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

Mishra, Sudhanshu

North-Eastern Hill University, Shillong (India)

23 August 2006

Online at https://mpra.ub.uni-muenchen.de/2718/
MPRA Paper No. 2718, posted 13 Apr 2007 UTC
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.
- $\omega$ 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.
- $\hat{x}_{gi}$ 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:

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

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

where,

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

Occasionally, when the process is caught in a local optimum, some 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 \( x^* = -10.8723 \) in the domain \( x_i \in [-10, 10], \ i = 1, 2 \).

\[
f(x) = -4 \left| \sin(x_1) \cos(x_2) e^{\cos((x_1^2 + x_2^2)/200)} \right|
\]

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

\[
f(x) = -4 \left| \sin(x_1) \cos(x_2) e^{\cos((x_1^2 + x_2^2)/200)} \right|
\]

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

\[
f(x) = -\left[ \cos(x_1) \cos(x_2) e^{\frac{\sqrt{(x_1^2 + x_2^2)}}{\pi}} \right]
\]

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

\[
f(x) = -\left[ \cos(x_1) \cos(x_2) e^{\frac{\sqrt{(x_1^2 + x_2^2)}}{\pi}} \right]^2 / 30
\]

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

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

6. Crowned cross function: This function is the negative form of the cross in tray function. It has \( f(x^*) = 0 \) 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:

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

7. Cross function: This is a multi-modal function with \( f(x^*) = 0 \). It is given as
$f(x) = \left[ \sin(x_1) \sin(x_2) e^{\frac{100 - [(x_1^2 + x_2^2) / 2]}{1}} + 1 \right]^{-0.1}$

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 $f(x') = -1$ 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:

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

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

$$f(x) = -\exp\left[ -\left[ \cos(x_1) \cos(x_2) e^{\frac{100 - [(x_1^2 + x_2^2) / 2]}{1}} \right]^{-1} \right]$$

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

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

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

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

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

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

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

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

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

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

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

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

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.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>470</td>
<td>300</td>
<td>10</td>
<td>8.55</td>
</tr>
<tr>
<td>285</td>
<td>80</td>
<td>10</td>
<td>3.79</td>
</tr>
<tr>
<td>470</td>
<td>300</td>
<td>120</td>
<td>4.82</td>
</tr>
<tr>
<td>470</td>
<td>80</td>
<td>120</td>
<td>0.02</td>
</tr>
<tr>
<td>470</td>
<td>80</td>
<td>10</td>
<td>2.75</td>
</tr>
<tr>
<td>100</td>
<td>190</td>
<td>10</td>
<td>14.39</td>
</tr>
<tr>
<td>100</td>
<td>80</td>
<td>65</td>
<td>2.54</td>
</tr>
<tr>
<td>470</td>
<td>190</td>
<td>65</td>
<td>4.35</td>
</tr>
<tr>
<td>100</td>
<td>300</td>
<td>54</td>
<td>13.00</td>
</tr>
<tr>
<td>100</td>
<td>80</td>
<td>120</td>
<td>0.05</td>
</tr>
<tr>
<td>285</td>
<td>300</td>
<td>10</td>
<td>11.32</td>
</tr>
<tr>
<td>285</td>
<td>190</td>
<td>120</td>
<td>3.13</td>
</tr>
</tbody>
</table>

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 $\gamma = a_3 = 0.0005$ and $\omega = 0.05$, 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:

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

We obtain $f_{\min}(512, 404.2319) = 959.64$. 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:

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

It is a difficult problem to optimize. For higher dimensions it gives repeating coupllets 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 \( f_{\text{min}}(5.90133, 0.5) = -43.3159 \).

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

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 \( f_{\text{min}}(-0.54719, -1.54719) = -1.9133 \).

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

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

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

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 \( f_{\text{min}}(0, 0) = 0 \).

\[
f(x) = 2x_1^2 - 1.05x_1^4 + x_1^6 / 6 + x_1x_2 + x_2^2.
\]

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

\[
f(x) = (x_1^2 + x_2^2 - 2x_1)^2 + 0.25x_1
\]

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

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

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.

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

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

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

\[
f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2
\]

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

\[
f(x) = 0.6 + \sum_{i=1}^{2} [\sin(16x_i - 1) + \sin^2(16x_i - 1) + \frac{1}{30}\sin(4(16x_i - 1))].
\]

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

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

\[
f(x) = 0.6 + \sum_{i=1}^{2} [\sin(16x_i - 1) + \sin^2(16x_i - 1) + \frac{1}{30}\sin(4(16x_i - 1))].
\]
\[ f(x) = 0.5 + \frac{\sin^2 \left[ \sqrt{x_1^2 + x_2^2} \right]}{[1 + 0.001(x_1^2 + x_2^2)]^2} - 0.5. \]

