Relaxed Gradient Descent and a New Gradient Descent Methods for Unconstrained Optimization

Neculai Andrei[1]

Abstract. In this work we present two techniques for steplength selection in the frame of gradient descent methods. Using a simple multiplicative modification of the steplength, given by a backtracking procedure, by means of a random variable uniformly distributed in (0,1] we get the relaxed gradient descent method. The second algorithm selects the steplength along the negative gradient using a new approximation of the Hessian of the minimizing function based on the function values and its gradients in two successive points along the iterations. Both algorithms belong to the same class of gradient descent with linear convergence property.

Some preliminary numerical experience shows that the second algorithm compares favourable with the Barzilai-Borwein approach. The main advantage of this new algorithm is the possibility to continue the iterations when the approximation of the Hessian is unproperly chosen.

Keywords: gradient descent methods, relaxed gradient descent, Barzilai-Borwein method

1. Introduction

One of the first and well known method for unconstrained optimization is the gradient descent method, designed by Cauchy early in 1847, in which the negative gradient direction is used to find local minimizers of a differentiable function. The method proved to be effective for functions very well conditioned, but for functions poorly conditioned the method is excessively slow, thus being of no practical value. Even for quadratic functions the gradient descent method with exact line search behave increasingly badly when the conditioning number of the matrix deteriorates. Early attempts to increase the performance of the method have been considered by Humphrey [19], Forsythe and Motzkin [11] and Schinzinger [31]. Even though the storage requirements for the gradient descent method are minimal (3n locations for a n-dimensional problem), the development of conjugate gradient and quasi-Newton methods for large-scale unconstrained optimization cast the gradient descent method in a penumbra.

In 1988 Barzilai and Borwein [2] proposed a gradient descent method (BB method) that uses a different strategy for choosing the step length. This is based on an interpretation of the quasi-Newton methods in a very simple manner. The steplength along the negative gradient direction is computed from a two-point approximation to the secant equation from quasi-Newton methods. In [2] Barzilai and Borwein proved that for the two-dimensional

[1] Research Institute for Informatics, 8-10, Averescu Avenue, Bucharest, Romania, E-mail: nandrei@u3.ici.ro

quadratic case the BB method is R-superlinear convergent. They present some numerical evidence showing that their method is remarkably superior to the classical gradient descent method for a quadratic function with four variables.

Raydan [28] proved that for strictly convex quadratic case with any number of variables the BB method is globally convergent. Using a globalization strategy, based on the non-monotone line search technique introduced by Grippo, Lampariello and Lucidi [17], Raydan [29] proved the global convergence of the BB method for non-quadratic functions and reports some numerical evidence on problems up to 10^4 variables showing that the BB method is competitive with the conjugate gradient Polak-Ribière [25] and CONMIN of Shanno and Phua [32] methods. A preconditioning technique for the BB method has been considered by Molina and Raydan [21]. Under a very restrictive assumption they established the Q-linear rate of convergence of the preconditioned BB method. Some applications of preconditioned BB method on a distance matrix problem are considered by Glunt, Hayden and Raydan [14, 15]. Extension of the BB method for box-constrained optimization problems have been considered by Friedlander, Martinez and Raydan [13] (for quadratic function) and by Birgin, Martinez and Raydan [3].

An analysis of the BB method stressing the importance of non-monotone line search as well as some open problems are presented by Fletcher [9]. Dai and Liao [5] refined the analysis in Raydan [28] and proved that the convergence rate is R-linear. New globalization strategies for BB method, based on relaxations of the monotonicity requirements, are considered by Grippo and Sciandrone [18] where the nonmonotone watchdog technique with nonmonotone linesearch rules are combined. Their algorithms are very sophisticated and dependent of a number of parameters. Numerical experience and comparisons with E04DGF routine of NAG library on some collections of problems, including CUTE, shows that their globalization strategy for the BB algorithm compares favorables with E04DGF algorithm. (However, for some difficult ill-conditioned problems, algorithm E04DGF is more efficient.)

Recently, for the quadratic positive definite case Raydan and Svaiter [30] consider the relaxed gradient method as well as the Cauchy-Barzilai-Borwein methods showing the superiority of the last one against the relaxed gradient descent and BB methods, Particularly, the Cauchy-Barzilai-Borwein method proves to be Q-linearly convergent in a norm defined by the matrix of the problem.

The purpose of this paper is twofold. The first one is to extend the relaxed gradient method to the convex, well conditioned, functions. It is shown that for strongly convex, well conditioned minimization problems the relaxed gradient descent algorithm is an improvement of the classical gradient descent version. The second purpose of the paper is to present a new algorithm of gradient descent type, in which the step size is computed by means of a simple approximation of the Hessian of the minimizing function. In contrast with the Barzilai and Borwein approach in which the steplength is computed from a simple interpretation of the secant equation, the new proposed algorithm considers another approximation of the Hessian based on the function values and its gradients in two successive points along the iterations. The corresponding algorithm belongs to the same class of linear convergent descent methods. The conclusion is that using only the local information given by the gradient, any procedure for step size computation, of any sophistication, does not change the linear convergence class of algorithms.

The paper is organized as follows. Section 2 is dedicated to present the relaxed gradient descent algorithm and its properties. In section 3 we present a new algorithm for unconstrained optimization in which the steplength is computed by backtracking starting with the inverse of a scalar approximation of the Hessian. Section 4 contains some numerical evidence and discussions of this approach.

2. The Relaxed Gradient Descent Method

In the following let us consider the problem:

$$\min f(x) \tag{1}$$

where $f : R^n \longrightarrow R$ is convex and twice continuously differentiable. A necessary and sufficient condition for a point x^* to be optimal for (1) is

$$\nabla f(x^*) = 0. \tag{2}$$

Usually, the problem is solved by an iterative algorithm which generates a sequence of points $x_0, x_1, ... \in dom f$, for which $f(x_k) \longrightarrow f^*$ as $k \longrightarrow \infty$. The algorithms for solving (1) generate a minimizing sequence x_k, $k = 0, 1, ...$ as:

$$x_{k+1} = x_k + t_k d_k, \tag{3}$$

where the scalar $t_k > 0$ is the step size, and the vector d_k is the search direction.

Many procedures for search direction computation have been proposed. One of the first method, and the simplest one, for solving (1) using (3) was the gradient descent method (Cauchy method [4]) where the choice for the search direction at the iteration x_k is the negative gradient, $-\nabla f(x_k)$. Some other known methods dedicated for large-scale problems are based on the conjugate gradient strategy. The simplest methods are those of Fletcher and Reeves [10] (using $3n$ locations) and of Polak and Ribière [25] (using $4n$ locations). More elaborate methods that use more storage are those of Shanno and Phua [32] (CONMIN that use $7n$ locations), the Limited Memory BFGS method of Nocedal [24] (using more than $9n$ locations), the Truncated Newton method of Dembo, Eisenstat and Steihaug [6] (an excellent survey of truncated-Newton methods has been given by Nash [23]), etc.

On the other hand, for step size selection two main algorithms have been considered: exact line search and backtracking line search. In exact line search the step t_k is selected as:

$$t_k = \arg\min_{t>0} f(x_k + td_k) \tag{4}$$

In some special cases (for example quadratic problems) it is possible to compute the step t_k analytically, but in most cases it is computed to approximately minimize f along the ray $\{x_k + td_k : t \geq 0\}$, or at least to reduce f enough. In practice the most used are the *inexact procedures* which try to reduce f enough. Many inexact line search methods have been proposed: Goldstein [16], Armijo [1], Wolfe [33], Powell [27], Denis and Schnabel [7], Fletcher [8], Potra and Shi [26], Lemaréchal [20], Moré and Thuente [22], and many others.

