Numer Algor (2010) 54:503–520
DOI 10.1007/s11075-009-9350-8

ORIGINAL PAPER

Accelerated gradient descent methods with line search

Predrag S. Stanimirović · Marko B. Miladinović

Received: 14 April 2009 / Accepted: 9 November 2009 /
Published online: 10 December 2009

Abstract We introduced an algorithm for unconstrained optimization based on the transformation of the Newton method with the line search into a gradient descent method. Main idea used in the algorithm construction is approximation of the Hessian by an appropriate diagonal matrix. The steplength calculation algorithm is based on the Taylor's development in two successive iterative points and the backtracking line search procedure. The linear convergence of the algorithm is proved for uniformly convex functions and strictly convex quadratic functions satisfying specified conditions.

Keywords Line search · Gradient descent methods · Newton method · Convergence rate

Mathematics Subject Classifications (2000) 90C30 · 90C06

1 Introduction

We consider the unconstrained optimization 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 in the set of real numbers $\mathbb{R}$ and $f : \mathbb{R}^n \to \mathbb{R}$ is given objective function. It is assumed that the function

P. S. Stanimirović (✉) · M. B. Miladinović
Department of Mathematics, Faculty of Science,
University of Niš, Višegradska 33, 18000 Niš, Serbia
e-mail: pecko@pmf.ni.ac.yu, pecko@pmf.ni.ac.rs

M. B. Miladinović
e-mail: markomiladinovic@gmail.com

$f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable. The most frequently used general iterative scheme aimed to solve the multivariable unconstrained minimization problem (1.1) is of the form

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

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

We use the following notations for convenience:

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

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

$$g_k^T d_k < 0.$$

There are several procedures for the choice of the search direction [16, 30]. The descent direction d_k in the Newton method with the line search is generated by solving the linear system $G_k d = -g_k$. In the gradient descent (GD) method d_k is defined by $d_k = -g_k$. Various conjugate gradient methods update the search direction by the general rule

$$d_k = \begin{cases} -g_k, & k = 0; \\ -g_k + \beta_k d_{k-1}, & k \geq 1, \end{cases}$$

where β_k is a scalar parameter depending of the method. For example, well-known methods are:

$$\beta_k^{FR} = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}, \qquad \text{(Fletcher-Reeves method [12])}$$

$$\beta_k^{PRP} = \frac{g_k^T (g_k - g_{k-1})}{g_{k-1}^T g_{k-1}}, \qquad \text{(Polak-Ribière-Polyak method [20, 21])}.$$

The steplength t_k can be also computed in various ways. There is a class of algorithms known as the line search, whose general strategy is to determine the stepsize t_k such that the objective function value decreases, i.e. $f(x_k + t_k d_k) < f(x_k)$. Two main strategies in the line search algorithms have been proposed: exact line search and inexact line search. The exact line search is based on well-known formula (see, for example [27, 30])

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

Theoretically exact optimal stepsize generally cannot be found in practical computation, and it is also expensive to find almost exact stepsize. Therefore

the most frequently used algorithm in practice is the inexact line search, which try to sufficiently decrease the value of f along the ray $x_k + td_k, t \geq 0$. Some of inexact line search methods are developed in [3, 12, 14, 15, 18, 22, 23, 27, 31].

On the other hand, there are a lot of heuristics for improving the stepsize t_k in conjunction with the negative gradient direction. Barzilai and Borwein suggested an algorithm in [4] (well known as BB 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 $G_k = \gamma_k I$ as an approximation to the Hessian of f at x_k, in order to make that the matrix G_k follows the quasi-Newton property, Barzilai and Borwein chose γ_k according to the rule

$$G_k = \arg \min_{G=\gamma I} \|Gs_{k-1} - y_{k-1}\|_2, \tag{1.2}$$

where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$, yielding the following iterative scheme:

$$x_{k+1} = x_k - \frac{1}{\gamma_k^{BB}} g_k, \quad \gamma_k^{BB} = \frac{s_{k-1}^T y_{k-1}}{s_{k-1}^T s_{k-1}}. \tag{1.3}$$

The result of Barzilai and Borwein was modified in many articles. For example, see [5–10, 13, 24, 25, 28, 29].

The general iterative scheme for the implementation of the line search methods is given as follows (see, for example [30]):

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, $f(x_k)$ 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 step size t_k such that $f(x_k + t_k d_k) < f(x_k)$.
5: Compute $x_{k+1} := x_k + t_k d_k$.
6: (Loop) Set $k := k + 1$ and go to Step 2.

The paper is organized as follows. The motivation of the article is given in the second section. We detect a class of gradient descent algorithms based on the multiplication of the stepsize t_k by an appropriate acceleration parameter, where t_k is computed by the line search procedure. We use the term **accelerated gradient descent algorithms with the line search** for such a class of methods. An explanation of an algorithm for choosing the acceleration parameter in an alternative way with respect to the algorithm used in [1] is given. In the third section we introduced an accelerated gradient descent method arising from the Newton method with the line search. The acceleration parameter is derived reducing the Hessian by an appropriately generated diagonal matrix, and the

steplength t_k is obtained according to the backtracking inexact line search. The convergence analysis of the algorithm for uniformly convex functions as well as strictly convex quadratic functions under certain assumptions is given in Section 4. Numerical results are presented in the last section.

