Repulsive Particle Swarm Method on Some Difficult Test Problems of Global Optimization

Mishra, SK

North-Eastern Hill University, Shillong (India)

5 October 2006

Online at https://mpra.ub.uni-muenchen.de/1742/
MPRA Paper No. 1742, posted 11 Feb 2007 UTC
Repulsive Particle Swarm Method on
Some Difficult Test Problems of Global Optimization

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

I. Introduction: Optimization of non-convex (multi-modal) functions is the subject matter of research in global optimization. During the 1970’s or before only little work was done in this field, but in the 1980’s it attracted the attention of many researchers. Since then, a number of methods have been proposed to find the global optima of non-convex (multi-modal) problems of combinatorial as well as continuous types. Among these methods, genetic algorithms, simulated annealing, particle swarm, ants colony, tunneling, taboo search, etc. have been quite successful as well as popular.

It may be noted that no method can guarantee that it would surely find the global optimum of an arbitrary function in a finite number of attempts, however large. There is one more point to be noted. A particular method might be quite effective in solving some (class of) problems, but it may cut a sorry figure at the others. Next, each of these methods operates with a number of parameters that may be changed at choice to make it more effective. This choice is often problem oriented. A particular choice may be extremely effective in a few cases, but it might be ineffective (or counterproductive) in certain other cases. Additionally, there is a relation of trade-off among those parameters. These features make all these methods a subject of trial and error exercises.

There is another feature of these methods (and the literature regarding them) that deserves a mention here. Each method of global optimization has quite many variants. The proponents of those variants introduce some changes into the original algorithm, test their variants on a few (popularly used) benchmark functions (often of too small dimensions) and haste to suggest that the proposed variant(s) performs better than the original (or other variants of the) method. There is no harm in introducing a variant of any method that functions well or better than the others. In a field of research, which is attractive as well as alive, this is expected and welcome. However, the observed tendency to test those variants on a couple of popular (and easy!) benchmark problems and push the method into the market does not augur well. The extant literature on the subject matter shows how some benchmark problems are in frequent use - Ackley, Griewank, Himmelblau, Levy, Michalewicz, Rastrigin, Rosenbrock, Schwefel and a couple of others, but much less frequently; so much so that some authors churn out ‘literature’ profusely with the test problems like Himmelblau’s, Griewank’s and Rastrigin’s functions alone. This is not to say that these test problems are simple or trivial. Intended is only to point out that frequent use of these functions introduces a specific bias into the research efforts and keeps us away from many harder problems that characterize the challenging task of global optimization research.

II. The Objectives: The objectives of this paper are plain and simple: to test a particular variant of the (Repulsive) Particle Swarm method on some rather difficult problems. A
number of such problems are collected from the extant literature and a few of them are newly introduced. First, we introduce the Particle Swarm (PS) method of global optimization and its variant called the ‘Repulsive Particle Swarm’ (RPS) method. Then we endow the particles with some stronger local search abilities – much like tunneling – so that each particle can make a search in its neighborhood to optimize itself. Next, we introduce the test problems, the existing as well as the new ones. We also give plots of some of these functions to help appreciation of the optimization problem. Finally, we present the results of the optimization exercise. We append the (Fortran) computer program that we have developed and used in this exercise. En passant we may add that this program has been used to optimize a large (over 60) number of benchmark problems (see Mishra, 2006 (c), (d)).

III. The Particle Swarm Method of Global Optimization: As it is well known, the problems of the existence of global order, its integrity, stability, efficiency, etc. emerging at a collective level from the selfish nature of individual human beings have been long standing. The laws of development of institutions, formal or informal that characterize ‘the settle habits of thinking and acting at a collective level’, have been sought in this order. Thomas Hobbes, George Berkeley, David Hume, John Locke and Adam Smith visualized the global system arising out of individual selfish 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 rational and selfish. The postulate of perfect competition was felt to be a necessary one in demonstrating that. However, Alfred Marshall limited himself to partial equilibrium analysis and, thus, indirectly allowed for the role of the invisible hand (while general equilibrium economists - Leon Walras, Kenneth Arrow, Gerard Debreu and John Nash, etc - have held that the establishment of order can be explained by their approach). Yet, Thorstein Veblen (1898, 1899) never believed in the mechanistic view and pleaded for economics as an evolutionary science. Friedrich von Hayek (1944) believed in a similar philosophy and held 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, Ilya Prigogine (1984) demonstrated how ‘order’ emerges from ‘chaos’.

The PS 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). It allows for limited knowledge, memory, habit formation, social learning, etc, not entertained before. In the animal world we observe that a swarm of birds or insects or a school of fish searches for food, protection, etc. in a very typical manner (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 PS method mimics this behavior (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 of the position and velocity are done in each iteration as follows:

\[
\begin{align*}
v_{i+1} &= \omega v_i + c_1 r_1 (\hat{x}_i - x_i) + c_2 r_2 (\hat{x}_g - x_i) \\
x_{i+1} &= x_i + v_{i+1}
\end{align*}
\]

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.
- \( \hat{x}_g \) 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 Repulsive Particle Swarm method of optimization is a variant of the classical Particle Swarm method invented by Eberhart and Kennedy (1995) (see Wikipedia, http://en.wikipedia.org/wiki/RPSO). It is particularly effective in finding out the global optimum in very complex search spaces (although it may be slower on certain types of optimization problems).

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

\[
\begin{align*}
v_{i+1} &= \omega v_i + \alpha r_1 (\hat{x}_i - x_i) + \omega \beta r_2 (\hat{x}_h - x_i) + \omega \gamma r_3 z \\
x_{i+1} &= x_i + v_{i+1}
\end{align*}
\]

where,

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

Occasionally, when the process is caught in a local optimum, some chaotic perturbation in position as well as velocity of some particle(s) may be needed.
IV. Additional Local Search by the Particles: The traditional RPS gives little scope of local search to the particles. They are guided by their past experience and the communication received from the others in the swarm. We have modified the traditional RPS method by endowing stronger (wider) local search ability to each particle. Each particle flies in its local surrounding and searches for a better solution. The domain of its search is controlled by a new parameter \((n_{\text{step}})\). This local search has no preference to gradients in any direction and resembles closely to tunneling. This added exploration capability of the particles brings the RPS method closer to what we observe in real life. However, in some cases moderately wide search \((n_{\text{step}}=9, \text{say}; \text{see program})\) works better.

V. Randomized Neighbourhood Topology: Each particle learns from its ‘chosen’ inmates in the swarm. At the one extreme is to learn from the best performer in the entire swarm. This is how the particles in the original PS method learn. However, such learning is not natural. How can we expect the individuals to know as to the best performer and interact with all others in the swarm? We believe in limited interaction and limited knowledge that any individual can possess and acquire. So, our particles do not know the ‘best’ in the swarm. Nevertheless, they interact with some chosen inmates that belong to the swarm. Now, the issue is: how does the particle choose its inmates? One of the possibilities is that it chooses the inmates closer (at lesser distance) to it. But, since our particle explores the locality by itself, it is likely that it would not benefit much from the inmates closer to it. Other relevant topologies are: (the celebrated) ring topology, ring topology hybridized with random topology, star topology, von Neumann topology, etc.

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

VI. The Benchmark Functions: It has already been mentioned that the RPS variant described above has been tested on over 60 box-bound benchmark functions. In a great majority of cases it has succeeded at locating the minimum of these functions. In this paper we propose to test the method on some new and some well known difficult problems.
1. New function #1: We introduce a 2-d problem with $f_{\text{min}} (-8.4666, -9.9988) = -0.18466.$

$$f(x) = \left[ \cos \sqrt{x_1^2 + x_2^2} \right]^{0.5} + (x_1 + x_2)/100; \quad x_i \in [-10, 10]; \quad i = 1, 2$$

2. New function #2: This is a variant of function #1, where $\cos(.)$ is replaced by $\sin(.)$. This function has the optimum $f_{\text{min}} (-9.94112, -9.99952) = -0.199441.$

$$f(x) = \left[ \sin \sqrt{x_1^2 + x_2^2} \right]^{0.5} + (x_1 + x_2)/100; \quad x_i \in [-10, 10]; \quad i = 1, 2$$

3. New function #3: In the domain $x \in [-10, 10]$ with $f_{\text{min}} (-1.98682, 10) = 1.01983$ this function is

$$f(x) = \sum_{i=1}^{m} \left[ (\sin((\cos(x_i) + \cos(x_i))^2) - (\cos((\sin(x_i) + \sin(x_i))^2))^2 \right] + x_i^2 + 0.01(x_i + x_2)$$

4. New function #4: In the domain $x \in [-10, 10]$ with $f_{\text{min}} (2.8863, 1.82326) = 2.28395$, this function is defined as

$$f(x) = -\ln \left[ ((\cos(x_i) + \cos(x_i))^2 - (\sin((\sin(x_i) + \sin(x_i))^2))^2 \right] + x_i^2 + (x_i - 1)^2)/10$$

5. New function #5 (Quintic function): In the domain $x \in [-10, 10]$ with $f_{\text{min}} (0) = 0$ for $x_i = -1$ or 2; $i = 1, 2, ..., m$ this function (with multiple global minima) is defined as

$$f(x) = \sum_{i=1}^{m} \left[ x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4 \right]; \quad x_i \in [-10, 10]; \quad i = 1, 2, ..., m$$

6. New function #6 (Needle-eye function): This function is m-dimensional (m≥1) and defined with a small (say 0.0001) eye. If $|x_i| \leq \text{eye} \forall i$ then $f(x) = 1$. Else $f(x) = \sum_{i=1}^{m} (100 + |x_i|)t_i; \quad t_i = 1 \text{ if } |x_i| > \text{eye}, 0 \text{ otherwise}$. Minimization of this function becomes more difficult with smaller eye and larger m (dimension).

