In Accurate CG-Algorithm for Unconstrained Optimization Problems

Nidhal Al-Assady  Maysoon M. Aziz  Ban Ahmed Mitras

dr.banah.mitras@gmail.com

College of Computer Sciences and Mathematics
University of Mosul, Iraq

Received on: 13/05/2002  Accepted on: 01/09/2002

Abstract

An algorithm for unconstrained minimization is proposed which is invariant to a non-linear scaling of a strictly convex quadratic function and which generates mutually conjugate directions for extended quadratic function. It is derived for inexact line searches and is designed for general use, it compares favorably numerical tests [over eight test functions and dimensionally up to (2-100) with the H/S, DX, F/R, P/R, and A/B algorithms on which this new algorithm is based.

Keyword: CG algorithm, in accurate CG method.

خوارزمية CG المقيسة لمسائل الأمثلية غير المقيدة

نضال الاسدي  ميسون مال الله عزيز  بان احمد متراس
كلية علوم الحاسوب والرياضيات، جامعة الموصل

تاريخ الاستلام: 13/05/2002  تاريخ القبول: 01/09/2002

الملخص

يتم تقديم خوارزمية للإمثالية غير المقيدة في هذا البحث التي اختصت بالتقييس غير الخطي للدوال التربيعية المحدبة بدقة وعممت لاتجاهات مترافقة موزعة
1. Introduction:

The searches related with Conjugate Gradient method begin to apply inexact line searches; i.e not to iterate until the “line minima” is found to some predetermined (small) tolerance in order to reduce the number of function evaluations (NOF).

In order to show improvement of the local rate of convergence and the efficiency of the traditional CG-method several well-known methods are discussed later, namely Dixon (1975) and Nazareth (1977).

The type of these algorithms has quadratic termination property by using an error vector even if inexact searches are used, while Sloboda (1982) presented an algorithm which retains the quadratic termination property without using an error vectors.

In this paper we develop a new general way of the CG algorithm with inexact line searches. This new algorithm is similar to that derived by Dixon (1975) CG method with inexact line searches but does not require the correction term as extra vector storage and different formula is obtained.

2. General Background:

In this section we shall present a brief description of the Dixon Sloboda, Nazareth and Nocedal algorithms. We shall then discuss properties of these algorithms.
In accurate CG-Algorithm ..

2.1 The Dixon’s Method (1975):

In Dixon’s method the idea is to determine directions parallel to the CG directions (for a quadratic) without line searches, and so he develops a conjugate set. Explicitly, the search directions \( d_k \), \( k = 1, 2, \ldots \), are given by

\[
d_{k+1} = -g_{k+1} + \beta_k d_k,
\]

where \( d_1 = -g_1 \),

and \( g_k^-, \beta_k \) are estimation values defined below. The new iterate is defined as:

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

where \( \lambda_k \) is chosen simply to satisfy conditions defined in the line search subprogram. The gradient \( g_k \) is estimate of the gradients at the points \( x_k \) that we would have reached, if we had performed exact line searches along the \( d_k \). For a quadratic, these gradients can be evaluated exactly, and defined from:

\[
\begin{align*}
g_{k+1} &= g_k + (1 - \frac{g_{k+1}^T d_k}{g_k^T d_k}) y_k, \quad k \geq 1, \\
g_{k+1} &= g_k + \frac{g_{k+1}^T d_k}{y_k^T d_k} y_k
\end{align*}
\]

and the approximated expressions for the coefficient \( \square_k \) (rather than the “true” \( \beta_k \) which occurs in classical CG methods) are defined:

\[
\begin{align*}
\bar{\beta}_k &= \frac{g_{k+1}^T (g_{k+1} - g_k)}{d_k^T (g_{k+1} - g_k)}, \quad (\text{Hestenes and Stiefel, 1952}) \\
\beta_k &= \frac{g_{k+1}^T g}{d_k^T g_k} \\
\end{align*}
\]

(Dixon, 1975)
we have used these formula in the computations which are reported later.

After \( n \) steps the error vector used to find the minimum of quadratic.

Dixon’s method moves along a direction parallel to the CG direction, and so retains the property of the quadratic convergence. (Dixon, 1975).

