Anderson Acceleration of Proximal Gradient Methods

Vien V. Mai 1  Mikael Johansson 1

Abstract

Anderson acceleration is a well-established and simple technique for speeding up fixed-point computations with countless applications. This work introduces novel methods for adapting Anderson acceleration to proximal gradient algorithms. Under some technical conditions, we extend existing local convergence results of Anderson acceleration for smooth fixed-point mappings to the proposed non-smooth setting. We also prove analytically that it is in general, impossible 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. Finally, we provide the first applications of Anderson acceleration to non-Euclidean geometry.

1. Introduction

The last few decades have witnessed significant advances in the theory and practice of convex optimization based on first-order information (Nesterov, 2004; Beck, 2017). The worst-case oracle complexity has been established for many function classes (Nemirovsky & Yudin, 1983) 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 (O’donoghue & Candes, 2015). Thus, strong practical performance of optimization algorithms requires local adaption and acceleration. Efficient line-search procedures (Nesterov, 2013), adaptive restart techniques (O’donoghue & Candes, 2015) and nonlinear acceleration schemes (Scieur et al., 2016) are therefore now receiving an increasing attention.

Extrapolation techniques have a long history in numerical analysis (see, e.g., (Sidi, 2017; Brezinski et al., 2018)). Recently, its idea has resurfaced in the first-order optimization literature (Scieur et al., 2016; Zhang et al., 2018; Mai & Johansson, 2019; Fu et al., 2019; Poon & Liang, 2019). Unlike momentum acceleration methods such as Polyak’s heavy ball (Polyak, 1964) and Nesterov’s fast gradient (Nesterov, 2004), which require knowledge of problem parameters, classical extrapolation techniques for vector sequences such as minimal polynomial extrapolation (Smith et al., 1987), reduced rank extrapolation (Eddy, 1979), vector epsilon algorithm (Wynn, 1962), and Anderson acceleration (Anderson, 1965) estimate the 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.

Related Work  Anderson acceleration (AA) was proposed in the 1960’s to expedite solution times for nonlinear integral equations (Anderson, 1965). 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. (Pulay, 1980; Eyert, 1996; Walker & Ni, 2011). However, AA and optimization algorithms have been developed quite independently and only limited connections were discovered and studied (Eyert, 1996; Fang & Saad, 2009). Very recently, the technique has started to gain a significant interest in the optimization community (see, e.g., (Scieur et al., 2016; 2017; Bollapragada et al., 2018; Zhang et al., 2018; Fu et al., 2019; Poon & Liang, 2019)). Specifically, a series of papers (Scieur et al., 2016; 2017; Bollapragada et al., 2018) adapt AA to accelerate several classical algorithms for unconstrained optimization; (Zhang et al., 2018) studies a variant of AA for non-expansive operators; (Fu et al.,
Although some initial success has been obtained for adapting AA to optimization algorithms, current research has mainly focused on unconstrained or linearly constrained minimization (e.g., (Scieur et al., 2016; Fu et al., 2019)). For non-smooth composite problems, asymptotic convergence results of AA are often achieved by additional safeguarding strategies (Zhang et al., 2018), without which even local convergence guarantees have not been available. This is because AA relies on linearization (and hence often requires only uses the last iterate to generate a new estimate, AA tries to make better use of past information. Concretely, let \( \{ x_i \}_{i=0}^n \) 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 \( k \)th 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 \( \bar{x}_k = \sum_{i=0}^{m} \alpha_i x_{k-i} \), AA seeks to find a vector of coefficients \( \alpha^k = [\alpha^k_0, \ldots, \alpha^k_m] \) such that
\[
\alpha^k = \arg\min_{\alpha^k} \| g(\bar{x}_k) - \bar{x}_k \|.
\]
However, since (1) can be hard to solve for a general non-linear mapping \( g \), AA uses
\[
\alpha^k = \arg\min_{\alpha^k} \| R_k \alpha^k \|.
\]
It is clear that Problems (1) and (2) are equivalent if \( g \) is an affine map. Let \( R_k = [r_0, \ldots, r_{k-m}] \) be the residual matrix at the \( k \)th iteration, Problem (2) can then be written as
\[
\alpha^k = \arg\min_{\alpha^{\top} = 1} \| R_k \alpha \|.
\]
With \( \alpha^k \) computed, the next AA iterate is then generated by
\[
x_{k+1} = \sum_{i=0}^{m} \alpha^k_i g(x_{k-i}),
\]
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.