7. New function #7 (Zero-sum function): Defined in the domain $x \in [-10, 10]$ this function (in m≥2) has $f(x) = 0$ if $\sum_{i=1}^{m} x_i = 0$. Otherwise $f(x) = 1 + \left( \frac{10000}{\sum_{i=1}^{m} |x_i|} \right)^{0.5}$. This function has innumerably many minima but it is extremely difficult to obtain any of them. Larger is the value of m (dimension), it becomes more difficult to optimize the function.

8. Corana function: On $x \in [-1000, 1000]; \quad i = 1, 2, ..., 4$ this four-variable function is defined as follows and has $f_{\text{min}} (0, 0, 0, 0) = 0$.

$$f(x) = \sum_{i=0}^{1} 0.15(z_i - 0.05 \text{sgn}(z_i))^2, \quad d_i = 1, 1000, 10, 100$$

9. Modified RCOS function: In the domain $x_1 \in [-5, 10], x_2 \in [0, 15]$ this 2-variable function has $f_{\text{min}} (-3.196989, 12.52626) = -0.179891$. It is specified as

$$f(x) = -1/(f_1 + f_3 + f_5 + e)$$

$$f_1 = a(x_2 - bx_1^2 + c x_i - d)^2; \quad f_2 = e(1-g)\cos(x_i)\cos(x_j); \quad f_3 = \log(x_1^2 + x_2^2 + 1)$$

where, $g=1/(8\pi); \quad b=5.1/(4\pi^2); \quad c=5/\pi; \quad a=1; \quad d=6; \quad e=10$.

10. Freudenstein Roth function: On $x_i \in [-10, 10]; \quad i = 1, 2$ this 2-variable function is defined as follows and has $f_{\text{min}} (5, 4) = 0$. 


\[ f(x) = -13 + x_1 + ((5 - x_2)x_3 - 2)x_4^2 + [-29 + x_1 + ((x_2 + 1)x_3 - 14)x_4] \]

11. **ANNs XOR function**: This function is in nine variables. It is defined as follows.
\[ f(x) = f_1 + f_2 + f_3 + f_4 \]
where,
\[ f_1 = \left[ 1 + \exp\{-x_1 / (1 + e^{-x_2 - x_3}) - x_4 / (1 + e^{-x_2 - x_3}) - x_6 \} \right] \]
\[ f_2 = \left[ 1 + \exp\{-x_1 / (1 + e^{x_2 - x_3}) - x_4 / (1 + e^{x_2 - x_3}) - x_6 \} \right] \]
\[ f_3 = \left[ 1 - 1 + \exp\{-x_1 / (1 + e^{x_2 - x_3}) - x_4 / (1 + e^{x_2 - x_3}) - x_6 \} \right] \]
\[ f_4 = \left[ 1 - 1 + \exp\{-x_1 / (1 + e^{x_2 - x_3}) - x_4 / (1 + e^{x_2 - x_3}) - x_6 \} \right] \]

It is very difficult to minimize this function. We obtain (by RPS) \( f_{\text{min}} = 0.95979 \) for:
\[ x = (0.99999, 0.99993, -0.89414, 0.99994, 0.55932, 0.99994, 0.99994, -0.99963, -0.08272) \]

12. **Perm function #1**: In the domain \( x \in [-4, 4] \), the function has \( f_{\text{min}} = 0 \) for \( x = (1, 2, 3, 4) \). It is specified as
\[ f(x) = \sum_{k=1}^{2} \left[ \sum_{i \in [1, 4]} (i^4 + \beta) \{(x_i / i)^k - 1\} \right] \]

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

13. **Perm function #2**: In the domain \( x \in [-1, 1] \), and for a given \( \beta \) (=10), this \( m \)-variable function has \( f_{\text{min}} = 0 \) for \( x_i = (i)^k \cdot i = 1, 2, ..., m \). It is specified as
\[ \sum_{k=1}^{2} \left[ \sum_{i=1}^{m} (i + \beta) \{(x_i)^k - (i)^k\} \right] \]

Smaller values of beta raise difficulty in optimization.

14. **Power-sum function**: Defined on four variables in the domain \( x \in [0, 4] \), this function has \( f_{\text{min}} = 0 \) for any permutation of \( x = (1, 2, 2, 3) \). The function is defined as
\[ f(x) = \sum_{k=1}^{2} \left[ b_k - \sum x_i^k \right] ; \quad b_k = (8, 18, 44, 114) \quad \text{for } k = (1, 2, 3, 4) \] respectively.

15. **Goldstein Price function**: On \( x_i \in [-10, 10] \); \( i = 1, 2 \) this 2-variable function is defined as follows and has \( f_{\text{min}} (0, -1) = 3 \).
\[ f(x) = (f_1)(f_2) \]
where,
\[ f_1 = 1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_2^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \]
\[ f_2 = 1 + (x_1 - x_2)^2(18 - 32x_1 + 12x_2^2 + 48x_2 - 36x_1x_2 + 27x_2^2) \]

16. **Bukin’s functions**: Bukin’s functions are almost fractal (with fine seesaw edges) in the surroundings of their minimal points. Due to this property, they are extremely difficult to optimize by any method of global (or local) optimization and find correct values of decision variables (i.e. \( x_i \) for \( i = 1, 2 \)). In the search domain \( x_i \in [-15, -5], x_2 \in [-3, 3] \) the 6th Bukin’s function is defined as follows.
\[ f_6(x) = 100 \sqrt{x_2^2 - 0.01x_1^2} + 0.01|x_1 + 10| ; \quad f_{\text{min}} (-10, 1) = 0 \]

This account does not exhaust the list of difficult (benchmark) functions. Bukin’s (others than specified above), Hougen’s, Giunta’s, Weierstrass’s, Factorial, Decanomial, SeqP, AMGM, etc. (see the computer program) are prominent among such difficult ones.
Elsewhere (Mishra, 2006 (a) and (b)) also we faced difficult global optimization problems.

VII. Performance of the RPS Method: In table-1 we present the results of our optimization efforts of the selected test functions described in the preceding section. For sake of comparison, we have optimized those functions with the Genetic Algorithm (GA) of David Carroll or Simulated Annealing (SA) of William Goffe and the (proposed variant of) RPS. We have changed three parameters in Carroll’s codes: maxgen in ga.inp file is set to 500; nparam is set to the required dimension in the same file for different functions; parameter (indmax=1000,nchrmx=60,nparmx=10) is set in the params.f file. Goffe’s SA program is used as it is. On the other hand, for RPS we have used our own program (appendix) with varying parameters. Note that instead of bringing them into the limits, we have placed heavy penalties whenever the arguments \( x_i; i = 1,2,\ldots,m \) are out of bounds (see the computer program).

<table>
<thead>
<tr>
<th>Sl No.</th>
<th>Function</th>
<th>m</th>
<th>The Minimal Value Obtained</th>
<th>GA or SA</th>
<th>RPS/SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>New function (N#1)</td>
<td>2</td>
<td>0.41490 GA</td>
<td>-1.18887</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>New function (N#2)</td>
<td>2</td>
<td>0.00000 GA</td>
<td>-0.19938</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>New function (N#3)</td>
<td>2</td>
<td>0.58581 GA</td>
<td>-1.01983</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>New function (N#4)</td>
<td>2</td>
<td>-1.14419 GA</td>
<td>-2.28395</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>New function (N#5) (Quintic function)</td>
<td>2</td>
<td>0.00000 SA</td>
<td>0.00094</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>New function (N#6) (Needle-eye function)</td>
<td>8</td>
<td>1 GA/SA</td>
<td>1.04573</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Zero-sum function (N#7)</td>
<td>8</td>
<td>0.00000 GA</td>
<td>1.04573</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Corana function</td>
<td>4</td>
<td>0.00000 GA/SA</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Modified RCOS function</td>
<td>2</td>
<td>-0.03786 GA</td>
<td>-0.17989</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Freundenstein Roth function</td>
<td>2</td>
<td>0.00000 SA</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>ANNs XOR function</td>
<td>9</td>
<td>0.959759 SA</td>
<td>0.95979</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Perm #1 function</td>
<td>4</td>
<td>16.000000 SA</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Perm #2 function</td>
<td>2</td>
<td>0.00011 (!!) GA/SA</td>
<td>0.00000 (!!)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Power-sum function</td>
<td>4</td>
<td>***** GA/SA</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Goldenstein Price function</td>
<td>2</td>
<td>***** GA/SA</td>
<td>3.00000</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Bukin’s 6th function</td>
<td>2</td>
<td>0.000000 (??) GA/SA</td>
<td>0.00000 (??)</td>
<td></td>
</tr>
</tbody>
</table>

Note: Shaded = Better results; Shaded = Perform equally well; (??) Decision variables far from the correct ones; ***** = Unstable; !! = Results degenerate fast as the dimension increases

The results indicate that none of the (three) method can assuredly find the optimum of an arbitrary function. In case of the Needle-eye and the Corana functions all three methods perform equally well while in case of Bukin’s 6th function all of them yield the values of decision variables far away from the right ones. In case of zero-sum function, GA performs better than the RPS. In case of the Perm #2 function, all of the methods fail when the dimension grows larger. In several cases, GA falters or fails while RPS succeeds. In case of N#1 through N#5 and the ANNs XOR functions the RPS performs better than the Genetic algorithm, but comparably or worse than SA. In case of the quintic function SA outperforms RPS.
VIII. Conclusion: From what we have seen above, one may jump at the conclusion that the RPS performs better than the GA at least. But we would like not to do so. We would only conclude that none could assure a supremacy over the other(s). Each one faltered in some case; each one succeeded in some others.

