Polynomial Smoothing of Time Series with Additive Step Discontinuities

Ivan W. Selesnick, Stephen Arnold, and Venkata R. Dantham

***Abstract*—This paper addresses the problem of estimating simultaneously a local polynomial signal and an approximately piecewise constant signal from a noisy additive mixture. The approach developed in this paper synthesizes the total variation filter and least-square polynomial signal smoothing into a unified problem formulation. The method is based on formulating an ℓ_1-norm regularized inverse problem. A computationally efficient algorithm, based on variable splitting and the alternating direction method of multipliers (ADMM), is presented. Algorithms are derived for both unconstrained and constrained formulations. The method is illustrated on experimental data involving the detection of nano-particles with applications to real-time virus detection using a whispering-gallery mode detector.**

I. Introduction

This paper addresses the problem of estimating simultaneously a local polynomial signal and an approximately piecewise constant signal from a noisy additive mixture.[1] Specifically, it is assumed that the given discrete-time data $y(n)$ can be well modeled as

$$y(n) = s(n) + x(n) + w(n), \quad n = 1, \ldots, N \qquad (1)$$

where:

1) The signal $s(n)$ is well approximated, on each interval of L samples, by a polynomial of degree d, with $d \ll L$.
2) $x(n)$ is approximately piecewise constant.
3) $w(n)$ is stationary white Gaussian noise.

Data of this form arises when a discrete event phenomenon is observed in the presence of a comparatively slow varying signal. For example, certain systems for the real-time optical detection of virus particles are based on the discrete change in resonance frequency of a microspherical cavity produced by the adsorption of individual nanoparticles [37]. As will be illustrated in Section VII, the signal measured by such a biosensor exhibits discrete steps, slow variation, and noise.

This paper develops a method to estimate both the slow varying signal $s(n)$ and the discrete step signal $x(n)$ from the measured signal $y(n)$. The developed method is a synthesis of two well-known but distinct methodologies: (1) least square polynomial smoothing, and (2) total variation filtering.

Contact author: I. Selesnick, email: selesi@poly.edu, phone: 718 260-3416, fax: 718 260-3906.

I. Selesnick is with the Department of Electrical and Computer Engineering, S. Arnold and V. R. Dantham are with the MicroParticle PhotoPhysics Lab. All authors are with the Polytechnic Institute of New York University, 6 Metrotech Center, Brooklyn, NY 11201.

I.S. acknowledges support from NSF under Grant No. CCF-1018020.

S.A. and V.R.D. would like to thank the NSF for supporting their contribution to this work under Grant No. CBET 0933531.

[1]MATLAB software related to this paper is available at http://eeweb.poly.edu/iselesni/patv/

Least square polynomial smoothing comprises of fitting a low-order polynomial to noisy data. When the polynomial fitting is performed in a sliding window, and the center value of the approximating polynomial is used as a signal sample estimate, then this is equivalent to low-pass FIR filtering and is known as the Savitzky-Golay filter [27], [30]–[32], [34, Section 11.4.1.7]. The topic of least square polynomial fitting of noisy data is well studied in linear algebra and numerical methods [21]. Polynomial smoothing filters are suitable for signals that are locally well approximated by polynomials [18].

On the other hand, total variation filtering is suitable when the underlying signal to be estimated is approximately piecewise constant. Total variation (TV) filtering, introduced by Rudin, Osher, and Fatemi [10], [29], is defined through the minimization of a non-quadratic cost function; therefore, the output of the TV filter is a nonlinear function of the data. Furthermore, the output of the TV filter can not be found in closed form as a function of the data; it can be obtained only as the result a numerical algorithm. Many algorithms have been developed for TV filtering with different convergence properties and computational requirements. It should be noted that the TV cost function is convex; consequently, algorithms for TV filtering can have guaranteed convergence. The algorithm by Chambolle [9] [33] is notable for its computational and algorithmic simplicity and its proven convergence properties; however, it converges slowly for some problems. At this time, TV filtering can be performed fast and efficiently (for example [39]). Recently, the method of 'alternating direction method of multipliers' (ADMM) has been used to develop fast algorithms for TV filtering and a variety of related signal restoration problems [1], [6], [13], [17], [39].

The problem of smoothing (filtering/denoising) a noisy observation of a piecewise smooth (or piecewise polynomial) function has been well studied and numerous approaches have been proposed. Methods for local polynomial approximation [18] have been significantly enhanced recently to account for discontinuities as described in [22]. In addition, several wavelet-based denoising methods have been developed specifically so as to enhance the denoising quality near edges; for example, ENO (essentially non-oscillatory) wavelets [11], wavelet footprints [15], interval wavelets [26], hidden Markov models [14], time-scale signal denoising [7], and wedgeprints for images [38]. The combined use of wavelets and total variation also leads to excellent results for filtering piecewise smooth signals [12], [16]. However, we note that although the problem addressed in this paper is simlar to filtering general piecewise smooth functions, there is an important difference. In this paper it is assumed that the signal of interest consists

of the sum of a smooth (low-pass) signal and a piecewise constant signal. In this case, the behavior of the signal to the left and to the right of each step discontinuity is essentially the same (i.e., the derivatives match). In contrast, a general piecewise smooth signal is generally understood to have different behavior on the two sides of a discontinuity and hence it is desirable when filtering such signals to avoid averaging across discontinuities. In contrast, for the signal model considered here, it is advantageous to exploit the common behavior on either side of each discontinuity.

This paper shows how to simultaneously accomplish both polynomial smoothing and total variation filtering, and achieves, as a byproduct, the separation of noisy data into a local polynomial signal and an approximately piecewise constant signal. The method described in this paper is based on the minimization of a suitably defined convex non-differentiable cost function. To minimize this cost function, variable splitting and ADMM are used to develop an efficient algorithm with robust convergence properties.

Section II introduces notation by way of reviewing the least square polynomial approximation problem and total variation filtering. Section III formulates the problem of simultaneous least square polynomial approximation and total variation filtering. In Section III, for clarity, the entire signal duration of N samples is modeled as a single polynomial. Section IV formulates the problem of simultaneous polynomial smoothing and total variation using a sliding window (analogous to the Savitzky-Golay filter). Section V addresses the constrained formulation. Section VI describes the use of the ℓ_p quasi-norm with $0 < p < 1$ so as to enhance the results obtained in the preceding sections. Section VII illustrates the method on experimental data produced by a whispering gallery mode nano-particle detector.

II. Preliminaries

This section reviews least square polynomial approximation (PA) and total variation (TV) filtering. The problem formulations and algorithms in the subsequent sections build upon these two methods.

A. Least Square Polynomial Approximation

To establish notation, consider first the problem of smoothing noisy data

$$y(n) = s(n) + w(n), \quad n = 1, \ldots, N \tag{2}$$

where $s(n)$ is well approximated by a polynomial of degree d, and $w(n)$ is stationary white Gaussian noise. The signal $s(n)$ can be estimated by minimizing the cost function:

$$\min_{a_0,\ldots,a_d} \sum_{n=1}^{N} |y(n) - p(n)|^2 \tag{3a}$$

where

$$p(n) = a_0 + a_1\, n + \cdots + a_d\, n^d \tag{3b}$$

with $d \ll N$. The minimization problem (3) can be written as

$$\min_{\mathbf{a}} \|\mathbf{y} - \mathbf{G}\,\mathbf{a}\|_2^2 \tag{4}$$

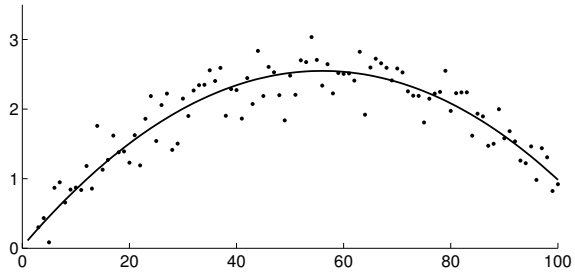


Fig. 1. Noisy data and least square polynomial approximation.

where $\mathbf{G}$ is a Vandermonde matrix and $\mathbf{a}$ is a vector of the polynomial coefficients,

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & & 2^d \\ 1 & 3 & & 3^d \\ \vdots & & & \vdots \\ 1 & N & \cdots & N^d \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} a_0 \\ \vdots \\ a_d \end{bmatrix} \tag{5}$$

The matrix $\mathbf{G}$ is of size $N \times (d+1)$. The solution to (4) is given by $\mathbf{a} = (\mathbf{G}^t\mathbf{G})^{-1}\mathbf{G}^t\mathbf{y}$ and the estimate of signal $\mathbf{s}$ is given by

$$\hat{\mathbf{s}} = \mathbf{G}\mathbf{a} = \mathbf{G}(\mathbf{G}^t\mathbf{G})^{-1}\mathbf{G}^t\mathbf{y}.$$

The residual is given by $\hat{\mathbf{w}} = \mathbf{y} - \hat{\mathbf{s}}$, which can be written as $\hat{\mathbf{w}} = \mathbf{H}\mathbf{y}$ where

$$\mathbf{H} := \mathbf{I} - \mathbf{G}(\mathbf{G}^t\mathbf{G})^{-1}\mathbf{G}^t. \tag{6}$$

Figure 1 shows an example of least square polynomial approximation, where $N = 100$ and $d = 2$.

B. Total Variation Filter

The total variation (TV) of the discrete signal $x(n)$ is defined the sum of absolute values of its first-order difference. Specifically, the total variation [10] of an N-point discrete-time signal $x(n)$, $1 \le n \le N$, is given by

$$\mathrm{TV}(\mathbf{x}) := \sum_{n=2}^{N} |x(n) - x(n-1)|.$$

Note that the TV of the signal $\mathbf{x}$ can be written as

$$\mathrm{TV}(\mathbf{x}) = \|\mathbf{D}\mathbf{x}\|_1$$

where

$$\mathbf{D} = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & & \ddots & \\ & & & -1 & 1 \end{bmatrix} \tag{7}$$

is a matrix of size $(N-1) \times N$. The notation $\|\cdot\|_1$ denotes the ℓ_1 norm.

As a measure of how much a signal fluctuates, $\mathrm{TV}(\mathbf{x})$ is to be contrasted with $\|\mathbf{D}\mathbf{x}\|_2^2$, the sum of squares of $\mathbf{D}\mathbf{x}$. The quantity $\|\mathbf{D}\mathbf{x}\|_2^2$ is more sensitive to the large values of $\mathbf{D}\mathbf{x}$ than is $\mathrm{TV}(\mathbf{x})$. Minimizing $\|\mathbf{D}\mathbf{x}\|_2^2$ does not produce a signal with sparse first-order difference, because the small values of

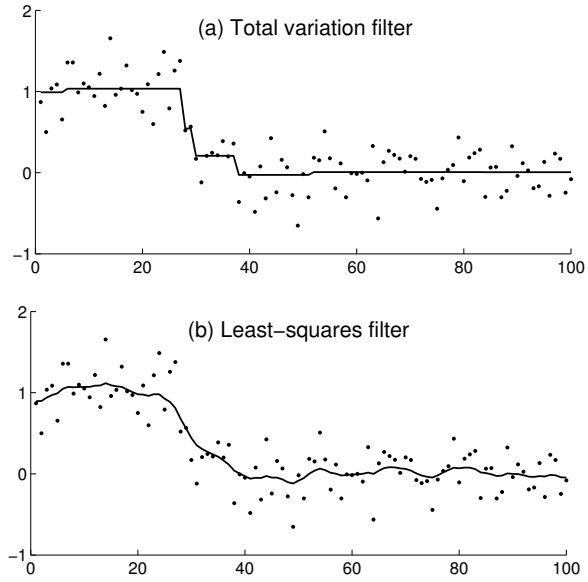


Fig. 2. (a) Noisy data and TV filtering obtained by minimization of (8). (b) Noisy data and least squares filtering obtained by minimization of (9).

$\mathbf{Dx}$ are relatively unimportant to $\|\mathbf{Dx}\|_2^2$. To obtain a signal $\mathbf{x}$ for which $\mathbf{Dx}$ is sparse, it is more effective to use the ℓ_1 norm. This is the basis of TV filtering, which is defined as follows.

Given noisy data $\mathbf{y}$, the output of the TV filter is defined as the solution $\mathbf{x}$ to the minimization problem:

$$\underset{\mathbf{x}}{\operatorname{argmin}} \ \lambda\|\mathbf{Dx}\|_1 + \|\mathbf{y}-\mathbf{x}\|_2^2. \tag{8}$$

To illustrate the TV filter, noisy data obtained by adding stationary white Gaussian noise to a step function is illustrated in Fig. 2a (dots). The output of the TV filter is illustrated also in Fig. 2a (solid line). The output of the TV filter is approximately a step function. (A perfect step function is not expected, TV filtering being a non-parametric signal estimation method; i.e., the output of the TV filter is not parameterized by step-location and step-height, etc.) It is informative to compare the TV filter output in Fig. 2a with the signal obtained by minimizing the cost function:

$$\underset{\mathbf{x}}{\operatorname{argmin}} \ \lambda\|\mathbf{Dx}\|_2^2 + \|\mathbf{y}-\mathbf{x}\|_2^2, \tag{9}$$

which is based on least squares rather than total variation. The solution to (9) is illustrated in Fig. 2b. Note that the filtered signal in Fig. 2b neither has the sharp discontinuity of the TV filter output, nor does it suppress the noise as much. Therefore, in contrast to linear filtering, the TV filter produces output signals with sharper discontinuities.

