Steepest Descent in Unconstrained Optimization with New Step Lengths[1]

B S Goh[2], Feng Ye[3], Shui Sheng Zhou[4]

Abstract. If steepest descent directions are used with exact line search step lengths it is well known that zigzag behavior may occur near a minimum point. We show that if the step lengths are chosen to be equal to the reciprocals of the eigenvalues of the matrix of a positive definite quadratic function of n variables then it is possible to have n-step convergence. This is similar to the well known property of conjugate directions when it is applied to a positive definite quadratic function. We shall show that such new step lengths are equal or less than those given by exact line searches. It is known that the Barzilai-Borwein step lengths can be equal approximately to the reciprocals of the eigenvalues. However a direct use of the BB-step lengths lead to function increases during the iterative process. We propose very simple way to ensure monotonic decreases of the function as the iteration progresses. This leads to very stable steepest descent algorithms may not have zigzag behavior even for very "stiff" problems. With these modifications to ensure monotonic function decreases, the BB-step lengths and exact line search step lengths in combination can then be used more effectively for nonlinear problems which are not quadratic.

Keywords. Unconstrained optimization. Steepest Descent Directions. Convergence. Barzilai-Borwein Steplengths.

1 Introduction

Goh (Ref.1) had used control theory concepts to analyze algorithms for unconstrained optimization problems. The control theory approach focusses on the properties of the whole trajectory generated by an algorithm in an unconstrained optimization problem. The whole trajectory is from an guessed initial point to the minimum point.

The steepest descent algorithm is the simplest algorithm for the numerical solution of an unconstrained optimization problem. If the exact line search step length is used in each iteration for a quadratic function then the trajectory can zigzag very badly near the minimum point, see Nocedal and Wright (Ref.2).

We shall show in this paper that the steepest descent algorithm can have n-step convergence property when it is applied to a positive definite quadratic function of n variables. The step lengths should be chosen to be equal to the

[1]The authors thank Mark Wu, Julian Goh for their assistance in this research.

[2]Research Fellow,Xidian University, Xian, China. Email: bsgoh33@yahoo.com

[3]Lecturer, Mathematics Dept, Xidian University, Xian, China.Email:fye@xidian.edu.cn

[4]Associate Professor, Mathematics Dept, Xidian University, Xian, China.

reciprocals of the eigenvalues. Thus it has a property that is similar to the n-step convergence of conjugate directions for a quadratic function. If the eigenvalues are ordered from the largest to the smallest then the function decreases as the iteration progresses. If not the n-step convergence is still possible but the function can increase in some iterations.

Barzilai-Borwein(Ref.3) introduced a way to compute step lengths which are sometimes exactly equal or approximately equal to the reciprocals of the eigenvalues. However these computed BB-step lengths are not ordered in magnitude. This leads to function increases during the iterative process. Here we shall propose very simple ways to ensure that the function decreases monotonically as the iteration progresses if the BB-step lengths are used in combination with exact line search step lengths.

2 Convergence Properties of Steepest Descent for a Quadratic Function

In an unconstrained optimization problem we seek a vector $x \in R^n$ which minimizes a smooth function $f(x)$.

Consider the iteration

$$x_{k+1} = x_k + \alpha_k p_k \tag{1}$$

where p_k is the search direction and α_k is the step length in the kth iteration.

In a steepest descent algorithm the direction vector is

$$p_k = -\nabla f(x_k) = -g_k \tag{2}$$

With a given direction vector p_k the step length α_k can be chosen so that it minimizes

$$f(x_k + \alpha p_k) \tag{3}$$

If the norm of the direction vector p_k is equal to one then α_k is precisely the step length. Otherwise α_k is just a measure of the step length. But for convenience and following common practice, α_k is called the step length.

By definition, a step length chosen in this way is called an "exact line search step length" if it minimizes the function in (3). We avoid saying that such a step length is an optimal step length. This is because, in the long run, this choice of step length is in fact not the best choice of the step length when the steepest descent directions are used. Furthermore, this choice of step lengths may cause the trajectory to zigzag when the current point is near the minimum point of a quadratic function with widely different eigenvalues. However in practice α_k is chosen by a numerical procedure which calculates it so that the function in (3) is minimized approximately. Such a choice of α_k is called an inexact line search step length.

