A Combined Cubic and Novel Line Search CG-Algorithm
Abbas Y. Al-Bayati
profabbasalbayati@yahoo.com
Hamsa TH. Chilmeran
hamsathrot@uomosul.edu.iq
College of Computer Sciences and Mathematics
University of Mosul, Iraq

Received on: 10/12/2006
Accepted on: 15/08/2007

ABSTRACT
In this paper a new line search technique is investigated. It uses (cubic and novel) line searches in the standard CG-algorithm for unconstrained optimization. Applying our new modified version on CG-method shows that, it is too effective when compared with other established algorithms, in this paper, to solve standard unconstrained optimization problems.

Keywords: CG-algorithms, step length, quadratic function.

1. Introduction.
The Conjugate Gradient (CG) method is particularly useful for minimizing function of many variables because it does not require the storage of any matrices. However, the rate of convergence of the algorithm is only linear unless the iterative procedure is "restarted" occasionally. At present it is usual to restart the CG-algorithm every n or (n+1) iterations, where n is number of variables, but it is known that frequency of restarts should depend on the objective function. CG-algorithm is useful to find minimum of a function \( f: \mathbb{R}^n \rightarrow \mathbb{R} \), In general, the method has the following form

\[
d_{k+1} = \begin{cases} -g_k & \text{for } k = 1 \\ -g_{k+1} + \beta_k d_k & \text{for } k > 1, \end{cases}
\]

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

where \( g_k \) denotes the gradient \( \nabla f(x_k) \), \( d_k \) is the search direction, \( \lambda_k \) is a step-length obtained by a line search, and \( \beta_k \) is chosen so that it becomes the \( k \)-th conjugate direction when the function is quadratic and the line search is exact (Sun, et al, 2003), a Well Known formula for \( \beta_k \) is given by:

\[
\beta_k = \frac{y_k^T g_{k+1}}{d_k^T y_k} \quad \text{(Hestenes & Stiefel, 1952)}
\]

\[
\beta_k = \frac{\|g_{k+1}\|^2}{\|g_k\|^2} \quad \text{(Fletcher & Reeves, 1964)}
\]
\[ \beta_k = \frac{g_{k+1}^T (g_{k+1} - g_k)}{\|g_k\|^2} \] (Polak & Ribiere, 1969)  
\[ \beta_k = -\frac{\|g_{k+1}\|^2}{d_k^T g_k} \] (Dixon, 1975)  
\[ \beta_k = -\frac{g_{k+1}^T y_k}{d_k^T g_k} \] (Al-Bayati & Al-Assady, 1986)

where \( y_k = g_{k+1} - g_k \)

The successive direction are conjugate vectors for successive gradients obtained as the method progresses. At step \( k \) one evaluates the current negative gradient and adds it to a linear combination of the previous direction vectors to obtain a new conjugate direction along which to move (Cristain, 2003).

2. Conjugate Gradient Algorithm with a Novel Line Search:

Let \( f = f(x_k) \) be continuously differentiable of \( x \in \mathbb{R}^n \) with gradient \( g(x) = \nabla f(x) \). The problem of finding a minimizer of \( f \) is often investigated by a CG-algorithm \( x_{k+1} = x_k + \lambda d_k \), where \( \lambda \) is the optimal step size, computed by a novel line search technique (Sun, et al, 2003).

2.1 The One–Dimensional Novel Line Search Procedure:

The following steps describe novel line search technique:

**Step 1:** Let \( f(x) \) be any unimodal function. Set \( x_i = a, f_i = f(x_i) \) and \( k = 1 \), \( \varepsilon \) is any small parameter. Set \( x_2 = T_0, f_2 = f(x_2) \), where \( T_0 \) is the initial step length \( 0 < T_0 < 1 \) and \( a \) is assumed to be available; \( a = 0.0 \) is normally taken.