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

<table>
<thead>
<tr>
<th>New Function #1</th>
<th>New Function #2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Function Chart" /></td>
<td><img src="image" alt="Function Chart" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>New Function #3</th>
<th>New Function #2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Function Chart" /></td>
<td><img src="image" alt="Function Chart" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>New Function #5 (Quintic function)</th>
<th>New function (Decanomial function)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Function Chart" /></td>
<td><img src="image" alt="Function Chart" /></td>
</tr>
</tbody>
</table>

Author’s Contact: mishrasknehu@hotmail.com
Bibliography

C PROGRAM TO FIND GLOBAL MINIMUM BY REPULSIVE PARTICLE SWARM METHOD
C WRITTEN BY SK MISHRA, DEPT. OF ECONOMICS, NEHU, SHILLONG (INDIA)
C
PARAMETER (N=100,NN=40,MX=100,NSTEP=15,ITRN=5000,NSIGMA=1,ITOP=3)
PARAMETER (NPNR=100) ! DISPLAYS RESULTS AT EVERY 100 TH ITERATION
PARAMETER (N=50,NN=25,MX=100,NSTEP=9,ITRN=10000,NSIGMA=1,ITOP=3)
PARAMETER (N=100,NN=15,MX=100,NSTEP=9,ITRN=10000,NSIGMA=1,ITOP=3)
C IN CERTAIN CASES THE ONE OR THE OTHER SPECIFICATION WORKS BETTER
C DIFFERENT SPECIFICATIONS OF PARAMETERS MAY SUIT DIFFERENT TYPES
C OF FUNCTIONS OR DIMENSIONS - ONE HAS TO DO SOME TRIAL AND ERROR
C
N = POPULATION SIZE. IN MOST OF THE CASES N=30 IS OK. ITS VALUE
C MAY BE INCREASED TO 50 OR 100 TOO. THE PARAMETER NN IS THE SIZE OF
C RANDOMLY CHOSEN NEIGHBOURS. 15 TO 25 (BUT SUFFICIENTLY LESS THAN
C N) IS A GOOD CHOICE. MX IS THE MAXIMAL SIZE OF DECISION VARIABLES.
C IN F(X1, X2, ..., XM) M SHOULD BE LESS THAN OR EQUAL TO MX. ITRN IS
C THE NO. OF ITERATIONS. IT MAY DEPEND ON THE PROBLEM. 200(AT LEAST)
C TO 500 ITERATIONS MAY BE GOOD ENOUGH. BUT FOR FUNCTIONS LIKE
C ROSENBROCK'S GRIEWANK OF LARGE SIZE (SAY M=50) IT IS NEEDED THAT
C ITRN IS LARGE, SAY 5000 OR EVEN 10000.
C
C SIGMA INTRODUCES PERTURBATION & HELPS THE SEARCH JUMP OUT OF LOCAL
C OPTIMA. FOR EXAMPLE : RASTRIGIN FUNCTION OF DMENSION 30 OR LARGER
C SIGMA INTRODUCES PERTURBATION & HELPS THE SEARCH JUMP OUT OF LOCAL
C ITRN IS LARGE, SAY 5000 OR EVEN 10000.
C 15, WHICH IS MUCH ON THE HIGHER SIDE.
C
C ITRN < 1 (RING); ITRN=2 (RING AND RANDOM); ITRN=3 (RANDOM)
C NSIGMA=0 (NO CHAOTIC PERTURBATION);NSIGMA=1 (CHAOTIC PERTURBATION)
C NOTE THAT NSIGMA=1 NEED NOT ALWAYS WORK BETTER (OR WORSE)
C SUBROUTINE FUNC1 : DEFINES OR CALLS THE FUNCTION TO BE OPTIMIZED.

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
COMMON /RNDM/IU,IV
COMMON /K/F/KF,NCALL
INTEGER IU,IV
CHARACTER *70 TIT
DIMENSION X(N,MX),V(N,MX),A(MX),V1(MX),V2(MX),V3(MX),V4(MX),BST(MX)
DIMENSION XX(N,MX),F(N),V1(MX),V2(MX),V3(MX),V4(MX),BST(MX)
A1 A2 AND A3 ARE CONSTANTS AND W IS THE INERTIA WEIGHT.
C OCCASSIONALLY, TINKERING WITH THESE VALUES, ESPECIALLY A3, MAY BE
C NEEDED.
DATA A1,A2,A3,W,SIGMA/.5000, .5000, .0005D00, .5000, 1.D-03/
WRITE('(*,*)') '-----------------------------------------------------'
DATA TIT(1) /'KF=1 NEW FUNCTION(N#1) 2-VARIABLES M=2'/
DATA TIT(2) /'KF=2 NEW FUNCTION(N#2) 2-VARIABLES M=2'/
DATA TIT(3) /'KF=3 NEW FUNCTION(N#3) 2-VARIABLES M=2'/
DATA TIT(4) /'KF=4 NEW FUNCTION(N#4) 2-VARIABLES M=2'/
DATA TIT(5) /'KF=5 NEW QUINTIC FUNCTION M-VARIABLES M=?'/
DATA TIT(6) /'KF=6 NEW NEEDLE-EYE FUNCTION (N#6) M-VARIABLES M=?'/
DATA TIT(7) /'KF=7 NEW ZERO-SUM FUNCTION (N#7) M-VARIABLES M=?'/
DATA TIT(8) /'KF=8 CORANA FUNCTION 4-VARIABLES M=4'/
DATA TIT(9) /'KF=9 MODIFIED RCOS FUNCTION 2-VARIABLES M=2'/
DATA TIT(10) /'KF=10 FREUDENSTEIN ROTH FUNCTION 2-VARIABLES M=2'/
DATA TIT(11) /'KF=11 ANNS XOR FUNCTION 9-VARIABLES M=9'/
DATA TIT(12) /'KF=12 PERM FUNCTION #1 (SET BETA) 4-VARIABLES M=4'/
DATA TIT(13) /'KF=13 PERM FUNCTION #2 (SET BETA) M-VARIABLES M=?'/
DATA TIT(14) /'KF=14 POWER-SUM FUNCTION 4-VARIABLES M=4'/
DATA TIT(15) /'KF=15 GOLDSTEIN PRICE FUNCTION 2-VARIABLES M=2'/
DATA TIT(16) /'KF=16 BUKIN 6TH FUNCTION 2-VARIABLES M=2'/
DATA TIT(17) /'KF=17 NEW FUNCTION (N#8) 2-VARIABLES M=2'/
DATA TIT(18) /'KF=18 DEFL CORRUG SPRING FUNCTION M-VARIABLES M=?'/
DATA TIT(19) /'KF=19 NEW FACTORIAL FUNCTION M-VARIABLES M=?'/
DATA TIT(20) /'KF=20 NEW DECANOMIAL FUNCTION 2-VARIABLES M=2'/
DATA TIT(21) /'KF=21 JUDGE FUNCTION 2-VARIABLES M=2'/
DATA TIT(22) /'KF=22 NEW DODECAK FUNCTION 3-VARIABLES M=3'/
DATA TIT(23) /'KF=23 NEW SUM-EQ-PROD FUNCTION 2-VARIABLES M=2'/
DATA TIT(24) /'KF=24 NEW AM-EQ-GM FUNCTION M-VARIABLES M=?'/
DATA TIT(25) /'KF=25 YAO-LIU FUNCTION #2 M-VARIABLES M=?'/
DATA TIT(26) /'KF=26 YAO-LIU FUNCTION #3 M-VARIABLES M=?'/
C RANDOM TOPOLOGY ******************************************
C
C FIND THE BEST AMONG THEM
C
NOW LET EVERY INDIVIDUAL RANDOMLY CONSULT NN(<N) COLLEAGUES AND

XX(I,J) IS THE M-TUPLE VALUE OF X ASSOCIATED WITH LOCAL BEST F(I)
F(I) CONTAINS THE LOCAL BEST VALUE OF FUNCTION FOR ITH INDIVIDUAL