**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. \( R_k = [r_0, \ldots, r_{k-m}] \), where \( r_i = g(x_i) - x_i \)

5. \( \alpha^k \leftarrow \arg\min_{\alpha^{\top} = 1} \| R_k \alpha \| \)

6. \( x_{k+1} \leftarrow \sum_{i=0}^{m_k} \alpha^k_i g(x_{k-i}) \)

7. **end for**

**Output:** \( x_K \)

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 that can be computed by first solving the \( m \times m \) normal equations \( R_k^T R_k x = 1 \) and then normalizing the result to obtain \( \alpha^k = x/(1^T x) \) (Scieur et al., 2016). 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 \( O(n^2 + mn) \) (Higham & Strabić, 2016). Since \( m \) is typically between 1 and 10 in practice, this additional cost is often negligible compared to the cost of evaluating \( g \).

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

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. The intuition behind AA for GD is that smooth functions are well approximated by quadratic ones. The intuition behind AA for GD is that smooth functions are well approximated by quadratic ones. The intuition behind AA for GD is that smooth functions are well approximated by quadratic ones. The intuition behind AA for GD is that smooth functions are well approximated by quadratic ones.

![Figure 1. Quadratic convex problems: Left: \( \lambda_1(A)/\lambda_{25}(A) = 10^3 \). Right: \( \lambda_1(A)/\lambda_{25}(A) = 10^4 \).](image)

### 2.1. Anderson acceleration for optimization algorithms

Since many optimization methods 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 \( f \):

\[ x_{k+1} = x_k - \nabla f(x_k), \]

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

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

\[ \min_{x \in \mathbb{R}^n} \frac{1}{2} x^T A x - b^T x, \]

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 (Walker & Ni, 2011; Potra & Engler, 2013) that AA with full information (i.e. setting \( m = \infty \) in Step 3 of Algorithm 1) is essentially equivalent to GMRES (Saad & Schultz, 1986). Therefore, AA admits the convergence rate (Nemirovski, 1995; Greenbaum, 1997)

\[ \| x_k - x^* \|_2^2 \leq O \left( \min \left\{ 1/k^2, e^{-k/\sqrt{n}} \right\} \right) \| x_0 - x^* \|_2^2, \]

where \( \kappa = \lambda_1(A)/\lambda_{n}(A) \) is the condition number. This rate shows a 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) (Nesterov, 2004) can only achieve this rate if \( \lambda_1(A) \) and \( \lambda_{n}(A) \) are known.

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. We compared AA-GD with GD, AGD, and the adaptive restart scheme (AGD-Restart) in (O’donoghue & Candes, 2015). 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 (6), even with a very small value of \( m \) and on an objective function which is not strongly convex.

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

\[ f(x) \approx f(x^*) + (x - x^*)^T \nabla^2 f(x^*)(x - x^*). \]

Note that the matrix \( \nabla^2 f(x^*) \) may have smallest eigenvalue \( \lambda_{\min} \) strictly greater than the \textit{global} strong convexity constant \( \mu \). 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 (Eyert, 1996; Fang & Saad, 2009; Walker & Ni, 2011). Here, we point out some connections between AA-GD and multi-step methods in optimization. To do so, let \( \gamma_i^k := \sum_{j=1}^{m_k} \alpha_j^k, \)

\[ \gamma_i^k := \sum_{j=1}^{m_k} \alpha_j^k, \]

\[ i \in \{1, \ldots, m_k\} \] and define \( y_i^k := \sum_{j=1}^{m_k} \alpha_j^k x_{k-j} \). AA-GD can then be written as

\[ y_k^\alpha = x_k - \sum_{i=1}^{m_k} \gamma_i^k (x_{k-i+1} - x_{k-i}), \]

\[ y_k^\alpha = x_k - \sum_{i=1}^{m_k} \gamma_i^k (x_{k-i+1} - x_{k-i}). \]

Recall that Nesterov’s accelerated gradient method (AGD) (Nesterov, 2004) can be written as

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

One of the key features of AA is its ability to adapt to the local properties of the function being optimized. This is achieved by constructing a series of quadratic approximations to the function, which are then used to construct a sequence of steps that converge to the optimum. The key idea behind AA is to use past steps to predict future steps, which can significantly speed up convergence.

The rate of convergence of AA is determined by the condition number of the function being optimized. In particular, AA enjoys a convergence rate of \( O(1/k^2) \) for strongly convex functions, which is faster than the linear convergence rate of standard gradient descent.

AA has been shown to be effective in a variety of applications, including machine learning, signal processing, and system identification. One of the key advantages of AA is its ability to handle non-smooth functions, which is a common feature of many optimization problems.

