Denosing Using Wavelets and Projections onto the ℓ_1-Ball

October 6, 2014

A. Enis Cetin, M. Tofighi
Dept. of Electrical and Electronic Engineering, Bilkent University, Ankara, Turkey
cetin@bilkent.edu.tr, tofighi@ee.bilkent.edu.tr

I. Scope

Both wavelet denoising and denoising methods using the concept of sparsity are based on soft-thresholding. In sparsity-based denoising methods, it is assumed that the original signal is sparse in some transform domains such as the Fourier, DCT, and/or wavelet domain. The transfer domain coefficients of the noisy signal are projected onto ℓ_1-balls to reduce noise. In this lecture note, we establish the relation between the standard soft-thresholding-based denoising methods and sparsity-based wavelet denoising. We introduce a new deterministic soft-threshold estimation method using the epigraph set of ℓ_1-ball cost function. It is shown that the size of the ℓ_1-ball determined using linear algebra. The size of the ℓ_1-ball in turn determines the soft threshold. The key step is an orthogonal projection onto the epigraph set of the ℓ_1-norm cost function.

II. Prerequisites

The prerequisites for understanding this article's material are linear algebra, discrete-time signal processing, and basic optimization theory.

III. Problem Statement

In standard wavelet denoising, a signal corrupted by additive noise is wavelet transformed and resulting wavelet subsignals are soft and/or hard thresholded. After this step the denoised signal is reconstructed from the thresholded wavelet subsignals [1, 2]. Thresholding the wavelet coefficients intuitively makes sense because wavelet subsignals obtained from an orthogonal or biorgthogonal wavelet filter-bank exhibit large amplitude coefficients only around the edges or change the locations of the original signal. Other small amplitude coefficients should be due to noise. Many other related wavelet denoising methods are developed based on the Donoho and Johnstone's idea, see e.g. [1–5]. Most denoising methods take advantage of the sparse nature of practical signals in wavelet domain to reduce the noise [6–8].

Consider the following basic denoising framework. Let $v[n]$ be a discrete-time signal and $x[n]$ be a noisy version of $v[n]$:

$$x[n] = v[n] + \xi[n], \quad n = 0, 1, 2, \dots, N-1, \tag{1}$$

where $\xi[n]$ is the additive, i.i.d, zero-mean, white Gaussian noise with variance σ^2. A L-level discrete wavelet transform of $x[n]/\sqrt{N}$ is computed and the lowband signal x_L and wavelet subsignals $w_1, w_2, \dots, w_L$ are obtained. After this step, wavelet subsignals are soft-thresholded. The soft threshold, θ, can be selected in many ways [2–4, 9] using statistical methods. One possible choice is

$$\theta = \gamma.\sigma.\sqrt{2log(N)/N}, \tag{2}$$

where γ is a constant [2]. In Eq. (2) the noise variance σ^2 has to be known or properly estimated from the observations, $x[n]$.

In this lecture note, soft threshold values θ_i for each wavelet subsignal w_i are determined using a deterministic approach based on linear algebra and orthogonal projections.

IV. Proposed Solution

As pointed out above denoising is possible with the assumption that wavelet subsignals are also sparse signals. In most natural and practical signals wavelet subsignals are sparse due to the high-pass filtering operation. Therefore, it is possible to denoise the wavelet subsignals $w_1, w_2, \dots, w_L$ by projecting them onto ℓ_1-balls which are closed and convex sets. Projection w_p of a vector w_i onto a convex set C_i is determined by finding the shortest distance between w_i and the set C_i, and it is is obtained by solving the following minimization problem:

$$w_{pi} = \arg\max_{w \in C_i} \|w - w_i\|_2^2, \tag{3}$$

where $\|.\|_2$ is the Euclidean norm. The ℓ_1-ball C_i with size d_i is defined as follows:

$$C_i = \{w | \sum_n |w[n]| < d_i\} \tag{4}$$

where $w[n]$ is the n-th component of the vector w, and d_i is the size of the ℓ_1-ball. Projection w_{pi} of w_i onto the ℓ_1-ball C_i is obtained as follows:

$$\begin{aligned} w_{pi} &= \arg\max \|w_i - w\|_2^2 \\ \text{such that} &\sum_n |w[n]| \le d_i, \end{aligned} \tag{5}$$

where w_i is the i-th wavelet subsignal. Projection of a vector onto an ℓ_1-ball reduces the amplitudes of small valued coefficients of the vector. Since w_i is the i-th level wavelet vector, small valued wavelet coefficients, which are probably due to noise will be eliminated by the projection operation. This minimization problem has been studied by many researchers and computationally efficient algorithms were developed (see e.g., [10]). When $\sum_n |w[n]| \le d_i$ is satisfied, the projection $w_{pi} = w_i$. Otherwise, the projection vector w_{pi} is basically obtained with soft-thresholding as in ordinary wavelet denoising [1]. The user is referred to [10] for more details about the solution of the optimization problem (5). Each wavelet coefficient is modified as follows:

$$w_{pi}[n] = \text{sign}(w_i[n]).\text{max}\{|w_i[n]| - \theta_i, 0\}, \tag{6}$$

where $sign(w_i[n])$ is the sign of $w_i[n]$, and θ_i is the soft-thresholding constant whose value is determined according to the size of the ℓ_1-ball, d_i, as described in Algorithm 1. As pointed above soft-thresholding of wavelet coefficients reduces noise of the original signal because small valued wavelet coefficients are probably due to noise [1]. Other computationally efficient algorithms capable of computing the projection vector w_{pi} in $O(K)$ time are also described in [10].

Algorithm 1 Order $(Klog(K))$ algorithm implementing projection onto the ℓ_1-ball with size d_i.

1: **Inputs:**
   A vector $w_i = [w_i[0], \ldots, w_i[K-1]]$ and scalar $d_i > 0$
2: **Initialize:**
   Sort $|w_i[n]|$ for $n = 0, 1, \ldots, K-1$ and obtain the rank ordered sequence $\mu_1 \ge \mu_2 \ge, \ldots, \ge \mu_K$. The Lagrange multiplier, θ_i is given by

$$\theta_i = \frac{1}{\rho}\big(\sum_{n=1}^{\rho} \mu_n - d_i\big) \quad \text{such that} \quad \rho = max\big\{j \in \{0, 1, 2, \ldots, K-1\} : \mu_j - \frac{1}{j}\big(\sum_{r=1}^{j} \mu_r - d_i\big) > 0\big\} \tag{7}$$

3: **Output:**
   $w_{pi}[n] = sign(w_i[n]).max\{|w_i[n]| - \theta_i, 0\}$, $n = 0, 1, 2, \ldots, K-1$

Projection operations onto ℓ_1-ball will force small valued wavelet coefficients to zero and retain the edges and sharp variation regions of the signal because wavelet subsignals have large amplitudes corresponding to edges in most natural signals. As in standard wavelet denoising methods the low-band subsignal x_L is not processed because x_L is not a sparse signal for most practical signals.

In standard wavelet denoising, noise variance has to be estimated to determine the soft-threshold value. In this case, the size of the ℓ_1-ball d_i in (5) has to be estimated. Another parameter that has to be determined in both standard wavelet denoising and the ℓ_1-ball based denoising is the number of wavelet decomposition levels. In the next two sub-sections we describe how the size of the ℓ_1-ball and the number of wavelet decomposition levels can be determined.

