Iraqi Journal of Statistical Sciences (19) 2011
p.p. [21-38]

Spectral Fletcher-Reeves Algorithm for Solving Non-Linear Unconstrained Optimization Problems

Khalil K. Abbo[*] Farah H. Mohammed[**]

Abstract

The non-linear conjugate gradient method is a very useful technique for solving Large-Scale minimization problems and has wide applications in many fields . In this paper, we present a new spectral type, a non-linear conjugate gradient algorithm the derivatation of this algorithm is based on Fletcher – Reeves and Newton algorithm, the descent property for the suggested algorithm is proved provided that the step size α_k satisfies the Wolfe conditions. Numerical results show that the new algorithm is efficient in practical computation and superior to the Fletcher – Reervs algorithm in many situations.

.

الملخص

تم في هذا البحث تطوير خوارزمية Fletcher-Reeves الى نوع من الخوارزميات المتجهات المترافقة الطيفية علما ان تطوير الخوارزمية اعتمد على طريقة Neuton . وبرهنا خاصية الانحدار السلبي للخوارزمية المطورة باستخدام شرط Wolfe . تشير النتائج العددية الى كفاءة الطريقة بالمقارنة مع بعض الخوارزميات المعروفة في هذا المجال .

[*] Department of Math /College of Computers Sciences and Mathematics/ University of Mosul

[**]Department of Math /College of Computers Sciences and Mathematics/ University of Mosul

Received: 8/12 /2010 ___________________Accepted:2 / 2 / 2011

1 . Introduction

Consider the Unconstrained Optimization problem

$$\min f(x), \quad x \in R^n \qquad (1)$$

where $f : R^n \to R$ is continuously differentiable function . The line search algorithm for (1) often generates a sequence of iterates (x_k) by letting

$$x_{k+1} = x_k + \alpha_k d_k \qquad (2)$$

Where x_k is the current iterate point, d_k is a search direction, and $\alpha_k > 0$ is step – length. Different choices of d_k and α_k will determine different line search methods [1,2] . These methods are devided into two stages at each iteration [3] :

1 . choose a descent search direction d_k i.e

$$g_k^T d_k < 0 \qquad (3)$$

2. choose a step – length α_k along the search direction d_k .

Throughout this paper, we denote $f(x_k)$ by f_k, $\nabla f(x_k)$ by g_k and $\nabla f(x_{k+1})$ by g_{k+1} , $\|.\|$ denotes the Euclidian norm of vectors .

One simple line search method is the steepest descent method, if we take $d_k = -g_k$ as a search direction at every iteration, which has wide application in solving large-scale minimization problems and training feed forward neural networks [4]. However, the steepest descent method often yields zig-zag phenomena in solving practical problems , which makes the algorithm converges to an optimal solution very slowly, or even fail to converge [5]. Then the steepest descent method is not the fastest one among the line search methods. If $d_k = -H_k g_k$ is the search direction at each iteration in the algorithm, where is an n x n matrix approximating $[\nabla^2 f(x)]^{-1}$, then the corresponding line search method is called Newton like method [5] such as Newton method, Quasi – Newton method, Variable metric method etc. Many papers [6] have been proposed by the method for optimization problems. However, one drawback of the Newton like methods is required to store and compute matrix H_k at each iteration and thus adds the cost of storage and computation. Accordingly, this method is not suitable to solve large scale optimization problems in many cases. When n is large the related problem is called large – scale minimization problem. In order to solve large–scale minimization problems, We need to design special algorithms that avoid the high storage and computation cost of some matrices.

The conjugate gradient method is a suitable approach for solving large – scale minimization problems. For strictly convex quadratic objective functions, the conjugate gradient method with exact line searches has finite convergence property [9]. If the objective function is not quadratic or inexact line searches are used, the conjugate gradient method has no quadratic convergence property [7]. When the conjugate gradient method is used to minimize non–quadratic objective functions, the related algorithm is called non – linear conjugate gradient method. Meanwhile, some new non – linear conjugate gradient methods have appeared in [8].

