Steepest descent method implementation on unconstrained optimization problem using C++ program

H Napitupulu$^{1,*}$, Sukono$^1$, I Bin Mohd$^2$, Y Hidayat$^3$, S Supian$^1$

$^1$Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran, Indonesia
$^2$Institute for Mathematical Research (INSPEM), Universiti Putera Malaysia, Selangor, Malaysia
$^3$Department of Statistics, Faculty of Mathematics and Natural Sciences, Universitas Padjadjaran, Indonesia

*Email: napitupuluherlina@gmail.com

Abstract. Steepest Descent is known as the simplest gradient method. Recently, many researches are done to obtain the appropriate step size in order to reduce the objective function value progressively. In this paper, the properties of steepest descent method from literatures are reviewed together with advantages and disadvantages of each step size procedure. The development of steepest descent method due to its step size procedure is discussed. In order to test the performance of each step size, we run a steepest descent procedure in C++ program. We implemented it to unconstrained optimization test problem with two variables, then we compare the numerical results of each step size procedure. Based on the numerical experiment, we conclude the general computational features and weaknesses of each procedure in each case of problem.

1. Introduction
Consider the following nonlinear unconstrained minimization problem

$$\text{find } x^* \in \mathbb{R}^n \text{ such that } f \left( x^* \right) = \min_{x \in \mathbb{R}^n} f \left( x \right)$$  \hspace{1cm} (1)

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Steepest descent method is the simplest gradient method for solving unconstrained optimization which is designed by Cauchy in 1847. This gradient method search along the negative gradient function, can ensure a reduction of the objective function as long as the current iterate point is not a stationary point ([1]). Many researches have been done which is concern in finding the better and more appropriate step size procedure which affecting to the significance of minimizing the objective functions. In this paper we give some reviews the properties of steepest descent method, its development, and compare the numerical results of steepest descent step size from some literatures on solving global optimization problem. We run a steepest descent in C++ program to test the performance of tested step size procedure, then compare the numerical results obtained of program execution. Based on the numerical experiment, the features as well as the weaknesses of each step size procedure in each case of problem are concluded.
2. Steepest Descent Method

Steepest descent method is the simple minimizing gradient method for solving nonlinear equations since it is based on the linear approximation of Taylor series

\[ f(x + \delta) = f(x) + \nabla f(x)^T \delta \]  

(1)

The term \( \nabla f(x)^T \delta \) is the directional derivatives of \( f \) at \( x \). The step \( \delta \) is a descent direction if the directional derivative is negative, which guarantees that the function \( f \) could be reduced along this direction. Furthermore, the step \( \delta \) needs to be chosen to make the linear directional derivative \( \nabla f(x)^T \delta \) as negative as possible, which gives the maximum reduction of \( f \) but no consuming too much time in making a choice.

Assume that a function \( f(x) \) is continuous in the neighborhood of point \( x \), \( d = -g \) is the steepest descent direction at point \( x \) and change \( \delta \) in \( x \) given by \( \delta = \alpha d \).

A small positive constant \( \alpha \) is called step size, which its selection makes an effect to the reduction value of \( f(x) \). By solving one dimensional global optimization problem

\[ \min_{\alpha > 0} \phi(\alpha) = f(x + \alpha d) \]  

(2)

the maximum reduction in \( f(x) \) can be obtained.

The procedure of steepest descent iteration is performed by formula

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

(3)

Starting with an initial point \( x_0 \), a direction \( d_0 = -g_0 \) and the step size \( \alpha_0 \) that minimizes \( f(x_0 + \alpha_0 d_0) \) can be determined so that the next point \( x_1 = x_0 + \alpha_0 d_0 \) is obtained. The procedure in (1) is repeated until the convergence is achieved or the value \( \|\alpha_k d_k\| \) is sufficiently small. The algorithm of steepest descent is presented as follows.

```
ALGORITHM STEEPEST DESCENT

given:
    starting point \( x_0 \in \text{domain } f \)

set \( k := 0 \)
repeat
    1. Compute steepest descent direction \( d_k = -g_k \)
    2. Update step size \( \alpha_k = \min_{\alpha > 0} \phi(\alpha) = f(x_k + \alpha d_k) \)
    3. Update \( x_{k+1} := x_k + \alpha_k d_k \)

until convergence or stopping criterion is satisfied
```

3. Steepest Descent Step Size

The step size in steepest descent method plays the important role in obtaining the minimizing the objective function. The step size of each iteration \( \alpha_k \), must be effectively chosen to give a significant reduction of function value, while the efficiently timing of choosing it also be optimized.