In this paper we focus our attention on the special case when

$$f(x) = (1/2)x^T Ax \tag{4}$$

where A is an $n \times n$ positive definite matrix.

Let $E_1, E_2, ..., E_n$ be the eigenvectors of the matrix A and $\lambda_1, \lambda_2, ..., \lambda_n$ be the eigenvectors of A. We shall initially assume that all the eigenvectors are distinct. Without loss of generality we assume that they are ordered in decreasing sizes as follows:

$$\lambda_1 > \lambda_2 > ... > \lambda_n > 0 \tag{5}$$

Theorem 2.1. Assume the eigenvalues of A satisfy (5) and the steepest descent directions (2.2) are used. If we choose the step lengths so that $\alpha_k = (1/\lambda_k)$ for $k = 1, 2, ..., n$, then the trajectory generated will converge to the minimum point of the quadratic function in (4) in no more than n steps.

Proof. At the kth iteration let

$$x_k = \sum c_{kj} E_j \tag{6}$$

where c_{kj} are constants.

We have

$$g_k = Ax_k. \tag{7}$$

With the search direction set equal to the steepest descent vectors we get

$$p_k = -g_k = -Ax_k = -\sum c_{kj} AE_j = -\sum c_{kj} \lambda_j E_j. \tag{8}$$

Substitute (6) and (8) into the iterative equation (1) we get

$$x_{k+1} = \sum c_{jk}(1 - \alpha_k \lambda_j) E_j, \tag{9}$$

$$g_{k+1} = Ax_{k+1} = \sum c_{kj}(1 - \alpha_k \lambda_k) \lambda_j E_j \tag{10}$$

with the choice of

$$\alpha_k = 1/\lambda_k \ for \ k = 1, 2, ..., n \tag{11}$$

we eliminate E_1 as a basis of the vectors x_2 and g_2. We continue in this manner and eliminate one eigenvector as a basis vector of the position and gradient vectors in one iteration in a successive manner. Finally E_n is the only basis vector of x_n. Then in the final iteration the choice

$$\alpha_n = 1/\lambda_n \tag{12}$$

generates the minimum point $x_{n+1} = 0$ and $g_{k+1} = 0$. Furthermore the final step length in (12) minimizes the function in (3) along the final steepest descent direction.

Corollary 2.1.1. When the matrix has multiple eigenvalues it is possible to use the above choice of step lengths so that the steepest descent directions generate a trajectory which can reach the minimum point in less than n steps. This follows immediately as we can eliminate the eigenvectors as basis vectors more rapidly.

Corollary 2.1.2. With a simple change of variables the above analysis can be applied to a more general quadratic function of the form

$$f(x) = (1/2)x^T Ax + b^T x + c \tag{13}$$

Corollary 2.1.3. If an initial point is on an eigenvector it can be driven to the minimum point in one iteration with the exact line search step length..

It follows that it is desirable to use steepest descent directions and drive a point to an eigenvector. This property of one step convergence from these special initial points is similar to the one iteration property of the Newton algorithm when it is applied to a positive definite quadratic function.

This single step convergence property of the steepest descent directions in a quadratic function can provide guidelines on how to develop new and effective algorithms for unconstrained optimization. This is being investigated. Some results have been obtained and they will be reported later. It should be noted again that if an initial point is on an eigenvector then the required step length in (12) does in fact minimizes the function (3).

Example. 2.1 Let $f(x) = (1/2)(4x_1^2 + 4x_1x_2 + 2x_2^2)$.

