      ENDIF
      ENDDO
      DO J=1,M
        IF(J.EQ.1) DJ=1.D00
        IF(J.EQ.2) DJ=1000.D00
        IF(J.EQ.3) DJ=10.D00
        IF(J.EQ.4) DJ=100.D00
            ISGNXJ=1
            IF(X(J).LT.0.D00) ISGNXJ=-1
            ZJ=(DABS(X(J)/0.2D00)+0.49999)*ISGNXJ*0.2D00
            ISGNZJ=1
            IF(ZJ.LT.0.D00) ISGNZJ=-1
            IF(DABS(X(J)-ZJ).LT.0.05D00) THEN
            F=F+0.15D00*(ZJ-0.05D00*ISGNZJ)**2 * DJ
            ELSE
            F=F+DJ*X(J)**2
            ENDIF
      ENDDO
      RETURN
      ENDIF
C     ----------------------------------------------------------------
      IF(KF.EQ.9) THEN
C     MODIFIED RCOS FUNCTION MIN=-0.179891 AT (-3.196989, 12.52626)APPRX
      F=0.D00
      IF(X(1).LT.-5.D00 .OR. X(1).GT.10.D00) THEN
      CALL RANDOM(RAND)
      X(1)=RAND*15.D00 -5.D00
      ENDIF
      IF(X(2).LT.0.D00 .OR. X(2).GT.15.D00) THEN
      CALL RANDOM(RAND)
      X(2)=RAND*15.D00
      ENDIF
      CA=1.D00
      CB=5.1/(4*PI**2)
      CC=5.D00/PI
      CD=6.D00
      CE=10.D00
      CF=1.0/(8*PI)
      F1=CA*(X(2)-CB*X(1)**2+CC*X(1)-CD)**2
      F2=CE*(1.D00-CF)*DCOS(X(1))*DCOS(X(2))
      F3=DLOG(X(1)**2+X(2)**2+1.D00)
      F=-1.0/(F1+F2+F3+CE)
      RETURN
      ENDIF
C     ----------------------------------------------------------------
      IF(KF.EQ.10) THEN
C     FREUDENSTEIN ROTH  FUNCTION : MIN  = 0 AT (5, 4)
      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
      F1=(-13.D00+X(1)+((5.D00-X(2))*X(2)-2)*X(2))**2
      F2=(-29.D00+X(1)+((X(2)+1.D00)*X(2)-14.D00)*X(2))**2
      F=F1+F2
      RETURN
      ENDIF
C     ----------------------------------------------------------------
      IF(KF.EQ.11) THEN
C     ANNS XOR FUNCTION (PARSOPOULOS, KE, PLAGIANAKOS, VP, MAGOULAS, GD
C     AND VRAHATIS, MN "STRETCHING TECHNIQUE FOR OBTAINING GLOBAL
C     MINIMIZERS THROUGH PARTICLE SWARM OPTIMIZATION")
C     MIN=0.9597588 FOR X=(1, -1, 1, -1, -1, 1, 1, -1, 0.421134) APPROX
C     OBTAINED BY DIFFERENTIAL EVOLUTION PROGRAM
```

```
      F=0.D00
      DO I=1,M
      IF(DABS(X(I)).GT.1.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-0.5D00)*2
      ENDIF
      ENDDO
      F11=X(7)/(1.D00+DEXP(-X(1)-X(2)-X(5)))
      F12=X(8)/(1.D00+DEXP(-X(3)-X(4)-X(6)))
      F1=(1.D00+DEXP(-F11-F12-X(9)))**(-2)
      F21=X(7)/(1.D00+DEXP(-X(5)))
      F22=X(8)/(1.D00+DEXP(-X(6)))
      F2=(1.D00+DEXP(-F21-F22-X(9)))**(-2)
      F31=X(7)/(1.D00+DEXP(-X(1)-X(5)))
      F32=X(8)/(1.D00+DEXP(-X(3)-X(6)))
      F3=(1.D00-(1.D00+DEXP(-F31-F32-X(9)))**(-1))**2
      F41=X(7)/(1.D00+DEXP(-X(2)-X(5)))
      F42=X(8)/(1.D00+DEXP(-X(4)-X(6)))
      F4=(1.D00-(1.D00+DEXP(-F41-F42-X(9)))**(-1))**2
      F=F1+F2+F3+F4
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.12) 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
      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
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.13) 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
      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
      ENDIF
```

```
C     ------------------------------------------------------------------
      IF(KF.EQ.14) THEN
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
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.15) THEN
C     GOLDSTEIN PRICE  FUNCTION  : MIN VALUE = 3 AT (0, -1)
      F=0.D00
      DO I=1,M
      IF(DABS(X(I)).GT.10.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-.5D00)*20
      ENDIF
      ENDDO
      F11=(X(1)+X(2)+1.D00)**2
      F12=(19.D00-14*X(1)+ 3*X(1)**2-14*X(2)+ 6*X(1)*X(2)+ 3*X(2)**2)
      F1=1.00+F11*F12
      F21=(2*X(1)-3*X(2))**2
      F22=(18.D00-32*X(1)+12*X(1)**2+48*X(2)-36*X(1)*X(2)+27*X(2)**2)
      F2=30.D00+F21*F22
      F= (F1*F2)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.16) 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
        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
       ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.17) THEN
C     NEW N#8 FUNCTION (MULTIPLE GLOBAL MINIMA)
C     MIN VALUE = -1 AT (AROUND .7 AROUND, 0.785 APPROX)
      F=0.D00
      DO I=1,M
      IF(X(I).LT.0.5D00 .OR. X(I).GT.1.D00) THEN
      CALL RANDOM(RAND)
```

```
671:        X(I)=RAND/2.D00
672:        ENDIF
673:        ENDDO
674:        F=-DEXP(-DABS(DLOG(.001D00+DABS((DSIN(X(1)+X(2))+DSIN(X(1)-X(2))+
675:       & (DCOS(X(1)+X(2))*DCOS(X(1)-X(2))+.001))**2)+
676:       & .01D00*(X(2)-X(1))**2)))
677:         RETURN
678:         ENDIF
679: C      ------------------------------------------------------------------
680:        IF(KF.EQ.18) THEN
681: C      DEFLECTED CORRUGATED SPRING FUNCTION
682: C      MIN VALUE = -1 AT (5, 5, ..., 5) FOR ANY K AND ALPHA=5; M VARIABLE
683:        CALL DCS(M,F,X)
684:        RETURN
685:        ENDIF
686: C      ------------------------------------------------------------------
687:        IF(KF.EQ.19) THEN
688: C      FACTORIAL FUNCTION, MIN =0 AT X=(1,2,3,...,M)
689:        CALL FACTOR1(M,F,X)
690:        RETURN
691:        ENDIF
692: C      ------------------------------------------------------------------
693:        IF(KF.EQ.20) THEN
694: C      DECANOMIAL FUNCTION, MIN =0 AT X=(2, -3)
695:          DO I=1,M
696:          IF(DABS(X(I)).GT.4.D00) THEN
697:          CALL RANDOM(RAND)
698:          X(I)= (RAND-0.5D00)*8
699:          ENDIF
700:          ENDDO
701:        CALL DECANOM(M,F,X)
702:        RETURN
703:        ENDIF
704: C      ------------------------------------------------------------------
705:        IF(KF.EQ.21) THEN
706: C      JUDGE'S FUNCTION  F(0.864, 1.23) = 16.0817; M=2
707:        CALL JUDGE(M,X,F)
708:        RETURN
709:        ENDIF
710: C      ------------------------------------------------------------------
711:        IF(KF.EQ.22) THEN
712: C      DODECAL FUNCTION
713:        CALL DODECAL(M,F,X)
714:        RETURN
715:        ENDIF
716: C      ------------------------------------------------------------------
717:        IF(KF.EQ.23) THEN
718: C      WHEN X(1)*X(2)=X(1)*X(2)  ? M=2
719:        CALL SEQP(M,F,X)
720:        RETURN
721:        ENDIF
722: C      ------------------------------------------------------------------
723:        IF(KF.EQ.24) THEN
724: C      WHEN ARITHMETIC MEAN = GEOMETRIC MEAN ? : M =>1
725:        CALL AMGM(M,F,X)
726:        RETURN
727:        ENDIF
728: C      ------------------------------------------------------------------
729:        IF(KF.EQ.25) THEN
730: C      M =>2
731:        CALL FUNCT2(M,F,X)
732:        RETURN
733:        ENDIF
734: C      ------------------------------------------------------------------
735:        IF(KF.EQ.26) THEN
736: C      M =>2
737:        CALL FUNCT3(M,F,X)
```

```
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.27) THEN
C     M =>2
      CALL FUNCT4(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.28) THEN
C     M =>2
      CALL FUNCT6(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.29) THEN
C     M =>2
      CALL FUNCT7(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.30) THEN
C     M =>2
      CALL FUNCT12(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.31) THEN
C     M =>2
      CALL FUNCT13(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.32) THEN
C     M =2
      CALL FUNCT14(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.33) THEN
C     M =4
      CALL FUNCT15(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.34) THEN
C     LINEAR PROGRAMMING : MINIMIZATION PROBLEM : M =2
      CALL LINPROG1(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.35) THEN
C     LINEAR PROGRAMMING : MINIMIZATION PROBLEM : M =3
      CALL LINPROG2(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.36) THEN
C     HOUGEN FUNCTION 5 VARIABLES : M =3
      CALL HOUGEN(X,M,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.37) THEN
C     GIUNTA FUNCTION 2 VARIABLES  :M =2
      CALL GIUNTA(M,X,F)
      RETURN
```