The early step size procedure of steepest descent is considered as step size which obtained by exact line search,

\[ \alpha_k = \arg \min_{\alpha} \{ f(x_k + \alpha(-g_k)) \} \]  

(4)

But if the steepest descent direction is used with exact line search step size would bring to the zigzag behavior which make the convergence is very slowly. Akaike ([2]) analyzing that by this search
procedure the converges is linearly and badly affected by ill-conditioning, by shown that the two asymptotic directions are in a two dimensional sub space spanned by two eigen vector of the Hessian matrix. Greenstadt ([3]) studied the efficiency of steepest descent method and shown the bound of the ratio between the reductions obtained by Cauchy step and by the Newton’s method, and the rate of convergence of steepest descent was proved by Forsythe ([4]).

Besides using exact line search, step size $\alpha_k$ could be computed by type of inexact line search methods, for instance Armijo condition ([5]), Goldstein condition ([6]), Wolfe condition ([7]), Powell ([8]), Dennis and Schnabel ([9]), Fletcher ([10]), Potra and Shi ([11]), Lemaréchal ([12]), Moré and Thüente ([13]) and the others. Steepest descent method with mentioned conditions is always convergent theoretically, which is will not terminate unless the stationary point is found. However, step size computed by those inexact line search procedure also appearing the bad zigzag behavior as well as in exact line search procedure.

Considering the weakness of existing step sizes, Barzilai and Borwein ([10]) proposed the the step size along negative gradient direction from a two point approximation to the secant equation from quasi-Newton’s method so called BB method. If the dimension is two, Barzilai and Borwein established R-superlinear convergence result for the method and their analyses indicated that the convergence rate is faster. For the general $n$-dimensional strictly convex quadratic function, Raydan ([14]) proved that the two-point step size gradient method is globally convergent, and the convergence rate is $R$-linear (Dai and Liao ([15]). For the non-quadratic case, Raydan ([14]) incorporated the globalization scheme of the two point step size gradient method using the traditional technique of non-monotone line search by Grippo et al ([16]). The resulted algorithm is competitive and sometimes preferable to several famous conjugate gradient algorithm for large scale unconstrained optimization. Due it simplicity and numerical efficiency the two-point step size gradient method has received many studies. They have been successfully applied to obtaining the local minimizers of large scale real problems ([17]). The algorithm of Raydan ([15]) was further generalized by Birgin et al ([18]) for the minimization of differentiable function on closed convex sets, yielding an efficient projected gradient method. Efficient projected algorithm based on BB-like methods have also been designed (Serafini et al ([19]) and Dai and Fletcher([20]) for special quadratic programs arising from training support vector machine, that has a singly linear constraint in addition to box constraints. The BB method has also received much attention in finding sparse approximation solution to large undetermined linear systems of equations from signal/ image processing and statics for example in Wright et al ([21]). However Fletcher ([22]) shown that for some problems of non-quadratic function this method may very slow.

There are many existing step sizes modification, some of them are easy to be applied but some are satisfied the complicated algorithm. Following are some step size procedure which is simple to be applied to steepest descent method.

