From Extrapolation to Quasi-Newton:
Stabilizing Type-I Anderson Mixing for Memory-Efficient, Line-Search Free and Black-Box Acceleration

Junzi Zhang

Stanford ICME, junziz@stanford.edu

Joint works with Brendan O’Donoghue, Anqi Fu, Stephen P. Boyd
Xin Guo, Anran Hu and Renyuan Xu

June 14, 2019
Overview

1. Motivation and Problem Statement
2. Acceleration: from extrapolation to quasi-Newton
3. Type-I Anderson acceleration and stabilization
4. Our algorithm
5. Numerical examples
Motivation and Problem Statement

1. Acceleration: from extrapolation to quasi-Newton
2. Type-I Anderson acceleration and stabilization
3. Our algorithm
4. Numerical examples
We consider solving a fixed-point problem \( x = f(x) \), where \( f : \mathbb{R}^n \to \mathbb{R}^n \) is potentially non-smooth.

\[
\|x\|_H = \sqrt{x^T H x}
\]
for some PSD matrix \( H \).
Fixed-point problems

- We consider solving a fixed-point problem $x = f(x)$, where $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is potentially non-smooth.

- **Assumption**: $f$ is non-expansive in $l_2$ (or $H$-norm\(^1\)), i.e.,
  \[
  \|f(x) - f(y)\|_2 \leq \|x - y\|_2 \text{ for any } x, y \in \mathbb{R}^n
  \]
  
or **contractive** in an arbitrary norm $\| \cdot \|$.

- Simplest solution: averaged iteration, a.k.a. Krasnosel’skiǐ-Mann (KM) iteration
  \[
  x^{k+1} = (1 - \alpha)x^k + \alpha f(x^k), \quad \alpha \in (0, 1).
  \]

- Convergence is robust, but sublinear in theory and slow in practice: 
can we **(safely)** do better?

\(^1\) $\|x\|_H = \sqrt{x^T H x}$ for some PSD matrix $H$
Many (potentially complicated) algorithms in optimization and beyond can be reformulated as "black-box" fixed-point problems. Examples:

- (Any) convex optimization with no strong convexity
  - minimize$_{x \in C} F(x)$, $C$ is convex, $F$ is convex and $L$-strongly smooth.
Many (potentially complicated) algorithms in optimization and beyond can be reformulated as \textit{“black-box” fixed-point} problems. Examples:

- (Any) convex optimization with no strong convexity
  - minimize$_{x \in C} F(x)$, $C$ is convex, $F$ is convex and $L$-strongly smooth.
  - Projected gradient descent: $x^{k+1} = \Pi_C (x^k - \frac{1}{L} \nabla F(x^k))$. 
Many (potentially complicated) algorithms in optimization and beyond can be reformulated as “black-box” fixed-point problems. Examples:

- (Any) convex optimization with no strong convexity
  - minimize\(_{x \in C} F(x)\), \(C\) is convex, \(F\) is convex and \(L\)-strongly smooth.
  - Projected gradient descent: \(x^{k+1} = \Pi_C (x^k - \frac{1}{L} \nabla F(x^k))\).
  - Optimality \(\Leftrightarrow x = f(x), f(x) := \Pi_C (x - \frac{1}{L} \nabla F(x))\).
Why non-smooth non-expansive fixed-point problems?

Many (potentially complicated) algorithms in optimization and beyond can be reformulated as "black-box" fixed-point problems. Examples:

- (Any) convex optimization with no strong convexity
  - minimize$_{x \in C} F(x)$, $C$ is convex, $F$ is convex and $L$-strongly smooth.
  - Projected gradient descent: $x^{k+1} = \Pi_C (x^k - \frac{1}{L} \nabla F(x^k))$.
  - Optimality $\iff x = f(x)$, $f(x) := \Pi_C (x - \frac{1}{L} \nabla F(x))$.
  - Projection is non-differentiable and non-expansive, but non-contractive without strong convexity.
Many (potentially complicated) algorithms in optimization and beyond can be reformulated as "black-box" fixed-point problems. Examples:

- **Discounted Markov decision processes (MDP)**
  - Value iteration: \( x^{k+1} = Tx^k \), where \( T \) is the Bellman operator:
    \[
    (Tx)_s = \max_{a=1,\ldots,A} R(s, a) + \gamma \sum_{s' = 1}^{S} P(s, a, s')x_{s'}.
    \]
  - Optimality \( \iff x = Tx \).
  - Contractive in \( l_\infty \), but still non-differentiable due to max.
Many (potentially complicated) algorithms in optimization and beyond can be reformulated as "black-box" fixed-point problems. Examples:

- Nash equilibrium in a multiplayer game $\iff$ monotone inclusion problem $\iff$ non-smooth non-expansive fixed-point problem.
1 Motivation and Problem Statement

2 Acceleration: from extrapolation to quasi-Newton

3 Type-I Anderson acceleration and stabilization

4 Our algorithm

5 Numerical examples
Algorithm 1 Extrapolation framework

Input: initial point $x_0$, fixed-point mapping $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$.

for $k = 0, 1, \ldots$ do

Choose $m_k$ (e.g., $m_k = \min\{m, k\}$ for some integer $m \geq 0$).
Select weights $\alpha^k_j$ based on the last $m_k$ iterations, with $\sum_{j=0}^{m_k} \alpha^k_j = 1$.

$x^{k+1} = \sum_{j=0}^{m_k} \alpha^k_j f(x^{k-m_k+j})$.

Such a framework subsumes many different algorithms, among which one of the most natural and popular method is Anderson acceleration (1965):

$$\text{minimize} \quad \| \sum_{j=0}^{m_k} \alpha^k_j g(x^{k-m_k+j}) \|_2^2 \quad \text{subject to} \quad \sum_{j=0}^{m_k} \alpha^k_j = 1,$$

where $g(x) := x - f(x)$ is the residual.
Also known as **Type-II Anderson acceleration** (AA-II), Anderson/Pulay mixing, Pulay’s direct inversion iterative subspace (DIIS), nonlinear GMRES, minimal polynomial extrapolation (MPE), reduced rank extrapolation (RRE), etc.
Also known as **Type-II Anderson acceleration** (AA-II), Anderson/Pulay mixing, Pulay’s direct inversion iterative subspace (DIIS), nonlinear GMRES, minimal polynomial extrapolation (MPE), reduced rank extrapolation (RRE), etc.

Widely used in computational quantum chemistry and material sciences, and recently introduced to optimization applications

- MLE, matrix completion, K-means, computer vision and deep learning.
Also known as **Type-II Anderson acceleration** (AA-II), Anderson/Pulay mixing, Pulay’s direct inversion iterative subspace (DIIS), nonlinear GMRES, minimal polynomial extrapolation (MPE), reduced rank extrapolation (RRE), etc.

Widely used in computational quantum chemistry and material sciences, and recently introduced to optimization applications

- MLE, matrix completion, K-means, computer vision and deep learning.

Equivalent to **multi-secant quasi-Newton** methods (see below) – development separated from the main-stream, connection established very recently in Fang and Saad 2009.
  - Extrapolation: good for intuition.
  - Quasi-Newton: good for derivations.
Recall the selection of $\alpha_j^k$ in AA-II (constrained least-squares):

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

Reformulation: minimize $\| g_k - Y_k \gamma \|_2$

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

$H_k:=$ argmin $H Y_k = S_k \| H - I \|_F$:

approximate inverse Jacobian of $g$.

multi-secant type-II (bad) Broyden's (quasi-Newton) method.
Recall the selection of $\alpha_j^k$ in AA-II (constrained least-squares):

$$\text{minimize} \quad \left\| \sum_{j=0}^{m_k} \alpha_j g(x^{k-m_k+j}) \right\|_2^2 \quad \text{subject to} \quad \sum_{j=0}^{m_k} \alpha_j = 1,$$

Reformulation: minimize $\| g_k - Y_k \gamma \|_2$

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

$x^{k+1} = \sum_{j=0}^{m_k} \alpha_j^k f(x^{k-m_k+j}) = x^k - H_k g_k$,

$H_k := I + (S_k - Y_k)(Y_k^T Y_k)^{-1} Y_k^T$.

$H_k = \arg\min_{HY_k = S_k} \| H - I \|_F$: approximate inverse Jacobian of $g$.

multi-secant type-II \textbf{(bad)} Broyden’s (quasi-Newton) method.
1. Motivation and Problem Statement

2. Acceleration: from extrapolation to quasi-Newton

3. Type-I Anderson acceleration and stabilization

4. Our algorithm

5. Numerical examples
Why not consider the type-I (good) counterpart?
Why not consider the \textbf{type-I (good)} counterpart?

Instead of inverse Jacobian (which itself \textbf{may not exist}), consider

\[ B_k := \arg\min_{BS_k = Y_k} \| B_k - I \|_F : \text{approximate Jacobian of } g. \]

\[ x^{k+1} = x^k - B_k^{-1} g_k, \text{ with } B_k^{-1} = I + (S_k - Y_k)(S_k^T Y_k)^{-1} S_k^T. \]
Why not consider the type-I (good) counterpart?