In both (8) and (9), λ is a positive scalar parameter specified by the user so as to control the trade-off between noise reduction and signal distortion.

III. Simultaneous Polynomial Approximation and TV Filtering

As illustrated in Section II, least square polynomial approximation (PA) and TV filtering are well suited for the estimation of smooth signals and (approximately) piecewise constant signals respectively. In this section it is assumed that the observed data is a noisy additive mixture of these two types of signal. Specifically, it is assumed that the noisy data $y(n)$ has the form

$$y(n) = s(n) + x(n) + w(n), \quad n = 1, \ldots, N, \tag{10}$$

where $s(n)$ is a low-order polynomial, and $x(n)$ is approximately piecewise constant.

In order to simultaneously estimate $s(n)$ and $x(n)$, it is proposed that the polynomial coefficients $\mathbf{a}$ and the signal $\mathbf{x}$ be jointly found so as to solve the minimization problem:

$$\min_{\mathbf{a},\mathbf{x}} \ \lambda \sum_{n=2}^{N} |x(n)-x(n-1)| + \sum_{n=1}^{N} |y(n)-p(n)-x(n)|^2 \tag{11a}$$

where

$$p(n) = a_1\, n + \cdots + a_d\, n^d. \tag{11b}$$

We refer to (11) as the PATV problem (for Polynomial Approximation–Total Variation). The estimate $\hat{\mathbf{s}}$ will be given by $\hat{\mathbf{s}} = \mathbf{Ga}$. Note that the polynomial $p(n)$ in (11b) does not include a constant term a_0 as it does in (3b) because the constant is accounted for by the signal $x(n)$. If $p(n)$ in (11b) were to include the constant term a_0, then the solution to the minimization problem (11) would not be unique (unless a constraint is imposed upon $x(n)$ such as a zero-mean constraint).

The PATV problem (11) can be written as

$$\underset{\mathbf{a},\mathbf{x}}{\operatorname{argmin}} \ \lambda\|\mathbf{Dx}\|_1 + \|\mathbf{y}-\mathbf{Ga}-\mathbf{x}\|_2^2, \tag{12}$$

where the matrix $\mathbf{G}$ is as given in (5) but with the first column of all 1's omitted (corresponding to the constant term a_0 in (3b)). The optimization problem (12) combines (4) and (8). The minimization is performed with respect to $\mathbf{a}$ and $\mathbf{x}$ jointly. The cost function (12) can not be minimized in explicit form, so an iterative algorithm is developed below to obtain the optimal $\mathbf{a}$ and $\mathbf{x}$.

Note that if $\mathbf{x}$ is fixed and the minimization is performed with respect to the polynomial coefficients $\mathbf{a}$ only, then the problem (12) is the same as (4), applied to the signal $\mathbf{y}-\mathbf{x}$. We refer to the signal $\mathbf{y}-\mathbf{x}$ as the 'TV-compensated data'. The minimization problem (12) can be understood as finding a signal $\mathbf{x}$ having small total variation so that $\mathbf{y}-\mathbf{x}$ is approximately polynomial. That is, an approximately piecewise constant signal $\mathbf{x}$ is to be determined so that the compensated data $\mathbf{y}-\mathbf{x}$ is well approximated by polynomial of degree d.

The modeling of the given data in (10) as a sum of two components of distinct behavior is similar to the approach of morphological component analysis (MCA) [4], [35], [36]. In MCA, each component is modeled as having a sparse representation with respect to a known transform. On the other hand, the approach used here (12) uses sparsity for only one of the two components ($\mathbf{Dx}$ is modeled as sparse).

A. Algorithm PATV

In order to solve the minimization problem (12), note that the minimization with respect to $\mathbf{a}$ can be expressed explicitly:

$$\mathbf{a} = (\mathbf{G}^t\mathbf{G})^{-1}\mathbf{G}^t(\mathbf{y}-\mathbf{x}). \tag{13}$$

Substituting (13) into (12), the cost function in (12) can hence be written as

$$J(\mathbf{x}) = \lambda\|\mathbf{D}\mathbf{x}\|_1 + \|(\mathbf{I}-\mathbf{G}(\mathbf{G}^t\mathbf{G})^{-1}\mathbf{G}^t)(\mathbf{y}-\mathbf{x})\|_2^2, \tag{14}$$

so that (12) can be written as:

$$\underset{\mathbf{x}}{\operatorname{argmin}} \quad \lambda\|\mathbf{D}\mathbf{x}\|_1 + \|\mathbf{H}(\mathbf{y}-\mathbf{x})\|_2^2, \tag{15}$$

where $\mathbf{H}$ is given by (6).

It remains to minimize (15) with respect to $\mathbf{x}$. In order to solve this problem, variable splitting and ADMM can be utilized as it has been for other ℓ_1-norm regularized inverse problems, for example the algorithm SALSA [1]. The method of ADMM and its application for the derivation of optimization algorithms of this type is described in [6] and references therein.

The algorithm SALSA and related algorithms are based in part on decomposing the optimization problem of interest into simpler problems which can be more readily solved. The optimization problem can often be decomposed into simpler optimization problems by performing a suitable *variable splitting*. It is clear that (15) is equivalent to the following optimization problem

$$\begin{aligned} \underset{\mathbf{x},\mathbf{u}_0,\mathbf{u}_1}{\operatorname{argmin}} \quad & \lambda\|\mathbf{u}_0\|_1 + \|\mathbf{H}(\mathbf{y}-\mathbf{u}_1)\|_2^2 && \text{(16a)}\\ \text{such that} \quad & \mathbf{u}_0 = \mathbf{D}\mathbf{x} && \text{(16b)}\\ & \mathbf{u}_1 = \mathbf{x}. && \text{(16c)} \end{aligned}$$

Compared with (15), problem (16) appears more difficult: additional variables $\mathbf{u}_0$ and $\mathbf{u}_1$ have been introduced and the problem has constraints. However, using ADMM, problem (16) can be solved by an iterative algorithm in which each iteration consists of two optimization problems:

$$\mathbf{u}_0,\mathbf{u}_1 \leftarrow \underset{\mathbf{u}_0,\mathbf{u}_1}{\operatorname{argmin}} \begin{cases} \lambda\|\mathbf{u}_0\|_1 + \|\mathbf{H}(\mathbf{y}-\mathbf{u}_1)\|_2^2 \\ +\mu_0\|\mathbf{u}_0-\mathbf{D}\mathbf{x}-\mathbf{d}_0\|_2^2 \\ +\mu_1\|\mathbf{u}_1-\mathbf{x}-\mathbf{d}_1\|_2^2 \end{cases} \tag{17a}$$

$$\mathbf{x} \leftarrow \underset{\mathbf{x}}{\operatorname{argmin}} \begin{cases} \mu_0\|\mathbf{u}_0-\mathbf{D}\mathbf{x}-\mathbf{d}_0\|_2^2 \\ +\mu_1\|\mathbf{u}_1-\mathbf{x}-\mathbf{d}_1\|_2^2 \end{cases} \tag{17b}$$

$$\begin{aligned} & \mathbf{d}_0 \leftarrow \mathbf{d}_0 - (\mathbf{u}_0 - \mathbf{D}\mathbf{x}) && \text{(17c)}\\ & \mathbf{d}_1 \leftarrow \mathbf{d}_1 - (\mathbf{u}_1 - \mathbf{x}) && \text{(17d)}\\ & \text{Go to (17a)} && \text{(17e)} \end{aligned}$$

The set of update rules in (17) are repeated until some stopping criterion is satisfied. The iterative algorithm (17) alternates between minimization with respect to $(\mathbf{u}_0,\mathbf{u}_1)$ in (17a) and $\mathbf{x}$ in (17b). The parameters μ_0 and μ_1 are user specified positive scalar parameters; they do not affect the solution to which the algorithm converges, but they do affect the speed of convergence. The algorithm requires initializations for $\mathbf{d}_0$, $\mathbf{d}_1$, and $\mathbf{x}$. Vectors $\mathbf{d}_0$ and $\mathbf{d}_1$ can be initialized to zero-valued vectors of length $N-1$ and N respectively; and $\mathbf{x}$ can be initialized to $\mathbf{y}$. However, due to the properties of ADMM [6], convergence of the iteration to the optimal solution is ensured regardless of the initialization.

Note that $\mathbf{u}_0$ and $\mathbf{u}_1$ are decoupled in (17a), so (17a) can be written as:

$$\begin{aligned} \mathbf{u}_0 &\leftarrow \underset{\mathbf{u}_0}{\operatorname{argmin}}\ \lambda\|\mathbf{u}_0\|_1 + \mu_0\|\mathbf{u}_0-\mathbf{D}\mathbf{x}-\mathbf{d}_0\|_2^2 && \text{(18a)}\\ \mathbf{u}_1 &\leftarrow \underset{\mathbf{u}_1}{\operatorname{argmin}}\ \|\mathbf{H}(\mathbf{y}-\mathbf{u}_1)\|_2^2 + \mu_1\|\mathbf{u}_1-\mathbf{x}-\mathbf{d}_1\|_2^2. && \text{(18b)} \end{aligned}$$

Now, each of the three minimizations (17b), (18a), (18b) admits an explicit solution, namely:

$$\begin{aligned} \mathbf{u}_0 &\leftarrow \operatorname{soft}(\mathbf{D}\mathbf{x}+\mathbf{d}_0, 0.5\lambda/\mu_0) && \text{(19a)}\\ \mathbf{u}_1 &\leftarrow (\mathbf{H}^t\mathbf{H}+\mu_1\mathbf{I})^{-1}\left(\mathbf{H}^t\mathbf{H}\mathbf{y}+\mu_1(\mathbf{x}+\mathbf{d}_1)\right) && \text{(19b)}\\ \mathbf{x} &\leftarrow (\mu_0\mathbf{D}^t\mathbf{D}+\mu_1\mathbf{I})^{-1}(\mu_0\mathbf{D}^t(\mathbf{u}_0-\mathbf{d}_0)+\mu_1(\mathbf{u}_1-\mathbf{d}_1)) \end{aligned}$$

In (19a), the function $\operatorname{soft}(\mathbf{v},T)$ is the soft-threshold function with threshold T applied to each element of vector $\mathbf{v}$.

Note that (19b) calls for solving a large system of equations (of size N, the length of the signal $\mathbf{y}$). In order to avoid the high computational cost, the matrix inverse lemma can be used as follows:

$$\begin{aligned} (\mathbf{H}^t\mathbf{H}+\mu_1\mathbf{I})^{-1} &= (\mathbf{H}+\mu_1\mathbf{I})^{-1} && \text{(20a)}\\ &= ((1+\mu_1)\mathbf{I}-\mathbf{G}(\mathbf{G}^t\mathbf{G})^{-1}\mathbf{G}^t)^{-1} && \text{(20b)}\\ &= \frac{1}{1+\mu_1}\Big(\mathbf{I}+\frac{1}{\mu_1}\mathbf{G}\left(\mathbf{G}^t\mathbf{G}\right)^{-1}\mathbf{G}^t\Big). && \text{(20c)} \end{aligned}$$

In (20a), the identity $\mathbf{H}^t\mathbf{H}=\mathbf{H}$ has been used. To obtain (20c), the matrix inverse lemma has been used. Using (6), the matrix $(\mathbf{H}^t\mathbf{H}+\mu_1\mathbf{I})^{-1}(\mathbf{H}^t\mathbf{H})$ can also be simplified:

$$(\mathbf{H}^t\mathbf{H}+\mu_1\mathbf{I})^{-1}\mathbf{H}^t\mathbf{H} = \frac{1}{1+\mu_1}\mathbf{H} \tag{21}$$

Using (20c) and (21) gives:

$$\begin{aligned} &(\mathbf{H}^t\mathbf{H}+\mu_1\mathbf{I})^{-1}\left(\mathbf{H}^t\mathbf{H}\,\mathbf{y}+\mu_1\mathbf{b}\right)\\ &\quad= \frac{1}{1+\mu_1}\Big((\mathbf{y}+\mu_1\mathbf{b})+\mathbf{G}\left(\mathbf{G}^t\mathbf{G}\right)^{-1}\mathbf{G}^t\left(\mathbf{b}-\mathbf{y}\right)\Big) \end{aligned}$$

Therefore, algorithm (17) can be written as:

$$\begin{aligned} & \mathbf{u}_0 \leftarrow \operatorname{soft}(\mathbf{D}\mathbf{x}+\mathbf{d}_0, 0.5\lambda/\mu_0) && \text{(22a)}\\ & \mathbf{b} \leftarrow \mathbf{x}+\mathbf{d}_1 && \text{(22b)}\\ & \mathbf{u}_1 \leftarrow [(\mathbf{y}+\mu_1\mathbf{b})+\mathbf{G}\left(\mathbf{G}^t\mathbf{G}\right)^{-1}\mathbf{G}^t(\mathbf{b}-\mathbf{y})]/(1+\mu_1) && \text{(22c)}\\ & \mathbf{x} \leftarrow (\mu_0\mathbf{D}^t\mathbf{D}+\mu_1\mathbf{I})^{-1}(\mu_0\mathbf{D}^t(\mathbf{u}_0-\mathbf{d}_0)+\mu_1(\mathbf{u}_1-\mathbf{d}_1)) && \text{(22d)}\\ & \mathbf{d}_0 \leftarrow \mathbf{d}_0-(\mathbf{u}_0-\mathbf{D}\mathbf{x}) && \text{(22e)}\\ & \mathbf{d}_1 \leftarrow \mathbf{d}_1-(\mathbf{u}_1-\mathbf{x}) && \text{(22f)}\\ & \text{Go to (22a)} && \text{(22g)} \end{aligned}$$

Note that $\left(\mathbf{G}^t\mathbf{G}\right)^{-1}$ is of size $d\times d$. This is a very small matrix because the polynomial order d is much smaller than the signal length N. Therefore (22c) requires minimal computation. Also note that the matrix $\mu_0\mathbf{D}^t\mathbf{D}+\mu_1\mathbf{I}$ in (22d) is a sparse

tridiagonal matrix; therefore, the system of equations in step (22d) can be solved very efficiently. We refer to (22) as the PATV algorithm.[2]

To summarize, the PATV problem (12) can be solved using the iterative algorithm (22) to find $\mathbf{x}$, and then (13) to find $\mathbf{a}$. Following the derivation of SALSA, the derivation here utilized both variable splitting and ADMM to decompose the original minimization problem into an iterative sequence of simpler minimization problems each of which in turn admits a computationally efficient explicit solution.