| No | Step Size | Remark |
|----|-----------|--------|
| 1 | $\alpha_k = \frac{g^T_k g_k}{g^T_k H_k g_k}$ | Step size method by Cauchy (1847) which is computed by exact line search (C step size). |
| 2 | Given $s > 0, \beta, \sigma \in (0, 1)$, $\alpha_k = \max \left\{ s, s \beta, s \beta^2, \ldots \right\}$ such that $f(x_k + \alpha_k d_k) \leq f(x_k) + \sigma \alpha_k g^T_k d_k$ | This procedure of step size using inexact line search so called Armijo’s line search (A step size). |
| 3 | Given $\beta, \sigma \in (0, 1), \tilde{\alpha}_0 = 1$ $\alpha_k = \beta \tilde{\alpha}_k$ | This procedure is so called Backtracking line search (B step size), this inexact line search is use just the sufficient decrease |
such that
\[ f(x_k + \alpha_k d_k) \leq f(x_k) + \sigma \alpha_k g_k^T d_k \]
condition to terminate the line search procedure.

4
\[
\alpha_k = \frac{s_k T \sqrt{y_{k-1}}}{\|y_{k-1}\|^2} \quad (BB1)
\]
and
\[
\alpha_k = \frac{s_k T \sqrt{y_{k-1}}}{s_{k-1} T \sqrt{y_{k-1}}} \quad (BB2)
\]
where
\[ s_{k-1} = x_k - x_{k-1} \]
\[ y_{k-1} = g_k - g_{k-1} \]
These two point step sizes are method of non-line search procedure which is namely after the inventors, Barzilai and Borwein’s Formula. These formula reduce the value of objective function and gradient and improved the convergence from linear to R–superlinear.

5
\[
\alpha_k = \frac{g_k T g_k \alpha_{k-1}^2}{2\left(f(x_k + \alpha_{k-1} d_k) - f(x_k) + \alpha_{k-1} g_k^T g_k\right)}
\]
This step size is so called elimination line search (EL step size), which is estimate the step size without compute the Hessian.

4. Numerical Experiment
In this section, several step size procedure of the steepest descent method shown in Table 1 are tested computationally. The programming is using Visual C++ 6.0, which is applied to 14 testing function of global optimization given in the following.

1. Beale Function
\[
\min f(x_1, x_2) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2
\]
\[
s.t. -4.5 \leq x_1, x_2 \leq 4.5, \quad x_0 = (1, 0)
\]

2. Bohachevsky I Function
\[
\min f(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) - 0.4 \cos(4\pi x_2) + 0.7
\]
\[
s.t. -50 \leq x_1, x_2 \leq 50, \quad x_0 = (0.3, 0.3)
\]

3. Bohachevsky II Function
\[
\min f(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) \cos(4\pi x_2) + 0.3
\]
\[
s.t. -50 \leq x_1, x_2 \leq 50, \quad x_0 = (0.1, 0.1)
\]

4. Booth Function
\[
\min f(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2
\]
\[
s.t. -10 \leq x_1, x_2 \leq 10, \quad x_0 = (-4, 4)
\]

5. Matyas Function
\[
\min f(x_1, x_2) = 0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2
\]
\[
s.t. -3 \leq x_1, x_2 \leq 3, \quad x_0 = (2, 2)
\]

6. Rosenbrock Simplified Function
\[
\min f(x_1, x_2) = 0.5\left(x_1^2 - x_2\right)^2 + (x_1 - 1)^2
\]
\[
s.t. -3 \leq x_1, x_2 \leq 3, \quad x_0 = (-1.6, 0.9)
\]
7. Six Hump Back Camel Function

\[
\min f(x_1, x_2) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 - x_1x_2 - 4x_2^2 + 4x_2^4
\]

s.t. \(-3 \leq x_1, x_2 \leq 3, \quad x_0 = (2, -1)\)

8. Rastrigin Function

\[
\min f(x_1, x_2) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)
\]

s.t. \(-2 \leq x_1, x_2 \leq 2, \quad x_0 = (-1, 1)\)

9. Three Hump Back Camel Function

\[
\min f(x_1, x_2) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} - x_1x_2 + x_2^2
\]

s.t. \(-3 \leq x_1, x_2 \leq 3, \quad x_0 = (-2, -1)\)

10. Treccani Function

\[
\min f(x_1, x_2) = x_1^4 + 4x_1^3 + 4x_1^2 + x_2^2
\]

s.t. \(-3 \leq x_1, x_2 \leq 3, \quad x_0 = (4, 2)\)

