Distributed Majorization-Minimization for Laplacian Regularized Problems

Jonathan Tuck, *Member, IEEE,* David Hallac, *Member, IEEE,* and Stephen Boyd, *Fellow, IEEE*

***Abstract*—We consider the problem of minimizing a block separable convex function (possibly nondifferentiable, and including constraints) plus Laplacian regularization, a problem that arises in applications including model fitting, regularizing stratified models, and multi-period portfolio optimization. We develop a distributed majorization-minimization method for this general problem, and derive a complete, self-contained, general, and simple proof of convergence. Our method is able to scale to very large problems, and we illustrate our approach on two applications, demonstrating its scalability and accuracy.**

***Index Terms*—Convex optimization, distributed optimization, graphical networks, Laplacian regularization.**

I. Introduction

MANY applications, ranging from multi-period portfolio optimization [1] to joint covariance estimation [2], can be modeled as convex optimization problems with two objective terms, one that is block separable and the other a Laplacian regularization term [3]. The block separable term can be nondifferentiable and may include constraints. The Laplacian regularization term is quadratic, and penalizes differences between individual variable components. These types of problems arise in several domains, including signal processing [4], machine learning [5], and statistical estimation or data fitting problems with an underlying graph prior [6], [7]. As such, there is a need for scalable algorithms to efficiently solve these problems.

In this paper we develop a distributed method for minimizing a block-separable convex objective with Laplacian regularization. Our method is iterative; in each iteration a convex problem is solved for each block, and the variables are then shared with each block's neighbors in the graph associated with the Laplacian term. Our method is an instance of a standard and well known general method, majorization-minimization (MM) [8], which recovers a wide variety of existing methods depending on the choice of majorization [9]. The advantages of MM over such methods as the alternating direction method of multipliers [10] and subgradient-based methods is that given an appropriate majorizer, which in general are not hard to find [9], MM can exploit structure (such as separability) in such a way that it can give better performance over these other methods [11]. In this paper, we derive a diagonal quadratic majorizer of the given Laplacian objective term, which has the benefit of separability. This separability allows for the minimization step in our MM algorithm to be carried out in parallel on a block-by-block basis. We develop a completely self-contained proof of convergence of our method, which relies on no further assumption than the existence of a solution. Finally, we apply our method to two separate applications, multi-period portfolio optimization and joint covariance estimation, demonstrating the scalable performance of our algorithm.

A. Related Work

There has been extensive research on graph Laplacians [12]−[14] and Laplacian regularization [15]−[17], and on developing solvers specifically for use in optimization over graphs [18]. In addition, there has been much research done on the MM algorithm [6], [8], [9], [17], [19], [20], including interpreting other well studied algorithms, such as the concave-convex procedure and the expectation-maximization algorithm [21], [22] as special cases of MM. We are not aware of any previous work that applies the MM algorithm to Laplacian regularization.

There has also been much work on the two specific application examples that we consider. Multi-period portfolio optimization is studied in, for example, [1], [23], [24], although scalability remains an issue in these studies. Our second application example arises in signal processing, specifically the joint estimation of inverse covariance matrices, which has been studied and applied in many different contexts, such as cell signaling [25], [26], statistical learning [27], radar signal processing [28], and medical imaging [29]. Again, scalability here is either not referenced or is still an ongoing issue in these fields.

B. Outline

In Section II we set up our notation, and describe the problem of Laplacian regularized minimization. In Section III we show how to construct a diagonal quadratic majorizer of a weighted Laplacian quadratic form. In Section IV we describe our distributed MM algorithm, and give a complete and self-contained proof of convergence. Finally, in Section V we present numerical results for two applications which demonstrates the effectiveness of our method.

Manuscript received September 27, 2018; revised November 16, 2018; accepted November 27, 2018. Recommended by Associate Editor Tengfei Liu. *(Corresponding author: Jonathan Tuck.)*

Citation: J. Tuck, D. Hallac, and S. Boyd, "Distributed majorization-minimization for Laplacian regularized problems," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 1, pp. 45−52, Jan. 2019.

J. Tuck, D. Hallac, and S. Boyd are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mails: {jtuck1, hallac, boyd}@stanford.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2019.1911321

II. Laplacian Regularized Minimization

We consider the problem of minimizing a convex function plus Laplacian regularization,

$$\text{minimize} \qquad F(x) = f(x) + \mathcal{L}(x), \tag{1}$$

with variable $x \in \mathbb{R}^n$. Here $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a proper closed convex function [30], [31], and $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ is the Laplacian regularizer (or Dirichlet energy [32]) $\mathcal{L}(x) = (1/2)x^T Lx$, where L is a weighted Laplacian matrix, i.e., $L = L^T$, $L_{ij} \leq 0$ for $i \neq j$, and $L\mathbf{1} = 0$, where $\mathbf{1}$ is the vector with all entries one [15]. Associating with $\mathcal{L}$ the graph with vertices $1, \ldots, n$, edges indexed by pairs (i, j) with $i < j$ and $L_{ij} \neq 0$, and (nonnegative) edge weights $w_{ij} = -2L_{ij}$, the Laplacian regularizer can be expressed as

$$\mathcal{L}(z) = \sum_{(i,j)\in\mathcal{E}} w_{ij}(z_i - z_j)^2.$$

We refer to the problem (1) as the *Laplacian regularized minimization problem* (LRMP). LRMPs are convex optimization problems, which can be solved by a variety of methods, depending on the specific form of f [33], [34]. We will let $F^\star$ denote the optimal value of the LRMP. Convex constraints can be incorporated into LRMP, by defining f to take value $+\infty$ when the constraints are violated. Note in particular that we specifically *do not* assume that the function f is finite, or differentiable (let alone with Lipschitz gradient), or even that its domain has affine dimension n. In this paper we will make only one additional analytical assumption about the LMRP (1): its sublevel sets are bounded. This assumption implies that the LRMP is solvable, i.e., there exists at least one optimal point $x^\star$, and therefore that its optimal value $F^\star$ is finite.