In practice, AA is often used in conjunction with other optimization methods, such as gradient descent or quasi-Newton methods, to improve performance. For example, AA can be used to accelerate the convergence of gradient descent, or to improve the performance of quasi-Newton methods in cases where the Hessian is difficult to compute.

In conclusion, Anderson Acceleration is a powerful technique for accelerating the convergence of optimization algorithms. Its ability to adapt to the local properties of the function being optimized makes it a valuable tool for many optimization problems.
while Polyak’s Heavy ball (HB) method (Polyak, 1964) 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 \( m_k = 1 \), AA-GD is analogous to AGD and HB with \( \beta_k, \beta'_k \) replaced by \( \gamma_k^2 \). However, their update directions are chosen differently: AGD takes a step using the gradient at the extrapolated point \( y_k \), while AA-GD 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 (Liang et al., 2016). However, unlike AA, there is currently no efficient way to select the coefficients in MiFB, thereby restricting its historical parameter to \( m = 1 \) or 2.

### 3. Anderson Acceleration for Proximal Gradient Method

Consider a composite minimization problem of the form

\[ \text{minimize} \quad \varphi(x) := f(x) + h(x), \quad (7) \]

where \( f: \mathbb{R}^n \to \mathbb{R} \) is \( L \)-Lipschitz smooth, i.e.

\[ \| \nabla f(x) - \nabla f(y) \|_2 \leq L \| x - y \|_2, \quad \forall x, y \in \text{dom} \ f. \]

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

\[ \text{prox}_h(y) := \arg\min_x \left\{ h(x) + \frac{1}{2} \| x - y \|_2^2 \right\}. \]

A classical method for solving (7) is the proximal gradient algorithm (PGA)

\[ x_{k+1} = \text{prox}_{\gamma h} (x_k - \gamma \nabla f(x_k)), \quad (8) \]

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

\[ g(x) = \text{prox}_{\gamma h} (x - \gamma \nabla f(x)). \quad (9) \]

It is not difficult to show that \( x^* \) is a minimizer of (7) if and only if

\[ x^* = \text{prox}_{\gamma h} (x^* - \gamma \nabla f(x^*)), \quad (10) \]

which implies that finding \( x^* \) amounts to finding a fixed-point of \( g \).

In light of our previous discussion, it would be natural to speed-up the PGA method by applying AA to the mapping \( g \) in (9). 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. As AA forms an affine (and not a convex) combination in each step, the resulting

![](Algorithm 2 AA-PGA)

\[ \text{Algorithm 2 AA-PGA} \]

**Input:** \( x_0 = y_0, \ m \geq 0 \)

1: \( y_1 \leftarrow x_0 - \gamma \nabla f(x_0), \ x_1 \leftarrow \text{prox}_{\gamma h}(y_1), \ g_0 \leftarrow y_1 \)
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: \( \alpha^k \leftarrow \arg\min_{\alpha \in R_k} \| R_k \alpha \| \)
7: \( y_{k+1} \leftarrow \sum_{i=0}^{m_k} \alpha^k_i r_{k-i} \)
8: \( x_{k+1} \leftarrow \text{prox}_{\gamma h}(y_{k+1}) \)
9: end for

**Output:** \( x_K \)

Algorithm 2 AA-PGA

Iterates can lie outside \( \text{dom} \ h \) (at which \( \nabla f \) may not exist). Nevertheless, if we rewrite the PGA iteration as

\[ y_{k+1} = x_k - \gamma \nabla f(x_k) \quad \text{and} \quad x_{k+1} = \text{prox}_{\gamma h}(y_{k+1}), \quad (11) \]

and consider the mapping \( g \) defined as

\[ g(y) = \text{prox}_{\gamma h}(y) - \gamma \nabla f(\text{prox}_{\gamma h}(y)), \quad (12) \]

then the fixed-point iteration \( y_{k+1} = g(y_k) \) recovers exactly the PGA iteration in (11). It is clear that if \( y^* \) is a fixed-point of \( g \), then \( x^* = \text{prox}_{\gamma h}(y^*) \) is an optimal solution to (7) since it satisfies condition (10). 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 \( x^* \) satisfies (10), then \( y^* = x^* - \gamma \nabla f(x^*) \) is a fixed-point of \( g \) defined in (12) and

\[ \| x_k - x^* \|_2 \leq \| y_k - y^* \|_2, \]

where the last step follows from the nonexpansiveness of proximal operators. The inequality implies that if one can quickly drive \( \{ y_k \} \) to \( y^* \), then \( \{ x_k \} \) will quickly converge to \( x^* \). It turns out that working with this \( g \) is also convenient in designing our safeguarding scheme later.