[1]http://videolectures.net/icml08_singer_ep/

A. Estimation of Denoising Thresholds Using the Epigraph Set of ℓ_1-ball

The soft-threshold θ_i is determined by the size of ℓ_1-ball d_i as described in Algorithm 1. The size of the ℓ_1-ball can vary between 0 and $d_{max,i}$, which is determined by the boundary of the ℓ_1-ball touching the wavelet subsignal w_i:

$$d_{max,i} = \sum_n sign(w_i[n])w_i[n], \tag{8}$$

i.e., the wavelet subsignal w_i is one of the boundary hyperplanes of the ℓ_1-ball. Orthogonal projection of w_i onto a ball with $d = 0$ produces an all-zero result. On the other hand, projection of w_i onto a ball with size $d_{max,i}$, does not change w_i because w_i is on the boundary of the ℓ_1-ball. Therefore, for meaningful results the ball size z must satisfy the inequality $0 < z < d_{max}$, for denoising. This ℓ_1-ball condition can be expressed as follows:

$$g(w) = \sum_{k=0}^{K-1} |w[k]| \leq z, \tag{9}$$

where $g(w)$ is the ℓ_1-ball cost function and it is assumed that the wavelet subsignal w is casual and K is the length of the corresponding vector $w = [w[0], w[1], \dots, w[K-1]]^T \in \mathbb{R}^K$. The condition (9) corresponds to the epigraph set of the ℓ_1-ball in $\mathbb{R}^{K+1}$ [6, 8]. In (9) there are $K+1$ variables. These are $w[0], \dots, w[K-1]$, and z. The epigraph set C of ℓ_1-ball cost function

$$C = \{\mathbf{w} = [w\ z]^T \in \mathbb{R}^{K+1} : g(w) \leq z\} \tag{10}$$

is shown in Fig. 1 for $w \in \mathbb{R}^2$. The epigraph set represents a family of ℓ_1-balls for $0 < z \leq d_{i,max}$ in $\mathbb{R}^{K+1}$. Let

$$\mathbf{w}_i = [w_i, 0]^T = [w_i[0], w_i[1], \dots, w_i[K-1], 0], \in \mathbb{R}^{K+1} \tag{11}$$

be the $K+1$ dimensional version of vector w_i in $\mathbb{R}^{K+1}$. From now on we embolden vectors in $\mathbb{R}^{K+1}$ to distinguish them from K dimensional vectors.

To denoise the vector $\mathbf{w}_i$ we can project it onto the epigraph set of ℓ_1-ball because it is supposed to contain sparse versions of the signals. It is possible to determine all of the $\mathbb{R}^{K+1}$ unknowns, $w_{pi}[n]$, $n = 0, 1, \dots, K-1$, and z_p, by projecting the wavelet subsignal $\mathbf{w}_i = [w_i[0], \dots, w_i[K-1], 0]^T$ onto the epigraph set as graphically illustrated in Fig. 1. The projection vector $\mathbf{w}_{pi} = [w_{pi}, d]^T$ is unique and it is the closest vector to the wavelet subsignal $\mathbf{w}_i = [w_i, 0]^T$ on the epigraph set because the epigraph set is a closed and convex set. Orthogonal projection onto the epigraph set can be computed in two steps. In the first step, $[w_i, 0]^T$ is projected onto the nearest boundary hyperplane of the epigraph set which is

$$\sum_{n=0}^{K-1} sign(w_i[n]).w[n] - z = 0. \tag{12}$$

This hyperplane is the side of the epigraph set C facing the vector $\mathbf{w}_i$. The projection vector $\widetilde{\mathbf{w}}_{pi}$ onto the hyperplane (12) in $\mathbb{R}^{K+1}$ is determined as in (3). To solve this minimization problem, the Lagrangian is constructed as follows:

$$\mathcal{L}(\mathbf{w}) = \|\mathbf{w}_i - \mathbf{w}\|_2^2 + \lambda\left(\sum_{n=0}^{K-1} sign(w_i[n]).w[n] - z\right), \tag{13}$$

where λ is the Lagrange multiplier. Differentiating with respect to $w[n]$, z, and λ and setting the result to zero, we obtain:

$$\frac{d\mathcal{L}(w)}{dw} = -2(w_i[n] - w[n]) + \lambda(sign(w_i[n])) = 0, \tag{14}$$

$$\frac{d\mathcal{L}(w)}{dz} = 2w[K] - \lambda = 0, \tag{15}$$

and the derivative of $\mathcal{L}(\mathbf{w})$ with respect to λ produces Eq. (12) as conditions to be satisfied. The projection vector can be also determined using the intersection of the line going through w_i along the surface normal of the hyperplane (12) and the epigraph set C without using the Lagrangian concept. Therefore, the obtained solution is

$$w_{pi}[n] = w_i[n] - \frac{\sum_{n=0}^{K-1} |w_i[n]|}{K+1} sign(w_i[n]) \quad n = 0, 1, \dots, K-1, \tag{16}$$

which we replaced $sign(w_i[n])w_i[n]$ with $|w_i[n]|$ to make equations more readable. The last component of $\widetilde{\mathbf{w}}_{pi}$ is given by

$$z_p = \frac{\sum_{n=0}^{K-1} sign(w_i[n])w_i[n]}{K+1} = \frac{\sum_{n=0}^{K-1} |w_i[n]|}{K+1}. \tag{17}$$

In Figure 1, $\widetilde{\mathbf{w}}_{pi} = [w_{pi}[0], w_{pi}[1], \dots, w_i[K-1], 0]^T$ which is the K dimensional version of w_{pi}. This orthogonal projection operation also determines the size of the ℓ_1-ball $d_i = z_p$, which can be verified using the geometry. The projection operation is graphically illustrated in Fig. 1 for $K = 2$. The projection

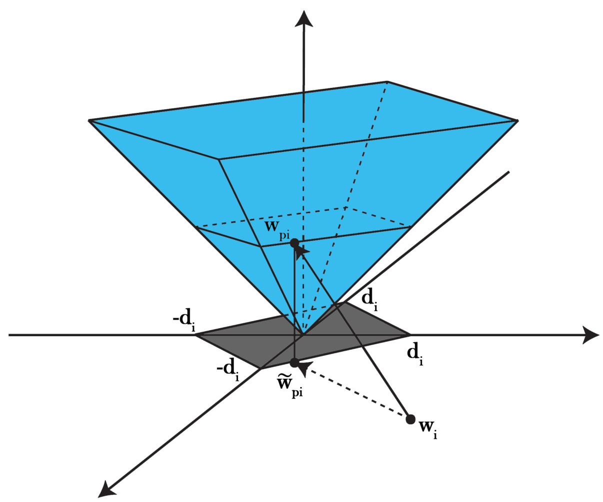


Fig. 1. Projection of $w_i[n]$ onto the epigraph set of ℓ_1-norm cost function: $C = \{\mathbf{w} : \sum_{n=0}^{K-1} |w[k]| \leq z\}$, gray shaded region