One which is very simple and efficient is the backtracking line search. This procedure considers two constants $0 < \alpha < 0.5$ and $0 < s < 1$ and takes the following steps:

Step 1. Consider the descent direction d_k for f in point x_k. Set $t = 1$.
Step 2. While $f(x_k + td_k) > f(x_k) + \alpha t \nabla f(x_k)^T d_k$, set $t = ts$.
Step 3. Set $t_k = t$.

Typically, $\alpha = 0.0001$ and $s = 0.8$, meaning that we accept a small decrease in f of the prediction based on the linear extrapolation.

One goal of this paper is to show that the linear behavior of the gradient descent method for well-conditioned functions could be improved by a subrelaxation of the step size. For quadratic positive definite problems an overrelaxation have been considered by Raydan and Svaiter [30]. They proved that the poor behavior of the steepest descent methods is due to the optimal Cauchy choice of step size and not to the choice of the search direction. In this paper we extend these results to convex, well conditioned functions. The idea is to modify the gradient descent method by introducing a relaxation of the following form:

$$x_{k+1} = x_k + \theta_k t_k d_k. \tag{5}$$

where θ_k is the relaxation parameter, a random variable uniformly distributed between 0 and 1. With this, the Relaxed Gradient Descent algorithm can be presented as:

Relaxed Gradient Descent Algorithm (RGD)
Step 1. Consider a starting point $x_0 \in dom f$. Set $k = 0$.
Step 2. Compute the search direction: $d_k = -\nabla f(x_k)$.
Step 3. Line search. Choose the step length t_k via exact or backtracking line search procedures.
Step 4. Update the variables: Select $\theta_k \in (0, 1)$ and update the variables: $x_{k+1} = x_k + \theta_k t_k d_k$.
Step 5. Test a criterion for stopping the iterations. If the test is satisfied, then stop; otherwise consider $k = k + 1$ and continue with step 2.

Clearly, if $\theta_k = 1$ for all k we get the classical Gradient Descent (GD) method.

Example 1. Let us illustrate the behavior of the relaxed gradient descent method in comparison with the classical gradient descent on the following function:

$$f(x) = \sum_{i=1}^{n} i x_i^2 + \frac{1}{100} \left(\sum_{i=1}^{n} x_i \right)^2 .$$

Considering $x_0 = [0.5, 0.5, ..., 0.5]$, $\alpha = 0.0001$ and $s = 0.8$ in backtracking procedure, as well as the following criteria for stopping the iterations

$$\|\nabla f(x_k)\| \le \varepsilon_g \quad \text{or} \quad \frac{|f(x_{k+1}) - f(x_k)|}{1 + |f(x_k)|} \le \varepsilon_f$$

with

$$\varepsilon_g = 10^{-6} \text{ and } \varepsilon_f = 10^{-16},$$

than for $n = 100$, the evolution of the norm $|f(x_k) - f^*|$ given by the gradient descent method and of the relaxed gradient descent method are presented in figure 1.

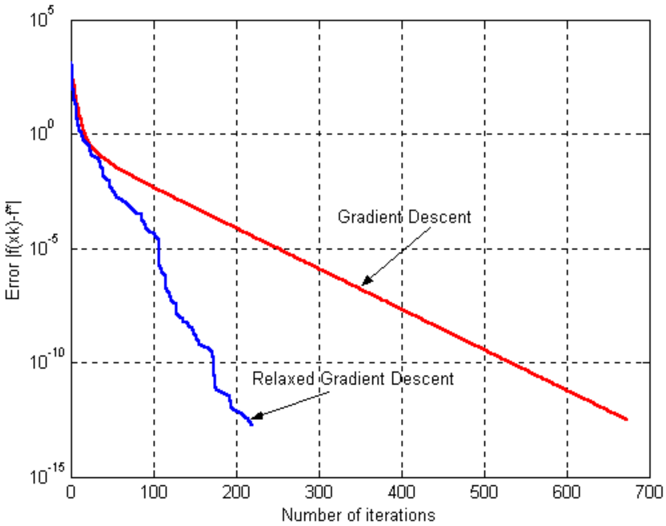

Figure 1: Gradient Descent versus Relaxed Gradient Descent

Table 1a shows the number of iterations corresponding to these algorithms, as well as the average steplength, for different values of n, using the above criteria for stopping.

Table 1a. Number of iterations and the average steplength.

	GD		**RGD**	
n	**# iter**	**average step**	**# iter**	**average step**
500	3105	0.002006	463	0.0209455
1000	6129	0.0010003	505	0.0144660
2000	12147	0.0005011	1256	0.0088871
3000	16773	0.0003349	1146	0.0097431
4000	22722	0.0002516	1177	0.0061838
5000	27910	0.0002013	1523	0.0071524

Observe that the relaxed version of gradient descent method clearly outperforms the classical gradient method. Both methods exhibits a linear convergence to the minimizer of the function, but RGD algorithm takes a significant smaller number of iterations. Similar results have been obtained using different random number generators in interval (0,1). The parameters α and s from backtracking linear search has a noticeable but not a dramatic influence on the number of iteration. Numerical experiments with different values for α leads to the same behaviour of the algorithm. Observe that the average stepsize in RGD algorithm is greater as in GD version. This explains its efficiency.

This numerical experiment reveals the serious limitation of the steplength choice by backtracking when searching is along the negative gradient. This reveals a lack of robustnees of the gradient descent algorithm with backtracking at steplength perturbations.

The convergence analysis of RGD algorithm is given by the following theorems.

Theorem 1. *If the sequence θ_k has an accumulation point $\overline{\theta} \in (0,1)$, then, for strongly convex functions, the sequence x_k generated by RGD algorithm converges linearly to x^*.*

Proof. Let us consider: $\Phi_k(\theta) = f(x_k - \theta t_k g_k)$, where $g_k = \nabla f(x_k)$. We can write:

$$f(x_k - \theta t_k g_k) = f(x_k) - \theta t_k g_k^T g_k + \frac{1}{2}\theta^2 t_k^2 g_k^T \nabla^2 f(x_k) g_k.$$

Since the function f is strongly convex it follows that $\Phi_k(\theta)$ is a convex function and $\Phi_k(0) = f(x_k)$. From the strong convexity of f it follows that:

$$f(x_k - \theta t_k g_k) \leq f(x_k) - \left(\theta - \frac{M t_k}{2}\theta^2\right) t_k \left\| g_k \right\|_2^2 . \tag{6}$$

But $\theta - \frac{M t_k}{2}\theta^2$ is a convex and nonnegative function on $(0, 2/Mt_k)$ and has the maximum value equal with $1/2Mt_k$ in $1/Mt_k$. Therefore $f(x_{k+1}) \leq f(x_k)$ for all k. Since f is bounded below, it follows that

$$\lim_{k\to\infty} \left(f(x_k) - f(x_{k+1})\right) = 0.$$

Now observe that $\Phi_k(\theta)$ is a convex function with the minimum value in point

$$\theta_m = \frac{g_k^T g_k}{t_k \left(g_k^T \nabla^2 f(x_k) g_k\right)} > 0.$$

On the other hand, $\Phi_k(0) = f(x_k)$ and $\Phi_k(\theta_u) = f(x_k)$, where $\theta_u = 2\theta_m$. But,

$$\Phi_k(\theta_m) = f(x_k) - \frac{\left(g_k^T g_k\right)^2}{2\left(g_k^T \nabla^2 f(x_k) g_k\right)} < f(x_k).$$

Therefore, for $\theta \in [0, 2\theta_m]$, $\Phi_k(\theta) \leq \Phi_k(0)$.