2.2 The Sloboda Method (1982):

Sloboda developed an algorithm which generates conjugate direction with imperfect searches and has the quadratic termination property without using an error vector. The algorithm for general function is as follows:

The Outline of the SLO-Algorithm:

**Step (1):** Set \( x_0, \ g_0 = g_0, d_0 = -g_0 \)

**Step (2):** For \( k = 1, 2, \ldots, n \)

Compute \( x_{k+1} = x_k + d_k \) where \( d_k \) chosen to satisfy the condition of the line search

**Step (3):** If \( \| g_{k+1} \| < \varepsilon \) then stop, else go to step (5)

**Step (4):** If \( k = n + 1 \) then set \( k = 0 \) and go to step (1), else compute

\[
\hat{g}_{k+1} = (g_{k+1} - g_k) - \left[ \frac{(g_{k+1} - g_k)^T \hat{g}_k}{\hat{g}_k^T g_k} \right] \hat{g}_k ,
\]
In accurate CG-Algorithm ..

**Step (5):** If \( \|g_{k+1}\| < \varepsilon \) go to step (1), else set \( k = k + 1 \), compute

\[
\hat{d}_{k+1} = -\hat{g}_{k+1} + \left[ \frac{(g_{k+1} - g_k)^T \hat{g}_{k+1}}{(g_{k+1} - g_k)^T d_k} \right] \hat{d}_k,
\]

set \( k = k + 1 \), and go to step (3)

**2.3 The Multi-Step Method, Nazareth and Nocedal (1978):**

Nazareth and Nocedal show that with inexact line searches a natural extension of conjugate gradient method, the algorithm can be obtained of this method is called NAZ-NOC.

This algorithm is considered as a modification of Gram–Schmidt orthogonalization process, where Nazareth and Nocedal show that not all the coefficients of the Gram-Schmidt process must be computed at every iteration, then suggested the following algorithm:

**Step (1):** Set \( x_1, d_1 = -g_1, e_1 = 0 \)

**Step (2):** For \( k = 1, 2, \ldots, n \) compute

\[
d_{k+1} = -g_{k+1} + \frac{y_k^T g_k}{y_k^T d_k} d_k + e_k
\]

**Step (3):** Check for convergence.

If \( \|g_{k+1}\| < \varepsilon \) then stop, else go to step (4)

**Step (4):** if \( k < n \), set \( k = k + 1 \), compute

\[
e_k = e_{k-1} + \frac{y_{k-1}^T g_{k+1}}{y_{k-1}^T d_{k-1}} d_{k-1},
\]

go to step (2), else set \( k = 1 \) and go to step (1)
The algorithm will have quadratic termination property when using the error term $e_{k+1}$ in the extra step, even if implemented without line search.

3. The New CG –Method with Inexact Line Searches:

In this section a new general way for the conjugate gradient type methods is presented. This new approach has the property of the quadratic termination even if the line search is not exact and the extra correction term is not essential.

Now let the $g_k$ be the gradient of the quadratic function and let

$$g_{k+1} = g_k + k g_k + 1$$

where $y_k = g_{k+1} - g_k$, then the following lemma is hold.

**Lemma(1):**

For the quadratic function, the term $g_{k+1}^*$ which is defined in eq.(1) is equivalent to that the gradient $g_{k+1}$ which is obtained by Hestenes and Stiefel method in (1952).

**Proof:**

Let We Let’s define $g_k = Gx_k - b$, and $d_0 = -g_0$ where $g_k$ is the gradient of quadratic function. The biothogonalization process of Hestenes and Stiefel is as follows:

$x_{k+1} = x_k + \hat{\lambda}_k d_k$, where
In accurate CG-Algorithm..

\[ \lambda_k = \frac{g_k^T d_k}{g_k^T G d_k} \]

for exact line searches and \( G \) is a symmetric positive define matrix. Thus

\[ g_{k+1} = g_k + \lambda_k G d_k, \quad \text{(2)} \]

and

\[ d_{k+1} = -g_{k+1} + \frac{y_k^T g_{k+1}}{d_k^T y_k} d_k \]

