Connecting Quasi-Newton with Extrapolation:

Black-box, Memory Efficient and Line-search Free Acceleration via Stabilized Anderson Mixing

Junzi Zhang

Stanford ICME, *junziz@stanford.edu*

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

CME 510 Linear Algebra and Optimization Seminar

November 7, 2019

Overview

1. Motivation and Problem Statement
2. Acceleration: connecting quasi-Newton with extrapolation
   - Good news and bad news
3. A generic stabilization scheme
   - Stabilization of AA-I
   - Stabilization of AA-II
   - Global convergence: solvable settings
   - A browse through effect of stabilization
4. Applications
   - Conic optimization + SCS 2.x
   - Prox-affine optimization + A2DR
5. Beyond convexity

1. Motivation and Problem Statement

2. Acceleration: connecting quasi-Newton with extrapolation
   - Good news and bad news

3. A generic stabilization scheme
   - Stabilization of AA-I
   - Stabilization of AA-II
   - Global convergence: solvable settings
   - A browse through effect of stabilization

4. Applications
   - Conic optimization + SCS 2.x
   - Prox-affine optimization + A2DR

5. Beyond convexity

Fixed-point problems

- We consider solving a fixed-point (FP) problem $v = F(v)$, where $F : \mathbb{R}^n \to \mathbb{R}^n$ is potentially non-smooth.

[1] $\|v\|_H = \sqrt{v^T H v}$ for some PD matrix H

Fixed-point problems

- We consider solving a fixed-point (FP) problem $v = F(v)$, where $F : \mathbb{R}^n \to \mathbb{R}^n$ is potentially non-smooth.
- **Assumption**: F is **non-expansive** in l_2 (or H-norm[1]), *i.e.*,

$$\|F(v) - F(w)\|_2 \leq \|v - w\|_2 \text{ for any } v,\ w \in \mathbb{R}^n$$

  or **contractive** in an arbitrary norm $\|\cdot\|$.
- Simplest solution: averaged iteration, a.k.a. Krasnosel'skiĭ-Mann (KM) iteration

$$v^{k+1} = F_\alpha(v^k) = (1-\alpha)v^k + \alpha F(v^k),\ \alpha \in (0,1).$$

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

[1] $\|v\|_H = \sqrt{v^T H v}$ for some PD matrix H

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
  - $\text{minimize}_{x \in C}\ f(x)$, C is convex, f is convex and L-strongly smooth.

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
  - $\text{minimize}_{x \in C}\ f(x)$, C is convex, f is convex and L-strongly smooth.
  - Projected gradient descent: $x^{k+1} = \Pi_C\left(x^k - \frac{1}{L}\nabla f(x^k)\right)$.

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
  - $\text{minimize}_{x\in C}\ f(x)$, C is convex, f is convex and L-strongly smooth.
  - Projected gradient descent: $x^{k+1} = \Pi_C\left(x^k - \frac{1}{L}\nabla f(x^k)\right)$.
  - Optimality $\Leftrightarrow$ $x = F(x)$, $F(x) := \Pi_C\left(x - \frac{1}{L}\nabla f(x)\right)$.

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
  - $\text{minimize}_{x \in C}\ f(x)$, C is convex, f is convex and L-strongly smooth.
  - Projected gradient descent: $x^{k+1} = \Pi_C\left(x^k - \frac{1}{L}\nabla f(x^k)\right)$.
  - Optimality $\Leftrightarrow$ $x = F(x),\ F(x) := \Pi_C\left(x - \frac{1}{L}\nabla f(x)\right)$.
  - Projection is non-differentiable and non-expansive, but non-contractive without strong convexity.

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:

- Discounted Markov decision processes (MDP)
  - Value iteration: $v^{k+1} = Tv^k$, where T is the Bellman operator:

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

  - Optimality $\Leftrightarrow$ $v = Tv$.
  - Contractive in l_∞, but still non-differentiable due to max.

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:

- Nash equilibrium in a multiplayer game $\Leftrightarrow$ monotone inclusion problem $\Leftrightarrow$ non-smooth non-expansive fixed-point problem.

1 Motivation and Problem Statement

2 Acceleration: connecting quasi-Newton with extrapolation
- Good news and bad news

3 A generic stabilization scheme
- Stabilization of AA-I
- Stabilization of AA-II
- Global convergence: solvable settings
- A browse through effect of stabilization

4 Applications
- Conic optimization + SCS 2.x
- Prox-affine optimization + A2DR

5 Beyond convexity

Algorithm 1 Extrapolation framework

Input: initial point x_0, fixed-point mapping $F : \mathbb{R}^n \to \mathbb{R}^n$.
for $k = 0, 1, \ldots$ **do**
  Choose m_k (*e.g.*, $m_k = \min\{m, k\}$ for some integer $m \geq 0$).
  Select weights α_j^k based on the last m_k iterations, with $\sum_{j=0}^{m_k} \alpha_j^k = 1$.
  $v^{k+1} = \sum_{j=0}^{m_k} \alpha_j^k F(v^{k-m_k+j})$.

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

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

where $G(v) := v - F(v)$ is the residual.

Literature comments

- 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.

Literature comments

- 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.

Literature comments

- 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.

From extrapolation to quasi-Newton

- Recall the selection of α_j^k in AA-II (constrained least-squares):

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

- Reformulation: minimize $\|g^k - Y_k \gamma\|_2$
  - variable $\gamma = (\gamma_0, \ldots, \gamma_{m_k-1})$.
  - $g^i = G(v^i)$, $Y_k = [y^{k-m_k} \ \ldots \ y^{k-1}]$ with $y^i = g^{i+1} - g^i$ for each i.
  - $\alpha_0 = \gamma_0$, $\alpha_i = \gamma_i - \gamma_{i-1}$ for $1 \le i \le m_k - 1$ and $\alpha_{m_k} = 1 - \gamma_{m_k-1}$.

From extrapolation to quasi-Newton

- Recall the selection of α_j^k in AA-II (constrained least-squares):

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

- Reformulation: minimize $\|g^k - Y_k\gamma\|_2$
  - variable $\gamma = (\gamma_0, \ldots, \gamma_{m_k-1})$.
  - $g^i = G(v^i)$, $Y_k = [y^{k-m_k} \ \ldots \ y^{k-1}]$ with $y^i = g^{i+1} - g^i$ for each i.
  - $\alpha_0 = \gamma_0$, $\alpha_i = \gamma_i - \gamma_{i-1}$ for $1 \le i \le m_k - 1$ and $\alpha_{m_k} = 1 - \gamma_{m_k-1}$.
- $v^{k+1} = \sum_{j=0}^{m_k} \alpha_j^k F(v^{k-m_k+j}) = v^k - H_k g^k$,
  - $H_k := I + (S_k - Y_k)(Y_k^T Y_k)^{-1} Y_k^T$
    - $S_k = [s^{k-m_k} \ldots s^{k-1}]$ with $s^i = v^{i+1} - v^i$ for each i.
  - $H_k = \text{argmin}_{HY_k = S_k} \|H - I\|_F$: **approximate inverse Jacobian** of G.
  - multi-secant type-II Broyden's (quasi-Newton) method.

Type-I Anderson acceleration

- Why not consider the **type-I** counterpart?

Type-I Anderson acceleration

- Why not consider the **type-I** counterpart?
- Instead of inverse Jacobian (which itself **may not exist**), consider $B_k := \operatorname{argmin}_{BS_k = Y_k} \|B_k - I\|_F$: **approximate Jacobian** of G.
- $v^{k+1} = v^k - B_k^{-1} g^k$, with $B_k^{-1} = I + (S_k - Y_k)(S_k^T Y_k)^{-1} S_k^T$.

Type-I Anderson acceleration

- Why not consider the **type-I (good)** counterpart?
- Instead of inverse Jacobian (which itself **may not exist**), consider $B_k := \text{argmin}_{BS_k=Y_k} \|B_k - I\|_F$: **approximate Jacobian** of G.
- $v^{k+1} = v^k - B_k^{-1} g_k$, 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: $\alpha_0^k = \tilde{\gamma}_0^k$, $\alpha_i^k = \tilde{\gamma}_i^k - \tilde{\gamma}_{i-1}^k$ $(1 \leq i \leq m_k - 1)$ and $\alpha_{m_k}^k = 1 - \tilde{\gamma}_{m_k-1}^k$.
5: $v^{k+1} = \sum_{j=0}^{m_k} \alpha_j^k F(v^{k-m_k+j})$.

1. Motivation and Problem Statement
2. Acceleration: connecting quasi-Newton with extrapolation
   - Good news and bad news
3. A generic stabilization scheme
   - Stabilization of AA-I
   - Stabilization of AA-II
   - Global convergence: solvable settings
   - A browse through effect of stabilization
4. Applications
   - Conic optimization + SCS 2.x
   - Prox-affine optimization + A2DR
5. Beyond convexity

Good news and bad news

Takeaway information:

Good news and bad news

Takeaway information:

- Extrapolation & quasi-Newton method accelerating FP iterations.