B. Examples

An example is illustrated in Fig. 3. The 100-point noisy data sequence $\mathbf{y}$ is generated by adding a step discontinuity, a second order polynomial, and IID Gaussian noise. The TV component $\mathbf{x}$ (Fig. 3b) is obtained using 100 iterations of algorithm (22). The TV-compensated data (Fig. 3c, dots) is given by $\mathbf{y}-\mathbf{x}$. The polynomial component $\mathbf{p}=\mathbf{G}\mathbf{a}$ (Fig. 3c, solid line) is obtained by performing least square polynomial approximation on the TV-compensated data using (13). The total signal estimate $\mathbf{p}+\mathbf{x}$ (Fig. 3d, solid line) is the sum of the TV and polynomial components; it attempts to capture both the smooth trend and the discontinuity in the noisy data. The first-order difference function, $\mathbf{D}\mathbf{x}$, illustrated in Fig. 4, is sparse — consisting of only three nonzero values.

The role of λ in (12) can be understood by considering two cases: $\lambda \to 0$ and $\lambda \to \infty$. If λ is set to a relatively small value, then the solution will have the property that $\|\mathbf{D}\mathbf{x}\|_1$ will be relatively large; hence, the TV component $\mathbf{x}$ will be relatively noisy. For example, Fig. 5a illustrates the result of minimizing (12) using too small a value of λ. The estimated TV component $\mathbf{x}$ is noisy; hence, the total signal estimate $\mathbf{x}+\mathbf{G}\mathbf{a}$ is also noisy. Too small a value of λ leads to an under-regularized TV component and a noisy estimate of the signal as a whole.

On the other hand, if λ is set to a relatively large value, then the solution will have the property that $\|\mathbf{D}\mathbf{x}\|_1$ will be relatively small; hence, the TV component $\mathbf{x}$ will be nearly a flat line, $x(n)=c$ for some constant c. An example is illustrated in Fig. 5b. The TV component $\mathbf{x}$ is nearly a constant function; hence, the TV-compensated data $\mathbf{y}-\mathbf{x}$ contains the step discontinuity. Therefore, the polynomial component must try to fit the step discontinuity. Too large a value of λ leads to an over-regularized TV component and an under-estimate of the additive step discontinuities.

We do not propose here a method to set λ, but this example shows the qualitative change in the solution as λ is increased or decreased. The interactive selection of λ is facilitated when one recognizes from the solution if λ should be reduced or increased.

IV. Local Polynomial Approximation and TV Filtering

In the previous section, a single polynomial was used to model the smooth component of the entire N-point data

[2]A MATLAB program `patv` implementing algorithm (22) is listed as Program 1 in the supplementary material.

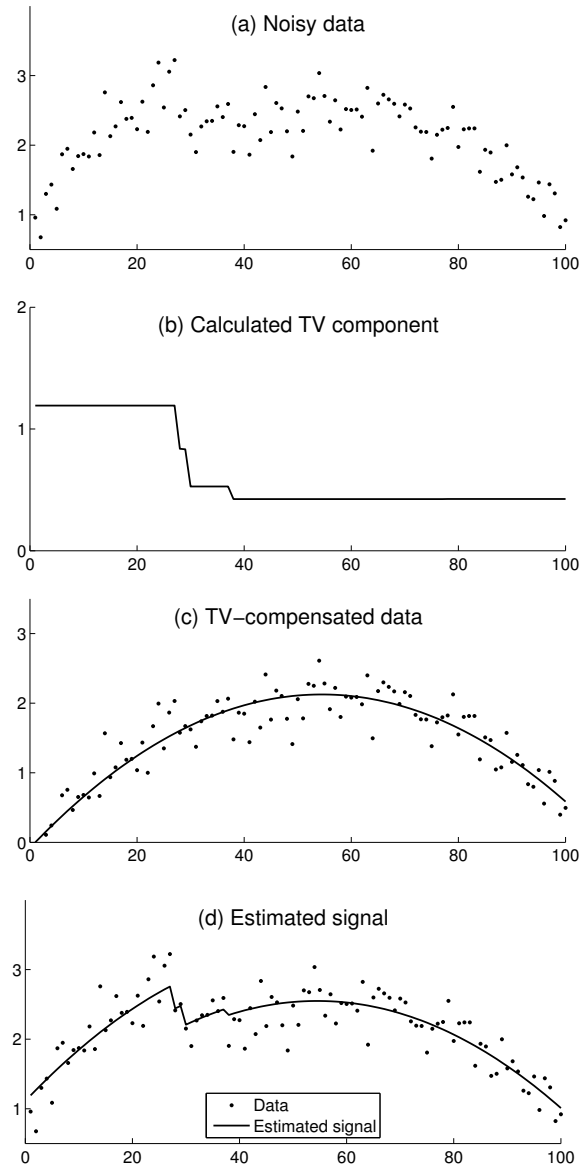


Fig. 3. (a) Noisy data $\mathbf{y}$. (b) Calculated TV component $\mathbf{x}$ obtained by minimization of (14). (c) TV-compensated data $\mathbf{y}-\mathbf{x}$ (dots) and least square polynomial approximation $\mathbf{p}$. (d) Noisy data $\mathbf{y}$ (dots) and total estimated signal $\mathbf{x}+\mathbf{p}$. The estimated signal captures both the additive step discontinuity and the polynomial behavior of the data.

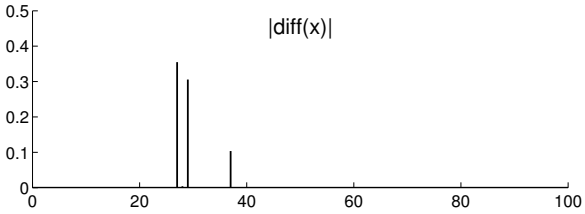


Fig. 4. The first-order difference function of the calculated TV component illustrated in Fig. 3b.

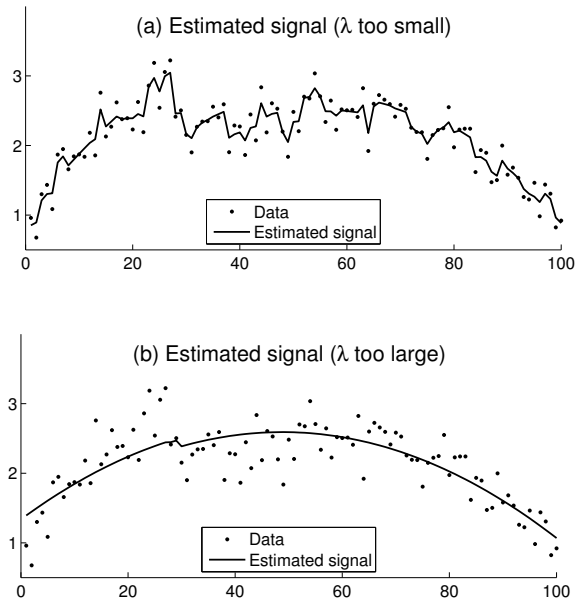


Fig. 5. (a) Effect of small λ (under-regularization of TV component). The calculated TV component is noisy. Hence, the estimated signal as a whole is noisy. (b) Effect of large λ (over-regularization of TV component). The calculated TV component is nearly constant. Hence, the estimated signal as a whole does not accurately capture the additive step discontinuity.

sequence. This is usually suitable only for relatively short signals. A long signal is usually approximated well by a low-order polynomial only over local blocks. Therefore, it is more common to perform polynomial approximation within a window of length L and to slide the L-point window along the data. If L is odd, then the value of the low-order polynomial at the center of the window can be used as an estimate of an underlying smooth signal at that point. Each data point $y(n_o)$ is approximated by a low-order polynomial fitted to the L data points centered at n_0. The method is parameterized by the window length L and the degree of the polynomial d. This is the idea of the Savitzky-Golay filter [27], [30], [32]. The procedure is known as 'polynomial smoothing' or local polynomial approximation.

This section develops an algorithm for simultaneous polynomial smoothing and total variation fitlering. It extends the method in the previous section so that low-order polynomial approximation is performed only over blocks (windows) instead of on the entire data set. In conventional polynomial smoothing, the window is shifted by a single point so that each L-point window has an overlap of $L-1$ points with the next window. In the following description, the overlap between consecutive windows can be less than that, in order to reduce the computational complexity.

Let the L-point vector $\mathbf{y}_i$ denote the i-th block of the N-point data sequence $\mathbf{y}$. The blocks $\mathbf{y}_i$ should be overlapping. At minimum, there should be no gaps between adjacent blocks, otherwise the algorithm developed here will involve the inverse of a singular matrix (in (32d) below). Let M denote the number of blocks. For each i, let the matrix $\mathbf{S}_i$ be defined such that $\mathbf{y}_i = \mathbf{S}_i\,\mathbf{y}$. The matrix $\mathbf{S}_i$ is of size $L \times N$, for $1 \le i \le M$. Note that each matrix $\mathbf{S}_i$ is a set of L consecutive rows of the $N \times N$ identity matrix. For example, if $N = 7$ and the second block consists of elements 3, 4, and 5, then

$$\mathbf{S}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{23}$$

Note that

$$\mathbf{S}_i\,\mathbf{S}_i^t = \mathbf{I}_L, \qquad \mathbf{S}_i^t\,\mathbf{S}_i = \mathrm{diag}(\mathbf{s}_i), \tag{24}$$

where $\mathrm{diag}(\mathbf{s})$ denotes the diagonal matrix diagonal with vector $\mathbf{s}$ on the diagonal. For example, with (23) one has $\mathbf{s}_2 = (0, 0, 1, 1, 1, 0, 0)$. Note that in the implementation described below, the matrices $\mathbf{S}_i$ are not constructed or stored. They are defined here only for notational convenience in deriving and describing the algorithm.

To perform simultaneous polynomial smoothing and total variation filtering, the following approach is taken: an approximately piecewise constant signal $\mathbf{x}$ must be determined so that the compensated data $\mathbf{y} - \mathbf{x}$ is locally well approximated by low-order polynomials. Specifically, a signal $\mathbf{x}$ having low total variation must be determined so that $\mathbf{y} - \mathbf{x}$ is well approximated by a polynomial of degree d over each L-point block i. This leads to the minimization of the cost function:

$$J(\mathbf{x}) = \lambda\|\mathbf{D}\mathbf{x}\|_1 + \sum_{i=1}^{M}\|\mathbf{H}\mathbf{S}_i(\mathbf{y}-\mathbf{x})\|_2^2 \tag{25}$$

analogous to (14), where $\mathbf{H}$ is given by (6). Each of the M terms in the sum corresponds to one of the M blocks. The i-th term in the sum measures the square error between the compensated data $\mathbf{y} - \mathbf{x}$ and its least square polynomial approximate over bock i, as in (14). We refer to the minimization of (25) as the LoPATV problem (for Local Polynomial Approximation–Total Variation).

Note that the polynomial approximation for each L-point block i requires a different constant term a_0. Therefore, the constant term can not be absorbed into the TV component $\mathbf{x}$ as it was in problem (11). Hence the Vandermonde matrix $\mathbf{G}$ in (25) must include the first column of all 1's as in (5). The matrix $\mathbf{G}$ in (25) is of size $L \times (d+1)$ where the polynomial degree d should be chosen sufficiently less than L to achieve the desired level of smoothing.

Note also that when $\mathbf{G}$ is chosen as described, then the minimizer of (25) is not unique because $J(\mathbf{x} + c) = J(\mathbf{x})$. Adding a constant to the signal $\mathbf{x}$ does not change the cost function. It will only change the polynomial approximation for each block $\mathbf{S}_i(\mathbf{y} - \mathbf{x})$ by an additive constant. Therefore, the minimizer of (25) is unique only up to an additive constant. The minimizer can be made unique by introducing an additional side constraint such as $x(1) = 0$, or a zero-mean constraint for $\mathbf{x}$. However, the additive constant does not alter the final signal estimate as whole. In addition, the algorithm developed below does not require it; hence, we do not introduce an additional side constraint here.