The non – linear conjugate gradient (CG) method has the form

$$x_{k+1} = x_k + \alpha_k \, d_k \qquad (4)$$

Where x_1 is an initial paint, α_k is a step – length and d_1 can be taken as $d_1 = -g_1$ and

$$d_{k+1} = -g_{k+1} + \beta_{k+1} d_k \qquad (5)$$

Different β_{k+1} will determine different CG method. Some famous formula for β_{k+1} as follows :

$$\beta_{k+1}^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} \qquad \text{(Fletcher-Reeves [9])} \qquad (6)$$

$$\beta_{k+1}^{HS} = \frac{g_{k+1}^T y_k}{y_k^T d_k} \qquad \text{(Hestenes-Stiefel [10])} \qquad (7)$$

$$\beta_{k+1}^{DY} = \frac{g_{k+1}^T g_{k+1}}{y_k^T d_k} \qquad \text{(Dai- Yuan [11])} \qquad (8)$$

Where $y_k = g_{k+1} - g_k$. Although some conjugate gradient methods have good numerical performance in solving Large-Scale minimization problems, they have no global convergence in some situations such as Hestenens – Stiefiel conjugate gradient method [12] , and others has global convergence theoretically but don't performe well in practice such as Fletcher – Reeves (β_{k+1}^{FR}) method. We often have two questions. Can we construct a conjugate gradient (CG) method that has both global convergence and good numerical performance in practical computation? or can we design a conjugate gradient method that is suitable to solve ill - conditioned minimization problems?

Yuan and Stoer in [13] studied the CG method on a subspace and they obtained a new conjugate gradient method. In their algorithm, the search direction was taken from the subspace [g_{k+1}, d_k] at the (k+1)th iteration (k ≥ 1) i.e .

$$d_{k+1} = -\gamma_{k+1}\, g_{k+1} + \beta_{k+1}\, d_k \qquad (9)$$

Where γ_{k+1} and β_{k+1} are two parameters. The other important aspect in the line search algorithms is the computing the step – length α_k, it has an important influence on the amount of calculations at each iteration. There are several line search rules for choosing step – length α_k for example, the exact minimization rule, the step – length α_k is chosen such that

$$f(x_k + \alpha_k d_k) = \text{ary}\min_{\alpha>0} f(x_k + \alpha\, d_k) \quad (10)$$

Armijo rule, Wolf rule, etc. In this paper we use the following two line search conditions:

1. The Wolfe – Powell (WWP) line search condition:

$$f_{k+1} \le f_k + \rho\, \alpha_k g_k^T d_k \quad (11a)$$

$$g_{k+1}^T d_k \ge \sigma\, g_k^T d_k \quad (11b)$$

2. The Strong Wolfe – Powell (SWP) : The equation (11a) with

$$\left| g_{k+1}^T d_k \right| \le -\sigma\, g_k^T d_k \quad (11c)$$

where $\rho \in (0,1)$ *and* $\sigma \in (\rho, 1)$

In general conjugate gradient methods usually implemented with restart since the rate of convergence of the algorithm is only linear unless the iterative procedure is restarted occasionally. It is usual to restart at every n or n+1 iterations but this not satisfactory since n is large, therefore other restarts are used such as Powell restarts [15] defined by

$$\left| g_{k+1}^T g_k \right| \ge 0.2 * \| g_k \| \quad (12)$$

2 . New Proposed Algorithm (SFR – CG say)

The search direction for the Fletcher – Reeves conjugate gradient (FR – CG) method is obtained by $d_1 = -g_1$ and

$$d_{k+1} = -g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} d_k \qquad (13)$$

Dai and Yuan in [14] established global convergence results for FR – CG method for any starting point with exact and inexact line searches, therefore it has better convergence property theoretically but it is not recommended for practical use since it has poor performance in practice see [16]. In order to accelerate the FR – CG method we use equation (9) as follows

Let $\gamma_{k+1} = 1 + \mu_{k+1}$ (14)

Where γ_{k+1} and μ_{k+1} are two parameters, then

$$d_{k+1} = -(1 + \mu_{k+1}) g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} d_k \qquad (15)$$

Or $d_{k+1} = -g_{k+1} - \mu_{k+1} g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} d_k$ (16)