Instead of inverse Jacobian (which itself may not exist), consider

\[ B_k := \arg\min_{BS_k=Y_k} \| B_k - I \|_F: \text{ approximate Jacobian of } g. \]

\[ x^{k+1} = x^k - B_k^{-1} g_k, \text{ with } B_k^{-1} = I + (S_k - Y_k)(S_k^T Y_k)^{-1}S_k^T. \]

---

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

1: for \( k = 0, 1, \ldots \) do
2: Choose \( m_k \leq m \) (e.g., \( m_k = \min\{m, k\} \) for some integer \( m \geq 0 \)).
3: Compute \( \tilde{\gamma}^k = (S_k^T Y_k)^{-1}(S_k^T g_k) \).
4: \( \alpha_0^k = \tilde{\gamma}^k, \alpha_i^k = \tilde{\gamma}_i^k - \tilde{\gamma}_{i-1}^k \) (\( 1 \leq i \leq m_k - 1 \)) and \( \alpha_{m_k}^k = 1 - \tilde{\gamma}_{m_k-1}^k \).
5: \( x^{k+1} = \sum_{j=0}^{m_k} \alpha_j^k f(x^{k-m_k+j}). \)
Good news and bad news

Good news:

- Compared to AA-II: early experiments applying AA to SCS (a popular convex optimization solver) show obvious advantage of AA-I over AA-II on some benchmark problems.

- AA is memory efficient (AA-I with $m = 5-10$ beats LBFGS/restarted Broyden with $m = 200-500$).

- AA is line-search free: just accept or reject is the best practice.

- AA is suitable to be used in a completely black-box way.

- PGD: don’t separate the gradient step and projection.

- ADMM: don’t separate the primal and dual steps.

- SCS itself is a non-smooth and non-expansive fixed-point iteration.
Good news and bad news

Good news:

- **Compared to AA-II**: early experiments applying AA to SCS (a popular convex optimization solver) show obvious advantage of AA-I over AA-II on some benchmark problems.

- **Compared to LBFGS and restarted Broyden**:
  - AA is *memory efficient* (AA-I with $m = 5 - 10$ beats LBFGS/restarted Broyden with $m = 200 - 500$)
Good news and bad news

Good news:

- Compared to **AA-II**: early experiments applying AA to SCS (a popular convex optimization solver) show obvious advantage of AA-I over AA-II on some benchmark problems.

- Compared to **LBFGS** and **restarted Broyden**:
  - AA is *memory efficient* (AA-I with $m = 5 - 10$ beats LBFGS/restarted Broyden with $m = 200 - 500$)
  - AA is *line-search free*: just accept or reject is the best practice
Good news and bad news

Good news:

- Compared to **AA-II**: early experiments applying AA to SCS (a popular convex optimization solver) show obvious advantage of AA-I over AA-II on some benchmark problems.

- Compared to **LBFGS** and **restarted Broyden**:
  - AA is *memory efficient* (AA-I with $m = 5 - 10$ beats LBFGS/restarted Broyden with $m = 200 - 500$)
  - AA is *line-search free*: just accept or reject is the best practice
  - AA is suitable to be used in a completely *black-box* way
    - PGD: don’t separate the gradient step and projection
    - ADMM: don’t separate the primal and dual steps
Good news and bad news

Good news:

- Compared to AA-II: early experiments applying AA to SCS (a popular convex optimization solver) show obvious advantage of AA-I over AA-II on some benchmark problems.
- Compared to LBFGS and restarted Broyden:
  - AA is memory efficient (AA-I with $m = 5 - 10$ beats LBFGS/restarted Broyden with $m = 200 - 500$)
  - AA is line-search free: just accept or reject is the best practice
  - AA is suitable to be used in a completely black-box way
    - PGD: don’t separate the gradient step and projection
    - ADMM: don’t separate the primal and dual steps
- SCS itself is a non-smooth and non-expansive fixed-point iteration.
Good news and bad news

Good news:

- Compared to **AA-II**:

  ![Histogram of run time ratio between SuperSCS (AA-II) and SCS v2 (AA-I).](image)

  ![DM profile of run time.](image)

**Figure**: Left: histogram of run time ratio between SuperSCS (AA-II) and SCS v2 (AA-I). Right: DM profile of run time.
Good news and bad news

Good news:

- Compared to **restarted Broyden**:

![DM profile](image_url)

**Figure**: DM profile. left: sparse PCA; right: sparse logistic regression. 
**SuperSCS**: *fast and accurate large-scale conic optimization*. Sopasakis, et al., 2019.
Bad news:

- **Numerical challenge:** both AA-I and AA-II are subject to potential *numerical instability*, and AA-I is more severe.
  - AA-II: $Y_k^T Y_k$ (close to) singular (degenerate least-squares system).
  - AA-I: $B_k$ can be (close to) singular.
Bad news:

- **Numerical challenge**: both AA-I and AA-II are subject to potential *numerical instability*, and AA-I is more severe.
  - AA-II: $Y_k^T Y_k$ (close to) singular (degenerate least-squares system).
  - AA-I: $B_k$ can be (close to) singular.