2 Accelerated gradient descent algorithms

The accelerated gradient descent iterative scheme of the form

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

is introduced in [1]. The iterative process (2.1) is based on the usage of the steplength t_k and the acceleration parameter $\theta_k > 0$ which improves the behavior of the gradient descent algorithm.

On the other hand the Newton's method with the line search, defined by the iterative scheme

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

is successful because it uses the Hessian which offers an useful curvature information. However, for various practical problems, the computation of the Hessian matrix and its inverse in each iterative step is very expensive, or even the Hessian is not available analytically. These difficulties initiate a class of methods that only use the function values and the gradients of the objective function and that are closely related to the Newton's method. Quasi-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. The basic motivation behind the quasi-Newton methods is to try to obtain, at least on the average, the rapid convergence associated with Newton's method without explicitly evaluating the Hessian at every step. This can be accomplished by constructing approximations of the inverse Hessian based on the information gathered during the descent process [16, 30]. It is well known that Quasi-Newton methods satisfy a quasi Newton equation given by

$$S_{k+1} y_k = s_k,$$

where

$$s_k = x_{k+1} - x_k, \quad y_k = g_{k+1} - g_k$$

and S_k is symmetric $n \times n$ matrix, selected as an approximation of the Hessian inverse. Let us now observe the general iterative scheme

$$x_{k+1} = x_k - t_k S_k g_k, \tag{2.2}$$

for which the matrix S_k possesses positive definiteness, and it is not required to satisfies the quasi-Newton equation. Such a method is so called a modified Newton method [30].

The main goal in the present paper is to investigate further possibilities of the general scheme (2.1), choosing the acceleration parameter θ_k in another way, using main properties of the modified Newton iterative scheme (2.2). If we take the following diagonal approximation to the inverse of the Hessian

$$S_k = \gamma_k^{-1} I, \quad \gamma_k \in \mathbb{R}, \tag{2.3}$$

the modified-Newton scheme (2.2) reduces to the iterative scheme

$$x_{k+1} = x_k - t_k \gamma_k^{-1} g_k, \tag{2.4}$$

which is essentially a technique for the steplength calculation in the frame of the gradient descent general iterative scheme. In this way, we combine ideas used in [1] and the modified Newton scheme (2.4).

Formal comparison of the iterative schemes (2.4) and (2.1) shows that (2.4) requires the computation of the real number γ_k with respect to the construction of the modification parameter θ_k in (2.1). The steplength t_k is a common scalar which appears in both iterative processes (2.4) and (2.1).

Following essential ideas in these iterative schemes, we observed that (2.1) tends to accelerate the basic gradient descent algorithm, while the iterations (2.4) appears as an aspect of the modified Newton method of the general form (2.2). Since the final result in both (2.4) and (2.1) is the acceleration of the gradient descent algorithm and since the scalar parameter t_k in both iterative methods is computed by means of the backtracking inexact line search procedure, we use the common term **accelerated gradient descent with line search** for all like methods.

In this paper we introduce a technique of taking the length of the acceleration parameter γ_k in the descent method (2.4) for the unconstrained optimization. Algorithm is developed using the approximation of the Hessian by means of the diagonal matrix whose entries are appropriately computed. Therefore, we derive the algorithm based on the general iterative scheme (2.4), where $\gamma_k = \gamma(x_k, x_{k-1}) \in \mathbb{R}$ determines appropriately computed approximation $\gamma_k I$ of the Hessian G_k and where t_k is a real parameter generated after the line search with backtracking.

3 Acceleration based on a modified Newton method

Main idea used in the algorithm construction is approximation of the Hessian inverse in (2.2) by a diagonal matrix using (2.3), where $\gamma_k = \gamma(x_k, x_{k-1})$ is appropriately selected real number based on Taylor's approximation of the function f at the point x_{k+1}, computed by means of (2.4). Therefore, we start from

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

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

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

Having in mind that the distance between x_k and x_{k+1} is small enough (using the local character of searching) we can take $\alpha = 1$ in (3.2) and get the approximation $\xi = x_{k+1}$. Thus we obtain

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

Now, from (3.1) and (3.3) it is not difficult to verify

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

As an straight implication from (3.4), for the choice of the acceleration parameter we obtain

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

It is well-known that if the point x_k is a local minimum of a smooth function, then the gradient g_k of the function vanishes and the Hessian is positive semidefinite (Second-Order Necessary Conditions); and conversely, if the gradient vanishes at some point and the Hessian is positive definite, then the objective function has a local minimum in that point (Second-Order Sufficient Conditions) [30]. Therefore, the condition $\gamma_{k+1} > 0$ is ultimate. In the case $\gamma_{k+1} < 0$ we take $\gamma_{k+1} = 1$. Motivation for such a solution is as follows. When G_k is not a positive definite matrix, then choosing $\gamma_{k+1} = 1$, produces that the next iterative point x_{k+2} is actually computed by the usual gradient descent method: $x_{k+2} = x_{k+1} - t_{k+1} g_{k+1}$.

