Globally Convergent Type-I Anderson Acceleration for Non-Smooth Fixed-Point Iterations

Junzi Zhang * Brendan O'Donoghue † Stephen Boyd ‡

December 3, 2018

Abstract

We consider the application of the type-I Anderson acceleration ([FS09]) to solving general non-smooth fixed-point problems. By interleaving with safe-guarding steps, and employing a Powell-type regularization and a re-start checking for strong linear independence of the updates, we propose the first globally convergent variant of Anderson acceleration assuming only that the fixed-point iteration is non-expansive. We show by extensive numerical experiments that many first order algorithms can be improved, especially in their terminal convergence, with the proposed algorithm. Our proposed method of acceleration is being implemented in SCS 2.0 [OCPB17], one of the default solvers used in the convex optimization parser-solver CVXPY 1.0 [AVDB18].

1 Introduction

We consider solving the following general fixed-point problem:

$$\text{Find } x \in \mathbf{R}^n \text{ such that } x = f(x), \tag{1}$$

where $f : \mathbf{R}^n \to \mathbf{R}^n$ is potentially non-smooth. Unless otherwise stated, we assume throughout the paper that f is non-expansive (in the ℓ_2-norm), *i.e.*, $\|f(x) - f(y)\|_2 \leq \|x - y\|_2$ for all $x,\ y \in \mathbf{R}^n$, and that the solution set $X = \{x^\star \mid x^\star = f(x^\star)\}$ of (1) is nonempty. With these assumptions, (1) can be solved by the Krasnosel'skiĭ-Mann (KM, or averaged) iteration algorithm, which updates x^k in iteration k to $x^{k+1} = (1-\alpha)x^k + \alpha f(x^k)$, where $\alpha \in (0, 1)$ is an algorithm parameter. An elementary proof shows the global convergence of KM iteration to some fixed-point $x^\star \in X$ [RB16]. In one sense, our goal is to accelerate the vanilla KM algorithm.

*ICME, Stanford University. **Email:** junziz@stanford.edu

†DeepMind, Google. **Email:** bodonoghue85@gmail.com

‡Department of Electrical Engineering, Stanford University. **Email:** boyd@stanford.edu

Fixed-point problems such as (1) arise ubiquitously in mathematics, natural science and social science. For example, to find a Nash equilibrium in a multi-player game, one can reformulate it as a monotone inclusion problem under mild assumptions on the utility functions [BAC13], which can then be further reformulated as a fixed-point problem of the corresponding (non-expansive) resolvent or Cayley operator [RB16]. In general, the solution of most, if not all, convex optimization problems falls into the above scenario. In fact, almost all optimization algorithms are iterative, and the goal is to solve the corresponding fixed-point problem (1), where $f : \mathbf{R}^n \to \mathbf{R}^n$ is the iteration mapping. When the optimization problem is convex, f is typically non-expansive, and the solution set of the fixed-point problem is the same as that of the original optimization problem, or closed related to it. Another related example is infinite-horizon discounted Markov Decision Process [Bel57, Ber15], in which the optimal policy can be found by solving the fixed-point problem of the associated Bellman operator, which is not non-expansive (in the ℓ_2-norm) but is contractive in the ℓ_∞-norm. Such kind of scenarios are also discussed in §5.1.3 as a variant of our main setting.

In spite of the robustness of the vanilla KM iteration algorithm, the convergence can be extremely slow in practice, especially when high or even just moderate accuracy is needed. Data pre-conditioning and step-size line-search are the two most commonly used generic approaches to accelerate the convergence of the KM method [GFB16]. To further accelerate the convergence, a trade-off between the number of iterations and per-iteration cost is needed. In this case, when $f(x) = x - \alpha \nabla F(x)$ is the gradient descent mapping for the minimization of the differentiable objective function $F(x)$, Newton, quasi-Newton, and accelerated gradient descent methods (*e.g.*, Nesterov's [Nes13], and more recently, [DD18]) can then be used to reduce the overall iteration complexity at the cost of increased cost in each step [LY84]. For more general f, Semi-smooth Newton [AWK17, XLWZ16] and B-differentiable (quasi-)Newton [Pan90, IK92], which generalize their classical counterparts, have also been proposed and widely studied. More recently, some hybrid methods, which interleave vanilla KM iterations with (quasi-)Newton type acceleration steps, are designed to enjoy smaller per-iteration cost while maintaining fast convergence in practice [SdB16, TP16].

Nevertheless, to our knowledge, apart from (pure) pre-conditioning and line-search (which can be superimposed on top of other acceleration schemes), the (local or global) convergence of most, if not all existing methods require additional assumptions, *e.g.*, some kind of differentiability around the solution [MT76, CCL14], symmetry of the Jacobian of f [LF99a, LF99b], or symmetry of the approximate Jacobians in the algorithm [ZL07, ZL08]. Moreover, line search is (almost) always enforced in these methods to ensure global convergence, which can be prohibitive when function evaluations are expensive. Our main goal in this paper is hence to provide a globally convergent acceleration method with relatively small per-iteration costs, without resorting to line search or any further assumptions other than non-expansiveness, thus guaranteeing improvement of a much larger class of algorithms ruled out by existing methods.

To achieve this goal, we propose to solve (1) using the type-I (or "good") Anderson acceleration (AA-I) [FS09], a natural yet underdeveloped variant of the original Anderson acceleration (AA), also known as the type-II Anderson acceleration (AA-II) [And65]. Despite its

elegance in implementation, popularity in chemistry and physics, and success in specific optimization problems, a systematic treatment of AA, especially AA-I in optimization-related applications is still lacking. One of the main purposes of this work is thus to showcase the impressive numerical performance of AA-I on problems from these fields.

On the other hand, both early experiments in [FS09] and our preliminary benchmark tests of SCS 2.0 [OCPB17] show that although AA-I outperforms AA-II in many cases (matching its name of "good"), it also suffers more from instability. Moreover, few convergence analysis of AA and its variants (and none for AA-I) for general nonlinear problems exist in the literature, and the existing ones all require f to be continuously differentiable (which excludes most algorithms involving projections, and in general proximal operators), and are either local [GS78, RS11, TK15] or assume certain non-singularity (*e.g.*, contractivity) conditions [SdB16, SBd17, SOdB18b]. Another goal of this paper is hence to provide modifications that lead to a stabilized AA-I with convergencence beyond differentiability, locality and non-singularity. As a result, we obtain global convergence to a fixed-point with no additional assumptions apart from non-expansiveness.

We emphasize that our analysis does not provide a rate of convergence. While it would be nice to formally establish that our modified AA-I algorithm converges faster than vanilla KM, we do not do this in this paper. Instead, we show only that convergence occurs. The benefit of our method is not an improved theoretical convergence rate; it is instead (a) a formal proof that the method always converges, under very relaxed conditions, and (b) empirical studies that show that terminal convergence, especially to moderately high accuracies, is almost always much better than vanilla methods.

Related work. As its name suggests, AA is an acceleration algorithm proposed by D. G. Anderson in 1965 [And65]. The earliest problem that AA dealt with was nonlinear integral equations. Later, developed by another two different communities [Pul80, Pul82], AA has enjoyed wide application in material sciences and computational quantum chemistry for the computation of electronic structures, where it is also known as Pulay/Anderson mixing and (Pulay's) direct inversion iterative subspace (DIIS), respectively. In contrast, its name is quite unpopular in the optimization community. As far as we know, it was not until [FS09] connected it with Broyden's (quasi-Newton) methods, that some applications of AA to optimization algorithms, including expectation-maximization (EM), alternating nonnegative least-squares (ANNLS) and alternating projections (AP), emerged [WN11, HR18, HS16]. More recently, applications are further extended to machine learning and control problems, including K-means clustering [ZYP$^+$18], robot localization [POD$^+$17] and computer vision [SOdB18a].

There has been a rich literature on applications of AA to specific problems, especially within the field of computational chemistry and physics [WTK14, AJW17, AUM$^+$16, WPT15, BRZS18, MST$^+$17]. In the meantime, an emerging literature on applications to optimization-related problems is also witnessed in recent years, as mentioned above.

Nevertheless, theoretical analysis of AA and its variants is relatively underdeveloped, and most of the theory literature is focused on the full memory AA-II, *i.e.*, $m_k = k$ for all

$k \geq 0$ in Algorithm 1 below. This deviates from the original AA-II [And65] and in general most of the numerical and application literature, where limited memory AA is predominant. For solving general fixed-point problems (or equivalently, nonlinear equations), perhaps the most related work to ours is [GS78] and [RS11], among which the former proves local Q-superlinear convergence of a full-memory version of AA-I, while the latter establishes local Q-linear convergence for the original (limited memory) AA-II with the re-starting strategy introduced in [GS78], both presuming continuous differentiability of f in (1) around the solutions. A slightly more stabilized version of full-memory AA-I is introduced in [BF94] by generalizing the re-starting strategy in [GS78], which is then later globalized using a non-monotone line search method [LF00], assuming Lipschitz differentiability of f [BK17]. In practice, the generalized re-starting strategy is more computationally expensive, yet the performance improvement is non-obvious [BK17]. This motivates us to keep to the original re-starting strategy in [GS78] in §3.2. By assuming in addition contractivity of f, a slightly stronger and cleaner local linear convergence of the original AA-II can also be obtained [TK15]. A similar analysis for noise-corrupted f is later conducted in [TEE$^+$17].

Interestingly, the three papers on full-memory AA-I, which to our knowledge are the only papers analyzing the convergence of AA-I (variants) for general nonlinear fixed-point problems, are not aware of the literature stemming from [And65], and the algorithms there are termed as projected Broyden's methods. We will discuss the connection between AA and the Broyden's methods in §2 following a similar treatment as in [FS09, WN11, RS11], which also paves the way for the analysis of our modified (limited-memory) AA-I.

On the other hand, stronger results have been shown for more special cases. When f is restricted to affine mappings, finite-step convergence of full-memory AA is discussed by showing its essential equivalence to GMRES and Arnoldi method [WN11, PE13]. More recently, a regularized variant of full-memory AA-II is rediscovered as regularized nonlinear acceleration (RNA) in [SdB16], in which f is the gradient descent mapping of a strongly convex and strongly smooth real-valued function. Local linear convergence with improved rates similar to Nesterov's accelerated gradient descent is proved using a Chebyshev's acceleration argument. The algorithm is then extended to accelerate stochastic [SBd17], momentum-based [SOdB18b] and primal-dual [BSd18] algorithms.

We are not aware of any previous work on convergence of the limited memory AA-I or its variants, let alone global convergence in the absence of (Fréchet continuous) differentiability and non-singularity (or contractivity), which is missing from the entire AA literature.

Outline. In §2, we introduce the original AA-I [FS09], and discuss its relation to quasi-Newton methods. In §3, we propose a stabilized AA-I with Powell-type regularization, re-start checking and safe-guarding steps. A self-contained convergence analysis of the stabilized AA-I is given in §4. Finally, we demonstrate the effectiveness of our proposed algorithms with various numerical examples in §5. Extensions and variants to our results are discussed in §6, followed by a few conclusive remarks in §7.

1.1 Notation and definitions

We list some basic definitions and notation to be used in the rest of the paper. We denote the set of real numbers as $\mathbf{R}$; $\mathbf{R}_+$ the set of non-negative real numbers; $\bar{\mathbf{R}} = \mathbf{R} \cup \{+\infty\}$ is the extended real line, and $\mathbf{R}^n$ the n-dimensional Euclidean space equipped with the inner product $x^T y$ for $x, y \in \mathbf{R}^n$ and the ℓ_2-norm $\|\cdot\|_2$. For notational compactness, we will alternatively use

$$(x_1, \ldots, x_n) \quad \text{and} \quad \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

to denote a vector in $\mathbf{R}^n$.

The *proximal operator* of a convex, closed and proper function $F : \mathbf{R}^n \to \bar{\mathbf{R}}$ is given by

$$\mathrm{prox}_F(x) = \mathrm{argmin}_y \{F(y) + \tfrac{1}{2}\|y - x\|_2^2\}.$$

For a nonempty, closed and convex set $\mathcal{C} \subseteq \mathbf{R}^n$, the indicator function of $\mathcal{C}$ is denoted as

$$\mathcal{I}_{\mathcal{C}}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ +\infty & \text{otherwise.} \end{cases}$$

Similarly, we denote the *projection* on $\mathcal{C}$ as

$$\Pi_{\mathcal{C}}(x) = \mathrm{argmin}_{y \in \mathcal{C}} \|x - y\|_2,$$

and the *normal cone* of $\mathcal{C}$ as

$$N_{\mathcal{C}}(x) = \{y \in \mathbf{R}^n \mid \sup_{x' \in \mathcal{C}} y^T(x' - x) \leq 0\}.$$

The projection $\Pi_{\mathcal{C}}$ is the proximal operator of $\mathcal{I}_{\mathcal{C}}$.

A mapping $f : \mathbf{R}^n \to \mathbf{R}^n$ is said to be *non-expansive* if for all $x,\ y \in \mathbf{R}^n$,

$$\|f(x) - f(y)\|_2 \leq \|x - y\|_2.$$

It is said to be γ-*contractive* in an (arbitrary) norm $\|\cdot\|$ if for all $x,\ y \in \mathbf{R}^n$,

$$\|f(x) - f(y)\| \leq \gamma \|x - y\|.$$

A relation $G : \mathbf{R}^n \to 2^{\mathbf{R}^n}$ is said to be *monotone*, if for all $x,\ y \in \mathbf{R}^n$,

$$(u - v)^T (x - y) \geq 0 \ \text{ for all } \ u \in G(x),\ v \in G(y).$$

It is said to be *maximal monotone* if there is no monotone operator that properly contains it (as a relation, *i.e.*, subset of $\mathbf{R}^n \times \mathbf{R}^n$). We refer interested readers to [RB16] for a detailed explanation of relations. When a relation G is single-valued, it becomes a usual mapping from $\mathbf{R}^n$ to $\mathbf{R}^n$, and the same definition of (maximal) monotonicity holds.

For a matrix $A = (a_{ij})_{n\times n} \in \mathbf{R}^{n\times n}$, its ℓ_2-norm (or spectral/operator norm) is denoted as $\|A\|_2 = \sup_{\|x\|_2=1} \|Ax\|_2$. The Frobenius norm of A is denoted as $\|A\|_F = \sqrt{\sum_{i,j=1}^n a_{ij}^2}$. The spectral radius of a square matrix A is the maximum absolute value eigenvalue, *i.e.*,

$$\rho(A) = \max\{|\lambda_1|, \ldots, |\lambda_n|\},$$

where $\lambda_1, \ldots, \lambda_n$ are eigenvalues of A (with repetitions counted).

For a description of strong convexity and strong smoothness of a function $F : \mathbf{R}^n \to \mathbf{R}$, see [RB16]. They will only be used when it comes to the examples in §5.

2 Type-I Anderson acceleration

In this section we introduce the original AA-I, with a focus on its relation to quasi-Newton methods. Following the historical development from [And65] to [FS09], we naturally motivate it by beginning with a brief introduction to the original AA-II, making explicit its connection to the type-II Broyden's method, and then move on to AA-I as a natural counterpart of the type-I Broyden's method.

2.1 General framework of AA

As illustrated in the prototype Algorithm 1, the main idea is to maintain a memory of previous steps, and update the iteration as a linear combination of the memory with dynamic weights. It can be seen as a generalization of the KM iteration algorithm, where the latter uses only the most recent two steps, and the weights are pre-determined, which leads to sub-linear convergence for non-expansive mappings in general [RB16], and linear convergence under certain additional assumptions [BNP15].

Algorithm 1 Anderson Acceleration Prototype (AA)

1: **Input:** initial point x_0, fixed-point mapping $f : \mathbf{R}^n \to \mathbf{R}^n$.
2: **for** $k = 0, 1, \ldots$ **do**
3: Choose m_k (*e.g.*, $m_k = \min\{m, k\}$ for some integer $m \geq 0$).
4: Select weights α_j^k based on the last m_k iterations satisfying $\sum_{j=0}^{m_k} \alpha_j^k = 1$.
5: $x^{k+1} = \sum_{j=0}^{m_k} \alpha_j^k f(x^{k-m_k+j})$.
6: **end for**

The integer m_k is the memory in iteration k, since the next iterate is a linear combination of the images of the last k iterates under the map f. Based on the choices of the weights α_j^k in line 4 of Algorithm 1, AA is classified into two subclasses [FS09], namely AA-I and AA-II. The terminology indicates a close relationship between AA and quasi-Newton methods, as we will elaborate in more details below. While existing literature is mainly focused on AA-II, our focus is on the less explored AA-I.

2.2 The original AA: AA-II

Define the residual $g : \mathbf{R}^n \to \mathbf{R}^n$ of f to be $g(x) = x - f(x)$. In AA-II [And65], for each iteration $k \geq 0$, we solve the following least squares problem with a normalization constraint:

$$\begin{array}{ll} \text{minimize} & \| \sum_{j=0}^{m_k} \alpha_j g(x^{k-m_k+j}) \|_2^2 \\ \text{subject to} & \sum_{j=0}^{m_k} \alpha_j = 1, \end{array} \tag{2}$$

with variable $\alpha = (\alpha_0, \ldots, \alpha_{m_k})$. The weight vector $\alpha^k = (\alpha_0^k, \ldots, \alpha_{m_k}^k)$ in line 4 of Algorithm 1 is then chosen as the solution to (2). The intuition is to minimize the norm of the weighted residuals of the previous $m_k + 1$ iterates. In particular, when g is affine, it is not difficult to see that (2) directly finds a normalized weight vector α, minimizing the residual norm $\|g(x^{k+1/2})\|_2$ among all $x^{k+1/2}$ that can be represented as $x^{k+1/2} = \sum_{j=0}^{m_k} \alpha_j x^{k-m_k+j}$, from which $x^{k+1} = f(x^{k+1/2})$ is then computed with an additional fixed-point iteration in line 5 of Algorithm 1.

