MATHEMATICS OF COMPUTATION
Volume 84, Number 296, November 2015, Pages 2791–2822
http://dx.doi.org/10.1090/mcom/2972
Article electronically published on May 28, 2015

A TIME CONTINUATION BASED FAST APPROXIMATE ALGORITHM FOR COMPRESSED SENSING RELATED OPTIMIZATION

FARZIN BAREKAT AND STANLEY OSHER

Abstract. In this paper we introduce a fast approximate algorithm to optimize an augmented version of the Basis Pursuit problem and subsequently find the solution to the compressed sensing problem. Our methodology is to first solve the Lagrangian dual formulation of the problem and then use the result to find an approximate solution to the primal problem. Although we emphasize that our algorithm finds an approximate solution, numerical experiments show that our algorithm perfectly recovers the solution when the solution is relatively sparse with respect to the number of measurements. In these scenarios, the recovery is extremely fast compared to other available methods. Numerical experiments also demonstrate that the algorithm exhibits a sharp phase transition in success rate of recovery of the solution to compressed sensing problems as sparsity of solution varies. The algorithm proposed here is parameter free (except a tolerance parameter due to numerical machine precision), and very easy to implement.

1. Introduction

The goal of compressed sensing is to find sparse solutions to equation $\mathbf{A}y = \mathbf{b}$, where $\mathbf{A}$ is an $m \times n$ matrix with $m \ll n$. The compressed sensing problem is formulated as

$$\underset{y\in\mathbb{R}^n}{\operatorname{argmin}} \|y\|_0 \quad \text{subject to} \quad \mathbf{A}y = \mathbf{b}, \tag{1.1}$$

where $\|y\|_0$ counts the number of nonzero entries in y.

It turns out (i.e. see [5]) that the sparse solution is often the same as the solution to the following minimization problem, also known as Basis Pursuit (i.e., see for example [6]),

$$\underset{y\in\mathbb{R}^n}{\operatorname{argmin}} \|y\|_1 \quad \text{subject to} \quad \mathbf{A}y = \mathbf{b}. \tag{1.2}$$

The objective function in (1.2) is nonsmooth, which poses a numerical challenge for minimization. Moreover, the Lagrangian dual of (1.2),

$$\underset{\mathbf{s}\in\mathbb{R}^m}{\operatorname{argmax}} \langle \mathbf{b}, \mathbf{s}\rangle \quad \text{subject to} \quad -\overrightarrow{\mathbf{1}} \le \mathbf{A}^T\mathbf{s} \le \overrightarrow{\mathbf{1}} \tag{1.3}$$

Received by the editor February 1, 2014.
2010 *Mathematics Subject Classification.* Primary 49M29, 65K10, 90C25; Secondary 49L99.
Key words and phrases. Augmented ℓ^1 minimization, homotopy methods, compressed sensing, exact regularization property.
This research was supported by DOE DE-FG02-05ER25710:005, ONR N00014-11-1-0749, and ONR N00014-11-1-7-11.

is constrained and is not strictly convex. Problem (1.2) can be relaxed in several ways:

For $\mu > 0$, we can introduce a regularization term and consider problem

$$\underset{y\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ \|y\|_1 + \frac{1}{2\mu}|y|^2 \right\} \quad \text{subject to} \quad \mathbf{A}y = \mathbf{b}. \tag{1.4}$$

Throughout this paper, $|\cdot|$ stands for Euclidean norm $\|\cdot\|_2$. It has been shown in [24] and [11] that the above problem has the *exact regularization property*: there exist large fixed μ_0, depending on $\mathbf{A}$ and $\mathbf{b}$, such that for $\mu > \mu_0$, the solution to (1.4) is the same as the solution to (1.2). In [17] it is shown that the exact regularization property still holds if the polyhedral norm is used in problem (1.4) instead of the L_1 norm. Moreover, as noted in [13], the Lagrange dual problem of (1.4) is unconstrained and differentiable; consequently, many classical techniques such as Nesterov's acceleration method [15], Barzilai–Borwein step sizes [1], and nonmonotone line search can be used to speed up computation.

Alternatively, we can add a penalty term to problem (1.2) and consider the following problem (known in the literature as LASSO [18] or Basis Pursuit Denoising [7]):

$$\underset{y\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ \|y\|_1 + \frac{1}{2t}|\mathbf{A}y - \mathbf{b}|^2 \right\}. \tag{1.5}$$

For certain applications, $\mathbf{b}$ may contain noise; which makes solving (1.5) more preferable than solving (1.2).

The primal problem (1.5) is unconstrained; however, its Lagrangian dual problem is constrained. In contrast, the primal problem (1.4) is constrained, whereas, its Lagrangian dual problem is unconstrained.

Motivated by the two relaxation procedures above, we now turn to problem (also known as Augmented Lasso),

$$\underset{y\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ \|y\|_1 + \frac{1}{2\mu}|y|^2 + \frac{1}{2t}|\mathbf{A}y - \mathbf{b}|^2 \right\}, \tag{1.6}$$

and a related problem

$$\underset{y\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ \|y\|_1 + \frac{1}{2\mu}|y|^2 - \sqrt{t^2 - |\mathbf{A}y - \mathbf{b}|^2} \right\} \quad \text{subject to} \quad |\mathbf{A}y - \mathbf{b}|^2 \le t. \tag{1.7}$$

Primal problems (1.6) and (1.7) can be written in the general form

$$\underset{y\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ \|y\|_1 + \frac{1}{2\mu}|y|^2 + R(t,y) \right\}, \tag{1.8}$$

where $R(t,y)$ incorporates the penalty term (i.e. $R(t,y) = \frac{1}{2t}|\mathbf{A}y - \mathbf{b}|^2$ for problem (1.6) and $R(t,y) = -\sqrt{t^2 - |\mathbf{A}y - \mathbf{b}|^2}$ for problem (1.7)).

The Lagrangian dual of the above problems is of the form

$$\underset{\mathbf{s}\in\mathbb{R}^m}{\operatorname{argmax}} \left\{ \langle \mathbf{b}, \mathbf{s}\rangle - \frac{\mu}{2}|\mathsf{shrink}(\mathbf{A}^T\mathbf{s}, \overrightarrow{\mathbf{1}})|^2 - tH(|\mathbf{s}|) \right\}, \tag{1.9}$$

with $H(x) = x^2/2$ for problem (1.6), and $H(x) = \sqrt{1+x^2}$ for problem (1.7). Here *shrink* is the shrinkage operator (also known as soft-thresholding operator), which for two vectors with the same length $\vec{x}$ and $\vec{v}$, is defined by

$$\mathsf{shrink}(\vec{x}, \vec{v})_i = \max(|x_i| - v_i, 0)\frac{x_i}{|x_i|}.$$

Consequently, we can use the Lagrangian dual formulation to solve problems (1.6) and (1.7) in the following way (see for example [9]): First compute

$$\mathbf{s}^* = \underset{\mathbf{s}\in\mathbb{R}^m}{\operatorname{argmax}} \left\{ \langle \mathbf{b}, \mathbf{s}\rangle - \frac{\mu}{2}|\mathsf{shrink}(\mathbf{A}^T\mathbf{s}, \overrightarrow{\mathbf{1}})|^2 - tH(|\mathbf{s}|) \right\},$$

then compute

$$y^* = \mu \cdot \mathsf{shrink}(\mathbf{A}^T\mathbf{s}^*, \overrightarrow{\mathbf{1}}).$$

For

- $H(x) = \frac{1}{2}x^2$, then y^* solves (1.6).
- $H(x) = \sqrt{1+x^2}$, then y^* solves (1.7).

We note that the idea of using duality for (1.8) was first used in [24], for $t = 0$ and then in [8] to obtain formula (1.9). We note that there is an interesting link between this duality formulation and viscosity solutions for certain Hamilton-Jacobi equations, pointed out in [8] (see also [9]). We also note that [3] developed a general framework to recover sparse data by first smoothing the dual of the conic formulation of the problem, and then applying optimal first-order methods.

In this paper we propose a new and practical approximate algorithm that uses the duality correspondence between (1.8) and (1.9) to find an approximate solution to the compressed sensing problem. We name our algorithm *Time Continuation Compressed Sensing*, or TCCS for short. The TCCS algorithm consists of two parts. The first part of the algorithm finds an approximate solution to the constrained problem

$$\underset{\mathbf{s}\in\mathbb{R}^m}{\operatorname{argmax}} \{\langle \mathbf{b}, \mathbf{s}\rangle - tH(|\mathbf{s}|)\} \quad \text{subject to} \quad -\overrightarrow{\mathbf{1}} \le \mathbf{A}^T\mathbf{s} \le \overrightarrow{\mathbf{1}}. \tag{1.10}$$

The second part of the TCCS algorithm uses the Lagrangian dual correspondence between (1.8) and (1.9), and a first-order approximation in terms of $1/\mu$ to find an approximate solution to

$$\underset{y\in\mathbb{R}^n}{\operatorname{argmin}} \{\|y\|_1 + R(t, y)\}. \tag{1.11}$$

As a consequence, the TCCS algorithm outputs a sparse solution to compressed sensing problems when we set $t = 0$.

Although the TCCS algorithm does not find the exact optimal argument for problem (1.11) (or (1.10)), it performs very well. In section 4 we provide numerical evidence that shows that when the TCCS algorithm is applied to compressed sensing problems (i.e. noiseless data), it perfectly recovers the solution when it is relatively sparse. Moreover, the TCCS algorithm is faster than some of the fastest methods used for solving compressed sensing problems (see section 4 for more details). Another advantage of using the TCCS algorithm for compressed sensing problems is that it is parameter free. In section 4, we also present numerical evidence that the TCCS algorithm performs well when the data are noisy.

We also derive some theoretical results that are important in their own right. In particular, in Corollary 2.7 we provide an alternative proof for the exact regularization property in problem (1.4). We also show in subsection 2.3 how the solution of problem (1.10) can be used to yield an approximate solution to primal problem (1.8) for sufficiently large μ.

Some readers might think that the idea used in the first part of the TCCS algorithm resembles the simplex method or the interior method used in optimization.

However, the resemblance is misleading. Suppose a polytope Ω is defined by

$$\Omega = \{\mathbf{s} \in \mathbb{R}^m : a_i^T \mathbf{s} \leq 1, \text{ for } i = 1 \dots, n\}.$$

For every point $\mathbf{s} \in \mathbb{R}^m$, let $J(\mathbf{s})$ be the set of indices $i \in \{1, \dots, n\}$ for which $a_i^T \mathbf{s} \geq 1$. We call $J(\mathbf{s})$ the set of *violations* of $\mathbf{s}$. We sometimes use J instead of $J(\mathbf{s})$ when this does not cause confusion. Let $\mathbf{A}_J$ be the matrix formed by appending the columns of $\mathbf{A}$ whose index belongs to J.

In each iteration of the simplex method, we move from an extreme point (a point on the boundary of Ω for which the corresponding matrix $\mathbf{A}_J$ has rank m) to another extreme point. On the other hand, in each iteration of the interior method, the corresponding points are inside Ω, and therefore their set of violations is empty (i.e., by convention $\mathbf{A}_J$ has rank zero for these points). However, in the TCCS algorithm, we start with a point inside the polytope Ω and at each iteration we move to a new point on a face of Ω in such a way that the rank of $\mathbf{A}_J$ increases by at least one. As a consequence the algorithm stops after a maximum of m iterations.

Indeed, the algorithm we used for the first part of the TCCS algorithm, belongs to a class of techniques called homotopy method (see, for example, [21, 22] for general applications of homotopy technique and [10, 20, 25] for applications of homotopy technique to L_1 minimization problems), suitably adapted for the particular problem at hand. Let

$$g_\tau(\mathbf{s}) = \langle \mathbf{b}, \mathbf{s} \rangle - \tau H(|\mathbf{s}|).$$

In essence, we generate a sequence $\tau_0 > \tau_1 > \cdots > \tau_\ell = t$ and iteratively find approximate maximizers $\mathbf{s}_{opt}^1, \dots, \mathbf{s}_{opt}^\ell$, respectively, for functions $g_{\tau_0}(\mathbf{s}), \dots, g_{\tau_\ell}(\mathbf{s})$ subject to constraint $-\overrightarrow{\mathbf{1}} \leq \mathbf{A}^T \mathbf{s} \leq \overrightarrow{\mathbf{1}}$. As a result, $\mathbf{s}_{opt}^\ell$ yields a good approximate maximizer for (1.10) .

The TCCS algorithm also resembles LARS [10] and adaptive inverse scale space [4], however, it is different from these compressed sensing related algorithms as the homotopy technique is applied to the dual problem.

The main contribution of our paper is to investigate application of homotopy technique to the dual formulation of L_1 minimization problems. We introduce an approximate algorithm that perfectly recovers the solution of compressed sensing problems (i.e. noiseless data) when the solution is relatively sparse with respect to the number of measurements. For this regime of sparsity, the recovery is extremely fast compared to other algorithms being used. Moreover, the TCCS algorithm has a fixed number of iterations. Also, the TCCS algorithm exhibits a sharp phase transition in the success rate of recovery of solution when sparsity of solution varies. The TCCS algorithm can also be used to solve problems such as the LASSO problem.

The reminder of this paper is organized as follows. In section 2 we provide some theoretical result that are used in developing the TCCS algorithm. Section 3 describes the general idea of the TCCS algorithm. Section 4 contains numerical evidence for the good performance of the TCCS algorithm. In section 5 we give some concluding remarks. In Appendix A we provide pseudo-codes for the TCCS algorithm.

2. Theoretical results

In this section we establish some theoretical results. These results justify and motivate the construction of the TCCS algorithm described in section 3. In subsections 2.1 and 2.2 we study the trajectory of the solutions for problems (1.9) and (1.10), respectively, as we vary parameter t. In subsection 2.3 we use a first-order approximation in terms of $1/\mu$ to establish a relation between the solution of (1.10) and the solution of the of the primal problem (1.8) when μ is large.

Indeed, in the rest of the paper, we consider the more general problems

$$\mathbf{s}_{opt}(t) = \operatorname{argmax}\{\langle \mathbf{b}, \mathbf{s}\rangle - tH(|\mathbf{s}|)\} \quad \text{subject to } \mathbf{A}^T\mathbf{s} \leq \overrightarrow{\mathbf{1}} \tag{2.1}$$

and

$$\mathbf{s}^*(t,\mu) = \operatorname*{argmax}_{\mathbf{s}\in\mathbb{R}^m} \left\{ \langle \mathbf{b}, \mathbf{s}\rangle - \frac{\mu}{2}|\mathcal{S}(\mathbf{A}^T\mathbf{s}, \overrightarrow{\mathbf{1}})|^2 - tH(|\mathbf{s}|)\right\}, \tag{2.2}$$

where operator $\mathcal{S}$ is defined by $\mathcal{S}(\vec{x},\vec{v})_i = \max(x_i - v_i, 0)$ and $\mathbf{A}^T$ is such that polytope

$$\Omega = \{\mathbf{s} : \mathbf{A}^T\mathbf{s} \leq \overrightarrow{\mathbf{1}}\}$$

is bounded. Here, $\mathbf{b}$, μ and t are fixed, $\mathbf{A}$ is $m \times n$ matrix with $m \ll n$. We make the following assumptions about function H:

(1) H is a function from positive real numbers to positive real numbers.
(2) H is strictly convex and strictly increasing.
(3) H is differentiable with h denoting its derivative.
(4) h is a composition of polynomials and radicals.
(5) h maps bounded sets to bounded sets.