There exists some $\beta \in (0, \theta_m)$ with $\beta < 1$ such that $\beta \leq \overline{\theta} \leq 2\theta_m - \beta$. Therefore, there exists a subsequence θ_{k_j} contained in the interval $[\beta, 2\theta_m - \beta]$. Using again the convexity of function $\Phi_k(\theta)$ we get that

$$\Phi_k(0) - \Phi_k(\beta\theta_m) \geq -\beta \left[\Phi_k(\theta_m) - \Phi_k(0)\right].$$

But,

$$\Phi_k(\theta_m) - \Phi_k(0) = -\frac{\left(g_k^T g_k\right)^2}{2\left(g_k^T \nabla^2 f(x_k) g_k\right)}.$$

Hence,

$$\Phi_k(0) - \Phi_k(\beta\theta_m) \geq \frac{\beta}{2} \frac{\left(g_k^T g_k\right)^2}{\left(g_k^T \nabla^2 f(x_k) g_k\right)} \geq \frac{\beta}{2\lambda_{\max}} \left\| g_k \right\|_2^2.$$

Therefore

$$f(x_{k_j}) - f(x_{k_j+1}) \geq \frac{\beta}{2\lambda_{\max}} \left\| g_k \right\|_2^2.$$

But, $f(x_{k_j}) - f(x_{k_j+1}) \to 0$ and as a consequence g_{k_j} goes to zero, i.e. x_{kj} converges to x^*. Having in view that $f(x_k)$ is an nonincreasing sequence, it follows that $f(x_k)$ converges to $f(x^*)$. ■

In the following let us assume that f is strongly convex and the sublevel set $S = \{x \in dom f : f(x) \leq f(x_0)\}$ is closed. Strong convexity of f on S involves that there exists the constants m and M such that $mI \leq \nabla^2 f(x) \leq MI$, for all $x \in S$. A consequence of strong convexity of f on S is that we can bound f^* as:

$$f(x) - \frac{1}{2m} \left\| \nabla f(x) \right\|_2^2 \leq f^* \leq f(x) - \frac{1}{2M} \left\| \nabla f(x) \right\|_2^2. \tag{7}$$

In these circumstances the following theorem can be proved.

Theorem 2. *For strongly convex functions the Relaxed Gradient Descent algorithm with backtracking is linear convergent and*

$$f(x_k) - f^* \leq \left(\prod_{i=0}^{k-1} c_i \right) \left(f(x_0) - f^* \right), \tag{8}$$

where

$$c_i = 1 - \min \left\{ 2m\alpha\theta_i, 2m\alpha\theta_i s/M \right\} < 1. \tag{9}$$

Proof. Consider $0 < \theta < 1$, then

$$f(x_k - \theta t_k g) \leq f(x_k) - \left(\theta - \frac{Mt_k}{2}\theta^2 \right) t_k \left\| g_k \right\|_2^2.$$

Notice that $\theta - \frac{Mt_k}{2}\theta^2$ is a concave function and for all $0 \leq \theta \leq 1/Mt_k$, $\theta - \frac{Mt_k}{2}\theta^2 \geq \frac{\theta}{2}$. Hence

$$f(x_k - \theta t_k g) \leq f(x_k) - \frac{\theta}{2} t_k \left\| g_k \right\|_2^2 \leq f(x_k) - \alpha\theta t_k \left\| g_k \right\|_2^2,$$

since $\alpha \leq 1/2$. Therefore the backtracking line search procedure terminates either with $t_k = 1$ or with a value $t_k \geq s/M$. With this, at step k we can get a lower bound on the decrease of the function. In the first case we have

$$f(x_{k+1}) \leq f(x_k) - \alpha\theta_k \left\| g_k \right\|_2^2,$$

and in the second one

$$f(x_{k+1}) \leq f(x_k) - \alpha\theta_k \frac{s}{M} \|g_k\|_2^2 .$$

Therefore, we have

$$f(x_{k+1}) \leq f(x_k) - \min\left\{\alpha\theta_k, \alpha\theta_k \frac{s}{M}\right\} \|g_k\|_2^2 .$$

Hence

$$f(x_{k+1}) - f^* \leq f(x_k) - f^* - \min\left\{\alpha\theta_k, \alpha\theta_k \frac{s}{M}\right\} \|g_k\|_2^2 .$$

But, from (7) it follows that $\|g_k\|_2^2 \geq 2m(f(x_k) - f^*)$. Therefore, combining this with the above relation we get

$$f(x_{k+1}) - f^* \leq \left(1 - \min\left\{2m\alpha\theta_k, 2m\alpha\theta_k \frac{s}{M}\right\}\right) (f(x_k) - f^*) .$$

Denoting $c_k = 1 - \min\left\{2m\alpha\theta_k, 2m\alpha\theta_k \frac{s}{M}\right\}$ it follows that for every $k = 0, 1, \ldots$

$$f(x_{k+1}) - f^* \leq c_k (f(x_k) - f^*) ,$$

which prove the second part of the theorem. Since $c_k < 1$, the sequence $f(x_k)$ converges to f^* like a geometric series with an exponent that partialy depends on the condition number bound M/m, the backtracking parameters α and s, the sequence θ_k of random uniform distributed numbers in $(0, 1)$ interval and on the initial suboptimality. Therefore the RGD algorithm is linear convergent. ■

For strongly convex functions the behavior of the gradient descent algorithm is given by:

$$f(x_k) - f^* \leq c^k (f(x_0) - f^*) , \tag{10}$$

where

$$c = 1 - \min\left\{2m\alpha, 2m\alpha\frac{s}{M}\right\} < 1. \tag{11}$$

Observe that for every $k = 0, 1, \ldots,$ since $\theta_k < 1$, it follows that $c_k > c$.

3. A New Algorithm for Unconstrained Optimization

As we mentioned in introduction, for problem (1) Barzilai and Borwein [2] suggested an algorithm which essentially is a gradient one, where the choice of the stepsize along the negative gradient is derived from a two-point approximation to the secant equation from quasi-Newton methods. Considering $D_k = \gamma_k I$ as an approximation to the Hessian of f at x_k,they chose γ_k such that

$$D_k = \arg\min \|Ds_k - y_k\|_2$$

where $s_k = x_k - x_{k-1}$ and $y_k = \nabla f(x_k) - \nabla f(x_{k-1})$, yielding

$$\gamma_k^{BB} = \frac{s_k^T y_k}{s_k^T s_k}. \tag{12}$$

With these, the method of Barzilai and Borwein is given by the following iterative scheme:

$$x_{k+1} = x_k - \frac{1}{\gamma_k^{BB}} \nabla f(x_k). \tag{13}$$

Mainly, the sequence $\{x_k\}$ generated by the BB method uses two initial vectors x_0 and x_1. Having in view its simplicity and numerical efficiency for well-conditioned problems, proved *inter alia* by Raydan [29] and Fletcher [9], the Barzilai and Borwein gradient method has received a great deal of attention. However, like all steepest descent and conjugate gradient methods, the BB method becomes slow when the problems happens to be more ill-conditioned [12] .