We have

$$\lambda_1 = (3 + \sqrt{5}) \; and \; \lambda_2 = (3 - \sqrt{5}), \tag{14}$$

and

$$E_1 = (2, -1 + \sqrt{5}) \; and \; E_2 = (2, -1 - \sqrt{5}). \tag{15}$$

With the use of these eigenvalues and eigenvectors all points can be driven to the minimum point in no more than two iterations with step lengths equal to the reciprocals of the eigenvalues and steepest descent direction vectors. Points which lie on an eigenvector can be driven to the minimum point in one iteration. Trajectories which converge in two iterations to the minimum point are shown in Fig.1.

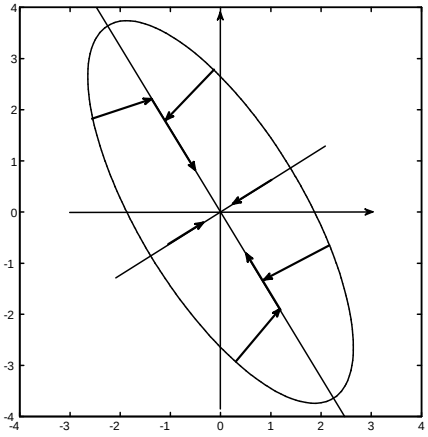

Fig.1. Two iterations convergence from different points with steepest descent directions and new step lengths.

Theorem 2.2. For the quadratic function (4) the exact line search step length in the first iteration is greater than or equal to the reciprocal of the largest eigenvalue of the matrix A.

Proof. Substitute (2) into (1) we get

$$x_2 = x_1 + \alpha_1 p_1 = x_1 - \alpha_1 g_1 \tag{16}$$

Substitute this into (4) and minimize the function with respect to α_1. Let α_1^{ES} denote the exact line search step length. We get

$$\alpha_1^{ES} = (x_1^T A g_1^T)/(g_1^T A g_1) = (g_1^T g_1)/(g_1^T A g_1) \geq (1/\lambda_1) \tag{17}$$

The inequality in (17) follows from the Rayleigh-Ritz ratio property for a symmetric A matrix, see Horn and Johnson (Ref.4).

Corollary 2.2.1. If the eigenvalues are ordered as in (5) and the step lengths are chosen according to (11) then the exact line search step length for the current position,

$$\alpha_k^{ES} = (x_k^T A g_k^T)/(g_k^T A g_k) = (g_k^T g_k)/(g_k^T A g_k) \geq (1/\lambda_k) = \alpha_k \tag{18}$$

Proof. Assume that the eigenvalues satisfy the order in (5). One eigenvector is eliminated as a member of the basis of vectors for the current point vector and the current gradient vector in each iteration. Apply an analysis similar to that in (17) to the subspace spanned by the remaining eigenvectors, we deduce that at each iteration the new step length, $\alpha_k = (1/\lambda_k)$, satisfies a condition similar to that in (17) in the reduced subspace. It is then the largest eigenvalue in the reduced subspace. Condition (18) implies that the new step length in each iteration is equal to or less than the exact line search step length for the current point. This analysis and condition (18) require that the eigenvalues

satisfy the ordering of the eigenvalues in decreasing values as in (5). If not the function may increase in an iteration.

Comments.

Consider a nonlinear unconstrained optimization problem for a non-quadratic function. Suppose (13) is a quadratic approximation. Then the first step to minimize the quadratic approximation is the most important step in the nonlinear problem. This is because the Hessian matrix of the quadratic approximation varies as the current point moves. Thus the largest eigenvalue of the quadratic approximation is the most relevant to calculate a good step length in a nonlinear problem which is non-quadratic.

Example 2.2. Consider the function $f(x) = (1/2)(x_1^2 + 4x_1^2)$. The eigenvalues are $\lambda_1 = 1$ and $\lambda_2 = 4$. Let the initial point A=(2,1) and ABC be the line in the steepest descent direction through A. This line intersects the x_1-axis at B=(1.5,0) and the x_2-axis at C=(0,-3). There are two ways to move in two iterations to the minimum point P=(0,0). We can move by ABP or ACP. The lines BP and CP are also in the steepest descent directions at B and C, respectively. To move from A to B we can use the steepest descent direction at A and the step length equal to 1/4. To move from A to C we use the steepest descent direction at A and the step length equal to 1/1. In each case we use a step length equal to the reciprocal of an eigenvalue. We note that the function is monotonic decreasing along ABP. But it increases along AC. Thus the order that the eigenvalues is used to calculate the step lengths is important, see Fig.2. This is to ensure that the function is monotonic decreasing along a trajectory.