Now, in order to prove that \( g_{k+1} \) which is obtained by Hestenes and Stiefel exact line searches algorithm is identical to the term \( g^* \) which is defined in eq.(1), we proceed as follows:

From the definition of \( \lambda_k = \frac{g_k^T d_k}{g_k^T G d_k} \), then rewriting eq.(2), we get

\[ g_{k+1} = g_k - \frac{g_k^T d_k}{g_k^T G d_k} G d_k \quad \text{(3)} \]

Now multiplying and dividing the second term of the eq. (3) by \( \lambda \), it becomes as:

\[ g_{k+1} = g_k - \frac{g_k^T d_k}{g_k^T G \lambda d_k} G d_k \]

from the definition we have \( y_k = g_{k+1} - g_k = \lambda_k G d_k \), then replacing \( y_k \) instead of \( \lambda_k G d_k \) in the above equation then we get:

\[ g_{k+1} = g_k - \frac{g_k^T d_k}{g_k^T y_k} y_k \]

thus this equation is identical to the equation which is defined in eq.(1) as:

\[ g^*_{k+1} = g_k - \frac{g_k^T d_k}{g_k^T y_k} y_k \]

40
thus the is of vectors $g_0, \ldots, g_k$ is orthogonal as in Hestenes and Stiefel method in quadratic function. From the above argument we have the following two corollaries.

**Corollary (1):**

The term $g_{k+1}^*$ which is defined in eq.(1) is used to be obtained by the following form:

$$d_{k+1}^* = -g_{k+1}^* + \left( g_{k+1}^* - g_k^* \right) d_k^*,$$

(4)

is parallel to that search direction given by Hestenes and Stiefel algorithm for quadratic function.

**Corollary (2):**

The search direction which is defined in eq. (4) is a descent direction even if for non-quadratic functions i.e.

$$d_{k+1}^T g_{k+1}^* < 0.$$

**Proof:**

Re-write the direction in eq. (4)

$$d_{k+1}^* = -g_{k+1}^* + \beta d_k^*,$$

Multiplying this direction by $g_{k+1}^*$, then we have:
**In accurate CG-Algorithm..**

\[ d_{k+1}^{*T}g_{k+1}^{*} = -g_{k+1}^{*T}g_{k+1}^{*} + \beta^{*} d_{k}^{T}g_{k+1}^{*} \]

and then \( d_{k+1}^{*T}g_{k+1}^{*} = -\left\| g_{k+1}^{*} \right\|^{2} \)

(which is always negative relation).

This result is true, because we have \( d_{k}^{T}g_{k+1}^{*} = 0 \), it is easy to prove this result as follows:

We have

\[ g_{k+1}^{*} = \frac{g_{k}^{T}g_{k}y_{k}}{g_{k}^{T}y_{k}} g_{k}^{*} \]

So \( y_{k} = g_{k+1} - g_{k} \). Now multiply eq. (5) by \( d_{k}^{T} \), then we obtain the following result:

\[ d_{k}^{*T}g_{k+1}^{*} = d_{k}^{T}g_{k}^{*} - \frac{d_{k}^{T}g_{k}}{g_{k}^{T}y_{k}} g_{k}^{T}y_{k} \]

thus, we get:

\[ d_{k}^{*T}g_{k+1}^{*} = 0 \]

**4. The Outline of the New Algorithm:**

**Step (1):** \( d_{0}^{*} = -g_{0} = -g^{0} \)

**Step (2):** For \( k=1,2... \) compute

\[ x_{k+1} = x_{k} + \lambda d_{k} \]

, for compute \( \lambda \). where \( \lambda \) is the step size.

42
Step (3): Check for convergence \( \|g_{k+i}\|_\infty > \epsilon \), then stop.

Otherwise go to step (4).