LET EACH INDIVIDUAL SEARCH FOR THE BEST IN ITS NEIGHBOURHOOD
V(I,J)=RAND
INITIALISE VELOCITIES V(I) FOR EACH INDIVIDUAL IN THE POPULATION
CASES MAY BE NEEDED
WE GENERATE RANDOM(-5,5). HERE MULTIPLIER IS 10. TINKERING IN SOME
GENERATE N-SIZE POPULATION OF M-TUPLE PARAMETERS X(I,J) RANDOMLY

-------------------------------------------------------------------
RPS-H2.f 2/21
IF
DO
DO
ENDDO
ENDDO
CALL
DO
DO
DO
DATA
READ
WRITE
WRITE
WRITE
ENDDO
WRITE
DO
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DATA
DO
DO
IF
ENDDO
VI
A
DO
I
J
RANDOM
1
1
FMIN
*
I
J
1
TIT
TIT
TIT
TIT
TIT
TIT
TIT
TIT
TIT
TIT
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0

ENDIF
ENDDO
BST
DO
F
I
J
F
I
J
I
J
I
J
I
J
I
J
I
J
I
J
I
J
I
J
I
J
I
J
DO
I=1,N
DO J=1,M
CALL RANDOM(RAND)
X(I,J)=(RAND-0.5000)*10
C WE GENERATE RANDOM(-5,5). HERE MULTIPLIER IS 10. TINKERING IN SOME
C CASES MAY BE NEEDED
C
ENDDO
F(I)=1.0D30
ENDDO
C INITIALISE VELOCITIES V(I) FOR EACH INDIVIDUAL IN THE POPULATION
DO I=1,N
DO J=1,M
CALL RANDOM(RAND)
V(I,J)=(RAND-0.5D+00)
C
ENDDO
ENDDO
C
DO 100 ITER=1,ITRN
C LET EACH INDIVIDUAL SEARCH FOR THE BEST IN ITS NEIGHBOURHOOD
DO I=1,N
DO J=1,M
A(J)=X(I,J)
VI(J)=V(I,J)
ENDDO
CALL LSRCH(A,M,VI,NSTEP,FI)
IF(FI.LT.F(I)) THEN
F(I)=FI
DO IN=1,M
BST(IN)=A(IN)
ENDDO
C F(I) CONTAINS THE LOCAL BEST VALUE OF FUNCTION FOR ITH INDIVIDUAL
C XX(I,J) IS THE M-TUPLE VALUE OF X ASSOCIATED WITH LOCAL BEST F(I)
DO J=1,M
XX(I,J)=A(J)
ENDDO
ENDDO
C
DO I=1,N
IF(ITOP.GE.3) THEN
C RANDOM TOPOLOGY ****************************************************
C CHOOSE NN COLLEAGUES RANDOMLY AND FIND THE BEST AMONG THEM
     BEST=1.0D30
     DO II=1,NN
        CALL RANDOM(RAND)
        NF=INT(RAND*NN)+1
        IF(BEST.GT.F(NF)) THEN
           BEST=F(NF)
           NFBEST=NF
        ENDIF
     ENDDO
     ENDIF
     IF(ITOP.EQ.2) THEN
     C RING + RANDOM TOPOLOGY *******************************
     C REQUIRES THAT THE SUBROUTINE NEIGHBOR IS TURNED ALIVE
     BEST=1.0D30
     CALL NEIGHBOR(I,N,I1,I3)
     DO II=1,NN
        IF(II.EQ.1) NF=I1
        IF(II.EQ.2) NF=I
        IF(II.EQ.3) NF=I3
        IF(BEST.GT.F(NF)) THEN
           CALL RANDOM(RAND)
           NF=INT(RAND*NN)+1
        ENDIF
        IF(BEST.GT.F(NF)) THEN
           BEST=F(NF)
           NFBEST=NF
        ENDIF
     ENDDO
     ENDIF
     IF(ITOP.LE.1) THEN
     C RING TOPOLOGY ************************************
     C REQUIRES THAT THE SUBROUTINE NEIGHBOR IS TURNED ALIVE
     BEST=1.0D30
     CALL NEIGHBOR(I,N,I1,I3)
     DO II=1,3
        IF(II.NE.1) THEN
        IF(II.EQ.1) NF=I1
        IF(II.EQ.3) NF=I3
        IF(BEST.GT.F(NF)) THEN
           CALL RANDOM(RAND)
           NF=INT(RAND*NN)+1
        ENDIF
        IF(BEST.GT.F(NF)) THEN
           BEST=F(NF)
           NFBEST=NF
        ENDIF
     ENDDO
     ENDIF
     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
     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)=-A*RAND^W*RND1
V3(J)=A*RAND^W
THEN ON PAST VELOCITY WITH INERTIA WEIGHT W
V4(J)=W*V(I,J)
FINALLY A SUM OF THEM
V(I,J)= V1(J)+V2(J)+V3(J)+V4(J)
ENDDO
CHANGE X
FINALLY A SUM OF THEM

C     ----------------------------------------------------------------
C      WRITE(*,*)'NO. OF FUNCTION CALLS = ',NFCALL
C     PARAMETER (N=100,NN=15,MX=100,NSTEP=9,ITRN=100000,NSIGMA=1,ITOP=2)
C     IN SOME CASES THIS PERTURBATION HAS WORKED VERY EFFECTIVELY WITH
C     SIGMA CONDITIONED RANDS INTRODUCES CHAOTIC ELEMENT IN TO LOCATION
C    ----------------------------------------------------------------------
C     CHANGE X
C     FINALLY A SUM OF THEM
C            V3(J)=A3*RAND*W
C     -------------------------------------------------------------------
C     CHANGE X
C     FINALLY A SUM OF THEM

RPS-H2.f 4/21
END

THIS SUBROUTINE IS NEEDED IF THE NEIGHBOURHOOD HAS RING TOPOLOGY EITHER PURE OR HYBRIDIZED

SUBROUTINE NEIGHBOR(I, N, J, K)

IF(I-1.GE.1 .AND. I.LT.N) THEN
  J=I-1
  K=I+1
ELSE
  IF(I-1.LT.1) THEN
    J=N-I+1
    K=I+1
 ENDIF
  IF(I.EQ.N) THEN
    J=I-1
    K=1
 ENDIF
ENDIF
RETURN
END

SUBROUTINE RANDOM(RAND1)

DOUBLE PRECISION RAND1
COMMON /RNDM/ IU, IV
INTEGER IU, IV

RAND=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)

IMPLICIT DOUBLE PRECISION (A-H, O-Z)
COMMON /RNDM/IU, IV
COMMON /KFF/KF, NFCALL
INTEGER IU, IV
DIMENSION X(*)

PI=4.D+00*DATAN(1.D+00)
NFCALL=NFCALL+1

IF(KF.EQ.1) THEN
  F=0.00
  FP=0.00
  DO I=1, M
    IF(DABS(X(I)).GT.10.D00) FP=FP+DEXP(DABS(X(I)))
  ENDDO
  IF(FP.NE.0.000) THEN
    F=FP
  ENDIF
  RETURN
ENDIF

IF(KF.EQ.2) THEN
  FUNCTION #2 MIN = -0.199409 APPROX AT (-9.9912, -9.99952) APPROX
  F=0.00
ENDIF
C      IF(RAND.LT.0.5D00) THEN   ! TURN IT ALIVE
C      CALL RANDOM(RAND)   ! TURN IT ALIVE
C     COLUMN OF THE LINE ---------------------------------------------
C     X=(2,2) OR X=(2,-1) OR X= (-1,2) OR TURN ALIVE THE FOLLOWING BY
C     CHECK FOR OPTIMUM SOLUTION : SET X(I) = -1 OR 2 SUCH AS X=(-1,-1)
C     ----------------------------------------------------------------
C     ----------------------------------------------------------------
C     ----------------------------------------------------------------
C     IF(KF.EQ.3) THEN
C     FUNCTION #3 MIN = -1.01983 APPROX AT (-1.98682, -10.00000) APPROX
C     F=0.D00
C     FP=0.D00
C     DO I=1,M
C     IF(DABS(X(I)).GT.10.D00) FP=FP+DEXP(DABS(X(I)))
C     ENDDO
C     IF(FP.NE.0.D00) THEN
C     RETURN
C     ELSE
C     F=DABS(DSQRT(DABS(X(1)**2+X(2)**2))**0.5 +0.01*X(1)+.01*X(2)
C     RETURN
C     ENDIF
C     ENDIF
C     C
C     IF(KF.EQ.4) THEN
C     FUNCTION #4 MIN = -2.28395 APPROX AT (2.88631, 1.82326) APPROX
C     F=0.D00
C     FP=0.D00
C     DO I=1,M
C     IF(DABS(X(I)).GT.10.D00) FP=FP+DEXP(DABS(X(I)))
C     ENDDO
C     IF(FP.NE.0.D00) THEN
C     RETURN
C     ELSE
C     F1=DSIN(( DCOS(X(1))+DCOS(X(2)) )**2)**2
C     F2=DCOS(( DSIN(X(1))+DSIN(X(2)) )**2)**2
C     F=(F1+F2+X(1))**2 ; IS MULTIMODAL
C     F=F+ 0.01*X(1)+0.1*X(2) ; MAKES UNIMODAL
C     RETURN
C     ENDIF
C     ENDIF
C     C
C     IF(KF.EQ.5) THEN
C     QUINTIC FUNCTION:GLOBAL MINIMA, EXTREMELY DIFFICULT TO OPTIMIZE
C     C
C     MIN VALUE = 0 AT PERMUTATION OF (2, 2, ..., 2, -1, -1, ..., -1)
C     C
C     CHECK FOR OPTIMUM SOLUTION : SET X(I) = -1 OR 2 SUCH AS X=(-1,-1)
C     X=(2,2) OR X=(2,-1) OR X= (-1,2) OR TURN ALIVE THE FOLLOWING BY
C     C
C     REMOVING C FROM THE FIRST COLUMN -------------------------------
C     C
C     X(1)=?  ; SET IT TO -1 OR 2
C     X(2)=?  ; SET IT TO -1 OR 2
C     C
C     OR FOR M => 1 TURN THE FOLLOWING ALIVE BY REMOVING C IN THE FIRST
C     C
C     COLUMN OF THE LINE -------------------------------
RPS-H2.f

