L1-NORM PENALIZED LEAST SQUARES WITH SALSA

IVAN SELESNICK

ABSTRACT. This lecture note describes an iterative optimization algorithm, 'SALSA', for solving L1-norm penalized least squares problems. We describe the use of SALSA for sparse signal representation and approximation, especially with overcomplete Parseval transforms. We also illustrate the use of SALSA to perform basis pursuit (BP), basis pursuit denoising (BPD), and morphological component analysis (MCA). The algorithm, 'SALSA', was developed by Afonso, Bioucas-Dias, and Figueiredo.

1. INTRODUCTION

Numerous sparsity-based signal processing methods are based on ℓ_1-norm penalized least squares. This approach has been used for denoising, deconvolution, missing data estimation, signal separation, and other problems. It has been demonstrated that combining the *augmented Lagrangian* approach and the variable splitting technique is an effective algorithmic approach for solving linear inverse problems with sparse regularization [1]. An algorithm, called SALSA, developed in Ref. [1], is based on this approach. This algorithm is notable due to (*i*) its flexibility in handling various problems, and (*ii*) its fast convergence in practice. More generally, the *alternating direction method of multipliers* (ADMM) has been shown lately to be highly effective for large scale non-smooth optimization [3].

This note is intended to complement the tutorial [7] which intentionally omitted detailed descriptions of algorithms for solving the ℓ_1-norm optimization problems described therein. In particular, this note describes the derivation of SALSA to solve two problems. The first problem is ℓ_1-norm penalized least squares; i.e.,

$$\mathbf{x}^{\mathsf{opt}} = \arg\min_{\mathbf{x}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1. \tag{BPD}$$

The second problem is that of finding the solution to a system of linear equations with minimal ℓ_1-norm; i.e.,

$$\begin{aligned}\mathbf{x}^{\mathsf{opt}} = &\arg\min_{\mathbf{x}} \ \|\mathbf{x}\|_1 \\ &\text{such that } \mathbf{A}\mathbf{x} = \mathbf{y}.\end{aligned} \tag{BP}$$

These problems are sometimes referred to as basis pursuit denoising (BPD) and basis pursuit (BP), respectively [4].

For a vector $\mathbf{x} \in \mathbb{C}^N$, the ℓ_1 and ℓ_2 norms are defined by

$$\|\mathbf{x}\|_1 := \sum_{n=0}^{N-1} |x_n|, \qquad \|\mathbf{x}\|_2^2 := \sum_{n=0}^{N-1} |x_n|^2. \tag{1}$$

Sections 2 and 3 derive iterative algorithms to solve BPD and BP, respectively. Based on these algorithms, Section 4 derives iterative algorithms for 'dual BPD' and 'dual BP'. These algorithms can be used to implement *morphological component analysis* (MCA) for nonlinear signal decomposition. Section 5 describes

Date: November, 2011. Last edit: January 20, 2014.

Matlab software available at `http://eeweb.poly.edu/iselesni/lecture_notes/SALSA`.

This tutorial is a Connexions module (m48933 at `http://cnx.org/`).

Cite as: I. Selesnick. L1-Norm Penalized Least Squares with SALSA. Connexions, 2014. http://cnx.org/content/m48933/.

Support from NSF under Grant CCF-1018020 is gratefully acknowledged.

transforms, $\mathbf{A}$, useful for sparse signal representation and approximation. Section 6 presents examples of BP, BPD, dual BP, and dual BPD applied to simple signals.

1.1. **The Augmented Lagrangian.** For the constrained optimization problem,

$$\begin{aligned} &\arg\min_{\mathbf{z}} \quad E(\mathbf{z}) \\ &\text{such that} \quad \mathbf{C}\mathbf{z} - \mathbf{b} = \mathbf{0}, \end{aligned} \tag{2}$$

/Users/chenhao/Dropbox/Math/MathWiki/lagrangian

the augmented Lagrangian is defined as

$$L_A(\mathbf{z}, \boldsymbol{\alpha}, \mu) = E(\mathbf{z}) + \boldsymbol{\alpha}^{\mathsf{T}}(\mathbf{C}\mathbf{z} - \mathbf{b}) + \mu\|\mathbf{C}\mathbf{z} - \mathbf{b}\|_2^2. \tag{3}$$

The vector, $\boldsymbol{\alpha}$, are Lagrange multipliers. Version-2 of the *augmented Lagrangian method* (ALM) [1], to solve the constrained problem is given by

initialize: $\mu > 0$, $\mathbf{d}$

repeat

$$\mathbf{z} \leftarrow \arg\min_{\mathbf{z}} \; E(\mathbf{z}) + \mu\|\mathbf{C}\mathbf{z} - \mathbf{d}\|_2^2 \tag{4a}$$

$$\mathbf{d} \leftarrow \mathbf{d} - (\mathbf{C}\mathbf{z} - \mathbf{b}) \tag{4b}$$

end

The indented assignment operations are iterated until convergence. This method is also known as the *method of multipliers* (MM); so this iterative algorithm is referred to as ALM/MM in [1].

The ALM/MM algorithm calls for a positive scalar, μ, which is like a step-size parameter. Its value can affect the convergence speed of the algorithm. But it does not affect the solution to which it converges.

2. L1 norm regularized least squares (BPD)

Given an observed vector $\mathbf{y}$ and matrix $\mathbf{A}$, consider the problem of finding a sparse vector $\mathbf{x}$ such that $\mathbf{y} \approx \mathbf{A}\mathbf{x}$. Using the ℓ_1 norm as a measure of sparsity, the problem can be formulated as

$$\mathbf{x}^{\mathsf{opt}} = \arg\min_{\mathbf{x}} \quad \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \|\boldsymbol{\lambda} \odot \mathbf{x}\|_1 \tag{5}$$

The notation $\boldsymbol{\lambda} \odot \mathbf{x}$ denotes element-wise multiplication of the equal-size vectors $\boldsymbol{\lambda}$ and $\mathbf{x}$; i.e., $[\boldsymbol{\lambda} \odot \mathbf{x}]_i = \lambda_i x_i$. When all elements of vector $\boldsymbol{\lambda}$ are the same value (i.e., $\boldsymbol{\lambda}_i = \lambda \in \mathbb{R}_+$), then (5) can be written as

$$\arg\min_{\mathbf{x}} \quad \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1 \tag{6}$$

which is the more common form. However, it will sometimes be useful to allow non-uniform regularization of $\mathbf{x}$, so we will use the form (5).

Applying variable splitting to (5) yields

$$\begin{aligned} &\arg\min_{\mathbf{x},\mathbf{u}} \quad \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \|\boldsymbol{\lambda} \odot \mathbf{u}\|_1 \\ &\text{such that} \quad \mathbf{u} - \mathbf{x} = \mathbf{0} \end{aligned} \tag{7}$$

Variable splitting introduces an auxiliary variable, but it also decouples the terms of the objective function. (Actually, it moves the coupling into the constraint, which is handled subsequently through alternating minimization.)

Problem (7) can be put written in the form of (2) by setting

$$\mathbf{z}_1 = \mathbf{x}, \;\; \mathbf{z}_2 = \mathbf{u}, \;\; \mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix}, \;\; \mathbf{C} = \begin{bmatrix} \mathbf{I}, & -\mathbf{I} \end{bmatrix}, \;\; \mathbf{b} = \mathbf{0}, \tag{8}$$

and

$$E(\mathbf{z}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{z}_1\|_2^2 + \|\boldsymbol{\lambda} \odot \mathbf{z}_2\|_1. \tag{9}$$

Now that the problem is expressed in the form of (2), the ALM/MM algorithm (4) can be applied. Using ALM/MM to solve problem (7), we obtain the iterative algorithm:

initialize: $\mu > 0,\ \mathbf{d}$

repeat

$$\mathbf{x}, \mathbf{u} \leftarrow \arg\min_{\mathbf{x},\mathbf{u}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \|\boldsymbol{\lambda} \odot \mathbf{u}\|_1 + \frac{\mu}{2}\|\mathbf{u} - \mathbf{x} - \mathbf{d}\|_2^2 \tag{10a}$$

$$\mathbf{d} \leftarrow \mathbf{d} - (\mathbf{u} - \mathbf{x}) \tag{10b}$$

end

The vector $\mathbf{d}$ must be initialized prior to the iteration. We usually initialize $\mathbf{d}$ to the zero vector.

As proven by Eckstein and Bertsekas, in a more general setting, if the minimization in (10a) is performed alternately between $\mathbf{x}$ and $\mathbf{u}$, the algorithm will still converge to the global minimum [5]. This technique is known as *alternating direction method of multipliers* (ADMM). Alternating between minimization with respect to each of $\mathbf{x}$ and $\mathbf{u}$, we obtain the algorithm:

initialize: $\mu > 0,\ \mathbf{d}$

repeat

$$\mathbf{u} \leftarrow \arg\min_{\mathbf{u}} \ \|\boldsymbol{\lambda} \odot \mathbf{u}\|_1 + \frac{\mu}{2}\|\mathbf{u} - \mathbf{x} - \mathbf{d}\|_2^2 \tag{11a}$$

$$\mathbf{x} \leftarrow \arg\min_{\mathbf{x}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \frac{\mu}{2}\|\mathbf{u} - \mathbf{x} - \mathbf{d}\|_2^2 \tag{11b}$$

$$\mathbf{d} \leftarrow \mathbf{d} - (\mathbf{u} - \mathbf{x}) \tag{11c}$$

end

This algorithm is called SALSA (split augmented Lagrangian shrinkage algorithm) in Ref. [1]. In fact, SALSA is more general, as it allows a general regularizer, $\phi(\mathbf{x})$, not just the ℓ_1 norm.

