Trends in Computational and Applied Mathematics, **24**, N. 3 (2023), 437-457
Sociedade Brasileira de Matemática Aplicada e Computacional
Online version ISSN 2676-0029
www.scielo.br/tcam
doi: 10.5540/tcam.2023.024.03.00437

On the Preconditioned Delayed Weighted Gradient Method

R. ALEIXO[1*] and H. LARA URDANETA[3]

Received on January 17, 2022 / Accepted on November 16, 2022

ABSTRACT. In this article a preconditioned version of the Delayed Weighted Gradient Method (DWGM) is presented and analyzed. In addition to the convergence, some nice properties as the A- orthogonality of the current transformed gradient with all the previous gradient vectors as well as finite convergence are demonstrated. Numerical experimentation is also offered, exposing the benefits of preconditioning.

Keywords: gradient methods, convex quadratic optimization, Krylov subspace methods, preconditioning.

1 INTRODUCTION

Many real-life applications lead to large-scale convex quadratic optimization problems. Among other, low-cost gradient methods have widely been effective in solving challenging instances of this class of optimization problems (see for instance [8, 16, 26] and references therein). Gradient methods for the unconstrained minimization problem

$$\text{minimize}_{x\in\mathbb{R}^n} \ f(x)$$

generate a sequence of solution approximations x_k satisfying

$$x_{k+1} = x_k - \alpha_k g_k,$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable, $g_k = \nabla f(x_k)$ and $\alpha_k > 0$. The selection of the step length α_k depends on the chosen method. The classical steepest decent (SD) method was proposed in [10] to solve nonlinear systems of equations. In this case,

$$\alpha_k^{\text{SD}} = \text{argmin}_\alpha f(x_k - \alpha g_k). \tag{1.1}$$

*Corresponding author: Rafael Aleixo – E-mail: rafael.aleixo@ufsc.br
[1]Departamento de Matemática, Universidade Federal de Santa Catarina, Campus Blumenau, R. João Pessoa, 2514, 89036-004, Blumenau, SC, Brazil – E-mail: rafael.aleixo@ufsc.br https://orcid.org/0000-0002-7865-6929
[2]Departamento de Engenharia de Controle, Automação e Computação, Universidade Federal de Santa Catarina, Campus Blumenau, R. João Pessoa, 2514, 89036-004, Blumenau, SC, Brazil – E-mail: hugo.lara.urdaneta@ufsc.br https://orcid.org/0000-0003-1670-5075

Assuming the objective function f to be a strictly convex quadratic function, that is, for a symmetric and positive definite (SPD) matrix $A \in \mathbb{R}^{n\times n}$, the unconstrained optimization problem becomes

$$\text{minimize}_{x\in\mathbb{R}^n} \; f(x) = \frac{1}{2}x^T Ax - b^T x. \tag{1.2}$$

It is well known that the unique minimizer of this problem also solves the linear equation $Ax = b$, and so we can denote it by $A^{-1}b$. Under this assumption, simple calculations on (1.1) give

$$\alpha_k^{\text{SD}} = \frac{g_k^T g_k}{g_k^T A g_k}.$$

It is proven the SD method converges Q-linearly [1]. Instead of minimizing the objective function f, the Minimum Gradient (MG) step length [3] aims to minimize the gradient norm, i.e.

$$\alpha_k^{\text{MG}} = \text{argmin}_\alpha ||\nabla f(x_k - \alpha g_k)||_2,$$

which can be written as

$$\alpha_k^{\text{MG}} = \frac{g_k^T A g_k}{g_k^T A^2 g_k}. \tag{1.3}$$

Despite the optimal properties on the definitions of α_k^{SD} and α_k^{MG}, the steepest descent method converges slowly and is badly affected by ill conditioning (see [1] and [18]). An overcome for this issue was proposed by Barzilai and Borwein [5]. The Barzilai-Borwein (BB) methods is based on a secant condition. They propose two possible step sizes

$$\alpha_k^{\text{BB1}} = \text{argmin}_\alpha \|\tilde{s}_{k-1} - \alpha\tilde{y}_{k-1}\|_2 \quad \text{and} \quad \alpha_k^{\text{BB2}} = \text{argmin}_\alpha \left\| \frac{1}{\alpha}\tilde{s}_{k-1} - \alpha\tilde{y}_{k-1} \right\|_2,$$

where $\tilde{s}_{k-1} = x_k - x_{k-1}$ and $\tilde{y}_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$. Thus obtaining, respectively

$$\alpha_k^{\text{BB1}} = \frac{\|\tilde{s}_{k-1}\|_2^2}{\tilde{s}_{k-1}^T \tilde{y}_{k-1}} \;\text{ and }\; \alpha_k^{\text{BB2}} = \frac{\tilde{s}_{k-1}^T \tilde{y}_{k-1}}{\|\tilde{y}_{k-1}\|_2^2}.$$

If we restrict f to be a strictly convex quadratic function, we obtain

$$\alpha_k^{\text{BB1}} = \frac{g_{k-1}^T g_{k-1}}{g_{k-1}^T A g_{k-1}} = \alpha_{k-1}^{\text{SD}} \quad \text{and} \quad \alpha_k^{\text{BB2}} = \frac{g_{k-1}^T A g_{k-1}}{g_{k-1}^T A^2 g_{k-1}} = \alpha_{k-1}^{\text{MG}}.$$

which satisfy [14]

$$\frac{1}{\lambda_1} \leqslant \alpha_k^{\text{BB2}} \leqslant \alpha_k^{\text{BB1}} \leqslant \frac{1}{\lambda_n}$$

where, λ_1 and λ_n are the maximum and the minimum eigenvalues of A, respectively. Basically, the BB step-sizes coincide with SD and MG with a retard of -1. The BB methods converge

R-linearly [12]. Friedlander et al. [19] generalized the BB approach and proposed the Gradient Methods with Retards (GMR). In this case,

$$\alpha_k^{\mathrm{GMR}} = \frac{g_{\nu(k)}^T g_{\nu(k)}}{g_{\nu(k)}^T A g_{\nu(k)}} = \alpha_{\nu(k)}^{\mathrm{SD}},$$

where $\nu(k)$ is chosen in the set $\{k, k-1, \max\{0, k-m\}\}$ and m is a given positive integer.

Yuan [29] proposed the following step length which also includes retard in its definition,

$$a_k^Y = 2\left(\frac{1}{\alpha_{k-1}^{\mathrm{SD}}} + \frac{1}{\alpha_k^{SD}} + \sqrt{\left(\frac{1}{\alpha_{k-1}^{\mathrm{SD}}} - \frac{1}{\alpha_k^{SD}}\right)^2 + \frac{4\|g_k\|_2^2}{(\alpha_{k-1}^{\mathrm{SD}}\|g_{k-1}\|_2)^2}}\right)^{-1}.$$

Yuan's method present a good performance for small-scale problems. Note that all the gradient method variants presented so far are one-step methods.

Methods which use two step-sizes are alternatives to accelerate gradient based methods, by imposing retard on the process (see [11]). Recently, Oviedo-Leon [25] proposed to combine a smoothing technique with a delayed gradient step to construct the promising Delayed Weighted Gradient Method (DWGM), which exhibits an impressive fast convergence behavior that compares favorable with the conjugate gradient method (CG), sharing other nice properties as finite termination and A-orthogonality of its iterated points (gradients) (see [2, 25]). The DWGM can be seen as a variant of the parallel tangent method (PARTAN) in the sense that it uses two line searches in the iteration, with information of previous points to accelerate the gradient method [18, 27, 28]. In [24] a smoothing technique is introduced to prevent the so called zigzagging behavior on the sequence of the gradient norms, which is characteristic of CG methods.

In most practical applications, it is convenient to introduce preconditioning to accelerate the convergence of the process. Preconditioners are useful in iterative methods to solve a linear system. Since the larger the condition number, the larger the rate of convergence, for most iterative solvers, and the preconditioning is meant to decrease the condition number, then it is expected to improve the convergence rate (see for example [7]). Given the positive definite matrix C^2 (preconditioner), the two sided preconditioning designated to deal with the system of equations $Ax = b$, first solves the related system $C^{-1}AC^{-1}y = C^{-1}b$, with better condition number, and then $Cx = y$ to obtain the solution x. There are different choices for preconditioning matrices, as Jacobi, incomplete LU factorization, incomplete Cholesky factorization, successive over-relaxation, etc. Preconditioning has been widely implemented to deal with conjugate gradient type methods, (see [7]). In [2] was noticed finite termination of DWGM in $p < n$ iterations, when the $n \times n$ Hessian matrix has only p distinct eigenvalues, as it also happens with CG methods, motivating the use of preconditioning strategies when solving large-scale symmetric and positive definite linear systems. The aim of this work is to propose, analyze and exhibit the numerical behavior of the preconditioned version of the recently introduced DWGM algorithm.

