SIAM J. SCI. COMPUT.
Vol. 42, No. 6, pp. A3560–A3583

ANDERSON ACCELERATED DOUGLAS–RACHFORD SPLITTING*

ANQI FU†, JUNZI ZHANG‡, AND STEPHEN BOYD†

Abstract. We consider the problem of nonsmooth convex optimization with linear equality constraints, where the objective function is only accessible through its proximal operator. This problem arises in many different fields such as statistical learning, computational imaging, telecommunications, and optimal control. To solve it, we propose an Anderson accelerated Douglas–Rachford splitting (A2DR) algorithm, which we show either globally converges or provides a certificate of infeasibility/unboundedness under very mild conditions. Applied to a block separable objective, A2DR partially decouples so that its steps may be carried out in parallel, yielding an algorithm that is fast and scalable to multiple processors. We describe an open-source implementation and demonstrate its performance on a wide range of examples.

Key words. Anderson acceleration, nonsmooth convex optimization, parallel and distributed optimization, proximal oracles, stabilization, global convergence, pathological settings

AMS subject classifications. 49J52, 65K05, 68W10, 68W15, 90C25, 90C53, 97N80

DOI. 10.1137/19M1290097

1. Introduction.

1.1. Problem setting.

Consider the convex optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b \end{array} \tag{1.1}$$

with variable $x \in \mathbf{R}^n$, where $f : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ is convex, closed, and proper (CCP), and $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$ are given. We assume that the linear constraint $Ax = b$ is feasible.

Block form. In this paper, we work with block separable f, i.e., $f(x) = \sum_{i=1}^N f_i(x_i)$ for individually CCP $f_i : \mathbf{R}^{n_i} \to \mathbf{R} \cup \{+\infty\}$, $i = 1, \ldots, N$. We partition $x = (x_1, \ldots, x_N)$ so that $n = \sum_{i=1}^N n_i$ and let $A = [A_1\ A_2\ \cdots\ A_N]$ with $A_i \in \mathbf{R}^{m\times n_i}$, $i = 1, \ldots, N$. Problem (1.1) can be written in terms of the block variables as

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^N f_i(x_i) \\ \text{subject to} & \sum_{i=1}^N A_i x_i = b. \end{array} \tag{1.2}$$

Many interesting problems have the form (1.2), such as consensus optimization [10] and cone programming [35]. In fact, by transforming nonlinear convex constraints (e.g., cone constraints) into set indicator functions and adding them to the objective function, any convex optimization problem can be written in the above form.

Optimality conditions. The point $x \in \mathbf{R}^n$ is a solution to (1.2) if there exist $g \in \mathbf{R}^n$ and $\lambda \in \mathbf{R}^m$ such that

$$Ax = b, \tag{1.3}$$

*Submitted to the journal's Methods and Algorithms for Scientific Computing section September 27, 2019; accepted for publication (in revised form) July 31, 2020; published electronically November 9, 2020. *Anqi Fu and Junzi Zhang contributed equally to this work.
https://doi.org/10.1137/19M1290097
Funding: The work of the first and second authors was each supported by a Stanford Graduate Fellowship.
†Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (anqif@stanford.edu, boyd@stanford.edu).
‡ICME, Stanford University, Palo Alto, CA 94304 USA (junziz@stanford.edu).

$$0 = g + A^T\lambda, \quad g \in \partial f(x), \tag{1.4}$$

where $\partial f(x)$ is the subdifferential of f at x. With block separability, (1.4) can be written as

$$0 = g_i + A_i^T\lambda, \quad g_i \in \partial f_i(x_i), \quad i = 1, \ldots, N.$$

We refer to (1.3) and (1.4) as the primal feasibility and dual feasibility conditions, and x and λ as the primal variable and dual variable, respectively. Together, these conditions are sufficient for optimality; they become necessary as well when Slater's constraint qualification is satisfied, i.e., $\mathbf{relint}\,\mathbf{dom}\, f \cap \{x : Ax = b\} \neq \emptyset$.

Proximal oracle. Methods for solving (1.2) vary depending on what oracle is available for f_i. If f_i and its subgradient can be queried directly, a variety of iterative algorithms may be used [11, 34, 33]. However, in our setting, we assume that each f_i can only be accessed through its proximal operator $\mathbf{prox}_{tf_i} : \mathbf{R}^{n_i} \to \mathbf{R}^{n_i}$, defined as

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

where $t > 0$ is a parameter. In particular, we assume neither direct access to the function f_i nor its subdifferential ∂f_i. The separability of f implies that [39]

$$\mathbf{prox}_{tf}(v) = \left(\mathbf{prox}_{tf_1}(v_1), \ldots, \mathbf{prox}_{tf_N}(v_N)\right)$$

for any $v = (v_1, \ldots, v_N) \in \mathbf{R}^n$.

While we cannot evaluate ∂f_i at a general point, we can find an element of ∂f_i at the proximal operator's image point:

$$x_i = \mathbf{prox}_{tf_i}(v_i) \iff 0 \in \partial f_i(x_i) + \tfrac{1}{t}(x_i - v_i) \iff \tfrac{1}{t}(v_i - x_i) \in \partial f_i(x_i).$$

Thus, by querying the proximal oracle of f_i at v_i, we obtain an element in the subgradient of f_i at $x_i = \mathbf{prox}_{tf_i}(v_i)$.

The optimality conditions can be expressed using the proximal operator as well. The point $x \in \mathbf{R}^n$ is a solution to (1.2) if there exist $v \in \mathbf{R}^n$ and $\lambda \in \mathbf{R}^m$ such that

$$Ax = b, \tag{1.5}$$

$$0 = \tfrac{1}{t}(v - x) + A^T\lambda, \quad x_i = \mathbf{prox}_{tf_i}(v_i), \quad i = 1, \ldots, N. \tag{1.6}$$

Residuals. From conditions (1.5) and (1.6), we define the primal and dual residuals at x, λ as

$$r_{\mathrm{prim}} = Ax - b, \tag{1.7}$$

$$r_{\mathrm{dual}} = \tfrac{1}{t}(v - x) + A^T\lambda, \tag{1.8}$$

and we define the overall residual as $r = (r_{\mathrm{prim}}, r_{\mathrm{dual}}) \in \mathbf{R}^{n+m}$.

Stopping criterion. If problem (1.2) is feasible and bounded, a reasonable stopping criterion is that the residual norm lies below some threshold, i.e., $\|r\|_2 \leq \epsilon_{\mathrm{tol}}$, where $\epsilon_{\mathrm{tol}} > 0$ is a user-specified tolerance. We refer to the associated x as an approximate solution to (1.2). We defer discussion of the criteria for pathological (infeasible/unbounded) cases to section 4.

Notice that given a candidate $v \in \mathbf{R}^n$, we can readily choose the primal point $x = \mathbf{prox}_{tf}(v)$ and dual point

$$\lambda = \tfrac{1}{t}(A^\dagger)^T(x - v) \in \mathrm{argmin}_{\hat{\lambda}} \ \|A^T\hat{\lambda} - \tfrac{1}{t}(x - v)\|_2, \tag{1.9}$$

a minimizer of the dual residual norm, where $A^\dagger$ denotes the pseudoinverse of A. Thus, any algorithm for solving (1.2) via the proximal oracle need only determine a v that produces a small residual norm.

1.2. Related work. When functional access is restricted to a proximal oracle, the most common approaches for solving (1.2) are the alternating direction method of multipliers (ADMM) [56, 38, 18, 6], Douglas–Rachford splitting (DRS) [22], and the augmented Lagrangian method [64] with appropriate problem reformulations (e.g., consensus). These algorithms take advantage of the separability of the objective function, making them well-suited for the nonsmooth convex optimization problem considered in this paper. Yet despite their robustness and scalability, they typically suffer from slow convergence. Researchers have proposed several acceleration techniques, including adaptive penalty parameters [21, 60], adaptive synchronization [9], and momentum methods [63]. In practice, improvement from these techniques is usually limited due to the first-order nature of the accelerated algorithms. Special cases of (1.2) can sometimes yield exploitable problem forms, such as the Laplacian regularized stratified model in [54]. There the authors use the structure of the Laplacian matrix to efficiently parallelize ADMM. However, for the general problem, further acceleration requires a quasi-Newton method with line search [51] or semismooth Newton method with access to the Clarke's generalized Jacobian of the objective's proximal operator [4, 59, 31], both of which typically impose high per-iteration costs and memory requirements.

The acceleration technique adopted in this paper, type-II Anderson acceleration (AA), dates back to the 1960s [5]. It belongs to the family of sequence acceleration methods, which achieve faster convergence through certain sequence transformations. The origin of these methods can be traced to Euler's transformation of series [1] from the 18th century. Several faster sequence acceleration techniques were proposed in the 20th century, including Aitken's Δ^2-process in 1926 [3] along with its higher-order [47, 57] and vector [30, 28, 48] extensions, of which AA is a member. We refer readers to [12, 7] for a thorough history. AA can be viewed as either an extrapolation method or a generalized quasi-Newton method [17]. However, unlike classical quasi-Newton methods, it is effective without a line search and requires less computation and memory per iteration so long as certain stabilization measures are adopted.

Type-II AA was initially proposed to accelerate solvers for nonlinear integral equations in computational chemistry and materials science; later, it was applied to general fixed-point problems [55]. It operates by using an affine combination of previous iterates to determine the next iterate of an algorithm, where the combination's coefficients are obtained by solving an unconstrained least squares problem. In this sense, it is a generalization of the averaged iteration algorithm and Nesterov's accelerated gradient method. Its local convergence properties have been analyzed in a range of settings, both deterministic [42, 53, 46, 16, 25, 29, 41] and stochastic [45, 52], but its global convergence properties remain largely unknown except for a variant called EDIIS [24]. EDIIS has been shown to converge globally assuming that the fixed-point mapping is contractive [13]. However, it adds nonnegativity constraints to the coefficients of AA, meaning each iteration must solve a nonnegative least squares problem, a more complex task than solving the unconstrained problem, which admits a closed-form solution. The technique proposed in this paper, by contrast, only requires nonexpansiveness for global convergence. Each of its iterations merely solves an unconstrained least squares problem, similar to the original type-II AA. Recently, [17] proposed another AA variant called type-I AA. While less stable than its type-II counterpart, this variant performs more favorably with appropriate stabilization and globalization [36, 61].

AA has been applied in the literature to several problems related to (1.2). The authors of [40] use AA to speed up a parallelized local-global solver for geometry

optimization and physics simulation problems, which may be viewed as a special case of our problem where f_i are projection operators. In a separate setting, [27] employs AA to solve large-scale fixed-point problems arising from partial differential equations, demonstrating performance improvements on a distributed memory platform. More generally, [61] uses type-I AA in conjunction with DRS and a splitting conic solver (SCS) [35] to solve problems in consensus and conic optimization. These results are extended by [49], which combines type-II AA with an SCS variant to produce SuperSCS, an efficient solver for large cone programs. AA has also seen success in nonconvex settings. Notably, [62] applies AA to ADMM and studies its empirical performance on nonconvex optimization problems arising in computer graphics.

