Global Optimization by Particle Swarm Method:A Fortran Program

Mishra, SK

North-Eastern Hill University, Shillong (India)

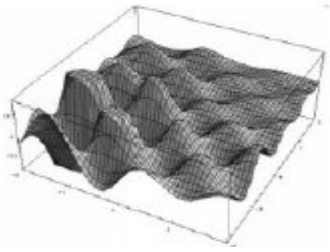
2 August 2006

Global Optimization by Particle Swarm Method: A Fortran Program

SK Mishra
Dept. of Economics
NEHU, Shillong (India)

Introduction: Optimization, that is finding minimum or maximum of a single-valued function $f(x)$ (often with constraints on x or some other function(s) of x, e.g. $h_i(x) \le \kappa_i : \ i = 1, 2, ..., k$) is required in many practical applications. If the problem is convex (it has only one minimum or maximum point), the local minimum (maximum) is also a global minimum (maximum). However, this is not always the case. One may have to optimize a function that has many local optima (and sometimes several global optima as well). For example, the function given below has many local optima.

Fig. I: Levy's Function # 5



Plot of Levy # 5 within the cube $[-2, 2]^2$

This function (within $x_i \in [-10,10]$; i = 1, 2) has about 760 local minima and one global minimum with function value

$$f(x^{**}) = -176.1375,$$

at the point $x^{**} = (-1.3068, -1.4248)$ as mentioned by Parsopoulos and Vrahatis (2002). The large number of local minimizers makes it difficult to locate the global minimum. Such surfaces are called multimodal.

There are many computer programs (in FORTRAN) that find the minimum (or maximum) when the problem is convex (with only peak or trough - that is when the local optimum is also a global optimum). Kuester and Mize (1973) has a good collection of such programs.

Global Optimization: Programs that work very well in optimizing convex functions very often perform poorly when the problem has multiple local minima or maxima. They are often caught or trapped in the local minima/maxima. Several methods have been developed to escape from being caught in such local optima.

(A). The Genetic Algorithm(s): Genetic algorithms are often successful in finding the global optimum of a multimodal function. Genetic algorithms are typically implemented as a computer simulation in which a population of abstract representations (called chromosomes) of candidate solutions (called individuals) to an optimization problem evolves toward better solutions. Solutions are often represented in binary as strings of 0s and 1s, but different encodings are also possible. The evolution starts from a population of completely random individuals and happens in generations. In each generation, the fitness of the whole population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), modified (mutated or recombined) to form a new population, which becomes current in the next iteration of the algorithm. Run over a large number (sequence) of generations or iterations, the algorithm

often finds the global (or near-global) optimum. There is a large body of literature on the Genetic algorithms. Also, see http://en.wikipedia.org/wiki/Genetic_algorithm .

A general-purpose optimization subroutine (FORTRAN-77) based on genetic algorithm (PIKAIA), downloadable from the anonymous ftp archive of the High Altitude Observatory, is available in the public domain. The subroutine is particularly useful (and robust) in treating multimodal optimization problems. The program may be downloaded from http://www.hao.ucar.edu/Public/models/pikaia/pikaia.html .

(B). The Classical Simulated Annealing Method: Simulated annealing (SA) is another effective method to find global optimum of a multimodal function. This method 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 (Kirkpatrick et al., 1983; Cerny, 1985). 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.

SIMANN, a global optimization algorithm using simulated annealing by William Goffe and others, is a very effective program written in FORTRAN-77. It may be downloaded from http://www.netlib.no/netlib/opt/simann.f absolutely free of cost.

(C). The Generalized Simulated Annealing Method: The method of simulated annealing was improved by Tsallis and Stariolo (1995) by the replacement of the Gaussian (Boltzmann-Gibbs) visiting distribution (used by SA) and the Cauchy-Lorentz visiting distribution (used by the Fast Simulated Annealing – FSA – method) by the Tsallis visiting distribution, making the earlier methods (SA and FSA) as special cases. This method is called the Generalized Simulated Annealing (GSA) method. Mundim (1996) provides GSA Fortran program to download absolutely free of cost from the website www.unb.br/iq/**kleber**/GSA/artigo2/node2.html .

(D). The Differential Evolution Method: 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, 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 (Storn and Price, 1995).

Although the DE does not perform well when there is an element of randomness in the objective function or the decision variables, it is perhaps the best among the algorithms that may be used to find out the global optimum of a non-stochastic continuous, real-valued, multi-modal functions (Mishra, 2006(f)). Its convergence is the fastest and it gives the most accurate results. A Fortran program of the differential evolution method may be downloaded freely from http://www1.webng.com/economics. It uses the most recent advances in the crossover scheme as recently suggested by Kenneth Price.

(E). The Particle Swarm Method of Global Optimization: This method is an instance of a successful application of the philosophy of *bounded rationality* and decentralized decision-making to solve the global optimization problems (Simon, 1982; Bauer, 2002; Fleischer, 2005). It is observed 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. Every member of the swarm searches for the best in its locality - learns from its own experience. Additionally, each member learns from the others, typically from the best performer among them. Even human beings show a tendency to learn from their own experience, their immediate neighbours and the ideal performers.

The Particle Swarm method of optimization mimics the said behaviour (see Wikipedia: http://en.**wikipedia**.org/wiki/Particle_**swarm**_optimization). 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 the position and velocity is done at each iteration as follows:

$$v_{i+1} = \omega v_i + c_1 r_1 (\hat{x}_i - x_i) + c_2 r_2 (\hat{x}_{gi} - x_i)$$
$$x_{i+1} = x_i + v_{i+1}$$

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.
- ω is the inertial constant. Good values are usually slightly less than 1.
- c_1 and c_2 are constants that say how much the particle is directed towards good positions. Good values are usually right around 1.
- r_1 and r_2 are random values in the range [0,1].
- $\hat{x}$ is the best that the particle has seen.
- $\hat{x}_g$ is the global best seen by the swarm. This can be replaced by $\hat{x}_L$, the local best, if neighborhoods are being used.

The Particle Swarm method (Eberhart and Kennedy, 1995) has many variants. The Repulsive Particle Swarm (RPS) method of optimization (see Wikipedia, http://en.wikipedia.org/wiki/RPSO), one of such variants, 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). Other variants use a dynamic scheme (Liang and Suganthan, 2005).

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

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

where,

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

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

Some Modifications in the RPS Method: 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.

It has been said that each particle learns from its 'chosen' inmates in the swarm. Now, 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.

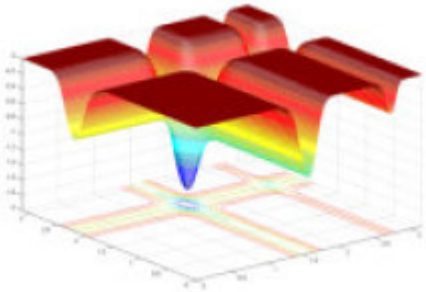
Let us visualize the possibilities of choosing (a predetermined number of) inmates randomly from among the members of the swarm. This is much closer to reality in the human world. When we are exposed to the mass media, we experience this. Alternatively, we may visualize our particles visiting a public 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.

A Computer program of (Modified) Repulsive Particle Swarm Method: Here we append a program of the (modified) RPS method. This program lists the Fortran codes of (over) 90 benchmark functions of different dimensions, complexities and difficulty levels, and the RPS method that minimizes them. A directly usable listing of the program can be obtained from http://www1.webng.com/economics/rps.txt.

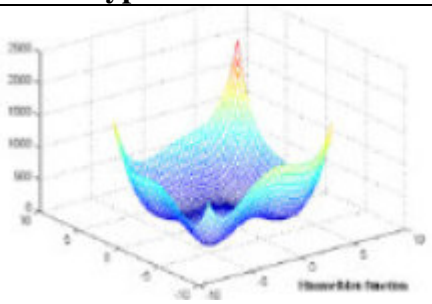
An Illustration: Let us choose some functions (listed in the program) rather arbitrarily. The first is a simple quadratic function (in $m = 2$ variables, KF=86). Here KF is the code of the function as listed in the program. The Quadratic function has only one local=global optimum. The second is the notorious Rosenbrock function (unimodal) in $m \geq 2$ variables, KF=59. The third is Levy's function # 5 ($m = 2$, KF= 43). This function is multimodal. The fourth is Levy's function # 8 ($m = 3$, kf=44). It is a multimodal function. The fifth function is of Michalewicz (KF=47). It may have several variables ($m \geq 2$). It is a

multimodal function. The next two are Himmelblau (KF=76; 4 minima, all global) and Trefethen (KF=90; $m = 2$) functions. The Trefethen function was posed by LN Trefethen of the Oxford University as a challenge (that appeared in the SIAM News, Jan-Feb. 2002). The eighth is the Griewank function (KF=40; $m \geq 2$, multimodal). Graphical presentations of some of these functions are presented in Fig.-II.

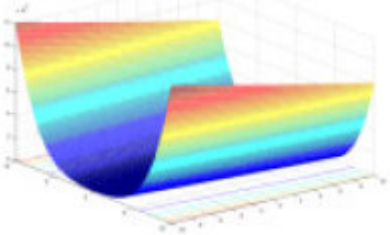
Fig.-II. Graphical Presentation of Some Typical Functions



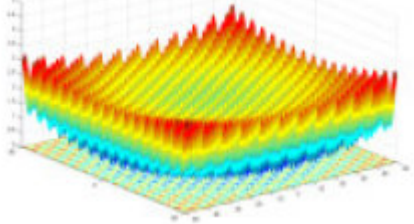
Michalewicz function



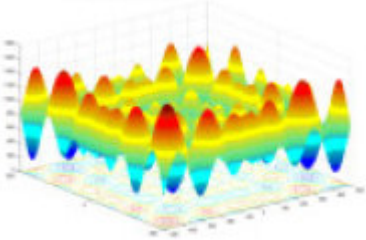
Himmelblau function



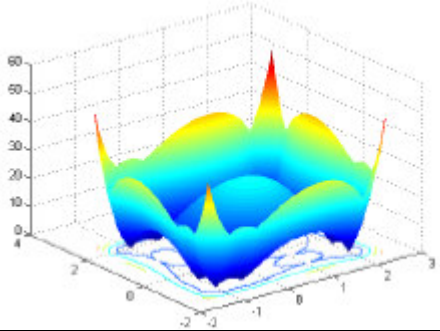
Rosenbrock function



Griewank function



Schwefel function



Quintic function

For most of the listed functions, the program has found the global or near-global optimum. In particular, the results of the eight functions chosen (for illustration) are given below.

Table-1: Minimization of some Benchmark functions for Illustration

Function	Quadratic	Rosenbrock	Levy5	Levy8	Michalewicz	Himmelblau	Trefethen	Griewank
Min F	-3873.72	2.51e-05	-176.1376	5.28e-12	-9.66014	1.89e-11	-3.3069	7.54e-09
Dim=m	2	10	2	3	10	2	2	10

When the program is run, it asks for choosing KF and specifying value of m (the number of decision variables). If KF=90 (this function is in two variables, x_1 and x_2 ; that

is m= decision variables = 2) the Trefethen function is optimized and so on. The program also asks for a seed to generate random numbers. A four-digit natural number (preferably odd) may be fed as a seed. The program gives the following type of output.

```
 KF=89 WAVY-2 FUNCTIONS: M-VARIABLES M=?
 KF=90 TREFETHEN FUNCTION: 2-VARIABLES M=2
 KF=91 ALPINE FUNCTION: M-VARIABLES M=?
 ------------------------------------------------------
 FUNCTION CODE [KF] AND NO. OF VARIABLES [M] ?
90 2
 4-DIGITS SEED FOR RANDOM NUMBER GENERATION
5511
 OPTIMAL SOLUTION UPTO THIS (FUNCTION CALLS= 751500)
 X =  -0.0243963447  0.210615718 MIN F =  -3.30686863
 OPTIMAL SOLUTION UPTO THIS (FUNCTION CALLS= 1501500)
 X =  -0.0192410078 -0.49615099 MIN F =  -3.30686865
 OPTIMAL SOLUTION UPTO THIS (FUNCTION CALLS= 2251500)
 X =  -0.0243980407  0.210608484 MIN F =  -3.30686865
 ------------------------------------------------------
 FINAL X =  -0.0243980407  0.210608484 FINAL MIN F =  -3.30686865
 COMPUTATION OVER:FOR
 KF=90 TREFETHEN FUNCTION: 2-VARIABLES M=2
 NO. OF VARIABLES= 2  END.
```

This program has also been used successfully for optimizing very complicated functions. For example, it has been used to fit Gielis superformula (Gielis, 2003; Mishra, 2006 (a)) modified by different trigonometric functions. The Gielis superformula is:

$$r(\theta) = f(\theta).\left[\left|\frac{1}{a}\cos\left(\frac{m}{4}\theta\right)\right|^{n_2} + \left|\frac{1}{b}\sin\left(\frac{m}{4}\theta\right)\right|^{n_3}\right]^{(-n_1^{-1})} = f(\theta).g(\theta)$$

The modifier functions are, for example,

Model-1 modifier : $f_1(\theta) = r = [n_4(3\cos(\theta) - \cos(3\theta))^2 + n_5(3\sin(\theta) - \sin(3\theta))^2]^{0.5}$

Model-2 modifier: $f_2(\theta) = r = n_4 + n_5\cos(\theta)$; Model-3 modifier: $f_2(\theta) = r = n_4 + n_5\cos(\theta)$

Model-4 modifier: $f_3(\theta) = r = n_4 - n_5\cos(\theta) + abs(\cos(\theta))^3$

In all modifier functions, n_4 and n_5 are parameters and $0 \le \theta \le 2\pi$.

The graphical presentations of Gielis curves are given in Fig.III(A) below. The red points are simulated by true parameters whereas blue points are obtained from the estimated parameters. It has also been used for extrapolating the points beyond the sample points used for fitting the curve (see Fig. III(B)). Extrapolation may have several applications, such as shape recovery (see Bhabhrawala and Krovi, 2005).

The RPS method has also been used to fit the generalized Gielis curves. The generalization of Gielis' superformula, $g(\theta)$, as suggested by Chacón (2004), is:

$$r(\theta) = f(\theta).\left[\left|\frac{1}{a}\cos\left(\phi(\theta)\right)\right|^{n_2} + \left|\frac{1}{b}\sin\left(\psi(\theta)\right)\right|^{n_3}\right]^{(-n_1^{-1})} = f(\theta).\gamma(\theta)$$

where, $\phi(\theta) = am\left[\frac{K(\mu)}{2\pi}(m\theta + \varphi), \mu\right]$; $\psi(\theta) = am\left[\frac{K(\mu)}{2\pi}(m\theta + \varphi'), \mu\right]$

In the expressions above, $am[u;\ \mu]$ is the Jacobian elliptic function (JEF) of the parameter μ, $K(\mu)$ is the complete elliptic integral of the first kind, and (φ, φ') are additional parameters (Whittaker and Watson, 1996; Abramowitz and Stegun, 1964).

The parameter m signifies rotational symmetry as in the Gielis superformula. The function $\gamma(\theta)$ degenerates into $g(\theta)$ for $\mu = \varphi = \varphi' = 0$ and in this sense the Chacón-Gielis superformula is a generalization of the Gielis superformula. Fitting Chacón-Gielis superformula (by Least Squares or by minimization of any other norm such as Minkowski's norm, etc) to data involves optimization of a highly complicated multimodal function, partly due to appearance of elliptic functions and complete integrals). This has been done quite successfully by the RPS method (see Mishra, 2006(b)).

Fig.-III(A). Fitting of Modified Gielis Functions by the Repulsive Particle Swarm Method

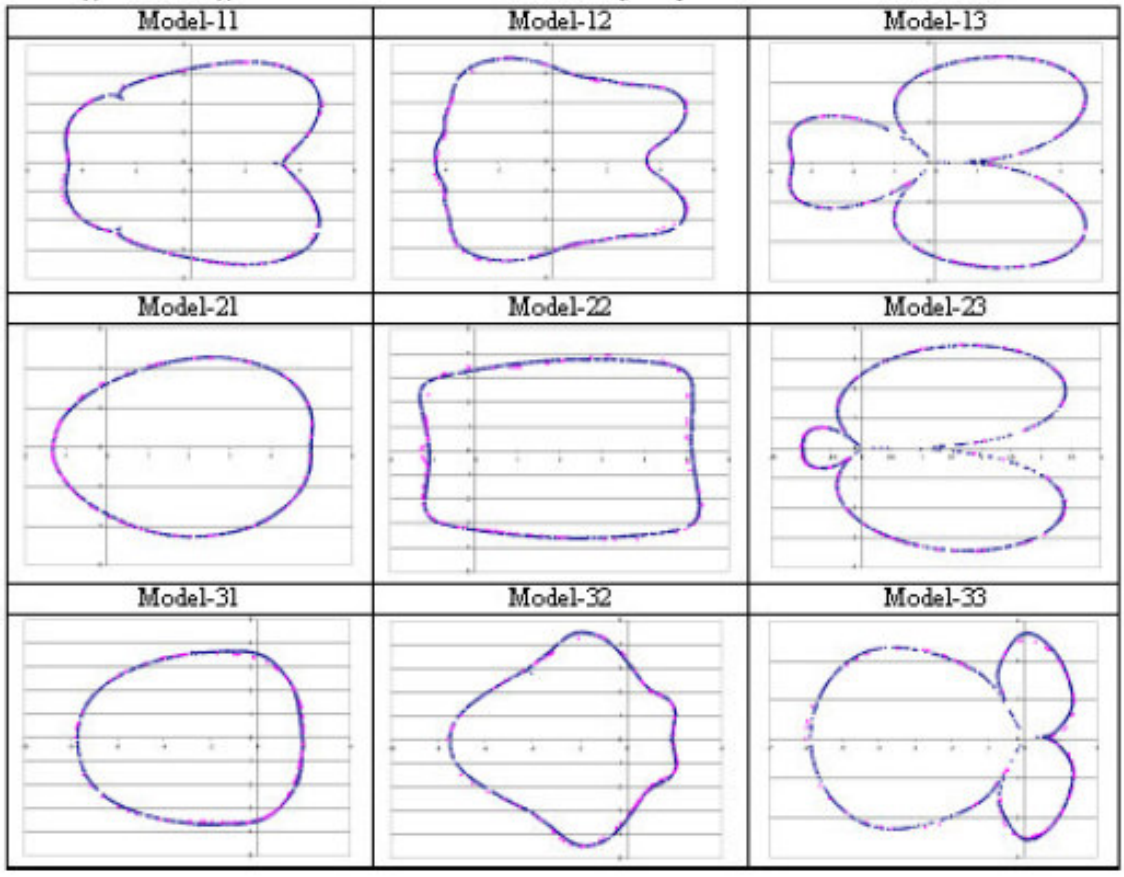
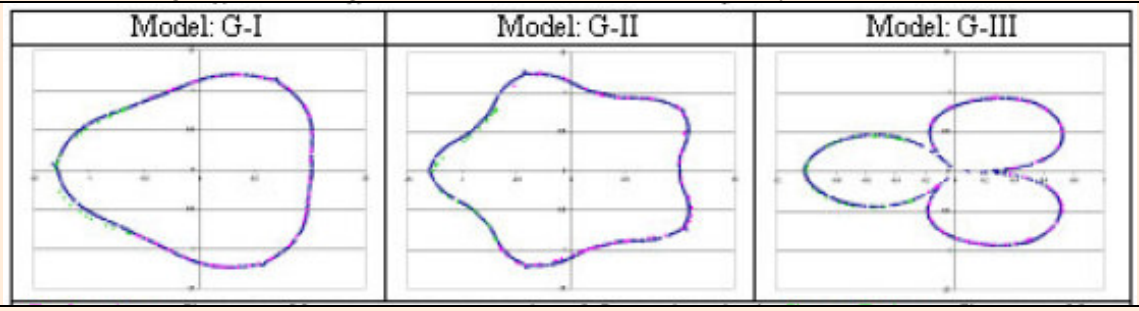


Fig.-III(B). Fitting of the Modified Gielis Functions by the RPS Method



Red points – generated by the true parameters (used for estimation); Green points – generated by the true parameters (not used for estimation); Blue points – estimated by the RPS using red points.