The remainder of this article is organized as follows: In the next section we describe the DWGM, and expose some of its properties. In section 3 we develop the preconditioned version. The analysis of our preconditioned version is the topic of section 4. Numerical experimentation, and conclusions are the topics of sections 5 and 6 respectively.

2 DELAYED WEIGHTED GRADIENT METHOD

We consider the strictly convex quadratic minimization problem (1.2). Since the gradient $g(x) \equiv \nabla f(x) = Ax - b$, then the unique global solution $A^{-1}b$ for the problem (1.2) also solves the linear system $Ax = b$. For large n, many low cost iterative methods have been proposed and analyzed. The so-called gradient type methods emerge as competitive choices since they show fast linear convergence (see [3, 4, 9, 11, 14, 21]).

From a starting point $x_0 \in \mathbb{R}^n$, consider $g_k = g(x_k)$. The minimum gradient method (MG) (see [3]) is given by the iteration

$$x_{k+1} = x_k - \alpha_k^{\mathrm{MG}} g_k.$$

Here, the step-size α_k^{MG} was defined in (1.3). Denoting $w_k := Ag_k$, we can write the step size as

$$\alpha_k^{\mathrm{MG}} = \frac{g_k^T w_k}{\|w_k\|_2^2}.$$

The minimum gradient norm method calculates the next iterated point as the vector alongside the current gradient at which the norm of the next gradient is minimized.

As a two step gradient method, DWGM incorporates a delaying step defined as follows [25]: The first stage uses the ordinary minimum gradient point $y_k = x_k - \alpha_k^{\mathrm{MG}} g_k$. Then, by defining $\beta_k = \operatorname{argmin}_{\beta \in \mathbb{R}} \|\nabla f(x_{k-1} + \beta(y_k - x_{k-1}))\|_2$, at the second stage

$$x_{k+1} = x_{k-1} + \beta_k (y_k - x_{k-1})$$

is calculated. It is straightforward to see that $\nabla f(x_{k-1} + \beta(y_k - x_{k-1})) = g_{k-1} - \beta(g_{k-1} - r_k)$, for $r_k = g_k - \alpha_k w_k$. This leads to

$$\beta_k = g_{k-1}^T (g_{k-1} - r_k) / \|g_{k-1} - r_k\|_2^2.$$

By merging the definition of y_k into x_{k+1} and after simple manipulation, the next iterated point can be rewritten as

$$x_{k+1} = (1 - \beta_k) x_{k-1} + \beta_k x_k - \beta_k \alpha_k g_k. \tag{2.1}$$

This expression leads us to interpret the choice of β_k as that it decides simultaneously for choosing a point on the line passing through the two previous iterated points, and how much to advance alongside the current gradient direction in such way that the gradient on the next iteration is minimized. The resulting DWGM algorithm is:

Algorithm 1 DWGM

Require: $A \in \mathbb{R}^{n\times n}$ SPD, $x_0 \in \mathbb{R}^n$, $x_{-1} = x_0$, $g_0 = g(x_0)$, $g_{-1} = g_0$, $\varepsilon > 0$.

$k = 0$
while $\|g_k\|_2 > \varepsilon$ **do**
  $w_k = Ag_k$
  $\alpha_k = g_k^T w_k / w_k^T w_k$
  $y_k = x_k - \alpha_k g_k$
  $r_k = g_k - \alpha_k w_k$
  $\beta_k = g_{k-1}^T (g_{k-1} - r_k) / \|g_{k-1} - r_k\|_2^2$
  $x_{k+1} = x_{k-1} + \beta_k (y_k - x_{k-1})$
  $g_{k+1} = g_{k-1} + \beta_k (r_k - g_{k-1})$
  $k = k + 1$
end while

Some of the properties that DWGM enjoys, established in [25] and [2] include the nonnegativity of β_k for all k, the monotonic decreasing of $\{\|g_k\|_2\}$ as well as the Q-linear convergence of $\{g_k\}$ to zero when k goes to infinity (which implies that $\{x_k\}$ converges to the unique global minimizer of f), and finite convergence by using A-orthogonality of the gradient vector at the current iteration with all previous gradient vectors.

3 PRECONDITIONED DWGM

In this section we present the preconditioned DWGM derivation. Preconditioners play an important role in the theory of iterative methods. Quoting [7] "... lack of robustness and sometimes erratic convergence behavior are recognized weakness of iterative solvers. These issues hamper the acceptance of iterative methods despite their intrinsic appeal for very large linear systems. Both the efficiency and robustness of iterative techniques can be very much improved by using preconditioning."

Let M be a positive definite preconditioner, then exists an unique symmetric positive definite matrix C, such that $M = C^2$ (see [20]). This positive definite matrix, as well as it inverse allow us to define the so-called "hat" transformations, that conduce us to transform the space where the original iterated point x_k belongs into the transformed space where the $\hat{x}_k$ are calculated:

$$\hat{A} = C^{-1}AC^{-1}; \quad \hat{b} = C^{-1}b; \quad \hat{x} = Cx. \tag{3.1}$$

Note that $\hat{A}$ is SPD. Let us define the auxiliary convex quadratic minimization problem

$$\text{minimize}_{\hat{x}\in\mathbb{R}^n} \ \hat{f}(\hat{x}) = \frac{1}{2}\hat{x}^T\hat{A}\hat{x} - \hat{b}^T\hat{x}. \tag{3.2}$$

Evaluating the function f at $x = C^{-1}\hat{x}$ we obtain $\hat{f}(\hat{x}) = f(x)$, where $f(x)$ is given in (1.2). Once we have the "hat" quadratic minimization problem (3.2), one could apply the delayed weighted gradient method to solve it. However, the algorithm is better defined if the preconditioning part is incorporated into the DWGM algorithm. The preconditioned DWGM algorithm generates a sequence $\{\hat{x}_k\}$ in the transformed space, which is translated to the original space of x. Let us denote the current iterated point by $\hat{x}_k \in \mathbb{R}^n$, and consider $\hat{g}_k = \nabla\hat{f}(\hat{x}_k)$ and $\hat{w}_k = \hat{A}\hat{g}_k$. Then the MG step size $\hat{\alpha}_k$ is defined by $\hat{\alpha}_k = \text{argmin}_{\alpha>0}\|\nabla f(\hat{x}_k - \alpha\hat{g}_k)\|_2$. Note that

$$\hat{g}_k = \hat{A}\hat{x}_k - \hat{b} = C^{-1}(Ax_k - b) = C^{-1}g_k,$$

and

$$\hat{w}_k = \hat{A}\hat{g}_k = \left(C^{-1}AC^{-1}\right)C^{-1}g_k.$$

Now define z_k, q_k and p_k such that,

$$Mz_k \coloneqq g_k, \;\; q_k \coloneqq C\hat{w}_k = Az_k \;\; \text{and} \;\; Mp_k \coloneqq q_k. \tag{3.3}$$

The first and third definitions in (3.3) are meant to avoid inverse matrix calculations. From the definition of the step size we obtain $\hat{\alpha}_k = \hat{g}_k^T\hat{w}_k/\|\hat{w}_k\|_2^2$. We transform through (3.3) the numerator as

$$\hat{g}_k^T\hat{w}_k = (C^{-1}g_k)^T\hat{A}(C^{-1}g_k) = g_k^TM^{-1}AM^{-1}g_k = z_k^TAz_k = z_k^Tq_k,$$

and the denominator becomes

$$\hat{w}_k^T\hat{w}_k = z_k^TAC^{-1}C^{-1}Az_k = q_k^TM^{-1}q_k = q_k^Tp_k,$$

obtaining

$$\hat{\alpha}_k = \frac{z_k^Tq_k}{q_k^Tp_k}. \tag{3.4}$$

Now, we shall reproduce the smoothing technique as in [24]. Consider the next MG iterated point

$$\hat{y}_k = \hat{x}_k - \hat{\alpha}_k\hat{g}_k = Cx_k - \hat{\alpha}_kC^{-1}g_k = C(x_k - \hat{\alpha}_kz_k),$$

and

$$\hat{r}_k = \hat{g}_k - \hat{\alpha}_k\hat{w}_k = C^{-1}(g_k - \hat{\alpha}_kAz_k) = C^{-1}(g_k - \hat{\alpha}_kq_k),$$

from which we express $u_k \coloneqq C^{-1}\hat{y}_k$ and $v_k \coloneqq C\hat{r}_k$ by

$$u_k = x_k - \hat{\alpha}_kz_k \;\; \text{and} \;\; v_k = g_k - \hat{\alpha}_kq_k. \tag{3.5}$$

