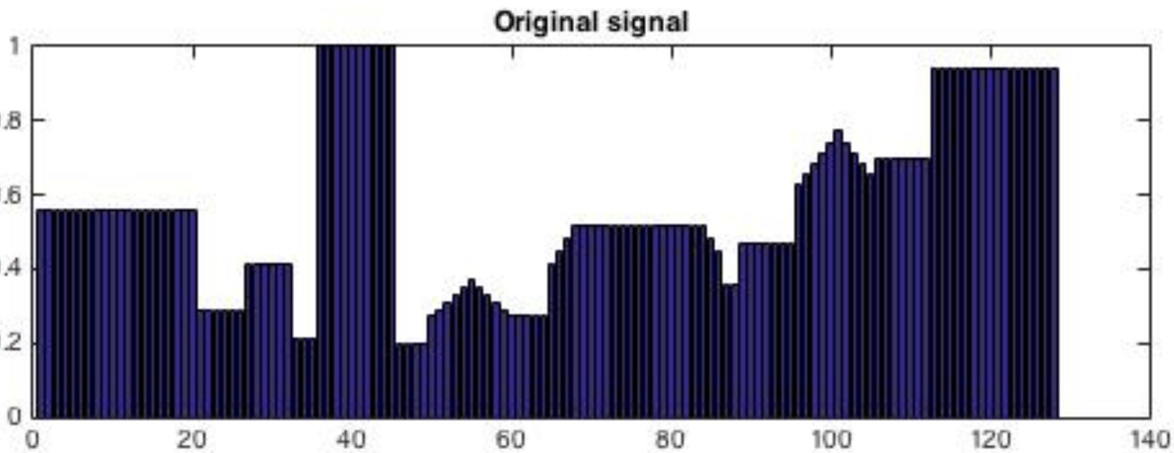
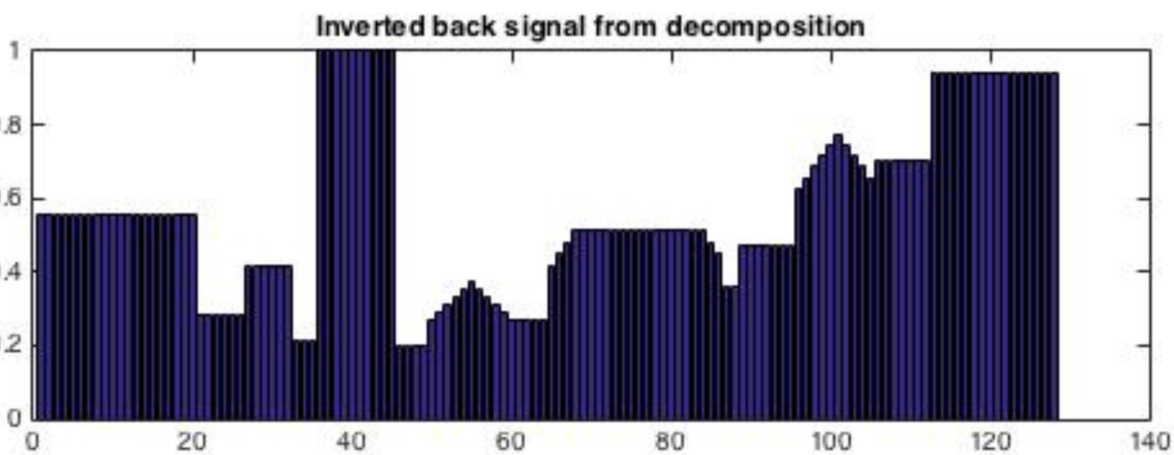
EL 7133 (DSP II)
Spring 2016
Homework Assignment - Week 01

Name: **Amitesh Kumar Sah**
NYU ID: **N19714360**

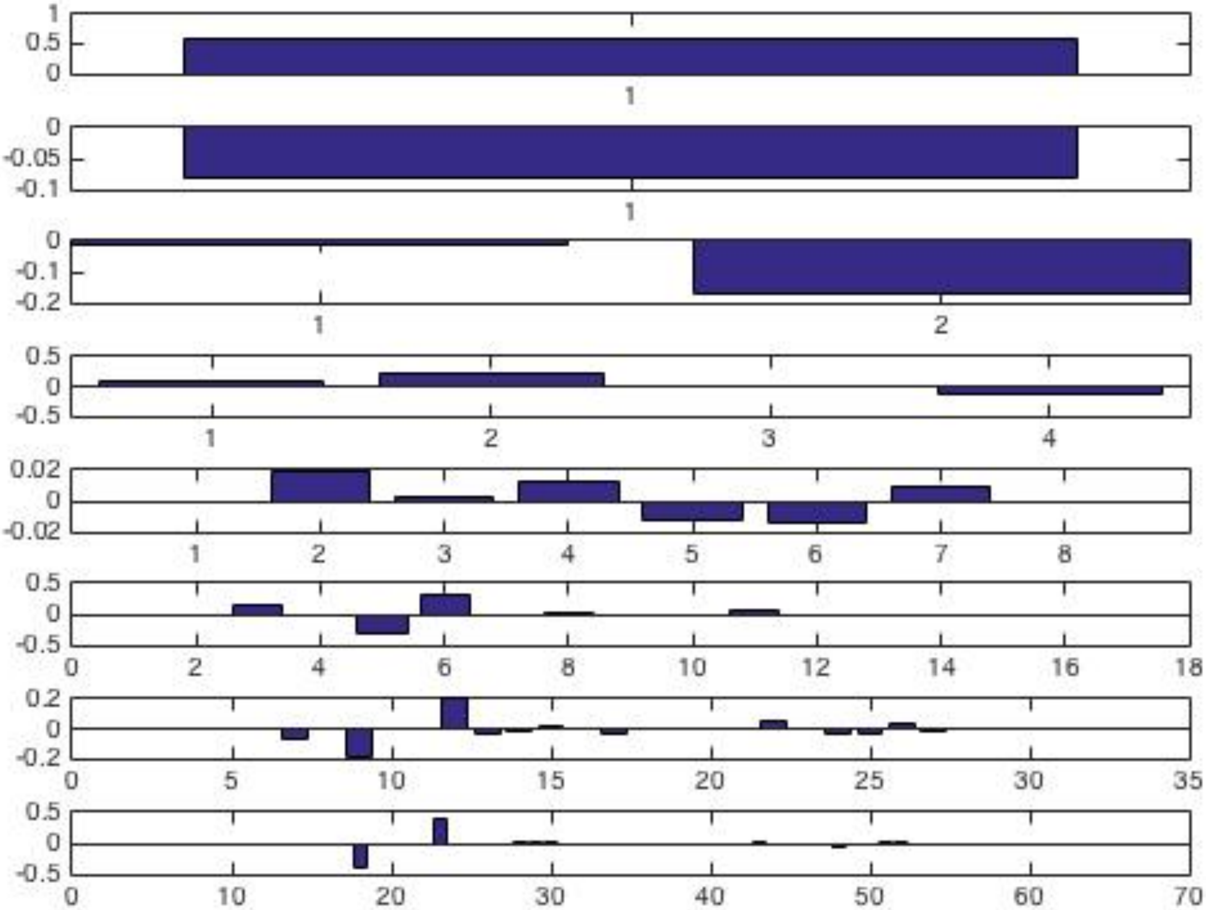
A) Haar wavelet transform: Write Matlab programs to implement forward and inverse Haar transform. The equations in lecture notes should be sufficient. Verify perfect reconstruction in Matlab