Bibliography

- Abramowitz, M. and Stegun, I.A.: *Handbook of Mathematical Functions*, Dover Publications, New York, 1964.
- Bauer, J.M.: "Harnessing the Swarm: Communication Policy in an Era of Ubiquitous Networks and Disruptive Technologies", *Communications and Strategies*, 45, 2002.
- Bhabhrawala, T. and Krovi, V.: "Shape Recovery from Medical Image Data using Extended Superquadrics", *ASME 2005 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Long Beach, California USA, September 24-28, 2005.
- 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.
- Chacón, R.: "A Mathematical Description of Natural Shapes in our Nonlinear World", Working paper, arXive:nlin. AO/0405057 v1 25 May, 2004.
- 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.
- Gielis, J.: "A Generic Geometric Transformation that unifies a Wide Range of Natural and Abstract Shapes", *American Journal of Botany,* 90(3): pp. 333–338, 2003.
- Glover F.," Future paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research*, 5:533-549, 1986.
- Holland, J.: *Adaptation in Natural and Artificial Systems,* Univ. of Michigan Press, Ann Arbor, 1975.
- Jung, B.S. and Karney, B.W.: "Benchmark Tests of Evolutionary Computational Algorithms", Environmental Informatics Archives (International Society for Environmental Information Sciences), 2, pp. 731-742, 2004.
- 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.
- Kuester, J.L. and Mize, J.H.: *Optimization Techniques with Fortran*, McGraw-Hill Book Co. New York, 1973.
- 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.
- Liang, J.J. and Suganthan, P.N. "Dynamic Multi-Swarm Particle Swarm Optimizer", *International Swarm Intelligence Symposium*, IEEE # 0-7803-8916-6/05/$20.00. pp. 124-129, 2005.
- 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).
- Mishra, S.K.: "Global Optimization by Differential Evolution and Particle Swarm Methods: Evaluation on Some Benchmark Functions". SSRN: http://ssrn.com/abstract=933827 , 2006 (f).
- Mundim, K.C.: "Generalized Simulated Annealing", (provides GSA Fortran Program to download) available at www.unb.br/iq/**kleber**/GSA/artigo2/node2.html , 1996.
- Mundim, K.C. and Tsallis, C.: "Geometry Optimization and Conformational Analysis through Generalized Simulated Annealing*",Int. Journal of Quantum Chemistry*, 58, pp. 373-381, 1996.
- 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.
- Storn, R. and Price, K: "Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces" : Technical Report, International Computer Science Institute, Berkley, 1995.
- 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.
- 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.
- Whittaker, E.T. and Watson, G.N.: *A Course of Modern Analysis*, Cambridge Univ. Press, 1996.
- Yao, X. and Liu, Y.: "Fast Evolutionary Programming", in Fogel, LJ, Angeline, PJ and Bäck, T (eds) *Proc. 5th Annual Conf. on Evolutionary programming*, pp. 451-460, MIT Press, Mass, 1996.

```
1: C     PROGRAM TO FIND GLOBAL MINIMUM BY REPULSIVE PARTICLE SWARM METHOD
2: C     WRITTEN BY SK MISHRA, DEPT. OF ECONOMICS, NEHU, SHILLONG (INDIA)
3: C     ------------------------------------------------------------------
4:       PARAMETER (N=100,NN=40,MX=100,NSTEP=15,ITRN=10000,NSIGMA=1,ITOP=3)
5:       PARAMETER (NPRN=500) ! DISPLAYS RESULTS AT EVERY 100 TH ITERATION
6: C     PARAMETER(N=50,NN=25,MX=100,NSTEP=9,ITRN=10000,NSIGMA=1,ITOP=3)
7: C     PARAMETER (N=100,NN=15,MX=100,NSTEP=9,ITRN=10000,NSIGMA=1,ITOP=3)
8: C     IN CERTAIN CASES THE ONE OR THE OTHER SPECIFICATION WORKS BETTER
9: C     DIFFERENT SPECIFICATIONS OF PARAMETERS MAY SUIT DIFFERENT TYPES
10: C     OF FUNCTIONS OR DIMENSIONS - ONE HAS TO DO SOME TRIAL AND ERROR
11: C     ------------------------------------------------------------------
12: C     N = POPULATION SIZE. IN MOST OF THE CASES N=30 IS OK. ITS VALUE
13: C     MAY BE INCREASED TO 50 OR 100 TOO. THE PARAMETER NN IS THE SIZE OF
14: C     RANDOMLY CHOSEN NEIGHBOURS. 15 TO 25 (BUT SUFFICIENTLY LESS THAN
15: C     N) IS A GOOD CHOICE. MX IS THE MAXIMAL SIZE OF DECISION VARIABLES.
16: C     IN F(X1, X2,...,XM) M SHOULD BE LESS THAN OR EQUAL TO MX. ITRN IS
17: C     THE NO. OF ITERATIONS. IT MAY DEPEND ON THE PROBLEM. 200(AT LEAST)
18: C     TO 500 ITERATIONS MAY BE GOOD ENOUGH. BUT FOR FUNCTIONS LIKE
19: C     ROSENBROCKOR GRIEWANK OF LARGE SIZE (SAY M=30) IT IS NEEDED THAT
20: C     ITRN IS LARGE, SAY 5000 OR EVEN 10000.
21: C     SIGMA INTRODUCES PERTURBATION & HELPS THE SEARCH JUMP OUT OF LOCAL
22: C     OPTIMA. FOR EXAMPLE : RASTRIGIN FUNCTION OF DMENSION 3O OR LARGER
23: C     NSTEP DOES LOCAL SEARCH BY TUNNELLING AND WORKS WELL BETWEEN 5 AND
24: C     15, WHICH IS MUCH ON THE HIGHER SIDE.
25: C     ITOP <=1 (RING); ITOP=2 (RING AND RANDOM); ITOP=>3 (RANDOM)
26: C     NSIGMA=0 (NO CHAOTIC PERTURBATION);NSIGMA=1 (CHAOTIC PERTURBATION)
27: C     NOTE THAT NSIGMA=1 NEED NOT ALWAYS WORK BETTER (OR WORSE)
28: C     SUBROUTINE FUNC( ) DEFINES OR CALLS THE FUNCTION TO BE OPTIMIZED.
29:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
30:       COMMON /RNDM/IU,IV
31:       COMMON /KFF/KF,NFCALL,FTIT
32:       INTEGER IU,IV
33:       CHARACTER *70 FTIT
34:       DIMENSION X(N,MX),V(N,MX),A(MX),VI(MX),TIT(50)
35:       DIMENSION XX(N,MX),F(N),V1(MX),V2(MX),V3(MX),V4(MX),BST(MX)
36: C     A1 A2 AND A3 ARE CONSTANTS AND W IS THE INERTIA WEIGHT.
37: C     OCCASIONALLY, TINKERING WITH THESE VALUES, ESPECIALLY A3, MAY BE
38: C     NEEDED.
39:       DATA A1,A2,A3,W,SIGMA /.5D00,.5D00,.0005D00,.5D00,1.D-03/
40:       EPSILON=1.D-08 ! ACCURACY NEEDED FOR TERMINATON
41: C     ---------------------CHOOSING THE TEST FUNCTION ------------------'
42:       CALL FSELECT(KF,M,FTIT)
43: C     ------------------------------------------------------------------
44:       FFMIN=1.D30
45:       LCOUNT=0
46:       NFCALL=0
47:       WRITE(*,*)'4-DIGITS SEED FOR RANDOM NUMBER GENERATION'
48:       READ(*,*) IU
49:       DATA FMIN /1.0E30/
50: C     GENERATE N-SIZE POPULATION OF M-TUPLE PARAMETERS X(I,J) RANDOMLY
51:       DO I=1,N
52:         DO J=1,M
53:         CALL RANDOM(RAND)
54:          X(I,J)=(RAND-0.5D00)*10
55: C     WE GENERATE RANDOM(-5,5). HERE MULTIPLIER IS 10. TINKERING IN SOME
56: C     CASES MAY BE NEEDED
57:          ENDDO
58:         F(I)=1.0D30
59:       ENDDO
60: C     INITIALISE VELOCITIES V(I) FOR EACH INDIVIDUAL IN THE POPULATION
61:       DO I=1,N
62:       DO J=1,M
63:       CALL RANDOM(RAND)
64:        V(I,J)=(RAND-0.5D+00)
65: C       V(I,J)=RAND
66:       ENDDO
67:       ENDDO
```

```
      DO 100 ITER=1,ITRN
C     LET EACH INDIVIDUAL SEARCH FOR THE BEST IN ITS NEIGHBOURHOOD
        DO I=1,N
          DO J=1,M
          A(J)=X(I,J)
          VI(J)=V(I,J)
          ENDDO
          CALL LSRCH(A,M,VI,NSTEP,FI)
          IF(FI.LT.F(I)) THEN
           F(I)=FI
           DO IN=1,M
           BST(IN)=A(IN)
           ENDDO
C     F(I) CONTAINS THE LOCAL BEST VALUE OF FUNCTION FOR ITH INDIVIDUAL
C     XX(I,J) IS THE M-TUPLE VALUE OF X ASSOCIATED WITH LOCAL BEST F(I)
             DO J=1,M
             XX(I,J)=A(J)
             ENDDO
             ENDIF
        ENDDO
C      NOW LET EVERY INDIVIDUAL RANDOMLY COSULT NN(<<N) COLLEAGUES AND
C      FIND THE BEST AMONG THEM
      DO I=1,N
C     ------------------------------------------------------------------
      IF(ITOP.GE.3) THEN
C     RANDOM TOPOLOGY ***************************************
C     CHOOSE NN COLLEAGUES RANDOMLY AND FIND THE BEST AMONG THEM
          BEST=1.0D30
           DO II=1,NN
                 CALL RANDOM(RAND)
               NF=INT(RAND*N)+1
                IF(BEST.GT.F(NF)) THEN
                 BEST=F(NF)
                NFBEST=NF
                 ENDIF
           ENDDO
      ENDIF
C-----------------------------------------------------------------------
      IF(ITOP.EQ.2) THEN
C     RING + RANDOM TOPOLOGY ********************************************
C     REQUIRES THAT THE SUBROUTINE NEIGHBOR IS TURNED ALIVE
       BEST=1.0D30
          CALL NEIGHBOR(I,N,I1,I3)
          DO II=1,NN
                IF(II.EQ.1) NF=I1
                 IF(II.EQ.2) NF=I
                  IF(II.EQ.3) NF=I3
                      IF(II.GT.3) THEN
                     CALL RANDOM(RAND)
                      NF=INT(RAND*N)+1
                     ENDIF
                  IF(BEST.GT.F(NF)) THEN
                  BEST=F(NF)
                  NFBEST=NF
                ENDIF
            ENDDO
       ENDIF
C-----------------------------------------------------------------------
      IF(ITOP.LE.1) THEN
C     RING TOPOLOGY *****************************************************
C     REQUIRES THAT THE SUBROUTINE NEIGHBOR IS TURNED ALIVE
        BEST=1.0D30
          CALL NEIGHBOR(I,N,I1,I3)
             DO II=1,3
             IF (II.NE.I) THEN
            IF(II.EQ.1) NF=I1
             IF(II.EQ.3) NF=I3
```

```
                  IF(BEST.GT.F(NF)) THEN
                   BEST=F(NF)
                   NFBEST=NF
                  ENDIF
                  ENDIF
              ENDDO
         ENDIF
C-------------------------------------------------------------------------
C      IN THE LIGHT OF HIS OWN AND HIS BEST COLLEAGUES EXPERIENCE, THE
C      INDIVIDUAL I WILL MODIFY HIS MOVE AS PER THE FOLLOWING CRITERION
C      FIRST, ADJUSTMENT BASED ON ONES OWN EXPERIENCE
C      AND OWN BEST EXPERIENCE IN THE PAST (XX(I))
           DO J=1,M
           CALL RANDOM(RAND)
           V1(J)=A1*RAND*(XX(I,J)-X(I,J))
C      THEN BASED ON THE OTHER COLLEAGUES BEST EXPERIENCE WITH WEIGHT W
C      HERE W IS CALLED AN INERTIA WEIGHT 0.01< W < 0.7
C      A2 IS THE CONSTANT NEAR BUT LESS THAN UNITY
           CALL RANDOM(RAND)
           V2(J)=V(I,J)
           IF(F(NFBEST).LT.F(I)) THEN
           V2(J)=A2*W*RAND*(XX(NFBEST,J)-X(I,J))
           ENDIF
C      THEN SOME RANDOMNESS AND A CONSTANT A3 CLOSE TO BUT LESS THAN UNITY
           CALL RANDOM(RAND)
           RND1=RAND
           CALL RANDOM(RAND)
            V3(J)=A3*RAND*W*RND1
C             V3(J)=A3*RAND*W
C      THEN ON PAST VELOCITY WITH INERTIA WEIGHT W
           V4(J)=W*V(I,J)
C      FINALLY A SUM OF THEM
           V(I,J)= V1(J)+V2(J)+V3(J)+V4(J)
           ENDDO
       ENDDO
C      CHANGE X
       DO I=1,N
       DO J=1,M
       RANDS=0.D00
C      ------------------------------------------------------------------
       IF(NSIGMA.EQ.1) THEN
        CALL RANDOM(RAND) ! FOR CHAOTIC PERTURBATION
        IF(DABS(RAND-.5D00).LT.SIGMA) RANDS=RAND-0.5D00
C      SIGMA CONDITIONED RANDS INTRODUCES CHAOTIC ELEMENT IN TO LOCATION
C      IN SOME CASES THIS PERTURBATION HAS WORKED VERY EFFECTIVELY WITH
C      PARAMETER (N=100,NN=15,MX=100,NSTEP=9,ITRN=100000,NSIGMA=1,ITOP=2)
       ENDIF
C      ------------------------------------------------------------------
       X(I,J)=X(I,J)+V(I,J)*(1.D00+RANDS)
       ENDDO
       ENDDO
        DO I=1,N
          IF(F(I).LT.FMIN) THEN
          FMIN=F(I)
          II=I
          DO J=1,M
          BST(J)=XX(II,J)
          ENDDO
          ENDIF
          ENDDO
       IF(LCOUNT.EQ.NPRN) THEN
       LCOUNT=0
       WRITE(*,*)'OPTIMAL SOLUTION UPTO THIS (FUNCTION CALLS=',NFCALL,')'
       WRITE(*,*)'X = ',(BST(J),J=1,M),' MIN F = ',FMIN
C       WRITE(*,*)'NO. OF FUNCTION CALLS = ',NFCALL
       IF(DABS(FFMIN-FMIN).LT.EPSILON) GOTO 999
       FFMIN=FMIN
```

```
202:       ENDIF
203:       LCOUNT=LCOUNT+1
204:   100 CONTINUE
205:   999 WRITE(*,*)'-------------------------------------------------------'
206:       WRITE(*,*)'FINAL X = ',(BST(J),J=1,M),' FINAL MIN F = ',FMIN
207:       WRITE(*,*)'COMPUTATION OVER:FOR ',FTIT
208:       WRITE(*,*)'NO. OF VARIABLES=',M,'  END.'
209:       END
210: C     ---------------------------------------------------------------
211:       SUBROUTINE LSRCH(A,M,VI,NSTEP,FI)
212:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
213:       CHARACTER *70 FTIT
214:       COMMON /KFF/KF,NFCALL,FTIT
215:       COMMON /RNDM/IU,IV
216:       INTEGER IU,IV
217:       DIMENSION A(*),B(100),VI(*)
218:       AMN=1.0D30
219:       DO J=1,NSTEP
220:          DO JJ=1,M
221:          B(JJ)=A(JJ)+(J-(NSTEP/2)-1)*VI(JJ)
222:          ENDDO
223:       CALL FUNC(B,M,FI)
224:         IF(FI.LT.AMN) THEN
225:         AMN=FI
226:         DO JJ=1,M
227:         A(JJ)=B(JJ)
228:         ENDDO
229:         ENDIF
230:       ENDDO
231:       FI=AMN
232:       RETURN
233:       END
234: C     ---------------------------------------------------------------------
235: C     THIS SUBROUTINE IS NEEDED IF THE NEIGHBOURHOOD HAS RING TOPOLOGY
236: C     EITHER PURE OR HYBRIDIZED
237:        SUBROUTINE NEIGHBOR(I,N,J,K)
238:        IF(I-1.GE.1 .AND. I.LT.N) THEN
239:        J=I-1
240:        K=I+1
241:        ELSE
242:        IF(I-1.LT.1) THEN
243:        J=N-I+1
244:        K=I+1
245:        ENDIF
246:        IF(I.EQ.N) THEN
247:        J=I-1
248:        K=1
249:        ENDIF
250:        ENDIF
251:        RETURN
252:        END
253: C     ---------------------------------------------------------------------
254:       SUBROUTINE RANDOM(RAND1)
255:        DOUBLE PRECISION  RAND1
256:        COMMON /RNDM/IU,IV
257:       INTEGER IU,IV
258:        RAND=REAL(RAND1)
259:        IV=IU*65539
260:        IF(IV.LT.0) THEN
261:        IV=IV+2147483647+1
262:        ENDIF
263:        RAND=IV
264:        IU=IV
265:        RAND=RAND*0.4656613E-09
266:        RAND1=DBLE(RAND)