11. Two Dimensional Function \(c = 0.05\)

\[
\min f(x_1, x_2) = \left(1 - 2x_2 + 0.05 \sin(4\pi x_2) - x_1\right)^2 + \left(x_2 - 0.5 \sin(2\pi x_1)\right)^2
\]

s.t. \(-10 \leq x_1, x_2 \leq 10, \quad x_0 = (0.5, 0.5)\)

12. Two Dimensional Function \(c = 0.2\)

\[
\min f(x_1, x_2) = \left(1 - 2x_2 + 0.2 \sin(4\pi x_2) - x_1\right)^2 + \left(x_2 - 0.5 \sin(2\pi x_1)\right)^2
\]

s.t. \(-10 \leq x_1, x_2 \leq 10, \quad x_0 = (2, -2)\)

13. Two Dimensional Function \(c = 0.5\)

\[
\min f(x_1, x_2) = \left(1 - 2x_2 + 0.5 \sin(4\pi x_2) - x_1\right)^2 + \left(x_2 - 0.5 \sin(2\pi x_1)\right)^2
\]

s.t. \(-10 \leq x_1, x_2 \leq 10, \quad x_0 = (0, 0)\)

14. Goldstein and Price Function

\[
\min f(x_1, x_2) = \left(1 + (x_1 + x_2 + 1) \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right) \left(30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right)
\]

s.t. \(-3 \leq x_1, x_2 \leq 3, \quad x_0 = (-1, -1)\)

By referring to [23] for Armijo procedure (\(A\)), \(\alpha = 1\), \(\beta = 0.75\), \(\sigma = 0.38\), and for Backtracking procedure \(\sigma = 0.0001\). The initial step size for Barzilai-Borwein step size 1 (BB1) and Barzilai-Borwein step size 2 (BB2) is 0.01, initial step size for elimination line search is \(\alpha_0 = 0.1\). The comparison of Numerical result will be based on: time execution, number of total iteration, total percentage of function, gradient and Hessian evaluation and the most decreased value of objective function obtained. Stopping criteria are \(\|d_k\| \leq 10^{-6}\) or \(\|d_k\| \leq 10^{-6}\). Following are the abbreviations used in of numerical results of Table 2.
Abbreviation in column one:
C = Cauchy step size
A = Armijo step size
B = Backtracking step size
BB1 = Barzilai & Borwein step size 1
BB2 = Barzilai & Borwein step size 2
EL = Elimination line search step size

Abbreviation in row one:
Iter = percentage of the total number of iteration until termination
Time = percentage of the total running time until the termination in second
Ft = percentage of the total number of function evaluation until termination
Gt = percentage of the total number of gradient evaluation until termination
Ht = percentage of the total number of Hessian evaluation until termination

| No. | Step | x | f   | Time | Iter % | Ft % | Gt % | Ht % |
|-----|------|---|-----|------|--------|------|------|------|
| 1.  | C    | (2.99995, 0.499987) | 4.73368e-10 | 0.009 | 22.06  | 0    | 18.80 | 100.0 |
|     | A    | (2.9999, 0.499975)  | 1.6441e-9    | 0.005 | 24.33  | 36.35 | 22.54 | 0     |
|     | B    | (2.9999, 0.499974)  | 1.69506e-9   | 0.004 | 29.79  | 41.21 | 25.55 | 0     |
|     | BB1  | (2.99985, 0.499963) | 3.44039e-9   | 0     | 2.97   | 0    | 7.59  | 0     |
|     | BB2  | (2.99987, 0.499968) | 2.7321e-9    | 0.002 | 4.53   | 0    | 11.58 | 0     |
|     | EL   | (2.9999, 0.499974)  | 1.74086e-9   | 0.004 | 16.29  | 22.42 | 13.90 | 0     |