The delaying smoothing step-size is defined for the "hat" system as

$$\hat{\beta}_k \coloneqq \text{argmin}_{\beta\in\mathbb{R}}\|\nabla\hat{f}(\hat{x}_{k-1} + \beta(\hat{y}_k - \hat{x}_{k-1}))\|_2,$$

which can be written as

$$\hat{\beta}_k = \hat{g}_{k-1}^T(\hat{g}_{k-1} - \hat{r}_k)/\|\hat{g}_{k-1} - \hat{r}_k\|_2^2.$$

Each factor gives

$$\hat{g}_{k-1}^T(\hat{g}_{k-1} - \hat{r}_k) = g_{k-1}^T C^{-1}(C^{-1}g_{k-1} - \hat{r}_k) = g_{k-1}^T M^{-1}(g_{k-1} - v_k)$$

and

$$\|\hat{g}_{k-1} - \hat{r}_k\|_2^2 = (\hat{g}_{k-1} - \hat{r}_k)^T(\hat{g}_{k-1} - \hat{r}_k) = (g_{k-1} - v_k)^T M^{-1}(g_{k-1} - v_k),$$

from which we obtain

$$\hat{\beta}_k = \frac{g_{k-1}^T s_k}{(g_{k-1} - v_k)^T s_k}, \text{ where } Ms_k := g_{k-1} - v_k. \tag{3.6}$$

Lastly, from

$$\hat{x}_{k+1} = \hat{x}_{k-1} + \hat{\beta}_k(\hat{y}_k - \hat{x}_{k-1})$$

and

$$\hat{g}_{k+1} = \hat{g}_{k-1} + \hat{\beta}_k(\hat{r}_k - \hat{g}_{k-1})$$

we obtain

$$x_{k+1} = x_{k-1} + \hat{\beta}_k(u_k - x_{k-1}) \text{ and } g_{k+1} = g_{k-1} + \hat{\beta}_k(v_k - g_{k-1}). \tag{3.7}$$

The preconditioned delayed weighted gradient method can be summarized by equations (3.3) – (3.7) in the Algorithm 2. The input data for the algorithm are the coefficient matrix A, the preconditioner M, the observation vector b, and the starting point x_0.

The computational cost of the preconditioned delayed weighted gradient method is the computational cost of DWGM plus the cost of solving three linear systems. Of course, three linear systems increases the computational cost. But, as we are working with symmetric definite systems, Jacobi, SSOR or incomplete Cholesky preconditioners are good choices that make the cost not increase excessively. In short, although it is necessary to solve three linear systems per iteration, they all have the same associated matrix which is typically diagonal or triangular. Thus, as mentioned in section 5, each linear system can be solved with a cost $\mathcal{O}(n)$ or $\mathcal{O}(n^2)$ operations. In the end, we expect the higher computational cost per iteration of PDWGM algorithm is compensated by the smaller number of iterations when compared to the DWGM algorithm. The convergence properties for PDWGM are inherited from DWGM, since the iterations are equivalent through the "hat" transformations.

Algorithm 2 PDWGM

Require: $A, M \in \mathbb{R}^{n\times n}$ SPD, $x_0 \in \mathbb{R}^n$, $x_{-1} = x_0$, $g_0 = g(x_0)$, $g_{-1} = g_0$, $\varepsilon > 0$.

Solve $Mz_0 = g_0$
$k = 0$
while $\|g_k\|_2 > \varepsilon$ **do**
    $q_k = Az_k$
    Solve $Mp_k = q_k$
    $\alpha_k = z_k^T q_k / q_k^T p_k$
    $u_k = x_k - \alpha_k z_k$
    $v_k = g_k - \alpha_k q_k$
    Solve $Ms_k = g_{k-1} - v_k$
    $\beta_k = g_{k-1}^T s_k / (g_{k-1} - v_k)^T s_k$
    $x_{k+1} = x_{k-1} + \beta_k(u_k - x_{k-1})$
    $g_{k+1} = g_{k-1} + \beta_k(v_k - g_{k-1})$
    Solve $Mz_{k+1} = g_{k+1}$
    $k = k + 1$
end while

4 CONVERGENCE

We now present the convergence result of the PDWGM. As the matrix C is a linear operator on a finite-dimensional linear space, then C is continuous. It means that convergence properties for $\{\hat{x}_k\}$ or $\{\hat{g}_k\}$ follow directly from the convergence of $\{x_k\}$ or $\{g_k\}$, and vice-versa. Associated to a positive definite matrix B, let us denote the $B-$norm by $\|\cdot\|_B = \|B(\cdot)\|_2$.

Lemma 4.1. *Let $\{x_k\}$ be the sequence generated by the Algorithm 1. Then for each k*

$$\|g_{k+1}\|^2_{M^{-1/2}} \leq \|r_k\|^2_{M^{-1/2}} < \|g_k\|^2_{M^{-1/2}} \leq \|r_{k-1}\|^2_{M^{-1/2}}. \tag{4.1}$$

Proof. From Lemma 1 in [25], when applied to the "hat" problem (3.2), we have $\|\hat{g}_{k+1}\|_2 \leq \|\hat{r}_k\|_2 < \|\hat{g}_k\|_2 \leq \|\hat{r}_{k-1}\|_2$, which in turn can be written as in (4.1). □

The next lemma establishes bounds on the eigenvalues of a product of definite positive matrices in terms of a product on the eigenvalues of their factors.

Lemma 4.2. *Given definite positive matrices A and B,* $\lambda_1(A) \geq \lambda_2(A) \geq, \ldots, \geq \lambda_n(A) > 0$ *and* $\lambda_1(B) \geq \lambda_2(B) \geq, \ldots, \geq \lambda_n(B) > 0$ *the respective eigenvalues, then for all* $i, j, k \in \{1, 2, \ldots, n\}$ *such that* $j + k \leq i + 1$,

$$\lambda_i(AB) \leq \lambda_j(A)\lambda_k(B) \tag{4.2}$$

$$\text{and} \quad \lambda_{n-j+1}(A)\lambda_{n-k+1}(B) \leq \lambda_{n-i+1}(AB). \tag{4.3}$$

In particular, for $i = 1, \ldots, n$

$$\lambda_i(A)\lambda_n(B) \leq \lambda_i(AB) \leq \lambda_1(A)\lambda_i(B). \tag{4.4}$$

For a proof see [6, Fact 8.18.17]. We shall apply this lemma to express the convergence rate of Algorithm 1 while running with preconditioning:

Theorem 4.1. *Consider the problem (1.2), the sequence* $\{x_k\}$ *generated by Algorithm 2, and let the eigenvectors for* $A^{1/2}$ *and* M^{-1} *be* $\lambda_1(A^{1/2}) > \lambda_2(A^{1/2}) > \ldots > \lambda_n(A^{1/2}) > 0$ *and* $\lambda_1(M^{-1/2}) > \lambda_2(M^{-1/2}) > \ldots > \lambda_n(M^{-1/2}) > 0$ *respectively. Then the sequence* $\{x_k\}$ *converges to* $A^{-1}b$ *Q-linearly with convergence factor*

$$\frac{\lambda_1(A^{1/2})\lambda_1(M^{-1/2}) - \lambda_n(A^{1/2})\lambda_n(M^{-1/2})}{(\lambda_1(A^{1/2}) + \lambda_n(A^{1/2}))\lambda_n(M^{-1/2})}.$$

Proof. By the Theorem 1 in [25] applied to the "hat" problem (3.2), we have for each k that

$$\|\hat{g}_{k+1}\| \leq \left(\frac{\hat{\lambda}_1 - \hat{\lambda}_n}{\hat{\lambda}_1 + \hat{\lambda}_n}\right)\|\hat{g}_k\|, \tag{4.5}$$

where $\hat{\lambda}_i$ stands for the i-th eigenvalue of $\hat{A}^{1/2}$. Now, since $\hat{A} = M^{-1/2}AM^{-1/2}$ we have for each i, $\lambda_i(\hat{A}^{1/2}) = \lambda_i((M^{-1/2}AM^{-1/2})^{1/2}) = \lambda_i(A^{1/2}M^{-1/2})$. Then we can write the factor as

$$\frac{\lambda_1(A^{1/2}M^{-1/2}) - \lambda_n(A^{1/2}M^{-1/2})}{\lambda_1(A^{1/2}M^{-1/2}) + \lambda_n(A^{1/2}M^{-1/2})}.$$

