A New Three-Term Preconditioned Gradient Memory Algorithm for Nonlinear Optimization Problems

Abbas Y. AL-Bayati and Ivan Subhi Latif

1Department of Mathematics, College of Computers Sciences and Mathematics, University of Mosul, Mosul-Iraq
2Department of Mathematics, College of Scientific Education, University of Salahaddin, Erbil, Iraq

Abstract: In the present study, we proposed a three-term of preconditioned gradient memory algorithms to solve a nonlinear optimization problem. The new algorithm subsumes some other families of nonlinear preconditioned gradient memory algorithms as its subfamilies with Powell’s Restart Criterion and inexact Armijo line searches. Numerical experiments on twenty one well-known test functions with various dimensions generally encouraged and showed that the new algorithm was more stable and efficient in comparison with the standard three-term CG-algorithm.

Key words: Unconstrained optimization, preconditioned conjugate gradient, self-scaling VM-updates, inexact line searches

INTRODUCTION

We consider the unconstrained optimization problem

\[ \min_{x \in \mathbb{R}^n} f(x) \quad x \in \mathbb{R}^n \]

(1)

where \( f : \mathbb{R}^n \rightarrow \mathbb{R}^1 \) is a continuously differentiable function in \( \mathbb{R}^n \) and \( \mathbb{R}^n \) is the n-dimensional euclidean space. Conjugate gradient methods were very useful for solving (1). They were of the form

\[ x_{k+1} = x_k + \alpha_k d_k \]

(2)

and

\[ d_k = -g_k + \beta_k d_{k-1} + \alpha_k d_{k-2} \]

(3)

where \( g_k \) denoted by \( \nabla f(x_k) \), \( \alpha_k \) is a step-length obtained by a line search, and \( \beta_k \) is a scalar. The memory gradient algorithm for problem (1) was first presented in Cragg and Levy [4] with the ordinary gradient method. This method has the advantage of high speed convergence since it produces a sequence of quadratic convergent points.

A new three-term memory gradient method for problem (1) whose search directions are defined by

\[ d_k = -g_k + \beta_k d_{k-1} + \alpha_k d_{k-2} \]

(4)

and

\[ x_{k+1} = x_k + \lambda_k d_k \]

(5)

where \( \beta_k \) and \( \alpha_k \) are parameters and \( \lambda_k \) is a step-size considered in Sun [8] and obtained by means of a one dimension search. The self-scaling Variable Metric (VM) algorithms were introduced, showing significant improvement in efficiency over earlier methods. The search direction

\[ d_k = -H_k g_k \]

(6)

\( H_k \) is an approximation to the inverse Hessian \( G^{-1} \).

For a given \( H_1 \), the matrix \( H_k \) was updated to \( H_{k+1} \) by a formula from the class of self-scaling updates satisfying the following QN-like condition given in Cohen [3]

\[ H_{k+1} y_k = \rho_k V_k \]

(7)

where

\[ V_k = x_{k+1} - x_k \]

(8)

There were infinite numbers of possible updates which satisfy the QN-condition. The class of these updates were written as (See Gill and Murray[5])

\[ H_{k+1} = \left( H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \sigma_k W_k W_k^T \right) \sigma_k + v_k y_k^T / y_k y_k \]

(9)
\[ \theta_k, \sigma_k \] parameters and

\[ W_k = \left( Y_k^T H_k Y_k \right)^{-1} \left[ V_k^T / V_k^T Y_k - H_k Y_k / Y_k^T H_k Y_k \right] \]  \hspace{1cm} (10)

The parameter \( \theta_k \) was chosen such that \( \theta_k \in [0,1] \) different choice of \( \theta_k \) defines different updates and different search conjugate directions (See Al-Assady and Al-Bayati[1]), the Davidson-Fletcher-Powell (DFP) update was defined as (9) where \( \sigma_k = 1 \) with \( \theta_k = 0 \) while Broynen-Fletcher-Goldfarb-Shanno (BFGS) was defined as an updated corresponds to \( \theta_k = 1 \). Oren found that a proper scaling of the objective function improve the performance of algorithms that use Broyden family of update where \( \sigma_k = V_k^T Y_k / Y_k^T H_k Y_k \). This choice for the scalar parameter \( \sigma_k \) was made primarily because in this case \( \sigma_k \) requires the quotient of two quantities which were already computed in the updating formula. For more details see Yabe and Takano[9].

