A New Family of Spectral CG-Algorithm

Abbas Y. Al Bayati
profabbasalbayati@yahoo.com
College of Computer Sciences and Mathematics
University of Mosul, Iraq
Received on: 07/01/2007
Accepted on: 23/01/2007

Runak M. Abdullah

College of Sciences
University of Suleimani

ABSTRACT

A new family of CG-algorithms for large-scale unconstrained optimization is introduced in this paper using the spectral scaling for the search directions, which is a generalization of the spectral gradient method proposed by Raydan [14].

Two modifications of the method are presented, one using Barzilai line search, and the others take \( \alpha = 1 \) at each iteration (where \( \alpha \) is step-size). In both cases tested for the Wolfe conditions, eleven test problems with different dimensions are used to compare these algorithms against the well-known Fletcher–Reeves CG-method, with obtaining a robust numerical results.

Keywords: Unconstrained optimization, spectral conjugate gradient method, inexact line search.

1. Introduction

Unconstrained optimization is one of the fundamental problems of numerical analysis with numerous applications.

The problem is the following:
For a function $f : R^n \to R$ and an initial point $x_0$, find a point $x^*$ (the minimizer of $f$) which minimizes the function $f(x)$, i.e.

$$\min_{x \in R^n} f(x) \quad \text{...(1)}$$

Usually $x^*$ exists and is locally unique. It is assumed that $f$ is continuously differentiable for all $k$ where $k$ is the number of iterations. Methods for unconstrained optimization are generally iterative methods in which the user typically provides an initial estimate $x_0$ of $x^*$ with possibly some additional information. A sequence of iterates $\{x_k\}$ is then generated according to some algorithm. Usually function values $\{f_k\}$ is monotonically decreasing ($f_k \text{ denotes } f(x_k)$).

A well-known algorithm for solving problem given in equation (1) is the Steepest Descent method first proposed by Cauchy in 1874. The iterations are made according to the following equation:

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 0, 1, \ldots \quad \text{...(2)}$$

where $d_k = -g_k$ and $\alpha_k$ is a step-size, which is obtained by carrying out an exact line search. It’s well-known that the negative gradient direction has the following optimal property (see [7]).

$$-g_k = \min_{d \in R^n, a \neq 0} \left[ f_k - f(x_k + \alpha g_k) \right] \frac{1}{\alpha} \quad \text{...(3)}$$

Despite the simplicity of the method and the optimal property (3), the Steepest Descent method converges slowly and is badly affected by ill-conditioning (see [9] or [15]).

In 1988, a paper by Barzilai and Borwein [5] proposed a Steepest Descent method (the BB method) that uses a different strategy for choosing the step-size $\alpha_k$ along the negative gradient direction which is obtained from two point approximation to the secant equation underlying Quasi-Newton methods.

Considering $H_k = y_k I_{n \times n}$ as an approximation to the Hessian of $f$ at $x_k$, they choose $\gamma_k$ such that $H_k$ such that

$$H_k = \arg \min \|H - y_k\|_2,$$

where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$, yielding (see[2] or [5]),

$$\gamma_k \equiv \frac{s_k^T y_k}{s_k^T s_k} \quad \text{...(4)}$$
with these, the method of Barzilai and Borwein is given by the following iterative scheme:

$$x_{k+1} = x_k - \alpha_k g_k$$

where $$\alpha_k = \frac{1}{\gamma^{BB}}$$

The scalar $$\gamma^{BB}$$ has been already used as scaling factor in the Quasi-Newton algorithms or Conjugate Gradient algorithms (see [4] and [11]).

The BB method has been shown to converge [14] and it’s convergence is linear [13], despite at these advances of BB method on quadratic functions, still there are many open questions about this method on non-quadratic functions although Fletcher [9] shows that the method be very low on some test functions.

In recent paper Abbo [1] proposed a modification of BB by the following way [1].

