Faculty of Sciences and Mathematics, University of Niš, Serbia
Available at: `http://www.pmf.ni.ac.rs/filomat`

Filomat **23:3** (2009), 23–36 DOI:10.2298/FIL0903023S

MULTIPLICATIVE PARAMETERS IN GRADIENT DESCENT METHODS

Predrag Stanimirović[*], Marko Miladinović [†] and Snežana Djordjević

Abstract

We introduced an algorithm for unconstrained optimization based on the reduction of the modified Newton method with line search into a gradient descent method. Main idea used in the algorithm construction is approximation of Hessian by a diagonal matrix. The step length calculation algorithm is based on the Taylor's development in two successive iterative points and the backtracking line search procedure.

1 Introduction

We consider the problem

$$\min f(x), \quad x \in \mathbb{R}^n, \tag{1.1}$$

where $\mathbb{R}^n$ denotes the set of n-tuples with elements from $\mathbb{R}$ and $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function. It is assumed that the function $f : \mathbb{R}^n \to \mathbb{R}$ is uniformly convex and twice continuously differentiable. The most frequently used general iterative scheme aimed to solve the problem (1.1) is given by

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

where x_{k+1} is new iterative point, x_k is previous iterative point, $t_k > 0$ is a step-length, and d_k is an appropriate search direction. The key problem is to find the descent direction vector d_k and a suitable step size t_k.

As usual, we use the following notations:

$$g(x) = \nabla f(x),\ G(x) = \nabla^2 f(x),\ g_k = \nabla f(x_k),\ G_k = \nabla^2 f(x_k),$$

[*]Corresponding author

[†]The authors gratefully acknowledge support from the research project 144011 of the Serbian Ministry of Science.

2000 *Mathematics Subject Classifications*. 90C30.

Key words and Phrases. Line search, gradient descent methods, Newton method, backtracking line search, convergence rate.

Received: March 20, 2009

Communicated by Dragan S. Djordjević

where $\nabla f(x)$ denotes the gradient of f and $\nabla^2 f(x)$ denotes the Hessian of f. For $x \in \mathbb{R}^n$, x^T denotes the transpose of x.

The search direction d_k is generally required to satisfy the descent condition

$$g_k^T d_k < 0.$$

For the choice of the search direction there are several procedures [13, 20]. In the Newton method with line search the descent direction d_k is generated by solving the system $G_k d = -g_k$. In the gradient descent method d_k is defined by $d_k = -g_k$.

Different choices of step-sizes lead to various gradient algorithms. The line search algorithms determine the step size t_k such that the objective function value decreases, i.e. $f(x_k + t_k d_k) < f(x_k)$. Two main algorithms have been proposed: exact line search and inexact line search. In the exact line search there is the well-known formula (see, for example [17], [20])

$$t_k = \arg\min_{t>0} f(x_k + t d_k).$$

In practice the most frequently used is the inexact line search which try to minimize f along the ray $x_k + t d_k$, $t \geq 0$, or to sufficiently decrease the value of f.

The general iterative scheme for the realization of the line search methods is given as follows [20]:

Algorithm 1.1 General scheme for line search methods.

```
Require: Objective function $f(x)$, initial point $x_0 \in R^n$ and tolerance $0 \leq \varepsilon \ll 1$.
1: $k = 0$.
2: (Verify termination criterion) If $||g_k|| \leq \varepsilon$ then return $x_k$ and $f(x_k)$ and stop
   the algorithm; else continue by the next step.
3: (Finding the direction) Find the vector $d_k$ which is a descent direction.
4: (Finding the stepsize) Find the stepsize $t_k$ such that
       $f(x_k + t_k d_k) < f(x_k)$.
5: (Loop) Compute $x_{k+1} := x_k + t_k d_k$; set $k := k + 1$ and go to Step 2.
```

For the sake of completeness, we describe the inexact line search known as the backtracking, introduced in [4].

Algorithm 1.2 The backtracking line search.

```
Require: Objective function $f(x)$, the direction $d_k$ of the search at the point $x_k$
   and numbers $0 < \sigma < 0.5$ and $\beta \in (0, 1)$.
1: $t = 1$.
2: While $f(x_k + t d_k) > f(x_k) + \sigma t g_k^T d_k$, take $t := t\beta$.
3: Return $t_k = t$.
```

The standard values of the parameters are $\sigma = 0.0001$ and $\beta = 0.8$.

The paper is organized as follows. In the second section we describe motivation and main directives used in the present article. In the third section we introduce several variants of a gradient descent method arising from the modified Newton method with line search, developed after the reduction of the Hessian by an appropriately generated diagonal matrix and using the backtracking line search for the step length calculation. Linear convergence of the algorithm is proved in Section 4. Numerical results obtained applying the implementation of the method in the programming language MATHEMATICA are presented in the last section.