Step (4): Compute \( g_{k+1}^* \) as defined in eq. (1)

Step (5): Find the new search direction

\[
d_{k+1}^* = -g_{k+1}^* + \beta_k^* d_k^*, \text{where}
\]

\[
\beta_k = -\frac{g_{k+1}^* g_{k+1}^T}{g_k^* g_k^T},
\]

\[
\beta_k = \frac{y_k^T g_{k+1}^*}{g_k^* g_k^*},
\]

\[
\beta_k = -\frac{g_{k+1}^* g_{k+1}^T}{d_k^* g_k^*},
\]

\[
\beta_k = \frac{y_k^T g_{k+1}^*}{y_k^T d_k^*},
\]

These definitions of \( \beta_k^* \) in eq. (6) due to Dixon (1975); in \( \beta_k^* \) eq. (7), due to Polok and Ribiere (1969); \( \beta_k^* \) in eq. (8) due to Al-Bayati and Al-Assady (1996); \( \beta_k^* \) in eq. (9) due to Hestenes and Stiefel (1952).

Step (6): Check for restarting criterion if \( k = n \) then set \( k = 0 \) and go to step (1) else go to step 2.
Because of the orthogonality property and the lemma (1) are held, this algorithm is identical to the original (H/S, Hestenes And Stiefel), (P/R; Polok and Ribiere), (DX; Dixon), and (AB; Al-Assady and Al-Bayati) algorithms in quadratic function.

Generally function is reduced P/R and A/B algorithms even it Inexact searches can be used as it well be shown in Corollary (2).

5. Numerical Results:

Several standard test functions were minimized to compare the new algorithm with standard CG algorithms. The same line search was employed in each of the algorithms. The cubic interpolation, and the algorithms were terminated if the norm of the gradient was reduced below $1 \times 10^{-5}$. We tabulate for all the algorithms the number of calls of the function evaluation (NOF), and the number of iterations (NOI). Overall totals are also given for NOF and NOI with each algorithm.

Table (1) contains the numerical results for the new algorithm with (DX) formula and the standard CG algorithm with the same formula. In this table we see that the new algorithm is more efficient than the standard CG algorithm, this is obtained from the NOF and the NOI of both algorithms.

Table (2) includes the results of the standard CG (H/S) formula and the new with (H/S) formula, this table indicates that the new algorithm is better than the standard CG (H/S) in (4) out of (12) cases and in (5) cases they are comparable.
Table (3) gives the comparison between the results of the new (A/B) algorithm which is presented in eq. (1) with classical CG (A/B), the results in this table indicate that the new method is better than the classical CG (A/B) algorithm in (6) out of (12) cases.

In table (4) we have compared the new algorithm with (P/R) formula with standard CG (P/R). It is obvious that the new algorithm improves the standard (P/R) algorithm in about (80.2%) NOI and (79.6%) NOF.

In table (5) we represent a numerical example to show that the performance of suggested algorithm is quick and has better performance since it requires less time to execute.
Table (1)

Comparative performance of the two algorithms for a group of test functions, by using $\beta : (DX)$.

<table>
<thead>
<tr>
<th>Test function</th>
<th>N</th>
<th>Standard CG</th>
<th>New method NOI(NOIF)</th>
<th>NOI(NOIF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Powell</td>
<td>4</td>
<td>76(197)</td>
<td>40(105)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>104(219)</td>
<td>85(189)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>114(241)</td>
<td>107(223)</td>
<td></td>
</tr>
<tr>
<td>Wood</td>
<td>4</td>
<td>37(85)</td>
<td>22(47)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>64(176)</td>
<td>54(127)</td>
<td></td>
</tr>
<tr>
<td>Rosen</td>
<td>2</td>
<td>37(88)</td>
<td>32(76)</td>
<td></td>
</tr>
<tr>
<td>Cubic</td>
<td>2</td>
<td>17(48)</td>
<td>17(48)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>64(138)</td>
<td>40(96)</td>
<td></td>
</tr>
<tr>
<td>Dixon</td>
<td>10</td>
<td>27(56)</td>
<td>24(50)</td>
<td></td>
</tr>
<tr>
<td>Beale</td>
<td>2</td>
<td>10(26)</td>
<td>10(25)</td>
<td></td>
</tr>
<tr>
<td>Reciep</td>
<td>30</td>
<td>8(22)</td>
<td>7(20)</td>
<td></td>
</tr>
<tr>
<td>3-Powell</td>
<td>3</td>
<td>15(34)</td>
<td>13(29)</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>573(1330)</td>
<td>451(1035)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tools</th>
<th>Standard CG</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOI</td>
<td>100</td>
<td>78.7</td>
</tr>
<tr>
<td>NOF</td>
<td>100</td>
<td>77.8</td>
</tr>
</tbody>
</table>
Table (2)
Comparative performance of the two algorithms for a group of test functions, by using $\beta : (H/S)$.