Solution:
Program is attached with this file.
Results:





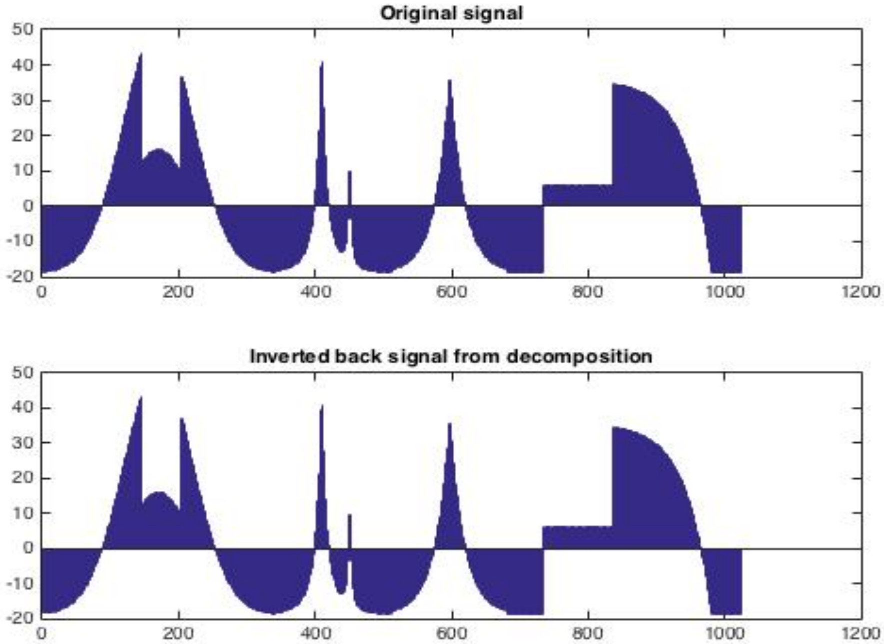
Decomposition Coefficient

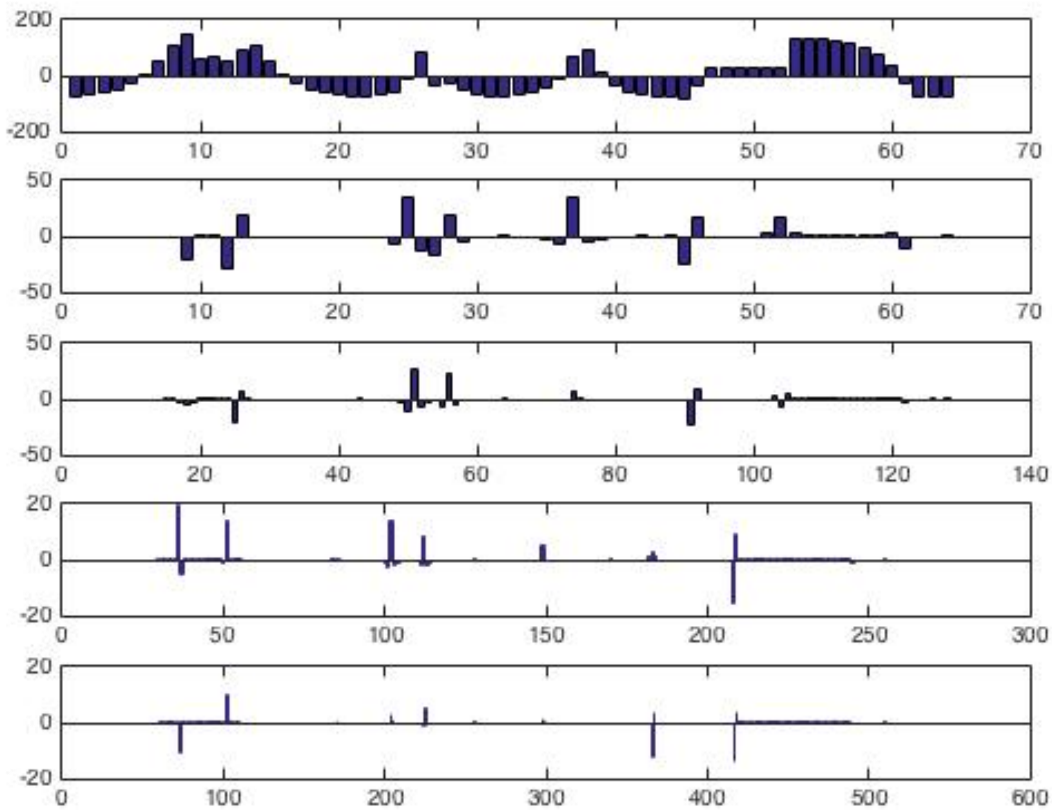


Here in this program , I took skyline signal as input. Read the file, calculated the length and found out the number of level decomposition required. If the signal is odd length, it is padded with zero at the end. Then I performed M-level operation forward transform using the equation of Forward Harr Transform and I plotted that signal. After I obtain the coefficients of Cn and Dn, I performed reconstruction i.e. Inverse Transform using reconstruction equation of Harr Transform . And then in a plot I verified the original signal with the reconstructed signal.