The minimizations (11a) and (11b) can be performed in explicit form. The minimization problem in (11a) is separable in u_i. Its solution is expressed explicitly in terms of the soft-thresholding rule (see Appendix A). The minimization problem in (11b) is a constrained least squares problem; hence, its solution is available in explicit form (in terms of a matrix inverse). Utilizing the explicit forms for the two minimization problems, we obtain the following algorithm.

initialize: $\mu > 0,\ \mathbf{d}$

repeat

$$\mathbf{u} \leftarrow \text{soft}(\mathbf{x} + \mathbf{d},\ \boldsymbol{\lambda}/\mu) \tag{12a}$$

$$\mathbf{x} \leftarrow \left(\mathbf{A}^{\mathsf{H}}\mathbf{A} + \mu\,\mathbf{I}\right)^{-1}\left(\mathbf{A}^{\mathsf{H}}\mathbf{y} + \mu\,(\mathbf{u} - \mathbf{d})\right) \tag{12b}$$

$$\mathbf{d} \leftarrow \mathbf{d} - \mathbf{u} + \mathbf{x} \tag{12c}$$

end

The operator $\mathbf{A}^{\mathsf{H}}$ is the complex conjugate (Hermitian) transpose of $\mathbf{A}$.

With a change of variables, $\mathbf{v} = \mathbf{u} - \mathbf{d}$, the arithmetic operations can be slightly reduced.

Algorithm 1: Algorithm for basis pursuit denoising (5). Using eq.12

initialize: $\mu > 0,\ \mathbf{d}$

repeat

$$\mathbf{v} \leftarrow \text{soft}(\mathbf{x} + \mathbf{d},\ \boldsymbol{\lambda}/\mu) - \mathbf{d} \tag{13a}$$

$$\mathbf{x} \leftarrow \left(\mathbf{A}^{\mathsf{H}}\mathbf{A} + \mu\,\mathbf{I}\right)^{-1}\left(\mathbf{A}^{\mathsf{H}}\mathbf{y} + \mu\,\mathbf{v}\right) \tag{13b}$$

$$\mathbf{d} \leftarrow \mathbf{x} - \mathbf{v} \tag{13c}$$

end

Sometimes, in (13b) it can be useful to use the matrix inverse lemma (see Appendix B) to write

$$\left(\mu\,\mathbf{I} + \mathbf{A}^{\mathsf{H}}\mathbf{A}\right)^{-1} = \frac{1}{\mu}\mathbf{I} - \frac{1}{\mu}\mathbf{A}^{\mathsf{H}}\left(\mu\,\mathbf{I} + \mathbf{A}\mathbf{A}^{\mathsf{H}}\right)^{-1}\mathbf{A}, \tag{14}$$

because in certain cases, $\mu\,\mathbf{I} + \mathbf{A}\mathbf{A}^{\mathsf{H}}$ is easier to invert than $\mu\,\mathbf{I} + \mathbf{A}^{\mathsf{H}}\mathbf{A}$.

2.1. **When A is a Tight Frame.** In some signal processing applications, $\mathbf{A}$ is a 'wide' matrix satisfying

$$\mathbf{A}\,\mathbf{A}^{\mathsf{H}} = p\,\mathbf{I}, \quad p > 0. \tag{15}$$

e.g In random demodulator, the sensing matrix is composed of {-1,+1} rand sequence on its diagonal elements

In this case, it is sometimes said that the columns of $\mathbf{A}$ form a tight frame. The matrix $\mathbf{A}^{\mathsf{H}}$ can also be considered an overcomplete Parseval transform. For example, the columns of $\mathbf{A}$ may be an overcomplete set of a complex sinusoids with closely spaced frequencies.

In many cases, we will have $p = 1$ in (15). However, for some problems (e.g., dual BPD in Sect. 4) we will have $p \neq 1$.

Using (15) in (14), we obtain:

$$\left(\mu\,\mathbf{I} + \mathbf{A}^{\mathsf{H}}\mathbf{A}\right)^{-1} = \frac{1}{\mu}\mathbf{I} - \frac{1}{\mu(\mu+p)}\mathbf{A}^{\mathsf{H}}\mathbf{A}. \tag{16}$$

Then the update equation for $\mathbf{x}$ in (13b) becomes:

$$\mathbf{x} \leftarrow \frac{1}{\mu}\left(\mathbf{A}^{\mathsf{H}}\mathbf{y} + \mu\,\mathbf{v}\right) - \frac{1}{\mu(\mu+p)}\mathbf{A}^{\mathsf{H}}\,\mathbf{A}\left(\mathbf{A}^{\mathsf{H}}\mathbf{y} + \mu\,\mathbf{v}\right) \tag{17}$$

which simplifies to

$$\mathbf{x} \leftarrow \frac{1}{\mu}\mathbf{A}^{\mathsf{H}}\mathbf{y} + \mathbf{v} - \frac{p}{\mu(\mu+p)}\mathbf{A}^{\mathsf{H}}\mathbf{y} - \frac{1}{\mu+p}\mathbf{A}^{\mathsf{H}}\,\mathbf{A}\,\mathbf{v} \tag{18a}$$

$$= \frac{1}{\mu+p}\mathbf{A}^{\mathsf{H}}\mathbf{y} + \mathbf{v} - \frac{1}{\mu+p}\mathbf{A}^{\mathsf{H}}\mathbf{A}\,\mathbf{v} \tag{18b}$$

$$= \mathbf{v} + \frac{1}{\mu+p}\mathbf{A}^{\mathsf{H}}\left(\mathbf{y} - \mathbf{A}\,\mathbf{v}\right) \tag{18c}$$

Therefore, Algorithm 1 can be written as follows.

initialize: $\mu > 0, \mathbf{d}$

repeat

$$\mathbf{v} \leftarrow \text{soft}(\mathbf{x} + \mathbf{d},\ \boldsymbol{\lambda}/\mu) - \mathbf{d} \tag{19a}$$

$$\mathbf{x} \leftarrow \mathbf{v} + \frac{1}{\mu+p}\mathbf{A}^{\mathsf{H}}\left(\mathbf{y} - \mathbf{A}\,\mathbf{v}\right) \tag{19b}$$

$$\mathbf{d} \leftarrow \mathbf{x} - \mathbf{v} \tag{19c}$$

end

The algorithm can be simplified by a slight rearrangement of operations, as follows.

Algorithm 2: Algorithm for basis pursuit denoising (5) with $\mathbf{A}\mathbf{A}^{\mathsf{H}} = p\,\mathbf{I}$.

Shown in the code 'BPD.m', where p = 1

initialize: $\mu > 0,\ \mathbf{d}$

repeat

$$\mathbf{v} \leftarrow \mathrm{soft}(\mathbf{x} + \mathbf{d},\ \boldsymbol{\lambda}/\mu) - \mathbf{d} \tag{20a}$$

$$\mathbf{d} \leftarrow \frac{1}{\mu + p}\mathbf{A}^{\mathsf{H}}\left(\mathbf{y} - \mathbf{A}\,\mathbf{v}\right) \tag{20b}$$

$$\mathbf{x} \leftarrow \mathbf{d} + \mathbf{v} \tag{20c}$$

end

Note that this algorithm does not involve any matrix inverse. If fast implementations are available for $\mathbf{A}$ and $\mathbf{A}^{\mathsf{H}}$, then each iteration of the algorithm is fast.