267:        RETURN
268:        END
```

```
C     ------------------------------------------------------------------
      SUBROUTINE FSELECT(KF,M,FTIT)
C     THE PROGRAM REQUIRES INPUTS FROM THE USER ON THE FOLLOWING ------
C     (1) FUNCTION CODE (KF), (2) NO. OF VARIABLES IN THE FUNCTION (M);
      CHARACTER *70 TIT(100),FTIT
      WRITE(*,*)'-----------------------------------------------------'
      DATA TIT(1)/'KF=1 NEW FUNCTION(N#1) 2-VARIABLES M=2'/
      DATA TIT(2)/'KF=2 NEW FUNCTION(N#2) 2-VARIABLES M=2'/
      DATA TIT(3)/'KF=3 NEW FUNCTION(N#3) 2-VARIABLES M=2'/
      DATA TIT(4)/'KF=4 NEW FUNCTION(N#4) 2-VARIABLES M=2'/
      DATA TIT(5)/'KF=5 NEW QUINTIC FUNCTION M-VARIABLES M=?'/
      DATA TIT(6)/'KF=6 NEW NEEDLE-EYE FUNCTION (N#6) M-VARIABLES M=?'/
      DATA TIT(7)/'KF=7 NEW ZERO-SUM FUNCTION (N#7) M-VARIABLES M=?'/
      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 WOOD FUNCTION : 4-VARIABLES M=4'/
      DATA TIT(35)/'KF=35 FENTON-EASON FUNCTION : 2-VARIABLES M=2'/
      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'/
      DATA TIT(71)/'KF=71 GLANKWAHMDEE FUNCTION : 5-VARIABLES M=5'/
      DATA TIT(72)/'KF=72 FLETCHER-POWELL FUNCTION : M-VARIABLES M=?'/
      DATA TIT(73)/'KF=73 POWELL FUNCTION: M-VARIABLES M (MULT OF 4)=?'/
      DATA TIT(74)/'KF=74 HARTMANN FUNCTION: 3-VARIABLES M=3'/
      DATA TIT(75)/'KF=75 COLVILLE FUNCTION: 4-VARIABLES M=4'/
      DATA TIT(76)/'KF=76 HIMMELBLAU FUNCTION: 2-VARIABLES M=2'/
      DATA TIT(77)/'KF=77 BEALE FUNCTION: 2-VARIABLES M=2'/
      DATA TIT(78)/'KF=78 BOOTH FUNCTION: 2-VARIABLES M=2'/
      DATA TIT(79)/'KF=79 HUMP FUNCTION: 2-VARIABLES M=2'/
      DATA TIT(80)/'KF=80 MATYAS FUNCTION: 2-VARIABLES M=2'/
      DATA TIT(81)/'KF=81 MISHRA_1 FUNCTION: M-VARIABLES M=?'/
      DATA TIT(82)/'KF=82 MISHRA_2 FUNCTION: M-VARIABLES M=?'/
      DATA TIT(83)/'KF=83 ZAKHAROV FUNCTION: M-VARIABLES M=?'/
      DATA TIT(84)/'KF=84 MULTIMOD FUNCTION: 2-VARIABLES M=2'/
      DATA TIT(85)/'KF=85 NONLINEAR FUNCTION: M-VARIABLES M=?'/
      DATA TIT(86)/'KF=86 QUADRATIC FUNCTION: 2-VARIABLES M=2'/
      DATA TIT(87)/'KF=87 TRIGON FUNCTION: M-VARIABLES M=?'/
      DATA TIT(88)/'KF=88 WAVY-1 FUNCTIONS: M-VARIABLES M=?'/
      DATA TIT(89)/'KF=89 WAVY-2 FUNCTIONS: M-VARIABLES M=?'/
      DATA TIT(90)/'KF=90 TREFETHEN FUNCTION: 2-VARIABLES M=2'/
      DATA TIT(91)/'KF=91 ALPINE FUNCTION: M-VARIABLES M=?'/
C     ------------------------------------------------------------------
      DO I=1,91
      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,FTIT
      INTEGER IU,IV
      DIMENSION X(*)
      CHARACTER *70 FTIT
      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
```

```
403: C     FUNCTION #2 MIN = -0.199409 APPROX AT (-9.94112, -10) APPROX
404:       F=0.D00
405:       DO I=1,M
406:       IF(DABS(X(I)).GT.10.D00) THEN
407:       CALL RANDOM(RAND)
408:       X(I)=(RAND-0.5D00)*20
409:       ENDIF
410:       ENDDO
411:       F=DABS(DSIN(DSQRT(DABS(X(1)**2+X(2)))))**0.5 +0.01*X(1)+.01*X(2)
412:       RETURN
413:       ENDIF
414: C     ------------------------------------------------------------------
415:       IF(KF.EQ.3) THEN
416: C     FUNCTION #3 MIN = -1.01983 APPROX AT (-1.98682, -10.00000) APPROX
417:       F=0.D00
418:       DO I=1,M
419:       IF(DABS(X(I)).GT.10.D00) THEN
420:       CALL RANDOM(RAND)
421:       X(I)=(RAND-0.5D00)*20
422:       ENDIF
423:       ENDDO
424:       F1=DSIN(( DCOS(X(1))+DCOS(X(2)) )**2)**2
425:       F2=DCOS(( DSIN(X(1))+DSIN(X(2)) )**2)**2
426:       F=(F1+F2+X(1))**2 ! IS MULTIMODAL
427:       F=F+ 0.01*X(1)+0.1*X(2) ! MAKES UNIMODAL
428:       RETURN
429:       ENDIF
430: C     ------------------------------------------------------------------
431:       IF(KF.EQ.4) THEN
432: C     FUNCTION  #4 MIN = -2.28395 APPROX AT (2.88631, 1.82326) APPROX
433:       F=0.D00
434:       DO I=1,M
435:       IF(DABS(X(I)).GT.10.D00) THEN
436:       CALL RANDOM(RAND)
437:       X(I)=(RAND-0.5D00)*20
438:       ENDIF
439:       ENDDO
440:       F1=DSIN((DCOS(X(1))+DCOS(X(2)))**2)**2
441:       F2=DCOS((DSIN(X(1))+DSIN(X(2)))**2)**2
442:       F3=-DLOG((F1-F2+X(1))**2 )
443:       F=F3+0.1D00*(X(1)-1.D00)**2+0.1D00*(X(2)-1.D00)**2
444:       RETURN
445:       ENDIF
446: C     ------------------------------------------------------------------
447:       IF(KF.EQ.5) THEN
448: C     QUINTIC FUNCTION:GLOBAL MINIMA,EXTREMELY DIFFICULT TO OPTIMIZE
449: C     MIN VALUE = 0 AT PERMUTATION OF (2, 2,..., 2, -1, -1, ..., -1,
450: C     -0.402627941) GIVES MIN F = 0.
451:       F=0.D00
452:       DO I=1,M
453:       IF(DABS(X(I)).GT.10.D00) THEN
454:       CALL RANDOM(RAND)
455:       X(I)=(RAND-0.5D00)*20
456:       ENDIF
457:       ENDDO
458:       CALL QUINTIC(M,F,X)
459:       RETURN
460:       ENDIF
461: C     ------------------------------------------------------------------
462:       IF(KF.EQ.6) THEN
463: C     NEEDLE-EYE FUNCTION M=>1;
464: C     MIN = 1 IF ALL ABS(X) ARE SMALLER THAN THE EYE
465: C     SMALLER THE VALUE OF ZZ, MORE DIFFICULT TO ENTER THE EYE
466: C     LARGER THE VALUE OF M, MORE DIFFICULT TO FIND THE OPTIMUM
467:       F=0.D00
468:       EYE=0.000001D00
469:       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)
```

```
604:       X(I)=(RAND-0.5D00)*2*M
605:       ENDIF
606:       ENDDO
607:       DO K=1,M
608:       SUM=0.D00
609:       DO I=1,M
610:       SUM=SUM+(I**K+BETA)*((X(I)/I)**K-1.D00)
611:       ENDDO
612:       F=F+SUM**2
613:       ENDDO
614:       RETURN
615:       ENDIF
616: C     ------------------------------------------------------------------
617:       IF(KF.EQ.13) THEN
618: C     PERM FUNCTION #2 MIN = 0 AT (1/1, 1/2, 1/3, 1/4,..., 1/M)
619: C     BETA => 0. CHANGE IF NEEDED. SMALLER BETA RAISES DIFFICULY
620: C     FOR BETA=0, EVERY PERMUTED SOLUTION IS A GLOBAL MINIMUM
621:       BETA=10.D00
622:       DO I=1,M
623:       IF(DABS(X(I)).GT.1.D00) THEN
624:       CALL RANDOM(RAND)
625:       X(I)=(RAND-.5D00)*2
626:       ENDIF
627:       SGN=X(I)/DABS(X(I))
628:       ENDDO
629:       F=0.D00
630:       DO K=1,M
631:       SUM=0.D00
632:       DO I=1,M
633:       SUM=SUM+(I+BETA)*(X(I)**K-(1.D00/I)**K)
634:       ENDDO
635:       F=F+SUM**2
636:       ENDDO
637:       RETURN
638:       ENDIF
639: C     ------------------------------------------------------------------
640:       IF(KF.EQ.14) THEN
641: C     POWER SUM FUNCTION; MIN = 0 AT PERM(1,2,2,3) FOR B=(8,18,44,114)
642: C     0 =< X <=4
643:       F=0.D00
644:       DO I=1,M
645: C     ANY PERMUTATION OF (1,2,2,3) WILL GIVE MIN = ZERO
646:       IF(X(I).LT.0.D00 .OR. X(I).GT.4.D00) THEN
647:       CALL RANDOM(RAND)
648:       X(I)=RAND*4
649:       ENDIF
650:       ENDDO
651:       DO K=1,M
652:       SUM=0.D00
653:       DO I=1,M
654:       SUM=SUM+X(I)**K
655:       ENDDO
656:       IF(K.EQ.1) B=8.D00
657:       IF(K.EQ.2) B=18.D00
658:       IF(K.EQ.3) B=44.D00
659:       IF(K.EQ.4) B=114.D00
660:       F=F+(SUM-B)**2
661:       ENDDO
662:       RETURN
663:       ENDIF
664: C     ------------------------------------------------------------------
665:       IF(KF.EQ.15) THEN
666: C     GOLDSTEIN PRICE  FUNCTION  : MIN VALUE = 3 AT (0, -1)
667:       F=0.D00
668:       DO I=1,M
669:       IF(DABS(X(I)).GT.10.D00) THEN
670:       CALL RANDOM(RAND)
```

```
671:       X(I)=(RAND-.5D00)*20
672:       ENDIF
673:       ENDDO
674:       F11=(X(1)+X(2)+1.D00)**2
675:       F12=(19.D00-14*X(1)+ 3*X(1)**2-14*X(2)+ 6*X(1)*X(2)+ 3*X(2)**2)
676:       F1=1.00+F11*F12
677:       F21=(2*X(1)-3*X(2))**2
678:       F22=(18.D00-32*X(1)+12*X(1)**2+48*X(2)-36*X(1)*X(2)+27*X(2)**2)
679:       F2=30.D00+F21*F22
680:       F= (F1*F2)
681:       RETURN
682:       ENDIF
683: C     ------------------------------------------------------------------
684:       IF(KF.EQ.16) THEN
685: C     BUKIN'S 6TH FUNCTION  MIN = 0 FOR (-10, 1)
686: C     -15. LE. X(1) .LE. -5 AND -3 .LE. X(2) .LE. 3
687:         IF(X(1).LT. -15.D00 .OR. X(1) .GT. -5.D00) THEN
688:         CALL RANDOM(RAND)
689:         X(1)=-(RAND*10+5.D00)
690:         ENDIF
691:         IF(DABS(X(2)).GT.3.D00) THEN
692:         CALL RANDOM(RAND)
693:         X(2)=(RAND-.5D00)*6
694:         ENDIF
695:       F=100.D0*DSQRT(DABS(X(2)-0.01D0*X(1)**2))+ 0.01D0*DABS(X(1)+10.D0)
696:        RETURN
697:        ENDIF
698: C     ------------------------------------------------------------------
699:       IF(KF.EQ.17) THEN
700: C     NEW N#8 FUNCTION (MULTIPLE GLOBAL MINIMA)
701: C     MIN VALUE = -1 AT (AROUND .7 AROUND, 0.785 APPROX)
702:       F=0.D00
703:       DO I=1,M
704:       IF(X(I).LT.0.5D00 .OR. X(I).GT.1.D00) THEN
705:       CALL RANDOM(RAND)
706:       X(I)=RAND/2.D00
707:       ENDIF
708:       ENDDO
709:       F=-DEXP(-DABS(DLOG(.001D00+DABS((DSIN(X(1)+X(2))+DSIN(X(1)-X(2))+
710:      & (DCOS(X(1)+X(2))*DCOS(X(1)-X(2))+.001))**2)+
711:      & .01D00*(X(2)-X(1))**2)))
712:        RETURN
713:        ENDIF
714: C     ------------------------------------------------------------------
715:       IF(KF.EQ.18) THEN
716: C     DEFLECTED CORRUGATED SPRING FUNCTION
717: C     MIN VALUE = -1 AT (5, 5, ..., 5) FOR ANY K AND ALPHA=5; M VARIABLE
718:       CALL DCS(M,F,X)
719:       RETURN
720:       ENDIF
721: C     ------------------------------------------------------------------
722:       IF(KF.EQ.19) THEN
723: C     FACTORIAL FUNCTION, MIN =0 AT X=(1,2,3,...,M)
724:       CALL FACTOR1(M,F,X)
725:       RETURN
726:       ENDIF
727: C     ------------------------------------------------------------------
728:       IF(KF.EQ.20) THEN
729: C     DECANOMIAL FUNCTION, MIN =0 AT X=(2, -3)
730:         DO I=1,M
731:         IF(DABS(X(I)).GT.4.D00) THEN
732:         CALL RANDOM(RAND)
733:         X(I)= (RAND-0.5D00)*8
734:         ENDIF
735:         ENDDO
736:       CALL DECANOM(M,F,X)
737:       RETURN
```

```
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.21) THEN
C     JUDGE'S FUNCTION  F(0.864, 1.23) = 16.0817; M=2
      CALL JUDGE(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.22) THEN
C     DODECAL FUNCTION
      CALL DODECAL(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.23) THEN
C     WHEN X(1)*X(2)=X(1)*X(2)  ? M=2
      CALL SEQP(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.24) THEN
C     WHEN ARITHMETIC MEAN = GEOMETRIC MEAN ? : M =>1
      CALL AMGM(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.25) THEN
C     M =>2
      CALL FUNCT2(M,F,X)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.26) THEN
C     M =>2
      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     WOOD FUNCTION : F MIN  : M=4
      CALL WOOD(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.35) THEN
C     FENTON & EASON FUNCTION :    : M=2
      CALL FENTONEASON(M,X,F)
      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
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.38) THEN
C     EGGHOLDER FUNCTION M VARIABLES
      CALL EGGHOLD(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.39) THEN
C     TRID FUNCTION M VARIABLES
      CALL TRID(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.40) THEN
C     GRIEWANK FUNCTION M VARIABLES
      CALL GRIEWANK(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.41) THEN
C     WEIERSTRASS FUNCTION M VARIABLES
      CALL WEIERSTRASS(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.42) THEN
C     LEVY-3 FUNCTION 2 VARIABLES
      CALL LEVY3(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
```

```
872:       IF(KF.EQ.43) THEN
873: C     LEVY-5 FUNCTION 2 VARIABLES
874:       CALL LEVY5(M,X,F)
875:       RETURN
876:       ENDIF
877: C     ------------------------------------------------------------------
878:       IF(KF.EQ.44) THEN
879: C     LEVY-8 FUNCTION 3 VARIABLES
880:       CALL LEVY8(M,X,F)
881:       RETURN
882:       ENDIF
883: C     ------------------------------------------------------------------
884:       IF(KF.EQ.45) THEN
885: C     RASTRIGIN FUNCTION M VARIABLES
886:       CALL RASTRIGIN(M,X,F)
887:       RETURN
888:       ENDIF
889: C     ------------------------------------------------------------------
890:       IF(KF.EQ.46) THEN
891: C     ACKLEY FUNCTION M VARIABLES
892:       CALL ACKLEY(M,X,F)
893:       RETURN
894:       ENDIF
895: C     ------------------------------------------------------------------
896:       IF(KF.EQ.47) THEN
897: C     MICHALEWICZ FUNCTION M VARIABLES
898:       CALL MICHALEWICZ(M,X,F)
899:       RETURN
900:       ENDIF
901: C     ------------------------------------------------------------------
902:       IF(KF.EQ.48) THEN
903: C     SCHWEFEL FUNCTION M VARIABLES
904:       CALL SCHWEFEL(M,X,F)
905:       RETURN
906:       ENDIF
907: C     ------------------------------------------------------------------
908:       IF(KF.EQ.49) THEN
909: C     SHUBERT FUNCTION 2 VARIABLES
910:       CALL SHUBERT(M,X,F)
911:       RETURN
912:       ENDIF
913: C     ------------------------------------------------------------------
914:       IF(KF.EQ.50) THEN
915: C     DIXON AND PRICE FUNCTION M VARIABLES
916:       CALL DIXPRICE(M,X,F)
917:       RETURN
918:       ENDIF
919: C     ------------------------------------------------------------------
920:       IF(KF.EQ.51) THEN
921: C     SHEKEL FUNCTION 4 VARIABLES
922:       CALL SHEKEL(M,X,F)
923:       RETURN
924:       ENDIF