403: C X(I)=2  ! TURN IT ALIVE
404: C ENDF      ! TURN IT ALIVE
405: C ENDDO     ! TURN IT ALIVE
406: C TEST OVER ------------------------------------------
407: 408: FP=0.D00
409:  DO I=1,M
410:  IF(DABS(X(I)).GT.10.D00) FP=FP+DEXP(DABS(X(I)))
412:  ENDDO
413:  IF(FP.NE.0.D00) THEN
414:    F=FP
415:    RETURN
416: ELSE
417:    CALL QUINTIC(M,F,X)
418:  ENDIF
419:  ENDIF
420:  IF(KF.EQ.6) THEN
421:    NEEDLE-EYE FUNCTION M=>1;
422:    MIN = 1 IF ALL ABS(X) ARE SMALLER THAN THE EYE
423:    SMALLER THE VALUE OF ZZ, MORE DIFFICULT TO ENTER THE EYE
424:    LARGER THE VALUE OF M, MORE DIFFICULT TO FIND THE OPTIMUM
425:    F=0.D00
426:    EYE=0.0001D00
427:    FP=0.D00
428:  DO I=1,M
429:    IF(DABS(X(I)).GT.EYE) THEN
430:      FP=1.D00
431:      F=F+100.D00+DABS(X(I))
432:    ELSE
433:      F=F+1.D00
434:    ENDDO
435:    IF(FP.NE.0.D00) THEN
436:      F=FP
437:      RETURN
438:    ENDIF
439:  ENDIF
440:  IF(KF.EQ.7) THEN
441:    ZERO SUM FUNCTION
442:    MIN = 0 AT SUM(X(I))=0
443:    F=0.D00
444:    DO I=1,M
445:      IF(DABS(X(I)).GT.10.D00) FP=FP+DEXP(DABS(X(I)))
446:    ENDDO
447:    IF(FP.NE.0.D00) THEN
448:      F=FP
449:      RETURN
450:    ELSE
451:      SUM=0.D00
452:      DO I=1,M
453:        SUM=SUM+X(I)
454:      ENDDO
455:      IF(SUM.NE.0.D00) F=1.D00+(10000*DABS(SUM))*0.5
456:      RETURN
457:    ENDIF
458:  ENDIF
459:  IF(KF.EQ.8) THEN
460:    CORANA FUNCTION
461:  ENDIF
462:  IF(KF.EQ.8) THEN
463:    CORANA FUNCTION
464:  ENDIF
465:  IF(KF.EQ.8) THEN
466:    F=0.D00
467:    FP=0.D00
468:    DO I=1,4
469:      -1000 TO 1000 M=4
IF(DBS(X(I)),GT.1000.D00) FP=FP+X(I)**2
ENDO
IF(FP.GT.0.D00) THEN
  F=FP
ELSE
  DO J=1,4
    IF(J.EQ.1) DJ=1.D00
    IF(J.EQ.2) DJ=1000.D00
    IF(J.EQ.3) DJ=10.D00
    IF(J.EQ.4) DJ=100.D00
    ISGNJ=-1
    IF(X(J).LT.0.D00) ISGNJ=-1
    ZJ=(DBS(X(J))/0.2D00)+0.49999)*ISGNJ*0.2D00
    ISGNJ=1
    IF(ZJ.LT.0.D00) ISGNJ=-1
    IF(DBS(X(J)-ZJ).LT.0.05D00) THEN
      F=FP+0.15D00*(ZJ-0.05D00)*ISGNJ)**2 * DJ
    ELSE
      F=FP+DJ*X(J)**2
    ENDIF
  ENDDO
ENDIF
RETURN
ENDIF
---------------------------------------------------------------------
C MODIFIED RCOS FUNCTION MIN=-0.179891 AT (-3.196989, 12.52626)APPRX
C
F=0.D00
FP=0.D00
IF(X(1).LT.-5.D00 .OR. X(1).GT.10.D00) FP=FP+DEXP(DBS(X(I)))
IF(X(2).LT.-0.D00 .OR. X(2).GT.15.D00) FP=FP+DEXP(DBS(X(2)))
IF(FP.NE.0.D00) THEN
  F=FP
RETURN
ELSE
  CA=1.D00
  CB=5.1/(4*PI)**2
  CC=5.D00/PI
  CD=6.D00
  CE=10.D00
  CF=1.0/(8*PI)
  F1=CA*X(2)-CB*X(1)**2+CC*X(1)-CD)**2
  F2=CE*(1.000CF)*DCOS(X(1))*DCOS(X(2))
  F3=DCOS(X(1))**2+X(2)**2+1.D00
  F=1.0/(F1+F2+F3+CE)
RETURN
ENDIF
---------------------------------------------------------------------
C FREUDENSTEIN ROTH FUNCTION
C MIN = 0 AT (5, 4)
F=0.D00
FP=0.D00
DO I=1,M
  IF(DBS(X(I)).GT.10.D00) FP=FP+DEXP(DBS(X(I)))
ENDDO
IF(FP.NE.0.D00) THEN
  F=FP
RETURN
ELSE
  F1=(-13.D00+X(1)+((5.D00-X(2))*X(2)-2)*X(2))**2
  F2=(-29.D00+X(1)+((X(2)+1.D00)*X(2)-14.D00)*X(2))**2
  F=F1+F2
RETURN
ENDIF
C     BETA => 0. CHANGE IF NEEDED. SMALLER BETA RAISES DIFFICULTY
C     PERM FUNCTION #2 MIN = 0 AT (1/1, 1/2, 1/3, 1/4,..., 1/M)
C     -----------------------------------------------------------------
C     FOR BETA=0, EVERY PERMUTED SOLUTION IS A GLOBAL MINIMUM
C     -----------------------------------------------------------------
C     0.55932, 0.99994, 0.99994, -0.99963, -0.08272).
C     MIN  = 0.959789 AT X = (0.99999, 0.99993, -0.89414,  0.99994,
C     MINIMIZERS THROUGH PARTICLE SWARM OPTIMIZATION
C     AND VRATHIS, MN "STRETCHING TECHNIQUE FOR OBTAINING GLOBAL
C     ANNS XOR FUNCTION (PARSOPOLUS, KE, PLAGIANAKOS, VP, MAGOULAS, GD
C     -----------------------------------------------------------------

DO
  FP
  F
  ENDDO

DO
  FP
  F
  ENDDO

DO
  IF(DABS(X(I)).GT.1.D00) FP=FP+DEXP(10.D00+DABS(X(I)))
ENDDO

IF(FP.NE.0.D00) THEN
  F=FP
RETURN
ELSE
  IF(KF.EQ.11) THEN
    IF(KF.EQ.11) THEN
      PERM FUNCTION #2 MIN = 0 AT (1, 2, 3, 4)
      BETA => 0. CHANGE IF NEEDED. SMALLER BETA RAISES DIFFICULTY
      FOR BETA=0, EVERY PERMUTED SOLUTION IS A GLOBAL MINIMUM
      BETA=50.D00
      F=0.D00
      FP=0.D00
      DO I=1,M
        IF(DABS(X(I)).GT.M) FP=FP+X(I)**2
      ENDDO
    ELSE
      IF(KF.EQ.12) THEN
        PERM FUNCTION #1 MIN = 0 AT (1, 2, 3, 4)
        BETA => 0. CHANGE IF NEEDED. SMALLER BETA RAISES DIFFICULTY
        FOR BETA=0, EVERY PERMUTED SOLUTION IS A GLOBAL MINIMUM
        BETA=10.D00
        F=0.D00
        FP=0.D00
        DO I=1,M
          SUM=SUM+I**K+BETA)*((X(I)/I)**K-1.D00)
        ENDDO
        F=F+SUM**2
      ENDIF
    ELSE
      IF(KF.EQ.13) THEN
        PERM FUNCTION #1 MIN = 0 AT (1, 1/2, 1/3, 1/4,..., 1/M)
        BETA => 0. CHANGE IF NEEDED. SMALLER BETA RAISES DIFFICULTY
        FOR BETA=0, EVERY PERMUTED SOLUTION IS A GLOBAL MINIMUM
        BETA=10.D00
        F=0.D00
        FP=0.D00
        DO I=1,M
          IF(X(I).EQ.0.D00) THEN
            TO CHECK MIN=0, TURN THE FOLLOWING STATEMENT [(X(I)=1.D00/I) ALIVE
C     MIN VALUE = 3 AT (0, -1)
C     GOLDSTEIN PRICE FUNCTION
C     -----------------------------------------------------------------
C     ANY PERMUTATION OF (1,2,2,3) WILL GIVE MIN = ZERO
C        IF(I.EQ.4) X(I)=2.D00 ! TURN IT ALIVE
C        IF(I.EQ.3) X(I)=2.D00 ! TURN IT ALIVE
C        IF(I.EQ.2) X(I)=1.D00 ! TURN IT ALIVE
C        IF(I.EQ.1) X(I)=3.D00 ! TURN IT ALIVE
C     PERMUTATION OF X=(1,2,2,3), FOR EXAMPLE
C     TURN THE FOLLOWING STATEMENTS ALIVE TO CHECK SOLUTION FOR ANY
C     0 <= X <=4
C     POWER SUM FUNCTION; MIN = 0 AT PERM(1,2,2,3) FOR B=(8,18,44,114)
C     -----------------------------------------------------------------
C      X(I)=1.D00/I   ! TURN IT ALIVE
C
10/21
F21=(2*X(1)-3*X(2))**2
F22=(18.D00-32*X(1)+12*X(1)**2+48*X(2)-36*X(1)*X(2)+27*X(2)**2)
F2=30.D00+F21+F22
F=(F1+F2)
RETURN
ENDIF
ENDIF

