Journal of Computational and Applied Mathematics 416 (2022) 114525

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

An extended delayed weighted gradient algorithm for solving strongly convex optimization problems

R. Andreani [a,*], H. Oviedo [b], M. Raydan [c], L.D. Secchin [d]

[a] Department of Applied Mathematics, IMECC-UNICAMP, University of Campinas, Distrito Barão Geraldo, 13083-859 Campinas SP, Brazil
[b] Escola de Matemática Aplicada, Fundação Getulio Vargas (FGV/EMAp), Rio de Janeiro, RJ, Brazil
[c] Center for Mathematics and Applications (NovaMath), FCT NOVA, 2829-516, Caparica, Portugal
[d] Department of Applied Mathematics, Federal University of Espírito Santo, Rodovia BR 101, Km 60, 29932-540, São Mateus, ES, Brazil

ARTICLE INFO

Article history:
Received 1 October 2021
Received in revised form 9 June 2022

MSC:
65K05
90C25
90C06

Keywords:
Gradient methods
Conjugate gradient methods
Strongly convex functions
Large-scale optimization

ABSTRACT

The recently developed delayed weighted gradient method (DWGM) is competitive with the well-known conjugate gradient (CG) method for the minimization of strictly convex quadratic functions. As well as the CG method, DWGM has some key optimality and orthogonality properties that justify its practical performance. The main difference with the CG method is that, instead of minimizing the objective function on the entire explored subspace, DWGM minimizes the 2-norm of the gradient vector on the same subspace. The main purpose of this study is to extend DWGM for solving strongly convex nonquadratic minimization problems while keeping a low computational cost per iteration. We incorporate the scheme into a tolerant line search globalization strategy, and we show that it exhibits q-linear convergence to the unique global solution. We compare the proposed extended DWGM with state-of-the-art methods for large-scale unconstrained minimization problems. We use some well-known strongly convex test problems, but also solve some regularized logistic regression problems that appear in machine learning. Our numerical results illustrate that the proposed scheme is promising and exhibits a fast convergence behavior. Moreover, we show through numerical experiments on CUTEst problems that the proposed extended DWGM can be very effective in accelerating the convergence of a well-established Barzilai–Borwein-type method when the iterates get close to minimizers of non-convex functions.

© 2022 Elsevier B.V. All rights reserved.

1. Introduction

Let us consider the strictly convex quadratic function given by

$$f(x) = \frac{1}{2}x^\top Ax - b^\top x, \tag{1.1}$$

where $b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$ is a symmetric and positive definite (SPD) matrix. Since A is SPD and the gradient $g(x) \equiv \nabla f(x) = Ax - b$, then the global minimizer of (1.1) is the unique solution $A^{-1}b$ of the linear system $Ax = b$.

* Corresponding author.
E-mail addresses: andreani@unicamp.br (R. Andreani), harry.leon@fgv.br (H. Oviedo), m.raydan@fct.unl.pt (M. Raydan), leonardo.secchin@ufes.br (L.D. Secchin).

https://doi.org/10.1016/j.cam.2022.114525
0377-0427/© 2022 Elsevier B.V. All rights reserved.

Recently, Oviedo [1] proposed a low-cost iterative method to minimize large-scale convex quadratic functions, namely the delayed weighted gradient method (DWGM). DWGM is based on a smoothing technique combined with a one-step delayed gradient method, which, starting at a given $x_0 = x_{-1}$, can be described by the following iterative recursive scheme:

$$\alpha_k^{\mathrm{MG}} := \arg\min_{\alpha} \|\nabla f(x_k - \alpha \nabla f(x_k))\|_2 = \frac{\nabla f(x_k)^\top A \nabla f(x_k)}{(A\nabla f(x_k))^\top (A \nabla f(x_k))}, \tag{1.2a}$$

$$z_k := x_k - \alpha_k^{\mathrm{MG}} \nabla f(x_k), \tag{1.2b}$$

$$\beta_k := \arg\min_{\beta} \|\nabla f(\beta z_k + (1-\beta) x_{k-1})\|_2 = \frac{\left(g_{k-1}^\top (g_{k-1} - \nabla f(z_k))\right)}{\|g_{k-1} - \nabla f(z_k)\|_2^2}, \tag{1.2c}$$

$$x_{k+1} := x_{k-1} + \beta_k (z_k - x_{k-1}). \tag{1.2d}$$

Notice that the gradient of f at z_k can be computed as $\nabla f(z_k) = \nabla f(x_k) - \alpha_k^{\mathrm{MG}} A \nabla f(x_k)$, and so the method only needs the matrix–vector product $A\nabla f(x_k)$ per iteration. In practice, DWGM exhibits a convergence behavior that competes favorably with the classical conjugate gradient (CG) method. It was recently established that DWGM has several key orthogonality properties that add understanding to the practical behavior of the method, including its finite termination; see [2]. Indeed, it was shown that if A has only $p < n$ distinct eigenvalues, then the method terminates in p iterations. Moreover, it was also established that the current iterate given by (1.2d) minimizes the 2-norm of $\nabla f(x)$ on the already explored subspace; see [2] for details. This optimality property motivates the use of DWGM for the minimization of nonquadratic functions. In this work, as a preliminary but fundamental step, we focus on the extension of DWGM to minimize strongly convex (nonquadratic) functions keeping its low computational cost per iteration as well as its simple algorithmic structure.

The rest of the paper is organized as follows. In Section 2, we describe and analyze the proposed extension of DWGM for strongly convex functions, which includes a suitable tolerant line search strategy. Section 3 is dedicated to an extensive numerical comparison between the proposed scheme and state-of-the-art modern methods for large-scale unconstrained minimization on some well-known sparse and dense test problems. We also solve some regularized logistic regression problems that appear in machine learning applications. In Section 4, we present our conclusions and provide some perspectives for a future work.

Notation. $\|\cdot\|_2$, $\|\cdot\|_\infty$ and $\|\cdot\|$ stands for the Euclidean norm, sup-norm and a generic norm, respectively.

2. Extension of DWGM for strongly convex functions

Let us consider the following optimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable and σ-strongly convex function, for which the following inequality holds for all points x and y in $\mathbb{R}^n$:

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\sigma}{2}\|y - x\|_2^2.$$

An equivalent condition is

$$(\nabla f(x) - \nabla f(y))^\top (x - y) \geq \sigma \|x - y\|_2^2.$$

Since f is twice continuously differentiable, another equivalent condition is that $\nabla^2 f(x) \succeq \sigma I$, for all x, that is, the least eigenvalue $\lambda_{\min}(\nabla^2 f(x)) \geq \sigma$ for all x. For a review on strongly convex functions we recommend the book by Bertsekas [3]. From now on we use the notation $g_k = \nabla f(x_k)$ and $H_k = \nabla^2 f(x_k)$.

Even though the function f is σ-strongly convex, it can be highly nonlinear. Therefore, if we apply directly the original version of the method given by (1.2), using at each k the matrix H_k instead of A, then DWGM may fail to obtain the solution of (2.1). This is a possibility since the step-size α_k in (1.2a) is not always able to reduce the gradient norm. Thus, to guarantee convergence of the method, it is crucial to select appropriately the step-size at each iteration. In the convex quadratic case, DWGM computes α_k carrying out an exact line search. However, the selection of an exact step-size for solving (2.1) will not be feasible, due to the nonlinearity of ∇f. Therefore, we propose to equip the method with an inexact line search, such that the kth step-size satisfies

$$\|\nabla f(x_k - \alpha_k g_k)\|_2^2 \leq \|g_k\|_2^2 - c_1 \alpha_k g_k^\top H(x_k) g_k, \tag{2.2}$$

where $c_1 \in (0, 1)$ is a constant. We clarify that it is always possible to find a positive real number α_k such that (2.2) holds. In fact, inequality (2.2) is exactly the Armijo rule applied to the square of the gradient norm of f. Notice that this relation forces the reduction in the norm of two consecutive gradients, which will be a convenient property to establish the global convergence of the proposed extension. In practice, to determine a step-size α_k satisfying (2.2), we use the well-known *backtracking* algorithm (see, e.g., [4]), starting at $\alpha_k^0 = \frac{g_k^\top H_k g_k}{\|H(x_k) g_k\|_2^2}$, since this quotient exploits the local information of the objective function.

In the second phase, the original DWGM also performs an exact line search to determine the parameter β_k. Although we can equip the method with a second inexact line search to choose the parameter β_k satisfying a condition similar to (2.2), in practice, this is an expensive task since this mechanism increases the number of gradient evaluations. In order to avoid extra computational effort, we adopt a simple acceptance–rejection strategy based on a non-monotone criterion on the gradient norm. Specifically, we compute the new iterate as follows

$$x_{k+1} = \begin{cases} x_{\beta,k} & \text{if } \|\nabla f(x_{\beta,k})\|_2^2 \leq \|\nabla f(z_k)\|_2^2 + \min\{\epsilon_k, \gamma t \alpha_k (g_k^\top w_k)\} \\ z_k & \text{otherwise,} \end{cases}$$

where $x_{\beta,k} = x_{k-1} + \beta_k(z_k - x_{k-1})$, β_k is given by (1.2c), $z_k = x_k - \alpha_k g_k$ is the auxiliary point obtained in the first phase of the algorithm, $\{\epsilon_k\} \subset \mathbb{R}_+$ is any sequence such that $\sum_{k=0}^{\infty} \epsilon_k < \infty$, and $t > 0$ and $\gamma \in (0, 1)$ are two constants.

Observe that the algorithmic essence of DWGM is retained since we still use the original formulas for the pair of parameters (α_k, β_k), while we promote the reduction in the gradient norm, which is one of the emblematic properties of DWGM. Additionally, under certain assumptions, we will prove in the next subsection that this extension generates a sequence for which the gradient norm converges q-linearly to zero. This is also a feature of the method introduced in [1]. Keeping in mind all these descriptions, we propose in Algorithm 1 the extension of DWGM to solve (2.1).

Algorithm 1 DWGM for σ-strongly convex functions.

Require: $x_0 \in \mathbb{R}^n$, $t > 0$, $x_{-1} = x_0$, $\{\epsilon_k\} \subset \mathbb{R}_+$ such that $\sum_{k=0}^{\infty} \epsilon_k < \infty$; $\gamma, \delta \in (0, 1)$, $g_0 = \nabla f(x_0)$, $g_{-1} = g_0$, $k \leftarrow 0$.

1: **while** $\|g_k\| > 0$ **do**
2: $\quad w_k = H_k g_k$
3: $\quad \alpha_k = (g_k^\top w_k)/(w_k^\top w_k)$
4: $\quad z_k = x_k - t\alpha_k g_k$
5: $\quad r_k = \nabla f(z_k)$
6: $\quad$ **while** $\|r_k\|_2^2 > \|g_k\|_2^2 - \gamma t \alpha_k (g_k^\top w_k)$ **do**
7: $\qquad \alpha_k \leftarrow \delta \alpha_k$
8: $\qquad z_k = x_k - t\alpha_k g_k, \quad r_k = \nabla f(z_k)$
9: $\quad$ **end while**
10: $\quad y_k = r_k - g_{k-1}$
11: $\quad \beta_k = -\left(g_{k-1}^\top y_k\right) / \left(y_k^\top y_k\right)$
12: $\quad x_{k+1} = x_{k-1} + \beta_k(z_k - x_{k-1})$
13: $\quad g_{k+1} = \nabla f(x_{k+1})$
14: $\quad$ **if** $\|g_{k+1}\|_2^2 > \|r_k\|_2^2 + \min\{\epsilon_k, \gamma t \alpha_k (g_k^\top w_k)\}$ **then**
15: $\qquad x_{k+1} = z_k, \quad g_{k+1} = r_k$
16: $\quad$ **end if**
17: $\quad k \leftarrow k + 1$
18: **end while**

2.1. Convergence analysis of Algorithm 1

In the sequel, we prove the global convergence of Algorithm 1. Let us consider the merit function

$$r(x) = \frac{1}{2}\|\nabla f(x)\|_2^2.$$

In addition to assuming that f is twice continuously differentiable, for the forthcoming results we will assume either one (or both) of the following hypotheses:

H1 f is σ-strongly convex,
H2 r is twice continuously differentiable.

A sufficient condition for H2 to be valid is that f has continuous third derivatives. Next, we prove the well definiteness of Algorithm 1 under H1.

