SIAM J. Optim. © 2020 Society for Industrial and Applied Mathematics
Vol. 30, No. 4, pp. 3170–3197

GLOBALLY CONVERGENT TYPE-I ANDERSON ACCELERATION FOR NONSMOOTH FIXED-POINT ITERATIONS*

JUNZI ZHANG†, BRENDAN O'DONOGHUE‡, AND STEPHEN BOYD§

Abstract. We consider the application of the type-I Anderson acceleration to solving general nonsmooth fixed-point problems. By interleaving with safeguarding steps and employing a Powell-type regularization and a restart 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 nonexpansive. 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 2.1, one of the default solvers used in the convex optimization parser-solver CVXPY 1.0.

Key words. Anderson acceleration, fixed-point, nonexpansive, nonsmooth, stabilization, global convergence

AMS subject classifications. 47H05, 47H09, 49J52, 65K05, 90C25, 90C53

DOI. 10.1137/18M1232772

1. Introduction.

We consider solving the following general fixed-point problem:

$$\text{Find } x \in \mathbf{R}^n \text{ such that } x = f(x), \tag{1.1}$$

where $f : \mathbf{R}^n \to \mathbf{R}^n$ is potentially nonsmooth. Unless otherwise stated, we assume throughout the paper that f is nonexpansive (in 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 (1.1) is nonempty. With these assumptions, (1.1) 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 $x^\star \in X$ [45]. In one sense, our goal is to accelerate the vanilla KM algorithm.

Fixed-point problems such as (1.1) arise ubiquitously in mathematics, natural science, and social science. For example, to find a Nash equilibrium in a multiplayer game, one can reformulate it as a monotone inclusion problem under mild assumptions on the utility functions [11], which can then be further reformulated as a fixed-point problem of the corresponding (nonexpansive) resolvent or Cayley operator [45]. 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 (1.1), where $f : \mathbf{R}^n \to \mathbf{R}^n$ is the iteration mapping. When the optimization problem is convex, f is typically nonexpansive, and the solution set of the fixed-point problem is the same as that of the original optimization problem, or closely related to it. Another related example is an infinite-horizon discounted Markov decision process (MDP), in which the optimal policy can be found by solving the fixed-point problem of the associated Bellman

*Received by the editors January 2, 2019; accepted for publication (in revised form) June 26, 2020; published electronically November 18, 2020.
https://doi.org/10.1137/18M1232772

†ICME, Stanford University, Stanford, CA 94305 USA (junziz@stanford.edu).

‡DeepMind, Google, London, UK (bodonoghue85@gmail.com).

§Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (boyd@stanford.edu).

operator, which is not nonexpansive (in ℓ_2-norm) but is contractive in ℓ_∞-norm. Such scenarios are also discussed in section 4.2.3 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 preconditioning and step-size line search are the two most commonly used generic approaches to accelerate the convergence of the KM method [21]. 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 a differentiable objective function $F(x)$, Newton, quasi-Newton, and accelerated gradient descent methods (e.g., Nesterov's [32], and more recently, [17]) can then be used to reduce the overall iteration complexity at the expense of an increased cost in each step [30]. For more general f, semismooth Newton [2, 59] and B-differentiable (quasi-)Newton [35, 26], 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 or Newton-type acceleration steps, are designed to enjoy a smaller per-iteration cost while maintaining fast convergence in practice [47, 52].

Nevertheless, to our knowledge, apart from (pure) preconditioning 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 requires additional assumptions, e.g., some kind of differentiability around the solution [31, 15], symmetry of the Jacobian of f [28, 29], or symmetry of the approximate Jacobians in the algorithm [64, 65]. 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 nonexpansiveness, thus guaranteeing improvement of a much larger class of algorithms ruled out by existing methods.

To achieve this goal, we propose to solve (1.1) using the type-I (also called "good") Anderson acceleration (AA-I) [19], a natural yet underdeveloped variant of the original Anderson acceleration (AA), also known as the type-II Anderson acceleration (AA-II) [4]. 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 [19] and our preliminary benchmark tests of SCS version 2 show that although AA-I outperforms AA-II in many cases (earning the name "good"), it also suffers more from instability. Moreover, few convergence analyses 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 either are local [20, 44, 54] or assume certain nonsingularity (e.g., contractivity) conditions [47, 46, 49]. Another goal of this paper is hence to provide modifications that lead to a stabilized AA-I with convergence beyond differentiability, locality, and nonsingularity. As a result, we obtain global convergence to a fixed-point with no additional assumptions apart from nonexpansiveness.

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 [4]. The earliest problem that AA dealt with was nonlinear integral equations. Later, developed by two different communities [41, 42], AA enjoyed wide applications 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, respectively. In contrast, AA is not well known in the optimization community. As far as we know, it was not until [19] connected it with Broyden's (quasi-Newton) methods that some applications of AA to optimization algorithms, including expectation-maximization (EM), alternating nonnegative least squares, and alternating projections (APs), emerged [56, 24, 25]. More recently, applications are further extended to machine learning and control problems, including K-means clustering [63], robot localization [37], and computer vision [48]. For a thorough treatment of the history of AA, we refer the readers to [10].

There has been a rich literature on applications of AA to specific problems, especially within the field of computational chemistry and physics [3, 40]. An emerging literature on applications to optimization-related problems has also been 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 AA-II. For solving general fixed-point problems (or equivalently, nonlinear equations), perhaps the work most related to ours is [20] and [44], of 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, both presuming continuous differentiability of f in (1.1) around the solutions. By assuming in addition contractivity of f, slightly stronger and cleaner local linear convergence of the original AA-II can be obtained [54]. A similar analysis for noise-corrupted f was later conducted in [53].

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 the Arnoldi method [56, 38]. More recently, a regularized variant of full-memory AA-II was rediscovered as regularized nonlinear acceleration in [47], 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 method is then extended to accelerate stochastic [46] and momentum-based [49] 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 nonsingularity (or contractivity), which is missing from the entire AA literature. However, we remark that the proposed stabilized AA-I method in this paper has now been adopted and generalized to various different applications and settings with convergence guarantees in a few follow-up works, including self-consistent field (SCF) computation [58], nonconvex statistical learning [23], and distributed optimization [51].

Outline. In section 2, we introduce the original AA-I [19] and discuss its relation to quasi-Newton methods. In section 3, we propose a stabilized AA-I with Powell-type regularization, restart checking, and safeguarding steps. A self-contained convergence analysis of the stabilized AA-I is given in section 4, together with several example applications. Finally, we demonstrate the effectiveness of our proposed algorithm with various numerical examples in section 5, followed by extensions to our results in section 6.

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 [4] to [19], 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 2.1, 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 two most recent steps, and the weights are predetermined, which leads to sublinear convergence for nonexpansive mappings in general [45], and linear convergence under certain additional assumptions [8].

Algorithm 2.1 Anderson acceleration prototype (AA).

```
1: Input: initial point $x_0$, fixed-point mapping $f : \mathbf{R}^n \to \mathbf{R}^n$.
2: for $k = 0, 1, \dots$ do
3:     Choose $m_k$ (e.g., $m_k = \min\{m, k\}$ for some integer $m \geq 0$).
4:     Select weights $\alpha_j^k$ 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 called the memory in iteration k, and the next iterate is a linear combination of the images of the last $m_k + 1$ iterates under the mapping f. Based on the choices of the weights α_j^k in line 4 of Algorithm 2.1, AA is classified into two subclasses [19], namely AA-I and AA-II. The terminology indicates a close relationship between AA and quasi-Newton methods, as we will elaborate in more detail 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 [4], for each iteration $k \geq 0$, we solve the following least squares problem with a normalization constraint:

$$
\begin{array}{ll}
\text{minimize} & \left\| \sum_{j=0}^{m_k} \alpha_j g(x^{k-m_k+j}) \right\|_2^2 \\
\text{subject to} & \sum_{j=0}^{m_k} \alpha_j = 1,
\end{array} \tag{2.1}
$$

with variable $\alpha = (\alpha_0, \dots, \alpha_{m_k})$. The weight vector $\alpha^k = (\alpha_0^k, \dots, \alpha_{m_k}^k)$ in line 4 of Algorithm 2.1 is then chosen as the solution to (2.1). 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 (2.1) 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 2.1.

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 (2.1) can be efficiently solved as an unconstrained least squares problem by a simple variable elimination [56]. More explicitly, we can reformulate (2.1) as follows:

$$\text{minimize} \quad \|g_k - Y_k \gamma\|_2 \tag{2.2}$$

with variable $\gamma = (\gamma_0, \dots, \gamma_{m_k-1})$. Here $g_i = g(x^i)$, $Y_k = [y_{k-m_k} \ \cdots \ y_{k-1}]$ 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 $1 \le i \le m_k - 1$ and $\alpha_{m_k} = 1 - \gamma_{m_k-1}$.

Assuming for now that Y_k has full column rank and the solution γ^k to (2.2) 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 $S_k = [s_{k-m_k} \ \cdots \ s_{k-1}]$, $s_i = x^{i+1} - x^i$ 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 multisecant condition $H_k Y_k = S_k$ [19, 56] 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 [12] of I satisfying the inverse multisecant condition.

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 [22].

In the same spirit, we define the type-I AA (AA-I) [19], in which we find an approximate Jacobian B_k of g minimizing $\|B_k - I\|_F$ subject to the multisecant condition $B_k S_k = Y_k$. Assuming for now that S_k is full column rank, we obtain (by symmetry) that

$$B_k = I + (Y_k - S_k)(S_k^T S_k)^{-1} S_k^T, \tag{2.3}$$

and the update scheme is defined as

$$x^{k+1} = x^k - B_k^{-1} g_k, \tag{2.4}$$

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

A direct application of the Woodbury matrix identity shows that

$$B_k^{-1} = I + (S_k - Y_k)(S_k^T Y_k)^{-1} S_k^T, \tag{2.5}$$

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, (2.4) can be rewritten as

$$x^{k+1} = x^k - g_k - (S_k - Y_k)\tilde{\gamma}^k = f(x^k) - \sum_{i=0}^{m_k-1} \tilde{\gamma}_i^k \left(f(x^{k-m_k+i+1}) - f(x^{k-m_k+i})\right), \tag{2.6}$$

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 2.1: 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 $1 \le i \le m_k - 1$ 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 instead of matrix-matrix multiplication.

For easier reference, we detail AA-I in Algorithm 2.2. As our focus is on the more numerically efficient limited-memory versions, we also specify a maximum-memory parameter m in the algorithm.

Algorithm 2.2 Type-I Anderson acceleration (AA-I-m).