**Step 2:** If \( f_2 \geq f_1 \) then, set \( T_0 = T_0 / 3 \) and go to step (1) else go to the next step (3).

**Step 3:** a) Set \( k = k + 1 \)
   b) \( f_k = f(x_k) \), where \( x_k = 2T_0 \)
   c) if \( f_{k+1} \geq f_k \), then go to step 4 else set \( T_0 = 2T_0 \) and go to position a.

**Step 4:** Steps (1) to (3) give the initial interval of uncertainty. Let:
\[
\begin{align*}
  y_1 &= x_{k-2}  \\
  f_{y_1} &= f_{k-2} \\
  y_3 &= x_{k-1}  \\
  f_{y_3} &= f_{k-1} \\
  y_5 &= x_k  \\
  f_{y_5} &= f_k 
\end{align*}
\]
also let \( f_s = f_{y_5} \)

**Step 5:** Obtain two more points in this interval of uncertainty as follows:
\[
\begin{align*}
  y_2 &= y_3 - [(f_{y_3} - f_{y_5})/f_{y_5}][y_5 - y_3/y_3];  \\
  f_{y_2} &= f(y_2) \\
  y_4 &= y_3 - [(f_{y_3} - f_{y_5})/f_{y_5}][y_5 - y_3/y_3] \\
  f_{y_4} &= f(y_4) \\
\end{align*}
\]
Now there are five points \( y_1, y_2, y_3, y_4 \) and \( y_5 \). if the value of points are not in the increasing order, order them such that \( y_1 < y_2 < y_3 < y_4 < y_5 \) and \( f_{y_1}, f_{y_2}, f_{y_3}, f_{y_4}, \) and \( f_{y_5} \) are their corresponding function values. choose a
point, which has the lowest function value among the five points and set it as \( B \). Let \( f_{\tilde{B}} \) be its corresponding function value. Take adjacent point of \( B \) and set them as \( \tilde{A} \) and \( \tilde{C} \), such that: \( \tilde{A} < \tilde{B} < \tilde{C} \rightarrow f_{\tilde{A}} \geq f_{\tilde{B}} \leq f_{\tilde{C}} \); \( X_0 = [(\tilde{A}^2 + \tilde{B}^2 + \tilde{C}^2)/3]^{1/2} \); \( f = f(X_0) \). Now we have four points \( X_0, \tilde{A}, \tilde{B}, \tilde{C} \) and their respective function values \( f_0, f_{\tilde{B}}, f_{\tilde{A}}, f_{\tilde{C}} \). Again choose three appropriate points and set them as \( y_1, y_3 \) and \( y_5 \) such that \( y_1 < y_3 < y_5 \Rightarrow f_{y_1} \geq f_{y_3} \leq f_{y_5} \); \( f_{y_1}, f_{y_3}, f_{y_5} \), their respective function values.

**Step 6:** If the inequality \(|(f_s - f_0)/f_0| \leq \varepsilon \), is satisfied, then the search is terminated with \( x^* = y_3 \) else \( f_0 = \min(f_s, f_0) \) and go to (5). The \( \varepsilon \) parameter control the accuracy of the solution – desired. it is value may be recommend a value of \( \varepsilon = 10^{-5} \) for most the usually encountered problems (Rao and Chandra, 1983).

3. Cubic Interpolation Technique:

In the proposed cubic polynomial method, a gradient and a function evaluation are made at every iteration at \( k \). At each iteration an update is performed when a new point \( x_{k+1} \) that satisfies the condition \( f(x_{k+1}) \leq f(x_k) \) in many of the n-dimensional routines, and the values of the first derivative are calculated. It may be desirable to use this information. An efficient algorithm based on fitting a cubic equation through the data at two points is often used this algorithm.