By using (4.4) twice, with $i = 1$ and $i = n$ we get

$$\lambda_1(A^{1/2})\lambda_n(M^{-1/2}) \leq \lambda_1(A^{1/2}M^{-1/2}) \leq \lambda_1(A^{1/2})\lambda_1(M^{-1/2})$$

and

$$\lambda_n(A^{1/2})\lambda_n(M^{-1/2}) \leq \lambda_n(A^{1/2}M^{-1/2}) \leq \lambda_1(A^{1/2})\lambda_n(M^{-1/2}).$$

From above relations we obtain

$$\lambda_1(A^{1/2}M^{-1/2}) - \lambda_n(A^{1/2}M^{-1/2}) \leq \lambda_1(A^{1/2})\lambda_1(M^{-1/2}) - \lambda_n(A^{1/2})\lambda_n(M^{-1/2})$$

and

$$\lambda_1(A^{1/2}M^{-1/2}) + \lambda_n(A^{1/2}M^{-1/2}) \geq \lambda_1(A^{1/2})\lambda_n(M^{-1/2}) + \lambda_n(A^{1/2})\lambda_n(M^{-1/2}),$$

which leads to the factor. Now, by using the relation $\|\hat{g}_k\| = \|g_k\|_{M^{-1/2}}$, we can write (4.5) as

$$\|g_{k+1}\|_{M^{-1/2}} \leq \left(\frac{\lambda_1(A^{1/2})\lambda_1(M^{-1/2}) - \lambda_n(A^{1/2})\lambda_n(M^{-1/2})}{(\lambda_1(A^{1/2}) + \lambda_n(A^{1/2}))\lambda_n(M^{-1/2})} \right) \|g_k\|_{M^{-1/2}}.$$

It follows that $\{g_k\}$ converges to zero Q-linearly with convergence factor

$$\frac{\lambda_1(A^{1/2})\lambda_1(M^{-1/2}) - \lambda_n(A^{1/2})\lambda_n(M^{-1/2})}{(\lambda_1(A^{1/2}) + \lambda_n(A^{1/2}))\lambda_n(M^{-1/2})}$$

and hence, since A is positive definite, we also conclude that $\{x_k\}$ tends to the unique minimizer of f when k goes to infinity. □

Let us consider the spectral condition number $\kappa_2(\hat{A})$ of $\hat{A}$. Since preconditioners are meant to improve the conditionning of a matrix, in general we will have $\kappa_2(A) \geqslant \kappa_2(\hat{A})$. The convergence factor can be rewritten as $(\kappa_2(\hat{A}) - 1)/(\kappa_2(\hat{A}) + 1)$. Since the function $f(x) = (x-1)/(x+1)$ is increasing and $\kappa_2(A) \geqslant \kappa_2(\hat{A})$, then the preconditioned DWGM converges in fewer iterations than the ordinary DWGM.

Now, we want to prove that PDWGM admits finite termination, in exact arithmetic.

In [2] it was demonstrated the $A-$orthogonality property of all previous gradients. Applied to the "hat" problem (3.2), it follows that for all k,

$$\hat{g}_k^T \hat{A} \hat{g}_j = 0, \ \ \forall j \leq k-1. \tag{4.6}$$

By translating to the original problem we obtain for each k,

$$g_k^T M^{-1} A M^{-1} g_j = 0, \ \ \forall j \leq k-1. \tag{4.7}$$

In other words, g_k is $M^{-1}AM^{-1}-$orthogonal to all previous gradient vectors. Notice that (4.7) can be written as $z_k^T A z_j = 0, \ \ \forall j \leq k-1$. The property (4.7) leads to the demonstration of the finite convergence theorem.

Theorem 4.2. *For any initial guess $x_0 \in \mathbb{R}^n$, PDWGM applied to problem (1.2) generates the iterated points x_k, $k \geq 1$ such that $x_n = A^{-1}b$.*

Proof. From (4.7) we have that the n vectors g_k, $k = 0, 1, \ldots, n-1$ form an $M^{-1}AM^{-1}-$orthogonal set, and then they form a linearly independent set of n vectors in $\mathbb{R}^n$. Therefore, the next vector, $g_n \in \mathbb{R}^n$ must be zero to be able to keep the $M^{-1}AM^{-1}-$orthogonality with all the previous gradient vectors. Since, $g_n = Ax_n - b = 0$ we obtain our claim. □

Another nice property of the DWGM algorithm preserved by our preconditioned version is that it terminates at p iterations, where p stands for the number of distinct eigenvalues of $M^{-1}AM^{-1}$. The key related results are Lemma 7 and Theorem 9 on [2], which we write without proof at the sequel. Notice that for each k the gradient vector generated by PDWGM belongs to the Krylov subspace $\mathscr{K}_{k+1}(M^{-1}AM^{-1}, g_0)$ depending on A, g_0 and M.

Lemma 4.3. *In the algorithm PDWGM, for all $k \geq 1$,*

$$g_k \in \mathscr{K}_{k+1}(M^{-1}AM^{-1}, g_0) := span\{g_0, M^{-1}AM^{-1}g_0, \ldots, (M^{-1}AM^{-1})^k g_0\}.$$

Theorem 4.3. *If $M^{-1}AM^{-1}$ has only $p < n$ distinct eigenvalues, then for any initial guess $x_0 \in \mathbb{R}^n$, the algorithm PDWGM generates the iterated points x_k, $k \geq 1$, such that $x_p = A^{-1}b$.*

Notice from (2.1) that to move from x_k to x_{k+1} the Algorithm 1 looks for a point in the line which passes through x_{k-1} and x_k from which walk in the gradient direction, involving these two search directions. Analogously, in the case of Algorithm 2, the search involves the directions $x_{k-1} - x_k$ and the transformed gradient z_k. The next theorem establishes the A-orthogonality of the sequences of these search directions. First notice that

$$\hat{g}_k^T \hat{A}(\hat{x}_k - \hat{x}_{k-1}) = g_k^T M^{-1}(g_k - g_{k-1}) = z_{k+1}^T (g_k - g_{k-1}).$$

Lemma 4.4. *In the algorithm PDWGM, the following statements hold for all $k \geq 1$.*

a) $z_{k+1}^T (v_k - g_{k-1}) = 0.$

b) $z_{k+1}^T (g_k - g_{k-1}) = 0.$

c) $z_{k+1}^T g_{k+1} = z_{k+1}^T g_k = z_{k+1}^T g_{k-1}.$

d) $\beta_k = z_{k-1}^T (g_{k-1} - v_k)/(\|g_{k-1} - g_k\|_{M^{-1/2}}^2 - [(z_k^T A z_k)^2/\|Az_k\|_{M^{-1/2}}^2]) > 1.$

Proof.

a) From steps 10, 12 and 13 of the algorithm we get

$$\begin{aligned}
&z_{k+1}^T (v_k - g_{k-1}) \\
&= g_{k+1}^T M^{-1}(v_k - g_{k-1}) \\
&= (g_{k-1} + \beta_k (v_k - g_{k-1}))^T M^{-1}(v_k - g_{k-1}) \\
&= g_{k-1}^T M^{-1}(v_k - g_{k-1}) + \frac{g_{k-1}^T M^{-1}(g_{k-1} - v_k)}{(g_{k-1} - v_k)^T M^{-1}(g_{k-1} - v_k)} (v_k - g_{k-1})^T M^{-1}(v_k - g_{k-1}) \\
&= 0.
\end{aligned}$$

b) From step 8, (a) and (4.7) we have

$$\begin{aligned} z_{k+1}^T(g_k - g_{k-1}) &= g_{k+1}^T M^{-1}(g_k - g_{k-1}) = g_{k+1}^T M^{-1}(v_k + \alpha_k A z_k - g_{k-1}) \\ &= g_{k+1}^T M^{-1}(v_k - g_{k-1}) + \alpha_k g_{k+1}^T M^{-1} A M^{-1} g_k = 0. \end{aligned}$$

c) By steps 8, 12 and 13

$$\begin{aligned} z_{k+1}^T g_{k+1} &= g_{k+1}^T M^{-1} g_{k+1} = g_{k+1}^T M^{-1}(g_{k-1} + \beta_k(v_k - g_{k-1})) \\ &= g_{k+1}^T M^{-1}(g_{k-1} + \beta_k(g_k - g_{k-1}) - \alpha_k \beta_k A z_k) \\ &= g_{k+1}^T M^{-1} g_{k-1} + \beta_k g_{k+1}^T M^{-1}(g_k - g_{k-1}) - \alpha_k \beta_k g_{k+1}^T M^{-1} A M^{-1} g_k. \end{aligned}$$