Remark 2.1. Assumption 4 above is used in Lemma 2.3 and assumption 5 is used in Theorem 2.4. One might be able to weaken these assumptions.

Remark 2.2. Functions $H(x) = \frac{1}{2}x^2$ and $H(x) = \sqrt{1+x^2}$ satisfy all these assumptions.

To see that the new formulation yields the original problem, observe that if

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & | & -\mathbf{A} \end{bmatrix}, \tag{2.3}$$

then $|\mathsf{shrink}(\mathbf{A}^T\mathbf{s}, \overrightarrow{\mathbf{1}})| = |\mathcal{S}(\tilde{\mathbf{A}}^T\mathbf{s}, \overrightarrow{\mathbf{1}})|$. Therefore, the solution to (2.2), with $\tilde{\mathbf{A}}$ in place of $\mathbf{A}$, is the same as the solution to (1.9). Moreover, if $y = \mu \cdot \mathsf{shrink}(\mathbf{A}^T\mathbf{s}, \overrightarrow{\mathbf{1}})$ and $\tilde{y} = \mu \cdot \mathcal{S}(\tilde{\mathbf{A}}^T\mathbf{s}, \overrightarrow{\mathbf{1}})$, then

$$y = \begin{bmatrix} \mathbf{I} & | & -\mathbf{I} \end{bmatrix} \tilde{y}, \tag{2.4}$$

where $\mathbf{I}$ is the identity matrix.

2.1. Unconstrained dual problem.

In this section we investigate the solution to problem (2.1).

Set

$$f_\tau(\mathbf{s}) = \langle \mathbf{b}, \mathbf{s}\rangle - \tau H(|\mathbf{s}|) - \frac{\mu}{2}|\mathcal{S}(\mathbf{A}^T\mathbf{s}, \overrightarrow{\mathbf{1}})|^2. \tag{2.5}$$

Observe that

$$\nabla f_\tau(\mathbf{s}) = \mathbf{b} - \tau h(|\mathbf{s}|)\frac{\mathbf{s}}{|\mathbf{s}|} - \mu\mathbf{A}\mathcal{S}(\mathbf{A}^T\mathbf{s}, \overrightarrow{\mathbf{1}}).$$

Because of the concavity of functions $f_\tau(\mathbf{s})$ and since $f_\tau(\mathbf{s})$ is bounded above when $\mu > 0$, we know that $\mathbf{s}^* = \mathbf{s}^*(\tau, \mu) = \text{argmax}_{\mathbf{s}}(f_\tau(\mathbf{s}))$ is either the origin or satisfies

$$0 = \mathbf{b} - \tau h(|\mathbf{s}^*|)\frac{\mathbf{s}^*}{|\mathbf{s}^*|} - \mu \mathbf{A}\mathcal{S}(\mathbf{A}^T\mathbf{s}^*, \overrightarrow{\mathbf{1}}). \tag{2.6}$$

It is clear that for nontrivial cases, the optimal argument $\mathbf{s}^*$ is not at the origin. Indeed, if $\mathbf{s}^*(\tau_1, \mu)$ is not at the origin, then, for $\tau_2 < \tau_1$, $\mathbf{s}^*(\tau_2, \mu)$ would not be at the origin. To see this note that

$$0 \le f_{\tau_1}(\mathbf{s}^*(\tau_1, \mu)) - f_{\tau_1}(\vec{0}) < f_{\tau_2}(\mathbf{s}^*(\tau_1, \mu)) - f_{\tau_2}(\vec{0}),$$

where the first inequality uses the definition of $\mathbf{s}^*(\tau_1, \mu)$ and the second inequality uses the assumption that function H is strictly increasing. On the other hand, by definition of $\mathbf{s}^*(\tau_2, \mu)$, $f_{\tau_2}(\mathbf{s}^*(\tau_1, \mu)) \le f_{\tau_2}(\mathbf{s}^*(\tau_2, \mu))$. Therefore, $f_{\tau_2}(\vec{0}) < f_{\tau_2}(\mathbf{s}^*(\tau_2, \mu))$, which means $\mathbf{s}^*(\tau_2, \mu)$ is not at the origin. For this reason, we usually discard the origin in the discussion that follows.

For every point $\mathbf{s} \in \mathbb{R}^m$, let $J(\mathbf{s})$ be the set of indices i for which

$$a_i^T \mathbf{s} \ge 1.$$

Here a_i^T denotes the ith row of matrix A^T. We call $J(\mathbf{s})$ the set of *violations* of $\mathbf{s}$. We sometimes use J instead of $J(\mathbf{s})$ where this does not cause confusion. Let $\mathbf{A}_J^T$ be the matrix formed by appending the rows of $\mathbf{A}^T$ whose index belongs to J. Let $\mathbf{A}_J$ be the transpose of $\mathbf{A}_J^T$; that is, the matrix formed by appending the columns of $\mathbf{A}$ whose index belongs to J. Now, equation (2.6) is equivalent to

$$0 = \mathbf{b} - \tau h(|\mathbf{s}^*|)\frac{\mathbf{s}^*}{|\mathbf{s}^*|} - \mu \mathbf{A}_J(\mathbf{A}_J^T\mathbf{s}^* - \overrightarrow{\mathbf{1}}).$$

Rearranging implies that

$$\mathbf{s}^* = \left(\tau\frac{h(|\mathbf{s}^*|)}{|\mathbf{s}^*|}\mathbf{I} + \mu \mathbf{A}_J\mathbf{A}_J^T\right)^{-1}(\mathbf{b} + \mu\mathbf{A}_J\overrightarrow{\mathbf{1}}), \tag{2.7}$$

where $\mathbf{I}$ is the identity matrix. The matrix inverse in equation (2.7) always exists for $\tau > 0$, $0 < \mu < \infty$, and $\mathbf{s}^*$ is not the origin[1].

Consider SVD decomposition

$$\mathbf{A}_J^T = U\Sigma\begin{bmatrix} V^T \\ W^T \end{bmatrix},$$

where rows of V^T (i.e., $\{v_1^T, \ldots, v_k^T\}$) and W^T (i.e., $\{w_1^T \ldots, w_{m-k}^T\}$) correspond, respectively, to nonzero and zero singular values of $\mathbf{A}_J^T$. In particular, $span\{a_i\}_{i\in J}$ $= span\{v_i\}_{i=1}^k$. Moreover, $\{v_1, \ldots, v_k, w_1 \ldots, w_{m-k}\}$ form an orthonormal set of basis. We also have,

$$\mathbf{A}_J^T = \sum_{i=1}^k u_i\sigma_i v_i^T \qquad \text{and} \qquad \mathbf{A}_J\mathbf{A}_J^T = \sum_{i=1}^k \sigma_i^2 v_i v_i^T.$$

Note that,

$$\tau\frac{h(|\mathbf{s}^*|)}{|\mathbf{s}^*|}\mathbf{I} + \mu\mathbf{A}_J\mathbf{A}_J^T = \tau\frac{h(|\mathbf{s}^*|)}{|\mathbf{s}^*|}\sum_{i=1}^{m-k} w_i w_i^T + \sum_{i=1}^k(\mu\sigma_i^2 + \tau\frac{h(|\mathbf{s}^*|)}{|\mathbf{s}^*|})v_i v_i^T. \tag{2.8}$$

[1]One way to see this is noting that eigenvalues of the matrix are strictly positive due to positivity of h. See also equation (2.8)

Taking an inverse and substituting in (2.7) yields

$$\mathbf{s}^* = \frac{|\mathbf{s}^*|}{\tau h(|\mathbf{s}^*|)}(\sum_{i=1}^{m-k} w_i w_i^T)(\mathbf{b} + \mu \mathbf{A}_J \overrightarrow{\mathbf{1}}) + \sum_{i=1}^{k} \frac{1}{\mu\sigma_i^2 + \tau h(|\mathbf{s}^*|)/|\mathbf{s}^*|} v_i v_i^T (\mathbf{b} + \mu \mathbf{A}_J \overrightarrow{\mathbf{1}}).$$

The w_i's are in the space perpendicular to $span\{a_i\}_{i\in J}$, $w_i^T \mathbf{A}_J = 0$. Therefore, the above equation simplifies to

$$\mathbf{s}^*(\tau,\mu) = \frac{|\mathbf{s}^*|}{\tau h(|\mathbf{s}^*|)}(\sum_{i=1}^{m-k} w_i w_i^T)\mathbf{b} + \sum_{i=1}^{k} \frac{1}{1 + \frac{\tau}{\mu\sigma_i^2} h(|\mathbf{s}^*|)/|\mathbf{s}^*|} \frac{v_i v_i^T}{\sigma_i^2}(\mathbf{b}/\mu + \mathbf{A}_J \overrightarrow{\mathbf{1}}). \tag{2.9}$$

This is one of the main equations that is used throughout this paper. Equation (2.9) describes an implicit equation for the trajectory of optimal argument $\mathbf{s}^*$ as the function of τ and μ. Note that $\mathbf{A}_J$, σ_i's, w_i's and v_i's depend on the set of violations of $\mathbf{s}^*$, $J(\mathbf{s}^*)$. Also equation (2.9) is valid for $0 < \tau$ and $0 \le \mu$ (since we needed the matrix in equation (2.7) to be invertible).

At first glance, equation (2.9) might not seem very helpful, as it is an implicit equation. However, we can infer many useful properties from this equation. What follows is not very practical to implement; however, it is useful in deriving some theoretical results. The TCCS algorithm inspired by these ideas is presented in section 3.

For every set J, τ and μ, we can write the implicit equation

$$\mathbf{s} = \frac{|\mathbf{s}|}{\tau h(|\mathbf{s}|)}(\sum_i w_i w_i^T)\mathbf{b} + \sum_i \frac{1}{1 + \frac{\tau}{\mu\sigma_i^2} h(|\mathbf{s}|)/|\mathbf{s}|} \frac{v_i v_i^T}{\sigma_i^2}(\mathbf{b}/\mu + \mathbf{A}_J \overrightarrow{\mathbf{1}}). \tag{2.10}$$

However, note that $\mathbf{s}$ in above equation is not necessarily the optimal solution because in equation (2.9), the set of violations J (i.e. and therefore $\mathbf{A}_J^T$, σ_i's, w_i's and v_i's) implicitly depends on τ and μ.

For every set J and $i \in \{1,\dots,n\}$, let $\Theta_i(J)$ be the set of τ of positive real numbers for which there exist $\mathbf{s}$ that satisfies both equation (2.10) and $a_i^T \mathbf{s} = 1$. The importance of the set $\Theta_i(J)$ will be discussed shortly. However, we first show the following lemma:

Lemma 2.3. *For fixed μ, J and i, set $\Theta_i(J)$ consists of a finite number of intervals in $\mathbb{R}$.*

Proof. First we take Euclidean norm of both sides of equation (2.10) and simplify. Orthogonality of the basis $\{w_1,\dots,w_{m-k},v_1,\dots,v_k\}$ considerably simplifies the expression. We perform a series of algebraic manipulations (depending on the form of function h), to reduce the expression to the form

$$P(|\mathbf{s}|,\tau) = 0, \tag{2.11}$$

where P is a polynomial. For example, in the case of $H(|\mathbf{s}|) = |\mathbf{s}|^2/2$, equation (2.10) becomes

$$\mathbf{s} = \frac{1}{\tau}\sum_i (w_i^T \mathbf{b}) w_i + \sum_i \frac{v_i^T(\mathbf{b}/\mu + \mathbf{A}_J \overrightarrow{\mathbf{1}})}{\sigma_i^2 + \tau/\mu} v_i. \tag{2.12}$$

It follows that

$$|\mathbf{s}|^2 = \sum_i \left|\frac{w_i^T \mathbf{b}}{\tau}\right|^2 + \sum_i \left|\frac{v_i^T(\mathbf{b}/\mu + \mathbf{A}_J \overrightarrow{\mathbf{1}})}{\sigma_i^2 + \tau/\mu}\right|^2.$$

Multiplying both sides by $\tau^2 \prod_{j=1}^{k}(\sigma_j^2 + \tau/\mu)^2$ and rearranging yields that in the case of $H(|\mathbf{s}|) = |\mathbf{s}|^2/2$,

$$P(|\mathbf{s}|,\tau) = \tau^2 \prod_j (\sigma_j^2 + \tau/\mu)^2 |\mathbf{s}|^2 - \prod_j (\sigma_j^2 + \tau/\mu)^2 \sum_i \left| w_i^T \mathbf{b} \right|^2 - \sum_i \tau^2 \prod_{j \neq i} (\sigma_j^2 + \tau/\mu)^2 \left| v_i^T (\mathbf{b}/\mu + \mathbf{A}_J \overrightarrow{\mathbf{1}}) \right|^2.$$

Next suppose $\mathbf{s}$ satisfies $a_i^T \mathbf{s} = 1$. We multiply both sides of equation (2.10) by a_i^T. The LHS is equal to 1. Again, we perform a series of algebraic manipulations to reduce the expression to the form

$$Q(|\mathbf{s}|,\tau) = 0, \tag{2.13}$$

where Q is a polynomial. For example in the case of $H(|\mathbf{s}|) = |\mathbf{s}|^2/2$, multiplying both sides of equation (2.12) by a_i^T yields

$$1 = \frac{1}{\tau} \sum_i (w_i^T \mathbf{b})(a_i^T w_i) + \sum_i \frac{v_i^T (\mathbf{b}/\mu + \mathbf{A}_J \overrightarrow{\mathbf{1}})}{\sigma_i^2 + \tau/\mu} (a_i^T v_i).$$

Multiplying both sides by $\tau \prod_{j=1}^{k}(\sigma_j^2 + \tau/\mu)$ and rearranging yields that in the case of $H(|\mathbf{s}|) = |\mathbf{s}|^2/2$,

$$Q(|\mathbf{s}|,\tau) = \tau \prod_j (\sigma_j^2 + \tau/\mu) - \prod_j (\sigma_j^2 + \tau/\mu) \sum_i (w_i^T \mathbf{b})(a_i^T w_i) - \sum_i \tau \prod_{j \neq i} (\sigma_j^2 + \tau/\mu) v_i^T (\mathbf{b}/\mu + \mathbf{A}_J \overrightarrow{\mathbf{1}})(a_i^T v_i).$$

If τ is chosen such that $\mathbf{s}$ satisfies equation (2.10) and $a_i^T \mathbf{s} = 1$, then $|\mathbf{s}|$ and τ must satisfy both (2.11) and (2.13). From Bezout's theorem, polynomials P and Q have a finite number of intersections, unless they have a common component. In either case a set of τ for which there exists $|\mathbf{s}|$ such that the pair $(|\mathbf{s}|,\tau)$ satisfies that both (2.11) and (2.13) consist of a finite number of intervals in $\mathbb{R}$. The result follows. □

Next partition $\mathbb{R}^m$ into regions where the points of each region have the same set of violations. Clearly there are a finite number of regions (although many of them). Now going back to equation (2.9) recall that $\mathbf{A}_J^T$, σ_i's, w_i's and v_i's depend on the set of violations of $\mathbf{s}^*$. Fix μ and think of $\mathbf{s}^*$ as a function of τ (to ease the notation we use $\mathbf{s}^*(\tau)$ in place of $\mathbf{s}^*(\tau,\mu)$ and $J(\tau)$ in place of $J(\mathbf{s}^*(\tau,\mu))$. Let τ_0 be sufficiently large so that $\mathbf{s}^*(\tau_0)$ is close to the origin and $J(\tau_0)$ is empty. Such τ_0 exist because $\mathbf{s}^*(\tau)$ is the maximizer of function (2.5) and it is assumed that function H is strictly convex and strictly increasing; consequently, as $\tau \to \infty$, $\mathbf{s}^*(\tau)$ is forced to converge to the origin. Set $\tau = \tau_0$.

