Some new test functions for global optimization and performance of repulsive particle swarm method

Sudhanshu Mishra

North-Eastern Hill University, Shillong (India)

23. August 2006

Some New Test Functions for Global Optimization and Performance of Repulsive Particle Swarm Method

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

I. Introduction: In this paper we introduce some new multi-modal test functions to assess the performance of global optimization methods. These functions have been selected partly because several of them are aesthetically appealing and partly because a few of them are really difficult to optimize. We also propose to optimize some important benchmark functions already in vogue. Each function has been graphically presented to appreciate its geometrical appearance. To optimize these functions we have used the Repulsive Particle Swarm (RPS) method, with wider local search abilities and randomized neighbourhood topology.

II. The Particle Swarm Method of Global Optimization: This method is an instance of successful application of the philosophy of Simon's *bounded rationality* and decentralized decision-making to solve the global optimization problems (Simon, 1982; Bauer, 2002; Fleischer, 2005). As it is well known, the problems of the existence of global order, its integrity, stability, efficiency, etc. have been long standing. The laws of development of institutions have been sought in this order. Newton, Hobbes, Adam Smith and Locke visualized the global system arising out of individual actions. In particular, Adam smith (1759) postulated the role of *invisible hand* in establishing the harmony that led to the said global order. The neo-classical economists applied the tools of equilibrium analysis to show how this grand synthesis and order is established while each individual is selfish. The postulate of perfect competition was felt to be a necessary one in demonstrating that. Yet, Alfred Marshall limited himself to partial equilibrium analysis and, thus, indirectly allowed for the role of *invisible hand* (while general equilibrium economists hoped that the establishment of order can be explained by their approach). Thorstein Veblen (1899) never believed in the mechanistic view and pleaded for economics as an evolutionary science. F. A. Hayek (1944) believed in a similar philosophy and believed that locally optimal decisions give rise to the global order and efficiency. Later, Herbert Simon (1982) postulated the '*bounded rationality*' hypothesis and argued that the hypothesis of perfect competition is not necessary for explaining the emergent harmony and order at the global level. Elsewhere, I. Prigogine (1984) demonstrated how the global 'order' emerges from chaos at the local level.

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 (Sumper, 2006). 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. The Particle Swarm method of optimization mimics this 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; Huang et al., 2006).

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 perturbation of v may be needed. We have modified the traditional RPS method by endowing stronger (wider) local search ability to each particle and the neighbourhood topology to each particle is randomized.

III. The New Test Functions: We used RPS method for a fairly large number of established test problems (Mishra, 2006 (c) reports about 30 benchmark functions). Here we introduce the new functions and the results obtained by the RPS program (appended). These new functions are as follows.

1. Test tube holder function (a): This multi-modal function is defined as follows. We obtain $x^* \simeq -10.8723$ in the domain $x_i \in [-10,\ 10],\ i = 1, 2$.

$$f(x) = -4\left|\sin(x_1)\cos(x_2)\,\mathrm{e}^{|\cos((x_1^2+x_2^2)/200)|}\right|.$$

2. Test tube holder function (b): This multi-modal function is defined as follows. We obtain $x^* \simeq -10.8723$ in the domain $x_1 \in [-9.5,\ 9.4],\ x_2 \in [-10.9,\ 10.9]$.

$$f(x) = -4\left|\sin(x_1)\cos(x_2)\,\mathrm{e}^{|\cos((x_1^2+x_2^2)/200)|}\right|.$$

3. Holder table function: This 'tabular holder' function has multiple local minima with four global minima at $f(x^*) \simeq 26.92$. This function is given as:

$$f(x) = -\left|\cos(x_1)\cos(x_2)\,\mathrm{e}^{|1-[(x_1^2+x_2^2)^{0.5}]/\pi|}\right|$$

4. Carrom table function: This function has multiple local minima with four global minima at $f(x^*) \simeq 24.1568155$ in the search domain $x_i \in [-10,\ 10],\ i = 1, 2$. This function is given as:

$$f(x) = -\left[\left\{\cos(x_1)\cos(x_2)\,\mathrm{e}^{|1-[(x_1^2+x_2^2)^{0.5}]/\pi|}\right\}^2\right]/30$$

5. Cross in tray function: This function has multiple local minima with the global minima at $f(x^*) \simeq -2.06261218$ in the search domain $x_i \in [-10,\ 10],\ i = 1, 2$. This function is given as:

$$f(x) = -0.0001\left[\left|\sin(x_1)\sin(x_2)e^{\left|100-[(x_1^2+x_2^2)^{0.5}]/\pi\right|}\right|+1\right]^{0.1}$$

6. Crowned cross function: This function is the negative form of the cross in tray function. It has $f(x^*) \simeq 0$ in the search domain $x_i \in [-10,\ 10],\ i = 1, 2$. It is a difficult function to optimize. The minimal value obtained by us is approximately 0.1. This function is given as:

$$f(x) = 0.0001\left[\left|\sin(x_1)\sin(x_2)e^{\left|100-[(x_1^2+x_2^2)^{0.5}]/\pi\right|}\right|+1\right]^{0.1}$$

7. Cross function: This is a multi-modal function with $f(x^*) \simeq 0$. It is given as

$$f(x) = \left[\left| \sin(x_1)\sin(x_2) e^{\left|100 - [(x_1^2 + x_2^2)^{0.5}]/\pi\right|} \right| + 1 \right]^{-0.1}$$

8. Cross-leg table function: This function is the negative form of the cross function and may also be called the 'inverted cross' function. It has $f(x^*) \simeq -1$ in the search domain $x_i \in [-10,\ 10],\ i = 1,2$. It is a difficult function to optimize. We have failed to optimize this function. This function is given as:

$$f(x) = -\left[\left| \sin(x_1)\sin(x_2) e^{\left|100 - [(x_1^2 + x_2^2)^{0.5}]/\pi\right|} \right| + 1 \right]^{-0.1}$$

9. Pen holder function: This is a multi-modal function with $f(x^*) \simeq -0.96354$ in the search domain $x_i \subseteq [-11, 11)$, given as

$$f(x) = -\exp\left[-\left| \cos(x_1)\cos(x_2) e^{\left|1 - [(x_1^2 + x_2^2)^{0.5}/\pi]\right|} \right|^{-1} \right]$$

10. Bird function: This is a bi-modal function with $f(x^*) \simeq -106.764537$ in the search domain $x_i \in [-2\pi,\ 2\pi];\ i = 1,2$ given as

$$f(x) = \sin(x_1) e^{[(1-\cos(x_2))^2]} + \cos(x_2) e^{[(1-\sin(x_1))^2]} + (x_1 - x_2)^2$$

11. Modified Schaffer function #1: In the search domain $x_1,\ x_2 \in [-100,\ 100]$ this function is defined as follows and has $f_{\min}(0,\ 0) = 0$.

$$f(x) = 0.5 + \frac{\sin^2(x_1^2 + x_2^2) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}.$$

12. Modified Schaffer function #2: In the search domain $x_1,\ x_2 \in [-100,\ 100]$ this function is defined as follows and has $f_{\min}(0,\ 0) = 0$.

$$f(x) = 0.5 + \frac{\sin^2(x_1^2 - x_2^2) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}.$$

13. Modified Schaffer function #3: In the search domain $x_1,\ x_2 \in [-100,\ 100]$ this function is defined as follows and has $f_{\min}(0,\ 1.253115) = 0.00156685$.

$$f(x) = 0.5 + \frac{\sin^2\left[\cos\left[\left|x_1^2 - x_2^2\right|\right]\right] - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}.$$

14. Modified Schaffer function #4: In the search domain $x_1,\ x_2 \in [-100,\ 100]$ this function is defined as follows and has $f_{\min}(0,\ 1.253132) = 0.292579$.

$$f(x) = 0.5 + \frac{\cos^2\left[\sin\left[\left|x_1^2 - x_2^2\right|\right]\right] - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}.$$

IV. Some Well-Established Benchmark Functions: As mentioned earlier, we have also tested the RPS in searching the optimum points of some well-established functions. These functions are:

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

$$rate = \frac{\beta_1 x_2 - x_3 / \beta_5}{1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 x_3}$$

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

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

Best results are obtained by the Rosenbrock-Quasi-Newton method: $\hat{\beta}_1 = 1.253031$; $\hat{\beta}_2 = 1.190943$; $\hat{\beta}_3 = 0.062798$; $\hat{\beta}_4 = 0.040063$; $\hat{\beta}_5 = 0.112453$. The sum of squares of deviations (S^2) is = 0.298900994 and the coefficient of correlation (R) between observed rate and expected rate is =0.99945. The second best results are obtained by Hooke-Jeeves-Quasi-Newton method with S^2 = 0.318593458. Most of the other methods do not perform well.