B) Like for (A) but for Daubechies length-4 filter. See the lecture notes for the filter coefficients and filter equations. Write Matlab programs to implement forward and inverse transform with length-4 Daubechies filters. Verify perfect reconstruction in Matlab.

Solution:
Program is attached with this report.
Result:

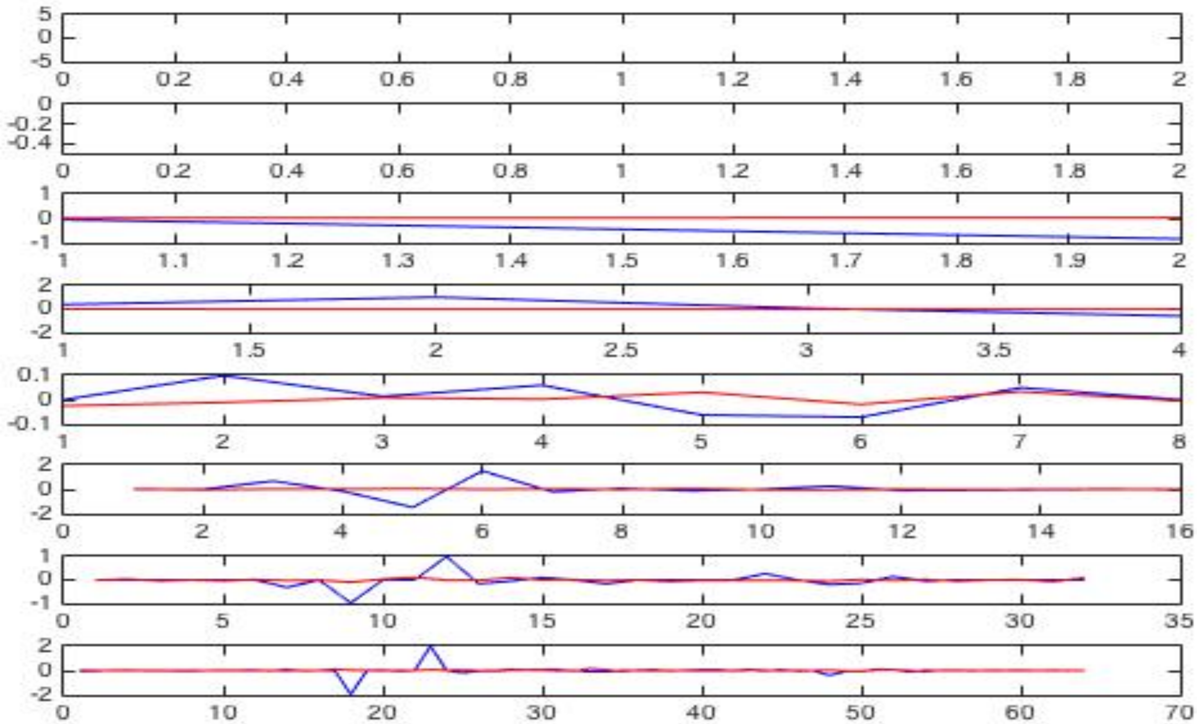




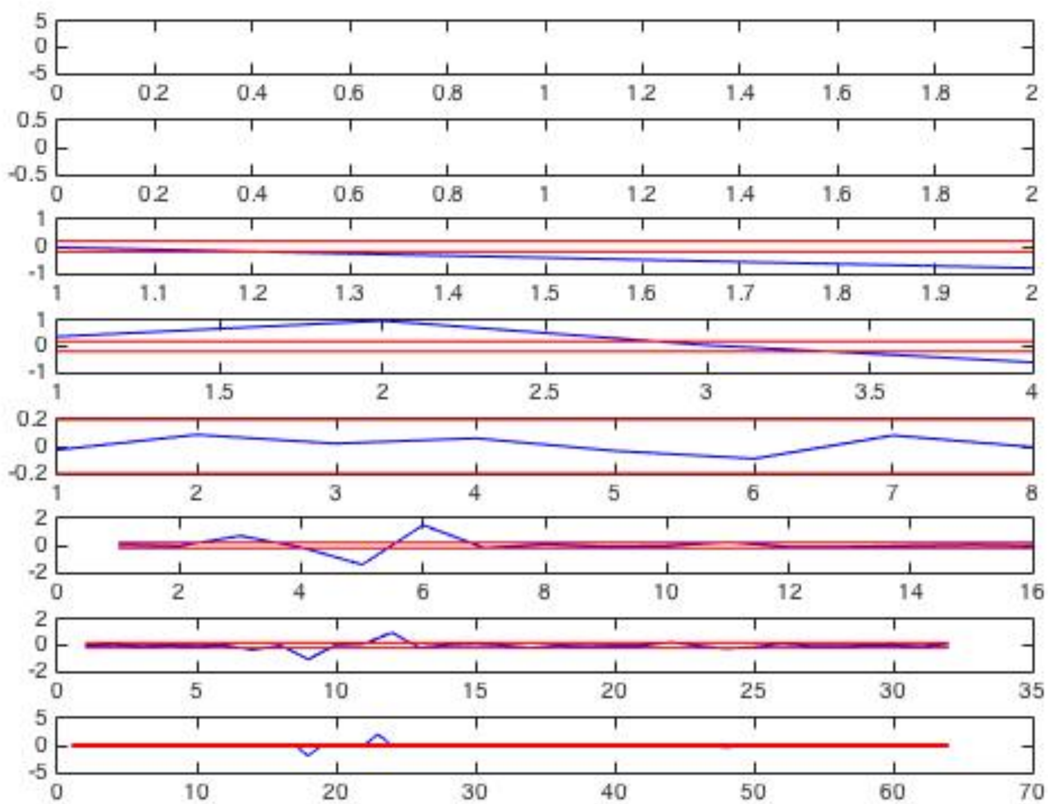
Here in this program , I took pwsmooth signal as input. Read the file. The number of level operation was 4 length. If the signal is odd length, it is padded with zero at the end. Then I performed M-level operation forward transform using the equation of Forward Harr Transform and I plotted that signal. After I obtain the coefficients of Cn and Dn, I performed reconstruction i.e. Inverse Transform using reconstruction equation of Harr Transform . And then in a plot I verified the original signal with the reconstructed signal.

