Department of Electronic & Telecommunication Engineering

University of Moratuwa

BM4151 : Biosignal processing

MATLAB Assignment 3 – Continuous and Discrete Wavelet Transforms

Name	Index number
D.S Weerapperuma	180685C

This is submitted as a partial fulfillment for the module BM4151 Biosignal Processing Department of Electronic and Telecommunication Engineering University of Moratuwa

Date of Submission:

10/01/2023

Table of contents

1 Continuous Wavelet Transform 5

1.1 Wavelet properties 5

1.2 Continuous Wavelet Decomposition 8

2 Discrete Wavelet Transform 10

2.1 Applying DWT with the Wavelet Toolbox in MATLAB 10

2.2 Signal Denoising with DWT 17

2.3 Signal Compression with DWT 23

Table of Figures

Figure 1.1 Time domain representation in wavelets 7
Figure 1.2 Mean and Energy at different scaling factors 7
Figure 1.3 Spectrum of wavelets at different scaling factors 8
Figure 1.4 input nonstationary signal x[n] 9
Figure 1.5 Spectrogram of x[n] 9
Figure 2.1 noisy y_1 and ideal x_1 in time domain 11
Figure 2.2 noisy y_2 and ideal x_2 in time domain 11
Figure 2.3 wavelet and scaling function of ‘db9’ wavelet 11
Figure 2.4 wavelet and scaling function of ‘haar’ wavelet 11
Figure 2.5 ‘ db9’ wavelet properties 12
Figure 2.6 ’haar’ wavelet properties 12
Figure 2.7 Y1[n] decomposition using ‘Haar’ wavelet 13
Figure 2.8 Y_1[n] decomposition using ‘db9’ wavelet 13
Figure 2.9 Y_2[n] decomposition using ‘Haar’ wavelet 14
Figure 2.10 Y_2[n] decomposition using ‘db9’ wavelet 14
Figure 2.11 Y_1[n] reconstructed | ‘haar’ wavelet 15
Figure 2.12 Y_1[n] reconstructed | ‘db9’ wavelet 15
Figure 2.13 Y_2[n] reconstructed |’ haar’ wavelet 16
Figure 2.14 Y_2[n] reconstructed | ‘db9’ wavelet 16
Figure 2.15 sorted haar wavelet coeff. (y_1) 17
Figure 2.16 constructed x_1 | harr wavelet 17
Figure 2.17 Comparison between original x_1 and reconstructed x_1 18
Figure 2.18 sorted ‘haar’ wavelet coeff. (x_1) 18
Figure 2.19 constructed x_1 | ‘db9’ wavelet 19
Figure 2.20 Comparison between original x_1 and reconstructed x_1 19
Figure 2.21 sorted ‘haar’ wavelet coeff. (y_2) 20
Figure 2.22 constructed x_2 | ‘harr’ wavelet 20
Figure 2.23 Comparison between original x2 and reconstructed x2 |’ haar’ 21
Figure 2.24 sorted ‘db9’ wavelet coeff. (x2) 21
Figure 2.25 constructed x_2 | ‘db9’ wavelet 22
Figure 2.26 Comparison between original x2 and reconstructed x2 | ‘db9’ 22
Figure 2.27 aVR signal in time domain 23

Figure 2.28 aVR decomposition using ‘haar’ wavelet 24
Figure 2.29 aVR decomposition using ‘db9’ wavelet 24
Figure 2.30 sorted coefficients(‘haar’) for aVR 25
Figure 2.31 constructed aVR signal using compressed data | ‘haar’ 25
Figure 2.32 aVR signal(original) and constructed using compressed data | ‘haar’ 26
Figure 2.33 sorted coefficients(‘db9’) for aVR 26
Figure 2.34 constructed aVR signal using compressed data | ‘db9’ 27
Figure 2.35 aVR signal(original) and constructed using compressed data | ‘db9’ 27

1 Continuous Wavelet Transform

Fourier transform (FT) decomposes a signal into linear combinations of complex exponentials of different frequencies. For stationary signals, it provides excellent localization in the frequency domain. For nonstationary signals, it provides perfect knowledge of frequencies, but no information about where these frequencies are located in time (Loss of time information). Also, accurate reconstruction of non-stationary signals is impossible with IFT since time information is lost at the FT. A technique called the wavelet transform provides excellent solutions for this by employing it to determine the time and frequency resolutions of a given signal.