Good news and bad news

Takeaway information:

- Extrapolation & quasi-Newton method accelerating FP iterations.
  - **Extrapolation** (Anderson, 1965)

Good news and bad news

Takeaway information:

- Extrapolation & quasi-Newton method accelerating FP iterations.
  - **Extrapolation** (Anderson, 1965)
  - **Multi-secant quasi-Newton** method (Fang & Saad, 2009).

Good news and bad news

Takeaway information:

- Extrapolation & quasi-Newton method accelerating FP iterations.
  - **Extrapolation** (Anderson, 1965)
  - **Multi-secant quasi-Newton** method (Fang & Saad, 2009).
  - **Type-I AA:** approximate the Jacobian of the FP mapping

Good news and bad news

Takeaway information:

- Extrapolation & quasi-Newton method accelerating FP iterations.
  - **Extrapolation** (Anderson, 1965)
  - **Multi-secant quasi-Newton** method (Fang & Saad, 2009).
  - **Type-I AA:** approximate the Jacobian of the FP mapping
  - **Type-II AA:** approximate the **inverse** Jacobian of the FP mapping

Good news and bad news

Takeaway information:

- Extrapolation & quasi-Newton method accelerating FP iterations.
  - **Extrapolation** (Anderson, 1965)
  - **Multi-secant quasi-Newton** method (Fang & Saad, 2009).
  - **Type-I AA:** approximate the Jacobian of the FP mapping
  - **Type-II AA:** approximate the **inverse** Jacobian of the FP mapping

Good news:

Takeaway information:

- Extrapolation & quasi-Newton method accelerating FP iterations.
  - **Extrapolation** (Anderson, 1965)
  - **Multi-secant quasi-Newton** method (Fang & Saad, 2009).
  - **Type-I AA:** approximate the Jacobian of the FP mapping
  - **Type-II AA:** approximate the **inverse** Jacobian of the FP mapping

Good news:

- 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

Takeaway information:

- Extrapolation & quasi-Newton method accelerating FP iterations.
  - **Extrapolation** (Anderson, 1965)
  - **Multi-secant quasi-Newton** method (Fang & Saad, 2009).
  - **Type-I AA:** approximate the Jacobian of the FP mapping
  - **Type-II AA:** approximate the **inverse** Jacobian of the FP mapping

Good news:

- 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

Takeaway information:

- Extrapolation & quasi-Newton method accelerating FP iterations.
  - **Extrapolation** (Anderson, 1965)
  - **Multi-secant quasi-Newton** method (Fang & Saad, 2009).
  - **Type-I AA:** approximate the Jacobian of the FP mapping
  - **Type-II AA:** approximate the **inverse** Jacobian of the FP mapping

Good news:

- 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:

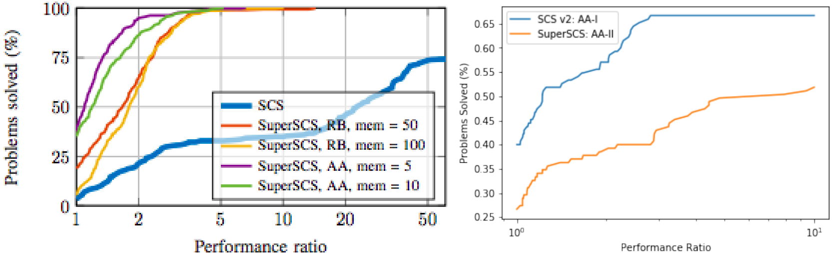


Figure: Sparse PCA: DM profiles of run time. Left: AA-II v.s. restarted Broyden, both in SuperSCS. Right: AA-I (SCS 2.x) v.s. AA-II (SuperSCS).

Good news and bad news

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.

Good news and bad news

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*.

Good news and bad news

Bad news: **Numerical challenge**

- Both AA-I and AA-II are subject to potential *numerical instability*, and AA-I is more severe.

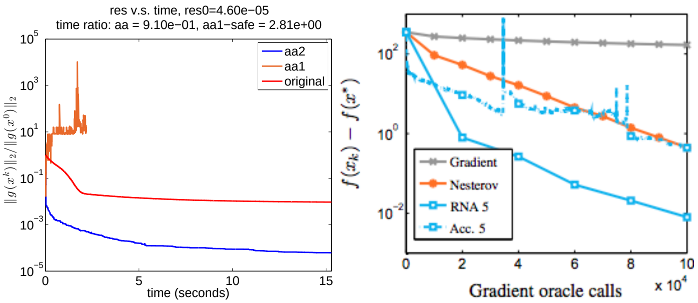


Figure: Divergence of AA + gradient descent on ℓ_2 regularized logistic regression **without** stabilization. Left: Failure of AA-I. Right: Failure of AA-II (*Regularized Nonlinear Acceleration*, Scieur et al., 2016).

Good news and bad news

Bad news: **Theoretical challenge**

- Type-II AA can even provably diverge when applied to the gradient descent on a one-dimensional smooth unconstrained optimization problem (Mai & Johansson, 2019).

Bad news: **Theoretical challenge**

- Type-II AA can even provably diverge when applied to the gradient descent on a one-dimensional smooth unconstrained optimization problem (Mai & Johansson, 2019).
  - (Scieur et al., 2016) showed that adding **constant quadratic regularization** to the objective leads to local convergence improvement.

Good news and bad news

Bad news: **Theoretical challenge**

- Type-II AA can even provably diverge when applied to the gradient descent on a one-dimensional smooth unconstrained optimization problem (Mai & Johansson, 2019).
  - (Scieur et al., 2016) showed that adding **constant quadratic regularization** to the objective leads to local convergence improvement.
  - **Insufficient** for global convergence both in theory and practice.

Good news and bad news

Bad news: **Theoretical challenge**

- Type-II AA can even provably diverge when applied to the gradient descent on a one-dimensional smooth unconstrained optimization problem (Mai & Johansson, 2019).
  - (Scieur et al., 2016) showed that adding **constant quadratic regularization** to the objective leads to local convergence improvement.
  - **Insufficient** for global convergence both in theory and practice.
- In general, most of the literature has been focused on AA-II:
  - AA-I is generally *missing both in theory and practice*.

Goal and contribution

- **Stabilize** AA with convergence beyond differentiability, locality and non-singularity

Goal and contribution

- **Stabilize** AA with convergence beyond differentiability, locality and non-singularity
  - The FP mapping of SCS is non-smooth and singular.

Goal and contribution

- **Stabilize** AA with convergence beyond differentiability, locality and non-singularity
  - The FP mapping of SCS is non-smooth and singular.
  - **Surprise:** stabilization also improves convergence consistently over both the original AA-I and AA-II.

Goal and contribution

- **Stabilize** AA with convergence beyond differentiability, locality and non-singularity
  - The FP mapping of SCS is non-smooth and singular.
  - **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
  - View an arbitrary unaccelerated algorithm as a **black-box** fixed-point iteration problem.
  - For example, concatenate successive iterates in momentum algorithms.

1 Motivation and Problem Statement

2 Acceleration: connecting quasi-Newton with extrapolation
- Good news and bad news

3 A generic stabilization scheme
- Stabilization of AA-I
- Stabilization of AA-II
- Global convergence: solvable settings
- A browse through effect of stabilization

4 Applications
- Conic optimization + SCS 2.x
- Prox-affine optimization + A2DR

5 Beyond convexity

A generic stabilization scheme

- Main idea: modified α^k (H_k/B_k) choice and a very relaxed safeguard.

A generic stabilization scheme

- Main idea: modified α^k (H_k/B_k) choice and a very relaxed safeguard.
- Iterates for $k = 1, 2, \ldots$, $(D,\ \epsilon > 0,\ M$ positive integer)

1. Compute $v_{\text{FP}}^{k+1} = F_\alpha(v^k), \quad g^k = v^k - v_{\text{FP}}^{k+1}$.
2. Update Y_k and S_k to include the new information
   & Compute a **modified** α^k (H_k/B_k)
3. Compute $v_{\text{AA}}^{k+1} = \sum_{j=0}^{M} \alpha_j^k v_{\text{FP}}^{k-M+j+1}$.
4. If the residual $\|G(v^k)\|_2 \le D\|g^0\|_2/n_{\text{AA}}^{1+\epsilon}$: (**safeguard**)
   Adopt $v^{k+i} = v_{\text{AA}}^{k+i}$ for $i = 1, \ldots, M$.
   (n_{AA}: # of adopted AA candidates)
5. Otherwise, take $v^{k+1} = v_{\text{FP}}^{k+1}$.

Motivations of the stabilization tricks

- Goal of modified α^k: boundedness of H_k and B_k.

Motivations of the stabilization tricks

- Goal of modified α^k: boundedness of H_k and B_k.
- Goal of safe-guard: avoid "wild" and "bad" extrapolation.