A. Algorithm LoPATV

As in Section III, variable splitting will be used in order to decompose the minimization problem (25) into simpler ones.

With variable splitting, problem (25) is equivalent to:

$$\underset{\mathbf{x},\mathbf{u}_i,i=0,\ldots,M}{\operatorname{argmin}} \quad \lambda\|\mathbf{u}_0\|_1 + \sum_{i=1}^{M}\|\mathbf{H}(\mathbf{S}_i\mathbf{y}-\mathbf{u}_i)\|_2^2 \tag{26a}$$

$$\text{such that} \quad \mathbf{u}_0 = \mathbf{D}\mathbf{x} \tag{26b}$$

$$\mathbf{u}_i = \mathbf{S}_i\mathbf{x}, \quad i=1,\ldots,M, \tag{26c}$$

where $\mathbf{H}$ is given in (6). The matrix $\mathbf{H}$ is of size $L \times L$. Using ADMM, problem (26) can be solved by the iterative algorithm:

$$\{\mathbf{u}_i\} \leftarrow \underset{\mathbf{u}_i,\,0\le i\le M}{\operatorname{argmin}} \begin{cases} \lambda\|\mathbf{u}_0\|_1 + \sum_{i=1}^{M}\|\mathbf{H}(\mathbf{S}_i\mathbf{y}-\mathbf{u}_i)\|_2^2 \\ +\,\mu_0\|\mathbf{u}_0 - \mathbf{D}\mathbf{x} - \mathbf{d}_0\|_2^2 \\ +\,\mu\sum_{i=1}^{M}\|\mathbf{u}_i - \mathbf{S}_i\mathbf{x} - \mathbf{d}_i\|_2^2 \end{cases} \tag{27a}$$

$$\mathbf{x} \leftarrow \underset{\mathbf{x}}{\operatorname{argmin}} \begin{cases} \mu_0\|\mathbf{u}_0 - \mathbf{D}\mathbf{x} - \mathbf{d}_0\|_2^2 \\ +\,\mu\sum_{i=1}^{M}\|\mathbf{u}_i - \mathbf{S}_i\mathbf{x} - \mathbf{d}_i\|_2^2 \end{cases} \tag{27b}$$

$$\mathbf{d}_0 \leftarrow \mathbf{d}_0 - (\mathbf{u}_0 - \mathbf{D}\mathbf{x}) \tag{27c}$$

$$\mathbf{d}_i \leftarrow \mathbf{d}_i - (\mathbf{u}_i - \mathbf{S}_i\mathbf{x}), \quad i=1,\ldots,M \tag{27d}$$

$$\text{Go to (27a)} \tag{27e}$$

Note that the $\mathbf{u}_i$ in (27a) are not coupled, so (27a) can be written as $M+1$ independent minimization problems:

$$\mathbf{u}_0 \leftarrow \underset{\mathbf{u}_0}{\operatorname{argmin}}\ \lambda\|\mathbf{u}_0\|_1 + \mu_0\|\mathbf{u}_0 - \mathbf{D}\mathbf{x} - \mathbf{d}_0\|_2^2 \tag{28}$$

$$\mathbf{u}_i \leftarrow \underset{\mathbf{u}_i}{\operatorname{argmin}}\ \|\mathbf{H}(\mathbf{S}_i\,\mathbf{y} - \mathbf{u}_i)\|_2^2 + \mu\|\mathbf{u}_i - \mathbf{S}_i\mathbf{x} - \mathbf{d}_i\|_2^2, \tag{29}$$

for $i=1,\ldots,M$. Each of the three minimizations (27b), (28), and (29) admits an explicit solution. The update of $\mathbf{u}_0$ in (28) is given by soft-thresholding. The update of $\mathbf{u}_i$ in (29) is given explicitly as

$$\mathbf{u}_i \leftarrow \left(\mathbf{H}^t\mathbf{H} + \mu\mathbf{I}\right)^{-1}\left(\mathbf{H}^t\mathbf{H}\mathbf{S}_i\mathbf{y} + \mu(\mathbf{S}_i\mathbf{x} + \mathbf{d}_i)\right), \tag{30}$$

for $i=1,\ldots,M$. The update (30) calls for the solution of an $L \times L$ system of equations. However, (30) is like (19b) so likewise (30) can be implemented as:

$$\mathbf{b} \leftarrow \mathbf{S}_i\,\mathbf{x} + \mathbf{d}_i \tag{31a}$$

$$\mathbf{u}_i \leftarrow [(\mathbf{S}_i\mathbf{y} + \mu_1\mathbf{b}) + \mathbf{G}\left(\mathbf{G}^t\mathbf{G}\right)^{-1}\mathbf{G}^t(\mathbf{b} - \mathbf{S}_i\mathbf{y})]/(1+\mu_1). \tag{31b}$$

The expression (31) involves the inverse of $\mathbf{G}^t\mathbf{G}$ which is of size only $(d+1)\times(d+1)$. Furthermore, $(\mathbf{G}^t\mathbf{G})$ needs to be computed only a single time and absorbed with $\mathbf{G}^t$ in (31), so it does not represent any real computational cost.

With explicit expressions for the minimization subproblems, the iterative algorithm (27) can be expressed as:

$$\mathbf{u}_0 \leftarrow \operatorname{soft}(\mathbf{D}\mathbf{x} + \mathbf{d}_0,\, 0.5\,\lambda/\mu_0) \tag{32a}$$

$$\mathbf{b} \leftarrow \mathbf{S}_i\,\mathbf{x} + \mathbf{d}_i \tag{32b}$$

$$\mathbf{u}_i \leftarrow [\mathbf{G}\left(\mathbf{G}^t\mathbf{G}\right)^{-1}\mathbf{G}^t(\mathbf{b} - \mathbf{S}_i\mathbf{y}) + (\mathbf{S}_i\mathbf{y} + \mu_1\mathbf{b})]/(1+\mu_1),\ i=1,\ldots,M \tag{32c}$$

$$\mathbf{x} \leftarrow \left(\mu_0\mathbf{D}^t\mathbf{D} + \mu\textstyle\sum_{i=1}^{M}\mathbf{S}_i^t\mathbf{S}_i\right)^{-1}\left(\mu_0\mathbf{D}^t(\mathbf{u}_0 - \mathbf{d}_0) + \mu\textstyle\sum_{i=1}^{M}\mathbf{S}_i^t(\mathbf{u}_i - \mathbf{d}_i)\right) \tag{32d}$$

$$\mathbf{d}_0 \leftarrow \mathbf{d}_0 - (\mathbf{u}_0 - \mathbf{D}\mathbf{x}) \tag{32e}$$

$$\mathbf{d}_i \leftarrow \mathbf{d}_i - (\mathbf{u}_i - \mathbf{S}_i\mathbf{x}), \quad i=1,\ldots,M \tag{32f}$$

$$\text{Go to (32a)} \tag{32g}$$

We refer to (32) as the LoPATV algorithm.[3] Note that the matrix $\sum_{i=1}^{M}\mathbf{S}_i^t\mathbf{S}_i$ in (32d) is diagonal, so as in (22d) the system matrix in (32d) is a sparse tridiagonal matrix, and hence the system (32d) can be solved very efficiently using a sparse system solver.

The coefficients $\mathbf{a}_i$ representing the polynomial approximation for block i are given by

$$\mathbf{a}_i = \left(\mathbf{G}^t\mathbf{G}\right)^{-1}\mathbf{G}^t\mathbf{S}_i(\mathbf{y} - \mathbf{x}),$$

where $\mathbf{S}_i(\mathbf{x}-\mathbf{y})$ represents the i-th block of the TV-compensated data. Hence $\mathbf{p}_i = \mathbf{G}\mathbf{a}_i$ is the polynomial approximation of the i-th block of $\mathbf{y}-\mathbf{x}$. Therefore, the problem formulation here results in overlapping signal estimates when the blocks are overlapping. To obtain a single estimate of the local polynomial signal the $\mathbf{p}_i$ can each be multiplied by a tapered window $\mathbf{w}$ and added, as expressed by:

$$\hat{\mathbf{s}} = \sum_{i=1}^{M}\mathbf{S}_i^t(\mathbf{w}\odot\mathbf{p}_i)./\sum_{i=1}^{M}\mathbf{S}_i^t\mathbf{w}. \tag{33}$$

In (33), $\odot$ denotes point-wise multiplication of two equal-length vectors and $./$ denotes point-wise division. The denominator in (33) is necessary for correct normalization. (For example, if all $\mathbf{p}_i = \mathbf{1}$, then we should have $\hat{\mathbf{s}} = \mathbf{1}$ regardless of window $\mathbf{w}$.) (Note that $\mathbf{S}_i^t\mathbf{v}$ places the L-point vector $\mathbf{v}$ into block i of an N-point signal.)

An example of simultaneous local polynomial approximation and TV filtering is illustrated in Fig. 6. The 300-point noisy data is generated by adding two step discontinuities, a sinusoidal signal, and IID Gaussian noise. In this example, we have used a block length $L=50$ with an overlap of 40 points, and polynomial degree $d=2$. Fig. 6a illustrates the noisy data $\mathbf{y}$. The calculated TV component $\mathbf{x}$ (Fig. 6b) is obtained using 200 iterations of the LoPATV algorithm (32). Fig. 6c illustrates the TV-compensated data $\mathbf{y}-\mathbf{x}$ (dots). We have used in (33) the L-point hamming window to combine the block-wise polynomial approximation $\mathbf{p}_i$ into a single local polynomial signal $\hat{\mathbf{s}}$. The total signal estimate (Fig. 6) captures the two discontinuities as well as the smooth (local polynomial) behavior of the signal.

[3] A MATLAB program `lopatv` implementing algorithm (32) is listed as Program 2 in the supplementary material.

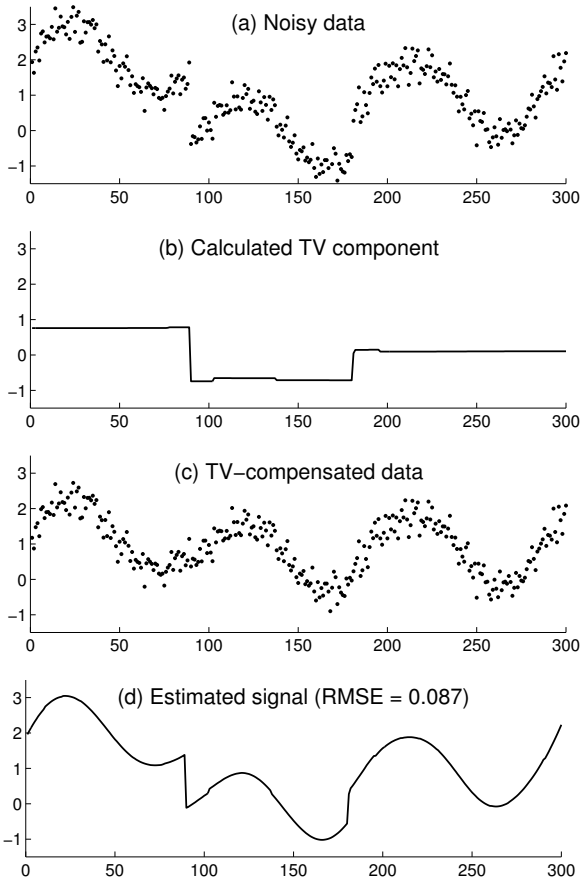


Fig. 6. (a) Noisy data $\mathbf{y}$. (b) Calculated TV component $\mathbf{x}$ obtained by minimization of (25). (c) TV-compensated data $\mathbf{y} - \mathbf{x}$. (d) Estimated signal obtained using LoPATV algorithm.

B. Comparisons

It was noted in the Introduction that the problem considered in this paper differs from the problem of denoising a general piecewise smooth signal. For a general piecewise smooth signal, the behavior to the left and right of a discontinuity are independent; hence an algorithm for denoising such signals should avoid smoothing across these points. Consequently, such algorithms are not expected to achieve the same level of smoothing without distorting the discontinuities.

