Anderson Acceleration of Proximal Gradient Methods

Vien V. Mai* Mikael Johansson*

October 16, 2019

Abstract

Anderson acceleration is a well-established and simple technique for speeding up fixed-point computations with countless applications. Previous studies of Anderson acceleration in optimization have only been able to provide convergence guarantees for unconstrained and smooth problems. This work introduces novel methods for adapting Anderson acceleration to (non-smooth and constrained) proximal gradient algorithms. Under some technical conditions, we extend the existing local convergence results of Anderson acceleration for smooth fixed-point mappings to the proposed scheme. We also prove analytically that it is not, in general, possible to guarantee global convergence of native Anderson acceleration. We therefore propose a simple scheme for stabilization that combines the global worst-case guarantees of proximal gradient methods with the local adaptation and practical speed-up of Anderson acceleration.

1 Introduction

The last few decades have witnessed significant advances in the theory and practice of convex optimization based on first-order information [32, 3]. The worst-case oracle complexity has been established for many function classes [31] and algorithms with matching worst-case performance have been developed. However, these methods are only optimal in a worst-case (resisting oracle) sense, and are developed under the assumption that global function properties are known and constant. In practice, however, such constants are almost never known a priori. Moreover, their local values, which determine the actual practical performance, may be very different from their conservative global bounds and often change as the iterates approach optimum. It is also observed that acceleration methods such as Nesterov's accelerated gradient are very sensitive to misspecified parameters; slightly over- or under-estimating the strong convexity constant can have a severe effect on the overall performance of the algorithm [34]. Thus, strong practical performance of optimization algorithms requires *local adaption and acceleration*. Efficient line-search procedures [33], adaptive restart techniques [34] and nonlinear acceleration schemes [47] are therefore now receiving an increasing attention.

Extrapolation techniques have a long history in numerical analysis (see, e.g., [48, 7]). Recently, its idea has resurfaced in the first-order optimization literature [47, 53, 27, 19, 39]. Unlike momentum acceleration methods such as Polyak's heavy ball [38] and Nesterov's fast gradient [32], which require knowledge of problem parameters, classical extrapolation techniques for vector sequences such as minimal polynomial extrapolation [49], reduced rank extrapolation [15], vector epsilon algorithm [52], and Anderson acceleration [1] estimate the

*Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden. Emails: {maivv, mikaelj}@kth.se.

solution directly from the available iterate sequence. These methods enjoy favorable theoretical properties of Krylov subspace methods on quadratic problems and often perform equally well in practice on non-quadratic problems.

1.1 Related Work

Anderson acceleration (AA) was proposed in the 1960's to expedite solution times for nonlinear integral equations [1]. The technique has then been generalized to general fixed-point equations and found countless applications in diverse fields such as computational chemistry, physics, material science, etc. [41, 16, 51]. However, AA and optimization algorithms have been developed quite independently and only limited connections were discovered and studied [16, 18]. Very recently, the technique has started to gain a significant interest in the optimization community (see, e.g., [47, 46, 5, 53, 19, 39]). Specifically, a series of papers [47, 46, 5] adapt AA to accelerate several classical algorithms for unconstrained optimization; [53] studies a variant of AA for non-expansive operators; [19] proposes an application of AA to Douglas-Rachford splitting; and [39] uses AA to improve the performance of the ADMM method. There is also an emerging literature on applications of AA in machine learning [23, 28, 20, 36].

Although some initial success has been obtained for adapting AA to optimization algorithms, current research has focused almost exclusively on smooth unconstrained optimization. None of the proposed techniques are able to guarantee acceleration, or even convergence, for problems with a general convex constraint set. We are only aware of the very recent work [19], which considers linearly constrained minimization. There, using an additional stabilization strategy similar to the one in [53], the authors show *global and asymptotic* convergence, albeit no convergence rate is given. The main difficulty in constrained problems is that the extrapolated point in AA may lie outside the feasible set. Moreover, since AA relies on linearization of a mapping around its fixed-point, convergence guarantees of AA often require differentiability of the associated mapping, which is not possible in non-smooth and constrained optimization. Our aim with this paper is to address these limitations. To this end, we make the following contributions:

1. We propose an efficient AA scheme for the classical proximal gradient algorithm (PGA), which has a simple interpretation and avoids the feasibility problems of naïve AA. Under certain technical conditions, we extend the *local* convergence guarantee of limited-memory AA for smooth problems in [50] to the non-smooth case.

2. Local convergence properties of native AA have been studied in various settings [50, 47, 37, 24, 27]. However, whether native AA converges globally still remains largely unknown (cf. [19]). Here, we show a negative answer to this question. More specifically, we construct an unconstrained strongly convex problem for which we can prove analytically that AA fails to converge. We therefore stabilize the proposed method by a simple guard step that preserves the global worst-case convergence guarantees of PGA without sacrificing the local adaption and acceleration abilities of AA.

3. We adapt AA to the Bregman proximal gradient (BPG) family, where the mirror descent [31] and NoLips [2] methods are special instances. The method respects the structure of the BPG family and admits a simple and elegant interpretation. To the best of our knowledge, these are the first applications of AA to non-Euclidean geometry.

4. We perform extensive experiments on several important classes of constrained optimization problems and demonstrate consistent and dramatic speedups on real-world data-sets.

1.2 Notation

We denote by $\mathbb{R}_+$ the set of nonnegative real numbers. For a set $\mathcal{X}$, $\overline{\mathcal{X}}$ and $\operatorname{int}\mathcal{X}$ denote its closure and interior, respectively. The notation $\|\cdot\|$ refers to a general norm, and $\|\cdot\|_2$ is the Euclidean norm. The all-ones vector is denoted by $\mathbf{1}$.

2 Anderson acceleration

Let $g : \mathbb{R}^n \to \mathbb{R}^n$ be a mapping and consider the problem of finding a fixed-point of g:

$$\text{Find } x \in \mathbb{R}^n \text{ such that } x = g(x).$$

In contrast to the fixed-point iteration $y_{k+1} = g(y_k)$, which only uses the last iterate to generate a new estimate, AA tries to make better use of past information. Concretely, let $\{x_i\}_{i=0}^k$ be the sequence of iterates generated by AA up to iteration k. Here, we refer the term $r_k := g(x_k) - x_k$ as the residual in the kth iteration. Then, to form x_{k+1}, it searches for a point that has smallest residual within the subspace spanned by the $m+1$ most recent iterates. In other words, if we let $\bar{x}_k = \sum_{i=0}^m \alpha_i^k x_{k-i}$, AA seeks to find a vector of coefficients $\alpha^k = [\alpha_0^k, \dots, \alpha_m^k]^\top$ such that

$$\alpha^k = \underset{\alpha:\alpha^\top \mathbf{1}=1}{\operatorname{argmin}} \left\| g(\bar{x}_k) - \bar{x}_k \right\|. \tag{1}$$

However, since (1) can be hard to solve for a general nonlinear mapping g, AA uses

$$\alpha^k = \underset{\alpha:\alpha^\top \mathbf{1}=1}{\operatorname{argmin}} \left\| \sum_{i=0}^m \alpha_i g(x_{k-i}) - \sum_{i=0}^m \alpha_i x_{k-i} \right\|. \tag{2}$$

It is clear that Problems (1) and (2) are equivalent if g is an affine mapping. Let $R_k = [r_k, \dots, r_{k-m}]$ be the residual matrix at the kth iteration, Problem (2) can then be written as

$$\alpha^k = \underset{\alpha^\top \mathbf{1}=1}{\operatorname{argmin}} \left\| R_k \alpha \right\|. \tag{3}$$

With α^k computed, the next iterate of AA is then generated by

$$x_{k+1} = \sum_{i=0}^m \alpha_i^k g\left(x_{k-i}\right), \tag{4}$$

which in the affine case, is equivalent to applying the operator g to $\bar{x}_k$. When $m = 0$, AA reduces to the fixed-point iteration.

One of the reason that AA is so popular in engineering and scientific applications is that it can speed-up convergence with almost no additional tuning parameters and the extrapolation coefficients can be computed very efficiently. When the Euclidean norm is considered, Problem (3) is a simple least-squares, which admits a closed-form solution given by

$$\alpha^k = \frac{(R_k^\top R_k)^{-1}\mathbf{1}}{\mathbf{1}^\top (R_k^\top R_k)^{-1}\mathbf{1}}. \tag{5}$$

This can be solved by first solving the $m \times m$ normal equations $R_k^\top R_k x = \mathbf{1}$ and then normalizing the result to obtain $\alpha^k = x/(\mathbf{1}^\top x)$ [47]. Indeed, the computations can be done even more efficiently using QR decomposition. When passing from R_{k-1} to R_k, only the last column of R_{k-1} is removed and a new column is added. Thus, the corresponding Q and R matrices can be easily updated and the total cost is at most $O\left(m^2 + mn\right)$ [23]. Since m is typically between 1 and 10 in practice, this additional cost is negligible compared to the cost of evaluating g.

Algorithm 1 Anderson Acceleration

```
Input: $x_0$, $m \geq 0$, $g(\cdot)$
1: $x_1 \leftarrow g(x_0)$
2: for $k = 1, \ldots, K-1$ do
3:     $m_k \leftarrow \min(m, k)$
4:     $R_k \leftarrow [r_k, \ldots, r_{k-m_k}]$, where $r_i = g(x_i) - x_i$
5:     $\alpha^k \leftarrow \operatorname{argmin}_{\alpha^\top \mathbf{1}=1} \|R_k \alpha\|$
6:     $x_{k+1} \leftarrow \sum_{i=0}^{m_k} \alpha_i^k g(x_{k-i})$
7: end for
Output: $x_K$
```

2.1 Anderson acceleration for optimization algorithms

Since many optimization algorithms can be written as fixed-point iterations, they can be accelerated by the memory-efficient, line search-free AA method with almost no extra cost. For example, the classical gradient descent (GD) method for minimizing a smooth convex function f defined by

$$x_{k+1} = x_k - \gamma \nabla f(x_k),$$

is equivalent to the fixed-point iteration applied to $g(x) = x - \gamma \nabla f(x)$. Clearly, a fixed-point of g corresponds to an optimum of f. The intuition behind AA for GD is that smooth functions are well approximated by quadratic ones around their (unconstrained) optimum, so their gradients and hence g are linear. In such regimes, AA enjoys several nice properties of Krylov subspace methods. Specifically, consider a convex quadratic minimization problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2} x^\top A x - b^\top x, \tag{6}$$

where $A \in \mathbb{R}^{n \times n}$ is a symmetric positive semidefinite matrix and $b \in \mathbb{R}^n$. It has been shown in [51, 40] that AA with full information (i.e. setting $m = \infty$ in Step 3 of Algorithm 1) is essentially equivalent to GMRES [45]. Therefore, AA admits the convergence rate [30, 21]

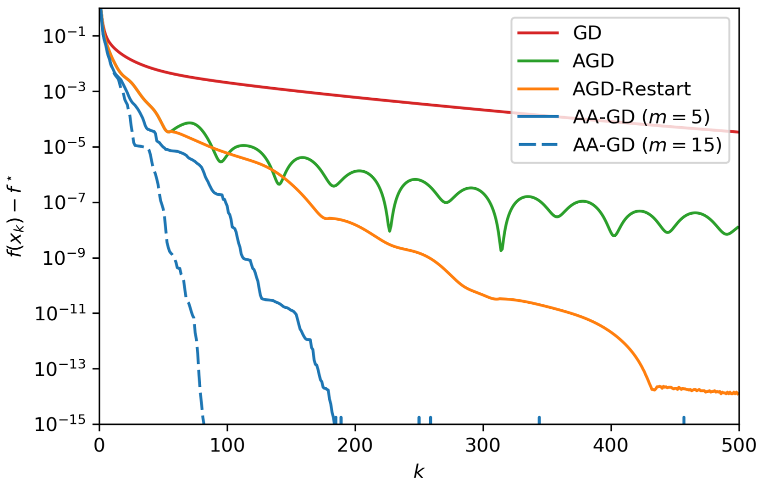
$$\|x_k - x^\star\|_2^2 \leq O\left(\min\left\{1/k^2, e^{-k/\sqrt{\kappa}}\right\}\right) \|x_0 - x^\star\|_2^2, \tag{7}$$

where $\kappa = \lambda_1(A)/\lambda_n(A)$ is the condition number. This rate shows a very strong adaptation ability and is attained without any knowledge of the problem at hand, a remarkable property of Krylov subspace methods. In contrast, Nesterov's accelerated gradient method (AGD) [32] can only achieve this rate if $\lambda_1(A)$ and $\lambda_n(A)$ are given a priori. When m is finite, it can be shown that AA will not do worse than GD for any m. More concretely, for any $m \in \{0, 1, \ldots\}$, AA achieves the worst-case guarantee

$$\|x_k - x^\star\|_2^2 \leq O\left(e^{-k/\kappa}\right) \|x_0 - x^\star\|_2^2,$$

which coincides with that of GD. The above observations imply that AA interpolates the two extreme regimes where $m = 0$ and $m = \infty$.