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

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

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

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

It is clear that in the case $\gamma_{k+1} > 0$ the function Φ is convex, and it is obvious that $\Phi_{k+1}(0) = f(x_{k+1})$ as well as $\Phi'_{k+1}(t) = (t-1)\gamma_{k+1}^{-1}\|g_{k+1}\|^2$. The function Φ decreases in the case $\Phi'_{k+1}(t) < 0$ which is true when $t \in (0, 1)$. Also we have

$$\Phi'_{k+1}(t) = 0 \Leftrightarrow \bar{t}_{k+1} = 1, \tag{3.6}$$

which means that the minimum of Φ_{k+1} is achieved for $t = 1$. Moreover, according to (3.4), the objective function f is decreasing in the case when the condition $\gamma_k > 0$ is ensured in each iteration. We determine the stepsize t_{k+1} using the backtracking line search procedure from [1]. In accordance with (3.6), we conclude that the best choice for the initial value of the line search parameter is $t = 1$. Therefore, we use the following algorithm for the inexact line search.

Algorithm 3.1 The backtracking line search starting from $t = 1$.

Require: Objective function $f(x)$, the direction d_k of the search at the point x_k and numbers $0 < \sigma < 0.5$ and $\beta \in (\sigma, 1)$.

1: $t = 1$.
2: While $f(x_k + td_k) > f(x_k) + \sigma t g_k^T d_k$, take $t := t\beta$.
3: Return $t_k = t$.

In this way, we introduce the following algorithm (called SM method), which is the main result of the article:

Algorithm 3.2 The gradient descent algorithm defined by (2.4) and (3.5).

Require: Objective function $f(x)$ and chosen initial point $x_0 \in dom(f)$.

1: Set $k = 0$, compute $f(x_0)$, $g_0 = \nabla f(x_0)$ and take $\gamma_0 = 1$.
2: If test criteria are fulfilled then stop the iteration; otherwise, go to the next step.
3: (Backtracking) Find the step size $t_k \in (0, 1]$ using Algorithm 3.1 with $d_k = -\gamma_k^{-1} g_k$.
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: Determine the scalar approximation γ_{k+1} of the Hessian of f at the point x_{k+1} using (3.5).
6: If $\gamma_{k+1} < 0$, then take $\gamma_{k+1} = 1$.
7: Set $k := k + 1$, go to the step 2.
8: Return x_{k+1} and $f(x_{k+1})$.

In order to make a comparison in the computational complexity between the AGD algorithm and SM method let us restate that AGD algorithm computes the acceleration parameter θ_k using the following steps 3 and 4 from AGD algorithm, introduced in [1]:

3: Compute: $z = x_k - t_k g_k$, $g_z = \nabla f(z)$ and $y_k = g_z - g_k$.
4: Compute $a_k = t_k g_k^T g_k$, $b_k = -t_k y_k^T g_k$ and $\theta_k = a_k / b_k$.

The main difference between the AGD algorithm and SM method is in the following: AGD algorithm is based on the usage of the acceleration parameter θ_k which is computed applying just restated steps 3 and 4, while SM method assumes a single computation of the real value γ_k by means of (3.5). It is clear that computation of the parameter γ_k in Algorithm 3.2 is simpler with respect to the calculation of the parameter θ_k. Also, a significant difference between the AGD and SM algorithm is in the following: AGD method calculates in Step 2 the steplength t_k using the backtracking line search with $d_k = -g_k$ and later computes the acceleration parameter θ_k in Steps 3 and 4; on the other hand, SM method, firstly, calculates the acceleration parameter γ_k and later applies the backtracking line search with $d_k = -\gamma_k^{-1} g_k$, in order to determine the stepsize t_k.

4 Convergence analysis

In this section we are concerned with the convergence of SM method. Firstly, we consider the set of uniformly convex functions and later analyze the convergence properties of the method for a subset of strictly convex quadratic functions.

In the following proposition and lemma we restate and derive some basic statements needful for analyzing the convergence properties of Algorithm 3.2 for uniformly convex functions. The proof of the next proposition can be found in [19, 26].

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*

$$\|g(x) - g(y)\| \leq L\|x - y\|, \quad \forall\, x, y \in B.$$

Lemma 4.1 *Under the assumptions of Proposition* 4.1 *there exist real numbers m, M satisfying*

$$0 < m \leq 1 \leq M, \tag{4.1}$$

such that $f(x)$ has an unique minimizer x^* and

$$m\|y\|^2 \leq y^T \nabla^2 f(x) y \leq M\|y\|^2, \quad \forall\, x, y \in \mathbb{R}^n; \tag{4.2}$$

$$\frac{1}{2} m\|x - x^*\|^2 \leq f(x) - f(x^*) \leq \frac{1}{2} M\|x - x^*\|^2, \quad \forall\, x \in \mathbb{R}^n; \tag{4.3}$$

$$m\|x - y\|^2 \leq \big(g(x) - g(y)\big)^T (x - y) \leq M\|x - y\|^2, \quad \forall\, x, y \in \mathbb{R}^n. \tag{4.4}$$

Proof Follows directly using known results from [19, 26]. □

The following lemma will tell us how much, at least, the value of an uniformly convex objective function is decreasing in each iteration.