Connection to quasi-Newton methods. To reveal the connection between AA-II and quasi-Newton methods, we begin by noticing that the inner minimization subproblem (2) can be efficiently solved as an unconstrained least squares problem by a simple variable elimination [WN11]. More explicitly, we can reformulate (2) as follows:

$$\text{minimize} \quad \|g_k - Y_k \gamma\|_2, \tag{3}$$

with variable $\gamma = (\gamma_0, \ldots, \gamma_{m_k - 1})$. Here $g_i = g(x^i)$, $Y_k = [y_{k-m_k} \ \ldots \ y_{k-1}]$ with $y_i = g_{i+1} - g_i$ for each i, and α and γ are related by $\alpha_0 = \gamma_0$, $\alpha_i = \gamma_i - \gamma_{i-1}$ for $1 \leq i \leq m_k - 1$ and $\alpha_{m_k} = 1 - \gamma_{m_k - 1}$.

Assuming for now that Y_k is full column rank, the solution γ^k to (3) is given by $\gamma^k = (Y_k^T Y_k)^{-1} Y_k^T g_k$, and hence by the relation between α^k and γ^k, the next iterate of AA-II can be represented as

$$\begin{aligned} x^{k+1} &= f(x^k) - \sum_{i=0}^{m_k - 1} \gamma_i^k \left(f(x^{k-m_k+i+1}) - f(x^{k-m_k+i}) \right) \\ &= x^k - g_k - (S_k - Y_k)\gamma^k \\ &= x^k - (I + (S_k - Y_k)(Y_k^T Y_k)^{-1} Y_k^T) g_k \\ &= x^k - H_k g_k, \end{aligned}$$

where $S_k = [s_{k-m_k} \ \ldots \ s_{k-1}]$, $s_i = x^{i+1} - x^i$ for each i, and $H_k = I + (S_k - Y_k)(Y_k^T Y_k)^{-1} Y_k^T$. It has been observed that H_k minimizes $\|H_k - I\|_F$ subject to the inverse multi-secant condition $H_k Y_k = S_k$ [FS09, WN11], and hence can be regarded as an approximate inverse Jacobian of g. The update of x^k can then be considered as a quasi-Newton-type update, with H_k being some sort of generalized second (or type-II) Broyden's update [Bro65] of I satisfying the inverse multi-secant condition.

It's worth noticing that a close variant of AA-II, with an additional non-negative constraint $\alpha \geq 0$, is also widely used to accelerate the SCF (self-consistent field) iteration in the electronic structure computation. Such methods are typically referred to as "energy DIIS" in literature [HY10]. However, the inner minimization problem of energy DIIS has to be solved as a generic convex quadratic program. Moreover, our preliminary experiments suggest that it may not work as well for many optimization algorithms (*e.g.*, SCS).

2.3 AA-I

In the quasi-Newton literature, the type-II Broyden's update is often termed as the "bad Broyden's method". In comparison, the so-called "good Broyden's method", or type-I Broyden's method, which directly approximates the Jacobian of g, typically seems to yield better numerical performance [Gri12].

In the same spirit, we define the type-I AA (AA-I) [FS09], in which we find an approximate Jacobian of g minimizing $\|B_k - I\|_F$ subject to the multi-secant condition $B_k S_k = Y_k$. Assuming for now that S_k is full column rank, we obtain (by symmetry) that

$$B_k = I + (Y_k - S_k)(S_k^T S_k)^{-1} S_k^T, \tag{4}$$

and the update scheme is defined as

$$x^{k+1} = x^k - B_k^{-1} g_k, \tag{5}$$

assuming B_k to be invertible. We will deal with the potential rank deficiency of S_k and singularity of B_k shortly in the next sections.

A direct application of Woodbury matrix identity shows that

$$B_k^{-1} = I + (S_k - Y_k)(S_k^T Y_k)^{-1} S_k^T, \tag{6}$$

where again we have assumed for now that $S_k^T Y_k$ is invertible. Notice that this explicit formula of B_k^{-1} is preferred in that the most costly step, inversion, is implemented only on a small $m_k \times m_k$ matrix.

Backtracking the derivation in AA-II, (5) can be rewritten as

$$x^{k+1} = x^k - g_k - (S_k - Y_k)\tilde{\gamma}^k = f(x^k) - \sum_{i=0}^{m_k - 1} \tilde{\gamma}_i^k \left(f(x^{k-m_k+i+1}) - f(x^{k-m_k+i}) \right), \tag{7}$$

where $\tilde{\gamma}^k = (S_k^T Y_k)^{-1} S_k^T g_k$. Now we can see how AA-I falls into the framework of Algorithm 1: here the weight vector α^k in line 4 is defined as $\alpha_0^k = \tilde{\gamma}_0^k$, $\alpha_i^k = \tilde{\gamma}_i^k - \tilde{\gamma}_{i-1}^k$ for $1 \leq i \leq m_k - 1$ and $\alpha_{m_k}^k = 1 - \tilde{\gamma}_{m_k-1}^k$. Note that although not as intuitive as the weight vector choice in AA-II, the computational complexity is exactly the same whenever matrix-vector multiplication is done prior to matrix-matrix multiplication.

For easier reference, we detail AA-I in the following Algorithm 2. As our focus is on the more numerically efficient limited-memory versions, we also specify a maximum-memory parameter m in the algorithm.

Algorithm 2 Type-I Anderson Acceleration (AA-I-m)

Input: initial point x_0, fixed-point mapping $f : \mathbf{R}^n \to \mathbf{R}^n$, max-memory $m > 0$.
for $k = 0, 1, \ldots$ **do**
  Choose $m_k \le m$ (*e.g.*, $m_k = \min\{m, k\}$ for some integer $m \ge 0$).
  Compute $\tilde{\gamma}^k = (S_k^T Y_k)^{-1}(S_k^T g_k)$.
  Compute $\alpha_0^k = \tilde{\gamma}_0^k$, $\alpha_i^k = \tilde{\gamma}_i^k - \tilde{\gamma}_{i-1}^k$ for $1 \le i \le m_k - 1$ and $\alpha_{m_k}^k = 1 - \tilde{\gamma}_{m_k-1}^k$.
  $x^{k+1} = \sum_{j=0}^{m_k} \alpha_j^k f(x^{k-m_k+j})$.
end for

Note that in the above algorithm, the iteration may get stuck or suffer from ill-conditioning if B_k, or equivalently either S_k or Y_k is (approximately) rank-deficient. This is also a major source of numerical instability in AA-I. We will solve this issue in the next section.

3 Stabilized type-I Anderson acceleration

In this section, we propose several modifications to the vanilla AA-I (Algorithm 2) to stabilize its convergence. We begin by introducing a Powell-type regularization to ensure the non-singularity of B_k. We then introduce a simple re-start checking strategy that ensures certain strong linear independence of the updates s_k. These together solve the stagnation problem mentioned at the end of the last section. Finally, we introduce safe-guarding steps that check the decrease in the residual norm, with which the modifications altogether lead to global convergence to a solution of (1), as we will show in §4.

Rank-one update. To motivate the modifications, we take a step back to the update formula (5) and formalize a closer connection between AA-I and the type-I Broyden's method in terms of rank-one update. The counterpart result has been proved for AA-II in [RS11].

Proposition 1. *Suppose that S_k is full rank, then B_k in (4) can be computed inductively from $B_k^0 = I$ as follows:*

$$B_k^{i+1} = B_k^i + \frac{(y_{k-m_k+i} - B_k^i s_{k-m_k+i})\hat{s}_{k-m_k+i}^T}{\hat{s}_{k-m_k+i}^T s_{k-m_k+i}}, \quad i = 0, \ldots, m_k - 1 \tag{8}$$

with $B_k = B_k^{m_k}$. Here $\{\hat{s}_i\}_{i=k-m_k}^{k-1}$ is the Gram-Schmidt orthogonalization of $\{s_i\}_{i=k-m_k}^{k-1}$, i.e.,

$$\hat{s}_i = s_i - \sum_{j=k-m_k}^{i-1} \frac{\hat{s}_j^T s_i}{\hat{s}_j^T \hat{s}_j} \hat{s}_j, \quad i = k - m_k, \ldots, k - 1. \tag{9}$$

We remark that another similar rank-one update formula for full-memory AA-I is presented in [GS78]. However, the result there corresponds to successive minimization of $\|B_k^{i+1} - B_k^i\|_F$ with the multi-secant constraints, instead of the direct minimization of $\|B_k - I\|_F$. It's thus non-obvious how we can apply their result here, and we instead provide a self-contained proof in the Appendix. The basic idea is to prove by induction, and to fix B_k by its restrictions to $\mathrm{span}(S_k)$ and its orthogonal complement, respectively.

3.1 Powell-type regularization

To fix the potential singularity of B_k, we introduce a Powell-type regularization to the rank-one update formula (8). The idea is to specify a parameter $\bar{\theta} \in (0, 1)$, and simply replace y_{k-m_k+i} in (8) with

$$\tilde{y}_{k-m_k+i} = \theta_k^i y_{k-m_k+i} + (1 - \theta_k^i) B_k^i s_{k-m_k+i}, \tag{10}$$

where $\theta_k^i = \phi_{\bar{\theta}}(\eta_k^i)$ is defined with

$$\phi_{\bar{\theta}}(\eta) = \begin{cases} 1 & \text{if } |\eta| \geq \bar{\theta} \\ \frac{1 - \mathbf{sign}(\eta)\bar{\theta}}{1 - \eta} & \text{if } |\eta| < \bar{\theta} \end{cases} \tag{11}$$

and $\eta_k^i = \frac{\hat{s}_{k-m_k+i}^T (B_k^i)^{-1} y_{k-m_k+i}}{\|\hat{s}_{k-m_k+i}\|_2^2}$. Here we adopt the convention that $\mathbf{sign}(0) = 1$. The formulation is almost the same as the original Powell's trick used in [Pow70], but we redefine η_k to take the orthogonalization into considerations. Similar ideas have also been introduced in [SdB16] and [HR18] by adding a Levenberg-Marquardt-type regularization. However, such tricks are designed for stabilizing least-squares problems in AA-II, which are not applicable here.

We remark that the update remains unmodified when $\bar{\theta} = 0$. On the other hand, when $\bar{\theta} = 1$, (5) reduces to the vanilla fixed-point iteration associated with (1). Hence $\bar{\theta}$ serves as a bridge between the two extremes. By definition, we immediately see that $\theta_k^i \in [1-\bar{\theta}, 1+\bar{\theta}]$, which turns out to be a useful bound in the subsequent derivations.

The following lemma establishes the non-singularity of the modified B_k, which also indicates how $\bar{\theta}$ trades off between stability and efficiency.

Lemma 2. *Suppose $\{s_i\}_{i=k-m_k}^{k-1}$ to be an arbitrary sequence in $\mathbf{R}^n$. Define $B_k = B_k^{m_k}$ inductively from $B_k^0 = I$ as*

$$B_k^{i+1} = B_k^i + \frac{(\tilde{y}_{k-m_k+i} - B_k^i s_{k-m_k+i}) \hat{s}_{k-m_k+i}^T}{\hat{s}_{k-m_k+i}^T s_{k-m_k+i}}, \quad i = 0, \ldots, m_k - 1, \tag{12}$$

with $\hat{s}_{k-m_k+i}$ and $\tilde{y}_{k-m_k+i}$ defined as in (9) and (10), respectively. Suppose that the updates above are all well-defined. Then $|det(B_k)| \geq \bar{\theta}^{m_k} > 0$, and in particular, B_k is invertible.

Proof. We prove by induction that $|\det(B_k^i)| \geq \bar{\theta}^i$. The base case when $i = 0$ is trivial. Now suppose that we have proved the claim for B_k^i. By Sylvester's determinant identity, we have

$$\begin{aligned} |\det(B_k^{i+1})| &= |\det(B_k^i)| \left| \det\left(I + \theta_k^i \frac{((B_k^i)^{-1} y_{k-m_k+i} - s_{k-m_k+i}) \hat{s}_{k-m_k+i}^T}{\hat{s}_{k-m_k+i}^T s_{k-m_k+i}} \right) \right| \\ &= |\det(B_k^i)| \left| 1 + \theta_k^i \frac{\hat{s}_{k-m_k+i}^T ((B_k^i)^{-1} y_{k-m_k+i} - s_{k-m_k+i})}{\hat{s}_{k-m_k+i}^T s_{k-m_k+i}} \right| \\ &= |\det(B_k^i)| \left| 1 - \theta_k^i (1 - \eta_k^i) \right| \geq \bar{\theta}^i \cdot \begin{cases} |\eta_k^i|, & |\eta_k^i| \geq \bar{\theta} \\ |\mathbf{sign}(\eta_k^i)\bar{\theta}|, & |\eta_k^i| < \bar{\theta} \end{cases} \geq \bar{\theta}^{i+1}. \end{aligned}$$

By induction, this completes our proof. □

Now that we have established the non-singularity of the modified B_k, defining $H_k = B_k^{-1}$, we can directly update $H_k = H_k^{m_k}$ from $H_k^0 = I$ as follows:

$$H_k^{i+1} = H_k^i + \frac{(s_{k-m_k+i} - H_k^i \tilde{y}_{k-m_k+i})\hat{s}_{k-m_k+i}^T H_k^i}{\hat{s}_{k-m_k+i}^T H_k^i \tilde{y}_{k-m_k+i}}, \quad i = 0, \ldots, m_k - 1, \tag{13}$$

again with $\hat{s}_{k-m_k+i}$ and $\tilde{y}_{k-m_k+i}$ defined as in (9) and (10), respectively. This can be easily seen by a direct application of the Sherman-Morrison formula. Notice that the H_k hereafter is different from the one in §2.2 for AA-II.

It's worth pointing out that [BK17] also considers selecting an appropriate θ_k^i in (10) to ensure the non-singularity of B_k, but an explicit choice of θ_k^i is not provided to guarantee its existence. Moreover, apart from the well-defined-ness of the iterations, the modification is neither needed in the proof, nor in the smooth numerical examples there as claimed by the authors. In contrast, in our general non-smooth settings the modification is both significant in theory and practice, as we will see below.

3.2 Re-start checking

In this section, we introduce a re-start checking strategy proposed in [GS78], and use it to establish uniform bounds on the approximate (inverse) Jacobians, which turns out to be essential to the final global convergence, as we will see in §4.

Notice that the update formula (12) is well-defined as long as $\hat{s}_{k-m_k+i} \neq 0$, in which case the denominator $\hat{s}_{k-m_k+i}^T s_{k-m_k+i} = \|\hat{s}_{k-m_k+i}\|_2^2 > 0$. However, unless $g_{k-m_k+i} = 0$ for some $i = 0, \ldots, m_k - 1$, in which case the problem is already solved, we will always have

$$s_{k-m_k+i} = -B_{k-m_k+i}^{-1} g_{k-m_k+i} \neq 0,$$

where we used Lemma 2 to deduce that B_{k-m_k+i} is invertible.

This means that the only case when the updates in (12) break down is $s_{k-m_k+i} \neq 0$ while $\hat{s}_{k-m_k+i} = 0$. Unfortunately, such a scenario is indeed possible if m_k is chosen as $\min\{m, k\}$ for some fixed $1 \leq m \leq \infty$ (with $m = \infty$ usually called "full"-memory), a fixed-memory strategy most commonly used in the literature. In particular, when m is greater than the problem dimension n, we will always have $\hat{s}_k = 0$ for $k > n$ due to linear dependence.

To address this issue, we enforce a re-start checking step that clears the memory immediately before the algorithm is close to stagnation. More explicitly, we keep m_k growing, until either $m_k = m + 1$ for some integer $1 \leq m < \infty$ or $\|\hat{s}_{k-1}\|_2 < \tau \|s_{k-1}\|_2$, in which case m_k is reset to 1 (*i.e.*, no orthogonalization). The process is then repeated. Formally, the following rule is adopted to select m_k in each iteration $k \geq 0$, initialized from $m_0 = 0$:

$$\text{Update } m_k = m_{k-1} + 1. \text{ If } m_k = m + 1 \text{ or } \|\hat{s}_{k-1}\|_2 < \tau \|s_{k-1}\|_2, \text{ then reset } m_k = 1. \tag{14}$$

Here $\tau \in (0, 1)$ is pre-specified. The main idea is to make sure that $\hat{s}_k \neq 0$ whenever s_k is so, which ensures that the modified updates (12) and (13) won't break down before reaching a solution. We actually require a bit more by imposing a positive parameter τ, which characterizes a strong linear independence between s_k and the previous updates. This leads to boundedness of B_k, as described in the following lemma.

Lemma 3. *Assume the same conditions as in Lemma 2, and in addition that m_k is chosen by rule (14). Then we have $\|B_k\|_2 \leq 3(1+\bar{\theta}+\tau)^m/\tau^m - 2$ for all $k \geq 0$.*

Proof. Notice that by rule (14), we have $\|\hat{s}_k\|_2 \geq \tau\|s_k\|_2$ and $m_k \leq m$ for all $k \geq 0$. Hence by (12), we have that

$$\begin{aligned}\|B_k^{i+1}\|_2 &\leq \|B_k^i\|_2 + \theta_k^i \frac{\|y_{k-m_k+i} - B_k^i s_{k-m_k+i}\|_2}{\|\hat{s}_{k-m_k+i}\|_2} \\ &\leq \|B_k^i\|_2 + \frac{1+\bar{\theta}}{\tau}\frac{\|y_{k-m_k+i} - B_k^i s_{k-m_k+i}\|_2}{\|s_{k-m_k+i}\|_2}.\end{aligned}$$

Noticing that $y_{k-m_k+i} = g(x^{k-m_k+i+1}) - g(x^{k-m_k+i})$ and that $f(x)(= x - g(x))$ is non-expansive, we see that

$$\|B_k^{i+1}\|_2 \leq \frac{1+\bar{\theta}+\tau}{\tau}\|B_k^i\|_2 + \frac{2(1+\bar{\theta})}{\tau},$$

and hence by telescoping the above inequality and the fact that $\|B_k^0\|_2 = 1$, we conclude that

$$\|B_k\|_2 = \|B_k^{m_k}\|_2 \leq 3\left(\frac{1+\bar{\theta}+\tau}{\tau}\right)^m - 2.$$

This completes our proof. □

In sum, combining the modified updates with the re-starting choice of m_k, the rank-deficiency problem mentioned at the end of §2.3 is completely resolved. In particular, the full-rank assumption on S_k is no longer necessary. Moreover, the inverse $H_k = B_k^{-1}$ is also bounded, as described in the following corollary.