For the noisy data shown in Fig. 6a, Fig. 7 shows the results of three denoising algorithms intended for general piecewise smooth signals: LPA-ICI [22], wavelet-HMT [14], and wavelet-TV. The LPA-ICI algorithm uses local polynomial approximation over centered and left- or right-facing windows, the widths of which are determined using the 'intersection of confidence intervals' (ICI) rule. For LPA-ICI, we set the polynomial degree to 2 as we did for LoPATV, and we used the public software provided by the authors.[4] The wavelet-HMT algorithm uses a hidden Markov tree (HMT) model. We symmetrically extended the noisy data up to length 512 (power of 2) and used the mid-phase (least asymmetric) orthonormal Daubechies wavelet filter of length 8. We used the public

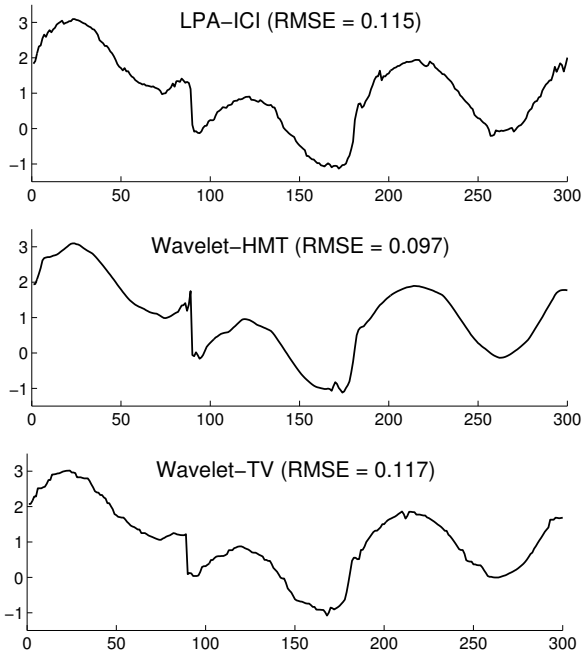


Fig. 7. Denoising using LPA-ICI, wavelet-HMT, and wavelet-TV algorithms on the noisy data shown in Fig. 6a. Compare with Fig. 6d.

software provided by the authors.[5] The wavelet-TV algorithm employed here, similar to [16], regularizes a weighted sum of the ℓ_1 norm of the wavelet coefficients and the total variation of the estimated signal. For wavelet-TV, we used the same wavelet transform as we used for the wavelet-HMT algorithm.

As shown in Fig. 7, denoising algorithms intended for general piecewise smooth signal can achieve substantial smoothing without overly distorting discontinuities. However, the LoPATV algorithm, which exploits the continuity of derivatives at additive step discontinuities, can provide further improvement for signals of this type.

V. Constrained Formulation

The effect of the regularization parameter λ was illustrated in Figs. 3 and 5. Clearly, in order to obtain a satisfactory result, it is important that λ be appropriately set. However, in practice it is often not clear in advance how to set λ. For this reason, an alternate problem formulation is of interest, namely the 'constrained' formulation. Note that in the cost function (15), both $\|\mathbf{D}\mathbf{x}\|_1$ and $\|\mathbf{H}(\mathbf{y} - \mathbf{x})\|_2^2$ are to be minimized, where λ specifies a relative weighting of the two terms. Instead of minimizing a weighted sum of the two terms as in (15), it is possible to minimize one of the two terms subject to a constraint on the other. Specifically, consider in place of (15) the constrained optimization problem:

$$\underset{\mathbf{x}}{\operatorname{argmin}} \quad \|\mathbf{D}\mathbf{x}\|_1 \tag{34a}$$

$$\text{such that} \quad \|\mathbf{H}(\mathbf{y} - \mathbf{x})\|_2 \leq r, \tag{34b}$$

where r is a positive scalar to be specified by the user. Like λ in (15), the parameter r controls the trade-off between the

[4] http://www.cs.tut.fi/~lasip/1D

[5] http://dsp.rice.edu/software/wavelet-domain-hidden-markov-models

two terms. While both (15) and (34) require that a scalar parameter be specified, the important advantage of (34) is that the parameter r can be set based on the variance of the additive noise in (10). To clarify, note that an estimate $\mathbf{x}$ provides also an estimate of the noise $\mathbf{H}(\mathbf{y}-\mathbf{x})$. Therefore, if the noise variance σ^2 is (approximately) known, then it is suitable to set r equal to $\sqrt{N}\sigma$.

We refer to (34) as the C-PATV problem (for Constrained Polynomial Approximation–Total Variation).

Even if the noise variance is unknown, the parameter, r, being the standard deviation of the residual, has a direct and clear interpretation, unlike λ in (15). Therefore, it can be more convenient to tune r in (34) than λ in (15). A disadvantage of the constrained problem (34) compared to the unconstrained problem (15) is that the constrained problem is generally a more difficult optimization problem to solve. However, recently, an algorithm, C-SALSA [2], has been introduced that is particularly effective for constrained problems such as (34). In the following, the C-SALSA approach is used to derive an algorithm for the constrained formulation of the PATV problem.

A. Algorithm C-PATV

As above, an iterative algorithm for solving (34) can be derived by a suitable variable splitting and the application of ADMM. Adapting the derivation of C-SALSA, problem (34) can be written as:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{u}_0,\mathbf{u}_1}{\operatorname{argmin}} \quad & \|\mathbf{u}_0\|_1 && \text{(35a)}\\ \text{such that} \quad & \|\mathbf{H}\mathbf{y}-\mathbf{u}_1\|_2 \le r && \text{(35b)}\\ & \mathbf{u}_0 = \mathbf{D}\mathbf{x} && \text{(35c)}\\ & \mathbf{u}_1 = \mathbf{H}\mathbf{x} && \text{(35d)} \end{aligned}$$

Using ADMM, problem (35) can be solved by the iterative algorithm:

$$\mathbf{u}_0, \mathbf{u}_1 \leftarrow \begin{cases} \underset{\mathbf{u}_0,\mathbf{u}_1}{\operatorname{argmin}} & \|\mathbf{u}_0\|_1 + \mu_0\|\mathbf{u}_0 - \mathbf{D}\mathbf{x} - \mathbf{d}_0\|_2^2 \\ & \quad + \mu_1\|\mathbf{u}_1 - \mathbf{H}\mathbf{x} - \mathbf{d}_1\|_2^2 \\ \text{such that} & \|\mathbf{H}\mathbf{y} - \mathbf{u}_1\|_2 \le r \end{cases} \tag{36a}$$

$$\mathbf{x} \leftarrow \underset{\mathbf{x}}{\operatorname{argmin}} \begin{cases} \mu_0\|\mathbf{u}_0 - \mathbf{D}\mathbf{x} - \mathbf{d}_0\|_2^2 \\ \quad + \mu_1\|\mathbf{u}_1 - \mathbf{H}\mathbf{x} - \mathbf{d}_1\|_2^2 \end{cases} \tag{36b}$$

$$\mathbf{d}_0 \leftarrow \mathbf{d}_0 - (\mathbf{u}_0 - \mathbf{D}\mathbf{x}) \tag{36c}$$

$$\mathbf{d}_1 \leftarrow \mathbf{d}_1 - (\mathbf{u}_1 - \mathbf{H}\mathbf{x}) \tag{36d}$$

$$\text{Go to (36a)} \tag{36e}$$

Note that $\mathbf{u}_0$ and $\mathbf{u}_1$ are decoupled in (36a); therefore, (36a) can be written as two independent minimization problems:

$$\mathbf{u}_0 \leftarrow \underset{\mathbf{u}_0}{\operatorname{argmin}} \|\mathbf{u}_0\|_1 + \mu_0\|\mathbf{u}_0 - \mathbf{D}\mathbf{x} - \mathbf{d}_0\|_2^2 \tag{37a}$$

$$\mathbf{u}_1 \leftarrow \begin{cases} \underset{\mathbf{u}_1}{\operatorname{argmin}} & \|\mathbf{u}_1 - \mathbf{H}\mathbf{x} - \mathbf{d}_1\|_2^2 \\ \text{such that} & \|\mathbf{H}\mathbf{y} - \mathbf{u}_1\|_2 \le r. \end{cases} \tag{37b}$$

Minimization (37a) is obtained by soft thresholding. Minimization (37b) can be obtained explicitly as

$$\mathbf{u}_1 \leftarrow \operatorname{proj}(\mathbf{H}\mathbf{x} + \mathbf{d}_1;\, \mathbf{H}\mathbf{y},\, r),$$

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

$$\operatorname{proj}(\mathbf{v};\, \mathbf{c}, r) := \begin{cases} \mathbf{c} + \dfrac{r}{\|\mathbf{v}-\mathbf{c}\|_2}(\mathbf{v}-\mathbf{c}), & \|\mathbf{v}-\mathbf{c}\| > r \\ \mathbf{v}, & \|\mathbf{v}-\mathbf{c}\| \le r. \end{cases} \tag{38}$$

The solution to (36b) is given by

$$\mathbf{x} \leftarrow (\mu_0\mathbf{D}^t\mathbf{D} + \mu_1\mathbf{H}^t\mathbf{H})^{-1}(\mu_0\mathbf{D}^t(\mathbf{u}_0 - \mathbf{d}_0) + \mu_1\mathbf{H}^t(\mathbf{u}_1 - \mathbf{d}_1)). \tag{39}$$

Note that the matrix $\mu_0\mathbf{D}^t\mathbf{D} + \mu_1\mathbf{H}^t\mathbf{H}$ in (39) is not sparse because $\mathbf{H}$ is not sparse; moreover, this matrix is large (size $N \times N$), so (39) calls for substantial computation if it is implemented directly. However, the computational cost of (39) can be reduced using the matrix inverse lemma as follows:

$$\begin{aligned} (\mu_0\mathbf{D}^t\mathbf{D} + \mu_1\mathbf{H}^t\mathbf{H})^{-1} &= (\mu_0\mathbf{D}^t\mathbf{D} + \mu_1\mathbf{H})^{-1} && \text{(40a)}\\ &= \left[(\mu_0\mathbf{D}^t\mathbf{D} + \mu_1\mathbf{I}) - \mu_1\mathbf{G}(\mathbf{G}^t\mathbf{G})^{-1}\mathbf{G}^t\right]^{-1} && \text{(40b)}\\ &= \mathbf{A}^{-1} + \mathbf{A}^{-1}\,\mathbf{G}\,\mathbf{F}^{-1}\,\mathbf{G}^t\,\mathbf{A}^{-1}, && \text{(40c)} \end{aligned}$$

where

$$\mathbf{A} := \mu_0\mathbf{D}^t\mathbf{D} + \mu_1\mathbf{I} \tag{41}$$

$$\mathbf{F} := \frac{1}{\mu_1}\mathbf{G}^t\mathbf{G} - \mathbf{G}^t\mathbf{A}^{-1}\,\mathbf{G}. \tag{42}$$

The expression (40c) can be implemented efficiently because $\mathbf{A}$ is sparse and $\mathbf{F}$ is small (size $d \times d$, with $d \ll N$). To clarify, (39) can be implemented as

$$\mathbf{b} \leftarrow \mathbf{A}^{-1}(\mu_0\mathbf{D}^t(\mathbf{u}_0 - \mathbf{d}_0) + \mu_1\mathbf{H}^t(\mathbf{u}_1 - \mathbf{d}_1)) \tag{43a}$$

$$\mathbf{x} \leftarrow \mathbf{b} + \mathbf{A}^{-1}(\mathbf{G}\,(\mathbf{F}^{-1}\,(\mathbf{G}^t\,\mathbf{b}))) \tag{43b}$$

Also note that the matrix $\mathbf{F}$ can be found efficiently in (42) due to $\mathbf{A}$ being sparse.

In summary, the algorithm (36) can be implemented explicitly as:

$$\mathbf{u}_0 \leftarrow \operatorname{soft}(\mathbf{D}\mathbf{x} + \mathbf{d}_0, 0.5/\mu_0) \tag{44a}$$

$$\mathbf{u}_1 \leftarrow \operatorname{proj}(\mathbf{H}\mathbf{x} + \mathbf{d}_1;\, \mathbf{H}\mathbf{y},\, r) \tag{44b}$$

$$\mathbf{b} \leftarrow \mathbf{A}^{-1}\,(\mu_0\mathbf{D}^t(\mathbf{u}_0 - \mathbf{d}_0) + \mu_1\mathbf{H}^t(\mathbf{u}_1 - \mathbf{d}_1)) \tag{44c}$$

$$\mathbf{x} \leftarrow \mathbf{b} + \mathbf{A}^{-1}(\mathbf{G}\,(\mathbf{F}^{-1}\,(\mathbf{G}^t\,\mathbf{b}))) \tag{44d}$$

$$\mathbf{d}_0 \leftarrow \mathbf{d}_0 - (\mathbf{u}_0 - \mathbf{D}\mathbf{x}) \tag{44e}$$

$$\mathbf{d}_1 \leftarrow \mathbf{d}_1 - (\mathbf{u}_1 - \mathbf{H}\mathbf{x}) \tag{44f}$$

$$\text{Go to (44a)} \tag{44g}$$

We refer to (44) as the C-PATV algorithm.[6] The algorithm is computationally efficient because inverse $\mathbf{A}^{-1}$ in (44) can be implemented by sparse system solvers, as $\mathbf{A}$ is a sparse tridiagonal matrix. The inverse matrix $\mathbf{F}^{-1}$ can be computed a single time and absorbed with $\mathbf{G}^t$; also $\mathbf{F}$ is a matrix of size only $d \times d$ where d is the polynomial degree.

[6] A MATLAB program `cpatv` implementing algorithm (44) is listed as Program 3 in the supplementary material.

Note that the variable splitting used in (35) is different than that used in (16). A different variable splitting is used here, because, unlike the variable splitting in (35), the variable splitting in (16) would not lead to the easy and explicit projection in (37b). The price to be paid for the simple projection using this variable splitting is that the update of $\mathbf{x}$ requires more computation: the C-PATV algorithm (44) requires the solution to two sparse systems (of size N) per iteration (in (44c) and (44d)), while the PATV algorithm (22) requires the solution to just one sparse system (of size N) per iteration (in (22d)).