The Particle Swarm method too does not ordinarily perform well in estimating the betas of the Hougen function. However, with γ(= a3) = 0.0005 and ω=0.05, run for 50,000 iterations we obtain: $\hat{\beta}_1 = 1.5575204$; $\hat{\beta}_2 = 0.0781010629$; $\hat{\beta}_3 = 0.050866667$; $\hat{\beta}_4 = 0.138796292$; $\hat{\beta}_5 = 0.955739322$. The sum of squares of deviations (S^2) is = 0.301933528. A comparison of Rosenbrock-Quasi-Newton results with these (RPS) results indicates that the betas exhibit very high degree of instability in the neighbourhood of the minimal S^2.

2. Egg holder function: This function is in m ($m \geq 2$) variables and given as:

$$f(x) = \sum_{i=1}^{m-1} \left(-(x_{i+1} + 47)\sin(\sqrt{|x_{i+1} + x_i/2 + 47|}) + \sin(\sqrt{|x_i - (x_{i+1} + 47)|})(-x_i) \right);\ -512 \leq x_i \leq 512;\ i = 1, 2, ..., m$$

We obtain $f_{\min}(512,\ 404.2319) \simeq 959.64$. It is a difficult function to optimize.

3. Sine envelope sine wave function: The function, also referred as the Schaffer function (m=2), is given as:

$$f(x) = \sum_{i=1}^{m-1} \left(\frac{\sin^2\left[\sqrt{x_{i+1}^2 + x_i^2}\right] - 0.5}{(0.001\left(x_{i+1}^2 + x_i^2\right) + 1)^2} + 0.5 \right);\ -100 \leq x_i \leq 100;\ i = 1, 2, ..., m$$

It is a difficult problem to optimize. For higher dimensions it gives repeating couplets of optimal values of x^*, except their sign.

4. Chichinadze function: In the search domain $x_1,\ x_2 \in [-30,\ 30]$ this function is defined as follows and has $f_{\min}(5.90133,\ 0.5) = -43.3159$.

$$f(x) = x_1^2 - 12x_1 + 11 + 10\cos(\pi x_1 / 2) + 8\sin(5\pi x_1) - (1/5)^{0.5} e^{-0.5(x_2 - 0.5)^2}$$

5. McCormick function: In the search domain $x_1 \in [-1.5,\ 4], x_2 \in [-3,\ 4]$ this function is defined as follows and has $f_{\min}(-0.54719,\ -1.54719) = -1.9133$.

$$f(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1.$$

6. Levy function (#13): In the search domain $x_1,\ x_2 \in [-10,\ 10]$ this function is defined as follows and has $f_{\min}(1,\ 1) = 0$.

$$f(x) = \sin^2(3\pi x_1) + (x_1 - 1)^2[1 + \sin^2(3\pi x_2)] + (x_2 - 1)^2[1 + \sin^2(2\pi x_2)].$$

7. Three-humps camel back function: In the search domain $x_1,\ x_2 \in [-5,\ 5]$ this function is defined as follows and has $f_{\min}(0,\ 0) = 0$.

$$f(x) = 2x_1^2 - 1.05x_1^4 + x_1^6 / 6 + x_1 x_2 + x_2^2.$$

8. Zettle function: In the search domain $x_1,\ x_2 \in [-5,\ 5]$ this function is defined as follows and has $f_{\min}(-0.0299,\ 0) = -0.003791$.

$$f(x) = (x_1^2 + x_2^2 - 2x_1)^2 + 0.25x_1$$

9. Styblinski-Tang function: In the search domain $x_1,\ x_2 \in [-5,\ 5]$ this function is defined as follows and has $f_{\min}(-2.903534,\ -2.903534) \simeq -78.332$.

$$f(x) = \frac{1}{2}\sum_{i=1}^{2}(x_i^4 - 16x_i^2 + 5x_i).$$

10. Bukin functions: Bukin functions are almost fractal (with fine seesaw edges) in the surroundings of their minimal points. Due to this property, they are extremely difficult to optimize by any method of global (or local) optimization. In the search domain $x_1 \in [-15,\ -5], x_2 \in [-3,\ 3]$ these functions are defined as follows.

$$f_4 = 100x_2^2 + 0.01|x_1 + 10| \quad ; \ f_{\min}(-10,\ 0) = 0$$

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

11. Leon function: In the search domain $x_1,\ x_2 \in [-1.2,\ 1.2]$ this function is defined as follows and has $f_{\min}(1,\ 1) = 0$.

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

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

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

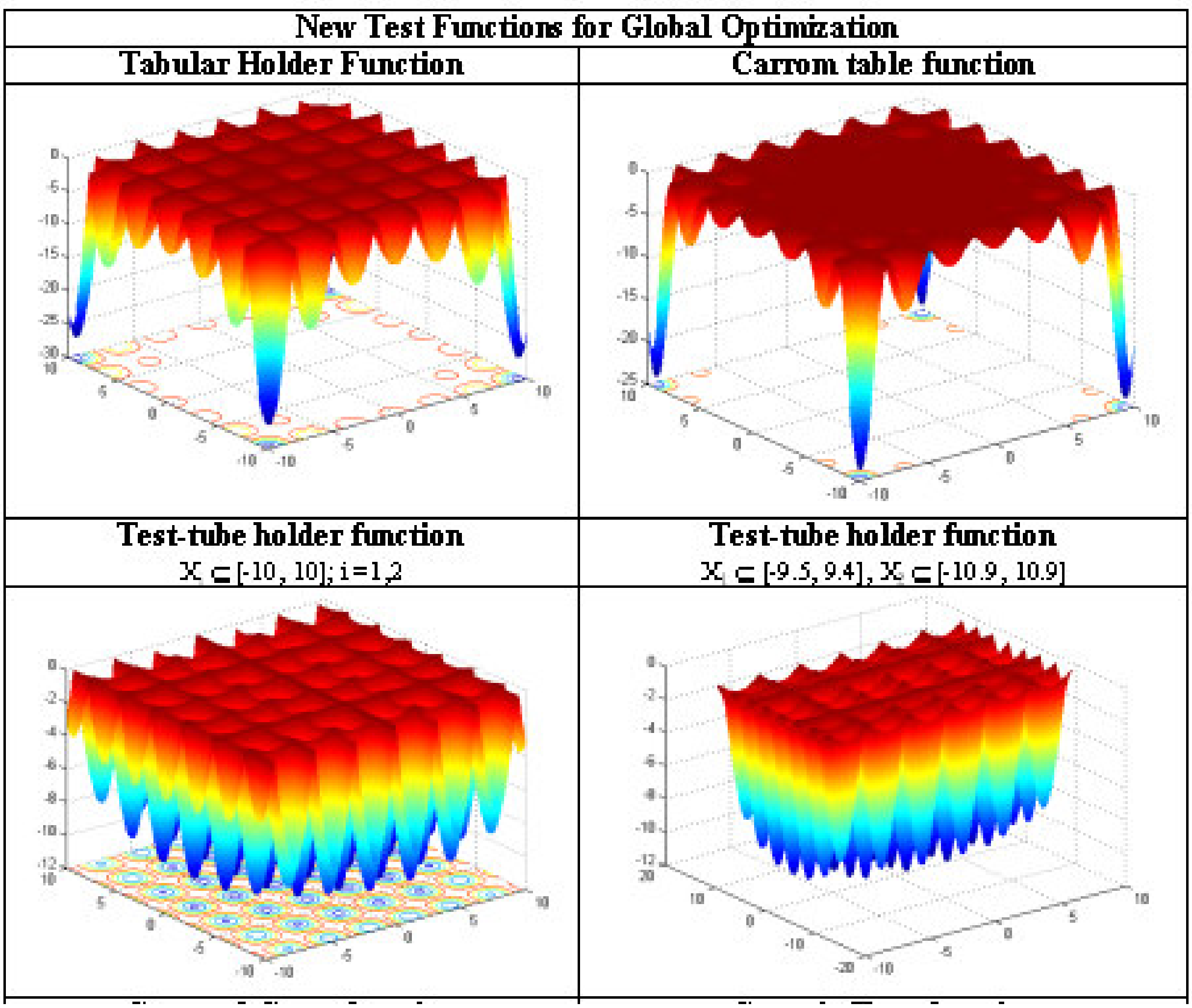
We have obtained $f_{\min}$ (0.4673199, 0.4673183) = 0.06447.

