Simultaneous Low-Pass Filtering and Total Variation Denoising

Ivan W. Selesnick, Harry L. Graber, Douglas S. Pfeil, and Randall L. Barbour

Abstract—This paper seeks to combine linear time-invariant (LTI) filtering and sparsity-based denoising in a principled way in order to effectively filter (denoise) a wider class of signals. LTI filtering is most suitable for signals restricted to a known frequency band, while sparsity-based denoising is suitable for signals admitting a sparse representation with respect to a known transform. However, some signals cannot be accurately categorized as either band-limited or sparse. This paper addresses the problem of filtering noisy data for the particular case where the underlying signal comprises a low-frequency component and a sparse-derivative component. A convex optimization approach is presented and two algorithms derived, one based on majorization-minimization (MM), the other based on the alternating direction method of multipliers (ADMM). It is shown that a particular choice of discrete-time filter, namely zero-phase non-causal recursive filters for finite-length data formulated in terms of banded matrices, makes the algorithms computationally efficient and effective. The computational efficiency of the algorithm stems from the use of fast algorithms for solving banded systems of linear equations. The method is illustrated using data from a physiological-measurement technique (i.e., near infrared spectroscopic time series imaging) that in many cases yields data that is well-approximated as the sum of low-frequency, sparse-derivative and noise components.

Keywords: total variation denoising, sparse signal, sparsity, low-pass filter, Butterworth filter, zero-phase filter.

I. INTRODUCTION

Linear time-invariant (LTI) filters are widely used in science, engineering, and general time series analysis. The properties of LTI filters are well understood, and many effective methods exist for their design and efficient implementation [46]. Roughly, LTI filters are most suitable when the signal of interest is (approximately) restricted to a known frequency band. At the same time, the effectiveness of an alternate approach to signal filtering, based on sparsity, has been increasingly recognized [17], [25], [44], [53]. Over the past 10-15 years, the development of algorithms and theory for sparsity-based signal processing has been an active research area, and many algorithms for sparsity-based denoising (and reconstruction, etc.) have been developed [49], [52]. These are most suitable when the signal of interest either is itself sparse or admits a sparse representation.

However, the signals arising in some applications are more complex: they are neither isolated to a specific frequency band nor do they admit a highly sparse representation. For such signals, neither LTI filtering nor sparsity-based denoising is appropriate by itself. Can conventional LTI filtering and more recent sparsity-based denoising methods be combined in a principled way, to effectively filter (denoise) a wider class of signals than either approach can alone?

This paper addresses the problem of filtering noisy data where the underlying signal comprises a low-frequency component and a sparse-derivative component. It is assumed here that the noisy data $y(n)$ can be modeled as

$$y(n) = f(n) + x(n) + w(n), \quad n = 0, \ldots, N - 1 \quad (1)$$

where $f$ is a low-pass signal, $x$ is a sparse-derivative signal, and $w$ is stationary white Gaussian noise. For noisy data such as $y$ in (1), neither conventional low-pass filtering nor sparsity-based denoising is suitable. Further, (1) is a good model for many types of signals that arise in practice, for example, in nano-particle biosensing (e.g., Fig. 3a in [22]) and near infrared spectroscopic imaging (e.g., Fig. 9 in [4]).

Note that if the low-pass signal $f$ were observed in noise alone ($y = f + w$), then low-pass filtering (LPF) would provide a good estimate of $f$; i.e., $f \approx \text{LPF}(f + w)$. On the other hand, if the sparse-derivative signal $x$ were observed in noise alone ($y = x + w$), then total variation denoising (TVD) would provide a good estimate of $x$; i.e., $x \approx \text{TVD}(x + w)$ [50]. Given noisy data of the form $y = f + x + w$, we seek a simple optimization-based approach so as to estimate $f$ and $x$ individually.

In this paper, an optimization approach is presented that enables the simultaneous use of low-pass filtering and sparsity-based denoising to estimate a low-pass signal and a sparse-derivative signal from a single noisy additive mixture, cf. (1). The optimization problem we formulate involves the minimization of a non-differentiable, strictly convex cost function. We present two iterative algorithms, the derivations of which are based on majorization-minimization (MM) and the alternating direction method of multipliers (ADMM) respectively.

In addition, this paper explains how a suitable choice of discrete-time filter makes the proposed approach effective and computationally efficient. Namely, we describe the design and implementation of a zero-phase non-causal recursive filter for finite-length data, formulated in terms of banded matrices. We choose recursive filters for their computational efficiency in comparison with non-recursive filters, and the zero-phase property to eliminate phase distortion (phase/time off-set issues). As the algorithms are intended primarily for batch-mode
processing, the filters need not be causal. We cast the recursive discrete-time filter in terms of a matrix formulation so as to easily and accurately incorporate it into the optimization framework and because it facilitates the implementation of the filter on finite-length data. Furthermore, the formulation is such that all matrix operations in the devised algorithms involve only banded matrices, thereby exploiting the high computational efficiency of solvers for banded linear systems [48, Sect 2.4] and of sparse matrix multiplication.

The computational efficiency of the proposed algorithms also draws on recent developments in sparse-derivative signal denoising (i.e., total variation (TV) denoising [9], [15], [50]). In particular, we note that the exact solution to the 1D TV denoising problem can be calculated by fast constructive algorithms [20], [34]. The algorithms presented here draw on this and the ‘fused lasso signal approximator’ [29].

The proposed algorithm is illustrated on simulated time series and on experimental data obtained from a NIRS time series measurement, which, as indicated earlier, frequently produces data that are well approximated by the model in (1).

After Sect. II on preliminaries, Sect. III presents the formulation of the optimization problem for simultaneous low-pass filtering and sparse-signal denoising. Section IV derives an iterative algorithm for solving the optimization problem. Section V presents recursive discrete-time filters to be used in the algorithm. Section VII addresses the case where both the signal itself and its derivative are sparse. Section VII-A presents the application of the proposed approach to an example of NIRS time-series measurement data. Section VII-B presents an algorithm for the constrained formulation.

A. Related work

The problem addressed in this paper is closely related to the problem addressed in Ref. [51]; however, the new approach described here has several advantages over the method described therein. While Ref. [51] uses least squares polynomial approximation on overlapping blocks for signal smoothing, the new approach uses LTI filtering. As a consequence, the new approach results in a time-invariant signal processing algorithm, in contrast to the approach of Ref. [51]. In addition, compared with Ref. [51], the new approach employs a more general sparse-derivative model that incorporates the sparsity of both the signal and its derivative. This is useful in practice for separating transient waveforms/pulses from a low frequency background signal. Also, unlike Ref. [51], one of the new algorithms is devised so that sparse-derivative denoising is an explicit step within the algorithm, which means that new fast algorithms for TV denoising (e.g. Ref. [20]) can be readily incorporated into the proposed algorithm. In addition, using the new approach, both the unconstrained and constrained formulations of the proposed optimization problem can exploit fast solvers for banded linear systems. In contrast, the approach of Ref. [51] allows this only for the unconstrained formulation, unless the blocks are non-overlapping. Consequently, the new approach is computationally efficient for the constrained formulation, unlike the approach of Ref. [51].

Many papers have addressed the problem of filtering/denoising piecewise smooth signals, a class of signals that includes the signals taken up in this paper, i.e. $y$ in (1). However, as noted in Ref. [51], much of the work on this topic explicitly or implicitly models the underlying signal of interest as being composed of unrelated smooth intervals/regions separated by discontinuities (or blurred discontinuities) [13], [24], [41]. This is particularly appropriate in image processing wherein distinct smooth regions correspond to distinct objects and discontinuities correspond to the edges of objects (e.g. one object occluding another) [30]. Under this model, smoothing across discontinuities should be avoided, to prevent blurring of edges. The signal model (1) taken up in this paper differs in an important way: it models the smooth behavior on the two sides of a discontinuity as being due to a common low-pass signal, i.e. $f$ in (1). In contrast to most methods developed for processing piecewise smooth signals, the proposed method seeks to exploit the common smooth behavior on both sides of a discontinuity, as in Ref. [51].