1.3. Contribution. In this paper, we consider the DRS algorithm for solving (1.2), which satisfies the proximal oracle assumption and admits a simple fixed-point (FP) formulation [43]. This FP format allows us to improve the convergence of DRS with AA, a memory efficient, line search free acceleration method that works on generic nonsmooth, nonexpansive FP mappings with almost no extra cost per iteration [61]. Motivated by the need for solver stability, we choose type-II AA in our current work and propose a robust stabilization scheme that maintains its speed and efficiency. We then apply it to DRS and show that the resulting Anderson accelerated Douglas–Rachford splitting (A2DR) algorithm always either converges or provides a certificate of infeasibility/unboundedness under very relaxed conditions. As a consequence, we obtain the first globally convergent type-II AA variant in nonsmooth, potentially pathological settings. Our convergence analysis only requires nonexpansiveness of the FP mapping, gracefully handling cases when a fixed-point does not exist. Finally, we release an open-source Python solver based on A2DR at

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

Outline. We begin in section 2 by introducing the basics of DRS. We then describe AA and propose A2DR in section 3. The global convergence properties of A2DR are established in section 4, along with an analysis of the infeasible and unbounded cases. We discuss the presolve, equilibration, and hyperparameter choices in section 5, followed by the implementation details in section 6. In section 7, we demonstrate the performance of A2DR on several examples. We conclude in section 8.

2. Douglas–Rachford splitting. Douglas–Rachford splitting (DRS) is an algorithm for solving problems of the form

$$\text{minimize} \quad g(x) + h(x)$$

with variable x, where g and h are CCP [43]. We can write problem (1.2) in this form by taking $g = f$ and $h = \mathcal{I}_{\{x\,:\,Ax=b\}}$, the indicator function of the linear equality constraint. Notice that $\mathbf{prox}_{th}$ is the projection onto the associated subspace, defined as

$$\Pi(v^{k+1/2}) = v^{k+1/2} - A^\dagger(Av^{k+1/2} - b) = v^{k+1/2} - A^T(AA^T)^\dagger(Av^{k+1/2} - b).$$

The DRS algorithm proceeds as follows.

Each iteration k requires the evaluation of the proximal operator of f and the projection onto a linear subspace.

Dual variable and residuals. We regard $x^{k+1/2}$, the proximal operator's image point, as our approximate primal optimal variable in iteration k. There are two ways to produce an approximate dual variable λ^k. The first way sets

$$\lambda^k = \tfrac{1}{t}(AA^T)^\dagger(Av^{k+1/2} - b),$$

Algorithm 2.1 Douglas–Rachford Splitting (DRS)

Input: initial point v^0, penalty coefficient $t > 0$.
for $k = 1, 2, \ldots$ **do**
  $x^{k+1/2} = \mathbf{prox}_{tf}(v^k)$
  $v^{k+1/2} = 2x^{k+1/2} - v^k$
  $x^{k+1} = \Pi(v^{k+1/2})$
  $v^{k+1} = v^k + x^{k+1} - x^{k+1/2}$
end for

an intermediate value from the projection step. (See Remark SM1.2 in the supplementary materials for the reasoning behind this choice.) The second way computes λ^k as the minimizer of $\|r^k_{\text{dual}}\|_2$, which necessitates solving the least squares problem (1.9) at each iteration. Our implementation uses the second method because the additional computational cost is minimal, and this choice of a dual optimal variable results in earlier stopping.

The primal and dual residuals can be calculated by plugging our DRS iterates into (1.7) and (1.8):

$$r^k_{\text{prim}} = Ax^{k+1/2} - b, \tag{2.1}$$

$$r^k_{\text{dual}} = \tfrac{1}{t}(v^k - x^{k+1/2}) + A^T\lambda^k. \tag{2.2}$$

Convergence. Define the fixed-point mapping $F_{\text{DRS}} : \mathbf{R}^n \to \mathbf{R}^n$ as

$$F_{\text{DRS}}(v) = v + \Pi\left(2\mathbf{prox}_{tf}(v) - v\right) - \mathbf{prox}_{tf}(v),$$

so that $v^{k+1} = F_{\text{DRS}}(v^k)$. It can be shown that F_{DRS} is 1/2-averaged (i.e., $F_{\text{DRS}} = \frac{1}{2}H + \frac{1}{2}I$, where H is nonexpansive and I is the identity mapping), and hence, v^k converges globally and sublinearly to a fixed-point of F_{DRS} whenever such a point exists. In this case, $x^{k+1/2}$ and x^{k+1} both converge to a solution of (1.2), implying that $\lim_{k\to\infty} \|r^k_{\text{prim}}\|_2 = \lim_{k\to\infty} \|r^k_{\text{dual}}\|_2 = 0$ [43].

3. Anderson accelerated DRS.

In this section, we give a brief overview of AA and propose a modification that improves its stability. We then combine stabilized AA with DRS to construct our main algorithm, Anderson accelerated DRS. A2DR always produces an approximate solution to (1.2) when the problem is feasible and bounded. We treat the infeasible/unbounded cases in section 4.

3.1. Anderson acceleration.

Consider a 1/2-averaged mapping $F : \mathbf{R}^n \to \mathbf{R}^n$. To solve the associated fixed-point problem $F(v) = v$, we can repeatedly apply the fixed-point iteration (FPI) $v^{k+1} = F(v^k)$, which is exactly DRS when $F = F_{\text{DRS}}$. However, convergence of FPI algorithms is usually slow in practice. Acceleration schemes are one way of addressing this flaw. AA is a special form of the generalized limited-memory quasi-Newton (LM-QN) method. It is one of the most successful acceleration schemes for general nonsmooth FPIs, exhibiting greater memory efficiency than classical LM-QN algorithms like the restarted Broyden's method [49].

We focus here on the original type-II AA [5]. Let $G(v) = v - F(v)$ be the residual function and $M^k \in \mathbf{Z}_+$ a nonnegative integer denoting the memory size. Typically,

$M^k = \min(M_{\max}, k)$ for some maximum memory $M_{\max} \geq 1$ [55]. At iteration k, type-II AA stores in memory the most recent $M^k + 1$ iterates $(v^k, \ldots, v^{k-M^k})$ and replaces $v^{k+1} = F(v^k)$ with $v^{k+1} = \sum_{j=0}^{M^k} \alpha_j^k F(v^{k-M^k+j})$, where $\alpha^k = (\alpha_0^k, \ldots, \alpha_{M^k}^k)$ is the solution to

$$\begin{array}{ll} \text{minimize} & \| \sum_{j=0}^{M^k} \alpha_j^k G(v^{k-M^k+j}) \|_2^2 \\ \text{subject to} & \sum_{j=0}^{M^k} \alpha_j^k = 1. \end{array} \tag{3.1}$$

AA then updates its memory to $(v^{k+1}, \ldots, v^{k+1-M^{k+1}})$ before repeating the process.

The accelerated v^{k+1} can be seen as an extrapolation from the original v^{k+1} and the fixed-point mappings of a few earlier iterates. It has the potential to reduce the residual by a significant amount. In particular, when F is affine, (3.1) seeks an affine combination $\tilde{v}^{k+1}$ of the last $M^k + 1$ iterates that minimizes the residual norm $\|G(\tilde{v}^{k+1})\|_2$, then computes $v^{k+1} = F(\tilde{v}^{k+1})$ by performing an additional FPI.

3.2. Main algorithm. Despite the popularity of type-II AA, it suffers from instability in its original form [46]. We propose a stabilized variant using adaptive regularization and a simple safeguarding globalization trick.

Adaptive regularization. Define $g^k = G(v^k)$, $y^k = g^{k+1} - g^k$, $s^k = v^{k+1} - v^k$, $Y_k = [y^{k-M^k} \ \cdots \ y^{k-1}]$, and $S_k = [s^{k-M^k} \ \cdots \ s^{k-1}]$. With a change of variables, (3.1) can be rewritten as [55]

$$\text{minimize} \quad \|g^k - Y_k \gamma^k\|_2 \tag{3.2}$$

with respect to $\gamma^k = (\gamma_0^k, \ldots, \gamma_{M^k-1}^k)$, where

$$\alpha_0^k = \gamma_0^k, \quad \alpha_i^k = \gamma_i^k - \gamma_{i-1}^k, \quad i = 1, \ldots, M^k - 1, \quad \alpha_{M^k}^k = 1 - \gamma_{M^k-1}^k. \tag{3.3}$$

To improve stability, we add an ℓ_2-regularization term to (3.2), scaled by the Frobenius norms of S_k and Y_k, which yields the problem

$$\text{minimize} \quad \|g^k - Y_k \gamma^k\|_2^2 + \eta \left(\|S_k\|_F^2 + \|Y_k\|_F^2 \right) \|\gamma^k\|_2^2, \tag{3.4}$$

where $\eta > 0$ is a parameter. The regularization adopted in (3.4) differs from the one introduced in [46] that directly regularizes α^k. We argue that with the affine constraint on α^k, it is more natural to regularize the unconstrained variables γ^k. This approach also allows us to establish global convergence in section 4. Intuitively, if the algorithm is converging, $\lim_{k\to\infty} \|S_k\|_F = \lim_{k\to\infty} \|Y_k\|_F = 0$, so the coefficient on the regularization term vanishes just like in the single iteration local analysis by [46].

A simple and relaxed safeguard. To achieve global convergence, we also need a safeguarding step. This step checks whether the current residual norm is sufficiently small. If true, the algorithm takes the AA update and skips the safeguarding check for the next $R - 1$ iterations. Otherwise, the algorithm replaces the AA update with the vanilla FPI update. Here $R \in \mathbf{Z}_{++}$ is a positive integer that determines the degree of safeguarding; smaller values are more conservative, since the safeguarding step is performed more often.

A2DR. We are finally ready to present A2DR (Algorithm 3.1). A2DR applies type-II AA with adaptive regularization (lines 10–11) and safeguarding (lines 13–17) to the DRS fixed-point mapping F_{DRS}. In our description, $G_{\text{DRS}} = I - F_{\text{DRS}}$ is the residual mapping, $D > 0$, $\epsilon > 0$ are constants that characterize the degree of safeguarding, and n_{AA}^k is the number of times the *AA* candidate has passed the safeguarding check up to iteration k.

Stopping criterion. As explained in section 1, to check optimality, we evaluate the primal and dual residuals r^k_{prim} and r^k_{dual}. We terminate the algorithm and output $x^{k+1/2}$ as the approximate solution if

$$\|r^k\|_2 \le \epsilon_{\text{tol}} = \epsilon_{\text{abs}} + \epsilon_{\text{rel}}\|r^0\|_2, \tag{3.5}$$

where $r^k = (r^k_{\text{prim}}, r^k_{\text{dual}})$ and $\epsilon_{\text{abs}} > 0, \epsilon_{\text{rel}} > 0$ are user-specified absolute and relative tolerances, respectively.