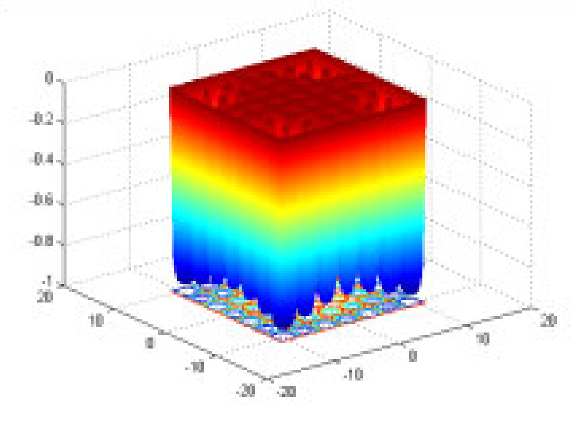
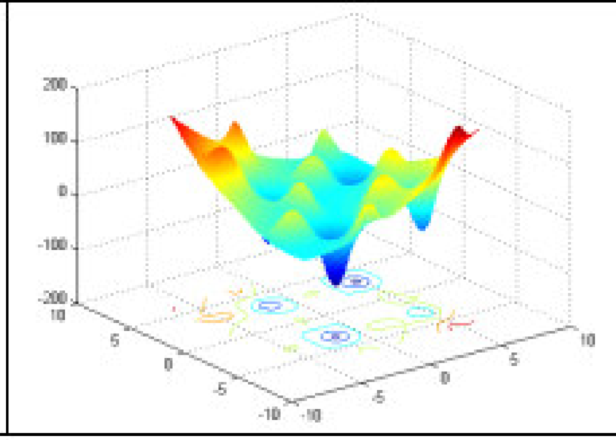
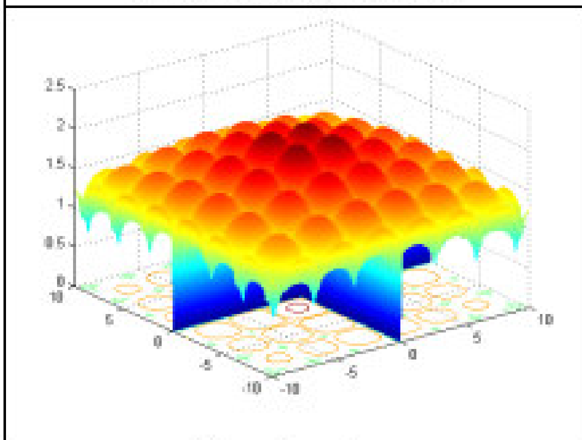
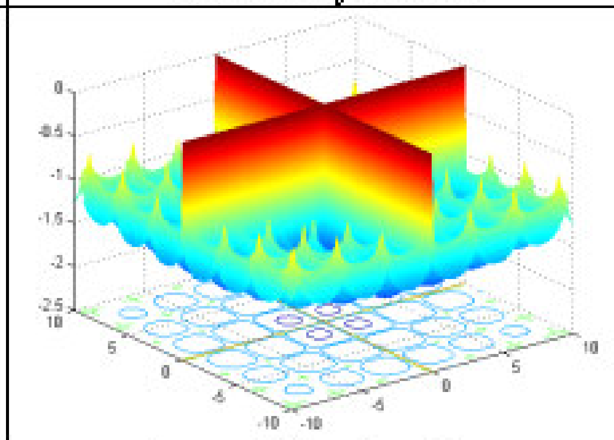
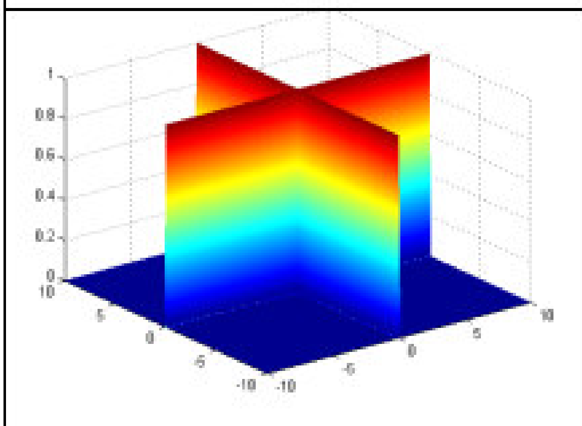
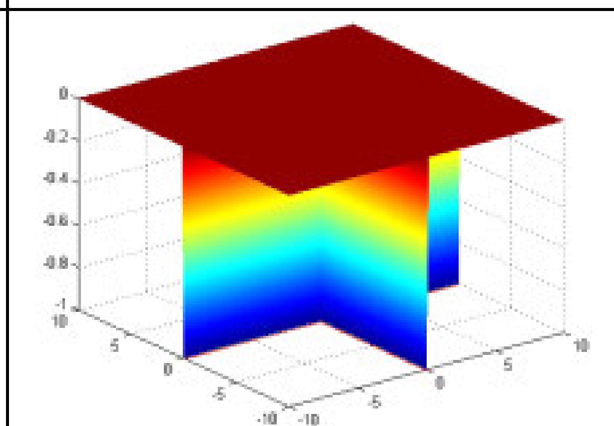
13. Schaffer function: In the search domain $x_1,\ x_2 \in [-100,\ 100]$ this function is defined as follows and has $f_{\min}(0,\ 0) = 0$.

$$f(x) = 0.5 + \frac{\sin^2\left[\sqrt{x_1^2 + x_2^2}\right] - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}.$$

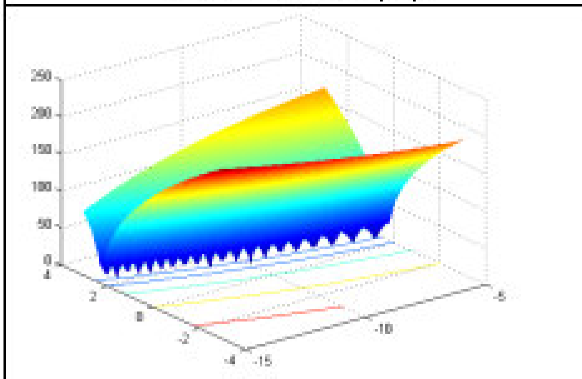
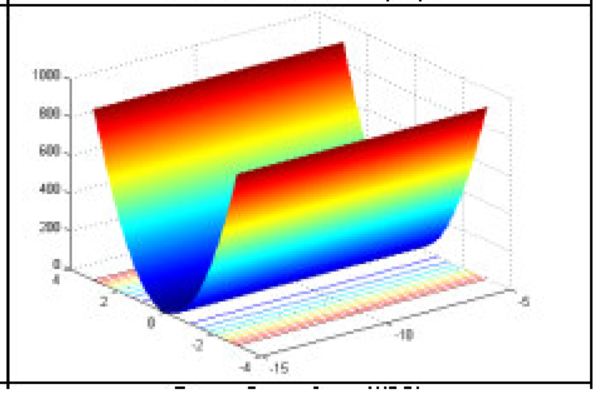
V. FORTRAN Program of RPS: We append a program of the Repulsive Particle Swarm method. The program has run successfully and optimized most of the functions. However, the crowned cross function and the cross-legged table functions have failed the program.