The problem addressed in this paper is a type of sparsity-based denoising problem, and as such, it is related to the general problem of sparse signal estimation. Many papers, especially over the last fifteen years, have addressed the problem of filtering/denoising signals, both 1D and multidimensional, using sparse representations via suitably chosen transforms (wavelet, etc.) [21], [38], [47]. The method described here has some similarities to sparse-transform domain filtering [44]. For example, in wavelet-domain thresholding (and more general nonlinear processing, data adaptive, etc.), the low-pass wavelet subband is often left intact (no thresholding is applied to it). In this case, a large threshold value leads to a denoised signal that is essentially a low-pass filtered version of the noisy data. When the threshold is small, the result of wavelet-domain thresholding is essentially the noisy data itself. Likewise, the proposed algorithms involve a regularization parameter $\lambda$. When $\lambda$ is set to a large value, the algorithms essentially perform low-pass filtering; when $\lambda$ is small, they have little effect and the output essentially is the noisy data.

More generally, as wavelet and related multiscale transforms [39] include a low-pass subband, which can be regularized separately from other subbands, wavelet-domain processing provides the opportunity to combine low-pass filtering and sparsity-based processing in a single framework. However, the proposed approach differs from many wavelet-based approaches in several aspects. For one, it completely decouples the low-pass filter from the sparse-signal description, while in wavelet-domain denoising the low-pass subband/filter is determined by the specific wavelet transform utilized. Hence, in the proposed approach, the design of the low-pass filter can be based on the properties of the low-pass component in the signal model ($f$ in (1)). Moreover, the proposed method, not being based on transform-domain sparsity, avoids the complications associated with selecting and implementing a suitable wavelet (or other) transform (choice of transform, choice of wavelet filters, boundary extensions, radix-2 length constraints, etc.). In addition, as the proposed approach is based on TV denoising, it avoids the ‘pseudo-Gibbs’ phenomenon that is present to varying degree in many wavelet-based methods [18].
II. Preliminaries

A. Notation

Vectors and matrices are represented by lower- and uppercase bold respectively (e.g. \( \mathbf{x} \) and \( \mathbf{H} \)). Finite-length discrete-time signals will be represented as lower-case italicized or bold. The \( N \)-point signal \( \mathbf{x} \) is represented by the vector

\[
\mathbf{x} = [x(0), \ldots, x(N-1)]^T
\]

where \([ \cdots ]^T\) denotes the transpose. Matrix \( \mathbf{D} \) is defined as

\[
\mathbf{D} = \begin{bmatrix}
-1 & 1 \\
-1 & 1 \\
\vdots & \vdots \\
-1 & 1
\end{bmatrix}.
\]

The first-order difference of an \( N \)-point signal \( \mathbf{x} \) is given by \( \mathbf{Dx} \) where \( \mathbf{D} \) is of size \((N-1) \times N\).

The notation \( \| \mathbf{v} \|_1 \) denotes the \( \ell_1 \) norm of the vector \( \mathbf{v} \), defined as \( \| \mathbf{v} \|_1 = \sum_n |v(n)| \). The notation \( \| \mathbf{v} \|_2 \) denotes the \( \ell_2 \) norm of the vector \( \mathbf{v} \), defined as \( \| \mathbf{v} \|_2 = (\sum_n |v(n)|^2)^{1/2} \).

The soft-threshold function [23] is defined as

\[
\text{soft}(x, T) := \begin{cases} x - T (x/|x|), & |x| > T \\ 0, & |x| \leq T \end{cases}
\]

for \( x \in \mathbb{C} \) and \( T > 0 \). This is the usual soft-threshold function on the real line, generalized here to the complex plane. For a vector \( \mathbf{x} \) or signal \( x(n) \), the notation \( \text{soft}(x, T) \) refers to the soft-threshold function applied element-wise to \( \mathbf{x} \).

B. Total Variation Denoising

Sparse-derivative signal denoising refers to the problem of estimating a signal \( x \), having a sparse or approximately sparse derivative, from a noisy observation, e.g. \( y = x + w \). As is well known, the \( \ell_1 \) norm is a convex proxy for sparsity, so it is practical to formulate sparse-derivative denoising as the problem of minimizing the \( \ell_1 \) norm of the derivative of \( x \) subject to a data fidelity constraint. For discrete-time data, the simplest approximation of the derivative is the first-order difference; hence, consider the minimization of \( \| \mathbf{Dx} \|_1 \).

Assuming the \( N \)-point signal \( \mathbf{x} \) is observed in additive white Gaussian noise with variance \( \sigma^2 \), a suitable data fidelity constraint is \( \| \mathbf{y} - \mathbf{x} \|_2^2 \leq N \sigma^2 \). This leads to the constrained optimization problem

\[
\begin{alignat}{2}
\arg \min_{\mathbf{x}} & \quad \| \mathbf{Dx} \|_1 \quad & \quad \text{(3a)} \\
\text{subject to} & \quad \| \mathbf{y} - \mathbf{x} \|_2^2 \leq N \sigma^2. & \quad \text{(3b)}
\end{alignat}
\]

Problem (3) is equivalent, for suitable \( \lambda \), to the unconstrained optimization problem

\[
\begin{alignat}{2}
\arg \min_{\mathbf{x}} & \quad \frac{1}{2} \| \mathbf{y} - \mathbf{x} \|_2^2 + \lambda \| \mathbf{Dx} \|_1 \quad & \quad \text{(4)} \\
\text{i.e.,} & \quad \arg \min_{\mathbf{x}} \frac{1}{2} \sum_{n=0}^{N-1} |y(n) - x(n)|^2 + \lambda \sum_{n=1}^{N-1} |x(n) - x(n-1)|. & \end{alignat}
\]

Problems (3) and (4) are two forms of the total variation denoising (TVD) problem [16]. The unconstrained form (4) is more commonly used than the constrained form (3).

We will denote the solution to problem (4) as \( \text{tvd}(\mathbf{y}, \lambda) \),

\[
\text{tvd}(\mathbf{y}, \lambda) := \arg \min_{\mathbf{x}} \left\{ \frac{1}{2} \| \mathbf{y} - \mathbf{x} \|_2^2 + \lambda \| \mathbf{Dx} \|_1 \right\}. \quad \text{(5)}
\]

There is no explicit solution to (4), but a fast algorithm to compute the exact solution has been developed [20] (with a C implementation).

Increasing the parameter \( \lambda \) has the effect of making the solution \( \mathbf{x} \) more nearly piecewise constant. Instead of the first-order difference, other approximations of derivatives can be used for sparse-derivative denoising. The notion of total variation has been further generalized in several ways to make it effective for a broader class of signals [12], [37], [40], [43].

C. Fused Lasso Signal Approximator

If both the signal \( \mathbf{x} \) and its derivative are sparse, then the denoising problem is more appropriately formulated as

\[
\arg \min_{\mathbf{x}} \left\{ \frac{1}{2} \| \mathbf{y} - \mathbf{x} \|_2^2 + \lambda_0 \| \mathbf{x} \|_1 + \lambda_1 \| \mathbf{Dx} \|_1 \right\}. \quad \text{(6)}
\]

This is a special case of a compound penalty function [1], [10], wherein two or more regularizers are used to promote distinct properties of the signal to be recovered.

The specific problem (6) is referred to as the ‘fused lasso signal approximator’ in Ref. [29]. Interestingly, Proposition 1 in Ref. [29] shows that problem (6) is equivalent to (4) in the sense that the solution to (6) can be obtained explicitly from the solution to (4). Specifically, the solution to (6) is given by

\[
\mathbf{x} = \text{soft}(\text{tvd}(\mathbf{y}, \lambda_1), \lambda_0). \quad \text{(7)}
\]

Hence, it is not necessary to have a separate algorithm for (6); it suffices to have an algorithm for the TVD problem (5).

D. Majorization-Minimization

The MM procedure replaces a difficult minimization problem with a sequence of simpler ones [27]. To minimize a function \( F(\mathbf{x}) \), the MM procedure produces a sequence \( \mathbf{x}_k \) according to

\[
\mathbf{x}_{k+1} = \arg \min_{\mathbf{x}} G_k(\mathbf{x}) \quad \text{with} \quad \mathbf{x}_0 \text{ initialized} \quad \text{to} \quad \mathbf{x}_{k+1} \quad \text{according to} \quad \mathbf{x}_k \quad \text{for} \quad k \geq 0. \quad \text{(8)}
\]

where \( k \) is the iteration index, \( k \geq 0 \). The function \( G_k(\mathbf{x}) \) is any convex majorizer of \( F(\mathbf{x}) \) that coincides with \( F(\mathbf{x}) \) at \( \mathbf{x} = \mathbf{x}_k \). That is, \( G_k(\mathbf{x}) \) should satisfy: \( G_k(\mathbf{x}) \geq F(\mathbf{x}) \) \( \forall \mathbf{x} \), and \( G_k(\mathbf{x}_k) = F(\mathbf{x}_k) \). With initialization \( \mathbf{x}_0 \), the update (8) produces a sequence \( \mathbf{x}_k \) converging to the minimizer of \( F(\mathbf{x}) \). For more details, see Ref. [27] and references therein.