C) Create examples of noise reduction using your wavelet transform programs with nonlinear thresholding. Compare Haar filters and Daubechies length-4 filters for noise reduction.

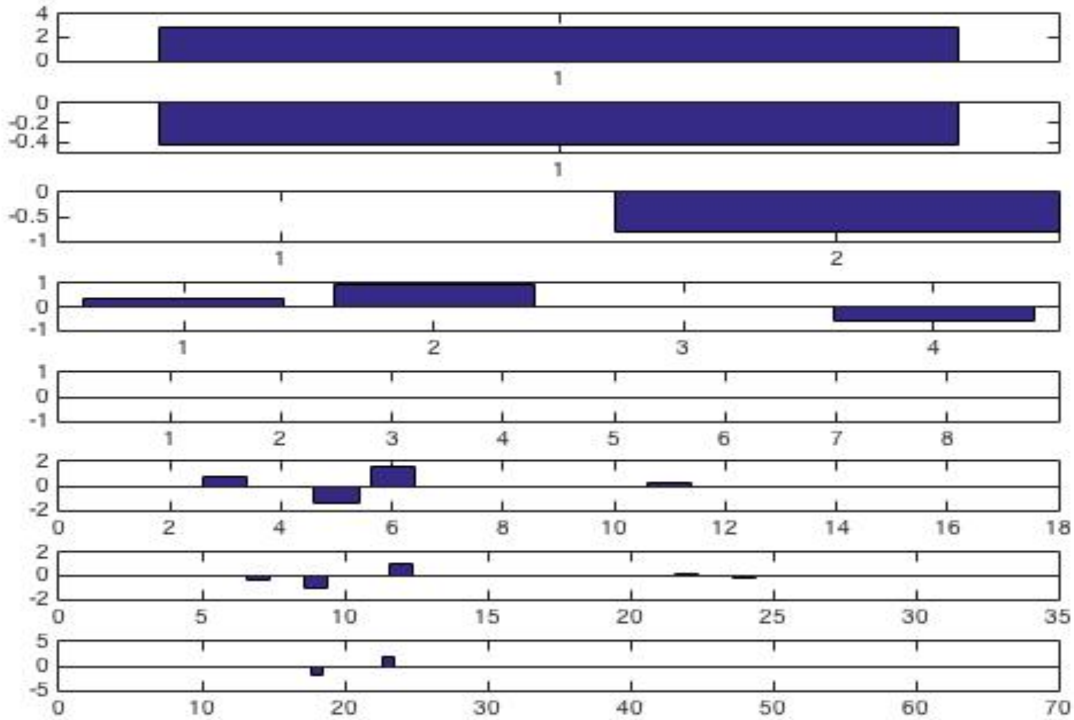
Solution
Using Haar filter for noise reduction:
Results
Decomposition of noise signal is represented by red and decomposition of noise free signal is represented by blue



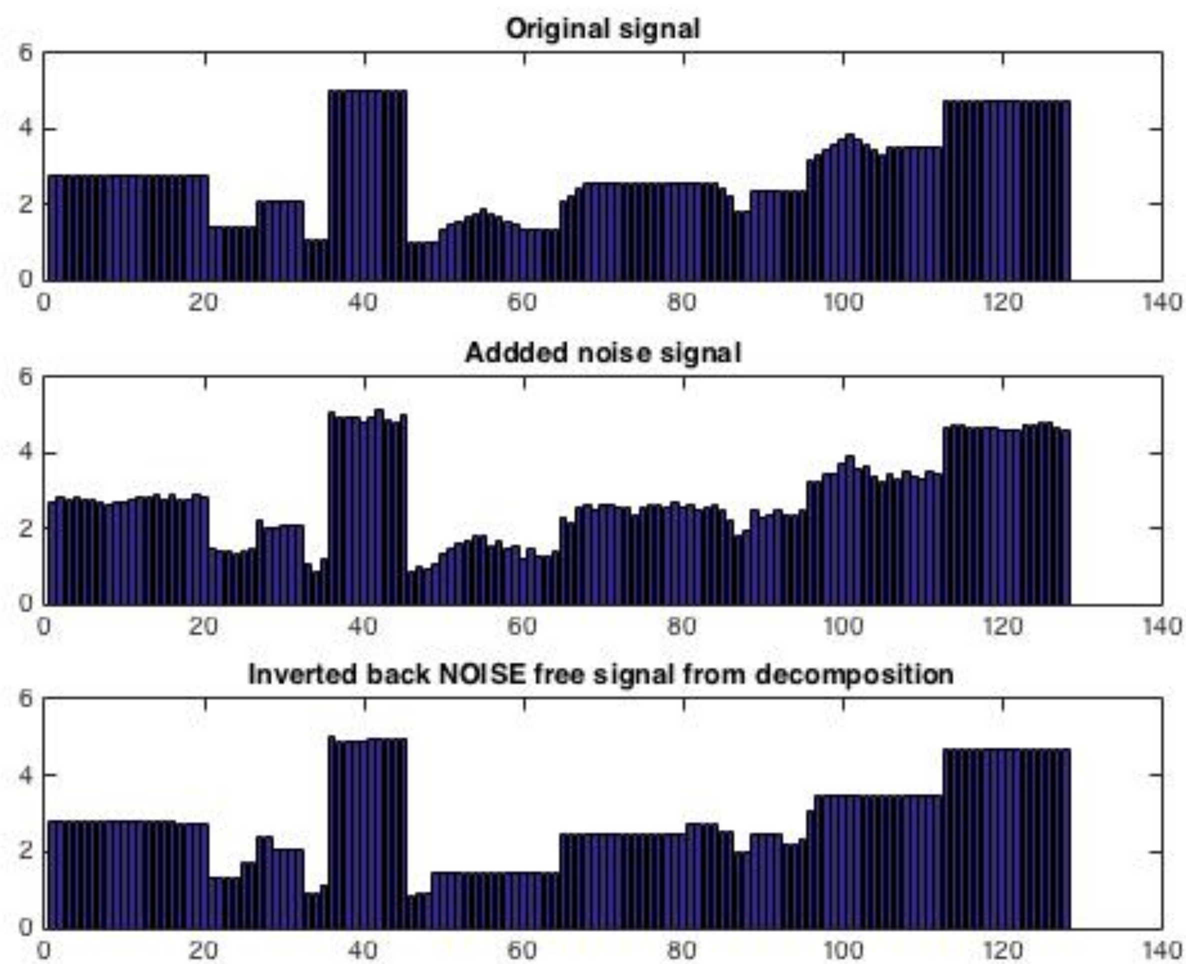
Decomposition of noisy signal with zero mean white Gaussian noise added with standard deviation 0.2.