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.
<table>
<thead>
<tr>
<th>Pen holder function</th>
<th>Bird function</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Pen holder function" /></td>
<td><img src="image2" alt="Bird function" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Crowned Cross function</th>
<th>Cross in Tray function</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3" alt="Crowned Cross function" /></td>
<td><img src="image4" alt="Cross in Tray function" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cross function</th>
<th>Inverted Cross function</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image5" alt="Cross function" /></td>
<td><img src="image6" alt="Inverted Cross function" /></td>
</tr>
</tbody>
</table>
Some Well-established Benchmark Functions

<table>
<thead>
<tr>
<th>Bukin function (#6)</th>
<th>Bukin function (#4)</th>
</tr>
</thead>
</table>

- **Schaffer function**
- **Levy function (#13)**

- **Leon function**
- **3-hump camel back function**
<table>
<thead>
<tr>
<th>Zettl function</th>
<th>Styblinski-Tang function</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Zettl function" /></td>
<td><img src="image2" alt="Styblinski-Tang function" /></td>
</tr>
<tr>
<td>Chichinadze function</td>
<td>McCormick function</td>
</tr>
<tr>
<td><img src="image3" alt="Chichinadze function" /></td>
<td><img src="image4" alt="McCormick function" /></td>
</tr>
<tr>
<td>Giunta function</td>
<td>Egg holder function</td>
</tr>
<tr>
<td><img src="image5" alt="Giunta function" /></td>
<td><img src="image6" alt="Egg holder function" /></td>
</tr>
<tr>
<td>Schaffer function (Closer view)</td>
<td>Modified Schaffer function #1</td>
</tr>
<tr>
<td><img src="image7" alt="Schaffer function (Closer view)" /></td>
<td><img src="image8" alt="Modified Schaffer function #1" /></td>
</tr>
</tbody>
</table>
Bibliography

- 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


Author’s Contact: mishrasknehu@hotmail.com
PROGRAM TO FIND GLOBAL MINIMUM BY REPELLENT PARTICLE SWARM METHOD

PARAMETER (N=50, NN=25, MX=100, NSTEP=21, ITRN=5000)

N - POPULATION SIZE. IN MOST OF THE CASES N=30 IS OK. ITS VALUE
MAY BE INCREASED TO 50 ALSO. THE PARAMETER NN IS THE SIZE OF
RANDOMLY CHOSEN NEIGHBOURS. 15 TO 25 (BUT SUFFICIENTLY LESS THAN
N) IS A GOOD CHOICE. MX IS THE MAXIMAL SIZE OF DECISION VARIABLES.
IN F(X1, X2, ..., XM) M SHOULD BE LESS THAN OR EQUAL TO MX. ITRN IS
THE NO. OF ITERATIONS. IT MAY DEPEND ON THE PROBLEM. 200 TO 500
ITERATIONS MAY BE GOOD ENOUGH. BUT FOR FUNCTIONS LIKE ROSENBROCK
OR GRIEWANK OF LARGE SIZE (SAY M=20) IT IS NEEDED THAT ITRN IS
LARGE, SAY 5000 OR 10000.

THE SUBROUTINE FUNC() DEFINES THE FUNCTION TO BE OPTIMIZED.

IMPLICIT DOUBLE PRECISION (A-H,O-Z)

COMMON /RNDM/IV4, IV
COMMON /KFF/KF
INTEGER IU4, IV
DIMENSION X(N, MX), V(N, MX), A(MX), VII(MX), TIT(50)
DIMENSION XX(N, MX), F(N), R(3), V1(MX), V2(MX), V3(MX), V4(MX), BST(MX)
CHARACTER *70 TIT

KF AND A3 ARE CONSTANTS AND W IS THE INERTIA WEIGHT.

OCCASIONALLY, TINKERING WITH THESE VALUES, ESPECIALLY A3, MAY BE
NEEDED.

WRITE(*,*)'CHOOSE KF AND SPECIFY M'

KFF=5 CROSS IN TRAY FUNCTION 2-VARIABLES M=2'
KFF=4 CARROM TABLE FUNCTION 2-VARIABLES M=2'
KFF=2 TEST TUBE HOLDER FUNCTION (B) 2-VARIABLES M=2'
KFF=25 EGG-HOLDER FUNCTION M=?'
KFF=23 HOUGEN REGRESSION FUNCTION 5-VARIABLE M=5'
KFF=22 BUKIN-6 FUNCTION 2-VARIABLE M=2'
KFF=19 ZETTLE FUNCTION 2-VARIABLE M=2'
KFF=14 SCHAFFER FUNCTION 2-VARIABLE M=2'
KFF=13 GIUNTA FUNCTION 2-VARIABLE M=2'
KFF=12 LEON FUNCTION 2-VARIABLE M=2'
KFF=11 DE JONG SPHERE FUNCTION M-VARIABLE M=?'
KFF=10 BIRD FUNCTION 2-VARIABLE M=2'
KFF=9 PEN HOLDER FUNCTION 2-VARIABLE M=2'
KFF=8 CROSS-LEGGED TABLE FUNCTION 2-VARIABLES M=2'
KFF=7 CROSS FUNCTION 2-VARIABLES M=2'
KFF=6 CROWNED CROSS FUNCTION 2-VARIABLES M=2'
KFF=5 CROSS IN TRAY FUNCTION 2-VARIABLES M=2'
KFF=4 CARROM TABLE FUNCTION 2-VARIABLES M=2'
KFF=3 HOLDER TABLE FUNCTION 2-VARIABLES M=2'
KFF=2 TEST TUBE HOLDER FUNCTION(A) 2-VARIABLES M=2'
KFF=1 DE JONG SPHERE FUNCTION M-VARIABLE M=?'