Motivations of the stabilization tricks

- Goal of modified α^k: boundedness of H_k and B_k.
- Goal of safe-guard: avoid "wild" and "bad" extrapolation.
  - Main idea: interleave AA steps with the vanilla KM iteration steps to safe-guard the decrease in residual norms G.

Motivations of the stabilization tricks

- Goal of modified α^k: boundedness of H_k and B_k.
- Goal of safe-guard: avoid "wild" and "bad" extrapolation.
  - Main idea: interleave AA 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, $\alpha \in (0,1)$. If F is averaged, just choose $\alpha = 1$.

Motivations of the stabilization tricks

- Goal of modified α^k: boundedness of H_k and B_k.
- Goal of safe-guard: avoid "wild" and "bad" extrapolation.
  - Main idea: interleave AA 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, $\alpha \in (0,1)$. If F is averaged, just choose $\alpha = 1$.
  - Can be seen as a much cheaper alternative to the expensive line-search.

1. Motivation and Problem Statement
2. Acceleration: connecting quasi-Newton with extrapolation
   - Good news and bad news
3. A generic stabilization scheme
   - Stabilization of AA-I
   - Stabilization of AA-II
   - Global convergence: solvable settings
   - A browse through effect of stabilization
4. Applications
   - Conic optimization + SCS 2.x
   - Prox-affine optimization + A2DR
5. Beyond convexity

Stabilization of AA-I: rank-one update

AA-I $\Longleftrightarrow$ 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 $B_k^0 = I$ as follows:

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

with $B_k = B_k^{m_k}$. Here $\{\hat{s}^i\}_{i=k-m_k}^{k-1}$ is the Gram-Schmidt orthogonalization of $\{s^i\}_{i=k-m_k}^{k-1}$, i.e., $\hat{s}^i = s^i - \sum_{j=k-m_k}^{i-1} \frac{(\hat{s}^j)^T s^i}{(\hat{s}^j)^T \hat{s}^j} \hat{s}^j, \quad i = k - m_k, \ldots, k-1.$

Stabilization of AA-I: 1. Powell-type regularization

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)

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

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)

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

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

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

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

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)

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

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

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

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

  - $|\det(B_k)|\geq\bar{\theta}^{m_k}>0$, and in particular, B_k is **invertible**!

Stabilization of AA-I: 2. Re-start checking

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

1 Motivation and Problem Statement

2 Acceleration: connecting quasi-Newton with extrapolation
- Good news and bad news

3 A generic stabilization scheme
- Stabilization of AA-I
- Stabilization of AA-II
- Global convergence: solvable settings
- A browse through effect of stabilization

4 Applications
- Conic optimization + SCS 2.x
- Prox-affine optimization + A2DR

5 Beyond convexity

Stabilization of AA-II: Adaptive Regularization

Our approach:

- Add *adaptive* regularization to the *unconstrained* formulation.

Our approach:

- Add *adaptive* regularization to the *unconstrained* formulation.
- Change variables to $\gamma^k \in \mathbf{R}^M$ (unconstrained LS):

$$\alpha_0^k = \gamma_0^k,\ \alpha_i^k = \gamma_i^k - \gamma_{i-1}^k,\ (i = 1, \dots, M-1),\ \alpha_M^k = 1 - \gamma_{M-1}^k$$

Stabilization of AA-II: Adaptive Regularization

Our approach:

- Add *adaptive* regularization to the *unconstrained* formulation.
- Change variables to $\gamma^k \in \mathbf{R}^M$ (unconstrained LS):

$$\alpha_0^k = \gamma_0^k,\ \alpha_i^k = \gamma_i^k - \gamma_{i-1}^k,\ (i = 1, \ldots, M-1),\ \alpha_M^k = 1 - \gamma_{M-1}^k$$

- Adaptive quadratic regularization: (adaptive LS)

$$\text{minimize} \quad \|g^k - Y_k\gamma^k\|_2^2 + \eta\left(\|S_k\|_F^2 + \|Y_k\|_F^2\right)\|\gamma^k\|_2^2,$$

where $\eta \geq 0$ is a regularization parameter and

$$g^k = G(v^k), \quad y^k = g^{k+1} - g^k, \quad Y_k = [y^{k-M} \ \ldots \ y^{k-1}]$$
$$s^k = v^{k+1} - v^k, \quad S_k = [s^{k-M} \ \ldots \ s^{k-1}]$$

Our approach:

- Adaptive quadratic regularization: (adaptive LS)

$$\text{minimize} \quad \|g^k - Y_k\gamma^k\|_2^2 + \eta\left(\|S_k\|_F^2 + \|Y_k\|_F^2\right)\|\gamma^k\|_2^2,$$

Lemma (Bounded approximate inverse Jacobian)

We have $v_{AA}^{k+1} = v^k - H_k g^k$, where $g^k = G(v^k)$ is the FP residual at v^k, and $\|H_k\|_2 \le 1 + 2/\eta$, where $\eta > 0$ is the regularization parameter in the adaptive LS subproblem.

1. Motivation and Problem Statement
2. Acceleration: connecting quasi-Newton with extrapolation
   - Good news and bad news
3. A generic stabilization scheme
   - Stabilization of AA-I
   - Stabilization of AA-II
   - Global convergence: solvable settings
   - A browse through effect of stabilization
4. Applications
   - Conic optimization + SCS 2.x
   - Prox-affine optimization + A2DR
5. Beyond convexity

Global convergence

Theorem

Suppose that f **has a fixed-point**. *Also suppose that f is non-expansive in l_2-norm or contractive in an arbitrary norm. Assume that $\{x^k\}_{k=0}^{\infty}$ is generated by the generic stabilization scheme wth α^k (H_k/B_k) chosen by either the stabilized AA-I or AA-II . Then we have*

$$\lim_{k\to\infty} x^k = x^\star,$$

where $x^\star = f(x^\star)$ is some fixed-point of f.

Global convergence

Theorem

Suppose that f **has a fixed-point***. Also suppose that f is non-expansive in l_2-norm or contractive in an arbitrary norm. Assume that $\{x^k\}_{k=0}^{\infty}$ is generated by the generic stabilization scheme wth α^k (H_k/B_k) chosen by either the stabilized AA-I or AA-II . Then we have*

$$\lim_{k\to\infty} x^k = x^\star,$$

where $x^\star = f(x^\star)$ is some fixed-point of f.

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

Global convergence

Theorem

Suppose that f **has a fixed-point**. *Also suppose that f is non-expansive in l_2-norm or contractive in an arbitrary norm. Assume that $\{x^k\}_{k=0}^{\infty}$ is generated by the generic stabilization scheme wth α^k (H_k/B_k) chosen by either the stabilized AA-I or AA-II . Then we have*

$$\lim_{k\to\infty} x^k = x^\star,$$

where $x^\star = f(x^\star)$ is some fixed-point of f.

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

What if f does not have a fixed-point? Pathological settings to be discussed later.

1. Motivation and Problem Statement
2. Acceleration: connecting quasi-Newton with extrapolation
   - Good news and bad news
3. A generic stabilization scheme
   - Stabilization of AA-I
   - Stabilization of AA-II
   - Global convergence: solvable settings
   - A browse through effect of stabilization
4. Applications
   - Conic optimization + SCS 2.x
   - Prox-affine optimization + A2DR
5. Beyond convexity

Effect of stabilization: AA-II + constant regularization

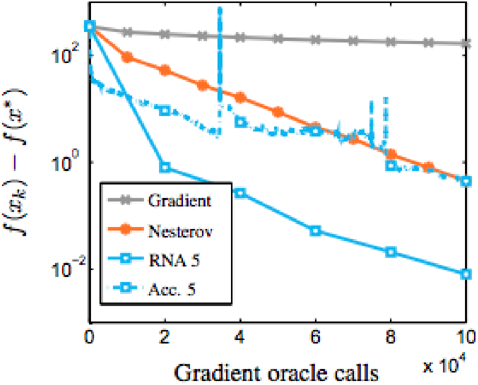


Figure: Effect of constant regularization (RNA, Scieur et al., 2016): ℓ_2 regularized logistic regression.

Effect of stabilization: AA-II + adaptive regularization

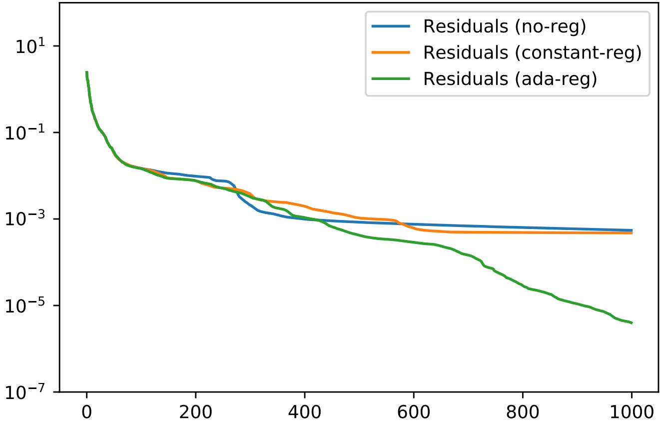


Figure: Further improvement of adaptive regularization (A2DR, FZB 2019): Nonnegative least squares.