Lemma 4.2 *For twice continuously differentiable and uniformly convex function f on $\mathbb{R}^n$, and for sequence $\{x_k\}$ generated by Algorithm 3.2 the following inequality is valid*

$$f(x_k) - f(x_{k+1}) \geq \mu \|g_k\|^2, \tag{4.5}$$

where

$$\mu = \min\left\{\frac{\sigma}{M}, \frac{\sigma(1-\sigma)}{L}\beta\right\}. \tag{4.6}$$

Proof Observe that our backtracking line search corresponds to the partial case $s_k = 1$ of the backtracking procedure described in [1]. Therefore, according to the backtracking line search Algorithm 3.1 we have the following inequality

$$f(x_k) - f(x_{k+1}) \geq -\sigma t_k g_k^T d_k, \quad \forall\, k \in \mathbb{N}. \tag{4.7}$$

In the rest of the proof we consider two different cases: $t_k < 1$ and $t_k = 1$.

In the case $t_k < 1$, similarly as in [27], we get

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

If we substitute $d_k = -\gamma_k^{-1} g_k$ immediately follows

$$t_k > \frac{\beta(1-\sigma)\gamma_k}{L}. \tag{4.8}$$

Applying (4.8) into the (4.7) we finally get

$$\begin{aligned} f(x_k) - f(x_{k+1}) &\geq -\sigma t_k g_k^T d_k \\ &> \frac{-\sigma(1-\sigma)\beta\gamma_k}{L} \cdot \frac{-g_k^T g_k}{\gamma_k} \\ &> \frac{\sigma(1-\sigma)\beta}{L}\|g_k\|^2. \end{aligned}$$

On the other hand, in the case $t_k = 1$, using the fact that $\gamma_k < M$ (follows from (4.2) and (4.1), having in mind the way of computing γ_k) we have

$$f(x_k) - f(x_{k+1}) \geq -\sigma g_k^T d_k \geq \frac{\sigma}{M}\|g_k\|^2.$$

Finally from the last two inequalities we get

$$f(x_k) - f(x_{k+1}) \geq \min\left\{\frac{\sigma}{M}, \frac{\sigma(1-\sigma)}{L}\beta\right\}\|g_k\|^2 \tag{4.9}$$

and the proof is completed. □

In the next theorem we prove a linear convergence of SM method for uniformly convex functions.

Theorem 4.1 *If the objective function f is twice continuously differentiable as well as uniformly convex on $\mathbb{R}^n$, and the sequence $\{x_k\}$ is generated by Algorithm 3.2 then*

$$\lim_{k\to\infty}\|g_k\| = 0, \tag{4.10}$$

and the sequence $\{x_k\}$ converges to x^* at least linearly.

Proof Since the objective f is bounded below and $f(x_k)$ decreases, it follows that

$$\lim_{k\to\infty}\big(f(x_k) - f(x_{k+1})\big) = 0,$$

which implies together with (4.5) and (4.6)

$$\lim_{k\to\infty}\|g_k\| = 0.$$

If we put $x^* = y$ in the inequality (4.4) and apply mean value Theorem as well as the Cauchy-Schwartz inequality we get (see, for example [19, 26])

$$m\|x - x^*\| \leq \|g(x)\| \leq M\|x - x^*\|, \quad \forall\, x \in \mathbb{R}^n. \tag{4.11}$$

Now, after applying (4.11) and (4.3) we obtain

$$\mu\|g_k\|^2 \geq \mu\, m^2\|x_k - x^*\|^2 \geq 2\mu\frac{m^2}{M}\big(f(x_k) - f(x^*)\big).$$

We conclude that the $\lim_{k\to\infty}\|x_k - x^*\| = 0$ which implies that the sequence $\{x_k\}$ converges to x^*.

To prove that the convergence is linear, it is necessary to show

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

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

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

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

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

because from (4.4) inequality $m \leq L$ holds.

Applying the result of Theorem 4.1 from [27] we obtain

$$\|x_k - x^*\| \leq \sqrt{\frac{2\big(f(x_0) - f(x^*)\big)}{m}}\sqrt{1-\rho^2}^k$$

and the proof is completed. □

Let us now observe strictly convex quadratic objective function f given by

$$f(x) = \frac{1}{2}x^T Ax - b^T x, \tag{4.12}$$

where A is a real $n \times n$ symmetric positive definite matrix and $b \in \mathbb{R}^n$. Denote the eigenvalues of the matrix A as $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

It is known that the convergence analysis of BB method, as well as other gradient methods, is difficult and non-standard so that the convergence results are often provided for convex quadratics (see [4, 11, 17]). Molina and Raydan in [17] established the Q-linear rate of convergence of the (preconditioned) BB method under the additional assumption $\lambda_n < 2\lambda_1$. We consider the convergence of SM method under similar assumptions.