Corollary 4. *Under the same assumptions in Lemma 3, we have for all $k \geq 0$ that*

$$\|H_k\|_2 \leq \left(3\left(\frac{1+\bar{\theta}+\tau}{\tau}\right)^m - 2\right)^{n-1}/\bar{\theta}^m. \tag{15}$$

Proof. Denote the singular values of B_k as $\sigma_1 \geq \cdots \geq \sigma_n$. Then by Lemma 2, we have $\prod_{i=1}^n \sigma_i \geq \bar{\theta}^{m_k} \geq \bar{\theta}^m$. On the other hand, by Lemma 3, we have $\sigma_1 \leq 3(1+\bar{\theta}+\tau)^m/\tau^m - 2$. Hence we obtain that

$$\|H_k\|_2 = 1/\sigma_n \leq \prod_{i=1}^{n-1}\sigma_i/\bar{\theta}^m \leq \left(3\left(\frac{1+\bar{\theta}+\tau}{\tau}\right)^m - 2\right)^{n-1}/\bar{\theta}^m,$$

which finishes our proof. □

We remark that for type-II methods as in [RS11], the algorithm can already get stuck if $g(x^{k+1}) = g(x^k)$, which is not informative enough for us to say anything. That's also one of the reasons for favoring the type-I AA in this paper. It's also worth mentioning that

empirical results in [PS15] and [HR18] have already suggested that cleaning memories from time to time improves performance significantly for self-consistent field (SCF) methods and EM-type algorithms, partially supporting our modification here.

Notice that when m_k is chosen by rule (14) and B_k is computed as in Lemma 2, we have $B_k^i = B_{k-m_k+i}$. This means that in iteration k, only a rank-one update (12) with $i = m_k - 1$ is needed, which yields $B_k = B_k^{m_k}$ from $B_{k-1} = B_k^{m_k-1}$. Moreover, we can remove the necessity of maintaining updates for B_k^i used in Powell's regularization by noticing that $B_k^i s_{k-m_k+i} = B_{k-m_k+i} s_{k-m_k+i} = -B_{k-m_k+i} B_{k-m_k+i}^{-1} g_{k-m_k+i} = -g_{k-m_k+i}$.

3.3 Safe-guarding steps

We are now ready to introduce the final piece for our modified AA-I algorithm. The main idea is to interleave AA-I steps with the vanilla KM iteration steps to safe-guard the decrease in residual norms g. In particular, we check if the current residual norm is sufficiently small, and replace it with the α-averaged (or KM) operator of f in (1) (defined as $f_\alpha(x) = (1-\alpha)x + \alpha f(x)$) whenever not.

The idea of interleaving AA with vanilla iterations has also been considered in [BSP16] with constant periods, and is observed to improve both accuracy and speed for a certain class of algorithms (*e.g.*, SCF), despite that no theoretical guarantees for convergence is provided. Similar ideas have been applied to regularized AA [SdB16] and the classical Broyden's methods [TP16] to seek for smaller per-iteration costs without sacrificing much the acceleration effects.

We remark that our safeguarding condition is closely related to the line search condition described in §5 of [GFB16], which guarantees global convergence of generic algorithms by checking the residual norm of the next candidate iteration. The major difference here is that we only check the residual norm of the current iteration, which leads to more relaxed safe-guards at the price of certain boundedness requirements as guaranteed by Lemma 3 and Corollary 4. A by-product is that our approach saves one potential extra fixed-point mapping evaluation in each iteration.

It is worth noticing that the SuperMann framework in [TP16] can also be seen as a general globalization strategy for several other acceleration schemes (apart from the basic Broyden's methods), although again an assumption (*cf.* Assumption II there) on certain boundedness related to Lemma 3 and Corollary 4 in this paper has to be made.

The resulting algorithm, combining all the aforementioned tricks, is summarized as Algorithm 3. Here, lines 4-8 perform re-start checking (rule (14)) described in §3.2, lines 9-11 perform the Powell-type regularization (update (12)) described in §3.1, and lines 12-14 execute the safe-guarding strategy described above. As mentioned at the end of §3.2, only a rank-one update of (12) from $i = m_k - 1$ is performed in iteration k, in which case the subscript $k - m_k + i$ becomes $k - 1$.

Notice that in line 5 of Algorithm 3, instead of defining $s_{k-1} = x^k - x^{k-1}$ and $y_{k-1} = g(x^k) - g(x^{k-1})$ as in §2.3, we redefine it using the AA-I trial update $\tilde{x}^k$ to ensure that $B_{k-1}s_{k-1} = -B_{k-1}B_{k-1}^{-1}g_{k-1} = -g_{k-1}$ still holds as mentioned at the end of §3.2, which makes it possible to get rid of maintaining an update for B_k.

Algorithm 3 Stablized Type-I Anderson Acceleration (AA-I-S-m)

1: **Input:** initial point x_0, fixed-point mapping $f : \mathbf{R}^n \to \mathbf{R}^n$, regularization constants $\bar{\theta},\ \tau,\ \alpha \in (0,1)$, safe-guarding constants $D,\ \epsilon > 0$, max-memory $m > 0$.
2: Initialize $H_0 = I,\ m_0 = n_{AA} = 0,\ \bar{U} = \|g_0\|_2$, and compute $x^1 = \tilde{x}^1 = f_\alpha(x^0)$.
3: **for** $k = 1,\ 2,\ \ldots$ **do**
4: $\quad m_k = m_{k-1} + 1$.
5: $\quad$ Compute $s_{k-1} = \tilde{x}^k - x^{k-1}$, $y_{k-1} = g(\tilde{x}^k) - g(x^{k-1})$.
6: $\quad$ Compute $\hat{s}_{k-1} = s_{k-1} - \sum_{j=k-m_k}^{k-2} \frac{\hat{s}_j^T s_{k-1}}{\hat{s}_j^T \hat{s}_j} \hat{s}_j$.
7: $\quad$ **If** $m_k = m + 1$ or $\|\hat{s}_{k-1}\|_2 < \tau \|s_{k-1}\|_2$
8: $\qquad$ reset $m_k = 1$, $\hat{s}_{k-1} = s_{k-1}$, and $H_{k-1} = I$.
9: $\quad$ Compute $\tilde{y}_{k-1} = \theta_{k-1} y_{k-1} - (1 - \theta_{k-1}) g_{k-1}$
10: $\qquad$ with $\theta_{k-1} = \phi_{\bar{\theta}}(\gamma_{k-1})$ and $\gamma_{k-1} = \hat{s}_{k-1}^T H_{k-1} y_{k-1} / \|\hat{s}_{k-1}\|^2$.
11: $\quad$ Update $H_k = H_{k-1} + \frac{(s_{k-1} - H_{k-1}\tilde{y}_{k-1})\hat{s}_{k-1}^T H_{k-1}}{\hat{s}_{k-1}^T H_{k-1} \tilde{y}_{k-1}}$, and $\tilde{x}^{k+1} = x^k - H_k g_k$.
12: $\quad$ **If** $\|g_k\| \le D\bar{U}(n_{AA} + 1)^{-(1+\epsilon)}$
13: $\qquad$ $x^{k+1} = \tilde{x}^{k+1}$, $n_{AA} = n_{AA} + 1$.
14: $\quad$ **else** $x^{k+1} = f_\alpha(x^k)$.
15: **end for**

We remark that the assumptions in Lemma 2, Lemma 3 and Corollary 4 all hold for Algorithm 3 unless a solution is reached and the problem is solved, despite that the updates are modified in line 4 and the safe-guarding strategy is introduced in lines 12-14. This comes immediately from the arbitrariness of the update sequence $\{s_i\}_{i=k-m_k}^{k-1}$ (*cf.* Lemma 2). Formally, we have the following corollary.

Corollary 5. *In Algorithm 3, unless a solution to (1) is found in finite steps, the inequality (15) holds for all $k \ge 0$, and the condition number of H_k is uniformly bounded by*

$$\boldsymbol{cond}(H_k) \le \left(3 \left(\frac{1 + \bar{\theta} + \tau}{\tau} \right)^m - 2 \right)^n / \bar{\theta}^m.$$

The proof is a simple combination of the results in Lemma 3 and Corollary 4.

4 Analysis of global convergence

In this section, we give a self-contained proof for global convergence of Algorithm 3. We temporarily assume for simplicity that a solution to (1) is *not* found in finite steps. The proof can be divided into three steps. Firstly, we prove that the residual g_k converges to 0. We then show that $\|x^k - y\|_2$ converges to some finite limit for any fixed point $y \in X$ of f. Finally, we we show that x^k converges to some solution to (1). We note that many of the arguments are adapted from the proofs in [Com01].

We begin by noticing that x^{k+1} either equals $x^k - H_k g_k$ or $f_\alpha(x^k)$, depending on whether the checking in line 12 of Algorithm 3 passes or not. We partition the iteration counts into

two subsets accordingly, with $K_{AA} = \{k_0, k_1, \dots\}$ being those iterations that passes line 12, while $K_{KM} = \{l_0, l_1, \dots\}$ being the rest that goes to line 14.

Step 1: Convergence of g_k. Consider $y \in X$ an arbitrary fixed point of f.

For $k_i \in K_{AA}$ $(i \geq 0)$, by Corollary 5, we have $\|H_{k_i}\|_2 \leq C$ for some constant C independent of the iteration count, and hence

$$\begin{aligned} \|x^{k_i+1} - y\|_2 &\leq \|x^{k_i} - y\|_2 + \|H_{k_i} g_{k_i}\|_2 \\ &\leq \|x^{k_i} - y\|_2 + C\|g_{k_i}\|_2 \leq \|x^{k_i} - y\|_2 + CD\bar{U}(i+1)^{-(1+\epsilon)}. \end{aligned} \tag{16}$$

For $l_i \in K_{KM}$ $(i \geq 0)$, since f is non-expansive, by Theorem 4.25(iii) in [BC10] or inequality (5) in [RB16], we have that

$$\|x^{l_i+1} - y\|_2^2 \leq \|x^{l_i} - y\|_2^2 - \alpha(1-\alpha)\|g_{l_i}\|_2^2 \leq \|x^{l_i} - y\|_2^2. \tag{17}$$

By telescoping the above inequalities, we obtain that

$$\|x^k - y\|_2 \leq \|x^0 - y\|_2 + CD\bar{U}\sum\nolimits_{i=0}^{\infty}(i+1)^{-(1+\epsilon)} = E < \infty, \tag{18}$$

and hence $\|x^k - y\|_2$ remains bounded for all $k \geq 0$.

Hence by squaring both sides of (16), we obtain that

$$\|x^{k_i+1} - y\|_2^2 \leq \|x^{k_i} - y\|_2^2 + \underbrace{(CD\bar{U})^2(i+1)^{-(2+2\epsilon)} + 2CDE\bar{U}(i+1)^{-(1+\epsilon)}}_{=\epsilon_{k_i}}. \tag{19}$$

Combining (17) and (19), we see that

$$\alpha(1-\alpha)\sum\nolimits_{i=0}^{\infty}\|g_{l_i}\|_2^2 \leq \|x^0 - y\|_2^2 + \sum\nolimits_{i=0}^{\infty}\epsilon_{k_i} < \infty, \tag{20}$$

and hence $\lim_{i\to\infty}\|g_{l_i}\|_2 = 0$. Noticing that $\|g_{k_i}\|_2 \leq D\bar{U}(i+1)^{-(1+\epsilon)}$ by line 12 of Algorithm 3, we also have $\lim_{i\to\infty}\|g_{k_i}\|_2 = 0$. Hence we see that

$$\lim_{k\to\infty}\|g_k\|_2 = 0. \tag{21}$$

Also notice that by defining $\epsilon_{l_i} = 0$, we again see from (17) and (19) that

$$\|x^{k+1} - y\|_2^2 \leq \|x^k - y\|_2^2 + \epsilon_k, \tag{22}$$

with $\epsilon_k \geq 0$ and $\sum_{k=0}^{\infty}\epsilon_k = \sum_{i=0}^{\infty}\epsilon_{k_i} < \infty$.

Notice that in the above derivation of (20)-(22), we have implicitly assumed that both K_{AA} and K_{KM} are infinite. However, the cases when either of them is finite is even simpler as one can completely ignore the finite index set.

Step 2: Convergence of $\|x^k-y\|_2$. Still consider $y \in X$ an arbitrary fixed point of f. We now prove that $\|x^k - y\|_2$ converges. Since $\|x^k - y\|_2 \geq 0$, there is a subsequence $\{j_0, j_1, \dots\}$ such that $\lim_{i\to\infty} \|x^{j_i} - y\|_2 = \underline{u} = \liminf_{k\to\infty} \|x^k - y\|_2$. For any $\delta > 0$, there exists an integer i_0 such that $\|x^{j_{i_0}} - y\|_2 \leq \underline{u} + \delta$ and $\sum_{k=j_{i_0}}^{\infty} \epsilon_k \leq \delta$. This, together with (22), implies that for any $k \geq j_{i_0}$,

$$\|x^k - y\|_2^2 \leq \|x^{j_{i_0}} - y\|_2^2 + \sum\nolimits_{k=j_{i_0}}^{\infty} \epsilon_k \leq \underline{u}^2 + 2\delta\underline{u} + \delta^2 + \delta, \tag{23}$$

and in particular, we have $\limsup_{k\to\infty} \|x^k - y\|_2^2 \leq \liminf_{k\to\infty} \|x^k - y\|_2^2 + \delta(2\underline{u} + \delta + 1)$. By the arbitrariness of $\delta > 0$, we see that $\|x^k - y\|_2^2$ (and hence $\|x^k - y\|_2$) is convergent.

Step 3: Convergence of x^k. Finally, we show that x^k converges to some solution $x^\star$ of (1), *i.e.*, $x^\star = f(x^\star)$. To see this, notice that since $\|x^k - y\|_2$ is bounded for $y \in X$, x^k is also bounded. Hence it must have a convergent subsequence by Weierstrass theorem.

Suppose on the contrary that x^k is not convergent, then there must be at least two different subsequences $\{k'_0, k'_1, \dots\}$ and $\{l'_0, l'_1, \dots\}$ converging to two different limits $y_1 \neq y_2$, both of which must be fixed points of f. This is because that by (21), we have

$$0 = \lim_{i\to\infty} \|g(x^{k'_i})\|_2 = \|g(y_1)\|_2, \quad 0 = \lim_{i\to\infty} \|g(x^{l'_i})\|_2 = \|g(y_2)\|_2,$$

where we used the fact that f is non-expansive and hence $g(x) = x - f(x)$ is (Lipschitz) continuous. Now notice that we have proved that $\alpha(y) = \lim_{k\to\infty} \|x^k - y\|_2$ exists for any $y \in X$. By the simple fact that $\|x^k - y\|_2^2 - \|y\|_2^2 = \|x^k\|_2^2 - 2y^T x^k$, we have

$$\begin{aligned}
\lim_{i\to\infty} \|x^{k'_i}\|_2^2 &= \lim_{k\to\infty} \|x^k - y\|_2^2 - \|y\|_2^2 + 2y^T \lim_{i\to\infty} x^{k'_i} = \alpha(y) - \|y\|_2^2 + 2y^T y_1, \\
\lim_{i\to\infty} \|x^{l'_i}\|_2^2 &= \lim_{k\to\infty} \|x^k - y\|_2^2 - \|y\|_2^2 + 2y^T \lim_{i\to\infty} x^{l'_i} = \alpha(y) - \|y\|_2^2 + 2y^T y_2.
\end{aligned}$$

Subtracting the above inequalities, we obtain that for any $y \in X$,

$$2y^T(y_1 - y_2) = \lim_{i\to\infty} \|x^{k'_i}\|_2^2 - \lim_{i\to\infty} \|x^{l'_i}\|_2^2.$$

By taking $y = y_1$ and $y = y_2$, we see that $y_1^T(y_1 - y_2) = y_2^T(y_1 - y_2)$, which implies that $y_1 = y_2$, a contradiction. Hence we conclude that x^k converges to some $\bar{x}$, which must be a solution as we have by (21) that $0 = \lim_{k\to\infty} \|g(x^k)\|_2 = \|g(\bar{x})\|_2$. Here we again used the fact that f is non-expansive, and hence $g(x) = x - f(x)$ is (Lipschitz) continuous.

In sum, together with the case that a fixed-point solution is found in finite steps, we have the following theorem.

Theorem 6. *Suppose that $\{x^k\}_{k=0}^{\infty}$ is generated by Algorithm 3, then we have $\lim_{k\to\infty} x^k = x^\star$, where $x^\star = f(x^\star)$ is a solution to (1).*

Remark 1. *Most part of steps 2 and 3 can be replaced by directly invoking Theorem 3.8 in [Com01]. The details are shown to make the proof self-contained and to facilitate the discussions about extensions in §6.*

5 Numerical results

In this section we present examples to demonstrate the power of AA-I. The major focus is on optimization problems and algorithms, where f in (1) comes from the iterative algorithms used to solve them. For each example, we specify the concrete form of f, verify its non-expansiveness, and check the equivalence between the fixed-point problem and the original problem.

We compare the performance of the following three algorithms for each experiment:

- The vanilla algorithm: *e.g.*, gradient descent;
- AA-I-m: Algorithm 2 with max-memory m, choosing $m_k = \min\{m, k\}$;
- AA-I-S-m: Algorithm 3 with max-memory m.

The convergence curves against both clock time (seconds) and iteration numbers will be shown.

5.1 Problems and algorithms

We begin by describing some example problems and the corresponding (unaccelerated) algorithms used to solve them.

5.1.1 Proximal gradient descent

Consider the following problem:

$$\text{minimize} \quad F_1(x) + F_2(x), \tag{24}$$

where $F_1,\ F_2 : \mathbf{R}^n \to \mathbf{R}$ are convex closed proper (CCP), and F_1 is L-strongly smooth.