Below, a majorizer for the \( \ell_1 \) norm will be used. To that end, note that

\[
\frac{1}{2} \mathbf{x}^H \mathbf{A}_k^{-1} \mathbf{x} + \frac{1}{2} \| \mathbf{x} \|_1 \geq \| \mathbf{x} \|_1, \quad \mathbf{A}_k = \text{diag}(|\mathbf{x}_k|), \quad \text{(9)}
\]

with equality when \( \mathbf{x} = \mathbf{x}_k \). Therefore, the left-hand-side of (9) is a majorizer of \( \| \mathbf{x} \|_1 \) and we will use it as \( G(\mathbf{x}) \) in the MM procedure.
III. LPF/TVD PROBLEM FORMULATION

Consider the problem of observing a noisy additive mixture of a low-pass signal $f$ and a sparse-derivative signal $x$, 

$$y = f + x + w,$$  

where is assumed that $w$ is stationary white Gaussian noise with variance $\sigma^2$. We seek estimates 

$$\hat{x} \approx x, \quad \hat{f} \approx f.$$  

(11)

Given an estimate $\hat{x}$ of $x$, we will estimate $f$ as 

$$\hat{f} := \text{LPF}(y - \hat{x}),$$  

(12)

where LPF is a specified low-pass filter. Therefore, the problem is to find $\hat{x}$.

Using (12) in (11) gives 

$$\text{LPF}(y - \hat{x}) \approx f.$$  

(13)

Using (10) in (13) gives 

$$\text{LPF}(y - \hat{x}) \approx y - x - w.$$  

(14)

Using (11) in (14) gives 

$$\text{LPF}(y - \hat{x}) \approx y - \hat{x} - w$$  

or 

$$(y - \hat{x}) - \text{LPF}(y - \hat{x}) \approx w.$$  

(16)

Note that the left-hand side of (16) constitutes a high-pass filter of $y - \hat{x}$. (This assumes that the frequency response of the low-pass filter is zero-phase or approximately zero-phase.)

Defining $\text{HPF} := I - \text{LPF}$, we write (16) as 

$$\text{HPF}(y - \hat{x}) \approx w.$$  

(17)

The expression (16) contains the data $y$, the estimate $\hat{x}$ that we seek to determine, and the noise signal $w$, but not the unknown signal $f$ or $x$; hence, it can be used to derive an estimate $\hat{x}$.

Using bold-face $H$ to represent the high-pass filter matrix, we have $H(y - \hat{x}) \approx w$.

Hence, $\hat{x}$ should be chosen so that $H(y - \hat{x})$ resembles a white Gaussian random vector with variance $\sigma^2$. At the same time, $\hat{x}$ should have a sparse derivative; i.e., the $\ell_1$ norm of $Dx$ should be small. Therefore, the estimation of $x$ can be formulated as the constrained optimization problem 

$$\arg\min_x \|Dx\|_1$$  

such that $\|H(y - x)\|_2^2 \leq N \sigma^2.$  

(18a,18b)

For suitable $\lambda$, an equivalent formulation is the unconstrained optimization problem: 

$$\arg\min_x \left\{ \frac{1}{2}\|H(y - x)\|_2^2 + \lambda\|Dx\|_1 \right\}.$$  

(19)

We refer to (18) and (19) as the LPF/TVD problem, the unconstrained form being computationally easier to solve. In Sect. IV, we derive an algorithm for solving (19), and consider the selection of a suitable $\lambda$. An algorithm for the constrained form (18) will be given in Sect. VII-B.

We will set the high-pass filter $H$ to be of the form 

$$H = A^{-1}B,$$  

(20)

where $A$ and $B$ are banded matrices. The design of the filter $H$ is presented in Sect. V, where it will be seen that the mathematical form of (20) flows naturally from the standard difference-equation formulation of LTI filtering. Note that while $A$ is banded, $A^{-1}$ is not, and hence neither is $H$.

The low-pass filter $LPF$ to estimate $f$ in (12) will be given by $LPF = I - HPF$ with filter matrix $L = I - A^{-1}B$.

IV. LPF/TVD ALGORITHM

Large-scale non-differentiable convex optimizations arise in many signal/image processing tasks (sparsity-based denoising, deconvolution, compressed sensing, etc.). Consequently, numerous effective algorithms have been developed for such problems, particularly for those of the form (19) [19], [26], [32]. In this section we apply the ‘majorization-minimization’ (MM) approach [27] to develop an algorithm for solving (19).

Note that the solution to (19) is unique only up to an additive constant. To make the solution unique, and to facilitate the subsequent use of MM, the following change of variables can be used. Let 

$$x = Su$$  

(21)

where $S$ is a matrix of the form 

$$S := \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$  

(22)

of size $N \times (N - 1)$. It represents a cumulative sum. Note that 

$$DS = I,$$  

(23)

i.e., $S$ is a discrete anti-derivative. Therefore, 

$$Dx = DSu = u.$$  

(24)

We also note that for the filters to be introduced in Sect. V, the matrix $B$ can be expressed as 

$$B = B_1D$$  

(25)

where $B_1$ is banded matrix.

With (21), problem (19) can be written as 

$$\arg\min_u \left\{ F(u) = \frac{1}{2}\|H(y - Su)\|_2^2 + \lambda\|u\|_1 \right\}.$$  

(26)

With the optimal solution $u$, the solution to (19) is obtained as $x = Su$. To minimize (26) using MM, we need a majorizer $G_k(u)$ of the cost function $F(u)$ in (26). Using (9), a majorizer of $F(u)$ is given by 

$$G_k(u) = \frac{1}{2}\|H(y - Su)\|_2^2 + \lambda \frac{u^T A_k^{-1} u + \lambda}{2} \|u_k\|_1,$$  

(27)

where $A_k$ is the diagonal matrix, 

$$[A_k]_{n,n} = |u_k(n)|.$$  

Using (23) and (25), 

$$HS = A^{-1}BS = A^{-1}B_1DS = A^{-1}B_1,$$  

(28)
and the majorizor can be written as

\[ G_k(u) = \frac{1}{2} \|A^{-1}B y - A^{-1}B_1 u\|^2 + \frac{\lambda}{2} u^T \Lambda^{-1} u + C \]

where \( C \) does not depend on \( u \). The MM update is given by

\[ u_{k+1} = \arg \min_u G_k(u) \tag{27} \]

which has the explicit form

\[ u_{k+1} = \left( B_1^T(AA^T)^{-1}B_1 + \lambda \Lambda_k^{-1} \right)^{-1} B_1^T(AA^T)^{-1}B y. \]

A numerical problem is that as the iterations progress, many values of \( u_k \) are expected to go to zero (due to the sparsity promoting properties of the \( \ell_1 \) norm), and therefore some entries of \( \Lambda_k^{-1} \) will go to infinity. This issue is addressed, as described in Ref. [28], by rewriting the equation using the matrix inverse lemma:

\[
\left( B_1^T(AA^T)^{-1}B_1 + \lambda \Lambda_k^{-1} \right)^{-1} = \frac{1}{\lambda} \Lambda_k - \frac{1}{\lambda} \Lambda_k B_1^T \left( \lambda(AA^T + B_1 \Lambda_k B_1^T) \right)^{-1} B_1 \Lambda_k.
\]

The indicated matrix is banded because \( A, B_1, \) and \( \Lambda_k \) are all banded. Using (28), the MM update (27) can be implemented as:

\[
b \leftarrow \frac{1}{\lambda} B_1^T(AA^T)^{-1}B y \\
\Lambda_k \leftarrow \text{diag}(|u_k|) \\
u_{k+1} \leftarrow \Lambda_k \left[ b - B_1^T \left( \lambda(AA^T + B_1 \Lambda_k B_1^T) \right)^{-1} B_1 \Lambda_k b \right].
\]

The update can be implemented using fast solvers for banded systems of linear equations [5], [33] [48, Sect 2.4]. Furthermore, as all matrices are banded, matrix-vector multiplications are also computationally efficient.