Since the set of violations is empty, from equation (2.9), we conclude that

$$\mathbf{s}^*(\tau) = \frac{|\mathbf{s}^*(\tau)|}{\tau h(|\mathbf{s}^*(\tau)|)} \mathbf{b}. \tag{2.14}$$

The above equation describes trajectory for $\mathbf{s}^*(\tau)$ in the region where the set of violations of the optimal argument is empty.

Next, we slowly decrease τ from τ_0. As $\tau \to t$, by continuity $\mathbf{s}^*(\tau) \to \mathbf{s}^*(t)$. Equation (2.14) continues to hold until the set of violations of $\mathbf{s}^*(\tau)$ changes; that is, $\mathbf{s}^*(\tau)$ enters into a new region. Let τ_1 denote the largest τ for which this happens. We continue this process.

Suppose at $\tau = \tau_r$, $\mathbf{s}^*(\tau)$ enters into a new region. Again by continuity, for $\tau < \tau_r$, $\mathbf{s}^*(\tau)$ follows a trajectory prescribed by equation (2.9) (using $J(\tau_r)$ in place of J), until the set of violations of $\mathbf{s}^*(\tau)$ changes; that is, $\mathbf{s}^*(\tau)$ enters into another region at $\tau = \tau_{r+1}$. Therefore, as τ decreases from τ_r, τ_{r+1} is the first τ for which $\mathbf{s}^*(\tau)$ passes through a plane given by $\{\mathbf{s} : a_k^T \mathbf{s} = 1\}$ for some $k \in \{1, \dots, n\}$. Thus, τ_{r+1} is equal to the largest element smaller than τ_r that belongs to $\Theta_k(J(\tau_r))$ for some k, where τ_r and τ_{r+1} are not on the same interval in $\Theta_k(J(\tau_r))$ (i.e., if τ_r and τ_{r+1} are on the same interval of $\Theta_k(J(\tau_r))$, it means that for $\tau_{r+1} \le \tau \le \tau_r$, $\mathbf{s}^*(\tau)$ lies on the plane given by $\{\mathbf{s} : a_k^T \mathbf{s} = 1\}$, consequently, the set of violations of $\mathbf{s}^*(\tau)$ does not change due to k).

We continue this process until τ reaches t.

The important observation is that because of Lemma 2.8 and since there are a finite number of regions (i.e. different set of violations) and indices i, the above process does not continue indefinitely. Therefore we have a sequence $\tau_0 > \tau_1 > \cdots > \tau_N > \tau_{N+1} = 0$, where for $\tau \in (\tau_{i+1}, \tau_i]$, $\mathbf{s}^*(\tau)$ lies in the same region.

Now we use the above observations to prove several results.

Theorem 2.4. *For every μ there exist $t_c > 0$ and matrix $\mathbf{A}_J$ with rank m such that for $0 < \tau \le t_c$,*

$$\mathbf{s}^*(\tau, \mu) = \sum_i \frac{1}{1 + \frac{\tau}{\mu\sigma_i^2} h(|\mathbf{s}^*|)/|\mathbf{s}^*|} \frac{v_i v_i^T}{\sigma_i^2} (\mathbf{b}/\mu + \mathbf{A}_J \overrightarrow{\mathbf{1}}). \tag{2.15}$$

Proof. In the discussions above, set $t_c = \tau_N$ and $\mathbf{A}_J = \mathbf{A}_{J(\tau_N)}$. First suppose that $J(\tau_N)$ is empty. From equation (2.9) (also see equation (2.14)) we conclude that for $\tau \in (0, t_c]$,

$$\mathbf{s}^*(\tau, \mu) = \frac{|\mathbf{s}^*|}{\tau h(|\mathbf{s}^*|)} \mathbf{b}.$$

For nontrivial cases, $\mathbf{s}^*$ is not at the origin (see the discussion after equation (2.6)). Therefore, we can take a Euclidean norm from both sides and simplify to conclude that

$$h(|\mathbf{s}^*|) = |\mathbf{b}|/\tau.$$

However, $\mathbf{s}^*(\tau, \mu)$ is bounded (indeed, because $J(\tau_N)$ is empty, $\mathbf{s}^*(\tau, \mu)$ lies within polytope $\Omega = \{\mathbf{A}^T \mathbf{s} < \overrightarrow{\mathbf{1}}\}$, which was assumed to be bounded). Therefore, by assumption 5 about function h, the LHS of the above equation is bounded for $\tau \in (0, t_c]$. However, as $\tau \to 0$, the RHS goes to infinity, which yields a contradiction.

Next suppose $J(\tau_N)$ is nonempty. From equation (2.9), we know that for $\tau \in (0, t_c]$,

$$\mathbf{s}^*(\tau, \mu) = \frac{|\mathbf{s}^*|}{\tau h(|\mathbf{s}^*|)} \Big(\sum_i^{m-k} w_i w_i^T \Big) \mathbf{b} + \sum_i^k \frac{1}{1 + \frac{\tau}{\mu\sigma_i^2} h(|\mathbf{s}^*|)/|\mathbf{s}^*|} \frac{v_i v_i^T}{\sigma_i^2} (\mathbf{b}/\mu + \mathbf{A}_J \overrightarrow{\mathbf{1}}),$$

where $\mathbf{A}_J$, w_i's, v_i's and σ_i's are fixed. Because the origin lies within the region corresponding to the empty violation set, $J(\tau_N)$ being nonempty implies that $\mathbf{s}^*(\tau, \mu)$ is bounded away from the origin for $\tau \in (0, t_c]$. Furthermore, we know that $\mathbf{s}^*(\tau, \mu)$ is bounded as $\tau \to 0$. Therefore, from assumption 5 about function h, $h(|\mathbf{s}^*|)$ is

bounded as $\tau \to 0$. Hence, unless $(\sum_i^{m-k} w_i w_i^T)\mathbf{b} = 0$, the first summand in the above equation would blow up as τ approaches 0; which yields a contradiction. Thus we must have $k = m$; that is, $\mathbf{A}_J^T$ has m nonzero singular values, which implies that $\mathbf{A}_J$ has rank m. The result follows. □

Remark 2.5. The values of t_c and matrix $\mathbf{A}_J$ in Theorem 2.4 are μ dependent.

Theorem 2.6. *For μ sufficiently large, $\mu\mathcal{S}(\mathbf{A}^T\mathbf{s}^*(0,\mu),1)$ is independent of μ.*

Proof. From the result of Theorem 2.4 we know that as τ approaches 0, the optimal argument $\mathbf{s}^*(\tau,\mu)$ eventually satisfies equation (2.15). Although at $\tau = 0$ there might not be a unique optimal argument $\mathbf{s}^*$; however, from equation (2.15), one of them is given by

$$\mathbf{s}^*(0,\mu) = \sum_i \frac{v_i v_i^T}{\sigma_i^2}(\mathbf{b}/\mu + \mathbf{A}_J \overrightarrow{\mathbf{1}}), \tag{2.16}$$

where $J = J(\mathbf{s}^*(0,\mu))$ denote the set of violations of $\mathbf{s}^*(0,\mu)$. Recall that $\mathbf{A}_J$, v_i's and σ_i's depend on the value of μ. Next think of μ as variable. As μ increases, $\mathbf{s}^*(0,\mu)$ follows a trajectory given by (2.16). We can use a similar argument to the one used for decreasing τ to conclude that there exist μ_c such that for $\mu \geq \mu_c$, $\mathbf{s}^*(0,\mu)$ stays in the same region (i.e. see footnote 2).

Let $\mu \geq \mu_c$. Observe that $\mathbf{s}^*(0,\mu)$ still satisfies equation (2.16); however, $\mathbf{A}_J$, v_i's and σ_i's no longer depend on μ and are fixed. From the definition of the violation set J, we can conclude that the only rows of $\mathcal{S}(\mathbf{A}^T\mathbf{s}^*(t,\mu), \overrightarrow{\mathbf{1}})$ that are nonzero are exactly the elements of J. Thus, it suffices to show the result for $\mu\mathcal{S}(\mathbf{A}_J^T\mathbf{s}^*(0,\mu),1)$.

Now recall that $\mathbf{A}_J^T = \sum_j u_j\sigma_j v_j^T$. Thus,

$$\mathbf{A}_J^T\mathbf{s}^*(0,\mu) = (\sum_j u_j\sigma_j v_j^T)(\sum_i \frac{v_i v_i^T}{\sigma_i^2}(\mathbf{b}/\mu + \mathbf{A}_J\overrightarrow{\mathbf{1}})) = \sum_i \frac{u_i v_i^T}{\sigma_i}(\frac{\mathbf{b}}{\mu} + \mathbf{A}_J\overrightarrow{\mathbf{1}}). \tag{2.17}$$

In the limiting case when μ approaches ∞, $\mathbf{s}^*(0,\mu)$ approaches to the boundary of Ω. Indeed, when $\mu = \infty$, $\mathbf{s}^*(0,\infty)$ is the optimal solution of the linear programming problem

$$\underset{\mathbf{s}}{\operatorname{argmax}} \ \langle \mathbf{b}, \mathbf{s}\rangle \quad \text{subject to} \quad \mathbf{A}^T\mathbf{s} \leq \overrightarrow{\mathbf{1}}. \tag{2.18}$$

Therefore, $\mathbf{A}_J^T\mathbf{s}^*(0,\infty) = \overrightarrow{\mathbf{1}}$. By formally substituting ∞ in equation (2.17), we conclude that $\overrightarrow{\mathbf{1}} = \sum_i \frac{u_i v_i^T}{\sigma_i}\mathbf{A}_J\overrightarrow{\mathbf{1}}$. Substituting this back into the above equation, we have

$$\mathbf{A}_J^T\mathbf{s}^*(0,\mu) = (\sum_i \frac{u_i v_i^T}{\sigma_i})\frac{\mathbf{b}}{\mu} + \overrightarrow{\mathbf{1}}.$$

Recall that by the definition of violation set J, entries of $\mathbf{A}_J^T\mathbf{s}^*(0,\mu)$ are greater than or equal to one. We conclude that entries of $(\sum_i \frac{u_i v_i^T}{\sigma_i})\frac{\mathbf{b}}{\mu}$ are greater than or equal to zero. Hence,

$$\mu\mathcal{S}(\mathbf{A}_J^T\mathbf{s}^*(0,\mu),1) = (\sum_i \frac{u_i v_i^T}{\sigma_i})\mathbf{b}.$$

Since the right-hand side is the same for all $\mu \geq \mu_c$, the result follows. Indeed, one can easily show that $\sum_i \frac{u_i v_i^T}{\sigma_i}$ is the pseudo-inverse of $\mathbf{A}_J$, $\mathbf{A}_J^\dagger$.

As a bonus, we also have that for sufficiently large μ, the nonzero elements of

$$\mu \mathcal{S}(\mathbf{A}^T \mathbf{s}^*(0,\mu), 1)$$

are equal to $\mathbf{A}_J^\dagger \mathbf{b}$, where J is the set of active constraints for the optimal solution of the linear programming problem (2.18). □

Using the result of the above theorem, we may conclude that we have the exact regularization property described in [24] and [11].

Corollary 2.7. *There exist fixed large μ_c such that for $\mu \geq \mu_c$, the solution to problem* (1.4) *is independent of μ. Indeed, let J denote the set of active constraints of the maximizer of the linear programming problem*

$$\underset{\mathbf{s}}{\operatorname{argmax}} \ \langle \mathbf{b}, \mathbf{s} \rangle \quad \text{subject to} \quad \tilde{A}^T \mathbf{s} \leq \overrightarrow{\mathbf{1}},$$

where $\tilde{A}$ is given by (2.3). *Let $\tilde{y}$ be a column vector with $2n$ entries where n is the number of columns of A. Moreover, those entries of $\tilde{y}$ whose index belong to J are given by $\tilde{y}(J) = \tilde{\mathbf{A}}_J^\dagger \mathbf{b}$, and the other entries of $\tilde{y}$ are zero. Then the minimizer of problem* (1.4) *is given by*

$$\begin{bmatrix} \mathbf{I} & | & -\mathbf{I} \end{bmatrix} \tilde{y}.$$

Proof. From the Lagrangian dual correspondence in section 1, we know that the solution of (1.4) is equal to $\mu \cdot \mathsf{shrink}(\mathbf{A}^T \mathbf{s}^*, \overrightarrow{\mathbf{1}})$ where $\mathbf{s}^*$ is the solution of (1.9) with $t = 0$. Now use the result of Theorem 2.6 and the relation (2.4). □

2.2. **Constrained dual problem.** In this section we analyze the solution to the constrained problem (2.1). This problem can be viewed as the limiting case of problem (2.2) as μ approaches ∞. Many of the ideas used in this section are similar to the previous section. However, the formulas in this case become simpler and the results enables us to devise a fast algorithm for finding the optimal argument.