In this study, we considered a new three-term PCG algorithm for problem (1) whose search directions were defined by

\[ d_k = -H_k s_k \quad \text{for} \quad k = 1 \]

\[ d_k = -g_k + \beta_k g_{k-1} + \alpha_k d_{k-2} \quad \text{for} \quad k \geq 2 \]  \hspace{1cm} (11)

by using a new line-search parameter and a positive definite matrix \( H_k \).

The Three-Term Memory Gradient Algorithm (TMG):

Consider the three term memory gradient method (4) and (5). Conditions are given on \( \beta_k \) and \( \alpha_k \) to ensure that \( d_k \) is a sufficient descent direction at the point \( x_k \). Now let \( S_k = -g_k + \beta_k g_{k-1} \) and assume that

\[ g_k^T s_k > \| \beta_k g_{k-1} \| \]
\[ g_k^T S_k \geq (1 + \Delta_1) \| \beta_k \| \| S_k \| \] \hspace{1cm} (12)

and

\[ g_k^T S_k \geq \| \alpha_k g_{k-1} \| \]
\[ g_k^T d_k \geq (1 + \Delta_2) \| \alpha_k \| \| d_k \| ] \hspace{1cm} (13)

where \( \Delta_1 > 0 \) and \( \Delta_2 > 0 \) are constants it follows from (12) that

\[ g_k^T s_k - \beta_k g_k^T d_{k-1} \geq (1 + \Delta_1) \| \beta_k \| \| g_k \| \] \hspace{1cm} (14)

Theorem:

If \( x_k \) is not a stationary point for problem (1) then:

\[ \| d_k \| \leq (1 + \frac{1}{\Delta_1} + \frac{1}{\Delta_2}) \| g_k \| \] \hspace{1cm} (15)

Moreover, if \( d_k \) is descent then:

\[ g_k^T d_k \leq \frac{1 + \Delta_1}{2 + \Delta_2} + \frac{1 + \Delta_2}{2 + \Delta_2} \| g_k \|^2 \] \hspace{1cm} (16)

for the prove of this theorem see [6].

Outline of the Three-Term Memory Gradient Algorithm (TMG):

Step1: let \( x_0 \in \mathbb{R}^n \) be initial point, \( \Delta_1 > 0, \Delta_2 > 0 \), compute \( g_0 \); if \( g_0 = 0 \) and \( x_0 \) is a stationary point of (1) stop; else set \( d_0 = -g_0 \), let \( k = 1 \) and go to step2.

Step2: let \( x_{k+1} = x_k + \lambda_k d_k \); the step size \( \lambda_k \) is defined in the following way \( \lambda_k = \min \{ \alpha > 0 : g_k^T d_k = \mu g_k^T d_k \text{where} \mu \in [0,1] \} \)

Step3: compute \( g_{k+1} \); if \( \| g_{k+1} \| = 0 \) and \( x_{k+1} \) is a stationary point of (1) stop; else let \( k = k + 1 \) go to step4.

Step4: set \( d_k = -g_k + \beta_k g_{k-1} + \alpha_k d_{k-2} \) where

\[ \beta_k = \begin{bmatrix} -\beta_k(\Delta_1), \beta_k(\Delta_2) \end{bmatrix} \]
\[ \alpha_k = \begin{bmatrix} -\alpha_k(\Delta_1, \Delta_2), \alpha_k(\Delta_1, \Delta_2) \end{bmatrix} \]

where

\[ \alpha_k(\Delta_1, \Delta_2) = \frac{1 + \Delta_1}{2 + \Delta_2} \frac{1}{(1 + \Delta_2) + \cos \theta_k} \| g_k \| \] \hspace{1cm} (17)

\[ \alpha_k(\Delta_1, \Delta_2) = \frac{1 + \Delta_1}{2 + \Delta_2} \frac{1}{(1 + \Delta_2) - \cos \theta_k} \| g_k \| \] \hspace{1cm} (18)

\( \theta_k \) angle between \( g_k, d_{k-2} \) and
A NEW THREE-TERM PRECONDITIONED GRADIENT MEMORY ALGORITHM

In this section we introduced a line search rule to find the best step-size parameter along the search direction at each iteration. We studied the convergent analysis of the modified Armijo step-size rules fully described in Armijo [2] given in step 3 of the following new algorithm.