Lemma 4.3 *For the strictly convex quadratic function f given by the expression (4.12) which involves symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$ and the gradient descent method (2.4), where the parameters γ_k and t_k are determined according to (3.5) and Algorithm 3.1, the following holds*

$$\lambda_1 \leq \frac{\gamma_{k+1}}{t_{k+1}} \leq \frac{2\lambda_n}{\sigma}, \quad k \in \mathbb{N}, \tag{4.13}$$

where λ_1 and λ_n are, respectively, the smallest and the largest eigenvalues of A.

Proof Let us start with

$$f(x_{k+1}) - f(x_k) = \frac{1}{2}x_{k+1}^T A x_{k+1} - b^T x_{k+1} - \frac{1}{2}x_k^T A x_k + b^T x_k.$$

After the replacement $x_{k+1} = x_k - t_k \gamma_k^{-1} g_k$ we obtain

$$\begin{aligned} f(x_{k+1}) - f(x_k) &= \frac{1}{2}\left(x_k - t_k\gamma_k^{-1}g_k\right)^T A\left(x_k - t_k\gamma_k^{-1}g_k\right) - b^T\left(x_k - t_k\gamma_k^{-1}g_k\right) \\ &\quad -\frac{1}{2}x_k^T A x_k + b^T x_k \\ &= \frac{1}{2}x_k^T A x_k - \frac{1}{2}t_k\gamma_k^{-1}g_k^T A x_k - \frac{1}{2}t_k\gamma_k^{-1}x_k^T A g_k + \frac{1}{2}t_k^2\gamma_k^{-2}g_k^T A g_k \\ &\quad -b^T x_k + t_k\gamma_k^{-1}b^T g_k - \frac{1}{2}x_k^T A x_k + b^T x_k. \end{aligned}$$

After the usage of the identity $g_k = Ax_k - b$, immediately follows

$$\begin{aligned} f(x_{k+1}) - f(x_k) &= t_k\gamma_k^{-1}\left(b^T g_k - x_k^T A g_k\right) + \frac{1}{2}t_k^2\gamma_k^{-2}g_k^T A g_k \\ &= -t_k\gamma_k^{-1}g_k^T g_k + \frac{1}{2}t_k^2\gamma_k^{-2}g_k^T A g_k. \end{aligned}$$

If we substitute the last equality in (3.5) we get

$$\begin{aligned} \gamma_{k+1} &= 2\gamma_k \frac{-t_k\|g_k\|^2 + \frac{1}{2}t_k^2\gamma_k^{-1}g_k^T A g_k + t_k\|g_k\|^2}{t_k^2\|g_k\|^2} \\ &= \frac{g_k^T A g_k}{g_k^T g_k}. \end{aligned}$$

Hence, the γ_{k+1} is the Rayleigh quotient of the real symmetric matrix A at the vector g_k, which means that the following holds

$$\lambda_1 \le \gamma_{k+1} \le \lambda_n, \quad k \in \mathbb{N}. \tag{4.14}$$

The left inequality in (4.13) immediately follows, since $0 < t_{k+1} \le 1$. To prove the right inequality in (4.13), let us observe the inequality

$$\frac{\gamma_{k+1}}{t_{k+1}} < \frac{L}{\beta(1-\sigma)}, \tag{4.15}$$

which follows straight from (4.8). Taking into account $g(x) = A(x) - b$, and the fact that the real matrix A is symmetric, we get

$$\|g(x)-g(y)\| = \|Ax - Ay\| = \|A(x-y)\| \le \|A\|\|x - y\| = \lambda_n \|x-y\|. \quad (4.16)$$

This means that for Lipschitz constant L in (4.15) we can take the largest eigenvalue λ_n of matrix A. This fact together with choices of parameters $0 < \sigma < 0.5$ and $\beta \in (\sigma, 1)$ from backtracking algorithm, produces the following

$$\frac{\gamma_{k+1}}{t_{k+1}} < \frac{L}{\beta(1-\sigma)} = \frac{\lambda_n}{\beta(1-\sigma)} < \frac{2\lambda_n}{\sigma},$$

and the proof is completed. □

Theorem 4.2 *For the strictly convex quadratic function f given by* (4.12) *and the gradient descent method* (2.4)*, under the additional assumption* $\lambda_n < 2\lambda_1$ *for the eigenvalues of matrix A, we have*

$$\left(d_i^{k+1}\right)^2 \le \delta^2 (d_i^k)^2, \quad (4.17)$$

where

$$\delta = \max\left\{1 - \frac{\sigma\lambda_1}{2\lambda_n}, \frac{\lambda_n}{\lambda_1} - 1\right\},$$

as well as

$$\lim_{k\to\infty} \|g_k\| = 0. \quad (4.18)$$

Proof Let $\{x_k\}$ be the sequence generated by Algorithm 3.2. Assume that orthonormal eigenvectors of A are $\{v_1, v_2, \ldots, v_n\}$. Then for arbitrary vector x_k, using $g_k = Ax_k - b$, we have that there exist real constants $d_1^k, d_2^k, \ldots, d_n^k$ such that

$$g_k = \sum_{i=1}^{n} d_i^k v_i. \quad (4.19)$$

Namely, using (2.4) as well as the fact that $g_k = Ax_k - b$, one can simple verify the following

$$g_{k+1} = A\left(x_k - t_k\gamma_k^{-1} g_k\right) - b = \left(I - t_k\gamma_k^{-1} A\right)g_k. \quad (4.20)$$