3. L1 Norm Regularized Solutions to Linear Systems (BP)

Given an observed signal $\mathbf{y}$, consider the problem of finding a sparse vector $\mathbf{x}$ that solves $\mathbf{A}\mathbf{x} = \mathbf{y}$. Using the ℓ_1 norm as a measure of sparsity, the problem can be formulated as:

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \|\boldsymbol{\lambda} \odot \mathbf{x}\|_1 \\ \text{such that} \quad & \mathbf{A}\,\mathbf{x} = \mathbf{y} \end{aligned} \tag{21}$$

This problem is known as *basis pursuit* [4]. By applying the variable splitting technique, we obtain an equivalent optimization problem:

$$\begin{aligned} \arg\min_{\mathbf{x},\mathbf{u}} \quad & \|\boldsymbol{\lambda} \odot \mathbf{u}\|_1 \\ \text{such that} \quad & \mathbf{A}\,\mathbf{x} = \mathbf{y} \\ & \mathbf{u} - \mathbf{x} = \mathbf{0} \end{aligned} \tag{22}$$

We will use the 'partly' augmented Lagrangian:

$$L_A(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \mu) = \|\boldsymbol{\lambda} \odot \mathbf{u}\|_1 + \boldsymbol{\lambda}^{\mathsf{T}}(\mathbf{u} - \mathbf{x}) + 0.5\mu\|\mathbf{u} - \mathbf{x}\|_2^2 + \boldsymbol{\lambda}_2(\mathbf{A}\,\mathbf{x} - \mathbf{y}) \tag{23}$$

Using ALM/MM so solve the problem, we obtain the algorithm:

initialize: $\mu > 0,\ \mathbf{d}$

repeat

$$\mathbf{x}, \mathbf{u} \leftarrow \begin{cases} \arg\min_{\mathbf{x},\mathbf{u}} \|\boldsymbol{\lambda} \odot \mathbf{u}\|_1 + 0.5\mu\|\mathbf{u} - \mathbf{x} - \mathbf{d}\|_2^2 \\ \text{such that} \quad \mathbf{A}\,\mathbf{x} = \mathbf{y} \end{cases} \tag{24a}$$

$$\mathbf{d} \leftarrow \mathbf{d} - (\mathbf{u} - \mathbf{x}) \tag{24b}$$

end

By alternately minimizing with respect to $\mathbf{x}$ and $\mathbf{u}$ (as in Sec. 2), we obtain the algorithm:

initialize: $\mu > 0,\ \mathbf{d}$

repeat

$$\mathbf{u} \leftarrow \arg\min_{\mathbf{u}} \|\boldsymbol{\lambda} \odot \mathbf{u}\|_1 + 0.5\mu\|\mathbf{u} - \mathbf{x} - \mathbf{d}\|_2^2 \tag{25a}$$

$$\mathbf{x} \leftarrow \begin{cases} \arg\min_{\mathbf{x}} \|\mathbf{u} - \mathbf{x} - \mathbf{d}\|_2^2 \\ \text{such that} \quad \mathbf{A}\,\mathbf{x} = \mathbf{y} \end{cases} \tag{25b}$$

$$\mathbf{d} \leftarrow \mathbf{d} - (\mathbf{u} - \mathbf{x}) \tag{25c}$$

end

The minimization with respect to $\mathbf{u}$ in (25a) can be expressed explicitly in terms of soft-thresholding. The minimization with respect to $\mathbf{x}$ in (25b) is a constrained least squares problem which admits an explicit solution in terms of matrix inverses. Using the explicit solution to each of the two minimization problems, we obtain the algorithm:

initialize: $\mu > 0,\ \mathbf{d}$

repeat

$$\mathbf{u} \leftarrow \text{soft}(\mathbf{x} + \mathbf{d},\ \boldsymbol{\lambda}/\mu) \tag{26a}$$

$$\mathbf{x} \leftarrow (\mathbf{u} - \mathbf{d}) + \mathbf{A}^{\mathsf{H}}(\mathbf{A}\mathbf{A}^{\mathsf{H}})^{-1}(\mathbf{y} - \mathbf{A}(\mathbf{u} - \mathbf{d})) \tag{26b}$$

$$\mathbf{d} \leftarrow \mathbf{d} - (\mathbf{u} - \mathbf{x}) \tag{26c}$$

end

With a change of variables, $\mathbf{v} = \mathbf{u} - \mathbf{d}$, the arithmetic operations can be slightly reduced, as follows.

initialize: $\mu > 0,\ \mathbf{d}$

repeat

$$\mathbf{v} \leftarrow \text{soft}(\mathbf{x} + \mathbf{d},\ \boldsymbol{\lambda}/\mu) - \mathbf{d} \tag{27a}$$

$$\mathbf{x} \leftarrow \mathbf{v} + \mathbf{A}^{\mathsf{H}}(\mathbf{A}\mathbf{A}^{\mathsf{H}})^{-1}(\mathbf{y} - \mathbf{A}\mathbf{v}) \tag{27b}$$

$$\mathbf{d} \leftarrow \mathbf{x} - \mathbf{v} \tag{27c}$$

end

The algorithm can be further simplified by a slight rearrangement of operations, as follows.

Algorithm 3: Algorithm for basis pursuit (21).

initialize: $\mu > 0,\ \mathbf{d}$

repeat

$$\mathbf{v} \leftarrow \text{soft}(\mathbf{x} + \mathbf{d},\ \boldsymbol{\lambda}/\mu) - \mathbf{d} \tag{28a}$$

$$\mathbf{d} \leftarrow \mathbf{A}^{\mathsf{H}}(\mathbf{A}\mathbf{A}^{\mathsf{H}})^{-1}(\mathbf{y} - \mathbf{A}\mathbf{v}) \tag{28b}$$

$$\mathbf{x} \leftarrow \mathbf{d} + \mathbf{v} \tag{28c}$$

end

Note that at every iteration, $\mathbf{x}$ satisfies $\mathbf{A}\mathbf{x} = \mathbf{y}$. This is because

$$\mathbf{A}(\mathbf{d} + \mathbf{v}) = \mathbf{A}\left[\mathbf{A}^{\mathsf{H}}(\mathbf{A}\mathbf{A}^{\mathsf{H}})^{-1}(\mathbf{y} - \mathbf{A}\mathbf{v}) + \mathbf{v}\right] \tag{29a}$$

$$= \mathbf{A}\mathbf{A}^{\mathsf{H}}(\mathbf{A}\mathbf{A}^{\mathsf{H}})^{-1}(\mathbf{y} - \mathbf{A}\mathbf{v}) + \mathbf{A}\mathbf{v} \tag{29b}$$

$$= (\mathbf{y} - \mathbf{A}\mathbf{v}) + \mathbf{A}\mathbf{v} \tag{29c}$$

$$= \mathbf{y} \tag{29d}$$

3.1. **When A is a Tight Frame.** Consider the BP problem (21) when the columns of $\mathbf{A}$ form a tight frame; i.e., when $\mathbf{A}$ satisfies (15). Then Algorithm 3 can be written as follows.

Algorithm 4: Algorithm for basis pursuit (21) with $\mathbf{A}\mathbf{A}^{\mathsf{H}} = p\,\mathbf{I}$.

initialize: $\mu > 0,\ \mathbf{d}$

repeat

$$\mathbf{v} \leftarrow \text{soft}(\mathbf{x} + \mathbf{d},\ \boldsymbol{\lambda}/\mu) - \mathbf{d} \tag{30a}$$

$$\mathbf{d} \leftarrow \frac{1}{p}\mathbf{A}^{\mathsf{H}}(\mathbf{y} - \mathbf{A}\mathbf{v}) \tag{30b}$$

$$\mathbf{x} \leftarrow \mathbf{d} + \mathbf{v} \tag{30c}$$

end

Note that this is very similar to Algorithm 2; only the constant in (20b) is different. Likewise, if $\mathbf{A}$ and $\mathbf{A}^{\mathsf{H}}$ are fast, then then the algorithm as a whole is fast.

4. Dual BP and dual BPD

In several signal processing applications, it is useful to to model a signal $\mathbf{y}$ as

$$\mathbf{y} \approx \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2. \tag{31}$$

In particular, this model is used in morphological component analysis (MCA) for the nonlinear separation of signal components [11, 10]. There are several ways to formulate the MCA problem. Two approaches for MCA are based on forms of BP and BPD.

In the following, we assume that $\mathbf{A}_i$ are tight frames with frame constant $p = 1$; i.e.,

$$\mathbf{A}_1\mathbf{A}_1^{\mathsf{H}} = \mathbf{I}, \qquad \mathbf{A}_2\mathbf{A}_2^{\mathsf{H}} = \mathbf{I}. \tag{32}$$

4.1. **Dual BPD.** If the signal, $\mathbf{y}$, is noisy, then it is appropriate to allow a residual. In this case, MCA may be formulated as,

$$\arg\min_{\mathbf{x}_1,\mathbf{x}_2} \quad \frac{1}{2}\|\mathbf{y} - \mathbf{A}_1\mathbf{x}_1 - \mathbf{A}_2\mathbf{x}_2\|_2^2 + \|\boldsymbol{\lambda}_1 \odot \mathbf{x}_1\|_1 + \|\boldsymbol{\lambda}_2 \odot \mathbf{x}_2\|_1. \tag{33}$$

This is a special case of BPD (5) with

$$\mathbf{A} = \begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{bmatrix}, \quad \boldsymbol{\lambda} = \begin{bmatrix}\boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2\end{bmatrix}. \tag{34}$$

Since $\mathbf{A}_i$ are tight frames (32), we have:

$$\mathbf{A}\mathbf{A}^{\mathsf{H}} = \begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix}\begin{bmatrix}\mathbf{A}_1^{\mathsf{H}} \\ \mathbf{A}_2^{\mathsf{H}}\end{bmatrix} = \mathbf{A}_1\mathbf{A}_1^{\mathsf{H}} + \mathbf{A}_2\mathbf{A}_2^{\mathsf{H}} = 2\,\mathbf{I}. \tag{35}$$

Therefore, we can use Algorithm 2 in Sect. 2.1 with $p = 2$. Hence, we obtain the following algorithm for dual BPD.

initialize: $\mu > 0$, $\mathbf{d}$

repeat

$$\begin{bmatrix}\mathbf{v}_1 \\ \mathbf{v}_2\end{bmatrix} \leftarrow \text{soft}\left(\begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{bmatrix} + \begin{bmatrix}\mathbf{d}_1 \\ \mathbf{d}_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\lambda}_1/\mu \\ \boldsymbol{\lambda}_2/\mu\end{bmatrix}\right) - \begin{bmatrix}\mathbf{d}_1 \\ \mathbf{d}_2\end{bmatrix} \tag{36a}$$

$$\begin{bmatrix}\mathbf{d}_1 \\ \mathbf{d}_2\end{bmatrix} \leftarrow \frac{1}{\mu + 2}\begin{bmatrix}\mathbf{A}_1^{\mathsf{H}} \\ \mathbf{A}_2^{\mathsf{H}}\end{bmatrix}\left(\mathbf{y} - \begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix}\begin{bmatrix}\mathbf{v}_1 \\ \mathbf{v}_2\end{bmatrix}\right) \tag{36b}$$

$$\begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{bmatrix} \leftarrow \begin{bmatrix}\mathbf{d}_1 \\ \mathbf{d}_2\end{bmatrix} + \begin{bmatrix}\mathbf{v}_1 \\ \mathbf{v}_2\end{bmatrix} \tag{36c}$$

end

This algorithm can be expressed as follows.