vector $\mathbf{w}_{pi}$ is the projection of $\mathbf{w}_i$ onto the hyperplane described in Eq. (12). It may or may not be the projection of $\mathbf{w}_i$ onto the epigraph set C as shown in Fig. 2 (view from top). When we project the vector $\mathbf{w}_l$ the projection vector $\mathbf{w}_{pl}$ turns out to be on another quadrant of $\mathbb{R}^{K+1}$. Therefore, $\mathbf{w}_{pl}$ is not on an ℓ_1-ball. We can easily determine if $\mathbf{w}_{pi}$ is the correct projection or not by checking the signs of the entries of $\mathbf{w}_{pi}$. If the signs of the projection vector entries $w_{pi}[n]$ are the same as $w_i[n]$ for all n then the $w_{pi}[n]$ is on the epigraph set C, otherwise $w_{pi}[n]$ is not on the ℓ_1-ball as shown in Fig. 2. If $w_{pi}[n]$ is not on the ℓ_1-ball we can still project w_i onto the ℓ_1-ball using Algorithm 1 or Duchi et al's ℓ_1-ball projection algorithm [10] using the value of $d_i = z_p$ determined in Eq. (17). The value of d_i is the same whether $\mathbf{w}_{pi}$ is on the ℓ_1-ball or not as shown in Fig. 2, because d_i turns out to be the intersection of the hyperplane with one of the axis of $\mathbb{R}^K$. This constitutes the second step of the projection operation onto the epigraph set C.

In summary, we have the following two steps: (i) Project $\mathbf{w}_i$ onto the boundary hyperplane and determine d_i. (ii) If $sign(w_i[n]) = sign(w_{pi}[n])$ for all n, $\mathbf{w}_{pi}$ is the projection vector. Otherwise use d_i value in Algorithm 1 to determine the final projection vector. Since we have $i = 1, 2, \dots, L$ wavelet subsignals, we should project each wavelet subsignal w_i onto possibly distinct ℓ_1-balls with sizes d_i, respectively. Notice that d_i is not the value of the soft-threshold, it is the size of the ℓ_1-ball. The value of the soft-threshold is detected using Algorithm 1. In the next subsection we describe a method to determine the number of wavelet decomposition level L.

V. How to Determine The Number of Wavelet Decomposition Levels

It is possible to use the Fourier transform of the noisy signal to estimate the bandwidth of the signal. Once the bandwidth ω_0 of the original signal is approximately determined it can be used to estimate the number of wavelet transform levels and the bandwidth of the low-band signal x_L. In an L-level wavelet decomposition the low-band signal x_L approximately comes from the $[0, \frac{\pi}{2^L}]$ frequency band of the signal $x[n]$. Therefore, $\frac{\pi}{2^L}$ must be greater than ω_0 so that the actual signal components are not

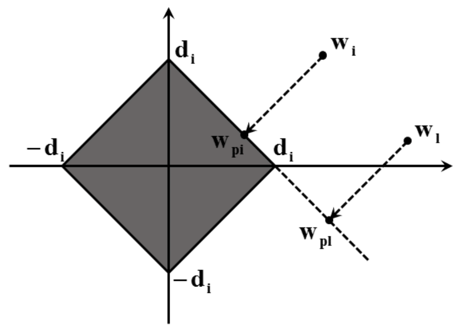


Fig. 2. Orthogonal projection operation onto a bounding hyperplane of ℓ_1-ball.

soft-thresholded. Only wavelet subsignals $w_L[n], w_{L-1}[n], \ldots, w_1[n]$, which come from frequency bands $[\frac{\pi}{2^L}, \frac{\pi}{2^{L-1}}]$, $[\frac{\pi}{2^{L-1}}, \frac{\pi}{2^{L-2}}]$, $\ldots$, $[\frac{\pi}{2}, \pi]$, respectively, should be soft-thresholded in denoising. For example,

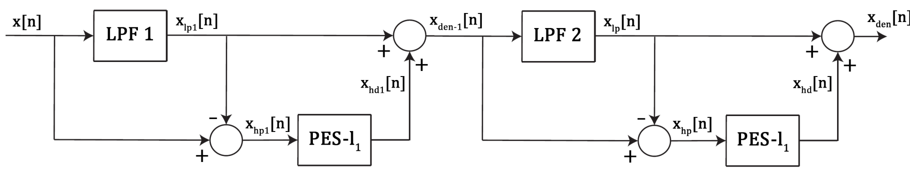


Fig. 3. Pyramidal filtering based denoising. the high-pass filtered signal is projected onto the epigraph set of ℓ_1.

in Fig. 4, the magnitude of Fourier transform of $x[n]$ is shown for "piece-regular" signal defined in MATLAB. This signal is corrupted by zero-mean white Gaussian noise with $\sigma = 10, 20$, and 30% of the maximum amplitude of the original signal, respectively. For this signal an $L = 3$ level wavelet decomposition is suitable because Fourier transform magnitude approaches to the noise floor level after $\omega_0 = \frac{58\pi}{512}$. It is also a good practice to allow a margin for signal harmonics. Therefore, $(\frac{\pi}{2^3} > \frac{58\pi}{512})$ is selected as the number of wavelet decomposition levels. It is also possible to use a pyramidal structure for

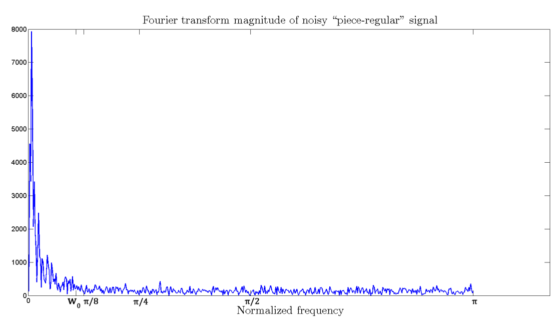
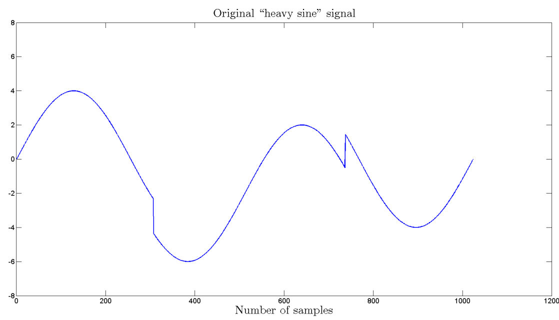
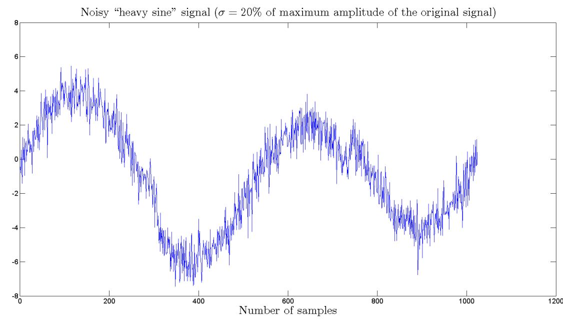


Fig. 4. Discrete-time Fourier transform magnitude of "piece-regular" signal corrupted by noise. The wavelet decomposition level L is selected as 3 to satisfy $\frac{\pi}{2^3} > \omega_0$, which is the approximate bandwidth of the signal.