WRITE(*,*)'----------------------------------------------------'

DATA A1, A2, A3, W / .5D00, .5D00, .0005D00, .5D00/
WRITE(*,*)'-----------------------------------------------'
WRITE(*,*)'KF=1 TEST TUBE HOLDER FUNCTION(A) 2-VARIABLES M=2'
WRITE(*,*)'KF=2 TEST TUBE HOLDER FUNCTION(B) 2-VARIABLES M=2'
WRITE(*,*)'KF=3 HOLDER TABLE FUNCTION 2-VARIABLES M=2'
WRITE(*,*)'KF=4 CARROM TABLE FUNCTION 2-VARIABLES M=2'
WRITE(*,*)'KF=5 CROSS IN TRAY FUNCTION 2-VARIABLES M=2'
WRITE(*,*)'KF=6 CROWNED CROSS FUNCTION 2-VARIABLES M=2'
WRITE(*,*)'KF=7 CROSS FUNCTION 2-VARIABLES M=2'
WRITE(*,*)'KF=8 CROSS-LEGGED TABLE FUNCTION 2-VARIABLES M=2'
WRITE(*,*)'KF=9 PEN HOLDER FUNCTION 2-VARIABLES M=2'
WRITE(*,*)'KF=10 BIRD FUNCTION 2-VARIABLES M=2'
WRITE(*,*)'KF=11 DE JONG SPHERE FUNCTION M-VARIABLE M=?'
WRITE(*,*)'KF=12 LEON FUNCTION 2-VARIABLE M=2'
WRITE(*,*)'KF=13 GIUNTA FUNCTION 2-VARIABLE M=2'
WRITE(*,*)'KF=14 SCHAFFER FUNCTION 2-VARIABLE M=2'
WRITE(*,*)'KF=15 CHICHINADZE FUNCTION 2-VARIABLE M=2'
WRITE(*,*)'KF=16 MCCORMICK FUNCTION 2-VARIABLE M=2'
WRITE(*,*)'KF=17 LEVY # 13 FUNCTION 2-VARIABLE M=2'
WRITE(*,*)'KF=18 3-HUMP CAMEL BACK FUNCTION 2-VARIABLE M=2'
WRITE(*,*)'KF=19 ZETTLE FUNCTION 2-VARIABLE M=2'
WRITE(*,*)'KF=20 STYBLINSKI-TANG FUNCTION 2-VARIABLE M=2'
WRITE(*,*)'KF=21 BUKIN-4 FUNCTION 2-VARIABLE M=2'
WRITE(*,*)'KF=22 BUKIN-6 FUNCTION 2-VARIABLE M=2'
WRITE(*,*)'KF=23 HOUGEN REGRESSION FUNCTION 5-VARIABLE M=5'
WRITE(*,*)'KF=24 SINE ENVELOPE SINE WAVE FUNCTION M=?'
WRITE(*,*)'KF=25 EGG-HOLDER FUNCTION M=?'
WRITE(*,*)'KF=26 MODIFIED SCHAFFER FUNCTION #1 2-VARIABLE M=2'
WRITE(*,*)'KF=27 MODIFIED SCHAFFER FUNCTION #2 2-VARIABLE M=2'
WRITE(*,*)'KF=28 MODIFIED SCHAFFER FUNCTION #3 2-VARIABLE M=2'
WRITE(*,*)'KF=29 MODIFIED SCHAFFER FUNCTION #4 2-VARIABLE M=2'
WRITE(*,*)'KF=30 QUARTIC(NOISE) FUNCTION M-VARIABLE M=?'

DO I=1,20
WRITE(*,*)TIT(I)
ENDDO
WRITE(*,*)'-----------------------------------------------'
WRITE(*,*)'CHOOSE KF AND SPECIFY M'
READ(*,*) KF, M
DSIGN=1.000
LCOUNT=0
WRITE(*,*)'4-DIGITS SEED FOR RANDOM NUMBER GENERATION'
READ(*,*) IU
DATA ZERO,ONE,FMIN /0.0D00,1.0D00,1.0E30/
C GENERATE M-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
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)
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)
ENDIF
ENDDO
C F(I) CONTAINS THE LOCAL BEST VALUE OF FUNCTION FOR ITH INDIVIDUAL
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
ENDDO
C NOW LET EVERY INDIVIDUAL RANDOMLY CONSULT NN(<N) COLLEAGUES AND
C FIND THE BEST AMONG THEM
DO I=1,N
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
DO J=1,M
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
C A3 IS THE CONSTANT CLOSE TO BUT LESS THAN UNITY
C THEN SOME RANDOMNESS AND A CONSTANT A3 CLOSE TO BUT LESS THAN UNITY
C THEN ON PAST VELOCITY WITH INERTIA WEIGHT W
C V3(J)=A3*RAND*W
C THEN BASED ON THE OTHER COLLEAGUES BEST EXPERIENCE WITH WEIGHT W