Outline of the New Three-Term Preconditioned Gradient Memory algorithm (NEW):

Step 1: Let \( x_0 \in \mathbb{R}^n \) be initial point, compute \( g_0 \); if \( g_0 = 0 \) and \( x_0 \) is a stationary point of (1) stop; else let \( H_1 \) is any positive definite matrix usually \( IH_1 = I \) and \( \varepsilon \) is a small positive value, let \( k = 1 \) set \( d_1 = -H_1 g_1 \).

Step 2: if \( \| g_k \| < \varepsilon \) then stop! Else go to step 3

Step 3: \( x_{k+1} = x_k + \alpha_k d_k \) the step size \( \alpha_k \) is chosen by the modified Armijo line search rule, namely: for given \( q > 1 \), \( \mu_i \in (0, 1), \lambda_i = q^{-i} \) and \( r^k \) is the smallest non negative integer such that

\[
 f(x_k + q^r d_k) \leq f(x_k) + \mu_r q^r g_k^T d_k
\]

Step 4: compute \( g_{k+1} \); if \( \| g_{k+1} \| = 0 \) and \( x_{k+1} \) is a stationary point of (1) stop; else let \( k = k + 1 \), go to step 5.

Step 5:

\[
d_k = \begin{cases} -H_1 g_k, & \text{for } k = 1 \\ -g_{k-2} + \beta_k H_{k-1} d_{k-1} + \alpha_k H_{k-2} d_{k-2}, & \text{for } k \geq 2 \end{cases}
\]

and \( H_{k+1} \) is updated by

\[
 H_{k+1} = \begin{cases} H_k - \frac{H_k V_k^T H_k}{V_k^T H_k V_k} + \theta_k W_k^T + V_k V_k^T / V_k V_k \text{ if } V_k^T H_k V_k > 0.5 \\ H_k, \text{ otherwise} \end{cases}
\]

where \( \theta_k = \| V_k^T H_k V_k \| V_k^T H_k V_k \).

Step 7: If the available storage is exceeded, then employ a restart option either with \( k = n \) or \( g_k^T g_k > g_{k+1}^T g_{k+1} \).

Step 8: Set \( k = k + 1 \) and go to step 2

The convergence analysis of the new proposed algorithm:

Consider the new three-term Preconditioned gradient memory defined in (11). Let \( S_k = -d_k g_k + \beta_k H_{k-1} d_{k-1} \) and order to ensure that \( d_k \) is a sufficient descent direction at the point \( x_k \), we assumed that for \( \Delta_1 = 0.67 \) and \( \Delta_2 = 5 \) then:

\[
 \begin{cases} g_k^T g_k > \beta_k g_k^T H_{k-1} d_{k-1} \\ \| g_k^T S_k \| \geq 1.067 \beta_k \| g_k \| \| H_{k-1} d_{k-1} \|
\end{cases}
\]

and

\[
 \begin{cases} \| g_k^T S_k \| \geq \alpha_k g_k^T H_{k-2} d_{k-2} \\ \| g_k^T d_{k-1} \| \geq 4 \alpha_k \| g_k \| \| d_{k-2} \|
\end{cases}
\]

from (22) we proposed the following property:

Property 1:

\[
g_k^T g_k - \beta_k g_k^T d_{k-1} \geq 1.067 \beta_k \| g_k \| \| H_{k-1} d_{k-1} \|
\]

Proof:

Case 1. To ensure that \( \beta_k > 0 \) let

\[
 \beta_k \leq \frac{g_k^T g_k}{1.067 \| g_k \| \| H_{k-1} d_{k-1} \| + g_k^T H_{k-1} d_{k-1}}
\]

\[
 \frac{1}{1.067 + \cos \theta_k} \| H_{k-1} d_{k-1} \|
\]

where \( \theta_k \) is the angle between \( g_k \) and \( H_{k-1} d_{k-1} \).