Theorem 2.1. *Assume that H1 holds. Then Algorithm 1 is well defined.*

Proof. For an arbitrary k, we have $\alpha_k > 0$ by the strong convexity of f whenever $g_k \neq 0$. Also, since $-g_k \neq 0$ is a descent direction for $r(x)$ at x_k (in fact, $-g_k^\top \nabla r(x_k) = -g_k^\top H_k g_k < 0$), the Armijo condition (row 6 in Algorithm 1) fails only finitely many times until

$$\|r_k\|_2^2 \leq \|g_k\|_2^2 - \gamma t \alpha_k (g_k^\top H_k g_k). \tag{2.3}$$

In particular, $\|r_k\|_2 < \|g_k\|_2$ whenever $g_k \neq 0$. For $k = 0$, we have $\|r_0\|_2 < \|g_0\|_2 = \|g_{-1}\|_2$, and thus β_0 is well defined if the method did not stop at iteration 0 with $g_0 = 0$. Now, suppose that at iteration $k-1$, $k \geq 1$, we have $g_{k-1} \neq 0$. If g_k computed at row 13 is null, then the next iterate x_k is the minimizer and the method stops. Otherwise, $g_k \neq 0$ and we have, by the test in rows 14–16 of Algorithm 1 and (2.3) for k and $k-1$, that

$$\|r_k\|_2^2 < \|g_k\|_2^2 \leq \|r_{k-1}\|_2^2 + \gamma t \alpha_{k-1} (g_{k-1}^\top H_{k-1} g_{k-1}) \leq \|g_{k-1}\|_2^2.$$

Thus, $r_k \neq g_{k-1}$, and then β_k is well defined. The statement follows by induction. □

Let us now prove the global convergence of Algorithm 1. From now on, we assume, without loss of generality, that Algorithm 1 never reaches $\|g_k\|_2 = 0$. Otherwise, the algorithm has been successfully stopped at a finite iteration and there is nothing to prove. First, we provide a necessary technical result.

Lemma 2.1. *Assume that H1 holds. In Algorithm 1, the Hessian of f remains uniformly bounded along the infinite sequence $\{x_k\}$ generated by the method, that is, there is a constant $M > 0$ such that*

$$\|H_k\|_2 \leq M, \quad \forall k \geq 0.$$

Furthermore, if H2 also holds, there exists a constant $L_r > 0$ such that the step-sizes α_k satisfy

$$0 < \min\left\{\frac{\sigma}{M^2}, \frac{\delta(2-\gamma)\sigma}{tL_r}\right\} \leq \alpha_k \leq \frac{M}{\sigma^2}, \quad \forall k \geq 0.$$

Proof. By the construction of Algorithm 1, we have, for all k,

$$\|\nabla f(x_{k+1})\|_2^2 \leq \|r_k\|_2^2 + \epsilon_k \leq \|\nabla f(x_k)\|_2^2 + \epsilon_k$$

since $t\alpha_k(g_k^\top w_k) > 0$. Take $\bar{\epsilon} \geq \sum_{k=0}^{\infty} \epsilon_k$ and consider the set

$$C = \{x \in \mathbb{R}^n \mid \|\nabla f(x)\|_2^2 \leq \|\nabla f(x_0)\|_2^2 + \bar{\epsilon}\}.$$

Immediately, $x_k \in C$ for all k. We affirm that C is compact. In fact, it is closed by the continuity of ∇f. By the σ-strong convexity of f we have

$$\nabla f(u_k)^\top \left(\frac{u_k - x^*}{\|u_k - x^*\|_2}\right) \geq \sigma \|u_k - x^*\|_2 \to \infty$$

for any sequence $\{u_k\}$ such that $\|u_k\|_2 \to \infty$, where x^* is the minimizer of f. In this case, we must have $\|\nabla f(u_k)\|_2 \to \infty$ and thus $\{u_k\}$ cannot be in C. Then, C is bounded. The compactness of C together the continuity of $\nabla^2 f$ guarantee the existence of a constant $M > 0$ such that

$$\|H_k\|_2 \leq M, \quad \forall k.$$

For each k, let $\alpha_k^{\mathrm{MG}} := (g_k^\top w_k)/(w_k^\top w_k)$. The upper bound on α_k follows from

$$\alpha_k \leq \alpha_k^{\mathrm{MG}} = \frac{g_k^\top H_k g_k}{g_k^\top [H_k]^2 g_k} \leq \frac{M\|g_k\|_2^2}{\sigma^2 \|g_k\|_2^2} = \frac{M}{\sigma^2}.$$

As $\{x_k\}$ is contained in the compact set C and f is continuously differentiable, the positive scalar

$$R = \limsup_k \{\|\nabla f(x_k - t\alpha d)\|_2^2 \mid \|d\|_2^2 \leq \|g_0\|_2^2 + \bar{\epsilon},\ 0 \leq \alpha \leq \delta^{-1} M/\sigma^2\}$$

is well defined. So, by H2 we can take a constant $L_r > 0$ that does not depend on k such that

$$\|\nabla r(y) - \nabla r(x_k)\|_2 \leq L_r \|y - x_k\|_2, \quad \forall y \in \bar{C},\ \forall k$$

where

$$\bar{C} = \{y \in \mathbb{R}^n \mid \|\nabla f(y)\|_2^2 \leq \|g_0\|_2^2 + \bar{\epsilon} + R\}$$

is a compact set containing C. Note that in particular $x_k, z_k \in \bar{C}$ for all k. The above condition is a kind of Lipschitz continuity, and implies the inequality

$$r(y) \leq r(x_k) + \nabla r(x_k)^\top (y - x_k) + \frac{L_r}{2}\|y - x_k\|_2^2, \quad \forall y \in \bar{C},\ \forall k. \tag{2.4}$$

Now, let $m_k \geq 0$ be the smallest integer such that the Armijo rule (row 6 of Algorithm 1) is satisfied. So,

$$\|\nabla f(x_k - t\,\delta^{m_k} \alpha_k^{\mathrm{MG}} g_k)\|_2^2 \leq \|g_k\|_2^2 - \gamma t \delta^{m_k} \alpha_k^{\mathrm{MG}} (g_k^\top w_k)$$

with $m_k = 0$ or

$$\|\nabla f(x_k - t\delta^{m_k-1}\alpha_k^{\mathrm{MG}} g_k)\|_2^2 > \|g_k\|_2^2 - \gamma t\delta^{m_k-1}\alpha_k^{\mathrm{MG}}(g_k^\top w_k). \tag{2.5}$$

If $m_k = 0$ then

$$\alpha_k = \alpha_k^{\mathrm{MG}} = \frac{g_k^\top H_k g_k}{g_k^\top [H_k]^2 g_k} \geq \frac{\sigma\|g_k\|_2^2}{\|H_k\|_2^2\|g_k\|_2^2} \geq \frac{\sigma}{M^2}.$$

Now, suppose that $m_k \geq 1$. We can rewrite (2.5) as

$$r(x_k - t\delta^{-1}\alpha_k g_k) > r(x_k) - \frac{\gamma t\delta^{-1}\alpha_k}{2}(g_k^\top w_k) \tag{2.6}$$

where $\alpha_k = \delta^{m_k}\alpha_k^{\mathrm{MG}}$. On the other hand, $x_k \in C$ implies $\|g_k\|_2^2 \leq \|g_0\|_2^2 + \bar{\epsilon}$, and $\delta^{-1}\alpha_k \leq \delta^{-1}M/\sigma^2$. Then it follows from the definition of $\bar{R}$ and (2.4) that

$$r(x_k - t\delta^{-1}\alpha_k g_k) \leq r(x_k) - t\delta^{-1}\alpha_k g_k^\top w_k + \frac{L_r t^2\delta^{-2}\alpha_k^2}{2}\|g_k\|_2^2. \tag{2.7}$$

Combining inequalities (2.6) and (2.7), we arrive at

$$\frac{\delta(2-\gamma)}{tL_r}\frac{g_k^\top w_k}{\|g_k\|_2^2} < \alpha_k. \tag{2.8}$$

Now, since f is σ-strongly convex, $g_k^\top w_k = g_k^\top H_k g_k \geq \sigma\|g_k\|_2^2$. Using this inequality in (2.8), we obtain

$$\alpha_k > \frac{\delta(2-\gamma)}{tL_r}\frac{g_k^\top w_k}{\|g_k\|_2^2} \geq \frac{\delta(2-\gamma)\sigma}{tL_r}.$$

This concludes the proof. □

The theorem below presents the global convergence of Algorithm 1.

Theorem 2.2. *Let $\{x_k\}$ be the sequence generated by Algorithm 1 and let us assume that H1–H2 are fulfilled. Then, the following conditions hold:*

(i) $\lim_{k\to\infty}\|g_k\|_2 = 0$;

(ii) if for some $c \in (0,1)$ we choose $\epsilon_k \leq c\gamma t\alpha_k(g_k^\top w_k)$, $\forall k$, then $\{\|g_k\|_2\}$ converges at least q-linearly to zero;

(iii) let $L_r > 0$ be the constant provided by Lemma 2.1. If we choose $t \in (0, \sigma^2/L_r]$ then no line search is performed, that is, the step-size $\alpha_k = (g_k^\top w_k)/(w_k^\top w_k)$ always satisfies the Armijo test (row 6 of Algorithm 1).

Proof. By the construction of Algorithm 1, we have, for all k,

$$\|r_k\|_2^2 \leq \|g_k\|_2^2 - \gamma t\alpha_k(g_k^\top H_k g_k) \quad \text{and} \quad \|g_{k+1}\|_2^2 \leq \|r_k\|_2^2 + \epsilon_k. \tag{2.9}$$

Combining these two inequalities, we get

$$t\alpha_k(g_k^\top H_k g_k) \leq \frac{\|g_k\|_2^2 - \|g_{k+1}\|_2^2}{\gamma} + \frac{\epsilon_k}{\gamma}.$$

Given an arbitrary positive integer $N \geq 1$, we have

$$s_N := \sum_{k=0}^{N} t\alpha_k(g_k^\top H_k g_k) \leq \frac{1}{\gamma}\sum_{k=0}^{N}(\|g_k\|_2^2 - \|g_{k+1}\|_2^2) + \frac{1}{\gamma}\sum_{k=0}^{N}\epsilon_k \leq \frac{\|g_0\|_2^2}{\gamma} + \frac{1}{\gamma}\sum_{k=0}^{\infty}\epsilon_k.$$

Thus the sequence of partial sums $\{s_N\}$ is bounded. In addition, observe that $\{s_N\}$ is monotonically increasing because it is a sum of positive scalars. Therefore, $\sum_{k=0}^{\infty} t\alpha_k(g_k^\top H_k g_k)$ is convergent, which implies $\lim_{k\to\infty} t\alpha_k(g_k^\top H_k g_k) = 0$. This last result together with Lemma 2.1 leads to

$$\lim_{k\to\infty} g_k^\top H_k g_k = 0. \tag{2.10}$$

Since f is a σ-strongly convex function, we have $g_k^\top H_k g_k \geq \sigma\|g_k\|_2^2 \geq 0$ for all k, which implies, together with (2.10), that

$$\lim_{k\to\infty}\|g_k\|_2 = 0,$$

and item (i) is established.

Let us prove item (ii). Assume that $\epsilon_k \leq c\gamma t\alpha_k(g_k^\top w_k)$ for all k. By the inequalities in (2.9), Lemma 2.1 and the σ-strong convexity of f we have

$$\frac{\|g_{k+1}\|_2^2}{\|g_k\|_2^2} \leq 1 - \gamma t\alpha_k\frac{(g_k^\top H_k g_k)}{\|g_k\|_2^2} + \frac{\epsilon_k}{\|g_k\|_2^2} \leq 1 - (1-c)\gamma t\nu\sigma$$

for all k and $0 < \nu \le \alpha_k$ as in Lemma 2.1. So, the q-linear convergence of $\{\|g_k\|_2\}$ to zero with rate $(1-(1-c)\gamma t\nu\sigma)^{1/2}$ follows from the last inequality by taking limits when $k \to \infty$.