A point x is optimal for the LRMP (1) if and only if there exists $g \in \mathbb{R}^n$ such that [30], [31]

$$g \in \partial f(x), \quad g + \nabla\mathcal{L}(x) = g + Lx = 0, \tag{2}$$

where $\partial f(x)$ is the subdifferential of f at x [30], [35]. For $g \in \partial f(x)$, we refer to

$$r = g + Lx$$

as the *optimality residual* for the LRMP (1). Our goal is to compute an x (and $g \in \partial f(x)$) for which the residual r is small.

We are interested in the case where f is block separable. We partition the variable x as $x = (x_1, \ldots, x_p)$, with $x_i \in \mathbb{R}^{n_i}$, $n_1 + \cdots + n_p = n$, and assume f has the form

$$f(x) = \sum_{i=1}^{p} f_i(x_i),$$

where $f_i : \mathbb{R}^n \to \mathbb{R}\cup\{\infty\}$ are closed convex proper functions.

The main contribution of this paper is a scalable and distributed method for solving LRMP in which each of the functions f_i is handled separately. More specifically, we will see that each iteration of our algorithm requires the evaluation of a diagonally scaled proximal operator [36] associated with each block function f_i, which can be done in parallel.

III. Diagonal Quadratic Majorization of the Laplacian

Recall that a function $\hat{\mathcal{L}} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a majorizer of $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ if for all x and z, $\hat{\mathcal{L}}(z; z) = \mathcal{L}(z)$, and $\hat{\mathcal{L}}(x; z) \geq \mathcal{L}(x)$ [8], [9]. In other words, the difference $\hat{\mathcal{L}}(x, z) - \mathcal{L}(z)$ is nonnegative, and zero when $x = z$.

We now show how to construct a quadratic majorizer of the Laplacian regularizer $\mathcal{L}$. This construction is known [9], but we give the proof for completeness. Suppose $\hat{L} = \hat{L}^T$ satisfies $\hat{L} \succeq L$, i.e., $\hat{L} - L$ is positive semidefinite. The function

$$\begin{aligned}\hat{\mathcal{L}}(x; z) = &\frac{1}{2}z^T Lz + z^T L(x - z)\\ &+ \frac{1}{2}(x - z)^T \hat{L}(x - z),\end{aligned} \tag{3}$$

which is quadratic in x, is a majorizer of $\mathcal{L}$. To see this, we note that

$$\begin{aligned}\hat{\mathcal{L}}(x; z) - \mathcal{L}(x) = &\frac{1}{2}z^T Lz + z^T L(x - z)\\ &+ \frac{1}{2}(x - z)^T \hat{L}(x - z) - \frac{1}{2}x^T Lx\\ = &\frac{1}{2}(x - z)^T (\hat{L} - L)(x - z),\end{aligned}$$

which is always nonnegative, and zero when $x = z$.

In fact, every quadratic majorizer of $\mathcal{L}$ arises from this construction, for some $\hat{L} \succeq L$. To see this we note that the difference $\hat{\mathcal{L}}(x; z) - \mathcal{L}(x)$ is a quadratic function of x that is nonnegative and zero when $x = z$, so it must have the form $(1/2)(x-z)^T P(x-z)$ for some $P = P^T \succeq 0$. It follows that $\hat{\mathcal{L}}$ has the form (3), with $\hat{L} = P + L \succeq L$.

We now give a simple scheme for choosing $\hat{L}$ in the diagonal quadratic majorizer. Suppose $\hat{L}$ is diagonal,

$$\hat{L} = \mathrm{diag}(\alpha) = \mathrm{diag}(\alpha_1, \ldots, \alpha_n),$$

where $\alpha \in \mathbb{R}^n$. A simple sufficient condition for $\hat{L} \succeq L$ is $\alpha_i \geq 2L_{ii}$, $i = 1, \ldots, n$. This follows from standard results for Laplacians [37], but it is simple to show directly. We note that for any $z \in \mathbb{R}^n$, we have

$$\begin{aligned}z^T(\hat{L} - L)z = &\sum_{i=1}^{n}(\alpha_i - L_{ii})z_i^2 + \sum_{i=1}^{n}\sum_{j\neq i}(-L_{ij})z_i z_j\\ \geq &\sum_{i=1}^{n} L_{ii}z_i^2 + \sum_{i=1}^{n}\sum_{j\neq i} L_{ij}|z_i||z_j|\\ = &|z|^T L|z| \geq 0,\end{aligned}$$

where the absolute value is elementwise. On the second line we use the inequalities $\alpha_i - L_{ii} \geq L_{ii}$ and for $j \neq i$, $-L_{ij}z_i z_j \geq L_{ij}|z_i||z_j|$, which follows since $L_{ij} \leq 0$.

In our algorithm described below, we will require that $\hat{L} \succ L$, i.e., $\hat{L} - L$ is positive definite. This can be accomplished by choosing

$$\alpha_i > 2L_{ii}, \quad i = 1, \ldots, n. \tag{4}$$

There are many other methods for selecting α, some of which have additional properties. For example, we can choose $\alpha = 2\lambda_{\max}(L)\mathbf{1}$, where $\lambda_{\max}(L)$ denotes the maximum eigenvalue of L. With this choice we have $\hat{L} = 2\lambda_{\max}(L)I$.

This diagonal majorization has all diagonal entries equal, i.e., it is a multiple of the identity.

Another choice (that we will encounter later in Section V-B) takes $\hat{L}$ to be a block diagonal matrix, conformal with the partition of x, with each block component a (possibly different) multiple of the identity,

$$\hat{L} = \operatorname{diag}(\alpha_1 I_{n_1}, \ldots, \alpha_p I_{n_p}), \tag{5}$$

where we can take

$$\alpha_i > \max_{j \in N_i} 2L_{jj}, \quad i = 1, \ldots, p,$$

where N_i is the index range for block i.