```
805:       ENDIF
806: C     ------------------------------------------------------------------
807:       IF(KF.EQ.38) THEN
808: C     EGGHOLDER FUNCTION M VARIABLES
809:       CALL EGGHOLD(M,X,F)
810:       RETURN
811:       ENDIF
812: C     ------------------------------------------------------------------
813:       IF(KF.EQ.39) THEN
814: C     TRID FUNCTION M VARIABLES
815:       CALL TRID(M,X,F)
816:       RETURN
817:       ENDIF
818: C     ------------------------------------------------------------------
819:       IF(KF.EQ.40) THEN
820: C     GRIEWANK FUNCTION M VARIABLES
821:       CALL GRIEWANK(M,X,F)
822:       RETURN
823:       ENDIF
824: C     ------------------------------------------------------------------
825:       IF(KF.EQ.41) THEN
826: C     WEIERSTRASS FUNCTION M VARIABLES
827:       CALL WEIERSTRASS(M,X,F)
828:       RETURN
829:       ENDIF
830: C     ------------------------------------------------------------------
831:       IF(KF.EQ.42) THEN
832: C     LEVY-3 FUNCTION 2 VARIABLES
833:       CALL LEVY3(M,X,F)
834:       RETURN
835:       ENDIF
836: C     ------------------------------------------------------------------
837:       IF(KF.EQ.43) THEN
838: C     LEVY-5 FUNCTION 2 VARIABLES
839:       CALL LEVY5(M,X,F)
840:       RETURN
841:       ENDIF
842: C     ------------------------------------------------------------------
843:       IF(KF.EQ.44) THEN
844: C     LEVY-8 FUNCTION 3 VARIABLES
845:       CALL LEVY8(M,X,F)
846:       RETURN
847:       ENDIF
848: C     ------------------------------------------------------------------
849:       IF(KF.EQ.45) THEN
850: C     RASTRIGIN FUNCTION M VARIABLES
851:       CALL RASTRIGIN(M,X,F)
852:       RETURN
853:       ENDIF
854: C     ------------------------------------------------------------------
855:       IF(KF.EQ.46) THEN
856: C     ACKLEY FUNCTION M VARIABLES
857:       CALL ACKLEY(M,X,F)
858:       RETURN
859:       ENDIF
860: C     ------------------------------------------------------------------
861:       IF(KF.EQ.47) THEN
862: C     MICHALEWICZ FUNCTION M VARIABLES
863:       CALL MICHALEWICZ(M,X,F)
864:       RETURN
865:       ENDIF
866: C     ------------------------------------------------------------------
867:       IF(KF.EQ.48) THEN
868: C     SCHWEFEL FUNCTION M VARIABLES
869:       CALL SCHWEFEL(M,X,F)
870:       RETURN
871:       ENDIF
```

```
C     ------------------------------------------------------------------
      IF(KF.EQ.49) THEN
C     SHUBERT FUNCTION 2 VARIABLES
      CALL SHUBERT(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.50) THEN
C     DIXON AND PRICE FUNCTION M VARIABLES
      CALL DIXPRICE(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.51) THEN
C     SHEKEL FUNCTION 4 VARIABLES
      CALL SHEKEL(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.52) THEN
C     PAVIANI FUNCTION 10 VARIABLES
      CALL PAVIANI(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.53) THEN
C     BRANIN FUNCTION#1 2 VARIABLES
      CALL BRANIN1(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.54) THEN
C     BRANIN FUNCTION#2 2 VARIABLES
      CALL BRANIN2(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.55) THEN
C     BOHACHEVSKY FUNCTION#1 2 VARIABLES
      CALL BOHACHEVSKY1(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.56) THEN
C     BOHACHEVSKY FUNCTION#2 2 VARIABLES
      CALL BOHACHEVSKY2(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.57) THEN
C     BOHACHEVSKY FUNCTION#3 2 VARIABLES
      CALL BOHACHEVSKY3(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.58) THEN
C     EASOM FUNCTION#3 2 VARIABLES
      CALL EASOM(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.59) THEN
C     ROSENBROCK FUNCTION  M VARIABLES
      CALL ROSENBROCK(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
```

```
      IF(KF.EQ.60) THEN
C     CROSS-LEGGED TABLE FUNCTION : 2 VARIABLES
      CALL CROSSLEG(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.61) THEN
C     CROSS FUNCTION : 2 VARIABLES
      CALL CROSS(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.62) THEN
C     CROSS-IN-TRAY FUNCTION : 2 VARIABLES
      CALL CROSSINTRAY(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.63) THEN
C     CROWNED-CROSS FUNCTION : 2 VARIABLES
      CALL CROWNEDCROSS(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.64) THEN
C     TT-HOLDER FUNCTION : 2 VARIABLES, MIN F([+/-]1.5706, 0)= -10.8723
      CALL TTHOLDER(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.65) THEN
C     HOLDER-TABLE FUNCTION : 2 VARIABLES
C     MIN F ([+/-] 9.64617, [+/-] 9.64617) APPROX = -26.92034 APPROX
      CALL HOLDERTABLE(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.66) THEN
C     CARROM-TABLE FUNCTION : 2 VARIABLES
C     MIN F ([+/-] 9.64617, [+/-] 9.64617) APPROX = -24.15682 APPROX
      CALL CARROMTABLE(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.67) THEN
C     PEN-HOLDER FUNCTION : 2 VARIABLES
C     MIN F ([+/-] 9.64617, [+/-] 9.64617) APPROX = -0.963535 APPROX
      CALL PENHOLDER(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.68) THEN
C     BIRD FUNCTION : 2 VARIABLES
C     MIN F (4.70104, 3.15294) APPROX = -106.764537 APPROX  OR
C     MIN F (-1.58214, -3.13024) APPROX = -106.764537 APPROX
      CALL BIRD(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.69) THEN
C     CHICHINADZE FUNCTION :  -30 <=X(I)<= 30;  M=2
C     MIN F (5.901329, 0.5) = -43.3158621
      CALL CHICHINADZE(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.70) THEN
```

```
1006: C     MCCORMICK FUNCTION : -1.5<= X(1)<=4; -3<=X(2)<=4 ; M=2
1007: C     MIN F (-0.54719755, -1.54719755) = -1.913223 APPROX
1008:       CALL MCCORMICK(M,X,F)
1009:       RETURN
1010:       ENDIF
1011: C     ================================================================
1012:       WRITE(*,*)'FUNCTION NOT DEFINED. PROGRAM ABORTED'
1013:       STOP
1014:       END
1015: C     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
1016:       SUBROUTINE DCS(M,F,X)
1017: C     FOR DEFLECTED CORRUGATED SPRING FUNCTION
1018:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1019:       DIMENSION X(*),C(100)
1020: C     OPTIMAL VALUES OF (ALL) X ARE ALPHA , AND K IS ONLY FOR SCALING
1021: C     OPTIMAL VALUE OF F IS -1. DIFFICULT TO OPTIMIZE FOR LARGER M.
1022:       DATA K,ALPHA/5,5.D00/ ! K AND ALPHA COULD TAKE ON ANY OTHER VALUES
1023:       R2=0.D00
1024:       DO I=1,M
1025:       C(I)=ALPHA
1026:       R2=R2+(X(I)-C(I))**2
1027:       ENDDO
1028:       R=DSQRT(R2)
1029:       F=-DCOS(K*R)+0.1D00*R2
1030:       RETURN
1031:       END
1032: C     ----------------------------------------------------------------
1033:       SUBROUTINE QUINTIC(M,F,X)
1034: C     QUINTIC FUNCTION: GLOBAL MINIMA,EXTREMELY DIFFICULT TO OPTIMIZE
1035: C     MIN VALUE = 0 AT PERM( -1, -0.402627941, 2)
1036:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1037:       DIMENSION X(*)
1038:       F=0.D00
1039:       DO I=1,M
1040:       F=F+DABS(X(I)**5-3*X(I)**4+4*X(I)**3+2*X(I)**2-10*X(I)-4.D00)
1041:       ENDDO
1042:       F=1000*F
1043:       RETURN
1044:       END
1045: C     ----------------------------------------------------------------
1046:       SUBROUTINE FACTOR1(M,F,X)
1047: C     FACTORIAL FUNCTION; MIN (1, 2, 3, ...., M) = 0
1048: C     FACT = FACTORIAL(M) = 1 X 2 X 3 X 4 X .... X M
1049: C     FIND X(I), I=1,2,...,M SUCH THAT THEIR PRODUCT IS EQUAL TO FACT.
1050: C     LARGER THE VALUE OF M (=>8) OR SO, HARDER IS THE PROBLEM
1051:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1052:       DIMENSION X(*)
1053:       F=0.D00
1054:       FACT=1.D00
1055:       P=1.D00
1056:       DO I=1,M
1057:       FACT=FACT*I
1058:       P=P*X(I)
1059:       F=F+DABS(P-FACT)**2
1060:       ENDDO
1061:       RETURN
1062:       END
1063: C     ----------------------------------------------------------------
1064:       SUBROUTINE DECANOM(M,F,X)
1065: C     DECANOMIAL FUNCTION; MIN (2, -3) = 0
1066:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1067:       DIMENSION X(*)
1068:       F1= DABS(X(1)**10-20*X(1)**9+180*X(1)**8-960*X(1)**7+
1069:      & 3360*X(1)**6-8064*X(1)**5+13340*X(1)**4-15360*X(1)**3+
1070:      & 11520*X(1)**2-5120*X(1)+2624.D00)
1071:       F2= DABS(X(2)**4+12*X(2)**3+54*X(2)**2+108*X(2)+81.D00)
1072:       F=0.001D00*(F1+F2)**2
```