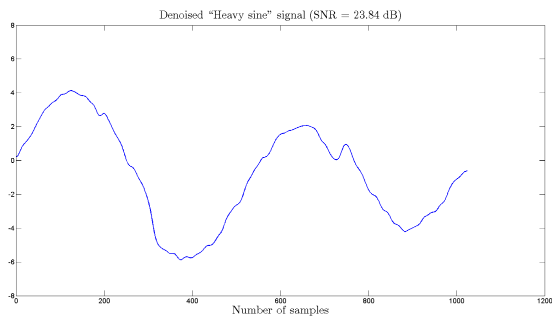
signal decomposition instead of the wavelet transform. The noisy signal is low-pass filtered with cut-off frequency $\frac{\pi}{8}$ for "piece-regular" signal and the output $x_{lp}[n]$ is subtracted from the noisy signal $x[n]$ to obtain the high-pass signal $x_{hp}[n]$ as shown in Fig. 3. The signal is projected onto the epigraph of ℓ_1-ball and $x_{hd}[n]$ is obtained. Projection onto the Epigraph Set of ℓ_1-ball (PES-ℓ_1), removes the noise by soft-thresholding. The denoised signal $x_{den}[n]$ is reconstructed by adding $x_{hd}[n]$ and $x_{lp}[n]$ as shown in Fig. 3. It is possible to use different thresholds for different subbands as in wavelet transform, using a multisatge pyramid as shown in Fig. 3. In the first stage a low-pass filter with cut-off $\frac{\pi}{2}$ can be used and $x_{hp1}[n]$ is projected onto the epigraph set of ℓ_1-ball producing a threshold for the subband $[\frac{\pi}{2}, \pi]$. In the second stage, another low-pass filter with cut-off $\frac{\pi}{4}$ can be used and $x_{hp}[n]$ is projected onto the epigraph set producing a threshold for $[\frac{\pi}{4}, \frac{\pi}{2}]$, etc.



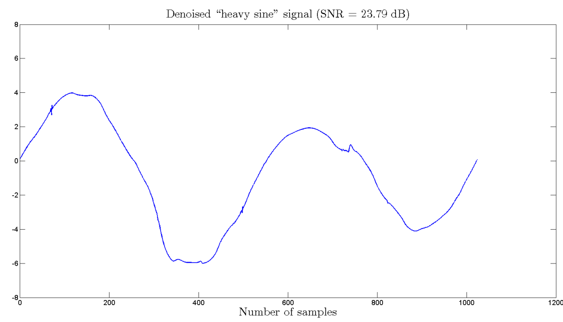
(a) Original signal



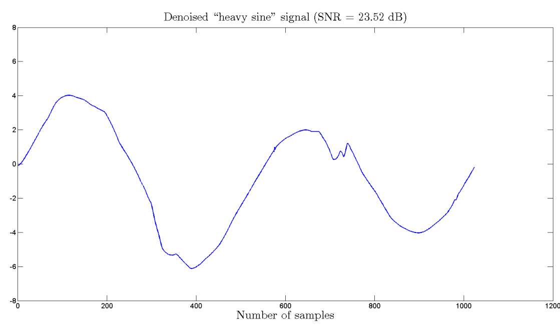
(b) Noisy signal



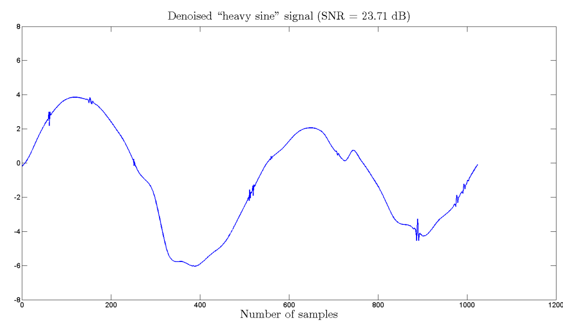
(c) PES-ℓ_1 with pyramid



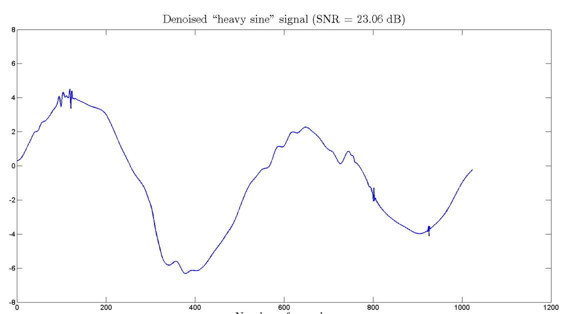
(d) PES-ℓ_1 with wavelet



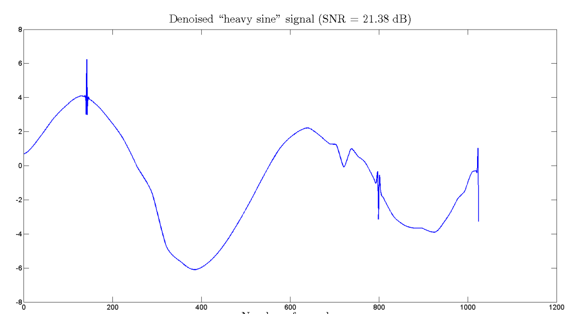
(e) Wavelet denoising in MATLAB [3**?**]



(f) Wavelet denoising “minimaxi” algorithm [2]



(g) Wavelet denoising “rigrsure” algorithm [5]

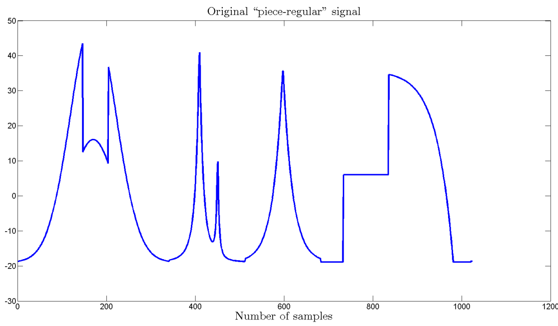


(h) Wavelet denoising with T = $3\hat{\sigma}$ [2, 9]

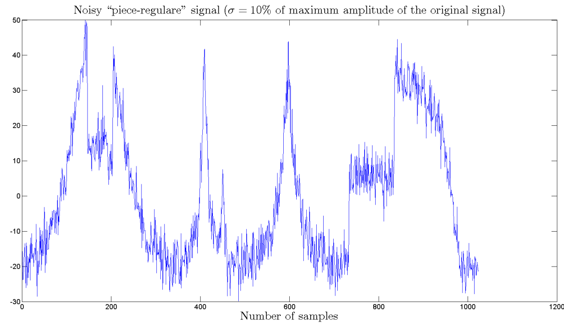
Fig. 5. (a) Original “heavy sine” signal, (b) signal corrupted with Gaussian noise with $\sigma = 20\%$ of maximum amplitude of the original signal, and denoised signal using (c) PES-ℓ_1-ball with pyramid; SNR = 23.84 dB and, (d) PES-ℓ_1-ball with wavelet; SNR = 23.79 dB, (e) Wavelet denoising in Matlab; SNR = 23.52 dB [3**?**], (f) Wavelet denoising “minimaxi” algorithm [2]; SNR = 23.71 dB, (g) Wavelet denoising “rigrsure” algorithm [5]; SNR = 23.06 dB, (h) Wavelet denoising with T = $3\hat{\sigma}$ [2, 9]; SNR = 21.38 dB.