Case 2. To ensure that \( \beta_k < 0 \) let
\[
\beta_k \geq \frac{\|g_k\|}{1.067\|H_{k-1}d_{k-1}\| - g_1^TH_{k-1}d_{k-1}} \tag{26}
\]

where \(\theta_{k1}\) is the angle between \(g_k\) and \(H_{k-1}d_{k-1}\).

Thus a new choice for \(\beta_k\) will be given by

\[
\beta_k = \frac{1}{1.067 - \cos \theta_{k1}} \frac{\|g_k\|}{\|H_{k-1}d_{k-1}\|} \tag{27}
\]

\[
\beta_{k1} = \frac{1}{1.067 + \cos \theta_{k1}} \frac{\|g_k\|}{\|H_{k-1}d_{k-1}\|} \tag{28}
\]

\[
\beta_{k2} = \frac{1}{1.067 - \cos \theta_{k1}} \frac{\|g_k\|}{\|H_{k-1}d_{k-1}\|} \tag{29}
\]

where \(\theta_{k1}\) is the angle between \(g_k\) and \(H_{k-1}d_{k-1}\).

from (23) we proposed the following property:

**Property 2:**

\[-g_k^TS_k - \alpha_k g_k^TH_{k-2}d_{k-2} \geq 4\alpha_k \frac{\|g_k\|}{\|H_{k-2}d_{k-2}\|} \tag{30}\]

**Proof:**

**Case 1.** To ensure that \(\alpha_k > 0\) let

\[
\alpha_k \geq \frac{-g_k^TS_k}{4\|g_k\|\|H_{k-2}d_{k-2}\| + g_k^Td_{k-2}} \tag{31}
\]

It follows from (23) that

\[g_k^TS_k \leq -\frac{1.067}{4} \|g_k\|^2\]

which implies that

\[
\alpha_k \geq \frac{1.067}{5} \frac{1}{4 + \cos \theta_{k2}} \frac{\|g_k\|^2}{\|H_{k-2}d_{k-2}\|} \tag{32}
\]

\[
\alpha_{k1} = \frac{1.067}{5} \frac{1}{4 + \cos \theta_{k2}} \frac{\|g_k\|^2}{\|H_{k-2}d_{k-2}\|} \tag{33}
\]

\[
\alpha_{k2} = \frac{1.067}{5} \frac{1}{4 + \cos \theta_{k2}} \frac{\|g_k\|^2}{\|H_{k-2}d_{k-2}\|} \tag{34}
\]

where \(\theta_{k2}\) is the angle between \(g_k\) and \(H_{k-2}d_{k-2}\).

The descent property of the new proposed algorithm:

If \(x_k\) is not a stationary point for problem (1) then the search directions \(d_k\) of the new proposed algorithm are descent directions i.e.

\[g_k^Td_k \leq \frac{1.067}{5} \|g_k\|^2\]

**Proof:** For \(k = 1\), it is clear that \(d_1 = -H_1g_1\) where \(H_1 = I\) identity matrix is a descent direction since for \(k \geq 2\) it follows from (22)

\[g_k^TS_k = -g_k^TH_kg_k + \beta_k g_k^TH_{k-1}d_{k-1} \]

\[= -\|g_kH_k\|^2 + \beta_k g_k^TH_{k-1}d_{k-1} \]

\[= -\|g_kH_k\|^2 + \frac{1}{1.067} g_k^TS_k \]
The above inequality \( \| g_k S_k \| = -g_k^T S_k \) imply that \( g_k^T d_k \leq \frac{1.067}{5} \| g_k \| \)

RESULTS AND DISCUSSION

In this section we report some numerical results obtained by newly-written Fortran procedure with double precision.