In the following I suggest another procedure for computing an approximation of the Hessian of the function f at x_k which can be considered to get the stepsize along the negative gradient. Let us consider the initial point x_0 where $f(x_0)$ and $g_0 = \nabla f(x_0)$ can immediately be computed. Using the backtracking procedure (initialized with $t = 1$) we can compute the steplength t_0 with which the next estimate $x_1 = x_0 - t_0 g_0$ is computed, where again we can compute $f(x_1)$ and $g_1 = \nabla f(x_1)$. So, the first step is computed using the backtracking along the negative gradient. Now, in point $x_{k+1} = x_k - t_k g_k$, $k = 0, 1, \ldots$we have:

$$f(x_{k+1}) = f(x_k) - t_k g_k^T g_k + \frac{1}{2} t_k^2 g_k^T \nabla^2 f(z) g_k, \tag{14}$$

where $z \in [x_k, x_{k+1}]$. Having in view the local character of the searching procedure and that the distance between x_k and x_{k+1} is enough small we can choose $z = x_{k+1}$ and consider $\gamma(x_{k+1})I$ as an approximation of the $\nabla^2 f(x_{k+1})$, where $\gamma(x_{k+1}) \in R$. This is an anticipative point of view, in which the approximation of the Hessian in point x_{k+1} is computed using the local information from point x_k. Therefore we can write:

$$\gamma(x_{k+1}) = \frac{2}{g_k^T g_k} \frac{1}{t_k^2} \left[f(x_{k+1}) - f(x_k) + t_k g_k^T g_k \right] . \tag{15}$$

Now, in order to compute the next estimation $x_{k+2} = x_{k+1} - t_{k+1} g_{k+1}$ we must consider a procedure to determine the stepsize t_{k+1}. For this let us consider the function:

$$\Phi_{k+1}(t) = f(x_{k+1}) - t g_{k+1}^T g_{k+1} + \frac{1}{2} t^2 \gamma(x_{k+1}) g_{k+1}^T g_{k+1}.$$

Observe that $\Phi_{k+1}(0) = f(x_{k+1})$ and $\Phi'_{k+1}(0) = -g_{k+1}^T g_{k+1} < 0$. Therefore $\Phi_{k+1}(t)$ is a convex function for all $t \geq 0$. To have a minimum for $\Phi_{k+1}(t)$ we must have $\gamma(x_{k+1}) > 0$. Considering for the moment that $\gamma(x_{k+1}) > 0$, then from $\Phi'_{k+1}(t) = 0$ we get

$$\bar{t}_{k+1} = \frac{1}{\gamma(x_{k+1})}, \tag{16}$$

as the minimum of $\Phi_{k+1}(t)$. Now,

$$\Phi_{k+1}(\bar{t}_{k+1}) = f(x_{k+1}) - \frac{1}{2\gamma(x_{k+1})} \|g_{k+1}\|_2^2 ,$$

which shows that if $\gamma(x_{k+1}) > 0$, then the value of function f is reduced. This suggests us to determine the stepsize t_{k+1} as:

$$t_{k+1} = \arg\min_{t \leq \bar{t}_{k+1}} f(x_{k+1} - tg_{k+1}) \tag{17}$$

using the backtracking procedure.

To complete the algorithm we must consider the situation when $\gamma(x_{k+1}) < 0$. If $f(x_{k+1}) - f(x_k) + t_k g_k^T g_k < 0$, then the reduction $f(x_k) - f(x_{k+1})$ is greater that $t_k g_k^T g_k$. In this case we change a little the stepsize t_k as $t_k + \eta_k$ in such a manner that

$$f(x_{k+1}) - f(x_k) + (t_k + \eta_k) g_k^T g_k > 0.$$

To get a value for η_k let us select a $\delta > 0$, enough small, and consider:

$$\eta_k = \frac{1}{g_k^T g_k} \left[f(x_k) - f(x_{k+1}) - t_k g_k^T g_k \right] + \delta \tag{18}$$

with which a new value for $\gamma(x_{k+1})$ can be computed as:

$$\gamma(x_{k+1}) = \frac{2}{g_k^T g_k} \frac{1}{(t_k + \eta_k)^2} \left[f(x_{k+1}) - f(x_k) + (t_k + \eta_k) g_k^T g_k \right]. \tag{19}$$

The corresponding New gradient descent Algorithm (NA) is as follows:

The New Algorithm (NA)

Step 1. Select $x_0 \in dom f$ and compute $f(x_0)$, $g_0 = \nabla f(x_0)$ and $t_0 = \arg\min_{t<1} f(x_0 - tg_0)$. Compute $x_1 = x_0 - t_0 g_0,$ $f(x_1)$ and $g_1 = \nabla f(x_1)$. Set $k = 0$.

Step 2. Test for continuation. If some criteria for stopping the algorithm are satisfied, then stop; otherwise continue with step 3.

Step3. Compute the (scalar) approximation, $\gamma(x_{k+1})$, of the Hessian of function f at x_{k+1} as in (15).

Step 4. If $\gamma(x_{k+1}) < 0$, then select $\delta > 0$ and compute a new value for $\gamma(x_{k+1})$ as in (19), where η_k is given by (18).

Step 5. Compute the initial stepsize

$$\bar{t}_{k+1} = \frac{1}{\gamma(x_{k+1})},$$

with which a backtracking procedure is performed in the next step.

Step 6. Using a backtracking procedure, determine the step length t_{k+1} as:

$$t_{k+1} = \arg\min_{t \leq \bar{t}_{k+1}} f(x_{k+1} - tg_{k+1}).$$

Step 7. Update the variables: $x_{k+2} = x_{k+1} - t_{k+1} g_{k+1},$ set $k = k + 1$ and go to step 2.

Example 1. (continued) For example 1 the number of iterations and the steplength corresponding to the Barzilai-Borwein algorithm and the New Algorithm are given in table 1b.

Table 1b. Number of iterations and the step length of BB and NA.

	Barzilai-Borwein (BB)		New Algorithm (NA)		
n	**# iter**	**average step**	**# iter**	**average step**	$\gamma < 0$
500	748	0.008818	706	0.009316	0
1000	1353	0.004934	1269	0.004967	0
2000	2675	0.002398	2410	0.002510	0
3000	3526	0.001740	3282	0.001687	0
4000	4194	0.001319	4895	0.001289	0
5000	6227	0.001031	6113	0.001020	0

where the elements in column below $\gamma < 0$ represents the number of iterations in which $\gamma(x_{k+1}) < 0$. Observe that the behaviour of the New Algorithm is very close to that of Barzilai-Borwein. In fact, the difference $\gamma(x_{k+1}) - \gamma_k^{BB}$ is very small, as it is illustrated in figure 2.

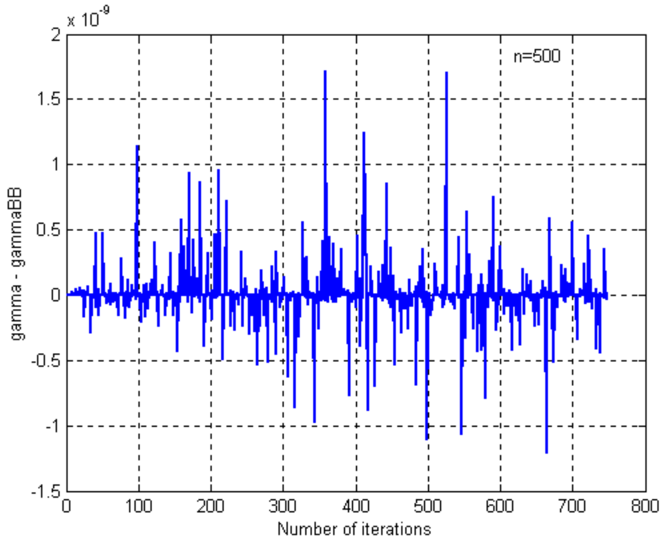

Figure 2: Difference $\gamma(x_{k+1}) - \gamma_k^{BB}$.