We thus propose to use AA for accelerating the auxiliary sequence \( \{ y_k \} \) governed by \( g \) defined in (12). 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. The resulting scheme, which we call AA-PGA, is summarized in Algorithm 2.

### 3.1. Local Convergence Guarantees

Although convergence properties of AA for linear mappings with full memory \( (m = \infty) \) are relatively well understood (Walker & Ni, 2011; Potra & Engler, 2013), much less is known in the case of nonlinear mappings and limited-memory. The work (Toth & Kelley, 2015) was the first to show that no matter what value \( m \in \{0, 1, \ldots \} \) is used, AA
Anderson Acceleration of Proximal Gradient Methods

does not harm the convergence of the fixed-point iteration when started near the fixed point. The proof requires continuous differentiability of $g$. However, in the context of composite convex optimization, the mapping $g$ defined in (12) is, in general, non-differentiable. Therefore, the analysis in (Toth & Kelley, 2015) is not applicable anymore. To circumvent this difficulty, we rely on the notion of generalized second-order differentiability, defined below. The interested reader is referred to (Rockafellar & Wets, 2009, Section 13) for a comprehensive treatment of epi-differentiability.

**Definition 3.1.** A function $f$ is twice epi-differentiable at $x$ for a vector $v \in \mathbb{R}^n$ if it is epi-differentiable at $x$ and the second-order quotient functions $\Delta^2_{x,v,f}$ defined by

$$\Delta^2_{x,v,f}(x') = \frac{[f(x+tv') - f(x) - t\langle v, x' \rangle] / (t^2/2)}{t > 0},$$

for $t > 0$, epi-converge to a proper function as $t \to 0$.

The limit, denoted by $\Delta^2_{x,v,f}$, is then the second-order epi-derivative of $f$.

We make the following assumption.

**Assumption A1.** Let $x^* \in \text{argmin}_x \varphi(x)$. We assume that:

(A1.i) the function $f$ is of class $C^2$ around $x^*$ and there exists a real $\nu > 0$ such that $\nabla^2 f(x^*) \succeq \nu I$.

(A1.ii) the convex function $h$ is twice epi-differentiable at $x^*$ for $-\nabla f(x^*)$ and the corresponding second-order epi-derivative is generalized quadratic:

$$\Delta^2_{x^*, -\nabla f(x^*)}h(\xi) = \begin{cases} \frac{1}{2} \langle \xi, Q\xi \rangle, & \xi \in L \\ \infty, & \text{otherwise,} \end{cases}$$

where $L$ is a linear subspace of $\mathbb{R}^n$ and $Q \in \mathbb{R}^{n \times n}$ is a symmetric matrix.

Twice epi-differentiable functions, introduced by Rockafellar in (Rockafellar, 1988), are remarkable in the sense that they may be both non-smooth and extended real-valued, but still have useful second-order properties. One important class of twice epi-differentiable functions are known as fully amenable (Poliquin & Rockafellar, 1995). In the context of (additive) composite optimization, full amenability is justified whenever $f \in C^2$ and $h$ is a polyhedral function (i.e., its epigraph is a polyhedral set). Indeed, (Poliquin & Rockafellar, 1995, Proposition 2.6) ensures that $\varphi = f + h$ is fully amenable at any feasible $x$, which in turn implies twice epi-differentiability of $h$ at $x$ for $-\nabla f(x)$ since $\partial \varphi(x) = \nabla f(x) + \partial h(x)$. Notable examples of polyhedral $h$ in machine learning applications are the $\ell_1$-norm, $\ell_{\infty}$-norm, total variation seminorm, and the indicator functions of polyhedral sets such as the non-negative orthant, box constraints and the probability simplex.

For the preceding $\varphi$, it is shown in (Poliquin & Rockafellar, 1995, Proposition 4.12), that the function $\Delta^2_{x^*, -\nabla f(x^*)}h$ is generalized quadratic if and only if $(x^*, \nabla f(x^*))$ satisfies the non-degeneracy condition:

$$-\nabla f(x^*) \in \text{relint}(\partial h(x^*)).$$

More broadly, if condition (13) holds, then any $C^2$-partly smooth function $h$ satisfies the properties in (A1.ii) (this follows by combining (Danilidis et al., 2006, Theorem 28) and (Poliquin & Rockafellar, 1996, Theorem 4.1(a) and (g)); see (Themelis et al., 2018) for detailed arguments). This allows to include regularizers which are not polyhedral, like the nuclear norm in matrix completion and the $\ell_1-\ell_2$-norm in group lasso (Liang et al., 2017).