<table>
<thead>
<tr>
<th>Test function</th>
<th>N</th>
<th>Standard CG</th>
<th>New method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NOI(NO)</td>
<td>NOI(NO)</td>
<td>NOI(NO)</td>
</tr>
<tr>
<td>Powell</td>
<td>4</td>
<td>50(114)</td>
<td>54(128)</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>34(78)</td>
<td>31(75)</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>105(244)</td>
<td>105(230)</td>
</tr>
<tr>
<td>Wood</td>
<td>4</td>
<td>21(46)</td>
<td>23(49)</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>48(101)</td>
<td>45(93)</td>
</tr>
<tr>
<td>Rosen</td>
<td>2</td>
<td>31(73)</td>
<td>31(73)</td>
</tr>
<tr>
<td>Cubic</td>
<td>2</td>
<td>18(50)</td>
<td>18(50)</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>13(35)</td>
<td>13(35)</td>
</tr>
<tr>
<td>Dixon</td>
<td>10</td>
<td>22(46)</td>
<td>21(44)</td>
</tr>
<tr>
<td>Beale</td>
<td>2</td>
<td>10(26)</td>
<td>10(26)</td>
</tr>
<tr>
<td>Reciep</td>
<td>30</td>
<td>9(24)</td>
<td>9(24)</td>
</tr>
<tr>
<td>3-Powell</td>
<td>3</td>
<td>18(40)</td>
<td>19(41)</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>379(877)</td>
<td>379(868)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tools</th>
<th>Standard CG</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOI</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>NOF</td>
<td>100</td>
<td>98.9</td>
</tr>
</tbody>
</table>
In accurate CG-Algorithm ..

Table (3)
Comparative performance of the two algorithms for a group of test functions, by using $\beta : (A/\beta)$.

<table>
<thead>
<tr>
<th>Test function</th>
<th>N</th>
<th>Standard CG</th>
<th>New method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NOI(NOIF)</td>
<td>NOI(NOIF)</td>
</tr>
<tr>
<td>Powell</td>
<td>4</td>
<td>65(144)</td>
<td>40(105)</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>68(155)</td>
<td>65(146)</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>105(250)</td>
<td>107(223)</td>
</tr>
<tr>
<td>Wood</td>
<td>4</td>
<td>25(58)</td>
<td>22(47)</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>44(95)</td>
<td>54(127)</td>
</tr>
<tr>
<td>Rosen</td>
<td>2</td>
<td>31(73)</td>
<td>32(76)</td>
</tr>
<tr>
<td>Cubic</td>
<td>2</td>
<td>19(52)</td>
<td>17(48)</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>14(36)</td>
<td>40(96)</td>
</tr>
<tr>
<td>Dixon</td>
<td>10</td>
<td>23(48)</td>
<td>24(50)</td>
</tr>
<tr>
<td>Beale</td>
<td>2</td>
<td>10(26)</td>
<td>10(25)</td>
</tr>
<tr>
<td>Reciep</td>
<td>30</td>
<td>9(24)</td>
<td>7(20)</td>
</tr>
<tr>
<td>3-Powell</td>
<td>3</td>
<td>19(41)</td>
<td>13(29)</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>432(1002)</td>
<td>431(992)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tools</th>
<th>Standard CG</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOI</td>
<td>100</td>
<td>99.7</td>
</tr>
<tr>
<td>NOF</td>
<td>100</td>
<td>99</td>
</tr>
</tbody>
</table>

48
### Table (4)

Comparative performance of the two algorithms for group of test functions, by using $\beta : (P/R)$.