In practice, significant speed-ups and strong adaptation are often observed even with very small m. As an example, Figure 1 shows the performance of different algorithms applied to minimize a quadratic convex function in $n = 100$ dimensions with 25 nonzero eigenvalues such



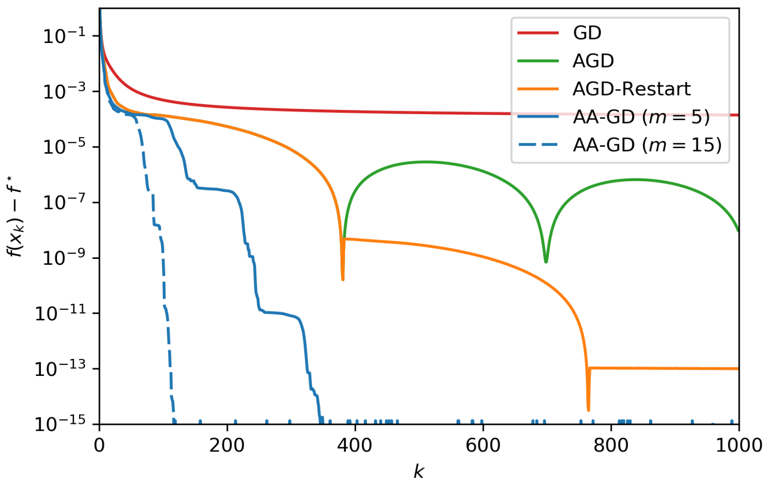


Figure 1: Performance of difference first-order optimization algorithms on quadratic convex problems. Here, $A \in \mathbb{R}^{100\times 100}$ is a symmetric positive semidefinite matrix with 25 nonzero eigenvalues. On the top $\lambda_1(A)/\lambda_{25}(A) = 10^3$ and on the bottom $\lambda_1(A)/\lambda_{25}(A) = 10^4$.

that $\lambda_1(A)/\lambda_{25}(A)$ equals 10^3 and 10^4 for the top and the bottom plots, respectively. We compared AA-GD with GD, AGD, and the adaptive restart scheme (AGD-Restart) in [34]. It should be noted that just like AA, the main objective of the AGD-Restart scheme is to achieve local adaptation. We can see that local adaptation and acceleration can dramatically improve the performance of an optimization algorithm. It is evident that AA initially converges at the same rate as AGD ($1/k^2$) and eventually switches to linear convergence, as suggested in (7), even with a very small value of m and a non-strongly convex objective.

If the function being minimized has a positive definite Hessian at the optimum, then near the solution it can be well approximated by a quadratic model

$$f(x) \approx f(x^\star) + (x - x^\star)^\top \nabla^2 f(x^\star)(x - x^\star).$$

Note that the matrix $\nabla^2 f(x^\star)$ may have smallest eigenvalue $\lambda_{\min}$ strictly greater than the *global* strong convexity constant μ. Thus, once we enter this regime, we may be able to achieve all the nice features of AA on quadratic problems discussed in the previous paragraphs.

2.2 Anderson acceleration as a multi-step method

It is known that AA is related to several iterative schemes such as multisecant quasi-Newton methods [16, 18, 51]. Here, we point out some connections between AA-GD and multi-step methods in optimization. To do so, let $\gamma_i^k := \sum_{j=i}^{m_k} \alpha_j^k$, $i \in \{1, \ldots, m_k\}$ and define $y_k^\alpha := \sum_{i=1}^{m_k} \alpha_i^k x_{k-i}$. AA-GD can then be written as

$$y_k^\alpha = x_k - \sum_{i=1}^{m_k} \gamma_i^k \left(x_{k-i+1} - x_{k-i}\right) \quad \text{and} \quad x_{k+1} = y_k^\alpha - \gamma \sum_{i=0}^{m_k} \alpha_i^k \nabla f(x_{k-i}). \tag{8}$$

Recall that Nesterov's accelerated gradient method (AGD) [32] can be written as

$$y_k = x_k + \beta_k(x_k - x_{k-1}) \quad \text{and} \quad x_{k+1} = y_k - \gamma \nabla f(y_k),$$

while Polyak's Heavy ball (HB) method [38] is given by

$$y_k' = x_k + \beta_k'(x_k - x_{k-1}) \quad \text{and} \quad x_{k+1} = y_k' - \gamma \nabla f(x_k),$$

where $\beta_k, \beta_k' > 0$ are extrapolation coefficients. Setting $m_k = 1$ in (8), the AA-GD method is analogous to AGD and HB with β_k, β_k' replaced by $-\gamma_1^k$. However, their update directions are chosen differently: AGD takes a step using the gradient at the extrapolated point y_k, HB uses the gradient at x_k, while AA-GD uses a combination of the gradients evaluated at x_k and x_{k-1}. For $m > 1$, AA-GD is similar to the MiFB method in [25]. However, unlike AA, there is currently no efficient way to select the coefficients in MiFB, thereby restricting its history parameter to $m = 1$ or 2.

3 Anderson Acceleration for Proximal Gradient Method

Consider a composite convex minimization problem of the form

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; f(x) + h(x), \tag{9}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is μ-strongly convex and L-Lipschitz smooth, *i.e.*

$$\begin{aligned} \|\nabla f(x) - \nabla f(y)\|_2 &\le L \|x - y\|_2, \quad \forall x, y \in \mathbb{R}^n, \\ f(x) \ge f(y) + \nabla f(y)^\top (x - y) &+ \frac{\mu}{2} \|x - y\|_2^2, \quad \forall x, y \in \mathbb{R}^n, \end{aligned}$$

and h is a proper closed and convex function. Recall that the proximal operator associated with h is defined as

$$\text{prox}_h(y) := \underset{x}{\text{argmin}} \left\{ \frac{1}{2} \|x - y\|_2^2 + h(x) \right\}. \tag{10}$$

A classical method for solving (9) is the proximal gradient algorithm (PGA)

$$x_{k+1} = \text{prox}_{\gamma h}\left(x_k - \gamma \nabla f(x_k)\right), \tag{11}$$

which can be seen as the fixed-point iteration for the mapping

$$g(x) = \operatorname{prox}_{\gamma h}\left(x - \gamma \nabla f(x)\right). \tag{12}$$

It is not difficult to show that $x^\star$ is a minimizer of (9) if and only if

$$x^\star = \operatorname{prox}_{\gamma h}\left(x^\star - \gamma \nabla f(x^\star)\right), \tag{13}$$

which implies that finding $x^\star$ amounts to finding a fixed-point of g.

Unlike the adaptive restart scheme [34], which has a straightforward extension to the proximal case, it is not that clear how to extend AA, since feasibility issues can arise from the extrapolation step. In light of our previous discussion, it would be natural to speed-up the PGA method by applying AA to the mapping g in (12). However, in many cases, the function h does not have full domain; for example, when h is the indicator function of some closed convex set. The resulting iterates can then become infeasible. Since AA forms an affine (and not a convex) combination in each step, there is no guarantee that the extrapolated point will lie inside $\operatorname{dom} h$. Nevertheless, if we rewrite the PGA iteration on the form:

$$y_{k+1} = x_k - \gamma \nabla f(x_k) \quad \text{and} \quad x_{k+1} = \operatorname{prox}_{\gamma h}\left(y_{k+1}\right), \tag{14}$$

and consider the mapping g defined as

$$g(y) = \operatorname{prox}_{\gamma h}\left(y\right) - \gamma \nabla f(\operatorname{prox}_{\gamma h}\left(y\right)), \tag{15}$$

then the fixed-point iteration $y_{k+1} = g(y_k)$ recovers exactly the PGA iteration in (14). It is clear that if $y^\star$ is a fixed-point of g, then $x^\star = \operatorname{prox}_{\gamma h}\left(y^\star\right)$ is an optimal solution to (9) since it satisfies condition (13). Now, to relate the convergence of the primal sequence $\{x_k\}$ and the auxiliary $\{y_k\}$, we use the following simple but useful observation: Suppose that $x^\star$ satisfies (13), then $y^\star = x^\star - \gamma \nabla f(x^\star)$ is a fixed-point of g defined in (15) and

$$\|x_k - x^\star\|_2 = \left\|\operatorname{prox}_{\gamma h}\left(y_k\right) - \operatorname{prox}_{\gamma h}\left(x^\star - \gamma \nabla f(x^\star)\right)\right\|_2 \leq \|y_k - y^\star\|_2,$$

where the last step follows from the nonexpansiveness of proximal operators. The inequality implies that if one can quickly drive $\{y_k\}$ to $y^\star$, then $\{x_k\}$ will quickly converge to $x^\star$.

From the above observations, we propose to use AA for accelerating the auxiliary sequence $\{y_k\}$ governed by g defined in (15). Since there are no restrictions on $\{y_k\}$, AA-PGA avoids the feasibility problems of naïve AA. Just like PGA, the algorithm requires only one gradient and one proximal evaluation per step. To the best of our knowledge, this is the first AA scheme that is able to handle a general convex constraint set, and it does so without adding extra computational cost to the AA algorithm. The resulting scheme, which we call AA-PGA, is summarized in Algorithm 2.

3.1 Convergence Guarantees

Having addressed the feasibility issue in the previous section, our next goal is to derive convergence guarantees for AA-PGA. To this end, we will extend the theory for limited-memory AA to a class of non-smooth problems which includes our proposed algorithm.