```
1073:       RETURN
1074:       END
1075: C     ----------------------------------------------------------------
1076:       SUBROUTINE JUDGE(M,X,F)
1077:       PARAMETER (N=20)
1078: C     THIS SUBROUTINE IS FROM THE EXAMPLE IN JUDGE ET AL., THE THEORY
1079: C     AND PRACTICE OF ECONOMETRICS, 2ND ED., PP. 956-7. THERE ARE TWO
1080: C     OPTIMA: F(0.86479,1.2357)=16.0817307 (WHICH IS THE GLOBAL MINUMUM)
1081: C     AND F(2.35,-0.319)=20.9805 (WHICH IS LOCAL). ADAPTED FROM BILL
1082: C     GOFFE'S SIMMAN (SIMULATED ANNEALING) PROGRAM
1083:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1084:       DIMENSION Y(N), X2(N), X3(N), X(*)
1085:        DATA (Y(I),I=1,N)/4.284,4.149,3.877,0.533,2.211,2.389,2.145,
1086:      & 3.231,1.998,1.379,2.106,1.428,1.011,2.179,2.858,1.388,1.651,
1087:      & 1.593,1.046,2.152/
1088:        DATA (X2(I),I=1,N)/.286,.973,.384,.276,.973,.543,.957,.948,.543,
1089:      & .797,.936,.889,.006,.828,.399,.617,.939,.784,.072,.889/
1090:       DATA (X3(I),I=1,N)/.645,.585,.310,.058,.455,.779,.259,.202,.028,
1091:      & .099,.142,.296,.175,.180,.842,.039,.103,.620,.158,.704/
1092:
1093:       F=0.D00
1094:       DO I=1,N
1095:       F=F+(X(1) + X(2)*X2(I) + (X(2)**2)*X3(I) - Y(I))**2
1096:       ENDDO
1097:       RETURN
1098:       END
1099: C     ----------------------------------------------------------------
1100:       SUBROUTINE DODECAL(M,F,X)
1101:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1102:       DIMENSION X(*)
1103: C     DODECAL POLYNOMIAL MIN F(1,2,3)=0
1104:       F=0.D00
1105:       F1=2*X(1)**3+5*X(1)*X(2)+4*X(3)-2*X(1)**2*X(3)-18.D00
1106:       F2=X(1)+X(2)**3+X(1)*X(2)**2+X(1)*X(3)**2-22.D00
1107:       F3=8*X(1)**2+2*X(2)*X(3)+2*X(2)**2+3*X(2)**3-52.D00
1108:       F=(F1*F3*F2**2+F1*F2*F3**2+F2**2+(X(1)+X(2)-X(3))**2)**2
1109:       RETURN
1110:       END
1111: C     ----------------------------------------------------------------
1112:       SUBROUTINE SEQP(M,F,X)
1113:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1114:       DIMENSION X(*)
1115: C     FOR WHAT VALUES X(1)+X(2)=X(1)*X(2) ? ANSWER: FOR (0,0) AND (2,2)
1116: C     WHILE X(1), X(2) ARE INTEGERS.
1117:       X(1)=INT(X(1)) ! X(1) CONVERTED TO INTEGER
1118:       X(2)=INT(X(2)) ! X(2) CONVERTED TO INTEGER
1119:
1120:       F1=X(1)+X(2)
1121:       F2=X(1)*X(2)
1122:       F=(F1-F2)**2     ! TURN ALIVE THIS  XOR
1123: C      F=DABS(F1-F2)    ! TURN ALIVE THIS  - BUT NOT BOTH --------------
1124:       RETURN
1125:       END
1126: C     ----------------------------------------------------------------
1127:       SUBROUTINE AMGM(M,F,X)
1128:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1129:       DIMENSION X(*)
1130: C     FOR WHAT VALUES ARITHMETIC MEAN = GEOMETRIC MEAN ? THE ANSWER IS:
1131: C     IF X(1)=X(2)=....=X(M) AND ALL X ARE NON-NEGATIVE
1132: C     TAKE ONLY THE ABSOLUTE VALUES OF X
1133:       SUM=0.D00
1134:       DO I=1,M
1135:       X(I)=DABS(X(I))
1136:       ENDDO
1137: C     SET SUM = SOME POSITIVE NUMBER. THIS MAKES THE FUNCTION UNIMODAL
1138:       SUM= 100.D00 ! TURNED  ALIVE FOR UNIQUE MINIMUM AND SET SUM TO
1139: C     SOME POSITIVE NUMBER. HERE IT IS 100; IT COULD BE ANYTHING ELSE.
```

```
1140:       F1=0.D00
1141:       F2=1.D00
1142:       DO I=1,M
1143:       F1=F1+X(I)
1144:       F2=F2*X(I)
1145:       ENDDO
1146:       XSUM=F1
1147:       F1=F1/M ! SUM DIVIDED BY M = ARITHMETIC MEAN
1148:       F2=F2**(1.D00/M) ! MTH ROOT OF THE PRODUCT = GEOMETRIC MEAN
1149:       F=(F1-F2)**2
1150:       IF(SUM.GT.0.D00) F=F+(SUM-XSUM)**2
1151:       RETURN
1152:       END
1153: C     ------------------------------------------------------------------
1154:       SUBROUTINE FUNCT2(M,F,X)
1155: C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
1156: C     IN FOGEL, L.J., ANGELIN, P.J. AND BACK, T. (ED) PROCEEDINGS OF THE
1157: C     FIFTH ANNUAL CONFERENCE ON EVOLUTIONARY PROGRAMMING, PP. 451-460,
1158: C     MIT PRESS, CAMBRIDGE, MASS.
1159: C     MIN F (0, 0, ..., 0) = 0
1160:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1161:       DIMENSION X(*)
1162:       F=0.D00
1163:       F1=1.D00
1164:       DO I=1,M
1165:       IF(DABS(X(I)).GT.10.D00) THEN
1166:       CALL RANDOM(RAND)
1167:       X(I)=(RAND-.5D00)*20
1168:       ENDIF
1169:       ENDDO
1170:       DO I=1,M
1171:       F=F+DABS(X(I))
1172:       F1=F1*DABS(X(I))
1173:       ENDDO
1174:       F=F+F1
1175:       RETURN
1176:       END
1177: C     ------------------------------------------------------------------
1178:       SUBROUTINE FUNCT3(M,F,X)
1179: C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
1180: C     MIN F (0, 0, ... , 0) = 0
1181:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1182:       DIMENSION X(*)
1183:       F=0.D00
1184:       F1=0.D00
1185:       DO I=1,M
1186:       IF(DABS(X(I)).GT.100.D00) THEN
1187:       CALL RANDOM(RAND)
1188:       X(I)=(RAND-.5D00)*200
1189:       ENDIF
1190:       ENDDO
1191:       DO I=1,M
1192:       F1=0.D00
1193:       DO J=1,I
1194:       F1=F1+X(J)**2
1195:       ENDDO
1196:       F=F+F1
1197:       ENDDO
1198:       RETURN
1199:       END
1200: C     ------------------------------------------------------------------
1201:       SUBROUTINE FUNCT4(M,F,X)
1202: C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
1203: C     MIN F (0, 0, ..., 0) = 0
1204:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1205:       DIMENSION X(*)
1206:       F=0.D00
```

```
      DO I=1,M
      IF(X(I).LT.0.D00 .OR. X(I).GE.M) THEN
      CALL RANDOM(RAND)
      X(I)=RAND*2*M
      ENDIF
      ENDDO
C     FIND MAX(X(I))=MAX(ABS(X(I))) NOTE: HERE X(I) CAN BE ONLY POSITIVE
      XMAX=X(1)
      DO I=1,M
      IF(XMAX.LT.X(I)) XMAX=X(I)
      ENDDO
      F=XMAX
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE FUNCT6(M,F,X)
C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
C     MIN F (-.5, -.5, ..., -.5) = 0
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
      F=0.D00
      DO I=1,M
      IF(DABS(X(I)).GT.100.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-.5D00)*200
      ENDIF
      ENDDO
      DO I=1,M
       F=F+(X(I)+0.5D00)**2
      ENDDO
      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 FUNCT12(M,F,X)
C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(100),Y(100)
      DATA A,B,C /10.D00,100.D00,4.D00/
      PI=4.D00*DATAN(1.D00)
      F=0.D00
C     MIN F (-1, -1, -1, ..., -1) = 0
C       X(I)=-1.D00  ! TO CHECK, TURN IT ALIVE
       DO I=1,M
        IF(DABS(X(I)).GT.50.D00) THEN
```

```
1274:         CALL RANDOM(RAND)
1275:         X(I)=(RAND-0.5D00)*100.D00
1276:         ENDIF
1277:         ENDDO
1278:       F1=0.D00
1279:       DO I=1,M
1280:       XX=DABS(X(I))
1281:       U=0.D00
1282:       IF(XX.GT.A) U=B*(XX-A)**C
1283:       F1=F1+U
1284:       ENDDO
1285:       F2=0.D00
1286:       DO I=1,M-1
1287:       Y(I)=1.D00+.25D00*(X(I)+1.D00)
1288:       F2=F2+ (Y(I)-1.D00)**2 * (1.D00+10.D00*(DSIN(PI*X(I+1))**2))
1289:       ENDDO
1290:       Y(M)=1.D00+.25D00*(X(M)+1.D00)
1291:       F3=(Y(M)-1.D00)**2
1292:       Y(1)=1.D00+.25D00*(X(1)+1.D00)
1293:       F4=10.D00*(DSIN(PI*Y(1)))**2
1294:       F=(PI/M)*(F4+F2+F3)+F1
1295:       RETURN
1296:       END
1297: C     ------------------------------------------------------------------
1298:       SUBROUTINE FUNCT13(M,F,X)
1299: C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
1300:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1301:       DIMENSION X(100)
1302:       DATA A,B,C /5.D00,100.D00,4.D00/
1303:       PI=4*DATAN(1.D00)
1304:       F=0.D00
1305: C     MIN F (1, 1, 1, ..., 4.7544 APPROX) = -1.15044 APPROX
1306: C        X(I)=1.D00  ! TO CHECK, TURN IT ALIVE
1307: C        X(M)=-4.7544  ! TO CHECK, TURN IT ALIVE
1308:       DO I=1,M
1309:       IF(DABS(X(I)).GT.50.D00) THEN
1310:       CALL RANDOM(RAND)
1311:       X(I)=(RAND-.5D00)*100.D00
1312:       ENDIF
1313:       ENDDO
1314:       F1=0.D00
1315:       DO I=1,M
1316:       XX=DABS(X(I))
1317:       U=0.D00
1318:       IF(XX.GT.A) U=B*(XX-A)**C
1319:       F1=F1+U
1320:       ENDDO
1321:       F2=0.D00
1322:       DO I=1,M-1
1323:       F2=F2+ (X(I)-1.D00)**2 * (1.D00+(DSIN(3*PI*X(I+1))**2))
1324:       ENDDO
1325:       F3=(X(M)-1.D00)*  (1.D00+(DSIN(2*PI*X(M)))**2)
1326:       F4=(DSIN(3*PI*X(1)))**2
1327:       F=0.1*(F4+F2+F3)+F1
1328:       RETURN
1329:       END
1330: C     ------------------------------------------------------------------
1331:       SUBROUTINE FUNCT14(M,F,X)
1332: C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
1333: C     MIN F (-31.98, 31.98) = 0.998
1334:       PARAMETER (N=25,NN=2)
1335:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1336:       DIMENSION X(2), A(NN,N)
1337:       DATA (A(1,J),J=1,N) /-32.D00,-16.D00,0.D00,16.D00,32.D00,-32.D00,
1338:      & -16.D00,0.D00,16.D00,32.D00,-32.D00,-16.D00,0.D00,16.D00,32.D00,
1339:      & -32.D0,-16.D0,0.D0,16.D0,32.D0,-32.D0,-16.D0,0.D0,16.D0,32.D0/
1340:       DATA (A(2,J),J=1,N) /-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,
```