925: C     ------------------------------------------------------------------
926:       IF(KF.EQ.52) THEN
927: C     PAVIANI FUNCTION 10 VARIABLES
928:       CALL PAVIANI(M,X,F)
929:       RETURN
930:       ENDIF
931: C     ------------------------------------------------------------------
932:       IF(KF.EQ.53) THEN
933: C     BRANIN FUNCTION#1 2 VARIABLES
934:       CALL BRANIN1(M,X,F)
935:       RETURN
936:       ENDIF
937: C     ------------------------------------------------------------------
938:       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
```

```
1006: C     MIN F ([+/-] 9.64617, [+/-] 9.64617) APPROX = -26.92034 APPROX
1007:       CALL HOLDERTABLE(M,X,F)
1008:       RETURN
1009:       ENDIF
1010: C     ----------------------------------------------------------------
1011:       IF(KF.EQ.66) THEN
1012: C     CARROM-TABLE FUNCTION : 2 VARIABLES
1013: C     MIN F ([+/-] 9.64617, [+/-] 9.64617) APPROX = -24.15682 APPROX
1014:       CALL CARROMTABLE(M,X,F)
1015:       RETURN
1016:       ENDIF
1017: C     ----------------------------------------------------------------
1018:       IF(KF.EQ.67) THEN
1019: C     PEN-HOLDER FUNCTION : 2 VARIABLES
1020: C     MIN F ([+/-] 9.64617, [+/-] 9.64617) APPROX = -0.963535 APPROX
1021:       CALL PENHOLDER(M,X,F)
1022:       RETURN
1023:       ENDIF
1024: C     ----------------------------------------------------------------
1025:       IF(KF.EQ.68) THEN
1026: C     BIRD FUNCTION : 2 VARIABLES
1027: C     MIN F (4.70104, 3.15294) APPROX = -106.764537 APPROX  OR
1028: C     MIN F (-1.58214, -3.13024) APPROX = -106.764537 APPROX
1029:       CALL BIRD(M,X,F)
1030:       RETURN
1031:       ENDIF
1032: C     ----------------------------------------------------------------
1033:       IF(KF.EQ.69) THEN
1034: C     CHICHINADZE FUNCTION :  -30 <=X(I)<= 30;  M=2
1035: C     MIN F (5.901329, 0.5) = -43.3158621
1036:       CALL CHICHINADZE(M,X,F)
1037:       RETURN
1038:       ENDIF
1039: C     ----------------------------------------------------------------
1040:       IF(KF.EQ.70) THEN
1041: C     MCCORMICK FUNCTION : -1.5<= X(1)<=4; -3<=X(2)<=4 ; M=2
1042: C     MIN F (-0.54719755, -1.54719755) = -1.913223 APPROX
1043:       CALL MCCORMICK(M,X,F)
1044:       RETURN
1045:       ENDIF
1046: C     ----------------------------------------------------------------
1047:       IF(KF.EQ.71) THEN
1048: C     GLANKWAHMDEE FUNCTION:
1049:       CALL GLANKWAHMDEE(M,X,F)
1050:       RETURN
1051:       ENDIF
1052: C     ----------------------------------------------------------------
1053:       IF(KF.EQ.72) THEN
1054: C     FLETCHER-POWELL FUNCTION
1055:       CALL FLETCHER(M,X,F)
1056:       RETURN
1057:       ENDIF
1058: C     ----------------------------------------------------------------
1059:       IF(KF.EQ.73) THEN
1060: C     POWELL FUNCTION
1061:       CALL POWELL(M,X,F)
1062:       RETURN
1063:       ENDIF
1064: C     ----------------------------------------------------------------
1065:       IF(KF.EQ.74) THEN
1066: C     HARTMANN FUNCTION
1067:       CALL HARTMANN(M,X,F)
1068:       RETURN
1069:       ENDIF
1070: C     ----------------------------------------------------------------
1071:       IF(KF.EQ.75) THEN
1072: C     COVILLE FUNCTION
```

```
1073:       CALL COLVILLE(M,X,F)
1074:       RETURN
1075:       ENDIF
1076: C     ------------------------------------------------------------------
1077:       IF(KF.EQ.76) THEN
1078: C     HIMMELBLAU FUNCTION
1079:       CALL HIMMELBLAU(M,X,F)
1080:       RETURN
1081:       ENDIF
1082: C     ------------------------------------------------------------------
1083:       IF(KF.EQ.77) THEN
1084: C     BEALE FUNCTION
1085:       CALL BEALE(M,X,F)
1086:       RETURN
1087:       ENDIF
1088: C     ------------------------------------------------------------------
1089:       IF(KF.EQ.78) THEN
1090: C     BOOTH FUNCTION
1091:       CALL BOOTH(M,X,F)
1092:       RETURN
1093:       ENDIF
1094: C     ------------------------------------------------------------------
1095:       IF(KF.EQ.79) THEN
1096: C     HUMP FUNCTION
1097:       CALL HUMP(M,X,F)
1098:       RETURN
1099:       ENDIF
1100: C     ------------------------------------------------------------------
1101:       IF(KF.EQ.80) THEN
1102: C     MATYAS FUNCTION
1103:       CALL MATYAS(M,X,F)
1104:       RETURN
1105:       ENDIF
1106: C     ------------------------------------------------------------------
1107:       IF(KF.EQ.81) THEN
1108: C     MISHRA_1 FUNCTION
1109:       CALL MISHRA_1(M,X,F)
1110:       RETURN
1111:       ENDIF
1112: C     ------------------------------------------------------------------
1113:       IF(KF.EQ.82) THEN
1114: C     MISHRA_2 FUNCTION
1115:       CALL MISHRA_2(M,X,F)
1116:       RETURN
1117:       ENDIF
1118: C     ------------------------------------------------------------------
1119:       IF(KF.EQ.83) THEN
1120: C     ZAKHAROV FUNCTION
1121:       CALL ZAKHAROV(M,X,F)
1122:       RETURN
1123:       ENDIF
1124: C     ------------------------------------------------------------------
1125:       IF(KF.EQ.84) THEN
1126: C     MULTOMOD FUNCTION
1127:       CALL MULTIMOD(M,X,F)
1128:       RETURN
1129:       ENDIF
1130: C     ------------------------------------------------------------------
1131:       IF(KF.EQ.85) THEN
1132: C     NONLINEAR  FUNCTION
1133:       CALL NONLIN(M,X,F)
1134:       RETURN
1135:       ENDIF
1136: C     ------------------------------------------------------------------
1137:       IF(KF.EQ.86) THEN
1138: C     QUADRATIC  FUNCTION
1139:       CALL QUADRATIC(M,X,F)
```

```
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.87) THEN
C     TRIGON FUNCTION
      CALL TRIGON(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.88) THEN
C     WAVY-1 FUNCTION
      CALL  WAVY1(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.89) THEN
C     WAVY-2 FUNCTION
      CALL  WAVY2(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.90) THEN
C     TREFETHEN FUNCTION
      CALL  TREFETHEN(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.91) THEN
C     ALPINE FUNCTION
      CALL  ALPINE(M,X,F)
      RETURN
      ENDIF
C     ------------------------------------------------------------------
C      IF(KF.EQ.99) THEN
C     *****  FUNCTION
C      CALL SUBROUTINE(M,X,F)
C      RETURN
C      ENDIF
C     ==================================================================
      WRITE(*,*)'FUNCTION NOT DEFINED. PROGRAM ABORTED'
      STOP
      END
C     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      SUBROUTINE DCS(M,F,X)
C     FOR DEFLECTED CORRUGATED SPRING FUNCTION
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*),C(100)
C     OPTIMAL VALUES OF (ALL) X ARE ALPHA , AND K IS ONLY FOR SCALING
C     OPTIMAL VALUE OF F IS -1. DIFFICULT TO OPTIMIZE FOR LARGER M.
      DATA K,ALPHA/5,5.D00/ ! K AND ALPHA COULD TAKE ON ANY OTHER VALUES
      R2=0.D00
      DO I=1,M
      C(I)=ALPHA
      R2=R2+(X(I)-C(I))**2
      ENDDO
      R=DSQRT(R2)
      F=-DCOS(K*R)+0.1D00*R2
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE QUINTIC(M,F,X)
C     QUINTIC FUNCTION: GLOBAL MINIMA,EXTREMELY DIFFICULT TO OPTIMIZE
C     MIN VALUE = 0 AT PERM( -1, -0.402627941, 2)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(*)
      F=0.D00
      DO I=1,M
```

```
1207:       F=F+DABS(X(I)**5-3*X(I)**4+4*X(I)**3+2*X(I)**2-10*X(I)-4.D00)
1208:       ENDDO
1209:       F=1000*F
1210:       RETURN
1211:       END
1212: C     ------------------------------------------------------------------
1213:       SUBROUTINE FACTOR1(M,F,X)
1214: C     FACTORIAL FUNCTION; MIN (1, 2, 3, ...., M) = 0
1215: C     FACT = FACTORIAL(M) = 1 X 2 X 3 X 4 X .... X M
1216: C     FIND X(I), I=1,2,...,M SUCH THAT THEIR PRODUCT IS EQUAL TO FACT.
1217: C     LARGER THE VALUE OF M (=>8) OR SO, HARDER IS THE PROBLEM
1218:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1219:       DIMENSION X(*)
1220:       F=0.D00
1221:       FACT=1.D00
1222:       P=1.D00
1223:       DO I=1,M
1224:       FACT=FACT*I
1225:       P=P*X(I)
1226:       F=F+DABS(P-FACT)**2
1227:       ENDDO
1228:       RETURN
1229:       END
1230: C     ------------------------------------------------------------------
1231:       SUBROUTINE DECANOM(M,F,X)
1232: C     DECANOMIAL FUNCTION; MIN (2, -3) = 0
1233:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1234:       DIMENSION X(*)
1235:       F1= DABS(X(1)**10-20*X(1)**9+180*X(1)**8-960*X(1)**7+
1236:      & 3360*X(1)**6-8064*X(1)**5+13340*X(1)**4-15360*X(1)**3+
1237:      & 11520*X(1)**2-5120*X(1)+2624.D00)
1238:       F2= DABS(X(2)**4+12*X(2)**3+54*X(2)**2+108*X(2)+81.D00)
1239:       F=0.001D00*(F1+F2)**2
1240:       RETURN
1241:       END
1242: C     ------------------------------------------------------------------
1243:       SUBROUTINE JUDGE(M,X,F)
1244:       PARAMETER (N=20)
1245: C     THIS SUBROUTINE IS FROM THE EXAMPLE IN JUDGE ET AL., THE THEORY
1246: C     AND PRACTICE OF ECONOMETRICS, 2ND ED., PP. 956-7. THERE ARE TWO
1247: C     OPTIMA: F(0.86479,1.2357)=16.0817307 (WHICH IS THE GLOBAL MINUMUM)
1248: C     AND F(2.35,-0.319)=20.9805 (WHICH IS LOCAL). ADAPTED FROM BILL
1249: C     GOFFE'S SIMMAN (SIMULATED ANNEALING) PROGRAM
1250:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1251:       DIMENSION Y(N), X2(N), X3(N), X(*)
1252:        DATA (Y(I),I=1,N)/4.284,4.149,3.877,0.533,2.211,2.389,2.145,
1253:      & 3.231,1.998,1.379,2.106,1.428,1.011,2.179,2.858,1.388,1.651,
1254:      & 1.593,1.046,2.152/
1255:        DATA (X2(I),I=1,N)/.286,.973,.384,.276,.973,.543,.957,.948,.543,
1256:      & .797,.936,.889,.006,.828,.399,.617,.939,.784,.072,.889/
1257:       DATA (X3(I),I=1,N)/.645,.585,.310,.058,.455,.779,.259,.202,.028,
1258:      & .099,.142,.296,.175,.180,.842,.039,.103,.620,.158,.704/
1259:
1260:       F=0.D00
1261:       DO I=1,N
1262:       F=F+(X(1) + X(2)*X2(I) + (X(2)**2)*X3(I) - Y(I))**2
1263:       ENDDO
1264:       RETURN
1265:       END
1266: C     ------------------------------------------------------------------
1267:       SUBROUTINE DODECAL(M,F,X)
1268:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1269:       DIMENSION X(*)
1270: C     DODECAL POLYNOMIAL MIN F(1,2,3)=0
1271:       DO I=1,M
1272:       IF(DABS(X(I)).GT.5.D0) THEN
1273:       CALL RANDOM(RAND)
```

```
1274:       X(I)=(RAND-0.5D00)*10
1275:       ENDIF
1276:       ENDDO
1277:       F=0.D00
1278:       F1=2*X(1)**3+5*X(1)*X(2)+4*X(3)-2*X(1)**2*X(3)-18.D00
1279:       F2=X(1)+X(2)**3+X(1)*X(2)**2+X(1)*X(3)**2-22.D00
1280:       F3=8*X(1)**2+2*X(2)*X(3)+2*X(2)**2+3*X(2)**3-52.D00
1281:       F=(F1*F3*F2**2+F1*F2*F3**2+F2**2+(X(1)+X(2)-X(3))**2)**2
1282:       RETURN
1283:       END
1284: C     ------------------------------------------------------------------
1285:       SUBROUTINE SEQP(M,F,X)
1286:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1287:       DIMENSION X(*)
1288: C     FOR WHAT VALUES X(1)+X(2)=X(1)*X(2) ? ANSWER: FOR (0,0) AND (2,2)
1289: C     WHILE X(1), X(2) ARE INTEGERS.
1290:       X(1)=INT(X(1)) ! X(1) CONVERTED TO INTEGER
1291:       X(2)=INT(X(2)) ! X(2) CONVERTED TO INTEGER
1292:
1293:       F1=X(1)+X(2)
1294:       F2=X(1)*X(2)
1295:       F=(F1-F2)**2     ! TURN ALIVE THIS  XOR
1296: C      F=DABS(F1-F2)    ! TURN ALIVE THIS  - BUT NOT BOTH --------------
1297:       RETURN
1298:       END
1299: C     ------------------------------------------------------------------
1300:       SUBROUTINE AMGM(M,F,X)
1301:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1302:       DIMENSION X(*)
1303: C     FOR WHAT VALUES ARITHMETIC MEAN = GEOMETRIC MEAN ? THE ANSWER IS:
1304: C     IF X(1)=X(2)=....=X(M) AND ALL X ARE NON-NEGATIVE
1305: C     TAKE ONLY THE ABSOLUTE VALUES OF X
1306:       SUM=0.D00
1307:       DO I=1,M
1308:       X(I)=DABS(X(I))
1309:       ENDDO
1310: C     SET SUM = SOME POSITIVE NUMBER. THIS MAKES THE FUNCTION UNIMODAL
1311:       SUM= 100.D00 ! TURNED  ALIVE FOR UNIQUE MINIMUM AND SET SUM TO
1312: C     SOME POSITIVE NUMBER. HERE IT IS 100; IT COULD BE ANYTHING ELSE.
1313:       F1=0.D00
1314:       F2=1.D00
1315:       DO I=1,M
1316:       F1=F1+X(I)
1317:       F2=F2*X(I)
1318:       ENDDO
1319:       XSUM=F1
1320:       F1=F1/M ! SUM DIVIDED BY M = ARITHMETIC MEAN
1321:       F2=F2**(1.D00/M) ! MTH ROOT OF THE PRODUCT = GEOMETRIC MEAN
1322:       F=(F1-F2)**2
1323:       IF(SUM.GT.0.D00) F=F+(SUM-XSUM)**2
1324:       RETURN
1325:       END
1326: C     ------------------------------------------------------------------
1327:       SUBROUTINE FUNCT2(M,F,X)
1328: C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
1329: C     IN FOGEL, L.J., ANGELIN, P.J. AND BACK, T. (ED) PROCEEDINGS OF THE
1330: C     FIFTH ANNUAL CONFERENCE ON EVOLUTIONARY PROGRAMMING, PP. 451-460,
1331: C     MIT PRESS, CAMBRIDGE, MASS.
1332: C     MIN F (0, 0, ..., 0) = 0
1333:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1334:       DIMENSION X(*)
1335:       F=0.D00
1336:       F1=1.D00
1337:       DO I=1,M
1338:       IF(DABS(X(I)).GT.10.D00) THEN
1339:       CALL RANDOM(RAND)
1340:       X(I)=(RAND-.5D00)*20
```

```
      ENDIF
      ENDDO
      DO I=1,M
      F=F+DABS(X(I))
      F1=F1*DABS(X(I))
      ENDDO
      F=F+F1
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE FUNCT3(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)
      DIMENSION X(*)
      F=0.D00