IV. Distributed Majorization-Minimization Algorithm

The majorization-minimization (MM) algorithm is an iterative algorithm that at each step minimizes a majorizer of the original function at the current iterate [9]. Since $\hat{\mathcal{L}}$, as constructed in Section III, using (4), majorizes $\mathcal{L}$, it follows that $\hat{F} = f + \hat{\mathcal{L}}$ majorizes $F = f + \mathcal{L}$. The MM algorithm for minimizing F is then

$$x^{k+1} = \arg\min_x \left(f(x) + \hat{\mathcal{L}}(x; x^k) \right), \tag{6}$$

where the superscripts k and $k+1$ denote the iteration counter. Note that since $\hat{L}$ is positive definite, $\hat{\mathcal{L}}$ is strictly convex in x, so the argmin is unique.

a) Stopping criterion: The optimality condition for the update (6) is the existence of $g^{k+1} \in \mathbb{R}^n$ with

$$g^{k+1} \in \partial f(x^{k+1}), \quad g^{k+1} + \nabla \hat{\mathcal{L}}(x^{k+1}; x^k) = 0. \tag{7}$$

From $\hat{\mathcal{L}}(x; z) - \mathcal{L}(x) = (1/2)(x - z)^T(\hat{L} - L)(x - z)$, we have

$$\nabla \hat{\mathcal{L}}(x^{k+1}; x^k) - \nabla \mathcal{L}(x^{k+1}) = (\hat{L} - L)(x^{k+1} - x^k).$$

Substituting this into (7) we get

$$g^{k+1} + \nabla \mathcal{L}(x^{k+1}) = (\hat{L} - L)(x^k - x^{k+1}). \tag{8}$$

Thus we see that

$$r^{k+1} = (\hat{L} - L)(x^k - x^{k+1})$$

is the optimality residual for x^{k+1}, i.e., the right-hand side of (2). We will use $\|r^{k+1}\|_2 \le \epsilon$, where $\epsilon > 0$ is a tolerance, as our stopping criterion. This guarantees that on exit, x^{k+1} satisfies the optimality condition (2) within ϵ.

b) Absolute and relative tolerance: When the algorithm is used to solve problems in which $x^\star$ or L vary widely in size, the tolerance ϵ is typically chosen as a combination of an absolute error ϵ_{abs} and a relative error ϵ_{rel}, for example,

$$\epsilon = \epsilon_{\text{abs}} + \epsilon_{\text{rel}}(\|\hat{L} - L\|_F + \|x\|_2),$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

c) Distributed implementation: The update (6) can be broken down into two steps. The first step requires multiplying by L, and in the other step, we carry out p independent minimizations in parallel. We partition the Laplacian matrix L into blocks $L_{ij}, i, j = 1, \ldots, p$, conformal with the partition $x = (x_1, \ldots, x_p)$. (In a few places above, we used L_{ij} to denote the i, j entry of L, whereas here we use it to denote the i, j submatrix. This slight abuse of notation should not cause any confusion since the index ranges, and the dimensions, make it clear whether the entry, or submatrix, is meant.) We then observe that our majorizer (3) has the form

$$\hat{\mathcal{L}}(x; z) = \sum_{i=1}^p \hat{\mathcal{L}}_i(x_i; z) + c,$$

where c does not depend on x, and

$$\hat{\mathcal{L}}_i(x_i; z) = (1/2)(x_i - z_i)^T \hat{L}_{ii}(x_i - z_i) + h_i^T x_i,$$

where z_i refers to the ith subvector of z, and h_i is the ith subvector Lz,

$$h_i = L_{ii} z_i + \sum_{j \neq i} L_{ij} z_j.$$

It follows that

$$\hat{F}(x; x^k) = \sum_{i=1}^p (f_i(x_i) + \hat{\mathcal{L}}(x_i; x_i^k)) + c$$

is block separable.

Algorithm 1 Distributed majorization-minimization.

```
given Laplacian matrix $L$, and initial starting point $x^0$
in the feasible set of the problem, with $f(x^0) < \infty$.
Form majorizer matrix. Form diagonal $\hat{L}$ with $\hat{L} \succ L$
(using (4)).
for $k = 1, 2, \ldots$
    1. Compute linear term. Compute $h^k = Lx^k$.
    2. Update in parallel. For $i = 1, \ldots, p$, update
    each $x_i$ (in parallel) as
    $x_i^{k+1} =$
      $\arg\min_{x_i} \left( f_i(x_i) + (1/2)(x_i - x_i^k)^T \hat{L}_{ii}(x_i - x_i^k) + (h_i^k)^T x_i \right)$.
    Compute residual $r^{k+1} = (\hat{L} - L)(x^k - x^{k+1})$.
    3. Test stopping criterion. Quit if $k \ge 2$ and
    $\|r^{k+1}\|_2 \le \epsilon$.
```

Step 1 couples the subvectors x_i^k; step 2 (the subproblem updates) is carried out in parallel for each i. We observe that the updates in step 2 are (diagonally scaled) proximal operator evaluations, i.e., they involve minimizing f_i plus a norm squared term, with diagonal quadratic norm; see, e.g., [36]. Our algorithm can thus be considered as a distributed proximal-based method, a method which has been extensively researched in signal processing [38], [39]. We also mention that as the algorithm converges (discussed in detail below), $x_i^{k+1} - x_i^k \to 0$, which implies that the quadratic terms $(1/2)(x_i - x_i^k)^T \hat{L}_{ii}(x_i - x_i^k)$ and their gradients in the update asymptotically vanish; roughly speaking, they "go away" as the algorithm converges. We will see below, however, that these quadratic terms are critical to convergence of the algorithm.

d) Warm start: Our algorithm supports *warm starting* by choosing the initial point x^0 as an estimate of the solution, for example, the solution of a closely related problem (e.g., a problem with slightly varying hyperparameters.) Warm starting can decrease the number of iterations required to converge [40], [41]; we will see an example in Section V.

A. Convergence

There are many general convergence results for MM methods, but all of them require varying additional assumptions about the objective function [8], [9]. Additionally, the MM algorithmic framework does not admit a general convergence rate, as convergence rates depend heavily on the choice of majorizer [11], [42], [43]. In this section we give a complete self-contained proof of convergence for our algorithm, that requires no additional assumptions. We will show that $F(x^k) - F^\star \to 0$, as $k \to \infty$, and also that the stopping criterion eventually holds, i.e., $(\hat{L} - L)(x^k - x^{k+1}) \to 0$.

We first observe a standard result that holds for all MM methods: the objective function is non-increasing. We have

$$F(x^{k+1}) \leq \hat{F}(x^{k+1}; x^k) \leq \hat{F}(x^k; x^k) = F(x^k),$$

where the first inequality holds since $\hat{F}$ majorizes F, and the second since x^{k+1} minimizes $\hat{F}(x; x^k)$ over x. It follows that $F(x^k)$ converges, and therefore $F(x^k) - F(x^{k+1}) \to 0$. It also follows that the iterates x^k are bounded, since every iterate satisfies $F(x^k) \leq F(x^0)$, and we assume that the sublevel sets of F are bounded.