Therefore, using the simple linear representation for g_{k+1} of the form (4.19), we obtain

$$g_{k+1} = \sum_{i=1}^{n} d_i^{k+1} v_i = \sum_{i=1}^{n} \left(1 - t_k\gamma_k^{-1}\lambda_i\right) d_i^k v_i. \quad (4.21)$$

To prove (4.17) we have to show that $|1-\lambda_i/\gamma_k t_k^{-1}| \le \delta$. Let us observe to different cases. In the case $\lambda_i \le \gamma_k/t_k$, as a straight implication from (4.13) we get

$$1 > \frac{\lambda_i}{\gamma_k t_k^{-1}} \ge \frac{\sigma\lambda_1}{2\lambda_n} \implies 1 - \frac{\lambda_i}{\gamma_k t_k^{-1}} \le 1 - \frac{\sigma\lambda_1}{2\lambda_n} \le \delta.$$

If the case $\gamma_k/t_k < \lambda_i$ is satisfied, we obtain

$$1 < \frac{\lambda_i}{\gamma_k t_k^{-1}} \le \frac{\lambda_n}{\lambda_1} \implies \left|1 - \frac{\lambda_i}{\gamma_k t_k^{-1}}\right| \le \frac{\lambda_n}{\lambda_1} - 1 \le \delta.$$

Now, in order to prove $\lim_{k\to\infty} \|g_k\| = 0$, we use the orthonormality of the eigenvectors $\{v_1, v_2, \ldots, v_n\}$ as well as (4.19) and get

$$\|g_k\|^2 = \sum_{i=1}^{n} \left(d_i^k\right)^2.$$

Since (4.17) is satisfied and $0 < \delta < 1$ holds, we get $\lim_{k\to\infty} \|g_k\| = 0$, which completes our proof. □

5 Numerical results

In this section we report some numerical results obtained by testing our method, named SM method, with respect to the, previously mentioned, AGD method, as well as the gradient descent (GD) method with the backtracking line search based on Algorithm 3.1. We have selected 30 functions from [2] given in generalized or extended form as a large scale unconstrained test problems. For each test problem we have considered 10 different numerical experiments with the number of variables: 100, 500, 1000, 2000, 3000, 5000, 7000, 8000, 10000 and 15000. The stopping conditions for Algorithm 3.2 as well as two other algorithms are:

$$\|g_k\| \le 10^{-6} \quad \text{and} \quad \frac{|f(x_{k+1}) - f(x_k)|}{1 + |f(x_k)|} \le 10^{-16}.$$

The backtracking parameters for all methods are $\sigma = 0.0001$ and $\beta = 0.8$, which means that we accept a small decrease in f predicted by linear approximation at the current point.

During the execution of algorithms we observed and later reported three different indicators: number of iterations, CPU time, as well as the number of function evaluations, in order to show dominance of SM algorithm with respect to AGD and GD method. The codes are written in the visual C++ programming language on a Workstation Intel Celeron 2.0 GHz.

Table 1 Summary numerical results for *SM*, *AGD* and *GD* tested on 30 large scale test functions