However, this is not the typical behaviour of the NA algorithm. Generally, for the most numerical experiments we notice that $\gamma(x_{k+1}) \geq \gamma_k^{BB}$.

The analysis of convergence of this algorithm is given by the following theorems.

Theorem 3. *For strongly convex functions the New Algorithm with backtracking is linear convergent and*

$$f(x_k) - f^* \leq \left(\prod_{i=0}^{k-1} c_i \right) (f(x_0) - f^*),$$

where

$$c_i = 1 - \min\{2m\alpha, 2m\alpha s^{p_i}\} < 1$$

and $p_i \geq 1$ *is an integer, (* $p_i = 1, 2, \ldots$ *given by the backtracking procedure).*

Proof. This is very similar with the proof in theorem 2 above. We can write:

$$f(x_{k+1}) = f(x_k) - \left(t - \frac{1}{2}t^2\gamma(x_{k+1})\right)\|g_k\|_2^2.$$

But, $t - t^2\gamma(x_{k+1})/2$ is a concave function, and for all $0 \leq t \leq 1/\gamma(x_{k+1})$, $t - t^2\gamma(x_{k+1})/2 \geq t/2$. Hence

$$f(x_{k+1}) \leq f(x_k) - \frac{t}{2}\|g_k\|_2^2 \leq f(x_k) - \alpha t\|g_k\|_2^2.$$

The backtracking procedure terminates either with $t = 1$ or with $t = s^{p_k}$, where p_k is an integer. Therefore

$$f(x_{k+1}) \leq f(x_k) - \min\{\alpha, \alpha s^{p_k}\}\|g_k\|_2^2.$$

Having in view that for strongly convex functions $\|g_k\|_2^2 \geq 2m(f(x_k) - f^*)$ it follows that

$$f(x_{k+1}) - f^* \leq c_k\left(f(x_k) - f^*\right),$$

where $c_k = 1 - \min\{2m\alpha, 2m\alpha s^{p_k}\}$. Since $c_k < 1$ the sequence $f(x_k)$ is linear convergent, like a geometric series, to f^*. ■

Theorem 4. *For every* $k = 0, 1, \ldots$ $\gamma(x_{k+1})$, *generated by the New Algorithm, is bounded away form zero.*

Proof. For every $k = 0, 1, \ldots$ we know that $f(x_{k+1}) - f(x_k) + t_k g_k^T g_k > 0$. Therefore, $f(x_k) - f(x_{k+1}) < t_k g_k^T g_k$. With this we have:

$$\gamma(x_{k+1}) = \frac{2}{t_k} - \frac{2\left(f(x_k) - f(x_{k+1})\right)}{t_k^2\left(g_k^T g_k\right)} > \frac{2}{t_k} - \frac{2t_k\left(g_k^T g_k\right)}{t_k^2\left(g_k^T g_k\right)} = 0. \blacksquare$$

Therefore the step 5 of the NA is well defined. However, towards the final iterations of the algorithm, especially when the accuracy requirements are too high, it is possible that $\left(f(x_{k+1}) + t_k g_k^T g_k\right) - f(x_k) < 0$, but very close to zero. This is because $g_k^T g_k$ is too small. That means the reduction in function values is too small. The remedy we have considered in this situation is to increase a little the steplength in order to compensate the accuracy requirements.

4. Numerical Experiments

In this section we report some numerical results obtained by a FORTRAN implemention of the above gradient descent algorithms for 11 functions. In all experiments the backtracking procedure considers $\alpha = 0.0001$ and $s = 0.8$. The criteria for stopping the algorithms are those used in example 1 above. For each function the initial point has been presented. Each table presents the number of iterations as well as the average stepsize corresponding to algorithms. Tables *a present the number of iterations and the average steplength for the

GD and the RGD algorithms. Tables *b give the number of iterations, corresponding to the BB and the NA algorithms. Tables *c show the number of iterations of an implementation of BB algorithm. Tables *d present the number of iterations of NA for different values of δ. The number of iterations in which $\gamma(x_{k+1}) < 0$ or $\gamma^{BB} < 0$ is shown in column below γ.

Example 2. $f(x) = \sum_{i=1}^{n} \frac{i}{10} \left(\exp(x_i) - x_i\right), x_0 = [1, 1, \ldots, 1]$.

Table 2a. Number of iterations and the average steplength of GD and RGD.

	GD		**RGD**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	2696	0.020215	1168	0.117336
2000	4380	0.010106	1159	0.080865
3000	5514	0.006744	1340	0.063615
4000	5788	0.005044	1177	0.057248
5000	6108	0.004036	1523	0.045275

Table 2b. Number of iterations and the average steplength of BB and NA.

	BB		**NA**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	744	0.08733	588	0.1199889
2000	1284	0.04795	758	0.0498313
3000	1487	0.03086	1465	0.0429736
4000	1812	0.02458	1401	0.0348057
5000	1676	0.01981	1316	0.0189150

Table 2c. Number of iterations of BB.

n	**# iter**	γ
10000	2155	0
20000	3181	0
30000	2747	0
40000	2073	0
50000	2214	0

Table 2d. Number of iterations of NA for different values of δ.

	$\delta = 0.01$		$\delta = 0.1$		$\delta = 1$		$\delta = 10$		$\delta = 100$	
n	**# iter**	γ	**# iter**	γ	**# iter**	γ	**# iter**	γ	**# iter**	γ
10000	2413	0	2413	0	2413	0	2413	0	2413	0
20000	2586	3	3091	19	1896	4	1702	4	1918	2
30000	2806	3	2405	2	2858	16	2173	2	2076	8
40000	2449	23	3099	4	2865	4	2504	8	2407	3
50000	3847	11	3931	11	3368	19	2915	1	3427	11

Example 3. (Tridiagonal function)

$$f(x) = \left((5-3x_1-x_1^2)x_1-3x_2+1\right)^2 + \sum_{i=2}^{m-1}\left((5-3x_i-x_i^2)x_i-x_{i-1}-3x_{i+1}+1\right)^2 + \left((5-3x_m-x_m^2)x_m-x_{m-1}+1\right)^2,$$
$$x_0 = [-1,-1,\dots,-1].$$

Table 3a. Number of iterations and the average steplength of GD and RGD.

	GD		**RGD**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	574	0.0017747	130	0.026428
2000	154	0.0087885	148	0.028887
3000	547	0.0021763	316	0.052925
4000	3681	0.0082045	226	0.061401
5000	147	0.0087238	279	0.043099

Table 3b. Number of iterations and the average steplength of BB and NA.

	BB		**NA**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	103	0.00678	112	0.00662298
2000	50	0.02207	116	0.01128028
3000	212	0.02659	101	0.01049011
4000	81	0.01034	71	0.01177585
5000	198	0.00946	147	0.00518654

Table 3c. Number of iterations of BB.

n	**# iter**	γ
10000	70	0
20000	241	5
30000	478	14
40000	48	0
50000	50	0

Table 3d. Number of iterations of NA for different values of δ.