Let $$G_k = \gamma^{BB}_k I_{n \times n}$$

where I is the identity matrix as an approximation of Hessian matrix $$G_k$$, from convex combination of forward and backward Euler’s scheme

$$x_{k+1} = x_k - h_k[(1 - \varepsilon)g_k + \varepsilon g_{k+1}], \ 0 \leq \varepsilon \leq 1, \ h \text{ is a step-size}$$

and using Taylor’s series for $$g(x)$$ about $$x_{k+1}$$, i.e.

$$g_{k+1} = g_k + G_k s_k + o(\|s\|^2)$$

2. Conjugate Gradient Method (CG-Methods)

Conjugate Gradient Methods depend on the fact that for quadratic function, if we search along a set of n mutually conjugate directions $$d_k$$, $$k = 1, 2, ... , n$$, then we will find the minimum in at most n steps if line searches are exact. Moreover, if we generate this set of directions by known gradients, then each direction can be simply expressed as

$$d_0 = -g_0$$

$$d_{k+1} = -g_{k+1} + \beta_k d_k$$

where $$\beta_k$$ can be calculated by

$$\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$$
\[ \beta_{\text{perry}} = \frac{(y_k - s_k)^T g_{k+1}}{s_k^T y_k} \] \quad \text{...(11)}

All these \( \beta \)'s are equivalent on quadratic function with exact line searches and starting with steepest descent direction, but when extended to
general non-linear functions, the conjugate gradient algorithm with different \( \beta \) are quite different in efficiency. Formula (11) gives better algorithms than (10) in practice, a reason for this is given by Powell [13]. One of the reasons for the inefficiency of CG-method is that none of the \( \beta \) in (10) and (11) takes into consideration the effect of inexact line searches [10]. To overcome this drawback some authors proposed the so called spectral conjugate gradient methods (see for example [3],[6]).

Birgin and Martinez in [6] introduced an spectral conjugate gradient (SCG), in which the search directions are generated by
\[
d_k = -\theta_k g_k \quad k = 0
\]
\[d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k s_k \] \quad \text{...(12)}

where \( \theta_{k+1} = \frac{s_k^T s_k}{s_k^T y_k} \) \quad \text{...(13)}

and \( \beta_k = \frac{(\theta_k y_k - s_k)^T g_{k+1}}{s_k^T y_k} \) \quad \text{...(14)}

For if \( \theta_k = 1 \) this formula was introduced by Perry in [12], if we assume that\[ s_j^T g_{j+1} = 0 \quad j = 0,1,...,k \] then
\[ \beta_k = \frac{\theta_k y_k^T g_{k+1}}{\alpha_k \theta_k g_k^T g_k} \] \quad \text{...(15)}

Finally, assuming that the successive gradients are orthogonal, we obtain the generalization of FR formula:
\[ \beta_k = \frac{\theta_k g_{k+1}^T g_k}{\alpha_k \theta_k g_k^T g_k} \] \quad \text{...(16)}

In fact, SCG algorithm is a generalization of the Raydan [14] spectral gradient algorithm defined by
\[ d_k = -\theta_k g_k \] \quad \text{...(17)}
where \( \theta \) as in (13).
3. Outlines of the spectral CG-algorithm algorithm

Let \( x_0 \in \mathbb{R}^n \) , \( d_0 = -g_0 \) , \( k = 0 \) , \( \alpha_0 = 1 \)

Step(1) : if \( g_k = 0 \) stop, otherwise go to step(2)

Step(2) : compute
\[
\alpha_k = \frac{\alpha_{k-1} \|d_{k-1}\|}{\|d_k\|} \quad \text{(18)}
\]
such that Wolfe-condition is satisfied and hence a new \( x_{k+1} \) is computed

Step(3) : compute \( \theta_{k+1} \) by (13) and \( \beta_k \) by (15) or (16) and define
\[
d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k s_k
\]

Step(4) : If \( d_k^T g_{k+1} \leq -10^{-3} \|d_k\| \|g_{k+1}\| \) \quad \text{(19)}
then set \( d_{k+1} = d_k \) else \( d_{k+1} = -\theta g_{k+1} \)

Step(5) : \( k=k+1 \) go to step(1)

4. New family of SCG methods (NSCG say)

In [10] Birgin gives a nice comparison by asking the following questions:

1- Is the choice (13) better than \( \theta = 1 \)?
2- Which is the best choice of \( \beta_k \) among (15) and (16)?
3- Which is the best choice of \( \alpha_k \)?