Algorithm 5: Algorithm for dual BPD (33) with $\mathbf{A}_i \mathbf{A}_i^{\mathsf{H}} = \mathbf{I}$.

initialize: $\mu > 0,\ \mathbf{d}_i$

repeat

$$\mathbf{v}_i \leftarrow \operatorname{soft}(\mathbf{x}_i + \mathbf{d}_i,\ \boldsymbol{\lambda}_i/\mu) - \mathbf{d}_i, \quad i = 1, 2 \tag{37a}$$

$$\mathbf{c} \leftarrow \mathbf{y} - \mathbf{A}_1 \mathbf{v}_1 - \mathbf{A}_2 \mathbf{v}_2 \tag{37b}$$

$$\mathbf{d}_i \leftarrow \frac{1}{\mu + 2} \mathbf{A}_i^{\mathsf{H}}\, \mathbf{c}, \qquad i = 1, 2 \tag{37c}$$

$$\mathbf{x}_i \leftarrow \mathbf{d}_i + \mathbf{v}_i, \qquad i = 1, 2 \tag{37d}$$

end

4.2. **Dual Basis Pursuit.** If the signal, $\mathbf{y}$, is noise-free, then it is appropriate to use an equality constraint. In this case MCA may be formulated as,

$$\begin{aligned} &\arg \min_{\mathbf{x}_1, \mathbf{x}_2} \quad \|\boldsymbol{\lambda}_1 \odot \mathbf{x}_1\|_1 + \|\boldsymbol{\lambda}_2 \odot \mathbf{x}_2\|_1 \\ &\text{such that} \quad \mathbf{y} = \mathbf{A}_1 \mathbf{x}_1 + \mathbf{A}_2 \mathbf{x}_2. \end{aligned} \tag{38}$$

This is a special case of BP (21) with

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}, \quad \boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \end{bmatrix}. \tag{39}$$

Since we assume that $\mathbf{A}_i$ are tight frames (32), we have (35). Therefore, we can use Algorithm 4 with $p = 2$. Hence, we obtain the following algorithm for dual BP.

initialize: $\mu > 0,\ \mathbf{d}$

repeat

$$\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} \leftarrow \operatorname{soft}\left(\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\lambda}_1/\mu \\ \boldsymbol{\lambda}_2/\mu \end{bmatrix} \right) - \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix} \tag{40a}$$

$$\begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix} \leftarrow \frac{1}{2} \begin{bmatrix} \mathbf{A}_1^{\mathsf{H}} \\ \mathbf{A}_2^{\mathsf{H}} \end{bmatrix} \left(\mathbf{y} - \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} \right) \tag{40b}$$

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix} \tag{40c}$$

end

This algorithm can be expressed as follows.

Algorithm 6: Algorithm for dual BP (38) with $\mathbf{A}_i \mathbf{A}_i^{\mathsf{H}} = \mathbf{I}$.

initialize: $\mu > 0,\ \mathbf{d}_i$

repeat

$$\mathbf{v}_i \leftarrow \operatorname{soft}(\mathbf{x}_i + \mathbf{d}_i,\ \boldsymbol{\lambda}_i/\mu) - \mathbf{d}_i, \quad i = 1, 2 \tag{41a}$$

$$\mathbf{c} \leftarrow \mathbf{y} - \mathbf{A}_1\, \mathbf{v}_1 - \mathbf{A}_2\, \mathbf{v}_2 \tag{41b}$$

$$\mathbf{d}_i \leftarrow \frac{1}{2} \mathbf{A}_i^{\mathsf{H}}\, \mathbf{c}, \qquad i = 1, 2 \tag{41c}$$

$$\mathbf{x}_i \leftarrow \mathbf{d}_i + \mathbf{v}_i, \qquad i = 1, 2 \tag{41d}$$

end

Note that this is the same as Algorithm 5 except for a constant in (37c).

Note that Algorithms 5 and 6 involve no matrix inverses. If $\mathbf{A}_i$ and $\mathbf{A}_i^{\mathsf{H}}$ are fast, then these algorithms as a whole are fast. For example, if the $\mathbf{A}_i$ are FFTs and/or short-time Fourier transforms, then they have low implementation complexity and can be admit high parallelism. Such a combination of transforms is useful for decomposing a signal into narrow-band and wide-band signal components, even when the components overlap in both time and frequency [9]. Alternately, by taking the $\mathbf{A}_i$ as wavelet transforms with different Q-factors, a signal can be decomposed into low and high resonance components [8].

5. Transforms for Sparse Signal Representation

In order to apply BP and BPD, a transform, $\mathbf{A}$, is need for the sparse representation of the signal of interest. In dual BP and dual BPD, two transforms, $\mathbf{A}_1$ and $\mathbf{A}_2$, are needed. The transforms should be chosen such that they enables a sparse representation (or approximation) of the signals of interest.

To emphasize that the representation of a signal $\mathbf{y}$ is in terms of transform coefficients, we use the letter 'c' for coefficients. For example, we write $\mathbf{y} = \mathbf{A}\mathbf{c}$ as a representation of signal $\mathbf{y}$ with respect to transform $\mathbf{A}$ where $\mathbf{c}$ is the vector of transform coefficients.

5.1. Zero-padded DFT.

To sparsely represent a real or complex set of sinusoids, we take $\mathbf{A}$ to be the normalized inverse of an K-point DFT with $K \geq N$. Specifically, $\mathbf{A} : \mathbb{C}^K \to \mathbb{C}^N$ is defined by

$$[\mathbf{A}\mathbf{c}]_n = \sqrt{K}\,[\mathrm{DFT}_K^{-1}\{\mathbf{c}\}]_n, \quad \mathbf{c} \in \mathbb{C}^K,\ n \in \mathbb{Z}_N. \tag{42}$$

where DFT_K is the discrete Fourier transform (DFT) with zero-padding up to a total length of K. The multiplication by $\sqrt{K}$ normalizes $\mathbf{A}$ so that $\mathbf{A}\mathbf{A}^{\mathsf{H}} = \mathbf{I}_N$. Note that, when $K > N$, the matrix $\mathbf{A}$ is 'wide' rather than square. Accordingly, in the definition of $\mathbf{A}$, the inverse K-point DFT is truncated down to N samples. The matrix $\mathbf{A}$ is a sub-matrix of the the inverse K-point DFT matrix (the first N rows of the $K \times K$ inverse DFT matrix). Consequently, $\mathbf{A}^{\mathsf{H}} : \mathbb{C}^N \to \mathbb{C}^K$ is defined by

$$[\mathbf{A}^{\mathsf{H}}\mathbf{x}]_k = \frac{1}{\sqrt{K}}\,[\mathrm{DFT}_K\{\mathbf{x}\}]_k, \quad \mathbf{x} \in \mathbb{C}^N,\ k \in \mathbb{Z}_K \tag{43}$$

where the N-point vector $\mathbf{x}$ is zero-padded to length K prior to the DFT computation.

The ℓ_2 norm of all the columns of $\mathbf{A}$ are equal, specifically,

$$\|\mathbf{a}\|_2 = \sqrt{\frac{N}{K}}. \tag{44}$$

This value can be used for setting regularization parameters, λ, in BPD and dual BPD. When the DFT is critically sampled (i.e., $K = N$), then $\mathbf{A}$ is simply the conventional DFT, normalized so as to be unitary, in which case (44) gives unity, as expected.

The DFT operator $\mathbf{A}$ can be implemented in MATLAB as

```
truncate = @(c, N) c(1:N);
A = @(c) sqrt(K) * truncate(ifft(c), N);
```

and $\mathbf{A}^{\mathsf{H}}$ as

```
AH = @(x) fft(x, K)/sqrt(K);
```

These fast matrix-free implementations of $\mathbf{A}$ and $\mathbf{A}^{\mathsf{H}}$ can be used for dual BP and dual BPD only if the utilized optimization algorithms are also 'matrix-free', as are the SALSA algorithms.

