Lab #4
Wavelet-based signal processing basics

Raed Suftah

110-44-889

[1] **Compression.** To achieve data compression, high-frequency wavelet coefficients can be removed. Compare the original and reconstructed signals with respect to different compression ratios. Use a l_2 norm to evaluate the reconstruction performance.

Compression ratio = original data size / compressed data size

Ans:

I use Compression ratio $1024/2^8$, $1024/2^6$, $1024/2^4$ to compare signals.

The l^2 norm is: $|s - s\_recon|_2 = \sqrt{\sum_i^N (s_i - s\_recon_i)^2}$. The smaller l^2 norm the better reconstruction signal is.

Compression Ratio	**$1024/2^8$**	**$1024/2^6$**	**$1024/2^4$**
L^2 norm	150.6249	392.1187	684.4828

Matlab code:

```
load test_eeg
n = 1024;
% filter bank
filter = MakeONFilter('Haar')
% wavelet matrix
W1 = WavMat(filter,n,1);
W = W1^9;
%original signal
xx0 = aa(1,1:n);
% compute the signal coefficients
w0 = W*xx0';
% reconstruct the signal
x0 = W'*w0;
% compression operation
[X Y] = wavedec(xx0,8,'db8')
out = appcoef(X,Y,'db8',1);
% Plot the figure
subplot(211)
plot(xx0)
title('Original signal')
subplot(212)
plot(out)
```

```
title('compressed signal level1')
```