Let

$$g_\tau(\mathbf{s}) = \langle \mathbf{b}, \mathbf{s} \rangle - \tau H(|\mathbf{s}|),$$

and set

$$\mathbf{s}_{opt}(\tau) = \underset{\mathbf{s}}{\operatorname{argmax}}\, g_\tau(\mathbf{s}) \qquad \text{subject to} \qquad \mathbf{A}^T \mathbf{s} \leq \overrightarrow{\mathbf{1}}.$$

As in the previous section, for any subset J of $\{1, \ldots, n\}$, let $\mathbf{A}_J^T$ denote the matrix formed from rows of $\mathbf{A}^T$ whose index belongs to J. The following lemma is essential in the analysis done in this subsection:

Lemma 2.8. *Suppose $\tau > 0$ and J is a fixed subset of $\{1, \ldots, n\}$. If function $g_\tau(\mathbf{s})$ has unique global max on the set $\{\mathbf{s} : \mathbf{A}_J^T \mathbf{s} = \overrightarrow{\mathbf{1}}\}$, then this global maximum is given by*

$$\mathbf{s}_J(\tau) := \underset{\mathbf{s}:\mathbf{A}_J^T \mathbf{s} = \overrightarrow{\mathbf{1}}}{\operatorname{argmax}} g_\tau(\mathbf{s}) = F(\tau, J) \overrightarrow{\alpha}(J) + \overrightarrow{\beta}(J) \tag{2.19}$$

where $\overrightarrow{\alpha}(J) := \mathbf{b} - \mathbf{A}_J (\mathbf{A}_J^T \mathbf{A}_J)^\dagger \mathbf{A}_J^T \mathbf{b}$ is the projection of $\mathbf{b}$ on the set $\{\mathbf{s} : \mathbf{A}_J^T \mathbf{s} = \overrightarrow{\mathbf{1}}\}$, and $\overrightarrow{\beta}(J) := \mathbf{A}_J (\mathbf{A}_J^T \mathbf{A}_J)^\dagger \overrightarrow{\mathbf{1}}$. Here, F is some function that is determined by values of τ, the set J and the function h (see the remark below).

Remark 2.9. For some specific functions $H(|\mathbf{s}|)$, closed expressions for $F(\tau, J)$ is available. For example in the case of $H(|\mathbf{s}|) = |\mathbf{s}|^2/2$, it is easy to show that

$$F(\tau, J) = \frac{1}{\tau};$$

therefore, equation (2.19) becomes

$$\mathbf{s}_J(\tau) = \frac{1}{\tau}\overrightarrow{\alpha}(J) + \overrightarrow{\beta}(J).$$

As another example, in the case of $H(|\mathbf{s}|) = \sqrt{1+|\mathbf{s}|^2}$, it is shown in the proof below that

$$F(\tau, J) = \sqrt{\frac{1+|\overrightarrow{\beta}(J)|^2}{\tau^2 - |\overrightarrow{\alpha}(J)|^2}};$$

therefore, equation (2.19) becomes

$$\mathbf{s}_J(\tau) = \sqrt{\frac{1+|\overrightarrow{\beta}(J)|^2}{\tau^2 - |\overrightarrow{\alpha}(J)|^2}}\,\overrightarrow{\alpha}(J) + \overrightarrow{\beta}(J).$$

Proof. Since the set $\{\mathbf{s} : \mathbf{A}_J^T\mathbf{s} = \overrightarrow{\mathbf{1}}\}$ is convex and g_τ is a concave function that has a global maximum on this set, it suffices to solve the first-order conditions for the Lagrangian

$$\mathcal{L}(\mathbf{s}, \overrightarrow{\lambda}) = \langle \mathbf{b}, \mathbf{s}\rangle - \tau H(|\mathbf{s}|) + \overrightarrow{\lambda}^T(\mathbf{A}_J^T\mathbf{s} - \overrightarrow{\mathbf{1}})$$

to find the global maximum.

First-order conditions with respect to $\mathbf{s}$ imply that

$$\mathbf{b} - \tau\frac{h(|\mathbf{s}_J|)}{|\mathbf{s}_J|}\mathbf{s}_J + \mathbf{A}_J\overrightarrow{\lambda}^* = 0, \tag{2.20}$$

and first-order conditions with respect to $\overrightarrow{\lambda}$ imply that

$$\mathbf{A}_J^T\mathbf{s}_J = \overrightarrow{\mathbf{1}}. \tag{2.21}$$

Multiplying both sides of (2.20) by $\mathbf{A}_J^T$, using (2.21), and rearranging yields that

$$\overrightarrow{\lambda}^* = (\mathbf{A}_J^T\mathbf{A}_J)^\dagger(\tau\frac{h(|\mathbf{s}_J|)}{|\mathbf{s}_J|}\overrightarrow{\mathbf{1}} - A_J^T\mathbf{b}).$$

Substituting the above into (2.20) and simplifying, yields that

$$\mathbf{s}_J = \frac{|\mathbf{s}_J|}{\tau h(|\mathbf{s}_J|)}\overrightarrow{\alpha} + \overrightarrow{\beta}, \tag{2.22}$$

where $\overrightarrow{\alpha} = \mathbf{b} - \mathbf{A}_J(\mathbf{A}_J^T\mathbf{A}_J)^\dagger\mathbf{A}_J^T\mathbf{b}$ and $\overrightarrow{\beta} = \mathbf{A}_J(\mathbf{A}_J^T\mathbf{A}_J)^\dagger\overrightarrow{\mathbf{1}}$. It is straightforward to verify that $\overrightarrow{\alpha}$ is the projection of vector $\mathbf{b}$ onto the set $\{\mathbf{s} : \mathbf{A}_J^T\mathbf{s} = 1\}$, and $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$ are perpendicular to each other.

Now taking the Euclidean norm square of (2.22), we have

$$|\mathbf{s}_J|^2 = \frac{|\mathbf{s}_J|^2}{\tau^2 h(|\mathbf{s}_J|)^2}|\overrightarrow{\alpha}|^2 + |\overrightarrow{\beta}|^2. \tag{2.23}$$

From the above equation and properties of function h, we conclude that $|\mathbf{s}_J|$ is the root of some polynomial that depends on τ and J (indeed, the coefficients of the polynomial are determined by τ, function h, $|\overrightarrow{\alpha}|$ and $|\overrightarrow{\beta}|$). Substituting these roots

in the RHS of equation (2.22) yields an explicit formula for $\mathbf{s}_J$. For example, in the case $H(|\mathbf{s}|) = \sqrt{1+|\mathbf{s}|^2}$, (2.23) becomes

$$|\mathbf{s}_J|^2 = \frac{1+|\mathbf{s}_J|^2}{\tau^2}|\overrightarrow{\alpha}|^2 + |\overrightarrow{\beta}|^2.$$

Solving for $|\mathbf{s}_J|$ implies that

$$|\mathbf{s}_J| = \sqrt{\frac{|\overrightarrow{\alpha}|^2 + \tau^2|\overrightarrow{\beta}|^2}{\tau^2 - |\overrightarrow{\alpha}|^2}},$$

which upon subtitution in the RHS of equation (2.22) yields that for $H(|\mathbf{s}|) = \sqrt{1+|\mathbf{s}|^2}$,

$$\mathbf{s}_J = \sqrt{\frac{1+|\overrightarrow{\beta}|^2}{\tau^2 - |\overrightarrow{\alpha}|^2}}\overrightarrow{\alpha} + \overrightarrow{\beta}.$$

As noted above, for other functions H, $|\mathbf{s}_J|$ is the root of some polynomial. In general, the polynomial might have several positive roots. However, because we are given that $g_\tau(\mathbf{s})$ has global maximum on the set (and global maximum is unique for $\tau > 0$ due to strict concavity), only one of the roots corresponds to $|\mathbf{s}_J|$. For general functions H, finding closed formulas for $|\mathbf{s}_J|$ and $\frac{|\mathbf{s}_J|}{\tau h(|\mathbf{s}_J|)}$ might not be possible; however, the value of $\frac{|\mathbf{s}_J|}{\tau h(|\mathbf{s}_J|)}$ is uniquely determined by τ and J. If we set $F(\tau, J) = \frac{|\mathbf{s}_J|}{\tau h(|\mathbf{s}_J|)}$, then

$$\mathbf{s}_J(\tau) = F(\tau, J)\overrightarrow{\alpha}(J) + \overrightarrow{\beta}(J). \qquad \square$$

We now analyze the path that the optimal argument $\mathbf{s}_{opt}(\tau)$ takes as τ decreases.

When τ is sufficiently large, $\mathbf{s}_{opt}(\tau)$ lies in the interior of Ω. We can find $\mathbf{s}_{opt}(\tau)$ by taking the gradient of $g_\tau(\mathbf{s})$ and setting it equal to zero. We conclude that

$$\mathbf{b} - \tau\frac{h(|\mathbf{s}_{opt}|)}{|\mathbf{s}_{opt}|}\mathbf{s}_{opt} = 0.$$

Taking the norm from both sides and simplifying yields that

$$\mathbf{s}_{opt}(\tau) = \frac{1}{|\mathbf{b}|}h^{-1}(\frac{|\mathbf{b}|}{\tau})\mathbf{b} = F(\tau, \emptyset)\alpha(\emptyset). \tag{2.24}$$

The second equality in the above equation comes from $\mathbf{b} = \overrightarrow{\alpha}(\emptyset)$. We have written this to highlight that the above equation is a special case of equation (2.19) when J is the empty set.

Now we decrease τ slowly. We know that $\mathbf{s}_{opt}(\tau)$ follows the path given by equation (2.24) until it hits a boundary of polytope Ω. Let τ_1 denote the first time that this happens. As a consequence, for $t \le \tau \le \tau_1$, the global maximum of $g_\tau(\mathbf{s})$ occurs outside of Ω if there was no constraints. Hence, because g_τ is a concave function and Ω is a convex set, $\mathbf{s}_{opt}(\tau)$ must be on a boundary of Ω for $\tau \le \tau_1$.

We use J_1 to denote the set of indices i for which $a_i^T\mathbf{s}_{opt}(\tau_1) = 1$. In the language of linear programming, J_1 is called the set of active constraints for point $\mathbf{s}_{opt}(\tau_1)$. From the discussion above, as τ decreases from τ_1, $\mathbf{s}_{opt}(\tau)$ moves along faces of Ω. Suppose the set of constraints K_1 determines the first face of Ω along which $\mathbf{s}_{opt}(\tau)$ moves for $\tau \le \tau_1$. Clearly, $K_1 \subset J_1$. From Lemma 2.8, for $\tau \le \tau_1$, $\mathbf{s}_{opt}(\tau)$ moves along the path given by

$$F(\tau, K_1)\overrightarrow{\alpha}(K_1) + \overrightarrow{\beta}(K_1),$$

until it hits another face of Ω. Let τ_2 denote the first time that this happens; that is, τ_2 is the largest $\tau < \tau_1$ for which the set of active constraints of $\mathbf{s}_{opt}(\tau)$ is different than K_1. Let J_2 denote the set of active constraints of $\mathbf{s}_{opt}(\tau_2)$. Again, as τ decreases from τ_2, $\mathbf{s}_{opt}(\tau)$ moves along a face of Ω. Let K_2 denote the set of constraints that determines this face. Again, $K_2 \subset J_2$, and we repeat as before.

We continue the above process until τ reaches t. From the arguments in section 2.1, we may conclude there is a sequence

$$\tau_0 > \tau_1 > \cdots > \tau_N > \tau_{N+1} = 0, \tag{2.25}$$

where for $\tau \in (\tau_{i+1}, \tau_i]$, $\mathbf{s}_{opt}(\tau)$ lies on the same face of the polytope. Therefore, we reach t by going through a finite number of faces. As mentioned earlier, the constrained problem can be viewed as the limiting case of an unconstrained problem as μ approaches ∞. Using this insight and Theorem 2.4, we have the following corollary:

Corollary 2.10. *There exist $t_c > 0$, such that for $0 < \tau \le t_c$ the optimal argument $\mathbf{s}_{opt}(\tau)$ for constrained problem*

$$\operatorname{argmax}\{\langle \mathbf{b}, \mathbf{s}\rangle - \tau H(|\mathbf{s}|)\} \quad \textit{subject to } \mathbf{A}^T\mathbf{s} \le \overrightarrow{\mathbf{1}},$$

are all the same and lie on an extreme point of the polytope Ω.

Proof. Formally substituting $\mu = \infty$ in equation (2.15) of Theorem 2.4, we observe that for $0 < \tau \le t_c$, $\mathbf{s}_{opt}(\tau)$ is independent of τ. Furthermore, since $\mathbf{A}_{J(t_c)}$ has rank m, we conclude that $\mathbf{s}_{opt}(t_c)$ is an extreme point of polytope Ω.

2.3. **Approximate solution of the augmented problem.** In this subsection we use first-order approximation in terms of $1/\mu$, to describe how $\mathbf{s}_{opt}(t)$ can be used to find an approximation for $y^*(t,\mu) = \mu\mathcal{S}(\mathbf{A}^T\mathbf{s}^*(t,\mu), \overrightarrow{\mathbf{1}})$ when μ is sufficiently large.

Note that formally, $\mathbf{s}_{opt}(t) = \mathbf{s}^*(t,\infty)$. For ease of notation, we occasionally use $\mathbf{s}^*$ and $\mathbf{s}_{opt}$ in place of $\mathbf{s}^*(t,\mu)$ and $\mathbf{s}_{opt}(t)$, respectively, in the remainder of this subsection. Using arguments similar to the one used in the proof of Theorem 2.6, we know that there exist μ_c such that for $\mu \ge \mu_c$, $\mathbf{s}^*(t,\mu)$ and $\mathbf{s}_{opt}$ have the same set of violations[2]. Suppose $\mu > \mu_c$. We may assume that $\mathbf{s}_{opt}$ does not lie in the interior of Ω as in that case $y^*(t,\mu) = \vec{0}$ and there is nothing to show.