5.2. **STFT.** To sparsely represent a signal composed of oscillatory pulses, we take $\mathbf{A}$ to be the normalized inverse of a short-time Fourier transform (STFT). The STFT has several parameters: the frame length, overlapping factor, and DFT length. We typically use 50% overlapping and a DFT length equal to at least the frame length. Consequently, the STFT is at least two-times over-sampled. If the time-frequency array of STFT coefficients is of size $M \times K$, for a signal of length N, then $\mathbf{A} : \mathbb{C}^{M\times K} \to \mathbb{C}^N$ is defined as

$$[\mathbf{A}\mathbf{c}]_n = [\mathrm{STFT}^{-1}\{\mathbf{c}\}]_n, \quad n \in \mathbb{Z}_N \tag{45}$$

and $\mathbf{A}^{\mathsf{H}} : \mathbb{C}^N \to \mathbb{C}^{M\times K}$ is defined as

$$[\mathbf{A}^{\mathsf{H}}\mathbf{x}]_{(m,k)} = [\mathrm{STFT}\{\mathbf{x}\}]_{(m,k)}, \quad m \in \mathbb{Z}_M,\ k \in \mathbb{Z}_K. \tag{46}$$

With a suitably implemented STFT, we have $\mathbf{A}\mathbf{A}^{\mathsf{H}} = \mathbf{I}_N$.

The ℓ_2 norm of all the columns of $\mathbf{A}$ are equal. Specifically, when 50% overlapping is used, the norm is given by

$$\|\mathbf{a}\|_2 = \sqrt{\frac{R}{2K}}. \tag{47}$$

This value can be used for setting regularization parameters, λ, in BPD and dual BPD.

We implement the STFT operators $\mathbf{A}$ and $\mathbf{A}^{\mathsf{H}}$ in MATLAB as

```
A = @(c) ipSTFT(c, R, N);
AH = @(x) pSTFT(x, R, K);
```

where `pSTFT` and `ipSTFT` are our implementations of the STFT, and its inverse, designed to satisfy $\mathbf{A}\mathbf{A}^{\mathsf{H}} = \mathbf{I}_N$. The 'p' stands for 'Parseval'. The parameter R is the frame length. The parameter K is the DFT length, with $K \geq R$. (The function `pSTFT` is not the built-in MATLAB spectrogram function, which is not designed to ensure invertibility.) The implementation of the STFT, so as to satisfy $\mathbf{A}\mathbf{A}^{\mathsf{H}} = \mathbf{I}$, is described in [6].

6. Examples

Example 1 (BP)**.** This example illustrates the sparse representation of complex sinusoids in white complex noise using BP with a zero-padded DFT. We assume that the signal $\mathbf{y}$ admits a sparse representation of the form

$$\mathbf{y} = \mathbf{A}\mathbf{c}, \quad \mathbf{y} \in \mathbb{C}^N,\ \mathbf{c} \in \mathbb{C}^K,\ \mathbf{A} \in \mathbb{C}^{N\times K} \tag{48}$$

where $\mathbf{A}$ is a zero-padded DFT and $\mathbf{c}$ is a sparse set of DFT coefficients. Given $\mathbf{y}$, we find a sparse $\mathbf{c}$ by solving the basis pursuit problem,

$$\begin{aligned} \mathbf{c}^{\mathsf{opt}} &= \arg\min_{\mathbf{c}} \|\mathbf{c}\|_1 \\ &\quad \text{such that } \mathbf{A}\mathbf{c} = \mathbf{y}. \end{aligned} \tag{49}$$

Figure 1 shows the (real part of the) complex signal, $\mathbf{y}$, in (a); the DFT coefficients, $\mathbf{A}^{\mathsf{H}}\mathbf{y}$ in (b); and the BPD-optimized coefficients, $\mathbf{c}^{\mathsf{opt}}$, in (c). In this example, the signal $\mathbf{y}$ is of length $N = 100$ samples. In Fig. 1 (and subsequent figures), the sampling rate is taken to be one sample/sec. in the axis labeling.

Example 2 (BPD)**.** This example illustrates the estimation of real-valued sinusoids in white noise using basis pursuit denoising (BPD). We assume the noisy data, $\mathbf{y}$, is given by

$$\mathbf{y} = \mathbf{A}\mathbf{c} + \mathbf{w}, \quad \mathbf{y}, \mathbf{w} \in \mathbb{C}^N,\ \mathbf{c} \in \mathbb{C}^K,\ \mathbf{A} \in \mathbb{C}^{N\times K} \tag{50}$$

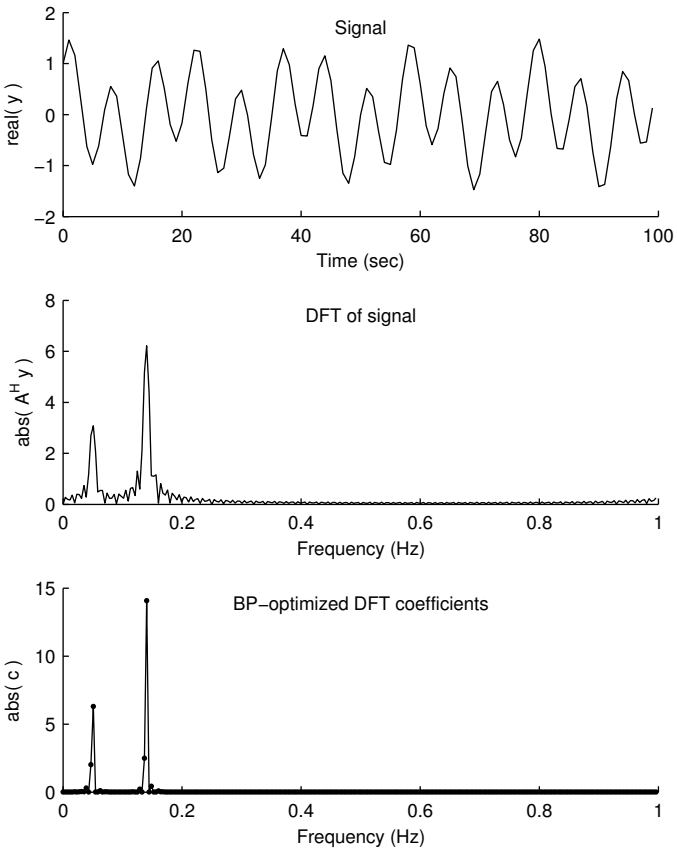

FIGURE 1. Example 1. Sparse representation of complex sinusoids in white complex noise using BP with a zero-padded DFT. (a) Complex sinusoids, $\mathbf{y}$. (b) DFT of data, $\mathbf{A}^{\mathsf{H}}\mathbf{y}$. (c) Sparse DFT coefficients, $\mathbf{c}^{\mathsf{opt}}$, obtained by solving the BP problem.

where $\mathbf{c}$ is a sparse set of DFT coefficients and $\mathbf{w}$ is a white Gaussian vector. This example uses real-valued sinusoids and noise, but we may still assume complex-valued signals to solve the problem. Due to conjugate symmetry properties of the DFT, the denoised signal will be real-valued.

Given $\mathbf{y}$, we estimate the sinusoids by solving the BPD problem,

$$\mathbf{c}^{\mathsf{opt}} = \arg\min_{\mathbf{c}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{c}\|_2^2 + \lambda\|\mathbf{c}\|_1. \tag{51}$$

For the transform, $\mathbf{A}$, we use the zero-padded DFT (100 signal samples in the time domain, 256 DFT coefficients in the frequency domain). The optimal coefficients, $\mathbf{c}^{\mathsf{opt}}$, are obtained using Algorithm 2. Figure 2 shows the noisy data, $\mathbf{y}$, in (a); the noisy DFT coefficients, $\mathbf{A}^{\mathsf{H}}\mathbf{y}$ in (b); the BPD-optimized coefficients, $\mathbf{c}^{\mathsf{opt}}$, in (c); and the denoised signal, $\mathbf{A}\mathbf{c}^{\mathsf{opt}}$, in (d). In the figure, only the positive frequency axis ($0 \leq f \leq 0.5$) is shown because the DFT coefficients are conjugate symmetric.

Example 3 (BPD)**.** This example illustrates the estimation of a real-valued pulse in white noise using basis pursuit denoising (BPD). We assume the noisy data, $\mathbf{y}$, is given by

$$\mathbf{y} = \mathbf{A}\mathbf{c} + \mathbf{w}, \quad \mathbf{y}, \mathbf{w} \in \mathbb{C}^N, \tag{52}$$

where $\mathbf{c}$ is a sparse set of STFT coefficients and $\mathbf{w}$ is a white Gaussian vector.