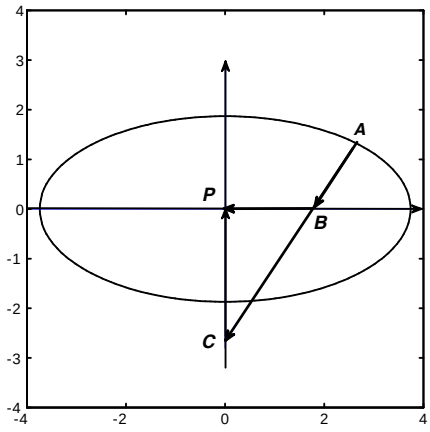

Fig.2. Two step convergence with function decrease along ABP and function increase along AC.

3 New step lengths for steepest descent

Barzilai and Borwein (Ref.3) had proposed step lengths which can compute approximately the reciprocals of the eigenvalues of the matrix of a quadratic function. A difficulty of the Barzilai and Borwein step lengths when they are applied to a non-quadratic problem is that the function may not decrease monotonically as the iteration progresses. This is because the sizes of the step lengths are not ordered as in (5). Dai et.al,(Ref.5) and others have used the Barzilai and Borwein step lengths in combinations with other step lengths and their algorithms have good convergence properties.

We shall use the properties described in Theorem 2.2 and corollary 2.2. We shall propose several simple ways to use the Barzilai and Borwein step lengths and the exact line search step lengths so that we have monotonic decreases in function values as the iteration progresses. In this paper we focus our attention on quadratic functions of the form (4) in order to gain insight into some of the properties of these algorithms.

Algorithm 3.1. Barzilai-Borwein Algorithm.

1. Chose the initial point x_0 and step length α_0 for the iterative process $x_{k+1} = x_k + \alpha_k p_k$.

2. Set the search direction $p_k = -g_k$.

3. Let $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. Compute the BB- step length $\alpha_k^{BB} = (s_{k-1}^T s_{k-1})/(s_{k-1}^T y_{k-1})$.

4. Compute $x_{k+1} = x_k - \alpha_k^{BB} g_k$.

5. End if $\parallel g_{k+1} \parallel < \epsilon$. Otherwise set $k = k + 1$.

There is no restriction on the choice the initial step length. There is a second Barzilai-Borwein formula for the step length. To distinguish it from the above formula we can replace α_k^{BB} in Algorithm. 3.1. by

$$\beta_k^{BB} = (y_{k-1}^T s_{k-1})/(y_{k-1}^T y_{k-1}) \tag{19}$$

Example 3.1. Let $f(x) = (1/2)x^T Ax$ where (i) $A = diag(10000, 10, 2)$; (ii) $A = diag(2.2, 2.1, 2)$. From the initial point (1,2,3) we find convergence occurs in 17 iterations in (i) and in 5 iterations in (ii). If $\{\beta_k^{BB}\}$ of (19) is used, convergence occurs in 12 iterations in (i) and in 5 iterations in (ii). Some but not all the step lengths are approximate values of the reciprocals of the eigenvalues of the matrix A. These are impressive results as the function in (i) is a stiff problem. We note in (i) the function had a very large increase in value in the seventh iteration. Similar results on these fast convergence of the BB-step lengths are obtained in many other examples with quadratic functions. Furthermore there is no zigzag behavior even with very stiff problems where the some eigenvalues are very large compared with others.

We now propose a new algorithm which uses the BB-step lengths. This algorithm is designed so that the function decreases monotonically as the iteration progresses.

Algorithm 3.2. BB-step lengths and monotonic decreases in function values.