Although convergence properties of AA for linear mappings with full memory ($m = \infty$) are relatively well understood [51, 40], much less is known in the case of nonlinear mappings and limited-memory. The work [50] was the first to show that no matter what value $m \in \{0, 1, \ldots\}$ is used, AA does not harm the convergence of the fixed-point iteration when started near the

Algorithm 2 AA-PGA

```
Input: $y_0 \in \operatorname{dom} h$, $x_0 = \operatorname{prox}_{\gamma h}(y_0)$, $m \geq 0$
1: $y_1 \leftarrow x_0 - \gamma \nabla f(x_0)$, $x_1 \leftarrow \operatorname{prox}_{\gamma h}(y_1)$, $g_0 \leftarrow y_1$
2: for $k = 1, \ldots, K-1$ do
3:     $m_k \leftarrow \min(m, k)$
4:     $g_k \leftarrow x_k - \gamma \nabla f(x_k)$ and $r_k \leftarrow g_k - y_k$
5:     $R_k \leftarrow [r_k, \ldots, r_{k-m_k}]$
6:     $\alpha^k \leftarrow \operatorname{argmin}_{\alpha^\top \mathbf{1}=1} \|R_k \alpha\|$
7:     $y_{k+1} \leftarrow \sum_{i=0}^{m_k} \alpha_i^k g_{k-i}$
8:     $x_{k+1} \leftarrow \operatorname{prox}_{\gamma h}(y_{k+1})$
9: end for
Output: $x_K$
```

fixed point. The proof requires differentiability of g. However, in the context of composite convex optimization, the mapping g defined in (15) is, in general, non-differentiable. Therefore, the analysis in [50, 10] is not applicable anymore. Nevertheless, the key steps in the proof of [50, Theorem 2.3] essentially only need that

$$\left\|g(x) - g'(x^\star)(x - x^\star)\right\|_2 \leq c\,\|x - x^\star\|_2^2, \tag{16}$$

holds for some constant $c > 0$ and for all x sufficiently close to $x^\star$. Interestingly, this condition is very similar to assumptions which guarantee convergence of Newton's method for solving non-smooth nonlinear equations (see [17, Chapter 7] for an excellent review). Two key ingredients in the analysis of non-smooth Newton methods are the use of Clarke's generalized Jacobian [12] and the concept of *semi-smoothness* [29, 42], defined below.

Definition 3.1 (Clarke's Generalized Jacobian)**.** *Consider the mapping $F : \Omega \to \mathbb{R}^m$, with $\Omega \subset \mathbb{R}^n$ open, and assume that F is Lipschitz continuous near a given point x of interest. Let $\mathcal{D}_F$ be the set where F is differentiable. We will write $JF(x)$ for the usual $m \times n$ Jacobian matrix whenever $x \in \mathcal{D}_F$. Then, the* B-differential *of F at x is defined as*

$$\partial_{\mathrm{B}} F(x) = \left\{ \lim_{k\to\infty} JF(x_k) : x_k \to x, x_k \in \mathcal{D}_F \right\}.$$

The set

$$\partial F(x) := \operatorname{conv}\{\partial_{\mathrm{B}} F(x)\},$$

where conv *denotes the convex hull, is called* Clarke's generalized Jacobian *of F at x,*

Definition 3.2 (Semi-Smoothness)**.** *Let $F : \Omega \to \mathbb{R}^n$ be a locally Lipschitz continuous function and $\partial F(x)$ be the Clarke generalized Jacobian of F at x. We say that F is strongly semismooth at $x \in \Omega$ if:*
i) F is directionally differentiable at x; and
ii) For any $J \in \partial F(x+h)$ and $h \in \mathbb{R}^n$,

$$\|F(x+h) - F(x) - Jh\|_2 \leq O(\|h\|_2^2) \text{ as } h \to 0.$$

If F is strongly semi-smooth at each $x \in \Omega$, then we say that F is strongly semi-smooth on Ω.

Some important classes of strongly semi-smooth functions include differentiable functions with a Lipschitz continuous gradient; the norm function $\|\cdot\|_p$ for every $p \in [1, \infty]$; and piecewise affine functions. This class of mappings is rich enough to cover most of the proximable functions arising in machine learning applications. For example, projections onto the non-negative orthant, box constraints and the probability simplex are component-wise piecewise affine [35]. More generally, the projection onto any polyhedral set is piecewise linear [44]. Projections onto the symmetric cone, hence the second-order and positive semidefinite cones, are also strongly semi-smooth [17]. The proximal operator of the ℓ_1-norm (a.k.a soft-thresholding operator) is component-wise separable and piecewise affine, hence strongly semi-smooth. By Moreau's decomposition, the proximal mapping of the ℓ_∞-norm is also strongly semi-smooth. The proximal operator of the nuclear norm, which encourages low-rank solutions, is strongly semi-smooth as a consequence of [11, Prop. 4.10].

To establish convergence, we impose the following assumption.

Assumption 1. *There exists a constant M_α such that $\|\alpha^k\|_1 \le M_\alpha$ for all $k \in \mathbb{N}_+$.*

This assumption is very common in the literature of AA and some effective solutions have been proposed to enfore it in practice. For example, one can monitor the condition number of the R matrix in the QR decomposition and drop the left-most column of the matrix if the number becomes too large [51], or one can add a Tikhonov regularization to the least squares as was done in [47]. The condition can also be imposed directly in the algorithm without changing the subsequent results. More specifically, if we detect that $\|\alpha^k\|_1$ is greater than M_α, we can set $\alpha^k = [0, \ldots, 1]^\top$, i.e., we simply perform a fixed-point iteration step.

The following lemma characterizes properties of the mapping g and its associated residual.

Lemma 3.1. *Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be μ-strongly convex and L-smooth and let $\gamma \in (0, 2/(\mu + L)]$. Define $\rho = \sqrt{1 - 2\gamma\mu L/(\mu + L)}$ and $F(y) := y - g(y)$ where g is defined as in* (15). *Let $y^\star$ be a fixed-point of g, i.e., $g(y^\star) = y^\star$. Then, it holds that*

$$\|g(x) - g(y)\|_2 \le \rho \|x - y\|_2, \quad \forall x, y \in \mathbb{R}^n, \tag{17}$$

and

$$(1 - \rho)\|y - y^\star\|_2 \le \|F(y)\|_2 \le (1 + \rho)\|y - y^\star\|_2, \quad \forall y \in \mathbb{R}^n. \tag{18}$$

Proof. We start by showing the contractivity of g defined in (15):

$$g(y) = \text{prox}_{\gamma h}(y) - \gamma \nabla f(\text{prox}_{\gamma h}(y)).$$

We have for any $x, y \in \mathbb{R}^n$ that

$$\begin{aligned}\|g(x) - g(y)\|_2^2 &= \left\|\text{prox}_{\gamma h}(x) - \text{prox}_{\gamma h}(y) - \gamma\left(\nabla f(\text{prox}_{\gamma h}(x)) - \nabla f(\text{prox}_{\gamma h}(y))\right)\right\|_2^2 \\ &= \left\|\text{prox}_{\gamma h}(x) - \text{prox}_{\gamma h}(y)\right\|_2^2 + \gamma^2 \left\|\nabla f(\text{prox}_{\gamma h}(x)) - \nabla f(\text{prox}_{\gamma h}(y))\right\|_2^2 \\ &\quad - 2\gamma \left\langle \text{prox}_{\gamma h}(x) - \text{prox}_{\gamma h}(y), \nabla f(\text{prox}_{\gamma h}(x)) - \nabla f(\text{prox}_{\gamma h}(y))\right\rangle \end{aligned} \tag{19}$$

Since f is μ-strongly convex and L-smooth, it follows from [32, Theorem 2.1.2] that

$$\begin{aligned}&\left\langle \text{prox}_{\gamma h}(x) - \text{prox}_{\gamma h}(y), \nabla f(\text{prox}_{\gamma h}(x)) - \nabla f(\text{prox}_{\gamma h}(y))\right\rangle \\ &\ge \frac{\mu L}{\mu + L}\left\|\text{prox}_{\gamma h}(x) - \text{prox}_{\gamma h}(y)\right\|_2^2 + \frac{1}{\mu + L}\left\|\nabla f(\text{prox}_{\gamma h}(x)) - \nabla f(\text{prox}_{\gamma h}(y))\right\|_2^2\end{aligned} \tag{20}$$

Plugging (20) into (19) yields

$$\begin{aligned}\|g(x)-g(y)\|_2^2 \le &\left(1-\frac{2\gamma\mu L}{\mu+L}\right)\left\|\mathrm{prox}_{\gamma h}(x)-\mathrm{prox}_{\gamma h}(y)\right\|_2^2 \\ &+\gamma\left(\gamma-\frac{2}{\mu+L}\right)\left\|\nabla f(\mathrm{prox}_{\gamma h}(x))-\nabla f(\mathrm{prox}_{\gamma h}(y))\right\|_2^2 .\end{aligned} \tag{21}$$

Since $\gamma \in (0, 2/(\mu+L))$, we can drop the last term in (21) and obtain

$$\|g(x)-g(y)\|_2^2 \le \left(1-\frac{2\gamma\mu L}{\mu+L}\right)\left\|\mathrm{prox}_{\gamma h}(x)-\mathrm{prox}_{\gamma h}(y)\right\|_2^2 ,$$

which by nonexpansiveness of proximal operators, can be further bounded as

$$\|g(x)-g(y)\|_2^2 \le \left(1-\frac{2\gamma\mu L}{\mu+L}\right)\|x-y\|_2^2 .$$

This implies that g is contractive with constant $\rho=\sqrt{1-2\gamma\mu L/(\mu+L)}$.

Next, we verify for every $y \in \mathbb{R}^n$ that

$$\|y-y^\star\|_2(1-\rho) \le \|F(y)\|_2 \le (1+\rho)\|y-y^\star\|_2 . \tag{22}$$

Since $g(y^\star)=y^\star$, $F(y^\star)=0$. By the contractivity of g, it holds for any $y\in\mathbb{R}^n$ that

$$\begin{aligned}\|F(y)-F(y^\star)\|_2 &= \|g(y)-y-g(y^\star)+y^\star\|_2 \\ &\le \|g(y)-g(y^\star)\|_2+\|y-y^\star\|_2 \\ &\le (1+\rho)\|y-y^\star\|_2 .\end{aligned}$$

Similarly, for the left-hand side of (22), we have

$$\begin{aligned}\|y-y^\star\|_2 &= \|g(y)-g(y^\star)+F(y)-F(y^\star)\|_2 \\ &\le \|g(y)-g(y^\star)\|_2+\|F(y)-F(y^\star)\|_2 \\ &\le \rho\|y-y^\star\|_2+\|F(y)-F(y^\star)\|_2 .\end{aligned}$$

□

Under Assumption 1, we can now quantify the convergence guarantee of AA-PGA.

Theorem 1. *Assume that f is μ-strongly convex and L-smooth, and that prox_h is strongly semi-smooth. Let $y^\star$ be a fixed-point of g and assume that y_0 is sufficiently close to $y^\star$. Let $\gamma \in (0, 2/(\mu+L)]$ and define $\rho=\sqrt{1-\gamma 2\mu L/(\mu+L)}$ and $\rho<\hat\rho<1$. Then, for any $m\in\mathbb{N}$ and $k\in\mathbb{N}$, the iterates $\{x_k\}$ and $\{y_k\}$ formed by AA-PGA satisfy:*

$$\|F(y_k)\|_2 \le \hat\rho^k\|F(y_0)\|_2 . \tag{23}$$

In addition, it holds for any $k\in\mathbb{N}$ that

$$\|x_k-x^\star\|_2 \le (1+\rho)/(1-\rho)\hat\rho^k\|y_0-y^\star\|_2 .$$

where $x^\star$ is an optimizer of (9).

Proof. Since $\text{prox}_h(\cdot)$ is strongly semi-smooth, and since strong semi-smoothness is closed under scalar multiplication, summation and composition, it follows that the mapping g is strongly semi-smooth. Thus, by Definition 3.2, there exists a ball $\mathcal{B}(y^\star, r)$ with $r > 0$ such that for any $x, y \in \mathcal{B}(y^\star, r)$ and $G \in \partial g(x)$ it holds that

$$\|g(x) - g(y) - G(x-y)\|_2 \leq c\,\|x - y\|_2^2,$$

for some constant c. Let $x = y^\star$ and $G^\star$ be any matrix in $\partial g(y^\star)$. We can then write $g(y)$ for any $y \in \mathcal{B}(y^\star, r)$ as

$$g(y) = y^\star + G^\star(y - y^\star) + \Delta(y), \tag{24}$$

where $\Delta(y)$ satisfies $\|\Delta(y)\|_2 \leq c\,\|y - y^\star\|_2^2$.