Note that condition (13) is very mild and can be seen as a geometric generalization of the well-known strict complementarity in nonlinear programming (Burke, 1990). For example, for the lasso problem with $f(x) = (1/2) \| Ax - b \|_2^2$ and $h(x) = \lambda \| x \|_1$, it is easy to verify that (13) is justified as long as $\| (A^T (Ax - b) - b) \|_1 \neq \lambda$ whenever $(x^*)_i = 0$. In fact, this condition has been considered almost necessary for identifying the support of $x^*$ (Liang et al., 2017).

An important consequence of Assumption A1 is that the proximal mapping $\text{prox}_g$ becomes differentiable at $x^* - \gamma \nabla f(x^*)$. This fact is summarized in the following lemma.

**Lemma 3.1.** Let Assumption A1 hold. Then, the proximal operator $\text{prox}_g$ is differentiable at $y^* = x^* - \gamma \nabla f(x^*)$ and its Jacobian $P_\gamma(y^*) := J_{\text{prox}_g}(y^*)$ is symmetric and positive semidefinite with $\| P_\gamma(y^*) \|_2 \leq 1$. Moreover, the mapping $g$ is differentiable at $y^*$ with Jacobian:

$$G = P_\gamma(y^*) (I - \gamma \nabla^2 f(x^*)) .$$

If, in addition, $\gamma \in (0, 1/L]$, then $\| G \|_2 \leq 1 - \gamma \nu \in [0, 1)$.

**Proof.** Detailed arguments for differentiability of $\text{prox}_g$ at $y^*$ can be found in (Themelis et al., 2018, Thm. 4.10). The Jacobian $G$ of $g$ at $y^*$ is a direct consequence of the chain rule. Finally, since $\nabla^2 f(x^*) \succeq \nu I$ and $f$ is $L$-smooth, we have $\| G \|_2 \leq \| P_\gamma(y^*) \|_2 \| (I - \gamma \nabla^2 f(x^*)) \|_2 \leq 1 - \gamma \nu$, as desired. \hfill $\square$

Our last assumption imposes a boundedness condition on the extrapolation coefficients.

**Assumption A2.** There exists a constant $M_\alpha$ such that $\| \alpha^k \|_1 \leq M_\alpha$ for all $k \in \mathbb{N}_+$.

This assumption is very common in the literature of AA and some effective solutions have been proposed to enforce 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 (Walker & Ni, 2011), or one can add a Tikhonov regularization to the least squares as was done in (Scieur
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 \( M_\alpha \), we can set \( \alpha^k = [0, \ldots, 1]^T \), i.e., we simply perform a fixed-point iteration step.

We can now state the main result of this section.

**Theorem 1.** Let Assumptions A1 and A2 hold. Let \( \gamma \in (0, 1/L) \) and define \( \hat{\rho}(G) = \|G\|_2 \). Let \( \hat{\rho} \) be some real constant satisfying \( \hat{\rho} \in (\rho(G), 1) \). Let \( F(g) = g(y) - y \) with \( g \) given in (12) and let \( y^\ast = x^\ast - \gamma \nabla f(x^\ast) \) be a fixed-point of \( g \). If \( y_0 \) is initialized sufficiently close to \( y^\ast \), then, for any fixed \( m \in \mathbb{N} \), the iterates \( \{x_k\} \) and \( \{y_k\} \) formed by AA-PGA satisfy:

\[
\|F(y_k)\|_2 \leq \hat{\rho}^k \|F(y_0)\|_2 \\
\|x_k - x^\ast\|_2 \leq c\hat{\rho}^k \|y_0 - y^\ast\|_2,
\]

where \( c = (3 + \rho(G))/(1 - \rho(G)) \). Moreover, we have

\[
\limsup_{k \to \infty} \left( \frac{\|F(y_k)\|_2}{\|F(y_0)\|_2} \right)^{1/k} \leq \rho(G) \\
\limsup_{k \to \infty} \left( \frac{\|x_k - x^\ast\|_2}{\|x_0 - x^\ast\|_2} \right)^{1/k} \leq \rho(G).
\]

**Proof.** See Appendix B.

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 (6) for even small values 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 Accelerated PGA

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 (Fu et al., 2019), is whether AA 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 (Walker & Ni, 2011; Scieur et al., 2016). 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 \( f \) whose gradient is given by

\[
\nabla f(x) = \begin{cases} 
\frac{x}{10} - 24.9 & \text{if } x < -1, \\
25x & \text{if } -1 \leq x < 1, \\
\frac{x}{10} + 24.9 & \text{if } x \geq 1.
\end{cases}
\]