We incorporate the second order information to the search direction in (16) by assuming, the direction in (16) is parallel to the Newton direction i.e $-G_{k+1}^{-1} g_{k+1} = -g_{k+1} - \mu_{k+1} g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} d_k$

(17) Where $-G_{k+1}^{-1}$ is the inverse Hessian matrix. Now suppose $-G_{k+1}^{-1}$ is symmetric $(G^{-1} = (G^{-1})^T)$, positive definite and satisfies the Quaci – Newton condition i.e

$$G_{k+1}^{-1} y_k = s_k \qquad (18)$$

Where $s_k = x_{k+1} - x_k$ multiply both sides of (17) by y_k and considering G_{k+1}^{-1} be symetric and positive definite, then

$$-(G_{k+1}^{-1} y_k)^T g_{k+1} = -y_k^T g_{k+1} - \mu_{k+1} y_k^T g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} y_k^T d_k$$

Use the relation given in (18) to get

$$-s_k^T g_{k+1} = -y_k^T g_{k+1} - \mu_{k+1} y_k^T g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k} y_k^T d_k$$

Divide both sides in the above equation by $y_k^T d_k$ then

$$-\frac{s_k^T g_{k+1}}{y_k^T d_k} = -\beta_{k+1}^{HS} - \mu_{k+1} \beta_{k+1}^{HS} + \beta_{k+1}^{FR}$$

Or $$-\frac{s_k^T g_{k+1}}{y_k^T d_k} = -(1 + \mu_{k+1}) \beta_{k+1}^{HS} + \beta_{k+1}^{FR}$$

Or $$(1 + \mu_{k+1}) \beta_{k+1}^{HS} = \beta_{k+1}^{FR} + \frac{s_k^T g_{k+1}}{y_k^T d_k}$$

$$\therefore \gamma_{k+1} = \frac{\beta_{k+1}^{FR}}{\beta_{k+1}^{HS}} + \frac{s_k^T g_{k+1}}{y_k^T g_{k+1}} \tag{19}$$

Use this value for the γ_{k+1} in (15) then

$$d_{k+1} = -\gamma_{k+1} \, g_{k+1} + \beta_{k+1}^{FR} \, d_k \tag{20}$$

We call the algorithm defined in (19) and (20) as spectral Fletcher – Reeves algorithm (SFR – CG) and we summarize it as the following algorithm SFR – CG.

Algorithm(SFR-CG) :

steep(1) : Initialization : select $x_1 \in R^n, \varepsilon > 0$, is small positive real value

and compute

$d_1 = -g_1, \quad \alpha_1 = 1 l \|g_1\|$ and $k = 1$

step(2) : Test for convergence: If $\|g_k\| \le \varepsilon$, stop x_k is optimal solution

else go to step(3).

step(3) : Line search : compute α_k satisfying the wolf conditions (11a),

(11b) and update the variable $x_{k+1} = x_k + \alpha_k d_k$, compute

$f_{k+1},\ g_{k+1},\ y_k \ and \ s_k$.

step(4) : Direction computation : compute γ_{k+1} from (19), if

$\gamma_{k+1} \ge 1 \ or \ \gamma_{k+1} \le 0$ set $\gamma_{k+1} = 1$ and $d = -\gamma_{k+1}\, g_{k+1} + \beta_{k+1}^{FR}\ d_k$

If Powell restart (12) is satisfied then $d_{k+1} = -\gamma_{k+1}\, g_{k+1}$ else $d_{k+1} = d$

and $\alpha_{k+1} = \alpha_k * \|d_k\| / \|d_{k+1}\|$, $k = k+1$ go to step(2).

3. . Descent property

In this section we prove that the our algorithm defined in the equations (19) and (20) generates descent directions for all iteration according to the following theorem

Theorem: Consider the algorithm defined in equation (4) where d_k computed from (19) and (20). Assume that the step size α_k satisfies the Wolfe conditions (11a) and (11b). Then the search directions d_k generated by the SFR – CG algorithm are descent for all k provided $y_k^T g_{k+1} > 0$.