Decomposition of noisy signal after nonlinear thresholding using hard threshold technique.

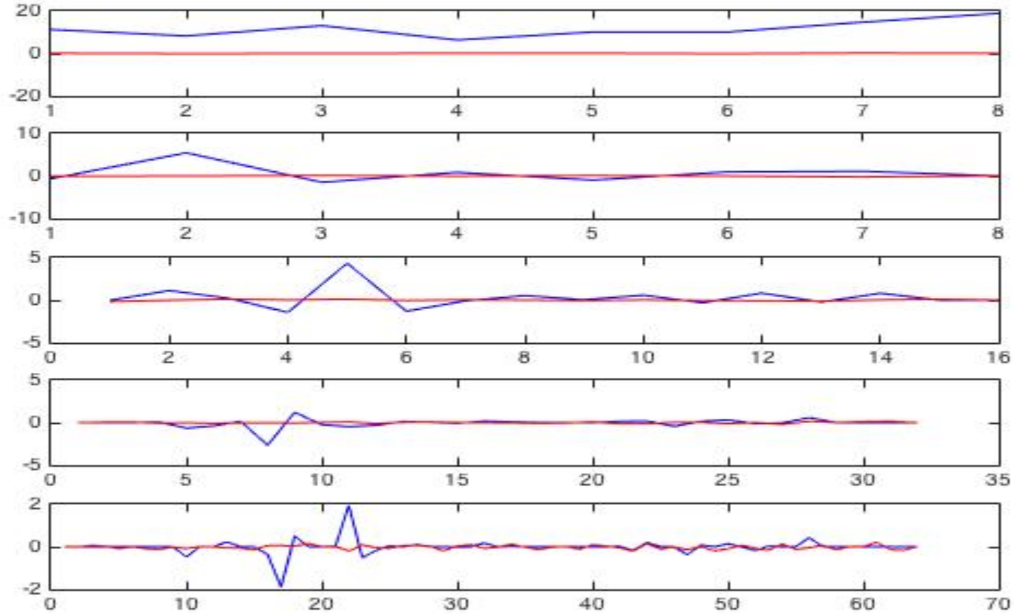


Reconstructed signal after removing the noise using inverse Haar filter

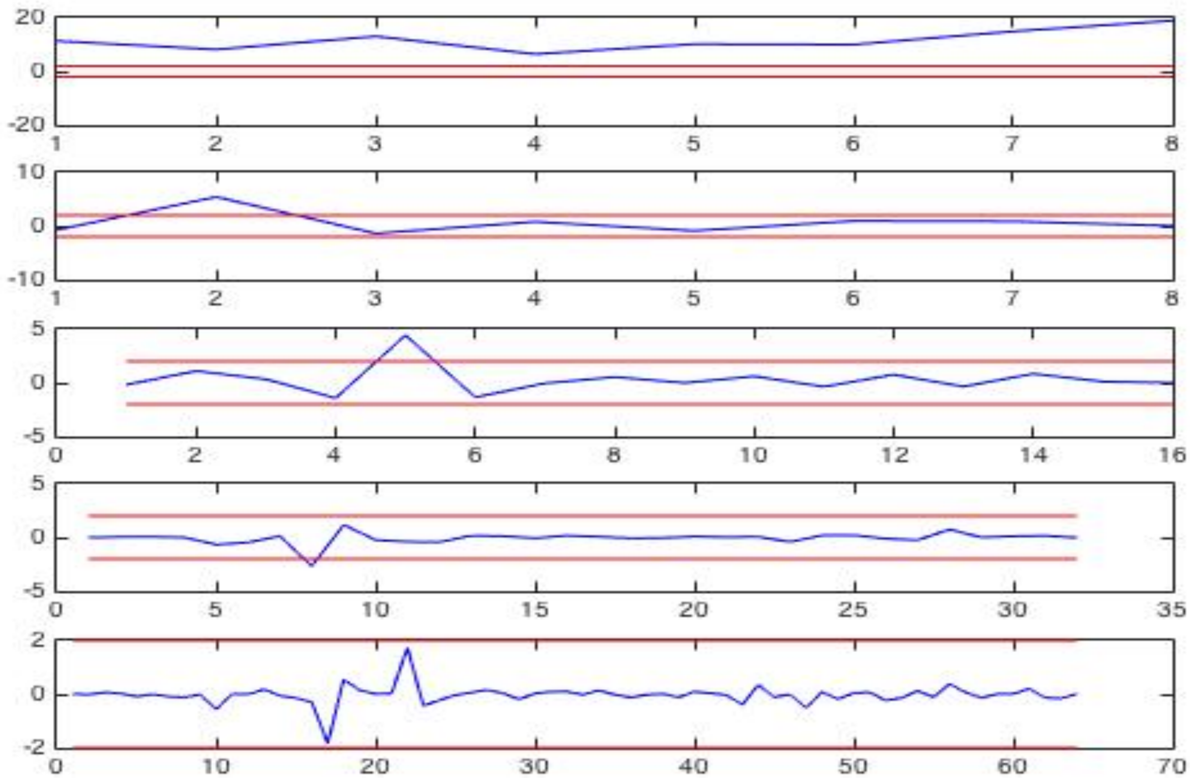


Daubechies length-4 filters