      F1=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
      F1=0.D00
      DO J=1,I
      F1=F1+X(J)**2
      ENDDO
      F=F+F1
      ENDDO
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE FUNCT4(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)
      DIMENSION X(*)
      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
```

```
1408:        F=F+(X(I)+0.5D00)**2
1409:       ENDDO
1410:       RETURN
1411:       END
1412: C     ------------------------------------------------------------------
1413:       SUBROUTINE FUNCT7(M,F,X)
1414: C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
1415: C     MIN F(0, 0, ..., 0) = 0
1416:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1417:       COMMON /RNDM/IU,IV
1418:       INTEGER IU,IV
1419:       DIMENSION X(*)
1420:       F=0.D00
1421:         DO I=1,M
1422:         IF(DABS(X(I)).GT.1.28D00) THEN
1423:         CALL RANDOM(RAND)
1424:         X(I)=(RAND-0.5D00)*2.56D00
1425:         ENDIF
1426:         ENDDO
1427:       DO I=1,M
1428:       CALL RANDOM(RAND)
1429:       F=F+(I*X(I)**4)
1430:       ENDDO
1431:        CALL RANDOM(RAND)
1432:        F=F+RAND
1433:       RETURN
1434:       END
1435: C     ------------------------------------------------------------------
1436:       SUBROUTINE FUNCT12(M,F,X)
1437: C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
1438:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1439:       DIMENSION X(100),Y(100)
1440:       DATA A,B,C /10.D00,100.D00,4.D00/
1441:       PI=4.D00*DATAN(1.D00)
1442:       F=0.D00
1443: C     MIN F (-1, -1, -1, ..., -1) = 0
1444: C       X(I)=-1.D00  ! TO CHECK, TURN IT ALIVE
1445:        DO I=1,M
1446:         IF(DABS(X(I)).GT.50.D00) THEN
1447:         CALL RANDOM(RAND)
1448:         X(I)=(RAND-0.5D00)*100.D00
1449:         ENDIF
1450:         ENDDO
1451:       F1=0.D00
1452:       DO I=1,M
1453:       XX=DABS(X(I))
1454:       U=0.D00
1455:       IF(XX.GT.A) U=B*(XX-A)**C
1456:       F1=F1+U
1457:       ENDDO
1458:       F2=0.D00
1459:       DO I=1,M-1
1460:       Y(I)=1.D00+.25D00*(X(I)+1.D00)
1461:       F2=F2+ (Y(I)-1.D00)**2 * (1.D00+10.D00*(DSIN(PI*X(I+1))**2))
1462:       ENDDO
1463:       Y(M)=1.D00+.25D00*(X(M)+1.D00)
1464:       F3=(Y(M)-1.D00)**2
1465:       Y(1)=1.D00+.25D00*(X(1)+1.D00)
1466:       F4=10.D00*(DSIN(PI*Y(1)))**2
1467:       F=(PI/M)*(F4+F2+F3)+F1
1468:       RETURN
1469:       END
1470: C     ------------------------------------------------------------------
1471:       SUBROUTINE FUNCT13(M,F,X)
1472: C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
1473:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1474:       DIMENSION X(100)
```

```
1475:       DATA A,B,C /5.D00,100.D00,4.D00/
1476:       PI=4*DATAN(1.D00)
1477:       F=0.D00
1478: C     MIN F (1, 1, 1, ..., 4.7544 APPROX) = -1.15044 APPROX
1479: C        X(I)=1.D00  ! TO CHECK, TURN IT ALIVE
1480: C        X(M)=-4.7544  ! TO CHECK, TURN IT ALIVE
1481:       DO I=1,M
1482:       IF(DABS(X(I)).GT.50.D00) THEN
1483:       CALL RANDOM(RAND)
1484:       X(I)=(RAND-.5D00)*100.D00
1485:       ENDIF
1486:       ENDDO
1487:       F1=0.D00
1488:       DO I=1,M
1489:       XX=DABS(X(I))
1490:       U=0.D00
1491:       IF(XX.GT.A) U=B*(XX-A)**C
1492:       F1=F1+U
1493:       ENDDO
1494:       F2=0.D00
1495:       DO I=1,M-1
1496:       F2=F2+ (X(I)-1.D00)**2 * (1.D00+(DSIN(3*PI*X(I+1))**2))
1497:       ENDDO
1498:       F3=(X(M)-1.D00)*  (1.D00+(DSIN(2*PI*X(M)))**2)
1499:       F4=(DSIN(3*PI*X(1)))**2
1500:       F=0.1*(F4+F2+F3)+F1
1501:       RETURN
1502:       END
1503: C     ------------------------------------------------------------------
1504:       SUBROUTINE FUNCT14(M,F,X)
1505: C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
1506: C     MIN F (-31.98, 31.98) = 0.998
1507:       PARAMETER (N=25,NN=2)
1508:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1509:       DIMENSION X(2), A(NN,N)
1510:       DATA (A(1,J),J=1,N) /-32.D00,-16.D00,0.D00,16.D00,32.D00,-32.D00,
1511:      & -16.D00,0.D00,16.D00,32.D00,-32.D00,-16.D00,0.D00,16.D00,32.D00,
1512:      & -32.D0,-16.D0,0.D0,16.D0,32.D0,-32.D0,-16.D0,0.D0,16.D0,32.D0/
1513:       DATA (A(2,J),J=1,N) /-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,
1514:      & -16.D00,-16.D00,-16.D00,-16.D00,-16.D00,0.D00,0.D00,0.D00,0.D00,
1515:      & 0.D00,16.D00,16.D00,16.D00,16.D00,16.D00,32.D00,32.D00,
1516:      & 32.D00,32.D00,32.D00/
1517:
1518:       F=0.D00
1519:       DO I=1,M
1520:       IF(DABS(X(I)).GT.100.D00) THEN
1521:       CALL RANDOM(RAND)
1522:       X(I)=(RAND-.5D00)*200.D00
1523:       ENDIF
1524:       ENDDO
1525:       F1=0.D00
1526:       DO J=1,N
1527:       F2=0.D00
1528:       DO I=1,2
1529:       F2=F2+(X(I)-A(I,J))**6
1530:       ENDDO
1531:       F2=1.D00/(J+F2)
1532:       F1=F1+F2
1533:       ENDDO
1534:       F=1.D00/(0.002D00+F1)
1535:       RETURN
1536:       END
1537: C     ------------------------------------------------------------------
1538:       SUBROUTINE FUNCT15(M,F,X)
1539: C     REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING
1540: C     MIN F(.19, .19, .12, .14) = 0.3075
1541:       PARAMETER (N=11)
```

```
1542:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1543:       DIMENSION X(*), A(N),B(N)
1544:       DATA (A(I),I=1,N) /.1957D00,.1947D00,.1735D00,.16D00,.0844D00,
1545:      & .0627D00,.0456D00,.0342D00,.0323D00,.0235D00,.0246D00/
1546:       DATA (B(I),I=1,N) /0.25D00,0.5D00,1.D00,2.D00,4.D00,6.D00,8.D00,
1547:      & 10.D00,12.D00,14.D00,16.D00/
1548:       DO I=1,N
1549:       B(I)=1.D00/B(I)
1550:       ENDDO
1551:       F=0.D00
1552:       DO I=1,M
1553:       IF(DABS(X(I)).GT.5.D00) THEN
1554:       CALL RANDOM(RAND)
1555:       X(I)=(RAND-.5D00)*10.D00
1556:       ENDIF
1557:       ENDDO
1558:       DO I=1,N
1559:       F1=X(1)*(B(I)**2+B(I)*X(2))
1560:       F2=B(I)**2+B(I)*X(3)+X(4)
1561:       F=F+(A(I)-F1/F2)**2
1562:       ENDDO
1563:       F=F*1000
1564:       RETURN
1565:       END
1566: C     ----------------------------------------------------------------
1567:       SUBROUTINE LINPROG1(M,F,X)
1568: C     LINEAR PROGRAMMING : MINIMIZATION PROBLEM
1569: C     IN THIS PROBLEM : M = NO. OF DECISION VARIABLES = 2
1570: C     MIN F (2.390, 2.033) = -19.7253 APPROX
1571: C     MIN F = OVER J=1, M : DO SUM(A(1,J)*X(J))  SUBJECT TO CONSTRAINTS
1572: C     OVER  J=1, M : DO SUM(A(I,J)*X(J)) <= C(I) ; I=2
1573: C     . . .          . . .          . . .          . . .
1574: C     OVER  J=1, M : DO SUM(A(I,J)*X(J)) <= C(I) ; I=N
1575: C     ALL X(I) => 0
1576:       PARAMETER (N=3) ! N IS THE NO. OF CONSTRAINTS + 1
1577:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1578:       DIMENSION X(*),A(20,10),C(20),FF(20)
1579:       DATA (A(1,J),J=1,2),C(1)/4.D0,5.D0,0.0D0/!COEFF OF OBJ FUNCTION
1580:       DATA (A(2,J),J=1,2),C(2)/10.D0,3.D0,30D0/!COEFF OF 1ST CONSTRAINT
1581:       DATA (A(3,J),J=1,2),C(3)/6.D0,20.D0,55.D0/!COEFF OF 2ND CONSTRAINT
1582: C     ----------------------------------------------------------------
1583: C     USING ONLY NON-NEGATIVE VALUES OF X(I)
1584:       DO I=1,M
1585:       X(I)=DABS(X(I))
1586:       ENDDO
1587: C     EVALUATION OF OBJ FUNCTION AND CONSTRAINTS
1588:       DO I=1,N
1589:       FF(I)=0.D00
1590:       DO J=1,M
1591:       FF(I)=FF(I)+A(I,J)*X(J)
1592:       ENDDO
1593:       ENDDO
1594:       F=-FF(1) ! CHANGE OF SIGN FOR MINIMIZATION
1595: C     CHECK FOR SATISFYING OR VIOLATING THE CONSTRAINTS
1596:       DO I=2,N
1597:       FF(I)=FF(I)-C(I) ! SLACK
1598: C     PENALTY FOR CROSSING LIMITS
1599:       IF(FF(I).GT.0) F=F+(10+FF(I))**2
1600:       ENDDO
1601:       RETURN
1602:       END
1603: C     ----------------------------------------------------------------
1604:       SUBROUTINE LINPROG2(M,F,X)
1605: C     LINEAR PROGRAMMING : MINIMIZATION PROBLEM
1606: C     IN THIS PROBLEM : M = NO. OF DECISION VARIABLES = 3
1607: C     MIN F (250, 625, 0) = -3250
1608: C     MIN F = OVER J=1, M : DO SUM(A(1,J)*X(J))   SUBJECT TO CONSTRAINTS
```

```
1609: C     OVER  J=1, M : DO SUM(A(I,J)*X(J)) <= C(I) ; I=2
1610: C     . . .          . . .          . . .       . . .
1611: C     OVER  J=1, M : DO SUM(A(I,J)*X(J)) <= C(I) ; I=N
1612: C     ALL X(I) => 0
1613:       PARAMETER (N=4) ! N IS THE NO. OF CONSTRAINTS + 1
1614:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1615:       DIMENSION X(*),A(20,10),C(20),FF(20)
1616:       DATA (A(1,J),J=1,3),C(1)/30.D0,40.D0,20.D0,0.0D0/! COEFF OF OBJ FUNCTION
1617:       DATA (A(2,J),J=1,3),C(2)/10.D0,12.D0,7.D0,10000.0D0/!COEFF OF 1ST CONSTRAINT
1618:       DATA (A(3,J),J=1,3),C(3)/7.D0,10.D0,8.D0,8000.D0/! COEFF OF 2ND CONSTRAINT
1619:       DATA (A(4,J),J=1,3),C(4)/1.D0,1.D0,1.D0,1000.D0/! COEFF OF 3RD CONSTRAINT
1620: C     ------------------------------------------------------------------
1621: C     USING ONLY NON-NEGATIVE VALUES OF X(I)
1622:       DO I=1,M
1623:       X(I)=DABS(X(I))
1624:       ENDDO
1625: C     EVALUATION OF OBJ FUNCTION AND CONSTRAINTS
1626:       DO I=1,N
1627:       FF(I)=0.D00
1628:       DO J=1,M
1629:       FF(I)=FF(I)+A(I,J)*X(J)
1630:       ENDDO
1631:       ENDDO
1632:       F=-FF(1) ! CHANGE OF SIGN FOR MINIMIZATION
1633: C     CHECK FOR SATISFYING OR VIOLATING THE CONSTRAINTS
1634:       DO I=2,N
1635:       FF(I)=FF(I)-C(I) ! SLACK
1636: C     PENALTY FOR CROSSING LIMITS
1637:       IF(FF(I).GT.0.D00) F=F+(100.D00+FF(I))**2
1638:       ENDDO
1639:       RETURN
1640:       END
1641: C     ------------------------------------------------------------------
1642:       SUBROUTINE HOUGEN(A,M,F)
1643:       PARAMETER(N=13,K=3)
1644:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1645:       DIMENSION X(N,K),RATE(N),A(*)
1646: C     ------------------------------------------------------------------
1647: C     HOUGEN FUNCTION (HOUGEN-WATSON MODEL FOR REACTION KINATICS)
1648: C     NO. OF PARAMETERS (A) TO ESTIMATE = 5 = M
1649:
1650: C     BEST RESULTS ARE:
1651: C     A(1)=1.253031; A(2)=1.190943; A(3)=0.062798; A(4)=0.040063
1652: C     A(5)=0.112453  ARE BEST ESTIMATES OBTAINED BY ROSENBROCK &
1653: C     QUASI-NEWTON METHOD WITH SUM OF SQUARES OF DEVIATION =0.298900994
1654: C     AND R=0.99945.
1655:
1656: C     THE NEXT BEST RESULTS GIVEN BY HOOKE-JEEVES & QUASI-NEWTON
1657: C     A(1)=2.475221;A(2)=0.599177; A(3)=0.124172; A(4)=0.083517
1658: C     A(5)=0.217886; SUM OF SQUARES OF DEVIATION = 0.318593458
1659: C     R=0.99941
1660: C     MOST OF THE OTHER METHODS DO NOT PERFORM WELL
1661: C     ------------------------------------------------------------------
1662:       DATA X(1,1),X(1,2),X(1,3),RATE(1) /470,300,10,8.55/
1663:       DATA X(2,1),X(2,2),X(2,3),RATE(2) /285,80,10,3.79/
1664:       DATA X(3,1),X(3,2),X(3,3),RATE(3) /470,300,120,4.82/
1665:       DATA X(4,1),X(4,2),X(4,3),RATE(4) /470,80,120,0.02/
1666:       DATA X(5,1),X(5,2),X(5,3),RATE(5) /470,80,10,2.75/
1667:       DATA X(6,1),X(6,2),X(6,3),RATE(6) /100,190,10,14.39/
1668:       DATA X(7,1),X(7,2),X(7,3),RATE(7) /100,80,65,2.54/
1669:       DATA X(8,1),X(8,2),X(8,3),RATE(8) /470,190,65,4.35/
1670:       DATA X(9,1),X(9,2),X(9,3),RATE(9) /100,300,54,13/
1671:       DATA X(10,1),X(10,2),X(10,3),RATE(10) /100,300,120,8.5/
1672:       DATA X(11,1),X(11,2),X(11,3),RATE(11) /100,80,120,0.05/
1673:       DATA X(12,1),X(12,2),X(12,3),RATE(12) /285,300,10,11.32/
1674:       DATA X(13,1),X(13,2),X(13,3),RATE(13) /285,190,120,3.13/
1675: C     WRITE(*,1)((X(I,J),J=1,K),RATE(I),I=1,N)
```

```
1676: C    1 FORMAT(4F8.2)
1677:       DO J=1,M
1678:       IF(DABS(A(J)).GT.5.D00) THEN
1679:       CALL RANDOM(RAND)
1680:       A(J)=(RAND-0.5D00)*10.D00
1681:       ENDIF
1682:       ENDDO
1683:       F=0.D00
1684:       DO I=1,N
1685:       D=1.D00
1686:          DO J=1,K
1687:          D=D+A(J+1)*X(I,J)
1688:          ENDDO
1689:       FX=(A(1)*X(I,2)-X(I,3)/A(M))/D
1690: C     FX=(A(1)*X(I,2)-X(I,3)/A(5))/(1.D00+A(2)*X(I,1)+A(3)*X(I,2)+
1691: C     A(4)*X(I,3))
1692:       F=F+(RATE(I)-FX)**2
1693:       ENDDO
1694:       RETURN
1695:       END
1696: C     ---------------------------------------------------------------
1697:       SUBROUTINE GIUNTA(M,X,F)
1698:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1699:       DIMENSION X(*)
1700: C     GIUNTA FUNCTION
1701: C     X(I) = -1 TO 1;  M=2
1702:         DO I=1,M
1703:         IF(DABS(X(I)).GT.1.D00) THEN
1704:         CALL RANDOM(RAND)
1705:         X(I)=(RAND-0.5D00)*2.D00
1706:         ENDIF
1707:         ENDDO
1708:       C=16.D00/15.D00
1709:       F=DSIN(C*X(1)-1.D0)+DSIN(C*X(1)-1.D0)**2+DSIN(4*(C*X(1)-1.D0))/50+
1710:      &DSIN(C*X(2)-1.D0)+DSIN(C*X(2)-1.D0)**2+DSIN(4*(C*X(2)-1.D0))/50+.6
1711:       RETURN
1712:       END
1713: C     ---------------------------------------------------------------
1714:       SUBROUTINE EGGHOLD(M,X,F)
1715: C     EGG HOLDER FUNCTION
1716:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1717:       DIMENSION X(*)
1718:       DO I=1,M
1719:        IF(DABS(X(I)).GT.512.D00) THEN
1720:         CALL RANDOM(RAND)
1721:         X(I)=(RAND-0.5D00)*1024.D00
1722:        ENDIF
1723:        ENDDO
1724:        F=0.D00
1725:       DO I=1,M-1
1726:       F1=-(X(I+1)+47.D00)
1727:       F2=DSIN(  DSQRT(  DABS(  X(I+1)+X(I)/2+47.D00  )  )  )
1728:       F3=DSIN(  DSQRT(  DABS(  X(I)-(X(I+1)+47.D00) ) ))
1729:       F4=-X(I)
1730:       F=F+ F1*F2+F3*F4
