Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 9, n. 1, 2022.

Trabalho apresentado no XLI CNMAC, Unicamp - Campinas - SP, 2022.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics
Preprint

Least-Squares Delayed Weighted Gradient Method

Rafael Aleixo [1]
Federal University of Santa Catarina, Department of Mathematics, Blumenau, Brazil
Hugo Lara Urdaneta [2]
Federal University of Santa Catarina, Department of Control and Automation Engineering, and Computing, Blumenau, Brazil

Abstract. The delayed weighted gradient algorithm (DWGM) is proved to be a robust iterative procedure to solve convex quadratic optimization problems. Its theoretical and numerical performance is similar to the conjugate gradient method. In this work we specialize the DWGM to deal with least-squares problems. Numerical experimentation is offered to show the effectiveness of the approach.

Keywords. Least squares, linear systems, iterative method, delayed weighted gradient method.

1 Introduction

Least-squares problems arise in different applications of mathematics, like statistics, econometrics, engineering et cetera. So, it is important to have algorithms that address these problems. Many algorithms were proposed to solving least-squares problems, among them we mention CGLS/CGNR [11], LSQR [17] and LSMR [10].

On the other hand, to minimize a convex quadratic form

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

where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite (SPD) and $b \in \mathbb{R}^n$ several methodologies were proposed [4, 7, 9, 12, 14, 15, 18]. Gradient methods play a key role in this matter. The steepest descent (SD) method proposed by Cauchy [5] 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

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

It is proven the SD method converges Q-linearly [1]. A variant of the SD method called minimal gradient (MG) step length [3] aims to minimize the gradient norm. The solution for the above problem is

$$\alpha_k^{\mathrm{MG}} = \frac{g_k^T A g_k}{g_k^T A^2 g_k}.$$

It is well known that gradient method either with α_k^{SD} or with α_k^{MG} performs very poorly [1, 6].

[1]rafael.aleixo@ufsc.br
[2]hugo.lara.urdaneta@ufsc.br

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics. v. 9, n. 1, 2022.

Methods which use two step-sizes are alternatives to accelerate gradient-based methods, by imposing retard on the process (see [7]). The Delayed Weighted Gradient Method (DWGM) is a two step-size gradient method that was introduced by Oviedo-Leon in 2019 [16]. DWGM is a gradient method developed to solving symmetric positive definite systems or a strictly convex minimization problem. The main objective of DWGM is to accelerate the convergence of the gradient method by a two-step iteration. Moreover, a smoothing process is applied. In this work, we present the least-squares version of the delayed weighted gradient method in order to solve problems like the unsymmetric squared linear equations system, linear least-squares and regularized least-squares.

The remainder of this article is organized as follows: In the next section we describe the Delayed Weighted Gradient Method, while in section 3 the least-squares version is proposed. Section 4 show the numerical experimentation, and finally at section 5 some concluding remarks.

2 Delayed Weighted Gradient Method

We consider the strictly convex quadratic minimization problem,

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

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

Let $x_0 \in \mathbb{R}^n$ be a starting point and $g_k = g(x_k)$. Asmundis et al. [3] present the minimum gradient method that is given by

$$x_{k+1} = x_k - \alpha_k^{\mathrm{MG}} g_k, \quad \text{with} \quad \alpha_k^{\mathrm{MG}} = g_k^T w_k / \|w_k\|_2^2,$$

where the step-size is defined as $\alpha_k^{\mathrm{MG}} = \operatorname{argmin}_{\alpha>0} \|\nabla f(x_k - \alpha g_k)\|_2$, and $w_k = Ag_k$. In short, the minimum gradient norm method calculates the next iterate as the point alongside the current gradient at which the norm of the next gradient is minimized.

As a two step gradient method, delayed weighted gradient method incorporates a delaying step defined as follows [16]: The first stage uses the ordinary minimum gradient point $y_k = x_k - \alpha_k^{\mathrm{MG}} g_k$. Then, calculates the next iterate as

$$x_{k+1} = x_{k-1} + \beta_k (y_k - x_{k-1}), \quad \text{where} \quad \beta_k = g_{k-1}^T (g_{k-1} - r_k) / \|g_{k-1} - r_k\|_2^2.$$

The step-size is defined by $\beta_k = \operatorname{argmin}_{\beta\in\mathbb{R}} \|\nabla f(x_{k-1} + \beta(y_k - x_{k-1}))\|_2$. It is straightforward to prove that $\nabla f(x_{k-1} + \beta(y_k - x_{k-1})) = g_{k-1} - \beta(g_{k-1} - r_k)$, for $r_k = g_k - \alpha_k^{\mathrm{MG}} w_k$. This leads to $\beta_k = \operatorname{argmin}_{\beta\in\mathbb{R}} \|g_{k-1} - \beta(g_{k-1} - r_k)\|_2 = g_{k-1}^T (g_{k-1} - r_k)/\|g_{k-1} - r_k\|_2^2$.