Test function	No. of iterations			CPU time			No. of funct. evaluation		
	SM	AGD	GD	SM	AGD	GD	SM	AGD	GD
Extended Rosenbrock	1735	86574	205737	15.1	3161.5	5942.0	14183	2323901	6003899
Extended penalty	547	276	1196	5.1	10.0	58.6	2872	7643	44104
Perturbed quadratic	79429	351987	384579	681.1	17132.7	29539.9	445587	13882174	16610440
Raydan-1	15117	21718	54183	50.2	190.1	895.0	82001	432311	1471691
Raydan-2	90	40	60	0.2	0.2	0.2	190	90	130
Diagonal 1	7557	38454	35025	29.2	881.3	866.5	40502	1319921	1327459
Diagonal 3	8664	107944	55385	56.3	4485.0	2679.3	47165	3855094	2083028
Hager	729	2410	3055	3.2	31.4	39.4	3474	36465	54374
Generalized tridiagonal - 1	278	570	662	1.6	10.9	15.2	952	7835	10948
Extended tridiagonal - 1	4320	3322	1250746	40.4	18.7	6910.9	38446	13318	6372398
Extended three expon	141	398	1874	0.6	2.4	12.0	700	3324	19117
Diagonal 4	80	100	8892	0.4	0.8	117.6	530	1110	173914
Diagonal 5	60	30	40	0.3	0.2	0.2	130	70	90
Extended Himmelblau	164	335	1355	1.0	7.9	32.9	558	5835	24705
Quadr. Diag. perturbed	31157	2005445	891082	238.5	63203.8	31549.2	316264	72179199	35601182
Quadratic QF1	43601	261485	211981	287.3	47169.1	33466.7	245060	9213885	7997891
Extended quad. penalty QP1	235	191	567	5.9	4.4	18.2	2398	2305	10748
Extended quad. penalty QP2	2232	247742	134484	19.7	12493.2	3281.2	16179	6172728	3878140
Quadratic QF2	62988	99547	309459	796.9	7164.1	31029.3	347920	3976538	13972221
Extended EP1	63	40	501	0.8	1.0	15.1	584	816	13776
Extended tridiagonal - 2	438	1433	1153	2.3	5.2	5.7	2429	7058	9504
Arwhead	256	684	43224	7.9	16.0	5646.4	4325	17997	1920203
Almost perturbed quadratic	43229	253483	200957	291.4	49128.1	36337.1	244132	9689916	8201237
Engval1	319	701	573	6.0	13.1	15.6	2601	6300	8724
Quartc	244	144	478799	0.6	0.5	1306.1	538	338	957648
Diagonal 6	90	40	60	0.2	0.1	0.1	216	90	130
Generalized quartic	157	158	1377	1.0	1.7	42.1	423	715	17683
Diagonal 7	90	60	544	0.3	0.4	3.6	223	284	3313
Diagonal 8	98	50	584	0.4	0.4	4.7	303	256	3840
Diagonal 9	12556	312344	211607	28.0	4814.4	1716.6	77830	12519797	9152384

The results in Table 1 represent the total number of iterative steps, total CPU time (given in seconds) and the total number of function evaluations, respectively, derived from 10 numerical experiments for each test function.

As one can see from Table 1 it is obvious that SM method outperforms other two methods. For 20 test functions SM method shows the best results in the sense of number of iterations needful to achieve requested accuracy, while

Table 2 Numerical results for 5 additional test functions

Test function	No. of iterations			CPU time			No. of funct. evaluation		
	SM	AGD	GD	SM	AGD	GD	SM	AGD	GD
Extended White & Holst	2962	621538	574171	3.1	1953.6	1918.3	26321	18638690	18808384
Tridia	56309	447900	346994	24.0	1629.1	1381.0	264738	17073495	14102900
LIARWHD	3841	64754	166334	7.1	423.6	1485.6	28819	2139319	6193310
DIXON3DQ	1123659	6543592	6493407	656.2	4032.8	5019.3	6544136	39261562	53196428
BIGGSB1	1015243	6001285	5959373	582.8	3659.1	4833.6	5917974	36007720	48821414

AGD is the best in the remaining 10 test problems. On the other hand, our method shows better performance for 23 test functions observing CPU time, as well as the number of function evaluations. Additionally, we selected 5 more test problems from [2] and considered 10 numerical experiments with number of variables $100, 200, \ldots, 1000$. In this case the dimensions of the problems are essentially smaller with respect to the previous experiment, applied on 30 functions. The reason is much bigger number of iterations, which requires a lot of the time for the program execution of the AGD and GD methods. The results are shown in Table 2.

It is clear that SM algorithm is superior in number of iterations, CPU time, as well as in the number of function evaluations for all 5 test problems. Moreover, these results demonstrate a greater superiority of SM algorithm with respect to the superiority derived upon Table 1.

It is important to point out that the difference between observed parameters in the case when AGD outperformed SM method is incomparable smaller with respect to the opposite case. As an approvement for this statement we have Table 3, where the average number of iterations, CPU time, and number of function evaluations for all 350 numerical experiments are given.

According to the presented results we conclude that our SM method has in average about 7 times better characteristics in number of iterations, and approximately 16 times smaller number of function evaluations with respect to AGD and GD method. On the other hand, referring to CPU time, it is evident that SM is almost 60 times faster than AGD and GD method.

The total number of iterations for SM method applied on all 350 numerical experiments is 2518678 while the number of cases from Algorithm 3.2 in which the gamma is negative is 716. This means that SM method behaved like GD method in only 0.03% of cases.

Table 3 Average numerical outcomes for 35 test functions tested on 10 numerical experiments

Average performances	SM	AGD	GD
Number of iterations	7196.22	49933.64	51514.34
CPU time (sec)	10.99	633.28	589.10
Number of function evaluations	42059.15	710851.71	734478.16

6 Conclusion

We present an idea for choosing the acceleration parameter in a gradient descent method using an alternative approach with respect to the algorithm used in [1]. More precisely, we introduce an accelerated gradient descent method, arising from the Newton method with the line search, reducing the Hessian by an appropriately generated diagonal matrix.

The linear convergence of the algorithm is proved for uniformly convex functions and for a subset of strictly convex quadratics.

From Table 1, Table 2 and Table 3 we conclude that the introduced SM method produces better results with respect to the AGD method from [1] and the GD method. Comparative criteria are the number of iterative steps, spent CPU time, and the number of the function evaluations.