Gradient Descent: stabilization from **divergence** to **convergence**

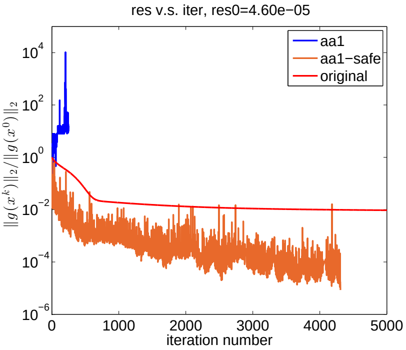


Figure: Gradient descent: regularized logistic regression. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

ISTA: elastic net regression – nonsmoothness coming from shrinkage

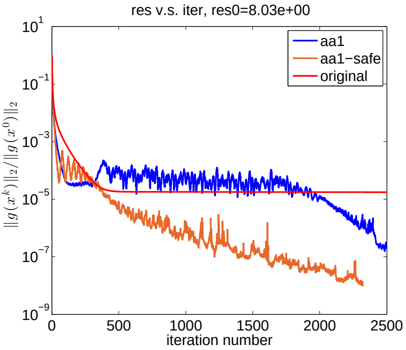


Figure: Iterative Shrinkage-Thresholding Algorithm: elastic-net linear regression. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

Effect of stabilization: AA-I

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

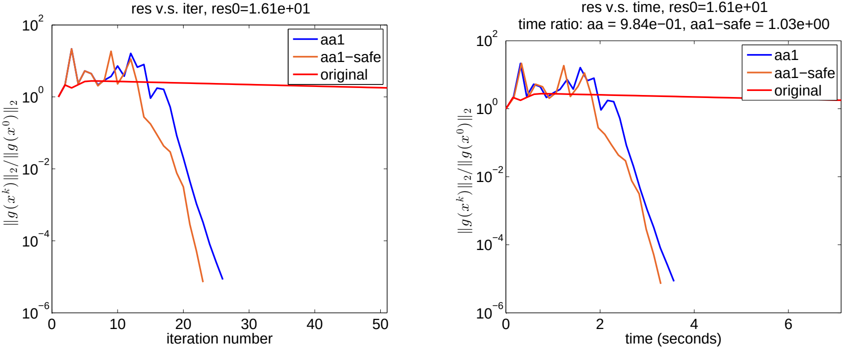


Figure: Value iteration: MDP. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

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

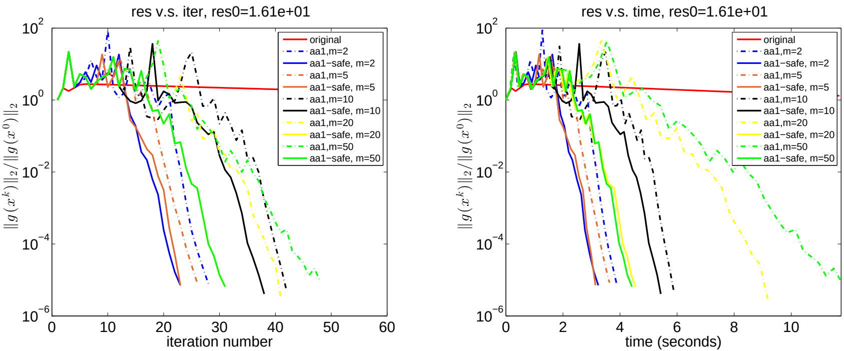


Figure: Value iteration: memory effect. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

1 Motivation and Problem Statement

2 Acceleration: connecting quasi-Newton with extrapolation
- Good news and bad news

3 A generic stabilization scheme
- Stabilization of AA-I
- Stabilization of AA-II
- Global convergence: solvable settings
- A browse through effect of stabilization

4 Applications
- Conic optimization + SCS 2.x
- Prox-affine optimization + A2DR

5 Beyond convexity

1. Motivation and Problem Statement

2. Acceleration: connecting quasi-Newton with extrapolation
   - Good news and bad news

3. A generic stabilization scheme
   - Stabilization of AA-I
   - Stabilization of AA-II
   - Global convergence: solvable settings
   - A browse through effect of stabilization

4. Applications
   - Conic optimization + SCS 2.x
   - Prox-affine optimization + A2DR

5. Beyond convexity

SCS as FP

- Problem: minimize$_x$ $c^T x$, subject to $Ax + s = b,\ s \in \mathcal{K}$.

SCS as FP

- Problem: minimize$_x$ $c^T x$, subject to $Ax + s = b, \ s \in \mathcal{K}$.
- Algorithm – SCS ($\mathcal{C} = \mathbb{R}^n \times \mathcal{K}^* \times \mathbb{R}_+$):

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

SCS as FP

- Problem: minimize$_x$ $c^T x$, subject to $Ax + s = b, \; s \in \mathcal{K}$.
- Algorithm – SCS ($\mathcal{C} = \mathbb{R}^n \times \mathcal{K}^* \times \mathbb{R}_+$):

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

- FP (don't apply AA to u and v separately):

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

- F is non-expansive (in ℓ_2), and 0 is always a fixed-point, and so the global convergence theorem goes through.

- We apply the stabilized AA-I to SCS.
- **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 $\frac{H_{k-j}^T \hat{s}_{k-j}}{\hat{s}_{k-j}^T (H_{k-j}\tilde{y}_{k-j})}$ for $j = 1, \ldots, m_k$, compute

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

  and then update $\tilde{x}^{k+1} = x^k - d_k$.
- **Problem scaling** is helpful when matrices are involved.

- Compared to **restarted Broyden**:

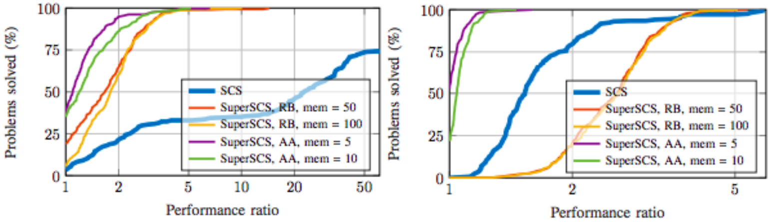


Figure: DM profile. left: sparse PCA; right: sparse logistic regression. *SuperSCS: fast and accurate large-scale conic optimization.* Sopasakis, et al., 2019.

Further success of AA-I: SCS 2.x

- Compared to **AA-II**:

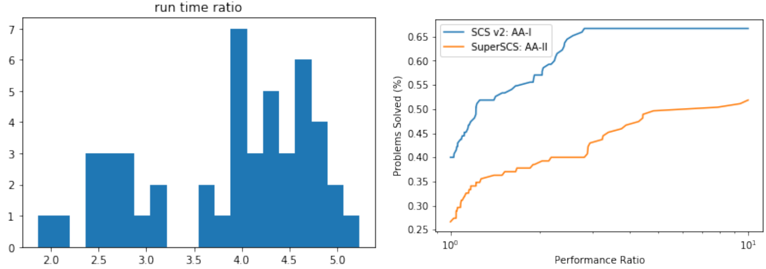


Figure: Sparse PCA. Left: histogram of run time ratio between SuperSCS (AA-II) and SCS 2.x (AA-I). Right: DM profile of run time.

- Still fail for 35% of the test cases.

Even further success with stabilized AA-I

SCS: LP – nonsmoothness coming from projections

- Implementation in progress in the next version of SCS.

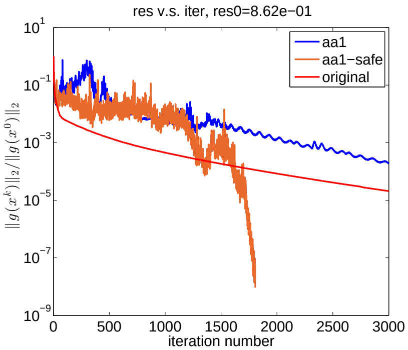


Figure: SCS: linear program. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

Even further success with stabilized AA-I

SCS: SOCP – nonsmoothness coming from projections

- Implementation in progress in the next version of SCS.

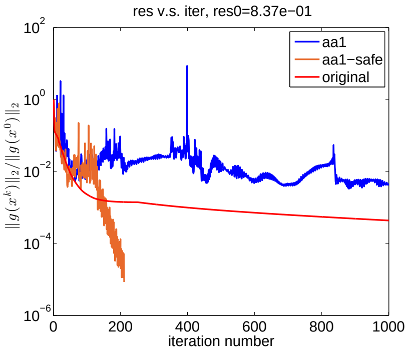


Figure: SCS: second-order cone program. Left: residual norm versus iteration. Right: residual norm versus time (seconds).