We solve it using proximal gradient descent, *i.e.*,

$$x^{k+1} = \text{prox}_{\alpha F_2}(x^k - \alpha \nabla F_1(x^k)),$$

where $\alpha \in (0, 2/L)$. In our notation, the fixed-point mapping is $f(x) = \text{prox}_{\alpha F_2}(x - \alpha \nabla F_1(x))$. For a proof of non-expansiveness for f and the equivalence between the fixed-point problem and the original optimization problem (24), see [PB14].

Gradient descent (GD). When $F_2 = 0$, proximal gradient descent reduces to vanilla gradient descent for unconstrained problems, *i.e.*, (denoting $F = F_1$)

$$x^{k+1} = x^k - \alpha \nabla F(x^k),$$

where $\alpha \in (0, 2/L)$, and the fixed-point mapping is $f(x) = x - \alpha \nabla F(x)$.

Projected gradient descent (PGD). When $F_2(x) = \mathcal{I}_{\mathcal{K}}(x)$, with $\mathcal{K}$ being a nonempty closed and convex set, problem (24) reduces to a constrained optimization problem. Accordingly, proximal gradient descent reduces to projected gradient descent, *i.e.*, (denoting $F = F_1$)

$$x^{k+1} = \Pi_{\mathcal{K}}(x^k - \alpha \nabla F(x^k)),$$

where $\alpha \in (0, 2/L)$, and the fixed-point mapping is $f(x) = \Pi_{\mathcal{K}}(x - \alpha \nabla F(x))$.

Alternating projection (AP). When $F_1(x) = \frac{1}{2}\mathrm{dist}(x, D)^2$ and $F_2(x) = \mathcal{I}_C(x)$, with $C,\ D$ being nonempty closed convex sets and $C \cap D \neq \emptyset$. The problem (24) then reduces to finding an element x in the intersection $C \cap D$. Noticing that F_1 is 1-smooth, by choosing $\alpha = 1$, proximal gradient descent reduces to alternating projection, *i.e.*,

$$x^{k+1} = \Pi_C \Pi_D(x^k),$$

with $f(x) = \Pi_C \Pi_D(x)$.

ISTA. When $F_2 = \mu\|x\|_1$, the problem reduces to sparsity-regularized regression (*e.g.*, Lasso, when F_1 is quadratic). Accordingly, proximal gradient descent reduces to Iterative Shrinkage-Thresholding Algorithm (ISTA), *i.e.*, (denoting $F = F_1$)

$$x^{k+1} = S_{\alpha\mu}(x^k - \alpha \nabla F(x^k)),$$

where $\alpha \in (0, 2/L)$, and

$$S_\kappa(x)_i = \mathbf{sign}(x_i)(|x_i| - \kappa)_+, \quad i = 1, \ldots, n,$$

is the shrinkage operator. The fixed-point mapping here is $f(x) = S_{\alpha\mu}(x - \alpha \nabla F(x))$.

5.1.2 Douglas-Rachford splitting

Consider the following problem:

$$\text{find } x \text{ such that } \ 0 \in (A + B)(x), \tag{25}$$

where $A,\ B : \mathbf{R}^n \to 2^{\mathbf{R}^n}$ are two maximal monotone relations.

Douglas-Rachford splitting (DRS) solves this problem by the following iteration scheme:

$$z^{k+1} = f(z^k) = z^k/2 + C_A C_B(z^k)/2, \tag{26}$$

where C_G is the Cayley operator of G, defined as $C_G(x) = 2(I + \alpha G)^{-1}(x) - x$, where I is the identity mapping, and $\alpha > 0$ is an arbitrary constant. Since the Cayley operator C_G of a maximal monotone relation G is non-expansive and defined over the entire $\mathbf{R}^n$, we see

that the fixed-point mapping $f(x) = x/2 + C_A C_B(x)/2$ is a $\frac{1}{2}$-averaged (and hence non-expansive) operator. The connection between (25) and (26) is established by the fact that x solves (25) if and only if z solves (26) and $x = R_B(z)$, where R_B is the resolvent operator of B, $R_B(x) = (I + \alpha B)^{-1}$.

Below we will implicitly use the facts that subgradients of CCP functions, linear mappings Mx with $M + M^T \succeq 0$, and normal cones of nonempty closed convex sets are all maximal monotone. These facts, as well as the equivalence between (25) and (26), can all be found in [RB16].

Notice that whenever z^k converges to a fixed-point of (26) (not necessarily following the DRS iteration (26)), $x^k = R_B(z^k)$ converges to a solution of problem (25), where $R_B(x) = (I + \alpha B)^{-1}(x)$ is the resolvent of B. This comes immediately from the equivalence between (25) and (26) and the fact that R_B is non-expansive [RB16] and hence continuous. Together with Theorem 6, this ensures that the application of Algorithm 3 to the DRS fixed-point problem (26) leads to the convergence of $x^k = R_B(z^k)$ to a solution of the original problem.

Consensus optimization (CO). In consensus optimization [RB16], we seek to solve

$$\text{minimize} \quad \textstyle\sum_{i=1}^m F_i(x) \tag{27}$$

where $F_i : \mathbf{R}^n \to \mathbf{R}$ are all CCP. Rewriting the problem as

$$\text{minimize} \quad \textstyle\sum_{i=1}^m F_i(x_i) + \mathcal{I}_{\{x_1 = x_2 = \cdots = x_m\}}(x_1, x_2, \ldots, x_m), \tag{28}$$

the problem reduces to (25) with

$$\begin{aligned} A(x) &= \mathcal{N}_{\{x_1 = x_2 = \cdots = x_m\}}(x_1, \ldots, x_m), \\ B(x) &= (\partial F_1(x_1), \ldots, \partial F_m(x_m))^T. \end{aligned}$$

Since for a CCP function $F : \mathbf{R}^n \to \mathbf{R}$ and a nonempty closed convex set C, $C_{\partial F}(x) = 2\mathrm{prox}_{\alpha F}(x) - x$ and $C_{\mathcal{N}_C}(x) = 2\Pi_C(x) - x$, we see that the DRS algorithm reduces to the following:

$$\begin{aligned} x_i^{k+1} &= \mathrm{argmin}_{x_i} \; F_i(x_i) + (1/2\alpha)\|x_i - z_i^k\|_2^2, \\ z_i^{k+1} &= z_i^k + 2\bar{x}^{k+1} - x_i^{k+1} - \bar{z}^k, \quad i = 1, \ldots, m. \end{aligned}$$

where $\bar{x}^k = \frac{1}{m}\sum_{i=1}^m x_i$, and the fixed-point mapping f is the mapping from z^k to z^{k+1}. As discussed above, x^{k+1} converges to the solution of (27) if z^k converges to the fixed-point of f, and hence can be deemed as approximate solutions to the original problem.

SCS. Consider the following generic conic optimization problem:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax + s = b, \quad s \in \mathcal{K}. \end{array} \tag{29}$$

where $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^m$, $c \in \mathbf{R}^n$, and $\mathcal{K}$ is a nonempty, closed and convex cone. Our goal here is to find both primal and dual solutions when they are available, and provide a certificate of infeasibility or unboundedness otherwise [OCPB16]. To this end, one seeks to solve the associated self-dual homogeneous embedding (SDHE) system [YTM94],

$$Qu = v, \quad (u, v)^T \in \mathcal{C} \times \mathcal{C}^*, \tag{30}$$

where $u = (x, y, \tau)^T \in \mathbf{R}^n \times \mathbf{R}^m \times \mathbf{R}$, $v = (r, s, \kappa)^T \in \mathbf{R}^n \times \mathbf{R}^m \times \mathbf{R}$, $\mathcal{C} = \mathbf{R}^n \times \mathcal{K}^* \times \mathbf{R}_+$, $\mathcal{C}^* = \{0\}^n \times \mathcal{K} \times \mathbf{R}_+$ is the dual cone of $\mathcal{C}$, and the SDHE embedding matrix

$$Q = \begin{bmatrix} 0 & A^T & c \\ -A & 0 & b \\ -c^T & -b^T & 0 \end{bmatrix}.$$

The SDHE system can then be further reformulated into (25) ([TP16]), with $A(u) = \mathcal{N}_{\mathcal{C}}(u)$, $B(u) = Qu$. Accordingly, DRS reduces to splitting conic solver (SCS) [OCPB16], *i.e.*,

$$\begin{aligned} \tilde{u}^{k+1} &= (I + Q)^{-1}(u^k + v^k) \\ u^{k+1} &= \Pi_{\mathcal{C}}(\tilde{u}^{k+1} - v^k) \\ v^{k+1} &= v^k - \tilde{u}^{k+1} + u^{k+1}, \end{aligned}$$

Notice that here we have actually used an equivalent form of DRS described in [Van16] with change of variables. In our notation, the fixed-point mapping f is

$$f(u, v) = \begin{bmatrix} \Pi_{\mathcal{C}}((I + Q)^{-1}(u + v) - v) \\ v - (I + Q)^{-1}(u + v) + u \end{bmatrix},$$

which is non-expansive (*cf.* the appendix in [OCPB16]).

Notice that with the transformations made, the equivalence and convergence properties of DRS can not be directly applied here as in the previous examples. Nevertheless, the equivalence between the fixed-point problem and the SDHE system here can be seen directly by noticing that $f(u, v) = (u, v)^T$ if and only if

$$(I + Q)^{-1}(u + v) = u, \quad \Pi_{\mathcal{C}}((I + Q)^{-1}(u + v) - v) = u,$$

i.e., $Qu = v$ and $\Pi_{\mathcal{C}}(u - v) = u$. By Moreau decomposition [PB14], we have

$$\Pi_{\mathcal{C}}(u - v) + \Pi_{-\mathcal{C}^*}(u - v) = u - v,$$

and hence

$$\Pi_{\mathcal{C}}(u - v) = u \Leftrightarrow \Pi_{-\mathcal{C}^*}(u - v) = -v \Leftrightarrow \Pi_{\mathcal{C}^*}(v - u) = v.$$

Hence we see that $f(u, v) = (u, v)^T \Rightarrow Qu = v,\ (u, v)^T \in \mathcal{C} \times \mathcal{C}^*$. On the other hand, when $Qu = v$ and $(u, v) \in \mathcal{C} \times \mathcal{C}^*$, we have $u^T v = u^T Q u = 0$ by the skew-symmetry of Q, and hence for any $w \in \mathcal{C}$,

$$\|u - v - w\|_2^2 = \|u - w\|_2^2 + \|v\|_2^2 - 2v^T(u - w) = \|u - w\|_2^2 + \|v\|_2^2 + 2v^T w \geq \|v\|_2^2,$$

where the last inequality comes from the fact that $v^T w \geq 0$ as $v \in \mathcal{C}^*$ and $w \in \mathcal{C}$, and the equality is achieved if and only if $u = w$. Hence we have $\Pi_{\mathcal{C}}(u - v) = u$, from which we conclude that $(u, v)^T$ is a fixed-point of f if and only if $Qu = v$, $(u, v)^T \in \mathcal{C} \times \mathcal{C}^*$, *i.e.*, $(u, v)^T$ solves the SDHE system.

5.1.3 Contractive mappings in different norms

As we can see from (17) in the proof of Theorem 6, which does not hold for general norms, the ℓ_2-norm in the definition of non-expansiveness is essential to our analysis of global convergence. Nevertheless, an expansive mapping in one norm may be non-expansive or even contractive in another norm, as we will see in the examples below. When a mapping is actually contractive in some (arbitrary) norm, the global convergence of Algorithm 3 can still be guaranteed. Formally, we have the following theorem.

Theorem 7. *Suppose that $\{x^k\}_{k=0}^{\infty}$ is generated by Algorithm 3, but with $\alpha = 1$, and instead of f being non-expansive (in ℓ_2-norm) in (1), f is γ-contractive in some (arbitrary) norm $\|\cdot\|$ (e.g., l_∞-norm) on $\mathbf{R}^n$, where $\gamma \in (0,1)$. Then we still have $\lim_{k\to\infty} x^k = x^\star$, where $x^\star = f(x^\star)$ is a solution to (1).*

The proof can be found in the appendix. Notice that the global convergence in the above algorithm also holds for $\alpha \in (0,1)$, and the proof is exactly the same apart from replacing γ with $(1-\alpha)+\alpha\gamma$, which is larger than γ but is still smaller than 1. The only reason for specifying $\alpha = 1$ is that it gives the fastest convergence speed both in theory and practice for contractive mappings.

Value iteration (VI). Consider solving a discounted Markov decision process (MDP) problem with (expected) reward $R(s,a)$, transition probability $P(s,a,s')$, initial state distribution $\pi(\cdot)$, and discount factor $\gamma \in (0,1)$, where s, $s' \in \{1,\ldots,S\}$ and $a \in \{1,\ldots,A\}$. The goal is to maximize the (expected) total reward $\mathbb{E}_\pi[\sum_{t=0}^{\infty}\gamma^t r(s_t,\mu(s_t))]$ over all possible (stationary) policies $\mu : \{1,\ldots,S\} \to \{1,\ldots,A\}$, where $s_{t+1} \sim P(s_t,\mu(s_t),\cdot)$.

One of the most basic algorithms to solve this problem is the well-known value iteration algorithm:

$$x^{k+1} = Tx^k,$$

where x^k approximates the optimal value function $V^\star(s) = \max_\mu \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t r(s_t,\mu(s_t))|s_0 = s]$, and $T : \mathbf{R}^S \to \mathbf{R}^S$ is the Bellman operator:

$$(Tx)_s = \max_{a=1,\ldots,A} R(s,a) + \gamma \sum\nolimits_{s'=1}^{S} P(s,a,s')x_{s'}.$$

In our notation, the fixed-point mapping $f(x) = T(x)$. A prominent property of T is that although not necessarily non-expansive in ℓ_2-norm, it is γ-contractive under the l_∞-norm, *i.e.*,

$$\|Tx - Ty\|_\infty \le \gamma\|x-y\|_\infty.$$

By Theorem 7, the global convergence is still guaranteed when Algorithm 3 is applied to VI here. We also remark that it would be interesting to apply the accelerated VI to solving the MDP subproblems in certain reinforcement learning algorithms (*e.g.*, PSRL [ORR13], UCRL2 [JOA10]), where the rewards r and transitions P are unknown.

Heavy ball (HB). Consider the following convex quadratic program (QP),

$$\text{minimize} \quad F(x) = \tfrac{1}{2}x^T Ax + b^T x + c \tag{31}$$

where $A \in \mathbf{R}^{n\times n}$ is positive definite with its eigenvalues lying between μ and L, b is a constant vector, and c is a constant scalar. Equivalently, we consider solving the nonsingular linear equation $Ax+b=0$. Notice that this is just a special case of the optimization problem for gradient descent described above, and the unique optimizer is simply $x^\star = -A^{-1}b$ and can be obtained by solving the corresponding linear equation. But here we instead consider solving it using the heavy-ball method, which enjoys a faster linear convergence rate than the vanilla gradient descent [Rec10].

The heavy ball (HB) method is a momentum-based variant of the usual gradient descent, which takes the following form of iterations:

$$x^{k+1} = x^k - \alpha(Ax^k + b) + \beta(x^k - x^{k-1}),$$

where $\alpha = \frac{4}{(\sqrt{L}+\sqrt{\mu})^2}$ and $\beta = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$.

Viewing $(x^k, x^{k-1})^T$ as the iteration variable, the fixed-point mapping f is

$$f(x', x) = \begin{bmatrix} x' - \alpha(Ax' + b) + \beta(x' - x) \\ x' \end{bmatrix} = T \begin{bmatrix} x' \\ x \end{bmatrix} + h,$$

where

$$T = \begin{bmatrix} (1+\beta)I - \alpha A & -\beta I \\ I & 0 \end{bmatrix}, \quad h = \begin{bmatrix} -\alpha b \\ 0 \end{bmatrix},$$

in which z lies on the segment between x and x', and I is the n-by-n identity matrix.

It's easy to see that $(x', x)^T$ is a fixed-point of f if and only if $x = x'$ and $Ax' + b = 0$, and hence $x = x'$ are both solutions to the original problem.

In general, f may not be non-expansive in ℓ_2-norms. However, for any norm $\|\cdot\|$ on $\mathbf{R}^n$,

$$\|f(x', x) - f(y', y)\| \le \|T\|\|(x' - y', x - y)^T\|,$$

where we use the same notation for the induced norm of $\|\cdot\|$ on $\mathbf{R}^{n\times n}$, *i.e.*, $\|T\| = \sup_{x\neq 0} \|Tx\|/\|x\|$. By noticing that the eigenvalues of A all lie between μ and L, we see that the spectral radius of T is upper bounded by [Rec10]

$$\rho(T) \le (\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1) < 1,$$

where $\kappa = L/\mu$. Hence for any sufficiently small ϵ satisfying $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} + \epsilon < 1$, we can define the norm $\|\cdot\|$ as $\|x\| = \|D(1/\epsilon)S^{-1}x\|_1$, where $\|\cdot\|_1$ is the l_1-norm,

$$T = S\ \mathbf{diag}\left(J_{n_1}(\lambda_1), J_{n_2}(\lambda_2), \ldots, J_{n_k}(\lambda_k)\right) S^{-1}$$

is the Jordan decomposition of T, and

$$D(\eta) = \mathbf{diag}\left(D_{n_1}(\eta), D_{n_2}(\eta), \ldots, D_{n_k}(\eta)\right),$$

in which $D_m(\eta) = \mathbf{diag}(\eta, \eta^2, \ldots, \eta^m)$. Then we have $\gamma = \|T\| \leq \rho(T) + \epsilon < 1$ [Fou12], and hence f is γ-contractive in the norm $\|\cdot\|$.

We remark that the although the above example seems to be a bit trivial as f is an affine mapping, there has been no global convergence result even for these simple cases as the existing analysis for applying AA to linear equations all require a full memory [WN11, PE13, RS11]. This indicates that even for affine mappings, one may not be able to avoid our analysis based on non-expansiveness or contractivity. And the HB example here further exemplifies the flexibility of choosing norms for verifying these properties, and restriction to the ℓ_2-norm is unnecessary.