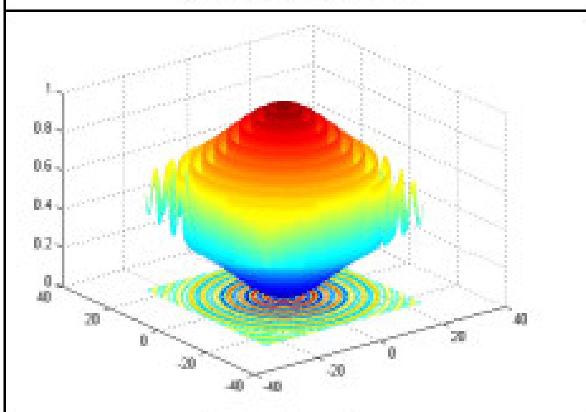
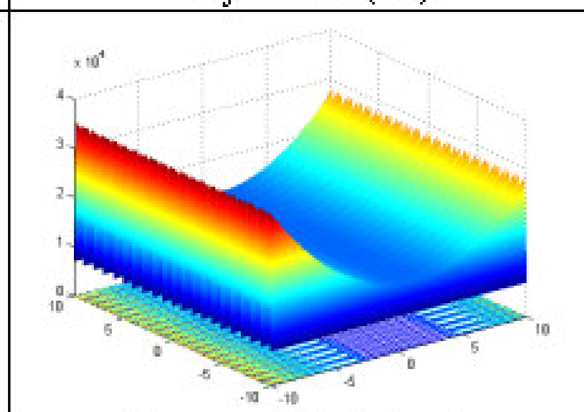
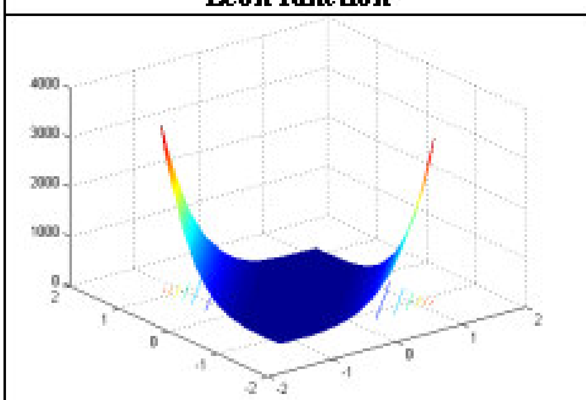
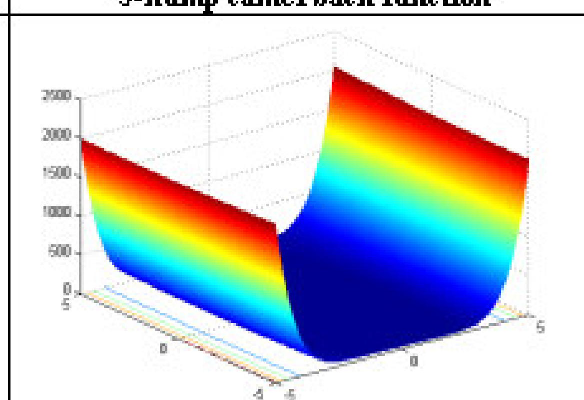
VI. Conclusion: Our program of the RPS method has succeeded in optimizing most of the established functions and the newly introduced functions. The functions (namely - Giunta, Bukin, cross-legged table, crowned cross and Hougen functions in particular) that have failed the RPS program miserably may be attractive to other methods such as Simulated Annealing, Genetic algorithms and tunneling methods. Improved versions of Particle Swarm method also may be tested.

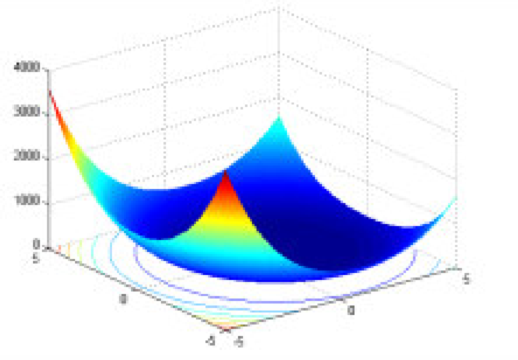
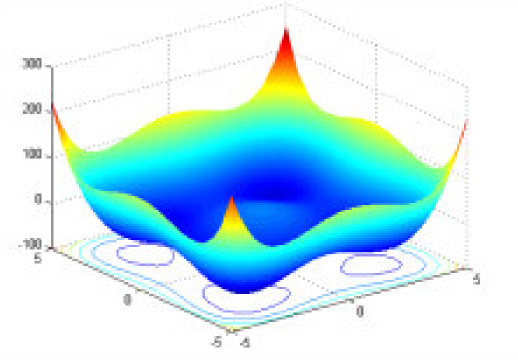
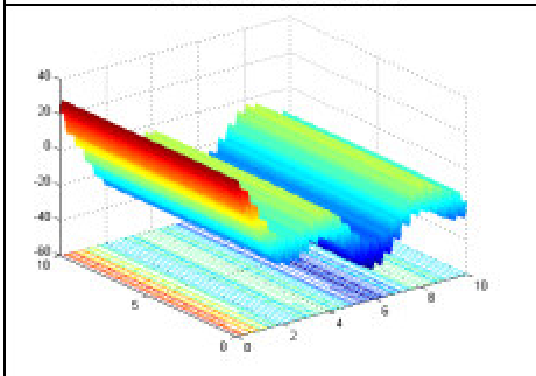
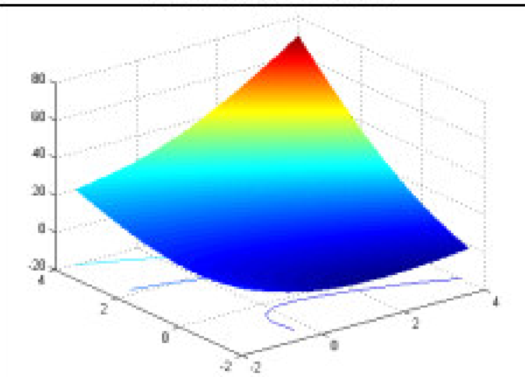
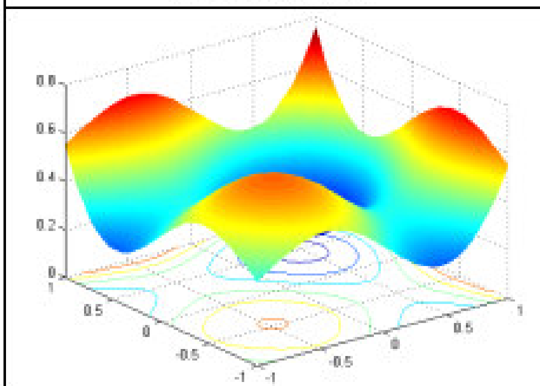
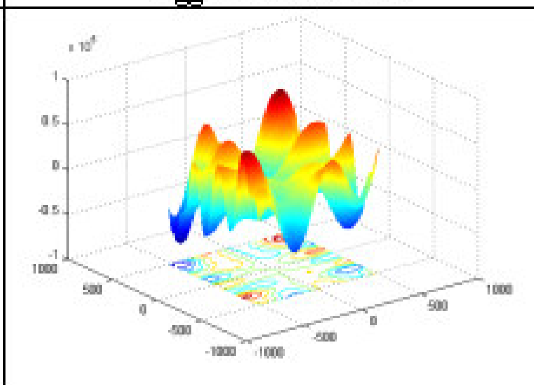
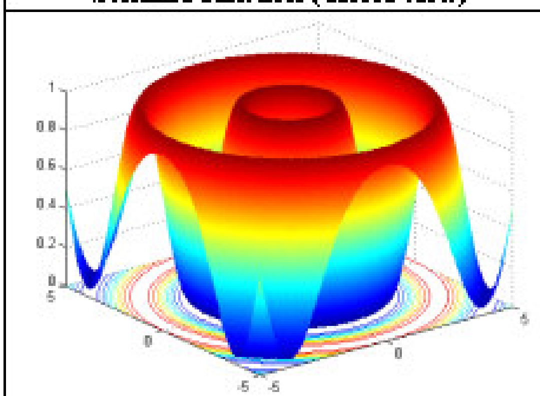
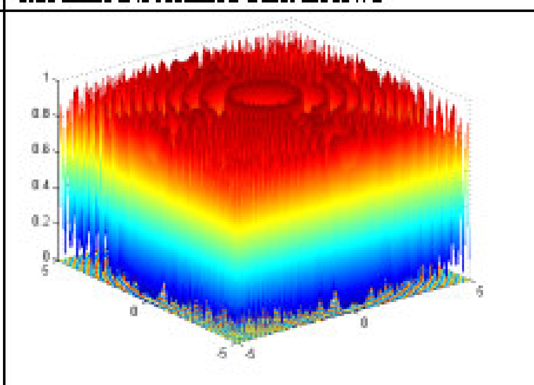