1 Motivation and Problem Statement

2 Acceleration: connecting quasi-Newton with extrapolation
- Good news and bad news

3 A generic stabilization scheme
- Stabilization of AA-I
- Stabilization of AA-II
- Global convergence: solvable settings
- A browse through effect of stabilization

4 Applications
- Conic optimization + SCS 2.x
- Prox-affine optimization + A2DR

5 Beyond convexity

- Open-sourced Python Solver for Prox-Affine Distributed Convex Optimization
- Combining AA-II with DRS (Douglas-Rachford Splitting).
- Available at `https://github.com/cvxgrp/a2dr`

Prox-affine form of generic convex optimization

We consider the following **prox-affine** representation/formulation of a **generic** convex optimization problem:

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=1}^N f_i(x_i) \\
\text{subject to} & \sum_{i=1}^N A_i x_i = b.
\end{array}
$$

with variable $x = (x_1, \ldots, x_N) \in \mathbf{R}^{n_1 + \cdots + n_N}$, $A_i \in \mathbf{R}^{m \times n_i}$, $b \in \mathbf{R}^m$.

Prox-affine form of generic convex optimization

We consider the following **prox-affine** representation/formulation of a **generic** convex optimization problem:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} & \sum_{i=1}^{N} A_i x_i = b. \end{array}$$

with variable $x = (x_1, \ldots, x_N) \in \mathbf{R}^{n_1+\cdots+n_N}$, $A_i \in \mathbf{R}^{m \times n_i}$, $b \in \mathbf{R}^m$.

- $f_i : \mathbf{R}^{n_i} \to \mathbf{R} \cup \{+\infty\}$ is closed, convex and proper (CCP).
- Each f_i can **only** be accessed through its proximal operator:

$$\mathbf{prox}_{tf_i}(v_i) = \mathrm{argmin}_{x_i} \ \left(f_i(x_i) + \frac{1}{2t}\|x_i - v_i\|_2^2\right).$$

Prox-affine form of generic convex optimization

Why **prox-affine** form?

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^N f_i(x_i) \\ \text{subject to} & \sum_{i=1}^N A_i x_i = b. \end{array}$$

- **Separability:** suitable for parallel and distributed implementation.

Prox-affine form of generic convex optimization

Why **prox-affine** form?

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^N f_i(x_i) \\ \text{subject to} & \sum_{i=1}^N A_i x_i = b. \end{array}$$

- **Separability:** suitable for parallel and distributed implementation.
- **Black-box proximal:** suitable for peer-to-peer optimization with privacy requirements.

Prox-affine form of generic convex optimization

Why **prox-affine** form?

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} & \sum_{i=1}^{N} A_i x_i = b. \end{array}$$

- **Separability:** suitable for parallel and distributed implementation.
- **Black-box proximal:** suitable for peer-to-peer optimization with privacy requirements.
- **New interface:** good substitute for the **conic** standard form.

Prox-affine form of generic convex optimization

Why **prox-affine** form?

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} & \sum_{i=1}^{N} A_i x_i = b. \end{array}$$

- **Separability:** suitable for parallel and distributed implementation.
- **Black-box proximal:** suitable for peer-to-peer optimization with privacy requirements.
- **New interface:** good substitute for the **conic** standard form.
  - Cone programs can be represented in prox-affine form by consensus without complication (but NOT vice versa).

Prox-affine form of generic convex optimization

Why **prox-affine** form?

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^N f_i(x_i) \\ \text{subject to} & \sum_{i=1}^N A_i x_i = b. \end{array}$$

- **Separability:** suitable for parallel and distributed implementation.
- **Black-box proximal:** suitable for peer-to-peer optimization with privacy requirements.
- **New interface:** good substitute for the **conic** standard form.
  - Cone programs can be represented in prox-affine form by consensus without complication (but NOT vice versa).
  - With log, exp, det involved, prox-affine form is much more compact.

Prox-affine form of generic convex optimization

Why **prox-affine** form?

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^N f_i(x_i) \\ \text{subject to} & \sum_{i=1}^N A_i x_i = b. \end{array}$$

- **Separability:** suitable for parallel and distributed implementation.
- **Black-box proximal:** suitable for peer-to-peer optimization with privacy requirements.
- **New interface:** good substitute for the **conic** standard form.
  - Cone programs can be represented in prox-affine form by consensus without complication (but NOT vice versa).
  - With log, exp, det involved, prox-affine form is much more compact.

Interface of a2dr:

```
x_vals, primal, dual, num_iters, solve_time = a2dr(p_list, A_list, b)
```

Try it out! Simply provide a list of proximal functions $\mathbf{prox}_{tf_i}(v_i)$ (p_list), list of A_i's (A_list), and b (b), and you are done!

Interface of a2dr:

```
x_vals, primal, dual, num_iters, solve_time = a2dr(p_list, A_list, b)
```

Try it out! Simply provide a list of proximal functions $\mathbf{prox}_{tf_i}(v_i)$ (p_list), list of A_i's (A_list), and b (b), and you are done!

Why a2dr?

a2dr: Solver interface

Interface of a2dr:

```
x_vals, primal, dual, num_iters, solve_time = a2dr(p_list, A_list, b)
```

Try it out! Simply provide a list of proximal functions $\mathbf{prox}_{tf_i}(v_i)$ (p_list), list of A_i's (A_list), and b (b), and you are done!

Why a2dr?

- Hundreds of papers on distributed/parallel optimization every year

Interface of a2dr:

```
x_vals, primal, dual, num_iters, solve_time = a2dr(p_list, A_list, b)
```

Try it out! Simply provide a list of proximal functions $\mathbf{prox}_{tf_i}(v_i)$ (p_list), list of A_i's (A_list), and b (b), and you are done!

Why a2dr?

- Hundreds of papers on distributed/parallel optimization every year
- Few solvers/softwares are written

Interface of a2dr:

```
x_vals, primal, dual, num_iters, solve_time = a2dr(p_list, A_list, b)
```

Try it out! Simply provide a list of proximal functions $\mathbf{prox}_{tf_i}(v_i)$ (p_list), list of A_i's (A_list), and b (b), and you are done!

Why a2dr?

- Hundreds of papers on distributed/parallel optimization every year
- Few solvers/softwares are written
- Existing good ones: CoCoA(+), TMAC, etc.
    - Efficient in communication cost
    - But hard to extend and use for general purposes.
    - Intended mostly for optimization experts.

a2dr: Solver interface

Interface of a2dr:

```
x_vals, primal, dual, num_iters, solve_time = a2dr(p_list, A_list, b)
```

Try it out! Simply provide a list of proximal functions $\mathbf{prox}_{tf_i}(v_i)$ (p_list), list of A_i's (A_list), and b (b), and you are done!

Why a2dr?

- Hundreds of papers on distributed/parallel optimization every year
- Few solvers/softwares are written
- Existing good ones: CoCoA(+), TMAC, etc.
  - Efficient in communication cost
  - But hard to extend and use for general purposes.
  - Intended mostly for optimization experts.

Finally: CVXPY + a2dr – Expression tree complier exists: Epsilon (Wytock et al., 2015).

Previous Work

Most common approaches for prox-affine formulation (sometimes goes by the name "distributed optimization"):

- Alternating direction method of multipliers (ADMM).
- Douglas-Rachford splitting (DRS).
- Augmented Lagrangian method (ALM).

Previous Work

Most common approaches for prox-affine formulation (sometimes goes by the name "distributed optimization"):

- Alternating direction method of multipliers (ADMM).
- Douglas-Rachford splitting (DRS).
- Augmented Lagrangian method (ALM).

These are typically slow to converge – acceleration techniques:

- Adaptive penalty parameters.
- Momentum methods.
- Quasi-Newton or Newton-type method with line search.

Our Method

A2DR: Stabilized AA-II applied to DRS

A2DR: Stabilized AA-II applied to DRS

- Why **AA**?

Our Method

A2DR: Stabilized AA-II applied to DRS

- Why **AA**?
  - *Fast and cheap:* As fast as (quasi-)Newton acceleration, but as memory efficient as adaptive penalty and momentum, and line-search free

Our Method

A2DR: Stabilized AA-II applied to DRS

- Why **AA**?
  - *Fast and cheap:* As fast as (quasi-)Newton acceleration, but as memory efficient as adaptive penalty and momentum, and line-search free
- Why **AA-II**?

Our Method

A2DR: Stabilized AA-II applied to DRS

- Why **AA**?
  - *Fast and cheap:* As fast as (quasi-)Newton acceleration, but as memory efficient as adaptive penalty and momentum, and line-search free
- Why **AA-II**?
  - Work better with DRS + prox-affine than type-I AA

Our Method

A2DR: Stabilized AA-II applied to DRS

- Why **AA**?
  - *Fast and cheap:* As fast as (quasi-)Newton acceleration, but as memory efficient as adaptive penalty and momentum, and line-search free
- Why **AA-II**?
  - Work better with DRS + prox-affine than type-I AA
  - Better stability for **general purpose** solvers and distributed settings.
    - **prox** operators have much larger diversity than solvable cones in SCS.