2 Motivation

We briefly describe our idea used in the present paper. The Newton's method with the line search is defined by the iterative scheme

$$x_{k+1} = x_k - t_k G_k^{-1} g_k.$$

However, for various practical problems, the efforts for computing the Hessian matrices and their inverses are very expensive, or the Hessian is not available analytically. These difficulties initiate a class of methods that only uses the function values and the gradients of the objective function and that is closely related to the Newton's method. Modified Newton methods do not need to compute the Hessian, but generates a series of Hessian approximations, and at the same time maintains a fast rate of convergence. This can be accomplished by constructing approximations to the inverse Hessian based on information gathered during the descent process [13, 20]. In this way, the general iterative scheme for modified Newton methods is

$$x_{k+1} = x_k - t_k S_k g_k, \tag{2.1}$$

where, in general, S_k is selected as an approximation to the inverse of the Hessian.

If we take the following diagonal approximation to the inverse of the Hessian

$$S_k = \gamma_k^{-1} I, \quad \gamma_k \in \mathbb{R}, \tag{2.2}$$

the modified Newton method (2.1) reduces to the iterative scheme

$$x_{k+1} = x_k - t_k \gamma_k^{-1} g_k, \tag{2.3}$$

which is essentially a technique for the step length calculation in the frame of the gradient descent general iterative scheme.

Our general decision to approximate the Hessian by a diagonal matrix was used by Barzilai and Borwein in [5]. They suggested an algorithm (called as *BB* algorithm) which essentially is a gradient one, where the choice of the step size along the negative gradient is derived from a two-point approximation to the secant equation underlying quasi-Newton methods:

$$S_k y_{k-1} = s_{k-1},$$

where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. Considering $S_k = \gamma_k I$ as an approximation to the Hessian of f at x_k, in order to make the matrix S_k have quasi-Newton property, Barzilai and Borwein chose γ_k according to the rule

$$S_k = \arg\min \|Gs_{k-1} - y_{k-1}\|_2,$$

yielding the following iterative scheme:

$$x_{k+1} = x_k - \frac{1}{\gamma_k^{BB}} g_k, \quad \gamma_k^{BB} = \frac{s_{k-1}^T y_{k-1}}{s_{k-1}^T s_{k-1}}. \tag{2.4}$$

In the recent years lots of researches have been done on how to choose the step size for the gradient method, following the amazing result by Barzilai and Borwein; for example, see [6, 7, 8, 9, 10, 11, 12, 14, 15, 18, 19].

Our diagonal matrix is computed without using the secant equation from quasi-Newton methods. Therefore, our method starts from the modified Newton methods with line search.

On the other hand, we observed that the iterative scheme (2.3) possesses the general form of the accelerated gradient descent methods from [2], which is defined as the gradient descent iterative scheme

$$x_{k+1} = x_k - \theta_k t_k g_k, \tag{2.5}$$

where $\theta_k > 0$ is the acceleration parameter which improves the behavior of the gradient descent algorithm.

In this way, our approach is between the ideas used in [1], [2] and [5].

Formal comparison of the iterative schemes (2.3) and (2.5) consists of the computation of the real number γ_k in (2.3) with respect to the construction of the modification parameter θ_k in (2.5). The step length t_k is a common scalar which appears in both iterative processes (2.3) and (2.5). Following essential ideas in these iterative schemes, we observed that (2.5) tends to accelerate the basic gradient descent algorithm, while the iterations (2.3) become from a modification of the Newton method given by the general form (2.1).

3 Gradient descent method

Main idea used in the algorithm construction is approximation of the Hessian in the modified Newton method by a diagonal matrix using (2.2), where $\gamma_k = \gamma(x_k, x_{k-1})$ is appropriately selected real number on the basis of Taylor's approximation in the point x_{k+1}, computed by means of (2.3). In order to select an appropriately real number γ_k we extend the idea for the step length selection in the frame of gradient descent method from [1]. In the $(k+1)$-th iteration of the iterative scheme (2.3) we have from the Taylors theorem

$$f(x_{k+1}) = f(x_k) - t_k g_k^T \gamma_k^{-1} g_k + \frac{1}{2} t_k^2 (\gamma_k^{-1} g_k)^T \nabla^2 f(\xi) \gamma_k^{-1} g_k, \tag{3.1}$$

where $\xi \in [x_k, x_{k+1}]$ is defined by

$$\xi = x_k + \alpha(x_{k+1} - x_k) = x_k - \alpha t_k \gamma_k^{-1} g_k,\ 0 \leq \alpha \leq 1. \tag{3.2}$$