1. Chose the initial point x_0, an initial step length α_0 and a deflation factor d where $0 < d < 1$.

2. Set the search direction $p_k = -g_k$.

3. Let $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. Compute the BB-step length $\alpha_k^{BB} = (s_{k-1}^T s_{k-1})/(s_{k-1}^T y_{k-1})$.

4. For $k \geq 1$, if $\Delta f = f(x_k - \alpha_k^{BB} g_k) - f(x_k) \leq 0$, set $\alpha_k = \alpha_k^{BB}$. Otherwise, replace α_k with $\alpha_k = d\alpha_k^{BB}$ and repeat this until $\Delta f < 0$.

5. End if $\parallel g_{k+1} \parallel < \epsilon$. Otherwise let $k = k + 1$.

Comments 3.2. This algorithm is designed for a convex quadratic function. Then the function would decrease in an iteration if the step is sufficiently small. This can be achieved with the repeated use of the deflation factor d.

Here attention is focussed on how to design algorithms for a convex quadratic function. Step 4 in Algorithm 3.2 can be replaced by another condition to enforce monotonic decrease of the function. We have

Algorithm 3.3. BB-step lengths and monotonic decreases in function values.

1. Chose the initial point x_0, initial step length and c where $0 < c < 2$.

2. Set the search direction $p_k = -g_k$.

3. Compute the exact line search step lengths $\alpha_k^{ES} = (g_k^T g_k)/(g_k^T A g_k)$.

4. Set the initial step length $\alpha_0 = \alpha_0^{ES}$.

5. Let $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. Compute the BB-step length $\alpha_k^{BB} = (s_{k-1}^T s_{k-1})/(s_{k-1}^T y_{k-1})$.

6. For $k \geq 1$ set $\alpha_k = \min(c\alpha_k^{ES}, \alpha_k^{BB})$.

7. End if $\parallel g_{k+1} \parallel < \epsilon$. Otherwise let $k = k + 1$.

We shall describe a simpler version of the above algorithm for a positive definite quadratic function. This new version is expected to be more robust when we have a nonlinear function which is not a non-quadratic function. For a non-quadratic function the function may increase if the step length satisfies the condition $0 < \alpha_k < 2\alpha_k^{ES}$. For this reason we impose a stronger condition to ensure monotonic decrease in the function value in an iteration.

Algorithm 3.4.

1. Chose the initial point x_0 for the iterative process $x_{k+1} = x_k + \alpha_k p_k$.

2. Set the search direction $p_k = -g_k$.

3. Compute the exact line search step lengths $\alpha_k^{ES} = (g_k^T g_k)/(g_k^T A g_k)$.

4. Set the initial step length $\alpha_0 = \alpha_0^{ES}$.

5. Let $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. Compute the BB step length $\alpha_k^{BB} = (s_{k-1}^T s_{k-1})/(s_{k-1}^T y_{k-1})$.

6. For $k \geq 1$, set $\alpha_k = \min(\alpha_k^{ES}, \alpha_k^{BB})$.

7. End if $\parallel g_{k+1} \parallel < \epsilon$. Otherwise set $k = k + 1$.

Comments 3.4. In this algorithm step 5 ensures that the function decreases monotonically for a convex quadratic function. Its convergence property is not as good as algorithms 3.2 and 3.3. However it is better designed for nonlinear functions which are not quadratic. For nonlinear functions which are not quadratic the Hessian matrix of the quadratic approximations will vary as the

current point moves when it is not near the minimum point. Step 5 of algorithm 3.4 is a very simple way to ensure that the quadratic function decreases monotonically as the iteration progresses.