Using first-order approximation,

$$\mathbf{s}^*(t,\mu) = \mathbf{s}_{opt} + \frac{1}{\mu}\Psi + O(1/\mu^2) \tag{2.26}$$

and

$$\frac{|\mathbf{s}^*(t,\mu)|}{h(|\mathbf{s}^*(t,\mu)|)} = \frac{|\mathbf{s}_{opt}|}{h(|\mathbf{s}_{opt}|)} + \frac{1}{\mu}\Gamma + O(1/\mu^2), \tag{2.27}$$

[2]Even though for $\mu \ge \mu_c$ the set of violations of $\mathbf{s}^*(t,\mu)$ are the same, $\mathbf{s}_{opt}$ is a limit point of the sequence $\mathbf{s}^*(t,\mu)$ as $\mu \to \infty$, and might have a larger set of violations. However, in our case, as we increase μ, the problem becomes more constrained and therefore we expect the set of violations to decrease. So we assume that $\mathbf{s}_{opt}$ and $\mathbf{s}^*(t,\mu)$, for $\mu \ge \mu_c$, have the same set of violations.

where Ψ and Γ are placeholders for the corresponding first-order derivatives. Recall from equation (2.9) that

$$\mathbf{s}^* = \frac{|\mathbf{s}^*|}{th(|\mathbf{s}^*|)}(\sum_{i=1}^{m-k} w_i w_i^T)\mathbf{b} + \sum_{i=1}^{k} \frac{1}{1+\frac{t}{\mu\sigma_i^2}h(|\mathbf{s}^*|)/|\mathbf{s}^*|}\frac{v_i v_i^T}{\sigma_i^2}(\mathbf{b}/\mu + \mathbf{A}_J\overrightarrow{\mathbf{1}}).$$

Substituting expansion

$$\frac{1}{1+\frac{t}{\mu\sigma_i^2}h(|\mathbf{s}^*|)/|\mathbf{s}^*|} = 1 - \frac{t}{\mu\sigma_i^2}h(|\mathbf{s}^*|)/|\mathbf{s}^*| + O(1/\mu^2)$$

in the second summand of the above expression yields that

$$\begin{aligned}\mathbf{s}^* = \frac{|\mathbf{s}^*|}{th(|\mathbf{s}^*|)}(\sum_{i=1}^{m-k} w_i w_i^T)\mathbf{b} + \sum_{i=1}^{k}\frac{v_i v_i^T}{\sigma_i^2}\mathbf{A}_J\overrightarrow{\mathbf{1}} + \frac{1}{\mu}\sum_{i=1}^{k}\frac{v_i v_i^T}{\sigma_i^2}\mathbf{b} \\ - \frac{t}{\mu}\frac{h(|\mathbf{s}^*|)}{|\mathbf{s}^*|}\sum_{i=1}^{k}\frac{v_i v_i^T}{\sigma_i^4}\mathbf{A}_J\overrightarrow{\mathbf{1}} + O(1/\mu^2).\end{aligned} \tag{2.28}$$

Substituting (2.26) and (2.27) into the equation (2.28), and equating coefficients of $1/\mu$ on both sides of the equality, we conclude that

$$\Psi = \frac{1}{t}\Gamma\sum_{i=1}^{m-k} w_i w_i^T\mathbf{b} + \sum_{i=1}^{k}\frac{v_i v_i^T}{\sigma_i^2}\mathbf{b} - t\frac{h(|\mathbf{s}_{opt}|)}{|\mathbf{s}_{opt}|}\sum_{i=1}^{k}\frac{v_i v_i^T}{\sigma_i^4}\mathbf{A}_J\overrightarrow{\mathbf{1}}.$$

Multiplying both sides of the above expression with $\mathbf{A}_J^T = \sum_i u_i\sigma_i v_i^T$ and using orthogonality of v_i's to w_i's, we have

$$\mathbf{A}_J^T\Psi = \mathbf{A}_J^T\sum_{i=1}^{k}\frac{v_i v_i^T}{\sigma_i^2}\mathbf{b} - t\frac{h(|\mathbf{s}_{opt}|)}{|\mathbf{s}_{opt}|}\mathbf{A}_J^T\sum_{i=1}^{k}\frac{v_i v_i^T}{\sigma_i^4}\mathbf{A}_J\overrightarrow{\mathbf{1}}. \tag{2.29}$$

From the definition of violation set J, we can conclude that the only rows of $\mathcal{S}(\mathbf{A}^T\mathbf{s}^*, \overrightarrow{\mathbf{1}})$ that are nonzero are exactly the elements of J. Therefore, it suffices to evaluate $\mathcal{S}(\mathbf{A}_J^T\mathbf{s}^*, \overrightarrow{\mathbf{1}})$. Furthermore, because $\mathbf{s}_{opt}$ is on the boundary of Ω and J is the set of violations of both $\mathbf{s}_{opt}$ and $\mathbf{s}^*$,

$$\mathbf{A}_J^T\mathbf{s}_{opt} = \overrightarrow{\mathbf{1}}, \qquad \text{and} \qquad \mathbf{A}_J^T\mathbf{s}^* \geq \overrightarrow{\mathbf{1}}. \tag{2.30}$$

Therefore,

$$\begin{aligned}\vec{0} \leq \mathbf{A}_J^T\mathbf{s}^* - \overrightarrow{\mathbf{1}} &= \mathbf{A}_J^T(\mathbf{s}_{opt} + \frac{1}{\mu}\Psi + O(1/\mu^2)) - \overrightarrow{\mathbf{1}} \\ &= (\mathbf{A}_J^T\mathbf{s}_{opt} - \overrightarrow{\mathbf{1}}) + \frac{1}{\mu}\mathbf{A}_J^T\Psi + O(1/\mu^2) \\ &= \frac{1}{\mu}\mathbf{A}_J^T\sum_{i=1}^{k}\frac{v_i v_i^T}{\sigma_i^2}\mathbf{b} - \frac{t}{\mu}\frac{h(|\mathbf{s}_{opt}|)}{|\mathbf{s}_{opt}|}\mathbf{A}_J^T\sum_{i=1}^{k}\frac{v_i v_i^T}{\sigma_i^4}\mathbf{A}_J\overrightarrow{\mathbf{1}} + O(1/\mu^2),\end{aligned} \tag{2.31}$$

where (2.30) and (2.29) was used to conclude the third line.

Because $y^*(t,\mu) = \mu\mathcal{S}(\mathbf{A}^T\mathbf{s}^*(t,\mu), \overrightarrow{\mathbf{1}})$ and the nonzero elements of $\mathcal{S}(\mathbf{A}^T\mathbf{s}^*(t,\mu), \overrightarrow{\mathbf{1}})$ are given by $\mathcal{S}(\mathbf{A}_J^T\mathbf{s}^*(t,\mu), \overrightarrow{\mathbf{1}})$, we conclude that the nonzero elements of $y^*(t,\mu)$,

for $\mu \geq \mu_c$, are given by

$$\begin{aligned}\mu\mathcal{S}(\mathbf{A}_J^T\mathbf{s}^*(t,\mu), \overrightarrow{\mathbf{1}}) &= \mathbf{A}_J^T\mathbf{s}^*(t,\mu) - \overrightarrow{\mathbf{1}} \\ &= \mathbf{A}_J^T \sum_{i=1}^{k} \frac{v_i v_i^T}{\sigma_i^2}\mathbf{b} - t\frac{h(|\mathbf{s}_{opt}|)}{|\mathbf{s}_{opt}|}\mathbf{A}_J^T \sum_{i=1}^{k} \frac{v_i v_i^T}{\sigma_i^4}\mathbf{A}_J \overrightarrow{\mathbf{1}} + O(1/\mu) \\ &= \mathbf{A}_J^T(\mathbf{A}_J\mathbf{A}_J^T)^\dagger\mathbf{b} - t\frac{h(|\mathbf{s}_{opt}|)}{|\mathbf{s}_{opt}|}\mathbf{A}_J^T((\mathbf{A}_J\mathbf{A}_J^T)^\dagger)^2\mathbf{A}_J \overrightarrow{\mathbf{1}} + O(1/\mu) \\ &= \mathbf{A}_J^\dagger\mathbf{b} - t\frac{h(|\mathbf{s}_{opt}|)}{|\mathbf{s}_{opt}|}\mathbf{A}_J^\dagger(\mathbf{A}_J^T)^\dagger \overrightarrow{\mathbf{1}} + O(1/\mu),\end{aligned}$$

where we used (2.31) for the second equality and the properties of the pseudo-inverse for the last equality.

Corollary 2.11. *In problem* (1.8)*, for each t, there exist fixed large μ_c such that for $\mu \geq \mu_c$, the solution to the corresponding problem is given as follows:*

Let J denote the set of violations of the maximizer of

$$\mathbf{s}_{opt} = \underset{\mathbf{s}}{\operatorname{argmax}} \{\langle \mathbf{b}, \mathbf{s}\rangle - tH(|\mathbf{s}|)\} \qquad \textit{subject to} \qquad \tilde{\mathbf{A}}^T\mathbf{s} \leq \overrightarrow{\mathbf{1}},$$

where $\tilde{\mathbf{A}}$ is given by (2.3)*. Let $\tilde{y}$ be a column vector with $2n$ entries where n is the number of columns of $\mathbf{A}$. Moreover, those entries of $\tilde{y}$ whose index belong to J are given by*

$$\tilde{y}(J) = \tilde{\mathbf{A}}_J^\dagger\mathbf{b} - t\frac{h(|\mathbf{s}_{opt}|)}{|\mathbf{s}_{opt}|}\tilde{\mathbf{A}}_J^\dagger(\tilde{\mathbf{A}}_J^T)^\dagger \overrightarrow{\mathbf{1}} + O(1/\mu),$$

and the other entries of $\tilde{y}$ are zero. Then the minimizer of problem (1.8) *is given by*

$$y^*(t,\mu) = \begin{bmatrix}\mathbf{I} & | & -\mathbf{I}\end{bmatrix}\tilde{y}.$$

Proof. From the Lagrangian dual correspondence in section 1, we know that the solution of problem (1.8) is equal to $\mu \cdot \mathsf{shrink}(\mathbf{A}^T\mathbf{s}^*, \overrightarrow{\mathbf{1}})$ where $\mathbf{s}^*$ is the solution of (1.9). Now use the result of this section and the relation (2.4). □

3. The TCCS algorithm

In this section we introduce the TCCS algorithm. The TCCS algorithm consists of two parts. The first part of the algorithm finds an approximate solution to the constrained problem (2.1). This part is described in subsection 3.1. The second part of the algorithm takes the result of the first part of the algorithm and then uses the result developed in subsection 2.3 to yield an approximate solution to problem (1.11). This part of the algorithm is described in subsection 3.2. Pseudo-codes for the TCCS algorithm are provided in the appendix. Although we emphasize that the TCCS algorithm finds an approximate solution to problem (1.11), numerical experiments in section 4 show that the algorithm perfectly recovers the solution when the solution is relatively sparse with respect to the number of measurements.

3.1. **First part.** For the first part of the TCCS algorithm we devise a fast algorithm that gives an approximate solution for constrained problem (2.1). Other methods could be used for this part of the TCCS algorithm. Nevertheless, we found the algorithm described here to be very effective and fast for optimizing constrained problems of form (2.1).

The idea of the algorithm is to trace out an approximation to the path of $\mathbf{s}_{opt}(\tau)$, as τ decreases, which was described in section 2.2. We denote the approximate path by $\tilde{\mathbf{s}}(\tau)$. Indeed to construct $\tilde{\mathbf{s}}(\tau)$, we proceed very similarly to the way the trajectory $\mathbf{s}_{opt}(\tau)$ was determined at the end of section 2.2. The only difference is that whenever we hit a new face of polytope Ω, say at τ_i, we use J_i as proxy for K_i and proceed. As a result, the trajectory of $\tilde{\mathbf{s}}(\tau)$ might be different than $\mathbf{s}_{opt}(\tau)$. We make this approximation to avoid cumbersome computations. This approximation is somehow similar to the approximation that was made in [10]. The details of the algorithm are provided below.

We initialize the algorithm by finding $\tilde{\tau}_0$ large enough so that

$$\frac{1}{|\mathbf{b}|}h^{-1}(\frac{|\mathbf{b}|}{\tilde{\tau}_0})\mathbf{b}$$

lies in the interior of Ω. Set $\tilde{\mathbf{s}}(\tilde{\tau}_0)$ to be equal to the above expression. Set $\tilde{J} = \emptyset$ and $\tau = \tilde{\tau}_0$. This completes the initialization of the algorithm. Next we go through a loop described as follows.

Suppose $\tau = \tilde{\tau}_r$, and $\tilde{J} = \tilde{J}_r$ denotes the set of active constraints for $\tilde{\mathbf{s}}(\tilde{\tau}_r)$. Since $\alpha(\tilde{J}_r)$ is the projection of $\mathbf{b}$ on the set $\{\mathbf{s} : \mathbf{A}_{\tilde{J}_r}^T \mathbf{s} = 1\}$, we have that $a_i^T \overrightarrow{\alpha}(\tilde{J}_r) = 0$ for $i \in \tilde{J}_r$. Therefore, when τ is decreasing from $\tilde{\tau}_r$ and $\tilde{\mathbf{s}}(\tau)$ is moving along $\overrightarrow{\alpha}(\tilde{J}_r)$, we have $a_i^T \tilde{\mathbf{s}}(\tau) = 1$ for $i \in \tilde{J}_r$. Thus, constraints given by $\tilde{J}_r$ continue to be active for $\tilde{\mathbf{s}}(\tau)$.

Let L be the complement of set $\tilde{J}_r$; that is, $L = \{1, \ldots, n\} \setminus \tilde{J}_r$. Let $\tilde{\tau}_{r+1}$ denote the largest τ smaller than $\tilde{\tau}_r$ such that the set of active constraints of

$$\tilde{\mathbf{s}}(\tau) = F(\tau, \tilde{J}_r)\overrightarrow{\alpha}(\tilde{J}_r) + \overrightarrow{\beta}(\tilde{J}_r) = \tilde{\mathbf{s}}(\tilde{\tau}_r) + \Big(F(\tau, \tilde{J}_r) - F(\tilde{\tau}_r, \tilde{J}_r)\Big)\overrightarrow{\alpha}(\tilde{J}_r) \tag{3.1}$$

is different than $\tilde{J}_r$. From the discussion above, $\tilde{\tau}_{r+1}$ is the largest τ smaller than $\tilde{\tau}_r$ so that for some $i \in L$, $a_i^T \tilde{\mathbf{s}}(\tau) = 1$, where $\tilde{\mathbf{s}}(\tau)$ is given by equation (3.1). Finding $\tilde{\tau}_{r+1}$ is problem specific and depends on the function $F(\tau, J)$.

To find $\tilde{\tau}_{r+1}$, we first evaluate the smallest positive number, denoted by k, such that

$$a_i^T \Big(\tilde{\mathbf{s}}(\tilde{\tau}_r) + k\overrightarrow{\alpha}(\tilde{J}_r)\Big) = 1$$

for some $i \in L$. Thus, k is the smallest positive number in the set

$$\left\{ \frac{1 - a_i^T \tilde{\mathbf{s}}(\tilde{\tau}_r)}{a_i^T \overrightarrow{\alpha}(\tilde{J}_r)} \right\}_{i \in L}$$

(such k always exist because we assumed that polytope Ω is bounded). Once k is evaluated, $\tilde{\tau}_{r+1}$ is determined such that

$$F(\tilde{\tau}_{r+1}, \tilde{J}_r) - F(\tilde{\tau}_r, \tilde{J}_r) = k. \tag{3.2}$$

For example, in the case of $H(|\mathbf{s}|) = |\mathbf{s}|^2/2$ (see remark 2.9), equation (3.2) becomes

$$\frac{1}{\tilde{\tau}_{r+1}} - \frac{1}{\tilde{\tau}_r} = k,$$

and in the case of $H(|\mathbf{s}|) = \sqrt{1 + |\mathbf{s}|^2}$ (see remark 2.9), equation (3.2) becomes

$$\sqrt{\frac{1 + |\overrightarrow{\beta}(\tilde{J}_r)|^2}{\tilde{\tau}_{r+1}^2 - |\overrightarrow{\alpha}(\tilde{J}_r)|^2}} - \sqrt{\frac{1 + |\overrightarrow{\beta}(\tilde{J}_r)|^2}{\tilde{\tau}_r^2 - |\overrightarrow{\alpha}(\tilde{J}_r)|^2}} = k.$$

If $\tilde{\tau}_{r+1} < t$, then

$$\tilde{\mathbf{s}}(t) = \tilde{\mathbf{s}}(\tilde{\tau}_r) + \Big(F(t, \tilde{J}_r) - F(\tilde{\tau}_r, \tilde{J}_r)\Big)\, \overrightarrow{\alpha}(\tilde{J}_r);$$

otherwise, we set $\tau = \tilde{\tau}_{r+1}$ and $J = \tilde{J}_{r+1}$ and repeat the above process.