Our Method

A2DR: Stabilized AA-II applied to DRS

- Why **AA**?
  - *Fast and cheap:* As fast as (quasi-)Newton acceleration, but as memory efficient as adaptive penalty and momentum, and line-search free
- Why **AA-II**?
  - Work better with DRS + prox-affine than type-I AA
  - Better stability for **general purpose** solvers and distributed settings.
    - **prox** operators have much larger diversity than solvable cones in SCS.
- Why **DRS**?

Our Method

A2DR: Stabilized AA-II applied to DRS

- Why **AA**?
  - *Fast and cheap:* As fast as (quasi-)Newton acceleration, but as memory efficient as adaptive penalty and momentum, and line-search free
- Why **AA-II**?
  - Work better with DRS + prox-affine than type-I AA
  - Better stability for **general purpose** solvers and distributed settings.
    - **prox** operators have much larger diversity than solvable cones in SCS.
- Why **DRS**?
  - Allows for a natural NEFP representation (ADMM not), and amenable to proximal evaluation (ALM not).

Challenges and contribution

Major Challenge:

- **Pathology:** The FP of DRS does not always have a fixed-point solution (unlike SCS).

Major Challenge:

- **Pathology:** The FP of DRS does not always have a fixed-point solution (unlike SCS).
- **Implementation:** tuning-free and off-the-shelf.

Challenges and contribution

Major Challenge:

- **Pathology:** The FP of DRS does not always have a fixed-point solution (unlike SCS).
- **Implementation:** tuning-free and off-the-shelf.

Theory: First globally convergent type-II AA variant in non-smooth and potentially **pathological** settings.

Challenges and contribution

Major Challenge:

- **Pathology:** The FP of DRS does not always have a fixed-point solution (unlike SCS).
- **Implementation:** tuning-free and off-the-shelf.

Theory: First globally convergent type-II AA variant in non-smooth and potentially **pathological** settings.

Practice: An open-source Python solver `a2dr` based on **A2DR**:

`https://github.com/cvxgrp/a2dr`.

DRS Algorithm

- Rewrite problem as ($\mathcal{I}_S$ is the indicator of set S)

$$\text{minimize} \quad \overbrace{\sum_{i=1}^{N} f_i(x_i)}^{f(x)} + \overbrace{\mathcal{I}_{Ax=b}(x)}^{g(x)}.$$

DRS Algorithm

- Rewrite problem as ($\mathcal{I}_S$ is the indicator of set S)

$$\text{minimize} \quad \overbrace{\sum_{i=1}^{N} f_i(x_i)}^{f(x)} + \overbrace{\mathcal{I}_{Ax=b}(x)}^{g(x)}.$$

- DRS iterates for $k = 1, 2, \ldots,$

$$\begin{aligned} x_i^{k+1/2} &= \mathbf{prox}_{tf_i}(v^k), \quad i = 1, \ldots, N \\ v^{k+1/2} &= 2x^{k+1/2} - v^k \\ x^{k+1} &= \Pi_{Av=b}(v^{k+1/2}) \\ v^{k+1} &= v^k + x^{k+1} - x^{k+1/2} \end{aligned}$$

$\Pi_S(v)$ is Euclidean projection of v onto S.

Convergence of DRS

- DRS iterations can be conceived as a fixed point (FP) mapping

$$v^{k+1} = F(v^k)$$

- F is **firmly non-expansive**.
- v^k converges to a fixed point of F (if it exists).
- x^k and $x^{k+1/2}$ converge to a solution of our problem.

Convergence of DRS

- DRS iterations can be conceived as a fixed point (FP) mapping

$$v^{k+1} = F(v^k)$$

- F is **firmly non-expansive**.
- v^k converges to a fixed point of F (if it exists).
- x^k and $x^{k+1/2}$ converge to a solution of our problem.

In practice, this convergence is often rather slow.

Convergence of DRS

- DRS iterations can be conceived as a fixed point (FP) mapping

$$v^{k+1} = F(v^k)$$

- F is **firmly non-expansive**.
- v^k converges to a fixed point of F (if it exists).
- x^k and $x^{k+1/2}$ converge to a solution of our problem.

In practice, this convergence is often rather slow.
So we add AA-II.

Performance/Stopping Criterion of A2DR

- Stop and output $x^{k+1/2}$ when $\|r^k\|_2 \le \epsilon_{\text{tol}} = \epsilon_{\rm abs} + \epsilon_{\rm rel}\|r^0\|_2$:

$$\begin{aligned} r^k_{\text{prim}} &= Ax^{k+1/2} - b, \\ r^k_{\text{dual}} &= \frac{1}{t}(v^k - x^{k+1/2}) + A^T\lambda^k, \\ r^k &= (r^k_{\rm prim}, r^k_{\rm dual}). \end{aligned}$$

Performance/Stopping Criterion of A2DR

- Stop and output $x^{k+1/2}$ when $\|r^k\|_2 \leq \epsilon_{\mathsf{tol}} = \epsilon_{\mathrm{abs}} + \epsilon_{\mathrm{rel}}\|r^0\|_2$:

$$
\begin{aligned}
r_{\mathsf{prim}}^k &= Ax^{k+1/2} - b, \\
r_{\mathsf{dual}}^k &= \tfrac{1}{t}(v^k - x^{k+1/2}) + A^T\lambda^k, \\
r^k &= (r_{\mathrm{prim}}^k, r_{\mathrm{dual}}^k).
\end{aligned}
$$

- Remark:
  - Just KKT conditions. Notice that $(v^k - x^{k+1/2})/t \in \partial f(x^{k+1/2})$.
  - $\mathbf{prox}_f$ is enough, and no need for access to f or its sub-gradient.
- Dual variable is solution to least-squares problem

$$
\lambda^k = \mathrm{argmin}_\lambda \; \|r_{\mathsf{dual}}^k\|_2
$$

Key lemma to the proof

Lemma (Connecting FP residuals with OPT residuals)

Suppose that $\liminf_{j\to\infty} \|G(v^j)\|_2 \leq \epsilon$ *for some* $\epsilon > 0$*, then*

$$\liminf_{j\to\infty} \|r^j_{\mathrm{prim}}\|_2 \leq \|A\|_2\epsilon, \quad \liminf_{j\to\infty} \|r^j_{\mathrm{dual}}\|_2 \leq \frac{1}{t}\epsilon.$$

Convergence of A2DR

Theorem (Solvable Case)

If the problem is solvable (e.g., feasible and bounded), then

$$\liminf_{k\to\infty} \|r^k\|_2 = 0$$

and the AA candidates are adopted infinitely often. Furthermore, if F has a fixed point, then

$$\lim_{k\to\infty} v^k = v^\star \text{ and } \lim_{k\to\infty} x^{k+1/2} = x^\star,$$

where $v^\star$ is a fixed-point of F and $x^\star$ is a solution to our problem.

Remark. when the proximal operators and projections are evaluated with *errors* bounded by ϵ, then $\liminf_{k\to\infty} \|r^k\|_2 = O(\sqrt{\epsilon})$.

Convergence of A2DR

Theorem (Pathological Case)

If the problem is pathological (strongly primal infeasible or strongly dual infeasible), then

$$\lim_{k\to\infty}\left(v^k - v^{k+1}\right) = \delta v \neq 0.$$

Furthermore, if $\lim_{k\to\infty} Ax^{k+1/2} = b$*, then the problem is unbounded and* $\|\delta v\|_2 = t\,\mathbf{dist}(\mathbf{dom}\, f^*, \mathbf{range}(A^T))$*.*
Otherwise, it is infeasible and $\|\delta v\|_2 \geq \mathbf{dist}(\mathbf{dom}\, f, \{x : Ax = b\})$ *with equality when the dual problem is feasible.*

Implementation

Pre-conditioning (convergence greatly improved by rescaling problem):

- Replace original A, b, f_i with

$$\hat{A} = DAE, \quad \hat{b} = Db, \quad \hat{f}_i(\hat{x}_i) = f_i(e_i \hat{x}_i)$$

- D and E are diagonal positive, $e_i > 0$ corresponds to ith block diagonal entry of E, and chosen by equilibrating A
- Proximal operator of $\hat{f}_i$ can be evaluated using proximal operator of f_i

$$\mathbf{prox}_{t\hat{f}_i}(\hat{v}_i) = \frac{1}{e_i}\mathbf{prox}_{(e_i^2 t)f_i}(e_i \hat{v}_i)$$

Pre-conditioning (convergence greatly improved by rescaling problem):

- Replace original A, b, f_i with

$$\hat{A} = DAE, \quad \hat{b} = Db, \quad \hat{f}_i(\hat{x}_i) = f_i(e_i \hat{x}_i)$$

- D and E are diagonal positive, $e_i > 0$ corresponds to ith block diagonal entry of E, and chosen by equilibrating A
- Proximal operator of $\hat{f}_i$ can be evaluated using proximal operator of f_i

$$\mathbf{prox}_{t\hat{f}_i}(\hat{v}_i) = \frac{1}{e_i}\mathbf{prox}_{(e_i^2 t) f_i}(e_i \hat{v}_i)$$

Choice of t (in DRS, $\mathbf{prox}_{tf_i}$): $t = \frac{1}{10}\left(\prod_{j=1}^N e_j\right)^{-2/N}$.