1731:       ENDDO
1732:       RETURN
1733:       END
1734: C     ---------------------------------------------------------------
1735:       SUBROUTINE TRID(M,X,F)
1736: C     TRID FUNCTION
1737:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1738:       DIMENSION X(*)
1739:       F1=0.D00
1740:       F2=0.D00
1741:       DO I=1, M
1742:       F1=F1+(X(I)-1.D00)**2
```

```
1743:       ENDDO
1744:       DO I=2, M
1745:       F2=F2+X(I)*X(I-1)
1746:       ENDDO
1747:       F=F1-F2
1748:       RETURN
1749:       END
1750: C     ------------------------------------------------------------------
1751:       SUBROUTINE GRIEWANK(M,X,F)
1752:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1753:       DIMENSION X(*)
1754: C     GRIEWANK FUNCTION
1755:       F1=0.D00
1756:       F2=1.0D00
1757:       DO I=1,M
1758:       F1=F1+X(I)**2
1759:       FI=DFLOAT(I)
1760:       F2=F2*DCOS(X(I)/DSQRT(FI))
1761:       ENDDO
1762:       F=F1/4000.D00-F2+1.0D00
1763:       RETURN
1764:       END
1765: C     ------------------------------------------------------------------
1766:       SUBROUTINE WEIERSTRASS(M,X,F)
1767:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1768:       DIMENSION X(*)
1769:       PI=4*DATAN(1.D00)
1770: C     WEIERSTRASS FUNCTION ----------------------------------
1771:       DATA AX,BX,KMAX/0.5D00,3.D00,20/
1772:       F1=0.D00
1773:       F2=0.D00
1774:
1775:         DO I=1,M
1776:         IF(DABS(X(I)).GT.0.5D00) THEN
1777:         CALL RANDOM(RAND)
1778:         X(I)=RAND-0.5D00
1779:         ENDIF
1780:         ENDDO
1781:
1782:       DO I=1,M
1783:       S=0.D00
1784:           DO KK=1,KMAX+1
1785:           K=KK-1
1786:           S=S+(AX**K*DCOS(2*PI*BX**K*(X(I)+0.5D00)))
1787:           ENDDO
1788:       F1=F1+S
1789:       ENDDO
1790:
1791:       DO KK=1,KMAX+1
1792:       K=KK-1
1793:       F2=F2+(AX**K*DCOS(2*PI*BX**K*0.5D00))
1794:       ENDDO
1795:       F=F1-M*F2
1796:       RETURN
1797:       END
1798: C     ------------------------------------------------------------------
1799:       SUBROUTINE LEVY3(M,X,F)
1800:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1801:       DIMENSION X(*)
1802: C     LEVY # 3 (LEVY ET AL. 1981) ---------------------------------
1803:          DO I=1,M
1804:          IF(DABS(X(I)).GT.10.D00) THEN
1805:          CALL RANDOM(RAND)
1806:          X(I)=(RAND-0.5D00)*20
1807:          ENDIF
1808:          ENDDO
1809:           F1=0.0D+00
```

```
1810:           F2=0.0D+00
1811:           DO I=1,5
1812:           F1=F1+(I*DCOS((I-1)*X(1)+I))
1813:           F2=F2+(I*DCOS((I+1)*X(2)+I))
1814:           ENDDO
1815:          F=F1*F2
1816:          RETURN
1817:          END
1818: C      ------------------------------------------------------------------
1819:       SUBROUTINE LEVY5(M,X,F)
1820:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1821:       DIMENSION X(*)
1822:             F1=0.0D+00
1823:          F2=0.0D+00
1824:        DO I=1,5
1825:       F1=F1+(I*DCOS((I-1)*X(1)+I))
1826:          F2=F2+(I*DCOS((I+1)*X(2)+I))
1827:          ENDDO
1828:          F3=(X(1)+1.42513D+00)**2
1829:         F4=(X(2)+0.80032D+00)**2
1830:         F=(F1*F2) + (F3+F4)
1831:         RETURN
1832:         END
1833: C      ------------------------------------------------------------------
1834:       SUBROUTINE LEVY8(M,X,F)
1835:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1836:       DIMENSION X(*),Y(3)
1837:       PI=4*DATAN(1.D00)
1838: C     LEVY # 8 FUNCTION ------------------------------------------------
1839:          DO I=1,3
1840:          Y(I)=1.D+00+(X(I)-1.D+00)/4.D+00
1841:          ENDDO
1842:          F1=DSIN(PI*Y(1))**2
1843:        F3=(Y(3)-1.D+00)**2
1844:       F2=0.D+00
1845:        DO I=1,2
1846:         F2=F2+((Y(I)-1.D+00)**2)*(1.D+00+10.D+00*(DSIN(PI*Y(I+1)))**2)
1847:         ENDDO
1848:       F=F1+F2+F3
1849:       RETURN
1850:       END
1851: C      ------------------------------------------------------------------
1852:       SUBROUTINE RASTRIGIN(M,X,F)
1853:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1854:       DIMENSION X(*)
1855:       PI=4*DATAN(1.D00)
1856: C     RASTRIGIN'S FUNCTION
1857:       F=0.D00
1858:       DO I=1,M
1859:       F=F+ X(I)**2 -10*DCOS(2*PI*X(I)) + 10.D00
1860:       ENDDO
1861:       RETURN
1862:       END
1863: C      ------------------------------------------------------------------
1864:       SUBROUTINE ACKLEY(M,X,F)
1865: C     ACKLEY FUNCTION
1866:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1867:       DIMENSION X(*)
1868:       PI=4*DATAN(1.D00)
1869:       F=20.D00+DEXP(1.D00)
1870:       DO I=1,M
1871:       IF(X(I).LT. -15.D00 .OR. X(I).GT.30.D00) THEN
1872:       CALL RANDOM(RAND)
1873:       X(I)=(RAND-0.5D00)*90 -15.D00
1874:       ENDIF
1875:       ENDDO
1876:       F1=0.D00
```

```
1877:       F2=0.D00
1878:         DO I=1,M
1879:          F1=F1+X(I)**2
1880:          F2=F2+DCOS(2*PI*X(I))
1881:         ENDDO
1882:       F1=-20*DEXP(-0.2D00*DSQRT(F1/M))
1883:       F2=-DEXP(F2/M)
1884:       F=F+F1+F2
1885:       RETURN
1886:       END
1887: C     ----------------------------------------------------------------
1888:       SUBROUTINE MICHALEWICZ(M,X,F)
1889:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1890:       DIMENSION X(*)
1891: C     MICHALEWICZ FUNCTION   [ 0 <= X(I) <= PI ] MP IS A PARAMETER
1892:       MP=10  ! SET IT TO THE DESIRED VALUE
1893:       PI=4*DATAN(1.D00)
1894:       DO I=1,M
1895:       IF(X(I).LT.0.D00 .OR. X(I).GT.PI)THEN
1896:       CALL RANDOM(RAND)
1897:       X(I)=RAND*PI
1898:       ENDIF
1899:       ENDDO
1900:       F=0.D00
1901:       DO I=1,M
1902:       F=F-DSIN(X(I))*(DSIN(I*X(I)**2/PI))**(2*MP)
1903:       ENDDO
1904:       RETURN
1905:       END
1906: C     ----------------------------------------------------------------
1907:       SUBROUTINE SCHWEFEL(M,X,F)
1908: C     SCHWEFEL FUNCTION
1909:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1910:       DIMENSION X(*)
1911:       DO I=1,M
1912:       IF(DABS(X(I)).GT.500) THEN
1913:       CALL RANDOM(RAND)
1914:       X(I)=(RAND-0.5D00)*1000
1915:       ENDIF
1916:       ENDDO
1917:       F=0.D00
1918:       DO I=1,M
1919:       F=F+ X(I)*DSIN(DSQRT(DABS(X(I))))
1920:       ENDDO
1921:       F=418.9829D00*M - F
1922:       RETURN
1923:       END
1924: C     ----------------------------------------------------------------
1925:       SUBROUTINE SHUBERT(M,X,F)
1926: C     SHUBERT FUNCTION
1927:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1928:       DIMENSION X(*)
1929:          DO I=1,M
1930:          IF(DABS(X(I)).GT.10.D00) THEN
1931:          CALL RANDOM(RAND)
1932:          X(I)=(RAND-0.5D00)*20
1933:          ENDIF
1934:          ENDDO
1935:       F1=0.D00
1936:       F2=0.D00
1937:       DO I=1,5
1938:       F1=F1+I*DCOS((I+1.D00)*X(1)+I)
1939:       F2=F2+I*DCOS((I+1.D00)*X(2)+I)
1940:       ENDDO
1941:       F=F1*F2
1942:       RETURN
1943:       END
```

```
1944: C     ------------------------------------------------------------------
1945:       SUBROUTINE DIXPRICE(M,X,F)
1946: C     DIXON & PRICE FUNCTION
1947:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1948:       DIMENSION X(*)
1949:        DO I=1,M
1950:          IF(DABS(X(I)).GT.10.D00) THEN
1951:          CALL RANDOM(RAND)
1952:          X(I)=(RAND-0.5D00)*20
1953:          ENDIF
1954:        ENDDO
1955:       F=0.D00
1956:       DO I=2, M
1957:       F=F + I*(2*X(I)**2-X(I-1))**2
1958:       ENDDO
1959:       F=F+(X(1)-1.D00)**2
1960:       RETURN
1961:       END
1962: C     ------------------------------------------------------------------
1963:       SUBROUTINE SHEKEL(M,X,F)
1964: C     SHEKEL FUNCTION FOR TEST OF GLOBAL OPTIMIZATION METHODS
1965:       PARAMETER(NROW=10,NCOL=4, NR=9)! NR MAY BE 2 TO 10
1966:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1967:       DIMENSION A(NROW,NCOL),C(NROW),X(*)
1968:       DATA ((A(I,J),J=1,NCOL),I=1,NROW)/4.,4.,4.,4.,1.,1.,1.,1.,8.,8.,
1969:      & 8.,8.,6.,6.,6.,6.,3.,7.,3.,7.,2.,9.,2.,9.,5.,5.,3.,3.,8.,1.,8.,
1970:      & 1.,6.,2.,6.,2.,7.,3.6D00,7.,3.6D00/
1971:       DATA (C(I),I=1,NROW)/0.1D00,0.2D00,0.2D00,0.4D00,0.4D00,0.6D00,
1972:      & 0.3D00,0.7D00,0.5D00,0.5D00/
1973:       F=0.D00
1974:       DO I=1,NR
1975:         S=0.D00
1976:         DO J=1,M
1977:         S=S+(X(J)-A(I,J))**2
1978:         ENDDO
1979:       F=F-1.D00/(S+C(I))
1980:       ENDDO
1981:       RETURN
1982:       END
1983: C     ------------------------------------------------------------------
1984:       SUBROUTINE PAVIANI(M,X,F)
1985: C     PAVIANI FUNCTION : MIN F(9.3502,...,9.3502)=45.77847 APPROX
1986: C     IN THE DOMAIN 2<= X(I) <= 10 FOR I=1,2,...,10.
1987:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1988:       DIMENSION X(*)
1989:       DO I=1,M
1990:       IF(X(I).LE.2.D00.OR.X(I).GE.10.D00) THEN
1991:       CALL RANDOM(RAND)
1992:       X(I)=RAND*8+2.D00
1993:       ENDIF
1994:       ENDDO
1995:       F1=0.D00
1996:       F2=1.D00
1997:       DO I=1,M
1998:       F1=F1+ DLOG(X(I)-2.D00)**2+DLOG(10.D00-X(I))**2
1999:       F2=F2*X(I)
2000:       ENDDO
2001:       F=F1-F2**0.2
2002:       RETURN
2003:       END
2004: C     ------------------------------------------------------------------
2005:       SUBROUTINE BRANIN1(M,X,F)
2006: C     BRANIN FUNCTION #1  MIN F (1, 0) = 0
2007:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2008:       DIMENSION X(*)
2009:       PI=4*DATAN(1.D00)
2010:       DO I=1,M
```

```
2011:       IF(DABS(X(I)).GT.10.D00) THEN
2012:       CALL RANDOM(RAND)
2013:       X(I)=(RAND-0.5D00)*20
2014:       ENDIF
2015:       ENDDO
2016:       F=(1.D00-2*X(2)+DSIN(4*PI*X(2))/2.D00-X(1))**2+(X(2)-
2017:      & DSIN(2*PI*X(1))/2.D00)**2
2018:       RETURN
2019:       END
2020: C     ------------------------------------------------------------------
2021:       SUBROUTINE BRANIN2(M,X,F)
2022: C     BRANIN FUNCTION #2 MIN F (3.1416, 2.25)= 0.397887 APPROX
2023:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2024:       DIMENSION X(*)
2025:       PI=4*DATAN(1.D00)
2026:       IF(X(1).LT.-5.D00 .OR. X(1).GT.10.D00) THEN
2027:       CALL RANDOM(RAND)
2028:       X(1)=RAND*15-5.D00
2029:       ENDIF
2030:       IF(X(2).LT.0.D00 .OR. X(2).GT.15.D00) THEN
2031:       CALL RANDOM(RAND)
2032:       X(2)=RAND*15
2033:       ENDIF
2034:       F=(X(2)-5.D00*X(1)**2/(4*PI**2)+5*X(1)/PI-6.D00)**2 +
2035:      &    10*(1.D00-1.D00/(8*PI))*DCOS(X(1))+10.D00
2036:       RETURN
2037:       END
2038: C     ------------------------------------------------------------------
2039:       SUBROUTINE BOHACHEVSKY1(M,X,F)
2040: C     BOHACHEVSKY FUNCTION #1 : MIN F (0, 0) = 0
2041:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2042:       DIMENSION X(*)
2043:       PI=4*DATAN(1.D00)
2044:       DO I=1,M
2045:       IF(DABS(X(I)).GT.100.D00) THEN
2046:       CALL RANDOM(RAND)
2047:       X(I)=(RAND-0.5D00)*200
2048:       ENDIF
2049:       ENDDO
2050:       F=X(1)**2+2*X(2)**2-0.3D00*DCOS(3*PI*X(1))-0.4D00*DCOS(4*PI*X(2))
2051:      & +0.7D00
2052:       RETURN
2053:       END
2054: C     ------------------------------------------------------------------
2055:       SUBROUTINE BOHACHEVSKY2(M,X,F)
2056: C     BOHACHEVSKY FUNCTION #2  : MIN F (0, 0) = 0
2057:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2058:       DIMENSION X(*)
2059:       PI=4*DATAN(1.D00)
2060:       DO I=1,M
2061:       IF(DABS(X(I)).GT.100.D00) THEN
2062:       CALL RANDOM(RAND)
2063:       X(I)=(RAND-0.5D00)*200
2064:       ENDIF
2065:       ENDDO
2066:       F=X(1)**2+2*X(2)**2-0.3D00*DCOS(3*PI*X(1))*DCOS(4*PI*X(2))+0.3D00
2067:       RETURN
2068:       END
2069: C     ------------------------------------------------------------------
2070:       SUBROUTINE BOHACHEVSKY3(M,X,F)
2071: C     BOHACHEVSKY FUNCTION #3  : MIN F (0, 0) = 0
2072:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2073:       DIMENSION X(*)
2074:       PI=4*DATAN(1.D00)
2075:       DO I=1,M
2076:       IF(DABS(X(I)).GT.100.D00) THEN
2077:       CALL RANDOM(RAND)
```

```
2078:       X(I)=(RAND-0.5D00)*200
2079:       ENDIF
2080:       ENDDO
2081:       F=X(1)**2+2*X(2)**2-0.3D00*DCOS(3*PI*X(1)+4*PI*X(2))+0.3D00
2082:       RETURN
2083:       END
2084: C     ------------------------------------------------------------------
2085:       SUBROUTINE EASOM(M,X,F)
2086: C     EASOM FUNCTION : 2-VARIABLES, MIN F (PI, PI) = -1.
2087:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2088:       DIMENSION X(*)
2089:       PI=4*DATAN(1.D00)
2090:       DO I=1,M
2091:       IF(DABS(X(I)).GT.100.D00) THEN
2092:       CALL RANDOM(RAND)
2093:       X(I)=(RAND-0.5D00)*200
2094:       ENDIF
2095:       ENDDO
2096:       F=-DCOS(X(1))*DCOS(X(2))*DEXP(-(X(1)-PI)**2 -(X(2)-PI)**2)
2097:       RETURN
2098:       END
2099: C     ------------------------------------------------------------------
2100:       SUBROUTINE ROSENBROCK(M,X,F)
2101: C     ROSENBROCK FUNCTION : M VARIABLE; MIN F (1, 1,...,1)=0
2102:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2103:       DIMENSION X(*)
2104:       DO I=1,M
2105:       IF(X(I).LT.-5.D00 .OR. X(I).GT.10.D00) THEN
2106:       CALL RANDOM(RAND)
2107:       X(I)=RAND*15-5.D00
2108:       ENDIF
2109:       ENDDO
2110:       F=0.D00
2111:       DO I=1,M-1
2112:       F=F+ (100.D00*(X(I+1)-X(I)**2)**2 + (X(I)-1.D00)**2)