The update equations constitute an algorithm, that we refer to as the LPF/TVD algorithm, for solving (19). The complete algorithm is presented as Algorithm 1. Once \( x \) is obtained, the low-pass component, \( f \), is obtained by applying the low-pass filter \( L = I - A^{-1}B \) to \( (y - x) \), cf. (12).

**Algorithm 1 LPF/TVD**

low-pass filtering / total variation denoising via (19)

**Input:** \( y \in \mathbb{R}^N \), \( \lambda > 0 \)

**Output:** \( x, f \in \mathbb{R}^N \)

1: \( b \leftarrow (1/\lambda) B_1^T(AA^T)^{-1}B y \\
2: \ u \leftarrow Dy \\
3: \text{repeat} \\
4: \ A \leftarrow \text{diag}(|u|) \\
5: \ Q \leftarrow \lambda(AA^T + B_1 \Lambda_k B_1^T) \\
6: \ u \leftarrow A \left[ b - B_1^T Q^{-1} B_1 \Lambda_k b \right] \\
7: \text{until convergence} \\
8: \ x \leftarrow Su \\
9: \ f \leftarrow (y - x) - A^{-1}B(y - x) \\
10: \text{return } x, f
\]

The change of variables \( x = Su \) is important above, because otherwise the MM approach leads here to dense system of equations. The efficiency of the LPF/TVD algorithm relies on the system being banded. Each iteration of the algorithm has \( O(dN) \) computational cost, where \( d \) is the order of the filter \( H \). Our implementation is programmed in MATLAB which in turn uses LAPACK for solving banded systems [5], [31].

**Optimality conditions.** The optimality conditions characterizing the minimizer of (19) can be adapted from [6, Prop 1.3]. Define

\[ g = S^T H^T H (y - x), \quad u = Dx. \tag{29} \]

Then \( x \) minimizes (19) if and only if

\[ g(n) = \text{sign}(u(n)) \cdot \lambda, \quad \text{for } u(n) \neq 0 \]

\[ |g(n)| \leq \lambda, \quad \text{for } u(n) = 0. \tag{30} \]

Using (30), one can readily verify the optimality of a result produced by a numerical algorithm.

**Setting \( \lambda.** The optimality condition (30) can be used as a guideline to set the regularization parameter \( \lambda \). Note that if \( y \) consists of noise only (i.e. \( y = w \)), then (ideally) \( x \) will be identically zero. From (29), \( g \) and \( u \) are given in this case by \( g = S^T H^T H w \) and \( u = 0 \). This is optimal, according to (30), if \( \lambda \leq \max(g) = \max(S^T H^T H w) \). Choosing the minimal \( \lambda \) so as to avoid unnecessary attenuation/distortion of \( x \), we get the value

\[ \lambda = \max(S^T H^T H w) \tag{31} \]

which assures availability of the noise signal \( w \). Start- and end-transients should be omitted when using (31). In practice, the noise is not known, but its statistics may be known and an approximate maximum value precomputed. For example, if the noise is zero-mean white Gaussian with variance \( \sigma^2 \), then we may compute the standard deviation \( \sigma_o \) of \( S^T H^T H w \) and use the ‘3-sigma’ rule (or similar) in place of the maximum value, to obtain the guideline \( \lambda = 3\sigma_o \).

Note that this approach for setting \( \lambda \) uses no information regarding the signal \( x \) or its statistics. Therefore, as a non-Bayesian procedure, it will not give a value for \( \lambda \) that is optimal in the mean-square sense. However, it can be useful in practice and can be used as a reasonable starting point for other schemes for optimizing regularization parameters.

**V. LTI FILTERS AND SPARSE MATRICES**

This section addresses the design and implementation of discrete-time filters for the method described in Sections III and IV. In particular, we describe the design and implementation of zero-phase non-causal recursive high-pass filters in terms of banded matrices.

A discrete-time filter is described by the difference equation

\[ \sum_k a(k) y(n-k) = \sum_k b(k) x(n-k) \tag{32} \]

where \( x(n) \) and \( y(n) \) are the input and output signals respectively. The frequency response of the discrete-time filter is \( H(e^{j\omega}) = B(e^{j\omega})/A(e^{j\omega}) \).

We are interested in filtering finite-length signals specifically, because sparsity-based signal processing problems are generally formulated in terms of finite-length signals, and the
developed algorithms are targeted for the finite-length case. In particular, this is the case for TV denoising. To implement the difference equation (32) for finite-length signals, we write

$$Ay = Bx$$

where $A$ and $B$ are banded matrices. The output $y$ of the filter can be written as

$$y = A^{-1}Bx$$

(33)

which calls for the solution to a banded system. Note that for (33) to be meaningful, $B$ need not be invertible, but $A$ must be. Hence, $B$ need not be square, but $A$ must be.

Typically, there are both start- and end-transients when applying a discrete-time filter to a finite-length signal. The start-transients depend on the initial states of the filter which, if not specified, are usually taken to be zero or optimized so as to minimize transients [35]. In the approach given here, based on $y = A^{-1}Bx$ with banded matrices, the explicit specification of initial states is avoided.

**Example.** Consider a causal first-order Butterworth high-pass filter. The difference equation has the form

$$a_0 y(n) + a_1 y(n-1) = x(n) - x(n-1)$$

(34)

which can be written and implemented as $y = A^{-1}Bx$. The matrix $B$ is given by $B = D$, the first-order difference matrix of size $(N-1) \times N$ defined in (2), where $N$ is the length of the input signal $x$. The matrix $A$ is given by

$$A = \begin{bmatrix}
a_0 & a_0 & \cdots & a_1 \\
a_1 & a_0 & \cdots & a_1 \\
\vdots & \vdots & \ddots & \vdots \\
a_1 & a_0 & \cdots & a_0 \\
\end{bmatrix}$$

(35)

and is of size $(N-1) \times (N-1)$. The filter can be implemented using a fast solver for banded linear systems.

Using $y = A^{-1}Bx$, the filter can be applied to a length-$N$ signal $x$. Note that the output $y$ is of length $N-1$.

**A. Zero-phase filters**

In order to avoid unnecessary distortion, the filter should be zero-phase; i.e., the frequency response $H(e^{j\omega})$ should be real-valued. Equivalently, the impulse (temporal) response should be symmetric. Besides, expression (16) in the derivation of the problem formulation assumes the zero-phase property.

The zero-phase property also implies specific properties of matrices $A$ and $B$. Note that for a finite-length signal $x$, the temporal symmetry property suggests that the filter should behave the same ‘backwards’ as it does ‘forwards’, cf. zero-phase forward-backward filtering [35]. That is, applying the filter $H$ to the reversed version of $x$, then reversing the filter output, should be the same as applying the filter directly to the data $x$. Letting $J$ denote the reversal matrix (square matrix with 1’s on the anti-diagonal), the filter should satisfy

$$JHJ = H$$

(36)

where the dimension of $J$ is determined by the dimensions of $H$ (recall $H$ is rectangular). Note that if $A$ and $B$ satisfy

$$JAJ = A, \quad JBJ = B$$

(37)

then $H = A^{-1}B$ satisfies (36).

For the proposed LPF/TVD algorithm, the filter matrices should satisfy (37). Note that (35) does not. The following examples illustrate recursive zero-phase filters satisfying (37).

**Example.** A second-order non-causal zero-phase Butterworth high-pass filter is described by the difference equation

$$a_0 y(n+1) + a_0 y(n) + a_1 y(n-1) = -x(n+1) + 2x(n) - x(n-1)$$

(38)

which can be defined and implemented as $y = A^{-1}Bx$, where $B$ has the form

$$B = \begin{bmatrix}
-1 & 2 & -1 \\
-1 & 2 & -1 \\
-1 & 2 & -1 \\
-1 & 2 & -1
\end{bmatrix}$$

(39)

and is of size $(N-2) \times N$, where $N$ is the length of the input signal $x$. Matrix $A$ has the form

$$A = \begin{bmatrix}
a_0 & a_1 & a_1 & a_0 \\
a_0 & a_1 & a_1 & a_0 \\
a_1 & a_0 & a_1 & a_0 \\
a_1 & a_0 & a_1 & a_0
\end{bmatrix}$$

(40)

and is of size $(N-2) \times (N-2)$. These $A$ and $B$ satisfy (37). Note that the output signal $y$ is two samples shorter than the input signal $x$.