Algorithm 3.1 Anderson Accelerated Douglas–Rachford Splitting (A2DR)

Input: initial point v^0, penalty coefficient $t > 0$, regularization coefficient $\eta > 0$, safeguarding constants $D > 0,\ \epsilon > 0,\ R \in \mathbf{Z}_{++}$, max-memory $M_{\max} \in \mathbf{Z}_+$.
Initialize $n_{\text{AA}} = 0,\ R_{\text{AA}} = 0,\ I_{\text{safeguard}} = \texttt{True}$.
Compute $v^1 = F_{\text{DRS}}(v^0)$, $g^0 = v^0 - v^1$.
for $k = 1, 2, \ldots$ **do**
    `# Memory update`
    Choose memory $M^k = \min(M_{\max}, k)$.
    Compute the DRS candidate: $v^{k+1}_{\text{DRS}} = F_{\text{DRS}}(v^k)$, $g^k = v^k - v^{k+1}_{\text{DRS}}$.
    Update Y_k and S_k with $y^{k-1} = g^k - g^{k-1}$ and $s^{k-1} = v^k - v^{k-1}$.
    `# Adaptive regularization`
    Solve for γ^k in regularized least squares (3.4) and compute weights α^k from (3.3).
    Compute the AA candidate: $v^{k+1}_{\text{AA}} = \sum_{j=0}^{M^k} \alpha^k_j v^{k-M^k+j+1}_{\text{DRS}}$.
    `# Safeguard`
    If $I_{\text{safeguard}}$ is `True` or $R_{\text{AA}} \ge R$:
        If $\|g^k\|_2 = \|G_{\text{DRS}}(v^k)\|_2 \le D\|g^0\|_2(n_{\text{AA}}/R + 1)^{-(1+\epsilon)}$:
            $v^{k+1} = v^{k+1}_{\text{AA}}$, $n_{\text{AA}} = n_{\text{AA}} + 1$, $I_{\text{safeguard}} = \texttt{False}$, $R_{\text{AA}} = 1$.
        else $v^{k+1} = v^{k+1}_{\text{DRS}}$, $R_{\text{AA}} = 0$.
    else $v^{k+1} = v^{k+1}_{\text{AA}}$, $n_{\text{AA}} = n_{\text{AA}} + 1$, $R_{\text{AA}} = R_{\text{AA}} + 1$.
    Terminate and output $x^{k+1/2}$ (cf. Algorithm 2.1) if stopping criterion (3.5) is satisfied.
end for

4. Global convergence.

We now establish the global convergence properties of A2DR. In particular, we show that under the general assumptions in section 1, A2DR either converges globally from any initial point or provides a certificate of infeasibility/unboundedness.

4.1. Infeasibility and unboundedness.

When the optimality conditions do not hold even in the asymptotic sense, i.e., if the infimum of the primal or dual residual over all possible x and v is nonzero, problem (1.2) is either infeasible or unbounded. We say that (1.2) is *infeasible* if $\mathbf{dom}\, f \cap \{x : Ax = b\} = \emptyset$, and we say that it is *unbounded* if (1.2) is feasible, but $\inf_{Ax=b} f(x) = -\infty$. The following proposition characterizes sufficient certificates of infeasibility and unboundedness.

PROPOSITION 4.1 (certificates of infeasibility and unboundedness). *Let* $f^* : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ *denote the conjugate function of* f*, defined as* $f^*(y) = \sup_{x \in \mathbf{dom}\, f} \left(y^T x - f(x)\right)$.

(i) *If* $\mathbf{dist}(\mathbf{dom}\, f, \{x : Ax = b\}) > 0$*, then problem* (1.2) *is infeasible.*

(ii) *If* $\mathbf{dist}(\mathbf{dom}\, f^*, \mathbf{range}(A^T)) > 0$*, then problem* (1.2) *is unbounded.*

When (i) holds, (1.2) is also called *(primal) strongly infeasible*, and when (ii) holds, (1.2) is called *dual strongly infeasible* [26]. We say that (1.2) is *pathological* if it is either primal or dual strongly infeasible, and *solvable* otherwise. Notice that when the problem is pathological, it is either infeasible or unbounded, but not both.

Proof. Claim (i) is true by definition. To prove claim (ii), observe that the dual problem of (1.2) is minimize$_\nu$ $f^*(\nu) + g^*(-\nu)$, where $g^*(\nu) = b^T\lambda$ when $\nu = A^T\lambda$, and $g^*(\nu) = +\infty$ otherwise. By Lemma 1 in [44], if $\mathbf{dist}(\mathbf{dom}\, f^*, \mathbf{range}(A^T)) > 0$, then the dual problem is strongly infeasible, and hence the primal problem (1.2) is unbounded. ∎

If (1.2) is pathological, an algorithm should provide a certificate of either (i) or (ii). We will show that A2DR achieves this goal by returning the distances in (i) and (ii) as a by-product of its iterations.

4.2. Convergence results. We are now ready to present the convergence results for A2DR. We begin by highlighting the contribution of adaptive regularization to the stabilization of AA. Indeed, by setting the gradient of the objective function in (3.4) to zero, we find the solution is

$$\gamma^k = (Y_k^T Y_k + \eta \left(\|S_k\|_F^2 + \|Y_k\|_F^2\right) I)^{-1} Y_k^T g^k.$$

Using the relationship between α^k and γ^k, we then write

$$v^{k+1} = v^k - (I + (S_k - Y_k)(Y_k^T Y_k + \eta \left(\|S_k\|_F^2 + \|Y_k\|_F^2\right) I)^{-1} Y_k^T) g^k = v^k - H_k g^k, \tag{4.1}$$

where $H_k = I + (S_k - Y_k)(Y_k^T Y_k + \eta \left(\|S_k\|_F^2 + \|Y_k\|_F^2\right) I)^{-1} Y_k^T$.

LEMMA 4.2. *The matrices* H_k $(k \geq 0)$ *satisfy* $\|H_k\|_2 \leq 1 + 2/\eta$.

Proof. Since $\|A\|_2 \leq \|A\|_F$ for any matrix A,

$$\begin{aligned}\|H_k\|_2 &\leq 1 + \frac{\|S_k - Y_k\|_2 \|Y_k\|_2}{\eta\left(\|S_k\|_F^2 + \|Y_k\|_F^2\right)} \leq 1 + \frac{\|S_k - Y_k\|_F \|Y_k\|_F}{\eta\left(\|S_k\|_F^2 + \|Y_k\|_F^2\right)} \\ &\leq 1 + \frac{\|S_k\|_F \|Y_k\|_F + \|Y_k\|_F^2}{\eta\left(\|S_k\|_F^2 + \|Y_k\|_F^2\right)} \leq 1 + 2/\eta.\end{aligned}$$

This completes the proof. ∎

The above lemma characterizes the stability ensured by regularization in (3.4), providing a stepping stone to our global convergence theorems.

4.2.1. Solvable case.

THEOREM 4.3. *Suppose that problem* (1.2) *is solvable. Then for any initialization* v^0 *and any hyperparameters* $\eta > 0,\ D > 0,\ \epsilon > 0,\ R \in \mathbf{Z}_{++},\ M_{\max} \in \mathbf{Z}_+$*, we have*

$$\liminf_{k \to \infty} \|r^k\|_2 = 0, \tag{4.2}$$

and the AA candidates are adopted infinitely often. Additionally, if F_{DRS} *has a fixed-point,* v^k *converges to a fixed-point of* F_{DRS} *and* $x^{k+1/2}$ *converges to a solution of* (1.2) *as* $k \to \infty$.

The proof is left to the supplementary materials. A direct corollary of Theorem 4.3 is that the primal and dual residuals of $x^{k+1/2}$ converge to zero so long as (1.2) is feasible and bounded. Even if (1.2) does not have a solution, A2DR still produces a sequence of asymptotically optimal points provided that (1.2) is not pathological. Thus, Algorithm 3.1 always terminates in a finite number of steps in these cases.

In practice, the proximal operators and projections are often evaluated with error, so lines 3 and 5 in Algorithm 2.1 become $\hat{x}^{k+1/2} = \mathbf{prox}_{tf}(v^k) + \zeta_1^k$ and $\hat{x}^{k+1} = \Pi(v^{k+1/2}) + \zeta_2^k$, where $\zeta_1^k,\ \zeta_2^k \in \mathbf{R}^n$ represent numerical errors. We use $\hat{x}^{k+1/2}$, $\hat{v}^{k+1/2}$, $\hat{x}^{k+1}$ to denote the error-corrupted intermediate F_{DRS} iterates, and $x^{k+1/2}$, $v^{k+1/2}$, x^{k+1} to denote the error-free intermediate F_{DRS} iterates. However, we still use the old notation (e.g., v^k and g^k) to denote the error-corrupted A2DR iterates in the body of Algorithm 3.1. For cases with such errors, we have the following convergence result.

THEOREM 4.4. *Suppose that problem* (1.2) *is solvable, but the* F_{DRS} *iterates are evaluated with errors* $\zeta_1^k,\ \zeta_2^k \in \mathbf{R}^n$. *Assume that* F_{DRS} *has a fixed-point and* $\exists\ \epsilon' > 0$ *such that* $\|\zeta_1^k\|_2 \le \epsilon'$ *and* $\|\zeta_2^k\|_2 \le \epsilon'$ *for all* $k \ge 0$. *Then for any initialization* v^0 *and any hyperparameters* $\eta > 0,\ D > 0,\ \epsilon > 0,\ R \in \mathbf{Z}_{++},\ M_{\max} \in \mathbf{Z}_+$, *if all* v^k *and some fixed-point* $v^\star$ *of* F_{DRS} *are uniformly bounded,* i.e., $\|v^k\|_2 \le L$ *and* $\|v^\star\|_2 \le L$ *for a constant* $L > 0$, *we have*

$$\liminf_{k\to\infty} \|r_{\mathrm{prim}}^k\|_2 \le \|A\|_2(4\epsilon' + 4\sqrt{L\epsilon'}), \quad \liminf_{k\to\infty} \|r_{\mathrm{dual}}^k\|_2 \le \tfrac{1}{t}(4\epsilon' + 4\sqrt{L\epsilon'}). \tag{4.3}$$

The residuals are computed by plugging v^k *(as output by A2DR) and the error-free intermediate iterates* $x^{k+1/2} = \boldsymbol{prox}_{tf}(v^k)$ *into* (2.1) *and* (2.2).

4.2.2. Pathological case.

THEOREM 4.5. *Suppose that problem* (1.2) *is pathological. Then for any initialization* v^0 *and any hyperparameters* $\eta > 0,\ D > 0,\ \epsilon > 0,\ R \in \mathbf{Z}_{++},\ M_{\max} \in \mathbf{Z}_+$, *the difference* $v^k - v^{k+1}$ *converges to some nonzero vector* $\delta v \in \mathbf{R}^n$. *If, furthermore,* $\lim_{k\to\infty} Ax^{k+1/2} = b$, *then* (1.2) *is unbounded, in which case* $\|\delta v\|_2 = t\,\mathbf{dist}(\mathbf{dom}\, f^*, \mathbf{range}(A^T))$. *Otherwise,* (1.2) *is infeasible and* $\|\delta v\|_2 \ge \mathbf{dist}(\mathbf{dom}\, f, \{x : Ax = b\})$ *with equality when the dual problem is feasible.*

The proof is given in the supplementary materials. Theorem 4.5 states that in pathological cases, the successive differences $\delta v^k = v^k - v^{k+1}$ can be used as certificates of infeasibility and unboundedness. We leave the practical design and implementation of these certificates to a future version of A2DR.

The same global convergence results (Theorems 4.3–4.5) can be shown for stabilized type-I AA [61], which sometimes exhibited better numerical performance in our early experiments. However, type-I AA introduces additional hyperparameters, and to ensure our solver is robust without the need for extra hyperparameter tuning, we restrict ourselves to type-II AA. We leave type-I Anderson accelerated DRS to a future paper.

5. Presolve, equilibration, and parameter selection. In this section, we introduce a few tricks that make A2DR more efficient in practice.

Infeasible linear constraints. In section 1, we assumed that the linear constraint $Ax = b$ is feasible. However, this assumption may be violated in practice. To address this issue, we first solve the least squares problem associated with the linear system. If the resulting residual is sufficiently small, we proceed to solve (1.2) using A2DR. Otherwise, we terminate and return a certificate of infeasibility.

Preconditioning. To precondition the problem, we scale the variables x_i and the linear constraints (rows of $Ax = b$), solve the problem with the scaled variables and data, then unscale to recover the original variables. Scaling the variables and constraints does not change the theoretical convergence, but can improve the practical convergence if the scaling factors are chosen well. A popular heuristic for improving the practical convergence is to choose the scalings to minimize, or at least reduce, the condition number of the coefficient matrix. In turn, a heuristic for reducing the condition number of the coefficient matrix is to equilibrate it, i.e., choose the scalings so that all rows have approximately equal norm and all columns have approximately equal norm. The regularized Sinkhorn–Knopp method described below does this, where the regularization allows it to gracefully handle matrices that cannot be equilibrated or would require very extreme scaling to equilibrate.