The second term is zero by part (b) and the third by (4.7). Then, we obtain $z_{k+1}^T g_{k+1} = z_{k+1}^T g_{k-1}$. Again from (b) we notice that $z_{k+1}^T g_k = z_{k+1}^T g_{k-1}$ obtaining our claim.

d) By Cauchy-Schwarz inequality and step 13 we obtain

$$\begin{aligned} z_{k-1}^T g_k &= g_{k-1}^T M^{-1} g_k = \|\hat{g}_{k-1}\| \|\hat{g}_k\| < \|\hat{g}_{k-1}\|^2 \\ &= g_{k-1}^T M^{-1} g_{k-1} = z_{k-1}^T g_{k-1}. \end{aligned}$$

Also,

$$z_{k-1}^T v_k = z_{k-1}^T(g_k - \alpha_k A z_k) = z_{k-1}^T g_k - \alpha_k z_{k-1}^T A z_k.$$

By using the A-orthogonality property (4.7) we obtain $z_{k-1}^T v_k = z_{k-1}^T g_k$. Therefore, the last two expressions lead to

$$z_{k-1}^T(g_{k-1} - v_k) = z_{k-1}^T(g_{k-1} - g_k) > 0.$$

This means that the numerator on step 10 is positive, and so, since the denominator can be written as $\|g_{k-1} - v_k\|_{M^{-1/2}}^2$, also $\beta_k > 0$ for all $k \geq 0$. Now we examine the denominator. By steps 4, 6 and 8, and algebraic manipulation we get

$$\begin{aligned} &\|g_{k-1} - v_k\|_{M^{-1/2}}^2 \\ &= \|g_{k-1} - g_k + \alpha_k A z_k\|_{M^{-1/2}}^2 \\ &= \|g_{k-1} - g_k\|_{M^{-1/2}}^2 + 2\alpha_k z_k^T A M^{-1}(g_{k-1} - g_k) + \alpha_k^2 z_k^T A M^{-1} A z_k \\ &= \|g_{k-1} - g_k\|_{M^{-1/2}}^2 + 2\alpha_k(z_k^T A z_{k-1} - z_k^T A z_k) + \alpha_k^2 z_k^T A M^{-1} A z_k \\ &= \|g_{k-1} - g_k\|_{M^{-1/2}}^2 - 2\left(\frac{z_k^T A z_k}{z_k A M^{-1} A z_k}\right) z_k^T A z_k + \left(\frac{z_k^T A z_k}{z_k^T A M^{-1} A z_k}\right)^2 z_k^T A M^{-1} A z_k \\ &= \|g_{k-1} - g_k\|_{M^{-1/2}}^2 - \frac{(z_k^T A z_k)^2}{z_k^T A M^{-1} A z_k}. \end{aligned}$$

So, $\beta_k = z_{k-1}^T(g_{k-1} - v_k)/(\|g_{k-1} - g_k\|_{M^{-1/2}}^2 - [(z_k^T A z_k)^2/\|A z_k\|_{M^{-1/2}}^2])$. Since β_k and the numerator $z_{k-1}^T(g_{k-1} - v_k)$ are strictly positive, then the denominator must also be strictly positive. Then,

$$0 < (z_k^T A z_k)^2/\|A z_k\|_{M^{-1/2}}^2 < \|g_{k-1} - g_k\|_{M^{-1/2}}^2.$$

From (c) we know $z_k^T(g_k - g_{k-1}) = 0$, and hence

$$0 = (g_k - g_{k-1} + g_{k-1})^T M^{-1}(g_k - g_{k-1}) = \|g_{k-1} - g_k\|^2_{M^{-1/2}} - z_{k-1}^T(g_{k-1} - g_k),$$

obtaining that $z_{k-1}^T(g_{k-1} - g_k) = \|g_{k-1} - g_k\|^2_{M^{-1/2}}$. By step 8 and (4.7) we have

$$z_{k-1}^T(g_k - v_k) = -\alpha_k z_{k-1}^T A z_k = 0,$$

so $z_{k-1}^T g_k = z_{k-1}^T v_k$. This fact is used to conclude that the numerator in (d) is strictly bigger than the denominator and since both are positive we finally have $\beta_k > 1$ for all k.

□

Now we are ready to establish the abovementioned orthogonality result:

Theorem 4.4. *The algorithm PDWGM generates sequences g_k and z_k such that, for $k \geq 2$,*

$$z_k^T(g_j - g_{j-1}) = 0, \ \ \forall 1 \leq j \leq k. \tag{4.8}$$

Proof. From Lemma 4.4 (b) we get $z_2^T(g_1 - g_0) = 0$, and so the result is true for $k = 2$. Let us assume by induction on k that (4.8) holds up to $k = \hat{k} \geq 3$ and consider the next iteration. So we need to show that $z_{\hat{k}+1}^T(g_j - g_{j-1}) = 0$ for all $1 \leq j \leq \hat{k}$. When $j = \hat{k}$ the result follows directly from part (b) of lemma above. Now, if $j \leq \hat{k} - 2$, by using steps 8 and 12, the inductive hypothesis and (4.7) we have

$$\begin{aligned} z_{\hat{k}+1}^T(g_j - g_{j-1}) &= g_{\hat{k}+1}^T M^{-1}(g_j - g_{j-1}) \\ &= [g_{\hat{k}-1} + \beta_{\hat{k}}(v_{\hat{k}} - g_{\hat{k}-1})]^T M^{-1}(g_j - g_{j-1}) \\ &= [(1-\beta_{\hat{k}})g_{\hat{k}-1} + \beta_k v_{\hat{k}}]^T M^{-1}(g_j - g_{j-1}) \\ &= \beta_{\hat{k}} v_{\hat{k}}^T M^{-1}(g_j - g_{j-1}) \\ &= \beta_{\hat{k}}(g_{\hat{k}} - \alpha_{\hat{k}} A z_{\hat{k}})^T M^{-1}(g_j - g_{j-1}) \\ &= -\beta_{\hat{k}}\alpha_{\hat{k}} g_{\hat{k}}^T M^{-1} A M^{-1}(g_j - g_{j-1}) = 0. \end{aligned}$$

Finally, when $j = \hat{k} - 1$, by using steps 8, 12 and 13, as well as the inductive hypothesis and (4.7) we have

$$\begin{aligned} z_{\hat{k}+1}^T(g_{\hat{k}-1} - g_{\hat{k}-2}) &= g_{\hat{k}+1}^T M^{-1}(g_{\hat{k}-1} - g_{\hat{k}-2}) \\ &= g_{\hat{k}+1}^T M^{-1}(g_{\hat{k}-1} - v_k + v_k - g_{\hat{k}-2}) \\ &= g_{\hat{k}+1}^T M^{-1}(g_{\hat{k}} - \alpha_{\hat{k}} A z_{\hat{k}} - g_{\hat{k}-2}) \\ &= g_{\hat{k}+1}^T M^{-1}(g_{\hat{k}} - g_{\hat{k}-2}) \\ &= g_{\hat{k}+1}^T M^{-1}(g_{\hat{k}-2} + \beta_{\hat{k}-1}(v_{\hat{k}-1} - g_{\hat{k}-2}) - g_{\hat{k}-2}) \\ &= \beta_{\hat{k}-1} g_{\hat{k}+1}^T M^{-1}(v_{\hat{k}-1} - g_{\hat{k}-2}) \\ &= \beta_{\hat{k}-1} z_{\hat{k}+1}^T(v_{\hat{k}-1} - g_{\hat{k}-2}). \end{aligned}$$

Therefore, $(\beta_{\hat{k}}-1)z_{\hat{k}-1}^T(v_{\hat{k}-1}-g_{\hat{k}-2})=0$. Since $\beta_{\hat{k}}>1$ for all $k\geq 1$ and noticing that $z_{\hat{k}-1}^T v_{\hat{k}-1}=z_{\hat{k}-1}^T g_{\hat{k}-1}$, we conclude $z_{\hat{k}-1}^T(v_{\hat{k}-1}-g_{\hat{k}-2})=0$, and so (4.8) is established. □

5 NUMERICAL EXPERIMENTS

In this section we show the preconditioned DWGM algorithm has a numerical behaviour similar to the preconditioned CG algorithm. We present some numerical experiments to validate our claim. In all examples we compare the performance of the CG [22], preconditioned CG [22], DWGM [25] and the preconditioned DWGM algorithms. All the experiments were performed on an intel(R) CORE(TM) i7-4770, CPU 3.40 GHz with 16 GB RAM.