Proof

The prove is by indication, for $k=1$, $d_1 = -g_1 \rightarrow d_1^T g_1 = -\|g_1\| < 0$

now suppose $d_k^T g_k < 0$ or $s_k^T g_k < 0$ since $s_k = \alpha_k d_k$, then for $k+1$ we have

$$d_{k+1} = -(\frac{\beta^{FR}}{\beta^{HS}} + \frac{s_k^T g_{k+1}}{y_k^T g_{k+1}})\, g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}\, d_k$$

$$d_{k+1}^T g_{k+1} = -(\frac{g_{k+1}^T g_{k+1}}{\alpha_k . g_k^T g_k} . \frac{s_k^T y_k}{y_k^T g_{k+1}} + \frac{s_k^T g_{k+1}}{y_k^T g_{k+1}})\, g_{k+1}^T g_{k+1} + \frac{g_{k+1}^T g_{k+1}}{\alpha_k . g_k^T g_k}\, s_k^T g_{k+1}$$

Divide both sides by $\dfrac{g_{k+1}^T g_{k+1}}{\alpha_k . g_k^T g_k}$ then

$$\alpha_k \frac{g_k^T g_k}{. g_{k+1}^T g_{k+1}}\, d_{k+1}^T g_{k+1} = -\frac{1}{y_k^T g_{k+1}} (s_k^T y_k g_k^T g_{k+1} + \alpha_k g_k^T g_k s_k^T g_{k+1}) + s_k^T g_{k+1}$$

$$\because\ s_k^T g_{k+1} = s_k^T g_{k+1} - s_k^T g_k + s_k^T g_k = s_k^T y_k + s_k^T g_k < s_k^T y_k$$

$$\therefore\ \alpha_k \frac{g_k^T g_k}{.g_{k+1}^T g_{k+1}} d_{k+1}^T g_{k+1} \le -\frac{1}{y_k^T g_{k+1}} (s_k^T y_k g_{k+1}^T g_{k+1} + \alpha_k g_k^T g_k s_k^T y_k) + s_k^T y_k$$

$$\therefore\ d_{k+1}^T g_{k+1} \le -\frac{s_k^T y_k}{y_k^T g_{k+1}} (g_{k+1}^T g_{k+1} + \alpha_k g_k^T g_k - y_k^T g_{k+1}) \frac{g_{k+1}^T g_{k+1}}{\alpha_k g_k^T g_k}$$

$$d_{k+1}^T g_{k+1} \le -\frac{s_k^T y_k}{y_k^T g_{k+1}} (\alpha_k g_k^T g_k + g_k^T g_{k+1}) \frac{g_{k+1}^T g_{k+1}}{\alpha_k g_k^T g_k}$$

$$= -\frac{s_k^T y_k}{y_k^T g_{k+1}} (g_{k+1}^T g_{k+1} + \frac{g_{k+1}^T g_{k+1} g_{k+1}^T g_k}{\alpha_k g_k^T g_k})$$

Use the Cuchy – Schuarz inequality then

$$\le -\frac{s_k^T y_k}{y_k^T g_{k+1}} \|g_{k+1}\|^2 (1 + \frac{\|g_{k+1}\|\|g_k\|}{\alpha_k \|g_k\|^2}) = -\frac{s_k^T y_k}{y_k^T g_{k+1}} \|g_{k+1}\|^2 (1 + \frac{1}{\alpha_k}\sqrt{\beta^{FR}})$$

$s_k^T y_k > 0$ by Wolfe condition and $y_k^T g_{k+1} > 0$ by assumption

$\therefore d_{k+1}^T g_{k+1} < 0$

The proof is complete.

4 . Computational Results and Comparisons.

This section presents the performance of FORTRAN implementation of our new spectral conjugate gradient algorithm (SFR -CG) on a set of unconstrained optimization test problems taken from [17]. We select (15) Lange – Scale test problems in extended or generalized from (see Appendix), for each function we have considered numerical experiments with the number of variables n=100 , 1000 and 10000. We have compared the performance of this algorithms versus To FR –

CG algorithm. These algorithms are implemented with standard Wolfe conditions with $\rho = 0.001$ *and* $\sigma = 0.9$ where the initial step – size $\alpha_1 = \frac{1}{\|g_1\|}$ and initial guess for other iterations i.e.