| 2.   | Step | x | f   | Time | Iter % | Ft % | Gt % | Ht % |
|------|------|---|-----|------|--------|------|------|------|
|      | C    | (0.360538, 0.266918) | 1.65345      | 0     | 15.78  | 0    | 7.51  | 100.0 |
|      | A    | (1.16965e-6, 0.469363) | 0.469528    | 0.001 | 19.73  | 62.74 | 28.32 | 0     |
|      | B    | (1.94405e-6, 0.469363) | 0.469528    | 0     | 13.15  | 20.91 | 9.82  | 0     |
|      | BB1  | (-6.15555e-7, 1.6234e-7) | 6.31717e-12 | 0     | 13.15  | 20.91 | 9.82  | 0     |
|      | BB2  | (9.47959e-7, 0.469363) | 0.469528    | 0     | 17.10  | 0    | 22.54 | 0     |
|      | EL   | (6.95877e-7, 0.469363) | 0.469528    | 0     | 15.78  | 16.33 | 7.51  | 0     |

| 3.   | Step | x | f   | Time | Iter % | Ft % | Gt % | Ht % |
|------|------|---|-----|------|--------|------|------|------|
|      | C    | (0.618401, 2.71507e-7) | 0.412619    | 0     | 15.71  | 0    | 8.00  | 100.0 |
|      | A    | (1.258e-6, 9.10494e-8) | 2.28987e-11 | 0     | 14.28  | 55.21 | 30.66 | 0     |
|      | B    | (8.44811e-7, 1.9562e-7) | 7.66054e-12 | 0.001 | 12.85  | 34.35 | 19.33 | 0     |
|      | BB1  | (8.44811e-7, 1.9562e-7) | 1.12145e-11 | 0.001 | 12.85  | 0    | 19.33 | 0     |
|      | BB2  | (2.70968e-7, -5.34494e-8) | 1.12599e-12 | 0     | 12.85  | 0    | 18.00 | 0     |
|      | EL   | (-5.56201e-7, 4.30077e-7) | 9.18932e-12 | 0     | 11.42  | 10.42 | 6.00  | 0     |

| 4.   | Step | x | f   | Time | Iter % | Ft % | Gt % | Ht % |
|------|------|---|-----|------|--------|------|------|------|
|      | C    | (0.99999, 3.00001) | 7.93254e-11 | 0     | 27.33  | 0    | 19.92 | 100.0 |
|      | A    | (0.99999, 3.00001) | 6.89661e-11 | 0.001 | 27.09  | 49.89 | 21.53 | 0     |
|      | B    | (-2.82681e-10, 0.459324) | 0.459962   | 0.001 | 24.38  | 42.82 | 18.50 | 0     |
|      | BB1  | (0.999999, 3) | 6.12972e-13 | 0     | 2.21   | 0    | 4.80  | 0     |
|      | BB2  | (0.99997, 3) | 1.31356e-11 | 0     | 14.77  | 0    | 32.02 | 0     |
|      | EL   | (0.999999, 3) | 2.04087e-12 | 0     | 4.18   | 7.27 | 3.20  | 0     |