	$\delta = 0.01$		$\delta = 0.1$		$\delta = 1$		$\delta = 10$		$\delta = 100$	
n	**# iter**	γ	**# iter**	γ	**# iter**	γ	**# iter**	γ	**# iter**	γ
10000	75	5	69	1	62	1	64	1	63	1
20000	52	1	65	1	249	7	51	1	76	2
30000	66	2	71	2	68	1	77	1	80	2
40000	51	0	51	0	51	0	51	0	51	0
50000	51	2	48	2	51	1	48	1	50	1

Example 4. (Penalty function)

$$f(x) = \sum_{i=1}^{n-1} (x_i - 1)^2 + \left(\sum_{j=1}^{n} x_j^2 - 0.25 \right)^2, \; x_0 = [1, 2, \ldots, n]\,.$$

Table 4a. Number of iterations and the average steplength of GD and RGD.

	GD		**RGD**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	88	0.0192150	42	0.0431264
2000	95	0.0115713	43	0.0282512
3000	243	0.0151839	42	0.0283190
4000	110	0.0071455	45	0.0296132
5000	138	0.0086804	42	0.0181590

Table 4b. Number of iterations and the average steplength of BB and NA.

	BB		**NA**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	45	0.008656	51	0.008363
2000	48	0.007326	50	0.006259
3000	45	0.006173	50	0.004794
4000	49	0.005273	53	0.003977
5000	49	0.004581	57	0.006126

Table 4c. Number of iterations of BB.

n	**# iter**	γ
10000	58	0
20000	58	0
30000	56	0
40000	58	0
50000	61	0

Table 4d. Number of iterations of NA for different values of δ.

	$\delta = 0.01$		$\delta = 0.1$		$\delta = 1$		$\delta = 10$		$\delta = 100$	
n	**# iter**	γ	**# iter**	γ	**# iter**	γ	**# iter**	γ	**# iter**	γ
10000	63	2	62	1	62	1	65	4	62	1
20000	66	1	68	3	69	4	66	1	70	5
30000	65	4	63	1	66	5	67	5	64	3
40000	67	1	69	3	68	1	70	4	73	7
50000	74	7	75	8	72	5	79	12	75	6

Example 5. (Tridiagonal function)

$$f(x) = \left((2+5x_1^2)x_1 + 2x_2 + 1\right)^2 + \sum_{i=2}^{n-1} \left((2+5x_i^2)x_i + x_{i-1} + 2x_{i+1} + 1\right)^2 +$$
$$\left((2+5x_n^2)x_n + x_{n-1} + 1\right)^2,$$
$$x_0 = [1, 1, \ldots, 1].$$

Table 5a. Number of iterations and the average steplength of GD and RGD.

	GD		**RGD**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	1087	0.031504	279	0.149108
2000	1059	0.031499	320	0.142032
3000	1061	0.031499	264	0.166463
4000	1063	0.031500	286	0.149378
5000	1063	0.031500	279	0.152496

Table 5b. Number of iterations and the average steplength of BB and NA.

	BB		**NA**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	224	0.11734	195	0.129313
2000	235	0.11257	193	0.130184
3000	219	0.11813	196	0.130313
4000	219	0.12023	201	0.129180
5000	190	0.12876	185	0.135398

Table 5c. Number of iterations of BB.

n	**# iter**	γ
10000	207	0
20000	226	0
30000	207	0
40000	192	0
50000	205	0

Table 5d. Number of iterations of NA

n	**# iter**	γ
10000	195	0
20000	188	0
30000	191	0
40000	192	0
50000	218	0

Example 6. $f(x) = \sum_{i=1}^{n-1} \left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2$, $x_0 = [-1.2, 1, \ldots, -1.2, 1]$.

Table 6a. Number of iterations and the average steplength of GD and RGD.

	GD		RGD	
n	**# iter**	**average step**	**# iter**	**average step**
1000	295	0.100293	177	0.309895
2000	298	0.100221	177	0.291935
3000	297	0.100196	172	0.315101
4000	296	0.100172	180	0.297528
5000	298	0.100149	176	0.295032

Table 6b. Number of iterations and the average steplength of BB and NA.

	BB		NA	
n	**# iter**	**average step**	**# iter**	**average step**
1000	89	0.28313	78	0.29727
2000	83	0.29775	74	0.29916
3000	86	0.28155	59	0.27871
4000	84	0.29868	86	0.29183
5000	88	0.29738	91	0.28947

Table 6c. Number of iterations of BB.

n	**# iter**	γ
10000	95	0
20000	68	0
30000	85	0
40000	96	0
50000	95	0

Table 6d. Number of iterations of NA

n	**# iter**	γ
10000	82	0
20000	84	0
30000	81	0
40000	79	0
50000	82	0

Example 7. (Trigonometric function)

$$f(x) = \sum_{i=1}^{n} \left(\left(n - \sum_{j=1}^{n} \cos x_j \right) + i\,(1 - \cos x_i) - \sin x_i \right)^2, \; x_0 = [0.2, 0.2, \dots, 0.2] .$$

Table 7a. Number of iterations and the average steplength of GD and RGD.

	GD		RGD	
n	**# iter**	**average step**	**# iter**	**average step**
1000	68	0.366606	355	0.974325
2000	108	0.313093	48	0.750819
3000	130	0.365581	33	0.596991
4000	76	0.191776	548	0.976957
5000	137	0.313370	43	0.702128

Table 7b. Number of iterations and the average steplength of BB and NA.

	BB		NA	
n	**# iter**	**average step**	**# iter**	**average step**
1000	44	0.51781	40	0.44384
2000	31	0.21982	36	0.26234
3000	31	0.19371	31	0.16299
4000	31	0.17401	30	0.09079
5000	32	0.17217	31	0.09094

Table 7c. Number of iterations of BB. **Table 7d.** Number of iterations of NA

n	**# iter**	γ
10000	33	0
20000	36	0
30000	36	0
40000	38	0
50000	39	0

n	**# iter**	γ
10000	33	0
20000	36	0
30000	37	0
40000	40	0
50000	41	0

Example 8. $f(x) = \sum_{i=1}^{n-1} \left(x_{i+1} - x_i^3\right)^2 + (1 - x_i)^2 \, , x_0 = [-1.2, 1, \ldots, -1.2, 1] \, .$

Table 8a. Number of iterations and the average steplength of GD and RGD.

	GD		RGD	
n	**# iter**	**average step**	**# iter**	**average step**
1000	1061	0.059832	418	0.258687
2000	1069	0.059531	451	0.246441
3000	1069	0.059513	438	0.257312
4000	1070	0.059523	449	0.252725
5000	1063	0.059831	485	0.231459

Table 8b. Number of iterations and the average steplength of BB and NA.

	BB		NA	
n	**# iter**	**average step**	**# iter**	**average step**
1000	230	0.22841	185	0.265502
2000	222	0.24470	217	0.242480
3000	242	0.21853	203	0.251972
4000	220	0.23544	218	0.237006
5000	233	0.22414	198	0.257031

Table 8c. Number of iterations of BB.

n	# iter	γ
10000	200	0
20000	219	0
30000	209	0
40000	230	0
50000	214	0

Table 8d. Number of iterations of NA for different values of δ.