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
C CALL RANDOM(RAND)
RND1=RAND
C 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)

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)
   ENDIF
ENDDO

DO J=1,M
   BST(J)=XX(J,J)
ENDDO

CALL RND1

WRITE(*,*)'INITIAL VALUE OF X'
WRITE(*,*)'INITIAL VALUE OF F'
WRITE(*,*)'INITIAL VALUE OF FMIN'
WRITE(*,*)'INITIAL VALUE OF W'
WRITE(*,*)'INITIAL VALUE OF A2'
WRITE(*,*)'INITIAL VALUE OF A3'
WRITE(*,*)'INITIAL VALUE OF KS'
WRITE(*,*)'INITIAL VALUE OF NSTEP'
WRITE(*,*)'INITIAL VALUE OF AMN'
WRITE(*,*)'INITIAL VALUE OF KF'
WRITE(*,*)'INITIAL VALUE OF IU'
WRITE(*,*)'INITIAL VALUE OF IV'
WRITE(*,*)'INITIAL VALUE OF JU'
WRITE(*,*)'INITIAL VALUE OF JV'
WRITE(*,*)'INITIAL VALUE OF MP'
WRITE(*,*)'INITIAL VALUE OF NFBEST'
WRITE(*,*)'INITIAL VALUE OF IU,'
WRITE(*,*)'INITIAL VALUE OF IV,'
WRITE(*,*)'INITIAL VALUE OF JU,'
WRITE(*,*)'INITIAL VALUE OF JV,'
WRITE(*,*)'INITIAL VALUE OF MP,'
WRITE(*,*)'INITIAL VALUE OF NFBEST,'
WRITE(*,*)'INITIAL VALUE OF IU,'
WRITE(*,*)'INITIAL VALUE OF IV,'
WRITE(*,*)'INITIAL VALUE OF JU,'
WRITE(*,*)'INITIAL VALUE OF JV,'
WRITE(*,*)'INITIAL VALUE OF MP,'
WRITE(*,*)'INITIAL VALUE OF NFBEST,'
WRITE(*,*)'INITIAL VALUE OF IU,'
WRITE(*,*)'INITIAL VALUE OF IV,'
WRITE(*,*)'INITIAL VALUE OF JU,'
WRITE(*,*)'INITIAL VALUE OF JV,'
WRITE(*,*)'INITIAL VALUE OF MP,'
WRITE(*,*)'INITIAL VALUE OF NFBEST,'
WRITE(*,*)'INITIAL VALUE OF IU,'
WRITE(*,*)'INITIAL VALUE OF IV,'
WRITE(*,*)'INITIAL VALUE OF JU,'
WRITE(*,*)'INITIAL VALUE OF JV,'
WRITE(*,*)'INITIAL VALUE OF MP,'
WRITE(*,*)'INITIAL VALUE OF NFBEST,'
WRITE(*,*)'INITIAL VALUE OF IU,'
WRITE(*,*)'INITIAL VALUE OF IV,'
WRITE(*,*)'INITIAL VALUE OF JU,'
WRITE(*,*)'INITIAL VALUE OF JV,'
WRITE(*,*)'INITIAL VALUE OF MP,'
WRITE(*,*)'INITIAL VALUE OF NFBEST,'
WRITE(*,*)'INITIAL VALUE OF IU,'
WRITE(*,*)'INITIAL VALUE OF IV,'
WRITE(*,*)'INITIAL VALUE OF JU,'
WRITE(*,*)'INITIAL VALUE OF JV,'
WRITE(*,*)'INITIAL VALUE OF MP,'
WRITE(*,*)'INITIAL VALUE OF NFBEST,'
WRITE(*,*)'INITIAL VALUE OF IU,'
WRITE(*,*)'INITIAL VALUE OF IV,'
WRITE(*,*)'INITIAL VALUE OF JU,'
WRITE(*,*)'INITIAL VALUE OF JV,'
WRITE(*,*)'INITIAL VALUE OF MP,'
WRITE(*,*)'INITIAL VALUE OF NFBEST,'
WRITE(*,*)'INITIAL VALUE OF IU,'
WRITE(*,*)'INITIAL VALUE OF IV,'
WRITE(*,*)'INITIAL VALUE OF JU,'
WRITE(*,*)'INITIAL VALUE OF JV,'
WRITE(*,*)'INITIAL VALUE OF MP,'
WRITE(*,*)'INITIAL VALUE OF NFBEST,'
WRITE(*,*)'INITIAL VALUE OF IU,'
WRITE(*,*)'INITIAL VALUE OF IV,'
WRITE(*,*)'INITIAL VALUE OF JU,'
WRITE(*,*)'INITIAL VALUE OF JV,'
WRITE(*,*)'INITIAL VALUE OF MP,'
WRITE(*,*)'INITIAL VALUE OF NFBEST,'
WRITE(*,*)'INITIAL VALUE OF IU,'
WRITE(*,*)'INITIAL VALUE OF IV,'
WRITE(*,*)'INITIAL VALUE OF JU,'
WRITE(*,*)'INITIAL VALUE OF JV,'
WRITE(*,*)'INITIAL VALUE OF MP,'
WRITE(*,*)'INITIAL VALUE OF NFBEST,'
WRITE(*,*)'INITIAL VALUE OF IU,'
WRITE(*,*)'INITIAL VALUE OF IV,'
WRITE(*,*)'INITIAL VALUE OF JU,'
WRITE(*,*)'INITIAL VALUE OF JV'
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
   ENDIF
   DO JJ=1,M
      A(JJ)=B(JJ)
   ENDDO
ENDDO