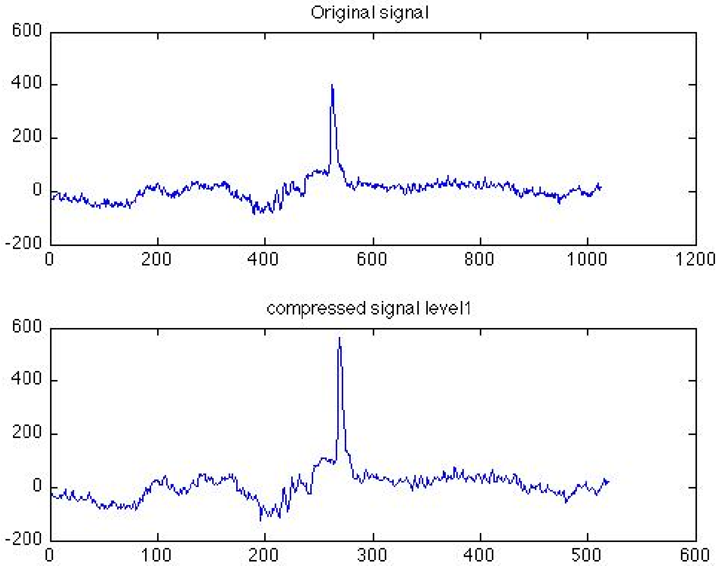


Figure 1: Compressed Signal Level 1

Compression ratio = original data size / compressed data size

1024/519=1.973025

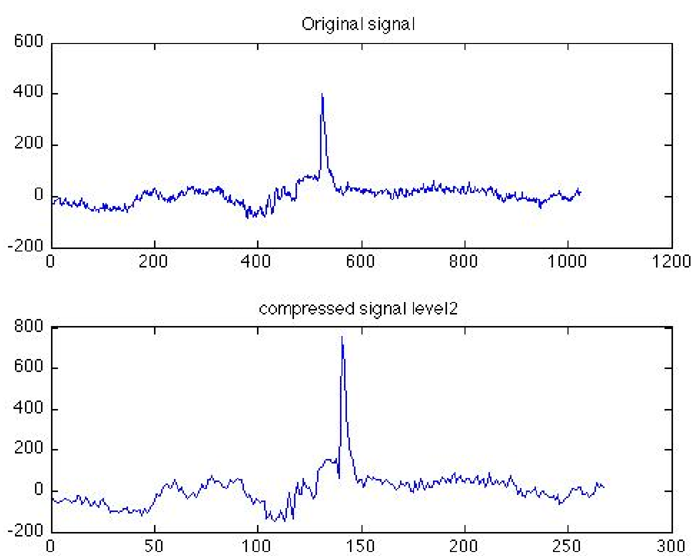


Figure 2: Compressed Signal Level 2

Compression ratio = original data size / compressed data size

1024/267=3.83520

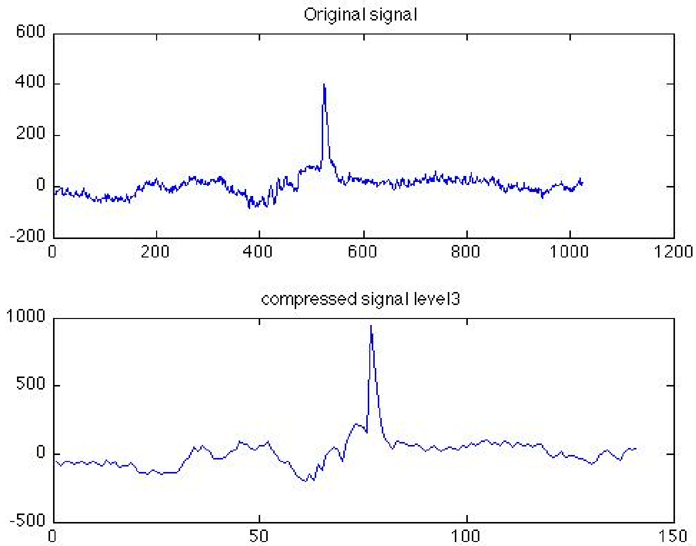


Figure 3: Compressed Signal Level 3

Compression ratio = original data size / compressed data size

1024/141=7.262411

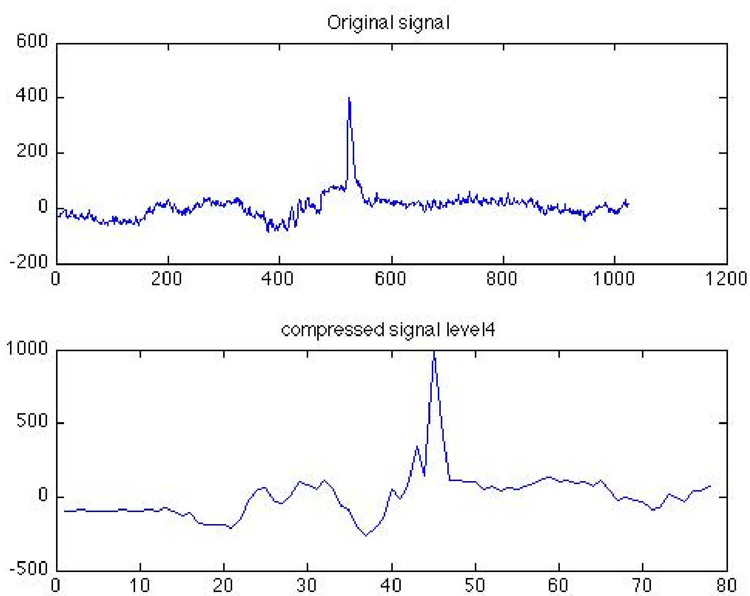


Figure 4: Compressed Signal Level 4

Compression ratio = original data size / compressed data size

1024/78=13.128205

[2] **Denoising**. Add noise to the original signal, assign zero values to the high-frequency wavelet coefficients, and reconstruct the smoothed signal. Compare the original and smoothed signals with respect to different signal-to-noise ratios and different compression ratios. Use the signal distortion ratio to evaluate the denoising performance.

Signal-to-noise ratio = power of original signal / power of noise

Signal distortion ratio = power of signal distortion / power of original signal

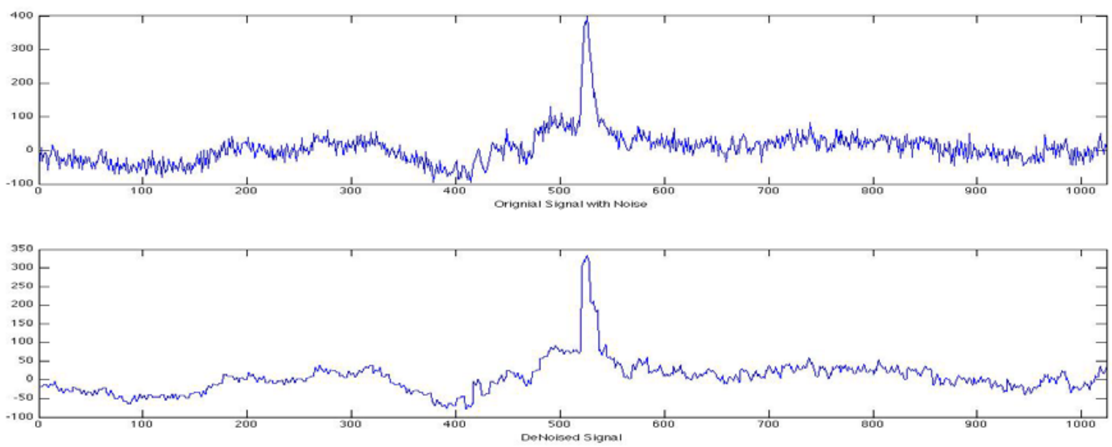
I use SNR: 10 dB, 20 dB, 30 dB and keep the compression ratio the same as previous problem. When SNR increases, the signal is more noisy.

The Signal distortion ratio $= \frac{P_{distortion}}{P_{signal}} = 10\log_{10}\left(\frac{A_{distortion}}{A_{signal}}\right)^2 = 20\log_{10}\frac{A_{distortion}}{A_{signal}}$, where A is the root mean square (RMS). The larger signal distortion ratio, the better denoiding performance is. However, the more high frequency information is lost.