To start the procedure, point (a) is selected. The derivative is evaluated and a step \( h \) taken in the direction of decreasing \( f \). The derivative \( f'(a + h) \) is then evaluated. If \( f'(a + h) \) has the same sign \( f'(a) \) then the point \( (a + h) \) is chosen as the new origin. The step size \( h \) is doubled and the step repeated. If \( f'(a + h) \) is zero or has changed sign, then a cubic is fitted to cubic and may be written as

\[
f(x) = Ax^3 + Bx^2 + Cx + D \tag{3}
\]

They may be found by solving a set of four equations, for convenience, the origin will be taken at the point a

\[
f(a) = D \tag{4}
\]

\[
f(b) = Ax^3 + Bx^2 + Cx + D \tag{5}
\]

\[
v = f'(a) = C \tag{6}
\]

\[
U = f'(b) = 3Ax^2 + 2Bx + C \tag{7}
\]

At the required solution:

\[
f'(x) = 3Ax^2 + 2Bx + C = 0 \tag{8}
\]

And

\[
f''(x) = 6Ax + 2Bv \tag{9}
\]

The solution of above equation are

\[
x = (-B\mu\sqrt{B^2 - 3AC})/3A \quad \text{if} \quad A \neq 0 \tag{10}
\]

\[
x = -C/2B \quad \text{if} \quad A = 0 \tag{11}
\]
And, of the former of these, only the positive single satisfies equation (9). Let us define:

\[ s = -3(f(a) - f(b)0/b = 3(Ab^2 + Bb + C) \]  
\[ \hat{z} = s - u - v = Bb + C \]  
\[ w^2 = \hat{z}^2 - uv = b^2(B^2 - 3AC) \]

Then the most readily obtainable solution is obtained from:

\[ v - \hat{z} + w = b(-B\mu\sqrt{B^2 - 3AC}) \]

And

\[ u + v - \hat{z} = 3Ab^2 \]

And the minimum is given by

\[ x = b(v - \hat{z} + w)/(u + v - \hat{z}) \]

This solution is not acceptable, however, as when \( A=0 \), it becomes indeterminate. Davidon (1959) therefore replaces it by:

\[ x = b\left(1 - \frac{u + w + \hat{z}}{u - v + 2w}\right) \]

In an n-dimensional problem. It is usual to accept the point \( x \) predicted by equation (18) as the minimum provided \( f(x) < f(a) \) and \( f(x) < f(b) \). To fit the cubic we will use the function and its gradient at two points as \( a, f(a), f'(a), b, f(b), f'(b) \) (Sun, J., etc, 2003, 2005).

3.1 Outline of the Cubic Technique (Bundy, 1984):

Step 1: Set \( a_1, h, \varepsilon \).

Step 2: Find \( v = f'(a) \) and if \( f'(a) > 0 \) then \( h = -h \) and go to step (3); else set \( h = +h \).

Step 3: Find \( b_1 = a_i + h \); compute \( u = f'(b) \).

Step 4: If \( uv < 0 \), then go to step (5); else set \( h = 2h, v = u, a_i = b_1 \) and go to step (3).

Step 5: For \( k = 1,2,\ldots, \) repeat.

Step 6: Set \( z = 3/(b_k - a_k)(f_a - f_b) + u + v \)

\[ w = (z^2 - uv)^{1/2} \]

\[ x^* = a_k + (b_k - a_k)\{1 - (u + w - z)/u - v + 2w\} \]

and evaluate \( f_x = f(x^*), g_x = g(x^*) \).

Step 7: If \( g_x < 0 \) or \( f_x < f(a_x) \), then set \( a_{k+1} = x^*, b_{k+1} = b_k \) and \( f_a = f_x, f_a' = g_x \) go to step (9), else set \( a_{k+1} = a_k, b_{k+1} = x^* \) and \( f_b = f_x, f_b' = g_x \).

Step 8: If \( b_{k+1} - a_{k+1} \leq \varepsilon \), then \( x^* \) is the root, else go to step (5).