We propose, initially, an experiment to compare the performance of DWGM and PDWGM with the most robust iterative method for solving strictly convex quadratic problems, the conjugate gradient method and its preconditioned version.

We obtained seventy matrices from the SuiteSparse Matrix Collection[1] [13, 23]. Then we solved the resulting seventy linear systems with the CG, preconditioned CG, DWGM and the preconditioned DWGM algorithms with $b=[1,1,\ldots,1]^T$ and $x_0=[0,0,\ldots,0]^T$. The stopping criterium used is

$$\|\nabla f(x_k)\|_2 \leqslant 10^{-5}.$$

For the seventy experiments we used the Jacobi preconditioner [20] to perform this experiment. But similar results are obtained if we use different preconditioners, like incomplete Cholesky factorization [17] or SSOR [17], for example.

Performance profiles [15] comparing CPU time and number of iterations are presented on Figure 1. Note that CG and DWGM algorithms have similar behaviour in terms of CPU time and number of iterations. On the other hand, the comparison between PCG and PDWGM present some differences. First, we can note that, in general, PDWGM converge in less iterations than PCG, but with a higher CPU time. But we emphasize the differences are not significant.

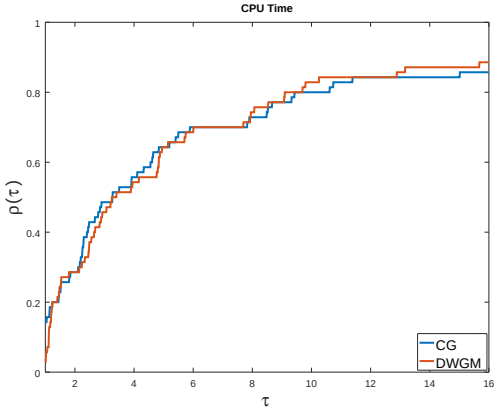
Table 1 presents, for fourteen selected matrices, the number of iterations (niter), the error norm ($e_k=\|x_k-x_*\|_2$) and the gradient error norm ($r_k=\|Ax_k-b\|_2$).

The most evident characteristic that we can observe in Table 1 is that the DWGM algorithm reaches the given tolerance in fewer iterations than the CG algorithm, this fact was also observed in [2]. A second observation is that both PCG and PDWGM have a similar behavior. It means they reach the given tolerance with a similar number of iterations. The CPU times of PCG and PDWGM are also similar, sometimes PCG is faster, sometimes PDWGM is faster. But, overall the CPU times do not differ a lot. Moreover, as shown in the next figure, the gradient curves of PCG and PDWGM are resemblant. In short, the differences between PDWGM and PCG are not significant.

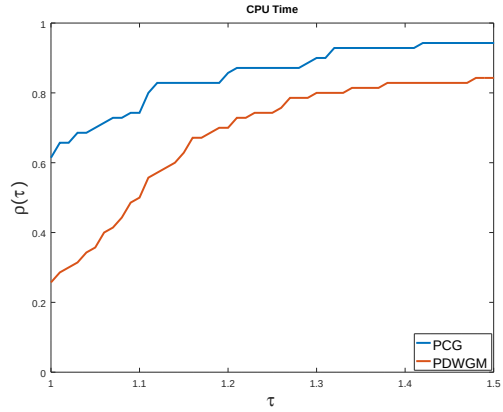
[1] formerly the University of Florida Sparse Matrix Collection.

Table 1: Iteration information of CG, PCG, DWGM and PDWGM for fourteen models.

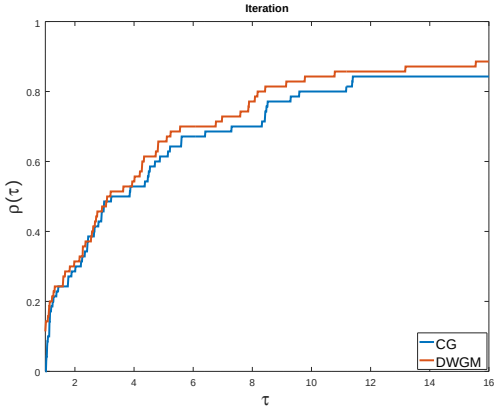
	mesh3em5				Muu			
	niter	e_k	r_k	CPU(s)	niter	e_k	r_k	CPU(s)
CG	**13**	2.7597e-005	4.1695e-005	0.0010	84	2.1429e-004	9.6983e-009	0.0849
PCG	**13**	1.8247e-005	3.5884e-005	**0.0009**	**30**	1.0021e-004	7.8161e-009	**0.0329**
DWGM	**13**	2.9586e-005	4.0226e-005	**0.0009**	83	3.3700e-004	7.6887e-009	0.1019
PDWGM	**13**	2.0587e-005	3.4235e-005	0.0019	**30**	1.2917e-004	6.8319e-009	0.0369
	Chem97ZtZ				bodyy4			
	niter	e_k	r_k	CPU(s)	niter	e_k	r_k	CPU(s)
CG	137	3.1782e-010	9.6090e-009	0.0199	273	1.4529e-009	9.7342e-009	0.274857
PCG	**32**	4.5905e-010	6.6680e-009	**0.0059**	214	3.7600e-009	9.7342e-009	**0.229854**
DWGM	135	9.0344e-010	7.5092e-009	0.0239	262	5.7226e-009	9.3859e-009	0.336092
PDWGM	34	1.0241e-010	3.1926e-009	0.0069	**212**	5.7226e-009	9.7141e-009	0.318831
	bundle1				nasa2146			
	niter	e_k	r_k	CPU(s)	niter	e_k	r_k	CPU(s)
CG	242	4.1278e-019	9.3649e-009	0.9597	429	3.9752e-014	9.2289e-009	0.1979
PCG	66	1.8484e-019	7.1148e-009	0.2953	323	4.1317e-014	9.7221e-009	**0.1479**
DWGM	241	8.7440e-019	9.4834e-009	0.9444	414	1.9318e-013	9.4622e-009	0.1971
PDWGM	**65**	3.7965e-019	8.3751e-009	**0.2758**	**312**	1.3832e-013	9.4843e-009	0.1559
	wathen120				msc00726			
	niter	e_k	r_k	CPU(s)	niter	e_k	r_k	CPU(s)
CG	413	2.2624e-009	9.2956e-009	1.4831	1512	9.5563e-014	7.8397e-009	0.3557
PCG	51	2.0597e-010	6.2655e-009	**0.1848**	126	2.9107e-015	7.0725e-009	0.0339
DWGM	405	6.9872e-009	9.2236e-009	1.6300	1520	3.4609e-013	1.9936e-008	0.3957
PDWGM	**50**	4.4364e-010	8.1356e-009	0.2158	**122**	1.7645e-014	7.0100e-009	**0.0309**
	bcsstm12				Pres_Poisson			
	niter	e_k	r_k	CPU(s)	niter	e_k	r_k	CPU(s)
CG	4313	1.5650e-005	6.9687e-009	0.8158	2243	5.3538e-005	9.8029e-007	8.7070
PCG	504	7.2098e-006	7.1205e-009	**0.0999**	767	2.8998e-005	9.8692e-007	**3.1102**
DWGM	3396	3.0353e-004	1.9472e-008	0.8019	2119	5.7822e-004	4.8639e-006	8.5900
PDWGM	**464**	9.3194e-005	8.9432e-009	0.1019	**758**	7.2819e-005	1.2784e-006	3.2261
	msc04515				1138_bus			
	niter	e_k	r_k	CPU(s)	niter	e_k	r_k	CPU(s)
CG	4812	4.8974e-014	9.6160e-007	2.9952	2000	4.1944e-005	8.8203e-005	0.2198
PCG	3825	1.4561e-012	8.6307e-007	**2.4625**	**970**	1.7307e-005	9.6794e-005	**0.1249**
DWGM	4698	1.0657e-012	6.0433e-005	3.1961	1966	2.5406e-004	1.0520e-004	0.2188
PDWGM	**3771**	1.7856e-012	6.8977e-005	2.4694	975	3.1397e-005	6.8587e-005	0.1329
	cbuckle				cvxbqp1			
	niter	e_k	r_k	CPU(s)	niter	e_k	r_k	CPU(s)
CG	4944	7.4180e-005	8.7746e-005	17.4160	8148	8.6930e-006	9.8426e-005	23.7433
PCG	970	7.9553e-005	9.7548e-005	3.7778	5103	8.6930e-006	9.9575e-005	16.9593
DWGM	3960	9.6921e-004	9.9202e-005	15.3422	5942	2.7696e-004	9.9960e-005	22.8188
PDWGM	**785**	0.0021630	9.6522e-005	**3.0412**	**3793**	3.1636e-004	9.9852e-005	**16.0228**



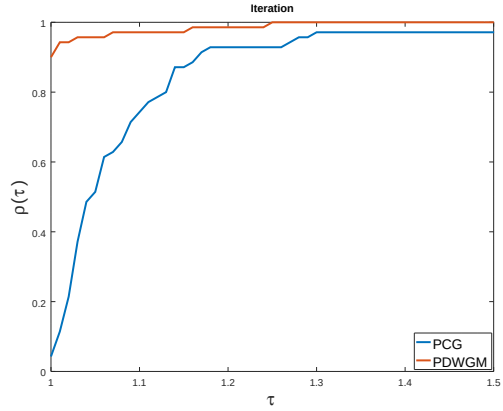
(a) CG/DWGM CPU time comparison.