<table>
<thead>
<tr>
<th>Test function</th>
<th>N</th>
<th>Standard CG</th>
<th>New method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NOI(NOF)</td>
<td>NOI(NOF)</td>
<td></td>
</tr>
<tr>
<td>Powell</td>
<td>4</td>
<td>77(183)</td>
<td>57(150)</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>76(173)</td>
<td>41(87)</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>105(227)</td>
<td>93(188)</td>
</tr>
<tr>
<td>Wood</td>
<td>4</td>
<td>33(75)</td>
<td>26(56)</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>44(93)</td>
<td>33(70)</td>
</tr>
<tr>
<td>Rosen</td>
<td>2</td>
<td>31(73)</td>
<td>31(73)</td>
</tr>
<tr>
<td>Cubic</td>
<td>2</td>
<td>19(52)</td>
<td>19(52)</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>16(38)</td>
<td>12(32)</td>
</tr>
<tr>
<td>Dixon</td>
<td>10</td>
<td>22(46)</td>
<td>21(44)</td>
</tr>
<tr>
<td>Beale</td>
<td>2</td>
<td>10(26)</td>
<td>10(26)</td>
</tr>
<tr>
<td>Reciep</td>
<td>30</td>
<td>9(24)</td>
<td>9(24)</td>
</tr>
<tr>
<td>3-Powell</td>
<td>3</td>
<td>18(43)</td>
<td>17(37)</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>460(1053)</td>
<td>369(839)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tools</th>
<th>Standard CG</th>
<th>NEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOI</td>
<td>100</td>
<td>80.2</td>
</tr>
<tr>
<td>NOF</td>
<td>100</td>
<td>79.6</td>
</tr>
</tbody>
</table>
Table (5)
The time of the execute, (sec) of the powell function at Dimension 100 by using $\beta$:H/S.

<table>
<thead>
<tr>
<th>Standard CG Method</th>
<th>New Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2517</td>
<td>1.9284</td>
</tr>
</tbody>
</table>
6. Appendix:

1- Generalized Powell Function:

\[ f(x) = \sum_{i=1}^{n/4} \left[ (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i}) + (x_{4i-2} - 2x_{4i-3})^4 + 10(x_{4i-3} - x_{4i})^4 \right] \]

\[ x_0 = (3,-1,0,1,\ldots)^T. \]

2- Generalized Wood Function:

\[ f(x) = \sum_{i=2}^{n/2} \left[ 100(x_{4i-2} - x_{4i-3})^2 + (1 - x_{4i-2})^2 + 9(x_{4i} - x_{4i-1})^2 + (1 - x_{4i-1})^2 \right. 

\[ \left. + 10.1((x_{4i-2} - 1)^2 + (x_{4i} - 1)^2) + 19.8(x_{4i-2} - 1)^2(x_{4i} - 1) \right], \]

\[ x_0 = (-3,-1,-3,-1,\ldots)^T. \]

3- Generalized Rosenbrock Function:

\[ f(x) = \sum_{i=1}^{n/2} \left[ 100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2 \right], \]

\[ x_0 = (-1.2,1;\ldots)^T. \]

4- Cubic Function:

\[ f(x) = \sum_{i=1}^{n/2} \left[ 100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2 \right], \]

\[ x_0 = (-1.2,1;\ldots)^T. \]

5- Generalized Dixon Function:

\[ f(x) = \sum_{i=1}^{n} \left[ (1 - x_i)^2 + (1 - x_n)^2 + \sum_{i=1}^{n-1} (x_i^2 - x_{i+1})^3 \right], \]

\[ x_0 = (-1;\ldots)^T. \]

6- Generalized Beale Function:

\[ f(x) = \sum_{i=1}^{n/2} \left[ \left( 1 - x_{2i-1} - (1 - x_{2i}) \right)^2 \right. \]

\[ \left. + \left( 2.25 - x_{2i-1} - (1 - x_{2i}) \right)^2 \right], \]

\[ x_0 = (1,1;\ldots)^T. \]

7- Generalized Recip Function:
In accurate CG-Algorithm.

\[
f(x) = \sum_{i=1}^{n/3} \left\{ (x_{3i-1} - 5)^2 + x_{3i-1}^2 + \frac{x_{3i}^2}{(x_{3i-1} - x_{3i-2})^2} \right\},
\]

\[x_0 = (2, 5, 1; ...)^T.\]

8- Generalized 3-Powell Function:

\[x_0 = (0, 1, 2; ...)^T.\]
REFERENCES