```
1341:      & -16.D00,-16.D00,-16.D00,-16.D00,-16.D00,0.D00,0.D00,0.D00,0.D00,
1342:      & 0.D00,16.D00,16.D00,16.D00,16.D00,16.D00,32.D00,32.D00,
1343:      & 32.D00,32.D00,32.D00/
1344:
1345:       F=0.D00
1346:       DO I=1,M
1347:       IF(DABS(X(I)).GT.100.D00) THEN
1348:       CALL RANDOM(RAND)
1349:       X(I)=(RAND-.5D00)*200.D00
1350:       ENDIF
1351:       ENDDO
1352:       F1=0.D00
1353:       DO J=1,N
1354:       F2=0.D00
1355:       DO I=1,2
1356:       F2=F2+(X(I)-A(I,J))**6
1357:       ENDDO
1358:       F2=1.D00/(J+F2)
1359:       F1=F1+F2
1360:       ENDDO
1361:       F=1.D00/(0.002D00+F1)
1362:       RETURN
1363:       END
1364: C     ------------------------------------------------------------------
1365:       SUBROUTINE FUNCT15(M,F,X)
1366: C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
1367: C     MIN F(.19, .19, .12, .14) = 0.3075
1368:       PARAMETER (N=11)
1369:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1370:       DIMENSION X(*), A(N),B(N)
1371:       DATA (A(I),I=1,N) /.1957D00,.1947D00,.1735D00,.16D00,.0844D00,
1372:      & .0627D00,.0456D00,.0342D00,.0323D00,.0235D00,.0246D00/
1373:       DATA (B(I),I=1,N) /0.25D00,0.5D00,1.D00,2.D00,4.D00,6.D00,8.D00,
1374:      & 10.D00,12.D00,14.D00,16.D00/
1375:       DO I=1,N
1376:       B(I)=1.D00/B(I)
1377:       ENDDO
1378:       F=0.D00
1379:       DO I=1,M
1380:       IF(DABS(X(I)).GT.5.D00) THEN
1381:       CALL RANDOM(RAND)
1382:       X(I)=(RAND-.5D00)*10.D00
1383:       ENDIF
1384:       ENDDO
1385:       DO I=1,N
1386:       F1=X(1)*(B(I)**2+B(I)*X(2))
1387:       F2=B(I)**2+B(I)*X(3)+X(4)
1388:       F=F+(A(I)-F1/F2)**2
1389:       ENDDO
1390:       F=F*1000
1391:       RETURN
1392:       END
1393: C     ------------------------------------------------------------------
1394:       SUBROUTINE LINPROG1(M,F,X)
1395: C     LINEAR PROGRAMMING : MINIMIZATION PROBLEM
1396: C     IN THIS PROBLEM : M = NO. OF DECISION VARIABLES = 2
1397: C     MIN F (2.390, 2.033) = -19.7253 APPROX
1398: C     MIN F = OVER J=1, M : DO SUM(A(1,J)*X(J))  SUBJECT TO CONSTRAINTS
1399: C     OVER  J=1, M : DO SUM(A(I,J)*X(J)) <= C(I) ; I=2
1400: C     . . .          . . .          . . .
1401: C     OVER  J=1, M : DO SUM(A(I,J)*X(J)) <= C(I) ; I=N
1402: C     ALL X(I) => 0
1403:       PARAMETER (N=3) ! N IS THE NO. OF CONSTRAINTS + 1
1404:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1405:       DIMENSION X(*),A(20,10),C(20),FF(20)
1406:       DATA (A(1,J),J=1,2),C(1)/4.D0,5.D0,0.0D0/!COEFF OF OBJ FUNCTION
1407:       DATA (A(2,J),J=1,2),C(2)/10.D0,3.D0,30D0/!COEFF OF 1ST CONSTRAINT
```

```
1408:       DATA (A(3,J),J=1,2),C(3)/6.D0,20.D0,55.D0/!COEFF OF 2ND CONSTRAINT
1409: C     ------------------------------------------------------------------
1410: C     USING ONLY NON-NEGATIVE VALUES OF X(I)
1411:       DO I=1,M
1412:       X(I)=DABS(X(I))
1413:       ENDDO
1414: C     EVALUATION OF OBJ FUNCTION AND CONSTRAINTS
1415:       DO I=1,N
1416:       FF(I)=0.D00
1417:       DO J=1,M
1418:       FF(I)=FF(I)+A(I,J)*X(J)
1419:       ENDDO
1420:       ENDDO
1421:       F=-FF(1) ! CHANGE OF SIGN FOR MINIMIZATION
1422: C     CHECK FOR SATISFYING OR VIOLATING THE CONSTRAINTS
1423:       DO I=2,N
1424:       FF(I)=FF(I)-C(I) ! SLACK
1425: C     PENALTY FOR CROSSING LIMITS
1426:       IF(FF(I).GT.0) F=F+(10+FF(I))**2
1427:       ENDDO
1428:       RETURN
1429:       END
1430: C     ------------------------------------------------------------------
1431:       SUBROUTINE LINPROG2(M,F,X)
1432: C     LINEAR PROGRAMMING : MINIMIZATION PROBLEM
1433: C     IN THIS PROBLEM : M = NO. OF DECISION VARIABLES = 3
1434: C     MIN F (250, 625, 0) = -3250
1435: C     MIN F = OVER J=1, M : DO SUM(A(1,J)*X(J))   SUBJECT TO CONSTRAINTS
1436: C     OVER  J=1, M : DO SUM(A(I,J)*X(J)) <= C(I) ; I=2
1437: C     . . .          . . .          . . .          . . .
1438: C     OVER  J=1, M : DO SUM(A(I,J)*X(J)) <= C(I) ; I=N
1439: C     ALL X(I) => 0
1440:       PARAMETER (N=4) ! N IS THE NO. OF CONSTRAINTS + 1
1441:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1442:       DIMENSION X(*),A(20,10),C(20),FF(20)
1443:       DATA (A(1,J),J=1,3),C(1)/30.D0,40.D0,20.D0,0.0D0/! COEFF OF OBJ FUNCTION
1444:       DATA (A(2,J),J=1,3),C(2)/10.D0,12.D0,7.D0,10000.0D0/!COEFF OF 1ST CONSTRAINT
1445:       DATA (A(3,J),J=1,3),C(3)/7.D0,10.D0,8.D0,8000.D0/! COEFF OF 2ND CONSTRAINT
1446:       DATA (A(4,J),J=1,3),C(4)/1.D0,1.D0,1.D0,1000.D0/! COEFF OF 3RD CONSTRAINT
1447: C     ------------------------------------------------------------------
1448: C     USING ONLY NON-NEGATIVE VALUES OF X(I)
1449:       DO I=1,M
1450:       X(I)=DABS(X(I))
1451:       ENDDO
1452: C     EVALUATION OF OBJ FUNCTION AND CONSTRAINTS
1453:       DO I=1,N
1454:       FF(I)=0.D00
1455:       DO J=1,M
1456:       FF(I)=FF(I)+A(I,J)*X(J)
1457:       ENDDO
1458:       ENDDO
1459:       F=-FF(1) ! CHANGE OF SIGN FOR MINIMIZATION
1460: C     CHECK FOR SATISFYING OR VIOLATING THE CONSTRAINTS
1461:       DO I=2,N
1462:       FF(I)=FF(I)-C(I) ! SLACK
1463: C     PENALTY FOR CROSSING LIMITS
1464:       IF(FF(I).GT.0.D00) F=F+(100.D00+FF(I))**2
1465:       ENDDO
1466:       RETURN
1467:       END
1468: C     ------------------------------------------------------------------
1469:       SUBROUTINE HOUGEN(A,M,F)
1470:       PARAMETER(N=13,K=3)
1471:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1472:       DIMENSION X(N,K),RATE(N),A(*)
1473: C     ------------------------------------------------------------------
1474: C     HOUGEN FUNCTION (HOUGEN-WATSON MODEL FOR REACTION KINATICS)
```