We also remark that similar analysis may be conducted for general strongly convex and strongly smooth objective F, and a convex set constraint may be included by adding a projection step on top of HB. But here we restrict to the above toy case for succinctness, and we leave the more general scenarios for future work.

5.2 Numerical experiments

We are now ready to illustrate the performance of the Anderson Acceleration algorithms with the example problems and (unaccelerated) algorithms above. All the experiments are run using Matlab 2014a on a system with two 1.7 GHz cores and 8 GB of RAM, running macOS X El Capitan.

For each experiment, we show the convergence curves of one representative run against clock time (seconds) and iteration numbers, respectively. The instances we show below are slightly biased towards more difficult ones to better exemplify the improvement of AA-I-S-m (Algorithm 3) over the original AA-I-m (Algorithm 2) . However, in fact our modified algorithm outperforms the original AA-I-m in more than 80% of the tests we tried, and is at least as good as AA-I-m in almost all cases, both in terms of iteration numbers and clock time.

The codes for the experiments, including some further comparisons with other algorithms (*e.g.*, AA-II and its regularized version [SdB16], which are also beaten by our algorithm in most cases, but we only present results focusing on the comparison within the AA-I algorithms) can be found in `https://github.com/cvxgrp/nonexp_global_aa1`. The random seeds are all set to 456, *i.e.*, the one used for producing the plots in this paper for reproducibility. Code in other languages, including Python and Julia, is being developed and will soon be posted.

5.2.1 Implementation details

Before we move on to the numerical results, we first describe in more details the implementation tricks for better efficiency.

Matrix-free updates. In line 11 of Algorithm 3, instead of computing and storing H_k, we actually store $H_{k-j}\tilde{y}_{k-j}$ and $\frac{H_{k-j}^T \hat{s}_{k-j}}{\hat{s}_{k-j}^T (H_{k-j}\tilde{y}_{k-j})}$ for $j = 1, \ldots, m_k$, compute

$$\begin{aligned} d_k &= H_{k-1}g_k + \frac{(s_{k-1} - H_{k-1}\tilde{y}_{k-1})\hat{s}_{k-1}^T H_{k-1} g_k}{\hat{s}_{k-1}^T H_{k-1}\tilde{y}_{k-1}} \\ &= g_k + \sum_{j=1}^{m_k} (s_{k-j} - (H_{k-j}\tilde{y}_{k-j})) \left(\frac{H_{k-j}^T \hat{s}_{k-j}}{\hat{s}_{k-j}^T (H_{k-j}\tilde{y}_{k-j})} \right)^T g_k, \end{aligned}$$

and then update $\tilde{x}^{k+1} = x^k - d_k$. This leads to a much more efficient matrix-free implementation. Another small trick we use is to normalize the $\hat{s}_k$ vectors, store them, and keep them transposed to save the computational overhead.

Termination criteria. In all our experiments, we simply terminate the experiment when either the iteration number reaches a pre-specified maximum $K_{\max}$, or the relative residual norm $\|g_k\|_2 / \|g_0\|_2$ is smaller than some tolerance *tol*. Accordingly, the residual norms in the plots are all rescaled by dividing $\|g_0\|_2$, so all of them starts with 1 in iteration 0. The initial residual norm $\|g_0\|_2$ is shown in the title as `res0`. Unless otherwise specified (*e.g.*, ISTA for elastic net regression), we always choose $K_{\max} = 1000$ and $tol = 10^{-5}$. In addition, we run the vanilla (original) algorithms for additional iterations and then truncate the convergence curves so that all three different algorithms can align in terms of running time (seconds). We remark that although not shown in the plots, the residual norms actually continue to decrease as iterations proceed in all the examples below.

Choice of hyper-parameters. Throughout the experiments, we use a single set of hyper-parameters to show the robustness of our algorithm (Algorithm 3). We choose $\bar{\theta} = 0.01$, $\tau = 0.001$, $D = 10^6$, $\epsilon = 10^{-6}$, and memory $m = 5$ (apart from the memory effect experiment on VI, in which we vary the memory sizes to see the performance change against memories). We choose a small averaging weight $\alpha = 0.1$ to make better use of the fact that most vanilla algorithms already correspond to averaged f.

Additional rules-of-thumb. In our algorithm, in general by setting a relatively large D and small ϵ, one enforces the modified algorithm to use safe-guarding steps less often, making it closer to the original AA-I-m. This may be wanted in case the problems are relatively easy and safe-guard checking is a slight waste of time. The Powell regularization parameter should not be set too large, as it will empirically break down the acceleration effect. For the re-start checking parameter τ, a choice ranging from 0.001 to 0.1 are all found reasonable in our experiments. A large τ will force the algorithm to re-start quite often, making it close to choosing the memory size $m = 1$. A memory size ranging from 2 to 50 are all found to be reasonable choices, with larger memories leading to more stable acceleration with slightly larger per-iteration costs. However, when the memory size becomes too large, especially

when it is close to the variable dimension, our algorithm (as well as the original AA-I-m) will again become unstable.

Although scaling is unnecessary for many of the random instances we show below, it does help a lot in practice if done appropriately. In fact, even for the synthetic but structural UCI Madelon dataset used in the regularized logistic regression example below, the AA algorithms will fail if we do not divide m in the objective. Similar issues occur when it comes to the heavy ball example below with an ill-conditioned linear system. For examples of pre-scaling and pre-conditioning, see [OCPB16].

5.2.2 Problem instances

We consider the following specific problem instances for the algorithms listed in Section 5.1, ranging from statistics, control to game theory and so on. For each plot, AA-I-m is labeled as *aa1*, AA-I-S-m is labeled as *aa1-safe*, and the original (vanilla) algorithm is labeled as *original*. The residual norms are computed in the ℓ_2-norm, *i.e.*, the vertical axis in the plots is $\|g_k\|_2$. In the title of the "residual norm versus time" figures, "time ratio" indicates the average time per iteration of the specified algorithm divided by that of the vanilla algorithm. The average is computed for the single run shown in the figure among all the iterations up to $K_{\max}$.

GD: Regularized logistic regression. We consider the following regularized logistic regression (Reg-Log) problem:

$$\text{minimize} \quad \frac{1}{m}\sum_{i=1}^{m}\log(1+\exp(-y_i\theta^T x_i)) + \frac{\lambda}{2}\|\theta\|_2^2, \tag{32}$$

where $y_i = \pm 1$ are the labels, and $x_i \in \mathbf{R}^n$ are the features and attributes. The minimization is over $\theta \in \mathbf{R}^n$. We use UCI Madelon dataset, which contains 2000 samples (*i.e.*, $m = 2000$) and 500 features (*i.e.*, $n = 500$) . We choose $\lambda = 0.01$, and initialize x^0 with independent normally distributed entries, *i.e.*, using `randn.m`. To avoid numerical overflow, we normalize x^0 to have a ℓ_2-norm equal to 0.001. The step size α is chosen as $2/(L+\lambda)$, where $L = \|X\|_2^2/4m$ is an upper bound on the largest eigenvalues of the objective Hessians [SdB16], and $X = [x_1, \ldots, x_m]$. The results are shown in Figure 1.

In this example, the original AA-I-m completely fails, and our modified AA-I-S-m obtains a 100x-1000x improvement over the original gradient descent algorithm in terms of the residual norms. Interestingly, although the residual norms of AA-I-S-m oscillate above the vanilla algorithm at several points, the improvement in terms of objective values is much more stable, as shown in Figure 2. Here the vertical axis is the objective value minus the smallest objective value found among all three algorithms. The objective value of the original AA-I-m is mostly not even plotted as plugging its corresponding iterates into (32) yields ∞.

HB: Linear system. As described in §5.1.3, we consider the simple problem of solving the nonsingular linear system $Ax + b = 0$, where $A \in \mathbf{R}^{n\times n}$ is positive definite and b is a

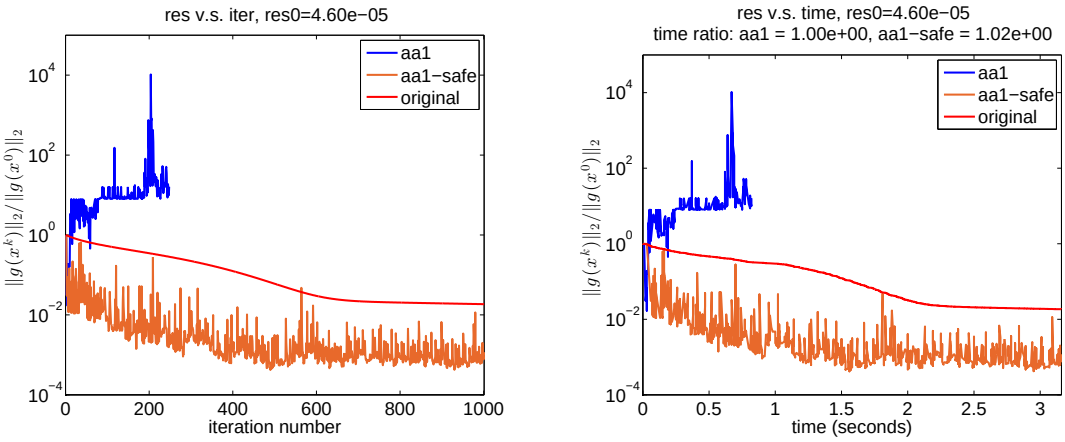

Figure 1: GD: Reg-Log. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

constant vector. We generate $A = B^T B + 0.005I$, where I is the n-by-n identity matrix, and $B \in \mathbf{R}^{\lfloor n/2 \rfloor \times n}$ is generated by `randn.m`. The vector b is also generated with `randn.m`. We choose $n = 1000$ in our experiments. To compute the step sizes α and β, we choose μ as 0.005 and $L = \|A\|_F$, which avoids the expensive eigenvalue decomposition.

Notice that here we deliberately choose B to be a "fat" matrix so that A is ill-conditioned. In our example, the condition number is $\mathbf{cond}(A) \approx 6.4629 \times 10^5$. And with $\kappa = L/\mu \geq \mathbf{cond}(A)$, the convergence of the vanilla HB algorithm will be rather slow, as can be seen from the theoretical convergence rate $(\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$ (which is super close to 1).

To remedy this, we adopt a simple diagonal scaling strategy that scales A and b by the row and column absolute value sums of $A = (a_{ij})_{n\times n}$. More explicitly, we compute $\hat{A} = D^{-1}A$ and $\tilde{b} = D^{-1}b$, where

$$D = \mathbf{diag}\left(\sum_{j=1}^{n} |a_{1j}|, \ldots, \sum_{j=1}^{n} |a_{nj}|\right).$$

We then further right diagonalize $\hat{A} = (\hat{a}_{ij})_{n\times n}$ as $\tilde{A} = \hat{A}E^{-1}$, where

$$E = \mathbf{diag}\left(\sum_{i=1}^{n} |\hat{a}_{i1}|, \ldots, \sum_{i=1}^{n} |\hat{a}_{in}|\right).$$

Essentially, this is exactly performing one step of Sinkhorn-Knopp algorithm to the absolute value matrix $|A| = (|a_{ij}|)_{n\times n}$ of A for matrix equilibration [Kni08]. Obviously, we see that $\tilde{x}$ is the solution to $\tilde{A}\tilde{x} + \tilde{b} = 0$ if and only if $x = E^{-1}\tilde{x}$ is the solution to $Ax + b = 0$. The results are shown in Figure 3, from which we again see the anticipated improvement.

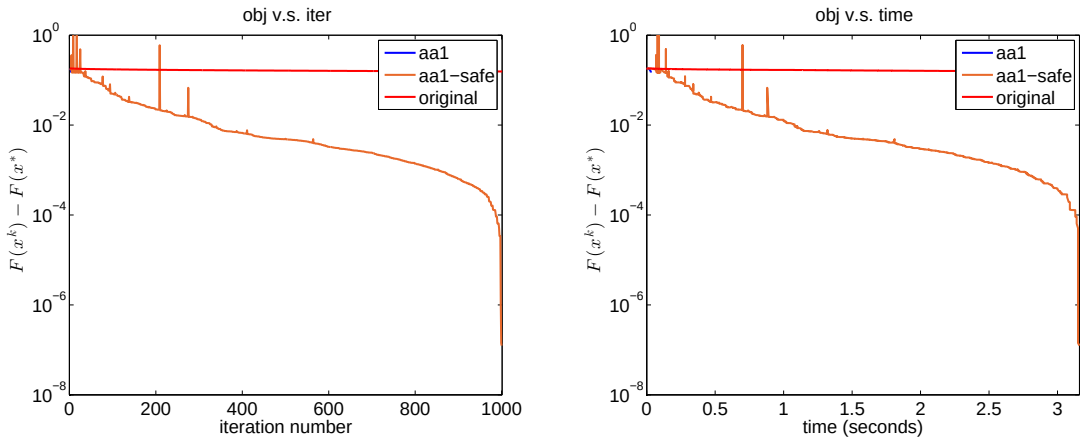

Figure 2: GD: Reg-Log. Left: objective value versus iteration. Right: objective value versus time (seconds).

AP: Linear program. We consider solving the following linear program (LP),

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b, \quad x \in \mathcal{K}, \end{array} \tag{33}$$

where $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^m$, $c \in \mathbf{R}^n$, and $\mathcal{K}$ is a nonempty, closed and convex cone. Notice that here we deliberate choose a different (dual) formulation of (29) to show the flexibility of our algorithm, which can be easily mounted on top of vanilla algorithms.

As in SCS, (33) can be similarly formulated as the self-dual homogeneous embedding (SDHE) system (30), but now with

$$Q = \begin{bmatrix} 0 & -A^T & c \\ A & 0 & -b \\ -c^T & b^T & 0 \end{bmatrix}, \quad \mathcal{C} = \mathcal{K} \times \mathbf{R}^m \times \mathbf{R}_+.$$

Under the notations of AP, solving the SDHE system above reduces to finding a point in the intersection of C and D, with $C = \{(u, v) \mid Qu = v\}$ and $D = \mathcal{C} \times \mathcal{C}^*$, which can then be solved by AP.

We generate a set of random data ensuring primal and dual feasibility of the original problem (33), following [OCPB16]. More specifically, we first generate A as a sparse random matrix with sparsity 0.1 using `sprandn.m`. We then generate $z^\star$ with `randn.m`, and take $x^\star = \max(z^\star, 0)$, $s^\star = \max(-z^\star, 0)$ where the maximum is taken component-wisely. We then also generate $y^\star$ with `randn.m`, and take $b = Ax^\star$, $c = A^T y^\star + s^\star$. In our experiments, we set $m = 500$ and $n = 1000$, and x^0 is simply initialized using `randn.m` and then normalized to have a unit ℓ_2-norm.

In addition, as in the HB example above and SCS [OCPB16], we perform diagonal scaling on the problem data. More explicitly, we compute $\tilde{A} = D^{-1}AE^{-1}$ exactly as in the HB

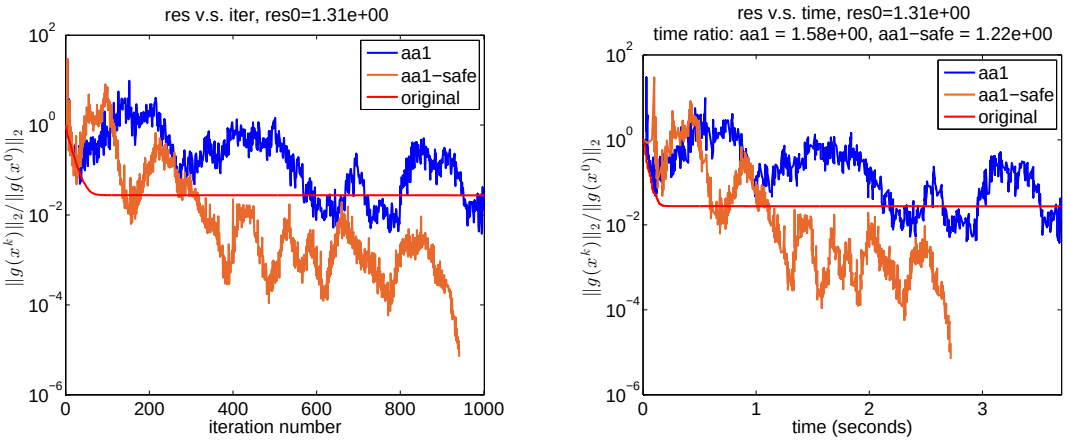

Figure 3: HB: linear system. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

example, and accordingly scale b to be $\tilde{b} = D^{-1}b$ and c to be $\tilde{c} = E^{-1}c$. Again, we see that $\tilde{x}$ is a solution to (33) with A, b, c replaced with the scaled problem data $\tilde{A}$, $\tilde{b}$, $\tilde{c}$, if and only if $x = E^{-1}\tilde{x}$ is a solution to the original problem.

The results are summarized in Figure 4. We can see that our algorithm AA-I-S-m com-

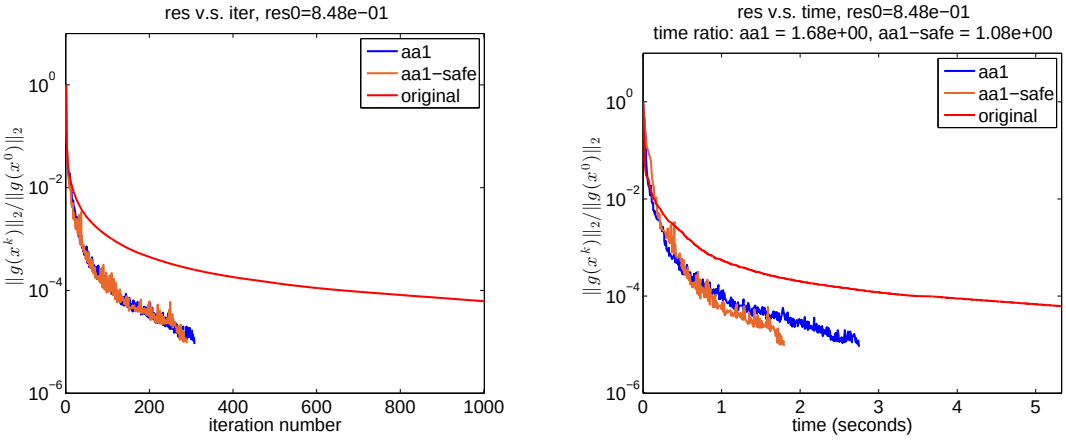

Figure 4: AP: LP as SDHE. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

pares favorably with the original AA-I-m in terms of iteration numbers, and both AA-I-S-m and AA-I-m outperform the vanilla AP algorithm. In terms of running time, we can see a further slight improvement over the original AA-I-m.