In this way we get a sequence $\tilde{\tau}_0 > \tilde{\tau}_1 > \cdots$. In general, this sequence is different than the sequence (2.25), except at $\tilde{\tau}_0 = \tau_0$ and $\tilde{\tau}_1 = \tau_1$. Also note that, at each step of the loop the rank of $\mathbf{A}_{\tilde{J}}^T$ strictly increases. Because of this, at $\tau = \tilde{\tau}_j$, for some $j \leq m$, $\tilde{\mathbf{s}}(\tau)$ reaches an extreme point of Ω. For $\tau < \tilde{\tau}_j$, $\tilde{\mathbf{s}}(\tau)$ stays at the same extreme point, and the algorithm outputs the same result. Thus, the algorithm goes through at most m loops. This feature makes this algorithm very efficient (in particular when $m \ll n$) in comparison to other available algorithms for optimizing problems of the form (2.1).

We also note that even though the first part of the algorithm outputs an approximate solution $\tilde{\mathbf{s}}(t)$ to constrained problem (2.1), using KKT conditions it is easy to verify whether $\tilde{\mathbf{s}}(t)$ is indeed the optimal solution $\mathbf{s}_{opt}(t)$ or not.

3.2. **Second part.** In this subsection we explain how $\mathbf{s}_{opt}(t)$ obtained in the first part of the TCCS algorithm is used to obtain an approximate solution to problem (1.11). Let $y_{opt}(t)$ denote the solution to problem (1.11). As $\mu \to \infty$, the solution to problem (1.8), $y^*(t, \mu)$, provides a good approximate for $y_{opt}(t)$. Hence, in view of Corollary 2.11, we find an approximation for $y_{opt}(t)$ using $\mathbf{s}_{opt}(t)$ in the following way:

Let J denote the set of violations of $\mathbf{s}_{opt}(t)$. Suppose $\tilde{y}_{opt}$ is a column vector with $2n$ entries where n is the number of columns of $\mathbf{A}$. Set those entries of $\tilde{y}_{opt}$ whose index belongs to J by

$$\tilde{y}_{opt}(J) = \tilde{\mathbf{A}}_J^{\dagger}\mathbf{b} - t\frac{h(|\mathbf{s}_{opt}|)}{|\mathbf{s}_{opt}|}\tilde{\mathbf{A}}_J^{\dagger}(\tilde{\mathbf{A}}_J^T)^{\dagger}\overrightarrow{\mathbf{1}},$$

and the other entries of $\tilde{y}_{opt}$ are set to zero. Now

$$\begin{bmatrix}\mathbf{I} & | & -\mathbf{I}\end{bmatrix}\tilde{y}_{opt}$$

provides a good approximation for $y_{opt}(t)$

3.3. **Application to compressed sensing.** Observe that problem (1.11) reduces to Basis Pursuit problem (1.2) when $t = 0$. Therefore, when $t = 0$, we expect the TCCS algorithm to output a sparse solution for compressed sensing problem (1.1). In section 4 we provide numerical evidence in support of this.

To summarize, the TCCS algorithm approaches the compressed sensing problem (1.1) in the following way. We try to find an approximate optimizer for problem (1.2). To that end, we generate the sequence $\tau_0 > \tau_1 > \cdots > \tau_\ell = 0$ and iteratively find approximate optimizers $\tilde{\mathbf{s}}(\tau_i)$ to problem (1.11) with τ_i in place of t. Finally, we use the result of Corollary 2.11 to find an approximate solution to problem (1.11).

One might question the necessity of introducing parameter μ in our methodology to find a solution to Basis Pursuit problem (1.2). Indeed, the value of μ does not appear in the TCCS algorithm (i.e. in the second part of the TCCS algorithm we only make the assumption that μ is sufficiently large). Nevertheless, parameter μ was important in establishing Corollary 2.11 and identifying a good approximate for $y_{opt}(t)$ in subsection 3.2.

On the other hand, introducing the parameter t (and eventually setting it equal to 0), enables us to use a homotopy technique to find an approximate solution to linear programming problem

$$\underset{\mathbf{s}}{\operatorname{argmax}} \ \langle \mathbf{b}, \mathbf{s} \rangle \quad \text{subject to} \quad -\overrightarrow{\mathbf{1}} \leq \mathbf{A}^T \mathbf{s} \leq \overrightarrow{\mathbf{1}}. \tag{3.3}$$

4. Numerical results

In this section we compare the performance of the TCCS algorithm with the LBSB algorithm [23], OMP algorithm [14,16,19] and FISTA [2]. It is important to note that the TCCS and FISTA algorithm can solve the more general problem of (1.11) (i.e. not only the Basis Pursuit problem (1.2), which corresponds to $t = 0$ in (1.11)); whereas, LBSB and OMP algorithm can be applied only to problems (1.2) and (1.1), respectively.

For the numerical experiments, we generate entries of the matrix $\mathbf{A}$ from i.i.d. normal distributions with mean 0 and standard deviation 1. We also generate the "true signal" y_s with a given sparsity level. Define the *sparsity level* of vector y_s to be the percentage ratio of the number of nonzero components of y_s with respect to the number of measurements (i.e. m). That is,

$$\text{sparsity level of } y_s = \frac{\#\text{ nonzero components of } y_s}{m} \times 100\%.$$

In each setting we run 10 independent trials and report the average processing time and the average relative error of each method for the 10 trials.

4.1. TCCS versus LBSB.

Here we compare the performance of the TCCS algorithm applied to the compressed sensing problem (i.e. $t = 0$) with the LBSB algorithm to solve problem (1.2). The LBSB algorithm is one of the fastest methods used for compressed sensing problems and in [23], extensive numerical experiments comparing this method with respect to other methods were done.

In each test, we generate the true signal y_s by first determining its support (i.e. the nonzero entries) at random and then generate each entry of the support using i.i.d. normal distributions with mean 0 and standard deviation 1. Vector $\mathbf{b}$ is obtained via

$$\mathbf{b} = \mathbf{A} y_s.$$

For the LBSB method, we set $\alpha = 10$, $\lambda = 0.4$ and the maximum number of iterations equal to 2000. The TCCS algorithm applied to compressed sensing problem does not require any parameter to be set. In this subsection, relative error is defined by

$$\frac{\|y - y_s\|_2}{\|y_s\|_2}.$$

In Tables 1 and 2 we compare performance of LBSB algorithm and TCCS algorithm when sparsity level is 2% and 10%, respectively.

For the numerical experiments that are presented in Tables 1 and 2, we chose matrix $\mathbf{A}$ to be very narrow and wide, to highlight the advantage of the TCCS algorithm over the LBSB algorithm in this regime, both in processing time and relative error of the solution. Remarkably, as shown in Table 1, when sparsity level is low, the TCCS algorithm perfectly recovers the sparse solution in all 10 independent trials.

TABLE 1. The comparison of the performance of the TCCS algorithm and the LBSB algorithm. "Time" and "error", respectively, refer to processing time (in seconds) and relative error for each method. Here, sparsity level is 2% and each row is the result of 10 independent trials.

m	n	LBSB time	LBSB error	TCCS time	TCCS error
200	10000	5.65	7.04e-03	0.13	1.07e-15
200	50000	56.71	1.21e-01	0.68	1.22e-15
200	100000	114.34	4.16e-01	1.55	9.03e-16
400	10000	11.22	1.66e-03	0.51	2.00e-15
400	50000	104.82	2.83e-02	2.65	1.53e-15
400	100000	229.97	1.11e-01	6.04	1.23e-15

TABLE 2. The comparison of the performance of the TCCS algorithm and the LBSB algorithm. "Time" and "error", respectively, refer to processing time (in seconds) and relative error for each method. Here, sparsity level is 10% and each row is the result of 10 independent trials.

m	n	LBSB time	LBSB error	TCCS time	TCCS error
200	10000	11.36	5.81e-02	2.22	2.62e-03
200	50000	56.72	3.59e-01	32.10	7.54e-01
200	100000	115.50	4.27e-01	71.05	3.49e-01
400	10000	22.23	3.80e-03	4.60	2.03e-15
400	50000	113.28	1.57e-01	83.21	4.77e-01
400	100000	230.65	3.62e-01	286.82	6.62e-01

In the next set of experiments, for different values of m and n, we compare the performance of the LBSB algorithm and the TCCS algorithm over a different range of sparsity levels. The results are shown in Figures 1, 2 and 3.

As can be seen from the figures, for low sparsity levels, the TCCS algorithm is again both much faster and more accurate in recovering the solution than the LBSB algorithm. Indeed, as observed earlier, for low sparsity levels, the TCCS algorithm perfectly recovers the sparse solution in all independent trials. For higher sparsity levels, the relative error of the TCCS algorithm and the LBSB algorithm is comparable. At these sparsity levels, the TCCS algorithm performs faster than the LBSB algorithm for very small ratios m/n, whereas, the converse becomes true for higher ratios of m/n (i.e., compare the right panels in Figure 2). It is also noteworthy that as the sparsity level increases, the TCCS algorithm exhibits a sharp phase transition in the success rate of recovery of the sparse solution.

4.2. **TCCS versus OMP.** Here we numerically investigate the difference between the TCCS algorithm applied to the compressed sensing problem (i.e. $t = 0$) and the OMP algorithm to solve problem (1.1). The OMP algorithm has been noted for its speed and ease of implementation. Furthermore, [19] provides some theoretical and numerical evidence for the reliability of the OMP algorithm.

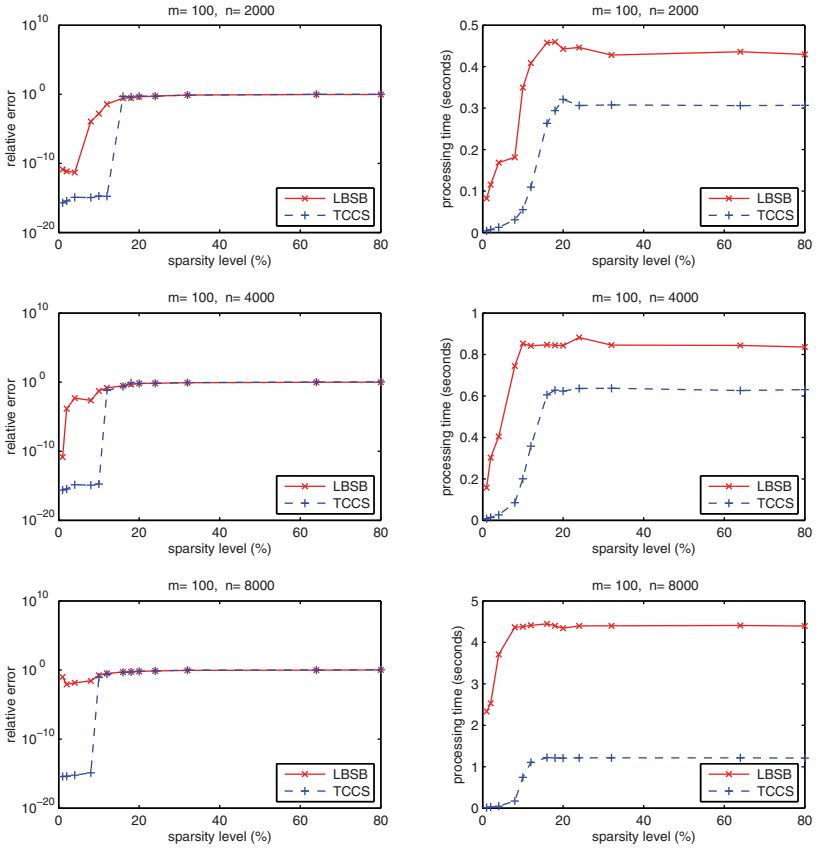

FIGURE 1. Comparison of the performance of the LBSB algorithm and the TCCS algorithm in terms of processing time and relative error for different sparsity levels. Here $m = 100$ and $n = 2000, 4000, 8000$. The left panels show semilog plot of the relative error, while the right panels show plot of processing time (in seconds). Each data point is the average of 10 independent trials.

It is important to note that the methodologies of these two algorithms are quite different despite some superficial similarities in their pseudo-codes. The OMP algorithm is a greedy based method that solves the compressed sensing problem by iteratively increasing the support of y with components whose correlation to the current residual is maximum. On the other hand, the TCCS algorithm uses a homotopy approach to trace the optimal solution of the dual problem.

In this subsection vector $\mathbf{b}$ is obtained via

$$\mathbf{b} = \mathbf{A}y_s,$$

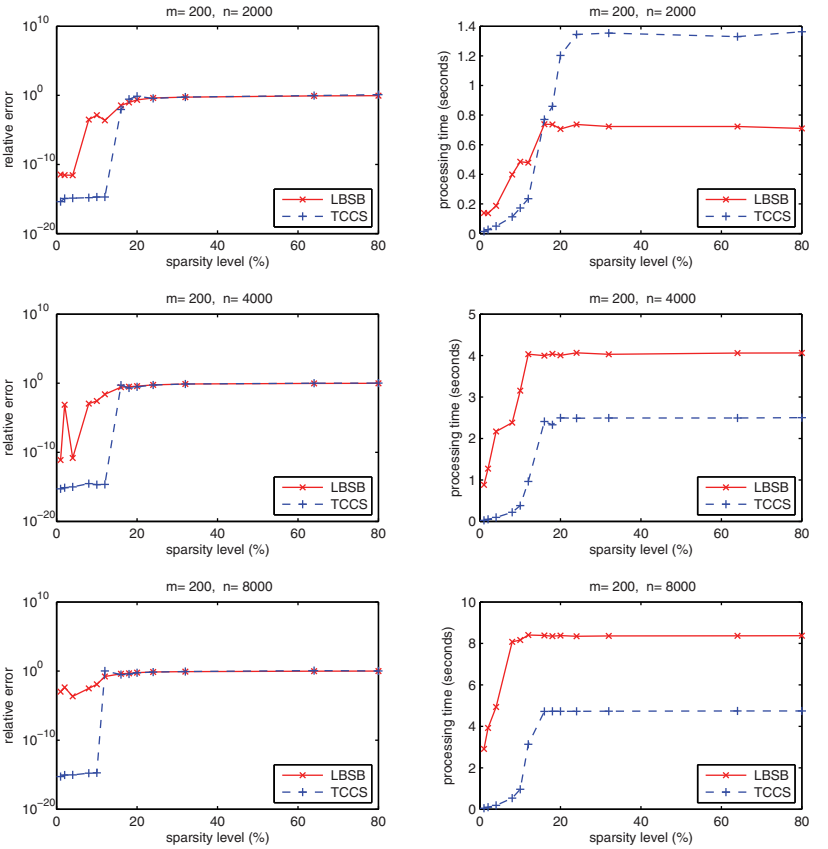

FIGURE 2. Comparison of the performance of the LBSB algorithm and the TCCS algorithm in terms of processing time and relative error for different sparsity levels. Here $m = 200$ and $n = 2000, 4000, 8000$. The left panels show semilog plot of the relative error, while the right panels show plot of processing time (in seconds). Each data point is the average of 10 independent trials.

and relative error is defined by

$$\frac{\|y - y_s\|_2}{\|y_s\|_2}.$$

We run two sets of experiments similar to the numerical experiments done in [4]. Because both the TCCS and the OMP algorithm run very fast, we only present the relative error of the two algorithms. In the first experiment, we generate the true signal y_s in the same fashion as in subsection 4.1. The results are shown in Figure 4.