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=DOUBLE(RAND)
RETURN
END

SUBROUTINE FUNC(X,M,F)
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)

IF(KF.EQ.1) THEN
  Test-tube holder function (A)
  FP=0.D00
  C=-10 TO 10 M=2
  F=0.D00
  IF (x(1).lt.-10.0d0 .or. x(1).gt. 10.0d0) fp=fp+x(1)**2
  IF (x(2).lt.-10.0d0 .or. x(2).gt. 10.0d0) fp=fp+x(1)**2
  IF (FP.GT.0.D00) THEN
    f=FP
  ELSE
    $=4*dabs(dsin(X(1))*dcos(x(2)))*dexp(dabs(dcos((X(1)**2+x(2)**2))/2))
  ENDIF
  RETURN
ENDIF

IF(KF.EQ.2) THEN
  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
    $=4*dabs(dsin(X(1))*dcos(x(2)))*dexp(dabs(dcos((X(1)**2+x(2)**2))/2))
  ENDIF
  RETURN
ENDIF

IF(KF.EQ.3) THEN
  Holder table function
  FP=0.D00
  C=-10 TO 10 M=2
  F=0.D00
  DO I=1,M
269:     IF( DABS(X(I)) .GT. 10.D00) FP=FP+X(I)**2
270:     ENDDO
271:     IF(FP .GT. 0.D00) THEN
272:       F=FP
273:     ELSE
274:       f=-dabs(dcos(x(1))*dcos(x(2)))*dexp(dabs(1.d00-(dsqrt(x(1)**2+
275:         & x(2)**2)/pi))))
276:     ENDIF
277:     RETURN
278:     ENDIF
279:     C
280:     IF(KF.EQ.4) THEN
281:       Carrom table function
282:       FP=0.D00
283:       C -10 TO 10 M=2
284:       F=0.D00
285:       DO I=1,M
286:       IF( DABS(X(I)) .GT. 10.D00) FP=FP+X(I)**2
287:       ENDDO
288:       IF(FP .GT. 0.D00) THEN
289:       F=FP
290:     ELSE
291:       f=-1.d00/30*(dcos(X(1))*dcos(x(2)))*dexp(dabs(1.d00-
292:         & (dsqrt(X(1)**2 + x(2)**2)/pi))))**2
293:     ENDIF
294:     RETURN
295:     ENDIF
296:     C
297:     IF(KF.EQ.5) THEN
298:       FP=0.D00
299:     Cross in tray function
300:     C -10 TO 10 M=2
301:     F=0.D00
302:     DO I=1,M
303:     IF( DABS(X(I)) .GT. 10.D00) FP=FP+X(I)**2
304:     ENDDO
305:     IF(FP .GT. 0.D00) THEN
306:     F=FP
307:     ELSE
308:     f=-0.0001d00*(dabs(dsin(X(1)))*dsin(x(2)))*dexp(dabs(100.d00-(dsqrt
309:         & (X(1)**2+x(2)**2)/pi)))+1.d00)**(.1)
310:     ENDIF
311:     RETURN
312:     ENDIF
313:     C
314:     IF(KF.EQ.6) THEN
315:       FP=0.D00
316:     Crowned Cross function
317:     C -10 TO 10 M=2
318:     F=0.D00
319:     DO I=1,M
320:     IF( DABS(X(I)) .GT. 10.d00) FP=FP+dexp(dabs(X(I)))
321:     ENDDO
322:     IF(FP .GT. 0.D00) THEN
323:     F=FP
324:     ELSE
325:     f=0.0001d00*(dabs(dsin(X(1)))*dsin(x(2)))*dexp(dabs(100.d00-
326:         & (dsqrt(X(1)**2+x(2)**2)/pi)))+1.d00)**(.1)
327:     ENDIF
328:     RETURN
329:     ENDIF
330:     C
331:     IF(KF.EQ.7) THEN
332:       FP=0.D00
333:     Cross function
334:     C -10 TO 10 M=2
335:     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     DE JONG SPHERE function
C     -----------------------------------------------------------------
C     -2Pi TO 2Pi M=2
C     Bird function
C     -----------------------------------------------------------------
C     -11 TO 11 M=2
C     Pen holder function
C     -----------------------------------------------------------------
C     -10 TO 10 M=2
C     Cross-legged table function