```
1475: C     NO. OF PARAMETERS (A) TO ESTIMATE = 5 = M
1476: C     -------------------------  BEST RESULTS  ---------------------
1477: C     FMIN =0.298900994 FOR A(1)=1.253031; A(2)=1.190943; A(3)=0.062798;
1478: C     A(4)=0.040063; A(5)=0.112453  BY ROSENBROCK & QUASI-NEWTON METHOD
1479: C     R=0.99945 = CORRELATION COEFF BETWEEN OBSERVED & EXPECTED RATE.
1480: C     next:fmin =0.298901 for x(1.25258611, 0.0627758222, 0.0400477567,
1481: C     0.112414812, 1.19137715) by de with ncross=0, pcros=0.9, fact=0.5
1482: C     THE NEXT BEST RESULTS GIVEN BY HOOKE-JEEVES & QUASI-NEWTON
1483: C     A(1)=2.475221;A(2)=0.599177; A(3)=0.124172; A(4)=0.083517
1484: C     A(5)=0.217886; SUM OF SQUARES OF DEVIATION = 0.318593458 = FMIN
1485: C     R=0.99941. MOST OF THE OTHER METHODS DO NOT PERFORM WELL.
1486: C     ---------------------------------------------------------------
1487:       DATA X(1,1),X(1,2),X(1,3),RATE(1) /470,300,10,8.55/
1488:       DATA X(2,1),X(2,2),X(2,3),RATE(2) /285,80,10,3.79/
1489:       DATA X(3,1),X(3,2),X(3,3),RATE(3) /470,300,120,4.82/
1490:       DATA X(4,1),X(4,2),X(4,3),RATE(4) /470,80,120,0.02/
1491:       DATA X(5,1),X(5,2),X(5,3),RATE(5) /470,80,10,2.75/
1492:       DATA X(6,1),X(6,2),X(6,3),RATE(6) /100,190,10,14.39/
1493:       DATA X(7,1),X(7,2),X(7,3),RATE(7) /100,80,65,2.54/
1494:       DATA X(8,1),X(8,2),X(8,3),RATE(8) /470,190,65,4.35/
1495:       DATA X(9,1),X(9,2),X(9,3),RATE(9) /100,300,54,13/
1496:       DATA X(10,1),X(10,2),X(10,3),RATE(10) /100,300,120,8.5/
1497:       DATA X(11,1),X(11,2),X(11,3),RATE(11) /100,80,120,0.05/
1498:       DATA X(12,1),X(12,2),X(12,3),RATE(12) /285,300,10,11.32/
1499:       DATA X(13,1),X(13,2),X(13,3),RATE(13) /285,190,120,3.13/
1500: C     WRITE(*,1)((X(I,J),J=1,K),RATE(I),I=1,N)
1501: C   1 FORMAT(4F8.2)
1502:       DO J=1,M
1503:       IF(DABS(A(J)).GT.5.D00) THEN
1504:       CALL RANDOM(RAND)
1505:       A(J)=(RAND-0.5D00)*10.D00
1506:       ENDIF
1507:       ENDDO
1508:       F=0.D00
1509:       DO I=1,N
1510:       D=1.D00
1511:          DO J=1,K
1512:          D=D+A(J+1)*X(I,J)
1513:          ENDDO
1514:       FX=(A(1)*X(I,2)-X(I,3)/A(M))/D
1515: C     FX=(A(1)*X(I,2)-X(I,3)/A(5))/(1.D00+A(2)*X(I,1)+A(3)*X(I,2)+
1516: C     A(4)*X(I,3))
1517:       F=F+(RATE(I)-FX)**2
1518:       ENDDO
1519:       RETURN
1520:       END
1521: C     ---------------------------------------------------------------
1522:       SUBROUTINE GIUNTA(M,X,F)
1523:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1524:       DIMENSION X(*)
1525: C     GIUNTA FUNCTION
1526: C     X(I) = -1 TO 1;  M=2
1527:         DO I=1,M
1528:         IF(DABS(X(I)).GT.1.D00) THEN
1529:         CALL RANDOM(RAND)
1530:         X(I)=(RAND-0.5D00)*2.D00
1531:         ENDIF
1532:         ENDDO
1533:       C=16.D00/15.D00
1534:       F=DSIN(C*X(1)-1.D0)+DSIN(C*X(1)-1.D0)**2+DSIN(4*(C*X(1)-1.D0))/50+
1535:      &DSIN(C*X(2)-1.D0)+DSIN(C*X(2)-1.D0)**2+DSIN(4*(C*X(2)-1.D0))/50+.6
1536:       RETURN
1537:       END
1538: C     ---------------------------------------------------------------
1539:       SUBROUTINE EGGHOLD(M,X,F)
1540: C     EGG HOLDER FUNCTION
1541:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
```

```
      DIMENSION X(*)
      DO I=1,M
       IF(DABS(X(I)).GT.512.D00) THEN
        CALL RANDOM(RAND)
        X(I)=(RAND-0.5D00)*1024.D00
       ENDIF
       ENDDO
       F=0.D00
      DO I=1,M-1
      F1=-(X(I+1)+47.D00)
      F2=DSIN(  DSQRT(  DABS(  X(I+1)+X(I)/2+47.D00   )   )   )
      F3=DSIN(  DSQRT(  DABS(  X(I)-(X(I+1)+47.D00) ) ))
      F4=-X(I)
      F=F+ F1*F2+F3*F4
      ENDDO
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE TRID(M,X,F)
C     TRID FUNCTION
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
      F1=0.D00
      F2=0.D00
      DO I=1, M
      F1=F1+(X(I)-1.D00)**2
      ENDDO
      DO I=2, M
      F2=F2+X(I)*X(I-1)
      ENDDO
      F=F1-F2
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE GRIEWANK(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
C     GRIEWANK FUNCTION
      F1=0.D00
      F2=1.0D00
      DO I=1,M
      F1=F1+X(I)**2
      FI=DFLOAT(I)
      F2=F2*DCOS(X(I)/DSQRT(FI))
      ENDDO
      F=F1/4000.D00-F2+1.0D00
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE WEIERSTRASS(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
      PI=4*DATAN(1.D00)
C     WEIERSTRASS FUNCTION m variable fmin (0, 0, ..., 0)= 0
      DATA AX,BX,KMAX/0.5D00,3.D00,20/
        DO I=1,M
        IF(DABS(X(I)).GT.0.5D00) THEN
        CALL RANDOM(RAND)
        X(I)=RAND-0.5D00
        ENDIF
        ENDDO
      f1=0.D00
      f2=0.D00
      DO I=1,M
      S=0.D00
          DO KK=1,KMAX+1
          K=KK-1
```

```
1609:           S=S+(AX**K*DCOS(2*PI*BX**K*(X(I)+0.5D00)))
1610:           ENDDO
1611:         F1=F1+S
1612:         ENDDO
1613:         DO KK=1,KMAX+1
1614:         K=KK-1
1615:         F2=F2+(AX**K*DCOS(2*PI*BX**K*0.5D00))
1616:         ENDDO
1617:         F=F1-M*F2
1618:         RETURN
1619:         END
1620: C       ------------------------------------------------------------------
1621:         SUBROUTINE LEVY3(M,X,F)
1622:         IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1623:         DIMENSION X(*)
1624: C       LEVY # 3 (LEVY ET AL. 1981) ----------------------------------
1625:            DO I=1,M
1626:            IF(DABS(X(I)).GT.10.D00) THEN
1627:            CALL RANDOM(RAND)
1628:            X(I)=(RAND-0.5D00)*20
1629:            ENDIF
1630:            ENDDO
1631:             F1=0.0D+00
1632:             F2=0.0D+00
1633:             DO I=1,5
1634:             F1=F1+(I*DCOS((I-1)*X(1)+I))
1635:             F2=F2+(I*DCOS((I+1)*X(2)+I))
1636:             ENDDO
1637:            F=F1*F2
1638:            RETURN
1639:            END
1640: C       ------------------------------------------------------------------
1641:         SUBROUTINE LEVY5(M,X,F)
1642:         IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1643:         DIMENSION X(*)
1644:               F1=0.0D+00
1645:             F2=0.0D+00
1646:           DO I=1,5
1647:          F1=F1+(I*DCOS((I-1)*X(1)+I))
1648:             F2=F2+(I*DCOS((I+1)*X(2)+I))
1649:             ENDDO
1650:             F3=(X(1)+1.42513D+00)**2
1651:            F4=(X(2)+0.80032D+00)**2
1652:            F=(F1*F2) + (F3+F4)
1653:            RETURN
1654:            END
1655: C       ------------------------------------------------------------------
1656:         SUBROUTINE LEVY8(M,X,F)
1657:         IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1658:         DIMENSION X(*),Y(3)
1659:         PI=4*DATAN(1.D00)
1660: C       LEVY # 8 FUNCTION -------------------------------------------------
1661:            DO I=1,3
1662:            Y(I)=1.D+00+(X(I)-1.D+00)/4.D+00
1663:            ENDDO
1664:            F1=DSIN(PI*Y(1))**2
1665:          F3=(Y(3)-1.D+00)**2
1666:         F2=0.D+00
1667:          DO I=1,2
1668:            F2=F2+((Y(I)-1.D+00)**2)*(1.D+00+10.D+00*(DSIN(PI*Y(I+1)))**2)
1669:            ENDDO
1670:         F=F1+F2+F3
1671:         RETURN
1672:         END
1673: C       ------------------------------------------------------------------
1674:         SUBROUTINE RASTRIGIN(M,X,F)
1675:         IMPLICIT DOUBLE PRECISION (A-H, O-Z)
```

```
      DIMENSION X(*)
      PI=4*DATAN(1.D00)
C     RASTRIGIN'S FUNCTION
      F=0.D00
      DO I=1,M
      F=F+ X(I)**2 -10*DCOS(2*PI*X(I)) + 10.D00
      ENDDO
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE ACKLEY(M,X,F)
C     ACKLEY FUNCTION
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
      PI=4*DATAN(1.D00)
      F=20.D00+DEXP(1.D00)
      DO I=1,M
      IF(X(I).LT. -15.D00 .OR. X(I).GT.30.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-0.5D00)*90 -15.D00
      ENDIF
      ENDDO
      F1=0.D00
      F2=0.D00
        DO I=1,M
         F1=F1+X(I)**2
         F2=F2+DCOS(2*PI*X(I))
        ENDDO
      F1=-20*DEXP(-0.2D00*DSQRT(F1/M))
      F2=-DEXP(F2/M)
      F=F+F1+F2
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE MICHALEWICZ(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
C     MICHALEWICZ FUNCTION   [ 0 <= X(I) <= PI ] MP IS A PARAMETER
C     min f= -1.8013 (M=2); -4.6877 (M=5); -6.6809 (M=7); -9.6602 (M=10)
C     -19.6370 (M=20); -29.6309 (M=30); -49.6248 (M=50), ETC
      MP=10  ! SET IT TO THE DESIRED VALUE
      PI=4*DATAN(1.D00)
      DO I=1,M
      IF(X(I).LT.0.D00 .OR. X(I).GT.PI)THEN
      CALL RANDOM(RAND)
      X(I)=RAND*PI
      ENDIF
      ENDDO
      F=0.D00
      DO I=1,M
      F=F-DSIN(X(I))*(DSIN(I*X(I)**2/PI))**(2*MP)
      ENDDO
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE SCHWEFEL(M,X,F)
C     SCHWEFEL FUNCTION
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
      DO I=1,M
      IF(DABS(X(I)).GT.500) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-0.5D00)*1000
      ENDIF
      ENDDO
      F=0.D00
      DO I=1,M
```