Although the OMP algorithm outperforms the TCCS algorithm in Figure 4, we will see that this is not the case in general. For the second set of experiments, we generate the true signal y_s by first determining its support (i.e. the nonzero entries) at random and then set each entry of the support to be $+1$ or -1 at random with

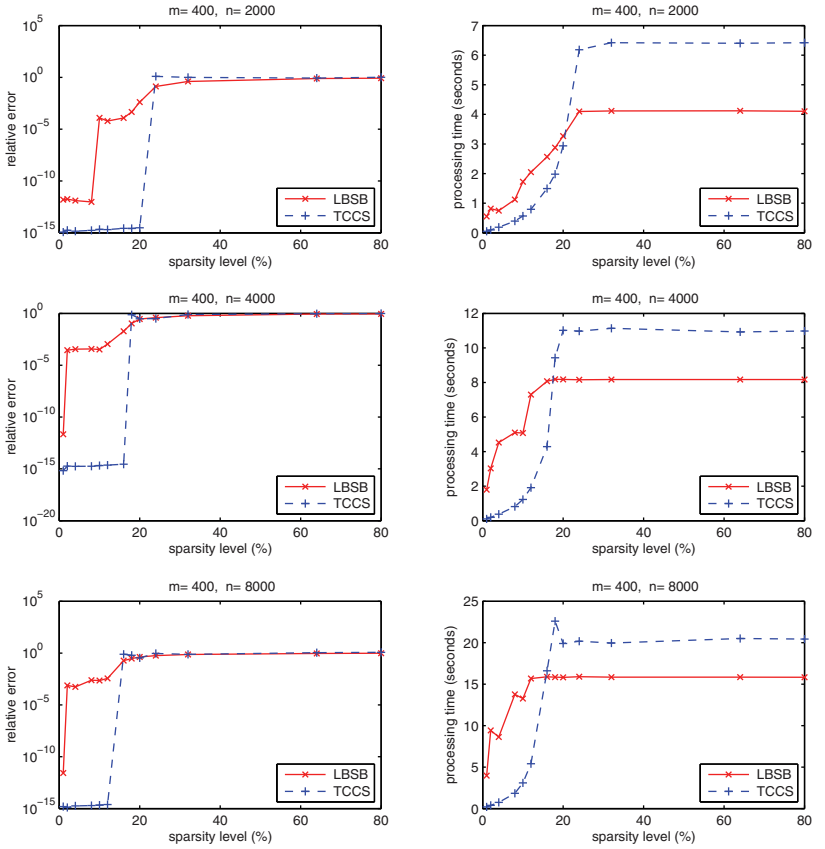

FIGURE 3. Comparison of the performance of the LBSB algorithm and the TCCS algorithm in terms of processing time and relative error for different sparsity levels. Here $m = 400$ and $n = 2000, 4000, 8000$. The left panels show semilog plot of the relative error, while the right panels show plot of processing time (in seconds). Each data point is the average of 10 independent trials.

the same probability. The results are shown in Figure 5. As can be seen, the TCCS algorithm outperforms the OMP algorithm in this example.

4.3. **TCCS versus FISTA.** Here we compare the performance of the TCCS algorithm with FISTA algorithm (with constant stepsize) to solve problem (1.5) with $t = 0.1$ and $t = 0.01$. FISTA is a very efficient algorithm that uses the forward-backward splitting method to solve a general class of optimization problems including (1.5). FISTA can be thought as an extension of FPC [12] (also called proximal gradient method (PGM)), as it applies the prox-operator at a specific linear combination of the previous two iterates. For the ease of reference, the outline of FISTA algorithm with constant stepsize applied to problem (1.5) is presented in Algorithm 1.

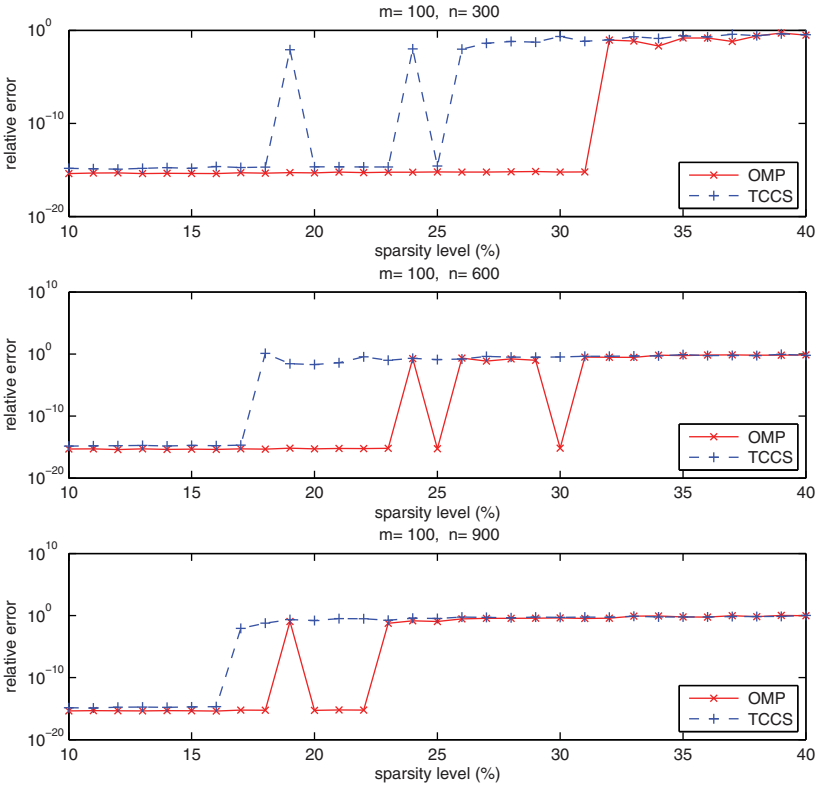

FIGURE 4. Comparison of the relative error of the OMP algorithm and the TCCS algorithm for different sparsity levels. Here the entries of the support of the true signal are chosen from i.i.d. normal distributions. Each data point is the average of 10 independent trials.

Algorithm 1: The FISTA algorithm with constant stepsize applied to LASSO problem (1.5).

Input: A, b and N.
Set $L = \lambda_{\max}(A^T A)/t$
Set $y_1 = z_0 = \vec{0}$, $t_1 = 1$
for $k = 1, \ldots, N$ **do**
  $z_k = \mathsf{shrink}(y_k - \frac{1}{tL} A^T(Ay_k - b), \frac{1}{L})$
  $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
  $y_{k+1} = z_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(z_k - z_{k-1})$
end
return y_N

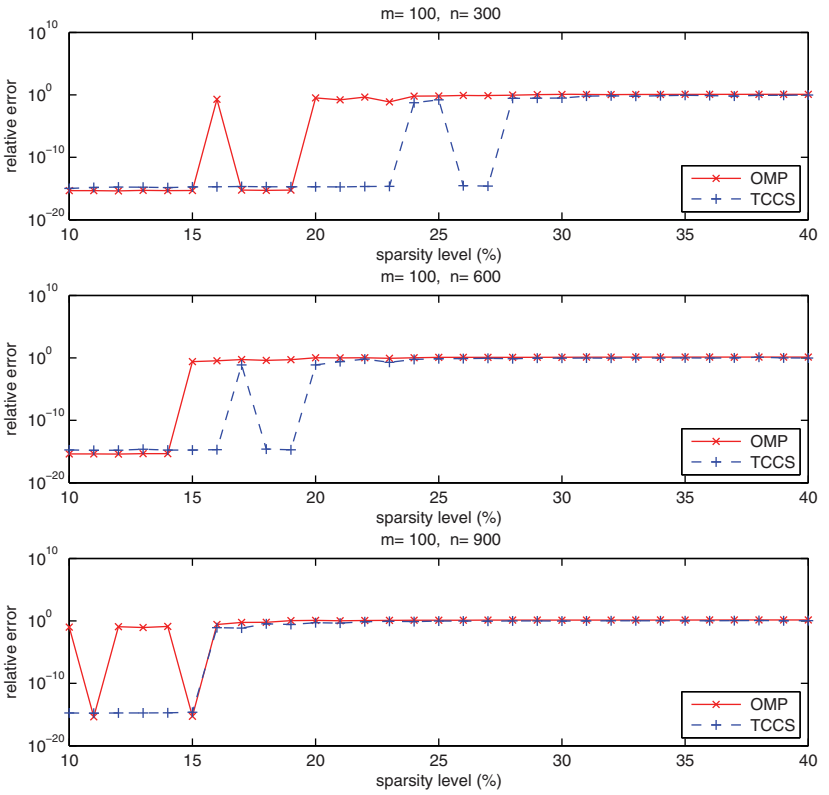

FIGURE 5. Comparison of the relative error of the OMP algorithm and the TCCS algorithm for different sparsity levels. Here the entries of the support of the true signal are chosen from $+1$ or -1 at random. Each data point is the average of 10 independent trials.

In each test, y_s is generated in the same way as in subsection 4.1. However, vector $\mathbf{b}$ is obtained via

$$\mathbf{b} = A(y_s + \epsilon_1) + \epsilon_2,$$

where ϵ_1 and ϵ_2 are vectors whose entries are generated using i.i.d. normal distributions with means 0 and standard deviations, respectively, σ_1 and σ_2 (i.e. both the signal and the measurements are corrupted with Gaussian white noise). In the experiments of this subsection, we set $\sigma_1 = 5 \times 10^{-5}$ and $\sigma_2 = 10^{-5}$. In this subsection, the relative error is defined by

$$\frac{\|y - y^*\|_2}{\|y^*\|_2},$$

where y^* is the proxy for the solution of (1.5) obtained by allowing FISTA algorithm to run for many iterations (e.g. $N = 5000$).

Figures 6 and 7 compare the performance of the FISTA algorithm (with the number of iterations $N = 200$) and the TCCS algorithm for different values of m and n and over different range of sparsity levels.

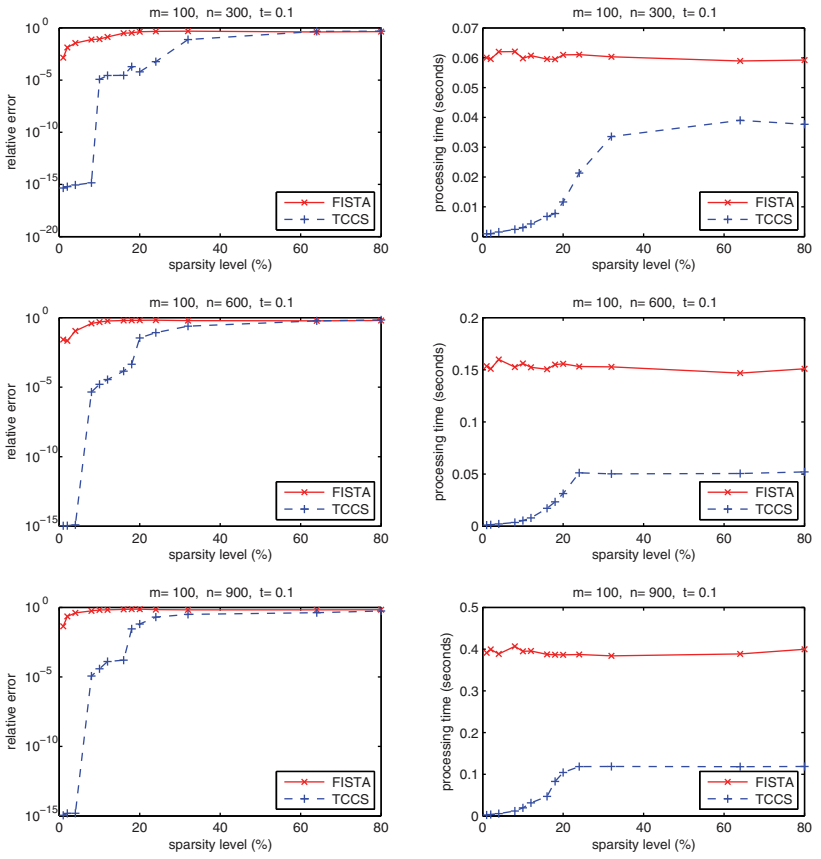

FIGURE 6. Comparison of the performance of the FISTA algorithm and the TCCS algorithm in terms of processing time and relative error for different sparsity levels to solve problem (1.5) with $t = 0.1$. The left panels show the semilog plot of the relative error, while the right panels show the plot of processing time (in seconds). Each data point is the average of 10 independent trials.

As it can be seen from the figures the performance of the TCCS algorithm compares well with the FISTA algorithm, specially for low sparsity levels. Clearly the relative error of FISTA algorithm would decrease if the number of iterations is increased; however, that would increase the processing time. In these experiments, the number of iterations $N = 200$ was chosen for FISTA algorithm, so that we can get a good comparison to the TCCS algorithm for both the relative error and the processing time.