We shall outline a proof that Algorithm 3.4 is convergent for a convex quadratic function. Let $\lambda_{\max}$ be the largest eigenvalue of the matrix of the quadratic function. We have

$$\alpha_k^{BB} = (g_{k-1}^T g_{k-1})/(g_{k-1}^T A g_{k-1}) \tag{20}$$

$$\alpha_k^{ES} = (g_k^T g_k)/(g_k^T A g_k) \tag{21}$$

By the Rayleigh-Ritz ratio property for matrix A we have

$$\sigma = (1/\lambda_{\max}) \leq \min(\alpha_k^{BB}, \alpha_k^{ES}) \text{ for all } k \tag{22}$$

Hence for all points x_k not equal to the minimum point we have

$$\Delta f(x_k) = f(x_{k+1}) - f(x_k) \leq f(x_k - \sigma g_k) - f(x_k) < 0 \tag{23}$$

By the inverse Lyapunov function Theorem conditions, see Goh(Ref.1), we can show that the trajectory will converge to the minimum point. In condition (23) the constant σ is independent of k. Thus $f(x_k - \sigma g_k) - f(x_k)$ is only a function of x_k and it is negative definite as a function of x. It is not a function of k explicitly. Here we can use the standard condition in the Lyapunov function Theorem that $\Delta f(x)$ must be negative definite for all points x_k in the admissible region. Thus we have global convergence.

There are well known examples, see Gould and Leyffer (Ref.6), that under the weaker condition that $\Delta f(x_k)$ is negative *along a trajectory* we may not get convergence. In the Gould and Leyffer counterexample $f(x) = x^2$ and $(\alpha_k, p_k) = (2+3/2^{k+1}, (-1)^{k+1})$. In this case (α_k, p_k) and $\Delta f(x_k)$ are functions of k rather than x_k. For this reason the inverse Lyapunov function Theorem cannot be used in this counterexample to establish convergence.

Example 3.2. Use $f(x) = (1/2)x^T Ax$ where $A = diag(10000, 10, 2)$ and initial point $x_0 = (1, 2, 3)$.We use Algorithms 3.1, 3.2 and 3.4 to study this problem. Here convergence to the minimum point occurs after 17 iterations with Algorithm 3.1. It converges after 18 iterations in Algorithm 3.2 with $d = 0.99$, see Fig.3.It is surprising that such a small change in the deflation factor affects the behavior of the solution so dramatically. With this deflation factor the function decreases monotonically. With Algorithm 3.4 the trajectory slows down near the minimum point due to zigzag behavior. Methods to accelerate the rate of convergence when zigzag behavior occurs will be addressed in a separate paper. Here the function decreases monotonically. In our computation we also modify slightly Algorithm 3.4 with the condition $\alpha_k \geq 0.1\alpha_k^{ES}$ for all k.

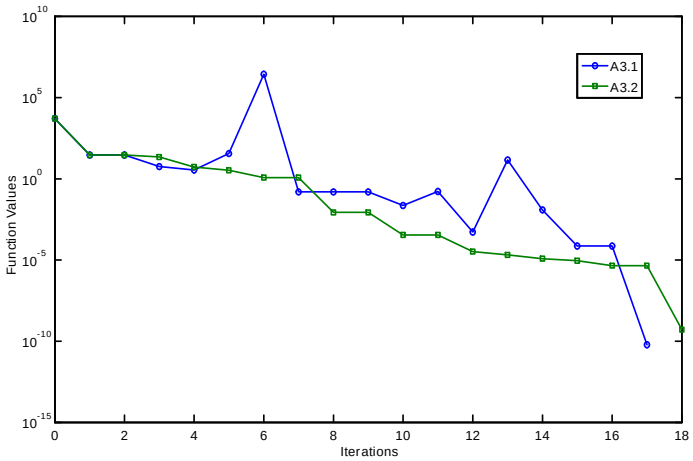

Fig.3. Function value changes with Algorithm 3.1 and 3.2

Example 3.3. Choose $A = diag(10000 : -500 : 900 : -100 : 100, 50, 10, 8, 6, 2)$ in Matlab notations. This is a 33th order diagonal matrix. The initial point is $(330 : -10 : 10)$ in Matlab notations.

With the α_k^{BB} BB-step length Algorithm 3.1. converges in 561 iterations. With β_k^{BB} BB-step lengths it converges in 616 iterations. There are many function increases during the iterations. But the speed of convergence is impressive.