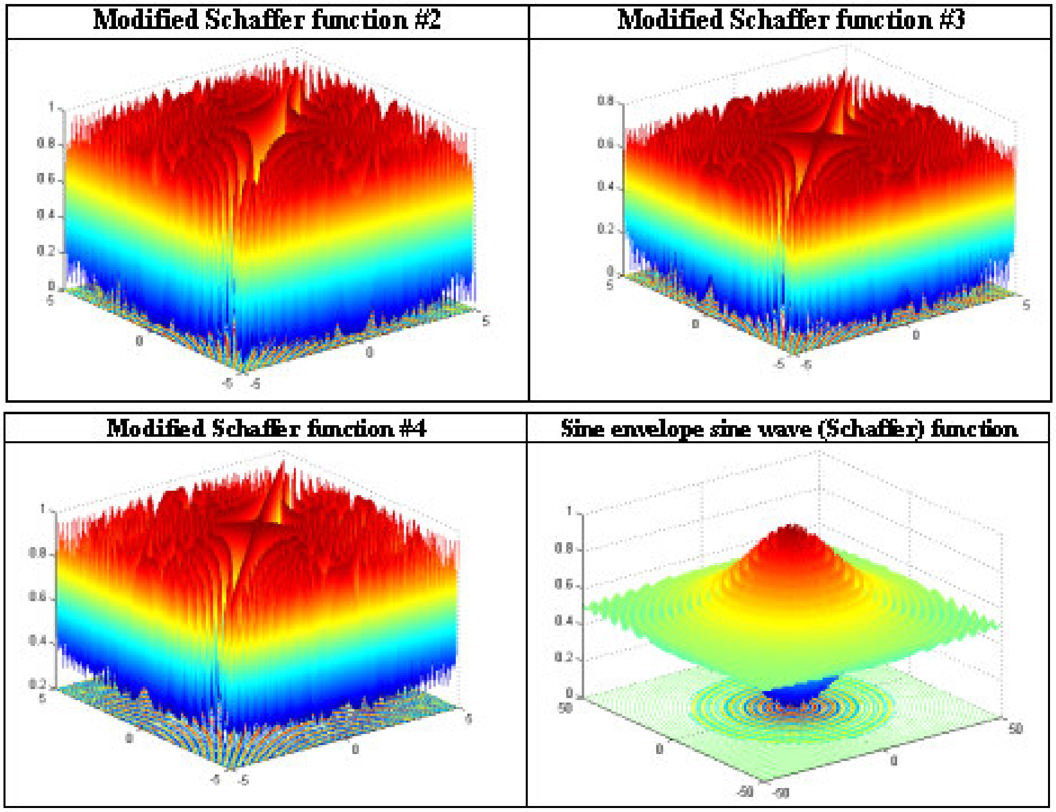
Pen holder function	Bird function
	
Crowned Cross function	**Cross in Tray function**
	
Cross function	**Inverted Cross function**
	

Some Well-established Benchmark Functions

Bukin function (#6)	Bukin function (#4)
	

Schaffer function	Levy function (#13)
	
Leon function	**3-hump camel back function**
	

Zettle function	Styblinski-Tang function
	
Chichinadze function	**McCormick function**
	
Giunta function	**Egg holder function**
	
Schaffer function (Closer view)	**Modified Schaffer function #1**
	



Bibliography

- Ackley, D. H.: *A Connectionist Machine for Genetic Hill-Climbing*, Kluwer Academic Publishers, Boston, 1987.
- Bauer, J.M.: "Harnessing the Swarm: Communication Policy in an Era of Ubiquitous Networks and Disruptive Technologies", *Communications and Strategies*, 45, 2002.
- Bukin, A. D.: *New Minimization Strategy For Non-Smooth Functions*, Budker Institute of Nuclear Physics preprint BUDKER-INP-1997-79, Novosibirsk 1997..
- Chichinadze, V.: "The ψ-transform for Solving Linear and Nonlinear Programming Problems", *Automata,* 5, 347–355, 1969.
- Easom, E. E.: *A Survey of Global Optimization Techniques*, M. Eng. thesis, Univ. Louisville, Louisville, KY, 1990.
- 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.
- Giunta, A. A.: *Aircraft Multidisciplinary Design Optimization using Design of Experiments Theory and Response Surface Modeling Methods*, MAD Center Report 97-05-01, Virginia Polytechnic Institute & State Univ. Blacksburg, VA, 1997.
- Hayek, F.A.: *The Road to Serfdom*, Univ. of Chicago Press, Chicago, 1944.
- Huang, V.L., Suganthan, P.N. and Liang, J.J. "Comprehensive Learning Particle Swarm Optimizer for Solving Multi-objective Optimization Problems", *International Journal of*

Intelligent Systems, 21, pp.209–226 (Wiley Periodicals, Inc. Published online in Wiley InterScience www.interscience.wiley.com) , 2006
- 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.
- Kuester, J.L. and Mize, J.H.: *Optimization Techniques with Fortran*, McGraw-Hill Book Co. New York, 1973.
- 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.
- Madsen, K. and Zilinskas, J.: *Testing Branch-and-Bound Methods for Global Optimization*, IMM technical report 05, Technical University of Denmark, 2000.
- 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 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).
- Nagendra, S.: *Catalogue of Test Problems for Optimization Algorithm Verification*, Technical Report 97-CRD-110, General Electric Company, 1997.
- 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.
- Schwefel, H.P.: *Numerical Optimization of Computer Models*, Wiley & Sons, Chichester, 1981.
- 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.
- Styblinski, M. and Tang, T.: "Experiments in Nonconvex Optimization: Stochastic Approximation with Function Smoothing and Simulated Annealing", *Neural Networks*, 3, 467-483, 1990.
- Sumper, D.J.T.: "The Principles of Collective Animal Behaviour", *Phil. Trans. R. Soc. B*. 361, pp. 5-22, 2006.
- Veblen, T.B.: *The Theory of the Leisure Class*, The New American library, NY. (Reprint, 1953), 1899.
- Whitley, D., Mathias, K., Rana, S. and Dzubera, J.: "Evaluating Evolutionary Algorithms", *Artificial Intelligence*, 85, 245-276, 1996.