```
1: Input: initial point $x_0$, fixed-point mapping $f : \mathbf{R}^n \to \mathbf{R}^n$, max-memory $m > 0$.
2: for $k = 0, 1, \dots$ do
3:     Choose $m_k \le m$ (e.g., $m_k = \min\{m, k\}$ for some integer $m \ge 0$).
4:     Compute $\tilde{\gamma}^k = (S_k^T Y_k)^{-1}(S_k^T g_k)$.
5:     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 $\alpha_{m_k}^k = 1 - \tilde{\gamma}_{m_k-1}^k$.
6:     $x^{k+1} = \sum_{j=0}^{m_k} \alpha_j^k f(x^{k-m_k+j})$.
7: 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 address 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 2.2) to stabilize its convergence. We begin by introducing a Powell-type regularization to ensure the nonsingularity of B_k. We then introduce a simple restart 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 safeguarding steps that check the decrease in the residual norm, with which the modifications altogether lead to global convergence to a solution of (1.1), as we will show in section 4.

Rank-one update. To motivate the modifications, we take a step back to the update formula (2.4) and formalize a closer connection between AA-I and the type-I Broyden's method in terms of a rank-one update. The counterpart result has been proved for AA-II in [44].

PROPOSITION 3.1. *Suppose that S_k is full rank; then B_k in* (2.3) *can be computed inductively from $B_k^0 = I$ as follows:*

$$B_k^{i+1} = B_k^i + \frac{(y_{k-m_k+i} - B_k^i s_{k-m_k+i})\hat{s}_{k-m_k+i}^T}{\hat{s}_{k-m_k+i}^T s_{k-m_k+i}}, \quad i = 0, \ldots, m_k - 1, \tag{3.1}$$

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

$$\hat{s}_i = s_i - \sum_{j=k-m_k}^{i-1} \frac{\hat{s}_j^T s_i}{\hat{s}_j^T \hat{s}_j} \hat{s}_j, \quad i = k - m_k, \ldots, k-1. \tag{3.2}$$

The proof of Proposition 3.1 is by induction and to fix B_k by its restrictions to span(S_k) and its orthogonal complement, respectively. The detailed proof can be found in the longer version of this paper [62] and is omitted here due to space limits.

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 (3.1). The idea is to specify a parameter $\bar{\theta} \in (0, 1)$ and simply replace y_{k-m_k+i} in (3.1) with

$$\tilde{y}_{k-m_k+i} = \theta_k^i y_{k-m_k+i} + (1 - \theta_k^i) B_k^i s_{k-m_k+i}, \tag{3.3}$$

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

$$\phi_{\bar{\theta}}(\eta) = \begin{cases} 1 & \text{if } |\eta| \geq \bar{\theta}, \\ \frac{1 - \mathbf{sign}(\eta)\bar{\theta}}{1-\eta} & \text{if } |\eta| < \bar{\theta} \end{cases} \tag{3.4}$$

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 [39], but we redefine η_k to take the orthogonalization into consideration. Similar ideas have also been introduced in [47] and [24] 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 $\bar{\theta} = 0$. 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 nonsingularity of the modified B_k, which also indicates how $\bar{\theta}$ trades off between stability and efficiency.

LEMMA 3.2. *Suppose $\{s_i\}_{i=k-m_k}^{k-1}$ and $\{y_i\}_{i=k-m_k}^{k-1}$ are arbitrary sequences in $\mathbf{R}^n$. Define $B_k = B_k^{m_k}$ inductively from $B_k^0 = I$ as*

$$B_k^{i+1} = B_k^i + \frac{(\tilde{y}_{k-m_k+i} - B_k^i s_{k-m_k+i})\hat{s}_{k-m_k+i}^T}{\hat{s}_{k-m_k+i}^T s_{k-m_k+i}}, \quad i = 0, \ldots, m_k - 1, \tag{3.5}$$

with $\hat{s}_{k-m_k+i}$ and $\tilde{y}_{k-m_k+i}$ defined as in (3.2) and (3.3), 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 nonsingularity 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:

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

again with $\hat{s}_{k-m_k+i}$ and $\tilde{y}_{k-m_k+i}$ defined as in (3.2) and (3.3), 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 section 2.2 for AA-II.

3.2. Restart checking. In this section, we introduce a restart checking strategy proposed in [20], which was used to establish local convergence results for (full-memory) AA-I. Here we instead use it to establish uniform bounds on the approximate (inverse) Jacobians for the practical limited-memory scenarios, which turns out to be essential to the final global convergence, as we will see in section 4.

Notice that the update formula (3.5) 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 $i = 0, \dots, m_k - 1$, 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 3.2 to deduce that B_{k-m_k+i} is invertible.

This means that the only case when the updates in (3.5) 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 restart 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 $m_0 = 0$:

$$
(3.7) \quad m_k = m_{k-1} + 1. \text{ If } m_k = m + 1 \text{ or } \|\hat{s}_{k-1}\|_2 < \tau \|s_{k-1}\|_2, \text{ then reset } m_k = 1.
$$

Here $\tau \in (0, 1)$ is prespecified. The main idea is to make sure that $\hat{s}_k \neq 0$ whenever s_k is so, which ensures that the modified updates (3.5) and (3.6) 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 3.3. *Assume the same conditions as in Lemma* 3.2 *and in addition that* $\|y_i\|_2 \le 2\|s_i\|_2$ *for* $i = k - m_k, \ldots, k-1$ *and* m_k *is chosen by rule* (3.7). *Then we have* $\|B_k\|_2 \le 3(1+\bar{\theta}+\tau)^m/\tau^m - 2$ *for all* $k \ge 0$.

Proof. Notice that by rule (3.7), we have $\|\hat{s}_k\|_2 \ge \tau\|s_k\|_2$ and $m_k \le m$ for all $k \ge 0$. Hence by (3.5), we have that

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

Noticing that $\|y_{k-m_k+i}\|_2 \le 2\|s_{k-m_k+i}\|_2$ for $i = 0, \ldots, m_k - 1$, we see that $\|B_k^{i+1}\|_2 \le \frac{1+\bar{\theta}+\tau}{\tau}\|B_k^i\|_2 + \frac{2(1+\bar{\theta})}{\tau}$, and hence by telescoping the above inequality and the fact that $\|B_k^0\|_2 = 1$, we conclude that $\|B_k\|_2 = \|B_k^{m_k}\|_2 \le 3(\frac{1+\bar{\theta}+\tau}{\tau})^m - 2$. This completes our proof. ∎

In sum, combining the modified updates with the restarting choice of m_k, the rank-deficiency problem mentioned at the end of section 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 3.4. *Under the same assumptions in Lemma* 3.3, *we have that*

$$\|H_k\|_2 \le \left(3\left(\frac{1+\bar{\theta}+\tau}{\tau}\right)^m - 2\right)^{n-1}/\bar{\theta}^m \quad \forall\ k \ge 0. \tag{3.8}$$

Proof. Denote the singular values of B_k as $\sigma_1 \ge \cdots \ge \sigma_n$. Then by Lemma 3.2, we have $\prod_{i=1}^n \sigma_i \ge \bar{\theta}^{m_k} \ge \bar{\theta}^m$. On the other hand, by Lemma 3.3, we have $\sigma_1 \le 3(1+\bar{\theta}+\tau)^m/\tau^m - 2$. Hence we obtain that

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

which finishes our proof. ∎

We remark that for type-II methods as in [44], 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 [40] and [24] have already suggested that cleaning memories from time to time improves performance significantly for SCF methods and EM-type algorithms, partially supporting our modification here.