With a deflation factor of $d = 0.99$ we apply Algorithm 3.2 with β_k^{BB} to the same problem. Convergence occurs after 705 iterations. But the function decreases monotonically. Fig.4 shows the dramatic difference in the changes in function values between Algorithm 3.1 and 3.2. When α_k^{BB} is used in Algorithm 3.2 the convergence is slower.

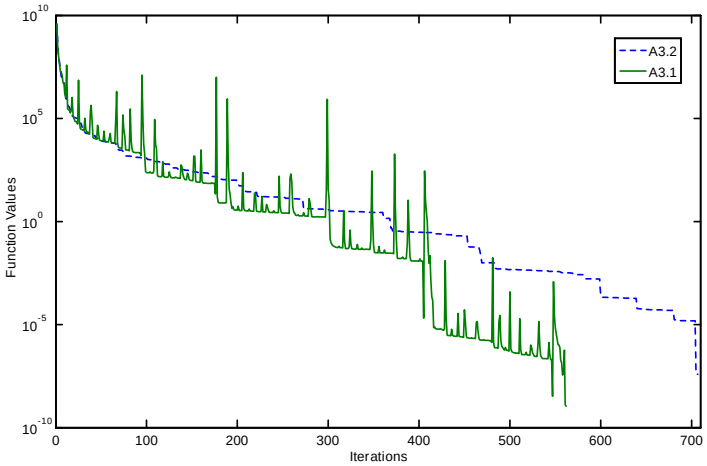

Fig.4. Function value changes with Algorithm 3.1 and 3.2

4 Conclusions

We have looked at the total trajectory in algorithms for unconstrained optimization problems when steepest descent directions are used. For a positive definite quadratic function in n variables we showed that we can have n-step convergence which is similar the well known property of conjugate directions. This is achieved when the step lengths is set equal to the reciprocals of the eigenvalues of the matrix of the quadratic function. We establish that in this n-step trajectory the step lengths must be ordered in an increasing order to get monotonic decrease in function values. Furthermore the new step lengths must be less than or equal those given by exact line search at the current point.

Barzilai and Borwein(Ref.3) have a method to compute step lengths which can be approximately equal to the reciprocals of the eigenvalues. The difficulty with these BB-step lengths is that the function may increase as the iteration progresses. This is because the computed BB-step lengths are not ordered in sizes.

Here we propose simple ways which make use of the BB-step lengths. We modify the algorithm to ensure that the function decreases monotonically as the iteration progresses. But with these modifications it appears the step lengths are less likely to be equal to the reciprocals of the eigenvalues. Thus convergence is slower but the modified algorithm would be better positioned for use with non-quadratic and nonlinear functions. We have many examples with non-quadratic functions where our modified algorithms converge but where the use of the BB-steps lengths alone would fail to converge. However we shall not discuss in details these non-quadratic function examples here as there are other issues to be addressed and more extensive testing are needed.

It appears that the algorithms we described here are able to eliminate the zigzag behavior of some very stiff quadratic function problems to some extent. In these problems the zigzag behavior would appear seriously if the steepest descent directions and exact line search step lengths are used.

References

[1] GOH, B. S., Algorithms for Unconstrained Optimization Problems via Control Theory, Journal of Optimization Theory and Applications, Vol. 92, pp. 581-604, 1997.

[2] NOCEDAL,J., and WRIGHT, S.J., Numerical Optimization, Springer-Verlag, Berlin, 1999.

[3] BARZILAI, J and BORWEIN, J M. Two-point step size gradient methods, IMA Journal Numerical Analysis vol.8 1988, pp141-148.

[4] HORN, R.A and JOHNSON, C.R., Matrix Analysis, Cambridge University Press, Cambridge, 1985.

[5] DAI, Y., YUAN, J. and YUAN, X., Modified Two-point Stepsize Gradient Methods for Unconstrained Optimization, Computational Optimization and Applications, vol.22 2002, pp.103-109.

[6] GOULD, N. I. M., LEYFFER, S. An Introduction to Algorithms for Nonlinear Optimization. Rutherford Appleton Laboratory, UK, 2002 Report No. RAL-TR-2002-031. pp 1-81.