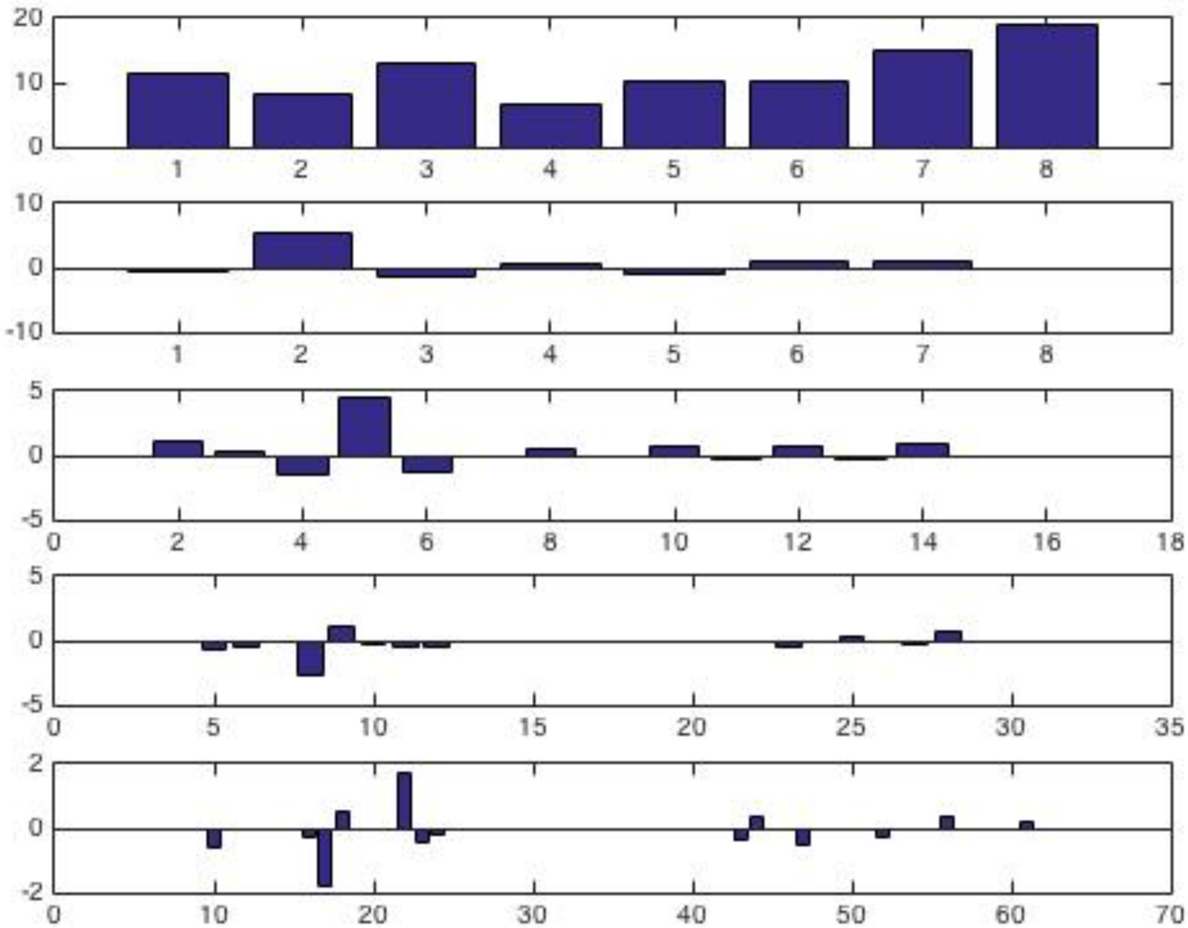
Decomposition of noise signal is represented by red and decomposition of noise free signal is represented by blue



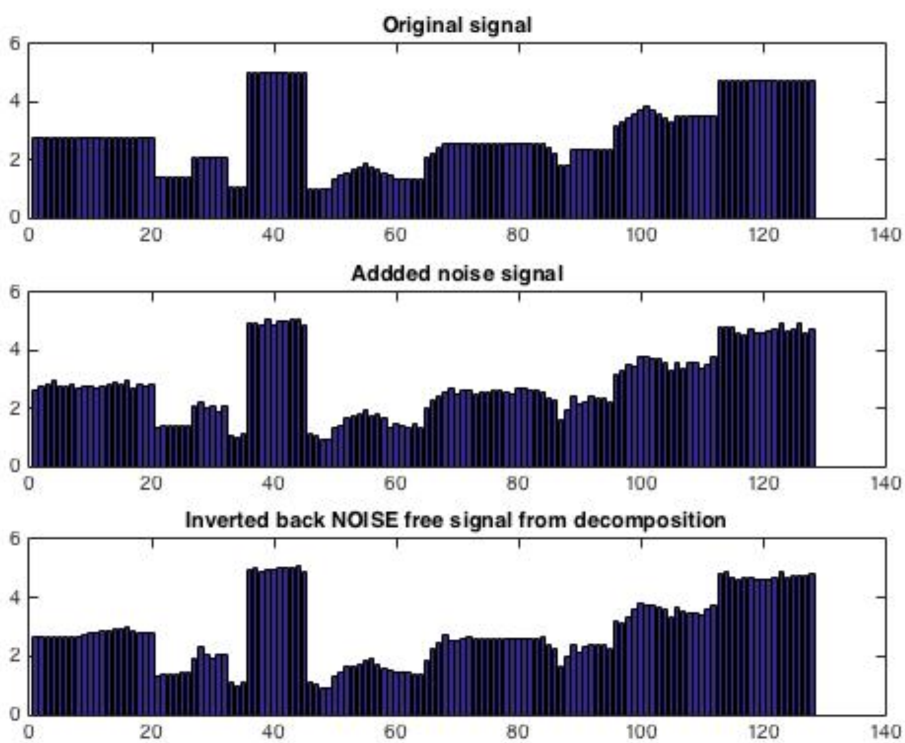
Decomposition of noisy signal with zero mean white Gaussian noise added with standard deviation 0.2.