Notice that when m_k is chosen by rule (3.7) and B_k is computed as in Lemma 3.2, we have $B_k^i = B_{k-m_k+i}$. This means that in iteration k, only a rank-one update (3.5) with $i = m_k - 1$ is needed, which yields $B_k = B_k^{m_k}$ from $B_{k-1} = B_k^{m_k-1}$. We can also remove the need of maintaining updates for B_k^i 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. Safeguarding 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 safeguard the decrease in residual norms $\|g\|_2$. In particular, we check if the current residual norm is sufficiently small and replace the AA-I trial update with the α-averaged (or KM) operator of f in (1.1) (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 [5] with constant periods and is observed to improve both accuracy and speed for a certain class of algorithms (e.g., SCF), although no theoretical guarantee for convergence is provided. Similar ideas have been applied to regularized AA [47] and the classical Broyden's methods [52] to seek smaller per-iteration costs without sacrificing much the acceleration effects. A related idea is discussed in the context of line search for general averaged operators in section 5 of [21].

It is worth noticing that the SuperMann framework in [52] can also be seen as a general globalization strategy for acceleration schemes. However, certain boundedness related to Lemma 3.3 and Corollary 3.4 in this paper has to be made as an assumption (cf. Assumption II there).

The resulting algorithm, combining all the aforementioned tricks, is summarized as Algorithm 3.1. Here, lines 4–8 perform restart checking (rule (3.7)) described in section 3.2, lines 9–11 perform the Powell-type regularization (update (3.5)) described in section 3.1, and lines 12–14 execute the safeguarding strategy described above. As mentioned at the end of section 3.2, only a rank-one update of (3.5) from $i = m_k - 1$ is performed in iteration k, in which case the subscript $k - m_k + i$ becomes $k - 1$. We remark that the right-hand side of the safeguarding step in line 12 can indeed be replaced with any positive sequence $\delta_{n_{AA}}$ satisfying $\sum_{n_{AA}=0}^{\infty} \delta_{n_{AA}} < \infty$ without breaking the global convergence below. We also remark that when the fixed-point problem (1.1) arises from an unconstrained optimization problem, comparisons of the objective function values may also be included in the safeguard to better guide the behavior of acceleration (see, e.g., [23] and more generally [43, 18]).

Notice that in line 5 of Algorithm 3.1, instead of defining $s_{k-1} = x^k - x^{k-1}$ and $y_{k-1} = g(x^k) - g(x^{k-1})$ as in section 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 section 3.2, which makes it possible to get rid of maintaining an update for B_k.

We remark that the assumptions in Lemma 3.2, Lemma 3.3, and Corollary 3.4 all hold for Algorithm 3.1 unless a solution is reached and the problem is solved, despite the fact that the updates are modified in line 5 and the safeguarding strategy is introduced in lines 12–14. This comes immediately from the arbitrariness of the update sequences $\{s_i\}_{i=k-m_k}^{k-1}$ and $\{y_i\}_{i=k-m_k}^{k-1}$, the observation that the updates are well-defined unless a solution is reached and the fact that f is nonexpansive (and hence g is 2-Lipschitz, which implies that $\|y_i\|_2 \le 2\|s_i\|_2$ for $i = k - m_k, \dots, k-1$) (cf. Lemmas 3.2 and 3.3). Formally, we have the following corollary.

COROLLARY 3.5. *In Algorithm* 3.1, *unless a solution to* (1.1) *is found in finite steps, the inequality* (3.8) *holds for all* $k \ge 0$, *and the condition number of* H_k *is uniformly bounded by* $\mathbf{cond}(H_k) \le (3(\frac{1+\bar{\theta}+\tau}{\tau})^m - 2)^n/\bar{\theta}^m$.

The proof is a simple combination of the results in Lemma 3.3 and Corollary 3.4.

4. Global convergence and example applications.

In this section, we give a self-contained proof for the global convergence of Algorithm 3.1, and then present example applications of the global convergence result.

4.1. Analysis of global convergence.

We temporarily assume for simplicity that a solution to (1.1) is *not* found in finite steps. The proof can be divided into three steps. First, 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 show that x^k converges to some solution to (1.1). We note that many of the arguments are adapted from the proofs in [16].