Some of the properties that DWGM enjoys, established in [2, 16] include the non negativity 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.

Algorithm 1 LSDWGM

Require: $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $x_0 \in \mathbb{R}^n$, $x_{-1} = x_0$, $\epsilon > 0$.

```
g = A x_0 - b
s_{-1} = s_0 = A^T g
p_0 = A s_0
k = 0
while ||s_k||_2 > ε do
    w_k = A^T p_k
    α_k = s_k^T w_k / w_k^T w_k
    y_k = x_k - α_k s_k
    r_k = s_k - α_k w_k
    β_k = s_{k-1}^T (s_{k-1} - r_k) / ||s_{k-1} - r_k||_2^2
    x_{k+1} = x_{k-1} + β_k (y_k - x_{k-1})
    s_{k+1} = s_{k-1} + β_k (r_k - s_{k-1})
    p_{k+1} = A s_{k+1}
    k = k + 1
end while
```

3 Least-Squares Delayed Weighted Gradient Method

In this section, we present a variant of the DWGM, called least-squares DWGM (LSDWGM), for computing the solution x to the following problems [17]:

$$\begin{aligned} &\text{Unsymmetric equations:} && \text{solve } Ax = b \\ &\text{Linear Least-Squares:} && \text{minimize } \|Ax - b\|_2 \\ &\text{Regularized Least-Squares:} && \text{minimize } \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2, \end{aligned} \tag{2}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $\lambda \geqslant 0$, with $m \geqslant n$ and $rank(A) = n$.

All the problems above can be solved by applying the DWGM to the normal equations

$$A^T Ax = A^T b.$$

Forming the normal equation is a simple form to symmetrize the problem we are trying to solve, but from the numerical point of view can be disastrous for large-scale problems or ill-conditioned ones. The main reason is the explicit use of A^TA that either impacts on the space complexity because in general if A is a sparse matrix then A^TA is less sparse or because $\kappa(A^TA) = \kappa(A)^2$, where $\kappa(A)$ is the condition number of A. But the time complexity is the most impacted with the calculation of A^TA because it requires $\mathcal{O}(m^2n)$ flops to multiply A^T by A.

Thus, an algorithm developed to solving problem (2) must have the ability to avoid forming the product A^TA. The implementation of the least-squares delayed weighted gradient method (LSDWGM) is straightforward. The ideia is to replace an iteration over $g_k = Ax_k - b$, by an iteration over $s_k = A^T g_k$, the residual of the normal linear system $A^TAx = A^Tb$. One possible algorithm is presented in Algorithm 1.

Table 1: Basic information of the models.

Model	Rows	Columns	Nonzeros	$\kappa_2(A)$
well1850	1850	712	8755	$1.113129e+02$
orani678	2529	2529	90158	$9.579953e+03$
lp_pilot*	4180	1441	44375	$2.661950e+03$
struct4	4350	4350	237798	$7.238826e+04$
lp_pilot87*	6680	2030	74949	$8.153260e+03$
rat*	9408	3136	268908	$1.269130e+00$
model9*	10939	2879	55956	$3.163690e+20$
model5*	11802	1888	89925	$7.244326e+19$
192bit	13691	13682	154303	$5.875923e+64$
lp_osa_07*	25067	1118	144812	$6.803249e+02$
testbig*	31223	17613	61639	$6.693068e+02$
car4*	33052	16384	63724	$1.193634e+00$
ts-palko*	47235	22002	1076903	$2.140518e+02$
mod2	66409	34774	199810	$8.527910e+03$

* for the tests we used the transpose of the model

The theoretical properties of the algorithm are inherited from the original DWGM [2, 16]. Note that problem (2) is a convex quadratic optimization problem with hessian given by A^TA.

4 Numerical Experiments

We chose fourteen datasets from the SuiteSparse Matrix Collection [8, 13] in order to evaluate the differences between the least-squares DWGM and the ordinary DWGM applied to the normal equations $A^TAb = A^Tb$. All the experiments were performed on a intel(R) CORE(TM) i7-4770, CPU 3.40 GHz with 16 GB RAM. Table 1 presents some basic information about the models we are using.