The transfer function of the filter (38) is given by

$$H(z) = \frac{B(z)}{A(z)} = \frac{-z + 2 - z^{-1}}{a_1 z + a_0 + a_1 z^{-1}}.$$  

(41)

In order that the filter defined by $L(z) = 1 - H(z)$ be a low-pass filter with a zero at the Nyquist frequency (i.e., $\omega = \pi$), the gain $c \in \mathbb{R}$ of $H(z)$ at the Nyquist frequency should be unity. For the system (38), $c$ is found by setting $x(n) = (-1)^n$ and $y(n) = c(-1)^n$ and solving for $c$ to obtain $c = 4/(a_0 - 2a_1)$. Equivalently, $c$ can be obtained as $c = H(-1)$. Hence, for the high-pass filter (38) to have unity Nyquist gain, the coefficients should satisfy

$$a_0 - 2a_1 = 4.$$  

Then the frequency response is given by

$$H(e^{j\omega}) = \frac{2 - 2\cos(\omega)}{a_0 + (a_0 - 4)\cos(\omega)}.$$  

The coefficient $a_0$ may be set so that the frequency response has a specified cut-off frequency $\omega_c$. Defining $\omega_c$ as that frequency where the frequency response is one half, $H(e^{j\omega_c}) = 0.5$, one obtains

$$a_0 = 4/(1 + \cos(\omega_c)).$$
For example, setting the cut-off frequency to $\omega_c = 0.1\pi$, gives $a_0 = 2.050$, $a_1 = -0.975$ This high-pass filter is illustrated in Fig. 1. The pole is at $z = 0.726$. This recursive filter is non-causal with a ‘symmetric’ time-domain response (due to finite-length processing, the time-domain response can not have infinite length).

We have referred to this filter as a zero-phase filter. That usually means that the filter has a symmetric impulse response. In the context of finite-length signals, the response to an impulse centered at $n = n_0$ is zero whenever the input signal is of the form $x(n) = k_0 + k_1 n$. Therefore, the low-pass filter, defined by $L = I - H = I - A^{-1}B$, exactly preserves polynomial signals of degree 1.

### B. Higher-order high-pass filter

Consider the transfer function

$$H(z) = \frac{\{-z^2 - z^{-1}\}^d}{\{-z^2 - 2z^{-1}\}^d + \alpha \{z^2 + 2z^{-1}\}^d}.$$  \hspace{1cm} (42)

The filter $H(z)$ has a 2d-order zero at $z = 1$, so the frequency response $H(e^{j\omega})$ is zero at $\omega = 0$, as are its first $2d - 1$ derivatives. Also, note that $H(z)$ can also be written as

$$H(z) = 1 - \frac{\alpha \{z^2 + 2z^{-1}\}^d}{\{-z^2 - 2z^{-1}\}^d + \alpha \{z^2 + 2z^{-1}\}^d}.$$  \hspace{1cm} (42)

The numerator of the second term has a zero of order 2d at $z = -1$, so the frequency response $H(e^{j\omega})$ has unity gain at $\omega = \pi$, and its first $2d - 1$ derivatives are zero there. That is, the frequency response is maximally-flat at dc and at the Nyquist frequency; hence, this is a zero-phase digital Butterworth filter.

The filter $H(z)$ in (42) is defined by the positive integer $d$ and by $\alpha$. The parameter $\alpha$ can be set so that the frequency response has a specified cut-off frequency $\omega_c$. Setting the gain at the cut-off frequency to one half, $H(e^{j\omega_c}) = 0.5$, gives the equation

$$\frac{[1 - \cos(\omega_c)]^d}{[1 - \cos(\omega_c)]^d + \alpha [1 + \cos(\omega_c)]^d} = \frac{1}{2}.$$  \hspace{1cm} (42)

Solving for $\alpha$ gives

$$\alpha = \left(\frac{1 - \cos(\omega_c)}{1 + \cos(\omega_c)}\right)^d.$$  \hspace{1cm} (42)

The zero-phase high-pass Butterworth filter (42) can be implemented as $y = A^{-1}Bx$ where

1) $B$ is a banded matrix of size $(N - 2d) \times N$.
2) $A$ is a square symmetric banded matrix of size $(N - 2d) \times (N - 2d)$.
3) Both $A$ and $B$ have bandwidth $2d + 1$; that is, in addition to the main diagonal, they have $d$ diagonals above and below the main diagonal.

Note that this filter maps an input signal of length $N$ to an output signal of length $N - 2d$. When $d = 1$, we get the high-pass filter (41) as a special case. In this case, $A$ and $B$, given by (39) and (40), are triadiagonal (a bandwidth of 3).

**Example.** Set $d = 2$ and the cut-off frequency to $\omega_c = 0.1\pi$. The matrix $B$ is of size $(N - 4) \times N$ and has the form

$$B = \begin{bmatrix}
1 & -4 & 6 & -4 & 1 \\
1 & -4 & 6 & -4 & 1 \\
& & \ddots & \ddots & \ddots \\
& & & 1 & -4 & 6 & -4 & 1
\end{bmatrix},$$

with bandwidth 5. With $\omega_c = 0.1\pi$, one obtains $\alpha = 6.29 \times 10^{-4}$. $A$ will be a banded symmetric square matrix with bandwidth 5. The coefficients of $A$ are $a_0 = 6.0038$, $a_1 = -3.9975$, $a_2 = 1.0006$, where $a_k$ lies on diagonal $\pm k$. The resulting fourth-order zero-phase filter is shown in Fig. 2.
C. Low-pass filter

The LPF/TVD algorithm provides an estimate, \( \hat{x} \), of the sparse-derivative component and calls for the high-pass filter \( H = A^{-1}B \). The algorithm does not use a low-pass filter. But, to obtain an estimate \( f \) of the low-pass component, recall that we need the low-pass filter denoted above as \( \text{LPF} = I - \text{HPF} \). A low-pass filter of this form is trivially performed by subtracting the high-pass filter output from its input. However, note that for the high-pass filter described in Section V-B, the matrices \( B \) and \( H \) are rectangular; the output of the high-pass filter is shorter than its input by \( 2d \) samples (\( d \) at the beginning and \( d \) at the end). Hence to implement the low-pass filter, the input signal should likewise be truncated so that the subtraction can be performed.

The low-pass filter matrix \( L \) is therefore given by \( L = I_d - A^{-1}B \) where \( I_d \) is the identity matrix with the first \( d \) and last \( d \) rows removed. The matrix \( I_d \) is of size \((N-2d)\times N\). The signal \( I_d x \) is obtained by deleting the first \( d \) and last \( d \) samples from \( x \).

Based on the high-pass filter (42), the low-pass filter \( L(z) := 1 - H(z) \) has the transfer function

\[
L(z) = \frac{\alpha (z + 2 + z^{-1})^d}{(-z + 2 - z^{-1})^d + \alpha (z + 2 + z^{-1})^d}
\]

with a \( 2d \)-order zero at \( z = -1 \). The filter matrix is given by \( L = I_d - A^{-1}B \).

**Example.** From the high-pass filter shown in Fig. 2 with \( d = 2 \), we obtain the low-pass filter illustrated in Fig. 3. The filter can be implemented as \( y = I_2 x - A^{-1}B x \). This filter passes third-order signals (of the form \( x(n) = k_0 + k_1 n + k_2 n^2 + k_3 n^3 \)) with no change, except for truncation by two samples at start and end.

**VI. LPF/TVD Example**

To illustrate simultaneous low-pass filtering and total-variation denoising using the LPF/TVD algorithm, we apply it to the noisy data \( y \) shown in Fig. 4. This is the same data used in the first example of Ref. [51], where smoothing was performed using a (non-time-invariant) least-squares polynomial approximation. The signal consists of a low-frequency sinusoid, two additive step discontinuities, and additive white Gaussian noise. In order to apply the new algorithm, we must specify a high-pass filter \( H \) and regularization parameter \( \lambda \). We use the fourth-order filter (42) with \( d = 2 \) and cut-off frequency \( \omega_c = 0.044\pi \). The parameter \( \lambda \) was set to 1.6 based on (31). The LPF/TVD algorithm was run for 30 iterations.

Figure 4 shows the sparse-derivative component \( x \) obtained from the algorithm. The low-pass component \( f \) is obtained by low-pass filtering \( y - x \); it is given by \( f = L(y - x) = I_2 (y - x) - H(y - x) \). The total LPF/TVD output, \( f + x \), shown in the second panel of Fig. 4, substantially smooths the data while preserving the discontinuities without introducing Gibbs-like phenomena.