Suppose $y_0 \in \mathcal{B}(y^\star, r_0)$ with $r_0 \leq r$. We will establish the bound (23) by induction. It is clear that (23) holds for $k = 0$. Suppose the bound holds up to iteration k. We will show that it also holds for iteration $k+1$. For brevity, define

$$y_k^\alpha = \sum_{i=0}^{m_k} \alpha_i^k y_{k-i}.$$

We then decompose $F(y_{k+1})$ as follows

$$\|F(y_{k+1})\|_2 = \|g(y_{k+1}) - y_{k+1}\|_2 \leq \|g(y_{k+1}) - g(y_k^\alpha)\|_2 + \|g(y_k^\alpha) - y_{k+1}\|_2\,. \tag{25}$$

Recall that $y_{k+1} = \sum_{i=0}^{m_k} \alpha_i^k g(y_{k-i})$ and that g is ρ-contractive. Hence,

$$\|g(y_{k+1}) - g(y_k^\alpha)\|_2 \leq \rho\,\|y_{k+1} - y_k^\alpha\|_2 = \rho\left\|\sum_{i=0}^{m_k} \alpha_i^k F(y_{k-i})\right\|_2 \leq \rho\,\|F(y_k)\|_2\,, \tag{26}$$

where the last step follows from the definition of α_i^k. To bound the remaining term in (25), we first show that $y_k^\alpha \in \mathcal{B}(y^\star, r)$. We have

$$\begin{aligned} \|y_k^\alpha - y^\star\| \leq \sum_{i=0}^{m_k} |\alpha_i^k|\,\|y_{k-i} - y^\star\| &\overset{\text{(a)}}{\leq} \frac{1}{1-\rho}\sum_{i=0}^{m_k} |\alpha_i^k|\,\|F(y_{k-i})\| \\ &\overset{\text{(b)}}{\leq} \frac{M_\alpha}{1-\rho}\hat{\rho}^{k-m}\,\|F(y_0)\| \\ &\overset{\text{(c)}}{\leq} \frac{M_\alpha(1+\rho)}{1-\rho}\hat{\rho}^{k-m}\,\|y_0 - y^\star\|\,, \end{aligned} \tag{27}$$

where we have used (18) in (a) and (c), and (b) follows from the induction hypothesis and the fact that $k - i \geq k - m_k \geq k - m$ for $i \in \{0, \ldots, m_k\}$. Therefore, by reducing $r_0 = \|y_0 - y^\star\|_2$ if necessary, $y_k^\alpha \in \mathcal{B}(y^\star, r)$. The above derivation clearly indicates that $y_{k-i} \in \mathcal{B}(y^\star, r)$ for any $i \in \{0, \ldots, m_k\}$. As a consequence, we can write $g(y_{k-i})$ as

$$g(y_{k-i}) = y^\star + G^\star(y_{k-i} - y^\star) + \Delta(y_{k-i}),$$

which implies

$$y_{k+1} = \sum_{i=0}^{m_k} \alpha_i^k \left[y^\star + G^\star(y_{k-i} - y^\star) + \Delta(y_{k-i})\right], \tag{28}$$

where $G^\star$ is any matrix in $\partial g(y^\star)$.

Fix a $G^\star \in \partial g(y^\star)$. Since $y_k^\alpha \in \mathcal{B}(y^\star, r)$, by (24), it holds that

$$\begin{aligned} g(y_k^\alpha) &= y^\star + G^\star \sum_{i=0}^{m_k} \alpha_i^k (y_{k-i} - y^\star) + \Delta(y_k^\alpha) \\ &= \sum_{i=0}^{m_k} \alpha_i^k \left[y^\star + G^\star(y_{k-i} - y^\star) \right] + \Delta(y_k^\alpha) \\ &= y_{k+1} - \sum_{i=0}^{m_k} \alpha_i^k \Delta(y_{k-i}) + \Delta(y_k^\alpha), \end{aligned}$$

where $\|\Delta(y_k^\alpha)\|_2 \leq c \|y_k^\alpha - y^\star\|_2^2$ and the last step follows from (28). We deduce that

$$\begin{aligned} \|g(y_k^\alpha) - y_{k+1}\|_2 &= \left\| \sum_{i=0}^{m_k} \alpha_i^k \Delta(y_{k-i}) - \Delta(y_k^\alpha) \right\|_2 \\ &\leq \left\| \sum_{i=0}^{m_k} \alpha_i^k \Delta(y_{k-i}) \right\|_2 + \|\Delta(y_k^\alpha)\|_2 . \end{aligned} \tag{29}$$

Note from the inequality (b) in (27) that

$$\|y_k^\alpha - y^\star\|_2 \leq \frac{M_\alpha}{1-\rho} \hat{\rho}^{k-m} \|F(y_0)\|_2 ,$$

which yields

$$\begin{aligned} \|\Delta(y_k^\alpha)\|_2 \leq c \|y_k^\alpha - y^\star\|_2^2 &\leq c \|y_k^\alpha - y^\star\|_2 \frac{M_\alpha}{1-\rho} \hat{\rho}^{k-m} \|F(y_0)\|_2 \\ &\leq \frac{c M_\alpha^2 (1+\rho)}{(1-\rho)^2} \hat{\rho}^{k-m} \|y_0 - y^\star\|_2 \hat{\rho}^{k-m} \|F(y_0)\|_2 , \end{aligned} \tag{30}$$

where we have used (27) in the second inequality. For the first term in (29), we have

$$\begin{aligned} \|\Delta(y_{k-i})\|_2 \leq c \|y_{k-i} - y^\star\|_2^2 &\overset{\text{(a)}}{\leq} \frac{c}{1-\rho} \|y_{k-i} - y^\star\|_2 \|F(y_{k-i})\|_2 \\ &\overset{\text{(b)}}{\leq} \frac{c(1+\rho)}{(1-\rho)^2} \hat{\rho}^{k-i} \|y_0 - y^\star\|_2 \|F(y_{k-i})\|_2 \\ &\overset{\text{(c)}}{\leq} \frac{c(1+\rho)}{(1-\rho)^2} \hat{\rho}^{k-m} \|y_0 - y^\star\|_2 \hat{\rho}^{k-m} \|F(y_0)\|_2 , \end{aligned} \tag{31}$$

where (a) follows from (18), (b) follows from the induction hypothesis and (18), and (c) follows from the induction hypothesis and the fact that $k - i \geq k - m$ for $i \in \{0, \ldots, m_k\}$. Since $M_\alpha \geq 1$, (31) implies that

$$\begin{aligned} \left\| \sum_{i=0}^{m_k} \alpha_i^k \Delta(y_{k-i}) \right\|_2 &\leq \sum_{i=0}^{m_k} |\alpha_i^k| \, \|\Delta(y_{k-i})\|_2 \\ &\leq \frac{c M_\alpha (1+\rho)}{(1-\rho)^2} \hat{\rho}^{k-m} \|y_0 - y^\star\|_2 \hat{\rho}^{k-m} \|F(y_0)\|_2 \\ &\leq \frac{c M_\alpha^2 (1+\rho)}{(1-\rho)^2} \hat{\rho}^{k-m} \|y_0 - y^\star\|_2 \hat{\rho}^{k-m} \|F(y_0)\|_2 . \end{aligned} \tag{32}$$

Choosing $r_0 = \|y_0 - y^\star\|_2$ sufficiently small such that

$$\frac{cM_\alpha^2(1+\rho)}{(1-\rho)^2}\hat{\rho}^{k-m}\|y_0 - y^\star\|_2 \le \frac{\hat{\rho}^{m+1}}{2}\left(1 - \frac{\rho}{\hat{\rho}}\right), \tag{33}$$

and plugging (30) and (32) into (29) yields

$$\|g(y_k^\alpha) - y_{k+1}\|_2 \le \left(1 - \frac{\rho}{\hat{\rho}}\right)\hat{\rho}^{k+1}\|F(y_0)\|_2. \tag{34}$$

Combining (25), (26), and (34), together with the induction hypothesis, we arrive at

$$\|F(y_{k+1})\| \le \rho\|F(y_k)\| + \left(1 - \frac{\rho}{\hat{\rho}}\right)\hat{\rho}^{k+1}\|F(y_0)\|_2 \le \hat{\rho}^{k+1}\|F(y_0)\|_2.$$

Finally, by the nonexpansiveness of the proximal operator,

$$\|x_k - x^\star\|_2 \le \|y_k - y^\star\|_2 \le \frac{\hat{\rho}^k\|F(y_0)\|_2}{1-\rho} \le \frac{1+\rho}{1-\rho}\hat{\rho}^k\|y_0 - y^\star\|_2,$$

which completes the proof. □

The theorem implies that when initialized near the optimal solution, even in the worst case, the use of multiple past iterates to construct a new update in AA will not slow down the convergence of the original PGA method, no matter how we choose $m \in \{0, 1, \ldots\}$. In most cases, near the solution, we would expect AA-PGA to enjoy the strong adaptive rate in (7) even for a small value of m. Therefore, we can see AA as interpolating between the two convergence rates corresponding to $m = 0$ (PGA) and $m = \infty$ (full-memory AA). Whether or not AA can attain a stronger convergence rate guarantees than PGA for finite m is still an open question, even with smooth and linear mappings.

4 Guarded Anderson acceleration

We have shown that when started from a point close to the optimal solution, AA-PGA is convergent under mild conditions. A natural question, which has also recently been raised in [19], is whether AA-PGA converges globally. We show that the answer is negative even when the problem has no constraint and the objective function is smooth. In this case, AA-PGA reduces to AA-GD, and hence the result is also valid for the AA methods in [51, 47]. To that end, we construct a one-dimensional smooth and strongly convex function and show analytically that AA will not converge to the optimum but get stuck in a periodic orbit. Concretely, consider the function whose gradient is given by

$$\nabla f(x) = \begin{cases} \frac{x}{10} - 24.9 & \text{if} \quad x < -1, \\ 25x & \text{if} \quad -1 \le x < 1, \\ \frac{x}{10} + 24.9 & \text{if} \quad x \ge 1, \end{cases} \tag{35}$$

which is strongly convex with $\mu = 1/10$ and smooth with $L = 25$.

We consider AA-GD with $m = 1$ for which the iterates can be expressed explicitly (detailed below) on the form of a nonlinear difference equation:

$$\begin{bmatrix} x_{k+1} \\ x_k \end{bmatrix} = \begin{bmatrix} \frac{\nabla f(x_{k-1})}{\nabla f(x_{k-1}) - \nabla f(x_k)} & \frac{-\nabla f(x_k)}{\nabla f(x_{k-1}) - \nabla f(x_k)} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix}. \tag{36}$$

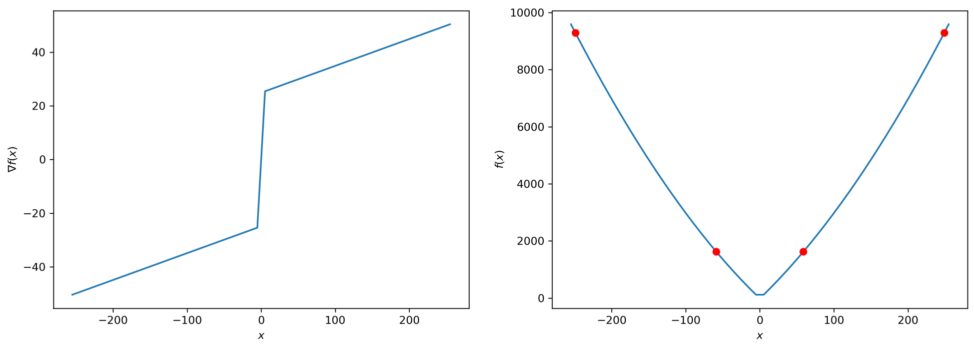


Figure 2: The graph of the gradient defined in (35) and the corresponding function. The circles in the right plot indicate the four limit points shown in Proposition 1.