If x_k and x_{k+1} are close enough, it is reasonable take $\alpha = 1$ and obtain the approximation $\xi = x_{k+1}$. Besides, we take the approximation (2.2) of the Hessian and obtain

$$\nabla^2 f(\xi) = \gamma_{k+1} I, \tag{3.3}$$

where $\gamma_{k+1} = \gamma(x_k, x_{k-1})$ is defined appropriately in $(k+1)$-th iteration. Now, from (3.1) and (3.3) it is not difficult to verify

$$f(x_{k+1}) = f(x_k) - t_k \gamma_k^{-1} \|g_k\|^2 + \frac{1}{2} t_k^2 \gamma_{k+1} \gamma_k^{-2} \|g_k\|^2, \tag{3.4}$$

and later

$$\gamma_{k+1} = 2\gamma_k \frac{\gamma_k \left[f(x_{k+1}) - f(x_k)\right] + t_k \|g_k\|^2}{t_k^2 \|g_k\|^2}. \tag{3.5}$$

It is clear that the condition $\gamma_k > 0$ is indispensable. We propose three solutions for the situation $\gamma_{k+1} < 0$.

(I) In the case $\gamma_{k+1} < 0$ simply take $\gamma_{k+1} = 1$. Motivation for such solution is: when G_k is not positive definite, the gradient descent direction $-g_k$ with the step size $t_k = 1$ is used.

(II) This solution is a generalization of the idea used in [1]. From (3.5) we obtain (assuming that in the previous iterative step $\gamma_k > 0$ is satisfied)

$$\gamma_{k+1} < 0 \Leftrightarrow f(x_k) - f(x_{k+1}) > t_k \gamma_k^{-1} \|g_k\|^2.$$

Let a real value η_k be chosen such that the following inequality holds:

$$f(x_k) - f(x_{k+1}) < (t_k + \eta_k) \gamma_k^{-1} \|g_k\|^2.$$

We compute η_k as a real number satisfying

$$t_k + \eta_k > \frac{\gamma_k (f(x_k) - f(x_{k+1}))}{\|g_k\|^2}.$$

It follows that

$$\eta_k > \frac{\gamma_k (f(x_k) - f(x_{k+1}))}{\|g_k\|^2} - t_k.$$

Hence, we can take

$$\eta_k = \frac{f(x_k) - f(x_{k+1})}{\gamma_k^{-1} \|g_k\|^2} - t_k + \delta,\ \delta > 0, \tag{3.6}$$

where $\delta > 0$ is a chosen positive real number. So, we obtain

$$\gamma_{k+1} = 2\gamma_k \frac{\gamma_k \left[f(x_{k+1}) - f(x_k)\right] + (t_k + \eta_k) \|g_k\|^2}{(t_k + \eta_k)^2 \|g_k\|^2}. \tag{3.7}$$

(III) The third idea is to use $\gamma_{k+1} = \alpha_{k+1} + \gamma_{k+1} = \delta > 0$. Motivation arises from the following approach: to avoid the case when G_{k+1} is not positive definite it is possible to change the Hessian G_{k+1} to $G_{k+1} + \alpha_{k+1} I$, where $\alpha_{k+1} \geq 0$ is chosen such that $G_{k+1} + \alpha_{k+1} I$ is positive definite and well-conditioned. This can be regarded as a kind of compromise between steepest descent (α_k very large) and Newtons method ($\alpha_k = 0$). Ideally, α_k is not much larger than the smallest that makes $G_{k+1} + \alpha_{k+1} I$ positive definite and well-conditioned.

In our implementation we used the solution **(I)**, since the solution **(II)** requires relatively complicated computations and implies greater CPU times in initiated iterative scheme.

After the parameter $\gamma_{k+1} > 0$ is found, we need to compute to find the parameter t_{k+1} to determine the next iterative estimation $x_{k+2} = x_{k+1} - t_{k+1}\gamma_{k+1}^{-1} g_{k+1}$ we use the inexact backtracking line search procedure for generating t_{k+1}. In order to derive an upper bound for the backtracking line search we analyze the function

$$\Phi_{k+1}(t) = f(x_{k+1}) - t g_{k+1}^T \gamma_{k+1}^{-1} g_{k+1} + \frac{1}{2} t^2 (\gamma_{k+1}^{-1} g_{k+1})^T \nabla^2 f(\xi) \gamma_{k+1}^{-1} g_{k+1},$$

where $\xi \in [x_{k+1}, x_{k+2}]$, $t \geq 0$ and $\gamma_{k+1} > 0$. In the case, when the approximation $\xi \approx x_{k+1}$ is applied, we obtain

$$\begin{aligned} \Phi_{k+1}(t) &= f(x_{k+1}) - t\gamma_{k+1}^{-1} \|g_{k+1}\|^2 + \frac{1}{2} t^2 \gamma_{k+1} \gamma_{k+1}^{-2} \|g_{k+1}\|^2 \\ &= f(x_{k+1}) - t\gamma_{k+1}^{-1} \|g_{k+1}\|^2 + \frac{1}{2} t^2 \gamma_{k+1}^{-1} \|g_{k+1}\|^2. \end{aligned}$$