The result shown in Fig. 4 is comparable to the result in Ref. [51], which compared favorably to several other methods as shown therein. While the method of Ref. [51] calls for the polynomial degree, block length, and overlapping-factor to be specified, the new method calls for the low-pass filter characteristics to be specified (filter order and cut-off frequency).
as the constrained problem:

$$\arg\min_{x,v} \left\{ \frac{1}{2} \|H y - Hx\|_2^2 + \lambda_0 \|v\|_1 + \lambda_1 \|Dv\|_1 \right\}$$

such that $v = x$. \hfill (45b)

Applying ADMM to (45) yields the iterative algorithm:

$$x \leftarrow \arg\min_x \|H y - Hx\|_2^2 + \mu \|v - x - d\|_2^2$$

$$v \leftarrow \arg\min_v \lambda_0 \|v\|_1 + \lambda_1 \|Dv\|_1 + 0.5 \mu \|v - x - d\|_2^2$$

$$d \leftarrow d - (v - x)$$

Go to (46a). \hfill (46d)

The iterative algorithm (46) alternates between minimization with respect to $x$ in (46a) and $v$ in (46b).

The algorithm (46) requires the parameter $\mu > 0$ be specified; the value of $\mu$ does not affect the solution to which the algorithm converges, but it does affect the overall convergence behavior. The convergence may be slow for a poor value of $\mu$. The variables $d$ and $v$ must also be initialized prior to the loop; however, as the cost function is convex, the algorithm converges to the unique minimizer regardless of the initialization [11]. We initialize $d$ and $v$ to the all-zero vector of size $y$. The loop is repeated until some stopping criterion is satisfied.

The solution to (46a) can be expressed as

$$x \leftarrow (H^T H + \mu I)^{-1} (H^T H y + \mu (v - d)).$$ \hfill (47)

From (20), we write

$$H^T H y = B^T (A A^T)^{-1} By.$$ \hfill (48)

Using the matrix inverse lemma, we obtain

$$\left(H^T H + \mu I\right)^{-1} = \frac{1}{\mu} \left[I - B^T \mu AA^T + BB^T \right]^{-1} B.$$ \hfill (49)

Using (48) and (49) in (47), line (46a) is implemented as

$$g \leftarrow \frac{1}{\mu} B^T (A A^T)^{-1} By + (v - d)$$ \hfill (50a)

$$x \leftarrow g - B^T (\mu AA^T + BB^T)^{-1} B g.$$ \hfill (50b)

Note that the first term on the right-hand side of (50a) need be computed only once, because $y$ is not updated in the loop (46); so it can be precomputed prior to the iteration.

Using (7), the solution to problem (46b) can be written as

$$v \leftarrow \text{soft}(tvd(x + d, \lambda_1/\mu), \lambda_0/\mu).$$

With these simplifications, the ADMM algorithm (46) for LPF/CSD can be readily implemented. The complete algorithm is listed as Algorithm 2. As in Sect. IV, all matrix operations involve only banded matrices and can therefore be implemented with high computational efficiency.

---

The latter parameters have the benefit of being more in line with conventional filtering practice and notions. In addition, the new method, being time-invariant, is not sensitive to the position of blocks relative to the signal, unlike that of [51].

This optimality conditions (30) are illustrated in Fig. 5 as a scatter plot. Each point represents a pair $(g(n), u(n))$, where $g(n)$ and $u(n)$ denote the $n$-th time samples of signals $g$ and $u$. Note that (30) means that each pair $(u, g)$ must lie on one of the line segments indicated by dashes. The sparsity of $Dx$ can be readily recognized from the fact that most of the points lie on the line $u = 0$.

---

VII. COMPOUND SPARSE DENOISING

In this section, the signal $x$ is modeled as being sparse itself and having a sparse derivative. As in Sect. III, we will estimate the low-pass component $f$ by applying a low-pass filter to $(y - \hat{x})$. In order to estimate $x$, instead of solving (19), we solve

$$\arg\min_x \left\{ \frac{1}{2} \|H (y - x)\|_2^2 + \lambda_0 \|x\|_1 + \lambda_1 \|Dx\|_1 \right\}$$

which promotes sparsity of both $x$ and its first-order derivative. The high-pass filter $H = \Phi^{-1} B$ is the same as used above, as it reflects the behavior of the low-pass component $f$. We refer to (44) as the LPF/CSD problem ("CSD" for compound sparse denoising).

The use of two regularizers as in (44) is referred to as compound regularization. Algorithms for general compound regularization are given in Refs. [10] and [1], which consider as an example the restoration of images that are both sparse and have sparse gradients. The particular compound regularization in (44) was also addressed in [29], as noted in Sec. II-C. In the following, we use Proposition 1 of Ref. [29], i.e., (7).

If the MM process is used, as in Sec. IV, to develop an algorithm for solving (44), then it leads to an algorithm calling for the solution to a dense (not-banded) system of $N$ linear equations, where $N$ is the length of signal $x$. This significantly increases the computational cost (by factor $N$). Therefore, in this section we apply the ‘alternating direction method of multipliers’ (ADMM) [2], [11]. ADMM is closely related to the split-Bregman algorithm and its variations [32], [55].

As in [2], we apply ‘variable splitting’ to decouple the terms of the cost function. In this case, problem (44) can be rewritten as the constrained problem:

$$\arg\min_{x,v} \left\{ \frac{1}{2} \|H (y - x)\|_2^2 + \lambda_0 \|v\|_1 + \lambda_1 \|Dv\|_1 \right\}$$

such that $v = x$. \hfill (45b)

---

Fig. 5. The optimality conditions (30) are satisfied.
Algorithm 2 LPF/CSD:
low-pass filtering / compound sparsity denoising via (44)

Input: $y \in \mathbb{R}^N$, $\lambda_0 > 0$, $\lambda_1 > 0$, $\mu > 0$
Output: $x, f \in \mathbb{R}^N$

1: $v \leftarrow 0$
2: $d \leftarrow 0$
3: $b \leftarrow (1/\mu)B^T(AA^T)^{-1}By$
4: repeat
5: \hspace{1em} $g \leftarrow b + v - d$
6: \hspace{1em} $x \leftarrow g - B^T(\mu AA^T + BB^T)^{-1}Bg$
7: \hspace{1em} $v \leftarrow \text{soft}(\text{tvd}(x + d, \lambda_1/\mu), \lambda_0/\mu)$
8: \hspace{1em} $d \leftarrow d - v + x$
9: until convergence
10: $f \leftarrow (y - x) - A^{-1}B(y - x)$
11: return $x, f$

A. NIRS Example

To illustrate simultaneous low-pass filtering and compound sparse denoising (LPF/CSD), the algorithm is applied to data acquired using a near infrared spectroscopic (NIRS) system [8]. The NIRS physiological modality uses light at two or more wavelengths in the $\sim (690-1000)$ nm range to monitor spatiotemporal fluctuations in tissue blood volume and blood oxygen saturation (we refer to these collectively as the ‘hemodynamic variables’) [7]. For a number of reasons, it is prone to producing time series data that are well described by the model (1):

1) Not uncommonly, there are long-term drifts in hemodynamic variables within the probed tissue volume (e.g., resulting from blood-pressure fluctuations) during the course of the measurement. These produce a low-frequency component in the data.
2) Additionally, the hemodynamic signal arises primarily from small blood vessels (arterioles, capillaries, venules) that tend to exhibit low-frequency oscillations called vasomotion [45].
3) Many NIRS measurement paradigms involve the intermittent presentation of stimuli to, or performance of tasks by, the human or animal subject [36]. These are intended to produce shifts in the magnitude of the hemodynamic variables approximately concurrent with the challenges, followed by a return to the previous level. That is, the signal is both sparse (i.e., resides at the baseline level most of the time) and has a sparse derivative (i.e., departs from the baseline a small number of times during the course of the measurement).
4) However, not all measurements are intervention-based. Resting-state monitoring also can be biologically informative and is commonly performed [54].
5) Unplanned events (e.g., postural shift, or subject sneezes) can introduce unwanted signal components that are sparse or have a sparse derivative.

The preceding considerations indicate that, depending on the experimental-design context, either the low-frequency or the sparse component may be the biological signal of interest.