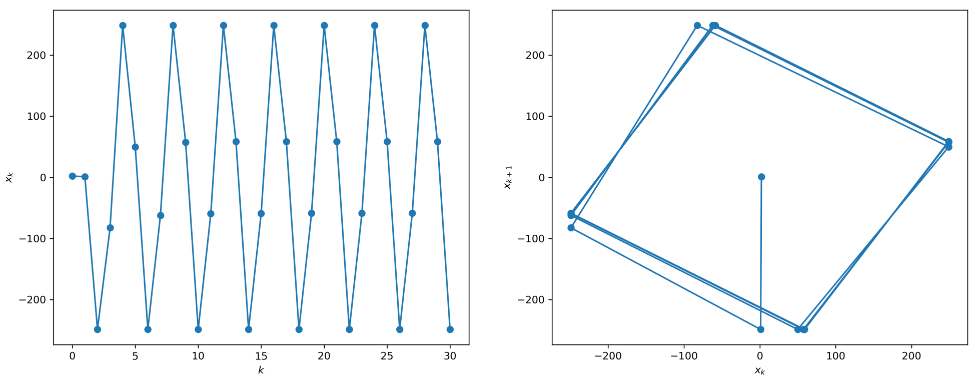


Figure 3: Left. Iterates of the AA-GD method when minimizing $f(x)$ defined in (35) with $x_0 = 2.1$; Right. The phase plane of the corresponding concatenated system (36). The iterates eventually converge to a periodic orbit of length 4.

A trajectory of this system started at $x_0 = 2.1$ is depicted in Figure 3 indicating that AA-GD converges to a periodic orbit instead of the origin. More formally, one can show the following.

Proposition 1. *Let f be the function defined in* (35)*. Suppose that the AA-GD method is applied to minimize f with the history parameter $m = 1$ and the step size $\gamma = 1/L$. Then, for any initial point $x_0 \in [2.01, 246.98]$, the iterates generated by AA-GD satisfy:*

$$x_{4n+3} \to -249(\sqrt{5}-2), \quad x_{4n+4} = 249, \quad x_{4n+5} \to 249(\sqrt{5}-2), \quad x_{4n+6} = -249, \ n = 0, 1, \ldots$$

The proposition indicates that in general, to achieve global convergence, it is necessary to stabilize the AA method. Recall that each iteration of AA-PGA consists of one original PGA step followed by an AA (affine combination) step. Thus, one natural strategy for stabilization would be to compare the objective value produced by the AA step with that of the PGA one and select the one with lower value as the next iterate. Since PGA converges globally, this ensures that AA-PGA converges at least as fast as PGA. However, doing so can be costly since one needs two function evaluations per step. Indeed, only the descent condition below is needed to achieve the same convergence rate as PGA:

$$f(x_{k+1}) \leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{1}{2\gamma} \|x_{k+1} - x_k\|_2^2 \leq f(x_k) - \frac{\gamma}{2} \|\nabla f(x_k)\|_2^2 . \quad (37)$$

Algorithm 3 Guarded AA-PGA

```
Input: $y_0 \in \mathrm{dom}\, h$, $x_0 = \mathrm{prox}_{\gamma h}(y_0)$, $m \geq 0$
 1: $y_1 \leftarrow x_0 - \gamma \nabla f(x_0)$, $x_1 \leftarrow \mathrm{prox}_{\gamma h}(y_1)$, $g_0 \leftarrow y_1$, and $r_0 = g_0 - y_0$
 2: for $k = 1, \dots, K-1$ do
 3:     $m_k \leftarrow \min(m, k)$
 4:     $g_k \leftarrow x_k - \gamma \nabla f(x_k)$ and $r_k \leftarrow g_k - y_k$
 5:     $R_k \leftarrow [r_k, \dots, r_{k-m_k}]$
 6:     $\alpha^k \leftarrow \mathrm{argmin}_{\alpha^\top \mathbf{1} = 1} \|R_k \alpha\|$
 7:     $y_{\text{ext}} \leftarrow \sum_{i=0}^{m_k} \alpha_i^k g_{k-i}$
 8:     $x_{\text{test}} \leftarrow \mathrm{prox}_{\gamma h}(y_{\text{ext}})$
 9:     if $f(x_{\text{test}}) \leq f(x_k) - \frac{\gamma}{2} \|\nabla f(x_k)\|_2^2$ then
10:         $x_{k+1} = x_{\text{test}}$
11:         $y_{k+1} = y_{\text{ext}}$
12:         $f(x_{k+1}) = f(x_{\text{test}})$
13:     else
14:         $x_{k+1} = \mathrm{prox}_{\gamma h}(g_k)$
15:         $y_{k+1} = g_k$
16:     end if
17: end for
Output: $x_K$
```

This suggests an alternative way for stabilization which is to compare the objective value of the AA step with the right-hand side of (37). The AA step is then accepted if sufficient descent was made, the PGA step is chosen otherwise. This allows to reuse the function values more efficiently. In particular, if the AA step is selected, only one function evaluation is needed. Moreover, in many applications, function values can be computed at a very small additional cost by reusing information readily available from gradient evaluations. Putting everything together, we arrive at Algorithm 3 that admits the global convergence rate of PGA with the potential for local adaptation and acceleration.

Proposition 2 (Global convergence)**.** *Let f be a μ-strongly convex and L-smooth function and let $\gamma \in (0, 2/(\mu + L)]$. Then, for any $k \in \mathbb{N}$, the iterates generated by Algorithm 3 satisfy*

$$\|x_k - x^\star\|_2^2 \leq O\left(1 - \frac{\gamma \mu L}{\mu + L}\right)^k \|x_0 - x^\star\|_2^2 . \quad (38)$$

The proof of this result is straightforward (see, e.g., [32, 3]), and follows directly from the descent condition (37) and a standard strong convexity inequality. Hence, we omit it here.

5 Extension to Bregman proximal gradient methods

Consider optimization problems of the form

$$\underset{x \in \mathcal{D}}{\text{minimize}} \; f(x) + h(x), \quad (39)$$

where $\mathcal{D} \subseteq \mathbb{R}^n$ is a closed convex set with nonempty interior. The formulation (39) often provides a more flexible way to handle the constraints, which are usually encoded by h in (9). This model is very rich and led to several recent advances in algorithmic developments of first-order methods. Bregman proximal gradient (BPG) is a general and powerful tool for solving (39) thanks to its ability to exploit the underlying geometry of the problem. The mirror descent method [31, 4] is a well-known instance of BPG when $h(x) = \mathrm{I}_{\mathcal{C}}(x)$ for some closed convex set $\mathcal{C} \subseteq \mathcal{D}$. Some more recent instances of BPG include the NoLips algorithm [2] and its accelerated version analysed in [22]. The number of applications of the BPG framework are growing rapidly [6, 13, 26].

The BPG method fits the geometry of the problem at hand, which is typically governed by the constraints and/or the objective, all-in-one by means of a kernel function. Popular examples include the energy function $\varphi(x) = (1/2)\|x\|_2^2$; the Shannon entropy $\varphi(x) = \sum_{i=1}^n x_i \log x_i$, $\operatorname{dom}\varphi = \mathbb{R}_+^n$ with $(0 \log 0 = 0)$; the Burg entropy $\varphi(x) = -\sum_{i=1}^n \log x_i$, $\operatorname{dom}\varphi = \mathbb{R}_{++}^n$; the Fermi-Dirac entropy $\varphi(x) = \sum_{i=1}^n (x_i \log x_i + (1-x_i)\log(1-x_i))$, $\operatorname{dom}\varphi = [0,1]^n$; the Hellinger entropy $\varphi(x) = -\sum_{i=1}^n \sqrt{1-x_i^2}$, $\operatorname{dom}\varphi = [-1,1]^n$; and the polynomial function $\varphi(x) = \frac{\alpha}{2}\|x\|_2^2 + \frac{1}{4}\|x\|_2^4$, $\alpha \geq 0$.

We impose the following assumption in this section.

Assumption 2. *The set* $\overline{\operatorname{dom}}\varphi = \mathcal{D}$ *is convex and the following conditions hold:*

1. $\varphi : \mathbb{R}^n \to (-\infty, +\infty]$ *is of Legendre type and its conjugate* φ^* *satisfies* $\operatorname{dom}\nabla\varphi^* = \mathbb{R}^n$.
2. $f : \mathbb{R}^n \to (-\infty, +\infty]$ *is proper closed convex and differentiable on* $\operatorname{int}\operatorname{dom}\varphi$.
3. $h : \mathbb{R}^n \to (-\infty, +\infty]$ *is proper closed convex and* $\operatorname{dom} h \cap \operatorname{int}\operatorname{dom}\varphi \neq \emptyset$.

When φ is Legendre, its gradient $\nabla\varphi$ is a bijection from $\operatorname{int}\operatorname{dom}\varphi$ to $\operatorname{int}\operatorname{dom}\varphi^*$ while $\nabla\varphi^*$ is a bijection from $\operatorname{int}\operatorname{dom}\varphi^*$ to $\operatorname{int}\operatorname{dom}\varphi$, i.e., $(\nabla\varphi)^{-1} = \nabla\varphi^*$ [43, Chapter 26]. Note that in all the above examples, φ is Legendre. Moreover, except from the Burg entropy, all the others share the useful property $\operatorname{dom}\nabla\varphi^* = \mathbb{R}^n$, which is critical for the development of our AA scheme.

The Bregman distance associated with φ is the function $D_\varphi : \operatorname{dom}\varphi \times \operatorname{int}\operatorname{dom}\varphi \to \mathbb{R}$ given by

$$D_\varphi(x,y) = \varphi(x) - \varphi(y) - \langle \nabla\varphi(y), x-y \rangle .$$

At the core of the BPG method is the Bregman proximal operator that generalizes the conventional one and is defined as [9]:

$$\operatorname{prox}_h^\varphi(y) = \operatorname*{argmin}_{x\in\mathbb{R}^n} \{h(x) + D_\varphi(x,y)\}, \quad y \in \operatorname{int}\operatorname{dom}\varphi. \tag{40}$$

BPG starts with some $x_0 \in \operatorname{int}\operatorname{dom}\varphi$ and performs the following operator at each iteration:

$$x_{k+1} = T_\gamma(x_k) := \operatorname*{argmin}_{x\in\mathbb{R}^n} \big\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \gamma^{-1} D_\varphi(x, x_k) + h(x) \big\}.$$

Assumptions 2 ensures that BPG iterates are well-defined and $x_k \in \operatorname{int}\operatorname{dom}\varphi$ for all k [2, Lemma 2]. Further simplification the update formula yields

$$x_{k+1} = \operatorname*{argmin}_{x\in\mathbb{R}^n} \{\langle \gamma\nabla f(x_k) - \nabla\varphi(x_k), x\rangle + \varphi(x) + h(x)\} .$$

Using the optimality condition and the fact that $(\nabla\varphi)^{-1} = \nabla\varphi^*$ yield

$$0 \in \gamma\partial h(x_{k+1}) + \nabla\varphi(x_{k+1}) - \nabla\varphi(\nabla\varphi^*\left(\nabla\varphi(x_k) - \gamma\nabla f(x_k)\right)). \tag{41}$$

Comparing (41) with the optimality condition of (40), we obtain an equivalent update rule for BPG:

$$x_{k+1} = \operatorname{prox}_{\gamma h}^{\varphi}\left(\nabla\varphi^*\left(\nabla\varphi(x_k) - \gamma\nabla f(x_k)\right)\right).$$

Note that when φ is the energy function, $\nabla\varphi$ and $\nabla\varphi^*$ are the identity map and we recover the PGA method. To apply AA, we further express the BPG iterations on the form

$$y_{k+1} = \nabla\varphi(x_k) - \gamma\nabla f(x_k) \quad \text{and} \quad x_{k+1} = \operatorname{prox}_{\gamma h}^{\varphi}\left(\nabla\varphi^*\left(y_{k+1}\right)\right). \tag{42}$$

In words, the mirror map $\nabla\varphi$ maps x_k from the primal space to a dual one, where the gradients live. A gradient step is then taken in the dual space to obtain y_{k+1}. Next, y_{k+1} is transferred back to the primal space by the inverse map $\nabla\varphi^*$. Finally, the Bregman proximal operator is performed in the primal space to produce x_{k+1}.