Algorithm 3.1 Stabilized 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 $\bar{\theta},\ \tau,\ \alpha \in (0,1)$, safeguarding 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 $k = 1,\ 2,\ \dots$ do
4:      $m_k = m_{k-1} + 1$.
5:      Compute $s_{k-1} = \tilde{x}^k - x^{k-1}$, $y_{k-1} = g(\tilde{x}^k) - g(x^{k-1})$.      ▷ $H_k$ updates prep
6:      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:      If $m_k = m + 1$ or $\|\hat{s}_{k-1}\|_2 < \tau \|s_{k-1}\|_2$      ▷ Restart checking
8:          reset $m_k = 1$, $\hat{s}_{k-1} = s_{k-1}$, and $H_{k-1} = I$.
9:      Compute $\tilde{y}_{k-1} = \theta_{k-1} y_{k-1} - (1 - \theta_{k-1}) g_{k-1}$      ▷ Powell regularization
10:         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:     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:     If $\|g_k\|_2 \le D\bar{U}(n_{AA} + 1)^{-(1+\epsilon)}$      ▷ Safeguard checking
13:         $x^{k+1} = \tilde{x}^{k+1}$, $n_{AA} = n_{AA} + 1$.
14:     else $x^{k+1} = f_\alpha(x^k)$.
15: end for
```

We begin by noticing that x^{k+1} equals either $x^k - H_k g_k$ or $f_\alpha(x^k)$, depending on whether the checking in line 12 of Algorithm 3.1 passes or not. We partition the iteration counts into two subsets accordingly, with $K_{AA} = \{k_0, k_1, \dots\}$ being those iterations passing line 12 and $K_{KM} = \{l_0, l_1, \dots\}$ being the rest going to line 14.

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

For $k_i \in K_{AA}$ $(i \ge 0)$, by Corollary 3.5, we have $\|H_{k_i}\|_2 \le C$ for some constant C independent of the iteration count, and hence

$$\begin{aligned} \|x^{k_i+1} - y\|_2 &\le \|x^{k_i} - y\|_2 + \|H_{k_i} g_{k_i}\|_2 \\ &\le \|x^{k_i} - y\|_2 + C\|g_{k_i}\|_2 \le \|x^{k_i} - y\|_2 + CD\bar{U}(i+1)^{-(1+\epsilon)}. \end{aligned} \tag{4.1}$$

For $l_i \in K_{KM}$ $(i \ge 0)$, since f is nonexpansive, by Proposition 4.25(iii) in [7] or inequality (5) in [45], we have that

$$\|x^{l_i+1} - y\|_2^2 \le \|x^{l_i} - y\|_2^2 - \alpha(1-\alpha)\|g_{l_i}\|_2^2 \le \|x^{l_i} - y\|_2^2. \tag{4.2}$$

By telescoping the above inequalities, we obtain that

$$\|x^k - y\|_2 \le \|x^0 - y\|_2 + CD\bar{U}\sum_{i=0}^{\infty}(i+1)^{-(1+\epsilon)} = E < \infty, \tag{4.3}$$

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

Hence by squaring both sides of (4.1), we obtain that

$$\|x^{k_i+1} - y\|_2^2 \le \|x^{k_i} - y\|_2^2 + \underbrace{(CD\bar{U})^2(i+1)^{-(2+2\epsilon)} + 2CDE\bar{U}(i+1)^{-(1+\epsilon)}}_{=\epsilon_{k_i}}. \tag{4.4}$$

Combining (4.2) and (4.4), we see that

$$\alpha(1-\alpha)\sum_{i=0}^{\infty}\|g_{l_i}\|_2^2 \le \|x^0-y\|_2^2+\sum_{i=0}^{\infty}\epsilon_{k_i}<\infty, \tag{4.5}$$

and hence $\lim_{i\to\infty}\|g_{l_i}\|_2=0$. Noticing that $\|g_{k_i}\|_2\le D\bar{U}(i+1)^{-(1+\epsilon)}$ by line 12 of Algorithm 3.1, we also have $\lim_{i\to\infty}\|g_{k_i}\|_2=0$. Hence we see that

$$\lim_{k\to\infty}\|g_k\|_2=0. \tag{4.6}$$

Also notice that by defining $\epsilon_{l_i}=0$, we again see from (4.2) and (4.4) that

$$\|x^{k+1}-y\|_2^2\le\|x^k-y\|_2^2+\epsilon_k \tag{4.7}$$

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

Notice that in the above derivation of (4.5)–(4.7), we have implicitly assumed that both K_{AA} and K_{KM} are infinite. However, the cases when either of them is finite are 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 $\|x^k-y\|_2\ge 0$, 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 $\|x^{j_{i_0}}-y\|_2\le\underline{u}+\delta$ and $\sum_{k=j_{i_0}}^{\infty}\epsilon_k\le\delta$. This, together with (4.7), implies that for any $k\ge j_{i_0}$,

$$\|x^k-y\|_2^2\le\|x^{j_{i_0}}-y\|_2^2+\sum_{k=j_{i_0}}^{\infty}\epsilon_k\le\underline{u}^2+2\delta\underline{u}+\delta^2+\delta, \tag{4.8}$$

and in particular, we have $\limsup_{k\to\infty}\|x^k-y\|_2^2\le\liminf_{k\to\infty}\|x^k-y\|_2^2+\delta(2\underline{u}+\delta+1)$. 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 (1.1), 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 the 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 $y_1\ne y_2$, both of which must be fixed-points of f. This is because by (4.6), we have $0=\lim_{i\to\infty}\|g(x^{k_i'})\|_2=\|g(y_1)\|_2$ and $0=\lim_{i\to\infty}\|g(x^{l_i'})\|_2=\|g(y_2)\|_2$, where we used the fact that f is nonexpansive 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 $\|x^k-y\|_2^2-\|y\|_2^2=\|x^k\|_2^2-2y^Tx^k$, we have

$$\lim_{i\to\infty}\|x^{k_i'}\|_2^2=\lim_{k\to\infty}\|x^k-y\|_2^2-\|y\|_2^2+2y^T\lim_{i\to\infty}x^{k_i'}=\alpha(y)-\|y\|_2^2+2y^Ty_1,$$
$$\lim_{i\to\infty}\|x^{l_i'}\|_2^2=\lim_{k\to\infty}\|x^k-y\|_2^2-\|y\|_2^2+2y^T\lim_{i\to\infty}x^{l_i'}=\alpha(y)-\|y\|_2^2+2y^Ty_2.$$

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 (4.6) that $0=\lim_{k\to\infty}\|g(x^k)\|_2=\|g(\bar{x})\|_2$. Here we again used the fact that f is nonexpansive, 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 4.1. *Suppose that $\{x^k\}_{k=0}^{\infty}$ is generated by Algorithm* 3.1*; then we have* $\lim_{k\to\infty} x^k = x^\star$*, where $x^\star = f(x^\star)$ is a solution to* (1.1).

Remark 4.2. Many parts of steps 2 and 3 can be replaced by directly invoking Theorem 3.8 in [16]. Step 3 also follows similarly to the proofs of [13, Corollary 3.3.3] and [6, Theorem 2.16(ii)]. The details are shown to make the proof self-contained and to facilitate the discussions about extensions in section 6.

4.2. Examples. Below we present several example problems and the corresponding (unaccelerated) algorithms used to solve them, to which Theorem 4.1 can be applied. The major focus is on optimization and decision-making problems and algorithms, where f in (1.1) comes from the iterative algorithms used to solve them. For each example, we specify the concrete form of f, verify its nonexpansiveness, and check the equivalence between the fixed-point problem and the original problem.

4.2.1. Proximal gradient descent. Consider the following problem:

$$\text{minimize} \quad F_1(x) + F_2(x), \tag{4.9}$$

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., $x^{k+1} = \mathbf{prox}_{\alpha F_2}(x^k - \alpha\nabla F_1(x^k))$, where $\alpha \in (0, 2/L)$. In our notation, the fixed-point mapping is $f(x) = \mathbf{prox}_{\alpha F_2}(x - \alpha\nabla F_1(x))$. For a proof of nonexpansiveness for f and the equivalence between the fixed-point problem and the original optimization problem (4.9), see [36].

Gradient descent. When $F_2 = 0$, proximal gradient descent reduces to vanilla gradient descent (GD) 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. When $F_2(x) = \mathcal{I}_{\mathcal{K}}(x)$, with $\mathcal{K}$ being a nonempty closed and convex set, problem (4.9) reduces to a constrained optimization problem. Accordingly, proximal gradient descent reduces to projected gradient descent (PGD), 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. When $F_1(x) = \frac{1}{2}\mathbf{dist}(x, D)^2$ and $F_2(x) = \mathcal{I}_C(x)$, with $C,\ D$ being nonempty closed convex sets and $C \cap D \neq \emptyset$. The problem (4.9) 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 AP, i.e., $x^{k+1} = \Pi_C\Pi_D(x^k)$, with $f(x) = \Pi_C\Pi_D(x)$.

Iterative shrinkage-thresholding algorithm. When $F_2 = \mu\|x\|_1$, the problem reduces to sparsity-regularized regression. 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 $S_\kappa(x)_i = \mathbf{sign}(x_i)(|x_i| - \kappa)_+$ $(i = 1, \ldots, n)$ is the shrinkage operator. The fixed-point mapping here is $f(x) = S_{\alpha\mu}(x - \alpha\nabla F(x))$.

4.2.2. Douglas–Rachford splitting. Consider the following problem:

$$\text{find } x \text{ such that } \ 0 \in (A + B)(x), \tag{4.10}$$