The details are as follows. First, we equilibrate A by choosing diagonal matrices $D = \mathbf{diag}(d_1, \ldots, d_m)$ and $E = \mathbf{diag}(e_1 I_{n_1}, \ldots, e_N I_{n_N})$, with $d_1 > 0, \ldots, d_m > 0$ and $e_1 > 0, \ldots, e_N > 0$, and forming the scaled matrix $\hat{A} = DAE$. The scaled problem is

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^N \hat{f}_i(\hat{x}_i) \\ \text{subject to} & \sum_{i=1}^N \hat{A}_i \hat{x}_i = \hat{b}, \end{array} \tag{5.1}$$

where

$$\hat{f}_i(\hat{x}_i) = f_i(e_i \hat{x}_i), \quad \hat{A} = D[A_1 \ A_2 \ \cdots \ A_N]E, \quad \hat{b} = Db.$$

We apply A2DR to (5.1) to obtain $\hat{x}^\star$ and recover the approximate solution to our original problem (1.2) via $x^\star = E\hat{x}^\star$.

To determine the scaling factors d_i and e_j, we use the regularized Sinkhorn–Knopp method [18]. First, we perform a change of variables to $u_i = 2\log(d_i)$ and $v_j = 2\log(e_j)$. Then we solve the optimization problem

$$\text{minimize} \quad \sum_{i=1}^m \sum_{j=1}^N B_{ij} e^{u_i + v_j} - N\mathbf{1}^T u - m\mathbf{1}^T v + \gamma \left(N \sum_{i=1}^m e^{u_i} + m \sum_{j=1}^N e^{v_j} \right) \tag{5.2}$$

for $u \in \mathbf{R}^m$ and $v \in \mathbf{R}^N$, where $B_{ij} = \sum_{l=n_1+\cdots+n_{j-1}+1}^{n_1+\cdots+n_j} A_{il}^2$ and $\gamma > 0$ is a regularization parameter. This problem is strictly convex. At its solution, the arithmetic means of the recovered scaling factors are equal. In our implementation, we set

$$\gamma = \frac{m+N}{mN} \sqrt{\epsilon^{\mathrm{mp}}},$$

where ϵ^{mp} is the machine precision. Notice that when $\gamma = 0$ and (5.2) has a solution, the resulting $\hat{A}$ is equilibrated exactly, i.e., the rows all have the same ℓ_2 norm, and the columns all have the same ℓ_2 norm in the blockwise sense (with block sizes $n_1, \ldots, n_N$).

We use coordinate descent to solve (5.2), which produces [18, Algorithm 2]. This algorithm typically returns a solution $\tilde{u}, \tilde{v}$ in only a handful of iterations. We then recover $\tilde{d}_i = e^{\tilde{u}_i/2}$ and $\tilde{e}_j = e^{\tilde{v}_j/2}$. Define $\tilde{D} = \mathbf{diag}(\tilde{d}_1, \ldots, \tilde{d}_m)$ and $\tilde{E} = \mathbf{diag}(\tilde{e}_1 I_{n_1}, \ldots, \tilde{e}_N I_{n_N})$. Although the arithmetic means of $(\tilde{d}_1, \ldots, \tilde{d}_m)$ and $(\tilde{e}_1, \ldots, \tilde{e}_N)$ are already equal, we also wish to enforce equality of their geometric means, which corresponds to equality of the arithmetic means of the problem variables. This leads to better performance in practice. Accordingly, we scale $\tilde{D}$ and $\tilde{E}$ to obtain D and E such that the geometric mean of $(d_1, \ldots, d_m)$ equals that of $(e_1, \ldots, e_N)$ and $\|DAE\|_F = \sqrt{\min(m, N)}$.

Since E is constant within each variable block, the proximal operator of $\hat{f}_i$ can be evaluated using the proximal operator of f_i via

$$\begin{aligned}\hat{x}_i &= \mathbf{prox}_{t\hat{f}_i}(\hat{v}_i) = \operatorname{argmin}_{\hat{x}_i}\ \left(f_i(e_i\hat{x}_i) + \tfrac{1}{2t}\|\hat{x}_i - \hat{v}_i\|_2^2\right) \\ &= \tfrac{1}{e_i}\operatorname{argmin}_{x_i}\ \left(f_i(x_i) + \tfrac{1}{2t}\|x_i/e_i - \hat{v}_i\|_2^2\right) \\ &= \tfrac{1}{e_i}\mathbf{prox}_{e_i^2 t f_i}(e_i\hat{v}_i).\end{aligned} \tag{5.3}$$

All other steps of A2DR (including the projection step in Algorithm 2.1, line 5) remain the same, except with A and b replaced by $\hat{A}$ and $\hat{b}$. We check the stopping criterion directly on (5.1), trusting that our equilibration scheme provides an appropriate scaling of the original problem. An alternative is to check the stopping criterion on (1.2) using the unscaled variables.

Choice of t. With equilibration, the choice of parameter

$$t = \frac{1}{10}\left(\prod_{j=1}^{N} e_j\right)^{-2/N}$$

works well across a wide variety of problems. (Recall that convergence is guaranteed in theory for any $t > 0$.) Our implementation uses this choice of t.

The intuition behind our choice is as follows. Consider the case of $f_i(x_i) = x_i^T Q_i x_i$ with $Q_i \in \mathbf{S}_+^{n_i}$, the set of symmetric positive semidefinite matrices. The associated $\mathbf{prox}_{tf_i}(v_i) = (2tQ_i + I)^{-1}v_i$ is linear, and by (5.3),

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

To avoid ill-conditioning when e_i is an extreme value, we want to choose t such that $e_i^2 t = c > 0$, a constant for $i = 1, \ldots, N$. However, this is impossible unless $e_1, \ldots, e_N$ are all equal, so instead we minimize $\sum_{i=1}^{N}(\log t - \log(ce_i^{-2}))^2$, where we have taken logs because e_i is on the exponential scale as discussed in the previous section. For $c = \frac{1}{10}$, the solution is precisely our choice of t.

6. Implementation. We now describe the implementation details and user interface of our A2DR solver.

Least squares evaluation. There are three places in A2DR that require the solution of a least squares problem. First, to evaluate the F_{DRS} projection

$$\Pi(v^{k+1/2}) = v^{k+1/2} - A^{\dagger}(Av^{k+1/2} - b),$$

we solve

$$\text{minimize} \quad \|Ad - (Av^{k+1/2} - b)\|_2$$

with respect to $d \in \mathbf{R}^n$ to obtain $d^k = A^{\dagger}(Av^{k+1/2} - b)$. This is accomplished in our implementation with LSQR, a conjugate gradient (CG) method [37]. Specifically, we store A as a sparse matrix and call `scipy.sparse.linalg.lsqr` with warm start at each iteration. LSQR has low memory requirements and converges extremely fast on well-conditioned systems, making it ideal for the problems we typically encounter.

Second, to compute the approximate dual variable λ^k in (2.2), we minimize $\|r_{\mathrm{dual}}^k\|_2$. We use LSQR with a warm start for this as well.

Finally, to solve the regularized least squares problem (3.4), we offer two options: the first is again LSQR, and the second is `numpy.linalg.lstsq`, an SVD-based least squares solver. Our implementation defaults to the second choice. This direct method is more stable, and since Y_k is a tall matrix with very few columns, the SVD is relatively efficient to compute at each iteration.

Solver interface. The A2DR solver is called with the command

```
result = a2dr(p_list, A_list, b)
```

where `p_list` is the list of proximal operators of f_i, `A_list` is the list of A_i, and `b` is the vector b. The lists `p_list` and `A_list` must be given in the same order of $i = 1, \ldots, N$. Each element of `p_list` is a Python function, which takes as input a vector v and parameter $t > 0$ and outputs the proximal operator of f_i evaluated at (v, t). For example, if $N = 2$ with $f_1(x_1) = \|x_1\|_2^2$ and $f_2(x_2) = \mathcal{I}_{\mathbf{R}^n_+}(x_2)$,

```
p_list = [lambda v, t:  v/(1.0 + 2*t), lambda v, t:
                  numpy.maximum(v,0)]
```

is a valid implementation. The `result` is a Python dictionary comprised of the key/value pairs `x_vals`: a list of $x_1^{k^\star+1/2}, \ldots, x_N^{k^\star+1/2}$ from the iteration $k^\star$ with the smallest $\|r^{k^\star}\|_2$, `primal` and `dual`: arrays containing the residual norms $\|r^k_{\text{prim}}\|_2$ and $\|r^k_{\text{dual}}\|_2$, respectively, at each iteration k, `num_iters`: the total number of iterations, and `solve_time`: the algorithm runtime.

Arguments `A_list` and `b` are optional, and when omitted, the solver recognizes the problem as (1.2) without the constraint $Ax = b$. All other hyperparameters in Algorithm 3.1, the initial point v^0, as well as the choice of whether to use preconditioning and/or AA, are also optional. By default, both preconditioning and AA are enabled.

Last but not least, the distributed execution of the iteration steps, including the evaluation of the proximal operators and componentwise summation and subtraction, is implemented with the `multiprocessing` package in Python.

7. Numerical experiments. The following experiments were carried out on a Linux server with 64 8-core Intel Xeon E5-4620 / 2.20 GHz processors and 503 GB of RAM. We used the default A2DR solver parameters throughout. In particular, the AA max-memory $M_{\max} = 10$, regularization coefficient $\eta = 10^{-8}$, safeguarding constants $D = 10^6$, $\epsilon = 10^{-6}$, and $R = 10$, and initial $v^0 = 0$. We set the stopping tolerances to $\epsilon_{\text{abs}} = 10^{-6}$ and $\epsilon_{\text{rel}} = 10^{-8}$ and limited the maximum number of iterations to 1000 unless otherwise specified. All data were generated such that the problems are feasible and bounded, and hence convergence of the primal and dual residuals is guaranteed. While it is possible to improve convergence with additional parameter tuning, we emphasize that A2DR consistently outperforms DRS by a factor of three or more using the solver defaults. This performance gain is robust across all problem instances.

For each experiment, we plotted the residual norm $\|r^k\|_2$ at each iteration k for both A2DR and vanilla DRS. The plots against runtime are very similar since the AA overhead is less than 10% of the per-iteration cost, so we refrain from showing them here. We also compared the final objective value and constraint violations with the solution obtained by CVXPY [15, 2]. In all but a few problem instances, the results match within 10^{-4}. The results that differ are due to CVXPY's solver failure, which we discuss in more detail below.