VI. Simulation Results

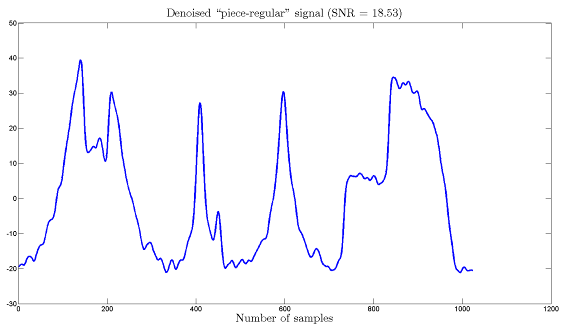
Epigraph set based threshold selection is compared with wavelet denoising methods used in MATLAB [2**?** –4]. The “heavy sine” signal shown in Fig. 7(a) is corrupted by a zero mean Gaussian noise with $\sigma =$



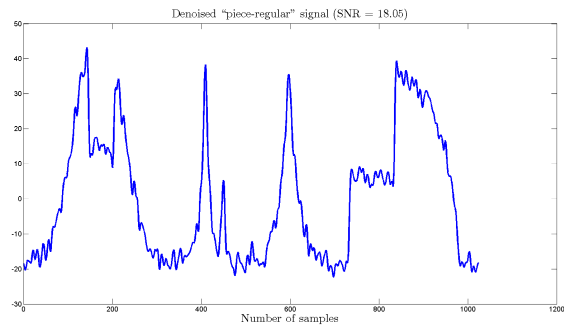
(a) Original signal



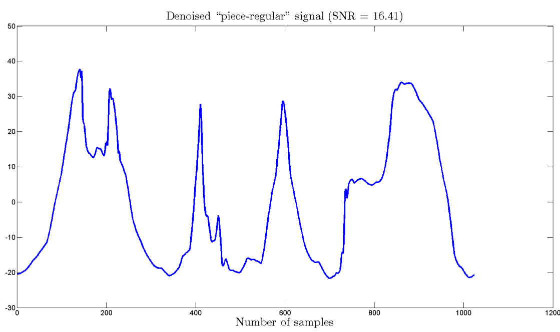
(b) Noisy signal



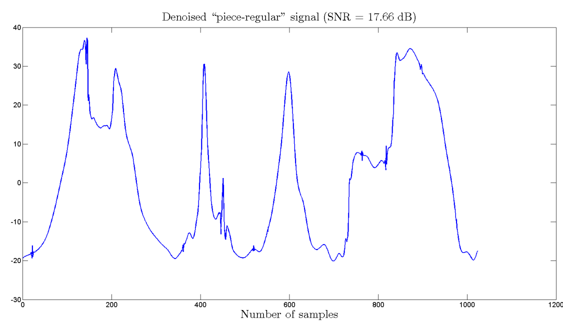
(c) PES-ℓ_1 with pyramid



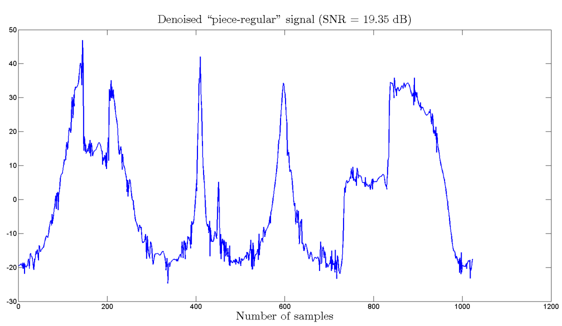
(d) PES-ℓ_1 with wavelet



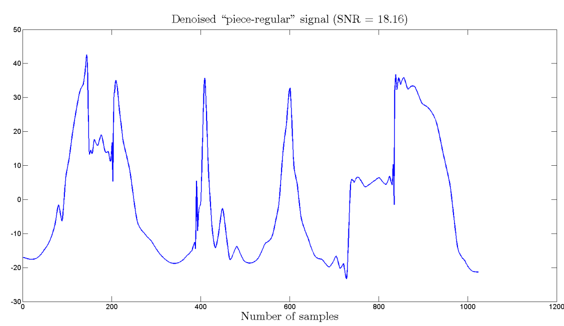
(e) Wavelet denoising in MATLAB [3**?**]



(f) Wavelet denoising "minimaxi" algorithm [2]



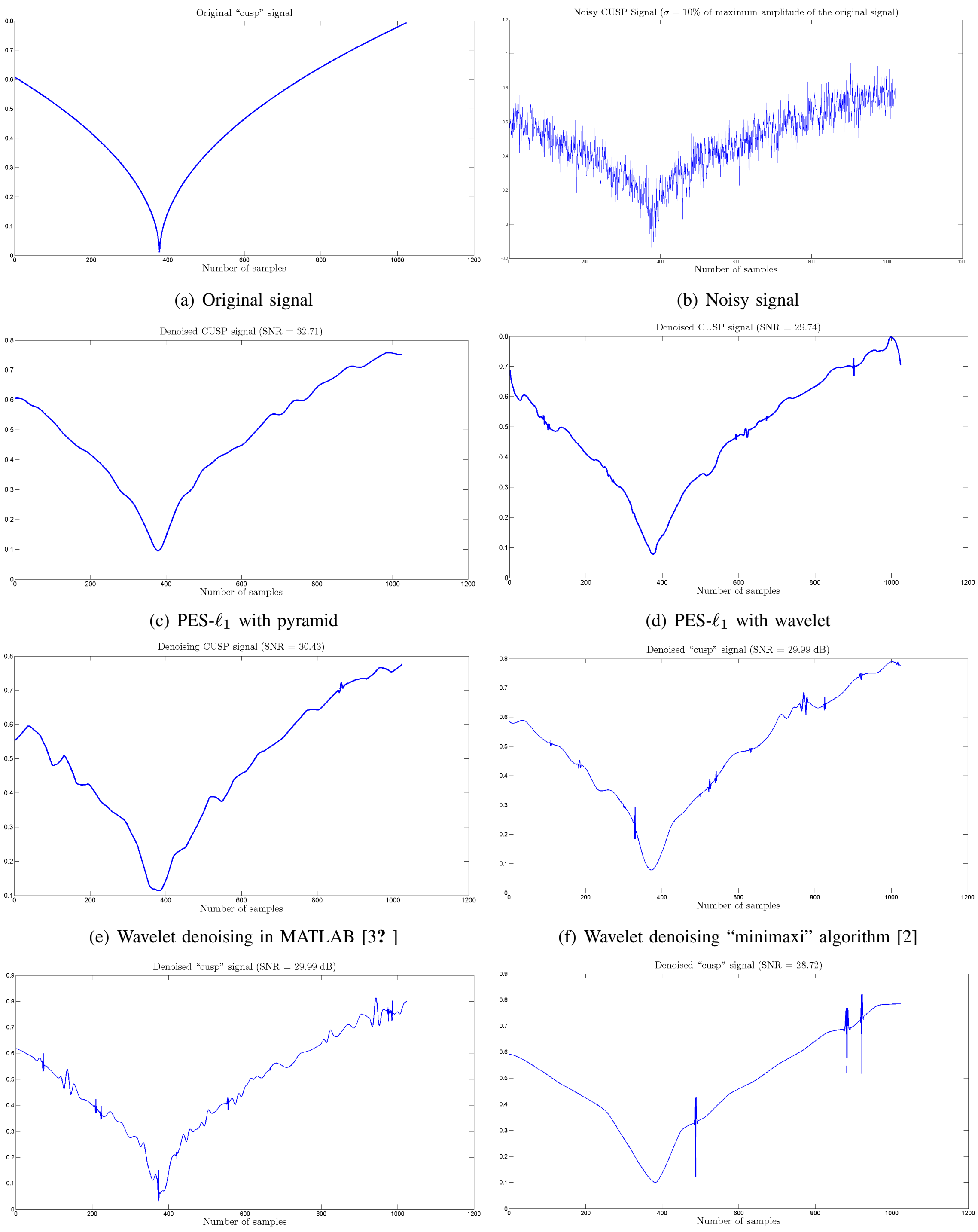
(g) Wavelet denoising "rigrsure" algorithm [5]