IF(KF.EQ.16) THEN
BUKIN'S 5TH FUNCTION MIN = 0 FOR (-10, 1)
FP=0.D00
ELSE
-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
ENDIF
ENDIF
ENDIF

IF(KF.EQ.17) THEN
NEW N#8 FUNCTION (MULTIPLE GLOBAL MINIMA)
MIN VALUE = -1 AT (AROUND .7 AROUND, 0.785 APPROX)
F=0.D00
DO I=1,M
IF(X(I).LT.0.5D00 .OR. X(I).GT.1.D00) FP=FP+DEXP(2.D00+DABS(X(I)))
ENDIF
ENDIF
ENDIF

IF(KF.EQ.18) THEN
DELECTED CORRUGATED SPRING FUNCTION
MIN VALUE = -1 AT (5, 5, ..., 5) FOR ANY K AND ALPHA=5; M VARIABLE
CALL DCS(M,F,X)
RETURN
ENDIF

IF(KF.EQ.19) THEN
FACTORIAL FUNCTION, MIN =0 AT X=(1,2,3,...M)
CALL FACTORI(M,F,X)
RETURN
ENDIF

IF(KF.EQ.20) THEN
DECANOMIAL FUNCTION, MIN =0 AT X=(2, -3)
FP=0.D00
IF(X(1).LT.-4.D00 .OR. X(1).GT.4.D00) FP=FP+(100+DABS(X(1)))**2
IF(X(2).LT.-4.D00 .OR. X(2).GT.4.D00) FP=FP+(100+DABS(X(2)))**2
ENDIF
ELSE
CALL DECANOM(M,F,X)
ENDIF
RETURN
ENDIF
ENDIF
ENDIF
IF (KF.EQ.21) THEN
    CALL JUDGE(M, X, F)
    RETURN
ENDIF