$(k > 1)$ is $\alpha_k = \alpha_{k-1} * \|d_{k-1}\| / d_k$. In the all cases the stopping criterion is $\|g_{k+1}\| \leq 10^{-6}$ and maximum number of iteration is 2000. Our comparison includes the following

1. NoI : Number of iterations.

2. FGE :Number of function and gradient evaluations.

3. LIN : the number of calling line search subroutin.

4. The total time required to solve (15) problem in particular dimension.

Tables (1) , (2) and (3) show the details of the results for (SFR – CG) algorithms versuse FR-CG algorithm

Table (1) Comparison of algorithms for N=100

Test Fun	Dim n	FR-CG algorithm NOI / FGE / LIN	SFR-CG algorithm NOI / FGE / LIN
1	**100**	**18 / 34 / 13**	**18 / 33 / 12**
2	**100**	**42 / 86 / 35**	**43 / 91 / 35**
3	**100**	**36 / 76 / 31**	**31 / 67 / 25**
4	**100**	**11 / 28 / 10**	**9 / 25 / 9**
5	**100**	**10 / 19 / 8**	**10 / 20 / 9**
6	**100**	**68 / 130 / 61**	**73 / 137 / 63**
7	**100**	**63 / 98 / 33**	**63 / 98 / 33**
8	**100**	**72 / 164 / 69**	**65 / 154 / 63**
9	**100**	**50 / 89 / 38**	**48 / 85 / 36**
10	**100**	**88 / 190 / 86**	**88 / 192 / 86**
11	**100**	**77 / 120 / 42**	**79 / 120 / 40**
12	**100**	**30 / 48 / 17**	**31 / 52 / 20**
13	**100**	**24 / 46 / 19**	**24 / 43 / 16**
14	**100**	**17 / 33 / 15**	**17 / 33 / 15**

15	**100**	**28 / 49 / 18**	**29 / 50 / 18**
Total		**634 / 1210 / 493**	**628 / 1138 / 444**

Total Time : 3.027 Sc 3.021 Sc

Table (2) Comparison of algorithms for N=1000

Test Fun	Dim N	FR-CG algorithm NOI / FGE / LIN	SFR-CG algorithm NOI / FGE / LIN
1	**1000**	**40 / 69 / 24**	**39 / 69 / 25**
2	**1000**	**42 / 96 / 40**	**38 / 84 / 34**
3	**1000**	**37 / 80 / 33**	**37 / 77 / 29**
4	**1000**	**94 / 96 / 86**	**22 / 47 / 13**
5	**1000**	**22 / 35 / 12**	**11 / 21 / 9**
6	**1000**	**85 / 62 / 76**	**78 / 149 / 70**
7	**1000**	**67 / 105 / 35**	**67 / 105 / 35**
8	**1000**	**170 / 336 / 86**	**73 / 179 / 71**
9	**1000**	**167 / 290 / 122**	**165 / 296 / 130**
10	**1000**	**86 / 184 / 84**	**93 / 199 / 91**
11	**1000**	**247 / 409 / 161**	**230 / 391 / 160**
12	**1000**	**33 / 63 / 26**	**36 / 62 / 22**
13	**1000**	**70 / 1313 / 60**	**67 / 1063 / 55**
14	**1000**	**19 / 42 / 18**	**22 / 49 / 21**
15	**1000**	**132 / 565 / 129**	**100 / 333 / 94**
Total		**1311 / 3842 / 992**	**1078 / 3124/ 859**

Total Time : 359 Sc 327 Sc

Table (3) Comparison of algorithms for N=10000

Test Fun	Dim n	FR-CG algorithm NOI / FGE / LIN	SFR-CG algorithm NOI / FGE / LIN
1	**10000**	**37 / 65 / 25**	**32 / 60 / 24**
2	**10000**	**40 / 91 / 34**	**37 / 83 / 31**
3	**10000**	**44 / 95 / 39**	**39 / 85 / 30**
4	**10000**	**25 / 65 / 16**	**23 / 59 / 15**