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
C     -----------------------------------------------------------------
C     -11 TO 11 M=2
C     DO I=1,M
C     IF(DABS(X(I)) .GT. 11.d00) FP=FP+X(I)**2
ENDDO

IF(FP .GT. 0.d00) THEN
  F=FP
ELSE
  f=-dexp(-((Dcos(X(1))*Dcos(X(2)))*Dexp(Dabs(1.d00-(Dsqrt(X(1)**2+X(2)**2)/pi))))**(-1)))
ENDIF
RETURN
ENDIF

C     IF(KF.EQ.10) THEN
C     Bird function
C     -----------------------------------------------------------------
C     -2Pi TO 2Pi M=2
C     DO I=1,M
C     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
C     -----------------------------------------------------------------
C     -2Pi TO 2Pi M=2
C     DO I=1,M

ENDIF
403: F=F+X(I)**2
404: ENDDO
405: RETURN
406: ENDF
407: C
408: IF(KF.EQ.12) THEN
409: C Leap function
410: FP=0.D00
411: C -1.2 TO 1.2 M=2
412: DO I=1,M
413: IF(DABS(X(I)).GT.1.2d00) FP=FP+X(I)**2
414: ENDDO
415: IF(FP.GT.0.D00) THEN
416: F=FP
417: ELSE
418: f=100*(x(2)-x(1)**2)**2+(1.d00-x(1))**2
419: ENDIF
420: RETURN
421: ENDF
422: C
423: IF(KF.EQ.13) THEN
424: C Giunta function
425: FP=0.D00
426: C -1 TO 1 M=2
427: DO I=1,M
428: IF(DABS(X(I)).GT.1.d00) FP=FP+X(I)**2
429: ENDDO
430: IF(FP.GT.0.D00) THEN
431: F=FP
432: ELSE
433: c=15.d00/15.d00
434: f=dsin(c*x(1)-1.d00)+dsin(c*x(1)-1.d00)**2+dsin(4*(c*x(1)-1.d00))/50+
435: &dsin(c*x(2)-1.d00)+dsin(c*x(2)-1.d00)**2+dsin(4*(c*x(2)-1.d00))/50+.6
436: ENDIF
437: RETURN
438: ENDF
439: C
440: IF(KF.EQ.14) THEN
441: C Schaffer function
442: FP=0.D00
443: C -100 TO 100 M=2
444: DO I=1,M
445: IF(DABS(X(I)).GT.100.d00) FP=FP+X(I)**2
446: ENDDO
447: IF(FP.GT.0.D00) THEN
448: F=FP
449: ELSE
450: f1=dsin(dsqr(x(1)**2+x(2)**2))*2-0.5d00
451: f2=(1.d00+ 0.001*(x(1)**2 + x(2)**2))*2
452: f=f1/f2 +0.5d00
453: ENDIF
454: RETURN
455: ENDF
456: C
457: IF(KF.EQ.15) THEN
458: C Chichinadze function
459: FP=0.D00
460: C 30 < X(I) < 30 M=2
461: DO I=1,M
462: IF(DABS(X(I)).GT.30.d00) FP=FP+X(I)**2
463: ENDDO
464: IF(FP.GT.0.D00) THEN
465: F=FP
466: ELSE
467: f=x(1)**2-12*x(1)+11.d00+10*DCOS(PI*X(1)/2)+8*DSIN(5*PI*X(1))-+
468: & (1/DSQRT(5.d00))**DEXP(-(X(2)-0.5d00)**2/2)
535:  ENDIF
536:  RETURN
537:  ENDIF
538:  C
539:  IF(KF.EQ.16) THEN
540:    McCormick function
541:    FP=0.0D0
542:  ENDIF
543:  C
544:  -1.5<=X(1)<=4; -3<=X(2)<=4 ; M=2
545:  IF(X(1).LT. -1.5D00  .OR.  X(1).GT. 4.D00) FP=FP*X(1)**2
546:  IF(X(2).LT. -3.D00  .OR.  X(2).GT. 4.D00) FP=FP*X(2)**2
547:  IF(FP.GT.0.0D0) THEN
548:    F=FP
549:  ELSE
550:    f=DSIN(X(1)+X(2))+(X(1)-X(2))**2-1.5*X(1)+2.5*X(2)+1.D00
551:  ENDIF
552:  RETURN
553:  ENDIF
554:  C
555:  IF(KF.EQ.17) THEN
556:    Levy #13 function
557:    FP=0.0D0
558:  ENDIF
559:  C
560:  -10<=X(I)<=10 M=2
561:  DO I=1,M
562:    IF(DABS(X(I)).GT.10.D00) FP=FP+X(I)**2
563:  ENDDO
564:  IF(FP.GT.0.0D0) THEN
565:    F=FP
566:  ELSE
567:    f=DSIN(3*PI*X(1))**2+(X(1)-1.D00)**2 *(1.D00+DSIN(3*PI*X(2))**2) +
568:    (X(2)-1.D00)**2 *(1.D00+DSIN(2*PI*X(2))**2)
569:  ENDIF
570:  RETURN
571:  ENDIF
572:  C
573:  IF(KF.EQ.18) THEN
574:    Three-hump Camel back function
575:    FP=0.0D0
576:  ENDIF
577:  C
578:  -5<=X(1)<=5 M=2
579:  DO I=1,M
580:    IF(DABS(X(I)).GT.5.D00) FP=FP+X(I)**2
581:  ENDDO
582:  IF(FP.GT.0.0D0) THEN
583:    F=FP
584:  ELSE
585:    f=2*X(1)**2-1.05*X(1)**4+X(1)**6/6 + X(1)*X(2)+X(2)**2
586:  ENDIF
587:  RETURN
588:  ENDIF
589:  C
590:  IF(KF.EQ.19) THEN
591:    Test function
592:    FP=0.0D0
593:  ENDIF
594:  C
595:  -5 <= X(I) <= 5 M=2
596:  DO I=1,M
597:    IF(DABS(X(I)).GT.5.D00) FP=FP+X(I)**2
598:  ENDDO
599:  IF(FP.GT.0.0D0) THEN
600:    F=FP
601:  ELSE
602:    f=(X(1)**2+X(2)**2-2*X(1))**2 + 0.25*X(1)
603:  ENDIF
604:  RETURN
605:  ENDIF
606:  C
607:  IF(KF.EQ.20) THEN
608:    Styblinski-Tang function
609:    FP=0.0D0
610:  ENDIF
C     SINE ENVELOPE SINE WAVE FUNCTION (Generalized Schaffer)
C     -----------------------------------------------------------------
C     NO. OF PARAMETERS TO ESTIMATE = 5 = M
C     HOUGEN FUNCTION (HOUGEN-WATSON MODEL FOR REACTION KINATICS)
C     -----------------------------------------------------------------
C     BUKIN-6 function
C     -----------------------------------------------------------------
C     -15. LE. X(1) .LE. -5 AND -3 .LE. X(2) .LE. 3
C     -----------------------------------------------------------------
C     -5 <=X(I) <=5 M=2