Now, let us prove item (*iii*). Assume that $t \le \sigma^2/L_r$ and define $\alpha_k^{\mathrm{MG}} := (g_k^\top w_k)/(w_k^\top w_k)$. It follows from (2.4) with $y = z_k$ that

$$\begin{aligned} r(x_k - t\alpha_k^{\mathrm{MG}} g_k) &\le r(x_k) - t\alpha_k^{\mathrm{MG}} \nabla r(x_k)^\top g_k + \frac{L_r}{2} t^2 (\alpha_k^{\mathrm{MG}})^2 \|g_k\|_2^2 \\ &\le r(x_k) - t\alpha_k^{\mathrm{MG}} g_k^\top w_k + \frac{1}{2} t (\alpha_k^{\mathrm{MG}})^2 \sigma^2 \|g_k\|_2^2. \end{aligned} \tag{2.11}$$

Since $\lambda_{\min}(H_k) \ge \sigma$ we have $\lambda_{\min}(H_k^\top H_k) \ge \sigma^2$, and then $\sigma^2\|g_k\|_2^2 \le \|H_k g_k\|_2^2 = w_k^\top w_k$. Using this last inequality in (2.11) we get

$$\begin{aligned} r(x_k - t\alpha_k^{\mathrm{MG}} g_k) &\le r(x_k) - t\alpha_k^{\mathrm{MG}} g_k^\top w_k + \frac{1}{2} t (\alpha_k^{\mathrm{MG}})^2 w_k^\top w_k \\ &= r(x_k) - t\alpha_k^{\mathrm{MG}} g_k^\top w_k + \frac{1}{2} t\alpha_k^{\mathrm{MG}} g_k^\top w_k = r(x_k) - \frac{1}{2} t\alpha_k^{\mathrm{MG}} g_k^\top w_k. \end{aligned}$$

Multiplying this expression by 2 leads to

$$\|\nabla f(x_k - t\alpha_k^{\mathrm{MG}} g_k)\|_2^2 \le \|g_k\|_2^2 - t\alpha_k^{\mathrm{MG}} g_k^\top w_k \le \|g_k\|_2^2 - \gamma t\alpha_k^{\mathrm{MG}} g_k^\top w_k.$$

Therefore, the Armijo test is fulfilled with $\alpha_k = \alpha_k^{\mathrm{MG}}$, and then item (*iii*) holds. This completes the proof. □

Remark 2.1. Condition $\epsilon_k \le c\gamma t\alpha_k(g_k^\top w_k)$ in item (*ii*) of Theorem 2.2 can be implemented in practice since ϵ_k is used after the computation of α_k, g_k and w_k in Algorithm 1. We can take, for example, $\epsilon_k = \min\{1/k^2, 0.9\gamma t\alpha_k(g_k^\top w_k)\}$. In this case, an increase in the magnitude of the gradient is allowed, especially at the first iterations for which $\|g_k\|_2$ is expected to be large. We emphasize that the q-linear rate of convergence in item (*ii*) is relative to the outer iterations, i.e., we do not count the possible updates of z_k performed by the Armijo test. Moreover, item (*iii*) says that no such update is performed.

Remark 2.2. If $t \in (0, \sigma^2/L_r]$ was chosen in Algorithm 1, no additional gradient evaluation is performed by the Armijo test. Note that the first evaluation $r_k = \nabla f(z_k)$ must always be computed to calculate β_k. In that sense, Algorithm 1 encapsulates the case where no line search is necessary.

As discussed above, condition (2.4) in essential to prove convergence of Algorithm 1. It says that ∇r is Lipschitz continuous over a compact set C containing all iterates, whose existence is naturally guaranteed by the method. Obviously, this condition is satisfied if we assume Lipschitz continuity of ∇r on the entire space, that is, that there exists a constant $L > 0$ such that

$$\|\nabla r(x) - \nabla r(y)\| \le L\|x - y\|, \quad \forall x, y \in \mathbb{R}^n. \tag{2.12}$$

In this case, item (*iii*) of Theorem 2.2 is valid for all $t \in (0, \sigma^2/L]$ (note that we can always choose $L_r \le L$). However, the above condition is restrictive and should only be expected for particular functions. One of them is the strictly convex quadratic function, which we consider in the sequel.

Suppose that f is given by (1.1) where A is SPD. Then it is easy to see that hypothesis H1 is fulfilled with $\sigma := \lambda_{\min}(A)$, and also that $L := \lambda_{\max}(A^2)$ is the tightest Lipschitz constant to satisfy (2.12). Now,

$$\frac{\sigma^2}{L} = \frac{(\lambda_{\min}(A))^2}{\lambda_{\max}(A^2)} = \left(\frac{\lambda_{\min}(A)}{\lambda_{\max}(A)}\right)^2 \le 1.$$

Thus, σ^2/L measures the square of the condition number of A. Choosing $t \le \sigma^2/L < 1$ is a conservative strategy for quadratics, since the step-size α in the original DWGM [1] (scheme (1.2)) corresponds exactly to that of Algorithm 1 with $t = 1$. The difference between the original DWGM and Algorithm 1 is the line search. While in DWGM for quadratics there is no line search, in Algorithm 1 we force a *sufficient* decrease of $\|\nabla f(z_k)\|_2$ in relation to $\|\nabla f(x_k)\|_2$. On the other hand, scheme (1.2) has excellent properties on quadratics, like finite termination and conjugacy [2]. So, it would be desirable for Algorithm 1 to generate the sequences (1.2) when applied on quadratics. Fortunately, this is true when we choose $t = 1$.

Theorem 2.3. *Let us assume that f is given by (1.1) where A is SPD. Then Algorithm 1 with $t = 1$ coincides with the original DWGM given by (1.2), independently of the choice of $\{\epsilon_k\}$ and γ. In this case, no line search is performed.*

Proof. Here we have $\nabla^2 f(x_k) = H_k = A$ for all k, and we set $t = 1$. Since β_k in (1.2c) is obtained by the minimization of $\|\nabla f(x_{k-1} + \beta(z_k - x_{k-1}))\|_2$, we have $\|g_{k+1}\|_2^2 \le \|r_k\|_2^2$. That is, the test in row 14 of Algorithm 1 never takes place, independently of $\epsilon_k \geq 0$. Therefore, it is sufficient to prove that the Armijo test in row 6 is always fulfilled with $\alpha_k^{\mathrm{MG}} := (g_k^\top w_k)/(w_k^\top w_k) = (g_k^\top A g_k)/(g_k^\top A^2 g_k)$. In fact, we have

$$\begin{aligned}
\|\nabla f(x_k - \alpha_k^{\mathrm{MG}} g_k)\|_2^2 &= \|A(x_k - \alpha_k^{\mathrm{MG}} g_k) - b\|_2^2 \\
&= \|g_k\|_2^2 - 2\alpha_k^{\mathrm{MG}}(g_k^\top A g_k) + (\alpha_k^{\mathrm{MG}})^2 (g_k^\top A^2 g_k) \\
&= \|g_k\|_2^2 - 2\alpha_k^{\mathrm{MG}}(g_k^\top A g_k) + \alpha_k^{\mathrm{MG}}(g_k^\top A g_k) \\
&= \|g_k\|_2^2 - \alpha_k^{\mathrm{MG}}(g_k^\top w_k) \\
&\leq \|g_k\|_2^2 - \gamma \alpha_k^{\mathrm{MG}}(g_k^\top w_k),
\end{aligned}$$

since $\gamma \in (0, 1)$. This concludes the proof. □

For solving (2.1), when f is strongly convex but not quadratic, the situation is more delicate. In practice, we do not know in general what is the tightest L_r so that (2.4) holds, and we do not know the value of σ. If we choose an arbitrary $t > 0$ in Algorithm 1, for example $t = 1$, Theorem 2.2 ensures its global convergence with the gradient sequence vanishing at a q-linear rate. On the other hand, item (*iii*) of Theorem 2.2 suggests that $t = \sigma^2/L_r$ is the best choice. Of course, if L_r (or even L) and σ are available, then we could define such a convenient value of t and establish a q-linear rate of convergence without imposing a line search strategy.

In practice, when L_r and σ are not available, a successful strategy could be to adjust t during the minimization process. In that case, the proof of Theorem 2.2 can be easily adapted to deal with a sequence $\{t_k\}$ bounded below away from zero. Nevertheless, Theorem 2.3 suggests that choosing $t = 1$ does not imply frequent reductions in the step size α_k, at least if the quadratic approximation of f locally around x_k is good enough along the direction $-g_k$. This will be illustrated in the numerical experiments of Section 3.

3. Numerical experiments

To illustrate the performance of our extended DWGM algorithm in solving nonquadratic strictly convex minimization problems, we consider the following algorithms:

- our extended DWGM (Algorithm 1) with $t = 1$;
- the globalized nonmonotone Barzilai–Borwein method, also known as the spectral BB gradient (SG-BB) algorithm (see [5,6]);
- the ABBmin 1 (or simply ABBmin) method that was developed as an acceleration of the SG-BB algorithm (see [7–9]);
- the Dai–Kou Conjugate Gradient method that was developed as a smoothing improvement of the SG-BB algorithm (see [10,11]);
- the CG_DESCENT method developed by Hager and Zhang [12], which has been considered so far as the best extension of CG in minimizing nonquadratic functions [13].

We implement these methods in Julia, except CG_DESCENT, where we use the implementation provided in the package `Optim.jl` [14] (github.com/JuliaNLSolvers/Optim.jl). Gradients are provided manually. For the SG-BB method, we follow the implementation in Fortran 90 provided by the TANGO project (https://www.ime.usp.br/~egbirgin/tango/codes.php) with its default parameters; in particular, we set $m = 100$, $\gamma = 10^{-4}$ and the maximum number of outer iterations equals to 50,000. This code presents a good behavior on CUTEst problems [15]. The ABBmin and Dai–Kou methods require a nonmonotone line search to guarantee convergence, and for that we impose the same strategy used in the SG-BB method. This line search depends on a parameter $m \geq 1$ that indicates the number of last iterations to be considered, and the well-known Armijo-type reduction parameter $\gamma \in (0, 1)$. As in the case of SG-BB, we set $m = 100$ and $\gamma = 10^{-4}$ for both methods. For Algorithm 1, we also set $\gamma = 10^{-4}$, $\delta = 0.9$ and $\epsilon_k = \min\{1/k^2, 0.9\gamma t \alpha_k (g_k^\top w_k)\}$ for all k (see Remark 2.1). Following SG-BB, the maximum number of outer iterations for the previous methods is set to 50,000. For CG_DESCENT, we use the parameters suggested originally by Hager and Zhang [12].

We note that in the original DWGM for strictly convex quadratics, the Hessian matrix A is only required to build the vector $w_k = Ag_k$. It means that the matrix A is not needed explicitly but instead we only need the product of the Hessian times g_k. From basic calculus, the product $H_k g_k$ in Algorithm 1 can be obtained with high numerical accuracy using a finite difference approximation that only requires an additional gradient evaluation:

$$H_k g_k = \nabla^2 f(x_k) g_k \approx (\nabla f(x_k + h g_k) - g_k)/h, \tag{3.1}$$

where $h > 0$ is a small number. In practice, for smooth convex functions, using the exact Hessian H_k or the finite difference expression in (3.1) produce the same iterations. In our tests, we take

$$h = \frac{10^{-5}}{\min\{1, \max\{10^{-3}, 10^5 \|g_k\|_2\}\}} \in [10^{-5}, 10^{-2}].$$

Thus, $h = 10^{-5}$ when $\|g_k\|_2 \geq 10^{-5}$ and $h > 10^{-5}$ otherwise. The idea is to take h larger if $\|g_k\|$ is much small, avoiding numerical instabilities associated with too small steps hg_k. This strategy proved to be more effective than simply taking h constant.

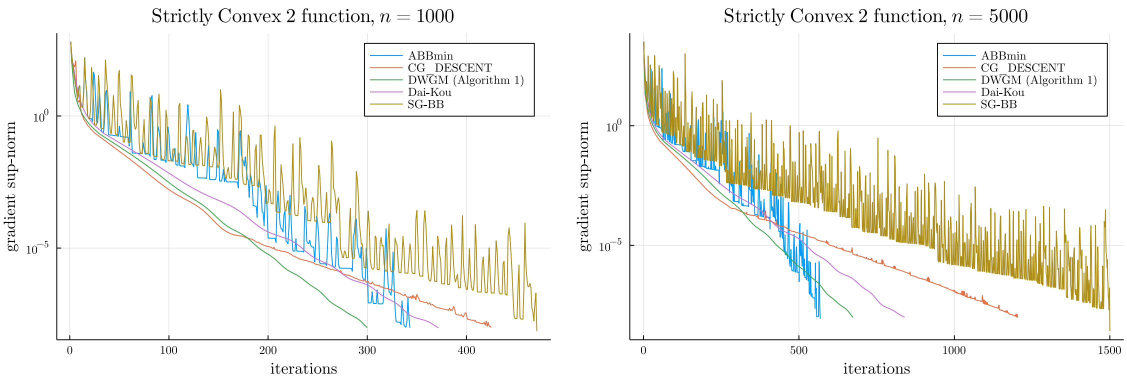

Fig. 1. Convergence history of all methods for SC2 function with $n = 1000$ (left) and $n = 5000$ (right), starting from $x_0 = 2 * \text{ones}(n)$.

In all our experiments we report the results using semilog curves, i.e., the y-axis is in logarithmic scale (base 10). The stopping criterion for all algorithms is

$$\|\nabla f(x_k)\|_\infty \leq 10^{-8}.$$

All the experiments were run in a computer equipped with Intel© Xeon© Silver 4114 CPU 2.20 GHz, 160 Gb RAM, GNU/Linux Ubuntu 20.04.3 LTS and Julia v1.6.2.