| 5.   | Step | x | f   | Time | Iter % | Ft % | Gt % | Ht % |
|------|------|---|-----|------|--------|------|------|------|
|      | C    | (2.44249e-15, 2.44249e-15) | 2.3863e-31  | 0.001 | 0.17   | 0    | 0.34  | 100.0 |
|      | A    | (1.7006e-5, 1.7006e-5) | 1.15682e-11 | 0.001 | 49.65  | 49.86 | 49.14 | 0     |
|      | B    | (1.7006e-5, 1.7006e-5) | 1.15682e-11 | 0.001 | 49.65  | 49.86 | 49.14 | 0     |
|      | BB1  | (2.21911e-12, 2.21911e-12) | 1.96979e-25 | 0     | 0.17   | 0    | 0.51  | 0     |
|      | BB2  | (2.21889e-12, 2.21889e-12) | 1.96939e-25 | 0     | 0.17   | 0    | 0.51  | 0     |
| Step | x | f | Time | Iter % | Ft % | Gr % | Ht % |
|------|---|---|------|--------|------|------|------|
| 6.   | C | -1.04076, 1.04076 | 0.179775 | 0 | 11.11 | 0 | 4.12 | 100.0 |
|      | A | -0.693844, 0.693844 | -1.03121 | 0 | 18.51 | 70.34 | 53.60 | 0 |
|      | B | -1.04076, 1.04076 | 0.179775 | 0 | 29.62 | 22.06 | 17.52 | 0 |
|      | BB1 | 1.04076, -1.04076 | 0.179775 | 0 | 11.11 | 0 | 9.27 | 0 |
|      | BB2 | -1.04076, 1.04076 | 0.179775 | 0 | 11.11 | 0 | 9.27 | 0 |
|      | EL | -1.04076, 1.04076 | 0.179775 | 0 | 18.51 | 7.58 | 6.18 | 0 |
| 7.   | C | (1, 1) | 8.87118e-13 | 0.001 | 28.07 | 0 | 21.30 | 100.0 |
|      | A | (0.999992, 0.999977) | 8.98588e-11 | 0.00 | 1 | 29.43 | 52.50 | 22.97 | 0 |
|      | B | (1, 1.00001) | 2.48894e-11 | 0.001 | 16.25 | 28.60 | 12.55 | 0 |
|      | BB1 | (0.999995, 0.999985) | 3.9723e-11 | 0 | 8.74 | 0 | 19.81 | 0 |
|      | BB2 | (0.999994, 0.999983) | 4.87009e-11 | 0 | 6.65 | 0 | 15.06 | 0 |
|      | EL | (0.999996, 0.999989) | 2.10425 | 0 | 10.83 | 18.89 | 8.27 | 0 |
| 8.   | C | (1.6071, -0.56865) | 2.10425 | 0 | 9.89 | 0 | 7.19 | 100.0 |
|      | A | (1.6071, -0.568655) | 2.10425 | 0.001 | 43.95 | 61.53 | 38.12 | 0 |
|      | B | (0.999992, 0.999977) | -1.03163 | 0 | 19.78 | 26.03 | 12.55 | 0 |
|      | BB1 | (0.999995, 0.999985) | 2.104 25 | 0 | 7.69 | 0 | 15.10 | 0 |
|      | BB2 | (0.999994, 0.999983) | 2.10425 | 0 | 7.69 | 0 | 15.10 | 0 |
|      | EL | (0.999996, 0.999989) | 2.10425 | 0 | 10.98 | 12.42 | 7.91 | 0 |
| 9.   | C | (-1.74755, -0.873778) | 0.298638 | 0.001 | 8.78 | 0 | 6.09 | 100.0 |
|      | A | (-1.74755, -0.873788) | 0.298638 | 0.001 | 46.86 | 68.14 | 34.34 | 0 |
|      | B | (-1.74755, -0.873772) | 0.298638 | 0 | 12.97 | 18.83 | 9.69 | 0 |
|      | BB1 | (-1.74755, -0.87378) | 0.298638 | 0 | 9.20 | 0 | 18.28 | 0 |
|      | BB2 | (-1.74755, -0.873778) | 0.298638 | 0 | 12.55 | 0 | 24.93 | 0 |
|      | EL | (-1.74755, -0.873778) | 0.298638 | 0 | 9.62 | 13.01 | 6.64 | 0 |
| 10.  | C | (7.64024e-19, 8.86385e-7) | 1.54548e-13 | 0 | 1.07 | 0 | 0.32 | 100.0 |
|      | A | (7.64024e-19, 8.86385e-5) | 7.85679e-9 | 0.003 | 67.89 | 70.55 | 19.62 | 0 |
|      | B | (2.3, 3.18205e-6) | 1.01254e-9 | 0.002 | 26.72 | 27.68 | 77.33 | 0 |
|      | BB1 | (6.26718e-8, 1.34651e-6) | 1.82881e-12 | 0 | 1.30 | 0 | 1.10 | 0 |
|      | BB2 | (6.26718e-8, 1.34651e-6) | 1.82881e-12 | 0 | 1.30 | 0 | 1.10 | 0 |
|      | EL | (2.73691, -0.787967) | 7.07142e-13 | 0 | 8.07 | 11.44 | 4.80 | 0 |
| 11.  | C | (0.402355, 0.287455) | 4.41419e-11 | 0 | 30.00 | 0 | 18.18 | 100.0 |
|      | A | (0.402355, 0.287455) | 4.30738e-11 | 0 | 31.76 | 76.92 | 23.07 | 0 |
|      | B | (3.7532e-13, 0.402127) | 5.29 | 0 | 14.20 | 4.54 | 0 |
|      | BB1 | (0.402359, 0.287451) | 6.32753e-12 | 0 | 10.00 | 0 | 17.83 | 0 |
|      | BB2 | (0.402356, 0.287453) | 1.02648e-11 | 0 | 18.82 | 0 | 33.56 | 0 |
|      | EL | (0.402357, 0.287452) | 1.42512e-14 | 0 | 4.11 | 8.87 | 2.79 | 0 |
| 12.  | C | (2.73691, -0.787967) | 0.0884257 | 0.001 | 7.76 | 0 | 4.44 | 100.0 |
|      | A | (1.59016, -0.270037) | 2.91343e-11 | 0.001 | 11.80 | 29.37 | 12.27 | 0 |
|      | B | (1.87797, -0.345587) | 2.2265e-11 | 0.001 | 40.99 | 59.17 | 24.55 | 0 |
|      | BB1 | (1.87797, -0.345586) | 3.68925e-13 | 0.001 | 19.56 | 0 | 33.62 | 0 |
|      | BB2 | (1.87797, -0.345586) | 5.19563e-12 | 0.001 | 11.80 | 0 | 20.28 | 0 |
|      | EL | (1.87797, -0.345586) | 7.07142e-13 | 0 | 8.07 | 11.44 | 4.80 | 0 |
| 13.  | C | (3.08067e-11, 3.08067e-11) | 3.79621e-23 | 0 | 0.17 | 0.26 | 0.34 | 0 |
From the numerical results of Table 2 we summarize the number of successful (ns), number of local minimizer obtained (nl), and number of failure (nf) presented in Table 3.