536:  C       -5 <=X(I) <=5 M=2
537:      DO I=1,M
538:      IF(DABS(X(I)).GT.5.d00) FP=FP+X(I)**2
539:      ENDDO
540:      IF(FP.GT.0.d00) THEN
541:      F=FP
542: ELSE
543: F=0.d00
544: DO I=1,M
545:  e=fp+(x(i)**4-16*x(i)**2+5*x(i))
546: ENDDO
547:  F=F/2
548:  ENDIF
549:  RETURN
550:  ENDIF

551:  C------------------------------------------------------------------------
552:  C IF(KF.EQ.21) THEN
553:  C SUNKIN-4 function
554:  FP=0.d00
555:  C------------------------------------------------------------------------
556:  C -15. LE. X(1) .LE. -5 AND -3 .LE. X(2) .LE. 3
557:  C------------------------------------------------------------------------
558:  IF(X(1).LT.-15.d00 .OR. X(1).GT.-5.d00) FP=FP+X(1)**2
559:  IF(DABS(X(2)).GT.3.d00) FP=FP+X(2)**2
560:  IF(FP.GT.0.d00) THEN
561:  F=FP
562: ELSE
563: F=100*X(2)**2 + 0.01*DABS(X(1)+10.d0)
564: ENDIF
565: RETURN
566: ENDIF

567:  C------------------------------------------------------------------------
568:  C IF(KF.EQ.22) THEN
569:  C SUNKIN-5 function
570:  FP=0.d00
571:  C------------------------------------------------------------------------
572:  C -15. LE. X(1) .LE. -5 AND -3 .LE. X(2) .LE. 3
573:  C------------------------------------------------------------------------
574:  IF(X(1).LT.-15.d00 .OR. X(1).GT.-5.d00) FP=FP+X(1)**2
575:  IF(DABS(X(2)).GT.3.d00) FP=FP+X(2)**2
576:  IF(FP.GT.0.d00) THEN
577:  F=FP
578: ELSE
579: F=100*DSQRT(DABS(X(2)-0.01d00*X(1)**2))+ 0.01*DABS(X(1)+10.d0)
580: ENDIF
581: RETURN
582: ENDIF

583:  C------------------------------------------------------------------------
584:  C IF(KF.EQ.23) THEN
585:  C HOUGEN FUNCTION (HOUGEN-WATSON MODEL FOR REACTION KINATICS)
586:  C NO. OF PARAMETERS TO ESTIMATE = 5 = M
587:  C------------------------------------------------------------------------
588:  CALL HOUGEN(M,X,F)
589: RETURN
590: ENDIF