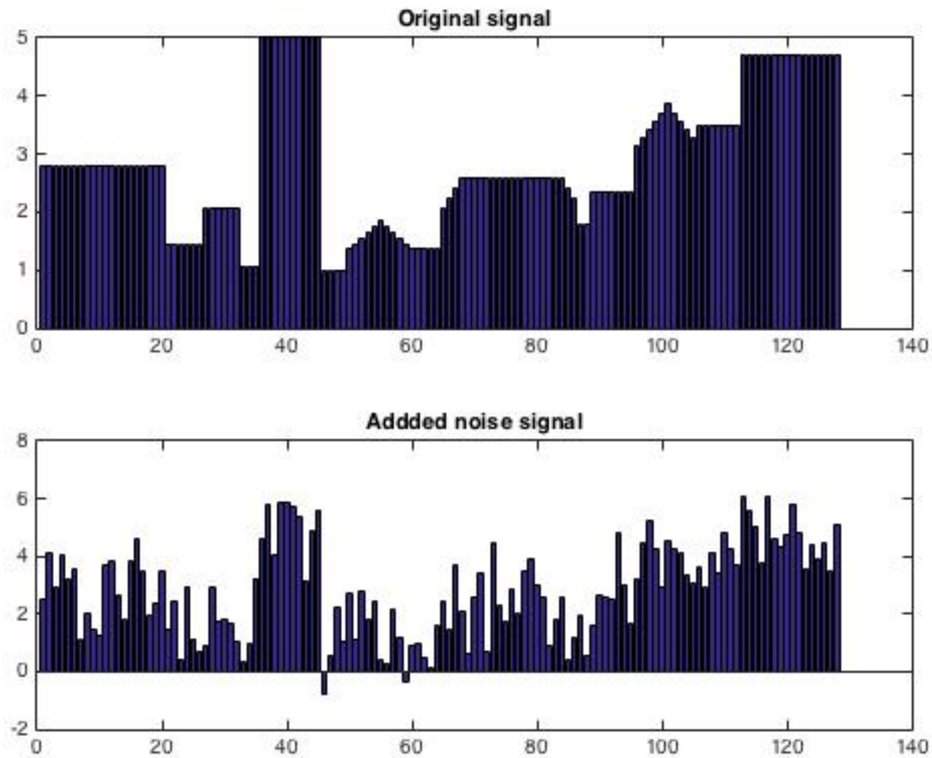
Decomposition of noisy signal after nonlinear thresholding using hard threshold technique.

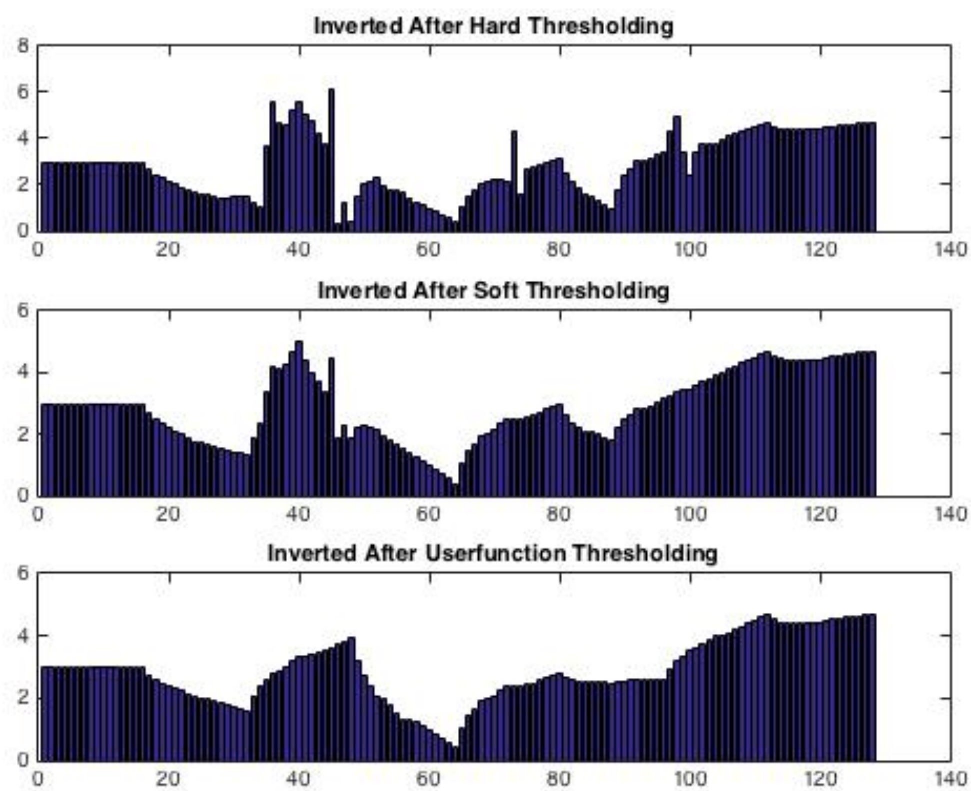


Reconstructed Signal using inverse Daubechies length 4 filter and hard thresholding



D) Design your own threshold function. Use it to perform denoising in the wavelet domain. Compare your threshold function with soft and hard thresholding. Can you improve upon hard and soft thresholding?






```
function f = threshfunc(dN,thr,delta,alpha)
[c r]=size(dN);
maxi=max(max(dN));
dN=dN/maxi
```

```
 range1=thr-(delta/2);
 range2=thr+(delta/2);
 %for hard thresholding
 r1=0;
if range1==range2
     r1=thr;
end
%  slope after r2
teta=atan((1-(range2-range1)^2)/(1-range2))
slope=tan(teta*alpha);

[c,r]=size(dN);

for i=1:c
    for j=1:r
       if dN(i,j)>=0
           if dN(i,j)<range1
           dNthreshf(i,j)=0;
           else if dN(i,j)>=range1 && dN(i,j)<=range2
               dNthreshf(i,j)=((dN(i,j)-range1)^2);
           else if  dN(i,j)>range2
                   dNthreshf(i,j)=r1+slope*(dN(i,j)-range2)+((range2-range1)^2);
               end
               end
           end
       else if dN(i,j)<0
           if dN(i,j)>-range1
           dNthreshf(i,j)=0;
           else if dN(i,j)<=-range1 && dN(i,j)>=-range2
               dNthreshf(i,j)=-((dN(i,j)-(-range1))^2);
           else if  dN(i,j)<range2
                   dNthreshf(i,j)=-r1+(slope*(dN(i,j)-(-range2))-(range2-range1)^2);
               end
               end
           end
           end
       end
    end
end
f=dNthreshf.*maxi;
end
```

Above program is the thresholding function created . It combiles both the soft and hard threshold technique in one.

```
function f = threshfunc(dN,thr,delta,alpha)
```