Since F is convex and $g^{k+1} + \nabla\mathcal{L}(x^{k+1}) \in \partial F(x^{k+1})$, we have (from the definition of subgradient)

$$F(x^k) \geq F(x^{k+1}) + (g^{k+1} + \nabla\mathcal{L}(x^{k+1}))^T(x^k - x^{k+1}).$$

Using this and (8), we have

$$F(x^k) - F(x^{k+1}) \geq (x^k - x^{k+1})^T(\hat{L} - L)(x^k - x^{k+1}).$$

Since $F(x^k) - F(x^{k+1}) \to 0$ as $k \to \infty$, and $\hat{L} - L \succ 0$, we conclude that $x^{k+1} - x^k \to 0$ as $k \to \infty$. This implies that our stopping criterion will eventually hold.

Now we show that $F(x^k) \to F^\star$. Let $x^\star$ be any optimal point. Then,

$$\begin{aligned} F^\star = F(x^\star) &\geq F(x^{k+1}) \\ &\quad + ((\hat{L} - L)x^k - \hat{L}x^{k+1} + Lx^{k+1})^T(x^\star - x^{k+1}) \\ &= F(x^{k+1}) + (x^k - x^{k+1})^T(\hat{L} - L)(x^\star - x^{k+1}). \end{aligned}$$

So we have

$$F(x^{k+1}) - F^\star \leq -(x^k - x^{k+1})^T(\hat{L} - L)(x^\star - x^{k+1}).$$

Since $x^k - x^{k+1} \to 0$ as $k \to \infty$, and x^{k+1} is bounded, the right-hand side converges to zero as $k \to \infty$, and so we conclude $F(x^{k+1}) - F^\star \to 0$ as $k \to \infty$.

B. Variations

a) Arbitrary convex quadratic regularization While our interest is in the case when $\mathcal{L}$ is Laplacian regularization, the algorithm (and convergence proof) work when $\mathcal{L}$ is any convex quadratic, i.e., $L \succeq 0$, with the choice

$$\alpha_i > \sum_{j=1}^{n} |L_{ij}|, \quad i = 1, \ldots, n,$$

replacing the condition (4). In fact, the condition (4) is a special case of this condition, for a Laplacian matrix.

b) Nonconvex f If the objective function in the LRMP is nonconvex, i.e., f is nonconvex, then the method proposed in this paper can be extended as a heuristic for solving (1) for nonconvex f. It is emphasized that the most the algorithm can guarantee is a local optimum, rather than a global optimum [33].

V. Examples

In this section we describe two applications of our distributed method for solving LRMP, and report numerical results demonstrating its convergence and performance. We run all numerical examples on a 32-core AMD machine with 64 hyperthreads, using the Pathos multiprocessing package to carry out computations in parallel [44]. Our code is available online at https://github.com/cvxgrp/mm_dist_lapl.

A. Multi-period Portfolio Optimization

We consider the problem of multi-period trading with quadratic transaction costs; see [1], [45] for more detail. We are to choose a portfolio of n holdings $x_t \in \mathbb{R}^n$, for periods $t = 1, \ldots, T$. We assume the nth holding is a riskless holding (i.e., cash). We choose the portfolios by solving the problem

$$\text{minimize} \quad \sum_{t=1}^{T}(f_t(x_t) + \frac{1}{2}(x_t - x_{t-1})^T D_t(x_t - x_{t-1})), \tag{9}$$

where f_t is the convex objective function (and constraints) for the portfolio in period t, and the D_t's are diagonal positive definite matrices. The initial portfolio x_0 is given and constant; $x_1, \ldots, x_T$ are the variables. The quadratic term $(1/2)(x_t - x_{t-1})^T D_t(x_t - x_{t-1})$ is the transaction cost, i.e., the additional cost of trading to move from the previous portfolio x_{t-1} to the current one x_t. We will assume that there is no transaction cost associated with cash, i.e., $(D_t)_{nn} = 0$.

The objective function f_t typically includes negative expected return, one or more risk constraints or risk avoidance terms, shorting or borrow costs, and possibly other terms. It also can include constraints, such as the normalization $\mathbf{1}^T x_t = 1$ (in which case x_t are referred to as the portfolio weights), limits on the holdings or the leverage of the portfolio, or a specified final portfolio; see [1], [45] for more detail.

We can express the transaction cost as Laplacian regularization on $x = (x_1, \ldots, x_T) \in \mathbb{R}^{Tn}$, plus a quadratic term involving x_1,

$$\begin{aligned}\sum_{t=1}^{T}\frac{1}{2}(x_t - x_{t-1})^T D_t(x_t - x_{t-1}) &= \frac{1}{2}x^T Lx + +\frac{1}{2}x_1^T D_1 x_1 \\ &\quad - (D_1 x_0)^T x_1 + +\frac{1}{2}x_0^T D_1 x_0.\end{aligned}$$

(Recall that the initial portfolio x_0 is given.) The Laplacian matrix L has block-tridiagonal form given by

$$L = \begin{bmatrix} D_2 & -D_2 & \dots & 0 & 0 \\ -D_2 & D_2 + D_3 & \dots & 0 & 0 \\ 0 & -D_3 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & D_{T-2} + D_{T-1} & 0 \\ 0 & 0 & \dots & -D_{T-1} + D_T & -D_T \\ 0 & 0 & \dots & -D_T & D_T \end{bmatrix}.$$

If we assume that the initial portfolio is cash, i.e., x_0 is zero except in the last component, then two of the three extra terms, $(D_1x_0)^T x_1$ and $(1/2)x_0^T D_1 x_0$, both vanish. If we lump the extra terms that depend on x_1 into f_1, the multi-period portfolio optimization problem (9) has the LRMP form, with $p = T$ and $n_i = n$. The total number of (scalar) variables is Tn. The graph associated with the Laplacian is the simple chain graph; roughly speaking, each portfolio x_t is linked to its predecessor x_{t-1} and its successor x_{t+1} by the transaction cost.