5	**10000**	**23 / 37 / 13**	**23 / 37 / 13**
6	**10000**	**104 / 199 / 93**	**61 / 116 / 53**
7	**10000**	**72 / 113 / 38**	**72 / 113 / 38**
8	**10000**	**77 / 174 / 73**	**70 / 70 / 68**
9	**10000**	**657 / 1158 / 500**	**579 / 1055 / 475**
10	**10000**	**86 / 182 / 84**	**85 / 181 / 83**
11	**10000**	**49 / 91 / 41**	**41 / 75 / 33**
12	**10000**	**96 / 145 / 46**	**88 / 153 / 62**
13	**10000**	**112 / 2589 / 102**	**68 / 1380 / 64**
14	**10000**	**16 / 40 / 15**	**17 / 40 / 16**
15	**10000**	**221 / 6472 / 215**	**297 / 7498 / 290**
Total		**1665/ 11516 / 1334**	**1533 / 11105 / 1294**

Total Time : 7423 Sc 6157 Sc

References

[1] Dai Y.and Yuan, Y. ' A non – Linear conjugate gradient method with a strong global convergence property ' SIAM J. Optim, 10. 1999.

[2] Dai Y. and Yuan Y. 'Non – Linear Conjugate Gradient Methods'.Shanghai Science and Technology press 2000.

[3] Dai,Y.and Yuan, Y. ' A class of globally convergence conjugate Gradient methods'. Sci. Chaina Ser A, 46, 2003.

[4] Fletcher, R. and Reeves, C. ' Function minimization by conjugate Gradient', Computer J.(7). 1964.

[5] Hestenes, M. and Stiefel, E. ' Methods of Conjugate Gradients for solving linear systems', J. Res. Nat. Bur. Standards, 49, 1952.

[6] Kincella J. 'Course Notes For MS4327 Optimization'

http//jkcray.Maths.ul.ie/ms4327/slides.pdf.

[7] Nocedal. J. and Wright S. 'Numerical Optimization' Spinger, Berlin, Heidelberg, New York 1999.

[8] Powell M. ' Restart procedure for the conjugate gradient methods' Mathematics Programming, (12), 1977.

[9] Wei Z., Li G. and Qi, L. 'New Quasi – Neuton Methods for Unconstrained Optimization Problems'. Applied Mathematics and Computation.Vol. 175, No. 1, 2006.

[10] Yuan, G. and Lu. X. 'A New Line search Method with Trust Region for Unconstrained Optimization'. Communications on Applied Non-Linear Analysis, Vol. 15, No. 1, 2008

[11] Yuan, G. and Lu. X. and Wei, Z. 'New Two – Point Stepsize Gradient Methods for Solving Unconstrained Optimization Problems'. Natural Science Journal of Xiangtan University, Vol. 29, No. 1, 2007

[12] Yuan, G. and Lu. S. and Wei, Z. 'A Line Search Algorithm for Unconstrained Optimization' . J – Software Engineering and Applications, 2010 doi: 10.4236/jsea, published on line at http:/www.sciRP.org/journal/jsea Scientific Research.

[13] Yuan Y. and Sun. W. 'Theory and Methods of Optimization'

Science Press of China, Beijing, 1999.

[14] Yuan Y. and Sun. W. 'Optimization Theory and Methods', Science Press, Beijing, 1997.

[15] Yuan Y. and Stoer, J. 'A subspace study on conjugate gradient Algorithms' Z. Angew Math. Mech., 75., 1995

[16] Zen. S. and Gou, J 'New Algorithm of Non – Linear Conjugate Gradient Method with strong convergence' Computational and Applied Mathematics, Vol. 27. No. 1

[17] More J. Garbow B. and Hillstrome K. ' Testing Unconstrained Optimization Softwore ' J.ACM Trans Math Softwore , (7) : 17-41. 1981