Experiment 1. For our first experiment we consider the so-called Strictly Convex 2 (SC2) function (see [6])

$$f(x) = \sum_{i=1}^{n} \frac{i}{10}(e^{x_i} - x_i),$$

and we start all the methods from $x_0 = (2, \dots, 2)^\top$. Clearly, the unique minimizer of SC2 is the origin $x^* = (0, \dots, 0)^\top$, and the Hessian at x^* has n distinct positive eigenvalues. For large values of n, the Hessian matrix is always diagonal but ill-conditioned. In Fig. 1 we show the convergence history of all algorithms for $n = 1000$ (left) and $n = 5000$ (right). All methods converge to the unique global minimizer. We note the smooth and monotone faster convergence behavior of Algorithm 1 compared to the other methods. It can be observed the monotone behavior of the Dai–Kou method, which exhibit a similar behavior on average to the highly nonmonotone SG-BB method. Finally, we note that ABBmin is also nonmonotone and that after reaching a certain accuracy it exhibits a significant acceleration when compare with SG-BB.

Experiment 2. Let us now consider the same experiment as before, but instead of using $x_0 = (2, \dots, 2)^\top$, let us choose the starting point randomly generated uniformly in $[-2, 2]^n$. In that case, the methods will converge to the zero vector, either from negative entries or positive entries. We ran each method starting from five random initial guess. Fig. 2 shows a typical convergence history for $n = 1000$ (left) and $n = 5000$ (right). It is worth noticing that for the chosen starting points with negative entries, some of the Hessian matrices during the convergence process are very ill-conditioned and that clearly reduces the speed of convergence of some of the methods. For $n = 1000$, we observe that Algorithm 1 is the best algorithm. For $n = 5000$, Algorithm 1 is better than CG_DESCENT. Surprisingly, for these problems, the Dai–Kou method presents a consistent decrease of the gradient norm, while ABBmin shows its typical oscillatory behavior and converges prematurely.

Table 1 presents the results for the SC2 function. Columns "iter", "f evals" and "g evals" bring the number of required iterations, the number of function evaluations and the number of gradient evaluations, respectively. Columns "f_{best}" and "$\|g\|_\infty$" brings the best functional value founded so far and the sup-norm of the gradient at the last iterate, respectively. Note that no function evaluation is necessary for the extended DWGM (Algorithm 1); the number 1 in column "f evals" indicates the evaluation of f at the final iterate to return the found functional value "f_{best}".

Experiment 3. For our next experiment, we consider the convex function

$$f(x) = -\log(\lambda^2 - x^\top x),$$

where $\lambda > 0$ is a given scalar and $\log(z)$ denotes the natural logarithm of z. Simple calculations reveal that

$$\nabla f(x) = \frac{2}{(\lambda^2 - x^\top x)} x \quad \text{and} \quad \nabla^2 f(x) = \frac{2}{(\lambda^2 - x^\top x)} I + \frac{4}{(\lambda^2 - x^\top x)^2} x x^\top,$$

where I is the $n \times n$ identity matrix. Hence, when we restrict the domain to $\{x \mid \|x\|_2^2 \leq \lambda^2 - \sigma\}$ for some $\sigma \in (0, \lambda^2)$, we have $\nabla^2 f(x) \succeq \sigma I$, and thus f is σ-strongly convex. Furthermore, the unique global minimizer is $x^* = (0, \dots, 0)^\top$ and the Hessian is dense for all $x \neq (0, \dots, 0)^\top$. In our experiments, we set $\lambda^2 = 10n$. Concerning Algorithm 1, we recall from

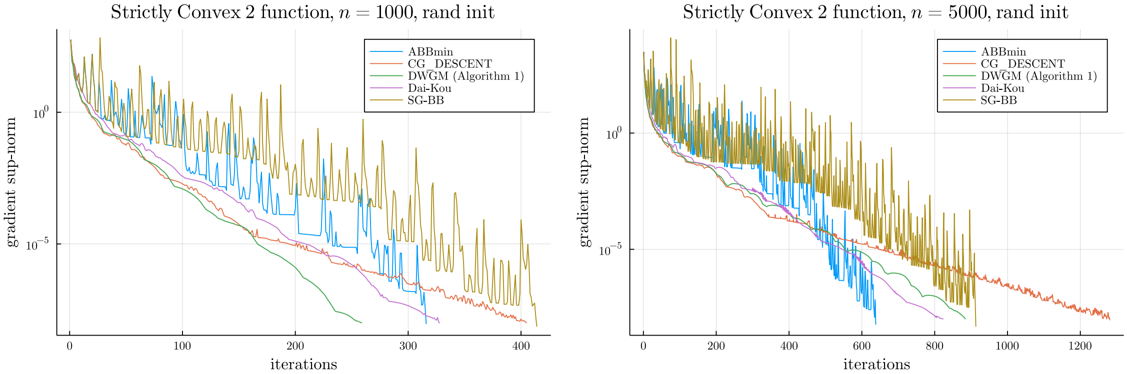

Fig. 2. Typical convergence history of all methods for the SC2 function with $n = 1000$ (left) and $n = 5000$ (right), starting from x_0 randomly chosen uniformly in $[-2, 2]^n$.

the proof of Lemma 2.1 that the gradient norm does not grow much from the initial point ($\|g_k\|_2^2 \le \|g_0\|_2^2 + \bar{\epsilon}$), so we can expect that $\|x_k\|_2^2 < 10n$ for all k.

We vary n between 1000, 2000, 3000, 4000 and 5000. For each of these n, we ran all algorithms starting from $x_0 = (2, \dots, 2)^\top$ and five times starting from a point with uniformly random entries chosen in $[-2, 2]$. In each test, all methods generate a monotonic sequence of gradients norms and converge to the global minimizer x^* in a few iterations (from 3 to 6). This is probably a consequence of the well-conditioned Hessian matrices of f at every iteration. Therefore, the density of the Hessian matrices does not produce a negative effect on the convergence process, whereas the condition number is the key factor to affect the convergence of the methods. We also note that Algorithm 1 converges generally requiring 1 or 2 less iterations than the other methods, and never activates the Armijo line search. We note that for this particular problem, with dense and well-conditioned Hessian matrices, all the considered methods seem to exhibit a q-superlinear rate of convergence.

Experiment 4. Given pairs of vectors $(z^i, y^i) \in \mathbb{R}^n \times \{-1, 1\}$, $i = 1, \dots, m$, let us consider the logistic loss function with ℓ_2-regularization

$$f(x) = \frac{\sigma}{2}\|x\|_2^2 + \sum_{i=1}^{m} \log(1 + e^{-(x^\top z^i)y^i}), \tag{3.2}$$

where $\sigma \ge 0$ is a parameter. By straightforward calculations, we obtain

$$\nabla f(x) = \sigma x - \sum_{i=1}^{m} \frac{y^i h_i(x)}{1 + h_i(x)} z^i$$

and

$$\nabla^2 f(x) = \sigma I + \sum_{i=1}^{m} \frac{(y^i)^2 h_i(x)}{1 + h_i(x)} \left[1 - \frac{1}{1 + h_i(x)}\right] z^i (z^i)^\top,$$

where $h_i(x) = e^{-(x^\top z^i)y^i}$. Immediately, f is σ-strongly convex if $\sigma > 0$. However, we also consider in our tests the case $\sigma = 0$. Actually, note that $h_i(x) > 0$, $i = 1, \dots, m$, remain bounded when minimizing f, and thus the Rayleigh quotients $(u^\top \nabla^2 f(x) u)/(u^\top u)$ remain uniformly above a positive scalar during the minimization process, at least when z forms a basis for $\mathbb{R}^n$ (usually, $m > n$). That is, $\nabla^2 f(x)$ has a great chance to be positive definite with a positive uniform lower bound for all its eigenvalues even if $\sigma = 0$.

The function (3.2) appears in *binary classification problems*. In fact, note that minimizing the sum in (3.2) leads each weighted data $x^\top z^i$ to have the same sign as y^i. In this sense, let us consider (3.2) constructed from the Ionosphere dataset, available from the UCI Machine Learning Repository [16]. This dataset consists of 351 radar returns z^i from the ionosphere together a binary label y^i that indicates weather or not each return is good for analysis (in our case, $y = 1$ for good returns and -1 otherwise). Each entry z^i encodes 34 continuous attributes, all normalized to $[-1, 1]$, so $z^i \in [-1, 1]^{34}$ for all i. It is worth mentioning that we are not training a model/neural network to predict the correct answer to an unknown data, as originally proposed [17]. In particular, we do not divide the dataset into training and test data.

Fig. 3 illustrates the behavior of $\|g_k\|_\infty$ for $\sigma = 0$ and $\sigma = 0.1$ of all methods, starting from $x_0 = (1, \dots, 1)^\top$. In both cases, Algorithm 1 needs fewer iterations to converge. We observe that the SG-BB, ABBmin and Dai–Kou methods suffer to converge if σ increase from 0 to 0.1, while Algorithm 1 and CG_DESCENT do not. For $\sigma = 0.4$, Algorithm 1 still needs significantly fewer iterations than the other methods to converge. In this case, ABBmin overcomes CG_DESCENT, while SG-BB and Dai–Kou take 31,729 and 10,014 iterations, respectively, to converge. See Table 2.

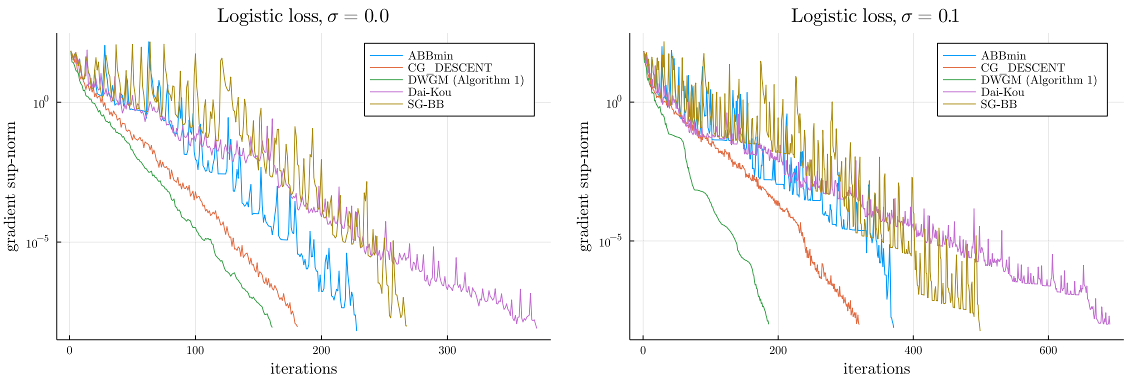

Fig. 3. Convergence history of all methods for the logistic loss (Ionosphere data) function with $\sigma = 0$ (left) and $\sigma = 0.1$ (right), starting from $x_0 = \text{ones}(n)$.

It is worth noticing that the extended DWGM does not present a perfect monotonicity of $\{\|g_k\|_\infty\}$ in Fig. 3. This is because Algorithm 1 is constructed to ensures monotonicity with respect to the Euclidean norm, not the sup-norm; if we plot using $\{\|g_k\|_2\}$, these oscillations vanish.

Finally, we ran the methods starting from initial random points, and also for problems with z_j^i randomly generated uniformly in $[-1, 1]$ and y^i chosen between ± 1. In all tests, Algorithm 1 was superior. For these experiments with random data, SG-BB, ABBmin 1 and Dai–Kou methods do not present the typical oscillatory behavior of $\{\|g_k\|_\infty\}$.

Experiment 5. Here we consider the minimization of

$$\tilde{f}(x) = \frac{1}{2}x^\top Ax + \frac{\beta}{4}\sum_{i=1}^{n} x_i^4$$

over the sphere $S = \{x \in \mathbb{R}^n \mid \|x\|_2 = 1\}$, where A is a Hermitian $n \times n$ matrix and $\beta > 0$ is a regularization parameter. This problem is related to the discretization of the energy function in Bose–Einstein condensates (see [18]). In such a problem, the data values are complex. In our case, however, we assume all data real and A positive definite. To transform this problem into an unconstrained one, we penalize the constraint $x^\top x = 1$ leading to the function

$$f(x) = \frac{1}{2}x^\top Ax + \frac{\beta}{4}\sum_{i=1}^{n} x_i^4 + \frac{\rho}{2}(x^\top x - 1)^2,$$

where $\rho > 0$ is a penalization parameter. We have

$$\nabla^2 f(x) = A + 3\beta\,\text{diag}(x_i^2) + 4\rho\, xx^\top + 2\rho(x^\top x - 1)I.$$

Note that $\nabla^2 f(x) \succeq \sigma(x)I$ where

$$\sigma(x) = \lambda_{\min}(A) + 3\beta \min_i x_i^2 + 2\rho(x^\top x - 1).$$

Thus, the smallest eigenvalue of the Hessian of f is greater or equal than $\lambda_{\min}(A)$ for $\|x\|_2 \geq 1$, but we do not have the guarantee that $\nabla^2 f(x)$ remains positive definite for $\|x\|_2 < 1$. Nevertheless, at the desirable points x where $\|x\|_2 \approx 1$, the Hessian is positive definite.