One of the main issues related to the least-squares solutions of linear systems is the possibility of forming the matrix A^TA. As explained before, the calculation of A^TA must be avoided. Note that DWGM was designed to run with SPD matrices, but our inputs are not SPD. Thus, in order to run DWGM for these type of datasets we must run it on the normal equations $A^TAx = A^Tb$, which is SPD (we selected full-rank matrices). With this in mind, we calculated the least-squares solutions of fourteen linear systems $Ax = b$ with the DWGM and LSDWGM algorithms, where the matrices A are described in Table 1, $b = [1, 1, \ldots, 1]^T$ and the starting point is $x_0 = [0, 0, \ldots, 0]^T$. The stopping criterium is $\|\nabla f(x_k)\|_2 \leqslant \epsilon$, for a given data-dependent ϵ, it varies from 10^{-5} to 10^{-8}.

In Tables 2 and 3 we present a performance comparison of DWGM performed on A^TA and the least-squares DWGM performed on A. We compare the norm of the solution error $e_k = x_k - x^*$, the norm of the residual error $r_k = Ae_k$ and the norm of the normal equations residual error $s_k = A^Tr_k$. The "theoretical solution" x^* used on the calculation of e_k was calculated using the Matlab's command $A\backslash b$. Besides, we present the CPU time comparison. Naturally, we expect that DWGM is faster than LSDWGM because LSDWGM requires an extra matrix-vector multiplication per iteration. Thus, for the CPU time comparison we compare the time of DWGM plus the time of A^TA calculation with the time of LSDWGM. Overall, the LSDWGM is faster than the DWGM plus A^TA calculation. It happens, as explained before, because the A^TA takes $\mathcal{O}(m^2n)$ flops to be

Table 2: Iteration information of DWGM and LSDWGM for fourteen models.

	well1850				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	400	2.5353e-004	1.4065e-005	9.6105e-007	0.0619
LSDWGM	400	2.6056e-004	1.4390e-005	9.8246e-007	0.0609
	orani678				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	2956	5.1285e-006	1.9783e-007	3.3051e-008	26.8766
LSDWGM	2989	5.2135e-006	2.0006e-007	2.5798e-008	2.13678
	lp_pilot				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	3844	2.4746e-005	3.8453e-006	1.0210e-006	2.59552
LSDWGM	3859	2.4810e-005	3.8552e-006	1.0199e-006	1.52112
	struct4				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	93622	6.0459e-005	6.4229e-007	1.9420e-005	756.879
LSDWGM	92221	6.0903e-005	6.1554e-007	1.7585e-005	141.407
	lp_pilot87				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	10236	7.2589e-008	3.2606e-008	2.3191e-006	13.5752
LSDWGM	10174	7.0216e-008	3.1441e-008	4.3149e-006	6.48128
	rat				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	9	3.5902e-007	5.5133e-007	8.6030e-007	0.1249
LSDWGM	9	3.5902e-007	5.5133e-007	8.6030e-007	0.0109
	model9				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	4607	1.3647e-005	1.0332e-005	3.9393e-005	3.22016
LSDWGM	4517	1.3534e-005	1.0222e-005	1.5405e-005	2.65256

performed which is more computationally expensive than the DWGM algorithm itself.

5 Conclusion

In this work we presented the least-squares version of the well known delayed weighted gradient method. Its implementation is straightforward and is based on iterations over the residual of the normal linear system or as a gradient of the convex quadratic form

$$f(x) = \frac{1}{2}x^T A^T Ax + x^T A^T b.$$

The numerical experiments demonstrated the method is robust and offers a good alternative to the DWGM applied to the normal equations.

Table 3: Iteration information of DWGM and LSDWGM for fourteen models.

	model5				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	6617	1.5582e-004	3.5101e-005	9.9800e-006	7.28182
LSDWGM	6625	1.5609e-004	3.5151e-005	9.9934e-006	5.11407
	192bit				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	4449	3.6584e-003	1.7346e-004	9.9971e-006	29.4261
LSDWGM	4311	3.6586e-003	1.7347e-004	9.9979e-006	8.70999
	lp_osa_07				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	171	3.4179e-008	1.6105e-007	8.1111e-007	0.1109
LSDWGM	190	3.1874e-008	1.4979e-007	7.2322e-007	0.2088
	testbig				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	49	6.1102e-007	1.4512e-006	5.4917e-006	1.1264
LSDWGM	47	4.9695e-007	7.5874e-007	5.0574e-006	0.0529
	car4				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	7	1.0495e-007	1.5882e-007	2.4144e-007	0.0330
LSDWGM	7	1.0495e-007	1.5882e-007	2.4144e-007	0.0090
	ts-palko				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	91	1.3661e-008	9.2081e-008	7.4641e-007	4.0327
LSDWGM	93	1.6256e-008	1.0862e-007	9.1126e-007	0.6766
	mod2				
	niter	$\|e_k\|_2$	$\|r_k\|_2$	$\|s_k\|_2$	CPU(s)
DWGM	25963	6.1060e-006	7.0713e-006	8.9153e-004	120.035
LSDWGM	26180	6.1070e-006	7.0724e-006	8.2159e-004	88.3913