Our strategy is to extrapolate the sequence $\{y_k\}$. Note that this sequence can be seen as the fixed-point iteration of

$$g(y) = \nabla\varphi(\operatorname{prox}_{\gamma h}^{\varphi} \circ \nabla\varphi^*(y)) - \gamma\nabla f(\operatorname{prox}_{\gamma h}^{\varphi} \circ \nabla\varphi^*(y)).$$

The AA scheme applied to this mapping (called AA-BPG) has a simple and elegant interpretation. Concretely, instead of accelerating the primal sequence, which is restricted to the constraint set, it extrapolates a sequence in the dual space, avoiding feasibility issues since $\nabla\varphi^*$ has full domain. To gain some intuition, we first recall the following useful property of Legendre functions:

$$D_{\varphi}\left(\nabla\varphi^*(y), \nabla\varphi^*(y')\right) = D_{\varphi^*}\left(y', y\right) \quad \forall y, y' \in \operatorname{int}\operatorname{dom}\varphi^*.$$

Assume that g has a fixed-point $y^\star$ and $\{y_k\}$ generated by AA-BPG is converging to $y^\star$. Let $\nabla\varphi^*(y_k)$ and $\nabla\varphi^*(y^\star)$ be the images of y_k and $y^\star$ on the primal space, then it holds that

$$D_{\varphi^*}\left(y^\star, y_k\right) = D_{\varphi}\left(\nabla\varphi^*(y_k), \nabla\varphi^*(y^\star)\right).$$

Applying the Bregman operator to the two images will give us x_k and $x^\star$, respectively. Since Bregman proximal operators possess certain nonexpansiveness property akin to their Euclidean counterpart [8, 14], it is thus reasonable to expect that $D_{\varphi}\left(x_k, x^\star\right)$ is well approximated by $D_{\varphi^*}\left(y^\star, y_k\right)$; for example, when $\operatorname{dom} h \subseteq \operatorname{int}\operatorname{dom}\varphi$, it is shown in [8] that $D_{\varphi}\left(x_k, x^\star\right) \leq D_{\varphi}\left(\nabla\varphi^*(y_k), \nabla\varphi^*(y^\star)\right)$. Moreover, $y_k = y^\star$ implies $x_k = x^\star$. Therefore, if AA can speed-up the convergence of $\{y_k\}$, one can achieve similar acceleration for $\{x_k\}$.

In the above discussion, we implicitly assumed that $x^\star \in \operatorname{int}\operatorname{dom}\varphi$. However, if $x^\star$ happens to be on the boundary of $\operatorname{dom}\varphi$, the mirror map $\nabla\varphi$ at $x^\star$ does not exist. One can then no longer express $x^\star$ as a fixed-point of some mapping involving $\nabla\varphi$. This makes it very hard to derive general theoretical guarantees for BPG since essentially all the current proofs of AA are heavily based on $g(x^\star)$. Therefore, a new proof technique that goes beyond linearization of g around $x^\star$ is needed, which we leave as a topic for future research. Nonetheless, since each iteration of AA-BPG consists of one BPG step, $T_\gamma(x_k)$, one can always compare the progress made by the AA step with the BPG one as was done in AA-PGA. A counterpart of the sufficient descent condition (37) that ensures the global convergence of BPG is [2]:

$$f(x_{k+1}) \leq f(x_k) + \langle\nabla f(x_k), T_\gamma(x_k) - x_k\rangle + \gamma^{-1}D_{\varphi}\left(T_\gamma(x_k), x_k\right).$$

Thus, a similar policy for stabilization as in AA-PGA will retains the convergence rate of BPG. The final AA-BPG algorithm is reported in Algorithm 4.

Algorithm 4 Guared AA-BPG

```
Input: $y_0 \in \operatorname{int}\operatorname{dom}\varphi^*$, $x_0 = \operatorname{prox}_{\gamma h}^{\varphi}\left(\nabla\varphi^*\left(y_0\right)\right)$, $m \geq 0$
 1: $y_1 \leftarrow \nabla\varphi(x_0) - \gamma\nabla f(x_0)$, $x_1 \leftarrow x_0 = \operatorname{prox}_{\gamma h}^{\varphi}\left(\nabla\varphi^*\left(y_0\right)\right)$, $g_0 \leftarrow y_1$, and $r_0 = g_0 - y_0$
 2: for $k = 1, \ldots, K-1$ do
 3:     $m_k \leftarrow \min(m, k)$
 4:     $g_k \leftarrow \nabla\varphi(x_k) - \gamma\nabla f(x_k)$ and $r_k \leftarrow g_k - y_k$
 5:     $R_k \leftarrow [r_k, \ldots, r_{k-m_k}]$
 6:     $\alpha^k \leftarrow \operatorname{argmin}_{\alpha^\top \mathbf{1}=1} \|R_k\alpha\|$
 7:     $y_{\text{ext}} \leftarrow \sum_{i=0}^{m_k} \alpha_i^k g_{k-i}$ and $x_{\text{test}} \leftarrow \operatorname{prox}_{\gamma h}^{\varphi}\left(\nabla\varphi^*\left(y_{\text{ext}}\right)\right)$
 8:     $x_{\text{BPG}} = \operatorname{prox}_{\gamma h}^{\varphi}\left(\nabla\varphi^*\left(g_k\right)\right)$
 9:     if $f(x_{\text{test}}) \leq f(x_k) + \langle\nabla f(x_k), x_{\text{BPG}} - x_k\rangle + \gamma^{-1}D_\varphi\left(x_{\text{BPG}}, x_k\right)$ then
10:         $x_{k+1} = x_{\text{test}}$
11:         $y_{k+1} = y_{\text{ext}}$
12:         $f(x_{k+1}) = f(x_{\text{test}})$
13:     else
14:         $x_{k+1} = x_{\text{BPG}}$
15:         $y_{k+1} = g_k$
16:     end if
17: end for
Output: $x_K$
```

6 Numerical Experiments

We will now illustrate the performance of (guarded) AA-PGA and AA-BPG on several constrained optimization problems with important applications in signal processing and machine learning. All the experiments are implemented in Python and run on a laptop with four 2.4 GHz cores and 16 GB of RAM, running Ubuntu 16.04 LTS.

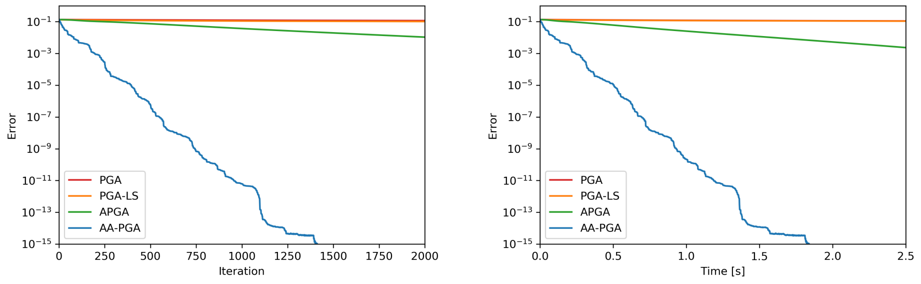
For AA-PGA, we compare it with PGA, PGA with adaptive line search (PGA-LS), and accelerated PGA (APGA) [3]. For AA-BPG, we compare AA-BPG with BPG, accelerated BPG (ABPG), ABPG with adaptive line search (ABPG-g), and restarted ABPG (ABPG-Restart) [22]. For the AA schemes, we use $m = 5$ in all plots and simply add a Tikhonov regularization of $10^{-10}\|R_k\|_2^2$ to (3) to avoid singularity, as was done in [46], without any tunning. For each experiment, we plot the errors, defined as $f(x_k) - f(x^\star)$, versus the number of iterations and wall-clock runtime. We have picked a few real-world data sets, which are known to be very ill-conditioned, and hence challenging for any first order methods.[1] All methods are initialized at $x_0 = \mathbf{0}$ unless otherwise stated.

6.1 Constrained logistic regression

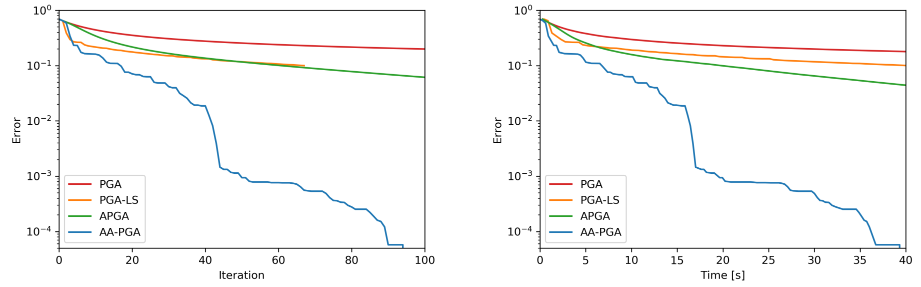
We start our experiments with the logistic regression with bounded constraint:

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \frac{1}{M}\sum_{i=1}^{M}\log(1+\exp(-y_i a_i^\top x)) + \mu\|x\|_2^2 \\ &\text{subject to}\ \|x\|_\infty \leq 1, \end{aligned}$$

[1]The data sets Madelon and Gisette are downloaded from: http://archive.ics.uci.edu/ml/datasets. The data sets Cina0 and Sido0 are downloaded from: http://www.causality.inf.ethz.ch



(a) Madelon: $\mu = 10$, $\kappa = 3 \times 10^6$



(b) Gisette: $\mu = 10$, $\kappa = 3.4 \times 10^6$

Figure 4: Constrained logistic regression on the Madelon and the Gisette data sets.

where $a_i \in \mathbb{R}^n$ are training samples and $y_i \in \{-1, 1\}$ are the corresponding labels. We set $\gamma = 1/L$, where $L = \|A\|_2^2 / 4M$ with $A = [a_1, \ldots, a_M]$.

Figures 4 and 5 show the performance of AA-PGA and other selected algorithms on four different data sets. As can be seen, AA consistently and dramatically improves the performance of standard first order methods both in number of iterations and wall-clock time. Since these data sets are very ill-conditioned, standard first order methods make very little progress, while AA can quickly find a high accuracy approximate solution. This once again demonstrates the great benefit of local adaptation and acceleration as previously seen in unconstrained quadratic problems (see., Figure 1). In most cases, the convergence rate is linear confirming our prediction. The result also highlights the importance of the guard step in Algorithm 3. Specifically, in some hard instances such as the one shown in Fig. 5(a), the iterates alternate between periods with big jumps due to AA steps, which often significantly reduce the objective, and slowly converging regimes governed by the PGA steps. The later steps help to guide the iterates through a tough regime until AA steps take over and make big improvement.