Let us, finally, remark that the problem stated in the paper can be exploited in many different ways, since the problem of finding an arbitrary acceleration parameter anyhow is still actual.

Acknowledgement The authors gratefully acknowledge support from the research project 144011 of the Serbian Ministry of Science.

References

1. Andrei, N.: An acceleration of gradient descent algorithm with backtracking for unconstrained optimization. Numer. Algor. **42**, 63–73 (2006)
2. Andrei, N.: An unconstrained optimization test functions collection. http://www.ici.ro/camo/journal/vol10/v10a10.pdf
3. Armijo, L.: Minimization of functions having Lipschitz first partial derivatives. Pac. J. Math **6**, 1–3 (1966)
4. Barzilai, J., Borwein, J.M.: Two point step size gradient method. IMA J. Numer. Anal. **8**, 141–148 (1988)
5. Dai, Y.H.: Alternate step gradient method. Report AMSS–2001–041, Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences (2001)
6. Dai, Y.H., Fletcher, R.: On the asymptotic behaviour of some new gradient methods. Numerical Analysis Report, NA/212, Dept. of Math. University of Dundee, Scotland, UK (2003)
7. Dai, Y.H., Yuan, J.Y., Yuan, Y.: Modified two-point step-size gradient methods for unconstrained optimization. Comput. Optim. Appl. **22**, 103–109 (2002)
8. Dai, Y.H., Yuan, Y.: Alternate minimization gradient method. IMA J. Numer. Anal. **23**, 377–393 (2003)
9. Dai, Y.H., Yuan, Y.: Analysis of monotone gradient methods. J. Ind. Manage. Optim. **1**, 181–192 (2005)
10. Dai, Y.H., Zhang, H.: An adaptive two-point step-size gradient method. Research report, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences (2001)
11. Dai, Y.H., Liao, L.Z.: R-linear convergence of the Barzilai and Borwein gradient method. IMA J. Numer. Anal. **22**, 1–10 (2002)
12. Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. Comput. J. **7**, 149–154 (1964)
13. Friedlander, A., Martinez, J.M., Molina, B., Raydan, M.: Gradient method with retards and generalizations. SIAM J. Numer. Anal. **36**, 275–289 (1999)
14. Goldstein, A.A.: On steepest descent. SIAM J. Control **3**, 147–151 (1965)

15. Lemaréchal, C.: A view of line search. In: Auslander, A., Oetti, W., Stoer J. (eds.) Optimization and Optimal Control, pp. 59–78. Springer, Berlin (1981)
16. Luenberg, D.G., and Ye, Y.: Linear and nonlinear programming. Springer Science + Business Media, LLC, New York (2008)
17. Molina, B., Raydan, M.: Preconditioned Barzilai–Borwein method for the numerical solution of partial differential equations. Numer. Algor. **13**, 45–60 (1996)
18. Moré, J.J., Thuente, D.J.: On line search algorithm with guaranteed sufficient decrease. Mathematics and Computer Science Division Preprint MCS-P153-0590, Argone National Laboratory, Argone (1990)
19. Ortega, J.M., Rheinboldt, W.C.: Iterative Solution Of Nonlinear Equation in Several Variables. Academic, London (1970)
20. Polak, E., Ribiére, G.: Note sur la convergence de méthodes de directions conjuguées. Revue Francaise Informat. Reserche Opérationnelle, 3e Année **16**, 35–43 (1969)
21. Polyak, B.T.: The conjugate gradient method in extreme problems. USSR Comp. Math. Math. Phys. **9**, 94–112 (1969)
22. Potra, F.A., Shi, Y.: Efficient line search algorithm for unconstrained optimization. J. Optim. Theory Appl. **85**, 677–704 (1995)
23. Powell, M.J.D.: Some Global Convergence Properties of a Variable-Metric Algorithm For Minimization Without Exact Line Search, vol. 9, pp. 53–72. AIAM-AMS Proc., Philadelphia (1976)
24. Raydan, M.: On the Barzilai and Borwein choice of steplength for the gradient method. IMA J. Numer. Anal. **13**, 321–326 (1993)
25. Raydan, M.: The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. SIAM J. Optim. **7**, 26–33 (1997)
26. Rockafellar, R.T.: Convex Analysis. Princeton University Press, Princeton (1970)
27. Shi, Z.-J.: Convergence of line search methods for unconstrained optimization. Appl. Math. Comput. **157**, 393–405 (2004)
28. Vrahatis, M.N., Androulakis, G.S., Lambrinos, J.N., Magoulas, G.D.: A class of gradient unconstrained minimization algorithms with adaptive step-size. J. Comp. and Appl. Math. **114**, 367–386 (2000)
29. Yuan, Y.: 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)
30. Sun, W., Yuan, Y.-X.: Optimization Theory and Methods: Nonlinear Programming. Springer, New York (2006)
31. Wolfe, P.: Convergence conditions for ascent methods. SIAM Rev. **11**, 226–235 (1968)