```
1743:       F=F+ X(I)*DSIN(DSQRT(DABS(X(I))))
1744:       ENDDO
1745:       F=418.9829D00*M - F
1746:       RETURN
1747:       END
1748: C     ------------------------------------------------------------------
1749:       SUBROUTINE SHUBERT(M,X,F)
1750: C     SHUBERT FUNCTION
1751:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1752:       DIMENSION X(*)
1753:          DO I=1,M
1754:          IF(DABS(X(I)).GT.10.D00) THEN
1755:          CALL RANDOM(RAND)
1756:          X(I)=(RAND-0.5D00)*20
1757:          ENDIF
1758:          ENDDO
1759:       F1=0.D00
1760:       F2=0.D00
1761:       DO I=1,5
1762:       F1=F1+I*DCOS((I+1.D00)*X(1)+I)
1763:       F2=F2+I*DCOS((I+1.D00)*X(2)+I)
1764:       ENDDO
1765:       F=F1*F2
1766:       RETURN
1767:       END
1768: C     ------------------------------------------------------------------
1769:       SUBROUTINE DIXPRICE(M,X,F)
1770: C     DIXON & PRICE FUNCTION
1771:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1772:       DIMENSION X(*)
1773:        DO I=1,M
1774:          IF(DABS(X(I)).GT.10.D00) THEN
1775:          CALL RANDOM(RAND)
1776:          X(I)=(RAND-0.5D00)*20
1777:          ENDIF
1778:        ENDDO
1779:       F=0.D00
1780:       DO I=2, M
1781:       F=F + I*(2*X(I)**2-X(I-1))**2
1782:       ENDDO
1783:       F=F+(X(1)-1.D00)**2
1784:       RETURN
1785:       END
1786: C     ------------------------------------------------------------------
1787:       SUBROUTINE SHEKEL(M,X,F)
1788: C     SHEKEL FUNCTION FOR TEST OF GLOBAL OPTIMIZATION METHODS
1789:       PARAMETER(NROW=10,NCOL=4, NR=9)! NR MAY BE 2 TO 10
1790:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1791:       DIMENSION A(NROW,NCOL),C(NROW),X(*)
1792:       DATA ((A(I,J),J=1,NCOL),I=1,NROW)/4.,4.,4.,4.,1.,1.,1.,1.,8.,8.,
1793:      & 8.,8.,6.,6.,6.,6.,3.,7.,3.,7.,2.,9.,2.,9.,5.,5.,3.,3.,8.,1.,8.,
1794:      & 1.,6.,2.,6.,2.,7.,3.6D00,7.,3.6D00/
1795:       DATA (C(I),I=1,NROW)/0.1D00,0.2D00,0.2D00,0.4D00,0.4D00,0.6D00,
1796:      & 0.3D00,0.7D00,0.5D00,0.5D00/
1797:       F=0.D00
1798:       DO I=1,NR
1799:         S=0.D00
1800:         DO J=1,M
1801:         S=S+(X(J)-A(I,J))**2
1802:         ENDDO
1803:       F=F-1.D00/(S+C(I))
1804:       ENDDO
1805:       RETURN
1806:       END
1807: C     ------------------------------------------------------------------
1808:       SUBROUTINE PAVIANI(M,X,F)
1809: C     PAVIANI FUNCTION : MIN F(9.3502,...,9.3502)=45.77847 APPROX
```

```
1810: C     IN THE DOMAIN 2<= X(I) <= 10 FOR I=1,2,...,10.
1811:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1812:       DIMENSION X(*)
1813:       DO I=1,M
1814:       IF(X(I).LE.2.D00.OR.X(I).GE.10.D00) THEN
1815:       CALL RANDOM(RAND)
1816:       X(I)=RAND*8+2.D00
1817:       ENDIF
1818:       ENDDO
1819:       F1=0.D00
1820:       F2=1.D00
1821:       DO I=1,M
1822:       F1=F1+ DLOG(X(I)-2.D00)**2+DLOG(10.D00-X(I))**2
1823:       F2=F2*X(I)
1824:       ENDDO
1825:       F=F1-F2**0.2
1826:       RETURN
1827:       END
1828: C     ---------------------------------------------------------------
1829:       SUBROUTINE BRANIN1(M,X,F)
1830: C     BRANIN FUNCTION #1  MIN F (1, 0) = 0
1831:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1832:       DIMENSION X(*)
1833:       PI=4*DATAN(1.D00)
1834:       DO I=1,M
1835:       IF(DABS(X(I)).GT.10.D00) THEN
1836:       CALL RANDOM(RAND)
1837:       X(I)=(RAND-0.5D00)*20
1838:       ENDIF
1839:       ENDDO
1840:       F=(1.D00-2*X(2)+DSIN(4*PI*X(2))/2.D00-X(1))**2+(X(2)-
1841:      & DSIN(2*PI*X(1))/2.D00)**2
1842:       RETURN
1843:       END
1844: C     ---------------------------------------------------------------
1845:       SUBROUTINE BRANIN2(M,X,F)
1846: C     BRANIN FUNCTION #2 MIN F (3.1416, 2.25)= 0.397887 APPROX
1847:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1848:       DIMENSION X(*)
1849:       PI=4*DATAN(1.D00)
1850:       IF(X(1).LT.-5.D00 .OR. X(1).GT.10.D00) THEN
1851:       CALL RANDOM(RAND)
1852:       X(1)=RAND*15-5.D00
1853:       ENDIF
1854:       IF(X(2).LT.0.D00 .OR. X(2).GT.15.D00) THEN
1855:       CALL RANDOM(RAND)
1856:       X(2)=RAND*15
1857:       ENDIF
1858:       F=(X(2)-5.D00*X(1)**2/(4*PI**2)+5*X(1)/PI-6.D00)**2 +
1859:      &    10*(1.D00-1.D00/(8*PI))*DCOS(X(1))+10.D00
1860:       RETURN
1861:       END
1862: C     ---------------------------------------------------------------
1863:       SUBROUTINE BOHACHEVSKY1(M,X,F)
1864: C     BOHACHEVSKY FUNCTION #1 : MIN F (0, 0) = 0
1865:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1866:       DIMENSION X(*)
1867:       PI=4*DATAN(1.D00)
1868:       DO I=1,M
1869:       IF(DABS(X(I)).GT.100.D00) THEN
1870:       CALL RANDOM(RAND)
1871:       X(I)=(RAND-0.5D00)*200
1872:       ENDIF
1873:       ENDDO
1874:       F=X(1)**2+2*X(2)**2-0.3D00*DCOS(3*PI*X(1))-0.4D00*DCOS(4*PI*X(2))
1875:      & +0.7D00
1876:       RETURN
```

```
      END
C     ------------------------------------------------------------------
      SUBROUTINE BOHACHEVSKY2(M,X,F)
C     BOHACHEVSKY FUNCTION #2  : MIN F (0, 0) = 0
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
      PI=4*DATAN(1.D00)
      DO I=1,M
      IF(DABS(X(I)).GT.100.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-0.5D00)*200
      ENDIF
      ENDDO
      F=X(1)**2+2*X(2)**2-0.3D00*DCOS(3*PI*X(1))*DCOS(4*PI*X(2))+0.3D00
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE BOHACHEVSKY3(M,X,F)
C     BOHACHEVSKY FUNCTION #3   : MIN F (0, 0) = 0
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
      PI=4*DATAN(1.D00)
      DO I=1,M
      IF(DABS(X(I)).GT.100.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-0.5D00)*200
      ENDIF
      ENDDO
      F=X(1)**2+2*X(2)**2-0.3D00*DCOS(3*PI*X(1)+4*PI*X(2))+0.3D00
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE EASOM(M,X,F)
C     EASOM FUNCTION : 2-VARIABLES, MIN F (PI, PI) = -1.
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
      PI=4*DATAN(1.D00)
      DO I=1,M
      IF(DABS(X(I)).GT.100.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-0.5D00)*200
      ENDIF
      ENDDO
      F=-DCOS(X(1))*DCOS(X(2))*DEXP(-(X(1)-PI)**2 -(X(2)-PI)**2)
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE ROSENBROCK(M,X,F)
C     ROSENBROCK FUNCTION : M VARIABLE; MIN F (1, 1,...,1)=0
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
      DO I=1,M
      IF(X(I).LT.-5.D00 .OR. X(I).GT.10.D00) THEN
      CALL RANDOM(RAND)
      X(I)=RAND*15-5.D00
      ENDIF
      ENDDO
      F=0.D00
      DO I=1,M-1
      F=F+ (100.D00*(X(I+1)-X(I)**2)**2 + (X(I)-1.D00)**2)
      ENDDO
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE CROSSLEG(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
```

```
1944: C     CROSS-LEGGED TABLE FUNCTION ; -10<= X(I) <=10; M=2
1945: C     MIN F(0 , X ) OR F(X, 0) = -1.
1946:       PI=4*DATAN(1.D00)
1947:         DO I=1,M
1948:         IF( DABS(X(I)).GT.10.D00) THEN
1949:         CALL RANDOM(RAND)
1950:         X(I)=(RAND-0.5D00)*20
1951:         ENDIF
1952:         ENDDO
1953:       F=-(DABS(DSIN(X(1))*DSIN(X(2))*DEXP(DABS(100.D00-(DSQRT
1954:      & (X(1)**2+X(2)**2)/PI))))+1.D00)**(-.1)
1955:       RETURN
1956:       END
1957: C     ------------------------------------------------------------------
1958:       SUBROUTINE CROSS(M,X,F)
1959:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1960:       DIMENSION X(*)
1961: C     CROSS FUNCTION ; -10<= X(I) <=10; M=2;
1962: C     MIN F(A, B)=0 APPROX; A, B=1.3494 APPROX OF EITHER SIGN (+ OR -)
1963:       PI=4*DATAN(1.D00)
1964:         DO I=1,M
1965:         IF( DABS(X(I)).GT.10.D00) THEN
1966:         CALL RANDOM(RAND)
1967:         X(I)=(RAND-0.5D00)*20
1968:         ENDIF
1969:         ENDDO
1970:       F=(DABS(DSIN(X(1))*DSIN(X(2))*DEXP(DABS(100.D00-(DSQRT
1971:      & (X(1)**2+X(2)**2)/PI))))+1.D00)**(-.1)
1972:       RETURN
1973:       END
1974: C     ------------------------------------------------------------------
1975:       SUBROUTINE CROSSINTRAY(M,X,F)
1976:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1977:       DIMENSION X(*)
1978: C     CROSS IN TRAY FUNCTION ; -10<= X(I) <=10; M=2;
1979: C     MIN F(A, B)=-20626.1218 APPROX; A, B=1.3494 APPROX OF EITHER SIGN
1980:       PI=4*DATAN(1.D00)
1981:         DO I=1,M
1982:         IF( DABS(X(I)).GT.10.D00) THEN
1983:         CALL RANDOM(RAND)
1984:         X(I)=(RAND-0.5D00)*20
1985:         ENDIF
1986:         ENDDO
1987:       F=-(DABS(DSIN(X(1))*DSIN(X(2))*DEXP(DABS(100.D00-(DSQRT
1988:      & (X(1)**2+X(2)**2)/PI))))+1.D00)**(.1)
1989:       RETURN
1990:       END
1991: C     ------------------------------------------------------------------
1992:       SUBROUTINE CROWNEDCROSS(M,X,F)
1993:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1994:       DIMENSION X(*)
1995: C     CROWNED CROSS FUNCTION ; -10<= X(I) <=10; M=2; MIN F = 1
1996:       PI=4*DATAN(1.D00)
1997:         DO I=1,M
1998:         IF( DABS(X(I)).GT.10.D00) THEN
1999:         CALL RANDOM(RAND)
2000:         X(I)=(RAND-0.5D00)*20
2001:         ENDIF
2002:         ENDDO
2003:       F=(DABS(DSIN(X(1))*DSIN(X(2))*DEXP(DABS(100.D00-
2004:      & (DSQRT(X(1)**2+X(2)**2)/PI))))+1.D00)**(.1)
2005:       RETURN
2006:       END
2007: C     ------------------------------------------------------------------
2008:       SUBROUTINE TTHOLDER(M,X,F)
2009:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2010:       DIMENSION X(*)
```