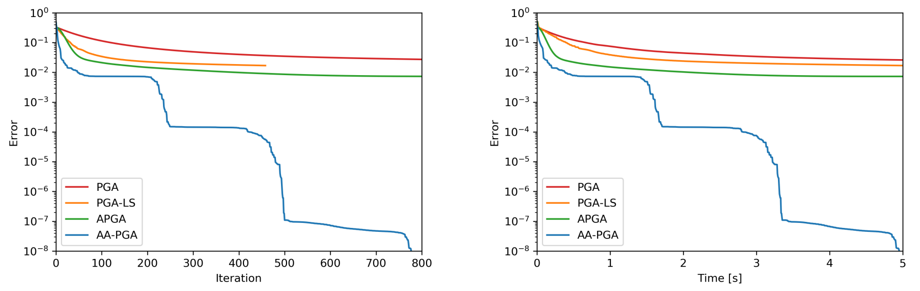
6.2 Nonnegative least squares

Next, we consider the nonnegative least squares problem:

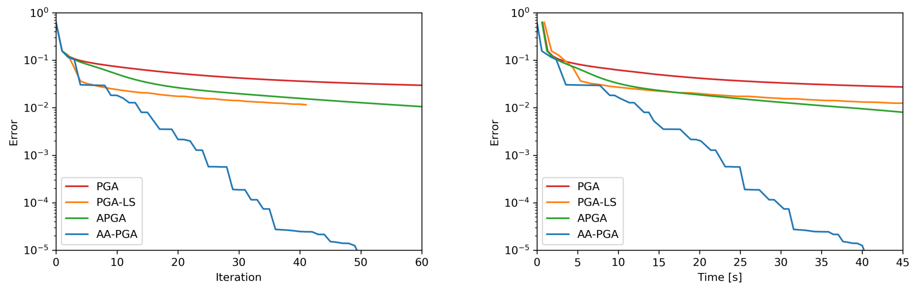
$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; \frac{1}{2M} \|Ax - b\|_2^2 + \mu \|x\|_2^2 \quad \text{subject to } \; x \geq 0,$$

which is a core step in many nonnegative matrix factorization algorithms. We set $\gamma = 1/L$, where $L = \|A\|_2^2 / M$.

Similarly to the previous problem, AA offers significant acceleration and often achieves several orders of magnitude speed-up over popular first order methods. Interestingly, in Fig. 6(a),



(a) Cina0: $\mu = 0.1$, $\kappa = 1.2 \times 10^7$



(b) Sido0: $\mu = 10^{-2}$, $\kappa = 3.7 \times 10^3$

Figure 5: Constrained logistic regression on the Cina0 and Sido0 data sets.

AA seems to identify the solution in finite time. This could be the case where the optimal solution lies in the subspace spanned by the past iterates.

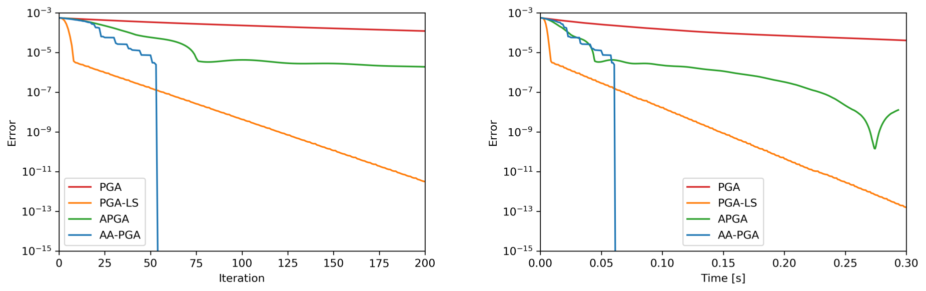
6.3 Relative-entropy nonnegative regression

The task is to reconstruct the signal $x \in \mathbb{R}_+^n$ by solving

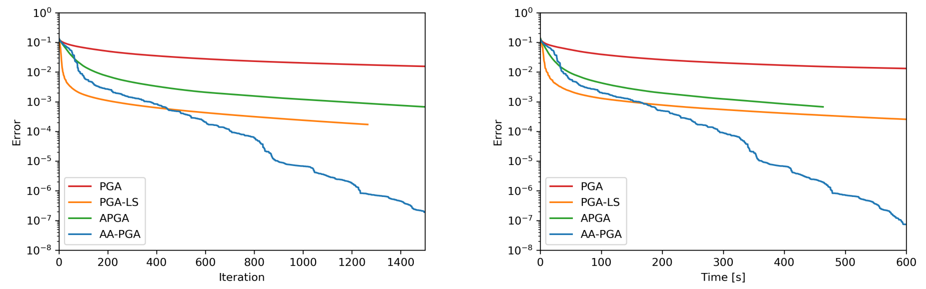
$$\underset{x}{\text{minimize}}\, D_{\text{KL}}(Ax, b) + \lambda \|x\|_1 \quad \text{subject to } x \geq 0,$$

where $A \in \mathbb{R}_+^{m \times n}$ is given nonnegative observation matrix and $b \in \mathbb{R}_{++}^m$ is a noisy measurement vector. We adapt the family of BPG methods with $\mathcal{D} = \mathbb{R}_+^n$, the Shannon entropy as the kernel φ, $f(x) = D_{\text{KL}}(Ax, b)$, and $h(x) = \lambda \|x\|_1$ with $\lambda = 0.001$. It is shown in [2] that f is L-smooth relative to φ with constant $L = \max_{1 \leq i \leq n} \|a_i\|_1$. We follow [22] and generate two problem instances with A and b having entries uniformly distributed over the interval $[0, 1]$. All methods are initialized at $x_0 = \mathbf{1}$.

Figure 8(a) shows the suboptimality for a randomly generated instance of the relative-entropy nonnegative regression problem with $m = 100$ and $n = 1000$. This instance is often referred as the easy case, and BPG converges linearly. Figure 8(b) shows similar results for the hard instance with $m = 1000$ and $n = 100$, where the BPG method converges sublinearly. In both cases, AA-BPG achieves the fastest convergence and significantly outperforms the others. Interestingly, AA-BPG is able to achieve linear convergence even in the hard case, which shows a clear evidence that our method adapts to the local strong convexity of the objective. This ability is observed consistently in all the problems and data sets we have considered, and confirms our theoretical predictions.

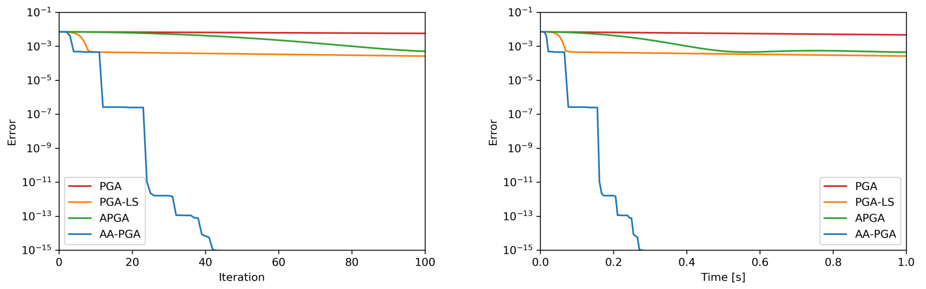


(a) Madelon: $\mu = 0.1$, $\kappa = 1.2 \times 10^9$

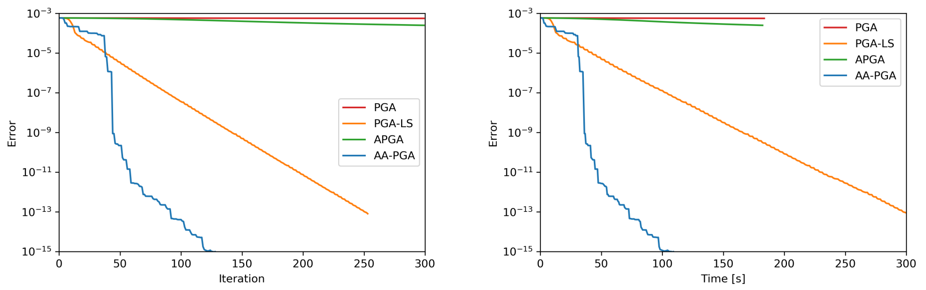


(b) Gisette: $\mu = 10$, $\kappa = 1.36 \times 10^7$

Figure 6: Nonnegative least-squares on the Madelon and Gisette data sets

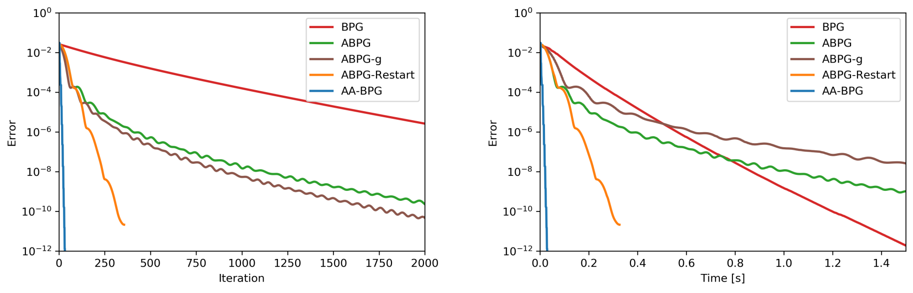


(a) Cina0: $\mu = 10$, $\kappa = 4.8 \times 10^5$

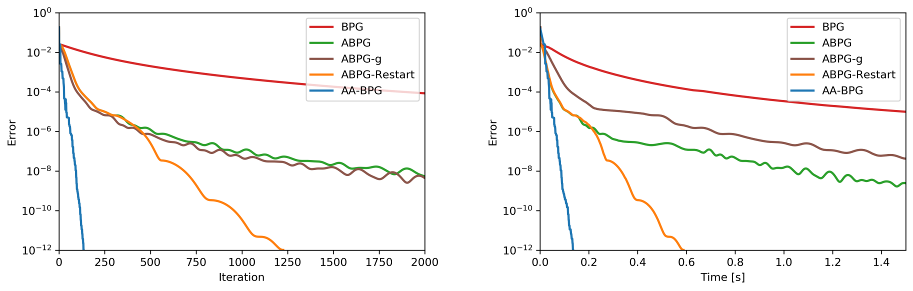


(b) Sido0: $\mu = 0.1$, $\kappa = 1.48 \times 10^6$

Figure 7: Nonnegative least-squares on the Cina0 and Sido0 data sets.



(a) $(m, n) = (100, 1000)$



(b) $(m, n) = (1000, 100)$

Figure 8: Relative-entropy nonnegative regression on two random problem instances.

7 Conclusion

We adapted Anderson acceleration to proximal gradient methods, retaining their global (worst-case) convergence guarantees while adding the potential for local adaption and acceleration. Key innovations include theoretical convergence guarantees for non-smooth mappings, techniques for avoiding potential infeasibilities, and stabilized algorithms with global convergence rate guarantees and strong practical performance. We also proposed an application of AA to non-Euclidean geometry. Given that AA can be applied to general fixed-point computations, the current literature has just scratched the surface of potential uses of AA in optimization. With its simplicity and evident promise, we feel that AA merits much further study.