Table 1: Comparison between the standard Three Term Memory Gradient (TMG) algorithm and New proposed algorithms using different values of 5≤N≤1000 for the 1st group of test functions

| N. Test function | TMG NOF(NOI) | New NOF(NOI) |
|------------------|--------------|--------------|
| 5 10 100 1000    | 5 10 100 1000|
| GEN-1 Trid1      | 117 169 374 365| 42 74 100 100 |
| Shanno           | 990 34 24 272| 25 35 21 25 |
| QF1              | 54 125 1187 1765| 16 26 114 532 |
| Rosen            | 1583 1655 1835 1015| 124 120 118 124 |
| NON-Digonal      | 306 389 367 385| 49 97 64 88 |
| TPQ              | 125 211 1593 1839| 20 30 144 656 |
| 6 7 8 GEN-1 GQ2  | 28 26 26 26| 16 18 24 30 |
| 24 24 24 24      | 12 12 12 12| 11 12 15 18 |
| 9 10 APQ         | 195 203 223 239| 28 28 30 30 |
| 3 4 GEN-Powell   | 68 68 68 68| 39 39 39 40 |
| Tridia           | 1583 1655 1835 1015| 124 120 118 124 |
| 5 7 Ex-wth host  | 68 68 68 68| 39 39 39 40 |
| 8 GEN-Wood       | 444 426 444 430| 244 231 246 234 |
| General total of 7 functions | 5494 4970 13569 9160 | 346 459 480 1456 |

In comparison of algorithms the function evaluation is normally assumed to be the most costly factor in each iteration and the number of iterations. The actual convergence criterion employed was \( \| g_k \| < 1 \times 10^{-6} \) for the two algorithms, twenty one well-known test functions (Appendices 1 and 2) and with dimensionality ranging (5-1000) are employed in the comparison. We solve each of these test function by the:

- Three Term Memory Gradient algorithm (TMG)
- The New proposed (New) algorithm

All the numerical results are summarized in Table 1, 2 and 3. They present the Number of Iterations(NOI) versus the Number of Function Evaluations (NOF) while Table 3 give the percentage performance of the new algorithm based on both (NOI) and (NOF) against the original (TMG) algorithm.

The important thing is that the new algorithm solves each particular problem measured by (NOI) and (NOF) respectively, while the other algorithm may fail in some cases. Moreover, the new proposed algorithm always performs more stably and efficiently.

Table 2: Comparison between the standard Three Term Memory Gradient (TMG) algorithm and New proposed algorithms using different value of 5≤N≤1000 for the 2nd group of test function

| N. Test function | TMG NOF(NOI) | New NOF(NOI) |
|------------------|--------------|--------------|
| 5 10 100 1000    | 5 10 100 1000|
| Biggsb F F F F   | 8 24 140 1234|
| 2 GEN-PowellI F F F F| 94 94 102 106 |
| 3 GEN-Cubic F F F F| 68 68 68 70 |
| 4 GEN-QDP F F F F| 18 18 478 128 |
| 5 Fred F F F F   | 12 13 248 70 |
| 6 Sinquad F F F F| 66 58 220 560 |
| 7 Ex-wth host F F F F| 68 68 68 70 |
| 8 GEN-Wood F F F F| 444 426 444 430 |
| General total of 7 functions | 244 231 246 234 |

Table 3: Percentage performance of the standard Three Term Memory Gradient (TMG) algorithm against and New algorithm for 100% in both NOI and NOF

| N Costs | New |
|---------|-----|
| 5 NOF NOI| 95.107 93.70 |
| 10 NOF NOI| 92.63 90.76 |
| 100 NOF NOI| 91.70 96.46 |
| 1000 NOF NOI| 79.47 84.11 |

Namely there are about (7-16)% improvements of NOI for all dimensions Also there are (5-21)% improvements of NOF for all test functions.

CONCLUSIONS

In this study, we have three parameter family of preconditioned gradient algorithm suitable to solve nonlinear unconstrained optimization problems. The directions \( d_k \) generated by the algorithm satisfy both the sufficient descent and lie search condition, with an
inexact line search under standard Wolfe line search condition. We have proved the global convergence of the new algorithm and examines their computational performances.

Computational experience shows that the new proposed algorithm performs better than the standard three parameter family of preconditioned gradient memory method.