Implementation

Pre-conditioning (convergence greatly improved by rescaling problem):

- Replace original A, b, f_i with

$$\hat{A} = DAE, \quad \hat{b} = Db, \quad \hat{f}_i(\hat{x}_i) = f_i(e_i \hat{x}_i)$$

- D and E are diagonal positive, $e_i > 0$ corresponds to ith block diagonal entry of E, and chosen by equilibrating A
- Proximal operator of $\hat{f}_i$ can be evaluated using proximal operator of f_i

$$\mathbf{prox}_{t\hat{f}_i}(\hat{v}_i) = \frac{1}{e_i}\mathbf{prox}_{(e_i^2 t)f_i}(e_i \hat{v}_i)$$

Choice of t (in DRS, $\mathbf{prox}_{tf_i}$): $t = \frac{1}{10}\left(\prod_{j=1}^N e_j\right)^{-2/N}$.

Parallelization: `multiprocessing` package in Python.

Nonnegative Least Squares (NNLS)

$$
\begin{array}{ll}
\text{minimize} & \|Fz - g\|_2^2 \\
\text{subject to} & z \geq 0
\end{array}
$$

with respect to $z \in \mathbf{R}^q$

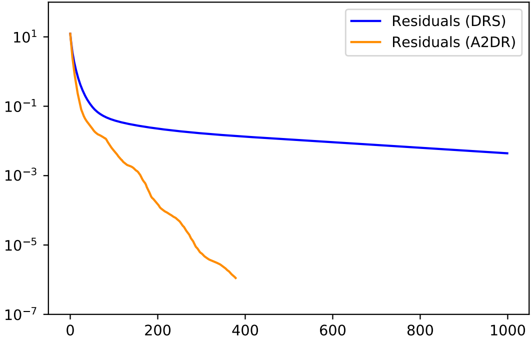
- Problem data: $F \in \mathbf{R}^{p \times q}$ and $g \in \mathbf{R}^p$
- Can be written in standard form with

$$
f_1(x_1) = \|Fx_1 - g\|_2^2, \quad f_2(x_2) = \mathcal{I}_{\mathbf{R}_+^n}(x_2)
$$
$$
A_1 = I, \quad A_2 = -I, \quad b = 0
$$

- We evaluate proximal operator of f_1 using LSQR

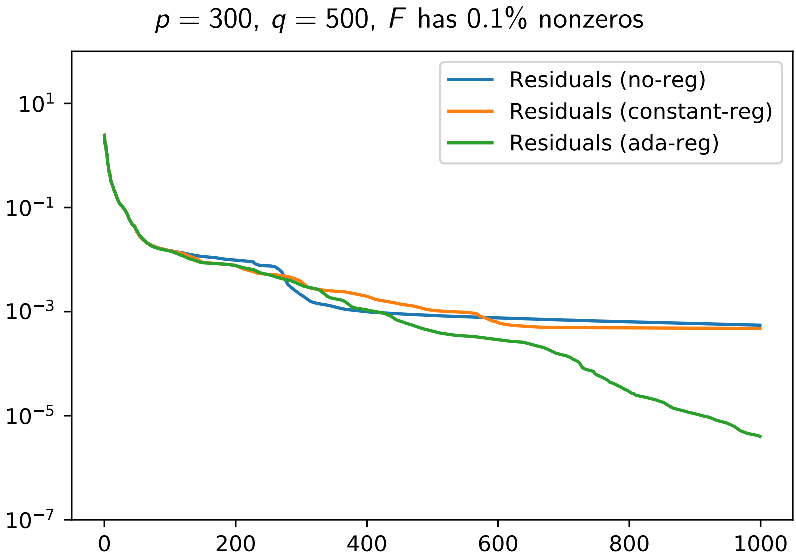
NNLS: Convergence of $\|r^k\|_2$

$p = 10^4$, $q = 8000$, F has 0.1% nonzeros



OSQP and SCS took respectively 349 and 327 seconds, while A2DR only took 55 seconds.

NNLS: Effect of regularization



Sparse Inverse Covariance Estimation

- Samples $z_1, \ldots, z_p$ IID from $\mathcal{N}(0, \Sigma)$
- Know covariance $\Sigma \in \mathbf{S}^q_+$ has **sparse** inverse $S = \Sigma^{-1}$
- One way to estimate S is by solving the penalized log-likelihood problem

$$\text{minimize} \quad -\log\det(S) + \mathrm{tr}(SQ) + \alpha\|S\|_1,$$

  where Q is the sample covariance, $\alpha \geq 0$ is a parameter
- Note $\log\det(S) = -\infty$ when $S \not\succ 0$

Sparse Inverse Covariance Estimation

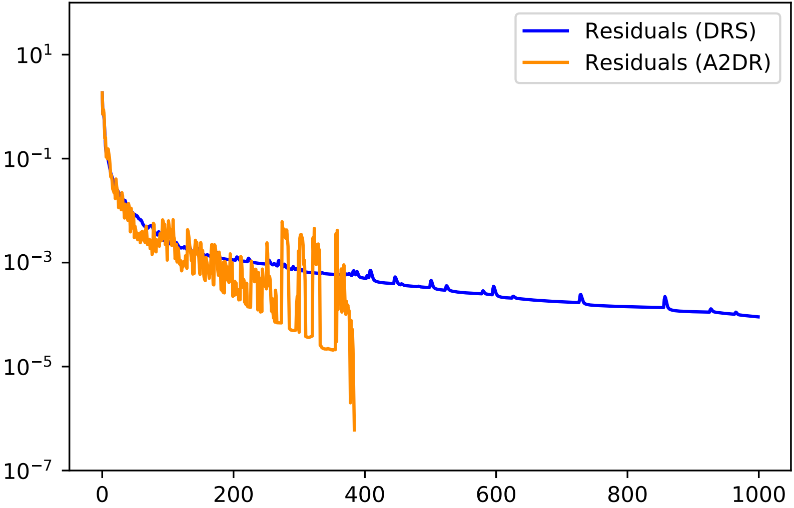
- Problem can be written in standard form with

$$f_1(S_1) = -\log\det(S_1) + \mathrm{tr}(S_1 Q), \quad f_2(S_2) = \alpha\|S_2\|_1,$$
$$A_1 = I, \quad A_2 = -I, \quad b = 0.$$

- Both proximal operators have closed-form solutions.

Covariance Estimation: Convergence of $\|r^k\|_2$

$p = 1000$, $q = 100$, S has 10% nonzeros



Covariance Estimation: larger examples

Ran A2DR on instances with $q = 1200$ and $q = 2000$ (vectorizations on the order of 10^6) and compared its performance to SCS:

Covariance Estimation: larger examples

Ran A2DR on instances with $q = 1200$ and $q = 2000$ (vectorizations on the order of 10^6) and compared its performance to SCS:

- In the former case, A2DR took 1 hour to converge to a tolerance of 10^{-3}, while SCS took 11 hours to achieve a tolerance of 10^{-1} and yielded a much worse objective value.

Covariance Estimation: larger examples

Ran A2DR on instances with $q = 1200$ and $q = 2000$ (vectorizations on the order of 10^6) and compared its performance to SCS:

- In the former case, A2DR took 1 hour to converge to a tolerance of 10^{-3}, while SCS took 11 hours to achieve a tolerance of 10^{-1} and yielded a much worse objective value.
- In the latter case, A2DR converged in 2.6 hours to a tolerance of 10^{-3}, while SCS failed immediately with an out-of-memory error.