It is obvious that $\Phi_{k+1}(0) = f(x_{k+1})$ and

$$\Phi'_{k+1}(0) = -\gamma_{k+1}^{-1} \|g_{k+1}\|^2 < 0.$$

In the case $\gamma_{k+1} > 0$ the function Φ is convex for every $t \geq 0$. We have

$$\Phi'_{k+1}(t) = 0 \Leftrightarrow \bar{t}_{k+1} = 1, \tag{3.8}$$

and the minimal value for $\Phi_{k+1}(t)$ is equal to

$$\Phi_{k+1}(1) = f(x_{k+1}) - \frac{1}{2} \gamma_{k+1}^{-1} \|g_{k+1}\|^2.$$

It is clear that f is reduced if $\gamma_{k+1} > 0$. In accordance with (3.8), we can take

$$\begin{aligned} t_{k+1} &= \arg\min_{t \leq 1} f(x_{k+1} - t G_{k+1}^{-1} g_{k+1}) \\ &\approx \arg\min_{t \leq 1} f(x_{k+1} - t \gamma_{k+1}^{-1} g_{k+1}) \end{aligned} \tag{3.9}$$

using the backtracking line search procedure.

Since the search direction and the step size are found, we are in a position to introduce the following algorithm:

Algorithm 3.1 The gradient descent algorithm defined by (2.3) and (3.5) (GDQN).

```
Require: Objective function $f(x)$ and chosen initial point $x_0 \in dom(f)$.
1: Set $k = 0$ and compute $f(x_0)$, $g_0 = \nabla f(x_0)$ and use $\gamma_0 = 1$.
2: If test criteria is fulfilled, then stop the iteration; otherwise, go to the next step.
3: Compute $t_k = \arg \min_{t<1} f(x_k - t_k \gamma_k^{-1} g_k)$ using the backtracking line search procedure.
4: Compute $x_{k+1} = x_k - t_k \gamma_k^{-1} g_k,\ f(x_{k+1})$ and $g_{k+1} = \nabla f(x_{k+1})$.
5: Compute the scalar approximation $\gamma_{k+1}$ of the Hessian of $f$ at the point $x_{k+1}$
   using (3.5). If $\gamma_{k+1} > 0$ go to Step 7, otherwise go to the next step.
6: If $\gamma_{k+1} < 0$, then apply one of the following two solutions:
   (1) Use $\gamma_{k+1} = 1$.
   (2) Use $\eta_k$ as it is defined in (3.6) and later take the positive value of $\gamma_{k+1}$ as in (3.7).
   (3) Select $\alpha_{k+1} \geq 0$ such that $\alpha_{k+1} + \gamma_{k+1} > 0$ and set $\gamma_{k+1} = \alpha_{k+1} + \gamma_{k+1}$.
7: Using the backtracking procedure, compute the length of the $t_{k+1}$ of the step as
   $$t_{k+1} = \arg \min_{t \leq 1} f(x_{k+1} - t\gamma_{k+1}^{-1} g_{k+1}).$$
8: Take $x_{k+2} = x_{k+1} - t_{k+1}\gamma_{k+1}^{-1} g_{k+1}$, set $k := k+1$, and go to Step 2.
9: Return $x_{k+1}$ and $f(x_{k+1})$.
```

In the rest of this section we introduce another variant of Algorithm *GDQN*. Starting from (3.2), instead of (3.3) and (3.5) we use

$$\nabla^2 f(\xi) = \gamma(\xi) I, \tag{3.10}$$

where

$$\gamma(\xi) = \gamma(x_k) + \alpha_k \left(\gamma_{k+1} - \gamma_k\right),\ \alpha_k \in (0,1),\ k = 0, 1, \ldots.$$

For example, we can use

$$\gamma(\xi) = \gamma_k + t_k \left(\gamma_{k+1} - \gamma_k\right),\ k = 0, 1, \ldots. \tag{3.11}$$

In view of (3.11) and (3.5) we derive the following heuristic for computing γ_{k+1}:

$$\gamma_{k+1} \approx \gamma(\xi) = \gamma_k \frac{2\gamma_k (f(x_{k+1}) - f(x_k)) + \|g_k\|^2 (3t_k - t_k^2)}{t_k \|g_k\|^2}. \tag{3.12}$$

In this way, we introduce the following algorithm, which differs from Algorithm 3 in Step 5, which is now defined as in the following:

5: Compute the scalar approximation γ_{k+1} of the Hessian of f at the point x_{k+1} using (3.12). If $\gamma_{k+1} > 0$ go to Step 7, otherwise go to the next step.