- **Theoretical challenge**: local convergence theory exists with further smoothness assumptions, but *no global convergence*.
Bad news:

- **Numerical challenge**: both AA-I and AA-II are subject to potential *numerical instability*, and AA-I is more severe.
  - AA-II: $Y_k^T Y_k$ (close to) singular (degenerate least-squares system).
  - AA-I: $B_k$ can be (close to) singular.

- **Theoretical challenge**: local convergence theory exists with further smoothness assumptions, but *no global convergence*.

In general, most of the literature has been focused on AA-II:
- AA-I is generally *missing both in theory and practice*. 
Bad news:

- **Numerical challenge**: both AA-I and AA-II are subject to potential *numerical instability*, and AA-I is more severe.

**Figure**: Convergence of Anderson accelerated gradient descent on $\ell_2$ regularized logistic regression without stabilization. Left: AA-I vs AA-II. Right: AA-II v.s. stabilized AA-II (*Regularized Nonlinear Acceleration*, Scieur et al., 2016.)
Goal and contribution

- **Stabilize** AA-I with convergence beyond **differentiability, locality and non-singularity**
  - **Surprise:** stabilization also improves convergence consistently over both the original AA-I and AA-II.
Goal and contribution

- **Stabilize** AA-I with convergence beyond **differentiability, locality and non-singularity**
  - **Surprise:** stabilization also improves convergence consistently over both the original AA-I and AA-II.
- Develop a “plug-and-play” acceleration scheme based on the stabilized AA-I
  - View an arbitrary unaccelerated algorithm as a **black-box** fixed-point iteration problem.
  - For example, concatenate successive iterates in momentum algorithms.
Stabilization of AA-I: rank-one update

AA-I $\iff$ Type-I Broyden’s rank-one update with **orthogonalization**:

**Proposition**

Suppose that $S_k$ is **full rank**, then $B_k$ can be computed inductively from $B^0_k = I$ as follows:

$$B^{i+1}_k = B^i_k + \frac{(y_{k-m_k+i} - B^i_k s_{k-m_k+i}) \hat{s}^T_{k-m_k+i}}{\hat{s}^T_{k-m_k+i} s_{k-m_k+i}}, \quad i = 0, \ldots, m_k - 1$$

with $B_k = B^{m_k}_k$. Here $\{\hat{s}_i\}_{i=k-m_k}^{k-1}$ is the Gram-Schmidt orthogonalization of $\{s_i\}_{i=k-m_k}^{k-1}$, i.e., $\hat{s}_i = s_i - \sum_{j=k-m_k}^{i-1} \frac{\hat{s}_j s_i}{\hat{s}_j^T \hat{s}_j} \hat{s}_j$, $\quad i = k - m_k, \ldots, k - 1$. 
Goal of regularization: avoid close to singularity ("lower bound" on $B_k$).
Stabilization of AA-I: 1. Powell-type regularization

Goal of regularization: avoid close to singularity ("lower bound" on $B_k$).

- AA-II: add ridge penalty (regularized nonlinear acceleration, 2016)

$$\text{minimize} \sum_{j=0}^{m_k} \alpha_j = 1 \| \sum_{j=0}^{m_k} \alpha_j g(x^{k-m_k+j}) \|^2_2 + \lambda \| \alpha \|^2_2$$

Help in extreme cases, but **impede the convergence** in general.
Stabilization of AA-I: 1. Powell-type regularization

Goal of regularization: avoid close to singularity (“lower bound” on $B_k$).

- AA-II: add ridge penalty (regularized nonlinear acceleration, 2016)
  \[
  \text{minimize} \sum_{j=0}^{m_k} \alpha_j = 1 \left\| \sum_{j=0}^{m_k} \alpha_j g(x^{k-m_k+j}) \right\|_2^2 + \lambda \|\alpha\|_2^2
  \]
  Help in extreme cases, but **impede the convergence** in general.