This \( f \) is strongly convex with \( \mu = 1/10 \) and smooth with \( L = 25 \). A trajectory of AA-GD with \( m = 1 \) started at \( x_0 = 2.1 \) is depicted in Figure 2 indicating that it 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 (14). 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 \( x_0 \in [2.01, 246.98] \) and \( n = 0, 1, \ldots \), the iterates generated by AA-GD satisfy:

\[
x_{4n+3} \to -249(\sqrt{5} - 2), \quad x_{4n+4} \to +249, \\
x_{4n+5} \to +249(\sqrt{5} - 2), \quad x_{4n+6} \to -249.
\]

**Proof.** See Appendix C.

The proposition confirms the necessity of a safeguarding step to ensure global convergence (see, e.g., (Zhang et al., 2018; Fu et al., 2019)). Note that such a step often only involves checking a simple condition, and hence is cheaper to execute than a line search.

Recall that each iteration of AA-PGA consists of one original PGA step followed by an AA step. Thus, one natural strategy for stabilization would be to compare the objective value produced by the AA step with that of PGA and select the one with lower value as the next iterate. However, this approach 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:

\[
f(x_{k+1}) \leq f(x_k) - \frac{\gamma}{2} \|\nabla f(x_k)\|^2_2.
\]

This suggests an alternative way for stabilization, which is to compare the objective value of the AA step with the
Algorithm 3 Guared AA-PGA

Input: \( y_0 = x_0, m \geq 0 \)

1: \( y_1 \leftarrow x_0 - \gamma \nabla f(x_0), x_1 \leftarrow \text{prox}_{\gamma h}(y_1), g_0 \leftarrow y_1, \)

and \( r_0 = g_0 - y_0 \)

2: for \( k = 1, \ldots, K - 1 \) do

3: \( m_k \leftarrow \min(m, k) \)

4: \( g_k \leftarrow x_k - \gamma \nabla f(x_k) \) and \( r_k \leftarrow g_k - y_k \)

5: \( R_k \leftarrow [r_k, \ldots, r_{k-m_k}] \)

6: \( \alpha_k \leftarrow \arg\min_{\alpha} \|R_k\alpha\| \)

7: \( y_{\text{ext}} \leftarrow \sum_{i=0}^{m_k} \alpha_i g_{k-i} \)

8: \( x_{\text{test}} \leftarrow \text{prox}_{\gamma h}(y_{\text{ext}}) \)

9: if \( f(x_{\text{test}}) \leq f(x_k) - \frac{\gamma}{2} \|\nabla f(x_k)\|^2 \) then

10: \( x_{k+1} \leftarrow x_{\text{test}} \quad \text{and} \quad y_{k+1} \leftarrow y_{\text{ext}} \)

11: else

12: \( x_{k+1} \leftarrow x_{k-1} \quad \text{and} \quad y_{k+1} \leftarrow g_k \)

13: end if

14: end for

Output: \( x_K \)

right-hand side of (15). If sufficient descent was made, the AA step is accepted, otherwise the PGA step is chosen. 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 \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) be \( \mu \)-strongly convex and \( L \)-smooth and let \( \gamma \in (0, 2/(\mu + L)] \). Then, the iterates \( \{x_k\} \) generated by Algorithm 3 satisfy

\[
\|x_k - x^*\|^2 \leq O \left( \frac{1 - \gamma \mu L}{\mu + L} \right)^k \|x_0 - x^*\|^2. \tag{16}
\]

The proof of this result is straightforward (see, e.g., (Nesterov, 2004; Beck, 2017)), and follows directly from the descent condition (15) and a standard strong convexity inequality. Hence, we omit it here.

5. AA-Bregman Proximal Gradient Methods

Consider optimization problems of the form

\[
\text{minimize} \quad f(x) + h(x), \tag{17}
\]

where \( D \subseteq \mathbb{R}^n \) is a closed convex set. The formulation (17) often provides a more flexible way to handle the constraints, which are usually encoded by \( h \) in (7). 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 (17) thanks to its ability to exploit the underlying geometry of the problem. The mirror descent method (Nemirovsky & Yudin, 1983; Beck & Teboulle, 2003) is a well-known instance of BPG when \( h(x) = 1_C(x) \) for some closed convex set \( C \subseteq D \). Some more recent instances of BPG include the NoLips method (Bauschke et al., 2016) and its accelerated version analysed in (Hanzely et al., 2018). The number of applications of the BPG framework are growing rapidly (Bolte et al., 2018; Dragomir et al., 2019; Lu et al., 2018).