	$\delta = 0.01$		$\delta = 0.1$		$\delta = 1$		$\delta = 10$		$\delta = 100$	
n	**# iter**	γ	**# iter**	γ	**# iter**	γ	**# iter**	γ	**# iter**	γ
10000	224	1	219	1	204	1	239	1	208	1
20000	207	1	200	1	230	1	210	1	190	1
30000	194	1	226	1	234	1	224	1	204	1
40000	195	1	207	1	215	1	222	1	226	1
50000	212	1	222	1	222	1	224	1	205	1

Example 9. $f(x) = \sum_{i=1}^{n/2} \left(x_{2i-1}^2 + x_{2i}^2 + x_{2i-1}x_{2i}\right)^2 + \sin^2(x_{2i-1}) + \cos^2(x_{2i})$, $x_0 = [3, 0.1, \ldots 3, 0.1]$.

Table 9a. Number of iterations and the average steplength of GD and RGD.

	GD		**RGD**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	145	0.326641	31	0.530351
2000	145	0.326641	31	0.530351
3000	142	0.326620	30	0.530963
4000	142	0.326620	30	0.530963
5000	142	0.326620	30	0.530963

Table 9b. Number of iterations and the average steplength of BB and NA.

	BB		**NA**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	15	0.26319	12	0.291550
2000	15	0.26319	12	0.277770
3000	15	0.26319	12	0.263386
4000	15	0.26319	12	0.255323
5000	15	0.26319	12	0.252502

Table 9c. Number of iterations of BB. **Table 9d.** Number of iterations of NA

n	# iter	γ
10000	15	0
20000	16	0
30000	16	0
40000	16	0
50000	16	0

n	# iter	γ
10000	12	0
20000	12	0
30000	12	0
40000	12	0
50000	12	0

Example 10. $f(x) = \sum_{i=1}^{n-1} \left(x_i^2 + x_{i+1}^2 + x_i x_{i+1}\right)^2 + \sin^2(x_i) + \cos^2(x_{i+1}),\ x_0 = [3, 0.1, \ldots 3, 0.1]$.

Table 10a. Number of iterations and the average steplength of GD and RGD.

	GD		RGD	
n	**# iter**	**average step**	**# iter**	**average step**
10	6301	0.414179	8396	0.961205
100	6587	0.414563	10736	0.962470
500	7014	0.414611	11193	0.963675
1000	6321	0.414590	8191	0.963520

Table 10b. Number of iterations and the average steplength of BB and NA.

	BB		NA	
n	**# iter**	**average step**	**# iter**	**average step**
10	1084	3.65298	1056	3.755755
100	1687	3.61832	1212	8.452615
500	2522	3.67988	533	8.248792
1000	2539	3.36897	566	34.69698

Table 10c. Number of iterations of BB.

n	# iter	γ
10000	2088	2
20000	2223	0
30000	1582	1
40000	1644	0
50000	1883	0

Table 10d. Number of iterations of NA for different values of δ.

	$\delta=0.01$		$\delta=0.1$		$\delta=1$		$\delta=10$		$\delta=100$	
n	**# iter**	γ	**# iter**	γ	**# iter**	γ	**# iter**	γ	**# iter**	γ
10000	690	70	598	82	646	83	398	23	888	82
20000	384	33	283	22	420	46	703	85	705	66
30000	624	46	325	23	560	62	737	73	798	62
40000	252	7	663	89	360	51	606	65	574	41
50000	321	21	488	48	1142	204	695	65	637	52

Example 11. (Beale function)

$$f(x) = \sum_{i=1}^{n/2} \left(1.5 - x_{2i-1}\left(1 - x_{2i}\right)\right)^2 +$$
$$\left(2.25 - x_{2i-1}\left(1 - x_{2i}^2\right)\right)^2 + \left(2.625 - x_{2i-1}\left(1 - x_{2i}^3\right)\right)^2,$$
$$x_0 = [1, 0.8, \ldots, 1, 0.8].$$

Table 11a. Number of iterations and the average steplength of GD and RGD.

	GD		**RGD**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	1236	0.0427481	449	0.192988
2000	1236	0.0427117	478	0.186444
3000	1281	0.0426884	482	0.187414
4000	1293	0.0426732	487	0.188121
5000	1302	0.0426619	487	0.188121

Table 11b. Number of iterations and the average steplength of BB and NA.

	BB		**NA**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	46	0.39584	63	0.385026
2000	46	0.39584	63	0.385114
3000	46	0.39584	63	0.385179
4000	46	0.39584	63	0.385108
5000	46	0.39584	63	0.385691

Table 11c. Number of iterations of BB.

n	**# iter**	γ
10000	46	0
20000	46	0
30000	46	0
40000	46	0
50000	46	0

Table 11d. Number of iterations of NA

n	**# iter**	γ
10000	63	0
20000	63	0
30000	63	0
40000	63	0
50000	63	0

Example 12. (Freudenstein and Roth function)

$$f(x) = \sum_{i=1}^{n/2} \left(-13 + x_{2i-1} + ((5 - x_{2i})\, x_{2i} - 2)\, x_{2i}\right)^2 +$$
$$\left(-29 + x_{2i-1} + ((x_{2i} + 1)\, x_{2i} - 14)\, x_{2i}\right)^2,$$
$$x_0 = [0.5, -2, \ldots, 0.5, -2].$$

Table 12a. Number of iterations and the average steplength of GD and RGD.

	GD		**RGD**	
n	**# iter**	**average step**	**# iter**	**Average step**
100	10381	0.0005367	679	0.0122165
200	10564	0.0005368	679	0.0122165
300	10564	0.0005368	679	0.0122165
400	11240	0.0005370	679	0.0122165
500	11240	0.0005370	679	0.0122165

Table 12b. Number of iterations and the average steplength of BB and NA.

	BB		**NA**	
n	**# iter**	**average step**	**# iter**	**average step**
1000	220	0.018691	25	0.04292
2000	245	0.018652	25	0.04292
3000	250	0.018069	25	0.04292
4000	173	0.025440	25	0.04292
5000	138	0.021845	25	0.04291

Table 12c. Number of iterations of BB.

n	**# iter**	γ
10000	183	0
20000	395	0
30000	380	0
40000	218	0
50000	205	0

Table 12d. Number of iterations of NA for different values of δ.

	$\delta = 0.01$		$\delta = 0.1$		$\delta = 1$		$\delta = 10$		$\delta = 100$	
n	**# iter**	γ	**# iter**	γ	**# iter**	γ	**# iter**	γ	**# iter**	γ
10000	25	0	25	0	25	0	25	0	25	0
20000	25	0	25	0	25	0	25	0	25	0
30000	25	0	25	0	25	0	25	0	25	0
40000	25	0	25	0	25	0	25	0	25	0
50000	29	1	27	1	27	1	32	1	27	1

Some comments are in order.

Both algorithms are linear convergent. Modifying only the steplength along the negative gradient we get linear convergent algorithms, i.e. the error $f(x_k) - f^*$ converges to zero approximately as a geometric series.

The convergence rate depends greatly on the condition number of the Hessian of the minimizing function. For well conditioned convex functions both algorithms are improvements of the classical gradient descent algorithm. For ill-conditioned functions these algorithms, like any gradient descent one, are so slow that they have no value in practice.