(b) PCG/PDWGM CPU time comparison.



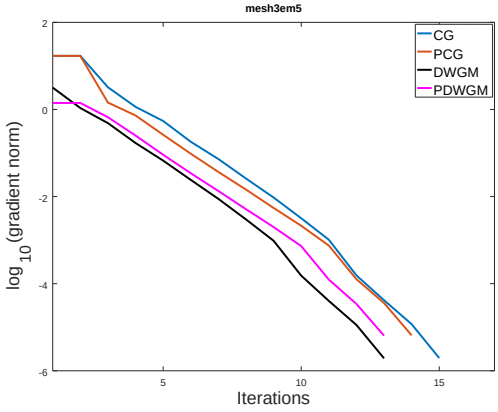
(c) CG/DWGM number of iterations comparison.



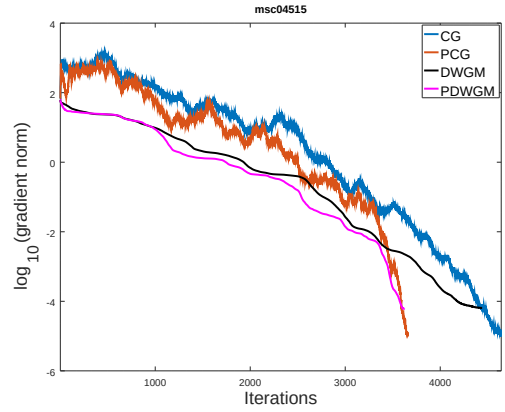
(d) PCG/PDWGM number of iterations comparison.

Figure 1: Performance profiles comparing CG, PCG, DWGM and PDWGM.

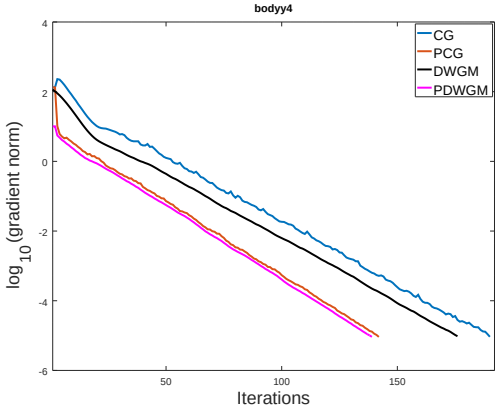
To corroborate the results presented in Table 1 we display six graphs with the conjugate gradient, preconditioned conjugate gradient, DWGM and PDWGM number of iterations versus $\log_{10}(\|\nabla f(x_k)\|_2)$ comparison. As pointed out above, PCG and PDWGM have a similar behaviour. Note that the gradient curves of PCG and PDWGM have similar pattern. Nevertheless, as pointed out by [25], the computational cost per iteration for CG is $2n^2+9n-2$ flops while the computation cost per iteration for PCG is $2n^2+17n-4$ flops. Thus, the computational cost of PCG is the cost of CG plus the resolution of a linear system, while the computational cost of PDWM is the cost of DWGM plus the resolution of three linear systems. Note that, by using the Jacobi preconditioner or a preconditioner based on the incomplete Cholesky factorization the linear systems to be solved, for each iteration are either diagonal or triangular with computational costs of $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ flops, respectively. As final observation, another effect we note is the smoothed curves of DWGM and PDWGM when compared to the well known "crispness" of the CG and PCG curves.



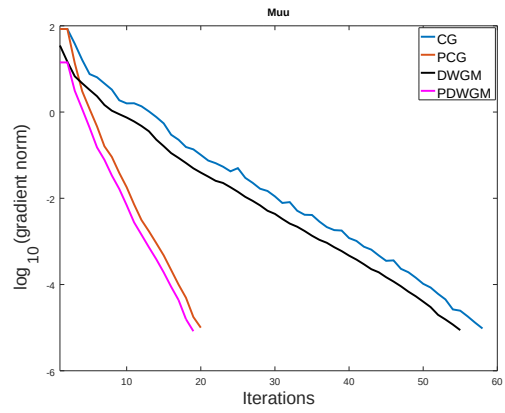
(a) mesh3em5.



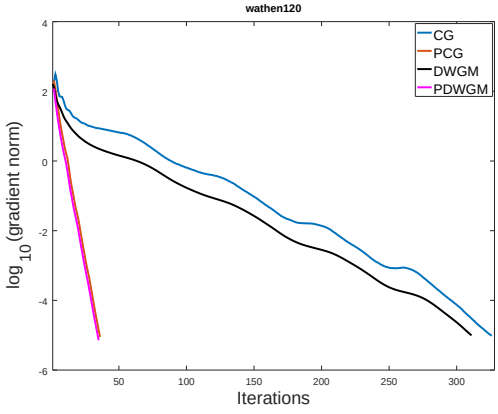
(b) msc04515.



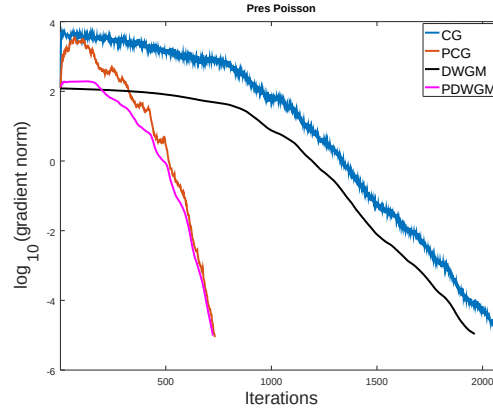
(c) bodyy4.



(d) Muu.



(e) wathen120.



(f) Pres_Poisson.

Figure 2: Comparison between CG, PCG, DWGM and PDWGM for six different models.

The experiment described above has, as objective, the comparison between the DWGM/PDWGM and CG/PCG algorithms. Now we propose a second experiment in order to illustrate the finite termination property in the number of distinct eigenvalues of the PDWGM algorithm (Theorem 4.3), and how preconditioning can influence over the number of steps.

This experiment is based on a construction of a preconditioned matrix with a predefined number of distinct eigenvalues. In the following procedure we present how to construct a matrix A and a preconditioner $M = C^2$ such that $\hat{A} = C^{-1}AC^{-1}$ has a defined number of distinct eigenvalues.

- Let $Q \in \mathbb{R}^{n\times n}$ be an orthogonal matrix;
- Let $v \in \mathbb{R}^n$ be a vector with random entries between 0 and 1;
- Define $A = QT_1Q^T$, with $T_1 = \text{diag}(v)$;
- Let $l \in \mathbb{R}^p$ such that $l_1 > l_2 > \cdots > l_p$ (eigenvalues);
- Define the algebraic multiplicity n_i of each l_i, $i = 1,\ldots,p$ such that $n_1 + n_2 + \cdots + n_p = n$;
- Define $L = \text{diag}(l_1,\ldots,l_1,l_2,\ldots,l_2,\ldots,l_p,\ldots,l_p) \in \mathbb{R}^{n\times n}$;
- Define a diagonal matrix T_2 such that $T_2(i,i) = L(i,i)/T_1(i,i)$, $i = 1,\ldots,n$;
- Define $M^{-1} = QT_2Q^T$ or $C^{-1} = QT_2^{1/2}Q^T$.

Note that,

$$\begin{aligned}\hat{A} &= C^{-1}AC^{-1} = \left(QT_2^{1/2}Q^T\right)\left(QT_1Q^T\right)\left(QT_2^{1/2}Q^T\right)\\ &= Q\left(T_2^{1/2}T_1T_2^{1/2}\right)Q^T = QLQ^T.\end{aligned}$$

Then, $\hat{A}$ is a preconditioned matrix with clustered eigenvalues.