To illustrate the C-PATV algorithm, consider the example illustrated in Fig. 3. For the length $N = 100$ noisy data illustrated in Fig. 3a, the additive noise has standard deviation $\sigma = 0.26$. Therefore, to test the C-PATV algorithm we set $r = \sqrt{N}\sigma = 2.6$ and perform the C-PATV algorithm (44) with polynomial degree $d = 2$. The result of the C-PATV algorithm is essentially the same as the result of the PATV algorithm illustrated in Fig. 3 (hence, we do not illustrate the result here). The history of the cost function through the progression of the C-PATV algorithm is illustrated in Fig. 8a. However, for the constrained formulation, the constraint (34b) should also be monitored to evaluate the convergence of the algorithm. Figure 8b illustrates the history of the constraint function. In Fig. 8c, the constraint and cost function are illustrated jointly; the algorithm starts in the upper right corner and progresses to a solution for which the constraint function $\|\mathbf{H}(\mathbf{y} - \mathbf{x})\|_2$ equals 2.6 as specified.

B. Local PA and TV Filtering: Constrained Formulation

The constrained formulation of the LoPATV problem (25) can be formulated as:

$$\underset{\mathbf{x}}{\text{argmin}} \quad \|\mathbf{D}\mathbf{x}\|_1 \tag{45a}$$

$$\text{such that} \quad \sum_{i=1}^{M} \|\mathbf{H}\mathbf{S}_i(\mathbf{y} - \mathbf{x})\|_2^2 \leq r^2. \tag{45b}$$

It is reasonable to set $r^2 = ML\sigma^2$, where σ^2 is the noise variance. The development of an algorithm to solve (45) follows along similar lines as above. However, we were not able to derive an algorithm that uses only sparse banded matrices; hence it is not as computationally efficient as the unconstrained formulation and we omit its derivation.

VI. Enhanced PA-TV via ℓ_p-norm Minimization

This section describes a modification of the preceding cost functions and algorithms so as to produce estimated signals having fewer extraneous small step discontinuities. Note that reducing the number of discontinuities in $\mathbf{x}$ is equivalent to making $\mathbf{D}\mathbf{x}$ more sparse. The ℓ_1-norm is often sub-optimal in terms of producing sparse signals, as illustrated for example in [28] in the case of image processing. There are several methods to obtain solutions with enhanced sparsity as compared to ℓ_1-norm minimization; for example iterative hard thresholding [5], [24], [28], reweighted ℓ_1-norm minimization [8], greedy ℓ_1-norm minimization [25], and hard thresholding pursuit [20].

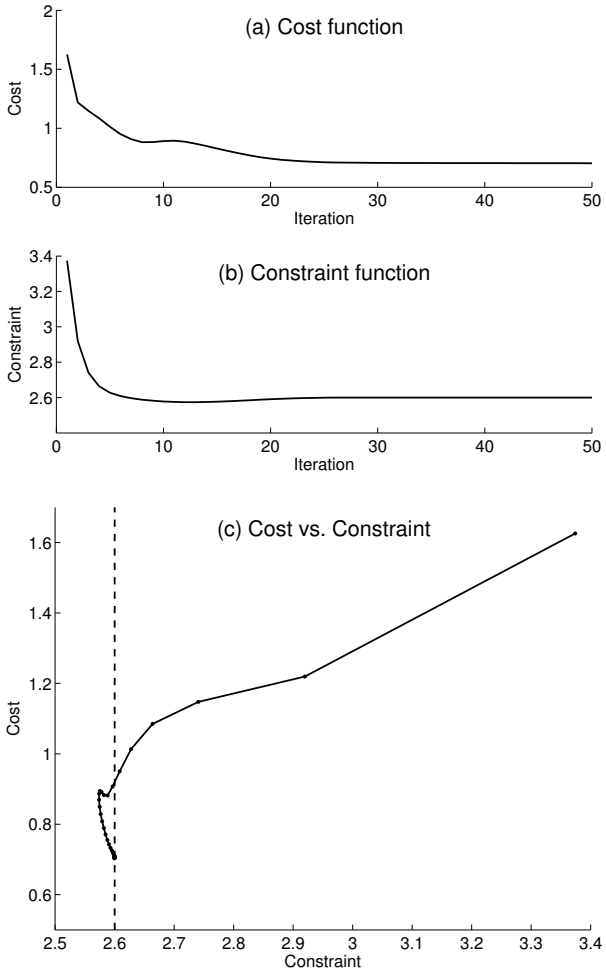


Fig. 8. Convergence of C-PATV algorithm (44) for constrained formulation for example data. The constraint is satisfied upon convergence of the algorithm. (a) The cost function $\|\mathbf{D}\mathbf{x}\|_1$ and (b) constraint $\|\mathbf{H}(\mathbf{y} - \mathbf{x})\|_2^2$ versus iteration. (c) Cost versus constraint, over the course of the algorithm.

In the following, in order to enhance the sparsity of $\mathbf{D}\mathbf{x}$, we will use the ℓ_p-norm with $0 < p < 1$ in place of the ℓ_1-norm in (15) and (25). Specifically, the new cost function is given by

$$J(\mathbf{x}) = \lambda R(\mathbf{D}\mathbf{x}) + F(\mathbf{y} - \mathbf{x}) \tag{46a}$$

where

$$R(\mathbf{v}) = \sum_{n=1}^{N} (|v(n)| + \epsilon)^p, \tag{46b}$$

with $0 < p \leq 1$ and $\epsilon > 0$. The function $F(\cdot)$ denotes the second term in (15) or (25) depending on the problem being addressed. When $p < 1$, the regularization function $R(\mathbf{v})$ promotes sparsity more strongly than the ℓ_1 norm. The small positive constant ϵ is used in (46b) so as to avoid numerical issues when any elements of $\mathbf{v}$ are equal to zero[7].

Although the signal $\mathbf{x}$ minimizing the new cost function (46) has fewer extraneous discontinuities compared to the signal obtained by ℓ_1-norm minimization, the cost function (46) is non-convex and more difficult to minimize. In order to

[7]In fact, it is fine to set ϵ to zero, provided it is understood to mean that subsequent soft-thresholding produces zeros corresponding to $v(n) = 0$.

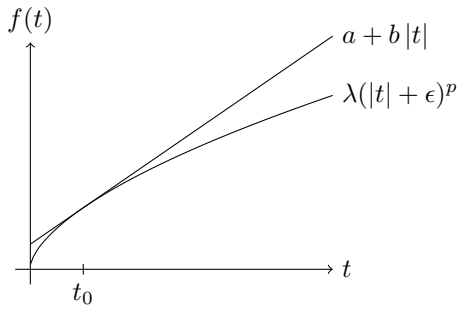


Fig. 9. Majorizer function as expressed in (50) with a tangent point at t_0.

minimize the cost function (46), the majorization-minimization (MM) optimization procedure can be effectively used [19].

Instead of minimizing $J(\mathbf{x})$ directly, the MM procedure minimizes a sequence of simpler functions $G_k(\mathbf{x})$ where each $G_k(\mathbf{x})$ is a majorizer of $J(\mathbf{x})$, i.e., $G_k(\mathbf{x}) \geq J(\mathbf{x})$ for all $\mathbf{x}$. Let $\mathbf{x}^{(k)}$ denote the minimizer of $G_{k-1}(\mathbf{x})$. Then the MM procedure asks that each $G_k(\mathbf{x})$ be chosen so that $G_k(\mathbf{x}^{(k)}) = J(\mathbf{x}^{(k)})$. That is, each majorizer $G_k(\mathbf{x})$ depends on the minimizer $\mathbf{x}^{(k)}$ of the previous majorizer $G_{k-1}(\mathbf{x})$. More details about the majorization-minimization procedure can be found in [19] and references therein. When $J(\mathbf{x})$ is convex, then under mild conditions, the sequence $\mathbf{x}^{(k)}$ produced by the MM procedure converges to the minimizer of $J(\mathbf{x})$.

Because (46) is non-convex when $p < 1$, the sequence $\mathbf{x}^{(k)}$ can not be guaranteed to converge to the global minimizer of $J(\mathbf{x})$, only to an approximate optimal solution, depending on the initialization $\mathbf{x}^{(0)}$ and the way in which the functions $G_k(\mathbf{x})$ are chosen. Here, we will initialize the MM process with the solution to (15) or (25) which are already reasonably sparse. The more tight the majorization functions $G_k(\mathbf{x})$ are to $J(\mathbf{x})$ the more effective the MM procedure will be. We will use

$$G_k(\mathbf{x}) = M_k(\mathbf{D}\mathbf{x}) + F(\mathbf{y} - \mathbf{x}) \tag{47}$$

where

$$M_k(\mathbf{v}) := \sum_{n=1}^{N} (a_n^{(k)} + b_n^{(k)}\,|v(n)|), \tag{48}$$

where $a_n^{(k)}$ and $b_n^{(k)}$ are chosen so as to satisfy:

$$G_k(\mathbf{x}) \geq J(\mathbf{x}),\ \forall \mathbf{x}, \tag{49a}$$

$$G_k(\mathbf{x}^{(k)}) = J(\mathbf{x}^{(k)}). \tag{49b}$$

Note that, to find a and b, such that

$$a + b\,|t| \geq \lambda(|t| + \epsilon)^p,\ \forall t \tag{50a}$$

$$a + b\,|t_0| = \lambda(|t_0| + \epsilon)^p \tag{50b}$$

as illustrated in Fig. 9, leads to

$$b = \lambda\, p\, (|t_0| + \epsilon)^{p-1},$$

$$a = \lambda\, (t_0 + \epsilon)^p - b\, t_0.$$

The value b is obtained as the derivative of $\lambda(|t| + \epsilon)^p$ at $t = t_0$. Therefore, to satisfy (49), $b_n^{(k)}$ should be chosen as

$$b_n^{(k)} = \lambda p(|(\mathbf{D}\mathbf{x}^{(k)})(n)| + \epsilon)^{p-1}.$$

The parameters $a_n^{(k)}$ should be chosen similarly, but will not be needed. Note that $M_k(\mathbf{v})$ in (48) can be written as

$$\begin{aligned} M_k(\mathbf{v}) &= \sum_{n=1}^{N} a_n^{(k)} + \sum_{n=1}^{N} b_n^{(k)}\,|v(n)| \\ &= C + \|\mathbf{b}^{(k)} \odot \mathbf{v}\|_1, \end{aligned}$$

where $\odot$ denotes element-wise multiplication of two vectors, and C is an additive constant that has no influence on the subsequent minimization. Hence $G_k(\mathbf{x})$ in (47) can be written as

$$G_k(\mathbf{x}) = C + \|\mathbf{b}^{(k)} \odot \mathbf{D}\mathbf{x}\|_1 + F(\mathbf{y} - \mathbf{x}).$$

Hence, an MM-based iterative algorithm to minimize the new cost function (46) is given by

$$\mathbf{x} \leftarrow \underset{\mathbf{x}}{\operatorname{argmin}}\ \lambda\|\mathbf{D}\mathbf{x}\|_1 + F(\mathbf{y} - \mathbf{x}) \tag{51a}$$

$$\mathbf{b} \leftarrow \lambda p\,(|(\mathbf{D}\mathbf{x})| + \epsilon)^{p-1} \tag{51b}$$

$$\mathbf{x} \leftarrow \underset{\mathbf{x}}{\operatorname{argmin}}\ \|\mathbf{b} \odot \mathbf{D}\mathbf{x}\|_1 + F(\mathbf{y} - \mathbf{x}) \tag{51c}$$

$$\text{Go to (51b)} \tag{51d}$$

In (51b), the absolute value and power operations are performed element-wise. The minimization problem (51c) can be solved with the preceding PATV and LoPATV algorithms with one minor change: (22a) and (32a) should be changed to

$$\mathbf{u}_0 \leftarrow \operatorname{soft}(\mathbf{D}\mathbf{x} + \mathbf{d}_0, 0.5\,\mathbf{b}/\mu_0),$$

which means that the n-th component of $\mathbf{D}\mathbf{x} + \mathbf{d}_0$ is soft-thresholded with threshold $0.5\,b(n)/\mu_0$. That is, the threshold is applied element-wise.

To illustrate the relative effectiveness of the cost function (46) compared with ℓ_1-norm cost function (15), Fig. 10 shows the result obtained using $p = 0.7$, which can be compared with the result illustrated in Fig. 3 obtained using the ℓ_1-norm. It can be seen in Fig. 4 that the TV component obtained using the ℓ_1-norm cost function has three step discontinuities, while the TV component in Fig. 10 obtained using the ℓ_p-norm cost function has a single step discontinuity. In both cases, the same value was used for λ. The ℓ_p-norm result was obtained with 15 iterations of the MM algorithm (51). Similarly, the effectiveness of cost function (46) compared with (25) is illustrated in Fig. 11. The ℓ_p-norm minimization produces a cleaner result than ℓ_1-norm minimization (compare with Fig. 6).