591:  C------------------------------------------------------------------------
592:  C IF(KF.EQ.24) THEN
593:  C SINE ENVELOPE SINE WAVE FUNCTION (Generalized Schaffer)
594:  IF(1.d00*d00)
595: do 1=1,m=1
596:  e1=d1n(dsqr(x(i)**2+x(i)**2))**2-0.5d00
597:  e2=(0.001d00*(x(i)**2+x(i)**2)+1.d00)**2
598:  e=1/(1/f2)+0.5d00
599:  enddo
600: do 1=1,m
601: if(dabs(x(i)).GT.100.d00) fp=fp+x(i)**2
602: enddo
603:     IF(fp.gt.0.d00) fp=fp  
604:     return  
605:     endif  
606:     C---------------------------------------------------------------------
607:     C     Modified (crossed) Schaffer function #3  
608:     C---------------------------------------------------------------------
609:     C     Modified (Hyperbolized) Schaffer function #2  
610:     C---------------------------------------------------------------------
611:     C     Modified Schaffer function #1  
612:     C---------------------------------------------------------------------
613:     EGG HOLDER FUNCTION  
614:     C---------------------------------------------------------------------
615:     C     EGG HOLDER FUNCTION  
616:     C---------------------------------------------------------------------
617:     IF(KF.EQ.25) THEN  
618:     ELSE  
619:       f1=x(I)**2+x(J)**2  
620:     ENDIF  
621:     f2=x(I)**2+0.5d00  
622:     ENDIF  
623:     C---------------------------------------------------------------------
624:     C     Modified Schaffer function #1  
625:     C---------------------------------------------------------------------
626:     IF(KF.EQ.26) THEN  
627:     ELSE  
628:       f1=x(I)**2  
629:       f2=(1.d00+0.001*(x(I)**2+x(J)**2))**2  
630:     ENDIF  
631:     RETURN  
632:     ENDIF  
633:     C---------------------------------------------------------------------
634:     C     Modified (Hyperbolized) Schaffer function #2  
635:     C---------------------------------------------------------------------
636:     IF(KF.EQ.27) THEN  
637:     ELSE  
638:       f1=x(I)**2-x(J)**2  
639:       f2=(1.d00+0.001*(x(I)**2+x(J)**2))**2  
640:     ENDIF  
641:     RETURN  
642:     ENDIF  
643:     C---------------------------------------------------------------------
644:     C     Modified (crossed) Schaffer function #3  
645:     C---------------------------------------------------------------------
646:     IF(KF.EQ.28) THEN  
647:     ELSE  
648:       f1=x(I)**2  
649:       f2=(1.d00+0.001*(x(I)**2-x(J)**2))**2-0.5d00  
650:     ENDIF  
651:     ENDIF  
652:     C---------------------------------------------------------------------
C     MOST OF THE OTHER METHODS DO NOT PERFORM WELL
C
C     A(1)=2.475221; A(2)=0.599177; A(3)=0.124172; A(4)=0.083517
C     AND R=0.99945.
C     A(5)=0.112453 ARE BEST ESTIMATES OBTAINED BY Rosenbrock &
C     A(1)=1.253031; A(2)=1.190943; A(3)=0.062798; A(4)=0.040063
C     BEST RESULTS ARE:
C     Quartic function with noise
C     -100 TO 100 M=2
C     HOUGEN
C
DOUBLE PRECISION (N,K),RATE(N),A(*)

HOUGEN FUNCTION (HOUGEN–WATSON MODEL FOR REACTION KINATICS)
C     NO. OF PARAMETERS (A) TO ESTIMATE = 5 = M
C
BEST RESULTS ARE:
A(1)=1.253031; A(2)=1.190943; A(3)=0.062798; A(4)=0.040063
A(5)=0.112453 ARE BEST ESTIMATES OBTAINED BY Rosenbrock &
Quasi-Newton METHOD WITH SUM OF SQUARES OF DEVIATION =0.298900994
AND R=0.99945.

THE NEXT BEST RESULTS GIVEN BY Hooke–Jeeves & Quasi-Newton
A(1)=2.475221;A(2)=0.599177; A(3)=0.124172; A(4)=0.083517
A(5)=0.217886; SUM OF SQUARES OF DEVIATION = 0.318593458
R=0.99941

MOST OF THE OTHER METHODS DO NOT PERFORM WELL

data x(1,1),x(1,2),x(1,3),rate(1) /470,300,10,8.55/
C     A(4)*X(I,3))
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   1 FORMAT(4F8.2)
C     WRITE(*,1)((X(I,J),J=1,K),RATE(I),I=1,N)

RPSWARM-NTEST.f 12/12

END
RETURN

if
if
do
ENDDO

F=

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
FX=(A(1)*X(I,2)-X(I,3)/A(5))/1.D00+A(2)*X(I,1)+A(3)*X(I,2)+
A(4)*X(I,3))
F=F+(RATE(I)-FX)**2
ENDDO
do
if

if(dp.gt.0.d00) f=fp
RETURN
END

12/12