IF (KF.EQ.22) THEN
    CALL DODECAL(M, F, X)
    RETURN
ENDIF

IF (KF.EQ.23) THEN
    WHEN X(1)*X(2)=X(1)*X(2) ? M=2
    CALL SEQP(M, F, X)
    RETURN
ENDIF

IF (KF.EQ.24) THEN
    WHEN ARITHMETIC MEAN = GEOMETRIC MEAN ?
    M =>1
    CALL AMGM(M, F, X)
    RETURN
ENDIF

IF (KF.EQ.25) THEN
    M =>2
    CALL FUNCT2(M, F, X)
    RETURN
ENDIF

IF (KF.EQ.26) THEN
    M =>2
    CALL FUNCT3(M, F, X)
    RETURN
ENDIF

IF (KF.EQ.27) THEN
    M =>2
    CALL FUNCT4(M, F, X)
    RETURN
ENDIF

IF (KF.EQ.28) THEN
    M =>2
    CALL FUNCT6(M, F, X)
    RETURN
ENDIF

IF (KF.EQ.29) THEN
    M =>2
    CALL FUNCT7(M, F, X)
    RETURN
ENDIF

IF (KF.EQ.30) THEN
    M =>2
    CALL FUNCT12(M, F, X)
    RETURN
ENDIF

IF (KF.EQ.31) THEN
    M =>2
    CALL FUNCT13(M, F, X)
    RETURN
ENDIF
805:     IF(KF.EQ.32) THEN
806:         M = 2
807:     CALL FUNCT14(M,F,X)
808:     RETURN
809:     ENDIF
810:     ---
811:     IF(KF.EQ.33) THEN
812:         M = 4
813:     CALL FUNCT15(M,F,X)
814:     RETURN
815:     ENDIF
816:     ---
817:     IF(KF.EQ.34) THEN
818:         LINEAR PROGRAMMING : MINIMIZATION PROBLEM
819:         M = 2
820:     CALL LINPROG1(M,F,X)
821:     RETURN
822:     ENDIF
823:     ---
824:     IF(KF.EQ.35) THEN
825:         LINEAR PROGRAMMING : MINIMIZATION PROBLEM
826:         M = 3
827:     CALL LINPROG2(M,F,X)
828:     RETURN
829:     ENDIF
830:     ---
831:     IF(KF.EQ.36) THEN
832:         TRIGONOMETRIC FUNCTION F(0, 0, ..., 0) = 0
833:     CALL TRIGON(M,F,X)
834:     RETURN
835:     ENDIF
836:     ---
837:     WRITE(*,*)'FUNCTION NOT DEFINED. PROGRAM ABORTED'
838:     STOP
839:     END
840:     ---
841:     SUBROUTINE DCS(M,F,X)
842:         FOR DEFLECTED CORRUGATED SPRING FUNCTION
843:         IMPLICIT DOUBLE PRECISION (A-H, O-Z)
844:         DIMENSION X(*),C(100)
845:         DATA K,ALPHA/5,5.000/ ! K AND ALPHA COULD TAKE ON ANY OTHER VALUES
846:         R2=0.000
847:         DO I=1,M
848:             C(I)=ALPHA
849:             R2=R2+X(I)-C(I)**2
850:         ENDDO
851:         R=DSQRT(R2)
852:         F=-DCOS(K*R)+0.1000*R2
853:     RETURN
854:     END
855:     ---
856:     SUBROUTINE QUINTIC(M,F,X)
857:         QUINTIC FUNCTION: GLOBAL MINIMA, EXTREMELY DIFFICULT TO OPTIMIZE
858:         IMPLICIT DOUBLE PRECISION (A-H, O-Z)
859:         DIMENSION X(*)
860:         DATA K,ALPHA/5,5.000/ ! K AND ALPHA COULD TAKE ON ANY OTHER VALUES
861:         R2=0.000
862:         DO I=1,M
863:             C(I)=ALPHA
864:             R2=R2+X(I)-C(I)**2
865:         ENDDO
866:         R=DSQRT(R2)
867:         F=-DCOS(K*R)+0.1000*R2
868:     RETURN
869:     END
870:     ---
871:     SUBROUTINE FACTOR1(M,F,X)
C DIMENSION X(*)
893: C X(I)=2  ! TO CHECK TURN IT ALIVE - REMOVE C FROM IST COLUMN
894: C X(2)=3  ! TO CHECK TURN IT ALIVE - REMOVE C FROM IST COLUMN
895: F1= DABS(X(1)**10-20*X(1)**9+180*X(1)**8-960*X(1)**7+  
896: & 3360*X(1)**6-8064*X(1)**5+13340*X(1)**4-15360*X(1)**3+  
897: & 11520*X(1)**2-5120*X(1)+2624.D00)
898: F2= DABS(X(2)**4+12*X(2)**3+54*X(2)**2+108*X(2)+81.D00)
899: F=F+F*DABS(P-FACT)**2
900: RETURN
901: END
902: C SUBROUTINE JUDGE(M,X,F)
903: C THIS SUBROUTINE IS FROM THE EXAMPLE IN JUDGE ET AL., THE THEORY  
904: C AND PRACTICE OF ECONOMETRICS, 2ND ED., PP. 956-7. THERE ARE TWO  
905: C OPTIMA: F(0.86479,1.2357)=16.0817307 (WHICH IS THE GLOBAL MINIMUM)  
906: C AND F(2.35,-0.319)=20.9805 (WHICH IS LOCAL). ADAPTED FROM BILL  
907: C GOFTE'S SIMMAN (SIMULATED ANNEALING) PROGRAM
908: C IMPPLICIT DOUBLE PRECISION (A-H, O-Z)
909: C DIMENSION Y(N), X2(N), X3(N), X(*)
910: & 3.231,1.998,1.379,2.106,1.428,1.011,2.179,2.858,1.388,1.651,
911: & 1.593,1.046,2.152/
912: DATA (Y(I),I=1,N)/4.286,4.149,3.877,0.533,2.211,2.389,2.145,
913: & .797,.936,.889,.006,.828,.399,.617,.939,.784,.072,.889/  
914: DATA (X2(I),I=1,N)/.265,.585,.310,.058,.455,.779,.259,.202,.028,
915: & .099,.142,.296,.175,.180,.842,.039,.103,.620,.158,.704/
916: & F=0.D00
917: DO I=1,N  
918: F=F+(X(I)) + (X(2)**2)*X2(I) + (X(2)**2)*X3(I) = Y(I))**2
919: ENDDO
920: RETURN
921: END
922: C SUBROUTINE DODECAL(M,F,X)
923: C DODECAL POLYNOMIAL MIN F(1,2,3)=0
924: C CHECK THESE VALUES ALIVE
925: C X(1)=1  I TURN ALIVE PY REMOVING C FROM THE FIRST COLUMN
926: C X(2)=2  I TURN ALIVE PY REMOVING C FROM THE FIRST COLUMN
927: C X(3)=3  I TURN ALIVE PY REMOVING C FROM THE FIRST COLUMN
928: F=F+D00
929: F1=2*X(1)**3+5*X(1)*X(2)+4*X(3)-2*X(1)**2*X(3)-18.D00
930: F2=X(1)+X(2)**3+X(1)*X(2)**2+X(1)*X(3)**2-22.D00
931: F3=8*X(1)**2+2*X(2)**2*X(3)+2*X(2)**2*X(3)+2*X(3)**3+52.D00
C     MIN F (0, 0, ..., 0) = 0
C     MIT PRESS, CAMBRIDGE, MASS.
C     SOME POSITIVE NUMBER. HERE IT IS 100; IT COULD BE ANYTHING ELSE.
C     SET SUM = SOME POSITIVE NUMBER. THIS MAKES THE FUNCTION UNIMODAL
C     TAKE ONLY THE ABSOLUTE VALUES OF X
C     IF X(1)=X(2)=....=X(M) AND ALL X ARE NON-NEGATIVE
C     -----------------------------------------------------------------
C      X(2)=DABS(X(2)) ! ONLY NON-NEGATIVE VALUES
C      X(1)=2  !SET X(1) TO 0 OR 2
C     ----------------------------------------------------------------
RPS-H2.f 15/21
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
DIMENSION X(*)
	C     FOR WHAT VALUES X(1)+X(2)=X(1)*X(2) ? ANSWER: FOR (0, 0) AND (2, 2)
	C     CHECK THE ANSWER FMIN = 0 FOR X=(0, 0) OR X=(2, 2)
	C     X(1)=2  ISET X(1) TO 0 OR 2
	C     X(2)=2  ISET X(2) TO X(1)
1000:  C     X(1)-DABS(X(1)) ! ONLY NON-NEGATIVE VALUES
1001:  C     X(2)=DABS(X(2)) ! ONLY NON-NEGATIVE VALUES
1002:  F1=X(1)+X(2)
1003:  F2=X(1)*X(2)
1004:  C     SET EITHER OF THE TWO ALIVE BY REMOVING C FROM THE FIRST COLUMN
1005:  F=(F1-F2)**2  ! TURN ALIVE THIS XOR
1006:  C     F=DABS(F1-F2)  ! TURN ALIVE THIS - BUT NOT BOTH
1007:  RETURN
1008:  END
1009:  C     FOR WHAT VALUES ARITHMETIC MEAN = GEOMETRIC MEAN ? THE ANSWER IS:
1010:  C     IF X(1)=X(2)=....=X(M) AND ALL X ARE NON-NEGATIVE
1011:  C     TAKE ONLY THE ABSOLUTE VALUES OF X
1012:  SUM=0.D00
1013:  DO I=1,M
1014:  X(I)=DABS(X(I))
1015:  ENDDO
1016:  C     SET SUM = SOME POSITIVE NUMBER. THIS MAKES THE FUNCTION UNIMODAL
1017:  SUM= 100.D00  ! TURNED ALIVE FOR UNIQUE MINIMUM AND SET SUM TO
1018:  C     SOME POSITIVE NUMBER. HERE IT IS 100; IT COULD BE ANYTHING ELSE.
1019:  F1=0.D00
1020:  F2=1.D00
1021:  DO I=1,M
1022:  IF I+X(I)
1023:  DO I=1,M
1024:  IF I+X(I)
1025:  END
1026:  XSUM=F1
1027:  F2=F2**((1.D00/M))  ! MTH ROOT OF THE PRODUCT = GEOMETRIC MEAN
1028:  F=(F1-F2)**2
1029:  IF (SUM GT 0.D00) F=F+(SUM-XSUM)**2
1030:  RETURN
1031:  END
1032:  C     SUBROUTINE FUNCT2(M,F,X)
1034:  C     IN FOGEL, L.J., ANGELIN, P. J. AND BACK, T. (ED) PROCEEDINGS OF THE
1035:  C     FIFTH ANNUAL CONFERENCE ON EVOLUTIONARY PROGRAMMING, PP. 451-460,
1036:  C     MIT PRESS, CAMBRIDGE, MASS.
1037:  C     MIN F (0, 0, ..., 0) = 0
1038:  C     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1039:  DIMENSION X(*)
1040:  F=0.D00
1041:  F1=1.D00
1042:  FP=0.D00
1043:  DO I=1,M
1044:  IF (DABS(X(I)) GT 10.D00) FP=FP+(100.D00+DABS(X(I)))**2
1045:  ENDDO
1046:  IF (FP NE 0.D00) THEN
1047:  F=FP
1048:  ELSE
1049:  DO I=1,M
1050:  END
1006: \[ F = F + \text{DABS}(X(I)) \]
1007: \[ F_1 = F_1 \times \text{DABS}(X(I)) \]
1008: END
1009: \( F = F + F_1 \)
1010: RETURN
1011: ENDIF
1012: END
1013: \[ \text{SUBROUTINE FUNCT3(M,F,X)} \]
1014: \[ \text{REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING} \]
1015: \[ \text{MIN F (0, 0, ... , 0) = 0} \]
1016: \[ \text{DIMENSION X(*)} \]
1017: \( F = 0.000 \)
1018: \( F_1 = 0.000 \)
1019: \( F_P = 0.000 \)
1020: DO I = 1, M
1021: IF(\text{DABS}(X(I)) .GT. 100.000) FP = FP + (100.000 + \text{DABS}(X(I)))**2
1022: ENDDO
1023: IF(FP .NE. 0.000) THEN
1024: \( F = FP \)
1025: RETURN
1026: ELSE
1027: DO I = 1, M
1028: \( F_1 = 0.000 \)
1029: DO J = 1, I
1030: \( F_1 = F_1 + X(J)**2 \)
1031: ENDDO
1032: \( F = F + F_1 \)
1033: \( F_1 = 0.000 \)
1034: \( F = F + F_1 \)
1035: \( F = F \)
1036: RETURN
1037: ENDIF
1038: END
1039: \[ \text{SUBROUTINE FUNCT4(M,F,X)} \]
1040: \[ \text{REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING} \]
1041: \[ \text{MIN F (0, 0, ... , 0) = 0} \]
1042: \[ \text{DIMENSION X(*)} \]
1043: \( F = 0.000 \)
1044: \( F_P = 0.000 \)
1045: DO I = 1, M
1046: IF(X(I) .LT. 0.000 .OR. X(I) .GE. M) FP = FP + (100.000 + \text{DABS}(X(I)))**2
1047: ENDDO
1048: IF(FP .NE. 0.000) THEN
1049: \( F = FP \)
1050: RETURN
1051: ELSE
1052: DO I = 1, M
1053: \( F = 0.000 \)
1054: \( X\_MAX = X(I) \)
1055: \( DO I = 1, M \)
1056: \( IF(X\_MAX .LT. X(I)) X\_MAX = X(I) \)
1057: \( END \)
1058: \( F = X\_MAX \)
1059: \( END \)
1060: RETURN
1061: \( END \)
1062: \( END \)
1063: \[ \text{SUBROUTINE FUNCT6(M,F,X)} \]
1064: \[ \text{REF: YAO, X. AND LIU, Y. (1996): FAST EVOLUTIONARY PROGRAMMING} \]
1065: \[ \text{MIN F (-0.5, -0.5, ... , -0.5) = 0} \]
1066: \[ \text{DIMENSION X(*)} \]
1067: \( F = 0.000 \)
1068: \( F_P = 0.000 \)
1069: DO I = 1, M
1070: \( F = 0.000 \)
1071: \( DO I = 1, M \)
1072: \( IF(DABS(X(I)) .GT. 100.000) FP = FP + (100.000 + \text{DABS}(X(I)))**2 \)
1073: \( END \)
1073:   ENDDO
1074:   IF (FP .NE. 0.00) THEN
1075:     F = FP
1076:   RETURN
1077:   ELSE
1078:     DO I = 1, M
1079:       F = F + (X(I) + 0.5D00)**2
1080:     ENDDO
1081:     RETURN
1082:   ENDF
1083:   END
1084:   ________________________________________________________________
1085:   C SUBROUTINE FUNCT7(M,F,X)
1087:   C MIN F(-1, -1, -1, ..., -1) = 0
1088:   C MIN F(0, 0, ..., 0) = 0
1089:   C MIN F(0, 0, ..., 0) = 0
1091:   ________________________________________________________________
1092: F2
1093: Y
1094: DO
1095: F2
1096: ENDDO
1097: F1
1098: U
1099: XX
1100: DO
1101: F1
1102: RETURN
1103: IF
1104: CALL RANDOM(RAND)
1105: F = F + (I*F(I)+**4)
1106: ENDDO
1107: CALL RANDOM(RAND)
1108: F = F + RAND
1109: RETURN
1110: ENDF
1111: END
1112: ________________________________________________________________
1113: C SUBROUTINE FUNCT12(M,F,X)
1115: C MIN F(-1, -1, -1, ..., -1) = 0
1116: C MIN F(0, 0, ..., 0) = 0
1117: C MIN F(0, 0, ..., 0) = 0
1119: C IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1120: C IMPLICIT DOUBLE PRECISION (A-H, O-Z)
1121: C DIMENSION X(*)
1122: C DIMENSION X(*)
1123: COMMON /RNDM/IV,IV
1124: COMMON /RNDM/IV,IV
1125: INTEGER IU,IV
1126: INTEGER IU,IV
1127: DATA A,B,C /10.D00,100.D00,4.D00/
1128: DATA A,B,C /10.D00,100.D00,4.D00/
1129: PI = 4.00*Datan(1.00)
1130: PI = 4.00*Datan(1.00)
1131: F = 0.00
1132: F = 0.00
1133: FP = 0.00
1134: FP = 0.00
1135: DO I = 1, M
1136: DO I = 1, M
1137: C IF(DABS(X(I)) .GT. 1.28D00) FP = FP + (100.D00 + DABS(X(I)))**2
1138: C IF(DABS(X(I)) .GT. 1.28D00) FP = FP + (100.D00 + DABS(X(I)))**2
1139: END DO
1140: IF (FP .NE. 0.00) THEN
1141: IF (FP .NE. 0.00) THEN
1142: F = FP
1143: F = FP
1144: RETURN
1145: RETURN
1146: ELSE
1147: ELSE
1148: F1 = 0.00
1149: F1 = 0.00
1150: DO I = 1, M
1151: DO I = 1, M
1152: XX = DABS(X(I))
1153: XX = DABS(X(I))
1154: U = 0.00
1155: U = 0.00
1156: IF (XX .GT. A) U = B*(XX - A)**2C
1157: IF (XX .GT. A) U = B*(XX - A)**2C
1158: F1 = F1 + U
1159: F1 = F1 + U
1160: ENDDO
1161: ENDDO
1162: F2 = 0.00
1163: F2 = 0.00
1164: DO I = 1, M - 1
1165: DO I = 1, M - 1
1166: Y(I) = 1.00 + .25D00*(X(I) + 1.00)
1167: Y(I) = 1.00 + .25D00*(X(I) + 1.00)
1168: F2 = F2 + (Y(I) - 1.00)**2 * (1.00 + 10.00*Dsin(PI*X(I+1))**2))
1169: F2 = F2 + (Y(I) - 1.00)**2 * (1.00 + 10.00*Dsin(PI*X(I+1))**2))
ENDO