Author's Contact: mishrasknehu@hotmail.com

```
C     PROGRAM TO FIND GLOBAL MINIMUM BY REPULSIVE PARTICLE SWARM METHOD
C     WRITTEN BY SK MISHRA, DEPT. OF ECONOMICS, NEHU, SHILLONG (INDIA)
      PARAMETER (N=50  ,NN=25 , MX=100, NSTEP=21, ITRN=5000)
C     N = POPULATION SIZE. IN MOST OF THE CASES N=30 IS OK. ITS VALUE
C     MAY BE INCREASED TO 50 ALSO. THE PARAMETER NN IS THE SIZE OF
C     RANDOMLY CHOSEN NEIGHBOURS. 15 TO 25 (BUT SUFFICIENTLY LESS THAN
C     N) IS A GOOD CHOICE. MX IS THE MAXIMAL SIZE OF DECISION VARIABLES.
C     IN F(X1, X2,...,XM) M SHOULD BE LESS THAN OR EQUAL TO MX. ITRN IS
C     THE NO. OF ITERATIONS. IT MAY DEPEND ON THE PROBLEM. 200 TO 500
C     ITERATIONS MAY BE GOOD ENOUGH. BUT FOR FUNCTIONS LIKE ROSENBROCK
C     OR GRIEWANK OF LARGE SIZE (SAY M=20) IT IS NEEDED THAT ITRN IS
C     LARGE, SAY 5000 OR 10000.
C     THE SUBROUTINE FUNC( ) DEFINES THE FUNCTION TO BE OPTIMIZED.

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /RNDM/IU,IV
      COMMON /KFF/KF
      INTEGER IU,IV
      DIMENSION X(N,MX),V(N,MX),A(MX),VI(MX),TIT(50)
      DIMENSION XX(N,MX),F(N),R(3),V1(MX),V2(MX),V3(MX),V4(MX),BST(MX)
      CHARACTER *70 TIT
C     A1 A2 AND A3 ARE CONSTANTS AND W IS THE INERTIA WEIGHT.
C     OCCASIONALLY, TINKERING WITH THESE VALUES, ESPECIALLY A3, MAY BE
C     NEEDED.
      DATA A1,A2,A3,W /.5D00,.5D00,.0005D00,.5D00/
      WRITE(*,*)'----------------------------------------------------'
      DATA TIT(1)/'KF=1 TEST TUBE HOLDER FUNCTION(A) 2-VARIABLES M=2'/
      DATA TIT(2)/'KF=2 TEST TUBE HOLDER FUNCTION(B) 2-VARIABLES M=2'/
      DATA TIT(3)/'KF=3 HOLDER TABLE FUNCTION 2-VARIABLES M=2'/
      DATA TIT(4)/'KF=4 CARROM TABLE FUNCTION 2-VARIABLES M=2'/
      DATA TIT(5)/'KF=5 CROSS IN TRAY FUNCTION 2-VARIABLES M=2'/
      DATA TIT(6)/'KF=6 CROWNED CROSS FUNCTION 2-VARIABLES M=2'/
      DATA TIT(7)/'KF=7 CROSS FUNCTION 2-VARIABLES M=2'/
      DATA TIT(8)/'KF=8 CROSS-LEGGED TABLE FUNCTION 2-VARIABLES M=2'/
      DATA TIT(9)/'KF=9 PEN HOLDER FUNCTION 2-VARIABLES M=2'/
      DATA TIT(10)/'KF=10 BIRD FUNCTION 2-VARIABLES M=2'/

      DATA TIT(11)/'KF=11 DE JONG SPHERE FUNCTION M-VARIABLE M=?'/
      DATA TIT(12)/'KF=12 LEON FUNCTION 2-VARIABLE M=2'/
      DATA TIT(13)/'KF=13 GIUNTA FUNCTION 2-VARIABLE M=2'/
      DATA TIT(14)/'KF=14 SCHAFFER FUNCTION 2-VARIABLE M=2'/
      DATA TIT(15)/'KF=15 CHICHINADZE FUNCTION 2-VARIABLE M=2'/
      DATA TIT(16)/'KF=16 MCCORMICK FUNCTION 2-VARIABLE M=2'/
      DATA TIT(17)/'KF=17 LEVY # 13 FUNCTION 2-VARIABLE M=2'/
      DATA TIT(18)/'KF=18 3-HUMP CAMEL BACK FUNCTION 2-VARIABLE M=2'/
      DATA TIT(19)/'KF=19 ZETTLE FUNCTION 2-VARIABLE M=2'/
      DATA TIT(20)/'KF=20 STYBLINSKI-TANG FUNCTION 2-VARIABLE M=2'/

      DATA TIT(21)/'KF=21 BUKIN-4 FUNCTION 2-VARIABLE M=2'/
      DATA TIT(22)/'KF=22 BUKIN-6 FUNCTION 2-VARIABLE M=2'/
      DATA TIT(23)/'KF=23 HOUGEN REGRESSION FUNCTION 5-VARIABLE M=5'/
      DATA TIT(24)/'KF=24 SINE ENVELOPE SINE WAVE FUNCTION  M=? '/
      DATA TIT(25)/'KF=25 EGG-HOLDER FUNCTION  M=?'/
      DATA TIT(26)/'KF=26 MODIFIED SCHAFFER FUNCTION #1 2-VARIABLE M=2'/
      DATA TIT(27)/'KF=27 MODIFIED SCHAFFER FUNCTION #2 2-VARIABLE M=2'/
      DATA TIT(28)/'KF=28 MODIFIED SCHAFFER FUNCTION #3 2-VARIABLE M=2'/
      DATA TIT(29)/'KF=29 MODIFIED SCHAFFER FUNCTION #4 2-VARIABLE M=2'/
      DATA TIT(30)/'KF=30 QUARTIC(+NOISE) FUNCTION M-VARIABLE M=?'/

      DO I=1,30
      WRITE(*,*)TIT(I)
      ENDDO
      WRITE(*,*)'----------------------------------------------------'
      WRITE(*,*)'CHOOSE KF AND SPECIFY M'
      READ(*,*) KF,M
      DSIGN=1.D00
      LCOUNT=0
```

```
      WRITE(*,*)'4-DIGITS SEED FOR RANDOM NUMBER GENERATION'
      READ(*,*) IU
      DATA ZERO,ONE,FMIN /0.0D00,1.0D00,1.0E30/
C     GENERATE N-SIZE POPULATION OF M-TUPLE PARAMETERS X(I,J) RANDOMLY
      DO I=1,N
        DO J=1,M
        CALL RANDOM(RAND)
         X(I,J)=(RAND-0.5D00)*10

C     WE GENERATE RANDOM(-5,5). HERE MULTIPLIER IS 10. TINKERING IN SOME
C     CASES MAY BE NEEDED
         ENDDO
        F(I)=1.0E30
      ENDDO
C     INITIALISE VELOCITIES V(I) FOR EACH INDIVIDUAL IN THE POPULATION
      DO I=1,N
      DO J=1,M
      CALL RANDOM(RAND)
       V(I,J)=(RAND-.5D+00)
C       V(I,J)=RAND
      ENDDO
      ENDDO
      ZZZ=1.0E+30
      ICOUNT=0
      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     AND XX(I,J) IS THE M-TUPLE VALUE OF X ASSOCIATED WITH THE 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      CHOOSE NN COLLEAGUES RANDOMLY AND FIND THE BEST AMONG THEM
          BEST=1.0E30
           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
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
      X(I,J)=X(I,J)+V(I,J)
      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
      Z=FMIN
      IF(LCOUNT.EQ.100) THEN

      LCOUNT=0
      WRITE(*,*)'OPTIMAL SOLUTION UPTO THIS'
      WRITE(*,*)'X = ',(BST(J),J=1,M),' MIN F = ',FMIN
      ENDIF
  999 FORMAT(5F15.6)
      LCOUNT=LCOUNT+1
  100 CONTINUE
      WRITE(*,*)'OVER:',TIT(KF)
      END

      SUBROUTINE LSRCH(A,M,VI,NSTEP,FI)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /KFF/KF
      COMMON /RNDM/IU,IV
      INTEGER IU,IV
      DIMENSION A(*),B(100),VI(*)
      AMN=1.0E30
      DO J=1,NSTEP
         DO JJ=1,M
         B(JJ)=A(JJ)+(J-NSTEP/2-1)*VI(JJ)
         ENDDO
      CALL FUNC(B,M,FI)
        IF(FI.LT.AMN) THEN
        AMN=FI
        DO JJ=1,M
        A(JJ)=B(JJ)
        ENDDO
        ENDIF
```

```
      ENDDO
      FI=AMN
      RETURN
      END

      SUBROUTINE RANDOM(RAND1)
       DOUBLE PRECISION  RAND1
       COMMON /RNDM/IU,IV
      INTEGER IU,IV
       RAND=REAL(RAND1)
       IV=IU*65539
       IF(IV.LT.0) THEN
       IV=IV+2147483647+1
       ENDIF
       RAND=IV
       IU=IV
       RAND=RAND*0.4656613E-09
       RAND1=DBLE(RAND)
       RETURN
       END

      SUBROUTINE FUNC(X,M,F)
C     New Test Functions
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
       COMMON /RNDM/IU,IV
       COMMON /KFF/KF
       INTEGER IU,IV
      DIMENSION X(*)
      PI=4.D+00*DATAN(1.D+00)
C     -----------------------------------------------------------------
      IF(KF.EQ.1) THEN
C     Test-tube holder function (A)
        FP=0.D00
C     -10 TO 10 M=2
      F=0.D00
        if(x(1).lt.-10.d00 .or. x(1).gt. 10.d00) fp=fp+x(1)**2
        if(x(2).lt.-10.d00 .or. x(2).gt. 10.d00) fp=fp+x(1)**2
        IF(FP.GT.0.D00) THEN
       f=FP
      ELSE
      f=-4*dabs(dsin(X(1))*dcos(x(2))*dexp(dabs(dcos((X(1)**2+x(2)**2)/
     & 200))))
      ENDIF
      RETURN
      ENDIF
C     -----------------------------------------------------------------
      IF(KF.EQ.2) THEN
C     Test-tube holder function (B)
      FP=0.D00
      F=0.D00
        if(x(1).lt.-9.5d00 .or. x(1).gt. 9.4d00) fp=fp+x(1)**2
        if(x(2).lt.-10.9d00 .or. x(2).gt. 10.9d00) fp=fp+x(1)**2
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f=-4*dabs(dsin(X(1))*dcos(x(2))*dexp(dabs(dcos((X(1)**2+x(2)**2)/
     & 200))))
      ENDIF
      RETURN
      ENDIF
C     -----------------------------------------------------------------