We can give a simple interpretation for the subproblem update in our method. The quadratic term of the subproblem update (which asymptotically goes away as we approach convergence) adds diagonal risk; the linear term h_t contributes an expected return to each asset. These additional risk and return terms come from both the preceding and the successor portfolios; they "encourage" the portfolios to move towards each other from one time period to the next, so as to reduce transaction cost. Each subproblem update minimizes negative risk-adjusted return, with the given return vector modified to encourage less trading.

1) Problem instance: We consider a problem with $n = 1000$ assets and $T = 30$ periods, so the total number of (scalar) variables is 30 000. The objective functions f_t include a negative expected return, a quadratic risk given by a factor (diagonal plus low rank) model with 50 factors [1], [46], and a linear shorting cost. We additionally impose the normalization constraint $\mathbf{1}^T x_t = 1$, so the portfolios x_t represent weights. The objective functions f_t have the form

$$f_t(x) = -\mu_t^T x + \gamma x^T \Sigma_t x + s_t^T (x)_-, \quad t = 1, \dots, T-1. \tag{10}$$

Here, $\gamma > 0$ is the *risk aversion parameter*, μ_t is the expected return, Σ_t is the return covariance, s_t is the (positive) shorting cost coefficient vector, and $(x)_- = \max\{-x, 0\}$, which is taken elementwise. The covariance matrices Σ_t are diagonal plus a rank 50 (factor) term, with zero entries in the last row and column (which correspond to the cash asset). We choose all these coefficients and the diagonal transaction cost matrices D_t randomly, but with realistic values. In our problem instance, we choose all of these parameters independent of t, i.e., constant.

We take f_T to be the indicator function for the constraint $x = e_n$ (i.e., $f_T(x) = 0$ if $x = e_n$, and ∞ otherwise), and the initial portfolio is all cash, $x_0 = e_n$. So in our multi-period portfolio optimization problem we are planning a sequence of portfolios that begin and end in cash.

We can see the interpretation of the subproblem updates in Section V-A by looking at the subproblem objective functions. Assuming we choose the diagonal elements of $\hat{L}$ to be $3L_{ii}$, we can rewrite the subproblem objective function (at time periods $t = 2, \dots, T-1$ and iteration k) as

$$\begin{aligned} &x_t^T\left(\gamma\Sigma_t + \frac{3}{2}(D_t + D_{t+1})\right)x_t \\ &\quad - (\mu_t + D_t(2x_t^k - x_{t-1}^k) + D_{t+1}(2x_t^k - x_{t+1}^k))^T x_t + c, \end{aligned}$$

where c is some constant that does not depend on x_t. We see that a diagonal risk term is added, and the mean return μ_t is offset by terms that depend on the past, current, and future portfolios x_{t-1}^k, x_t^k, and x_{t+1}^k.

2) Numerical results: We first solve the problem instance using CVXPY [47] and solver OSQP [48], which is single-thread. The solve time for this baseline method was 120 minutes.

We then solved the problem instance using our method. We initialized all portfolios as e_n, i.e., all cash, and use stopping criterion tolerance $\epsilon = 10^{-6}$. Our algorithm took 8 iterations and 19 seconds to converge, and produced a solution that agreed very closely with the CVXPY/OSQP solution. Fig. 1 shows a plot of the residual norm $\|r^k\|_2$ versus iteration k. This plot shows nearly linear convergence, with a reduction in residual norm by around a factor of 5 each iteration.

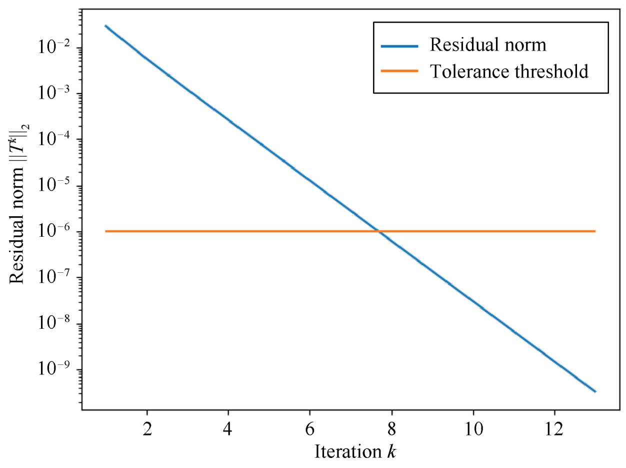


Fig. 1. Residual norm versus iteration for multi-period portfolio optimization problem.

B. Laplacian Regularized Estimation

We consider estimation of parameters in a statistical model. We have a graph, with some data associated with each node; the goal is to fit a model to the data at each node, with Laplacian regularization used to make neighboring models similar.

The model parameter at node i is $\theta_i \in \mathbb{R}^{n_i}$. The vector of all node parameters is $\theta = (\theta_1, \dots, \theta_p) \in \mathbb{R}^n$, with $n = n_1 + \cdots + n_p$. We choose θ by minimizing a local loss function and regularizer at each node, plus Laplacian regularization:

$$\text{minimize} \quad \sum_{i=1}^{p} f_i(\theta_i) + \mathcal{L}(\theta),$$

where $f_i(\theta_i) = \ell_i(\theta_i) + r_i(\theta_i)$, where $\ell_i : \mathbb{R}^{n_i} \to \mathbb{R}$ is the loss function (for example, the negative log-likelihood of θ_i) for the data at node i, and $r_i : \mathbb{R}^{n_i} \to \mathbb{R}$ is a regularizer on the parameter θ_i. Without the Laplacian term, the problem is separable, and corresponds to fitting each parameter separately by minimizing the local loss plus regularizer. The Laplacian term is an additional regularizer that encourages various entries in the parameter vectors to be close to each other.

a) Laplacian regularized covariance estimation:

We now focus on a more specific case of this general problem, Laplacian regularized covariance estimation. At each node, we have some number of samples from a zero-mean Gaussian distribution on $\mathbb{R}^d$, with covariance matrix Σ_i, assumed positive definite. We will estimate the natural parameters (as an exponential family), the inverse covariance matrices $\theta_i = \Sigma_i^{-1}$. So here we take the node parameters θ_i to be symmetric positive definite $d \times d$ matrices, with $n_i = d(d+1)/2$. (In the discussion of the general case above, θ_i is a vector in $\mathbb{R}^{n_i}$; in the rest of this section, θ_i will denote a symmetric $d \times d$ matrix.)