4 Convergence of algorithm GDQN

Proposition 4.1. *If the function $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and uniformly convex on $\mathbb{R}^n$ then:*

1) *the function f has a lower bound on $L_0 = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\}$, where $x_0 \in \mathbb{R}^n$ is available;*

2) *the gradient g is Lipschitz continuous in an open convex set B which contains L_0, i.e. there exists $L > 0$ such that*
$$\|g(x) - g(y)\| \leq L\|x - y\|, \quad \forall\, x, y \in B;$$

3) *$f(x)$ has a unique minimizer x^*, and there exist real numbers m, M satisfying $0 < m \leq M$ such that*
$$m\|y\|^2 \leq y^T \nabla^2 f(x) y \leq M\|y\|^2, \quad \forall\, x, y \in \mathbb{R}^n; \tag{4.1}$$
$$\frac{1}{2}m\|x - x^*\|^2 \leq f(x) - f(x^*) \leq \frac{1}{2}M\|x - x^*\|^2, \quad \forall\, x \in \mathbb{R}^n; \tag{4.2}$$
$$m\|x - y\|^2 \leq (g(x) - g(y))^T(x - y) \leq M\|x - y\|^2, \quad \forall\, x, y \in \mathbb{R}^n. \tag{4.3}$$

The proof can be found in [16].

If we put $x^* = y$ in the last inequality, and apply mean value Theorem as well as the Cauchy-Schwartz inequality we get
$$m\|x - x^*\| \leq \|g(x)\| \leq M\|x - x^*\|, \quad \forall\, x \in \mathbb{R}^n. \tag{4.4}$$

Lemma 4.1. *For twice continuously differentiable and uniformly convex function f on $\mathbb{R}^n$, and for sequence $\{x_k\}$ generated by Algorithm GDQN we have*
$$f(x_k) - f(x_{k+1}) \geq \mu\|g_k\|^2, \tag{4.5}$$
where
$$\mu = \min\left\{\frac{\sigma}{M}, \frac{\sigma(1-\sigma)}{L}\beta\right\}. \tag{4.6}$$

Proof. Let us observe the following sets $K_1 = \{k \mid t_k = 1\}$, $K_2 = \{k \mid t_k < 1\}$ from Algorithm 3. Using the backtracking scheme from Algorithm 1.2 we have the following two inequalities:
$$f(x_k) - f(x_{k+1}) \geq -\sigma g_k^T d_k, \quad \forall\, k \in K_1, \tag{4.7}$$
$$f(x_k) - f(x_{k+1}) \geq -\sigma t_k g_k^T d_k, \quad \forall\, k \in K_2. \tag{4.8}$$

Since $t_k/\beta \leq 1$, $\forall\, k \in K_2$, we get
$$f(x_k) - f\left(x_k + \frac{t_k}{\beta}d_k\right) < -\sigma\frac{t_k}{\beta}g_k^T d_k, \quad \forall\, k \in K_2.$$

Now, applying the mean value Theorem on the left hand side of the above inequality, we have that $\exists\theta_k \in (0,1)$ such that
$$-\frac{t_k}{\beta}g\left(x_k + \theta\frac{t_k}{\beta}d_k\right)^T d_k < -\sigma\frac{t_k}{\beta}g_k^T d_k, \quad k \in K_2,$$

and hence

$$g\left(x_k+\theta\frac{t_k}{\beta}d_k\right)^T d_k > \sigma g_k^T d_k, \quad k\in K_2. \tag{4.9}$$

If we subtract $g_k^T d_k$ from both hand sides of the above inequality, and apply Cauchy-Schwartz inequality as well as the statement 2) from Lemma 4.1 we have

$$-(1-\sigma)g_k^T d_k < \left(g\left(x_k+\theta\frac{t_k}{\beta}d_k\right)-g_k\right)^T d_k \le \frac{L}{\beta}t_k\|d_k\|^2, \quad k\in K_2,$$

which implies

$$t_k > -\frac{\beta(1-\sigma)}{L}\frac{g_k^T d_k}{\|d_k\|^2}, \quad k\in K_2.$$

After the substitution $d_k=-\gamma_k^{-1}g_k$ we obtain from the last inequality

$$t_k > \frac{\beta(1-\sigma)\gamma_k}{L}, \quad k\in K_2. \tag{4.10}$$

Applying (4.10) into the (4.8) we get

$$f(x_k)-f(x_{k+1}) \ge -\sigma t_k g_k^T d_k > \frac{-\sigma(1-\sigma)\beta\gamma_k}{L}\frac{-g_k^T g_k}{\gamma_k} > \frac{\sigma(1-\sigma)\beta}{L}\|g_k\|^2, \quad \forall\, k\in K_2.$$

On the other hand, using the fact that $\gamma_k < M$ (follows from (4.1)) in the case $k\in K_1$ we have

$$f(x_k)-f(x_{k+1}) \ge -\sigma g_k^T d_k \ge \frac{\sigma}{M}\|g_k\|^2, \quad \forall\, k\in K_1.$$

Finally from the last two inequalities the proof follows immediately. □

Theorem 4.1. *If the objective function f is twice continuously differentiable as well as uniformly convex on $\mathbb{R}^n$, then the sequence $\{x_k\}$ generated by Algorithm GDQN satisfies*

$$\lim_{k\to\infty}\|g(x_k)\|=0, \tag{4.11}$$

and converges to x^* at least linearly.