- AA-I: Powell-type trick (**turns out helpful also in practice**)!
  - Replace $y_{k-m_k+i}$ with $\tilde{y}_{k-m_k+i} = \theta^i_k y_{k-m_k+i} + (1 - \theta^i_k) B^i_k s_{k-m_k+i}$,
    where $\theta^i_k = \phi_{\bar{\theta}}(\eta^i_k)$, with $\eta^i_k = \frac{\hat{s}_{k-m_k+i}^T (B^i_k)^{-1} y_{k-m_k+i}}{\|\hat{s}_{k-m_k+i}\|_2^2}$.

  \[
  \phi_{\bar{\theta}}(\eta) = \begin{cases} 
  1 & \text{if } |\eta| \geq \bar{\theta} \\
  1 - \text{sign}(\eta) \bar{\theta} & \text{if } |\eta| < \bar{\theta}
  \end{cases}
  \]
Stabilization of AA-I: 1. Powell-type regularization

Goal of regularization: avoid close to singularity ("lower bound" on $B_k$).

- AA-II: add ridge penalty (regularized nonlinear acceleration, 2016)

$$\text{minimize } \sum_{j=0}^{m_k} \alpha_j = 1 \quad || \sum_{j=0}^{m_k} \alpha_j g(x^{k-m_k+j}) ||_2^2 + \lambda ||\alpha||_2^2$$

Help in extreme cases, but impede the convergence in general.

- AA-I: Powell-type trick (turns out helpful also in practice!)

  - Replace $y_{k-m_k+i}$ with $\tilde{y}_{k-m_k+i} = \theta_i^k y_{k-m_k+i} + (1 - \theta_i^k) B_i^k s_{k-m_k+i}$,
  
  where $\theta_i^k = \phi_{\bar{\theta}}(\eta_i^k)$, with $\eta_i^k = \frac{\hat{s}_{k-m_k+i}^T (B_i^k)^{-1} y_{k-m_k+i}}{||\hat{s}_{k-m_k+i}||_2^2}$,

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

- $|\text{det}(B_k)| \geq \bar{\theta}^{m_k} > 0$, and in particular, $B_k$ is invertible!
Goal of re-start: avoid blow-up ("upper bound" on $B_k$).
Goal of re-start: avoid blow-up ("upper bound" on $B_k$).

- $\hat{s}_{k-m_k+i}^T s_{k-m_k+i} = \|\hat{s}_{k-m_k+i}\|_2^2$ appears in the denominators: but $\hat{s}_{k-m_k+i}$ becomes 0 when $m_k > n$ due to orthogonalization.
Stabilization of AA-I: 2. Re-start checking

Goal of re-start: avoid blow-up ("upper bound" on $B_k$).

- $\hat{s}_{k-m_k+i}^T s_{k-m_k+i} = \|\hat{s}_{k-m_k+i}\|_2^2$ appears in the denominators: but $\hat{s}_{k-m_k+i}$ becomes 0 when $m_k > n$ due to orthogonalization.

- Solution: update $m_k = m_{k-1} + 1$. If $m_k = m + 1$ or $\|\hat{s}_{k-1}\|_2 < \tau\|s_{k-1}\|_2$, then reset $m_k = 1$. 

Goal of re-start: avoid blow-up ("upper bound" on $B_k$).

- $\hat{s}_{k-m_k+i}^T s_{k-m_k+i} = \|\hat{s}_{k-m_k+i}\|^2_2$ appears in the denominators: but $\hat{s}_{k-m_k+i}$ becomes 0 when $m_k > n$ due to orthogonalization.

- Solution: update $m_k = m_{k-1} + 1$. If $m_k = m + 1$ or $\|\hat{s}_{k-1}\|_2 < \tau \|s_{k-1}\|_2$, then reset $m_k = 1$.

- Then $\|B_k\|_2 \leq 3(1 + \bar{\theta} + \tau)^m/\tau^m - 2!$

- (Re)define $H_k := B_k^{-1}$: $\|H_k\|_2 \leq \left(3 \left(\frac{1 + \bar{\theta} + \tau}{\tau} \right)^m - 2 \right)^{n-1}/\bar{\theta}^m$. 
Goal of safe-guard: avoid “wild” and “bad” extrapolation.
Goal of safe-guard: avoid “wild” and “bad” extrapolation.

- Main idea: interleave AA-I steps with the vanilla KM iteration steps to safe-guard the decrease in residual norms $g$. 

$\text{Check if the current residual norm is sufficiently small, and replace it with } f(\alpha)(x) = (1 - \alpha)x + \alpha f(x) \text{ whenever not.}$

Can be seen as a cheap alternative to the expensive line-search.
Goal of safe-guard: avoid “wild” and “bad” extrapolation.