      IF(KF.EQ.3) THEN
C     Holder table function
        FP=0.D00
C     -10 TO 10 M=2
      F=0.D00
        DO I=1,M
```

```
          IF( DABS(X(I)).GT.10.D00) FP=FP+X(I)**2
          ENDDO
          IF(FP.GT.0.D00) THEN
          F=FP
        ELSE
        f=-dabs(dcos(X(1))*dcos(x(2))*dexp(dabs(1.d00-(dsqrt(X(1)**2+
     & x(2)**2)/pi))))
        ENDIF
        RETURN
        ENDIF
C       -----------------------------------------------------------------
        IF(KF.EQ.4) THEN
C       Carrom table function
          FP=0.D00
C       -10 TO 10 M=2
        F=0.D00
          DO I=1,M
          IF( DABS(X(I)).GT.10.D00) FP=FP+X(I)**2
          ENDDO
          IF(FP.GT.0.D00) THEN
          F=FP
        ELSE
        f=-1.d00/30*(dcos(X(1))*dcos(x(2))*dexp(dabs(1.d00-
     & (dsqrt(X(1)**2 + x(2)**2)/pi))))**2
        ENDIF
        RETURN
        ENDIF
C       -----------------------------------------------------------------
        IF(KF.EQ.5) THEN
          FP=0.D00
C       Cross in tray function
C       -10 TO 10 M=2
        F=0.D00
          DO I=1,M
          IF( DABS(X(I)).GT.10.d00) FP=FP+X(I)**2
          ENDDO
          IF(FP.GT.0.D00) THEN
          F=FP
        ELSE
        f=-0.0001d00*(dabs(dsin(X(1))*dsin(x(2))*dexp(dabs(100.d00-(dsqrt
     & (X(1)**2+x(2)**2)/pi))))+1.d00)**(.1)
        ENDIF
        RETURN
        ENDIF
C       -----------------------------------------------------------------
        IF(KF.EQ.6) THEN
        FP=0.D00
C       Crowned Cross function
C       -10 TO 10 M=2
        F=0.D00
          DO I=1,M
          IF( DABS(X(I)).GT.10.d00) FP=FP+dexp(dabs(X(I)))
          ENDDO
          IF(FP.GT.0.D00) THEN
          F=FP
        ELSE
        f=0.0001d00*(dabs(dsin(X(1))*dsin(x(2))*dexp(dabs(100.d00-
     & (dsqrt(X(1)**2+x(2)**2)/pi))))+1.d00)**(.1)
        ENDIF
        RETURN
        ENDIF
C       -----------------------------------------------------------------
        IF(KF.EQ.7) THEN
          FP=0.D00
C       Cross function
C       -10 TO 10 M=2
        F=0.D00
```

```
        DO I=1,M

        IF( DABS(X(I)).GT.10.d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f=(dabs(dsin(X(1))*dsin(x(2))*dexp(dabs(100.d00-(dsqrt
     & (X(1)**2+x(2)**2)/pi))))+1.d00)**(-.1)
      ENDIF
      RETURN
      ENDIF
C     -----------------------------------------------------------------
      IF(KF.EQ.8) THEN
      FP=0.D00
C     Cross-legged table function
C     -10 TO 10 M=2
      F=0.D00
        DO I=1,M

        IF( DABS(X(I)).GT.10.d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f=-(dabs(dsin(X(1))*dsin(x(2))*dexp(dabs(100.d00-(dsqrt
     & (X(1)**2+x(2)**2)/pi))))+1.d00)**(-.1)
      ENDIF
      RETURN
      ENDIF
C     -----------------------------------------------------------------
      IF(KF.EQ.9) THEN
C     Pen holder function
        FP=0.D00
C     -11 TO 11 M=2
        DO I=1,M
        IF(DABS(X(I)).GT.11.D00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f=-Dexp(-(Dabs(Dcos(X(1))*Dcos(X(2))*Dexp(Dabs(1.D0-(Dsqrt
     & (X(1)**2+X(2)**2)/pi))))**(-1)))
      ENDIF
      RETURN
      ENDIF
C     -----------------------------------------------------------------
      IF(KF.EQ.10) THEN
C     Bird function
        FP=0.D00
C     -2Pi TO 2Pi M=2
        DO I=1,M
        IF(DABS(X(I)).GT.2*pi) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f=(dsin(x(1))*dexp((1.d00-dcos(x(2)))**2) +
     & dcos(x(2))*dexp((1.d00-dsin(x(1)))**2))+(x(1)-x(2))**2
      ENDIF
      RETURN
      ENDIF
C     -----------------------------------------------------------------
      IF(KF.EQ.11) THEN
C     DE JONG SPHERE function
      F=0.D00
      DO I=1,M
```

```
      F=F+X(I)**2
      ENDDO
      RETURN
      ENDIF
C     ----------------------------------------------------------------
      IF(KF.EQ.12) THEN
C     Leon function
        FP=0.D00
C     -1.2 TO 1.2 M=2
        DO I=1,M
        IF(DABS(X(I)).GT.1.2d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f=100*(x(2)-x(1)**2)**2+(1.d00-x(1))**2
      ENDIF
      RETURN
      ENDIF
C     ----------------------------------------------------------------     -----------
---------------------------------------------------------------------
      IF(KF.EQ.13) THEN
C     Giunta function
        FP=0.D00
C     -1 TO 1 M=2
        DO I=1,M
        IF(DABS(X(I)).GT.1.d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      c=16.d00/15.d00
      f=dsin(c*x(1)-1.d0)+dsin(c*x(1)-1.d0)**2+dsin(4*(c*x(1)-1.d0))/50+
     &dsin(c*x(2)-1.d0)+dsin(c*x(2)-1.d0)**2+dsin(4*(c*x(2)-1.d0))/50+.6
      ENDIF
      RETURN
      ENDIF
C     ----------------------------------------------------------------
      IF(KF.EQ.14) THEN
C     Schaffer function
        FP=0.D00
C     -100 TO 100 M=2
        DO I=1,M
        IF(DABS(X(I)).GT.100.d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f1=dsin(dsqrt(x(1)**2+x(2)**2))**2-0.5d00
      f2=(1.d00+ 0.001*(x(1)**2 + x(2)**2))**2
      f=f1/f2 +0.5d00
      ENDIF
      RETURN
      ENDIF
C     ----------------------------------------------------------------
       IF(KF.EQ.15) THEN
C     Chichinadze function
        FP=0.D00
C     -30 <=X(I)<= 30 M=2
        DO I=1,M
        IF(DABS(X(I)).GT.30.d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f=X(1)**2-12*X(1)+11.D00+10*DCOS(PI*X(1)/2)+8*DSIN(5*PI*X(1))-
     & (1/DSQRT(5.D00))*DEXP(-(X(2)-0.5D00)**2/2)
```

```
      ENDIF
      RETURN
      ENDIF
C     -----------------------------------------------------------------
      IF(KF.EQ.16) THEN
C     McCormick function
        FP=0.D00
C     -1.5<= X(1)<=4; -3<=X(2)<=4 ; M=2
        IF(X(1).LT. -1.5D00 .OR. X(1) .GT. 4.D00) FP=FP+X(1)**2
        IF(X(2).LT. -3.D00  .OR. X(2) .GT. 4.D00) FP=FP+X(2)**2
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f=DSIN(X(1)+X(2))+(X(1)-X(2))**2-1.5*X(1)+2.5*X(2)+1.D00
      ENDIF
      RETURN
      ENDIF
C     -----------------------------------------------------------------
      IF(KF.EQ.17) THEN
C     Levy #13 function
        FP=0.D00
C     -10 <=X(I) <=10 M=2
        DO I=1,M
        IF(DABS(X(I)).GT.10.d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f=DSIN(3*PI*X(1))**2+(X(1)-1.D00)**2*(1.D00+DSIN(3*PI*X(2))**2) +
     & (X(2)-1.D00)**2*(1.D00+DSIN(2*PI*X(2))**2)
      ENDIF
      RETURN
      ENDIF
C     -----------------------------------------------------------------
      IF(KF.EQ.18) THEN
C     Three-hump Camel back function
        FP=0.D00
C     -5 <=X(I) <=5 M=2
        DO I=1,M
        IF(DABS(X(I)).GT.5.d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f=2*X(1)**2-1.05*X(1)**4+X(1)**6/6 + X(1)*X(2)+X(2)**2
      ENDIF
      RETURN
      ENDIF
C     -----------------------------------------------------------------
      IF(KF.EQ.19) THEN
C     Zettle function
        FP=0.D00
C     -5 <=X(I) <=5 M=2
        DO I=1,M
        IF(DABS(X(I)).GT.5.d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f=(X(1)**2+X(2)**2-2*X(1))**2 + 0.25*X(1)
      ENDIF
      RETURN
      ENDIF
C     -----------------------------------------------------------------
      IF(KF.EQ.20) THEN