VII. Nano-particle Detection

The Whispering Gallery Mode (WGM) biosensor, designed to detect individual nano-particles in real time, is based on detecting discrete changes in the resonance frequency of a microspherical cavity. The adsorption of a nano-particle onto the surface of a custom designed microsphere produces a theoretically predicted but minute change in the optical properties of the sphere [3]. By measuring the response of a frequency-swept laser through an optical fibre coupled with

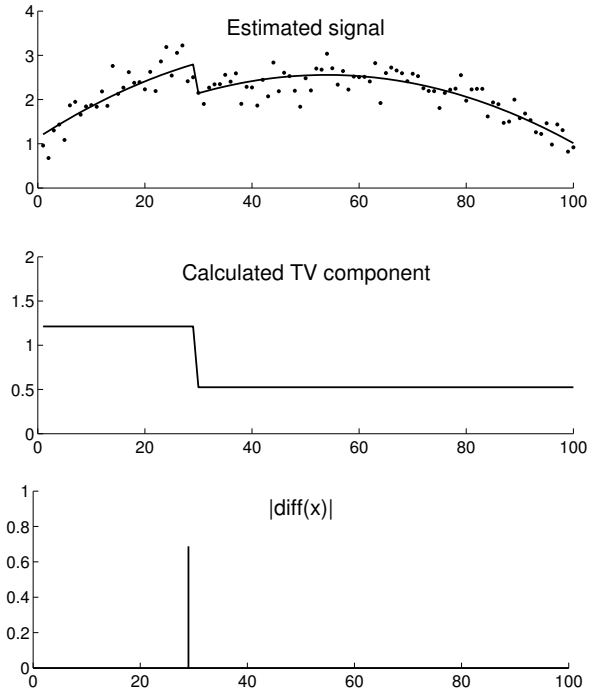


Fig. 10. Enhanced PATV applied to data of Fig. 3. Compare with Figs. 3d. The discontinuities are captured more accurately than by ℓ_1-norm minimization.

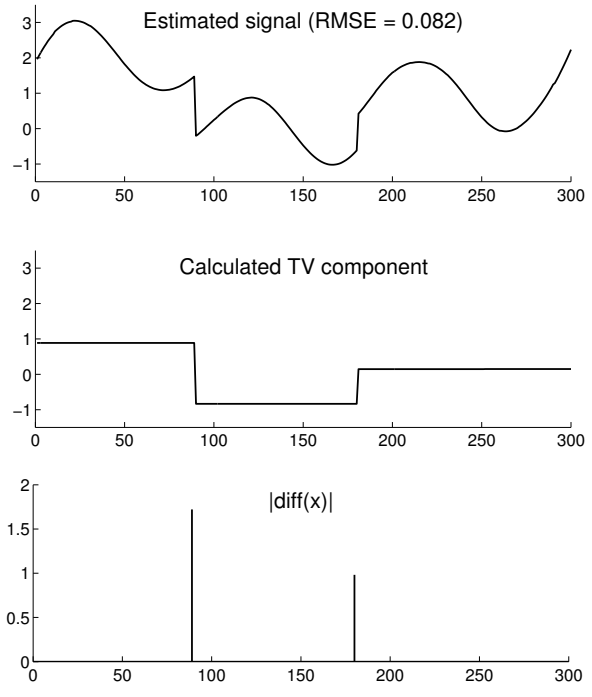


Fig. 11. Enhanced LoPATV applied to data of Fig. 6. Compare with Fig. 6. The discontinuities are captured more accurately than by ℓ_1-norm minimization.

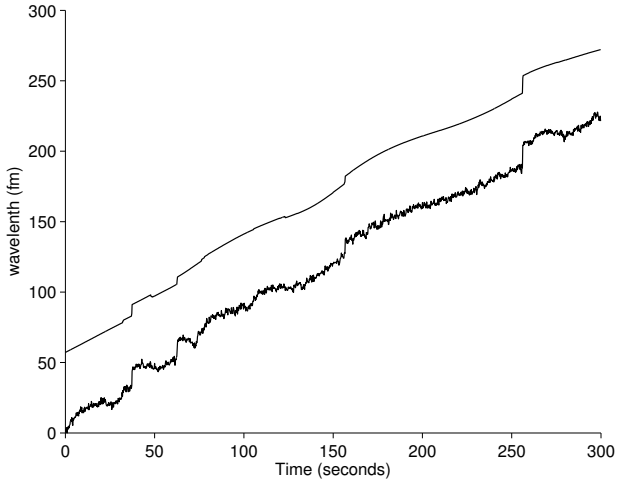
the microsphere, the spectral response characteristic of the sphere can be recorded as a function of time. In particular, the resonance frequency of a whispering gallery mode of the sphere will change due to the adsorption of a nano-particle, and this peak in the spectral response can be traced as a function of time. Such data from a WGM biosensor of this kind is illustrated in Fig. 12. In the figure it can be observed that the free-space resonance wavelength shift above 780.674 nm exhibits discrete steps with step sizes between 5 and 10 femtometers (10^{-15} m). (Note, in Fig. 12a the vertical axis is offset; at time zero the wavelength is 780.674 nm.)

The data illustrated in Fig. 12 was acquired with a sampling rate of 5 samples/second. The LoPATV algorithm was used to jointly estimate the step (TV) and smooth (local polynomial) components. For the LoPATV algorithm, we used a block length of 200 samples (40 seconds), an overlap of 150 samples (30 seconds), and a polynomial degree $d = 1$. Figure 12b illustrates the TV component and its first-order difference function which can be seen to be sparse. Figure 13 illustrates the results of the enhanced LoPATV algorithm (ℓ_p-norm minimization) with $p = 0.7$ and $\epsilon = 0.001$. It can be seen that ℓ_p norm minimization suppresses the smaller potentially extraneous steps edges.

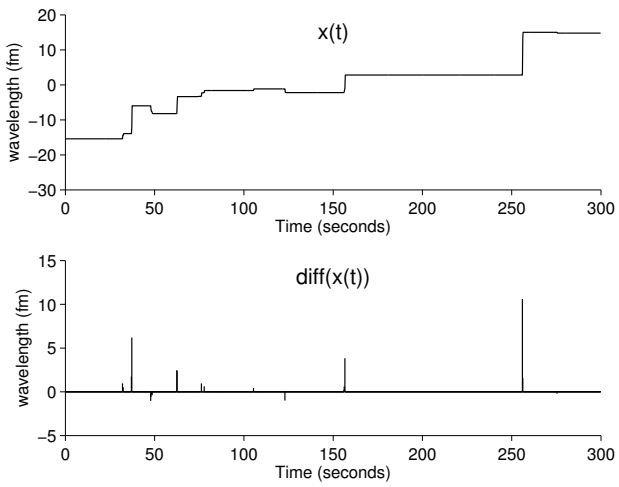
We note that the problem of estimating steps in noisy data can be addressed using several existing methods. For example, the step fitting procedure of Kerssemakers et. al. [23] was developed for a related problem in biosensing. However, algorithms, of which we are aware, appear to assume or model the step signal as being free of the smoothly varying background activity that can be seen in Fig. 12. The method developed here explicitly allows for the presence of an additive low frequency background signal and models it as being locally well approximated by a low order polynomial.

VIII. Conclusion

This paper has described the joint use of least square polynomial approximation and total variation for the smoothing of time series with additive step discontinuities. The problem is formulated as an optimization problem in which the derivative (first-order difference) of the step component is modeled as being sparse (via total regularization). The formulation requires the specification of only a few parameters: the polynomial degree d, the size of the window L (block-length over which the polynomial approximation is valid), and a regularization parameter λ. An iterative algorithm is derived using ADMM; each iteration requires the solution to only a sparse (banded) set of equations and is therefore computationally efficient. The method does not require the explicit segmentation of the time series data into disjoint segments (however, the method can be used to estimate the jump points which can then be utilized for further signal estimation). The constrained form of the problem is also addressed, wherein the energy of the residual is constrained. The method is illustrated on data produced by a biosensor that functions by detecting discrete changes (on the order of femtometers) in the resonance frequency of a microspherical cavity.

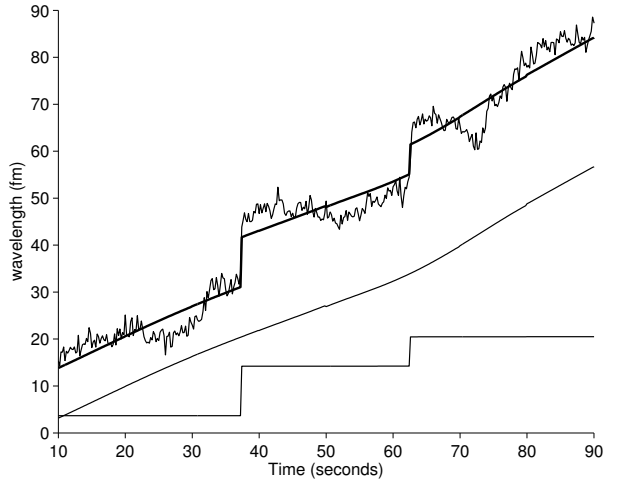


(a) Biosensor data and result of LoPATV

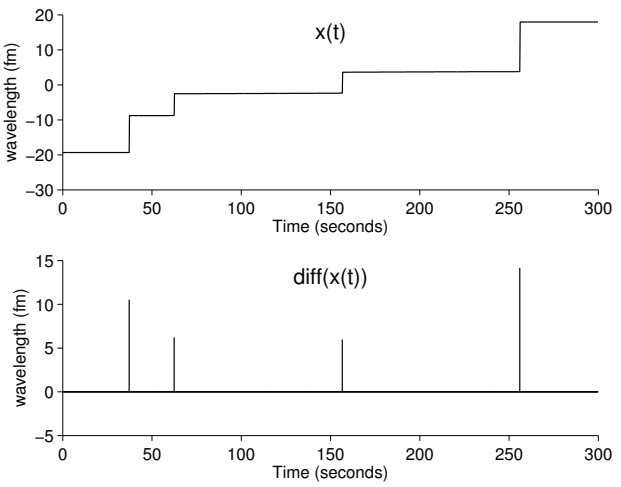


(b) Calculated TV component

Fig. 12. LoPATV algorithm applied to the nano-particle detection problem. (a) Biosensor data and result of LoPATV (vertically offset for clarity). (b) Calculated TV component and its first-order difference function.



(a) Biosensor data and result of enhanced LoPATV



(b) Calculated TV component

Fig. 13. Enhanced LoPATV algorithm applied to the nano-particle detection problem. (a) Biosensor data and result of enhanced LoPATV. For clarity, an 80 second segment (400 points) is illustrated, and the local polynomial and TV components are illustrated individually. (b) Calculated TV component and its first-order difference function. Minimization of the ℓ_p norm results in fewer estimated step discontinuities (equivalently, a sparser first-order difference function) compared with the ℓ_1 norm. Compare with Fig. 12b.

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. Arnold, M. Khoshima, I. Teraoka, S. Holler, and F. Vollmer. Shift of whispering-gallery modes in microspheres by protein adsorption. *Opt. Lett*, 28(4):272–274, February 15, 2003.

[4] J.-F. Aujol, G. Aubert, L. Blanc-Féraud, and A. Chambolle. Image decomposition into a bounded variation component and an oscillating component. *J. Math. Imag. Vis.*, 22:71–88, 2005.

[5] T. Blumensath and M. E. Davies. Normalized iterative hard thresholding: Guaranteed stability and performance. *IEEE. J. Sel. Top. Signal Processing*, 4(2):298–309, April 2010.

[6] 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.

[7] V. Bruni, B. Piccoli, and D. Vitulano. A fast computation method for time scale signal denoising. *Signal Image and Video Processing*, 3(1):63–83, 2008.

[8] E. J. Candès, M. B. Wakin, and S. Boyd. Enhancing sparsity by reweighted l1 minimization. *J. Fourier Anal. Appl.*, 14(5):877–905, December 2008.

[9] A. Chambolle. An algorithm for total variation minimization and applications. *J. of Math. Imaging and Vision*, 20:89–97, 2004.

[10] T. F. Chan, S. Osher, and J. Shen. The digital TV filter and nonlinear denoising. *IEEE Trans. Image Process.*, 10(2):231–241, February 2001.

[11] T. F. Chan and H. M. Zhou. ENO-wavelet transforms for piecewise smooth functions. *SIAM J. Numer. Anal.*, 40(4):1369–1404, April 2002.

[12] T. F. Chan and H.-M. Zhou. Total variation wavelet thresholding. *J. Sci. Comput.*, 32(2):315–341, August 2007.

[13] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In H. H. Bauschke, R. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. Springer-Verlag (New York), 2010.

[14] M. S. Crouse, R. D. Nowak, and R. G. Baraniuk. Wavelet-based signal processing using hidden Markov models. *IEEE Trans. Signal Process.*, 46(4):886–902, April 1998.

[15] P. L. Dragotti and M. Vetterli. Wavelet footprints: theory, algorithms,

and applications. *IEEE Trans. Signal Process.*, 51(5):1306–1323, May 2003.

[16] S. Durand and J. Froment. Reconstruction of wavelet coefficients using total variation minimization. *SIAM J. Sci. Comput.*, 24(5):1754–1767, 2003.

[17] E. Esser, X. Zhang, and T. F. Chan. A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. *SIAM Journal on Imaging Sciences*, 3(4):1015–1046, 2010.

[18] J. Fan and I. Gijbels. *Local polynomial modelling and its applications*. Chapman & Hall, 1996.