PGD: Non-negative least squares and convex-concave matrix game. We consider the following non-negative least squares (NNLS) problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|Ax-b\|_2^2 \\ \text{subject to} & x \geq 0, \end{array} \tag{34}$$

where $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$.

Such a problem arises ubiquitously in various applications, especially when x has certain physical interpretation [CP10]. We consider the more challenging high dimensional case, *i.e.*, $m < n$ [SH13]. The gradient of the objective function can be simply evaluated as $A^TAx - A^Tb$, and hence the PGD algorithm can be efficiently implemented.

We generate both A and b using `randn.m`, with $m = 500$ and $n = 1000$. We again initialize x^0 using `randn.m` and then normalize it to have a unit ℓ_2-norm. The step size α is set to $1.8/\|A^TA\|_2$. The results are summarized in Figure 5.

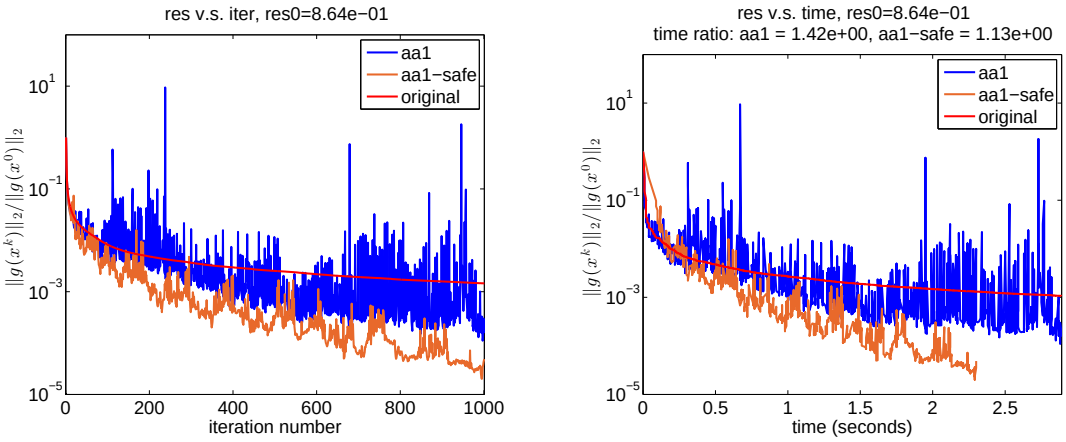

Figure 5: PGD: NNLS. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

We also consider a more specialized and structured problem: convex-concave matrix game (CCMG), which can be reformulated into a form solvable by PGD, as we show below.

A CCMG can be formulated as the following LP [BV04]:

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & u \geq 0, \quad \mathbf{1}^Tu = 1, \quad P^Tu \leq t\mathbf{1}, \end{array} \tag{35}$$

where $t \in \mathbf{R}$, $u \in \mathbf{R}^m$ are variables, and $P \in \mathbf{R}^{m\times n}$ is the pay-off matrix. Of course we can again reformulate it as an SDHE system and solve it by AP as above. But here we instead consider a different reformulation amenable to PGD.

To do so, we first notice that the above LP is always feasible. This can be seen by choosing u to be an arbitrary probability vector, and setting $t = \|P^Tu\|_\infty$. Hence the above

LP can be further transformed into

$$\begin{array}{ll} \text{minimize} & t + \frac{1}{2}\|P^T u + s - t\mathbf{1}\|_2^2 \\ \text{subject to} & u \geq 0, \quad 1^T u = 1, \quad s \geq 0, \end{array} \tag{36}$$

where we introduce an additional (slack) variable $s \in \mathbf{R}^n$. Using the efficient projection algorithm onto the probability simplex set [WCP13, BV04], the above problem can be solved efficiently by PGD.

We generate P using `randn.m` with $m = 500$ and $n = 1500$. Again, x^0 is initialized using `randn.m` and then normalized to have a unit ℓ_2-norm. The step size α is set to $1.8/\|\tilde{A}^T\tilde{A}\|_2$, where $\tilde{A} = [P^T, I, e]$, in which I is the n-by-n identity matrix and $e \in \mathbf{R}^n$ is an all-one vector. The results are summarized in Figure 6.

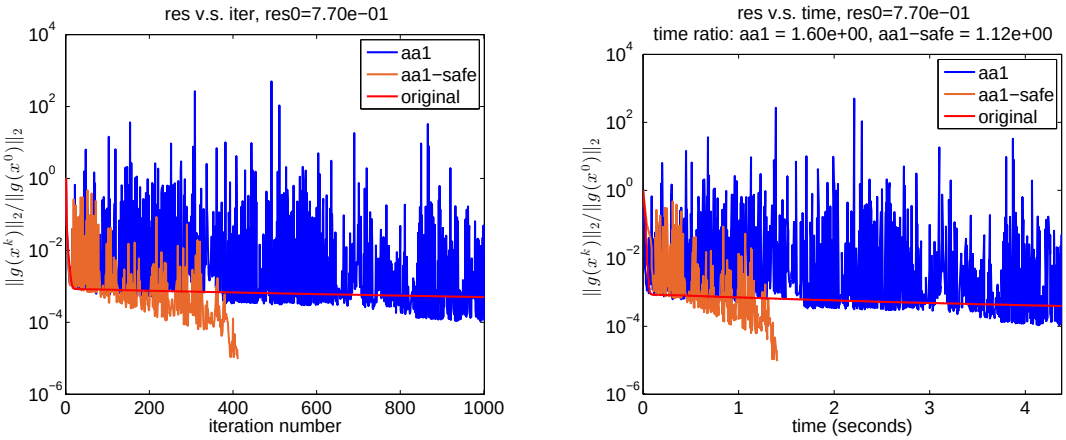

Figure 6: PGD: CCMG. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

ISTA: Elastic net regression. We consider the following elastic net regression (ENR) problem [ZH05]:

$$\text{minimize} \quad \tfrac{1}{2}\|Ax - b\|_2^2 + \mu\left(\tfrac{1-\beta}{2}\|x\|_2^2 + \beta\|x\|_1\right), \tag{37}$$

where $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^m$. In our experiments, we take $\beta = 1/2$ and $\mu = 0.001\mu_{\max}$, where $\mu_{\max} = \|A^T b\|_\infty$ is the smallest value under which the ENR problem admits only the zero solution [OCPB16]. ENR is proposed as a hybrid of Lasso and ridge regression, and has been widely used in practice, especially when one seeks both sparsity and overfitting prevention.

Applying ISTA to ENR, we obtain the following iteration scheme:

$$x^{k+1} = S_{\alpha\mu/2}\left(x^k - \alpha\left(A^T(Ax - b) + \frac{\mu}{2}x\right)\right),$$

in which we choose $\alpha = 1.8/L$, with $L = \lambda_{\max}(A^T A) + \mu/2$.

We again consider a harder high dimensional case, where $m = 500$ and $n = 1000$. The data is generated similar to the Lasso example in [OCPB16]. More specifically, we generate A using `randn.m`, and then generate $\hat{x} \in \mathbf{R}^n$ using `sprandn.m` with sparsity 0.1. We then generate b as $b = A\hat{x} + 0.1w$, where w is generated using `randn.m`. The initial point x^0 is again generated by `randn.m` and normalized to have a unit ℓ_2-norm. The step size is chosen as $\alpha = 1.8/L$, where $L = \|A^T A\|_2 + \mu/2$. The results are shown in Figure 7. Here we set the tolerance *tol* to 10^{-8} to better exemplify the performance improvement of our algorithm in a relative long run.

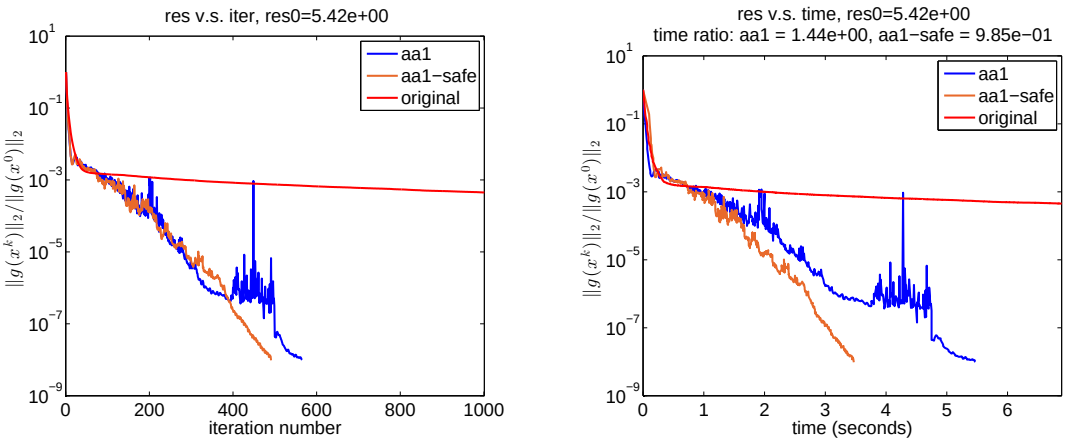

Figure 7: ISTA: ENR. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

CO: Facility location. Consider the following facility location problem [XY97]:

$$\text{minimize} \quad \sum_{i=1}^m \|x - c_i\|_2, \tag{38}$$

where $c_i \in \mathbf{R}^n$, $i = 1, \ldots, m$ are locations of the clients, and the goal is to find a facility location that minimizes the total distance to all the clients.

Applying CO to this problem with $\alpha = 1$, we obtain that ([PB14])

$$\begin{array}{ll} x_i^{k+1} & = \mathbf{prox}_{\|\cdot\|_2}(z_i^k + c_i) - c_i \\ z_i^{k+1} & = z_i^k + 2\bar{x}^{k+1} - x_i^{k+1} - \bar{z}^k, \quad i = 1, \ldots, m, \end{array}$$

where $\mathbf{prox}_{\|\cdot\|_2}(v) = (1 - 1/\|v\|_2)_+ v$.

Notice that all the updates can be parallelized. In particular, in the Matlab implementation no for loops is needed within one iteration, which is important to the numerical efficiency. We generate c_i using `sprandn.m`, with $m = 500$ and $n = 300$ and sparsity 0.01. The results are summarized in Figure 8. Notice that here we again set the tolerance *tol* to 10^{-8} to better

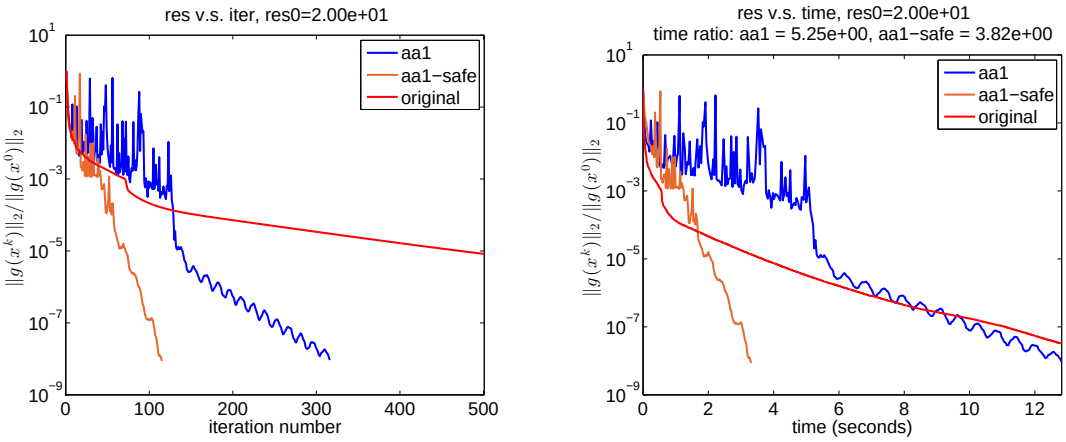

Figure 8: CO: facility location. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

demonstrate the improvement of our algorithm, and we truncate the maximum iteration number to $K_{\max} = 500$ for better visualization.

We remark that in general, the ℓ_2-norm can also be replaced with an arbitrary p-norm, and more generally any function for which the proximal operators can be easily evaluated.

SCS: Cone programs. Consider (29) with $\mathcal{K} = \mathbf{R}^m_+$ (resp. $\mathcal{K} = \{s \in \mathbf{R}^m \mid \|s_{1:m-1}\|_2 \leq s_m\}$), *i.e.*, a generic LP (resp. SOCP). We solve it using a toy implementation of SCS, *i.e.*, one without approximate projection, CG iterations, fine-tuned over-relaxation and so on.

We make use of the following explicit formula for the projection onto the second order cone $\mathcal{K} = \{s \in \mathbf{R}^m \mid \|s_{1:m-1}\|_2 \leq s_m\}$ ([PB14]):

$$\Pi_{\mathcal{K}}(s) = \begin{cases} s & \text{if } \|s_{1:n-1}\|_2 \leq s_n \\ 0 & \text{if } \|s_{1:n-1}\|_2 \leq -s_n \\ \frac{\|s_{1:n-1}\|_2 + s_n}{2} \left[\frac{s_{1:n-1}}{\|s_{1:n-1}\|_2}, 1\right]^T & \text{otherwise.} \end{cases}$$

For both LP and SOCP, we choose $m = 500$ and $n = 700$, and x^0 is initialized using `randn.m` and then normalized to have a unit ℓ_2-norm. We again follow [OCPB16] to generate data that ensures primal and dual feasibility of the original cone programs.

For LP, we generate A as a horizontal concatenation of `sprandn(m,⌊n/2⌋,0.1)` and identity matrix of size $m \times \lfloor n/2 \rfloor$, added with a noisy term `1e-3 * randn(m, n)`. We then generate $z^\star$ using `randn.m`, and set $s^\star = \max(z^\star, 0)$ and $y^\star = \max(-z^\star, 0)$, where the maximum is also taken component-wisely. We then also generate $x^\star$ using `randn.m`, and take $b = Ax^\star + s^\star$ and $c = -A^T y^\star$.

For SOCP, we similarly generate A exactly the same as in LP. We then generate $z^\star$ using `randn.m`, and set $s^\star = \Pi_{\mathcal{K}}(z^\star)$ and $y^\star = s^\star - z^\star$, where the maximum is also taken

component-wisely. We then once again generate $x^\star$ using `randn.m`, and take $b = Ax^\star + s^\star$ and $c = -A^T y^\star$.

The results are summarized in Figure 9 and Figure 10.

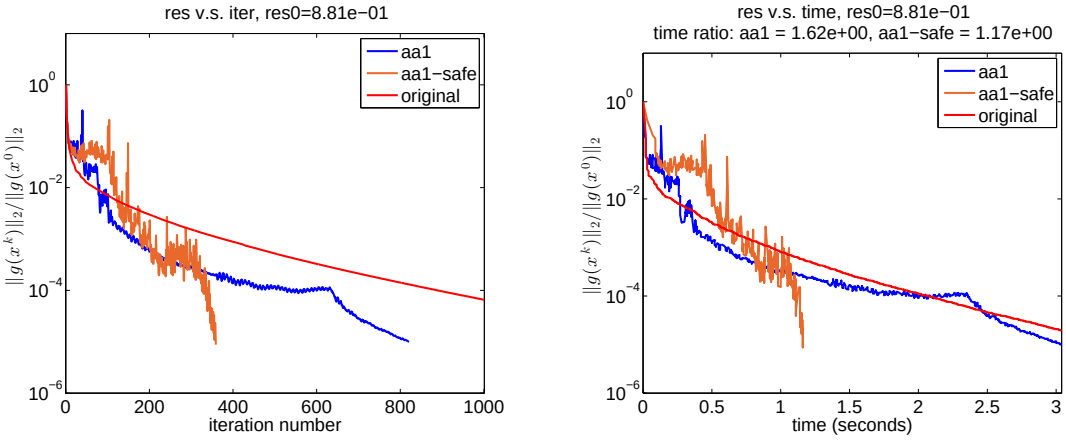

Figure 9: SCS: LP. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

VI: Markov decision process. As described in §5.1.3, we consider solving a general random Markov decision process (MDP) using VI. In our experiments, we choose $S = 300$ and $A = 200$, and we choose a large discount factor $\gamma = 0.99$ to make the problem more difficult, thusly making the improvement of AA more explicit.

The transition probability matrices $P_a \in \mathbf{R}^{S\times S}$, $a = 1, \ldots, A$ are first generated as `sprand` $(S, S, 0.01) + 0.001I$, where I is the S-by-S identity matrix, and then row-normalized to be a stochastic matrix. Here the addition of $0.001I$ is to ensure that no all-zero row exists. Similarly, the reward matrix $R \in \mathbf{R}^{S\times A}$ is generated by `sprandn.m` with sparsity 0.01. The results are summarized in Figure 11. Notice that the maximum iteration $K_{\max}$ is set to 50 for better visualization.

It would be interesting to test the algorithms on more structured MDP, *e.g.*, the chain MDP, frozen lake, grid world, and more practically an energy storage problem (`http://castlelab.princeton.edu/html/datasets.htm#storagesalas`).

Moreover, it would also be interesting to see how our algorithm helps as a sub-solver in other reinforcement learning algorithms.

Influence of memory sizes. Finally, we rerun the VI experiments above with different memories $m = 2,\ 5,\ 10,\ 20,\ 50$. All other data are exactly the same as in the previous example. The results are summarized in Figure 12. Notice that again the maximum iteration $K_{\max}$ is set to 50 for better visualization.