- Main idea: interleave AA-I steps with the vanilla KM iteration steps to safe-guard the decrease in residual norms $g$.
- Check if the current residual norm is sufficiently small, and replace it with $f_\alpha(x) = (1 - \alpha)x + \alpha f(x)$ whenever not.
Goal of safe-guard: *avoid “wild” and “bad” extrapolation.*

- **Main idea:** Interleave AA-I steps with the vanilla KM iteration steps to safe-guard the decrease in residual norms $g$.
- **Check if the current residual norm is sufficiently small,** and replace it with $f_\alpha(x) = (1 - \alpha)x + \alpha f(x)$ whenever not.
- **Can be seen as a cheap alternative to the expensive line-search.**
Motivation and Problem Statement

Acceleration: from extrapolation to quasi-Newton

Type-I Anderson acceleration and stabilization

Our algorithm

Numerical examples
Stabilized AA-I

Combine Powell-type regularization, re-start checking and safe-guard checking (with some simplifications using Woodbury formula, etc.)

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

1: **Input:** initial point \( x_0 \), fixed-point mapping \( f : \mathbb{R}^n \rightarrow \mathbb{R}^n \), regularization constants \( \bar{\theta}, \tau, \alpha \in (0, 1) \), safe-guarding constants \( D, \epsilon > 0 \), max-memory \( m > 0 \).

2: Initialize \( H_0 = I, m_0 = n_{AA} = 0, \bar{U} = \|g_0\|_2 \), and compute \( x^1 = \tilde{x}^1 = f_\alpha(x^0) \).

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

4: \( m_k = m_{k-1} + 1 \).

5: Compute \( s_{k-1} = \tilde{x}^k - x^{k-1}, y_{k-1} = g(\tilde{x}^k) - g(x^{k-1}) \).

6: Compute \( \hat{s}_{k-1} = s_{k-1} - \sum_{j=k-m_k}^{k-2} \hat{s}_j^T s_{k-1} \hat{s}_j \).

7: \[ \text{If } m_k = m + 1 \text{ or } \|\hat{s}_{k-1}\|_2 < \tau \|s_{k-1}\|_2 \] \{Re-start checking\}

8: reset \( m_k = 1, \hat{s}_{k-1} = s_{k-1} \), and \( H_{k-1} = I \).

9: Update \( H_k \) with \{Powell-type regularization\}, compute \( \tilde{x}^{k+1} = x^k - H_k g_k \).

10: \[ \text{If } \|g_k\| \leq D \bar{U}(n_{AA} + 1)^{- (1 + \epsilon)} \] \{Safe-guard checking\}

11: \( x^{k+1} = \tilde{x}^{k+1}, n_{AA} = n_{AA} + 1 \).

12: else \( x^{k+1} = f_\alpha(x^k) \).
Global convergence

Theorem

Suppose that $f$ is non-expansive in $l_2$-norm or contractive in an arbitrary norm, and assume that $\{x^k\}_{k=0}^\infty$ is generated by Algorithm 3. Then we have $\lim_{k \to \infty} x^k = x^*$, where $x^* = f(x^*)$.

Key: bounds on $H_k$ and $B_k$ ensure that the deviation is not too much from the safe-guarding paths.
Implementation details

- **Hyper-parameters choice:** $\bar{\theta} = 0.01$, $\tau = 0.001$, $D = 10^6$, $\epsilon = 10^{-6}$, memory $m = 5$, averaging weight $\alpha = 0.1$.

- **Matrix-free updates:** instead of computing and storing $H_k$, we store $H_{k-j}\tilde{y}_{k-j}$ and $\frac{H^T_{k-j}\hat{s}_{k-j}}{\hat{s}^T_{k-j}(H_{k-j}\tilde{y}_{k-j})}$ for $j = 1, \ldots, m_k$, compute

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

  and then update $\tilde{x}^{k+1} = x^k - d_k$.

- **Problem scaling** is helpful when matrices are involved.
Motivation and Problem Statement

Acceleration: from extrapolation to quasi-Newton

Type-I Anderson acceleration and stabilization

Our algorithm

Numerical examples
General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by concatenation and neglecting (intermediate variables).
Apart from PGD ($\min_{x \in C} F(x)$) and value iteration (MDP):
More examples: Problem + ALG $\Leftrightarrow$ black-box FP

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by concatenation and neglecting (intermediate variables).

Apart from PGD (min$_{x \in C} F(x)$) and value iteration (MDP):

- Problem 1: find $x \in C \cap D$. 
General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by concatenation and neglecting (intermediate variables).
Apart from PGD ($\min_{x \in C} F(x)$) and value iteration (MDP):

- Problem 1: find $x \in C \cap D$.
- Algorithm – alternating projection: $x^{k+1} = f(x^k) = \Pi_C(\Pi_D(x^k))$. 
More examples: Problem + ALG $\iff$ black-box FP

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by concatenation and neglecting (intermediate variables).