According to these inquiries let us consider the following:

From the last term in (7) and substituting in (6) we obtain
\[
x_{k+1} - x_k = -h_k \{ (1-\varepsilon) g_k + \varepsilon (g_k + G_k s_k) \}
\]
\[
s_k = -h_k \{ g_k + \varepsilon G_k s_k \}
\]
\[
s_k + \partial h_k G_k s_k = -h_k g_k
\]
\[
(I + \partial h_k G_k) s_k = -h_k g_k
\]
\[
\frac{x_{k+1} - x_k}{h_k} = -(I + \partial h_k G_k)^{-1} g_k \quad \text{(20)}
\]

Let \( L_k = \frac{\| g_{k+1} - g_k \|^2}{\|x_{k+1} - x_k\|^2} \), Lipschitz constant, let \( G_k = \lambda_k I \) where \( I \) is \( n \times n \) identity matrix and put \( h_k = L_k \) in (20)
\[
x_{k+1} - x_k = -L_k \{ I + L_k \partial \lambda_k I \}^{-1} g_k
\]
\[ \frac{1}{L_k} (x_{k+1} - x_k) = - \left[ I + \varepsilon \frac{y^T y}{s^T s} \right]^{-1} g_k \]

\[ \frac{1}{L_k} s_k = - \left[ \frac{s_k^T s_k}{s_k^T s_k + \varepsilon s^T y} \right] g_k \]

\[ d_k = - \frac{s_k^T s_k}{s_k^T s_k + \varepsilon s^T y} g_k \]

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

where \( \theta = \frac{s^T s}{s^T s + \varepsilon s^T y} \) \( \cdots (21) \)

From (21) it is clear that setting \( \varepsilon = 0 \) this gives \( \theta = \frac{s^T s}{s^T s} = 1 \), this will answer one of the inquiries of Birgin. Also taking \( \varepsilon = 1 \) will give \( \theta = \frac{s^T s}{s^T s + s^T y} \). To answer the 2\(^{nd} \) inquiry, it is clear that \( \beta_k \) in (14) is very effective since the line search which is used in this paper is not exact. To answer the 3\(^{rd} \) inquiry we suggest a new hybrid computations for the scalar \( \alpha \) as shown in step(2) from the new algorithm.

We are going to list outlines of the new proposed algorithm (NSCG).

4.1 Outline of the algorithm (NSCG)

Let \( x_0 \in \mathbb{R}^n \), \( 0 < \sigma < \gamma < 1 \), \( d_0 = -g_0 \), \( k = 0 \)

Step(1): if \( g_k = 0 \) stop, else go to step(2)

Step(2): First compute \( \alpha_k = 1 \) and second compute

\[ \alpha_k = \begin{cases} 1 & k = 0 \\ \alpha_{k-1} [d_{k-1}] & k > 0 \end{cases} \]

Such that \( f(x_k + \alpha_k d_k) \leq f(x_k) + \sigma \alpha_k g_k^T d_k \) \( \cdots (22) \)

And \( g_{k+1}^T d_k \geq \gamma g_k^T d_k \) \( \cdots (23) \)

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

Step(3) : compute \( \theta \) by (21) and \( \beta_k \) by (16) and define
A new family of spectral CG-algorithm

\[ d_{k+1} = -\theta_{k+1}g_{k+1} + \beta_k s_k \]

Step (4): If \[ d^T_k g_{k+1} \leq -10^{-3}\|d_k\|\|g_{k+1}\| \]
then \[ d_{k+1} = d_k \] else \[ d_{k+1} = -\theta_{k+1}g_{k+1} \]

Step (5): \( k = k+1 \) go to step (1)

4.2 Some theoretical results

4.2.1 Theorem:
If \( \alpha_k \) satisfies Wolf condition defined by (22) and (23) then the search direction will be descent, i.e. \( y^T_k s_k > 0 \).

For proof see [5].

4.2.2 Theorem:
Suppose that \( f \) is bounded below in \( \mathbb{R}^n \) and that \( f \) is continuously differentiable in neighborhood of the level set \( L = \{ x : f(x) \leq f(x_0) \} \). Assume also that the gradient \( g_k \) is Lipschitz continuous i.e. there exists a constant \( c > 0 \) s.t. \( \| g(x) - g(y) \| \leq c\| x - y \| \) \( \forall x, y \in \mathbb{R}^n \).

Consider any iteration of the form \( x_{k+1} = x_k + \alpha_k d_k \) where \( \alpha = 1 \) and if \( d_k = -g_k \) and \( \alpha_k \) satisfies Wolf conditions defined in (22) and (23) then \( \lim_{k \to \infty} \| g_k \| = 0 \).