The data samples at node i have empirical covariance S_i (which is not positive definite if there are fewer than d samples). The negative log-likelihood for node i is (up to a constant and a positive scale factor)

$$\ell_i(\theta_i) = \mathrm{Tr}(S_i \theta_i) - \log\det \theta_i.$$

We use trace regularization on the parameter,

$$r_i(\theta_i) = \kappa \mathrm{Tr}(\theta_i),$$

where $\kappa > 0$ is the local regularization hyperparameter. We note that we can minimize $f_i(\theta_i) = \ell_i(\theta_i) + r_i(\theta_i)$ analytically; the minimizer is

$$\theta_i = (S_i + \kappa I)^{-1}.$$

(See, e.g., [27].)

The Laplacian regularization is used to encourage neighboring inverse covariance matrices in the given graph to be near each other. It has the specific form

$$\mathcal{L}(\theta_1, \dots, \theta_p) = \lambda \sum_{(i,j) \in \mathcal{E}} \|\theta_i - \theta_j\|_F^2 = \mathrm{Tr}(\theta^T L \theta),$$

where the norm is the Frobenius norm, L is the associated weighted Laplacian matrix for the graph with vertices $1, \dots, p$ and edges $\mathcal{E}$, and $\lambda \geq 0$ is a hyperparameter that controls the amount of Laplacian regularization. When $\lambda = 0$, the estimation problem is separable, with analytical solution

$$\theta_i = (S_i + \kappa I)^{-1}, \quad i = 1, \dots, p.$$

For $\lambda \to \infty$, assuming the graph is connected, the estimation problem reduces to finding a single covariance matrix for all the data.

We choose the majorizer to be block diagonal with each block a multiple of the identity, as in (5). The update at each node in our algorithm can be expressed as minimizing over θ_i the function

$$\mathrm{Tr}((S_i + H_i^k)\theta_i) - \log\det \theta_i + \kappa \mathrm{Tr}(\theta_i) + \frac{\alpha_i}{2}\|\theta_i - \theta_i^k\|_F^2,$$

where

$$H^k = L\theta^k.$$

As an aside, we note that the optimal θ_i can be rewritten as

$$\begin{aligned}&\arg\min_{\theta_i}(-\log\det \theta_i \\ &\quad + \frac{\alpha_i}{2}\|\theta_i - (\theta_i^k - S_i - H_i^k - \kappa I)\|_F^2),\end{aligned}$$

i.e., the proximal operator of $-\log\det$ with parameter $1/\alpha_i$.

This minimization can be carried out analytically. By taking the gradient of the subproblem objective function with respect to θ_i and equating to zero, we see that

$$S_i + H_i^k - \theta_i^{-1} + \kappa I + \alpha_i(\theta_i - \theta_i^k) = 0,$$

or

$$\theta_i^{-1} - \alpha_i \theta_i = S_i + H_i^k + \kappa I - \alpha_i \theta_i^k.$$

This implies that θ_i and $S_i + H_i^k + \kappa I - \alpha_i \theta_i^k$ share the same eigenvectors [2], [26], [49]. Let $Q_i \Lambda_i Q_i^T$ be the eigenvector decomposition of $S_i + H_i^k + \kappa I - \alpha_i \theta_i^k$. We find that the eigenvalues of $\theta_i, v_{ij}, j = 1, \dots, n$, are

$$v_{ij} = \frac{1}{2\alpha_i}\left(-(\Lambda_i)_{jj} + \sqrt{(\Lambda_i)_{jj}^2 + 4\alpha_i}\right).$$

We have $\theta_i^{k+1} = Q_i V_i Q_i^T$, where $V_i = \mathrm{diag}(v_{i1}, \dots, v_{in})$. The computational cost per iteration is primarily in computing the eigenvector decomposition of $S_i + H_i^k + \kappa I - \alpha_i \theta_i^k$, which has order d^3.

1) Problem instance: The graph is a 15×15 grid, with 420 edges, so $p = 225$. The dimension of the data is $d = 30$, so each θ_i is a symmetric 30×30 matrix. The total number of (scalar) variables in our problem instance is $225 \times 30(30 + 1)/2 = 104\,625$.

We generate the data for each node as follows. First, we choose the four corner covariance matrices randomly. The other 221 nodes are given covariance matrices using bilinear interpolation from the corner covariance matrices. We then generate 20 independent samples of data from each of the node distributions. (The samples are in $\mathbb{R}^{30}$, so the empirical covariance matrices are singular.) In our problem instance we used hyperparameter values $\lambda = 0.053$ and $\kappa = 0.08$, which were chosen to give good estimation performance.

2) Numerical results: The problem instance is too large to reliably solve using CVXPY and the solver SCS [50], which stops after two hours with the status message that the computed solution may be inaccurate.

We solved the problem using our distributed method, with absolute tolerance $\epsilon_{\text{abs}} = 10^{-5}$ and relative tolerance $\epsilon_{\text{rel}} = 10^{-3}$. The method took 54 iterations and 13 seconds to converge. Fig. 2 is a plot of the residual norm $\|r^k\|_F$ versus iteration k.

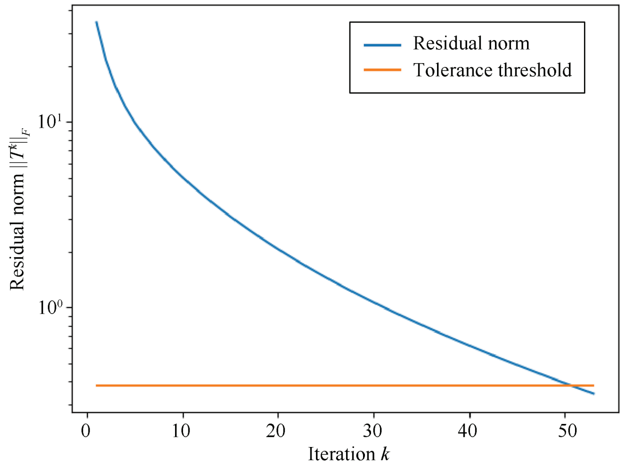


Fig. 2. Residual norm vs. iteration for Laplacian regularized covariance estimation problem.

a) Regularization path via warm-start: To illustrate the advantage of warm-starting our algorithm, we compute the entire regularization path, i.e., the solutions of the problem for 100 values of λ, spaced logarithmically between 10^{-5} and 10^4.