We take $\rho = 2 \times 10^5$ and, following [18], we set $\beta = 500$. We consider positive definite matrices from the University of Florida Sparse Matrix Collection [19]. We select the matrices from the `HB` group with $n \leq 10,000$. The initial point is taken as $1.1 \times v/\|v\|_2$, where v is the smallest eigenvector computed by the implicitly restarted Lanczos method implemented in the package `Arpack.jl` (https://github.com/JuliaLinearAlgebra/Arpack.jl). This choice is justified by the fact that the problem becomes an eigenvalue problem when $\beta = 0$. So, we aim to start the methods close to the minimizer, although there is no guarantee that this is going to happen due to the presence of the quartic terms x_i^4.

For these experiments, we declare convergence when

$$\|\nabla f(x_k)\|_\infty \leq 10^{-4}$$

due to numerical difficulties in dealing with the terms $\beta x_i^4/4$. Also, we set the maximum number of outer iterations to 100,000 for all methods. Algorithm 1 has computed a negative α_k during the minimization process in 14 out of 54 problems. This indicates that the extended DWGM reaches a non-convexity region, where $\|x\|_2 \ll 1$, and therefore these problems were discarded. Additionally, other 7 problems were rejected because no method was able to solve them. So, we considered 33 problems.

Table 1
Computational results for Experiments 1 and 2.

Function	n	Method	iter	f_{best}	$\|g\|_\infty$	f evals	g evals
SC2	1000	ABBmin	342	5.01E+04	9.59E−09	343	343
		CG_DESCENT	424	5.01E+04	9.90E−09	1446	1023
		DWGM	**299**	**5.01E+04**	**9.76E−09**	**1**	**898**
		Dai–Kou	371	5.01E+04	9.61E−09	372	372
		SG_BB	470	5.01E+04	7.23E−09	471	471
SC2 (rand 1)	1000	ABBmin	305	5.01E+04	8.67E−09	306	306
		CG_DESCENT	423	5.01E+04	8.57E−09	1399	978
		DWGM	**431**	**5.01E+04**	**9.49E−09**	**1**	**1294**
		Dai–Kou	350	5.01E+04	9.56E−09	351	351
		SG_BB	373	5.01E+04	4.85E−09	374	374
SC2 (rand 2)	1000	ABBmin	393	5.01E+04	6.39E−09	394	394
		CG_DESCENT	448	5.01E+04	9.59E−09	1446	999
		DWGM	**361**	**5.01E+04**	**9.60E−09**	**1**	**1084**
		Dai–Kou	374	5.01E+04	9.77E−09	375	375
		SG_BB	418	5.01E+04	1.08E−09	419	419
SC2 (rand 3)	1000	ABBmin	351	5.01E+04	9.46E−09	352	352
		CG_DESCENT	416	5.01E+04	6.91E−09	1430	1015
		DWGM	**292**	**5.01E+04**	**9.87E−09**	**1**	**877**
		Dai–Kou	350	5.01E+04	9.64E−09	351	351
		SG_BB	435	5.01E+04	7.60E−09	436	436
SC2 (rand 4)	1000	ABBmin	380	5.01E+04	4.29E−09	381	381
		CG_DESCENT	461	5.01E+04	9.70E−09	1501	1041
		DWGM	**413**	**5.01E+04**	**9.26E−09**	**1**	**1240**
		Dai–Kou	406	5.01E+04	9.71E−09	407	407
		SG_BB	490	5.01E+04	9.80E−09	491	491
SC2 (rand 5)	1000	ABBmin	315	5.01E+04	9.02E−09	316	316
		CG_DESCENT	404	5.01E+04	9.83E−06	1371	968
		DWGM	**258**	**5.01E+04**	**9.98E−09**	**1**	**775**
		Dai–Kou	327	5.01E+04	9.85E−09	328	328
		SG_BB	413	5.01E+04	7.23E−09	415	414
SC2	5000	ABBmin	568	1.25E+06	8.70E−09	571	569
		CG_DESCENT	1203	1.25E+06	9.74E−09	4592	3390
		DWGM	**673**	**1.25E+06**	**9.83E−09**	**1**	**2020**
		Dai–Kou	839	1.25E+06	9.86E−09	840	840
		SG_BB	1499	1.25E+06	2.62E−09	1529	1500
SC2 (rand 1)	5000	ABBmin	599	1.25E+06	9.31E−09	604	600
		CG_DESCENT	1292	1.25E+06	9.70E−09	4820	3529
		DWGM	**1202**	**1.25E+06**	**9.95E−09**	**1**	**3607**
		Dai–Kou	898	1.25E+06	9.92E−09	899	899
		SG_BB	1143	1.25E+06	5.73E−09	1154	1144
SC2 (rand 2)	5000	ABBmin	558	1.25E+06	8.17E−09	561	559
		CG_DESCENT	1186	1.25E+06	9.69E−09	4549	3364
		DWGM	**651**	**1.25E+06**	**9.75E−09**	**1**	**1954**
		Dai–Kou	791	1.25E+06	9.93E−09	792	792
		SG_BB	916	1.25E+06	4.85E−09	921	917
SC2 (rand 3)	5000	ABBmin	582	1.25E+06	7.48E−09	585	583
		CG_DESCENT	1233	1.25E+06	9.54E−09	4564	3332
		DWGM	**1233**	**1.25E+06**	**9.89E−09**	**1**	**3700**
		Dai–Kou	860	1.25E+06	9.96E−09	861	861
		SG_BB	954	1.25E+06	8.62E−09	977	955
SC2 (rand 4)	5000	ABBmin	723	1.25E+06	9.81E−09	726	724
		CG_DESCENT	1319	1.25E+06	9.21E−09	4871	3553
		DWGM	**1154**	**1.25E+06**	**9.84E−09**	**1**	**3463**
		Dai–Kou	872	1.25E+06	9.70E−09	873	873
		SG_BB	1474	1.25E+06	9.84E−09	1545	1475
SC2 (rand 5)	5000	ABBmin	637	1.25E+06	5.73E−09	639	638
		CG_DESCENT	1280	1.25E+06	8.89E−09	4793	3514
		DWGM	**884**	**1.25E+06**	**9.99E−09**	**1**	**2653**
		Dai–Kou	823	1.25E+06	9.99E−09	824	824
		SG_BB	912	1.25E+06	4.72E−09	926	913

Table 2
Computational results for Experiment 4.

Function	n	Method	iter	f_{best}	$\|g\|_\infty$	f evals	g evals
Logistic loss, $\sigma = 0$	34	ABBmin	227	9.58E+01	6.56E−09	231	228
		CG_DESCENT	180	9.58E+01	9.28E−09	430	268
		DWGM	**160**	**9.58E+01**	**8.85E−09**	**1**	**489**
		Dai–Kou	370	9.58E+01	8.14E−09	372	371
		SG_BB	267	9.58E+01	9.95E−09	272	268
Logistic loss, $\sigma = 0.1$	34	ABBmin	370	1.01E+05	7.44E−09	375	371
		CG_DESCENT	319	1.01E+05	9.73E−09	773	486
		DWGM	**185**	**1.01E+05**	**9.96E−09**	**1**	**564**
		Dai–Kou	690	1.01E+05	9.90E−09	692	691
		SG_BB	498	1.01E+05	5.67E−09	502	499
Logistic loss, $\sigma = 0.4$	34	ABBmin	596	9.58E+01	9.96E−09	610	597
		CG_DESCENT	762	9.58E+01	8.97E−09	2,162	1,541
		DWGM	**367**	**9.58E+01**	**7.62E−09**	**1**	**1,110**
		Dai–Kou	10,014	9.58E+01	9.89E−09	10,087	10,015
		SG_BB	31,729	9.58E+01	9.96E−09	40,450	31,730

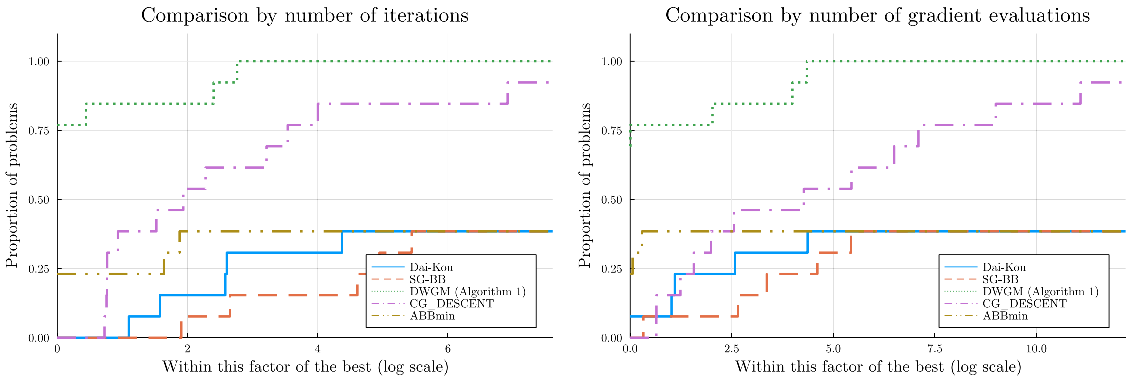

Fig. 4. Performance profile for the data of Table 3, where $\lambda_{\min}(A)$ is large. On the left, we present a comparison based on the number of outer iterations; and on the right, a comparison based on the number of gradient evaluations.

Tables 3 and 4 show the results. Column "st" stands for the resolution status: 0 for convergence and 1 for maximum number of iterations reached. An asterisk indicates that an error has occurred. In column "$\lambda_{\min}(A)$" we report an estimation of the smallest eigenvalue of A computed using the `Arpack.jl` package.

We separate tests by the magnitude of $\lambda_{\min}(A)$. Algorithm 1 behaves very well in problems where the smallest eigenvalue has an order of magnitude one or more (Table 3). In fact, this seems to be a favorable situation for the extended DWGM, since $\nabla^2 f$ tends to be $\succeq \sigma I$ with a large constant σ on a wide set containing the solutions. As a consequence, the step-sizes α_k tends to be larger. In the results reported in Table 3, Algorithm 1 usually converges with much less outer iterations, leading to a smaller overall number of gradient evaluations. We also note that Algorithm 1 converges while other methods do not. As before, we emphasize that in the extended DWGM, f is evaluated only at the final iteration to return the functional value at the solution. Fig. 4 brings the performance profiles [20] related to Table 3, which were generated using the package `BenchmarkProfiles.jl` [21]. For these problems, it can be observed that Algorithm 1 overcomes, on average, all other methods in terms of the number of outer iterations and the number of gradient evaluations.

The scenario changes if $\lambda_{\min}(A)$ is small (Table 4). Although this is not always the case, Algorithm 1 suffers to converge, or it is not able to solve the problem. This is critical in problems where $\lambda_{\min}(A) < 1$. Notice that for all problems of Tables 3 and 4, the Euclidean norm of the final iterate x is approximately 0.99 or even 0.9999, except for `bcsstk03` and `bcsstk09`, where $\|x\|_2 \approx 0.923340$ and ≈ 0.982075, respectively.

3.1. Tests with Algorithm 1 varying t

As we already mentioned in the end of Section 2, no Armijo line search is performed when $t \le \sigma^2/L \le 1$, where L is the tightest Lipschitz constant associated with $\nabla r(x)$. This upper bound can probably be relaxed in view of item (*iii*) of Theorem 2.2, which says that L can be exchanged for a constant L_r that acts only locally around the iterates x_k (Eq. (2.4)). Also, Theorem 2.3 says that for strictly convex quadratic functions, no line search is performed with $t = 1$. In view of our numerical tests, where $t = 1$, we could ask if Algorithm 1 remains working without line search for $t > 1$.