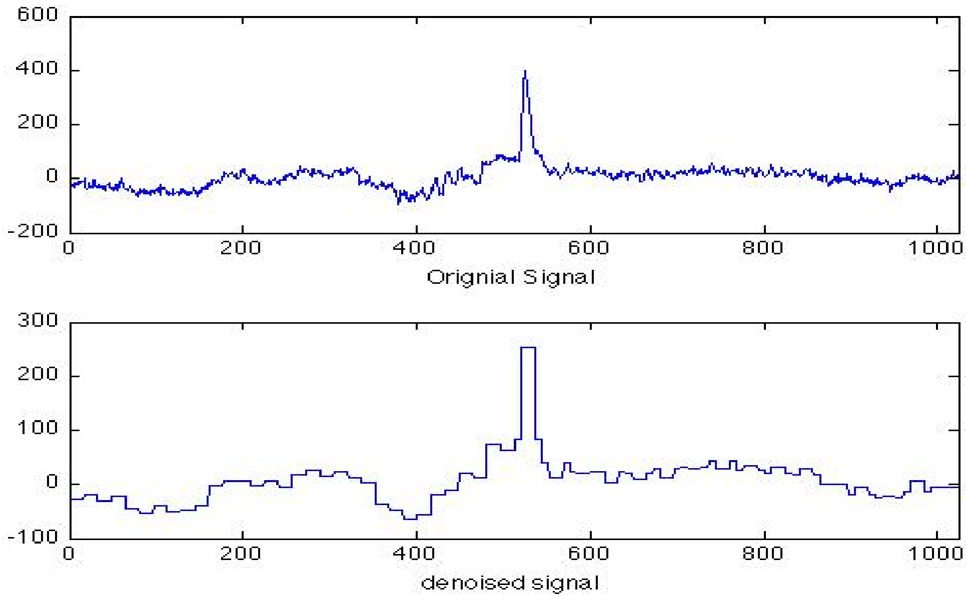
MatLab code :

```
load test_eeg
n = 1024;
% filter bank
filter = MakeONFilter('Haar')
% wavelet matrix
W1 = WavMat(filter,n,1);
W = W1^9;
%original signal
xx0 = aa(1,1:n);
% compute the signal coefficients
w0 = W*xx0';
% reconstruct the signal
x0 = W'*w0;

%% Addition of noise. By adding SNRs
xx1 = awgn(xx0,5,'measured');

% compute the DWT coefficients for the noisy signal
w1 = W*xx1';
w2 = w1;

% remove the coefficients lower than a certain threshold
w2(abs(w1)< 100) = 0;
```

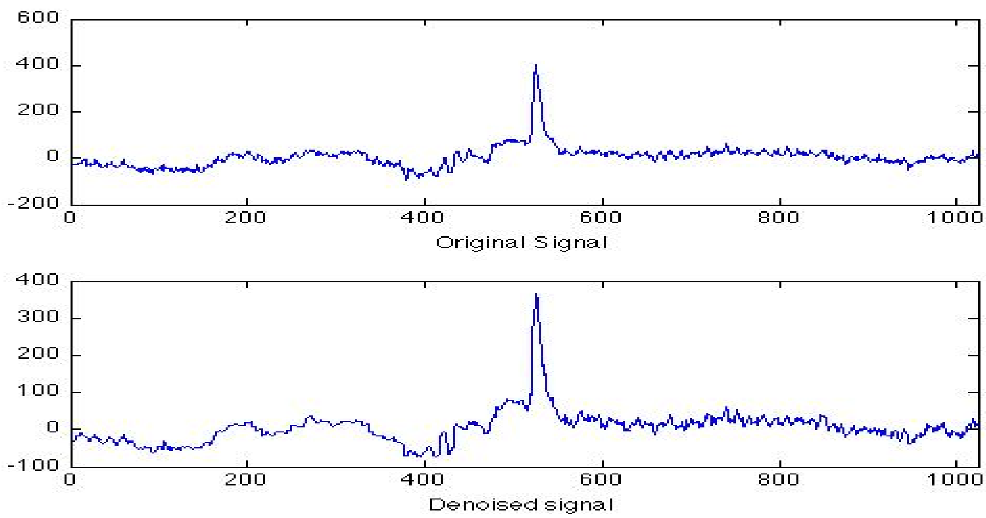
```
% reconstruct the original and the noisy signal from the DWT coefficients
x1 = W'*w1;
x2 = W'*w2;

 % plot the figures
subplot(211)
plot(x1)
xlim([0 n])
subplot(212)
plot(x2)
xlim([0 n])


% power of the signal and the distorted signal:
px = sum(xx0.^2)/n;

pxn = sum(xx1.^2)/n;

% signal distortion ratio
sgDis = pxn / px
```



Threshold	SNR	Power of signal with noise	Distortion ratio
100	5	3.0194426e+03	1.2878



Threshold	SNR	Power of signal with noise	Distortion ratio
50	10	2.4108030e+03	1.0663



Threshold	SNR	Signal-to-noise ratio	Distortion ratio
150	15	2.3372783e+03	1.0338