Proof. Starting from the inequality (4.5), after applying (4.4) and (4.2) we have

$$\begin{aligned} f(x_k)-f(x_{k+1}) &\ge \mu\|g_k\|^2 \ge \mu m^2\|x_k-x^*\|^2 \\ &\ge 2\mu\frac{m^2}{M}(f(x_k)-f(x^*)). \end{aligned}$$

Since the objective f is bounded below and $f(x_k)$ decreases, it follows that $f(x_k)\to f(x_{k+1})$. Therefore we proved $\lim_{k\to\infty}\|g(x_k)\|=0$ as well as convergence of the sequence $\{f(x_k)\}$ to $f(x^*)$, which implies $\lim_{k\to\infty}x_k=x^*$. It remains to prove that

$$2\mu\frac{m^2}{M}<1.$$

In the case $\mu = \sigma/M$ we have

$$2\mu\frac{m^2}{M} = 2\frac{\sigma m^2}{M^2} < 1.$$

Similarly, in the case $\mu = \sigma(1-\sigma)\beta/L$ we have

$$\mu\frac{m^2}{M} = 2\beta\frac{\sigma(1-\sigma)}{L}\frac{m^2}{M} < \frac{m^2}{ML} < 1,$$

because from (4.3) we have $m \le L$. □

Lemma 4.1 and Theorem 4.1 are derived using analogous results from [17]. In the following theorem we introduce some additional results about the behavior of *GDQN* algorithm.

Theorem 4.2. *Values of the goal function f decrease during the iterative steps of Algorithm GDQN inside the closed interval*

$$\left[f(x_k) - \frac{\|g_k\|^2}{2\gamma_{k+1}}, f(x_k)\right]. \tag{4.12}$$

Proof. From (3.4) we have

$$f(x_{k+1}) - f(x_k) = \frac{t_k\|g_k\|^2}{\gamma_k}\left(\frac{t_k\gamma_{k+1}}{2\gamma_k} - 1\right).$$

For every $k = 0, 1, \dots$ we have that the values γ_k, generated by Algorithm *GDQN*, are bounded away from zero, so that that the minimum of the function

$$\Psi(t) = \frac{t\|g_k\|^2}{\gamma_k}\left(\frac{t\gamma_{k+1}}{2\gamma_k} - 1\right)$$

is achieved at the point

$$t_{opt} = \frac{\gamma_k}{\gamma_{k+1}}$$

and it is equal to

$$\min \Psi(t) = -\frac{\|g_k\|^2}{2\gamma_{k+1}}.$$

So, the following holds

$$f(x_{k+1}) \ge f(x_k) - \frac{\|g_k\|^2}{2\gamma_{k+1}} \tag{4.13}$$

On the other hand, using (4.5) and (4.6), we obtain

$$f(x_{k+1}) \le f(x_k),$$

which means, in conjunction with (4.13) that values of the goal function f decrease during the iterative steps and the decreasing is bounded inside the closed interval (4.12). □

5 Numerical results

Implementation of considered algorithms is performed in package MATHEMATICA. In Table 1. are compared results obtained applying the gradient descent (GD) method, AGD method from [2] and our method, denoted as $GDQN$using dimensions $n = 30$ for test functions from [3]. Numerical results corresponding to Algorithm $GDQN$ are derived using solution (1) in Step 6 in the case $\gamma_{k+1} < 0$. A streak in the table means a great number of iterative steps while the symbol ▲ means a wrong result produced. Summarized results are arranged in the last row of the table.