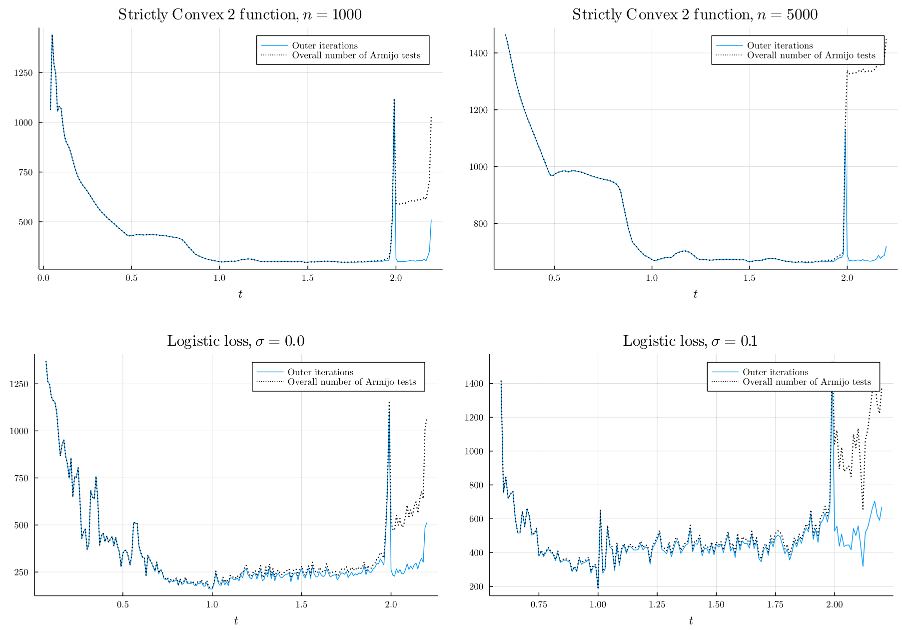

Fig. 5. Relation of outer iterations and the number of Armijo tests in the DWGM for different values of t.

We aim to evaluate the ambiguous effect of increasing t: on the one hand, α_k tends to be reduced more times by the Armijo line search, increasing the number of gradient evaluations; on the other hand, the number of outer iterations tends to be smaller since larger steps are obtained. For this analysis, we selected the following previous experiments:

- SC2 function starting from $x_0 = (2, \dots, 2)^\top$ (experiment 1);
- logistic loss function (3.2) constructed with Ionosphere data (experiment 4).

Let us bound σ^2/L for the SC2 function. In this case, the smallest eigenvalue of $\nabla^2 f$ is $\sigma = 0.1$ and

$$[\nabla r(x)]_i = \left(\frac{i}{10}\right)^2 e^{x_i}(e^{x_i} - 1), \quad i = 1, \dots, n.$$

The derivative of this expression in relation to x_i is $(i/10)^2(2e^{2x_i} - e^{x_i})$, which assumes the value $(i/10)^2$ at $x_i = 0$. So, the Lipschitz constant L for ∇r is at least $(n/10)^2$. This implies that, for the SC2 function,

$$\frac{\sigma^2}{L} \le \frac{1}{n^2} \ll 1. \tag{3.3}$$

For function (3.2), this computation is more complicated and depends on the data y, z.

In Fig. 5 we plot the number of outer iterations performed by Algorithm 1 to declare convergence for different values of t (solid lines). The dotted lines represent the number of times the Armijo condition (row 6 of Algorithm 1) has been checked. For a better visualization, we cut extreme values of t. When solid and dotted lines are together, only one Armijo test is made per iteration, and thus no line search is performed. Otherwise, separate lines indicate reductions in the step-size α during the minimization process.

We observe that Algorithm 1 keeps working without line search until $t \approx 1.75$ for the SC2 function. This value is much higher than the bound (3.3). Also, note that the behavior is the same for $n = 1000$ and $n = 5000$, while (3.3) is different for each case (10^{-3} for $n = 1000$ and 2×10^{-4} for $n = 5000$). For the logistic loss function (3.2), the line search starts to be activated before $t = 1$, but its use remains moderate until $t \approx 2$.

Finally, we note that, as illustrated in Fig. 5, a very small t leads the extended DWGM to require more outer iterations.

3.2. Towards an effective hybrid strategy for general non-convex smooth functions

Our extension of DWGM to strongly convex functions decreases the gradient 2-norm q-linearly to zero. Besides, the previous numerical tests indicate that it is effective to minimize different functions. So, we can expect this method to work effectively near local isolated minimizers x^* of non-convex functions, where $\nabla^2 f(x^*)$ is positive definite, since in this case f is locally strongly convex.

Table 3
Computational results for Experiment 5—matrices with large smallest eigenvalue.

Matrix	n	$\lambda_{\min}(A)$	Method	iter	f_{best}	$\|g\|_\infty$	st	f evals	g evals
bcsstk01	48	3.42E+03	ABBmin	100,000	1.74E+03	8.70E+00	1	16,298,527	100,001
			CG_DESCENT	12,367	1.74E+03	8.90E−05	0	50,679	44,778
			DWGM	**772**	**1.74E+03**	**8.15E−05**	**0**	**1**	**2,317**
			Dai–Kou	*	*	*	*	*	*
			SG_BB	100,000	1.74E+03	4.28E−01	1	15,512,241	100,001
bcsstk03	112	2.94E+04	ABBmin	100,000	1.47E+04	5.21E+05	1	3,964,206	100,001
			CG_DESCENT	61,585	1.42E+04	9.61E−05	0	2,182,236	2,170,022
			DWGM	**5,298**	**1.47E+04**	**9.97E−05**	**0**	**1**	**15,895**
			Dai–Kou	100,000	1.47E+04	3.42E+05	1	3,201,574	100,001
			SG_BB	100,000	1.47E+04	2.20E+01	1	2,796,735	100,001
bcsstk06	420	4.61E+02	ABBmin	100,000	2.32E+05	3.70E−01	1	16,198,528	100,001
			CG_DESCENT	11,846	2.32E+05	9.20E−05	0	38,717	32,706
			DWGM	**6,961**	**2.32E+05**	**1.00E−04**	**0**	**1**	**20,884**
			Dai–Kou	*	*	*	*	*	*
			SG_BB	100,000	2.32E+05	6.72E−02	1	13,897,216	100,001
bcsstk07	420	4.61E+02	ABBmin	100,000	2.32E+05	4.28E−01	1	16,224,302	100,001
			CG_DESCENT	11,420	2.32E+05	9.44E−05	0	37,350	31,527
			DWGM	**6,721**	**2.32E+05**	**9.97E−05**	**0**	**1**	**20,164**
			Dai–Kou	*	*	*	*	*	*
			SG_BB	100,000	2.32E+05	3.50E−02	1	12,089,251	100,001
bcsstk08	1074	2.95E+03	ABBmin	100,000	1.49E+03	6.60E+00	1	4,884,570	100,001
			CG_DESCENT	81,213	1.49E+03	9.70E−05	0	475,215	436,216
			DWGM	**49,097**	**1.49E+03**	**1.00E−04**	**0**	**1**	**147,292**
			Dai–Kou	*	*	*	*	*	*
			SG_BB	100,000	1.49E+03	6.39E+00	1	4,821,927	100,001
bcsstk09	1083	7.10E+03	ABBmin	221	3.53E+03	8.88E−05	0	227	222
			CG_DESCENT	8,572	3.52E+03	7.48E−05	0	461,395	461,277
			DWGM	**71**	**3.55E+03**	**9.93E−05**	**0**	**1**	**214**
			Dai–Kou	212	3.53E+03	9.28E−05	0	215	213
			SG_BB	266	3.53E+03	9.86E−05	0	273	267
bcsstk10	1086	8.54E+01	ABBmin	14,255	4.46E+01	9.78E−05	0	14,600	14,256
			CG_DESCENT	40,967	4.46E+01	9.40E−05	0	642,568	621,225
			DWGM	**19,352**	**4.46E+01**	**9.92E−05**	**0**	**1**	**58,057**
			Dai–Kou	30,545	4.46E+01	9.82E−05	0	103,713	30,546
			SG_BB	89,617	4.46E+01	6.22E−05	0	109,361	89,618
bcsstk26	1922	9.54E+02	ABBmin	100,000	4.95E+05	2.75E+05	1	3,002,140	100,001
			CG_DESCENT	73,319	4.95E+05	9.18E−05	0	309,999	271,875
			DWGM	**38,436**	**4.96E+05**	**1.00E−04**	**0**	**1**	**115,309**
			Dai–Kou	100,000	4.96E+02	2.86E+01	1	3,561,618	100,001
			SG_BB	100,000	4.96E+05	3.32E+01	1	14,364,476	100,001
bcsstm19	817	1.69E+02	ABBmin	613	1.16E+05	8.71E−05	0	622	614
			CG_DESCENT	1,036	1.16E+05	2.99E−05	0	55,483	55,429
			DWGM	**4,160**	**1.16E+05**	**8.39E−05**	**0**	**1**	**12,508**
			Dai–Kou	3,669	1.16E+05	8.95E−05	0	9,989	3,670
			SG_BB	14,965	1.16E+05	9.30E−05	0	18,264	14,966
bcsstm20	485	1.87E+02	ABBmin	181	1.56E+05	3.77E−05	0	184	182
			CG_DESCENT	1,679	1.56E+05	7.24E−05	0	92,930	92,876
			DWGM	**956**	**1.56E+05**	**8.76E−05**	**0**	**1**	**2,897**
			Dai–Kou	3,752	1.56E+05	8.33E−05	0	7,428	3,753
			SG_BB	7,874	1.56E+05	9.42E−05	0	9,238	7,875
lund_a	147	8.00E+01	ABBmin	1,427	4.49E+01	9.28E−05	0	1,472	1,428
			CG_DESCENT	1,485	4.49E+01	9.61E−05	0	5,413	4,628
			DWGM	**388**	**4.49E+01**	**9.94E−05**	**0**	**1**	**1,165**
			Dai–Kou	2,359	4.48E+04	8.69E−05	0	6,110	2,360
			SG_BB	11,976	4.49E+01	4.99E−05	0	14,243	11,977
nos1	237	1.23E+02	ABBmin	100,000	6.40E+01	4.55E−01	1	2,589,828	100,001
			CG_DESCENT	3,132	6.40E+01	9.81E−05	0	12,802	11,387
			DWGM	**646**	**6.40E+01**	**9.48E−05**	**0**	**1**	**1,939**
			Dai–Kou	100,000	6.40E+01	8.06E−01	1	4,931,890	100,001
			SG_BB	100,000	6.40E+01	3.52E−02	1	1,850,983	100,001
nos2	957	3.08E+01	ABBmin	100,000	1.60E+01	6.62E−02	1	2,387,577	100,001
			CG_DESCENT	100,000	1.60E+01	1.26E−01	1	325,260	275,413
			DWGM	**20,275**	**1.60E+01**	**1.00E−04**	**0**	**1**	**60,826**
			Dai–Kou	100,000	1.60E+01	5.40E−01	1	4,012,386	100,001
			SG_BB	100,000	1.60E+04	6.92E−02	1	1,483,670	100,001

Table 4
Computational results for Experiment 5—matrices with small smallest eigenvalue.