Proof: From equation (22) we have \( (g_{k+1} - g_k)^T d_k \geq (\sigma_2 - 1)\| g_k \|^2 \) \( d_k \) \( \ldots(24) \) on the other hand, the lipchitz condition \( (g_{k+1} - g_k)^T d_k \leq \alpha_k \| d_k \|^2 \) \( \ldots(25) \) from (24) and (25) we get \( \alpha_k \geq \left( \frac{\sigma_2 - 1}{c} \right) \frac{(g_k^T d_k)^2}{\| d_k \|^2} \) \( \ldots(26) \)

using equations (22) and (26) we have \( f_{k+1} \leq f_k + \sigma_1 \left( \frac{\sigma_2 - 1}{c} \right) \frac{(g_k^T d_k)^2}{\| d_k \|^2} \) \( \ldots(27) \)

now using the relation \( \| g_k \| \| d_k \| \cos \gamma_k = -g_k^T d_k \) where \( \gamma_k \) is the angle between \( g_k \) and \( d_k \).

then the equation (27) can be written as \( f_{k+1} \leq f_k + \| g_k \| \cos \gamma_k \) \( \ldots(28) \)

where \( r = \frac{\sigma_1(\sigma_2 - 1)}{c} \) and \( \sigma_1, \sigma_2 \in (0, \frac{1}{2}) \)
summing the expression in equation (28) and since \( f \) is bounded below, we obtain

\[
\sum \cos^2 \gamma_k \|g_k\|^2 < \infty
\]

assuming that \( \cos^2 \gamma_k > \delta > 0 \) for all \( k \), then we conclude that

\[
\lim_{k \to \infty} \|g_k\| = 0
\]

(29)

(30)

5. Numerical results

The comparative test involves eleven well-known standard test functions (given in the appendix) with different dimensions. The results are given in the Table(1) is specifically quoting the number of function evaluations (NOF). All programs are written in FORTRAN 90 language and for all cases the stopping criterion is taken to be \( \|g_{k+1}\| < 1 \times 10^{-5} \). The results are given in table (1):

<table>
<thead>
<tr>
<th>Test Function</th>
<th>New (SCG)</th>
<th>Standard SCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended Trigonometric</td>
<td>1000</td>
<td>44</td>
</tr>
<tr>
<td>Trigonometric</td>
<td>5000</td>
<td>101</td>
</tr>
<tr>
<td>Extended</td>
<td>10000</td>
<td>86</td>
</tr>
<tr>
<td>Extended Rosenbrock</td>
<td>1000</td>
<td>59</td>
</tr>
<tr>
<td>Extended</td>
<td>5000</td>
<td>60</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>10000</td>
<td>64</td>
</tr>
<tr>
<td>Perturbed Quadratic</td>
<td>1000</td>
<td>662</td>
</tr>
<tr>
<td>Extended</td>
<td>5000</td>
<td>1239</td>
</tr>
<tr>
<td>Extended</td>
<td>10000</td>
<td>1504</td>
</tr>
<tr>
<td>Extended</td>
<td>1000</td>
<td>1619</td>
</tr>
<tr>
<td>Extended</td>
<td>5000</td>
<td>873</td>
</tr>
<tr>
<td>Raydan 1</td>
<td>10000</td>
<td>2327</td>
</tr>
<tr>
<td>Generalized Tridiagonal-1</td>
<td>1000</td>
<td>307</td>
</tr>
<tr>
<td>Generalized Tridiagonal-1</td>
<td>5000</td>
<td>486</td>
</tr>
<tr>
<td>Diagonal 2</td>
<td>10000</td>
<td>522</td>
</tr>
<tr>
<td>Generalized Tridiagonal-1</td>
<td>2000</td>
<td>54</td>
</tr>
<tr>
<td>Generalized Tridiagonal-1</td>
<td>5000</td>
<td>429</td>
</tr>
<tr>
<td>Generalized Tridiagonal-1</td>
<td>10000</td>
<td>1324</td>
</tr>
<tr>
<td>Extended Exponential</td>
<td>3000</td>
<td>1769</td>
</tr>
<tr>
<td>Exponential Terms</td>
<td>4000</td>
<td>1911</td>
</tr>
</tbody>
</table>

Note: For some cases, the values are not provided due to computational limitations or the program's failure to converge.
A new family of spectral CG-algorithm

<table>
<thead>
<tr>
<th></th>
<th>NOF</th>
<th>NOG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generalized PSC1</td>
<td>5000</td>
<td>#</td>
</tr>
<tr>
<td>Extended Powell</td>
<td>1000</td>
<td>172</td>
</tr>
<tr>
<td></td>
<td>3000</td>
<td>146</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>158</td>
</tr>
<tr>
<td>Extended Maratos</td>
<td>1000</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>6000</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>98</td>
</tr>
<tr>
<td>Extended Wood</td>
<td>1000</td>
<td>184</td>
</tr>
<tr>
<td></td>
<td>5000</td>
<td>192</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>178</td>
</tr>
<tr>
<td>Total</td>
<td>16076</td>
<td>9170</td>
</tr>
</tbody>
</table>

From Table (1) taking the standard Birgin (SCG) as %100 NOF we can get the following values.