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:

$$z^{k+1} = f(z^k) = z^k/2 + C_A C_B(z^k)/2, \tag{4.11}$$

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 nonexpansive 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 nonexpansive) operator. The connection between (4.10) and (4.11) is established by the fact that x solves (4.10) if and only if z solves (4.11) and $x = R_B(z)$, where R_B is the resolvent operator of B, i.e., $R_B(x) = (I + \alpha B)^{-1}$.

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 (4.10) and (4.11), can all be found in [45].

Notice that whenever z^k converges to a fixed-point of (4.11) (not necessarily following the DRS iteration (4.11)), $x^k = R_B(z^k)$ converges to a solution of problem (4.10), where $R_B(x) = (I + \alpha B)^{-1}(x)$ is the resolvent of B. This comes immediately from the equivalence between (4.10) and (4.11) and the fact that R_B is nonexpansive [45] and hence continuous. Together with Theorem 4.1, this ensures that the application of Algorithm 3.1 to the DRS fixed-point problem (4.11) leads to the convergence of $x^k = R_B(z^k)$ to a solution of the original problem.

Consensus optimization. In consensus optimization (CO) [45], we seek to solve

$$\text{minimize} \quad \sum_{i=1}^m F_i(x), \tag{4.12}$$

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

$$\text{minimize} \quad \sum_{i=1}^m F_i(x_i) + \mathcal{I}_{\{x_1 = x_2 = \cdots = x_m\}}(x_1, x_2, \ldots, x_m), \tag{4.13}$$

the problem reduces to (4.10) 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, $C_{\partial F}(x) = 2\text{prox}_{\alpha F}(x) - x$ and $C_{\mathcal{N}_C}(x) = 2\Pi_C(x) - x$, we see that the DRS algorithm reduces to the following:

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

where $\bar{x}^k = \frac{1}{m}\sum_{i=1}^m x_i^k$, and the fixed-point mapping f is the mapping from $z^k = (z_1^k, \ldots, z_m^k)^T$ to $z^{k+1} = (z_1^{k+1}, \ldots, z_m^{k+1})^T$. As discussed above, x^{k+1} converges to the solution of (4.12) 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:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax + s = b, \quad s \in \mathcal{K}, \end{array} \tag{4.14}$$

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 [33]. To this end, one seeks to solve the associated self-dual homogeneous embedding (SDHE) system [61],

$$Qu = v, \quad (u, v)^T \in \mathcal{C} \times \mathcal{C}^*, \tag{4.15}$$

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 is

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

The SDHE system can then be further reformulated into (4.10) [52], with $A(u) = \mathcal{N}_{\mathcal{C}}(u)$, $B(u) = Qu$. Accordingly, DRS reduces to SCS [33], 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 [55] with a 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 nonexpansive (cf. the appendix in [33]).

Notice that with the transformations made, the equivalence and convergence properties of DRS cannot 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 $(I + Q)^{-1}(u + v) = u$ and $\Pi_{\mathcal{C}}((I + Q)^{-1}(u + v) - v) = u$, i.e., $Qu = v$ and $\Pi_{\mathcal{C}}(u - v) = u$. By Moreau decomposition [36], 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 $(u, v)^T \in \mathcal{C} \times \mathcal{C}^*$, we have $u^T v = u^T Qu = 0$ 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.

4.2.3. Contractive mappings in different norms. As we can see from (4.2) in the proof of Theorem 4.1, which does not hold for general norms, the ℓ_2-norm in the definition of nonexpansiveness is essential to our analysis of global convergence. Nevertheless, an expansive mapping in one norm may be nonexpansive 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 3.1 can still be guaranteed. Formally, we have the following theorem.

THEOREM 4.3. *Suppose that $\{x^k\}_{k=0}^{\infty}$ is generated by Algorithm* 3.1*, but with $\alpha = 1$, and instead of f being nonexpansive (in ℓ_2-norm) in* (1.1)*, 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* (1.1)*.*

The detailed proof can be found in the longer version of this paper [62] and is again skipped here due to space limits. 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. Consider solving a discounted 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 $s,\, s' \in \{1,\dots,S\}$ and $a \in \{1,\dots,A\}$.

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 policies $\mu : \{1,\dots,S\} \to \{1,\dots,A\}$, where $s_{t+1} \sim P(s_t,\mu(s_t),\cdot)$. One of the most basic algorithms used to solve this problem is the well-known value iteration (VI) 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:

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

In our notation, the fixed-point mapping $f(x) = T(x)$. A prominent property of T is that although not necessarily nonexpansive in ℓ_2-norm, it is γ-contractive under the l_∞-norm, i.e., $\|Tx - Ty\|_\infty \le \gamma\|x-y\|_\infty$.

By Theorem 4.3, the global convergence is still guaranteed when Algorithm 3.1 is applied to VI here. We also remark that it would be interesting to apply the accelerated VI to solving the MDP subproblems in model-based reinforcement learning algorithms (e.g., UCRL2 [27]), where the rewards r and transitions P are unknown.

Another example that falls into the scenario of Theorem 4.3 is the heavy-ball algorithm. Here we do not go into detail on this topic due to space limits, and interested readers may refer to the longer version of this paper [62].

5. Numerical results.

We are now ready to illustrate the performance of the AA 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 Catalina.

We compare the performance of three algorithms for each experiment: (a) the vanilla algorithm (e.g., gradient descent); (b) AA-I-m, Algorithm 2.2 with maximum memory m, choosing $m_k = \min\{m,k\}$; (c) AA-I-S-m, Algorithm 3.1 with maximum memory m. For each experiment, we show the convergence curves of one representative run against clock time (seconds) and iteration numbers, respectively. The codes for the experiments, including some further comparisons with other algorithms (e.g., AA-II and its regularized version [47], 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 at https://github.com/cvxgrp/nonexp_global_aa1. The proposed acceleration method is also being implemented in SCS 2.1 [34], one of the default solvers in CVXPY 1.0 [1]

5.1. Implementation details.

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

Matrix-free updates. In line 11 of Algorithm 3.1, 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 prespecified 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 start 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 (ENR)), we always choose $K_{\max}=1000$ and $tol=10^{-5}$.

Choice of hyper-parameters. Throughout the experiments, we use a single set of hyper-parameters to show the robustness of our algorithm (Algorithm 3.1). 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. We remark that further improvement might be obtained by adopting an adaptive and problem-dependent strategy for choosing the hyper-parameters.

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 safeguarding steps less often, making it closer to the original AA-I-m. The Powell regularization parameter should not be set too large, as it will empirically break down the acceleration effect. A large τ will force the algorithm to restart quite often, making it close to choosing the memory size $m=1$. A restart checking parameter τ between 0.001 and 0.1 and a memory size ranging from 2 to 50 are found to be reasonable choices.