Appendix

1. Extended Trigonometric Function

$$f(x)=\sum_{i=1}^{n}\left(\left(n-\sum_{j=1}^{n}\cos x_j\right)+i(1-\cos x_i)-\sin x_i\right)^2,$$

$$x_0=[0.2\ ,\ 0.2\ ,..................,\ 0.2].$$

2. Extended Rosenbrock Function

$$f(x)=\sum_{i=1}^{n/2}c(x_{2i}-x_{2i-1}^2)^2+(1-x_{2i-1})^2,$$

$$x_0=[-1.2\ ,\ 1\ ,..........,\ -1.2\ ,\ 1].c=100$$

3. Extended White & Host Function

$$f(x)=\sum_{i=1}^{n/2}c(x_{2i}-x_{2i-1}^2\ )^2+(1-x_{2i-1})^2,$$

$$x_0=[-1.2,1,.........,-1.2,1].c=100$$

4. Extended Penalty Function

$$f(x)=\sum_{i=1}^{n-1}(x_i-1)^2+(\sum_{j=1}^{n}x_j^2-0.25)^2,$$
$$x_0=[1,2,..............,n]$$

5. Extended Himmelblau Function

$$f(x)=\sum_{i=1}^{n/2}(x_{2i-1}^2+x_{2i}-11)^2+(x_{2i-1}+x_{2i}^2-7)^2,$$
$$x_0=[1,1,.........,1].$$

6. Generalized PSC1 Function

$$f(x)=\sum_{i=1}^{n-1}(x_i^2+x_{i+1}^2+x_i\,x_{i+1})^2+\sin^2(x_i)+\cos^2(x_i),$$
$$x_0=[3,0.1,............,3,0.1]$$

7. Extended PSC1 Function

$$f(x)=\sum_{i=1}^{n/2}(x_{2i-1}^2+x_{2i}^2+x_{2i-1}x_{2i})^2+\sin^2(x_{2i-1})+\cos^2(x_{2i}),$$
$$x_0=[3,0.1,.............,3,0.1].$$

8. Extended Powell Function

$$f(x)=\sum_{i=1}^{n/4}(x_{4i-3}+10x_{4i-1})^2+5(x_{4i-1}-x_{4i})^2+(x_{4i-2}-2x_{4i-1})^4+10(x_{4i-3}-x_{4i})^4,$$
$$x_0=[3,-1,0,1,.............,3,-1,0.1]$$

9. Full Hessian FH2 Function

$$f(x)=(x_1-5)^2+\sum_{i=2}^{n}(x_1+x_2+.......+x_i-1)^2,$$
$$x_0=[0.01,0.01,...................,0.01].$$

10. Extended Maratos function (c=100)

$$f(x)=\sum_{i=1}^{n/2}x_{2i-1}+c(x_{2i-1}^2+x_{2i}^2-1)^2,$$
$$x_0=[1.1,0.1,................,1.1,0.1].c=100$$

11. NONDQUAR Function (CUTE)

$$f(x) = (x_1 - x_2)^2 + \sum_{i=1}^{n-2} (x_i + x_{i+1} x_n)^4 + (x_{n-1} + x_n)^2,$$
$$x_0 = [1.,-1.,.............................,1.,-1.,].$$

12. DQDRTIC function (CUTE)

$$f(x) = \sum_{i=1}^{n-2} (x_i^2 + c x_{i+1}^2 + d x_{i+2}^2),$$
$$c = 100., d = 100.$$
$$x_0 = [3.,3.,...................,3.].$$

13. DIXMAANA-DIXMAANL Function

$$f(x) = 1 + \sum_{i=1}^{n} \alpha x_i^2 (\frac{i}{n})^{k1} + \sum_{i=1}^{n-1} \beta x_i^2 (x_{i+1} + x_{i+1}^2)^2 (\frac{i}{n})^{k2} + \sum_{i=1}^{2m} \gamma x_i^2 x_{i+m}^4 (\frac{i}{n})^{k3} + \sum_{i=1}^{m} \delta x_i x_{i+2m} (\frac{i}{n})^{k4}$$
$$m = n/3$$

14. Almost Pertubed Quadratic Function

$$f(x) = \sum_{i=1}^{n} i x_i^2 + \frac{1}{100} (x_1 + x_n)^2,$$
$$x_0 = [0.5, 0.5, ..., 0.5]$$

15. Staircase 2 Function

$$f(x) = \sum_{i=1}^{n} \left[(\sum_{j=1}^{i} x_j) - i \right]^2$$
$$x_0 = [0,0,.................,0].$$