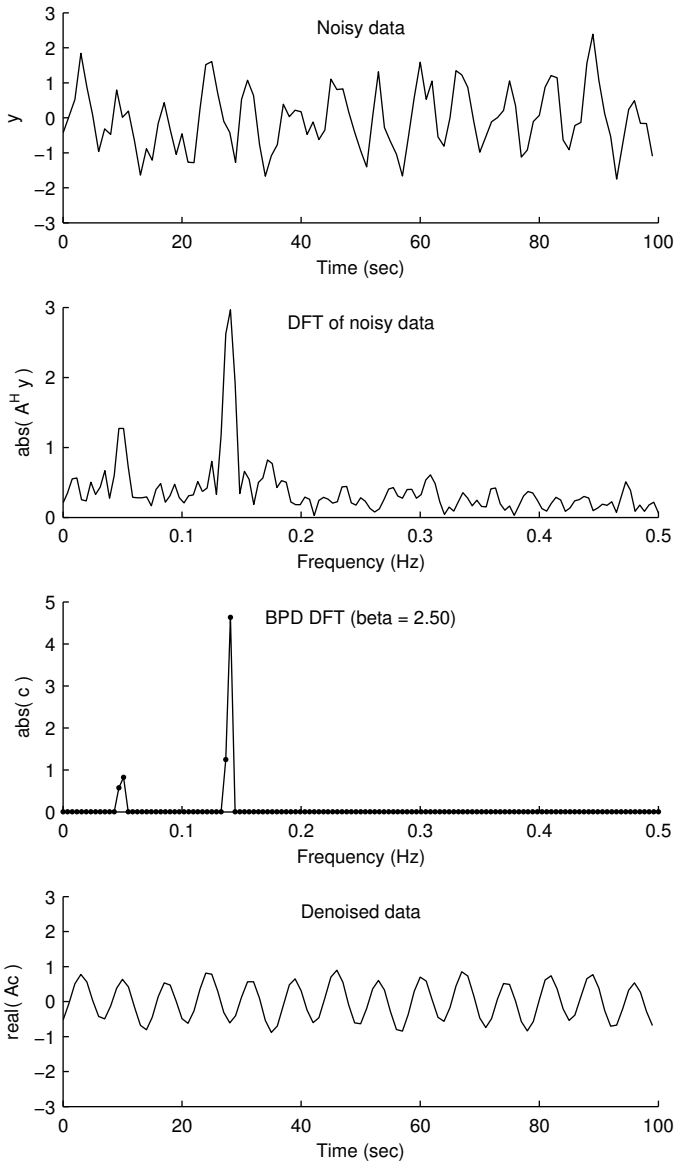

FIGURE 2. Example 2. Estimation of sinusoids in white noise using BPD with a zero-padded DFT. (a) Sinusoids in white noise, $\mathbf{y}$. (b) DFT of noisy data, $\mathbf{A}^{\mathsf{H}}\mathbf{y}$. (c) Sparse DFT coefficients, $\mathbf{c}^{\mathsf{opt}}$, obtained by solving the BPD problem. (d) Denoised signal, $\mathbf{A}\mathbf{c}^{\mathsf{opt}}$.

Given $\mathbf{y}$, we estimate the pulse by solving the dual BP problem,

$$\mathbf{c}^{\mathsf{opt}} = \arg\min_{\mathbf{c}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{c}\|_2^2 + \lambda\|\mathbf{c}\|_1. \tag{53}$$

For the transform, $\mathbf{A}$, we use the STFT. The optimal coefficients, $\mathbf{c}^{\mathsf{opt}}$, are obtained using Algorithm 2. Figure 3 shows the noisy data, $\mathbf{y}$, in (a); the noisy STFT coefficients, $\mathbf{A}^{\mathsf{H}}\mathbf{y}$ in (b); the BPD-optimized coefficients, $\mathbf{c}^{\mathsf{opt}}$, in (c); and the denoised signal, $\hat{\mathbf{s}} = \mathbf{A}\mathbf{c}^{\mathsf{opt}}$, in (d).

Example 4 (dual-BP)**.** We illustrate the separation of a sinusoid and a pulse in the noise-free case. The data is given by

$$\mathbf{y} = \mathbf{s}_1 + \mathbf{s}_2, \quad \mathbf{y}, \mathbf{s}_1, \mathbf{s}_2 \in \mathbb{R}^N. \tag{54}$$

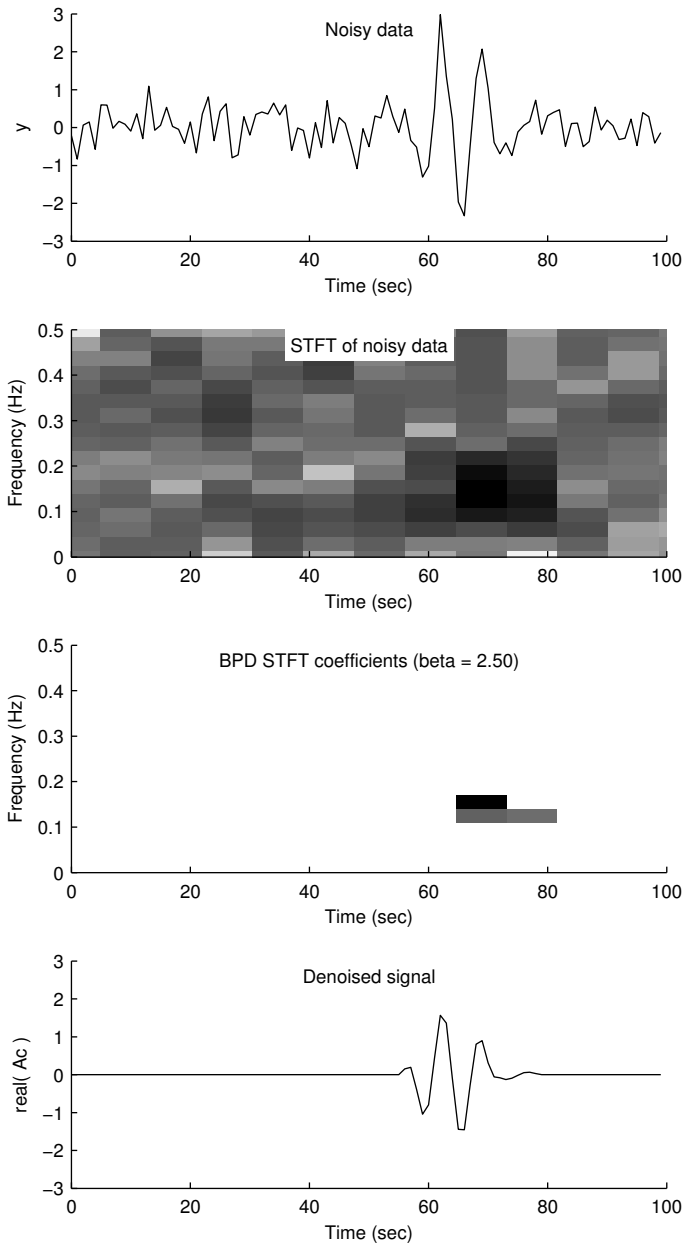

FIGURE 3. Example 3. Estimation of a pulse in white noise using BPD with a STFT. (a) Pulse in white noise, $\mathbf{y}$. (b) STFT of noisy data, $\mathbf{A}^{\mathsf{H}}\mathbf{y}$. (c) Sparse STFT coefficients, $\mathbf{c}^{\mathsf{opt}}$, obtained by solving the BPD problem. (d) Denoised signal, $\mathbf{A}\mathbf{c}^{\mathsf{opt}}$.

Given $\mathbf{y}$, we estimate $\mathbf{s}_1$ and $\mathbf{s}_2$ by solving the dual BP problem,

$$\begin{aligned} \{\mathbf{c}_1^{\mathsf{opt}}, \mathbf{c}_2^{\mathsf{opt}}\} = \arg\min_{\mathbf{c}_1,\mathbf{c}_2} \quad & \lambda_1\|\mathbf{c}_1\|_1 + \lambda_2\|\mathbf{c}_2\|_1 \\ \text{such that} \quad & \mathbf{y} = \mathbf{A}_1\mathbf{c}_1 + \mathbf{A}_2\mathbf{c}_2. \end{aligned} \tag{55}$$

The optimal coefficients, $\mathbf{c}_i$, are found using Algorithm 6. We then set $\hat{\mathbf{s}}_i = \mathbf{A}_i\mathbf{c}_i^{\mathsf{opt}}$ for $i = 1, 2$.

The example is illustrated in Fig. 4. For transform $\mathbf{A}_1$, we use a zero-padded DFT with two-times oversampling; i.e., $N = 100$ and $K = 200$ in (42) and (43). For transform $\mathbf{A}_2$, we use the STFT with frame

length $R = 16$, DFT length $K = 16$, and 50% overlapping. Hence, both $\mathbf{A}_1$ and $\mathbf{A}_2$ are oversampled by two, and $\|\mathbf{a}_1\|_2 = \|\mathbf{a}_1\|_2 = 1/\sqrt{2}$.

In the dual BP problem, we set $\lambda_1 = \lambda_2 = 0.5$. To solve the dual BP problem, we run Algorithm 6. As a result, we obtain sparse coefficients, $\mathbf{c}_i^{\mathsf{opt}}$, as illustrated in Fig. 4. The coefficient vector, $\mathbf{c}_1^{\mathsf{opt}}$, is a sparse set of DFT coefficients. The coefficient array, $\mathbf{c}_2^{\mathsf{opt}}$, is a sparse set of STFT coefficients, displayed as an image in a time-frequency plane. From the coefficients, we construct the two signals, $\mathbf{x}_1$ and $\mathbf{x}_2$ as $\mathbf{x}_i = \mathbf{A}_i\mathbf{c}_i$, for $i = 1, 2$.