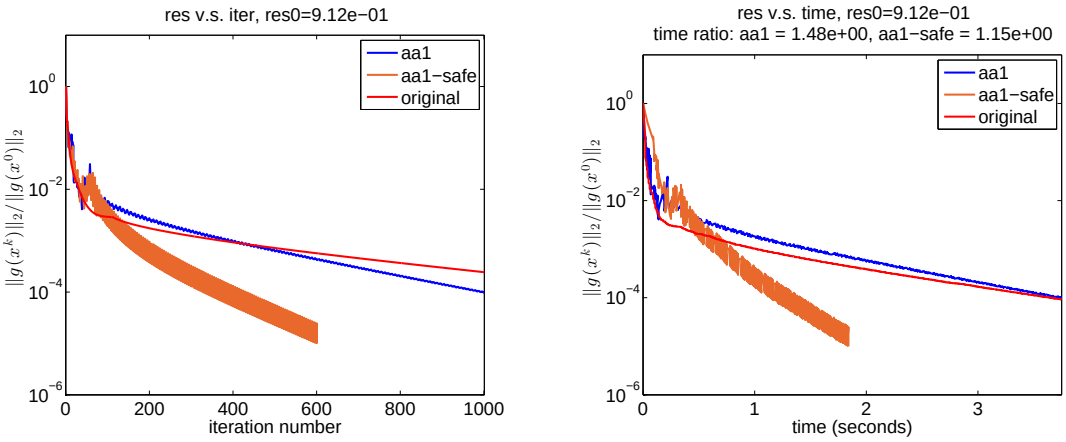

Figure 10: SCS: SOCP. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

We can see from the figures that the best performance is achieved when $m = 2,\ 5$, and AA-I-S-m consistently improves over the original AA-I-m. In addition, although increasing the memory size may accelerate the convergence in terms of iteration numbers, the per-iteration cost may eventually dominate, as can be seen from the comparison between $m = 10$ and $m = 20,\ 50$.

5.2.3 Summary of numerical results

As we have seen above, surprisingly our algorithm actually performs better than the original AA-I-m in various cases, sometimes even when AA-I-m does not seem to suffer much from instability (*e.g.*, SCS for SOCP), and the improvement is more significant when the latter does (*e.g.*, GD for regularized logistic regression).

In terms of running time, the safe-guard checking steps does seem to slightly slow down our algorithm AA-I-S-m, as expected. This is more obvious for simple problems with larger sizes (*e.g.*, VI for MDP). Nevertheless, due to the easiness of such problems the extra time is still quite affordable, making this a minor sacrifice for robustness. In addition, as in our algorithm the approximate Jacobians are computed with rank-one updates and re-starts are invoked from time to time, it is indeed potentially faster than the original fixed-memory AA-I-m, in which the approximate Jacobian is computed from scratch with m memories. This is also exemplified in most of our numerical experiments shown above.

Finally, by better parallelization and GPU acceleration, the per-iteration running time of both AA-I-S-m and AA-I-m should be further improved compared to the current single CPU version without any nontrivial parallelization. This may relieve the contrary result of acceleration and deceleration of the original AA-I-m in terms of iteration numbers and time respectively in some examples above (*e.g.*, CO for facility location), and may further improve the acceleration effect of our algorithm in terms of computational time.

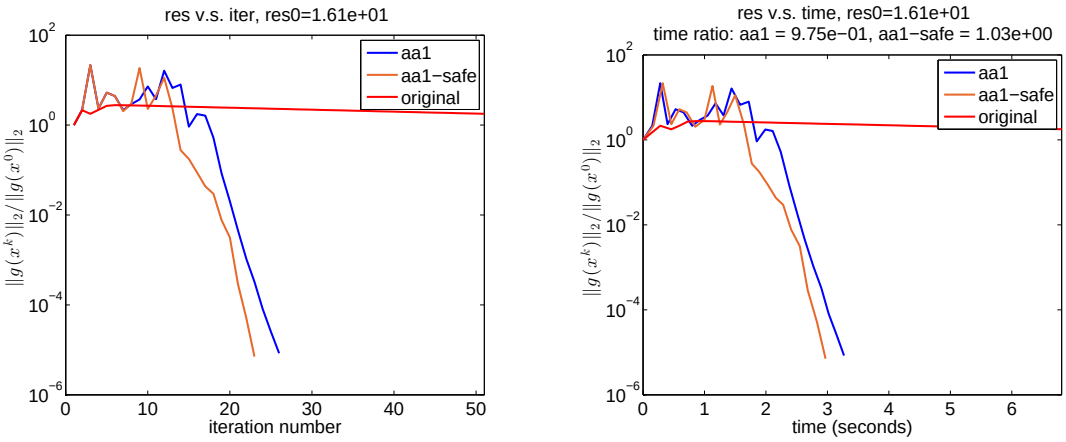

Figure 11: VI: MDP. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

6 Extensions to more general settings

In this section, we briefly outline some extended convergence analysis and results of our algorithm in several more general settings, and discuss the necessity of potential modifications of our algorithm to better suit some more challenging scenarios.

Quasi-nonexpansive mappings. A mapping $f : \mathbf{R}^n \to \mathbf{R}^n$ is called *quasi-nonexpansive* if for any $y \in X$ a fixed-point of f, $\|f(x) - y\|_2 \leq \|x - y\|_2$ for any $x \in \mathbf{R}^n$. Obviously, non-expansive mappings are quasi-nonexpansive.

Our convergence theorems actually already hold for these slightly more general mappings, with an additional mild assumption on continuity. By noticing that non-expansiveness is only applied between an arbitrary point and a fixed-point of f in the proof of Theorem 6, and that the continuity of g can be relaxed to closedness, we immediately see that the same global convergence result hold if f is only assumed to be quasi-nonexpansive and closed. Here we say that a mapping $f : \mathbf{R}^n \to \mathbf{R}^n$ is closed if for any sequence $\{x_n\}_{n=0}^{\infty} \subseteq \mathbf{R}^n$, $x_n \to x$ and $f(x_n) \to y$ imply that $f(x) = y$.

Similarly, Theorem 7 remain true if f is closed and the contractivity is assumed only between an arbitrary point and a fixed-point of f, *i.e.*, $\|f(x) - f(y)\| \leq \gamma\|x - y\|$ for any $x \in \mathbf{R}^n$ and $y \in X$, which we term as *quasi-γ-contractive*.

Formally, we have the following corollary:

Corollary 8. *Suppose that $\{x^k\}_{k=0}^{\infty}$ is generated by Algorithm 3, and instead of f being non-expansive (in ℓ_2-norm) in (1), we only assume that f is closed, and is either quasi-nonexpansive (in ℓ_2-norm), or quasi-γ-contractive in some (arbitrary) norm $\|\cdot\|$ (e.g., l_∞-norm) on $\mathbf{R}^n$, where $\gamma \in (0, 1)$. Then we still have $\lim_{k\to\infty} x^k = x^\star$, where $x^\star = f(x^\star)$ is a*

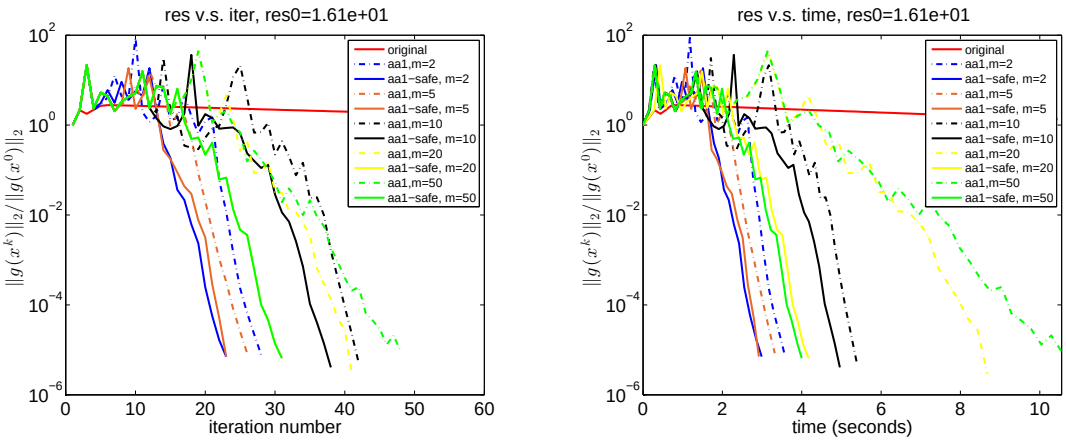

Figure 12: VI: memory effect. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

solution to (1). In the latter (quasi-γ-contractive) case, the averaging weight α can also be taken as 1.

Iteration-dependent mappings. Consider the case when the mapping f varies as iteration proceeds, *i.e.*, instead of a fixed f, we have $f_k : \mathbf{R}^n \to \mathbf{R}^n$ for each $k = 0, 1, \ldots$. The goal is to find the common fixed-point of all f_k (assuming that it exists), *i.e.*, finding $x^\star \in \cap_{k\geq 0} X_k$ with X_k being the fixed-point set of f_k. For example, in GD, we may consider a changing (positive) step size, which will result in a varying mapping f. However, the common fixed-point of all f_k is still exactly the optimal solution to the original optimization problem. In fact, all f_k have the same fixed-point set.

Assuming non-expansiveness (actually quasi-nonexpansiveness suffices) of each f_k, $k \geq 0$, and that the fixed-point set $X_k = X$ of f_k is the same across all $k \geq 0$, both of which hold for GD with positive varying step sizes described above, we can still follow exactly the same steps 1 and 2 of the proof for Theorem 6 to obtain that $\|g_k\|_2 \to 0$ as $k \to \infty$, where $g_k = x^k - f_k(x^k)$, and that $\|x^k - y\|_2$ converges for any fixed-point $y \in X$.

Unfortunately, in general step 3 does not go through with these changing mappings. However, if we in addition assume that for any sequence $x^k \in \mathbf{R}^n$, $\lim_{k\to\infty} \|x^k - f_k(x^k)\|_2 = 0$ and $x^k \to \bar{x} \Rightarrow \bar{x} \in X$, then any limit point of x^k is a common fixed-point of f_k's in X. The rest of step 3 then follows exactly unchanged, which finally shows that Theorem 6 still holds in this setting.

Formally, we have the following corollary:

Corollary 9. *Suppose that $f_k : \mathbf{R}^n \to \mathbf{R}^n$, $k \geq 0$ are all quasi-nonexpansive, and that the fixed-point sets $X_k = \{x \in \mathbf{R}^n \mid f_k(x) = x\}$ of f_k are equal to the same set $X \subseteq \mathbf{R}^n$. Assume in addition that for any sequence $\{z^k\}_{k=0}^\infty \subseteq \mathbf{R}^n$, if $\lim_{k\to\infty} \|z^k - f_k(z^k)\|_2 = 0$ and $z^k \to \bar{z}$ for some $\bar{z} \in \mathbf{R}^n$, then $\bar{z} \in X$. Suppose that $\{x^k\}_{k=0}^\infty$ is generated by Algorithm 3, with f*

replaced with f_k in iteration k. Then we have $\lim_{k\to\infty} x^k = x^\star$, where $x^\star = f(x^\star)$ is a solution to (1).

Although the additional assumption about "z^k" seems to be a bit abstract, it does hold if we nail down to the aforementioned specific case, the GD example with varying step sizes, *i.e.*, $f_k(x^k) = x^k - \alpha^k \nabla F(x^k)$, and if we assume in addition that the step size α^k is bounded away from 0, *i.e.*, $\alpha^k \geq \epsilon > 0$ for some positive constant ϵ for all $k \geq 0$.

In fact, by $\lim_{k\to\infty} \|g_k\|_2 = \lim_{k\to\infty} \|x^k - f_k(x^k)\|_2 = 0$, we have $\lim_{k\to\infty} \alpha^k \|\nabla F(x^k)\|_2 = 0$, which implies that $\lim_{k\to\infty} \|\nabla F(x^k)\|_2 = 0$ as $\alpha^k \geq \epsilon > 0$. In particular, any limit point $\bar{x}$ of x^k satisfies $\nabla F(\bar{x}) = 0$ by the continuity of ∇F assumed in §5.1.1, *i.e.*, $\bar{x} \in X$. Hence we see that the assumptions made in Corollary 9 all hold in this example, and hence global convergence of x^k is ensured.

A similar analysis can be carried out to reprove Theorem 7 in this setting.

Nevertheless, it remains open what assumptions are needed in general to obtain global convergence as in Theorem 6 and Theorem 7. In particular, the above analysis fails if α^k is vanishing, which may arise in many practical cases, *e.g.*, training of deep neural networks (using stochastic algorithms, which is to be discussed below). It might be true that some adaptive mechanisms need to be included in the design of our algorithm to fully fit to this changing mapping scenario.

Non-expansive mappings in non-Euclidean norms. Theorem 7 establishes global convergence for contractive mappings in arbitrary norms. It is hence natural to ask what happens if f is only non-expansive (instead of contractive) in an arbitrary norm different from the ℓ_2-norm. More generally, the norm in which the mapping f is non-expansive or contractive may also change as the iterations proceed, which is exactly the case if we perform the same analysis of HB for a general strongly convex and strongly smooth objective function. In general, finding out the additional assumptions needed for the global convergence of the current algorithm in these settings, or a way to further modify our algorithm to work here if necessary, may largely contribute to a more flexible algorithm.

7 Conclusions

In this paper, we modify the type-I Anderson acceleration (AA-I) to propose a globally convergent acceleration algorithm that works for general non-expansive non-smooth fixed-point problems, with no additional assumptions required. We list 9 problem-algorithm combinations, each supported by one or more concrete problem instances. Our extensive numerical results show that our modified algorithm is not only more robust, but also more efficient than the original AA-I. Finally, extensions to different settings are discussed, and in particular another theorem is established to ensure global convergence of our algorithm on value iteration in MDPs and heavy ball methods in QPs.

Despite the success of our algorithm both in theory and practice, several problems remain open. In particular, the convergence of our proposed algorithm on general momentum

methods (*e.g.*, HB for general convex constrained optimization, and Nesterov's accelerated gradient descent), and the potential modifications needed in the absence of such convergence, deserves a more thorough study. In addition, it is also interesting to see how our algorithm performs in stochastic settings, *i.e.*, when the evaluation of f is noisy. Moreover, some popular algorithms are still ruled out from our current scenario, *e.g.*, Frank-Wolfe and Sinkhorn-Knopp. It is thus desired to push our algorithm more beyond non-expansiveness (apart from quasi-nonexpansiveness and contractivity in a non-Euclidean norm) to incorporate these interesting examples, which may also help address the convergence for algorithms in non-convex optimization settings. In the meantime, a theoretical characterization of the acceleration effect of our algorithm is still missing, and in particular no convergence rate has been established for our algorithm. Although some partial unpublished results relying on certain differentiability has been obtained by us, it remains super challenging how we can include all the (non-smooth) mappings listed in §5 into the assumptions. Last but not least, numerical tests with larger sizes and real-world datasets, and more systematic comparisons with other acceleration methods (*e.g.*, AA-II), are yet to be conducted, which may finally contribute to a new automatic acceleration unit for general existing solvers.

8 Acknowledgements

We thank Ernest Ryu for his comments on the possibility and difficulty of analyzing Frank-Wolfe and Nesterov's algorithms in fixed-point mapping frameworks. We thank Tianyi Lin for his advice on generating appropriate random data for LPs, and his comments on the challenge of convergence order analysis in quasi-Newton methods. We thank Qingyun Sun for the general discussions in the early stage of the project, especially those related to interpreting AA as multi-secant methods [FS09, WN11]. We also thank Michael Saunders for his constant encouragement and positive feedback on the progress of the project on a high level, and his detailed suggestions on an early draft of this paper. We are also grateful to Zaiwen Wen for the inspiring discussions on related literature. We also thank Anran Hu for her suggestions on the organization and presentation of the paper, and her suggestion on considering the heavy ball algorithm. We thank Pontus Giselsson for pointing out the relationship between the safeguarding strategy in this paper and the line search condition in [GFB16] as well as the necessity of an additional closedness assumption for the quasi-nonexpansive extension in §6. Last but not least, we also thank Zico Kolter for his positive feedback on an early draft of this paper, and Panagiotis Patrinos for some detailed remarks on SuperMann as a general globalization framework.

References

[AJW17] H. An, X. Jia, and H. F. Walker. Anderson acceleration and application to the three-temperature energy equations. *Journal of Computational Physics*, 347:1–19, 2017.

[And65] D. G. Anderson. Iterative procedures for nonlinear integral equations. *J. Assoc. Comput. Mach.*, 12:547–560, 1965.

[AUM+16] A. Atanasov, B. Uekermann, C. A. P. Meja, H. J. Bungartz, and P. Neumann. Steady-state Anderson accelerated coupling of lattice boltzmann and navierstokes solvers. *Computation*, 4(4):38, 2016.

[AVDB18] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[AWK17] A. Ali, E. Wong, and J. Z. Kolter. A semismooth Newton method for fast, generic convex programming. *arXiv preprint arXiv:1705.00772*, 2017.

[BAC13] L. M. Briceno-Arias and P. L. Combette. Monotone operator methods for Nash equilibria in non-potential games. *Computational and Analytical Mathematics*, pages 143–159, 2013.

[BC10] H. H. Bauschke and P. L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 2011. New York: Springer, 2010.

[Bel57] R. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

[Ber15] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control 4th Edition, Volume II*. Athena Scientific, 2015.

[BF94] O. Burdakov and U. Felgenhauer. Stable multipoint secant methods with released requirements to points position. *System Modelling and Optimization*, pages 225–236, 1994.

[BK17] O. Burdakov and A. Kamandi. Multipoint secant and interpolation methods with nonmonotone line search for solving systems of nonlinear equations. *arXiv preprint arXiv:1712.01142*, 2017.

[BNP15] H. H. Bauschke, D. Noll, and H. M. Phan. Linear and strong convergence of algorithms involving averaged nonexpansive operators. *Journal of Mathematical Analysis and Applications*, 421(1):1–20, 2015.