Multi-Task Logistic Regression

$$\text{minimize} \quad \phi(W\theta, Y) + \alpha \textstyle\sum_{l=1}^{L} \|\theta_l\|_2 + \beta\|\theta\|_*$$

with respect to $\theta = [\theta_1 \cdots \theta_L] \in \mathbf{R}^{s \times L}$

- Problem data: $W \in \mathbf{R}^{p \times s}$ and $Y = [y_1 \cdots y_L] \in \mathbf{R}^{p \times L}$
- Regularization parameters: $\alpha \geq 0, \beta \geq 0$
- Logistic loss function

$$\phi(Z, Y) = \sum_{l=1}^{L} \sum_{i=1}^{p} \log\left(1 + \exp(-Y_{il} Z_{il})\right)$$

Multi-Task Logistic Regression

- Rewrite problem in standard form with:

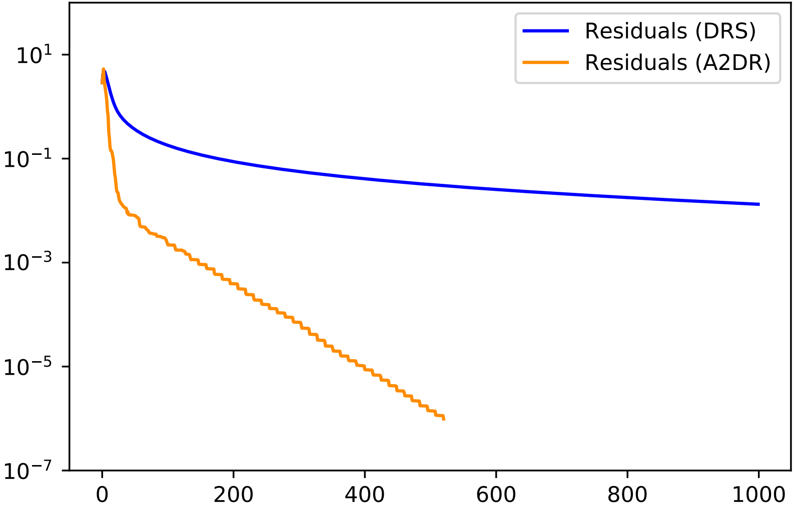
$$f_1(Z) = \phi(Z, Y), \quad f_2(\theta) = \alpha \sum_{l=1}^{L} \|\theta_l\|_2, \quad f_3(\tilde{\theta}) = \beta \|\tilde{\theta}\|_*,$$

$$A = \begin{bmatrix} I & -W & 0 \\ 0 & I & -I \end{bmatrix}, \quad x = \begin{bmatrix} Z \\ \theta \\ \tilde{\theta} \end{bmatrix}, \quad b = 0$$

- We evaluate proximal operator of f_1 using Newton-CG method, and the rest with closed-form formulae.

Multi-Task Logistic: Convergence of $\|r^k\|_2$

$p = 300,\ s = 500,\ L = 10,\ \alpha = \beta = 0.1$



Other examples

A (very) brief summary of other examples (see the paper for more details):

- l_1 trend filtering.
- Stratified models.
- Single commodity flow optimization (match the performance of OSQP, and largely outperform SCS).
- Optimal control (largely outperform both SCS and OSQP).
- Coupled quadratic program (match the performance of OSQP and SCS).

Other examples

A (very) brief summary of other examples (see the paper for more details):

- l_1 trend filtering.
- Stratified models.
- Single commodity flow optimization (match the performance of OSQP, and largely outperform SCS).
- Optimal control (largely outperform both SCS and OSQP).
- Coupled quadratic program (match the performance of OSQP and SCS).

Remark. The advantage compared to OSQP probably comes from the inclusion of AA, while the advantage compared to SCS (which includes type-I AA) is probably due to the more compact standard form representation.

Summary of A2DR

- A2DR is a fast, robust algorithm for solving generic (non-smooth) convex optimization problems in the prox-affine form.

Summary of A2DR

- A2DR is a fast, robust algorithm for solving generic (non-smooth) convex optimization problems in the prox-affine form.
- Parallelized, scalable and memory-efficient.

Summary of A2DR

- A2DR is a fast, robust algorithm for solving generic (non-smooth) convex optimization problems in the prox-affine form.
- Parallelized, scalable and memory-efficient.
- Consistent and fast convergence with no parameter tuning, and beat SOTA open source solvers like SCS (2.x) and OSQP in many cases.

Summary of A2DR

- A2DR is a fast, robust algorithm for solving generic (non-smooth) convex optimization problems in the prox-affine form.
- Parallelized, scalable and memory-efficient.
- Consistent and fast convergence with no parameter tuning, and beat SOTA open source solvers like SCS (2.x) and OSQP in many cases.
- Produces primal and dual solutions, or a certificate of infeasibility/unboundedness.

Summary of A2DR

- A2DR is a fast, robust algorithm for solving generic (non-smooth) convex optimization problems in the prox-affine form.
- Parallelized, scalable and memory-efficient.
- Consistent and fast convergence with no parameter tuning, and beat SOTA open source solvers like SCS (2.x) and OSQP in many cases.
- Produces primal and dual solutions, or a certificate of infeasibility/unboundedness.
- Python library:

`https://github.com/cvxgrp/a2dr`

Future Work on A2DR

- More work on feasibility detection.
- Expand library of proximal operators (non-convex proximal).
- User-friendly interface with CVXPY (with the help of `Epsilon`).
- GPU parallelization and cloud computing,

1 Motivation and Problem Statement

2 Acceleration: connecting quasi-Newton with extrapolation
- Good news and bad news

3 A generic stabilization scheme
- Stabilization of AA-I
- Stabilization of AA-II
- Global convergence: solvable settings
- A browse through effect of stabilization

4 Applications
- Conic optimization + SCS 2.x
- Prox-affine optimization + A2DR

5 Beyond convexity

Beyond non-expansiveness (convexity)

- Our stabilization technique can actually be extended to generic **non-convex** optimization settings.

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.

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.

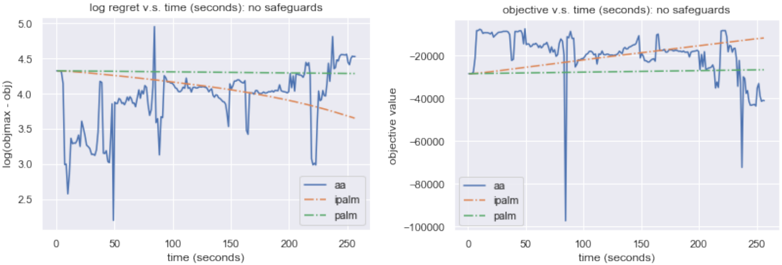


Figure: MLE of MHPs: exponential hawkes. No safe-guards. Left: log-regret v.s. time (seconds). Right: objective v.s. time (seconds).

Safe-guards in non-convex optimization

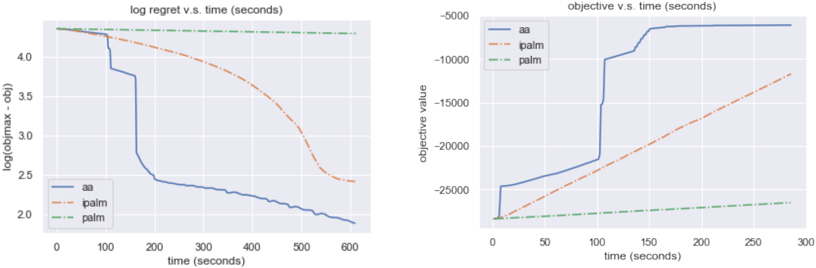
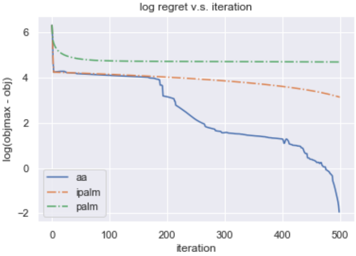


Figure: MLE of MHPs: synthetic exponential hawkes. With safe-guards. Left: log-regret v.s. time (seconds). Right: objective v.s. time (seconds).

Safe-guards in non-convex optimization



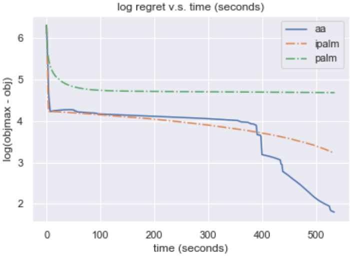


Figure: MLE of MHPs: synthetic power law hawkes. With safe-guards. Left: log-regret v.s. iterations Right: log-regret v.s. time (seconds).

Safe-guards in non-convex optimization

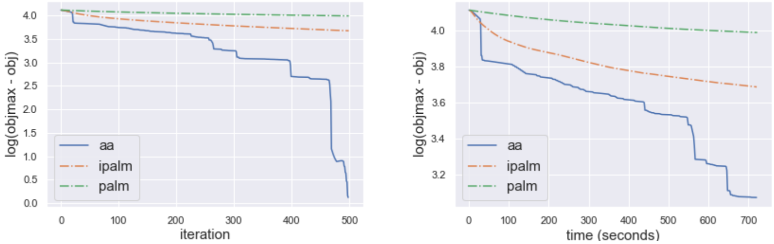


Figure: MLE of MHPs: memetracker dataset + exponential hawkes. With safe-guards. Left: log-regret v.s. iterations Right: log-regret v.s. time (seconds).

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.

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.

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).

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.

References

Zhang, J., O'Donoghue, B, and Boyd, S. P. (2018).
Globally Convergent Type-I Anderson Acceleration for Non-Smooth Fixed-Point Iterations.
arXiv preprint arXiv:1808.03971.

Fu, A.*, Zhang, J.* and Boyd, S. P. (2019). (*equal contribution)
Anderson Accelerated Douglas-Rachford Splitting.
arXiv preprint arXiv:1908.11482.

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

Acknowledgment

- Thanks to Brendan O'Donoghue for his advice on pre-conditioning and his inspirational ideas of developing solvers with Anderson acceleration, pioneered by SCS 2.x:
    - Zhang, J., O'Donoghue, B. and Boyd, S. P. (2018).
- Thanks to Anqi Fu for the input to the A2DR part of the slides.

Thanks for listening!

Any questions?