| Step | C | fail | A | fail | B | fail | BB1 | fail | BB2 | fail | EL | fail |
|------|---|------|---|------|---|------|------|------|------|------|----|------|
|      |   | (0.102535, 0.300418) |   | 2.8179e-10 |   | 0.517355 |   | 0.517355 |   | 0.517355 |   | 0.517355 |
|      |   | 37.16 |   | 66.66 |   | 33.88 |   | 0 |   | 0 |   | 0 |

Table 3. Summary of Numerical Results

| Step Size | ns | nl | nf |
|-----------|----|----|----|
| C         | 7  | 6  | 1  |
| A         | 10 | 4  | 0  |
| B         | 9  | 5  | 0  |
| BB1       | 10 | 3  | 1  |
| BB2       | 9  | 4  | 1  |
| EL        | 9  | 4  | 1  |

From the numerical result obtained, the A method and BB1 method are the method with the most successful among others for the case of two dimensional global optimization given. A method never failed in solving the testing problems but have 4 times go into another local minimizer. The C method is the least in obtaining the minimizer successfully with one failure. The B, BB2 and EL method reach the same level in successfully obtain the minimizer. The B method reach without any fail, but BB2 and EL have 1 fail. From this numerical experiment we can conclude that A is better than others based on the number of minimizer obtained in the testing example given in given initial condition.

5. Conclusion

The advantages and disadvantages of steepest descent method are, this method is sensitive to the initial point; this method has descent property and it is a logical starting procedure for all gradient based methods. Since $\left\| x_{k+1} - x_k \right\| = \left\| x_k \right\| \left\| \nabla f(x_k) \right\| \to 0$ as $\nabla f(x) = 0$, so $x_k$ approaches the minimizer rather slowly, in fact in a zigzag way. It is not economical to do linear searches thoroughly. All that is necessary to do is to obtain reduction in function values at successive iterates.

According to the numerical experiment of the given test function, we can conclude that in the case of two dimensional unconstrained global optimization problem given, the A and BB1 is considered as the better method among others and C method is good enough, for the case of testing example given with given initial condition.