**APPENDIX**

**Appendix 1:** All the test functions used in Table (1) are from[2]:

**Generalized tridiagonal-1 function:**
\[ f(x) = \sum_{i=1}^{n} (x_{2i-1} + x_{2i-3})^2 + (x_{2i-1} - x_{2i} + 1)^4, \]
\[ x_0 = [2.2, 2, 2, \cdots, 2, 2]. \]

**Non-diagonal (Shanno-78) Function (Cute):**
\[ f(x) = (x_{i} - 1)^2 + \sum_{i=1}^{n} 100(x_{i} - x_{i}^2)^2, \]
\[ x_0 = [-1.1, -1, \cdots, -1, 1]. \]

**Quadratic QF1 function:**
\[ f(x) = \sum_{i=1}^{n} x_{i}^2 - x_{n}, \]
\[ x_0 = [1, 1, \cdots, 1, 1]. \]

**Generalized rosen brock banana function:**
\[ f(x) = \sum_{i=1}^{n} 100(x_{2i} - x_{2i-1})^2 + (1 - x_{2i-1})^2, \]
\[ x_0 = [-1.2, 1, \cdots, -1, 1]. \]

**Generalized Non diagonal function:**
\[ f(x) = \sum_{i=1}^{n} 100(x_{i} - x_{i}^3)^2 + (1 - x_{i})^2, \]
\[ x_0 = [-1, \cdots, -1]. \]

**Tri-diagonal perturbed quadratic function:**
\[ f(x) = x_{i}^2 + \sum_{i=1}^{n} x_{i}^2 + (x_{i-1} + x_{i} + x_{i+1})^2, \]
\[ x_0 = [0.5, 0.5, \cdots, 0.5, 0.5]. \]

**Generalized quadratic function GQ2:**
\[ f(x) = (x_{i}^2 - 1)^2 + \sum_{i=2}^{n} (x_{i}^2 - x_{i-1} - 2)^2, \]
\[ x_0 = [1, 1, \cdots, 1, 1]. \]

**Generalized Powell3 function:**
\[ f(x) = \sum_{i=1}^{n} \left[ 3 - \frac{1}{x_{i} x_{i+1} x_{i+2}} \right] \sin \left( \frac{x_{i} x_{i+1} x_{i+2}}{2} \right) - \exp \left[ - \left( \frac{x_{i} x_{i+1} x_{i+2}}{2} - 2 \right)^2 \right] \]
\[ x_0 = [0, 1, 2, \cdots, 0, 1, 2]. \]

**Tri-diagonal function:**
\[ f(x) = \gamma (\delta x_i - 1)^2 + \sum_{i=1}^{n} i (\alpha x_i - \beta x_{i+1})^2, \]
\[ x_0 = [1, 1, \cdots, 1], \alpha = 1, \beta = 1, \gamma = 1, \delta = 1. \]

**Almost perturbed quadratic function:**
\[ f(x) = \sum_{i=1}^{n} x_{i}^2 + \frac{1}{100} (x_i + x_{i+1})^2, \]
\[ x_0 = [0.5, 0.5, \cdots, 0.5, 0.5]. \]

**General helical function:**
\[ f(x) = \sum_{i=1}^{n} \left( 100 x_{i} - 10^* H_{i} \right)^2 + 100 (R_{i} - 1)^2 + x_{i}^2, \]
\[ \tan^{-1} \frac{x_{i}^{2.5}}{2.5}, \]
where \( R_{i} = \sqrt{(x_{i-1} + x_{i+1})}, H_{i} = \frac{x_{i-2}}{2.51} \]
\[ x_0 = [-1, 0, \cdots, -1, 0]. \]

**D-quadratic Function (CUTE):**
\[ f(x) = \sum_{i=1}^{n} x_{i}^2 + \sum_{i=1}^{n} \left( x_{i}^2 + x_{i+1}^2 + d x_{i+1}^2 \right), \]
\[ x_0 = [3, 3, \cdots, 3, 3], c = 100, d = 100. \]

**Generalized Beale Function:**
\[ f(x) = 1.5 x_{1} (1 - x_{1})^2 + 2.25 x_{2} (1 - x_{2})^2 + 2.625 x_{3} (1 - x_{3})^2, \]
\[ x_0 = [1, 1, \cdots, 1]. \]