Computing these 100 estimates by running the algorithm for each value of λ sequentially, without warm-start, requires 26 000 total iterations (an average of 260 iterations per choice of λ) and 81 minutes. Computing these 100 estimates by running the algorithm using warm-start, starting from $\lambda = 10^{-5}$, requires only 2000 total iterations (an average of 20 iterations per choice of λ) and 7.1 minutes. For the specific instance solved above, the algorithm converges in only 2.5 seconds and 10 iterations using warm-start, compared to 13 seconds and 54 iterations using cold-start.

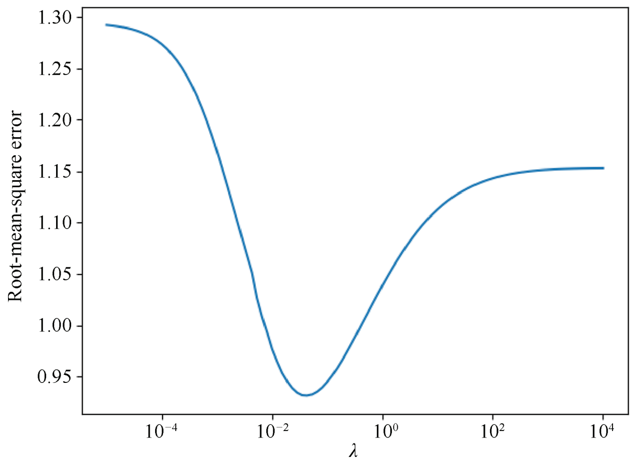


Fig. 3. Root-mean-square error of the optimal estimates vs. λ.

While the point of this example is the algorithm that computes the estimates, we also explore the performance of the method. For each of the 100 values of λ we compute the root-mean-square error between our estimate of the inverse covariance and the true inverse covariance, which we know, since we generated them. Fig. 3 shows a plot of the root-mean-square error of our estimate versus the value of λ. This plot shows that the method works, i.e., produces better estimates of the inverse covariance matrices than handling them separately (small λ) or fitting one inverse covariance matrix for all nodes (large λ).

Acknowledgments

The authors would like to thank Peter Stoica for his insights and comments on early drafts of this paper. We would also like to thank the Air Force Research Laboratory, and in particular Muralidhar Rangaswamy, for discussions of covariance estimation arising in radar signal processing.

References

[1] S. Boyd, E. Busseti, S. Diamond, R. N. Kahn, K. Koh, P. Nystrup, and J. Speth, “Multi-period trading via convex optimization,” *Foundations and Trends in Optimization*, vol. 3, no. 1, pp. 1− 76, Apr. 2017.

[2] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, “Network inference via the time-varying graphical lasso,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 205−213, 2017.

[3] M. Yin, J. Gao, and Z. Lin, “Laplacian regularized low-rank representation and its applications,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 3, pp. 504−517, Mar. 2016.

[4] J. Pang and G. Cheung, “Graph laplacian regularization for image denoising: Analysis in the continuous domain,” *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1770− 1785, April 2017. [Online]. Available: https://doi.org/10.1109/ TIP.2017.2651400

[5] D. Slepcev and M. Thorpe, “Analysis of p-laplacian regularization in semi-supervised learning,” *ArXiV preprint*, Oct. 2017.

[6] R. K. Ando and T. Zhang, “Learning on graph with Laplacian regularization,” *Conference on Neural Information Processing Systems*, 2006.

[7] S. Melacci and M. Belkin, “Laplacian support vector machines trained in the primal,” *Journal of Machine Learning Research*, vol. 12, pp. 1149−1184, Jul. 2011.

[8] K. Lange, *MM Optimization Algorithms*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2016.

[9] Y. Sun, P. Babu, and D. Palomar, “Majorization-minimization algorithms in signal processing, communications, and machine learning.”*IEEE Transactions in Signal Processing*, vol. 65, no. 3, pp. 794−816, 2017.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the Alternating Direction Method of Multipliers,” *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1−122, Jan. 2011.

[11] J. de Leeuw and K. Lange, “Sharp quadratic majorization in one dimension,” *Computational Statistics & Data Analysis*, vol. 53, no. 7, pp. 2471−2484, 2009. [Online]. Available: https://doi.org/10.1016/j.csda.2009.01.002

[12] C. Zhang, D. Florencio, and P. Chou, “Graph signal processing−a probabilistic framework,” Tech. Rep., April 2015. [Online]. Available: https: //www. microsoft. com/ en-us/ research/ publication/ graph-signal-processing-a-probabilistic-framework/

[13] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, “Learning laplacian matrix in smooth graph signal representations,” *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160−6173, Dec. 2016. [Online]. Available: https://doi.org/10.1109/TSP.2016.2602809

[14] H. E. Egilmez, E. Pavez, and A. Ortega, “Graph learning from data under laplacian and structural constraints,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 825−841, Sept 2017.

[15] C. Godsil and G. Royle, *The Laplacian of a Graph*. Springer, 2001.

[16] K. Q. Weinberger, F. Sha, Q. Zhu, and L. K. Saul, “Graph Laplacian regularization for large-scale semidefinite programming,” in *Advances in Neural Information Processing Systems 19*. MIT Press, 2007, pp. 1489−1496.

[17] M. Razaviyayn, M. Hong, and Z. Luo, “A unified convergence analysis of block successive minimization methods for nonsmooth optimization,” *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 1126−1153, 2013.

[18] D. Hallac, C. Wong, S. Diamond, A. Sharang, R. Sosic, S. Boyd, and J. Leskovec, "SnapVX: A network-based convex optimization solver." *Journal of Machine Learning Research*, vol. 18, 2017.