For the particular example presented here, we have obtained data from a dynamic tissue-simulating phantom [8], while varying the strength of its absorption for NIR light in a manner that emulates the hemodynamic response of a human brain to intermittently delivered stimuli. Figure 6 illustrates data acquired by the system. The ‘hemodynamic’ pulses are observed in the presence of unavoidable low-frequency background processes and wideband noise. The signal of interest and its derivative are sparse relative to the low-frequency background signal and noise; for the illustrated measurement channel, the noise standard deviation is greater than 10% of the largest-magnitude hemodynamic pulse. The LPF/CSD algorithm simultaneously estimates and separates the low-pass background signal and the hemodynamic pulses, as illustrated in Fig. 6. For this computation, 50 iterations of the algorithm were performed, in a time of 0.14 seconds on a 2010 base-model Apple MacBook Pro (2.4 GHz Intel Core 2 Duo) with Matlab version 7.8 (R2009a). Note that the shape of the pulses are well preserved, in contrast to the commonly observed amplitude-reducing, edge-spread, and plateau-rounding, of LTI filtering alone. Figure 6 also illustrates the output of a band-pass filter (BPF) applied to the noisy time series data; neither the baseline drift nor residual noise can be further reduced without further distorting the hemodynamic pulses. Note that the BPF obscures the amplitude of the hemodynamic pulses relative to the baseline.

B. Constrained Formulation

When the noise statistics and a reasonable estimate of the energy of the residual are known, then the constrained form of the LPF/CSD problem (44) may be preferred. The constrained form is:

\[
\arg\min_x \{\lambda_0\|x\|_1 + \lambda_1\|Dx\|_1\} \quad (51a)
\]

such that $\|H(y - x)\|_2 \leq r$, \hspace{1em} (51b)

where $r$ is the energy of the residual. We refer to problem (51) as the C-LPF/CSD problem. Note that $\lambda_0$ and $\lambda_1$ can each be multiplied by the same constant without affecting the solution. Therefore, it can be assumed that $\lambda_0 + \lambda_1 = 1$ with $\lambda_i \geq 0$, for $i = 1, 2$. That is, compared to (44), which requires two independent parameters $\lambda_i$, the constrained form requires only one, but it assumes that $r$, the energy of the residual, is known. If $r$ is known (e.g., if it can be predicted based on the noise variance), then the advantage of (51) is that fewer $\lambda_i$ parameters need be specified.

In this section, the C-SALSA algorithm [3] is utilized to solve (51). First, variable splitting is performed to obtain the equivalent constrained problem:

\[
\arg\min_{x,v_0,v_1} \{\lambda_0\|v_0\|_1 + \lambda_1\|Dv_0\|_1\} \quad (52a)
\]

such that $\|Hy - v_1\|_2 \leq r$, \hspace{1em} (52b)

\[v_0 = x\] \hspace{1em} (52c)

\[v_1 = Hx\] \hspace{1em} (52d)

Second, the ADMM approach applied to (52) yields an iterative algorithm wherein each iteration consists of minimization...
Fig. 6. Near infrared spectroscopic (NIRS) data and LPF/CSD processing. The method simultaneously performs low-pass filtering and sparse signal denoising. In so doing, it separates the low-pass and sparse components.

with respect to \( v_i \) and \( x \).

\[
\begin{align*}
\arg \min_{v_0, v_1} & \left\{ \frac{\lambda_0}{2} \|v_0\|_1 + \lambda_1 \|Dv_0\|_1 \\
& + \mu_1 \|v_1 - Hx - d_1\|_2^2 \\
\right. \\
\text{such that} & \left. \|Hy - v_1\|_2 \leq r \right\} \quad (53a)
\end{align*}
\]

\[
x \leftarrow \arg \min_x \left\{ \frac{\mu_0}{2} \|v_0 - x - d_0\|_2^2 \\
+ \mu_1 \|v_1 - Hx - d_1\|_2^2 \right\} \quad (53b)
\]

\[
d_0 \leftarrow d_0 - (v_0 - x) \quad (53c)
\]

\[
d_1 \leftarrow d_1 - (v_1 - Hx) \quad (53d)
\]

Go to (53a).

In (53a), the variables \( v_0 \) and \( v_1 \) are decoupled and can be expressed as

\[
v_0 \leftarrow \text{soft}(\text{tvd}(x + d_0, \lambda_1/\mu_0), \lambda_0/\mu_0) \quad (54)
\]

\[
v_1 \leftarrow \text{proj}(Hx + d_1; Hy, r) \quad (55)
\]

where \( \text{proj}(v; c, r) \) denotes the projection of the point \( v \) onto the ball centered at \( c \) with radius \( r \). It is given explicitly by

\[
\text{proj}(v; c, r) := \begin{cases} 
\frac{r}{\|v - c\|_2} (v - c), & \|v - c\| > r \\
v, & \|v - c\| \leq r.
\end{cases}
\]

The solution to (53b) is given by

\[
x \leftarrow \left( \mu_0 I + \mu_1 H^T H \right)^{-1} \left( \mu_0 (v_0 - d_0) + \mu_1 H^T (v_1 - d_1) \right)
\]

which can simplified using the matrix inverse lemma, as in (50a), to obtain

\[
\left( \mu_0 I + \mu_1 H^T H \right)^{-1} = \frac{1}{\mu_0} I - \frac{\mu_1}{\mu_0} B^T (\mu_0 AA^T + \mu_1 BB^T)^{-1} B.
\]

With these simplifications, the iterative ADMM algorithm (53) can be readily implemented. The complete algorithm for C-LPF/CSD is listed as Algorithm 3.

Algorithm 3 C-LPF/CSD: low-pass filtering / compound sparsity denoising, constrained form (51)

**Input:** \( y \in \mathbb{R}^N, \lambda_0 > 0, \lambda_1 > 0, r > 0, \mu_0 > 0, \mu_1 > 0 \)

**Output:** \( x, f \in \mathbb{R}^N \)

1: \( x \leftarrow y \)
2: \( d_0 \leftarrow 0 \)
3: \( d_1 \leftarrow 0 \)
4: \( \overline{y} \leftarrow A^{-1} By \)
5: \( x \leftarrow A^{-1} B x \)
6: **repeat**
7: \( v_0 \leftarrow \text{soft}(\text{tvd}(x + d_0, \lambda_1/\mu_0), \lambda_0/\mu_0) \)
8: \( v_1 \leftarrow \text{proj}(x + d_1; \overline{y}, r) \)
9: \( b \leftarrow (v_0 - d_0) + (\mu_0/\mu_1) B^T A^{-1} (v_1 - d_1) \)
10: \( x \leftarrow b - \mu_1 B^T (\mu_0 AA^T + \mu_1 BB^T)^{-1} Bb \)
11: \( x \leftarrow A^{-1} B x \)
12: \( d_0 \leftarrow d_0 - v_1 + x \)
13: \( d_1 \leftarrow d_1 - v_1 + \overline{x} \)
14: **until** convergence
15: \( f \leftarrow (y - x) - A^{-1} B (y - x) \)
16: **return** \( x, f \)

**VIII. Conclusion**

Sparsity-based signal processing methods are now highly developed, yet LTI filtering is still predominant for noise reduction for 1-D signals in practice. This paper presents a convex optimization approach for combining low-pass filtering and sparsity-based denoising so as to more effectively filter (denoise) a wider class of signals. The LPF/TVD algorithm (Alg. 1) assumes that the signal of interest is composed of a low-frequency component and a sparse-derivative component. The LPF/CSD algorithm (Alg. 2) assumes the second component is both sparse and has a sparse derivative. Both algorithms draw on the computational efficiency of fast solvers for banded linear systems, available for example as part of LAPACK [5].

The problem formulation and algorithms described in this paper can be extended in several ways. As in [51], enhanced sparsity can be achieved by replacing the \( \ell_1 \) norm by regularizers that promote sparsity more strongly, such as the \( \ell_p \) pseudo-norm (0 < p < 1), or by reweighted \( \ell_1 \) [14], greedy \( \ell_1 \)
[42], etc. In addition, in place of total variation, higher-order or generalized total variation can be used [40]. The use of LTI filters other than a low-pass filter may also be useful; for example, the use of band-pass or notch filters may be appropriate for specific signals.