Apart from PGD ($\min_{x \in C} F(x)$) and value iteration (MDP):

- **Problem 1**: find $x \in C \cap D$.
- **Algorithm – alternating projection**: $x^{k+1} = f(x^k) = \Pi_C(\Pi_D(x^k))$.
- **FP**: $x = \Pi_C(\Pi_D(x^k))$. 
General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by concatenation and neglecting (intermediate variables).

Apart from PGD ($\min_{x \in C} F(x)$) and value iteration (MDP):

- **Problem 2**: minimize $x F(x) + \mu \|x\|_1$.
- **Algorithm – ISTA**: $x^{k+1} = S_{\alpha \mu}(x^k - \alpha \nabla F(x^k))$, with $S_{\kappa}(x)_i = \text{sign}(x_i)(|x_i| - \kappa)_+$ for $i = 1, \ldots, n$.
- **FP**: $x = S_{\alpha \mu}(x - \alpha \nabla F(x))$. 
More examples: Problem + ALG ⇔ black-box FP

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by concatenation and neglecting (intermediate variables).

Apart from PGD ($\min_{x \in C} F(x)$) and value iteration (MDP):

- Problem 3: minimize $\sum_{i=1}^{m} F_i(x)$.
- Algorithm – consensus DRS:

  $x_i^{k+1} = \arg\min_{x_i} F_i(x_i) + (1/2\alpha)\|x_i - z_i^k\|_2^2$,
  
  $z_i^{k+1} = z_i^k + 2\bar{x}^{k+1} - x_i^{k+1} - \bar{z}^k$, $i = 1, \ldots, m$.

- FP: $f$ defined as the mapping from $z^k$ to $z^{k+1}$.
- Wrong approach: apply AA to both $x$ and $z$.  

More examples: Problem + ALG ⇔ black-box FP

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by concatenation and neglecting (intermediate variables).

Apart from PGD ($\min_{x \in C} F(x)$) and value iteration (MDP):

- Problem 4: minimize $\min_x c^T x$, subject to $Ax + s = b$, $s \in \mathcal{K}$.
- Algorithm – SCS ($\mathcal{C} = \mathbb{R}^n \times \mathcal{K}^* \times \mathbb{R}_+$):
  \[
  \begin{align*}
  \tilde{u}^{k+1} &= (I + Q)^{-1}(u^k + v^k) \\
  u^{k+1} &= \Pi_\mathcal{C}(\tilde{u}^{k+1} - v^k) \\
  v^{k+1} &= v^k - \tilde{u}^{k+1} + u^{k+1}.
  \end{align*}
  \]

- FP (don’t apply AA to $u$ and $v$ separately):
  \[
  f(u, v) = \begin{bmatrix}
  \Pi_\mathcal{C}((I + Q)^{-1}(u + v) - v) \\
  v - (I + Q)^{-1}(u + v) + u
  \end{bmatrix}.
  \]
More examples: Problem $+$ ALG $\iff$ black-box FP

General idea: rewrite an algorithm into $x^{k+1} = f(x^k)$ by concatenation and neglecting (intermediate variables).

Apart from PGD ($\min_{x \in C} F(x)$) and value iteration (MDP):

- **Problem 5:** minimize $x^T Ax + b^T x + c$.
- **Algorithm:** momentum GD: $x^{k+1} = x^k - \alpha(Ax^k + b) + \beta(x^k - x^{k-1})$.
- **FP** (concatenate two successive iterates):

$$f(x', x) = \left[ x' - \alpha(Ax' + b) + \beta(x' - x) \right]_{x'}.$$  

- Remember to concatenate, don’t simply neglect $x^{k-1}$ as in RNA.
Numerical examples

Gradient Descent: stabilization from divergence to convergence

Figure: Gradient descent: regularized logistic regression. Left: residual norm versus iteration. Right: residual norm versus time (seconds).
**Numerical examples**

**SCS (ADMM):** SOCP – nonsmoothness coming from projections

**Figure:** SCS: second-order cone program. Left: residual norm versus iteration. Right: residual norm versus time (seconds).
**Numerical examples**

**ISTA**: elastic net regression – nonsmoothness coming from shrinkage

**Figure**: Iterative Shrinkage-Thresholding Algorithm: elastic-net linear regression. Left: residual norm versus iteration. Right: residual norm versus time (seconds).
Numerical examples

**MDP (value iteration)** (discount factor $\gamma = 0.99$):

Figure: Value iteration: MDP. Left: residual norm versus iteration. Right: residual norm versus time (seconds).
Numerical examples