Generally, $\gamma(x_{k+1}) \geq \gamma_k^{BB}$ showing that the initial step in backtracking procedure of NA is lower than the corresponding initial step of Barzilai-Borwein approach. However, along the iterations $\gamma(x_{k+1})$ is very close to γ_k^{BB}. This give us the motivation to modify the Barzilai-Borwein algorithm when $\gamma_k^{BB} < 0$. If $\gamma_k^{BB} < 0$, as in exemples 3 and 10 above, then we can consider in the Barzilai-Borwein algorithm $\gamma(x_{k+1})$ instead of γ_k^{BB}.

Refering to the number of iterations corresponding to GD, RGD, BB and NA algorithms, from the above tables the following cumulative results have been obtained.

Table 13. Number of iterations. Cumulative results for tables *a and *b.

Nr. Ex.	**GD**	**RGD**	**BB**	**NA**
Ex1	88786	6070	18723	18675
Ex2	24480	6367	7003	5528
Ex3	5103	1099	644	547
Ex4	674	214	236	261
Ex5	5333	1428	1087	970
Ex6	1484	882	430	388
Ex7	519	1027	169	168
Ex8	5332	2241	1147	1021
Ex9	716	152	75	60
Ex10	26223	38516	7832	3367
Ex11	6375	2383	230	315
Ex12	53989	3395	1026	125
TOTAL	219014	63774	38602	31425

Table 14. Number of iterations. Cumulative results for tables *c and *d.

Large-scale problems. (NA with $\delta = 100$)

Nr. Ex.	BB	NA
Ex2	12370	12304
Ex3	887	320
Ex4	291	344
Ex5	1037	984
Ex6	439	408
Ex7	182	187
Ex8	1072	1033
Ex9	79	60
Ex10	9420	3602
Ex11	230	315
Ex12	1381	127
TOTAL	27388	19684

5. Conclusion

The main contributions of this paper are as following. Firstly, it extends the relaxation idea considered by Raydan and Svaiter [30] for the quadratic positive definite functions to the nonlinear convex optimization. It is shown that a simple modification of the steplength by means of a random variable uniformly distributed in $(0, 1)$, for the strongly convex functions, represents an improvement of the classical gradient descent algorithm. Secondly, a new gradient descent algorithm is proposed in which the step length is computed by backtracking using a simple approximation of the Hessian. This new approach compares favourable with Barzilai-Borweins, for well conditioned convex functions this being an improvement of the classical gradient descent algorithm. The general conclusion is that using only the local information given by the gradient of the minimizing function, any procedure for step length computation, does not change the linear convergence property of the gradient descent algorithms.

References

1. L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. Pacific Journal of Mathematics, vol.6:1-3, 1966.

2. J. Barzilai and J.M. Borwein. Two point step size gradient method. IMA J. Numer. Anal., 8: 141-148, 1988.

3. E.G. Birgin, J.M. Martinez and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. SIAM J. Optim., 10:1196-1211, 2000.

4. A. Cauchy, Méthodes générales pour la résolution des systèmes déquations simultanées, C.R. Acad. Sci. Par., vol.25, pp.536-538, 1847.

5. Y.H. Dai and L.Z. Liao. R-linear convergence of the Barzilai and Borwein gradient method. IMA J. Numer. Anal., 22:1-10, 2002.

6. R.S. Dembo, S.C. Eisenstat and T. Steihaug. Inexact Newton methods. SIAM J. Numer. Anal., vol.19:400-408, 1982.

7. J.E. Dennis ans R.B. Schnabel. Numerical Methods for Unconstrained Optimization

and Nonlinear Equations. Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

8. R. Fletcher. Practical Methods of Optimization. John Wiley and Sons, New York, 1987.

9. R. Fletcher. On the Barzilai-Borwein method. Numerical Analysis Report NA/207, 2001.

10. R. Fletcher and C.M. Reeves. Function minimization by conjugate gradients. Computer Journal, vol.7:149-154, 1964.

11. G.E., Forsythe and T.S., Motzkin, Asymptotic properties of the optimum gradient method. Bull. American Society, vol.57, 1951, pp.183.

12. A. Friedlander, J.M. Martinez, B. Molina and M. Raydan. Gradient method with retards and generalizations. SIAM J. Numer. Anal. vol.36:275-289, 1999.

13. A. Friedlander, J.M. Martinez and M. Raydan. A new method for large-scale box constrained convex quadratic minimization problems. Optimization Methods and Software, 5:55-74, 1995.

14. W. Glunt, T.L. Hayden and M. Raydan. Molecular conformations from distance matrices, J. Comput. Chem., 14:114-120, 1993.

15. W. Glunt, T.L. Hayden and M. Raydan. Preconditioners for distance matrix algorithms. J. Comput. Chem., 15:227-232, 1994.

16. A.A. Goldstein. On steepest descent. SIAM Journal on Control, vol.3:147-151, 1965.

17. L. Grippo, F. Lampariello and S. Lucidi. A nonmonotone line search technique for Newtons method. SIAM J. Numer. Anal., 23:707-716, 1986.

18. L. Grippo and M. Sciandrone. Nonmonotone globalization techniques for the Barzilai-Borwein gradient method. Computational Optimization and Applications, 23:143-169, 2002.

19. W. E., Humphrey, A general minimising routine - minfun. in A. Lavi, and T.P. Vogl, (Eds.), Recent Advances in Optimisation Techniques, John Wiley, 1966.

20. C. Lemaréchal. A view of line search. in A. Auslander, W. Oettli and J. Stoer (Eds.) Optimization and Optimal Control, Springer Verlag, pp.59-78, 1981.

21. B. Molina and M. Raydan. Preconditioned Barzilai-Borwein method for the numerical solution of partial differential equations. Numerical Algorithms, 13:45-60, 1996.

22. J. Moré and D.J. Thuente. On line search algorithms with guaranteed sufficient decrease. Mathematics and Computer Science Division Preprint MCS-P153-0590, Argonne National Laboratory, Argonne, 1990.

23. S. Nash. A survey of Truncated-Newton methods. Journal of Computational and Applied Mathematics, vol.124:45-59, 2000.

24. J. Nocedal. Updating quasi-Newton matrices with limited starage. Math. of Comput., vol.35:773-782, 1980.

25. E. Polak and G. Ribière. Note sur la convergence de methodes de directions conjugat. Revue Francaise dInformatique et Recherche Operationnelle, 16:35-43, 1969.

26. F.A. Potra and Y. Shi. Efficient line search algorithm for unconstrained optimization. JOTA, vol.85, no.3:677-704, 1995.

27. M.J.D. Powell. Some global convergence properties of a variable-metric algorithm for minimization without exact line searches. SIAM-AMS Proceedings, Philadel-

phia, vol.9:53-72, 1976.

28. M. Raydan. On the Barzilai and Borwein choice of the steplength for the gradient method. IMA J. Numer. Anal., 13:618-622, 1993.

29. M. Raydan. The Barzilai and Borwein gradient method for the lasge scale unconstrained minimization problem. SIAM J. Optim., 7:26-33, 1997.

30. M. Raydan and B.F. Svaiter, Relaxed Steepest Descent and Cauchy-Barzilai-Borwein Method, Computational Optimization and Applications, 21, pp.155-167, 2002.

31. R., Schinzinger,Optimization in electromagnetic system design. in A. Lavi, and T.P. Vogl (Eds.), Recent Advances in Optimisation Techniques, John Wiley, 1966.

32. D.F. Shanno and K.H. Phua. Remark on Algorithm 500: Minimization of unconstrained multivariate functions. ACM Trans. Math. Software, vol.6:618-622, 1980.

33. P. Wolfe. Convergence conditions for ascent methods. SIAM Review, vol.11:226-235, 1968.