C     Styblinski-Tang function
        FP=0.D00
```

```
C       -5 <=X(I) <=5 M=2
          DO I=1,M
          IF(DABS(X(I)).GT.5.d00) FP=FP+X(I)**2
          ENDDO
          IF(FP.GT.0.D00) THEN
          F=FP
        ELSE
        F=0.D00
        DO I=1,M
        f=F+(X(I)**4-16*X(I)**2+5*X(I))
        ENDDO
        F=F/2
        ENDIF
        RETURN
        ENDIF
C       -----------------------------------------------------------------
        IF(KF.EQ.21) THEN
C       BUKIN-4 function
          FP=0.D00
C       -15. LE. X(1) .LE. -5 AND -3 .LE. X(2) .LE. 3
          IF(X(1).LT. -15.D00 .OR. X(1) .GT. -5.D00) FP=FP+X(1)**2
          IF(DABS(X(2)).GT.3.D00) FP=FP+X(2)**2
          IF(FP.GT.0.D00) THEN
          F=FP
        ELSE
        F=100*X(2)**2 + 0.01*DABS(X(1)+10.D0)
        ENDIF
        RETURN
        ENDIF
C       -----------------------------------------------------------------
        IF(KF.EQ.22) THEN
C       BUKIN-6 function
          FP=0.D00
C       -15. LE. X(1) .LE. -5 AND -3 .LE. X(2) .LE. 3
          IF(X(1).LT. -15.D00 .OR. X(1) .GT. -5.D00) FP=FP+X(1)**2
          IF(DABS(X(2)).GT.3.D00) FP=FP+X(2)**2

          IF(FP.GT.0.D00) THEN
          F=FP
        ELSE
        F=100*DSQRT(DABS(X(2)-0.01D00*X(1)**2))+ 0.01*DABS(X(1)+10.D0)
        ENDIF
        RETURN
        ENDIF
C       -----------------------------------------------------------------
        IF(KF.EQ.23) THEN
C       HOUGEN FUNCTION (HOUGEN-WATSON MODEL FOR REACTION KINATICS)
C       NO. OF PARAMETERS TO ESTIMATE = 5 = M
        CALL HOUGEN(M,X,F)
        RETURN
        ENDIF
C       -----------------------------------------------------------------
        IF(KF.EQ.24) THEN
C       SINE ENVELOPE SINE WAVE FUNCTION (Generalized Schaffer)
        f=0.d00
        fp=0.d00
        f1=0.d00
        f2=0.d00
        do I=1,m-1
        f1=dsin(dsqrt(x(I+1)**2+x(i)**2))**2-0.5d00
        f2=(0.001d00*(x(I+1)**2+x(i)**2)+1.d00)**2
        f=f+(f1/f2)+0.5d00
        enddo

        do I=1,m
        if(dabs(x(i)).gt.100.d00) fp=fp+x(i)**2
        enddo
```

```
      if(fp.gt.0.d00) f=fp
      return
      endif
C     ----------------------------------------------------------------
      IF(KF.EQ.25) THEN
C     EGG HOLDER FUNCTION
      f=0.d00
      fp=0.d00
      do I=1,m-1
      f1=-(x(I+1)+47.d00)
      f2=dsin(  dsqrt(  dabs(  x(I+1)+x(i)/2+47.d00   )   )   )
      f3=dsin(  dsqrt(  dabs(  x(i)-(x(I+1)+47.d00) ) ))
      f4=-x(i)
      f=f+ f1*f2+f3*f4
      enddo
      do I=1,m
       if(dabs(x(i)).gt.512.d00) fp=fp+x(i)**2
       enddo
       if(fp.gt.0.d00) f=fp
      return
      endif
C     ----------------------------------------------------------------
      IF(KF.EQ.26) THEN
C     Modified Schaffer function #1
        FP=0.D00
C     -100 TO 100 M=2
        DO I=1,M
        IF(DABS(X(I)).GT.100.d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f1=dsin(x(1)**2  +  x(2)**2)**2-0.5d00
      f2=(1.d00+ 0.001*(x(1)**2 + x(2)**2))**2
      f=f1/f2 +0.5d00
      ENDIF
      RETURN
      ENDIF
C     ----------------------------------------------------------------
      IF(KF.EQ.27) THEN
C     Modified (Hyperbolized) Schaffer function #2
        FP=0.D00
C     -100 TO 100 M=2
        DO I=1,M
        IF(DABS(X(I)).GT.100.d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f1=dsin(x(1)**2  - x(2)**2)**2-0.5d00
      f2=(1.d00+ 0.001*(x(1)**2 + x(2)**2))**2
      f=f1/f2 +0.5d00
      ENDIF
      RETURN
      ENDIF
C     ----------------------------------------------------------------
      IF(KF.EQ.28) THEN
C     Modified (crossed) Schaffer function #3
        FP=0.D00
C     -100 TO 100 M=2
        DO I=1,M
        IF(DABS(X(I)).GT.100.d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f1=dsin(dcos(   dabs(x(1)**2  - x(2)**2)   ))**2-0.5d00
```

```
      f2=(1.d00+ 0.001*(x(1)**2 + x(2)**2))**2
      f=f1/f2 +0.5d00
      ENDIF
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.29) THEN
C     Modified (crossed) Schaffer function #4
        FP=0.D00
C     -100 TO 100 M=2
        DO I=1,M
        IF(DABS(X(I)).GT.100.d00) FP=FP+X(I)**2
        ENDDO
        IF(FP.GT.0.D00) THEN
        F=FP
      ELSE
      f1=dcos(dsin(   dabs(x(1)**2  - x(2)**2)   ))**2-0.5d00
      f2=(1.d00+ 0.001*(x(1)**2 + x(2)**2))**2
      f=f1/f2 +0.5d00
      ENDIF
      RETURN
      ENDIF
C     ------------------------------------------------------------------
      IF(KF.EQ.30) THEN
C     Quartic function with noise
      f=0.d00
      FP=0.D00
      DO I=1,M
      IF(DABS(X(I)).GT.4.28D00) FP=FP+DEXP(DABS(X(I)))
      ENDDO
         IF(FP.NE.0.D00) THEN
      F=FP
      RETURN
          ELSE
      DO I=1,M
      CALL RANDOM(RAND)
      F=F+(I*X(I)**4)+RAND
      ENDDO
      RETURN
          ENDIF
      ENDIF
C     -----------------------------------------------------------------
      WRITE(*,*)'FUNCTION NOT DEFINED. PROGRAM ABORTED'
      STOP
      END

      SUBROUTINE HOUGEN(M,A,F)
      PARAMETER(N=13,K=3)
      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(N,K),RATE(N),A(*)
C     -----------------------------------------------------------------
C     HOUGEN FUNCTION (HOUGEN-WATSON MODEL FOR REACTION KINATICS)
C     NO. OF PARAMETERS (A) TO ESTIMATE = 5 = M

C     BEST RESULTS ARE:
C     A(1)=1.253031; A(2)=1.190943; A(3)=0.062798; A(4)=0.040063
C     A(5)=0.112453  ARE BEST ESTIMATES OBTAINED BY Rosenbrock &
C     Quasi-Newton METHOD WITH SUM OF SQUARES OF DEVIATION =0.298900994
C     AND R=0.99945.

C     THE NEXT BEST RESULTS GIVEN BY Hooke-Jeeves & Quasi-Newton
C     A(1)=2.475221;A(2)=0.599177; A(3)=0.124172; A(4)=0.083517
C     A(5)=0.217886; SUM OF SQUARES OF DEVIATION = 0.318593458
C     R=0.99941
C     MOST OF THE OTHER METHODS DO NOT PERFORM WELL
C     ------------------------------------------------------------------
      data x(1,1),x(1,2),x(1,3),rate(1) /470,300,10,8.55/
```

```
      data x(2,1),x(2,2),x(2,3),rate(2) /285,80,10,3.79/
      data x(3,1),x(3,2),x(3,3),rate(3) /470,300,120,4.82/
      data x(4,1),x(4,2),x(4,3),rate(4) /470,80,120,0.02/
      data x(5,1),x(5,2),x(5,3),rate(5) /470,80,10,2.75/
      data x(6,1),x(6,2),x(6,3),rate(6) /100,190,10,14.39/
      data x(7,1),x(7,2),x(7,3),rate(7) /100,80,65,2.54/
      data x(8,1),x(8,2),x(8,3),rate(8) /470,190,65,4.35/
      data x(9,1),x(9,2),x(9,3),rate(9) /100,300,54,13/
      data x(10,1),x(10,2),x(10,3),rate(10) /100,300,120,8.5/
      data x(11,1),x(11,2),x(11,3),rate(11) /100,80,120,0.05/
      data x(12,1),x(12,2),x(12,3),rate(12) /285,300,10,11.32/
      data x(13,1),x(13,2),x(13,3),rate(13) /285,190,120,3.13/
C     WRITE(*,1)((X(I,J),J=1,K),RATE(I),I=1,N)
C   1 FORMAT(4F8.2)

      F=0.D00
      fp=0.d00
      DO I=1,N
      D=1.D00
         DO J=1,K
         D=D+A(J+1)*X(I,J)
         ENDDO
      FX=(A(1)*X(I,2)-X(I,3)/A(M))/D
C     FX=(A(1)*X(I,2)-X(I,3)/A(5))/(1.D00+A(2)*X(I,1)+A(3)*X(I,2)+
C     A(4)*X(I,3))
      F=F+(RATE(I)-FX)**2
      ENDDO
      do j=1,m
      if(dabs(a(j)).gt.5.d00) fp=fp+dexp(dabs(a(j)))
      enddo
      if(fp.gt.0.d00) f=fp
      RETURN
      END
```