**Table (2)**

<table>
<thead>
<tr>
<th>NOF+NOG</th>
<th>$\alpha_k = \frac{\alpha_{k-1} |d_{k-1}|}{|d_k|}$</th>
<th>$\alpha_k = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard SCG</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>New SCG</td>
<td>64%</td>
<td>56%</td>
</tr>
</tbody>
</table>

From table (2) it is clear that the new proposed algorithm with it’s both versions has an improvements of about (33-36)% NOF according to our selected number of test functions.
6. Appendix:

All the test functions used in this paper are from general literature:

1. Extended Trigonometric Function
   \[ f(x) = \sum_{i=1}^{n} \left[ (n - \sum_{j=1}^{n} \cos x_j) + i(1 - \cos x_i) \right]^2, \quad x_0 = [0.2, 0.2, \ldots, 0.2]^T \]

2. Extended Rosenbrock Function
   \[ f(x) = \sum_{i=1}^{n/2} c(x_{2i} - x_{2i-1})^2 + (1 - x_{2i-1})^2, \quad x_0 = [-1.2, 1, \ldots, -1.2, 1]^T \]

3. Perturbed Quadratic Function
   \[ f(x) = \sum_{i=1}^{n} x_i^2 + \frac{1}{100} \sum_{i=1}^{n} x_i^2, \quad x_0 = [0.5, 0.5, \ldots, 0.5]^T \]

4. Raydan1 Function
   \[ f(x) = \sum_{i=1}^{n} \left( \exp(x_i) - x_i \right), \quad x_0 = [1, 1, 1]^T \]

5. Diagonal2 Function
   \[ f(x) = \sum_{i=1}^{n} \left( \exp(x_i) - \frac{x_i}{i} \right), \quad x_0 = [1/1, 1/2, \ldots, 1/n]^T \]

6. Generalized Tridigonal-1 Function
   \[ f(x) = \sum_{i=1}^{n-1} (x_i + x_{i+1} - 3)^2 + (x_i - x_{i+1} + 1)^4, \quad x_0 = [2, 2, \ldots, 2]^T \]

7. Extended Three Exponential Terms
   \[ f(x) = \sum_{i=1}^{n/2} \left( \exp(x_{2i-1}) + 3x_{2i} - 0.1 \right) + \exp(x_{2i} - 3x_{2j} - 0.1) + \exp(-x_{2j-1} - 0.1) \]
   \[ x_0 = [0.5, 0.5, \ldots, 0.5]^T \]

8. Generalized PSC1 Function
   \[ f(x) = \sum_{i=1}^{n-1} \left( x_i^2 + x_{i+1}^2 + x_{i+1}x_{i+1} \right)^2 + \sin^2(x_i) + \cos^2(x_i), \quad x_0 = [3, 0.1, \ldots, 3, 0.1]^T \]

9. Extended Powell Function
   \[ f(x) = \sum_{i=1}^{n/4} \left( x_{4i-3} + 10x_{4i-2} \right)^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4 \]
   \[ x_0 = [3, -1, 0.1, \ldots, 3, -1, 0.1]^T \]

10. Extended Maratos Function
A new family of spectral CG-algorithm

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

11. Extended Wood Function

\[ f(x) = \sum_{i=1}^{n/4} 100(x_{4i-3}^2 - x_{4i-1}^2)^2 + (x_{4i-3} - 1)^2 + 90(x_{4i-1}^2 - x_{4i})^2 + (1 - x_{4i-1})^2 + 10.1(x_{4i-2} - 1)^2 + (x_{4i} - 1) + 19.8(x_{4i-2} - 1)(x_{4i} - 1) \quad , \quad x_0 = [-3, -1, -3, -1, \ldots, -3, -1, -3, -1]^T \]

79
REFERENCES