5.2. Problem instances. We consider the following specific problem instances for the algorithms listed in section 4.2, ranging from statistics to control to game theory and so on. For each plot, AA-I-m is labeled as *aa*1, AA-I-S-m is labeled as *aa*1-*safe*, and the original (vanilla) algorithm is labeled as *original*. The residual norms are computed in the ℓ_2-norm, and the vertical axis in the plots is $\|g_k\|_2/\|g_0\|_2$. 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:

$$\text{minimize} \quad \frac{1}{m}\sum_{i=1}^{m}\log(1+\exp(-y_i\theta^T x_i))+\frac{\lambda}{2}\|\theta\|_2^2, \tag{5.1}$$

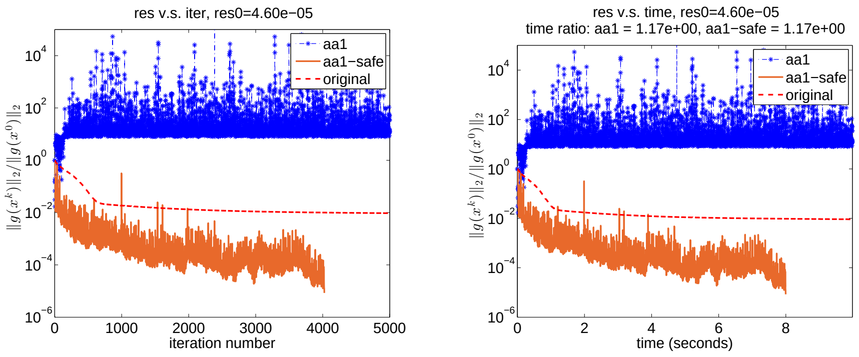

FIG. 1. *GD: Reg-Log. Left: residual versus iteration. Right: residual versus time.*

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 the 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 an 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 [47], and $X = [x_1, \dots, x_m]$. The results are shown in Figure 1. Here we set $K_{\max} = 5000$ to better demonstrate the improvement of our algorithm.

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.

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

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b, \quad x \in \mathcal{K}, \end{array} \tag{5.2}$$

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 deliberately choose a different (dual) formulation of (4.14) to show the flexibility of our algorithm, which can be easily mounted on top of vanilla algorithms.

As in SCS, (5.2) can be similarly formulated as the SDHE system (4.15), 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 notation 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 (5.2), following [33]. 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 $x^\star = \max(z^\star, 0)$, $r^\star = \max(-z^\star, 0)$, where the maximum is taken componentwisely. We then also generate $y^\star$ with `randn.m` and take $b = Ax^\star$, $c = A^T y^\star + r^\star$. 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.

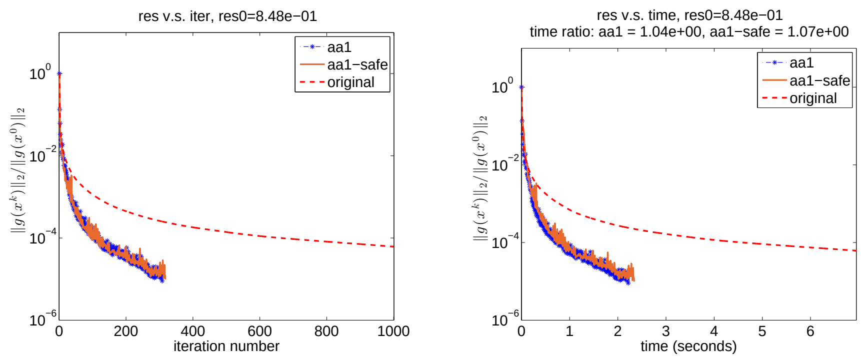

FIG. 2. *AP: LP as SDHE. Left: residual versus iteration. Right: residual versus time.*

In addition, as in SCS [33], we perform diagonal scaling on the problem data. More explicitly, we compute $\tilde{A} = D^{-1}AE^{-1}$ and accordingly scale b to be $\tilde{b} = D^{-1}b$ and c to be $\tilde{c} = E^{-1}c$. Here $D = \mathbf{diag}(\sum_{j=1}^n |a_{1j}|, \dots, \sum_{j=1}^n |a_{mj}|)$ and $E = \mathbf{diag}(\sum_{i=1}^m |\hat{a}_{i1}|, \dots, \sum_{i=1}^m |\hat{a}_{in}|)$, with $\hat{A} = (\hat{a}_{ij})_{m\times n} = D^{-1}A$. We can see that $\tilde{x}$ is a solution to (5.2) 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 2.

We can see that our algorithm AA-I-S-m compares 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.

PGD: Nonnegative least squares and convex-concave matrix game. We consider the following nonnegative least squares (NNLS) problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|Ax-b\|_2^2 \\ \text{subject to} & x \geq 0, \end{array} \tag{5.3}$$

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 a certain physical interpretation [14]. We consider the more challenging high dimensional case, i.e., $m < n$ [50]. The gradient of the objective function can be 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 3. Here we set $K_{\max} = 1500$ to better demonstrate the improvement of our algorithm.

We also consider a more specialized and structured problem: the 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 [9]:

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & u \geq 0, \quad \mathbf{1}^Tu = 1, \quad P^Tu \leq t\mathbf{1}, \end{array} \tag{5.4}$$

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.

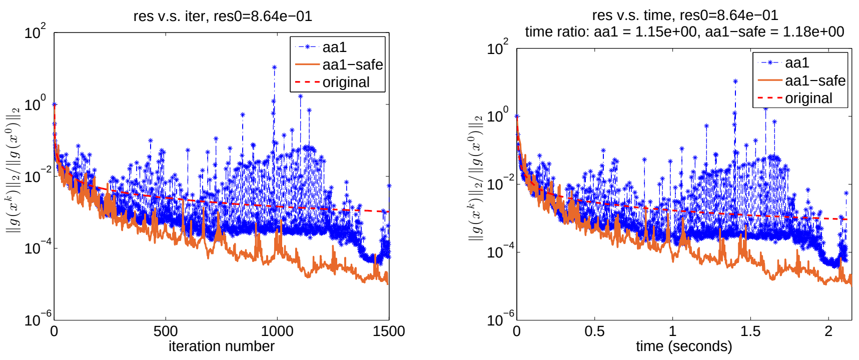

FIG. 3. *PGD: NNLS. Left: residual versus iteration. Right: residual versus time.*

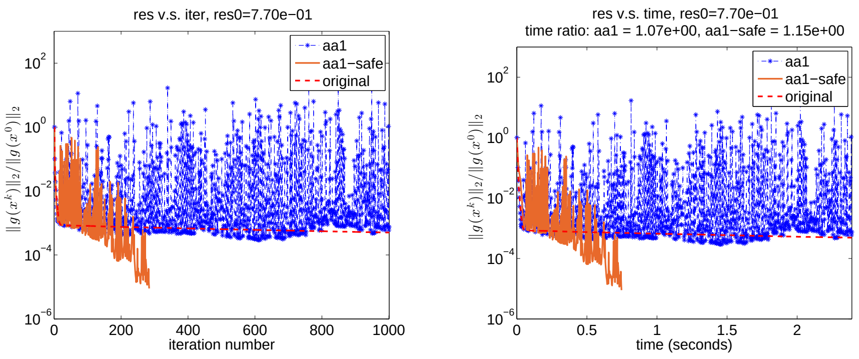

FIG. 4. *PGD: CCMG. Left: residual versus iteration. Right: residual versus time.*

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 $t = \|P^T u\|_\infty$. Hence the above LP can be further transformed into

$$\begin{array}{ll} \text{minimize} & t + \frac{1}{2}\|P^T u + s - t\mathbf{1}\|_2^2 \\ \text{subject to} & u \geq 0, \quad 1^T u = 1, \quad s \geq 0, \end{array} \tag{5.5}$$

where we introduce an additional (slack) variable $s \in \mathbf{R}^n$. Using the efficient projection algorithm onto the probability simplex set [57, 9], 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 $\tilde{A} = [P^T, I, -\mathbf{1}]$, in which I is the n-by-n identity matrix and $\mathbf{1} \in \mathbf{R}^n$ is the all-one vector. The results are summarized in Figure 4.

ISTA: Elastic net regression. We consider the following ENR problem [66]:

$$\text{minimize} \quad \frac{1}{2}\|Ax - b\|_2^2 + \mu\left(\frac{1-\beta}{2}\|x\|_2^2 + \beta\|x\|_1\right), \tag{5.6}$$

where $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^m$. In our experiments, we take $\beta = 1/2$ and $\mu = 10^{-5}\mu_{\max}$, where $\mu_{\max} = \|A^T b\|_\infty$ is the smallest value under which the ENR problem admits only the zero solution [33]. 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 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 $L = \lambda_{\max}(A^TA) + \mu/2$.

We again consider a harder high dimensional case, where $m = 1500$ and $n = 2000$. The data is generated similarly to the Lasso example in [33]. 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 $L = \|A^TA\|_2 + \mu/2$. The results are shown in Figure 5. Here we set the tolerance *tol* to 10^{-8} and $K_{\max}$ to 2500 to better exemplify the performance improvement of our algorithm in a relatively long run.

CO: Facility location. Consider the following facility location problem [60]:

$$\text{minimize} \quad \sum_{i=1}^{m} \|x - c_i\|_2, \tag{5.7}$$

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

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

where $\mathbf{prox}_{\|\cdot\|_2}(v) = (1 - 1/\|v\|_2)_+ v$ [36].

Notice that all the updates can be parallelized. In particular, in the MATLAB implementation no "for" loops are 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 initialization x^0 is again generated by `randn.m` and normalized to have a unit ℓ_2-norm. The results are summarized in Figure 6. Notice that here we again set the tolerance *tol* to 10^{-8} and we truncate the maximum iteration number to $K_{\max} = 500$ for better visualization.

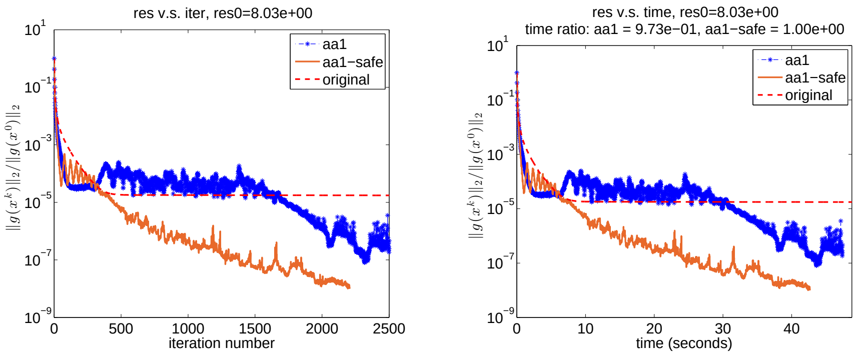

FIG. 5. *ISTA: ENR. Left: residual versus iteration. Right: residual versus time.*

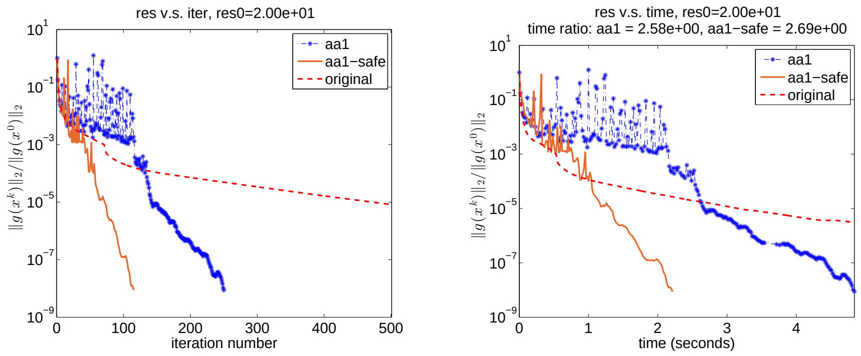

FIG. 6. *CO: facility location. Left: residual versus iteration. Right: residual versus time.*

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

SCS: Cone programs. Consider (4.14) with $\mathcal{K} = \mathbf{R}^m_+$ or $\{s \in \mathbf{R}^m \mid \|s_{1:m-1}\|_2 \le s_m\}$), i.e., a generic LP or second order cone program (SOCP). We solve it using a toy implementation of SCS, i.e., one without approximate projection, CG iterations, fine-tuned overrelaxation, and so on. However, we do perform diagonal scaling on the problem data as in the AP example above.

We make use of the following explicit formula for the projection onto the second order cone $\mathcal{K} = \{s \in \mathbf{R}^m \mid \|s_{1:m-1}\|_2 \le s_m\}$ [36]:

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

For both LP and SOCP, x^0 is initialized using `randn.m` and then normalized to have a unit ℓ_2-norm. We again follow [33] to generate data that ensures primal and dual feasibility of the original cone programs.