References

[1] H. Akaike. "On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method". In: **Annals of the Institute of Statistical Mathematics** 11 (1959), pp. 1–16. DOI: 10.1007/BF01831719.

[2] R. Andreani and M. Raydan. "Properties of the delayed weighted gradient method". In: **Computational Optimization and Applications** 78 (2021), pp. 167–180. DOI: 10.1007/s10589-020-00232-9.

[3] R. De Asmundis et al. "An efficient gradient method using the Yuan steplength". In: **Computational Optimization and Applications** 59 (2014), pp. 541–563. DOI: 10.1007/s10589-014-9669-5.

[4] J. Barzilai and J. M. Borwein. "Two-point step size gradient methods". In: **IMA Journal of Numerical Analysis** 8.1 (1988), pp. 141–148. DOI: 10.1093/imanum/8.1.141.

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

[6] Y. H. Dai. "Alternate step gradient method". In: **Optimization** 52.4-5 (2003), pp. 395–415. DOI: 10.1080/02331930310001611547.

[7] Y. H. Dai, Y. Huang, and X. W. Liu. "A family of spectral gradient methods for optimization". In: **Computational Optimization and Applications** 74 (2019), pp. 43–65. DOI: 10.1007/s10589-019-00107-8.

[8] T. A. Davis and Y. Hu. "The University of Florida Sparse Matrix Collection". In: **ACM Trans. Math. Softw.** 38.1 (2011), pp. 1–25. DOI: 10.1145/2049662.2049663.

[9] D. di Serafino et al. "On the steplength selection in gradient methods for unconstrained optimization". In: **Applied Mathematics and Computation** 318 (2018), pp. 176–195. DOI: https://doi.org/10.1016/j.amc.2017.07.037.

[10] D. C. L. Fong and M. Saunders. "LSMR: An Iterative Algorithm for Sparse Least-Squares Problems". In: **SIAM Journal on Scientific Computing** 33.5 (2011), pp. 2950–2971. DOI: 10.1137/10079687X.

[11] R. W. Freund, G. H. Golub, and N. M. Nachtigal. "Iterative solution of linear systems". In: **Acta Numerica** 1 (1992), pp. 57–100. DOI: 10.1017/S0962492900002245.

[12] A. Friedlander et al. "Gradient method with retards and generalizations". In: **SIAM Journal on Numerical Analysis** 36.1 (1999), pp. 275–289. DOI: 10.1137/S003614299427315X.

[13] S. P. Kolodziej et al. "The SuiteSparse Matrix Collection website interface". In: **Journal of Open Source Software** 4.35 (2019), pp. 1244:1–1244:4. DOI: 10.21105/joss.01244.

[14] J. L. Lamotte, B. Molina, and M. Raydan. "Smooth and adaptive gradient method with retards". In: **Mathematical and Computer Modelling** 36.9 (2002), pp. 1161–1168. DOI: 10.1016/S0895-7177(02)00266-2.

[15] H. Oviedo, R. Andreani, and M. Raydan. "A family of optimal weighted conjugate-gradient-type methods for strictly convex quadratic minimization". In: **Numerical Algorithms** 90 (2022), pp. 1225–1252. DOI: 10.1007/s11075-021-01228-0.

[16] H. F. Oviedo-Leon. "A delayed weighted gradient method for strictly convex quadratic minimization". In: **Computational Optimization and Applications** 74 (2019), pp. 729–746. DOI: 10.1007/s10589-019-00125-6.

[17] C. C. Paige and M. A. Saunders. "LSQR: An algorithm for sparse linear equations and sparse least squares". In: **ACM Trans. Math. Softw.** 8.1 (1982), pp. 43–71. DOI: 10.1145/355984.355989.

[18] Y. X. Yuan. "A new stepsize for the steepest descent method". In: **Journal of Computational Mathematics** 24.2 (2006), pp. 149–156.