REFERENCES


SUPPLEMENTARY MATERIAL

The LPF/TVD algorithm is implemented by the MATLAB program lpftvd.
The LPF/CSD algorithm is implemented by the MATLAB program lpfcscd.
The C-LPF/CSD algorithm is implemented by the MATLAB program lpfcscd_constr.
MATLAB program for simultaneous low-pass filtering and total variation denoising (LPF/TVD).

```matlab
function [x, f, cost] = lpftvd(y, d, fc, lam, Nit)
% [x, f, cost] = lpftvd(y, d, fc, lam, Nit)
% Simultaneous low-pass filtering (LPF) and total variation denoising (TVD).
%
% INPUT
% y - noisy data
% d - degree of filter is 2d (d : small positive integer 1, 2, 3..)  
% fc - cut-off frequency (normalized frequency, 0 < fc < 0.5)  
% lam - regularization parameter (proportional to noise sigma)  
% Nit - number of iterations  
%
% OUTPUT
% x - TV component  
% f - LPF component  
% cost - cost function history
%
% The algorithm is based on majorization-minimization and fast  
% solvers for banded linear systems.
%
% Ivan Selesnick, NYU-Poly, 2012

y = y(:);  % convert to column
cost = zeros(1,Nit);  % cost function history
N = length(y);

[A, B, B1] = ABfilt(d, fc, N);  % banded filter matrices [sparse]
H = @(x) A\(B*x);  % H : high-pass filter
S = @(x) [0; cumsum(x)];  % S : cumulative sum
Id = @(x) x(d+1:N-d);  
AAT = A*A';  % A *A' : banded matrix [sparse]
b = (1/lam) * B1'*(AAT\(B*y));  % b : (1/lam) * S * H'*H*y
u = -diff(y);  % initialization
for k = 1:Nit
    Lam = spdiags(abs(u), 0, N-1, N-1);  % Lam : diag(abs(u)) [sparse]
    F = lam*AAT + B1*Lam*B1';  % F : banded matrix [sparse]
    u = Lam * (b - (B1'*(F\(B1*(Lam*b)))));  % update
    cost(k) = 0.5*sum(abs(H(y-S(u))).^2) + lam * sum(abs(u));
end
x = S(u);  % x : tv (step) component
bn = nan + zeros(d,1);  % bn : nan’s to extend f to length N
f = y - x - [bn; H(y-x); bn];  % f : low-pass component
```

% convert to column
% cost function history

% banded filter matrices [sparse]
% high-pass filter
% cumulative sum
% banded matrix [sparse]
% (1/lam) * S * H'*H*y
% initialization

% diag(abs(u)) [sparse]
% banded matrix [sparse]
% update
% cost function history

% step component
% nan’s to extend f to length N
% low-pass component
MATLAB program for simultaneous low-pass filtering and sparsity-based denoising (LPF/CSD).

```matlab
function [x,f,cost] = lpfcsd(y, d, fc, lam0, lam1, Nit, mu)
% [x, f, cost] = lpfcsd(y, d, fc, lam0, lam1, Nit, mu)
% Simultaneous low-pass filtering and compound sparsity denoising
% INPUT
% y - noisy data
% d - degree of filter is 2d (use d = 1, 2, or 3)
% fc - cut-off frequency (normalized frequency, 0 < fc < 0.5)
% lam0, lam1 - regularization parameter for x and diff(x)
% Nit - number of iterations
% mu - Augmented Lagrangian parameter
% OUTPUT
% x - TV component
% f - LPF component
% cost - cost function history

y = y(:); % convert to column vector
N = length(y);
bn = nan + zeros(d,1); % bn : nan's to extend f to length N
[A, B] = ABfilt(d, fc, N);
Id = @(x) x(d+1:N-d);
H = @(x) A\(B*x); % H: high-pass filter
G = mu*(A*A') + B*B'; % G: banded matrix [sparse]
v = zeros(N,1); % initializations
d = zeros(N,1);
b = (1/mu) * B'*((A*A')\(B*y));

for k = 1:Nit
    g = b + v - d;
    x = g - B'*((G \ (B*g)));
    v = tvd(x + d, N, lam1/mu); % TV denoising
    v = soft(v, lam0/mu);
    v = v(:);
    d = d + x - v;
    cost(k) = lam0 * sum(abs(x)) + lam1 * sum(abs(diff(x))) + 0.5 * sum(abs(H(x-y)).^2);
end
f = y - x - [bn; H(y-x); bn]; % f : low-pass component
```
MATLAB program for the constrained form of simultaneous low-pass filtering and sparsity denoising (C-LPF/CSD).

```matlab
function [x, f, cost, constr] = lpfcsd_constr(y, d, fc, r, lam0, lam1, Nit, mu0, mu1)
% [x, f, cost, constr] = lpfcsd_constr(y, d, fc, r, lam0, lam1, Nit, mu0, mu1)
% Simultaneous low-pass filtering and compound sparsity denoising
% constrained formulation
% INPUT
% y - noisy data
% d - degree of filter is 2d (use d = 1, 2, or 3)
% fc - cut-off frequency (normalized frequency, 0 < fc < 0.5)
% r - constraint
% Nit - number of iterations
% mu0, mu1 - Augmented Lagrangian parameters
% OUTPUT
% x - sparse / sparse-derivative component
% f - LPF component
% cost - cost function history
% constr - constraint function history
y = y(:); % convert to column vector
cost = zeros(1, Nit); % cost function history
constr = zeros(1, Nit); % constraint function history
N = length(y);

[A, B] = ABfilt(d, fc, N);
Id = @(x) x(d+1:N-d);
H = @(x) A' \ (B*x); % H: high-pass filter
Id = @(x) x(d+1:N-d);
F = mu0*(A*A') + mu1*(B*B'); % F: banded matrix [sparse]
x = zeros(N,1);
d0 = zeros(N,1);
d1 = zeros(N-2*d,1);
Hy = H(y);
Hx = H(x);
for k = 1:Nit
    v0 = tvd(x + d0, N, lam1/mu0); % TV denoising
    v0 = soft(v0, lam0/mu0);
    v0 = v0(:);
    v1 = projball(Hx + d1, Hy, r);
    b = (v0 - d0) + (mu1/mu0) * B'*A' \ (v1-d1)); % banded system solve (A')
    x = b - mu1*B'*F \ (B*b)); % banded system solve (F)
    Hx = H(x);
    d0 = d0 - (v0 - x);
    d1 = d1 - (v1 - Hx);
    cost(k) = lam0 * sum(abs(x)) + lam1 * sum(abs(diff(x))); % cost function history
    constr(k) = sqrt(sum(abs(H(x-y)).^2)); % constraint function history
end
bn = nan + zeros(d,1); % bn : nan’s to extend f to length N
f = y - x - [bn; H(y-x); bn]; % f : low-pass component
```

% SIMULTANEOUS LOW-PASS FILTERING AND COMPOUND SPARSITY DENOISING
% CONSTRAINED FORMULATION

% INPUT
% y - noisy data
% d - degree of filter is 2d (use d = 1, 2, or 3)
% fc - cut-off frequency (normalized frequency, 0 < fc < 0.5)
% r - constraint
% Nit - number of iterations
% mu0, mu1 - Augmented Lagrangian parameters
% OUTPUT
% x - sparse / sparse-derivative component
% f - LPF component
% cost - cost function history
% constr - constraint function history
function [A, B, B1] = ABfilt(d, fc, N)
% [A, B] = ABfilt(d, fc, N)
% Banded matrices for zero-phase high-pass filter.
% The matrices are 'sparse' data type in MATLAB.
% INPUT
% d : degree of filter is 2d
% fc : cut-off frequency (normalized frequency, 0 < fc < 0.5)
% N : length of signal
% [A, B, B1] = ABfilt() also returns B1 such that B = B1 * D
% where D is the first-order different matrix (diff())

b1 = [1 -1];
for i = 1:d-1
    b1 = conv(b1, [-1 2 -1]);
end
b = conv(b1, [-1 1]);

omc = 2*pi*fc;
t = ((1-cos(omc))/(1+cos(omc)))ˆd;

a = 1;
for i = 1:d
    a = conv(a,[1 2 1]);
end
a = b + t*a;
A = spdiags( a(ones(N-2*d,1), :) , -d:d, N-2*d, N-2*d); % A: Symmetric banded matrix

B1 = spdiags(b1(ones(N,1), :) , 0:2*d-1, N-2*d, N-1); % B1: banded matrix

e = ones(N, 1);
D = spdiags([-e e] , 0:1, N-1, N);
B = B1 * D; % B: banded matrix

% Verify that B = B1*D
x = randn(N,1);
err = B*x - B1*diff(x);
if max(abs(err(:))) > 1e-5
    disp('Error: B not equal to B1*D')
end