2113:       ENDDO
2114:       RETURN
2115:       END
2116: C     ------------------------------------------------------------------
2117:       SUBROUTINE CROSSLEG(M,X,F)
2118:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2119:       DIMENSION X(*)
2120: C     CROSS-LEGGED TABLE FUNCTION ; -10<= X(I) <=10; M=2
2121: C     MIN F(0 , X ) OR F(X, 0) = -1.
2122:       PI=4*DATAN(1.D00)
2123:         DO I=1,M
2124:         IF( DABS(X(I)).GT.10.D00) THEN
2125:         CALL RANDOM(RAND)
2126:         X(I)=(RAND-0.5D00)*20
2127:         ENDIF
2128:         ENDDO
2129:       F=-(DABS(DSIN(X(1))*DSIN(X(2))*DEXP(DABS(100.D00-(DSQRT
2130:      & (X(1)**2+X(2)**2)/PI))))+1.D00)**(-.1)
2131:       RETURN
2132:       END
2133: C     ------------------------------------------------------------------
2134:       SUBROUTINE CROSS(M,X,F)
2135:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2136:       DIMENSION X(*)
2137: C     CROSS FUNCTION ; -10<= X(I) <=10; M=2;
2138: C     MIN F(A, B)=0 APPROX; A, B=1.3494 APPROX OF EITHER SIGN (+ OR -)
2139:       PI=4*DATAN(1.D00)
2140:         DO I=1,M
2141:         IF( DABS(X(I)).GT.10.D00) THEN
2142:         CALL RANDOM(RAND)
2143:         X(I)=(RAND-0.5D00)*20
2144:         ENDIF
```

```
2145:         ENDDO
2146:       F=(DABS(DSIN(X(1))*DSIN(X(2))*DEXP(DABS(100.D00-(DSQRT
2147:      & (X(1)**2+X(2)**2)/PI))))+1.D00)**(-.1)
2148:       RETURN
2149:       END
2150: C     ------------------------------------------------------------------
2151:       SUBROUTINE CROSSINTRAY(M,X,F)
2152:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2153:       DIMENSION X(*)
2154: C     CROSS IN TRAY FUNCTION ; -10<= X(I) <=10; M=2;
2155: C     MIN F(A, B)=-20626.1218 APPROX; A, B=1.3494 APPROX OF EITHER SIGN
2156:       PI=4*DATAN(1.D00)
2157:         DO I=1,M
2158:         IF( DABS(X(I)).GT.10.D00) THEN
2159:         CALL RANDOM(RAND)
2160:         X(I)=(RAND-0.5D00)*20
2161:         ENDIF
2162:         ENDDO
2163:       F=-(DABS(DSIN(X(1))*DSIN(X(2))*DEXP(DABS(100.D00-(DSQRT
2164:      & (X(1)**2+X(2)**2)/PI))))+1.D00)**(.1)
2165:       RETURN
2166:       END
2167: C     ------------------------------------------------------------------
2168:       SUBROUTINE CROWNEDCROSS(M,X,F)
2169:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2170:       DIMENSION X(*)
2171: C     CROWNED CROSS FUNCTION ; -10<= X(I) <=10; M=2; MIN F = 1
2172:       PI=4*DATAN(1.D00)
2173:         DO I=1,M
2174:         IF( DABS(X(I)).GT.10.D00) THEN
2175:         CALL RANDOM(RAND)
2176:         X(I)=(RAND-0.5D00)*20
2177:         ENDIF
2178:         ENDDO
2179:       F=(DABS(DSIN(X(1))*DSIN(X(2))*DEXP(DABS(100.D00-
2180:      & (DSQRT(X(1)**2+X(2)**2)/PI))))+1.D00)**(.1)
2181:       RETURN
2182:       END
2183: C     ------------------------------------------------------------------
2184:       SUBROUTINE TTHOLDER(M,X,F)
2185:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2186:       DIMENSION X(*)
2187: C     TEST-TUBE HOLDER FUNCTION ; -10<= X(I) <=10; M=2;
2188: C     MIN F([+/-]1.5706, 0)= -10.8723
2189:       PI=4*DATAN(1.D00)
2190:         DO I=1,M
2191:         IF( DABS(X(I)).GT.10.D00) THEN
2192:         CALL RANDOM(RAND)
2193:         X(I)=(RAND-0.5D00)*20
2194:         ENDIF
2195:         ENDDO
2196:       F=-4*DABS(DSIN(X(1))*DCOS(X(2))*DEXP(DABS(DCOS((X(1)**2+X(2)**2)/
2197:      & 200))))
2198:       RETURN
2199:       END
2200: C     ------------------------------------------------------------------
2201:       SUBROUTINE HOLDERTABLE(M,X,F)
2202:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2203:       DIMENSION X(*)
2204: C     HOLDER-TABLE FUNCTION  ; -10<= X(I) <=10; M=2;
2205: C     MIN F ([+/-] 9.64617, [+/-] 9.64617) APPROX = -26.92034 APPROX
2206:       PI=4*DATAN(1.D00)
2207:         DO I=1,M
2208:         IF( DABS(X(I)).GT.10.D00) THEN
2209:         CALL RANDOM(RAND)
2210:         X(I)=(RAND-0.5D00)*20
2211:         ENDIF
```

```
2212:         ENDDO
2213:        F=-DABS(DCOS(X(1))*DCOS(X(2))*DEXP(DABS(1.D00-(DSQRT(X(1)**2+
2214:       & X(2)**2)/PI))))
2215:        RETURN
2216:        END
2217: C     ------------------------------------------------------------------
2218:        SUBROUTINE CARROMTABLE(M,X,F)
2219:        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2220:        DIMENSION X(*)
2221: C      CARROM-TABLE FUNCTION  ; -10<= X(I) <=10; M=2;
2222: C      MIN F ([+/-] 9.64617, [+/-] 9.64617) APPROX = -24.15682 APPROX
2223:        PI=4*DATAN(1.D00)
2224:         DO I=1,M
2225:         IF( DABS(X(I)).GT.10.D00) THEN
2226:         CALL RANDOM(RAND)
2227:         X(I)=(RAND-0.5D00)*20
2228:         ENDIF
2229:         ENDDO
2230:        F=-1.D00/30*(DCOS(X(1))*DCOS(X(2))*DEXP(DABS(1.D00-
2231:       & (DSQRT(X(1)**2 + X(2)**2)/PI))))**2
2232:        RETURN
2233:        END
2234: C     ------------------------------------------------------------------
2235:        SUBROUTINE PENHOLDER(M,X,F)
2236:        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2237:        DIMENSION X(*)
2238: C      PENHOLDER FUNCTION  ; -11<= X(I) <=11; M=2;
2239: C      MIN F ([+/-] 9.64617, [+/-] 9.64617) APPROX = -0.963535 APPROX
2240:        PI=4*DATAN(1.D00)
2241:         DO I=1,M
2242:         IF( DABS(X(I)).GT.11.D00) THEN
2243:         CALL RANDOM(RAND)
2244:         X(I)=(RAND-0.5D00)*22
2245:         ENDIF
2246:         ENDDO
2247:        F=-DEXP(-(DABS(DCOS(X(1))*DCOS(X(2))*DEXP(DABS(1.D0-(DSQRT
2248:       & (X(1)**2+X(2)**2)/PI))))**(-1)))
2249:        RETURN
2250:        END
2251: C     ------------------------------------------------------------------
2252:        SUBROUTINE BIRD(M,X,F)
2253:        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2254:        DIMENSION X(*)
2255: C      BIRD FUNCTION ; -2PI<= X(I) <=2PI; M=2;
2256: C      MIN F (4.70104, 3.15294) APPROX = -106.764537 APPROX  OR
2257: C      MIN F (-1.58214, -3.13024) APPROX = -106.764537 APPROX
2258:        PI=4*DATAN(1.D00)
2259:         DO I=1,M
2260:         IF( DABS(X(I)).GT.2*PI) THEN
2261:         CALL RANDOM(RAND)
2262:         X(I)=(RAND-0.5D00)*4*PI
2263:         ENDIF
2264:         ENDDO
2265:        F=(DSIN(X(1))*DEXP((1.D00-DCOS(X(2)))**2) +
2266:       & DCOS(X(2))*DEXP((1.D00-DSIN(X(1)))**2))+(X(1)-X(2))**2
2267:        RETURN
2268:        END
2269: C     ------------------------------------------------------------------
2270:        SUBROUTINE CHICHINADZE(M,X,F)
2271:        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2272:        DIMENSION X(*)
2273: C      CHICHINADZE FUNCTION :  -30 <=X(I)<= 30;  M=2
2274: C      MIN F (5.901329, 0.5) = -43.3158621
2275:         PI=4*DATAN(1.D00)
2276:         DO I=1,M
2277:         IF( DABS(X(I)).GT.30) THEN
2278:         CALL RANDOM(RAND)
```

```
2279:         X(I)=(RAND-0.5D00)*60
2280:         ENDIF
2281:         ENDDO
2282:       F=X(1)**2-12*X(1)+11.D00+10*DCOS(PI*X(1)/2)+8*DSIN(5*PI*X(1))-
2283:      & (1.D00/DSQRT(5.D00))*DEXP(-(X(2)-0.5D00)**2/2)
2284:       RETURN
2285:       END
2286: C     ------------------------------------------------------------------
2287:       SUBROUTINE MCCORMICK(M,X,F)
2288:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2289:       DIMENSION X(*)
2290: C     MCCORMICK FUNCTION : -1.5<= X(1)<=4; -3<=X(2)<=4 ; M=2
2291: C     MIN F (-0.54719755, -1.54719755) = -1.913223 APPROX
2292:         IF(X(1).LT. -1.5D00 .OR. X(1) .GT. 4.D00) THEN
2293:         CALL RANDOM(RAND)
2294:         X(1)=RAND*5.5D00-1.5D00
2295:         ENDIF
2296:         IF(X(2).LT. -3.D00 .OR. X(2) .GT. 4.D00) THEN
2297:         CALL RANDOM(RAND)
2298:         X(2)=RAND*7.D00-3.D00
2299:         ENDIF
2300:       F=DSIN(X(1)+X(2))+(X(1)-X(2))**2-1.5*X(1)+2.5*X(2)+1.D00
2301:       RETURN
2302:       END
2303: C     ------------------------------------------------------------------
2304:       SUBROUTINE FENTONEASON(M,X,F)
2305:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2306:       DIMENSION X(*)
2307: C     FENTON & EASON FUNCTION  FMIN(1.74345, -2.029695) = 1.744152
2308:       DO I=1,M
2309:       IF(DABS(X(I)).GT.100.D00) THEN
2310:       CALL RANDOM(RAND)
2311:       X(I)=(RAND-0.5D00)*200
2312:       ENDIF
2313:       ENDDO
2314:       F=1.2D00+0.1*X(1)**2 +(0.1D00+0.1*X(2)**2)/X(1)**2+
2315:      & (.1*X(1)**2*X(2)**2+10.D00)/((X(1)*X(2))**4)
2316:       RETURN
2317:       END
2318: C     ------------------------------------------------------------------
2319:       SUBROUTINE WOOD(M,X,F)
2320:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2321:       COMMON /RNDM/IU,IV
2322:       INTEGER IU,IV
2323:       DIMENSION X(*)
2324: C     WOOD FUNCTION:FMIN(0.443546,-0.194607,1.466077,2.15115)=1.09485393
2325:       DO I=1,M
2326:       IF(DABS(X(I)).GT.5.D00) THEN
2327:       CALL RANDOM(RAND)
2328:       X(I)=(RAND-0.5D00)*10
2329:       ENDIF
2330:       ENDDO
2331:       F1=100*(X(2)+X(1)**2)**2 + (1.D0-X(1))**2 +90*(X(4)-X(3)**2)**2
2332:       F2=(1.D0-X(3))**2 +10.1*((X(2)-1.D0)**2+(X(4)-1.D0)**2)
2333:       F3=19.8*(X(2)-1.D0)*(X(4)-1.D0)
2334:       F=F1+F2+F3
2335:       RETURN
2336:       END
2337: C     ------------------------------------------------------------------
2338:       SUBROUTINE GLANKWAHMDEE(M,X,F)
2339:       PARAMETER (N=5)
2340:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2341:       COMMON /RNDM/IU,IV
2342:       INTEGER IU,IV
2343:       DIMENSION X(*),A(N,N), B(N)
2344: C     ------------------ GLANKWAHMDEE FUNCTION -------------------------
2345: C     GLANKWAHMDEE A, LIEBMAN JS AND HOGG GL (1979) "UNCONSTRAINED
```

```
2346: C     DISCRETE NONLINEAR PROGRAMMING. ENGINEERING OPTIMIZATION 4: 95-107
2347: C     PARSOPOULOS, KE AND VRAHATIS, MN (2002) RECENT APPROACHES TO
2348: C     GLOBAL OPTIMIZATION PROBLEMS THROUGH PARTICLE SWARM OPTIMIZATION,
2349: C     NATURAL COMPUTING 1: 235-306, 2002.  REPORT THE BEST OBTAINED
2350: C     FMIN (0,12,23,17,6)= -737 OR  MIN F(0, 11,22,16,6)= -737
2351: C     WE GET FMIN(-.232, 11.489, 22.273, 16.540, 6.115) = -739.822991
2352: C     ------------------------------------------------------------------
2353:       DATA ((A(I,J),J=1,N),I=1,N) /35,-20,-10,32,-10,-20, 40,-6,-31,32,
2354:      & -10,-6,11,-6,-10,32,-31,-6,38,-20,-10,32,-10,-20,31/
2355:       DATA (B(J),J=1,N) /-15,-27,-36,-18,-12/
2356: C     ------------------------------------------------------------------
2357:       DO I=1,M
2358:       IF(DABS(X(I)).GT.100.D00) THEN
2359:       CALL RANDOM(RAND)
2360:       X(I)=(RAND-0.5D00)*200
2361:       ENDIF
2362:       ENDDO
2363:       F=0.D0
2364:       DO J=1,M
2365:       F=F+B(J)*X(J)
2366:       ENDDO
2367:       DO I=1,M
2368:       C=0.D0
2369:       DO J=1,M
2370:       C=C+X(J)*A(J,I)
2371:       ENDDO
2372:       F=F+C*X(I)
2373:       ENDDO
2374:       RETURN
2375:       END
2376: C     ------------------------------------------------------------------
2377:       SUBROUTINE FLETCHER(M,X,F)
2378: C     FLETCHER-POWELL FUNCTION, M <= 10, ELSE IT IS VERY MUCH SLOW
2379: C     SOLUTION: MIN F = 0 FOR X=(C1, C2, C3,...,CM)
2380:       PARAMETER(N=10) ! FOR DIMENSION OF DIFFERENT MATRICES AND VECTORS
2381:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2382:       COMMON /RNDM/IU,IV
2383:       INTEGER IU,IV
2384:       DIMENSION X(*),A(N,N),B(N,N),AA(N),BB(N),AL(N),C(N),C1(N)
2385:       PI=4*DATAN(1.D00)
2386: C     GENERATE A(I,J) AND B(I,J) BETWEEN (-100, 100) RANDOMLY.
2387: C     C(I) = BETWEEN (-PI, PI) IS EITHER GIVEN OR RANDOMLY GENERATED.
2388: C     DATA (C(I),I=1,10)/1,2,3,-3,-2,-1,0,1,2,3/ ! BETWEEN -PI AND PI
2389:       DATA (C1(I),I=1,N)/-3,-3.02,-3.01,1,1.03,1.02,1.03,-.08,.001,3/
2390: C     DATA (C1(I),I=1,N)/0,0,0,0,0,0,0,0,0,0/ ! ANOTHER EXAMPLE C1 = 0
2391:       NC=0  ! DEFINE NC HERE 0 OR 1 OR 2
2392: C     IF NC=0, C1 FROM DATA IS USED (THAT IS FIXED C);
2393: C     IF NC=1, C IS MADE FROM C1 BY ADDING RANDOM PART - THAT IS C=C1+R
2394: C     IF NC=2 THEN C IS PURELY RANDOM THAT IS C= 0 + R
2395: C     IN ANY CASE C LIES BETWEEN -PI AND PI.
2396: C     ------------------------------------------------------------------
2397: C     FIND THE MAX MAGNITUDE ELEMENT IN C1 VECTOR (UPTO M ELEMENTS)
2398:       CMAX=DABS(C1(1))
2399:       DO J=2,M
2400:       IF(DABS(C1(J)).GT.CMAX) CMAX=DABS(C1(J))
2401:       ENDDO
2402:       RANGE=PI-CMAX
2403: C     ------------------------------------------------------------------
2404:       DO J=1,M
2405:       DO I=1,M
2406:       CALL RANDOM(RAND)
2407:       A(I,J)=(RAND-0.5D00)*200.D00
2408:       CALL RANDOM(RAND)
2409:       B(I,J)=(RAND-0.5D00)*200.D00
2410:       ENDDO
2411:       IF(NC.EQ.0) AL(J)=C1(J) ! FIXED OR NON-STOCHASTIC C
2412:       IF(NC.EQ.1) THEN
```

```
2413:       CALL RANDOM(RAND)
2414:       AL(J)=C1(J)+(RAND-0.5D0)*2*RANGE ! A PART FIXED, OTHER STOCHASTIC
2415:       ENDIF
2416:       IF(NC.EQ.2) THEN
2417:       CALL RANDOM(RAND)
2418:       AL(J)=(RAND-0.5D00)*2*PI ! PURELY STOCHASTIC
2419:       ENDIF
2420:       ENDDO
2421:       DO I=1,M
2422:       AA(I)=0.D00
2423:       DO J=1,M
2424:       AA(I)=AA(I)+A(I,J)*DSIN(AL(J))+B(I,J)*DCOS(AL(J))
2425:       ENDDO
2426:       ENDDO
2427: C     ------------------------------------------------------------------
2428:       DO I=1,M
2429:       IF(DABS(X(I)).GT.PI) THEN
2430:       CALL RANDOM(RAND)
2431:       X(I)=(RAND-0.5D00)*2*PI