For LP, we choose $m = 2000$ and $n = 3000$. We generate A as a horizontal concatenation of `sprandn(m,⌊n/2⌋,0.1)` and an identity matrix of size $m \times \lfloor n/2 \rfloor$, added with a noise 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 componentwisely. We then also generate $x^\star$ using `randn.m` and take $b = Ax^\star + s^\star$ and $c = -A^T y^\star$. We set the tolerance *tol* to 10^{-8} and $K_{\max}$ to 5000 for better visualization.

For SOCP, we choose $m = 3000$ and $n = 5000$. 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 componentwisely. We then once again generate $x^\star$ using `randn.m` and take $b = Ax^\star + s^\star$ and $c = -A^T y^\star$. Here we use the default *tol* and $K_{\max}$.

The results are summarized in Figures 7 and 8.

VI: Markov decision process. As described in section 4.2.3, we consider solving a general random 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, thus making the improvement of AA more explicit.

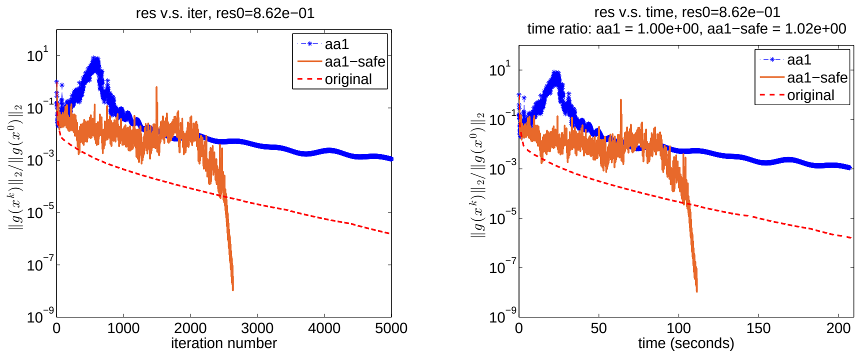

FIG. 7. *SCS: LP. Left: residual versus iteration. Right: residual norm versus time.*

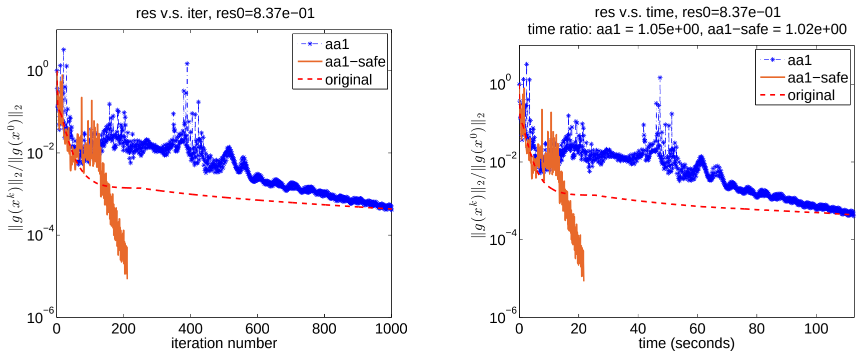

FIG. 8. *SCS: SOCP. Left: residual versus iteration. Right: residual versus time.*

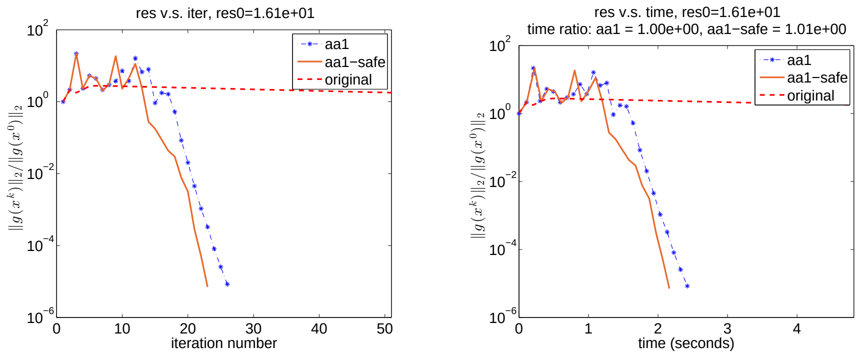

FIG. 9. *VI: MDP. Left: residual versus iteration. Right: residual versus time.*

The transition probability matrices $P_a \in \mathbf{R}^{S\times S}$, $a = 1, \dots, A$, 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 9. Notice that the maximum iteration $K_{\max}$ is set to 50 for better visualization.

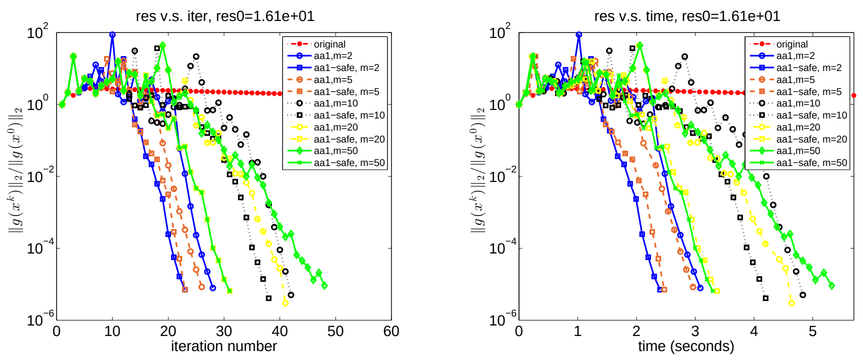

FIG. 10. *VI: memory effect. Left: residual versus iteration. Right: residual versus time.*

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 10. Notice that again the maximum iteration $K_{\max}$ is set to 50 for better visualization.

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.

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, $\|f(x) - y\|_2 \le \|x - y\|_2$ for any $x \in \mathbf{R}^n$. Obviously, nonexpansive 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 nonexpansiveness is only applied between an arbitrary point and a fixed-point of f in the proof of Theorem 4.1, and that the continuity of g can be relaxed to closedness, we immediately see that the same global convergence result holds 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 4.3 remains true if f is closed and the contractivity is assumed only between an arbitrary point and a fixed-point of f, i.e., $\|f(x) - f(y)\| \le \gamma\|x - y\|$ 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 6.1. *Suppose that $\{x^k\}_{k=0}^{\infty}$ is generated by Algorithm* 3.1, *and instead of f being nonexpansive (in ℓ_2-norm) in* (1.1), *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 solution to* (1.1). *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 a common fixed-point of all f_k's (assuming that it exists), i.e., find $x^\star \in \cap_{k\ge 0} X_k$ 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 f. However, a common fixed-point of all f_k's is still exactly an optimal solution to the original optimization problem and all f_k's have the same fixed-point set.

Assuming nonexpansiveness (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 4.1 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 4.1 still holds in this setting, as formally stated below.

COROLLARY 6.2. *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* 3.1*, 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* (1.1).

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 the fact that $\lim_{k\to\infty} \|g_k\|_2 = \lim_{k\to\infty} \|x^k - f_k(x^k)\|_2 = 0$, 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 section 4.2.1, i.e., $\bar{x} \in X$. Hence we see that the assumptions made in Corollary 6.2 all hold in this example, and hence global convergence of x^k is ensured.

A similar analysis can be carried out to reprove Theorem 4.3 in this setting.

Nevertheless, it remains open what assumptions are needed in general to obtain global convergence as in Theorems 4.1 and 4.3. 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. There are also cases when the norm in which nonexpansiveness holds changes as the iteration proceeds. We leave these as future works.

Nonexpansive mappings in non-Euclidean norms. Theorem 4.3 establishes global convergence for contractive mappings in arbitrary norms. It is hence natural to ask what happens if f is only nonexpansive (instead of contractive) in an arbitrary norm different from the ℓ_2-norm. A direct generalization holds when the ℓ_2-norm is replaced with an H-norm $\|\cdot\|_H$, defined as $\|x\|_H = \sqrt{x^T H x}$, where $H \in \mathbf{R}^{n\times n}$ is symmetric positive-definite. The general extensions are yet to be explored in future works.

Acknowledgments. The authors would like to thank Ernest Ryu, Tianyi Lin, Qingyun Sun, Michael Saunders, Zaiwen Wen, Anran Hu, Pontus Giselsson, Zico Kolter, and Panagiotis Patrinos for the fruitful discussions and feedback on an early draft of this paper. The authors also thank the associate editor, Wotao Yin, and the anonymous referees for detailed reviews and the insightful suggestions.

REFERENCES

[1] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, *A rewriting system for convex optimization problems*, J. Control Decis., 5 (2018), pp. 42–60.

[2] A. Ali, E. Wong, and J. Z. Kolter, *A semismooth Newton method for fast, generic convex programming*, in Proceedings of the 34th International Conference on Machine Learning 2017, pp. 70–79.

[3] H. An, X. Jia, and H. F. Walker, *Anderson acceleration and application to the three-temperature energy equations*, J. Comput. Phys., 347 (2017), pp. 1–19.

[4] D. G. Anderson, *Iterative procedures for nonlinear integral equations*, J. ACM, 12 (1965), pp. 547–560.

[5] A. S. Banerjee, P. Suryanarayana, and J. E. Pask, *Periodic Pulay method for robust and efficient convergence acceleration of self-consistent field iterations*, Chem. Phys. Lett., 647 (2016), pp. 31–35.

[6] H. H. Bauschke and J. M. Borwein, *On projection algorithms for solving convex feasibility problems*, SIAM Rev., 38 (1996), pp. 367–426.

[7] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, New York, 2011.

[8] H. H. Bauschke, D. Noll, and H. M. Phan, *Linear and strong convergence of algorithms involving averaged nonexpansive operators*, J. Math. Anal. Appl., 421 (2015), pp. 1–20.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[10] C. Brezinski, M. Redivo-Zagila, and Y. Saad, *Shanks sequence transformations and Anderson acceleration*, SIAM Rev., 60 (2018), pp. 646–669.

[11] L. M. Briceño-Arias and P. L. Combettes, *Monotone operator methods for Nash equilibria in non-potential games*, in Computational and Analytical Mathematics, D. H. Bailey, H. H. Bauschke, P. Borwein, F. Garvan, M. Théra, J. D. Vanderwerff, and H. Wolkowicz, eds., Springer, New York, 2013, pp. 143–159.

[12] C. G. Broyden, *A class of methods for solving nonlinear simultaneous equations*, Math. Comp., 19 (1965), pp. 577–593.

[13] A. Cegielski, *Iterative Methods for Fixed Point Problems in Hilbert Spaces*, Springer, Berlin, 2012.

[14] D. Chen and R. J. Plemmons, *Nonnegativity constraints in numerical analysis*, in The Birth of Numerical Analysis, World Scientific, 2010, pp. 109–139.

[15] Z. Chen, W. Cheng, and X. Li, *A global convergent quasi-Newton method for systems of monotone equations*, J. Appl. Math. Comput., 44 (2014), pp. 455–465.