A Proof of Proposition 1

We start by recalling the following useful result. For $a, b \in \mathbb{R}$ satisfying $a \neq b$, the solution to the minimization problem

$$\begin{array}{ll}\underset{\alpha_0,\alpha_1\in\mathbb{R}}{\text{minimize}} & (\alpha_0 a + \alpha_1 b)^2\\ \text{subject to} & \alpha_0 + \alpha_1 = 1,\end{array}$$

is given by

$$\alpha_0 = \frac{b}{b-a} \quad \text{and} \quad \alpha_1 = \frac{-a}{b-a}. \tag{43}$$

Recall also that the AA-GD method is the application of Algorithm 1 to the mapping $g(x) = x - \gamma\nabla f(x)$. Since $m = 1$, the k-th subproblem ($k \geq 1$) in Step 5 of Algorithm 1 boils down to computing

$$\alpha^k = \underset{\alpha_0+\alpha_1=1}{\text{argmin}} \left(\alpha_0 \nabla f(x_k) + \alpha_1 \nabla f(x_{k-1})\right)^2,$$

which together with (43) imply that

$$\alpha_0^k = \frac{\nabla f(x_{k-1})}{\nabla f(x_{k-1}) - \nabla f(x_k)} \quad \text{and} \quad \alpha_1^k = \frac{-\nabla f(x_k)}{\nabla f(x_{k-1}) - \nabla f(x_k)}.$$

Consequently, we can explicitly compute the next iterate defined in Step 6 of Algorithm 1 as

$$x_{k+1} = \alpha_0^k g(x_k) + \alpha_1^k g(x_{k-1}) = \frac{\nabla f(x_{k-1})}{\nabla f(x_{k-1}) - \nabla f(x_k)} x_k - \frac{\nabla f(x_k)}{\nabla f(x_{k-1}) - \nabla f(x_k)} x_{k-1}. \quad (44)$$

By the construction of $\nabla f(x)$ and (44), it follows that whenever x_k and x_{k-1} belong to the interval $[1, +\infty)$, the next iterate x_{k+1} will take the value -249. Similarly, if x_k and x_{k-1} belong to the interval $(-\infty, -1]$, then $x_{k+1} = +249$. This motivates us to select the initial interval so that some subsequnece of $\{x_k\}$ will always take the value $+249$ or -249, and hence never converge to the origin. To do so, let us examine the pattern of the first few iterates.

First, let $x_0, x_1 \in [1, +\infty)$ so that $x_2 = -249$. Given x_1 and x_2, it is easy to verify that

$$x_3 = \frac{249(x_1 + x_2)}{x_1 - x_2 + 498} = \frac{249(x_1 - 249)}{x_1 + 747}.$$

Since $x_2 < -1$, if we ensure that $x_3 \le -1$, we will have $x_4 = +249$. Note that for $x_1 \ge 1$, the right-hand-side of the preceding equation is an increasing function of x_1, therefore $x_3 < -1$ when $x_1 \le 245$. Also, since $x_1 \ge 1$, we have $x_3 > -83$. In summary, for $x_1 \in [1, 245]$, we have $x_3 \in (-83, -1)$ and $x_4 = +249$. A similar calculation yields

$$x_5 = \frac{-249(x_3 + x_4)}{x_3 - x_4 - 498} = \frac{-249(x_3 + 249)}{x_3 - 747}.$$

Similarly, for $x_3 \in (-83, -1)$, x_5 is an increasing function of x_3, therefore $x_5 \in (49.8, 83)$. Now, since $x_4, x_5 \in [1, \infty)$, $x_6 = -249$. The process is now repeated with x_1 replaced by x_5, x_2 replaced by x_6, and so on. Note that since $x_5 \in (49.8, 83) \subset [1, 245]$, all the above results are still valid and can be summarized as:

$$x_{4n+3} \in (-83, -1), \quad x_{4n+4} = +249, \quad x_{4n+5} \in [1, 245], \quad x_{4n+6} = -249, \quad \text{for} \quad n = 0, 1, 2, \ldots,$$

which implies that AA-GD will never converges to the optimal solution.

Indeed, it can be shown that all the four subsequences above will eventually converge. Under our initial condition, for $n = 0, 1, 2, \ldots$, the iterates x_{4n+3} and x_{4n+5} have the forms

$$x_{4n+3} = \frac{249(x_{4n+1} - 249)}{x_{4n+1} + 747}$$
$$x_{4n+5} = \frac{-249(x_{4n+3} + 249)}{x_{4n+3} - 747}.$$

Thus, we can find a transformation from x_{4n+1} to x_{4n+5} as

$$x_{4n+5} = \frac{249(x_{4n+1} + 249)}{x_{4n+1} + 1245}.$$

Define $y_n = x_{4n+1}$, then the previous equation can be seen as a fixed-point iteration $y_{n+1} = G(y_n)$ with $G(y) := 249(y + 249)/(y + 1245)$. It is easy to verify that for $y \in [1, 245]$, the mapping G is contractive, and hence $\{y_n\}$ converges to the unique fixed-point of G in $[1, 245]$, which is $+249(\sqrt{5} - 2)$. A parallel argument yields $x_{4n+3} \to -249(\sqrt{5} - 2)$ as $n \to \infty$.

Finally, since $x_1 = x_0 - (1/L)\nabla f(x_0)$, to guarantee $x_1 \in [1, 245]$, a sufficient condition is $x_0 \in [2.01, 246.98]$. This completes the proof.

References

[1] Donald G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12(4):547–560, 1965.

[2] Heinz H. Bauschke, Jérôme Bolte, and Marc Teboulle. A descent lemma beyond Lipschitz gradient continuity: First-Order methods revisited and applications. *Mathematics of Operations Research*, 42(2):330–348, 2016.

[3] Amir Beck. *First-order methods in optimization*, volume 25. SIAM, 2017.

[4] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[5] Raghu Bollapragada, Damien Scieur, and Alexandre d'Aspremont. Nonlinear acceleration of momentum and primal-dual algorithms. *arXiv preprint arXiv:1810.04539*, 2018.

[6] Jérôme Bolte, Shoham Sabach, Marc Teboulle, and Yakov Vaisbourd. First order methods beyond convexity and lipschitz gradient continuity with applications to quadratic inverse problems. *SIAM Journal on Optimization*, 28(3):2131–2151, 2018.

[7] Claude Brezinski, Michela Redivo-Zaglia, and Yousef Saad. Shanks sequence transformations and Anderson acceleration. *SIAM Review*, 60(3):646–669, 2018.

[8] Dan Butnariu and Alfredo N. Iusem. *Totally convex functions for fixed points computation and infinite dimensional optimization*, volume 40. Springer Science & Business Media, 2012.

[9] Yair Censor and Stavros A. Zenios. Proximal minimization algorithm with D-functions. *Journal of Optimization Theory and Applications*, 73(3):451–464, 1992.

[10] Xiaojun Chen and C. T. Kelley. Convergence of the EDIIS algorithm for nonlinear equations. *SIAM Journal on Scientific Computing*, 41(1):A365–A379, 2019.

[11] Xin Chen, Houduo Qi, and Paul Tseng. Analysis of nonsmooth symmetric-matrix-valued functions with applications to semidefinite complementarity problems. *SIAM Journal on Optimization*, 13(4):960–985, 2003.

[12] Frank H. Clarke. *Optimization and nonsmooth analysis*, volume 5. SIAM, 1990.

[13] Radu-Alexandru Dragomir, Jérôme Bolte, and Alexandre d'Aspremont. Fast gradient methods for symmetric nonnegative matrix factorization. *arXiv preprint arXiv:1901.10791*, 2019.

[14] Jonathan Eckstein. Nonlinear proximal point algorithms using Bregman functions, with applications to convex programming. *Mathematics of Operations Research*, 18(1):202–226, 1993.

[15] R. P. Eddy. Extrapolating to the limit of a vector sequence. In *Information linkage between applied mathematics and industry*, pages 387–396. Elsevier, 1979.

[16] V. Eyert. A comparative study on methods for convergence acceleration of iterative vector sequences. *Journal of Computational Physics*, 124(2):271–285, 1996.

[17] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional variational inequalities and complementarity problems.* Springer Science & Business Media, 2007.

[18] Haw-ren Fang and Yousef Saad. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16(3):197–221, 2009.

[19] Anqi Fu, Junzi Zhang, and Stephen Boyd. Anderson accelerated Douglas-Rachford splitting. *arXiv preprint arXiv:1908.11482*, 2019.

[20] Matthieu Geist and Bruno Scherrer. Anderson acceleration for reinforcement learning. *arXiv preprint arXiv:1809.09501*, 2018.

[21] Anne Greenbaum. *Iterative methods for solving linear systems*, volume 17. SIAM, 1997.

[22] Filip Hanzely, Peter Richtarik, and Lin Xiao. Accelerated Bregman proximal gradient methods for relatively smooth convex optimization. *arXiv preprint arXiv:1808.03045*, 2018.

[23] Nicholas J. Higham and Nataša Strabić. Anderson acceleration of the alternating projections method for computing the nearest correlation matrix. *Numerical Algorithms*, 72(4):1021–1042, 2016.

[24] Zhize Li and Jian Li. An Anderson-Chebyshev mixing method for nonlinear optimization. *arXiv preprint arXiv:1809.02341*, 2018.

[25] Jingwei Liang, Jalal Fadili, and Gabriel Peyré. A multi-step inertial forward-backward splitting method for non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 4035–4043, 2016.

[26] Haihao Lu, Robert M. Freund, and Yurii Nesterov. Relatively smooth convex optimization by first-order methods, and applications. *SIAM Journal on Optimization*, 28(1):333–354, 2018.

[27] Vien V. Mai and Mikael Johansson. Nonlinear acceleration of constrained optimization algorithms. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4903–4907, 2019.

[28] Mathurin Massias, Joseph Salmon, and Alexandre Gramfort. Celer: A fast solver for the lasso with dual extrapolation. In *International Conference on Machine Learning*, pages 3321–3330, 2018.

[29] Robert Mifflin. Semismooth and semiconvex functions in constrained optimization. *SIAM Journal on Control and Optimization*, 15(6):959–972, 1977.

[30] Arkadi Nemirovski. Information-based complexity of convex programming. *Lecture Notes*, 1995.

[31] Arkadii Nemirovsky and David Yudin. *Problem complexity and method efficiency in optimization.* Wiley, 1983.

[32] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course.* Springer, New York, USA, 2004.

[33] Yurii Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[34] Brendan O'donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015.

[35] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[36] Yue Peng, Bailin Deng, Juyong Zhang, Fanyu Geng, Wenjie Qin, and Ligang Liu. Anderson acceleration for geometry optimization and physics simulation. *ACM Transactions on Graphics*, 37(4):42, 2018.

[37] Sara Pollock and Leo Rebholz. Anderson acceleration for contractive and noncontractive operators. *arXiv preprint arXiv:1909.04638*, 2019.

[38] Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[39] Clarice Poon and Jingwei Liang. Trajectory of alternating direction method of multipliers and adaptive acceleration. *arXiv preprint arXiv:1906.10114*, 2019.

[40] Florian A. Potra and Hans Engler. A characterization of the behavior of the Anderson acceleration on linear problems. *Linear Algebra and its Applications*, 438(3):1002–1011, 2013.

[41] Péter Pulay. Convergence acceleration of iterative sequences. The case of SCF iteration. *Chemical Physics Letters*, 73(2):393–398, 1980.

[42] Liqun Qi and Jie Sun. A nonsmooth version of Newton's method. *Mathematical programming*, 58(1-3):353–367, 1993.

[43] R. Tyrrell Rockafellar. *Convex Analysis*, volume 28. Princeton University Press, 1970.

[44] R. Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis.* Springer Science & Business Media, 2009.

[45] Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[46] Damien Scieur, Francis Bach, and Alexandre d'Aspremont. Nonlinear acceleration of stochastic algorithms. In *Advances in Neural Information Processing Systems*, pages 3982–3991, 2017.

[47] Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. In *Advances In Neural Information Processing Systems*, pages 712–720, 2016.

[48] Avram Sidi. *Vector extrapolation methods with applications*, volume 17. SIAM, 2017.

[49] David A. Smith, William F. Ford, and Avram Sidi. Extrapolation methods for vector sequences. *SIAM review*, 29(2):199–233, 1987.

[50] Alex Toth and CT Kelley. Convergence analysis for Anderson acceleration. *SIAM Journal on Numerical Analysis*, 53(2):805–819, 2015.

[51] Homer F. Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

[52] Peter Wynn. Acceleration techniques for iterated vector and matrix problems. *Mathematics of Computation*, 16(79):301–322, 1962.

[53] Junzi Zhang, Brendan O'Donoghue, and Stephen Boyd. Globally convergent type-I Anderson acceleration for non-smooth fixed-point iterations. *arXiv preprint arXiv:1808.03971*, 2018.