4. A new combined Novel and Cubic Line Searches:

This new technique combines between the cubic interpolation & Novel line searches to find the minimum of multi-dimension function subject to necessary condition \( f'(x) = 0 \) and sufficient condition \( f(x_{k+1}) < f(x_k) \).
4.1 Outline of the Combined Technique are as follows:

**Step 1:** Let \( x_0 \) be the initial point; set \( \varepsilon, H(1) = 0, T_0 = 1 \).

**Step 2:** Find the function of \( x_k, f(x_k) = f_k \).

**Step 3:** Set \( k = 1 \).

**Step 4:** Set \( d_k = -g_k \).

**Step 5:** Set \( x_{k+1} = x_k + T_0 d_k \).

**Step 6:** Find the function of \( x_{k+1}, f(x_{k+1}) = f_{k+1} \).

**Step 7:** Check if \( f_{k+1} \geq f_k \) is satisfied then set \( T_0 = T_0 / 3 = H(2) \) and go to step (5); otherwise go to step (8).

**Step 8:** Set \( T_0 = 2T_0 = H(3) \); \( x_{k+2} = x_{k+1} + T_0 d_k \).

**Step 9:** Check if \( f_{k+2} \geq f_{k+1} \) is satisfied then set \( \tilde{B} = H(1) \).

**Step 10:** Set \( x_{k+1} = x_k + \tilde{C} d_k \).

**Step 11:** Set \( x_{k+1} = x_k + \tilde{B} d_k \) and \( x_{k+1} = x_k + \tilde{A} d_k \).

**Step 12:** Compute \( f_p = f(x_{k+1}) \) and \( f_Q = f(x_{k+1}) \).

**Step 13:** Compute \( G_p = g^T d_{k+1} \) and \( G_Q = g^T d_{k+1} \).

**Step 14:** Put \( B = hh \).

**Step 15:** Set \( \tilde{A} = 3*(F_p - F_Q) / hh \) and \( \tilde{B} = zz + G_p + G_Q \).

**Step 16:** Compute \( w = \tilde{B} - \tilde{A} \).

**Step 17:** Check if \( w < 0 \), then set \( w = 0 \) and go to step (19) else put \( w = (ww)^{1/2} \).

**Step 18:** Compute \( \tilde{A} = hh^*(1 - (G_Q + w - zz)/(G_Q - G_p + 2*w)) \) and set \( \lambda_k = \tilde{A} \).

**Step 19:** Find \( x_{k+1} = x_k + \lambda_k d_k \).

**Step 20:** Find \( f_R = f(x_{k+1}) \).

**Step 21:** Find \( G_R = g(x_{k+1})^T d(x_{k+1}) \).

**Step 22:** Check if \( |G_R| < 0.9|G_p| \) and \( |f_R - f_p| \leq 0.01 dd * G_p \) is satisfied then go to step(26); otherwise

**Step 23:** Check if \( G_R > 0 \).

**Step 24:** Set \( hh = dd, G_Q = G_R \) Go to step (16); else set \( hh = hh - dd \) and \( G_p = G_R \) then go to step (16).

**Step 25:** Check if \( |Z| \leq \varepsilon \). Print minimum found; else

**Step 26:** Set \( d_{k+1} = -g_{k+1} + \beta_k d_k \).

**Step 27:** Check if \( k > n \) go to step (3); else set \( k = k + 1 \) and go to step (4).
2. Flowchart of CG algorithm with New Combined Novel and Cubic Line Searches:

Set $x_0$ is the initial point, $\varepsilon$, $T_0$ is scalar, $H(1) = 0.0$

$K = 1$

Find $f(x_k) = f_k$

$d_k = -g_k$

$x_{k+1} = x_k + T_0 d_k$

Find $f(x_{k+1}) = f_{k+1}$

If $f_{k+1} \geq f_k$

$T_0 = 2 * T_0 = H(3)$

$x_{k+2} = x_{k+1} + T_0 * d_k$

Find $f(x_{k+2}) = f_{k+2}$

If $f_{k+2} \geq f_{k+1}$

$T_0 = T_0 / 3 = H(2)$

YES

NO
A Combined Cubic and Novel Line Search CG-Algorithm

\[
\tilde{B} = H(1) \\
\tilde{A} = \tilde{B} / 2 \\
\tilde{C} = 2 \tilde{B}
\]