(h) Wavelet denoising with T = $3\hat{\sigma}$ [2, 9]

Fig. 6. (a) Original "cusp" signal, (b) signal corrupted with Gaussian noise with $\sigma = 10\%$ of maximum amplitude of the original signal, and denoised signal using (c) PES-ℓ_1-ball with pyramid; SNR = 23.84 dB and, (d) PES-ℓ_1-ball with wavelet; SNR = 23.79 dB, (e) Wavelet denoising in Matlab; SNR = 23.52 dB [3**?**], (f) Wavelet denoising "minimaxi" algorithm [2]; SNR = 23.71 dB, (g) Wavelet denoising "rigrsure" algorithm [5]; SNR = 23.06 dB, (h) Wavelet denoising with T = $3\hat{\sigma}$ [2, 9]; SNR = 21.38 dB.

20% of the maximum amplitude of the original signal. The signal is restored using PES-ℓ_1 with pyramid structure, PES-ℓ_1 with wavelet, MATLAB's wavelet multivariate denoising algorithm [3**?**], MATLAB's



(a) Original signal

(b) Noisy signal

(c) PES-ℓ_1 with pyramid

(d) PES-ℓ_1 with wavelet

(e) Wavelet denoising in MATLAB [3?]

(f) Wavelet denoising "minimaxi" algorithm [2]

(g) Wavelet denoising "rigrsure" algorithm [5]

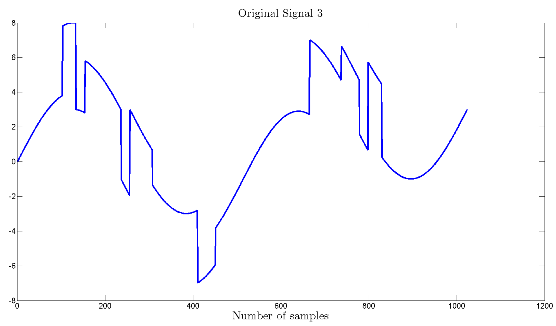
(h) Wavelet denoising with T = $3\hat{\sigma}$ [2, 9]

Fig. 7. (a) Original "cusp" signal, (b) signal corrupted with Gaussian noise with $\sigma = 10\%$ of maximum amplitude of the original signal, and denoised signal using (c) PES-ℓ_1-ball with pyramid; SNR = 23.84 dB and, (d) PES-ℓ_1-ball with wavelet; SNR = 23.79 dB, (e) Wavelet denoising in Matlab; SNR = 23.52 dB [3?], (f) Wavelet denoising "minimaxi" algorithm [2]; SNR = 23.71 dB, (g) Wavelet denoising "rigrsure" algorithm [5]; SNR = 23.06 dB, (h) Wavelet denoising with T = $3\hat{\sigma}$ [2, 9]; SNR = 21.38 dB.

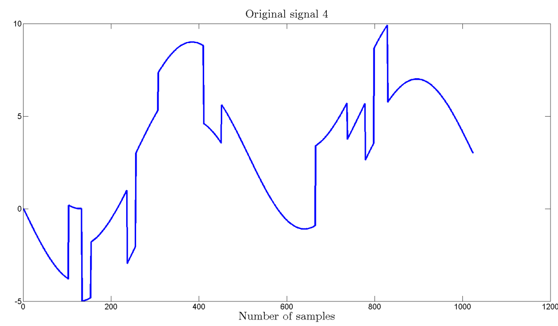
soft-thresholding denoising algorithm (for "minimaxi" and "rigrsure" thresholds), and wavelet thresholding denoising method. The denoised signals are shown in Fig. 7(c), 7(d), 7(e), 7(f), 7(g), and 7(h) with

SNR values equal to 23.84, 23.79, 23.52, 23.71, 23.06 dB, and 21.38, respectively. On the average, the proposed PES-ℓ_1 with pyramid and PES-ℓ_1 with wavelet method produce better thresholds than the other soft-thresholding methods. MATLAB codes of the denoising algorithms and other simulation examples are available in the following web-page: http://signal.ee.bilkent.edu.tr/1DDenoisingSoftware.html.

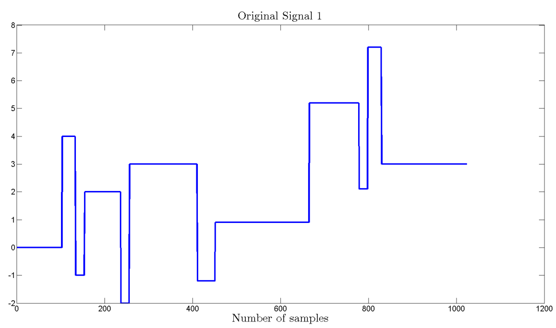
Results for other test signals in MATLAB are presented in Table I. These results are obtained by averaging the SNR values after repeating the simulations for 300 times. The SNR is calculated using the formula: SNR $= 20 \times log_{10}(\|\mathbf{w}_{orig}\|/\|\mathbf{w}_{orig} - \mathbf{w}_{rec}\|)$. In this lecture note, it is shown that soft-denoising threshold can be determined using basic linear algebra.