Effect of **different memories** $m$:

![Graph showing residual norm versus iteration and time](image)

**Figure**: Value iteration: memory effect. Left: residual norm versus iteration. Right: residual norm versus time (seconds).
Starting point: Early empirical success in applying AA-I to SCS, but unstable performance
Summary

- **Starting point:** Early empirical success in applying AA-I to SCS, but unstable performance
- **Destination:**
  - the first *globally convergent* Anderson acceleration variant under very relaxed conditions.
Starting point: Early empirical success in applying AA-I to SCS, but unstable performance

Destination:
- the first globally convergent Anderson acceleration variant under very relaxed conditions.
- online pre-conditioning/stabilization tricks useful both in theory and practice (Powell, re-start and safe-guard).
Starting point: Early empirical success in applying AA-I to SCS, but unstable performance

Destination:
- the first globally convergent Anderson acceleration variant under very relaxed conditions.
- online pre-conditioning/stabilization tricks useful both in theory and practice (Powell, re-start and safe-guard).
- flexible applications to many different problems/algorithms with general black-box fixed-point reformulation, with stable performance.
Summary

- **Starting point:** Early empirical success in applying AA-I to SCS, but unstable performance

- **Destination:**
  - the first **globally convergent** Anderson acceleration variant under very relaxed conditions.
  - online pre-conditioning/stabilization tricks useful both in theory and practice (Powell, re-start and safe-guard).
  - flexible applications to many different problems/algorithms with general **black-box** fixed-point reformulation, with stable performance.
  - Now being implemented and tested in SCS 2.0.
Our stabilization technique can actually be extended to generic non-convex optimization settings.
Our stabilization technique can actually be extended to generic non-convex optimization settings.

- **Safe-guard** becomes central here (unlike non-expansive cases), and need to be exclusive designed for each algorithm.
Beyond non-expansiveness (convexity)

- Our stabilization technique can actually be extended to generic non-convex optimization settings.
  - **Safe-guard** becomes central here (unlike non-expansive cases), and need to be exclusive designed for each algorithm.
  - Example: We proposed *Anderson accelerated iPALM* [GHXZ2018] with an exclusive safe-guard for iPALM for computing the MLEs multivariate Hawkes processes.
Safe-guards in non-convex optimization

Figure: MLE of MHPs: exponential hawkes. **No safe-guards.** Left: log-regret v.s. time (seconds). Right: objective v.s. time (seconds).
Figure: MLE of MHPs: exponential hawkes. With safe-guards. Left: log-regret v.s. time (seconds). Right: objective v.s. time (seconds).
Future work

- Can we extract some general design rules of safe-guards formally?
Future work

- Can we extract some general design rules of safe-guards formally?
  - Find a **balance** between practical efficiency and theoretical guarantee.

**Failure example:** apply AA-II to Nesterov, but require monotonic decrease in the objective values, which breaks the non-monotonic acceleration of Nesterov.

**More examples for applying AA-I:**
Nesterov's accelerated gradient descent, Frank-Wolfe, stochastic gradient descent and its variants (e.g., ADAM), ... (a ongoing tutorial paper).

Adaptive choices/line-search of the hyper-parameters in our stabilized AA-I.
Future work

- Can we extract some general design rules of safe-guards formally?
  - Find a **balance** between practical efficiency and theoretical guarantee.
  - Failure example: apply AA-II to Nesterov, but require **monotonic** decrease in the objective values, which breaks the **non-monotonic acceleration** of Nesterov.
Can we extract some general design rules of safe-guards formally?

- Find a **balance** between practical efficiency and theoretical guarantee.
- Failure example: apply AA-II to Nesterov, but require **monotonic** decrease in the objective values, which breaks the **non-monotonic acceleration** of Nesterov.

More examples for applying AA-I:

- Nesterov’s accelerated gradient descent, Frank-Wolfe, stochastic gradient descent and its variants (e.g., ADAM), ... (a ongoing tutorial paper).
Can we extract some general design rules of safe-guards formally?
- Find a **balance** between practical efficiency and theoretical guarantee.
- Failure example: apply AA-II to Nesterov, but require **monotonic** decrease in the objective values, which breaks the **non-monotonic acceleration** of Nesterov.

More examples for applying AA-I:
- Nesterov’s accelerated gradient descent, Frank-Wolfe, stochastic gradient descent and its variants (e.g., ADAM), ... (a ongoing tutorial paper).
- Adaptive choices/line-search of the hyper-parameters in our stabilized AA-I.

Thanks for listening!

Any questions?