7.1. Nonnegative least squares. The nonnegative least squares problem is

$$\begin{array}{ll} \text{minimize} & \|Fz - g\|_2^2 \\ \text{subject to} & z \geq 0, \end{array} \tag{7.1}$$

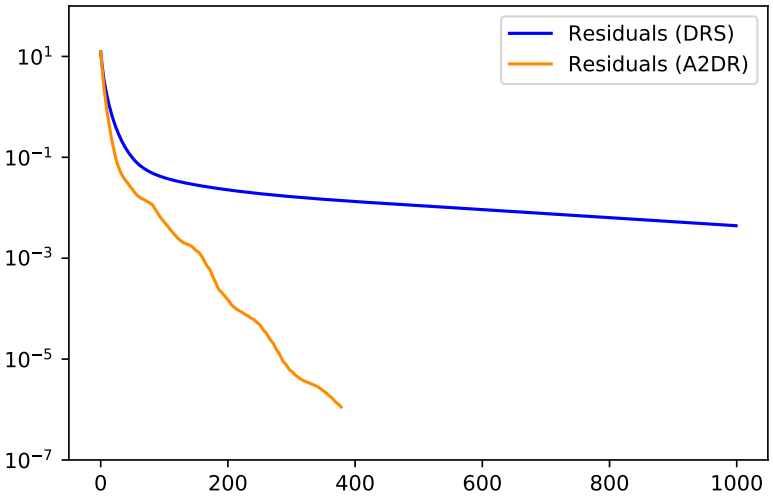

FIG. 1. *Nonnegative least squares: convergence of residual norms* $\|r^k\|_2$.

where $z \in \mathbf{R}^q$ is the variable, and $F \in \mathbf{R}^{p\times q}$ and $g \in \mathbf{R}^p$ are problem data. This problem may be rewritten in form (1.2) by letting

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

for $x_1, x_2 \in \mathbf{R}^q$ and enforcing the constraint $x_1 = x_2$ with $A_1 = I, A_2 = -I$, and $b = 0$. The proximal operators of f_1 and f_2 are

$$\begin{aligned} &\mathbf{prox}_{tf_1}(v) = \operatorname{argmin}_{x_1} \left\| \begin{bmatrix} F \\ \frac{1}{\sqrt{2t}} I \end{bmatrix} x_1 - \begin{bmatrix} g \\ \frac{1}{\sqrt{2t}} v \end{bmatrix} \right\|_2^2, \\ &\mathbf{prox}_{tf_2}(v) = (v)_+. \end{aligned} \tag{7.2}$$

We evaluate $\mathbf{prox}_{tf_1}$ using LSQR.

Problem instance. Let $p = 10000$ and $q = 8000$. We took F to be a sparse random matrix with 0.1% nonzero entries, which are drawn i.i.d. (independently and identically distributed) from $\mathcal{N}(0,1)$, and g to be a random vector from $\mathcal{N}(0, I)$.

The convergence results are shown in Figure 1. A2DR achieves $\|r^k\|_2 \le 10^{-6}$ in under 400 iterations, while DRS flattens out at $\|r^k\|_2 \approx 10^{-2}$ until the maximum number of iterations is reached. Our algorithm's speed is a notable improvement over other popular solvers. We solved the same problem using an operator splitting quadratic program (OSQP) solver [50] and SCS, which took, respectively, 349 and 327 seconds to return a solution with tolerance 10^{-6}. In contrast, A2DR converged in only 55 seconds and produced the smallest objective value up to a precision of 10^{-10}.

In a second experiment, we set $p = 300$ and $q = 500$ and compared the performance under adaptive regularization, as described in (3.4), with no regularization and constant regularization. Figure 2 shows that adaptive regularization results in better convergence. By 1000 iterations, the residual norm is nearly 10^{-6} in the adaptive case, while it is roughly 10^{-3} under the other two regularization schemes. Similar improvement arises in the examples below, but we have not included the plots for the sake of brevity.

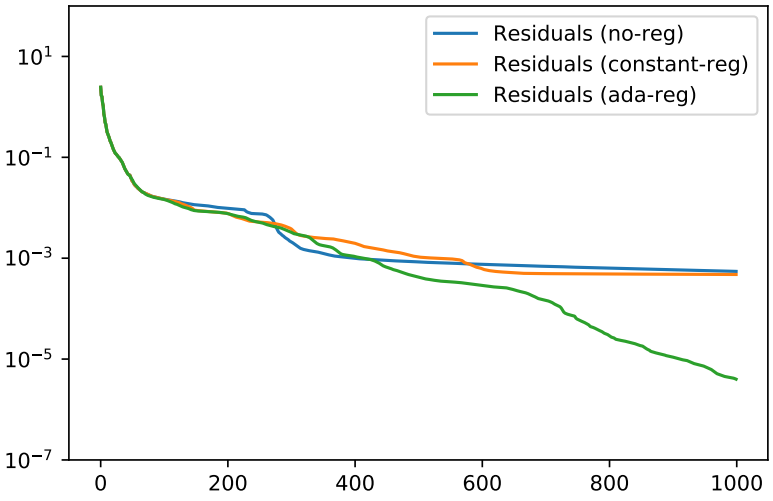

FIG. 2. *Nonnegative least squares: A2DR with no, constant, and adaptive regularization.*

7.2. Sparse inverse covariance estimation. Suppose that $z_1, \ldots, z_p$ are i.i.d. $N(0, \Sigma)$ with Σ^{-1} known to be sparse. We can estimate the covariance matrix $\Sigma \in \mathbf{S}_+^q$ by solving the optimization problem [19, 8]

$$\text{minimize} \quad -\log\det(S) + \operatorname{tr}(SQ) + \alpha\|S\|_1, \tag{7.3}$$

where $S \in \mathbf{S}^q$ (the set of symmetric matrices) is the variable, $Q = \frac{1}{p}\sum_{l=1}^p z_l z_l^T$ is the sample covariance, and $\alpha > 0$ is a hyperparameter. We then take $\hat{\Sigma} = S^{-1}$ as an estimate of Σ. Here $\|S\|_1$ is the elementwise ℓ_1 norm and $\log\det$ is understood to be an extended real-valued function, i.e., $\log\det(S) = -\infty$ whenever $S \not\succ 0$.

Let $x_i \in \mathbf{R}^{q(q+1)/2}$ be some vectorization of $S_i \in \mathbf{S}^q$ for $i = 1, 2$. Problem (7.3) can be represented in standard form (1.2) by setting

$$f_1(x_1) = -\log\det(S_1) + \operatorname{tr}(S_1 Q), \quad f_2(x_2) = \alpha\|S_2\|_1,$$

and $A_1 = I$, $A_2 = -I$, and $b = 0$.

The proximal operator of f_1 can be computed by combining the affine addition rule in [39, section 2.2] with [39, section 6.7.5], while the proximal operator of f_2 is simply the shrinkage operator [39, section 6.5.2]. The overall computational cost is dominated by the eigenvalue decomposition involved in evaluating $\mathbf{prox}_{tf_1}$, which has complexity $O(q^3)$.

Problem instance. We generated $S \in \mathbf{S}_{++}^q$, the set of symmetric positive definite matrices, with $q = 100$ and approximately 10% nonzero entries. Then we calculated Q using $p = 1000$ i.i.d. samples from $\mathcal{N}(0, S^{-1})$. Let $\alpha_{\max} = \sup_{i \neq j} |Q_{ij}|$ be the smallest α for which the solution of (7.3) is trivially the diagonal matrix $(\mathbf{diag}(Q) + \alpha I)^{-1}$ [8]. We solved (7.3) using $\alpha = 0.001\alpha_{\max}$, which produced an estimate of S with 7% nonzero entries.

Figure 3 depicts the residual norm curves. A2DR achieves $\|r^k\|_2 \leq 10^{-6}$ in less than 400 iterations, while DRS fails to fall below 10^{-4} even at 1000 iterations. The fluctuations in the A2DR residuals may be smoothed out by increasing the adaptive regularization coefficient η, but this generally leads to slower convergence.

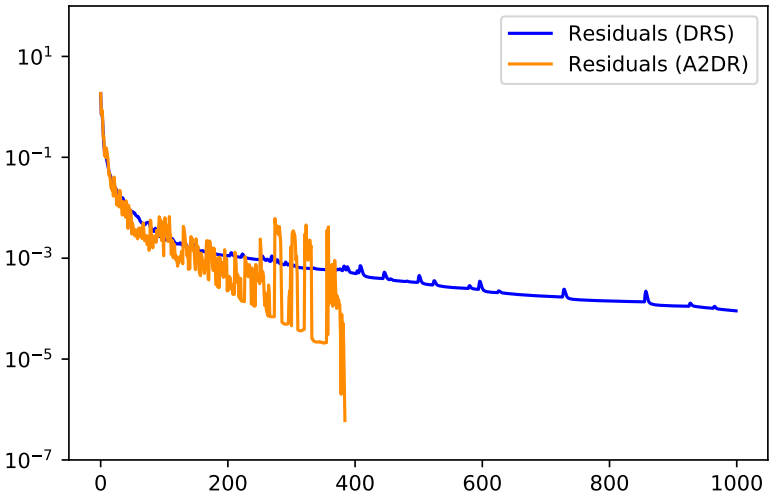

FIG. 3. *Sparse inverse covariance estimation: convergence of residual norms* $\|r^k\|_2$.

We also 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.

7.3. ℓ_1 trend filtering. The ℓ_1 trend filtering problem is [23]

$$\text{minimize} \quad \tfrac{1}{2}\|y - z\|_2^2 + \alpha\|Dz\|_1, \tag{7.4}$$

where $z \in \mathbf{R}^q$ is the variable, $y \in \mathbf{R}^q$ is the problem data (e.g., time series), $\alpha \geq 0$ is a smoothing parameter, and $D \in \mathbf{R}^{(q-2)\times q}$ is the second difference operator

$$D = \begin{bmatrix} 1 & -2 & 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -2 & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 & -2 & 1 \end{bmatrix}.$$

Again, we can rewrite the above problem in standard form (1.2) by letting

$$f_1(x_1) = \frac{1}{2}\|y - x_1\|_2^2, \quad f_2(x_2) = \alpha\|x_2\|_1$$

with variables $x_1 \in \mathbf{R}^q, x_2 \in \mathbf{R}^{q-2}$ and constraint matrices $A_1 = D, A_2 = -I$, and $b = 0$. The proximal operator of f_1 is simply $\mathbf{prox}_{tf_1}(v) = \frac{ty+v}{t+1}$, and the proximal operator of f_2 is the shrinkage operator [39, section 6.5.2]. Since D is tridiagonal, the projection $\Pi(v^{k+1/2})$ can be computed in $O(q)$.

Problem instance. We drew y from $\mathcal{N}(0, I)$ with $q = 10^6$ and solved (7.4) using $\alpha = 0.01\alpha_{\max}$, where $\alpha_{\max} = \|y\|_\infty$ is the smallest α for which the solution is trivially zero.

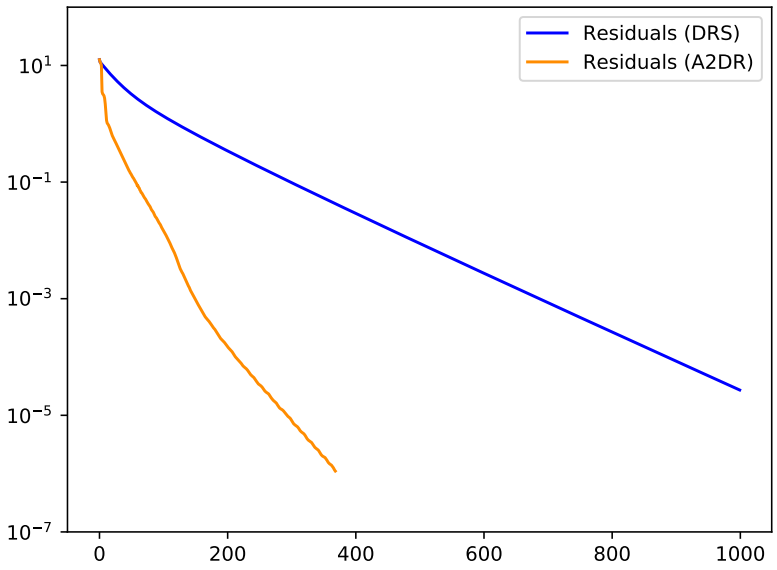

FIG. 4. *ℓ_1 trend filtering: convergence of residual norms $\|r^k\|_2$.*

The results are shown in Figure 4. A2DR converges about three times faster than DRS, reaching a tolerance of 10^{-6} in 360 iterations.