Matrix	n	$\lambda_{\min}(A)$	Method	iter	f_{best}	$\|g\|_\infty$	st	f evals	g evals
1138_bus	1138	3.52E−03	ABBmin	1,581	1.12E−01	9.57E−05	0	1,648	1,582
			CG_DESCENT	1,373	1.12E−01	9.48E−05	0	11,825	10,940
			DWGM	**6,737**	**1.12E−01**	**9.98E−05**	**0**	**1**	**20,253**
			Dai–Kou	4,330	1.12E−01	9.39E−05	0	4,407	4,331
			SG_BB	1,248	1.12E−01	8.33E−05	0	1,463	1,249
662_bus	662	5.05E−03	ABBmin	525	1.92E−01	9.21E−05	0	552	526
			CG_DESCENT	479	1.92E−01	8.79E−05	0	4,864	4,601
			DWGM	**100,000**	**1.92E−01**	**1.04E−03**	**1**	**1**	**300,153**
			Dai–Kou	1,313	1.92E−01	9.65E−05	0	1,344	1,314
			SG_BB	521	1.92E−01	1.08E−05	0	544	522
685_bus	685	6.19E−02	ABBmin	2,283	2.26E−01	7.29E−05	0	2,318	2,284
			CG_DESCENT	1,949	2.26E−01	9.31E−05	0	31,821	30,519
			DWGM	**87,335**	**2.26E−01**	**1.00E−04**	**0**	**1**	**262,007**
			Dai–Kou	3,270	2.26E−01	7.18E−05	0	3,340	3,271
			SG_BB	1,676	2.26E−01	8.62E−05	0	1,925	1,677
bcsstk02	66	4.21E+00	ABBmin	255	5.03E+00	4.02E−05	0	261	256
			CG_DESCENT	510	5.03E+00	9.57E−05	0	20,248	20,109
			DWGM	**15,238**	**5.03E+00**	**5.64E−07**	**0**	**1**	**45,715**
			Dai–Kou	679	5.03E+00	7.66E−05	0	685	680
			SG_BB	270	5.03E+00	9.40E−05	0	276	271
bcsstk04	132	4.21E+00	ABBmin	2,370	5.03E+00	3.13E−05	0	2,477	2,371
			CG_DESCENT	2,118	5.03E+00	9.51E−05	0	41,940	40,920
			DWGM	**2,192**	**5.03E+00**	**9.85E−05**	**0**	**1**	**6,577**
			Dai–Kou	3,089	5.03E+00	9.21E−05	0	5,497	3,090
			SG_BB	18,199	5.03E+00	4.21E−05	0	21,987	18,200
bcsstk11	1473	2.96E+00	ABBmin	100,000	2.44E+00	4.02E−02	1	4,519,434	100,001
			CG_DESCENT	100,000	2.44E+00	1.09E+00	1	320,008	254,311
			DWGM	**37,722**	**2.43E+00**	**9.99E−05**	**0**	**1**	**113,167**
			Dai–Kou	*	*	*	*	*	*
			SG_BB	100,000	2.44E+00	1.83E−01	1	122,400	100,001
bcsstk12	1473	2.96E+00	ABBmin	100,000	2.43E+00	1.80E−02	1	10,403,286	100,001
			CG_DESCENT	100,000	2.44E+00	9.95E−01	1	323,747	259,216
			DWGM	**37,774**	**2.43E+00**	**9.99E−05**	**0**	**1**	**113,323**
			Dai–Kou	*	*	*	*	*	*
			SG_BB	100,000	2.46E+00	7.92E−02	1	122,719	100,001
bcsstk21	3600	7.21E+00	ABBmin	9,528	4.21E+00	9.25E−05	0	9,838	9,529
			CG_DESCENT	39,440	4.21E+00	9.82E−05	0	133,136	109,947
			DWGM	**4,836**	**4.21E+00**	**1.00E−04**	**0**	**1**	**14,509**
			Dai–Kou	47,913	4.21E+00	9.99E−05	0	162,530	47,914
			SG_BB	37,169	4.21E+03	1.00E−04	0	45,085	37,170
bcsstk28	4410	8.14E−01	ABBmin	100,000	5.06E−01	1.03E−02	1	1,325,301	100,001
			CG_DESCENT	100,000	5.06E−01	3.32E−01	1	273,847	194,065
			DWGM	**82,076**	**5.05E−01**	**9.99E−05**	**0**	**1**	**246,229**
			Dai–Kou	*	*	*	*	*	*
			SG_BB	100,000	5.17E−01	5.51E−02	1	124,051	100,001
bcsstm07	420	3.30E−01	ABBmin	9,451	1.13E+00	5.76E−05	0	9,585	9,452
			CG_DESCENT	2,810	1.13E+00	8.43E−05	0	81,336	79,835
			DWGM	**97,466**	**1.13E+00**	**9.91E−05**	**0**	**1**	**336,288**
			Dai–Kou	4,726	1.13E+00	8.32E−05	0	5,012	4,727
			SG_BB	1,759	1.13E+00	9.02E−05	0	2,125	1,760
bcsstm08	1074	1.75E−01	ABBmin	6	1.25E+02	2.12E−07	0	8	7
			CG_DESCENT	4	1.25E+05	3.10E−10	0	13	10
			DWGM	**3**	**1.25E+05**	**4.07E−05**	**0**	**1**	**10**
			Dai–Kou	6	1.25E+02	2.12E−07	0	8	7
			SG_BB	6	1.25E+02	2.12E−07	0	8	7
bcsstm11	1473	2.67E−04	ABBmin	44	1.04E+01	3.54E−07	0	48	45
			CG_DESCENT	175	1.04E+01	4.71E−05	0	6,858	6,817
			DWGM	**636**	**1.04E+01**	**9.07E−06**	**0**	**1**	**3,496**
			Dai–Kou	200	1.04E+01	1.45E−05	0	223	201
			SG_BB	52	1.04E+01	4.12E−05	0	76	53
bcsstm22	138	1.03E−05	ABBmin	15	6.25E+01	8.79E−08	0	18	16
			CG_DESCENT	377	6.25E+01	2.90E−05	0	19,490	19,480
			DWGM	**36**	**6.25E+01**	**5.84E−07**	**0**	**1**	**190**
			Dai–Kou	50	6.25E+01	4.04E−05	0	53	51
			SG_BB	21	6.25E+01	2.02E−08	0	28	22

(*continued on next page*)

Table 4 (*continued*).

Matrix	n	$\lambda_{\min}(A)$	Method	iter	f_{best}	$\|g\|_\infty$	st	f evals	g evals
bcsstm23	3134	8.74E−03	ABBmin	10	1.25E+05	4.91E−05	0	12	11
			CG_DESCENT	4	1.25E+05	6.84E−07	0	13	10
			DWGM	**3**	**1.25E+05**	**4.07E−05**	**0**	**1**	**10**
			Dai–Kou	6	1.25E+05	2.12E−07	0	8	7
			SG_BB	10	1.25E+05	4.91E−05	0	12	11
bcsstm26	1922	5.78E−06	ABBmin	6	1.25E+05	2.12E−07	0	8	7
			CG_DESCENT	4	1.25E+05	3.21E−10	0	13	10
			DWGM	**3**	**1.25E+05**	**4.07E−05**	**0**	**1**	**10**
			Dai–Kou	6	1.25E+05	2.12E−07	0	8	7
			SG_BB	6	1.25E+05	2.12E−07	0	8	7
gr_30_30	900	6.15E−02	ABBmin	106	2.22E−01	9.69E−05	0	114	107
			CG_DESCENT	188	2.22E−01	7.73E−05	0	1,199	1,075
			DWGM	**7,103**	**2.22E−01**	**9.90E−05**	**0**	**1**	**40,685**
			Dai–Kou	503	2.22E−01	9.93E−05	0	524	504
			SG_BB	76	2.22E−01	8.44E−05	0	114	77
nos3	960	1.83E−02	ABBmin	2,257	3.35E−01	8.17E−05	0	2,316	2,258
			CG_DESCENT	1,106	3.35E−01	8.27E−05	0	26,665	26,026
			DWGM	**100,000**	**3.35E−01**	**4.59E−04**	**1**	**1**	**318,430**
			Dai–Kou	1,889	3.35E−01	7.26E−05	0	2,019	1,890
			SG_BB	565	3.35E−01	9.22E−05	0	628	566
nos4	100	5.38E−04	ABBmin	61	1.40E+00	4.57E−05	0	66	62
			CG_DESCENT	227	1.40E+00	2.42E−05	0	5,604	5,479
			DWGM	**3,961**	**1.40E+00**	**6.09E−05**	**0**	**1**	**22,270**
			Dai–Kou	560	1.40E+00	7.53E−05	0	585	561
			SG_BB	94	1.40E+00	4.86E−05	0	134	95
nos7	729	4.15E−03	ABBmin	28,724	1.74E−01	3.51E−05	0	29,121	28,725
			CG_DESCENT	11,653	1.74E−01	7.95E−05	0	33,196	23,501
			DWGM	**76,729**	**1.74E−01**	**9.96E−05**	**0**	**1**	**230,188**
			Dai–Kou	12,163	1.74E−01	9.94E−05	0	15,612	12,164
			SG_BB	100,000	1.74E−01	6.99E−02	1	122,995	100,001
plat362	362	3.55E−12	ABBmin	423	3.85E−01	9.39E−05	0	456	424
			CG_DESCENT	484	3.85E−01	8.34E−05	0	4,304	3,935
			DWGM	**21,204**	**3.85E−01**	**9.56E−05**	**0**	**1**	**119,987**
			Dai–Kou	1,627	3.84E−01	8.41E−05	0	1,778	1,628
			SG_BB	241	3.85E−01	4.48E−05	0	474	242

To illustrate our expectation, we conduct some numerical tests with a simple scheme that hybridizes the extended DWGM and ABBmin methods. ABBmin was chosen because it was the best gradient-type method in our previous tests. So, we compare the following two strategies:

- pure ABBmin method with optimality tolerance $\varepsilon = 10^{-8}$, that is, declaring convergence if $\|\nabla f(x_k)\|_\infty \leq 10^{-8}$;
- start running ABBmin, and at the first iterate x_k satisfying $\|\nabla f(x_k)\|_\infty \leq \sqrt{10^{-8}} = 10^{-4}$ (if found), switch to the extended DWGM with optimality tolerance $\varepsilon = 10^{-8}$.

The maximum number of iterations is set to 50,000 in both strategies, which has been the same maximum number of iterations allowed to ABBmin and DWGM in our previous tests. The other parameters were the same used in the previous experiments for each method.

The second strategy, which we will refer as ABBmin+DWGM, is a simple way to decide if we are close to a minimizer. Evidently, there are more effective heuristics to do this, which should be carefully studied in future work. Nevertheless, this simple hybrid scheme leads to encouraging results. Table 5 shows the comparison of the two above strategies on 38 selected unconstrained functions from the CUTEst collection, with number of variables between 50 and 2000. Columns "DWGM activation/iter, $\|g\|_\infty$" show the iteration and the gradient norm at which Algorithm 1 is activated, respectively. Column "ABBmin+DWGM/it total" is the number of ABBmin and extended DWGM iterations combined. Asterisks indicate failure. It is worth noticing that when both strategies converges successfully (columns "st" equal to 0), they reached the same functional value. In Fig. 6 we show the convergence history for some selected problems.

Although the extended DWGM fails on some problems, we can observe that ABBmin+DWGM generally needs fewer iterations than ABBmin, especially for large-scale problems (compare columns "ABBmin/it" and "ABBmin+DWGM/it total"). Also, Algorithm 1 was able to solve two problems not solved by ABBmin, while for `PENALTY3` the situation is the opposite. In other 7 problems, namely `DIAMON2DLS`, `DIAMON3DLS`, `DMN15102LS`, `DMN15103LS`, `DMN15333LS`, `DMN37143LS` and `MNISTS5LS`, a negative α_k was encountered during the extended DWGM execution, and thus they were discarded. This could be due to numerical instabilities in approximating $H_k g_k$ by (3.1), or because 10^{-4} is not small enough to ensure an adequate initialization of the extended DWGM, or even because f is not locally strongly convex around x^*. Strategies to identify and escape from those situations must be considered in future work. Finally, we discard problems where ABBmin never reaches $\|\nabla f(x_k)\|_\infty \leq 10^{-4}$, since in this case we do not switch to the extended DWGM.

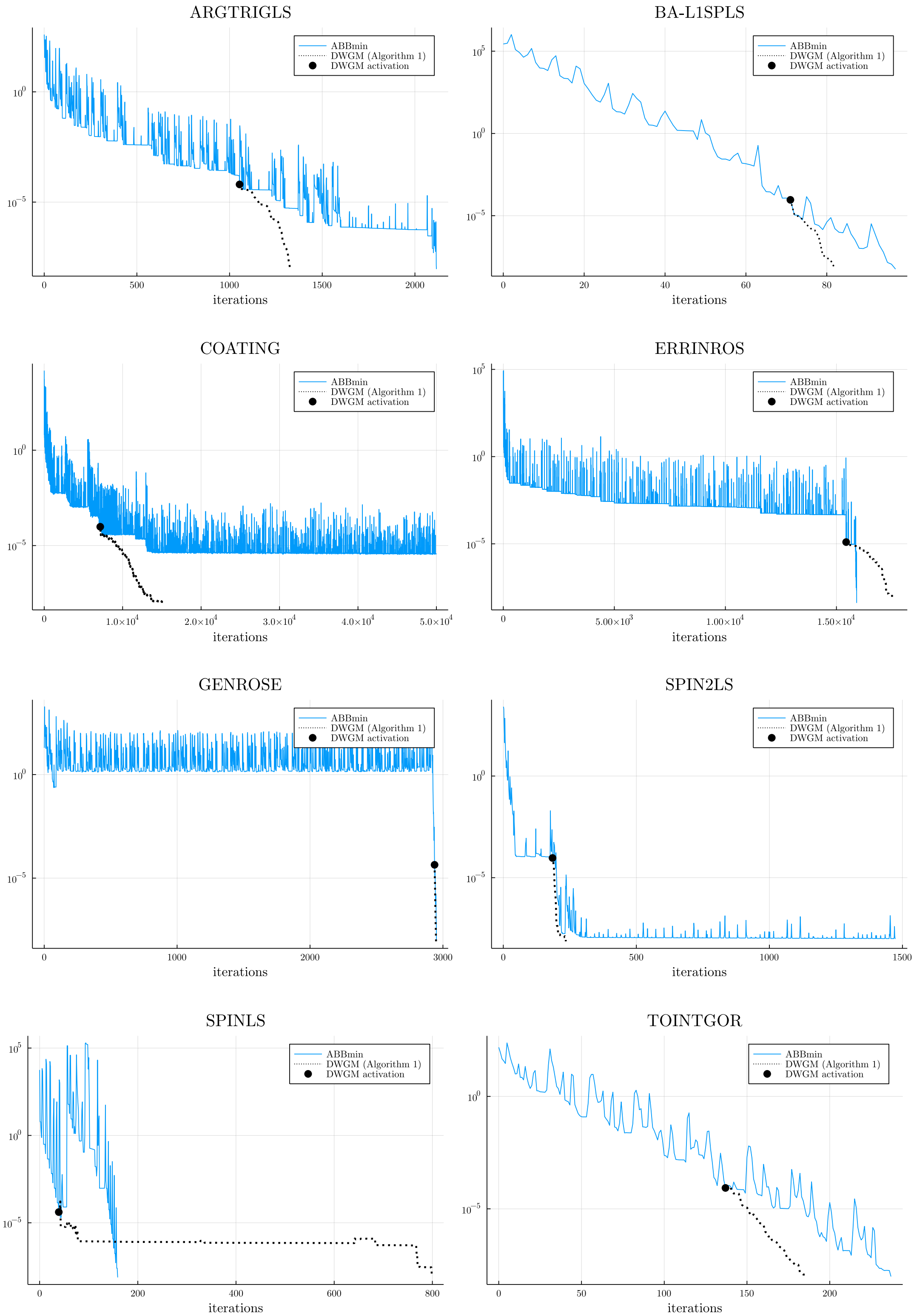

Fig. 6. Convergence history of ABBmin and ABBmin+DWGM on CUTEst problems. The vertical axis stands for the gradient sup-norm in logarithmic scale. The extended DWGM clearly speeds up the convergence for `ARGTRIGLS`, `BA-L1SPLS`, `COATING`, `SPIN2LS` and `TOINTGOR`. In particular, ABBmin was not able to solve `COATING`. Both strategies behave similarly on problem `GENROSE`. For `ERRINROS` and `SPINLS`, ABBmin reaches the solution quickly, turning the extended DWGM ineffective.