	Number of iterations			CPU time		
Test function	GD	GDQN	AGD	Steepest	GDQN	AGD
Extended Penalty	114	29	17	4	0.6	0.7
Perturbed quadratic	124	50	147	4.1	1.2	5
Raydan-1	102	35	99	1.8	0.6	1.8
Raydan-2	6	8	4	0.1	0.1	0.1
Diagonal1	48	33	50	1.2	0.6	1.2
Diagonal2	49	35	12	0.9	0.7	0.4
Diagonal3	95	35	108	2.5	0.7	2.8
Hager	51	15	23	1.1	0.4	0.6
Generalized Tridiagonal - 1	38	19	35	1.6	0.5	1.5
Extended Tridiagonal - 1	455	88	54	7.6	1.5	1
Extended Three Expon	123	11	25	3.8	0.3	0.9
Diagonal4	471	8	8	9.4	0.1	0.1
Diagonal5	4	6	3	0.2	0.2	0.3
Extended Himmelblau	87	15	23	2.5	0.3	0.7
Generalized PSC1	506	25	14	20	1	0.7
Extended PSC1	92	10	16	2.4	0.2	0.5
Extended Block Diagonal BD1	45	16	17	1.2	0.3	0.5
Quadr. Diag. Perturbed	200	17	230	6.2	0.4	7.1
Quadratic QF1	132	44	154	2.3	0.5	2.6
Extended Quad. Penalty QP1	42	14	9	1.5	0.4	0.4
Extended EP1	32	5	3	1.7	0.1	0.2
Extended Tridiagonal - 2	39	22	66	1.4	0.7	2.4
Tridia	1055	253	1427	43.8	5.3	60.6
Arwhead	61	13	11	2.5	0.3	0.5
Dqdrtic	819	106	1071	18.7	1.2	23.9
Almost Perturbed	119	50	148	2.2	0.5	2.7
Tridiagonal Perturbed Quadratic	365	49	181	14.1	1.3	7.4
Liarwhd	415	46	302	22.4	1.5	16.3
Power (cute)	2764	851	3941	75.2	9.5	107.3
Quartc	2102	22	13	19.7	0.2	0.2
Dixon3dq (cute)	1377	320	1785	26.9	5.5	36.8
Biggsb1 (cute)	1165	353	1651	22.8	6.8	34.1
Generalized quartic	50	14	12	1.3	0.3	0.3
Diagonal 7	39	8	5	1.1	0.2	0.2
Diagonal 8	40	8	5	1.4	0.3	0.3
Full Hessian FH3	41	6	4	2.5	0.4	0.5
Himmelbg	/	3	5	/	0.1	0.1
Average	368.53	71.41	315.62	9.23	1.21	8.72

Table 1. Numerical results derived comparing GD, AGD and GDQN

In Table 2 are compared results obtained applying the methods $GDQN$ with the results obtained applying the Barzilai and Borwein from [5], (denoted by BB), as well as with the results obtained using the method called as NA in [1]. Comparison is performed using dimensions $n = 100$ for test functions from [3]. Summarized results are presented in the last row of the table.

	Number of iterations			CPU time		
Test function	GDQN	BB	NA	GDQN	BB	NA
1 Extended Penalty	36	32	35	26.9	9.8	28.6
2 Perturbed quadratic	78	72	72	48.5	6.8	47.4
3 Raydan-1	62	64	74	22.2	80.4	28.3
4 Raydan-2	8	7	7	3.9	10.3	4.1
5 Diagonal1	45	▲	/	18.5	/	/
6 Diagonal2	75	18	37	29.6	43.4	20.6
7 Diagonal3	75	51	50	30.2	108.4	23.2
8 Hager	23	18	18	11	34	10.4
9 Generalized Tridiagonal - 1	17	19	18	10.2	2.6	11.8
10 Extended Tridiagonal - 1	77	32	33	26.3	1.9	11.3
11 Extended Three Expon	11	8	10	4.8	1.5	5.1
12 Diagonal4	8	7	7	1.5	0.5	1.5
13 Diagonal5	6	4	5	7	16.3	9,y2
14 Extended Himmelblau	15	12	39	4.7	1.1	14.2
15 Generalized PSC1	25	22	24	19.8	10,y1	20.6
16 Extended PSC1	10	14	9	4.3	1.8	4.6
17 Extended Block Diagonal BD1	18	17	44	6.1	1.7	16.5
18 Quadr. Diag. Perturbed	27	19	19	14,5	2.5	10.9
19 Quadratic QF1	119	86	86	24.8	2.2	18.3
20 Extended Quad. Penalty QP1	12	10	11	8.4	4.1	9.4
21 Quadratic QF2	130	85	96	56	3.8	44.1
22 Extended EP1	5	3	4	2.3	1.4	2.6
23 Extended Tridiagonal - 2	21	23	24	13.2	3	16.25
24 Arwhead	12	11	15	5.8	1.9	8
25 Dqdrtic	104	35	35	21.6	1.2	7.5
26 Almost Perturbed	94	83	83	19.5	2.2	17.6
27 Tridiagonal Perturbed Quadratic	120	79	79	72.3	5.5	49.1
28 Liarwhd	84	39	44	50.3	4.5	26.6
29 Quartc	22	19	21	4.1	0.9	4.4
30 Generalized quartic	13	13	15	6	1.6	7.8
31 Diagonal 7	8	6	8	5.7	13.4	6.9
32 Diagonal 8	8	6	7	7.2	16	8
33 Full Hessian FH3	5	4	4	7.7	16.4	8.9
34 Himmelbg	3	7	2	0.8	1.5	0.9
Average	40.47	28.03	31.36	17.52	12.51	15.29

Table 2. Numerical results derived comparing GD, AGD and GDQN

6 Conclusion

If we approximate the Hessian of the objective function by an appropriate diagonal matrix we derive an accelerated gradient descent method (2.3) from the modified Newton method (2.1). Value lying on the main diagonal of the diagonal matrix becomes he acceleration parameter. In this way, our investigations continue results of articles [1], [2] and [5].