7.4. Single commodity flow optimization. Consider a network with p nodes and q (directed) arcs described by an incidence matrix $B \in \mathbf{R}^{p\times q}$ with

$$B_{ij} = \begin{cases} 1 & \text{arc } j \text{ enters node } i, \\ -1 & \text{arc } j \text{ leaves node } i, \\ 0 & \text{otherwise.} \end{cases}$$

Suppose a single commodity flows in this network. Let $z \in \mathbf{R}^q$ denote the arc flows and $s \in \mathbf{R}^p$ the node sources. We have the flow conservation constraint $Bz + s = 0$. This in turn implies $\mathbf{1}^T s = 0$ since $B^T\mathbf{1} = 0$ by construction. The total cost of traffic on the network is the sum of a flow cost, represented by $\psi : \mathbf{R}^q \to \mathbf{R} \cup \{\infty\}$, and a source cost, represented by $\phi : \mathbf{R}^p \to \mathbf{R} \cup \{\infty\}$. We assume that these costs are separable with respect to the flows and sources, i.e., $\psi(z) = \sum_{j=1}^q \psi_j(z_j)$ and $\phi(s) = \sum_{i=1}^p \phi_i(s_i)$. Our goal is to choose flow and source vectors such that the network cost is minimized:

$$\begin{array}{ll} \text{minimize} & \psi(z) + \phi(s) \\ \text{subject to} & Bz + s = 0 \end{array} \tag{7.5}$$

with respect to z and s.

We consider a special case modeled on the DC power flow problem in power engineering [32]. The flow costs are quadratic with a capacity constraint:

$$\psi_j(z_j) = \begin{cases} c_j z_j^2, & |z_j| \le z_j^{\max}, \\ +\infty & \text{otherwise.} \end{cases}$$

The source costs are determined by the node type, which can fall into one of three categories:

1. *Transfer/way-point nodes* fixed at $s_i = 0$, i.e., $\phi_i(s_i) = \mathcal{I}_{\{0\}}(s_i)$.
2. *Sink nodes* fixed at $s_i = L_i$ (for $L_i < 0$), i.e., $\phi_i(s_i) = \mathcal{I}_{\{L_i\}}(s_i)$.
3. *Source nodes* with cost

$$\phi_i(s_i) = \begin{cases} d_i s_i^2, & 0 \le s_i \le s_i^{\max}, \\ +\infty & \text{otherwise.} \end{cases}$$

The vectors $c \in \mathbf{R}^q_+, d \in \mathbf{R}^p_+, z^{\max} \in \mathbf{R}^q$, and $s^{\max} \in \mathbf{R}^p$ are constants.

This problem may be restated as (1.2) with $x_1 \in \mathbf{R}^q, x_2 \in \mathbf{R}^p, f_1(x_1) = \psi(x_1)$, $f_2(x_2) = \phi(x_2), A_1 = B, A_2 = I$, and $b = 0$. Since costs are separable, the proximal operators can be calculated elementwise as

$$\begin{aligned} (\mathbf{prox}_{tf_1}(v))_j &= \Pi_{[-z_j^{\max}, z_j^{\max}]}\left(\frac{v_j}{2tc_j + 1}\right), \quad j = 1, \dots, q, \\ (\mathbf{prox}_{tf_2}(w))_i &= \Pi_{[0, s_i^{\max}]}\left(\frac{w_i}{2td_i + 1}\right), \quad i \text{ is a source node.} \end{aligned} \tag{7.6}$$

Here $\Pi_{\mathcal{C}}$ denotes the projection onto the set $\mathcal{C}$. Notice that in evaluating the proximal operator, we implicitly solve a linear system related to $L = BB^T$, which is the Laplacian associated with the network.

Problem instance. We set $p = 4000$ and $q = 7000$ and generated the incidence matrix as follows. Let $\tilde{B} \in \mathbf{R}^{p \times (q-p+1)}$, where each column j is zero except for two entries $\tilde{B}_{ij} = 1$ and $\tilde{B}_{i'j} = -1$, whose positions are chosen uniformly at random. Define $\hat{B} \in \mathbf{R}^{p \times (p-1)}$ with $\hat{B}_{ii} = 1$ and $\hat{B}_{(i+1)i} = -1$ for $i = 1, \dots, p-1$. The final incidence matrix is $B = [\tilde{B}\ \hat{B}]$.

To construct the source vector, we first drew $\breve{s} \in \mathbf{R}^p$ i.i.d. from $\mathcal{N}(0, I)$ and defined

$$\tilde{s}_i = \begin{cases} 0, & i = 1, \dots, \lfloor \frac{p}{3} \rfloor, \\ -|\breve{s}_i|, & i = \lfloor \frac{p}{3} \rfloor + 1, \dots, \lfloor \frac{2p}{3} \rfloor, \\ \sum_{l=\lfloor p/3 \rfloor + 1}^{\lfloor 2p/3 \rfloor} |\breve{s}_l| / (p - \lfloor \frac{2p}{3} \rfloor), & i = \lfloor \frac{2p}{3} \rfloor + 1, \dots, p. \end{cases}$$

We took the first $\lfloor \frac{p}{3} \rfloor$ entries to be the transfer nodes, the second $\lfloor \frac{2p}{3} \rfloor - \lfloor \frac{p}{3} \rfloor$ entries to be the sink nodes with $L_i = \tilde{s}_i$, and the last $p - \lfloor \frac{2p}{3} \rfloor$ entries to be the source nodes, where

$$s_i^{\max} = \begin{cases} \tilde{s}_i + 0.001, & i = \lfloor \frac{2p}{3} \rfloor, \dots, \lfloor \frac{5p}{6} \rfloor, \\ 2(\tilde{s}_i + 0.001), & i = \lfloor \frac{5p}{6} \rfloor, \dots, p. \end{cases}$$

To get the flow bounds, we solved $B\tilde{x} = -\tilde{s}$ for $\tilde{x}$, and let

$$x_j^{\max} = \begin{cases} |\tilde{x}_j| + 0.001, & j = 1, \dots, \lfloor \frac{q}{2} \rfloor, \\ 2(|\tilde{x}_j| + 0.001), & j = \lfloor \frac{q}{2} \rfloor + 1, \dots, q. \end{cases}$$

Finally, the entries of c and d were drawn i.i.d. from Uniform$(0, 1)$.

Figure 5 depicts the results of our experiment. A2DR converges to a tolerance of 10^{-6} in less than 1200 iterations, while DRS remains above 10^{-4} even once the maximum iterations of 2000 is reached. For this problem, we also attempted to find a solution using SCS, but the solver failed to converge to its default tolerance of 10^{-5} in 5000 iterations, finishing with a linear constraint violation of $\|Bz + s\|_2 > 0.3$. In contrast, A2DR's final result yields $\|Bz + s\|_2 \approx 10^{-6}$.

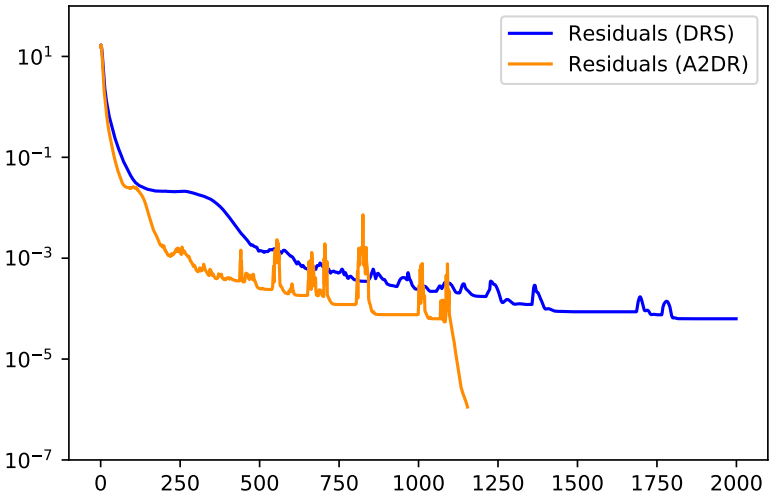

FIG. 5. *Single commodity flow: convergence of residual norms* $\|r^k\|_2$.

7.5. Optimal control. We are interested in the following finite-horizon optimal control problem:

$$\begin{array}{ll} \text{minimize} & \sum_{l=1}^{L} \phi_l(z_l, u_l) \\ \text{subject to} & z_{l+1} = F_l z_l + G_l u_l + h_l, \quad l = 1, \ldots, L-1, \\ & z_1 = z_{\text{init}}, \quad z_L = z_{\text{term}} \end{array} \tag{7.7}$$

with state variables $z_l \in \mathbf{R}^q$, control variables $u_l \in \mathbf{R}^p$, and cost functions $\phi_l : \mathbf{R}^q \times \mathbf{R}^p \to \mathbf{R} \cup \{\infty\}$. The data consist of an initial state $z_{\text{init}} \in \mathbf{R}^q$, a terminal state $z_{\text{term}} \in \mathbf{R}^q$, and dynamics matrices $F_l \in \mathbf{R}^{q\times q}, G_l \in \mathbf{R}^{q\times p}$, and $h_l \in \mathbf{R}^q$ for $l = 1, \ldots, L-1$. Let $z = (z_1, \ldots, z_L) \in \mathbf{R}^{Lq}$ and $u = (u_1, \ldots, u_L) \in \mathbf{R}^{Lp}$. If we define

$$\tilde{F} = \begin{bmatrix} I & 0 & \ldots & 0 & 0 \\ -F_1 & I & \ldots & 0 & 0 \\ 0 & -F_2 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & -F_{L-1} & I \\ 0 & 0 & \ldots & 0 & I \end{bmatrix}, \quad \tilde{G} = \begin{bmatrix} 0 & 0 & \ldots & 0 & 0 \\ -G_1 & 0 & \ldots & 0 & 0 \\ 0 & -G_2 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & -G_{L-1} & 0 \\ 0 & 0 & \ldots & 0 & 0 \end{bmatrix},$$

and $\tilde{h} = (z_{\text{init}}, h_1, \ldots, h_{L-1}, z_{\text{term}})$, then the constraints can be written compactly as $\tilde{F}z + \tilde{G}u = \tilde{h}$.

We focus on a time-invariant linear quadratic version of (7.7) with $F_l = F, G_l = G, h_l = 0$, and

$$\phi_l(z_l, u_l) = \|z_l\|_2^2 + \|u_l\|_2^2 + \mathcal{I}_{\{u\,:\,\|u\|_\infty \le 1\}}(u_l), \quad l = 1, \ldots, L.$$

This problem is equivalent to (1.2) with $x_1 \in \mathbf{R}^{Lq}, x_2 \in \mathbf{R}^{Lp}$,

$$f_1(x_1) = \|x_1\|_2^2, \quad f_2(x_2) = \|x_2\|_2^2 + \mathcal{I}_{\{u\,:\,\|u\|_\infty \le 1\}}(x_2),$$

and constraint matrices $A_1 = \tilde{F}, A_2 = \tilde{G}$, and $b = \tilde{h}$. The proximal operators of f_i have closed forms $\mathbf{prox}_{tf_1}(v) = \frac{v}{2t+1}$ and $\mathbf{prox}_{tf_2}(w) = \Pi_{[-1,1]}\left(\frac{w}{2t+1}\right)$.