Example 5 (dual-BPD)**.** We illustrate the separation and estimation of a sinusoid and a pulse in the case of additive white Gaussian noise. The data is given by

$$\mathbf{y} = \mathbf{s}_1 + \mathbf{s}_2 + \mathbf{w}, \quad \mathbf{y}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{w} \in \mathbb{R}^N \tag{56}$$

where $\mathbf{w}$ is a zero-mean white Gaussian vector with variance σ^2. Figure 5 shows the noisy signal, $\mathbf{y}$.

Given $\mathbf{y}$, we estimate $\mathbf{s}_1$ and $\mathbf{s}_2$ by solving the dual BPD problem,

$$\{\mathbf{c}_1^{\mathsf{opt}}, \mathbf{c}_2^{\mathsf{opt}}\} = \arg\min_{\mathbf{c}_1, \mathbf{c}_2} \ \frac{1}{2}\|\mathbf{y} - \mathbf{A}_1\mathbf{c}_1 - \mathbf{A}_2\mathbf{c}_2\|_2^2 + \lambda_1\|\mathbf{c}_1\|_1 + \lambda_2\|\mathbf{c}_2\|_1. \tag{57}$$

The components $\mathbf{s}_i$ are then estimated as,

$$\hat{\mathbf{s}}_i = \mathbf{A}_i\mathbf{c}_i^{\mathsf{opt}}, \quad i = 1, 2. \tag{58}$$

For transform $\mathbf{A}_1$, we use a zero-padded DFT; i.e., $N = 100$ and $K = 256$ in (42) and (43) (100 signal samples in the time domain, 256 DFT coefficients in the frequency domain). Hence, the dual BPD algorithm can be run using radix-2 FFTs exclusively. For this $\mathbf{A}_1$, we have $\|\mathbf{a}_1\|_2 = \sqrt{100/256} = 5/8$. For transform $\mathbf{A}_2$, we use the STFT with frame length $R = 16$, DFT length $K = 16$, and 50% overlapping. For this $\mathbf{A}_2$, we have $\|\mathbf{a}_2\|_2 = 1/\sqrt{2}$.

In the dual BPD problem, we set $\lambda_1 = \beta\|\mathbf{a}_1\|_2\sigma$, and $\lambda_2 = \beta\|\mathbf{a}_2\|_2\sigma$ where $\beta = 2.5$. This choice of λ_i is discussed in [other notes]. Generically, one may set $\beta \in [2.5, 3]$. To solve the dual BPD problem, we run Algorithm 5. As a result, we obtain sparse coefficients, $\mathbf{c}_1^{\mathsf{opt}}$ and $\mathbf{c}_2^{\mathsf{opt}}$, as illustrated in Fig. 5. ($\mathbf{c}_1$ is a sparse vector of DFT coefficients, $\mathbf{c}_2$ is a sparse two-dimensional array of STFT coefficients). From the coefficients, we obtain the two signals, $\mathbf{x}_1$ and $\mathbf{x}_2$ as $\mathbf{x}_i = \mathbf{A}_i\mathbf{c}_i$, for $i = 1, 2$.

Example 6 (dual-BP)**.** This example illustrates dual BPD with a speech waveform.

To be completed ....

7. Conclusion

This note has described the algorithm, SALSA, for standard ℓ_1 norm minimization problems arising in sparse signal processing. SALSA can also be used for more general problems (not only the quadratic data fidelity) and more general regularization terms (not only the ℓ_1 norm penalty); see Ref. [1] for more details. An extension of SALSA to the constrained formulation of the sparsity-penalized least squares problem, called CSALSA, is developed in Ref. [2].

In this note, we emphasize that when $\mathbf{A}_i$ are tight frames, then the presented algorithms for BP, BPD, dual BP, and dual BPD are:

(1) Matrix-free: The $\mathbf{A}_i$ and $\mathbf{A}_i^{\mathsf{H}}$ appear only at operators. No elements of $\mathbf{A}_i$ need to be individually accessed. Hence, $\mathbf{A}_i$ do not need to be stored as matrices. It is sufficient to implement the operators as algorithms. Fast algorithm for $\mathbf{A}_i$ and $\mathbf{A}_i$ can be exploited.
(2) Low complexity: The main computation is $\mathbf{A}_i$ and $\mathbf{A}_i^{\mathsf{H}}$.
(3) Globally convergent: Any initialization leads to an optimal solution (the objective functions are convex).

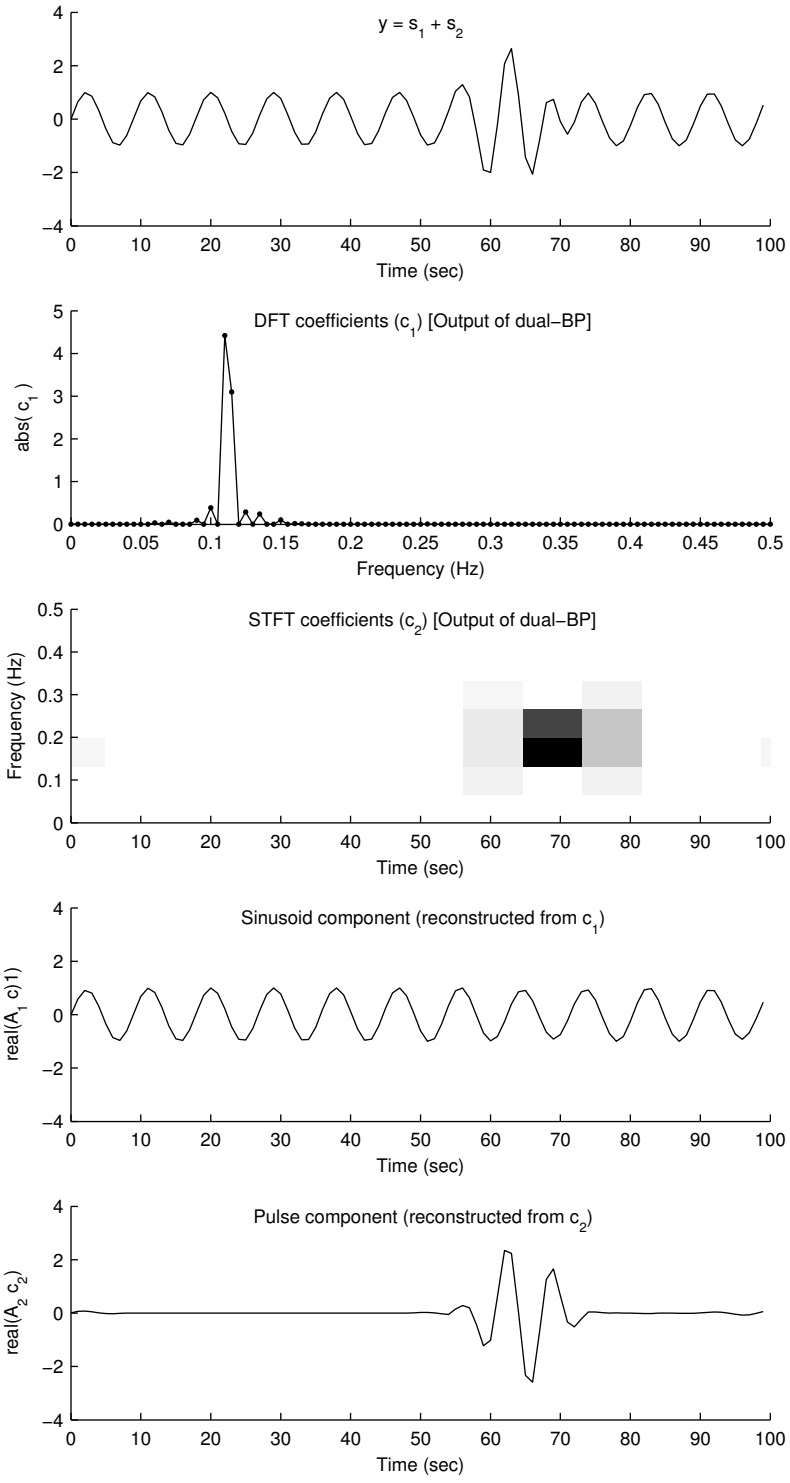

FIGURE 4. Example 4. Separation of a sinusoid and pulse using dual-BP with a zero-padded DFT and STFT. (a) Data, $\mathbf{x}$. (b) Sparse DFT coefficients, $\mathbf{c}_1^{\mathsf{opt}}$. (c) Sparse STFT coefficients, $\mathbf{c}_2^{\mathsf{opt}}$. (b) Estimated sinusoid, $\mathbf{A}_1\mathbf{c}_1^{\mathsf{opt}}$. (c) Estimated pulse, $\mathbf{A}_2\mathbf{c}_2^{\mathsf{opt}}$. The coefficients, $\mathbf{c}_1^{\mathsf{opt}}$ and $\mathbf{c}_2^{\mathsf{opt}}$, are obtained by solving the dual-BP problem.