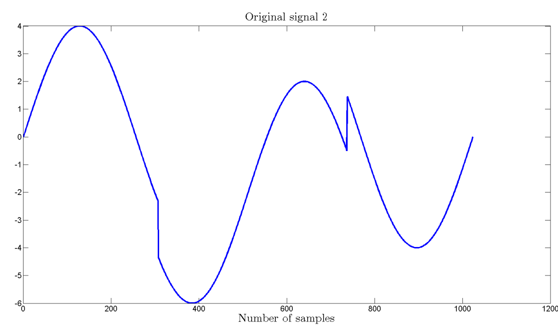
(a) Signal 1



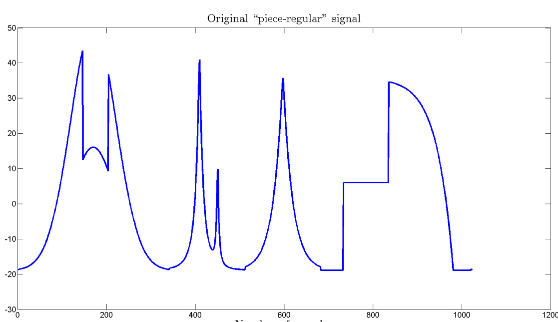
(b) Signal 2



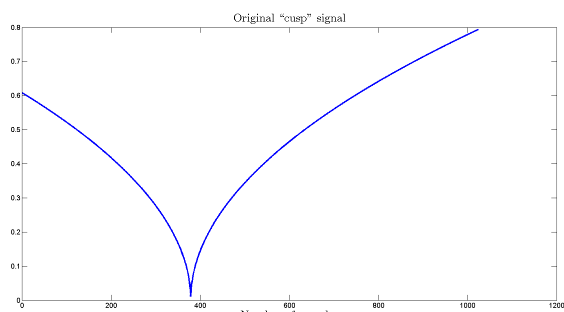
(c) Blocks



(d) Heavy sine



(e) Piece-regular



(f) CUSP

Fig. 8. Signals which are used in the simulations.

TABLE I
COMPARISON OF THE RESULTS FOR DENOISING ALGORITHMS WITH GAUSSIAN NOISE WITH $\sigma = 10$, 20, AND 30 % OF MAXIMUM AMPLITUDE OF ORIGINAL SIGNAL.

Signal	Input SNR (dB)	**PES-ℓ_1 Pyramid**	PES-ℓ_1 Wavelet	MATLAB [3**?**]	Soft-threshold $3\hat{\sigma}$	MATLAB "rigrsure" [5]	MATLAB "mimimaxi" [2]
Blocks	12.30	17.27	17.08	15.73	17.64	18.32	16.59
Heavy sine	17.77	26.17	26.62	26.87	26.22	27.82	27.75
Signal 1	13.09	18.43	18.10	16.63	17.80	19.18	17.41
Signal 2	13.83	20.37	19.94	18.39	18.92	20.53	19.08
Piece-Regular	12.32	18.53	18.05	16.41	18.16	19.35	17.66
CUSP	16.29	32.58	29.40	30.43	28.72	29.10	29.99
Blocks	6.28	14.34	13.98	12.92	12.87	14.18	13.43
Heavy sine	11.75	23.84	23.79	23.52	21.38	23.06	23.71
Signal 1	7.07	15.70	15.28	14.00	13.30	15.15	14.42
Signal 2	7.80	17.13	17.07	15.84	14.56	16.65	16.20
Piece-Regular	6.27	15.24	14.47	13.11	13.14	14.94	13.99
CUSP	10.25	28.24	24.89	25.04	23.27	23.48	24.44
Blocks	2.76	12.52	12.55	11.37	10.13	12.05	11.64
Heavy sine	9.20	20.89	21.78	21.32	18.79	20.17	21.05
Signal 1	3.56	13.50	13.65	12.37	10.44	13.06	12.72
Signal 2	4.26	15.14	14.25	14.06	12.05	14.37	14.30
Piece-Regular	2.77	13.21	12.70	11.37	9.94	12.43	12.05
CUSP	6.73	25.10	23.47	21.73	19.67	19.69	21.02
Average	9.13	**19.68**	18.73	17.84	17.06	18.53	18.19

MATLAB CODE

PES-ℓ_1 with pyramid method

In the codes for PES-ℓ_1 with pyramid method, first it starts with loading the original signals as:

```
x_orig = zeros(1024,6);
load ex4mwden
x_orig(:, 5) = load_signal('Piece-Regular', 1024);
x_orig(:, 6) = load_signal('cusp', 1024);
```

Then the white Gaussian noise is added as below:

```
    amp_perc = 0.1;
    for m = 1:kk
        sigma(m) = amp_perc*max(x_orig(:, m));
        NoisySignal(:, m) = x_orig(:, m) + sigma(m)*randn(size(x_orig
            (:, m)));
    end
```

which the noise standard deviation is determined with “amp_perc” which in our software it is 0.1, 0.2, and 0.3. Then the iteration number is determined according to the noise power. After all the signal will enter the denoising algorithm which is applied to the noisy signal in “PES_L1_Pyramid.m” function, therefore:

```
    DenoisedSignal = PES_L1_Pyramid(iter, NoisySignal, kk);
```

which “iter” is the number of the iterations, “NoisySignal” is the corrupted signal, and “kk” is the number of the signals, which here we have six signals in our simulations. In the main function of PES-ℓ_1 denoising “PES_L1_Pyramid.m”, first the signal is passed through the high-pass filter and signal’s high and low frequencies are separated, and then the PES-ℓ_1 algorithm is applied to the high-passed signal. After that, the denoised high-pass signal is added to the unchanged low-pass signal and the main denoised signal is obtained. AS mentioned above, the performance of the algorithms are evaluated by SNR. which are calculated as:

```
    for d = 1:kk
        SNR_in(d) = snr(x_orig(:,d), NoisySignal(:,d));
        SNR_out(d) = snr(x_orig(:,d), DenoisedSignal(:,d));
    end
```

Since the additive noise is random then we have to run the codes repeatedly for enough times and average them to get the rational SNR value. Which averaging is done as:

```
SNR_In_Ave = mean(SNR_in1, 1)
SNR_Out_Ave = mean(SNR_out1, 1)
```

Then all the signals (original, noisy, and denoised) are ploted as:

```
kp = 0;
figure(2)
for f = 1:kk
    subplot(kk,3,kp+1), plot(x_orig(:,f)); axis tight;
    title(['Original signal ',num2str(f)])
    subplot(kk,3,kp+2), plot(NoisySignal(:,f)); axis tight;
    title(['Observed signal ',num2str(f)])
    subplot(kk,3,kp+3), plot(DenoisedSignal(:,f)); axis tight;
    title(['Denoised signal ',num2str(f)])
    kp = kp + 3;
end
```

The resulting plot are: and the SNRs are:

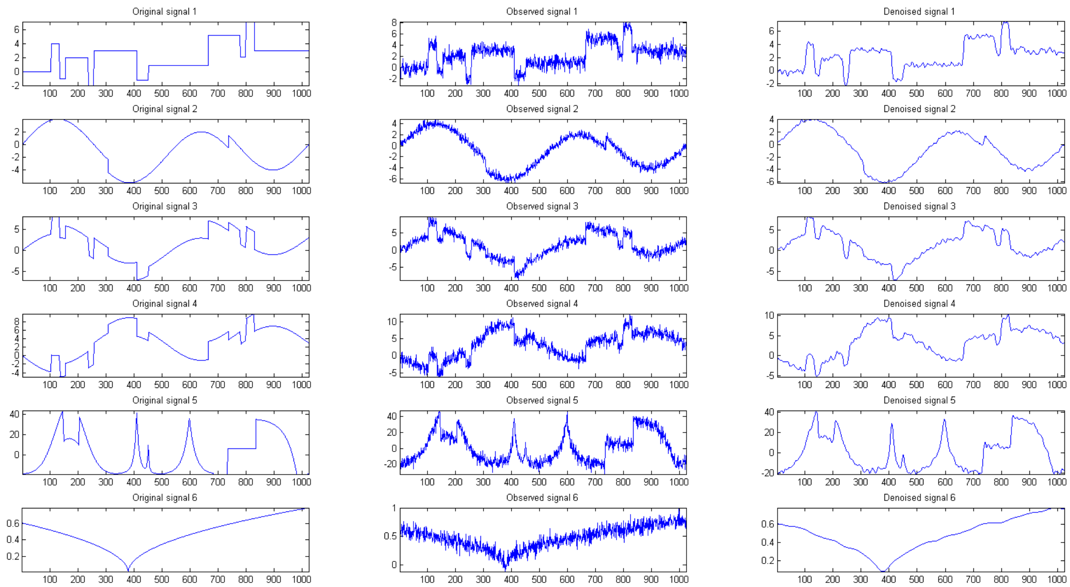


Fig. 9. All the original, noisy, and denoised signals for PES-ℓ_1 with pyramid method.

```
SNR_In_Ave =

    6.2994    11.7275    7.0773    7.7911    6.2919    10.2356


SNR_Out_Ave =

   14.3881    23.8448    15.6371    17.1841    15.3298    28.1067


ans =

   19.0818
```

PES-ℓ_1 with wavelet method

All the preliminary steps for this codes are as the same for PES-ℓ_1 with pyramid method. Here, instead of "PES_L1_Pyramid.m", the function for PES-ℓ_1 with wavelet method "PES_L1_Wavelet.m" is used. In this function the "farras" filter bank is used for wavelet decomposition and the decomposition level is determined as explained in previous sections. It is done as:

```
        x_dwt1 =  circshift(NoisySignal(:, k), f-1); % Shifting the
            signal
        [af, sf] = farras; % Wavelet transforms filters
        J = Jk(k); % Decomposition level
        x_dwt = dwt_C(x_dwt1,J,af); % Wavelet decomposition
```

then the high subsignals are denoised with PES-ℓ_1 algorithm and the low subsignal is transfered to the output without without any change, then the main denoised signal is reconstructed as follows;

```
        % Reconstructing the signal from its subsignals
        im_den(J+1) = x_dwt(J+1);
        x_idwt1 = idwt_C(im_den, J, sf);
        x_idwt2(:, f) = circshift(x_idwt1, -f+1); % Shift back
```

again the SNR is calculated as before, and the signals are plotted as follows:

```
kp = 0;
figure
for f = 1:kk
    subplot(kk,3,kp+1), plot(x_orig(:,f)); axis tight;
    title(['Original signal ',num2str(f)])
    subplot(kk,3,kp+2), plot(NoisySignal(:,f)); axis tight;
    title(['Observed signal ',num2str(f)])
    subplot(kk,3,kp+3), plot(DenoisedSignal(:,f)); axis tight;
    title(['Denoised signal ',num2str(f)])
    kp = kp + 3;
end
```

and the SNR values are averaged as before and the signals are plotted as following figure: and the SNRs

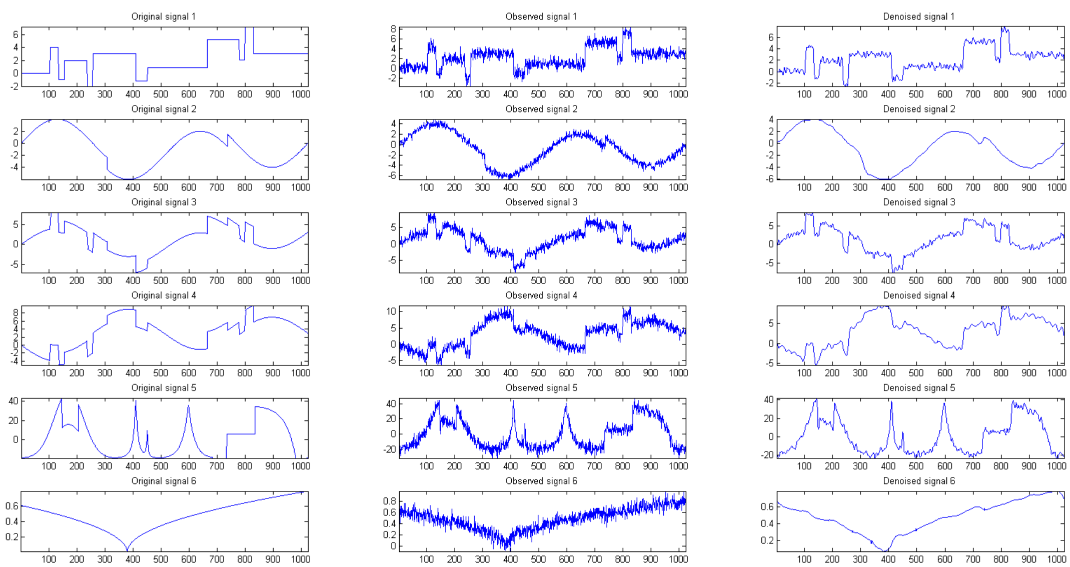


Fig. 10. All the original, noisy, and denoised signals for PES-ℓ_1 with pyramid method.

are:

```
    SNR_In_Ave =

      6.2991    11.7385    7.0891    7.7825    6.2741    10.2704


SNR_Out_Ave =

    13.9855    23.7737    15.2842    17.0870    14.4843    24.8300


ans =

    18.2408
```

References

[1] S. Mallat and W.-L. Hwang, "Singularity detection and processing with wavelets," *Information Theory, IEEE Transactions on*, vol. 38, no. 2, pp. 617–643, Mar 1992.

[2] D. Donoho, "De-noising by soft-thresholding," *Information Theory, IEEE Transactions on*, vol. 41, no. 3, pp. 613–627, May 1995.

[3] M. Aminghafari, N. Cheze, and J.-M. Poggi, "Multivariate denoising using wavelets and principal component analysis," *Computational Statistics and Data Analysis*, vol. 50, no. 9, pp. 2381 – 2398, 2006.

[4] S. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression," *IEEE Transactions on Image Processing*, vol. 9, no. 9, pp. 1532–1546, Sep 2000.

[5] D. L. Donoho and I. M. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1200–1224, 1995.

[6] G. Chierchia, N. Pustelnik, J.-C. Pesquet, and B. Pesquet-Popescu, "An epigraphical convex optimization approach for multicomponent image restoration using non-local structure tensor," in *IEEE ICASSP, 2013*, 2013, pp. 1359–1363.

[7] M. Tofighi, K. Kose, and A. Cetin., "Denoising using projections onto the epigraph set of convex cost functions," in *IEEE International Conference on Image Processing (ICIP'14).*, Oct 2014.

[8] G. Chierchia, N. Pustelnik, J.-C. Pesquet, and B. Pesquet-Popescu, "Epigraphical projection and proximal tools for solving constrained convex optimization problems: Part i," *Arxiv, CoRR*, vol. abs/1210.5844, 2012.

[9] J. Fowler, "The redundant discrete wavelet transform and additive noise," *IEEE Signal Processing Letters*, vol. 12, no. 9, pp. 629–632, Sept 2005.

[10] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l1-ball for learning in high dimensions," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 272–279.