```
2011: C      TEST-TUBE HOLDER FUNCTION ; -10<= X(I) <=10; M=2;
2012: C      MIN F([+/-]1.5706, 0)= -10.8723
2013:        PI=4*DATAN(1.D00)
2014:          DO I=1,M
2015:          IF( DABS(X(I)).GT.10.D00) THEN
2016:          CALL RANDOM(RAND)
2017:          X(I)=(RAND-0.5D00)*20
2018:          ENDIF
2019:          ENDDO
2020:        F=-4*DABS(DSIN(X(1))*DCOS(X(2))*DEXP(DABS(DCOS((X(1)**2+X(2)**2)/
2021:       & 200))))
2022:        RETURN
2023:        END
2024: C      ------------------------------------------------------------------
2025:        SUBROUTINE HOLDERTABLE(M,X,F)
2026:        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2027:        DIMENSION X(*)
2028: C      HOLDER-TABLE FUNCTION  ; -10<= X(I) <=10; M=2;
2029: C      MIN F ([+/-] 9.64617, [+/-] 9.64617) APPROX = -26.92034 APPROX
2030:        PI=4*DATAN(1.D00)
2031:          DO I=1,M
2032:          IF( DABS(X(I)).GT.10.D00) THEN
2033:          CALL RANDOM(RAND)
2034:          X(I)=(RAND-0.5D00)*20
2035:          ENDIF
2036:          ENDDO
2037:        F=-DABS(DCOS(X(1))*DCOS(X(2))*DEXP(DABS(1.D00-(DSQRT(X(1)**2+
2038:       & X(2)**2)/PI))))
2039:        RETURN
2040:        END
2041: C      ------------------------------------------------------------------
2042:        SUBROUTINE CARROMTABLE(M,X,F)
2043:        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2044:        DIMENSION X(*)
2045: C      CARROM-TABLE FUNCTION  ; -10<= X(I) <=10; M=2;
2046: C      MIN F ([+/-] 9.64617, [+/-] 9.64617) APPROX = -24.15682 APPROX
2047:        PI=4*DATAN(1.D00)
2048:          DO I=1,M
2049:          IF( DABS(X(I)).GT.10.D00) THEN
2050:          CALL RANDOM(RAND)
2051:          X(I)=(RAND-0.5D00)*20
2052:          ENDIF
2053:          ENDDO
2054:        F=-1.D00/30*(DCOS(X(1))*DCOS(X(2))*DEXP(DABS(1.D00-
2055:       & (DSQRT(X(1)**2 + X(2)**2)/PI))))**2
2056:        RETURN
2057:        END
2058: C      ------------------------------------------------------------------
2059:        SUBROUTINE PENHOLDER(M,X,F)
2060:        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2061:        DIMENSION X(*)
2062: C      PENHOLDER FUNCTION  ; -11<= X(I) <=11; M=2;
2063: C      MIN F ([+/-] 9.64617, [+/-] 9.64617) APPROX = -0.963535 APPROX
2064:        PI=4*DATAN(1.D00)
2065:          DO I=1,M
2066:          IF( DABS(X(I)).GT.11.D00) THEN
2067:          CALL RANDOM(RAND)
2068:          X(I)=(RAND-0.5D00)*22
2069:          ENDIF
2070:          ENDDO
2071:        F=-DEXP(-(DABS(DCOS(X(1))*DCOS(X(2))*DEXP(DABS(1.D0-(DSQRT
2072:       & (X(1)**2+X(2)**2)/PI))))**(-1)))
2073:        RETURN
2074:        END
2075: C      ------------------------------------------------------------------
2076:        SUBROUTINE BIRD(M,X,F)
2077:        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
```

```
      DIMENSION X(*)
C     BIRD FUNCTION ; -2PI<= X(I) <=2PI; M=2;
C     MIN F (4.70104, 3.15294) APPROX = -106.764537 APPROX  OR
C     MIN F (-1.58214, -3.13024) APPROX = -106.764537 APPROX
      PI=4*DATAN(1.D00)
        DO I=1,M
        IF( DABS(X(I)).GT.2*PI) THEN
        CALL RANDOM(RAND)
        X(I)=(RAND-0.5D00)*4*PI
        ENDIF
        ENDDO
      F=(DSIN(X(1))*DEXP((1.D00-DCOS(X(2)))**2) +
     & DCOS(X(2))*DEXP((1.D00-DSIN(X(1)))**2))+(X(1)-X(2))**2
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE CHICHINADZE(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
C     CHICHINADZE FUNCTION :  -30 <=X(I)<= 30;  M=2
C     MIN F (5.901329, 0.5) = -43.3158621
        PI=4*DATAN(1.D00)
        DO I=1,M
        IF( DABS(X(I)).GT.30) THEN
        CALL RANDOM(RAND)
        X(I)=(RAND-0.5D00)*60
        ENDIF
        ENDDO
      F=X(1)**2-12*X(1)+11.D00+10*DCOS(PI*X(1)/2)+8*DSIN(5*PI*X(1))-
     & (1.D00/DSQRT(5.D00))*DEXP(-(X(2)-0.5D00)**2/2)
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE MCCORMICK(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
C     MCCORMICK FUNCTION : -1.5<= X(1)<=4; -3<=X(2)<=4 ; M=2
C     MIN F (-0.54719755, -1.54719755) = -1.913223 APPROX
        IF(X(1).LT. -1.5D00 .OR. X(1) .GT. 4.D00) THEN
        CALL RANDOM(RAND)
        X(1)=RAND*5.5D00-1.5D00
        ENDIF
        IF(X(2).LT. -3.D00 .OR. X(2) .GT. 4.D00) THEN
        CALL RANDOM(RAND)
        X(2)=RAND*7.D00-3.D00
        ENDIF
      F=DSIN(X(1)+X(2))+(X(1)-X(2))**2-1.5*X(1)+2.5*X(2)+1.D00
      RETURN
      END
C     ==================================================================
C     ==============  REPULSIVE PARTICLE SWARM METHOD  ================
      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=50,NSTEP=11,ITRN=10000,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.
```

```
2145: C     IN F(X1, X2,...,XM) M SHOULD BE LESS THAN OR EQUAL TO MX. ITRN IS
2146: C     THE NO. OF ITERATIONS. IT MAY DEPEND ON THE PROBLEM. 200(AT LEAST)
2147: C     TO 500 ITERATIONS MAY BE GOOD ENOUGH. BUT FOR FUNCTIONS LIKE
2148: C     ROSENBROCKOR GRIEWANK OF LARGE SIZE (SAY M=30) IT IS NEEDED THAT
2149: C     ITRN IS LARGE, SAY 5000 OR EVEN 10000.
2150: C     SIGMA INTRODUCES PERTURBATION & HELPS THE SEARCH JUMP OUT OF LOCAL
2151: C     OPTIMA. FOR EXAMPLE : RASTRIGIN FUNCTION OF DMENSION 3O OR LARGER
2152: C     NSTEP DOES LOCAL SEARCH BY TUNNELLING AND WORKS WELL BETWEEN 5 AND
2153: C     15, WHICH IS MUCH ON THE HIGHER SIDE.
2154: C     ITOP <=1 (RING); ITOP=2 (RING AND RANDOM); ITOP=>3 (RANDOM)
2155: C     NSIGMA=0 (NO CHAOTIC PERTURBATION);NSIGMA=1 (CHAOTIC PERTURBATION)
2156: C     NOTE THAT NSIGMA=1 NEED NOT ALWAYS WORK BETTER (OR WORSE)
2157: C     SUBROUTINE FUNC( ) DEFINES OR CALLS THE FUNCTION TO BE OPTIMIZED.
2158:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2159:       COMMON /RNDM/IU,IV
2160:       COMMON /KFF/KF,NFCALL
2161:       INTEGER IU,IV
2162:       CHARACTER *70 FTIT
2163:       DIMENSION X(N,MX),V(N,MX),A(MX),VI(MX)
2164:       DIMENSION XX(N,MX),F(N),V1(MX),V2(MX),V3(MX),V4(MX),BST(MX)
2165: C     A1 A2 AND A3 ARE CONSTANTS AND W IS THE INERTIA WEIGHT.
2166: C     OCCASIONALLY, TINKERING WITH THESE VALUES, ESPECIALLY A3, MAY BE
2167: C     NEEDED.
2168:       DATA A1,A2,A3,W,SIGMA,EPSI /.5D0,.5D0,5.D-04,.5D00,1.D-03,1.D-08/
2169: C     ----------------------------------------------------------------
2170: C     CALL SUBROUTINE FOR CHOOSING FUNCTION (KF) AND ITS DIMENSION (M)
2171:       CALL FSELECT(KF,M,FTIT)
2172: C     ----------------------------------------------------------------
2173:       GGBEST=1.D30 ! TO BE USED FOR TERMINATION CRITERION
2174:       LCOUNT=0
2175:       NFCALL=0
2176:       WRITE(*,*)'4-DIGITS SEED FOR RANDOM NUMBER GENERATION'
2177:       WRITE(*,*)'A FOUR-DIGIT POSITIVE ODD INTEGER, SAY, 1171'
2178:       READ(*,*) IU
2179:       DATA FMIN /1.0E30/
2180: C     GENERATE N-SIZE POPULATION OF M-TUPLE PARAMETERS X(I,J) RANDOMLY
2181:       DO I=1,N
2182:         DO J=1,M
2183:         CALL RANDOM(RAND)
2184:          X(I,J)=(RAND-0.5D00)*2000
2185: C     WE GENERATE RANDOM(-5,5). HERE MULTIPLIER IS 10. TINKERING IN SOME
2186: C     CASES MAY BE NEEDED
2187:          ENDDO
2188:         F(I)=1.0D30
2189:       ENDDO
2190: C     INITIALISE VELOCITIES V(I) FOR EACH INDIVIDUAL IN THE POPULATION
2191:       DO I=1,N
2192:       DO J=1,M
2193:       CALL RANDOM(RAND)
2194:        V(I,J)=(RAND-0.5D+00)
2195: C       V(I,J)=RAND
2196:       ENDDO
2197:       ENDDO
2198:       DO 100 ITER=1,ITRN
2199: C     LET EACH INDIVIDUAL SEARCH FOR THE BEST IN ITS NEIGHBOURHOOD
2200:         DO I=1,N
2201:            DO J=1,M
2202:            A(J)=X(I,J)
2203:            VI(J)=V(I,J)
2204:            ENDDO
2205:            CALL LSRCH(A,M,VI,NSTEP,FI)
2206:            IF(FI.LT.F(I)) THEN
2207:             F(I)=FI
2208:             DO IN=1,M
2209:             BST(IN)=A(IN)
2210:             ENDDO
2211: C     F(I) CONTAINS THE LOCAL BEST VALUE OF FUNCTION FOR ITH INDIVIDUAL
```