[Bro65] C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of computation*, 19(92):577–593, 1965.

[BRZS18] C. Brezinski, M. Redivo-Zagila, and Y. Saad. Shanks sequence transformations and Anderson acceleration. *SIAM Review*, 60(3):646–669, 2018.

[BSd18] R. Bollapragada, D. Scieur, and A. d'Aspremont. Nonlinear acceleration of momentum and primal-dual algorithms. *arXiv preprint arXiv:1810.04539*, 2018.

[BSP16] A. S. Banerjee, P. Suryanarayana, and J. E. Pask. Periodic Pulay method for robust and efficient convergence acceleration of self-consistent field iterations. *Chemical Physics Letters*, 647:31–35, 2016.

[BV04] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[CCL14] Z. Chen, W. Cheng, and X. Li. A global convergent quasi-newton method for systems of monotone equations. *Journal of Applied Mathematics and Computing*, 44(1-2):455–465, 2014.

[Com01] P. L. Combettes. Quasi-Fejérian analysis of some optimization algorithms. *Studies in Computational Mathematics*, 8:115–152, 2001.

[CP10] D. Chen and R. J. Plemmons. Nonnegativity constraints in numerical analysis. *The birth of numerical analysis*, pages 109–139, 2010.

[DD18] W. Drummond and S. Duncan. Accelerated gradient methods with memory. *arXiv preprint arXiv:1805.09077*, 2018.

[Fou12] S. Foucart. Lecture 6: Matrix norms and spectral radii, `http://www.math.drexel.edu/~foucart/TeachingFiles/F12/M504Lect6.pdf`. 2012.

[FS09] H. Fang and Y. Saad. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16(3):197–221, 2009.

[GFB16] P. Giselsson, M. Fält, and S. Boyd. Line search for averaged operator iteration. *Decision and Control (CDC), 2016 IEEE 55th Conference on*, IEEE, 2016.

[Gri12] A. Griewank. Broyden updating, the good and the bad! *Optimization Stories, Documenta Mathematica*, Extra: Optimization Stories:301–315, 2012.

[GS78] D. M. Gay and R. B. Schnabel. Solving systems of nonlinear equations by Broyden's method with projected updates. *Nonlinear Programming*, 3:245–281, 1978.

[HR18] N. C. Henderson and R.Varadhan. Damped Anderson acceleration with restarts and monotonicity control for accelerating EM and EM-like algorithms. *arXiv preprint arXiv:1803.06673*, 2018.

[HS16] N. 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.

[HY10] X. Hu and W. Yang. Accelerating self-consistent field convergence with the augmented Roothaan-Hall energy function. *The Journal of chemical physics*, 132(5):054109, 2010.

[IK92] C. M. Ip and J. Kyparisis. Local convergence of quasi-Newton methods for B-differentiable equations. *Mathematical Programming*, 56(1-3):71–89, 1992.

[JOA10] T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

[Kni08] P. A. Knight. The Sinkhorn-Knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008.

[LF99a] D. Li and M. Fukushima. A derivative-free line search and DFP method for symmetric equations with global and superlinear convergence. *Numerical functional analysis and optimization*, 20(1-2):59–77, 1999.

[LF99b] D. Li and M. Fukushima. A globally and superlinearly convergent Gauss-Newton-based BFGS method for symmetric nonlinear equations. *SIAM Journal on Numerical Analysis*, 37(1):152–172, 1999.

[LF00] D. Li and M. Fukushima. A derivative-free line search and global convergence of Broyden-like method for nonlinear equations. *Optimization Methods and Software*, 13(3):181–201, 2000.

[LY84] D. G. Luenberger and Y. Ye. *Linear and nonlinear programming*, volume 2. Reading, MA: Addison-wesley, 1984.

[MST+17] S. A. Matveev, V. I. Stadnichuk, E. E. Tyrtyshnikov, A. P. Smirnov, N.V. Ampilogova, and N. V. Brilliantov. Anderson acceleration method of finding steady-state particle size distribution for a wide class of aggregationfragmentation models. *Computer Physics Communications*, 2017.

[MT76] J. J. Moré and J. A. Trangenstein. On the global convergence of Broydens method. *Mathematics of Computation*, 30(135):523–540, 1976.

[Nes13] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[OCPB16] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.

[OCPB17] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. SCS: Splitting conic solver, version 2.0.2. `https://github.com/cvxgrp/scs`, November 2017.

[ORR13] I. Osband, D. Russo, and B. Van Roy. (More) efficient reinforcement learning via posterior sampling. *In Advances in Neural Information Processing System*, pages 3003–3011, 2013.

[Pan90] J. S. Pang. Newton's method for B-differentiable equations. *Mathematics of Operations Research*, 15(2):311–341, 1990.

[PB14] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[PE13] F. A. Potra and H. Engler. A characterization of the behavior of the Anderson acceleration on linear problems. *Linear Algebra and its Applications*, 438(3):1002–1011, 2013.

[POD+17] A. L. Pavlov, G. V. Ovchinnikov, D. Yu Derbyshev, D. Tsetserukou, and I. V. Oseledets. AA-ICP: Iterative closest point with Anderson acceleration. *arXiv preprint arXiv:1709.05479*, 2017.

[Pow70] M. JD. Powell. A hybrid method for nonlinear equations. *Numerical Methods for Nonlinear Algebraic Equations*, pages 87–144, 1970.

[PS15] P. P. Pratapa and P. Suryanarayana. Restarted Pulay mixing for efficient and robust acceleration of fixed-point iterations. *Chemical Physics Letters*, 635:69–74, 2015.

[Pul80] P. Pulay. Convergence acceleration of iterative sequences. the case of SCF iterations. *Chem. Phys. Lett.*, 73:393–398, 1980.

[Pul82] P. Pulay. Improved SCF convergence. *J. Comput. Chem.*, 3:556–560, 1982.

[RB16] E. K. Ryu and S. Boyd. Primer on monotone operator methods. *Appl. Comput. Math*, 15(1):3–43, 2016.

[Rec10] B. Recht. Cs726-Lyapunov analysis and the heavy ball method. 2010.

[RS11] T. Rohwedder and R. Schneider. An analysis for the DIIS acceleration method used in quantum chemistry calculations. *Journal of mathematical chemistry*, 49(9):1889–1914, 2011.

[SBd17] D. Scieur, F. Bach, and A. d'Aspremont. Nonlinear acceleration of stochastic algorithms. *Advances In Neural Information Processing Systems*, 2017.

[SdB16] D. Scieur, A. d'Aspremont, and F. Bach. Regularized nonlinear acceleration. *Advances In Neural Information Processing Systems*, 2016.

[SH13] M. Slawski and M. Hein. Non-negative least squares for high-dimensional linear models: Consistency and sparse recovery without regularization. *Electronic Journal of Statistics*, 7:3004–3056, 2013.

[SOdB18a] D. Scieur, E. Oyallon, A. d'Aspremont, and F. Bach. Nonlinear acceleration of CNNs. *arXiv preprint arXiv:1806.003709*, 2018.

[SOdB18b] D. Scieur, E. Oyallon, A. d'Aspremont, and F. Bach. Nonlinear acceleration of deep neural networks. *arXiv preprint arXiv:1805.09639*, 2018.

[TEE+17] A. Toth, J. A. Ellis, T. Evans, S. Hamilton, C. T. Kelley, R. Pawlowski, and S. Slattery. Local improvement results for Anderson acceleration with inaccurate function evaluations. *SIAM Journal on Scientific Computing*, 39(5):S47–S65, 2017.

[TK15] A. Toth and C. T. Kelley. Convergence analysis for Anderson acceleration. *SIAM Journal on Numerical Analysis*, 53(2):805–819, 2015.

[TP16] A. Themelis and P. Patrinos. Supermann: a superlinearly convergent algorithm for finding fixed points of nonexpansive operators. *arXiv preprint arXiv:1609.06955*, 2016.

[Van16] L. Vandenberghe. 13. Douglas-Rachford method and ADMM. *EE236C*, 2016.

[WCP13] W. Wang and M. Á. Carreira-Perpiñán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv preprint arXiv:1309.1541*, 2013.

[WN11] H. F. Walker and P. Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

[WPT15] J. Willert, H. Park, and W. Taitano. Using Anderson acceleration to accelerate the convergence of neutron transport calculations with anisotropic scattering. *Nuclear Science and Engineering*, 181(3):342–350, 2015.

[WTK14] J. Willert, W. T. Taitano, and D. Knoll. Leveraging Anderson acceleration for improved convergence of iterative solutions to transport systems. *Journal of Computational Physics*, 273:278–286, 2014.

[XLWZ16] X. Xiao, Y. Li, Z. Wen, and L. Zhang. A regularized semi-smooth Newton method with projection steps for composite convex programs. *Journal of Scientific Computing*, pages 1–26, 2016.

[XY97] G. Xue and Y. Ye. An efficient algorithm for minimizing a sum of euclidean norms with applications. *SIAM Journal on Optimization*, 7(4):1017–1036, 1997.

[YTM94] Y. Ye, M. J. Todd, and S. Mizuno. An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 19(1):53–67, 1994.

[ZH05] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[ZL07] W. Zhou and D. Li. Limited memory BFGS method for nonlinear monotone equations. *Journal of Computational Mathematics*, pages 89–96, 2007.

[ZL08] W. Zhou and D. Li. A globally convergent BFGS method for nonlinear monotone equations without any merit functions. *Mathematics of Computation*, 77(264):2231–2240, 2008.

[ZYP+18] J. Zhang, Y. Yao, Y. Peng, H. Yu, and B. Deng. Fast k-means clustering with Anderson acceleration. *arXiv preprint arXiv:1805.10638*, 2018.

Appendices

Proof of Proposition 1.

Proof. Suppose that $Z_k \in \mathbf{R}^{n\times n-m_k}$ is a basis of $\mathrm{span}(S_k)^\perp$. Then from (4), we see that $B_kS_k = Y_k$ and $B_kZ_k = Z_k$. Now we prove that $B_k^{m_k}$ also satisfies these linear equations.

Firstly, we show by induction that $B_k^iS_k^i = Y_k^i$, where $S_k^i = (s_{k-m_k}, \ldots, s_{k-m_k+i-1})$ and $Y_k^i = (y_{k-m_k}, \ldots, y_{k-m_k+i-1})$, $i = 1, \ldots, m_k$.

Base case. For $i = 1$, we have $B_k^0 = I$, $S_k^1 = s_{k-m_k}$ and $Y_k^1 = y_{k-m_k}$, and hence

$$B_k^1S_k^1 = B_k^1s_{k-m_k} = s_{k-m_k} + \frac{(y_{k-m_k} - s_{k-m_k})\hat{s}_{k-m_k}^Ts_{k-m_k}}{\hat{s}_{k-m_k}^Ts_{k-m_k}} = y_{k-m_k} = Y_k^1.$$

Induction. Suppose that we have proved the claim for $i = l$. Then for $i = l + 1$,

$$B_k^{l+1}S_k^l = B_k^lS_k^l + \frac{(y_{k-m_k+l} - B_k^ls_{k-m_k+l})\hat{s}_{k-m_k+l}^TS_k^l}{\hat{s}_{k-m_k+l}^Ts_{k-m_k+l}} = Y_k^l,$$

where we used the hypothesis and the fact that by orthogonalization, $\hat{s}_{k-m_k+l}^TS_k^l = 0$.

In addition, we also have

$$B_k^{l+1}s_{k-m_k+l} = B_k^ls_{k-m_k+l} + \frac{(y_{k-m_k+l} - B_k^ls_{k-m_k+l})\hat{s}_{k-m_k+l}^Ts_{k-m_k+l}}{\hat{s}_{k-m_k+l}^Ts_{k-m_k+l}} = y_{k-m_k+l},$$

which shows that $B_k^{l+1}S_k^{l+1} = Y_k^{l+1}$ together with the equalities above. This completes the induction, and in particular shows that

$$B_k^{m_k}S_k = B_k^{m_k}S_k^{m_k} = Y_k^{m_k} = Y_k.$$

Secondly, we show that $B_k^{m_k}Z_k = Z_k$. To see this, notice that by $\mathrm{span}(\hat{s}_{k-m_k}, \ldots, \hat{s}_{k-1}) = \mathrm{span}(S_k)$, we have $\hat{s}_{k-i}^TZ_k = 0$ $(i = 1, \ldots, m_k)$, and hence

$$B_k^{m_k}Z_k = Z_k + \sum_{i=0}^{m_k-1} \frac{(y_{k-m_k+i} - B_k^is_{k-m_k+i})\hat{s}_{k-m_k+i}^TZ_k}{\hat{s}_{k-m_k+i}^Ts_{k-m_k+i}} = Z_k.$$

Finally, since (S_k, Z_k) is invertible, we see that the equation $B(S_k, Z_k) = (Y_k, Z_k)$ has a unique solution, and hence $B_k = B_k^{m_k}$. □

Proof of Theorem 7.

Proof. Below we use the same notation for the vector norm $\|\cdot\|$ on $\mathbf{R}^n$ and its induced matrix norm on $\mathbf{R}^{n\times n}$, *i.e.*, $\|A\| = \sup_{x\neq 0} \|Ax\|/\|x\|$ for any $A \in \mathbf{R}^{n\times n}$. Again, we partition the iteration counts into two subsets accordingly, with $K_{AA} = \{k_0, k_1, \ldots\}$ being those iterations that passes line 12, while $K_{KM} = \{l_0, l_1, \ldots\}$ being the rest that goes to line 14. Denote as y the unique fixed point of f in (1), where the uniqueness comes from contractivity of f.

The proof is completed by considering two scenarios separately. The first is when K_{KM} is *finite*, in which case the proof is identical to Theorem 6, as neither the contractivity nor the non-expansiveness comes into play after a finite number of iterations. The second is when K_{KM} is *infinite*, the proof of which is given below.

Suppose from now on that K_{KM} is an infinite set.

On one hand, for $k_i \in K_{AA}$, by Corollary 5 and norm equivalence on $\mathbf{R}^n$, we have $\|H_{k_i}\| \leq C'$ for some constant C' independent of the iteration count, and similarly $\|g_{k_i}\| \leq C''\|g_{k_i}\|_2 \leq C''D\bar{U}(i+1)^{-(1+\epsilon)}$. Hence

$$\begin{aligned}\|x^{k_i+1} - y\| &\leq \|x^{k_i} - y\| + \|H_{k_i}g_{k_i}\| \\ &\leq \|x^{k_i} - y\| + C'\|g_{k_i}\| \leq \|x^{k_i} - y\| + \underbrace{C'C''D\bar{U}(i+1)^{-(1+\epsilon)}}_{\epsilon'_{k_i}}. \end{aligned} \tag{39}$$

On the other hand, for $l_i \in K_{KM}$ $(i \geq 0)$, (17) does not hold anymore. Instead, we have by γ-contractivity that

$$\|x^{l_i+1} - y\| \leq \gamma\|x^{l_i} - y\| \leq \|x^{l_i} - y\|. \tag{40}$$

Hence by defining $\epsilon'_{l_i} = 0$, we again see from (39) and (40) that

$$\|x^{k+1} - y\| \leq \|x^k - y\| + \epsilon'_k, \tag{41}$$

with $\epsilon'_k \geq 0$ and $\sum_{k=0}^{\infty}\epsilon'_k = \sum_{i=0}^{\infty}\epsilon'_{k_i} < \infty$.

Now define $a_j = \sum_{l_j+1\leq k<l_{j+1}}\epsilon'_k = \sum_{l_j\leq k_i<l_{j+1}}\epsilon'_{k_i}$. Then we have $\sum_{j=0}^{\infty}a_j = \sum_{k=0}^{\infty}\epsilon'_k < \infty$, and in particular $\lim_{j\to\infty}a_j = 0$ and $0 \leq a_j \leq E'$ for some $E' > 0$. Then we have

$$\|x^{l_i} - y\| \leq \|x^{l_{i-1}+1} - y\| + \sum\nolimits_{l_{i-1}+1\leq k<l_i}\epsilon'_k \leq \gamma\|x^{l_{i-1}} - y\| + a_{i-1}. \tag{42}$$

By telescoping the above inequality, we immediately see that

$$\begin{aligned}\|x^{l_i} - y\| &\leq \gamma^i\|x^{l_0} - y\| + \sum_{k'=0}^{i-1}\gamma^{k'}a_{i-1-k'} \\ &\leq \gamma^i\|x^0 - y\| + E'\sum_{k'=\lfloor(i-1)/2\rfloor}^{i-1}\gamma^{k'} + \sum_{k'=0}^{\lfloor(i-1)/2\rfloor-1}a_{i-1-k'} \\ &\leq \gamma^i\|x^0 - y\| + \frac{E'}{1-\gamma}\gamma^{\lfloor(i-1)/2\rfloor} + \sum_{k'=\lceil(i-1)/2\rceil}^{\infty}a_{k'}\end{aligned} \tag{43}$$

where we used the fact that $l_0 = 0$ by Line 2 of Algorithm 3. In particular, by using the fact that $\gamma^i \to 0$ as $i \to \infty$, and that $\sum_{k'=k}^{\infty}a_{k'} \to 0$ as $k \to \infty$, we see that

$$\lim_{i\to\infty}\|x^{l_i} - y\| = 0. \tag{44}$$

Finally, for any $k > 0$, define $i_k = \text{argmax}_i \ \{l_i < k\}$. Then we have $\lim_{k\to\infty} i_k = \infty$ as K_{KM} is infinite, and moreover, $l_{i_k+1} \geq k$. Hence we obtain that

$$\begin{aligned} \|x^k - y\| &\leq \|x^{l_{i_k}+1} - y\| + \sum\nolimits_{l_{i_k}+1\leq k'\leq k-1} \epsilon'_{k'} \\ &\leq \gamma\|x^{l_{i_k}} - y\| + a_{i_k}, \end{aligned} \tag{45}$$

from which and (44) we immediately conclude that

$$\lim_{k\to\infty} \|x^k - y\| = 0, \tag{46}$$

i.e., $\lim_{k\to\infty} x^k = y$, where y is the unique solution of (1). This completes our proof. □