Y(N)=1.D00+.25D00*(X(M)+1.D00)

F3=(Y(M)-1.D00)**2

Y(1)=1.D00+.25D00*(X(1)+1.D00)

F4=10.d00*(DSIN(Pi*Y(1)))**2

F=(Pi/M)*(F4+F2+F3)+F1

RETURN

ENDIF

END

C

SUBROUTINE FUNCT13(M,F,X)

C

CALL YAO, X, AND LIU, Y (1996): FAST EVOLUTIONARY PROGRAMMING

C

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

C

DIMENSION X(100)

C

DATA A,B,C /5.D00,100.D00,4.D00/

C

PI=4*DATAN(1.D00)

C

FP=0.D00

C

DO I=1,M

C

MIN F (1, 1, 1, ..., 4.7544 APPROX) = -1.15044 APPROX

C

X(I)=-1.D00 ! TO CHECK, TURN IT ALIVE

C

MIN F (1, 1, 1, ..., 4.7544 APPROX) = -1.15044 APPROX

ENDO

IF(FP.DEB.0.D00) THEN

F=FP

RETURN

ELSE

F1=0.D00

DO I=1,M

XX=DSABS(X(I))

U=0.D00

IF(XX.GT.A) U=B*(XX-A)**2*C

F1=F1+U

ENDDO

F2=0.D00

DO I=1,M

F2=F2+ (X(I)-1.D00)**2 * (1.D00+((DSIN(3*PI*X(I+1)))**2))

ENDDO

F3=(X(M)-1.D00)* (1.D00+((DSIN(2*PI*X(M)))**2)

F4=((DSIN(3*PI*X(1)))**2

F=-0.1*(F4+F2+F3)+F1

RETURN

ENDIF

END

C

SUBROUTINE FUNCT14(M,F,X)

C

CALL YAO, X, AND LIU, Y (1996): FAST EVOLUTIONARY PROGRAMMING

C

MIN F (-31.98, 31.98) = 0.998

C

PARAMETER (N=25, NN=2)

C

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

C

DIMENSION X (2), A(NN,N)

C

DATA (A(1,J),J=1,N) /-32.D00,-16.D00,0.D00,16.D00,32.D00,-32.D00,

& -16.D00,0.D00,16.D00,32.D00,-32.D00,-16.D00,0.D00,16.D00,32.D00,

& -32.D00,-16.D00,0.D00,16.D00,32.D00,-32.D00,-16.D00,0.D00,16.D00,32.D00/}

C

DATA (A(2,J),J=1,N) /-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,-32.D00,


& 32.D00,32.D00,32.D00,32.D00/}

C

FOR TEST TURN VALUES OF X(1) AND X(2) BELOW ALIVE

C

X(1)=31.98 ! TURN ALIVE

C

X(2)=-31.98 ! TURN ALIVE

F=0.D00

FP=0.D00

DO I=1,M

IF(DABS(X(I)).GT.100.D00) FP=FP+(DABS(X(I)))**2

ENDO
DATA A(1),A(2),A(3),A(4),A(5),A(6),A(7),A(8),A(9),A(10),A(11),A(12),A(13),A(14)
DIMENSION A(1),A(2),A(3),A(4),A(5),A(6),A(7),A(8),A(9),A(10),A(11),A(12),A(13),A(14)
DO I=1, N
FL=F+I**2+F**2
ENDO
RETURN
ENDIF
RETURN
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
;t
 Lori


DO I=1,M
X(I)=DABS(X(I))
ENDDO
C EVALUATION OF OBJ FUNCTION AND CONSTRAINTS
DO I=1,N
FF(I)=-0.D00
DO J=1,M
FF(I)=FF(I)+A(I,J)*X(J)
ENDDO
ENDDO
DO I=2,N
FF(I)=FF(I)-C(I) ! SLACK
ENDDO
IF(FF(I).GT.0) F=F+(10+FF(I))**2
ENDDO
RETURN
ENDDO
C
PARAMETER (N=4) ! N IS THE NO. OF CONSTRAINTS + 1
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
DIMENSION X(*),A(20,10),C(20),FF(20)
DATA (A(1,J),J=1,3),C(1)/30.D0,40.D0,20.D0,0.0D0/! COEFF OF OBJ FUNCTION
DATA (A(2,J),J=1,3),C(2)/10.D0,12.D0,7.D0,10000.0D0/!COEFF OF 1ST CONSTRAINT
DATA (A(3,J),J=1,3),C(3)/7.D0,10.D0,8.D0,8000.0D0/! COEFF OF 2ND CONSTRAINT
DATA (A(4,J),J=1,3),C(4)/1.D0,1.D0,1.D0,1000.D0/! COEFF OF 3RD CONSTRAINT
C USING ONLY NON-NEGATIVE VALUES OF X(I)
DO I=1,M
X(I)=DABS(X(I))
ENDDO
C EVALUATION OF OBJ FUNCTION AND CONSTRAINTS
DO I=1,N
FF(I)=-0.D00
DO J=1,M
FF(I)=FF(I)+A(I,J)*X(J)
ENDDO
ENDDO
DO I=2,N
FF(I)=FF(I)-C(I) ! SLACK
ENDDO
IF(FF(I).GT.0) F=F+(10+FF(I))**2
ENDDO
RETURN
ENDDO
C
SUBROUTINE TRIGON(M,F,X)
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
DIMENSION X(*)
F MIN (PI, 0, 0, ..., 0) OR (PI, 0, 0, ..., 0) = 0
PI=4*DATAN(1.D00)
F=0.D00
DO I=1,M
IF(X(I).lt.0.0D0 .or. X(I).GT.PI) THEN
F=F+(DABS(X(I))+10.0D0)**2
1341:     ENDIF
1342:     ENDDO
1343:     DO I=2,M
1344:     F=F+(dcos(I+0.D00)*DSIN(X(I)-X(I-1)))**2 +
1345:     & (i-1.D00)*(1.DO-DCOS(X(I))))**2
1346:     ENDDO
1347:     RETURN
1348:     END
1349:     C  ===============================================================