```
2212: C     XX(I,J) IS THE M-TUPLE VALUE OF X ASSOCIATED WITH LOCAL BEST F(I)
2213:              DO J=1,M
2214:              XX(I,J)=A(J)
2215:              ENDDO
2216:              ENDIF
2217:         ENDDO
2218: C      NOW LET EVERY INDIVIDUAL RANDOMLY COSULT NN(<<N) COLLEAGUES AND
2219: C      FIND THE BEST AMONG THEM
2220:       DO I=1,N
2221: C     ----------------------------------------------------------------------
2222:       IF(ITOP.GE.3) THEN
2223: C     RANDOM TOPOLOGY ***************************************
2224: C     CHOOSE NN COLLEAGUES RANDOMLY AND FIND THE BEST AMONG THEM
2225:           BEST=1.0D30
2226:            DO II=1,NN
2227:                 CALL RANDOM(RAND)
2228:                NF=INT(RAND*N)+1
2229:                 IF(BEST.GT.F(NF)) THEN
2230:                  BEST=F(NF)
2231:                 NFBEST=NF
2232:                  ENDIF
2233:           ENDDO
2234:       ENDIF
2235: C-----------------------------------------------------------------------
2236:       IF(ITOP.EQ.2) THEN
2237: C     RING + RANDOM TOPOLOGY ********************************************
2238: C     REQUIRES THAT THE SUBROUTINE NEIGHBOR IS TURNED ALIVE
2239:        BEST=1.0D30
2240:           CALL NEIGHBOR(I,N,I1,I3)
2241:           DO II=1,NN
2242:                 IF(II.EQ.1) NF=I1
2243:                  IF(II.EQ.2) NF=I
2244:                   IF(II.EQ.3) NF=I3
2245:                       IF(II.GT.3) THEN
2246:                      CALL RANDOM(RAND)
2247:                       NF=INT(RAND*N)+1
2248:                      ENDIF
2249:                   IF(BEST.GT.F(NF)) THEN
2250:                   BEST=F(NF)
2251:                   NFBEST=NF
2252:                  ENDIF
2253:             ENDDO
2254:        ENDIF
2255: C-----------------------------------------------------------------------
2256:       IF(ITOP.LE.1) THEN
2257: C     RING TOPOLOGY *****************************************************
2258: C     REQUIRES THAT THE SUBROUTINE NEIGHBOR IS TURNED ALIVE
2259:         BEST=1.0D30
2260:            CALL NEIGHBOR(I,N,I1,I3)
2261:               DO II=1,3
2262:               IF (II.NE.I) THEN
2263:              IF(II.EQ.1) NF=I1
2264:               IF(II.EQ.3) NF=I3
2265:                   IF(BEST.GT.F(NF)) THEN
2266:                    BEST=F(NF)
2267:                    NFBEST=NF
2268:                   ENDIF
2269:                   ENDIF
2270:             ENDDO
2271:        ENDIF
2272: C-----------------------------------------------------------------------
2273: C     IN THE LIGHT OF HIS OWN AND HIS BEST COLLEAGUES EXPERIENCE, THE
2274: C     INDIVIDUAL I WILL MODIFY HIS MOVE AS PER THE FOLLOWING CRITERION
2275: C     FIRST, ADJUSTMENT BASED ON ONES OWN EXPERIENCE
2276: C     AND OWN BEST EXPERIENCE IN THE PAST (XX(I))
2277:           DO J=1,M
2278:           CALL RANDOM(RAND)
```

```
2279:            V1(J)=A1*RAND*(XX(I,J)-X(I,J))
2280: C      THEN BASED ON THE OTHER COLLEAGUES BEST EXPERIENCE WITH WEIGHT W
2281: C      HERE W IS CALLED AN INERTIA WEIGHT 0.01< W < 0.7
2282: C      A2 IS THE CONSTANT NEAR BUT LESS THAN UNITY
2283:            CALL RANDOM(RAND)
2284:            V2(J)=V(I,J)
2285:            IF(F(NFBEST).LT.F(I)) THEN
2286:            V2(J)=A2*W*RAND*(XX(NFBEST,J)-X(I,J))
2287:            ENDIF
2288: C      THEN SOME RANDOMNESS AND A CONSTANT A3 CLOSE TO BUT LESS THAN UNITY
2289:            CALL RANDOM(RAND)
2290:            RND1=RAND
2291:            CALL RANDOM(RAND)
2292:             V3(J)=A3*RAND*W*RND1
2293: C             V3(J)=A3*RAND*W
2294: C      THEN ON PAST VELOCITY WITH INERTIA WEIGHT W
2295:            V4(J)=W*V(I,J)
2296: C      FINALLY A SUM OF THEM
2297:            V(I,J)= V1(J)+V2(J)+V3(J)+V4(J)
2298:            ENDDO
2299:         ENDDO
2300: C      CHANGE X
2301:         DO I=1,N
2302:         DO J=1,M
2303:         RANDS=0.D00
2304: C      ------------------------------------------------------------------
2305:         IF(NSIGMA.EQ.1) THEN
2306:          CALL RANDOM(RAND) ! FOR CHAOTIC PERTURBATION
2307:          IF(DABS(RAND-.5D00).LT.SIGMA) RANDS=RAND-0.5D00
2308: C      SIGMA CONDITIONED RANDS INTRODUCES CHAOTIC ELEMENT IN TO LOCATION
2309: C      IN SOME CASES THIS PERTURBATION HAS WORKED VERY EFFECTIVELY WITH
2310: C      PARAMETER (N=100,NN=15,MX=100,NSTEP=9,ITRN=100000,NSIGMA=1,ITOP=2)
2311:         ENDIF
2312: C      ------------------------------------------------------------------
2313:         X(I,J)=X(I,J)+V(I,J)*(1.D00+RANDS)
2314:         ENDDO
2315:         ENDDO
2316:          DO I=1,N
2317:            IF(F(I).LT.FMIN) THEN
2318:            FMIN=F(I)
2319:            II=I
2320:            DO J=1,M
2321:            BST(J)=XX(II,J)
2322:            ENDDO
2323:            ENDIF
2324:            ENDDO
2325:         IF(LCOUNT.EQ.NPRN) THEN
2326:         LCOUNT=0
2327:         WRITE(*,*)'OPTIMAL SOLUTION UPTO THIS (FUNCTION CALLS=',NFCALL,')'
2328:         WRITE(*,*)'X = ',(BST(J),J=1,M),' MIN F = ',FMIN
2329: C       WRITE(*,*)'NO. OF FUNCTION CALLS = ',NFCALL
2330:         IF(DABS(FMIN-GGBEST).LT.EPSI) THEN
2331:            WRITE(*,*)'COMPUTATION OVER'
2332:            FMINIM=FMIN
2333:            RETURN
2334:            ELSE
2335:            GGBEST=FMIN
2336:            ENDIF
2337:         ENDIF
2338:         LCOUNT=LCOUNT+1
2339:   100 CONTINUE
2340:         WRITE(*,*)'COMPUTATION OVER:',FTIT
2341:         FMINIM=FMIN
2342:         RETURN
2343:         END
2344: C      ------------------------------------------------------------------
2345:         SUBROUTINE LSRCH(A,M,VI,NSTEP,FI)
```

```
2346:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2347:       COMMON /KFF/KF,NFCALL
2348:       COMMON /RNDM/IU,IV
2349:       INTEGER IU,IV
2350:       DIMENSION A(*),B(100),VI(*)
2351:       AMN=1.0D30
2352:       DO J=1,NSTEP
2353:          DO JJ=1,M
2354:          B(JJ)=A(JJ)+(J-(NSTEP/2)-1)*VI(JJ)
2355:          ENDDO
2356:       CALL FUNC(B,M,FI)
2357:         IF(FI.LT.AMN) THEN
2358:         AMN=FI
2359:         DO JJ=1,M
2360:         A(JJ)=B(JJ)
2361:         ENDDO
2362:         ENDIF
2363:       ENDDO
2364:       FI=AMN
2365:       RETURN
2366:       END
2367: C     ----------------------------------------------------------------------
2368: C     THIS SUBROUTINE IS NEEDED IF THE NEIGHBOURHOOD HAS RING TOPOLOGY
2369: C     EITHER PURE OR HYBRIDIZED
2370:        SUBROUTINE NEIGHBOR(I,N,J,K)
2371:        IF(I-1.GE.1 .AND. I.LT.N) THEN
2372:        J=I-1
2373:        K=I+1
2374:        ELSE
2375:        IF(I-1.LT.1) THEN
2376:        J=N-I+1
2377:        K=I+1
2378:        ENDIF
2379:        IF(I.EQ.N) THEN
2380:        J=I-1
2381:        K=1
2382:        ENDIF
2383:        ENDIF
2384:        RETURN
2385:        END
```