Acknowledgments

Authors wishing to the Rector, Director of DRPMI, and Dean of FMIPA, Universitas Padjadjaran, which provides a grant program of the Academic Leadership Grant (ALG) and a grant program of Competence Research of Lecturer of Unpad (Riset Kompetensi Dosen Unpad/RKDU).
References
[1] Yuan Y 2008 Step-sizes for the gradient method In Third International Congress of Chinese Mathematicians. Part 1, 2 AMS/IP Stud. Adv. Math., 42, Pt. 1 2 785–796. Amer. Math. Soc., Providence, RI
[2] Akaike H 1959 On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method Ann, Inst. Statist. Math. Tokyo 11 pp 1-7
[3] Greenstadt J 1967 On the relative efficiencies of gradient methods Math. Comp. 21 pp 360-367
[4] Forshyte G E 1968 On the Asymptotic directions of the s-dimensional optimum gradient method Numerische Mathematik 11 pp 57-76
[5] Armijo L 1966 Minimization of function having Lipschitz continuous first partial derivatives Pacific J. of Mathematics 6 pp 1-3
[6] Goldstein A A 1965 On steepest descent. SIAM Journal on Control, vol. 3, 147-151.
[7] Wolfe P 1968 Convergence conditions for ascent methods SIAM Review 11 pp 226-235
[8] Powell M J D 1976 Some global convergence properties of a variable-metric algorithm for minimization without exact line searches SIAM-AMS Proceedings, Philadelphia, 9 pp 53-72
[9] Dennis J E and Schnabel R B 1983 Numerical Methods for Unconstrained Optimization and Nonlinear Equations Prentice-Hall Englewood Cliffs New Jersey
[10] Barzilai J and Borwein J 1988 Two-point step size gradient methods IMA J. of Numer. Anal. 8 141-148
[11] Potra F A and Shi Y 1995 Efficient line search algorithm for unconstrained optimization JOTA 85, no.3 pp 677-704
[12] Lemaréchal C 1981 A view of line search in A. Auslander, W. Oettli and J. Stoer (Eds.). Optimization and Optimal Control Springer Verlag, pp 59-78
[13] Moré J and Thuente D J On line search algorithms with guaranteed sufficient decrease. Mathematics and Computer Science Division Preprint MCS-P153-0590 1990 Argonne National Laboratory Argonne.
[14] Raydan M 1993 On the Barzilai and Borwein choice of steplength for the gradient method IMA J. Numer. Anal. 13 pp 321-326
[15] Dai Y H, and Liao L Z. 2002. R-linear convergence of the Barzilai and Borwein gradient method. IMA, J. Numer. Anal. 26, 1-10
[16] Grippo L, Lampariello F, and Lucidi S 1986. A nonmonotone line search technique for Newton’s Method SIAM J. Numer. Anal. 23 pp707-716
[17] Luengo F and Raydan M 2003 Gradient method with dynamical retards for large scale Optimization problems Electronic Transactions on Numer. Anal. vol 16 Kentstate University
[18] Birgin E G, Floudas C A, Martínez J M Math. Programming; Heidelberg 125 Iss. 1 pp 139-162
[19] Serafini T G, Zanlungato and Zanni L 2005 Gradient projection methods for quadratic programs and application in training support vector machine. Optim. Meth. Software 20 pp 347-372
[20] Dai Y H and Fletcher 2006 New algorithms for singly linearly constrained quadratic subject to lower and upper bound Math Program Ser A. 106, pp 403-421
[21] Wright J S, Nowak R D, and Figueiredo M A T 2009 Sparse Reconstruction by separable approximation. IEEE Transaction on Signal processing. 57, 2479-2493
[22] Fletcher R. 2001 On the Barzilai-Borwein Method Numer. Anal. Report NA/20
[23] Wen G K, Mamat M, Mohd B I, and. Dasril Y 2012 A novel Step Size Selection Procedures for Steepest Descent Method Appl. Math. Sci. 6 No. 51, pp 2507-2518