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 \( \varphi(x) = (1/2) \|x\|^2 \); the Shannon entropy \( \varphi(x) = \sum_{i=1}^n x_i \log x_i \), \( \text{dom}\, \varphi = \mathbb{R}_+^n \) with \( (\log 0 = 0) \); the Burg entropy \( \varphi(x) = -\sum_{i=1}^n \log x_i \), \( \text{dom}\, \varphi = \mathbb{R}_+^n \); the Fermi-Dirac entropy \( \varphi(x) = \sum_{i=1}^n (x_i \log x_i + (1 - x_i) \log(1 - x_i)) \), \( \text{dom}\, \varphi = [0, 1]^n \); the Hellinger entropy \( \varphi(x) = -\sum_{i=1}^n \sqrt{1 - x_i^2} \), \( \text{dom}\, \varphi = [-1, 1]^n \); and the polynomial function \( \varphi(x) = \frac{a}{2} \|x\|^2 + \frac{1}{4} \|x\|^4, a \geq 0 \).

We impose the following assumption in this section.

Assumption A3. The set \( \text{dom}\varphi = D \) is convex and the following conditions hold:

1. \( \varphi : \mathbb{R}^n \rightarrow (-\infty, +\infty) \) is of Legendre type and its conjugate \( \varphi^* \) satisfies \( \text{dom}\, \nabla \varphi^* = \mathbb{R}^n \).

2. \( f : \mathbb{R}^n \rightarrow (-\infty, +\infty) \) is proper closed convex and differentiable on \( \text{int}\, \text{dom}\, \varphi \).

3. \( h : \mathbb{R}^n \rightarrow (-\infty, +\infty) \) is proper closed convex and \( \text{dom}\, h \cap \text{int}\, \text{dom}\, \varphi \neq \emptyset \).

When \( \varphi \) is Legendre, its gradient \( \nabla \varphi \) is a bijection from \( \text{int}\, \text{dom}\, \varphi \) to \( \text{int}\, \text{dom}\, \varphi^* \) while \( \nabla \varphi^* \) is a bijection from \( \text{int}\, \text{dom}\, \varphi^* \) to \( \text{int}\, \text{dom}\, \varphi \), i.e., \( (\nabla \varphi)^{-1} = \nabla \varphi^* \) (Rockafellar, 1970, Chapter 26). Note that in all the above examples, \( \varphi \) is Legendre. Moreover, except from the Burg entropy, all the others share the useful property \( \nabla \varphi^* = \mathbb{R}^n \), which is critical for the development of our AA scheme.

The Bregman distance associated with \( \varphi \) is the function \( D_\varphi : \text{dom}\, \varphi \times \text{int}\, \text{dom}\, \varphi \rightarrow \mathbb{R} \) 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 for \( y \in \text{int}\, \text{dom}\, \varphi \) as (Censor & Zenios, 1992):

\[
\text{prox}_h^\varphi(y) = \arg\min_{x \in \mathbb{R}^n} \{ h(x) + D_\varphi(x, y) \}. \tag{18}
\]

BPG starts with some \( x_0 \in \text{int}\, \text{dom}\, \varphi \) and performs the following operator at each iteration:

\[
x_{k+1} = \arg\min_{x \in \mathbb{R}^n} \{ (\nabla f(x_k), x - x_k) + \gamma^{-1} D_\varphi(x, x_k) + h(x) \},
\]
Anderson Acceleration of Proximal Gradient Methods

which can also be expressed equivalently as (Teboulle, 2018)

\[ x_{k+1} = \text{prox}_{\gamma_h}^\varphi \left( \nabla \varphi^* \left( \nabla \varphi(x_k) - \gamma \nabla f(x_k) \right) \right). \]

Assumption A3 ensures that BPG iterates are well-defined and \( x_k \in \text{int dom } \varphi \) for all \( k \) (Bauschke et al., 2016, Lemma 2). Note that when \( \varphi = \frac{1}{2} \| \cdot \|^2 \), \( \nabla \varphi \) 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

\[
\begin{align*}
    y_{k+1} &= \nabla \varphi(x_k) - \gamma \nabla f(x_k) \\
    x_{k+1} &= \text{prox}_{\gamma_h}^\varphi \left( \nabla \varphi^* \left( y_{k+1} \right) \right).
\end{align*}
\] (19)

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

\[ g(y) = \nabla \varphi \left( \text{prox}_{\gamma_h}^\varphi \circ \nabla \varphi^* \left( y \right) \right) - \gamma \nabla f \left( \text{prox}_{\gamma_h}^\varphi \circ \nabla \varphi^* \left( y \right) \right). \]

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:

\[ D_\varphi \left( \nabla \varphi^* \left( y \right), \nabla \varphi^* \left( y' \right) \right) = D_{\varphi^*} \left( y', y \right) \forall y, y' \in \text{int dom } \varphi^*. \]

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

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

Applying the Bregman operator to the two images will give us \( x_k \) and \( x^\star \), respectively.