\[
\tilde{A} \pi \tilde{B} \pi \tilde{C} \\
f_{\tilde{A}} \geq f_{\tilde{B}} \geq f_{\tilde{C}}
\]

\[
x_{\tilde{A}} = x_k + \tilde{A} d_k \\
x_{\tilde{B}} = x_k + \tilde{B} d_k \\
x_{\tilde{C}} = x_k + \tilde{C} d_k
\]

\[
f_p = f(x_{\tilde{A}}) \\
f_Q = f(x_{\tilde{B}})
\]

\[
G_p = g(x_{\tilde{A}})^T d_k \\
G_Q = g(x_{\tilde{B}})^T d_k
\]

\[
hh = \tilde{B}
\]

\[
ZG = 3(\tilde{F}_Q - f_p) / \tilde{B} \\
ZG = ZG + GP + GQ
\]

\[
ww = (z^2 - G_p + G_Q) / 2
\]

\[
ww = 0 \\
ww < 0
\]

Yes

*****
5. Numerical Results:

In this section, we discuss computational results of H/S-CG method using cubic and novel line searches techniques for finding the minimum of nonlinear objective functions. The results are obtained using Pentium IV Computer. The programs were written in FORTRAN Language. The comparative performance of the algorithms is evaluated by considering $NOF$, $NOI$, and $TW$ is the best measure of actual work done. Table (1) includes the numerical results of (H/S) CG-algorithm by using cubic line search and Novel line search for $4 \leq n \leq 1000$. Table (2) includes the numerical results of (F/R)-CG method by using cubic line search and employ new combined cubic-Novel line research for test function of $4 \leq n \leq 1000$.

**Table (1).** Comparisons of CG-algorithm (Novel line search) with CG-algorithm (cubic line search)

<table>
<thead>
<tr>
<th>Test fun.</th>
<th>Dim</th>
<th>H/S- algorithm (cubic and novel line searches)</th>
<th>H/S- algorithm (cubic line searches)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$TW=NOF$</td>
<td>$NOI$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Central</td>
<td>4</td>
<td>733</td>
<td>50</td>
</tr>
<tr>
<td>Mniele</td>
<td>4</td>
<td>1857</td>
<td>126</td>
</tr>
<tr>
<td>Powell</td>
<td>4</td>
<td>2311</td>
<td>163</td>
</tr>
<tr>
<td>Sum</td>
<td>4</td>
<td>153</td>
<td>8</td>
</tr>
<tr>
<td>Osp</td>
<td>4</td>
<td>138</td>
<td>8</td>
</tr>
<tr>
<td>Central</td>
<td>100</td>
<td>453</td>
<td>27</td>
</tr>
<tr>
<td>Powell3</td>
<td>100</td>
<td>331</td>
<td>29</td>
</tr>
<tr>
<td>Powell</td>
<td>100</td>
<td>2152</td>
<td>120</td>
</tr>
<tr>
<td>Herical</td>
<td>100</td>
<td>488</td>
<td>41</td>
</tr>
<tr>
<td>Osp</td>
<td>1000</td>
<td>1600</td>
<td>159</td>
</tr>
<tr>
<td>Wolfe</td>
<td>1000</td>
<td>456</td>
<td>46</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>10672</strong></td>
<td><strong>777</strong></td>
</tr>
</tbody>
</table>

**Table (2).** Comparisons of our new algorithm with standard CG-algorithm

<table>
<thead>
<tr>
<th>Test fun.</th>
<th>Dim</th>
<th>F/R- algorithm (cubic and novel line searches)</th>
<th>F/R- algorithm (cubic line searches)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$TW=NOF(novel)+(n+1)NOF$ (cubic)</td>
<td>$NOI$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rosen</td>
<td>4</td>
<td>1593</td>
<td>62</td>
</tr>
<tr>
<td>Miele</td>
<td>4</td>
<td>425</td>
<td>16</td>
</tr>
<tr>
<td>Cubic</td>
<td>4</td>
<td>469</td>
<td>18</td>
</tr>
</tbody>
</table>