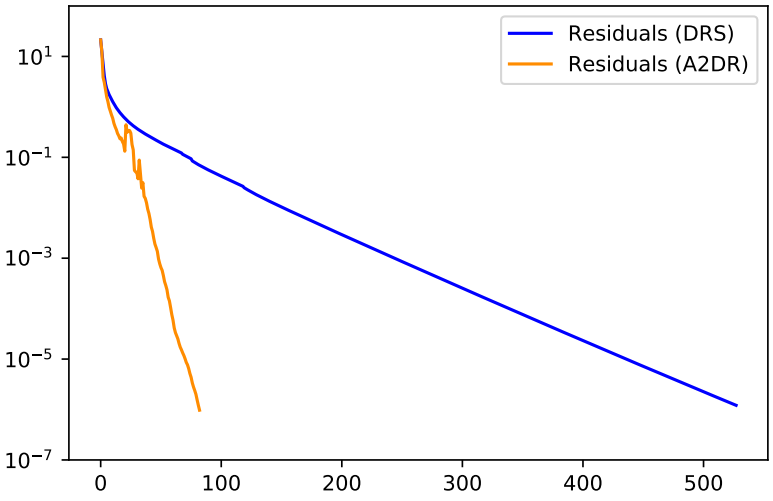

FIG. 6. *Optimal control: convergence of residual norms* $\|r^k\|_2$.

Problem instance. We set $p = 80, q = 150$, and $L = 20$ and drew the entries of F, G, h, and z_{init} i.i.d. from $\mathcal{N}(0,1)$. The matrix F was scaled by its spectral radius so its largest eigenvalue has magnitude one. To determine z_{term}, we drew $\hat{u}_l \in \mathbf{R}^p$ i.i.d. from $\mathcal{N}(0, I)$, normalized to get $\tilde{u}_l = \hat{u}_l / \|\hat{u}_l\|_\infty$, and computed $\tilde{z}_{l+1} = F\tilde{z}_l + G\tilde{u}_l + h$ for $l = 1, \ldots, L-1$ starting from $\tilde{z}_1 = z_{\text{init}}$. We then chose the terminal state to be $z_{\text{term}} = \tilde{z}_L$.

Figure 6 depicts the residual curves for problem (7.7). DRS requires over five times as many iterations to converge as A2DR, which reaches a tolerance of 10^{-6} in just under 100 iterations. For comparison, we solved the same problem in CVXPY with OSQP and SCS and found that neither solver converged to its default tolerance (10^{-4} and 10^{-5}, respectively) by its maximum number of iterations. Indeed, OSQP returned a solver error, while SCS terminated with a linear constraint violation of $\|\tilde{F}z + \tilde{G}u - \tilde{h}\|_2 > 0.9$. A2DR's final constraint violation is only about 10^{-6}.

7.6. Coupled quadratic program. We consider a quadratic program in which L variable blocks are coupled through a set of s linear constraints, represented as

$$\begin{array}{ll} \text{minimize} & \sum_{l=1}^{L} z_l^T Q_l z_l + c_l^T z_l \\ \text{subject to} & F_l z_l \le d_l, \quad l = 1, \ldots, L, \\ & \sum_{l=1}^{L} G_l z_l = h \end{array} \tag{7.8}$$

with respect to $z = (z_1, \ldots, z_L)$, where $z_l \in \mathbf{R}^{q_l}, Q_l \in \mathbf{S}_+^{q_l}, c_l \in \mathbf{R}^{q_l}, F_l \in \mathbf{R}^{p_l \times q_l}, d_l \in \mathbf{R}^{p_l}, G_l \in \mathbf{R}^{s \times q_l}$, and $h \in \mathbf{R}^s$ for $l = 1, \ldots, L$.

We can rewrite (7.8) in standard form with $N = L$, $x = z$,

$$f_i(x_i) = x_i^T Q_i x_i + c_i^T x_i + \mathcal{I}_{\{x\,:\,F_i x \le d_i\}}(x_i), \quad i = 1, \ldots, L,$$

$A = [G_1 \ \cdots \ G_L]$, and $b = h$. The proximal operator $\mathbf{prox}_{tf_i}(v_i)$ is evaluated by solving

$$\begin{array}{ll} \text{minimize} & x_i^T \left(Q_i + \frac{1}{2t} I\right) x_i + (c_i - \frac{1}{t} v_i)^T x_i \\ \text{subject to} & F_i x_i \le d_i \end{array} \tag{7.9}$$

with respect to $x_i \in \mathbf{R}^{q_i}$.

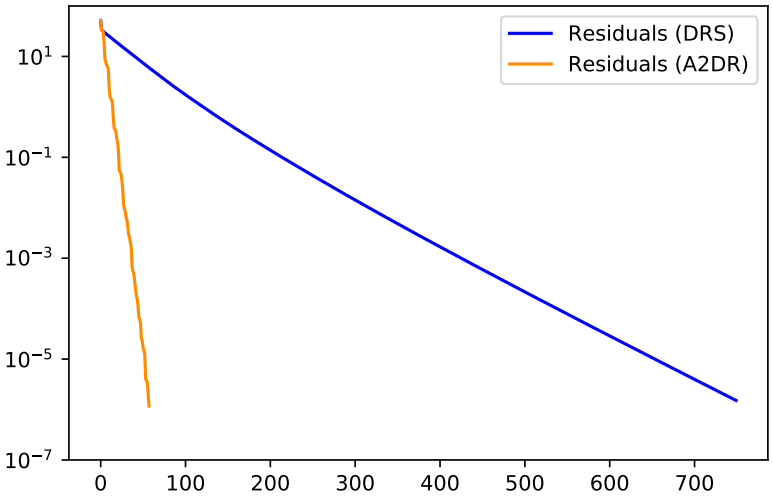

FIG. 7. *Coupled quadratic program: convergence of residual norms* $\|r^k\|_2$.

Problem instance. Let $L = 8$, $s = 50$, $q_l = 300$, and $p_l = 200$ for $l = 1, \ldots, L$. We generated the entries of $c_l \in \mathbf{R}^{q_l}, F_l \in \mathbf{R}^{p_l \times q_l}, G_l \in \mathbf{R}^{s \times q_l}, \tilde{z}_l \in \mathbf{R}^{q_l}$, and $H_l \in \mathbf{R}^{q_l \times q_l}$ i.i.d. from $\mathcal{N}(0,1)$. We then formed $d_l = F_l \tilde{z}_l + 0.1, Q_l = H_l^T H_l$, and $h = \sum_{l=1}^{L} G_l \tilde{z}_l$. To evaluate the proximal operators, we constructed problem (7.9) in CVXPY and solved it using OSQP with the default tolerance.

The results of our experiment are shown in Figure 7. A2DR produces an over ten-fold speedup, converging to the desired tolerance of 10^{-6} in only 60 iterations.

7.7. Multitask regularized logistic regression. Consider the following multitask regression problem:

$$\text{minimize} \quad \phi(W\theta, Y) + r(\theta) \tag{7.10}$$

with variable $\theta = [\theta_1 \cdots \theta_L] \in \mathbf{R}^{s \times L}$. Here $\phi : \mathbf{R}^{p \times L} \times \mathbf{R}^{p \times L} \to \mathbf{R}$ is the loss function, $r : \mathbf{R}^{s \times L} \to \mathbf{R}$ is the regularizer, $W \in \mathbf{R}^{p \times s}$ is the feature matrix shared across the L tasks, and $Y = [y_1 \cdots y_L] \in \mathbf{R}^{p \times L}$ contains the p class labels for each task $l = 1, \ldots, L$.

We focus on the binary classification problem, so that all entries of Y are ± 1. Accordingly, we take our loss function to be the logistic loss summed over samples and tasks,

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

where $Z \in \mathbf{R}^{p \times L}$, and our regularizer to be a linear combination of the group lasso penalty [20] and the nuclear norm,

$$r(\theta) = \alpha \|\theta\|_{2,1} + \beta \|\theta\|_*,$$

where $\|\theta\|_{2,1} = \sum_{l=1}^{L} \|\theta_l\|_2$ and $\alpha > 0,\ \beta > 0$ are regularization parameters.

Problem (7.10) can be converted to standard form (1.2) by letting

$$f_1(Z) = \phi(Z, Y), \quad f_2(\theta) = \alpha \|\theta\|_{2,1}, \quad f_3(\tilde{\theta}) = \beta \|\tilde{\theta}\|_*,$$

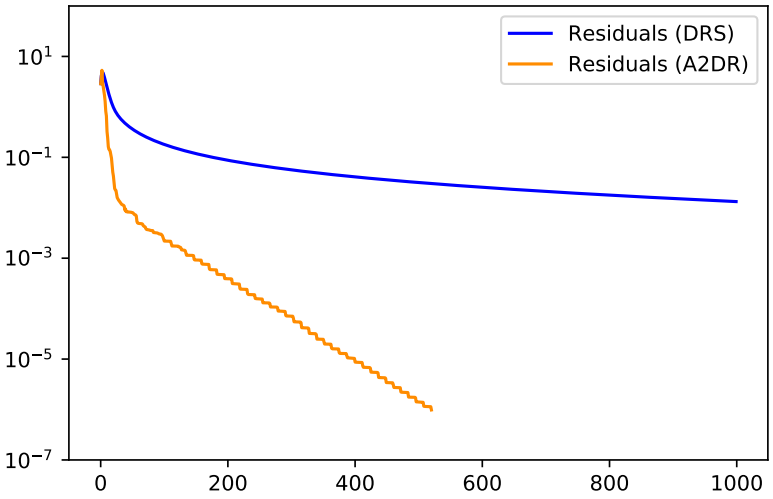

FIG. 8. *Multitask regularized logistic regression: convergence of residual norms* $\|r^k\|_2$.

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

The proximal operator of f_1 can be evaluated efficiently via Newton type methods applied to each component in parallel [14], while the proximal operators of the regularization terms have closed-form expressions [39, sections 6.5.4 and 6.7.3].

Problem instance. We let $p = 300$, $s = 500$, $L = 10$, and $\alpha = \beta = 0.1$. The entries of $W \in \mathbf{R}^{p\times s}$ and $\theta^\star \in \mathbf{R}^{s\times L}$ were drawn i.i.d. from $\mathcal{N}(0,1)$. We calculated $Y = \mathbf{sign}(W\theta^\star)$, where the signum function is applied elementwise with the convention $\mathbf{sign}(0) = -1$. To evaluate $\mathbf{prox}_{tf_1}$, we used the Newton-CG method from `scipy.optimize.minimize`, warm starting each iteration with the output from the previous iteration. (Further performance improvements may be achieved by implementing Newton's method with unit step size and initial point zero for each component in parallel [14].)

Figure 8 shows the residual plots for A2DR and DRS. The A2DR curve exhibits a steep drop in the first few steps and continues falling until convergence at 500 iterations. In contrast, the DRS residual norms never make it below a tolerance of 10^{-2}.

8. Conclusions. We have presented an algorithm for solving linearly constrained convex optimization problems, where the objective function is only accessible via its proximal operator. Our algorithm is an application of type-II Anderson acceleration to Douglas–Rachford splitting (A2DR). Under relatively mild conditions, we prove that A2DR either converges to a global optimum or provides a certificate of infeasibility/unboundedness. Moreover, when the objective is block separable, its steps partially decouple so that they may be computed in parallel, enabling fast distributed implementations. We provide one such Python implementation at https://github.com/cvxgrp/a2dr. Using only the default parameters, we show that our solver achieves rapid convergence on a wide range of problems, making it a robust choice for general large-scale convex optimization.

In the future, we plan to release a user-friendly interface, which automatically reduces a problem to the standard form (1.2) input of the A2DR solver, similar to the Epsilon system [58]. This will allow us to integrate A2DR into a high-level domain specific language for convex optimization. We also intend to expand the library of proximal operators. As problems grow larger, we aim to support more parallel computing architectures, allowing users to leverage GPU acceleration and high-performance clusters for distributed optimization.