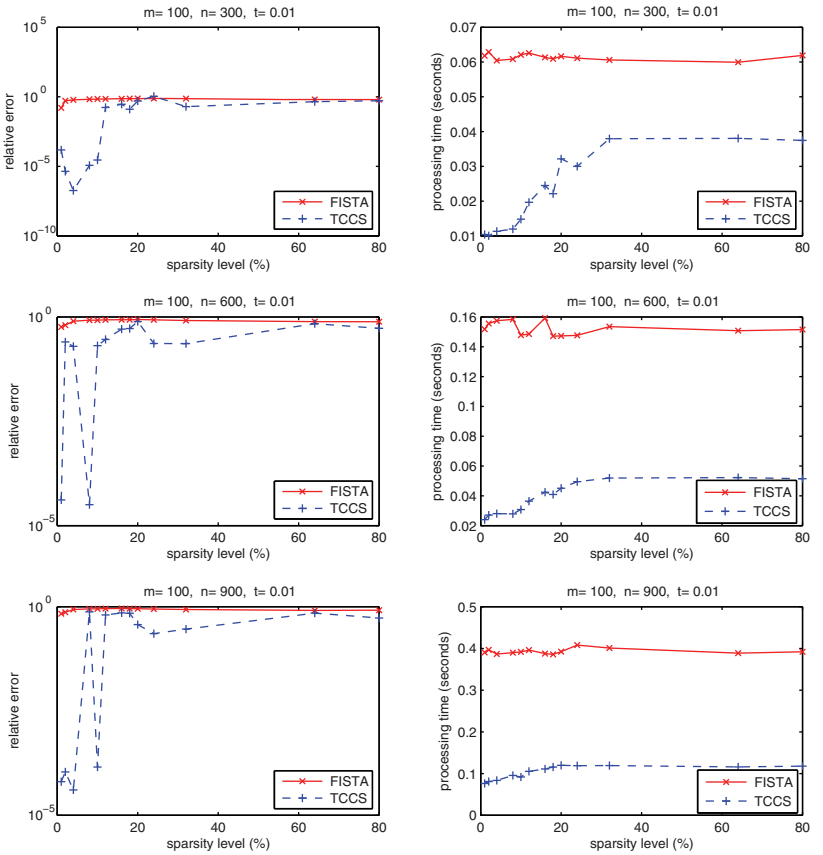

FIGURE 7. Comparison of the performance of the FISTA algorithm and the TCCS algorithm in terms of processing time and relative error for different sparsity levels to solve problem (1.5) with $t = 0.01$. The left panels show semilog plot of the relative error, while the right panels show plot of processing time (in seconds). Each data point is the average of 10 independent trials.

5. Conclusion

In this paper we developed an algorithm for finding the approximate solution to problems of the form (1.11). In particular, by setting $t = 0$, our algorithm outputs a good approximate for the solution to the compressed sensing problem (1.1). Numerical results show that the algorithm perfectly recovers the solution to the compressed sensing problem when it is relatively sparse with respect to the number of measurements. For this regime of sparsity, the algorithm recovers the solution extremely fast. Numerical results are also presented to show the advantages of the algorithm to solve problems such as the LASSO problem.

Another advantage of our algorithm is that it is parameter free except a tolerance parameter that is due to numerical machine precision. In many other optimization

methods, one needs to introduce extra parameters whose value greatly affect the performance of the algorithm.

We also observed a sharp phase transition in the success rate of recovery of the solution of the compressed sensing problem by our algorithm as the sparsity of the solution varies. This suggests that a theoretical analysis should be conducted relating to this phenomena.

Appendix A. Pseudo-code

In this section we provide a pseudo-code for the TCCS algorithm when $H(|\mathbf{s}|) = |\mathbf{s}|^2/2$ and $t \geq 0$, Algorithm 2, and for $t = 0$, Algorithm 3. Indeed, Algorithm 3 is an easy adaptation of Algorithm 2. The output of Algorithms 2 and 3 yiels an approximate solution to the LASSO problem (1.5) and the compressed sensing problem (1.1), respectively.

Some remarks regarding the implementation of the pseudo-codes are in order. In each iteration of the algorithms we need to compute α, which is the projection of b on the space $\{u : A_J^T u = \overrightarrow{\mathbf{1}}\}$. There are several ways to do this. One way is to use the formula $b - \tilde{A}_J(\tilde{A}_J^T\tilde{A}_J)^\dagger \tilde{A}_J^T b$ and find the pseudo-inverse via the SVD decomposition. Another way is to use a QR decomposition to compute the least square solution. For example, we can use the expression $\alpha = b - A_J((A_J^T A_J)\backslash(A_J^T b))$ in Matlab for this purpose (i.e., note that $A_J^T A_J$ is a square matrix, and therefore, backslash operator in Matlab yields the least square solution).

Also observe that in the pseudo-codes, we introduced a tolerance parameter ϵ. Due to numerical machine precision, we need to use lines

$$\|\alpha\|_\infty < \epsilon \qquad \text{and} \qquad J = \{i : |a_i^T s - 1| < \epsilon\},$$

in place of lines

$$\alpha == 0 \qquad \text{and} \qquad J = \{i : a_i^T s == 1\}.$$

The pseudo-codes provided here are meant to clarify the ideas and they do not necessarily implement the TCCS algorithm in the most efficient way. In particular, one could avoid introducing the matrix $\tilde{\mathbf{A}}$ and work with the matrix $\mathbf{A}$ throughout the algorithm.

Algorithm 2: The TCCS algorithm when $H(|\mathbf{s}|) = |\mathbf{s}|^2/2$ and $t \geq 0$. The output y_{opt} of this pseudo-code yields approximate solution to problem (1.5). Also, s_{opt} is the approximate solution to (1.10). Note that ϵ is not a real parameter of the algorithm, it is rather a tolerance parameter that is due to numerical machine precision.

Input: A, t, b, and ϵ.
First part: $\tilde{A} = [A \mid -A]$
Choose τ_0 large enough so that $\tilde{A}^T(b/\tau_0) \leq 1$.
$J = \emptyset$
$s = b/\tau_0$
while $r = 0, \ldots$ **do**
  $\alpha =$ projection of b on the space $\{u : A_J^T u = \overrightarrow{\mathbf{1}}\}$.
  if $\|\alpha\|_\infty < \epsilon$ **then**
    // s has reached an extreme point of polytope at τ_r and will not move thereafter.
    $s_{opt} = s$
    break
  end
  $L = \{1, \ldots, n\} \setminus J_r$
  $v = (\overrightarrow{\mathbf{1}} - \tilde{A}_L^T s)./\tilde{A}_L^T \alpha$ // entrywise division.
  $k =$ minimum positive entry of v.
  // k is the smallest positive number such that $a_i^T(s + k\alpha) = 1$ for some $i \in L$.
  Find τ_{r+1} such that $\frac{1}{\tau_{r+1}} - \frac{1}{\tau_r} = k$.
  if $\tau_{r+1} < t$ **then**
    $\tilde{k} = \frac{1}{t} - \frac{1}{\tau_r}$
    $s_{opt} = s + \tilde{k}\alpha$
    break
  end
  $s = s + k\alpha$
  $J = \{i : |a_i^T s - 1| < \epsilon\}$
end
Second part: $\tilde{y}_{opt} = \vec{0}$
$\tilde{y}_{opt}(J) = \tilde{A}_J^\dagger b - t\tilde{A}_J^\dagger (\tilde{A}_J^T)^\dagger \overrightarrow{\mathbf{1}}$ // set entries of $\tilde{y}_{opt}$ that belong to J.
$y_{opt} = [I \mid -I]\tilde{y}_{opt}$
return y_{opt}

Algorithm 3: The TCCS Algorithm for compressed sensing problems. This algorithm is an adaptation of Algorithm 2 when $t = 0$. The output of this pseudo-code yields an approximate solution to problem (1.1). Also, s_{opt} is the approximate solution to (3.3). Note that ϵ is not a real parameter of the algorithm, it is rather a tolerance parameter that is due numerical machine precision.

Input: A, b, and ϵ.
First part: $\tilde{A} = [A \mid -A]$
Choose τ_0 large enough so that $\tilde{A}^T(b/\tau_0) \leq 1$.
$J = \emptyset$
$s = b/\tau_0$
while $r = 0, \ldots$ **do**
  $\alpha =$ projection of b on the space $\{u : A_J^T u = \overrightarrow{1}\}$.
  if $\|\alpha\|_\infty < \epsilon$ **then**
    // s has reached an extreme point of polytope at τ_r and will not move thereafter.
    $s_{opt} = s$
    break
  end
  $L = \{1, \ldots, n\} \setminus J_r$
  $v = (\overrightarrow{1} - \tilde{A}_L^T s)./\tilde{A}_L^T \alpha$ // entrywise division.
  $k =$ minimum positive entry of v.
  // k is the smallest positive number such that $a_i^T(s + k\alpha) = 1$ for some $i \in L$.
  Find τ_{r+1} such that $\frac{1}{\tau_{r+1}} - \frac{1}{\tau_r} = k$.
  $s = s + k\alpha$
  $J = \{i : |a_i^T s - 1| < \epsilon\}$
end
Second part: $\tilde{y}_{opt} = \vec{0}$
$\tilde{y}_{opt}(J) = \tilde{A}_J^\dagger b$ // set entries of $\tilde{y}_{opt}$ that belong to J.
$y_{opt} = [I \mid -I]\tilde{y}_{opt}$
return y_{opt}

Acknowledgements

The authors benefited greatly from valuable conversations with Russel Caflisch, Jerome Darbon, Damek Davis, Yi Yang and Wotao Yin during the completion of this project. The authors would also like to thank the anonymous referees for their insightful remarks and constructive suggestions.

References

[1] J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA J. Numer. Anal. **8** (1988), no. 1, 141–148, DOI 10.1093/imanum/8.1.141. MR967848 (90h:65113)

[2] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci. **2** (2009), no. 1, 183–202, DOI 10.1137/080716542. MR2486527 (2010d:35390)

[3] S. R. Becker, E. J. Candès, and M. C. Grant, *Templates for convex cone problems with applications to sparse signal recovery*, Math. Program. Comput. **3** (2011), no. 3, 165–218, DOI 10.1007/s12532-011-0029-5. MR2833262 (2012h:90107)

[4] M. Burger, M. Möller, M. Benning, and S. Osher, *An adaptive inverse scale space method for compressed sensing*, Math. Comp. **82** (2013), no. 281, 269–299, DOI 10.1090/S0025-5718-2012-02599-3. MR2983025

[5] E. J. Candes and T. Tao, *Decoding by linear programming*, IEEE Trans. Inform. Theory **51** (2005), no. 12, 4203–4215, DOI 10.1109/TIT.2005.858979. MR2243152 (2007b:94313)

[6] S. Chen and D. Donoho, *Basis Pursuit*, Conference on Signals, Systems and Computers., v1, (1994), pp. 41–44.

[7] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM Rev. **43** (2001), no. 1, 129–159, DOI 10.1137/S003614450037906X. Reprinted from SIAM J. Sci. Comput. **20** (1998), no. 1, 33–61 (electronic) [MR1639094 (99h:94013)]. MR1854649

[8] J. Darbon, *On convex finite-dimensional variational methods in imaging sciences and Hamilton-Jacobi equations*,UCLA CAM report 13-59, (2013).

[9] J. Darbon and S. Osher, *Initial Value Problems for Hamilton-Jacobi Equations and Sparsity for Linear Systems via ℓ^1 Related Optimization*, (2013), preprint.

[10] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, *Least angle regression*, Ann. Statist. **32** (2004), no. 2, 407–499, DOI 10.1214/009053604000000067. With discussion, and a rejoinder by the authors. MR2060166 (2005d:62116)

[11] M. P. Friedlander and P. Tseng, *Exact regularization of convex programs*, SIAM J. Optim. **18** (2007), no. 4, 1326–1350, DOI 10.1137/060675320. MR2373304 (2008k:90092)

[12] E. T. Hale, W. Yin, and Y. Zhang, *Fixed-point continuation for l_1-minimization: methodology and convergence*, SIAM J. Optim. **19** (2008), no. 3, 1107–1130, DOI 10.1137/070698920. MR2460734 (2009j:90070)

[13] M.-J. Lai and W. Yin, *Augmented ℓ_1 and nuclear-norm models with a globally linearly convergent algorithm*, SIAM J. Imaging Sci. **6** (2013), no. 2, 1059–1091, DOI 10.1137/120863290. MR3062582

[14] S. G. Mallat, Z. Zhang, *Matching pursuits with time-frequency dictionaries*, IEEE Transactions on Signal Processing, Issue 12, (1993), pp. 3397–3415.

[15] Yu. E. Nesterov, *A method for solving the convex programming problem with convergence rate $O(1/k^2)$* (Russian), Dokl. Akad. Nauk SSSR **269** (1983), no. 3, 543–547. MR701288 (84i:90119)

[16] Y. C. Pati, R. Rezaiifar, P. S. Krishnaprasad, *Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition*, Proceedings of the 27th Annual Asilomar Conference on Signals, Systems, and Computers, (1993), pp. 40–44.

[17] F. Schöpfer, *Exact regularization of polyhedral norms*, SIAM J. Optim. **22** (2012), no. 4, 1206–1223, DOI 10.1137/11085236X. MR3023770

[18] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. Roy. Statist. Soc. Ser. B **58** (1996), no. 1, 267–288. MR1379242 (96j:62134)

[19] J. A. Tropp and A. C. Gilbert, *Signal recovery from random measurements via orthogonal matching pursuit*, IEEE Trans. Inform. Theory **53** (2007), no. 12, 4655–4666, DOI 10.1109/TIT.2007.909108. MR2446929 (2009h:94042)

[20] D. L. Donoho and Y. Tsaig, *Fast solution of l_1-norm minimization problems when the solution may be sparse*, IEEE Trans. Inform. Theory **54** (2008), no. 11, 4789–4812, DOI 10.1109/TIT.2008.929958. MR2589865 (2011a:94027)

[21] L. T. Watson, *Theory of globally convergent probability-one homotopies for nonlinear programming*, SIAM J. Optim. **11** (2000/01), no. 3, 761–780 (electronic), DOI 10.1137/S105262349936121X. MR1814041 (2001m:90106)

[22] L. T. Watson and R. T. Haftka, *Modern homotopy methods in optimization*, Comput. Methods Appl. Mech. Engrg. **74** (1989), no. 3, 289–305, DOI 10.1016/0045-7825(89)90053-4. MR1020627 (90j:90130)

[23] Y. Yang, M. Möller, and S. Osher, *A dual split Bregman method for fast ℓ^1 minimization*, Math. Comp. **82** (2013), no. 284, 2061–2085, DOI 10.1090/S0025-5718-2013-02700-7. MR3073192

[24] W. Yin, *Analysis and generalizations of the linearized Bregman model*, SIAM J. Imaging Sci. **3** (2010), no. 4, 856–877, DOI 10.1137/090760350. MR2735964 (2011j:68172)

[25] M. R. Osborne, B. Presnell, and B. A. Turlach, *A new approach to variable selection in least squares problems*, IMA J. Numer. Anal. **20** (2000), no. 3, 389–403, DOI 10.1093/imanum/20.3.389. MR1773265 (2001g:65054)

Mathematics Department, University of California at Los Angeles, Los Angeles, California 90095-1555

E-mail address: fbarekat@math.ucla.edu

Mathematics Department, University of California at Los Angeles, Los Angeles, California 90095-1555

E-mail address: sjo@math.ucla.edu