Linear convergence is proved using the general principles from [17] about the convergence of line search techniques.

Implementation of the GDQN methods as well as related methods AGD, NA and BB is performed in the package MATHEMATICA. Comparative criteria between these methods are both number of iterative steps and the CPU time. From Table 1 we conclude that our method $GDQN$ produces better results with respect to the gradient descent method and the AGD method from [2]. From Table 2 we conclude that the average CPU time as well as the average number of iterations is slightly smaller in the NA method with respect to $GDQN$ method. But, there are several test functions when the method $GDQN$ produces better results. The best performances shows BB method. Let us mention that the comparison of methods $GDQN$, AGD and NA with BB is relative, since $GDQN$, AGD and NA are based on the usage of the line search procedures in each iterative step, while BB uses the line search only in the initial step.

References

[1] N. Andrei, *A New Gradient Descent Method for Unconstrained Optimization*, http://www.ici.ro/camo/neculai/newstep.pdf, April 8, 2004.

[2] N. Andrei, *An acceleration of gradient descent algorithm with backtracking for unconstrained optimization*, Numer. Algor. **42** (2006), 63–73.

[3] N. Andrei, *An Unconstrained Optimization Test Functions Collection*, http://www.ici.ro/camo/journal/vol10/v10a10.pdf.

[4] L. Armijo, *Minimization of functions having Lipschitz first partial derivatives*, Pac. J. Math, **6** (1966), 1–3.

[5] J. Barzilai and J.M. Borwein, *Two point step size gradient method*, IMA J. Numer. Anal., **8** (1988) 141–148.

[6] Y.H. Dai, *Alternate step gradient method*, Report AMSS–2001–041, Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, 2001.

[7] Y.H. Dai and R. Fletcher, *On the asymptotic behaviour of some new gradient methods*, Numerical Analysis Report, NA/212, Dept. of Math. University of Dundee, Scotland, UK (2003).

[8] Y.H. Dai, J.Y. Yuan, and Y. Yuan, *Modified two-point step-size gradient methods for unconstrained optimization*, Computational Optimization and Applications, **22** (2002), 103–109.

[9] Y.H. Dai and Y. Yuan, *Alternate minimization gradient method*, IMA Journal of Numerical Analysis, **23** (2003) 377–393.

[10] Y.H. Dai and Y. Yuan, *Analysis of monotone gradient methods*, J. Industrial and Management Optimization, **1** (2005) 181–192.

[11] Y. H. Dai and H. Zhang, *An Adaptive Two-Point Step-size Gradient Method*, Research report, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, 2001.

[12] A. Friedlander, J. M. Martinez, B. Molina, and M. Raydan, *Gradient method with retards and generalizations*, SIAM J. Numer. Anal., **36** (1999), 275–289.

[13] D.G. Luenberg and Y. Ye, *Linear and nonlinear programming*, Springer Science+Business Media, LLC, New York, 2008.

[14] M. Raydan, *On the Barzilai and Borwein choice of steplength for the gradient method*, IMA J. Numer. Anal. **13** (1993) 321–326.

[15] M. Raydan, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM J. Optim., **7** (1997) 26–33.

[16] R.T.Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

[17] Z.-Jun Shi, *Convergence of line search methods for unconstrained optimization*, Appl. Math. Comput. **157** (2004), 393–405.

[18] M.N. Vrahatis, G.S. Androulakis, J.N. Lambrinos and G.D. Magoulas, *A class of gradient unconstrained minimization algorithms with adaptive step-size*, J. Comp. and Appl. Math. **114** (2000) 367–386.

[19] Y. Yuan, *A new stepsize for the steepest descent method*, Research report, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, 2004.

[20] Wenyu Sun and Ya-Xiang Yuan, *Optimization theory and methods: nonlinear programming*, Springer, 2006.

University of Niš, Department of Mathematics, Faculty of Science and Mathematics, Višegradska 33, 18000 Niš, Serbia

E-mail
P. Stanimirović: `pecko@pmf.ni.ac.rs`
M. Miladinović: `markomiladinovic@gmail.com`
S. Djordjević: `gospava48@ptt.rs`