Here, there are four arguments to the function. The arguments are:

dN= It is the decomposition value of difference that is passed for thresholding

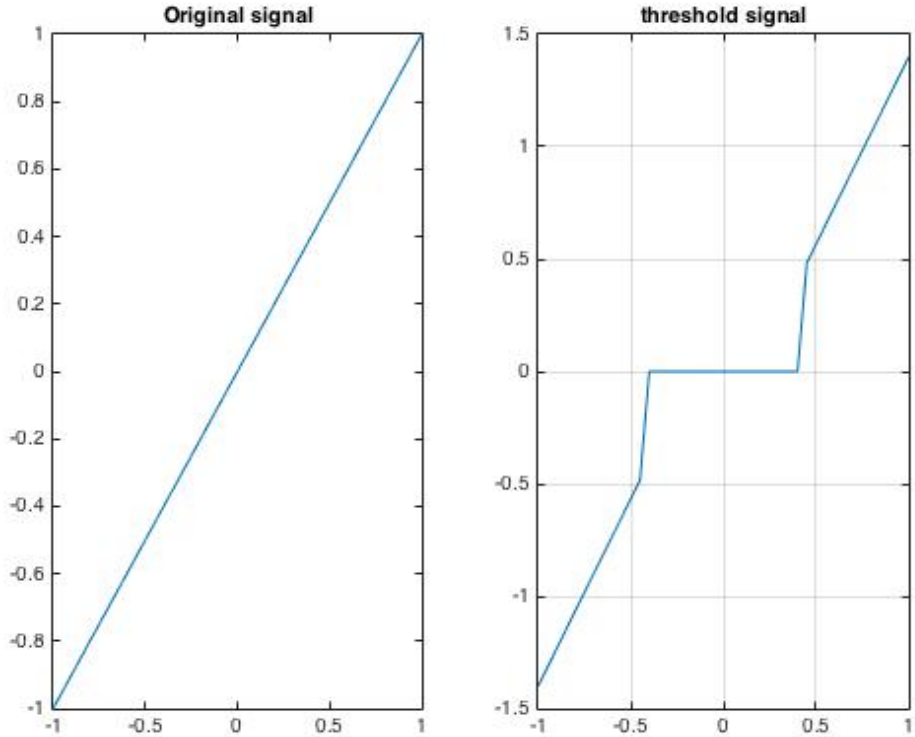
thr= it's the threshold value (deviation od noise signal)(value from 0 to 1)

delta= it's the small range near threshold where the signal cannot be noise. (0 to 1)

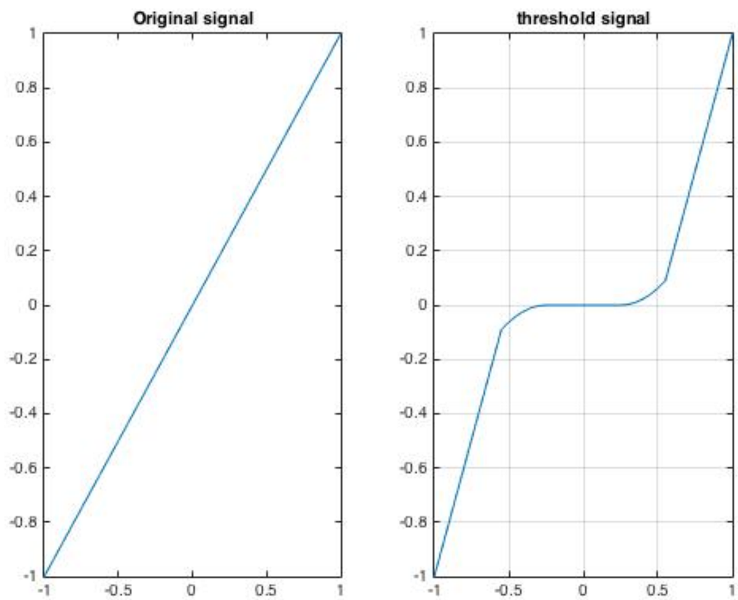
alpha- it determines whether it should cover the full range or just the range till threshold(0 to 1)

when 1, it means it covers the output till 1, when 0 , its output is till threshold

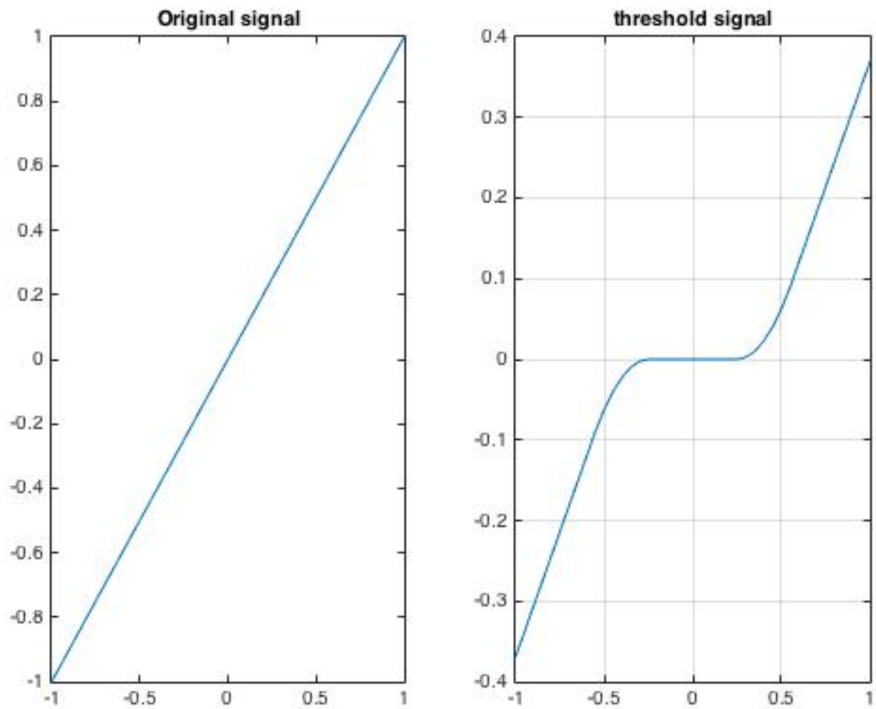
```
thr=0.4;
delta=0;
alpha=1;
```



when delta is 0 , it behaves as hard threshold



When delta =0.3, it considers the value near threshold although its magnitude is low.



when alpha is 0.5, its behaving as soft threshold and it is also preserving the edges as delta is 0.3