The objective of this experiment is to present some examples of the finite termination of DWGM and PDWGM. Firstly, a random orthogonal matrix Q is defined based on the QR decomposition [20] of a given random matrix. Then, the procedure above is run 15 times, generating 15 pairs (A,M). As in the first example we compare DWGM and PDWGM 15 times and then we take the average and the standard deviation of the results.

The results are presented in Table 2. For DWGM and PDWGM we compare the number of iterations, the absolute error, the norm of the residual and CPU time. Moreover, n an p represent the problem dimension and the number of distinct eigenvalues of $\hat{A}$, respectively. Observe that for small instances the number of iterations of the algorithm without preconditioning is proportional to the dimension n, while the number of iterations of the preconditioned is proportional to the number of different eigenvalues p on the problem, as Theorem 4.3 claim.

Table 2: Iteration information of CG, PCG, DWGM and PDWGM.

	n=7 and p=3			
	niter	e_k	r_k	CPU(s)
DWGM	7 ± 0	2.8042e-013 $\pm$ 8.5845e-013	1.7165e-013 $\pm$ 5.2349e-013	1.0973e-003 $\pm$ 1.0944e-003
PDWGM	3 ± 0	4.6075e-013 $\pm$ 1.4180e-012	1.5011e-013 $\pm$ 4.6798e-013	6.9978e-004 $\pm$ 9.4906e-004
	n=100 and p=11			
	niter	e_k	r_k	CPU(s)
DWGM	50.70 $\pm$ 10.79	1.3530e-005 $\pm$ 1.4383e-005	7.7019e-007 $\pm$ 1.7123e-007	1.1192e-002 $\pm$ 2.3468e-003
PDWGM	10.90 $\pm$ 0.31	1.2297e-007 $\pm$ 3.8885e-007	7.0609e-008 $\pm$ 2.2328e-007	4.4976e-003 $\pm$ 7.0674e-004
	n=1000 and p=51			
	niter	e_k	r_k	CPU(s)
DWGM	143.50 $\pm$ 26.69	1.0365e-004 $\pm$ 4.9420e-005	9.4354e-007 $\pm$ 4.3212e-008	0.0785 $\pm$ 0.0138
PDWGM	33.00 $\pm$ 4.24	1.6387e-006 $\pm$ 8.6370e-007	5.6995e-007 $\pm$ 1.5505e-007	0.0620 $\pm$ 0.0101
	n=7000 and p=623			
	niter	e_k	r_k	CPU(s)
DWGM	373.40 $\pm$ 73.30	1.1075e-003 $\pm$ 7.3672e-004	9.6649e-007 $\pm$ 1.7105e-008	14.9948 $\pm$ 2.9115
PDWGM	121.40 $\pm$ 31.77	2.3420e-006 $\pm$ 7.2454e-007	9.1388e-007 $\pm$ 5.6455e-008	12.2710 $\pm$ 3.2368

6 CONCLUSIONS

The delayed weighted gradient method is a two-step gradient method that aims to correct the Cauchy's point and produce a faster solution than the classical gradient method. We have presented and discussed the derivation of the preconditioned delayed weighted gradient method. Also we have shown the preconditioned conjugate gradient and the preconditioned delayed weighted gradient method have similar results, and preconditioned conjugate gradient has a similar computational cost per iteration than the preconditioned delayed weighted gradient method. Convergence and theoretical properties of the preconditioned delayed weighted gradient method are proved.

REFERENCES

[1] H. Akaike. On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. *Annals of the Institute of Statistical Mathematics*, **11** (1959), 1–16.

[2] R. Andreani & M. Raydan. Properties of the delayed weighted gradient method. *Computational Optimization and Applications*, **78** (2021), 167–180.

[3] R.D. Asmundis, D. di Serafino, W.W. Hager, G. Toraldo & H. Zhang. An efficient gradient method using the Yuan steplength. *Computational Optimization and Applications*, **59** (2014), 541–563.

[4] R.D. Asmundis, D. di Serafino, F. Riccio & G. Toraldo. On spectral properties of steepest descent methods. *IMA Journal of Numerical Analysis*, **33**(4) (2013), 1416–1435.

[5] J. Barzilai & J.M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, **8**(1) (1988), 141–148.

[6] D.S. Bernstein. "Matrix Mathematics: Theory, Facts, and Formulas". Princeton University Press, second ed. (2011).

[7] D. Bertaccini & F. Durastante. "Iterative Methods and Preconditioning for Large and Sparse Linear Systems with Applications". Chapman and Hall/CRC, New York (2018).

[8] E. Birgin, I. Chambouleyron & J.M. Martínez. Estimation of the optical constants and the thickness of thin films using unconstrained optimization. *Computational Physics*, (151) (1999), 862–880.

[9] E.G. Birgin, J.M. Martínez & M. Raydan. Spectral projected gradient methods: review and perspectives. *Journal of Statistical Software*, **60**(3) (2014), 1–21.

[10] A. Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comptes Rendus Sci. Paris*, **25** (1847), 536–538.

[11] Y.H. Dai, Y. Huang & X.W. Liu. A family of spectral gradient methods for optimization. *Computational Optimization and Applications*, **74** (2019), 43–65.

[12] Y.H. Dai & L.Z. Liao. R-linear convergence of the Barzilai and Borwein gradient method. *IMA Journal of Numerical Analysis*, **22**(1) (2002), 1–10.

[13] T.A. Davis & Y. Hu. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Softw.*, **38**(1) (2011). doi:10.1145/2049662.2049663.

[14] D. di Serafino, V. Ruggiero, G. Toraldo & L. Zanni. On the steplength selection in gradient methods for unconstrained optimization. *Applied Mathematics and Computation*, **318** (2018), 176–195.

[15] E.D. Dolan & J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, **91** (2002), 201–213. doi:10.1007/s101070100263.

[16] M.A. Figueiredo, R. Nowak & S. Wright. Projection for sparse reconstruction: application to compressed sensing and other inverse problems. *IEEE J. Sel. Top. Signal Process*, (1) (2007), 586–597.

[17] W. Ford. "Numerical Linear Algebra with Applications: Using MATLAB". Elsevier Inc, New York (2015).

[18] G.E. Forsythe & T.S. Motzkin. Asymptotic properties of the optimum gradient method, Preliminary report. *Bull. Amer. Math. Soc*, **57** (1951), 304–305.

[19] A. Friedlander, J.M. Martínez, B. Molina & M. Raydan. Gradient method with retards and generalizations. *SIAM Journal on Numerical Analysis*, **36**(1) (1999), 275–289.

[20] G.H. Golub & C.F. Van Loan. "Matrix Computations". Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 4th ed. (2012).

[21] Y. Huang, Y.H. Dai, X.W. Liu & H. Zhang. Gradient methods exploiting spectral properties. *Optimization Methods and Software*, **35**(4) (2020), 681–705.

[22] C.T. Kelley. "Iterative Methods for Linear and Nonlinear Equations". Society for Industrial and Applied Mathematics (1995).

[23] S.P. Kolodziej, M. Aznaveh, M. Bullock, J. David, T.A. Davis, M. Henderson, Y. Hu & R. Sandstrom. The SuiteSparse Matrix Collection Website Interface. *Journal of Open Source Software*, **4**(35) (2019), 1244. doi:10.21105/joss.01244.

[24] J.L. Lamotte, B. Molina & M. Raydan. Smooth and adaptive gradient method with retards. *Mathematical and Computer Modelling*, **36** (2002), 1161–1168.

[25] H.F. Oviedo-Leon. A delayed weighted gradient method for strictly convex quadratic minimization. *Computational Optimization and Applications*, **74** (2019), 729–746.

[26] T. Serafini, G. Zanghirati & L. Zanni. Gradient projection methods for large quadratic programs and applications in training support vector machines. *Optimization Methods and Software*, (20) (2005), 353–378.

[27] B.V. Shah, R.J. Buehler & O. Kempthorne. Some Algorithms for Minimizing a Function of Several Variables. *Journal of the Society for Industrial and Applied Mathematics*, **12**(1) (1964), 74–92.

[28] H. Sorenson. Comparison of some conjugate direction procedures for function minimization. *Journal of the Franklin Institute*, **288**(6) (1969), 421–441.

[29] Y.X. Yuan. A new stepsize for the steepest decent method. *Journal of Computational Mathematics*, **24**(2) (2006), 149–156.