Table 5
Computational results on CUTEst problems.

Problem	n	ABBmin			ABBmin+DWGM					
					DWGM activation					
		$\|g\|_\infty$	iter	st	iter	$\|g\|_\infty$	$\|g\|_\infty$	iter	st	it total
ARGLINB	200	5.05E−04	50,000	1	6	4.69E−05	*	*	*	*
ARGLINC	200	1.63E−01	50,000	1	49	6.89E−05	*	*	*	*
ARGTRIGLS	200	9.18E−09	2,116	0	1,054	6.34E−05	9.77E−09	275	0	**1,329**
BA-L1LS	57	3.83E−09	45	0	31	6.08E−05	6.17E−09	6	0	**37**
BA-L1SPLS	57	5.69E−09	97	0	71	9.21E−05	6.18E−09	11	0	**82**
BROWNAL	200	6.56E−02	50,000	1	5	1.50E−06	*	*	*	*
CHNROSNB	50	4.98E−09	1,265	0	1,131	8.79E−05	8.92E−09	34	0	**1,165**
CHNRSNBM	50	8.80E−09	1,630	0	1,552	8.09E−05	7.89E−06	27	0	**1,579**
COATING	134	3.70E−06	50,000	1	7,146	9.69E−05	9.81E−09	7937	0	**15,083**
DMN15332LS	66	1.38E−09	42	0	40	2.33E−05	1.38E−09	1	0	**41**
EDENSCH	2000	9.10E−09	48	0	33	3.42E−05	7.31E−09	9	0	**42**
EG2	1000	7.34E−14	5	0	4	1.46E−06	1.48E−13	1	0	5
ERRINROS	50	4.09E−09	**15,911**	0	15,437	1.26E−05	9.96E−09	2224	0	17,661
FLETCHCR	1000	2.10E−09	325	0	281	3.61E−05	7.52E−09	22	0	**303**
GENROSE	500	9.79E−09	2,951	0	2,937	4.40E−05	8.88E−09	11	0	**2,948**
INTEQNELS	502	9.28E−10	9	0	5	1.19E−05	3.84E−09	3	0	**8**
LUKSAN11LS	100	5.63E−11	4,347	0	4,346	2.99E−05	2.19E−12	1	0	4,347
LUKSAN12LS	98	4.75E−09	443	0	308	5.06E−05	7.40E−09	25	0	**333**
LUKSAN13LS	98	3.35E−09	319	0	253	2.51E−05	3.59E−09	17	0	**270**
LUKSAN14LS	98	3.98E−09	249	0	175	5.98E−05	9.54E−09	25	0	**200**
LUKSAN15LS	100	2.72E−09	36	0	24	5.76E−05	6.69E−09	9	0	**33**
LUKSAN16LS	100	8.56E−09	43	0	29	3.42E−05	4.98E−09	9	0	**38**
LUKSAN17LS	100	9.02E−09	554	0	359	7.56E−05	9.93E−09	97	0	**456**
LUKSAN21LS	100	9.21E−09	1,248	0	1,028	6.74E−05	7.90E−09	113	0	**1,141**
LUKSAN22LS	100	1.09E−08	50,000	1	1,458	3.79E−05	9.83E−09	7528	0	**8,986**
MSQRTALS	1024	9.96E−09	8,355	0	1,076	8.36E−05	9.99E−09	3996	0	**5,072**
MSQRTBLS	1024	9.43E−09	5,287	0	885	9.74E−05	9.70E−09	2315	0	**3,200**
OSCIPATH	500	1.68E−09	15	0	9	1.30E−05	4.76E−09	5	0	**14**
PENALTY1	1000	8.13E−09	1,608	0	42	2.19E−05	9.40E−09	232	0	**274**
PENALTY2	200	8.28E−09	340	0	255	5.09E−05	9.01E−09	38	0	**293**
PENALTY3	200	9.37E−09	233	0	95	8.26E−05	*	*	*	*
QING	100	1.08E−09	107	0	67	7.75E−05	5.26E−09	20	0	**87**
SENSORS	100	1.42E−09	252	0	251	1.50E−06	1.33E−09	1	0	252
SPIN2LS	102	1.00E−08	1,474	0	185	9.30E−05	7.40E−09	50	0	**235**
SPINLS	1327	7.46E−09	**159**	0	39	4.11E−05	8.32E−09	760	0	799
TOINTGOR	50	9.83E−09	237	0	137	8.52E−05	8.96E−09	49	0	**186**
TOINTPSP	50	8.55E−09	314	0	248	4.34E−05	5.04E−09	24	0	**272**
TOINTQOR	50	1.93E−11	79	0	41	6.08E−05	4.22E−09	14	0	**55**

4. Conclusions and future work

In this work, we developed an extension of DWGM for strongly convex functions. Under mild assumptions, we established its global and q-linear convergence. Furthermore, we established that if the Lipschitz constant of the gradient norm of the function and a uniform positive bound of the Hessian eigenvalues are known in advance, the new method preserves the same convergence status without ever activating the line search globalization strategy. Our numerical experiments, on a variety of test problems, show that the new method is competitive and computationally efficient. In particular, it outperforms state-of-the-art methods for solving large-scale regularized logistic regression problems that appear in machine learning applications.

Motivated by the obtained theoretical results and by the observed practical behavior on different strongly convex test problems, we should expect that the proposed new algorithm works effectively near local isolated minimizers of non-convex functions. The preliminary results, with a simple scheme to detect if the iterations are close to a local minimizer, are encouraging; see Table 5 and Fig. 6. Hence, we believe that the use of the extended DWGM as a local search method embedded into a more robust hybrid globalization strategy is a promising topic for future research.

Another research topic is inspired by our numerical tests with $t > 1$ (Section 3.1). They indicate that, for moderate values of t, the line search strategy rarely reduces the step-size during the convergence process of Algorithm 1. This could be beneficial for some problems since larger step-sizes may reduce the number of iterations (this is illustrated in Fig. 5). Thus, it can be a good strategy to adjust $t > 0$ heuristically considering the history of $\|\nabla f(x)\|_2$ and the number of recent line search reductions of the step-size.

Acknowledgments

The first author was financially supported by FAPESP, Brazil (Projects 2013/05475-7 and 2017/18308-2), CEPID-CeMEAI, Brazil (FAPESP 2013/07375-0) and the National Council for Scientific and Technological Development – CNPq, Brazil (Projects 301888/2017-5 and 306988/2021-6). The second author was financially supported by FGV, Brazil (Fundação Getulio Vargas) through the excellence post-doctoral fellowship program. The third author was financially supported by the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) through the projects UIDB/00297/2020 and UIDP/00297/2020 (Center for Mathematics and Applications). The fourth author was financially supported by FAPES (Fundação de Amparo à Pesquisa e Inovação do Espírito Santo) (grant 116/2019) and CNPq (Project 309136/2021-0). We would like to thank two anonymous referees and the Principal Editor for their comments and suggestions that helped us to improve the final version of this paper.

References

[1] H. Oviedo, A delayed weighted gradient method for strictly convex quadratic minimization, Comput. Optim. Appl. 74 (2019) 729–746, http://dx.doi.org/10.1007/s10589-019-00125-6.
[2] R. Andreani, M. Raydan, Properties of the delayed weighted gradient method, Comput. Optim. Appl. 78 (2021) 167–180, http://dx.doi.org/10.1007/s10589-020-00232-9.
[3] D.P. Bertsekas, Convex Analysis and Optimization, Athena Scientific, Belmont, Massachusetts, 2003.
[4] J. Nocedal, S. Wright, Numerical Optimization, Springer Science & Business Media, New York, 2006.
[5] J. Barzilai, J.M. Borwein, Two point step size gradient methods, IMA J. Numer. Anal. 8 (1988) 141–148, http://dx.doi.org/10.1093/imanum/8.1.141.
[6] M. Raydan, The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem, SIAM J. Optim. 7 (1) (1997) 26–33, http://dx.doi.org/10.1137/S1052623494266365.
[7] D. di Serafino, V. Ruggiero, G. Toraldo, L. Zanni, On the steplength selection in gradient methods for unconstrained optimization, Appl. Math. Comput. 318 (2018) 176–195, http://dx.doi.org/10.1016/j.amc.2017.07.037, Recent Trends in Numerical Computations: Theory and Algorithms.
[8] G. Frassoldati, L. Zanni, G. Zanghirati, New adaptive stepsize selections in gradient methods, J. Ind. Manag. Optim. 4 (2008) 299–312, http://dx.doi.org/10.3934/jimo.2008.4.299.
[9] B. Zhou, L. Gao, Y.-H. Dai, Gradient methods with adaptive step-sizes, Comput. Optim. Appl. 35 (1) (2006) 69–86, http://dx.doi.org/10.1007/s10589-006-6446-0.
[10] Y. Dai, C. Kou, A Barzilai-Borwein conjugate gradient method, Sci. China Math. 59 (2016) 1511–1524, http://dx.doi.org/10.1007/s11425-016-0279-2.
[11] H. Liu, Z. Liu, An efficient Barzilai–Borwein conjugate gradient method for unconstrained optimization, J. Optim. Theory Appl. 180 (2019) 879–906, http://dx.doi.org/10.1007/s10957-018-1393-3.
[12] W.W. Hager, H. Zhang, Algorithm 851: CG_DESCENT, a conjugate gradient method with guaranteed descent, ACM Trans. Math. Software 32 (1) (2006) 113–137, http://dx.doi.org/10.1145/1132973.1132979.
[13] W.W. Hager, H. Zhang, A survey of nonlinear conjugate gradient methods, Pac. J. Optim. 2 (1) (2006) 35–58.
[14] P.K. Mogensen, A.N. Riseth, Optim: A mathematical optimization package for Julia, J. Open Source Softw. 3 (24) (2018) 615, http://dx.doi.org/10.21105/joss.00615.
[15] E.G. Birgin, J.M. Martínez, M. Raydan, Algorithm 813: SPG-software for convex-constrained optimization, ACM Trans. Math. Software 27 (3) (2001) 340–349, http://dx.doi.org/10.1145/502800.502803.
[16] V.G. Sigillito, S.P. Wing, L.V. Hutton, K.B. Baker, Ionosphere, UCI Machine Learning Repository, 1988.
[17] V.G. Sigillito, S.P. Wing, L.V. Hutton, K.B. Baker, Classification of radar returns from the ionosphere using neural networks, Johns Hopkins APL Tech. Dig. 10 (3) (1989).
[18] J. Hu, A. Milzarek, Z. Wen, Y. Yuan, Adaptive quadratically regularized Newton method for Riemannian optimization, SIAM J. Matrix Anal. Appl. 39 (3) (2018) 1181–1207, http://dx.doi.org/10.1137/17M1142478.
[19] T.A. Davis, Y. Hu, The university of Florida sparse matrix collection, ACM Trans. Math. Software 38 (1) (2011) http://dx.doi.org/10.1145/2049662.2049663.
[20] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2) (2002) 201–213, http://dx.doi.org/10.1007/s101070100263.
[21] D. Orban, A.N. Riseth, E. Saba, T. Kelman, BenchmarkProfiles.jl: A simple Julia package to plot performance and data profiles, 2019, http://dx.doi.org/10.5281/zenodo.6214088, https://github.com/JuliaSmoothOptimizers/BenchmarkProfiles.jl.