Since Bregman proximal operators possess certain non-expansiveness property akin to their Euclidean counterpart (Butnariu & Iusem, 2012; Eckstein, 1993), 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 \( \text{dom } \varphi \subseteq \text{int dom } \varphi \), it is shown in (Butnariu & Iusem, 2012) that \( D_\varphi \left( x_k, x^\star \right) \leq D_\varphi \left( \nabla \varphi^* \left( y_k \right), \nabla \varphi^* \left( y^\star \right) \right) \). 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 \( x^\star \in \text{int dom } \varphi \). However, if \( x^\star \) happens to be on the boundary of \( \text{dom } \varphi \), 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, a similar policy for stabilization as in AA-PGA retains the convergence rate of BPG. The final guarded AA-BPG algorithm is reported in Appendix A.

6. Numerical Experiments

We will now illustrate the performance of (guarded) AA-PGA and AA-BPG on several constrained optimization problems with important machine learning applications.

We compare AA-PGA to PGA, PGA with adaptive line search (PGA-LS), and accelerated PGA (APGA) (Beck, 2017). For AA-BPG, we compare to BPG, accelerated BPG (ABPG), ABPG with adaptive line search (ABPG-g), and restarted ABPG (ABPG-Restart) (Hanzely et al., 2018). 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 (Scieur et al., 2017), without further tuning. 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. All methods are initialized at \( x_0 = 0 \) unless otherwise stated.

Constrained logistic regression

We start our experiments with box-constrained logistic regression

\[
\min_{\|x\|_2 \leq 1} \frac{1}{M} \sum_{i=1}^{M} \log(1 + \exp(-y_i a_i^\top x)) + \mu \|x\|_2^2,
\]

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 \( A = [a_1, \ldots, a_M] \).

Figure 3 shows the performance of AA-PGA and other selected algorithms on four different data sets. As can be seen, AA consistently and dramatically outperforms the standard first-order methods both in wall-clock time. Since these data sets are very ill-conditioned, standard first-order methods make very little progress, while AA quickly finds a high accuracy approximate solution. This once again demonstrates the benefit of local adaptation and acceleration

\footnote{All experiments are implemented in Python and run in Ubuntu 16.04 LTS on a laptop with four 2.4 GHz cores and 16 GB RAM.}

\footnote{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}
Figure 3. Constrained logistic regression on difference data sets.

as previously seen in unconstrained quadratic problems (cf. Figure 1). In most cases, the convergence rate is linear confirming our prediction.

Nonnegative least squares Next, we consider the non-negative least squares problem:

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

which is a core component of 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-ups over popular first-order methods. Interestingly, in Fig. 4(a), 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.

Relative-entropy nonnegative regression The task is to reconstruct the signal $x \in \mathbb{R}^+_n$ by solving

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

where $A \in \mathbb{R}^M_{+n}$ is a nonnegative observation matrix and $b \in \mathbb{R}^M_{+}$ is a noisy measurement vector. We adapt the family of BPG methods with $\mathcal{D} = \mathbb{R}^n_+$ as the Shannon entropy as kernel $\varphi$, $f(x) = D_{\text{KL}}(Ax, b)$, and $h(x) = \lambda \|x\|_1$ with $\lambda = 0.001$. It is shown in (Bauschke et al., 2016) that $f$ is $L$-smooth relative to $\varphi$ with constant $L = \max_{1 \leq i \leq n} \|a_i\|_1$. We follow (Hanzely et al., 2018) and generate two problem instances with $A$ and $b$ having entries uniformly distributed on $[0, 1]$. All methods are initialized at $x_0 = 1$.

Figure 5(a) shows the suboptimality for a randomly generated problem instance with $M = 100$ and $n = 1000$. These dimensions is often referred to as the easy case, and BPG converges linearly. Figure 5(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.

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.
Acknowledgements

This work was supported in part by the Knut and Alice Wallenberg Foundation, the Swedish Research Council and the Swedish Foundation for Strategic Research. We would like to thank Wenqing Ouyang for his useful feedback and suggestions on an early draft of this paper. We also thank the anonymous reviewers for their useful comments and suggestions.

References