[16] P. L. Combettes, *Quasi-Fejérian analysis of some optimization algorithms*, in Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications, D. Butnariu, Y. Censor, and S. Reich, eds., Stud. Comput. Math. 8, Elsevier, New York, 2001, pp. 115–152.

[17] W. Drummond and S. Duncan, *Accelerated Gradient Methods with Memory*, preprint, arXiv:1805.09077, 2018.

[18] A. Dutta, E. H. Bergou, Y. Xiao, M. Canini, and P. Richtárik, *Direct Nonlinear Acceleration*, preprint, arXiv:1905.11692, 2019.

[19] H. Fang and Y. Saad, *Two classes of multisecant methods for nonlinear acceleration*, Numer. Linear Algebra Appl., 16 (2009), pp. 197–221.

[20] D. M. Gay and R. B. Schnabel, *Solving systems of nonlinear equations by Broyden's method with projected updates*, Nonlinear Program., 3 (1978), pp. 245–281.

[21] P. Giselsson, M. Fält, and S. Boyd, *Line search for averaged operator iteration*, in Proceedings of the IEEE 55th Conference on Decision and Control, 2016, pp. 1015–1022.

[22] A. Griewank, *Broyden Updating, the Good and the Bad!*, Documenta Mathematica, Extra Volume: Optimization Stories, 2012, pp. 301–315.

[23] X. Guo, A. Hu, R. Xu, and J. Zhang, *Consistency and Computation of Regularized MLEs for Multivariate Hawkes Processes*, preprint, arXiv:1810.02955, 2018.

[24] N. C. Henderson and R. Varadhan, *Damped Anderson acceleration with restarts and monotonicity control for accelerating EM and EM-like algorithms*, J. Comput. Graph. Statist., 28 (2019), pp. 834–846.

[25] N. J. Higham and N. Strabić, *Anderson acceleration of the alternating projections method for computing the nearest correlation matrix*, Numer. Algorithms, 72 (2016), pp. 1021–1042.

[26] C. M. Ip and J. Kyparisis, *Local convergence of quasi-Newton methods for B-differentiable equations*, Math. Program., 56 (1992), pp. 71–89.

[27] T. Jaksch, R. Ortner, and P. Auer, *Near-optimal regret bounds for reinforcement learning*, J. Mach. Learn. Res., 11 (2010), pp. 1563–1600.
[28] D. Li and M. Fukushima, *A derivative-free line search and DFP method for symmetric equations with global and superlinear convergence*, Numer. Funct. Anal. Optim., 20 (1999), pp. 59–77.
[29] D. Li and M. Fukushima, *A globally and superlinearly convergent Gauss-Newton-based BFGS method for symmetric nonlinear equations*, SIAM J. Numer. Anal., 37 (1999), pp. 152–172.
[30] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, 4th ed., Springer, Cham, 2016.
[31] J. J. Moré and J. A. Trangenstein, *On the global convergence of Broyden's method*, Math. Comp., 30 (1976), pp. 523–540.
[32] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Springer, 2004.
[33] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, *Conic optimization via operator splitting and homogeneous self-dual embedding*, J. Optim. Theory Appl., 169 (2016), pp. 1042–1068.
[34] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, *SCS: Splitting Conic Solver*, version 2.1.2, https://github.com/cvxgrp/scs (2019).
[35] J. S. Pang, *Newton's method for B-differentiable equations*, Math. Oper. Res., 15 (1990), pp. 311–341.
[36] N. Parikh and S. Boyd, *Proximal algorithms*, Found. Trends Optim., 1 (2014), pp. 127–239.
[37] A. L. Pavlov, G. W. Ovchinnikov, D. Y. Derbyshev, D. Tsetserukou, and I. V. Oseledets, *AA-ICP: Iterative closest point with Anderson acceleration*, in Proceedings of the IEEE International Conference on Robotics and Automation, 2018, pp. 3407–3412.
[38] F. A. Potra and H. Engler, *A characterization of the behavior of the Anderson acceleration on linear problems*, Linear Algebra Appl., 438 (2013), pp. 1002–1011.
[39] M. J. D. Powell, *A hybrid method for nonlinear equations*, in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon and Breach, New York, 1970, pp. 87–144.
[40] P. P. Pratapa and P. Suryanarayana, *Restarted Pulay mixing for efficient and robust acceleration of fixed-point iterations*, Chem. Phys. Lett., 635 (2015), pp. 69–74.
[41] P. Pulay, *Convergence acceleration of iterative sequences. The case of SCF iterations*, Chem. Phys. Lett., 73 (1980), pp. 393–398.
[42] P. Pulay, *Improved SCF convergence acceleration*, J. Comput. Chem., 3 (1982), pp. 556–560.
[43] A. N. Riseth, *Objective acceleration for unconstrained optimization*, Numer. Linear Algebra Appl., 26 (2019), e2216.
[44] T. Rohwedder and R. Schneider, *An analysis for the DIIS acceleration method used in quantum chemistry calculations*, J. Math. Chem., 49 (2011), pp. 1889–1914.
[45] E. K. Ryu and S. Boyd, *Primer on monotone operator methods*, Appl. Comput. Math., 15 (2016), pp. 3–43.
[46] D. Scieur, F. Bach, and A. d'Aspremont, *Nonlinear acceleration of stochastic algorithms*, in Advances in Neural Information Processing Systems 30, Curran Associates, 2017, pp. 3982–3991.
[47] D. Scieur, A. d'Aspremont, and F. Bach, *Regularized nonlinear acceleration*, in Advances in Neural Information Processing Systems, 2016, pp. 712–720.
[48] D. Scieur, E. Oyallon, A. d'Aspremont, and F. Bach, *Nonlinear Acceleration of CNNs*, preprint, arXiv:1806.003709, 2018.
[49] D. Scieur, E. Oyallon, A. d'Aspremont, and F. Bach, *Nonlinear Acceleration of Deep Neural Networks*, preprint, arXiv:1805.09639, 2018.
[50] M. Slawski and M. Hein, *Non-negative least squares for high-dimensional linear models: Consistency and sparse recovery without regularization*, Electron. J. Stat., 7 (2013), pp. 3004–3056.
[51] W. Tang and P. Daoutidis, *Fast and Stable Nonconvex Constrained Distributed Optimization: The ELLADA algorithm*, preprint, arXiv:2004.01977, 2020.
[52] A. Themelis and P. Patrinos, *SuperMann: A superlinearly convergent algorithm for finding fixed points of nonexpansive operators*, IEEE Trans. Automat. Control, 64 (2019), pp. 4875–4890.
[53] 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 J. Sci. Comput., 39 (2017), pp. S47–S65.
[54] A. Toth and C. T. Kelley, *Convergence analysis for Anderson acceleration*, SIAM J. Numer. Anal., 53 (2015), pp. 805–819.
[55] L. Vandenberghe, *Douglas-Rachford Method and ADMM*, ECE236C, http://www.seas.ucla.edu/~vandenbe/ee236c.html (2020).

[56] H. F. Walker and P. Ni, *Anderson acceleration for fixed-point iterations*, SIAM J. Numer. Anal., 49 (2011), pp. 1715–1735.

[57] W. Wang and M. A. Carreira-Perpiñán, *Projection onto the Probability Simplex: An Efficient Algorithm with a Simple Proof, and an Application*, preprint, arXiv:1309.1541, 2013.

[58] N. Woods, M. Payne, and P. Hasnip, *Computing the self-consistent field in Kohn–Sham density functional theory*, J. Phys. Condens. Matter, 31 (2019), 453001.

[59] X. Xiao, Y. Li, Z. Wen, and L. Zhang, *A regularized semi-smooth Newton method with projection steps for composite convex programs*, J. Sci. Comput., (2016), pp. 1–26.

[60] G. Xue and Y. Ye, *An efficient algorithm for minimizing a sum of Euclidean norms with applications*, SIAM J. Optim., 7 (1997), pp. 1017–1036.

[61] Y. Ye, M. J. Todd, and S. Mizuno, *An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm*, Math. Oper. Res., 19 (1994), pp. 53–67.

[62] J. Zhang, B. O'Donoghue, and S. Boyd, *Globally Convergent Type*-I *Anderson Acceleration for Non-smooth Fixed-point Iterations*, preprint, arXiv:1808.03971, 2018.

[63] J. Zhang, Y. Yao, Y. Peng, H. Yu, and B. Deng, *Fast K-means Clustering with Anderson Acceleration*, preprint, arXiv:1805.10638, 2018.

[64] W. Zhou and D. Li, *Limited memory BFGS method for nonlinear monotone equations*, J. Comput. Math., (2007), pp. 89–96.

[65] W. Zhou and D. Li, *A globally convergent BFGS method for nonlinear monotone equations without any merit functions*, Math. Comp., 77 (2008), pp. 2231–2240.

[66] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, J. R. Stat. Soc. Ser. B. Stat. Methodol., 67 (2005), pp. 301–320.