[19] D. R. Hunter and K. Lange, "A tutorial on mm algorithms," *The American Statistician*, vol. 58, no. 1, pp. 30−37, 2004. [Online]. Available: https://doi.org/10.1198/0003130042836

[20] M. W. Jacobson and J. A. Fessler, "An expanded theoretical treatment of iteration-dependent majorize-minimize algorithms," *IEEE Transactions on Image Processing*, vol. 16, no. 10, pp. 2411−2422, Oct 2007.

[21] A. L. Yuille and A. Rangarajan, "The concave-convex procedure," *Neural Computing*, vol. 15, no. 4, pp. 915−936, Apr. 2003.

[22] T. T. Wu and K. Lange, "The MM alternative to EM," *Statistical Science*, vol. 25, no. 4, pp. 492−505, 11 2010.

[23] R. Almgren and N. Chriss, "Optimal execution of portfolio transactions," *Journal of Risk*, pp. 5−39, 2000.

[24] J. Skaf and S. Boyd, "Multi-period portfolio optimization with constraints and transaction costs," 2009.

[25] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432−441, Aug. 2008.

[26] P. Danaher, P. Wang, and D. Witten, "The joint graphical lasso for inverse covariance estimation across multiple classes," *Journal of the Royal Statistical Society*, vol. 76, no. 2, pp. 373−397, 2014.

[27] O. Banerjee, L. El Ghaoui, and A. d'Aspremont, "Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data," *Journal of Machine Learning Research*, vol. 9, pp. 485−516, 2008.

[28] I. Soloveychik and A. Wiesel, "Joint estimation of inverse covariance matrices lying in an unknown subspace," *IEEE Transactions on Signal Processing*, vol. 65, no. 9, pp. 2379−2388, 2017.

[29] E. Levitan and G. T. Herman, "A maximum a posteriori probability expectation maximization algorithm for image reconstruction in emission tomography," *IEEE Transactions on Medical Imaging*, vol. 6, no. 3, pp. 185−192, Sept 1987.

[30] R. T. Rockafellar, *Convex Analysis, ser. Princeton Mathematical Series*. Princeton University Press, 1970.

[31] J. M. Borwein and A. S. Lewis, *Convex Analysis and Nonlinear Optimization, Theory and Examples*. Springer, 2000.

[32] L. C. Evans, *Partial differential equations.* Providence, R.I.: American Mathematical Society, 2010.

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

[34] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 2006.

[35] F. H. Clarke, *Optimization and Nonsmooth Analysis*. Society for Industrial and Applied Mathematics, 1990.

[36] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127−239, Jan. 2014.

[37] B. Bollobas, *Modern Graph Theory*, ser. Graduate Texts in Mathematics. Heidelberg: Springer, 1998.

[38] P. Combettes and J. Pesquet, *Proximal Splitting Methods in Signal Processing*. New York, NY: Springer New York, 2011, pp. 185−212. [Online]. Available: https://doi.org/10.1007/978-1-4419-9569-8_10.

[39] M. Burger, A. Sawatzky, and G. Steidl, *First Order Algorithms in Variational Image Processing*. Cham: Springer International Publishing, 2016, pp. 345−407. [Online]. Available: https: //doi.org/10.1007/978-3-319-41589-5_10.

[40] E. Yildirim and S. Wright, "Warm-start strategies in interior-point methods for linear programming," *SIAM Journal on Optimization*, vol. 12, no. 3, pp. 782−810, 2002. [Online]. Available: https://doi.org/10.1137/S1052623400369235

[41] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, pp. 267−278, March 2010.

[42] D. Kim, D. Pal, J.-B. Thibault, and J. A. Fessler, "Accelerating ordered subsets image reconstruction for X-ray CT using spatially non-uniform optimization transfer," *IEEE Transactions on Medical Imaging*, vol. 32, no. 11, pp. 1965−78, Nov. 2013.

[43] M. J. Muckley, D. C. Noll, and J. A. Fessler, "Fast parallel MR image reconstruction via B1-based, adaptive restart, iterative soft thresholding algorithms (BARISTA)," *IEEE Transactions on Medical Imaging*, vol. 34, no. 2, pp. 578−88, Feb. 2015.

[44] M. McKerns, "Pathos multiprocessing," July 2017. [Online]. Available: https: //pypi.python.org/pypi/pathos

[45] S. Boyd, M. T. Mueller, B. O'Donoghue, and Y. Wang, "Performance bounds and suboptimal policies for multi-period investment," *Foundations and Trends in Optimization*, vol. 1, no. 1, pp. 1−72, Jan. 2014.

[46] G. Chamberlain and M. Rothschild, "Arbitrage, factor structure, and mean-variance analysis on large asset markets," *Econometrica*, vol. 51, no. 5, pp. 1281−1304, 1983. [Online]. Available: http://www.jstor.org/stable/1912275

[47] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1−5, 2016.

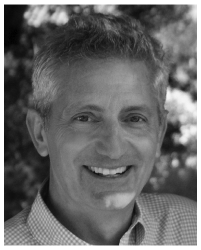
[48] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *ArXiv e-prints*, Nov. 2017.

[49] D. M. Witten and R. Tibshirani, "Covariance-regularized regression and classification for high dimensional problems," *Journal of the Royal Statistical Society*, vol. 71, no. 3, pp. 615−636, 2009.

[50] B. O' Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *Journal of Optimization Theory and Applications*, vol. 169, no. 3, pp. 1042−1068, June 2016.

Jonathan Tuck is a Ph.D. candidate at Stanford University in the Electrical Engineering Department. He received his B.S. degree in electrical engineering from the Georgia Institute of Technology in 2016. His research interests include distributed convex optimization algorithms in machine learning, signal processing, and finance.

David Hallac is a Ph.D. candidate at Stanford University in the Electrical Engineering Department. He received his M.S. degree from Stanford in 2015 and his B.S. degree from the University of Pennsylvania in 2013. His research interests include distributed convex optimization and machine learning on time series data.



Stephen Boyd (F'99) is the Samsung Professor of Engineering, and Professor of Electrical Engineering in the Information Systems Laboratory at Stanford University, with courtesy appointments in Computer Science and Management Science and Engineering. He received the A.B. degree in mathematics from Harvard University in 1980, and the Ph.D. in electrical engineering and computer science from the University of California, Berkeley, in 1985, and then joined the faculty at Stanford. His current research focus is on convex optimization applications in control, signal processing, machine learning, finance, and circuit design.