Acknowledgment. The authors would like to thank Brendan O'Donoghue for his advice on preconditioning and his inspirational work developing solvers with Anderson acceleration, pioneered by SCS 2.0.

REFERENCES

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, U.S. National Bureau of Standards, Washington, DC, 1964.

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

[3] A. C. Aitken, *On Bernoulli's numerical solution of algebraic equations*, Proc. Roy. Soc. Edinburgh, 46 (1927), pp. 289–305.

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

[5] D. G. Anderson, *Iterative procedures for nonlinear integral equations*, J. Assoc. Comput. Mach., 12 (1965), pp. 547–560.

[6] N. S. Aybat, Z. Wang, T. Lin, and S. Ma, *Distributed linearized alternating direction method of multipliers for composite convex consensus optimization*, IEEE Trans. Automat. Control, 63 (2018), pp. 5–20.

[7] F. Bach, *Acceleration without pain*, https://francisbach.com/acceleration-without-pain, Feb. 4, 2020.

[8] O. Banerjee, L. E. Ghaoui, and A. d'Aspremont, *Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data*, J. Mach. Learn. Res., 9 (2008), pp. 485–516.

[9] P. Bansode, K. C. Kosaraju, S. R. Wagh, R. Pasumarthy, and N. M. Singh, *Accelerated distributed primal-dual dynamics using adaptive synchronization*, IEEE Access, 7 (2019), pp. 120424–120440.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Found. Trends Mach. Learn., 3 (2011), pp. 1–122.

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

[12] C. Brezinski, M. Redivo-Zaglia, and Y. Saad, *Shanks sequence transformations and Anderson acceleration*, SIAM Rev., 60 (2018), pp. 646–669, https://doi.org/10.1137/17M1120725.

[13] X. Chen and C. T. Kelley, *Convergence of the EDIIS algorithm for nonlinear equations*, SIAM J. Sci. Comput., 41 (2019), pp. A365–A379, https://doi.org/10.1137/18M1171084.

[14] A. Defazio, *A simple practical accelerated method for finite sums*, in Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 676–684.

[15] S. Diamond and S. Boyd, *CVXPY: A Python-embedded modeling language for convex optimization*, J. Mach. Learn. Res., 17 (2016), pp. 1–5.

[16] C. Evans, S. Pollock, L. G. Rebholz, and M. Xiao, *A Proof that Anderson Acceleration Increases the Convergence Rate in Linearly Converging Fixed Point Methods (But Not in Quadratically Converging Ones)*, preprint, https://arxiv.org/abs/1810.08455, 2018.

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

[18] C. Fougner and S. Boyd, *Parameter selection and preconditioning for a graph form solver*, in Emerging Applications of Control and Systems Theory, R. Tempo, S. Yurkovich, and P. Misra, eds., Lect. Notes Control Inf. Sci. Proc., Springer, Cham, pp. 41–61.

[19] J. Friedman, T. Hastie, and R. Tibshirani, *Sparse inverse covariance estimation with the graphical lasso*, Biostatistics, 9 (2008), pp. 432–441.
[20] J. Friedman, T. Hastie, and R. Tibshirani, *A Note on the Group Lasso and a Sparse Group Lasso*, preprint, https://arxiv.org/abs/1001.0736, 2010.
[21] B. S. He, H. Yang, and S. L. Wang, *Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities*, J. Optim. Theory Appl., 106 (2000), pp. 337–356.
[22] H. He and D. Han, *A distributed Douglas-Rachford splitting method for multi-block convex minimization problems*, Adv. Comput. Math., 42 (2016), pp. 27–53.
[23] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky, ℓ_1 *trend filtering*, SIAM Rev., 51 (2009), pp. 339–360, https://doi.org/10.1137/070690274.
[24] K. N. Kudin and G. E. Scuseria, *A black-box self-consistent field convergence algorithm: One step closer*, J. Chem. Phys., 116 (2002), pp. 8255–8261.
[25] Z. Li and J. Li, *An Anderson-Chebyshev Mixing Method for Nonlinear Optimization*, preprint, https://arxiv.org/abs/1809.02341, 2018.
[26] Y. Liu, E. K. Ryu, and W. Yin, *A new use of Douglas-Rachford splitting for identifying infeasible, unbounded, and pathological conic programs*, Math. Program., 177 (2019), pp. 225–253.
[27] J. Loffeld and C. S. Woodward, *Considerations on the implementation and use of Anderson acceleration on distributed memory and GPU-based parallel computers*, in Advances in the Mathematical Sciences, Springer, Cham, 2016, pp. 417–436.
[28] A. J. Macleod, *Acceleration of vector sequences by multi-dimensional* Δ^2 *methods*, Commun. Appl. Numer. Methods, 2 (1986), pp. 385–392.
[29] V. V. Mai and M. Johansson, *Nonlinear acceleration of constrained optimization algorithms*, in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2019, pp. 4903–4907.
[30] M. Mešina, *Convergence acceleration for the iterative solution of the equations* $X = AX + f$, Comput. Methods Appl. Mech. Eng., 10 (1977), pp. 165–173.
[31] A. Milzarek, X. Xiao, S. Cen, Z. Wen, and M. Ulbrich, *A stochastic semismooth Newton method for nonsmooth nonconvex optimization*, SIAM J. Optim., 29 (2019), pp. 2916–2948, https://doi.org/10.1137/18M1181249.
[32] N. Moehle, E. Busseti, S. Boyd, and M. Wytock, *Dynamic energy management*, in Large Scale Optimization in Supply Chains and Smart Manufacturing, J. M. Velásquez-Bermúdez, M. Khakifirooz, and M. Fathi, eds., Springer Optim. Appl. 149, Springer, Cham, 2019, pp. 69–126.
[33] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Kluwer Academic Publishers, Boston, 2004.
[34] J. Nocedal and S. Wright, *Numerical Optimization*, Springer, New York, 2006.
[35] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, *Conic optimization via operator splitting and homogeneous self-dual embedding*, J. Optim. Theory Appl., 169 (2016), pp. 1042–1068.
[36] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, *SCS: Splitting conic solver, version* 2.1.2. https://github.com/cvxgrp/scs, November 2019.
[37] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Softw., 8 (1982), pp. 43–71.
[38] N. Parikh and S. Boyd, *Block splitting for distributed optimization*, Math. Program. Comput., 6 (2014), pp. 77–102.
[39] N. Parikh and S. Boyd, *Proximal algorithms*, Found. Trends Optim., 1 (2014), pp. 127–239.
[40] Y. Peng, B. Deng, J. Zhang, F. Geng, W. Qin, and L. Liu, *Anderson acceleration for geometry optimization and physics simulation*, ACM Trans. Graph., 37 (2018), 42.
[41] S. Pollock and L. Rebholz, *Anderson Acceleration for Contractive and Noncontractive Operators*, preprint, https://arxiv.org/abs/1909.04638, 2019.
[42] T. Rohwedder and R. Schneider, *An analysis for the DIIS acceleration method used in quantum chemistry calculations*, J. Math. Chem., 49 (2011), pp. 1889–1914.
[43] E. K. Ryu and S. Boyd, *A primer on monotone operator methods*, Appl. Comput. Math, 15 (2016), pp. 3–43.
[44] E. K. Ryu, Y. Liu, and W. Yin, *Douglas-Rachford splitting and ADMM for pathological convex optimization*, Comput. Optim. Appl., 74 (2019), pp. 747–778.
[45] D. Scieur, F. Bach, and A. d'Aspremont, *Nonlinear acceleration of stochastic algorithms*, in Advances in Neural Information Processing Systems, Curran Associates, Red Hook, NY, 2017, pp. 3982–3991.
[46] D. Scieur, A. d'Aspremont, and F. Bach, *Regularized nonlinear acceleration*, in Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016,

pp. 712–720.
[47] D. Shanks, *Non-linear transformations of divergent and slowly convergent sequences*, J. Math. and Phys., 34 (1955), pp. 1–42.
[48] D. A. Smith, W. F. Ford, and A. Sidi, *Extrapolation methods for vector sequences*, SIAM Rev., 29 (1987), pp. 199–233, https://doi.org/10.1137/1029042.
[49] P. Sopasakis, K. Menounou, and P. Patrinos, *SuperSCS: Fast and accurate large-scale conic optimization*, in Proceedings of the European Control Conference, 2019, pp. 1500–1505.
[50] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, *OSQP: An operator splitting solver for quadratic programs*, in Proceedings of the UKACC International Conference on Control, 2018, p. 339.
[51] A. Themelis and P. Patrinos, *SuperMann: A superlinearly convergent algorithm for finding fixed points of nonexpansive operators*, IEEE Trans. Automat. Control, 64 (2019), pp. 4875–4890.
[52] A. Toth, J. A. Ellis, T. Evans, S. Hamilton, C. T. Kelley, R. Pawlowski, and S. Slattery, *Local improvement results for Anderson acceleration with inaccurate function evaluations*, SIAM J. Sci. Comput., 39 (2017), pp. S47–S65, https://doi.org/10.1137/16M1080677.
[53] A. Toth and C. T. Kelley, *Convergence analysis for Anderson acceleration*, SIAM J. Numer. Anal., 53 (2015), pp. 805–819, https://doi.org/10.1137/130919398.
[54] J. Tuck, S. Barratt, and S. Boyd, *A Distributed Method for Fitting Laplacian Regularized Stratified Models*, preprint, https://arxiv.org/abs/1904.12017, 2019.
[55] H. F. Walker and P. Ni, *Anderson acceleration for fixed-point iterations*, SIAM J. Numer. Anal., 49 (2011), pp. 1715–1735, https://doi.org/10.1137/10078356X.
[56] E. Wei and A. Ozdaglar, *Distributed alternating direction method of multipliers*, in Proceedings of the IEEE Conference on Decision and Control, 2012, pp. 5445–5450.
[57] P. Wynn, *On a device for computing the $e_m(S_n)$ transformation*, Math. Comp., 10 (1956), pp. 91–96.
[58] M. Wytock, P.-W. Wang, and J. Z. Kolter, *Convex Programming with Fast Proximal and Linear Operators*, preprint, https://arxiv.org/abs/1511.04815, 2015.
[59] X. Xiao, Y. Li, Z. Wen, and L. Zhang, *A regularized semi-smooth Newton method with projection steps for composite convex programs*, J. Sci. Comput., 76 (2018), pp. 364–389.
[60] Z. Xu, G. Taylor, H. Li, M. Figueiredo, X. Yuan, and T. Goldstein, *Adaptive consensus ADMM for distributed optimization*, in Proceedings of the International Conference on Machine Learning, 2017, pp. 3841–3850.
[61] J. Zhang, B. O'Donoghue, and S. Boyd, *Globally Convergent type-*I *Anderson Acceleration for Non-smooth Fixed-Point Iterations*, preprint, https://arxiv.org/abs/1808.03971, 2018.
[62] J. Zhang, Y. Peng, W. Ouyang, and B. Deng, *Accelerating ADMM for efficient simulation and optimization*, ACM Trans. Graph., 38 (2019), 163.
[63] J. Zhang, C. A. Uribe, A. Mokhtari, and A. Jadbabaie, *Achieving acceleration in distributed optimization via direct discretization of the heavy-ball ODE*, in Proceedings of the American Control Conference, 2019, pp. 3408–3413.
[64] Y. Zhang and M. M. Zavlanos, *A consensus-based distributed augmented Lagrangian method*, in Proceedings of the IEEE Conference on Decision and Control, 2018, pp. 1763–1768.