$TW=NOF(N+1)$

$NOI$
From the above two tables we can find the following:

**Table (3).** Performance percentage of the new algorithm compared with standard CG-algorithm (cubic line search)

<table>
<thead>
<tr>
<th>Tools</th>
<th>NOI</th>
<th>TW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>%100</td>
<td>%64</td>
</tr>
</tbody>
</table>

**Appendix**

1. **Generalized Osp (Oren and Spedicato Function):**

\[ f(x) = \left[ \sum_{i=1}^{n} i x_i^2 \right]^2, \quad x_0 = (1, ..., T)^T. \]

2. **Wolfe Function:**

\[ f(x) = \left[ -x_i (3 - x_i / 2) + 2x_2 - 1 \right]^2 + \sum_{i=1}^{n} [x_{i-1} - x_i (3 - x_i / 2 + 2x_{i+1} - 1)]^2 + [x_{n-1} - x_n (3 - x_n / 2 - 1)]^2, \quad x_0 = (-1, ..., T)^T. \]

3. **Generalized Cantrell Function:**

\[ f(x) = \sum_{i=1}^{n} \left[ \left( \exp(x_{i-1}) - x_{i-2} \right)^4 + 100(x_{i-2} - x_{i-1})^6 + \right] \left( \arctan(x_{i-1} - x_i) \right)^4 + x_{i-3}, \quad x_0 = (1, 2, 2, 2; ...)^T. \]
4. Generalized Miele Function:
\[
f(x) = \sum_{i=1}^{n/2} \left[ \exp(x_{4i-3} - x_{4i-1})^2 + 100(x_{4i-2} - x_{4i-1})^2 + x_{4i-3}^4 + (x_{4i} - l)^2 \right] , \quad x_0 = (1,2,2,2;...)^T.
\]

5. Generalized Powell Function:
\[
f(x) = \sum_{i=1}^{n/2} \left[ (x_{4i-3} + 10x_{4i-2})^4 + 5(x_{4i-1} - x_{4i})^2 \right] , \quad x_0 = (3,-1,0,3;...)^T.
\]

6. Generalized Powell 3 Function:
\[
f(x) = \sum_{i=1}^{n/2} \left[ \frac{1}{1 + (x_i - x_{2i})^2} - \sin(\frac{\pi x_{2i} x_{3i}}{2}) - \exp\left[ -\left( \frac{x_{i} + x_{3i}}{x_{2i}} - 2 \right)^2 \right] \right] , \quad x_0 = (0,1,2;...)^T.
\]

7. Rosenborck Function:
\[
f(x) = \sum_{i=1}^{n/2} [100(x_{2i} - x_{2i-1})^2 + (1 - x_{2i-1})^2] , \quad x_0 = (-1,2,1;...)^T.
\]

8. Generalized Cubic Function:
\[
f(x) = \sum_{i=1}^{n/2} [100(x_{2i} - x_{2i-1})^2 + (1 - x_{2i-1})^2] , \quad x_0 = (-1,2,1;...)^T.
\]

9. Generalized Shallow Function:
\[
f(x) = \sum_{i=1}^{n/2} [x_{2i-1}^2 - x_{2i}]^2 + (1 - x_{2i-1})^2 , \quad x_0 = (-2, -2;...)^T.
\]

10. Sum of Quatrics (SUM) function:
\[
f(x) = \sum_{i=1}^{n/2} (x_j - i)^4 , \quad x_0 = (1, ...)^T.
\]

11. Herical function:
\[
f(x) = 100(x_{3i}) - 10 \left( \frac{1}{2\pi} \arctan \left( \frac{x_{3i-1}}{x_{3i-2}} \right) \right)^2 + 100(\sqrt{x_{3i-2}^2 + x_{3i-1}^2} - 1)^2 + x_{3i}^2 , \quad x_0 = (-1, 0, 0;...)^T.
\]
REFERENCES