2432:       ENDIF
2433:       ENDDO
2434:       DO I=1,M
2435:       BB(I)=0.D00
2436:       DO J=1,M
2437:       BB(I)=BB(I)+A(I,J)*DSIN(X(J))+B(I,J)*DCOS(X(J))
2438:       ENDDO
2439:       ENDDO
2440:       F=0.D00
2441:       DO I=1,M
2442:       F=F+(AA(I)-BB(I))**2
2443:       ENDDO
2444:       RETURN
2445:       END
2446: C-----------------------------------------------------------------------
2447:       SUBROUTINE POWELL(M,X,F)
2448:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2449:       DIMENSION X(*)
2450: C     POWELL FUNCTION  ; -4<= X(I) <=5; M=A MULTIPLE OF 4;
2451: C     MIN F  =  0.0
2452:         DO I=1,M
2453:         IF(X(I).LT.-4.D00 .OR. X(I).GT.5.D00) THEN
2454:         CALL RANDOM(RAND)
2455:         X(I)=(RAND-0.5D00)*9+.5D00
2456:         ENDIF
2457:         ENDDO
2458:       M4=M/4
2459:       F=0.D00
2460:       DO I=1,M4
2461:       J=4*I
2462:       F=F+(X(J-3)+10*X(J-2))**2+5*(X(J-1)-X(J))**2+(X(J-2)-X(J-1))**4 +
2463:      & 10*(X(J-3)-X(J))**4
2464:       ENDDO
2465:       RETURN
2466:       END
2467: C-----------------------------------------------------------------------
2468:       SUBROUTINE HARTMANN(M,X,F)
2469:       PARAMETER (N=4)
2470:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2471:       DIMENSION X(*),P(N,3),A(N,3),C(N)
2472: C     HARTMANN FUNCTION
2473: C     MIN F  =  -3.86278 APPROX  : 0 < X < 1.
2474:       DATA ((P(I,J),J=1,3),I=1,4) /0.6890,0.1170,0.2673,0.4699,0.4387,
2475:      &  0.7470,0.1091,0.8732,0.5547,0.0381,0.5743,0.8828/
2476:       DATA ((A(I,J),J=1,3),I=1,4) /3.0,10.0,30.0,0.1,10.0,35.0,3.0,
2477:      &  10.0,30.0,0.1,10.0,35.0/
2478:       DATA (C(J),J=1,4) /1.0,1.2,3.0,3.2/
2479:         DO I=1,M
```

```
2480:         IF(X(I).LE.0.D00 .OR. X(I).GE.1.D00) THEN
2481:         CALL RANDOM(RAND)
2482:         X(I)=RAND
2483:         ENDIF
2484:         ENDDO
2485:        F=0.D00
2486:        DO I=1,N
2487:        S=0.D00
2488:        DO J=1,M
2489:        S=S+A(I,J)*(X(J)-P(I,J))**2
2490:        ENDDO
2491:        F=F+C(I)*DEXP(-S)
2492:        ENDDO
2493:        F=-F
2494:        RETURN
2495:        END
2496: C-----------------------------------------------------------------------
2497:        SUBROUTINE COLVILLE(M,X,F)
2498:        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2499:        DIMENSION X(*)
2500: C      COLVILLE FUNCTION  ; -10<= X(I) <=10; M= 4;
2501: C      MINF(1,1,1,1)= 0.0
2502:         DO I=1,M
2503:         IF(X(I).LT.-10.D00 .OR. X(I).GT.10.D00) THEN
2504:         CALL RANDOM(RAND)
2505:         X(I)=(RAND-0.5D00)*20
2506:         ENDIF
2507:         ENDDO
2508:        F=100*(X(1)**2-X(2))**2 + (X(1)-1.D00)**2 +(X(3)-1.D00)**2+
2509:       & 90*(X(3)**2-X(4))**2+10.1*((X(2)-1.D0)**2+(X(4)-1.D00)**2)+
2510:       & 19.8*(X(2)-1.D00)*(X(4)-1.D00)
2511:        RETURN
2512:        END
2513: C      -----------------------------------------------------------------
2514:        SUBROUTINE HIMMELBLAU(M,X,F)
2515:        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2516:        DIMENSION X(*)
2517:        NF=0    ! SET NF = 0 MULTIPLE OPTIMA NF=1 SINGLE OPTIMUM
2518:         DO I=1,M
2519:         IF(X(I).LT.-10.D00 .OR. X(I).GT.10.D00) THEN
2520:         CALL RANDOM(RAND)
2521:         X(I)=(RAND-0.5D00)*20
2522:         ENDIF
2523:         ENDDO
2524: C      HIMMELBLAU FUNCTION. IT HAS MULTIPLE (4) GLOBAL OPTIMA : MINF=0
2525: C      (3, 2); (-2.8051, 3.1313); (3.5744, -1.8481); (-3.779, -3.283)
2526:        IF(NF.EQ.0) THEN
2527:        F= (X(1)**2+X(2)-11)**2+  (X(1)+X(2)**2-7)**2
2528:        RETURN
2529:        ENDIF
2530:        IF(NF.EQ.1) THEN
2531: C      MODIFIED HIMMELBLAU FUNCTION. IT HAS ONLY ONE GLOBAL OPTIMUM
2532: C      MINF=0 AT (3,2)
2533:        F= (X(1)**2+X(2)-11)**2+(X(1)+X(2)**2-7)**2+0.1D00*((X(1)-3)**2 +
2534:       1   (X(2)-2)**2)
2535:        ENDIF
2536:        RETURN
2537:        END
2538: C      -----------------------------------------------------------------
2539:        SUBROUTINE BEALE(M,X,F)
2540:        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2541:        DIMENSION X(*)
2542:         DO I=1,M
2543:         IF(X(I).LT.-4.500 .OR. X(I).GT.4.500) THEN
2544:         CALL RANDOM(RAND)
2545:         X(I)=(RAND-0.5D00)*9
2546:         ENDIF
```

```
2547:         ENDDO
2548: C      BEALE FUNCTION  : MINF =0 AT (3, 0.5)
2549:       F1=(1.5D00-X(1)+X(1)*X(2))**2
2550:       F2=(2.25D00-X(1)+X(1)*X(2)**2)**2
2551:       F3=(2.625D00-X(1)+X(1)*X(2)**3)**2
2552:       F=F1+F2+F3
2553:       RETURN
2554:       END
2555: C     ------------------------------------------------------------------
2556:       SUBROUTINE BOOTH(M,X,F)
2557:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2558:       DIMENSION X(*)
2559:         DO I=1,M
2560:         IF(X(I).LT.-10.D00 .OR. X(I).GT.10.D00) THEN
2561:         CALL RANDOM(RAND)
2562:         X(I)=(RAND-0.5D00)*20
2563:         ENDIF
2564:         ENDDO
2565: C      BOOTH FUNCTION  MINF (1,3)=0
2566:       F=(X(1)+2*X(2)-7.0D00)**2+(2*X(1)+X(2)-5.0D00)**2
2567:       RETURN
2568:       END
2569: C     ------------------------------------------------------------------
2570:       SUBROUTINE HUMP(M,X,F)
2571:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2572:       DIMENSION X(*)
2573:         DO I=1,M
2574:         IF(X(I).LT.-5.D00 .OR. X(I).GT.5.D00) THEN
2575:         CALL RANDOM(RAND)
2576:         X(I)=(RAND-0.5D00)*10
2577:         ENDIF
2578:         ENDDO
2579: C      HUMP FUNCTION  MINF (0.0898, -0.7127) = -1.0316
2580:       F=4*X(1)**2 - 2.1D00*X(1)**4 + (X(1)**6)/3.D00 + X(1)*X(2) -
2581:      & 4*X(2)**2 + 4*X(2)**4
2582:       RETURN
2583:       END
2584: C     ------------------------------------------------------------------
2585:       SUBROUTINE MATYAS(M,X,F)
2586:       IMPLICIT DOUBLE PRECISION (A-H, O-Z)
2587:       DIMENSION X(*)
2588:         DO I=1,M
2589:         IF(X(I).LT.-10.D00 .OR. X(I).GT.10.D00) THEN
2590:         CALL RANDOM(RAND)
2591:         X(I)=(RAND-0.5D00)*20
2592:         ENDIF
2593:         ENDDO
2594: C      MATYAS FUNCTION  MIN F (0, 0) = 0
2595:       F=0.26D00*(X(1)**2 + X(2)**2) - 0.48D00*X(1)*X(2)
2596:       RETURN
2597:       END
2598: C     ------------------------------------------------------------------
2599:       SUBROUTINE MISHRA_1(M,X,F)
2600:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2601:       COMMON /RNDM/IU,IV
2602:       INTEGER IU,IV
2603:       DIMENSION X(*)
2604: C      MIN F (1, 1, ..., 1) =2
2605:       DO I=1,M
2606:       IF(X(I).LT.0.D00 .OR. X(I).GT.1.D00) THEN
2607:       CALL RANDOM(RAND)
2608:       X(I)=RAND
2609:       ENDIF
2610:       ENDDO
2611:       S=0.D00
2612:       DO I=1,M-1
2613:       S=S+X(I)
```

```
      ENDDO
      X(M)=(M-S)
      F=(1.D00+X(M))**X(M)
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE MISHRA_2(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*)
C     MIN F (1, 1, ..., 1) =2
      DO I=1,M
      IF(X(I).LT.0.D00 .OR. X(I).GT.1.D00) THEN
      CALL RANDOM(RAND)
      X(I)=RAND
      ENDIF
      ENDDO
      S=0.D00
      DO I=1,M-1
      S=S+(X(I)+X(I+1))/2.D00
      ENDDO
      X(M)=(M-S)
      F=(1.D00+X(M))**X(M)
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE ZAKHAROV(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*)
C     ZAKHAROV FUNCTIONMIN MINF = (0, 0, ..., 0) =0
      DO I=1,M
      IF(X(I).LT.-5.D00 .OR. X(I).GT.10.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-.5D0)*15 + 2.5D0
      ENDIF
      ENDDO
      F1=0.D00
      F2=0.D00
      DO I=1, M
      F1=F1+ X(I)**2
      F2=F2 + I*X(I)/2.D00
      ENDDO
      F=F1+F2**2+F2**4
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE MULTIMOD(M,X,F)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION X(*)
C     MULTIMODAL  FUNCTION  MINF = (0,0)= -1  : M=2
      DO I=1,M
      IF(X(I).LT.-10.D00 .OR. X(I).GT.10.D00) THEN
      CALL RANDOM(RAND)
      X(I)=(RAND-.5D0)*20
      ENDIF
      ENDDO
C     A TYPICAL MULTIMODAL FUNCTION
      F=-DCOS(X(1))*DCOS(X(2))*DEXP(-(X(1)**2 + X(2)**2)**2/4.D00)
      RETURN
      END
C     ------------------------------------------------------------------
      SUBROUTINE NONLIN(M,X,F)
```

```
2681:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2682:       COMMON /RNDM/IU,IV
2683:       INTEGER IU,IV
2684:       DIMENSION X(*)
2685: C     NONLINEAR  MULTIMODAL FUNCTION  MINF = 0
2686:       DO I=1,M
2687:       IF(X(I).LT.-10.D00 .OR. X(I).GT.10.D00) THEN
2688:       CALL RANDOM(RAND)
2689:       X(I)=(RAND-.5D0)*20
2690:       ENDIF
2691:       ENDDO
2692:       F=0.D0
2693:       DO I=2,M
2694:       F=F+DCOS(DABS(X(I)-X(I-1)) / DABS(X(I-1)+X(I)))
2695:       ENDDO
2696:       F=F+(M-1.D00)
2697: C     IF 0.001*X(1) IS ADDED TO F, IT BECOMES UNIMODAL
2698: C     F=F+0.001*X(1)
2699:       RETURN
2700:       END
2701: C     ------------------------------------------------------------------
2702:       SUBROUTINE QUADRATIC(M,X,F)
2703:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2704:       COMMON /RNDM/IU,IV
2705:       INTEGER IU,IV
2706:       DIMENSION X(*)
2707: C     QUADRATIC FUNCTION  MINF (0.19388, 0.48513) = -3873.7243 (M=2)
2708:       DO I=1,M
2709:       IF(X(I).LT.-10.D00 .OR. X(I).GT.10.D00) THEN
2710:       CALL RANDOM(RAND)
2711:       X(I)=(RAND-.5D0)*200
2712:       ENDIF
2713:       ENDDO
2714:       F=-3803.84-138.08*X(1)-232.92*X(2)+128.08*X(1)**2+203.64*X(2)**2+
2715:      & 182.25*X(1)*X(2)
2716:       RETURN
2717:       END
2718: C     ------------------------------------------------------------------
2719:       SUBROUTINE TRIGON(M,X,F)
2720:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2721:       COMMON /RNDM/IU,IV
2722:       INTEGER IU,IV
2723:       DIMENSION X(*)
2724: C     TRIGON FUNCTION F MIN (0, 0, 0,..., 0) OR (PI, 0, 0, ...,0) =0
2725:       PI=4*DATAN(1.D00)
2726:         DO I=1,M
2727:         IF(X(I).LT.0.D00 .OR. X(I).GT.PI) THEN
2728:         CALL RANDOM(RAND)
2729:         X(I)=RAND*PI
2730:         ENDIF
2731:         ENDDO
2732:       F=0.D00
2733:       DO I=2,M
2734:       F=F+(DCOS(I+0.D00)*DSIN(X(I)-X(I-1))**2 +
2735:      & (I-1.D00)*(1.D0-DCOS(X(I))))**2
2736:       ENDDO
2737:       RETURN
2738:       END
2739: C     ------------------------------------------------------------------
2740:       SUBROUTINE WAVY1(M,X,F)
2741:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2742:       COMMON /RNDM/IU,IV
2743:       INTEGER IU,IV
2744:       DIMENSION X(*)
2745: C     WAVY-1 FUNCTION F MIN (24, 24,...,24)=0
2746:         DO I=1,M
2747:         IF(X(I).LT.-100.D00 .OR. X(I).GT.100.D0) THEN
```

```
2748:         CALL RANDOM(RAND)
2749:         X(I)=(RAND-.5D0)*200
2750:         ENDIF
2751:         ENDDO
2752:       F=0.D00
2753:       DO I=1,M
2754:       F=F+DABS(2*(X(I)-24.D0)+(X(I)-24.D0)*DSIN(X(I)-24.D0))
2755:       ENDDO
2756:       RETURN
2757:       END
2758: C     ------------------------------------------------------------------
2759:       SUBROUTINE WAVY2(M,X,F)
2760:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2761:       COMMON /RNDM/IU,IV
2762:       INTEGER IU,IV
2763:       DIMENSION X(*)
2764: C     WAVY-2 FUNCTION F MIN (0,0,...,0)=0
2765:         DO I=1,M
2766:         IF(X(I).LT.-100.D00 .OR. X(I).GT.100.D0) THEN
2767:         CALL RANDOM(RAND)
2768:         X(I)=(RAND-.5D0)*200
2769:         ENDIF
2770:         ENDDO
2771:       F=0.D00
2772:       DO I=1,M
2773:       F=F+X(I)**6*(2.D0+DSIN(1.D0/X(I)))
2774:       ENDDO
2775:       RETURN
2776:       END
2777: C     ------------------------------------------------------------------
2778:       SUBROUTINE TREFETHEN(M,X,F)
2779:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2780:       COMMON /RNDM/IU,IV
2781:       INTEGER IU,IV
2782:       DIMENSION X(*)
2783: C     TREFETHEN FUNCTION:FMIN(-0.02440307923,0.2106124261)=-3.3068686475
2784:         DO I=1,M
2785:         IF(X(I).LT.-10.D00 .OR. X(I).GT.10.D0) THEN
2786:         CALL RANDOM(RAND)
2787:         X(I)=(RAND-.5D0)*20
2788:         ENDIF
2789:         ENDDO
2790:       F=0.D00
2791: C     THE JAN-FEB 2002 ISSUE OF SIAM NEWS CONTAINED THE FOLLOWING
2792: C     CHALLENGE FROM L. N. TREFETHEN OF OXFORD UNIVERSITY.
2793: C     FMIN = F(-0.02440307923, 0.2106124261)= -3.3068686475
2794:       F=DEXP(DSIN(50*X(1))) + DSIN(60*DEXP(X(2))) + DSIN(70*DSIN(X(1)))
2795:      &  + DSIN(DSIN(80*X(2))) - DSIN(10*(X(1)+X(2))) +
2796:      & 1.D0/4*(X(1)**2 + X(2)**2)
2797:       RETURN
2798:       END
2799: C     ------------------------------------------------------------------
2800:       SUBROUTINE ALPINE(M,X,F)
2801:       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
2802:       COMMON /RNDM/IU,IV
2803:       INTEGER IU,IV
2804:       DIMENSION X(*)
2805: C     ALPINE FUNCTION F MIN (7.917, 7.917, ..., 7.917)= -2.808131 APPROX
2806: C     THE DOMAIN : X IN [-10, 10].
2807: C     NOTE: IN THE ORIGINAL FUNCTION MIN F = -(2.808131)^M
2808:         DO I=1,M
2809:         IF(X(I).LT.0.D00 .OR. X(I).GT.10.D0) THEN
2810:         CALL RANDOM(RAND)
2811:         X(I)=RAND*10.D0
2812:         ENDIF
2813:         ENDDO
2814:       F1=1.D0
```

```
2815:       F2=1.D0
2816:       DO I=1,M
2817:       F1=F1*DSIN(X(I))
2818:       F2=F2*X(I)
2819:       ENDDO
2820:       F=-DEXP(DLOG(F1*DSQRT(F2))/M)
2821:       RETURN
2822:       END
2823:
```