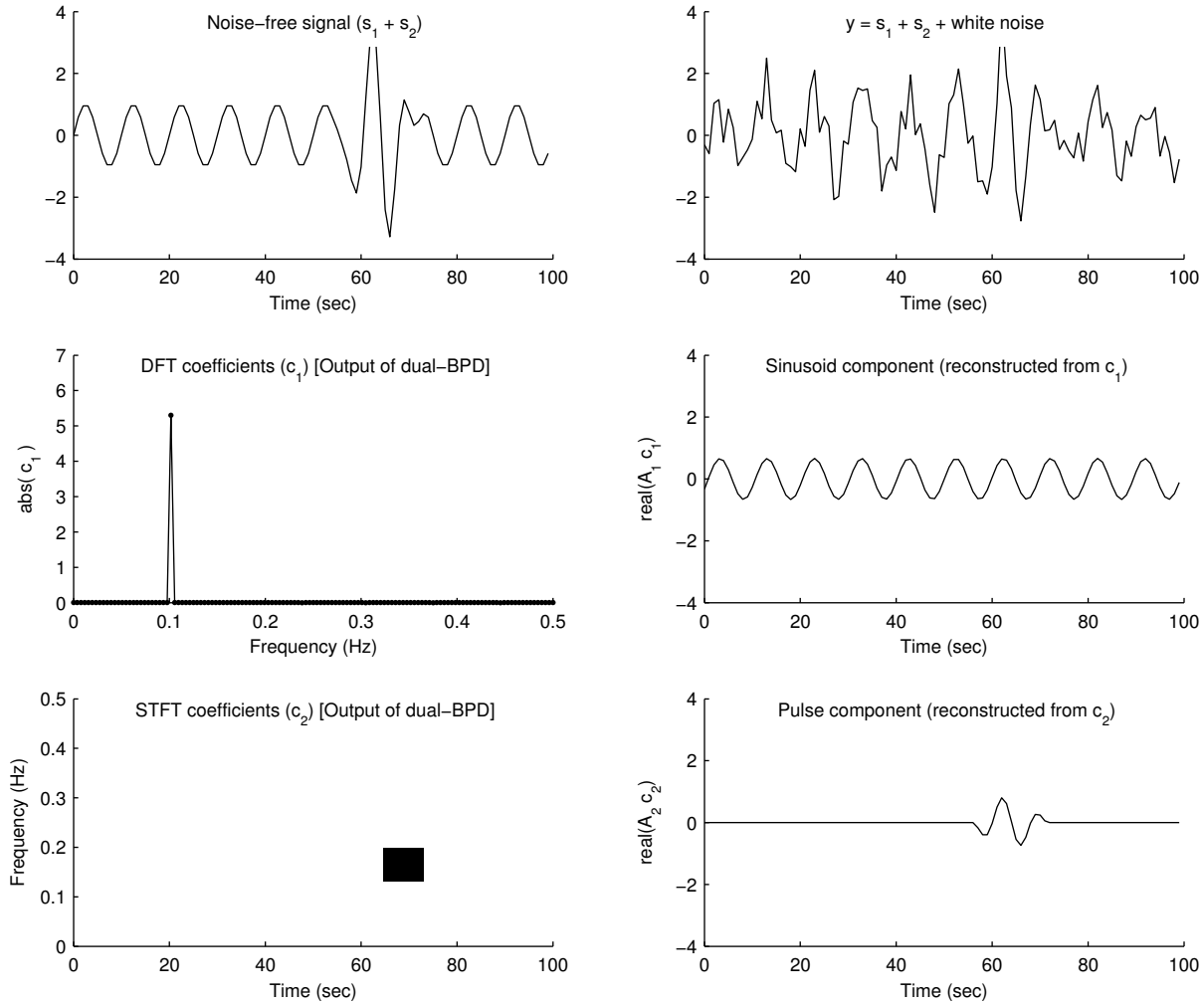

FIGURE 5. Example 5. Estimation of a sinusoid and pulse in white noise using dual-BPD with a zero-padded DFT and STFT. (a) Nois-free data. (b) Noisy data, $\mathbf{x}$. (c) Sparse DFT coefficients, $\mathbf{c}_1^{\mathsf{opt}}$. (d) Estimated sinusoid, $\mathbf{A}_1\mathbf{c}_1^{\mathsf{opt}}$. (e) Sparse STFT coefficients, $\mathbf{c}_2^{\mathsf{opt}}$. (f) Estimated pulse, $\mathbf{A}_2\mathbf{c}_2^{\mathsf{opt}}$. The coefficients, $\mathbf{c}_1^{\mathsf{opt}}$ and $\mathbf{c}_2^{\mathsf{opt}}$, are obtained by solving the dual-BPD problem.

If $\mathbf{A}_i$ and $\mathbf{A}_i^{\mathsf{H}}$ are fast, then these algorithms as a whole are fast. For example, if the $\mathbf{A}_i$ are FFTs and/or short-time Fourier transforms, then they have low implementation complexity and can admit high parallelism. Such a combination of transforms is useful for decomposing a signal into narrow-band and wide-band signal components, even when the components overlap in both time and frequency. An application of this method to radar signal processing is described in Ref. [9]. Alternately, by taking the $\mathbf{A}_i$ as wavelet transforms with different Q-factors, a signal can be decomposed into low and high resonance components [8].

APPENDIX A. SOFT THRESHOLD FUNCTION

The soft-thresholding function, $\text{soft} : \mathbb{C} \times \mathbb{R}_+ \to \mathbb{C}$, is defined as

$$\text{soft}(x, T) = \max\big(1 - T/|x|,\, 0\big) \cdot x. \tag{59}$$

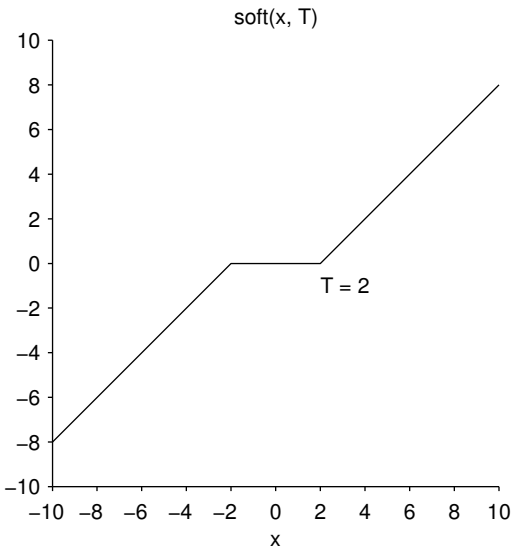

FIGURE 6. The soft threshold function on the real line.

The soft-threshold function on the real-line is illustrated in Fig. 6.

When we apply soft-thresholding to a vector, $\mathbf{x} \in \mathbb{C}^N$, we apply it component-wise; i.e.

$$[\text{soft}(\mathbf{x}, T)]_i = \text{soft}(x_i, T). \tag{60}$$

If both $\mathbf{x}$ and $\mathbf{T}$ are vectors of equal length, then

$$[\text{soft}(\mathbf{x}, \mathbf{T})]_i = \text{soft}(x_i, T_i). \tag{61}$$

APPENDIX B. MATRIX INVERSE LEMMA

The matrix inverse lemma is given by

$$(\mathbf{A} + \mathbf{BCd}D)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}\right)^{-1}\mathbf{DA}^{-1}. \tag{62}$$

From (62), we obtain

$$\left(\mu\,\mathbf{I} + \mathbf{A}^{\mathsf{H}}\mathbf{A}\right)^{-1} = \frac{1}{\mu}\mathbf{I} - \frac{1}{\mu}\mathbf{A}^{\mathsf{H}}\left(\mu\,\mathbf{I} + \mathbf{AA}^{\mathsf{H}}\right)^{-1}\mathbf{A}. \tag{63}$$

REFERENCES

[1] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. Fast image recovery using variable splitting and constrained optimization. *IEEE Trans. Image Process.*, 19(9):2345–2356, September 2010.

[2] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems. *IEEE Trans. Image Process.*, 20(3):681–695, March 2011.

[3] 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*, 3(1):1–122, 2011.

[4] S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20(1):33–61, 1998.

[5] J. Eckstein and D. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Math. Program.*, 5:293–318, 1992.

[6] I. Selesnick. The short-time Fourier transform and speech denoising. *Connexions*, 2009. http://cnx.org/content/m32294.

[7] I. Selesnick. Introduction to sparsity in signal processing. *Connexions*, 2012. http://cnx.org/content/m43545/1.3/.

[8] I. W. Selesnick. Resonance-based signal decomposition: A new sparsity-enabled signal analysis method. *Signal Processing*, 91(12):2793 – 2809, 2011.

[9] I. W. Selesnick, K. Y. Li, S. U. Pillai, and B. Himed. Doppler-streak attenuation via oscillatory-plus-transient decomposition of IQ data. In *IET Int. Conf. Radar Systems*, 2012.

[10] J.-L. Starck, M. Elad, and D. Donoho. Redundant multiscale transforms and their application for morphological component analysis. *Advances in Imaging and Electron Physics*, 132:287–348, 2004.

[11] J.-L. Starck, M. Elad, and D. Donoho. Image decomposition via the combination of sparse representation and a variational approach. *IEEE Trans. Image Process.*, 14(10):1570–1582, October 2005.