**Appendix 2:** All the test functions used in Table (2) are from[3]:

**Biggsb1 Function (CUTE):**
\[ f(x) = (x_i - 1)^2 + \sum_{i=1}^{n} (x_{i-1} - x_i)^2 + (1 - x_{i})^2, \]
\[ x_0 = [1, 1, \cdots, 1, 1]. \]

**Generalized Powell function:**
\[ f(x) = \sum_{i=1}^{n} \left[ 3 - \frac{1}{x_{i} x_{i+1} x_{i+2}} \right] \sin \left( \frac{x_{i} x_{i+1} x_{i+2}}{2} \right) - \exp \left[ - \left( \frac{x_{i} x_{i+1} x_{i+2}}{2} - 2 \right)^2 \right] \]
\[ x_0 = [0, 1, 2, \cdots, 0, 1, 2]. \]

**Generalized Cubic function:**
$f(x) = \sum_{i=1}^{n/2} [100(x_{2i} - x_{2i-1}^3) + (1 - x_{2i-1})^2],$

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

**Quadratic diagonal perturbed function:**

$$f(x) = \left(\sum_{i=1}^{n} x_i^2\right)^{\frac{3}{2}} + \sum_{i=1}^{n} x_i^2,$$

$x_0 = [0.5,0.5,...,0.5,0.5].$

**Extended fred function:**

$$f(x) = \sum_{i=1}^{n/2} (-13 + x_{2i-1} + (5 - x_{2i}) + (x_{2i} - 2)(x_{2i}))^2,$$

$x_0 = (1,2,...,n)^T.$

**Sinquad Function (CUTE):**

$$f(x) = (x_1 - 1)^4 + \sum_{i=1}^{n/2} (\sin(x_i - x_{i+1}) - x_i^2 + x_i^2)^2 + (x_i^2 - x_{i+1}^2)^2,$$

$x_0 = [0.1,0.1,...,0.1].$

**Extended white and holst function:**

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

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

**Generalized wood function:**

$$f(x) = \sum_{i=1}^{n/2} [100(x_{4i-2} - x_{4i-1}^3) + (1 - x_{4i-1})^2] + 90(x_{4i} - x_{4i-1}^2)^2,$$

$x_0 = [-3.,-1.,-3.,-1.,...,-3.,-1.,-3.,-1.].$

**REFERENCES**

1. Al-Assady, N.H., Al-Bayati, A.Y., 1994. Minimization of extended quadratic functions with inexact line Searches. J. Optim. Theo. Appl. 82, 139–147, doi> 10.1007/BF02191784.

2. Armijo, L., 1966. Minimization of function having Lipschitz continuous first partial derivatives, pacific. J. Math., 16, 1-13. http://projecteuclid.org/euclid.pjm/1102995080

3. Cohen, A., 1981. Step-size analysis for descent methods, IEEE Xplore, 12, 417-421. doi> 10.1109/CDC.1979.1126079.

4. Cragg, E.E., Levy, A.V., 1969. Study on a super memory gradient method for the minimization of functions, J. Optim. Theo. Appl.4, 191–205. doi> 10.1007/BF00930579.

5. Gill, P.E., Murray, W., 1975. Conjugate gradient methods for large scale nonlinear optimization, Technical Report SOL 79-15, Stanford University, http://stinet.dtic.mil/stinet/jsp/advanced-tr.jsp

6. Hager, W., Zhang, H., 2005. A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J. Optim. 16, 170–192. http://portal.acm.org/citation.cfm?id=1085581&dl= &coll=

7. More, J.J, Garbow, B.S., Hillstrom, K.E., 1981. Testing unconstrained optimization software. ACM Transactions on Mathematical Software 7, 17–41, http://portal.acm.org/citation.cfm?doid=355934.355936

8. Sun, Q., 2004. Global Convergence of a Class of New Three-term Memory Gradient Methods with Curry-Altman and Armijo Step-size Rules, Soochow Journal of Mathematics 30, 55–66. http://143.248.27.21/mathnet/thesis_content.php?no=367448

9. Yabe, H., Takano, M., 2004. Global convergence properties of nonlinear conjugate gradient methods with modified secant condition. Computational Optim. Appl. 28,203–225. doi>10.1023/B:COAP.0000026885. 81997. 88