[19] M. Figueiredo, J. Bioucas-Dias, and R. Nowak. Majorization-minimization algorithms for wavelet-based image restoration. *IEEE Trans. Image Process.*, 16(12):2980–2991, December 2007.

[20] S. Foucart. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM J. Numer. Anal.*, 49(6):2543–2563, 2010.

[21] F. B. Hildebrand. *Introduction to Numerical Analysis: Second Edition*. Dover, 1987.

[22] V. Katkovnik, K. Egiazarian, and J. Astola. *Local Approximation Techniques in Signal and Image Processing*. SPIE Press, 2006.

[23] J. W. J. Kerssemakers, E. L. Munteanu, L. Laan, T. L. Noetzel, M. E. Janson, and M. Dogterom. Assembly dynamics of microtubules at molecular resolution. *Nature*, 442:709–712, 10 August 2006. Supplementary Methods 3: Step fitting algorithm.

[24] N. Kingsbury and T. Reeves. Redundant representation with complex wavelets: how to achieve sparsity. In *Proc. IEEE Int. Conf. Image Processing*, 2003.

[25] I. Kozlov and A. Petukhov. Sparse solutions of underdetermined linear systems. In W. Freeden et al., editor, *Handbook of Geomathematics*. Springer, 2010.

[26] W. S. Lee and A. A. Kassim. Signal and image approximation using interval wavelet transform. *IEEE Trans. Image Process.*, 16(1):46–56, January 2007.

[27] S. J. Orfanidis. *Introduction to Signal Processing*. Prentice Hall, 1996.

[28] J. Portilla and L. Mancera. L0-based sparse approximation: two alternative methods and some applications. In *Proceedings of SPIE*, volume 6701 (Wavelets XII), 2007.

[29] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[30] A. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[31] R. W. Schafer. What is a Savitzky-Golay filter? *IEEE Signal Processing Magazine*, 28(4):111–117, July 2011.

[32] H. W. Schüssler and P. Steffen. Some advanced topics in filter design. In J. S. Lim and A. V. Oppenheim, editors, *Advanced Topics in Signal Processing*, chapter 8, pages 416–491. Prentice Hall, 1988.

[33] I. Selesnick and I. Bayram. Total variation filtering. *Connexions Web site*, 2009. http://cnx.org/content/m31292/1.1/.

[34] I. W. Selesnick and C. S. Burrus. Design of FIR digital filters. In V. K. Madessetti and Douglas B. Williams, editors, *The Digital Signal Processing Handbook*. CRC Press, Boca Raton, 1997.

[35] 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.

[36] L. A. Vese and S. Osher. Image denoising and decomposition with total variation minimization and oscillatory functions. *J. Math. Imag. Vis.*, 20:7–18, 2004.

[37] F. Vollmer, S. Arnold, and D. Keng. Single virus detection from the reactive shift of a whispering-gallery mode. *Proc. Nat. Acad. Sciences USA (PNAS)*, 105(52):20701–20704, 2008.

[38] M. B. Wakin, J. K. Romberg, H. Choi, and R. G. Baraniuk. Wavelet-domain approximation and compression of piecewise smooth images. *IEEE Trans. Image Process.*, 15(5):1071–1087, May 2006.

[39] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM J. on Imaging Sciences*, 1(3):248–272, 2008.

SUPPLEMENTARY MATERIAL

A MATLAB program `patv` implementing the PATV algorithm (22) is listed as Program 1.
A MATLAB program `lopatv` implementing LoPATV algorithm (32) is listed as Program 2.
A MATLAB program `cpatv` implementing C-PATV algorithm (44) is listed as Program 3.
These programs require several subprograms which are listed below.

A more complete MATLAB software related to this paper is available at http://eeweb.poly.edu/iselesni/patv/
The software package includes programs to reproduce the examples in the paper.

```
function [x, p, cost] = patv(y, d, lambda, Nit, mu0, mu1)
% [x, p, cost] = patv(y, d, lambda, Nit, mu0, mu1)
% PATV: Simultaneous polynomial approximation and total variation
% filtering
%
% INPUT
%   y - noisy data
%   d - order of polynomial
%   lambda - regularization parameter
%   Nit - number of iterations
%   mu0, mu1 - augmented Lagrangian parameters
%
% OUTPUT
%   x - TV component
%   p - polynomial component
%   cost - cost function history

% Ivan Selesnick
% Polytechnic Institute of New York University
% December 2011
% Reference: Polynomial Smoothing of Time Series with Additive Step Discontinuities
% I. W. Selesnick, S. Arnold, and V. R. Dantham

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

n = (0:N-1)';
G = zeros(N,d);
for k = 1:d, G(:,k) = n.^k;  end        % exclude dc term (included in TV component)
G = orth(G);                            % orthogonalize cols of G (so that G' G = I)

e = ones(N-1,1);
D = spdiags([-e e],[0 1],N-1,N);        % D : first-order difference (sparse matrix)
A = mu0*(D'*D) + mu1*speye(N);          % A : mu0 D'D + mu1 I (sparse matrix)
D = @(x) diff(x);                       % D (operator)
DT = @(x) [-x(1); -diff(x); x(end)];    % D' (operator)
H = @(x) x - G * (G'*x);

x = zeros(N,1);                         % initializations
d0 = zeros(N-1,1);
d1 = zeros(N,1);

for k = 1:Nit
    u0 = soft(D(x)+d0, 0.5*lambda/mu0);
    b = x + d1;
    u1 = ((y + mu1*b) + G*(G'*(b-y))) / (1+mu1);
    x = A \ (mu0*DT(u0-d0) + mu1*(u1-d1));         % sparse system solve
    d0 = d0 - (u0-D(x));
    d1 = d1 - (u1-x);
    cost(k) = sum(abs(lambda .* D(x))) + sum(abs(H(x-y)).^2);
end
p = G * (G' * (y - x));
```

Program 1: MATLAB program for simultaneous polynomial approximation and total variation filtering (PATV). The program implements algorithm (22).

```
function [x, p, cost] = lopatv(y, L, P, deg, lambda, Nit, mu0, mu)
% [x, p, cost] = lopatv(y, L, P, deg, lambda, Nit, mu0, mu)
% LoPATV: Simultaneous local polynomial approximation and total variation filtering
% (sliding window with overlapping)
%
% INPUT
%   y - noisy data
%   L - block length
%   P - overlapping (number of samples common to adjacent blocks)
%   deg - polynomial degree
%   lambda - regularization parameter
%   Nit - number of iterations
%   mu0, mu - augmented Lagrangian parameters
%
% OUTPUT
%   x - step function (TV component)
%   p - local polynomial component
%   cost - cost function history
%
% Number of blocks = (length(y)-L)/(L-P)+1
% If this is not an integer, then input signal y will be truncated.

% Ivan Selesnick
% Polytechnic Institute of New York University
% December 2011
% Reference: Polynomial Smoothing of Time Series with Additive Step Discontinuities
% I. W. Selesnick, S. Arnold, and V. R. Dantham

y = y(:);                               % convert to column vector
N = length(y);

M = (N-L)/(L-P);                        % M : number of blocks - 1
if M > floor(M)
    N = floor(M)*(L-P)+L;
    y = y(1:N);
    fprintf('Note in lopatv.m: The input signal will be truncated down to length %d\n',N)
    fprintf('so it is consistent with the block length %d and overlap %d.\n', L, P);
end

cost = zeros(1,Nit);                    % cost function history
G = zeros(L,deg+1);
for k = 0:deg, G(:,k+1) = (0:L-1)'.^k; end
G = orth(G);                            % orthogonalize columns of G (so that G' G = I)
H = @(x) x - G * (G'*x);
buff = @(x) buffer(x, L, P, 'nodelay');
s = invbuffer(buff(ones(1,N)), P);

e = ones(N-1,1);
D = spdiags([-e e],[0 1],N-1,N);                % D (sparse matrix)
A = mu0*(D'*D) + mu*spdiags(s,0,N,N);           % mu0 D'D + mu S (sparse matrix)
D = @(x) diff(x);                               % D (operator)
DT = @(x) [-x(1); -diff(x); x(end)];            % D' (operator)
x = zeros(size(y));                             % initialization
yb = buff(y);
xb = buff(x);
d0 = zeros(N-1,1);
d = zeros(size(xb));

for it = 1:Nit
    u0 = soft(D(x)+d0, 0.5*lambda/mu0);
    b = xb + d;
    u = ((yb + mu*b) + G*(G'*(b-yb))) / (1+mu);
    x = A \ (mu0*DT(u0-d0) + mu*invbuffer(u-d, P));    % sparse system solve
    xb = buff(x);
    d0 = d0 - (u0-D(x));
    d = d - (u-xb);
    cost(it) = sum(abs(lambda .* D(x))) + sum(sum(abs(H(buff(y-x))).^2));
end
p_blocks = G * (G' * buff(y - x));

w = hamming(L);                     % column vector
p = invbuffer(p_blocks, P, w);      % compute local polynomial estimate
```

Program 2: MATLAB program for simultaneous local polynomial approximation and total variation filtering (LoPATV). The

```
function [x,p,cost,constr] = cpatv(y, d, r, Nit, mu0, mu1)
% [x,p,cost,constr] = cpatv(y, d, r, Nit, mu0, mu1)
% C-PATV: Simultaneous polynomial approximation and total variation filtering,
% constrained formulation: ||H(y-x)||_2 <= r
%
% INPUT
%   y - noisy data
%   d - order of polynomial
%   r - constraint parameter
%   Nit - number of iterations
%   mu0, mu1 - augmented Lagrangian parameters
%
% OUTPUT
%   x - TV component
%   p - polynomial component
%   cost - cost function history
%   constr - constraint function history

% Ivan Selesnick
% Polytechnic Institute of New York University
% December 2011
% Reference: Polynomial Smoothing of Time Series with Additive Step Discontinuities
% I. W. Selesnick, S. Arnold, and V. R. Dantham

y = y(:);                               % convert to column vector
cost = zeros(1,Nit);                    % cost function history
constr = zeros(1,Nit);                  % constraint function history
N = length(y);
n = (0:N-1)';
G = zeros(N,d);
for k = 1:d, G(:,k) = n.^k;  end        % exclude dc term (included in TV component)
G = orth(G);                            % orthogonalize cols of G (so that G' G = I)
e = ones(N-1,1);
D = spdiags([-e e],[0 1],N-1,N);        % D (sparse matrix)
A = mu0*(D'*D) + mu1*speye(N);          % mu0 D'D + mu1 I (sparse matrix)
D = @(x) diff(x);                       % D (operator)
DT = @(x) [-x(1); -diff(x); x(end)];    % D' (operator)
H = @(x) x - G * (G'*x);                % H = I - G'*G (operator)
F = (1/mu1)*eye(d) - (G' * (A \ G));    % F (sparse matrix solve)
FG = F\G';
x = zeros(N,1);                         % initializations
d0 = zeros(N-1,1);
d1 = zeros(N,1);
Hy = H(y);
for k = 1:Nit
    u0 = soft(D(x) + d0, 0.5/mu0);
    u1 = projball(H(x) + d1, Hy, r);
    b = A \ (mu0*DT(u0-d0) + mu1*H(u1-d1));     % sparse matrix solve
    x = b + A \ ( G * (FG * b));                % sparse matrix solve
    d0 = d0 - (u0 - D(x));
    d1 = d1 - (u1 - H(x));
    cost(k) = sum(abs(D(x)));
    constr(k) = sqrt(sum(abs(H(x-y)).^2));
end
p = G * (G' * (y - x));
```

Program 3: MATLAB program for constrained formulation of simultaneous polynomial approximation and total variation filtering (C-PATV). The program implements algorithm (44).

The following program implements (38).

```
function v = projball(x, y, r);

R = sqrt( sum(abs(x(:) - y(:)).^2) );
if R <= r
    v = x;
else
    v = y + (r/R) * (x-y);
end
```

The following program implements the soft-threshold function.

```
function y = soft(x, T)
% Soft-threshold function (real or complex x)
% y = soft(x, T)
% Input
%    x : data
%    T : threshold
%
% If x and T are both multidimensional, then they must be of the same size.

y = max(1 - T./abs(x), 0) .* x;
```

The following program is required for the LoPATV program. It inverts the `buffer` function.

```
function x = invbuffer(X, P, w)
% x = invbuffer(X, P)
% inverts the buffer function
%
% x = invbuffer(X, P, w)
% Multiplies each frame by window w
%
% Each column of X is one block of x
% P : overlap

% Ivan Selesnick
% Polytechnic Institute of New York University
% December 2011


[L, M] = size(X);
% L : length of block
% M : number of blocks

x = zeros(L+(M-1)*(L-P),1);

if nargin < 3
    for i = 1:M
        x((i-1)*(L-P)+(1:L)) = x((i-1)*(L-P)+(1:L)) + X(:,i);
    end
else
    s = x; % zeros
    w = w(:);
    for i = 1:M
        x((i-1)*(L-P)+(1:L)) = x((i-1)*(L-P)+(1:L)) + w .* X(:,i);
        s((i-1)*(L-P)+(1:L)) = s((i-1)*(L-P)+(1:L)) + w;
    end
    x = x./s;
end
```