The following is an equation for the continuous wavelet transform.

$$W(s,\tau) = \int x(t)\frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right)dt$$

Where, $\mathrm{s = scaling\ factor}, \tau = \mathrm{translation\ and}\ \psi = \mathrm{wavelet\ function}$

There are many wavelet families defined such as Haar, Shannon, Mexican hat, Morlet, Daubechies, etc. depending on the application.

In this section, we will construct the Mexican hat mother wavelet to check wavelet properties and then implement CWT on a non-stationary signal.

1.1 Wavelet properties

Given the Gaussian function $g(t)$ (μ= 0 and σ = 1), the Mexican hat function $m(t)$ can be obtained as follows.

$$m(t) = -\frac{d^2 g(t)}{dt^2}$$

Where,

$$g(t) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}t^2}$$

$$\frac{d}{dt}g(t) = \frac{-t}{\sqrt{2\pi}}e^{-\frac{1}{2}t^2}$$

$$\frac{d^2}{dt^2}g(t) = \frac{t^2}{\sqrt{2\pi}}e^{-\frac{1}{2}t^2} + \frac{-1}{\sqrt{2\pi}}e^{-\frac{1}{2}t^2}$$

Therefore,

$$\therefore m(t) = \frac{(1 - t^2)}{\sqrt{2\pi}}e^{-\frac{1}{2}t^2}$$

Since the wavelength function has unity energy let's calculate the normalizing factor (NF) using that fact.

$$Enerey\ of\ m(t) = E = \int_{-\infty}^{\infty} m(t)^2\,dt$$

$$E = \int_{-\infty}^{\infty} \frac{(1-t^2)^2 e^{-t^2}}{2\pi}\,dt$$

$$E = \left[\frac{3\,\mathrm{erf}(t)}{16\sqrt{\pi}} - \frac{t(2t^2-1)e^{-t^2}}{8\pi}\right]_{-\infty}^{\infty} = \frac{3}{8\sqrt{\pi}}$$

Since the energy should be 1, we can calculate NF as follows.

$$Normalization\ factor(NF) = \frac{1}{\sqrt{E}} = \sqrt{\frac{8\sqrt{\pi}}{3}} = \frac{2\sqrt{2}\pi^{\frac{1}{4}}}{\sqrt{3}}$$

normalized Mexican hat function can be derived as follows.

$$m_{normalized}(t) = NF\,.\,m(t)$$

$$m_{normalized}(t) = \frac{2\sqrt{2}\pi^{\frac{1}{4}}}{\sqrt{3}} \times \frac{(1-\ t^2)}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2}$$

$$m_{normalized}(t) = \frac{2\ (1-\ t^2)}{\sqrt{3}\pi^{\frac{1}{4}}} e^{-\frac{1}{2}t^2}$$

Using the above equations, we can derive normalized Mexican hat mother wavelet $\psi_{s,\tau}(t)$ as follows.

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right)$$

$$\boxed{\psi_{s,\tau}(t) = \frac{2}{\sqrt{3s}\pi^{\frac{1}{4}}}\left[1-\left(\frac{t-\tau}{s}\right)^2\right] e^{-\frac{1}{2}\left(\frac{t-\tau}{s}\right)^2}}$$

Zero mean, unity power and compact support (limited in the time domain) are specialties in the wavelet functions and because of these valuable properties, the wavelet's function can use in a lot of applications and this document discuss some of them as well.

Using the provided script 'wavelet_construction.m', we generated the Mexican hat daughter wavelet for scaling factors of 0.01:0.1:2. Following figures show the resultant time domain and frequency domain waveforms.

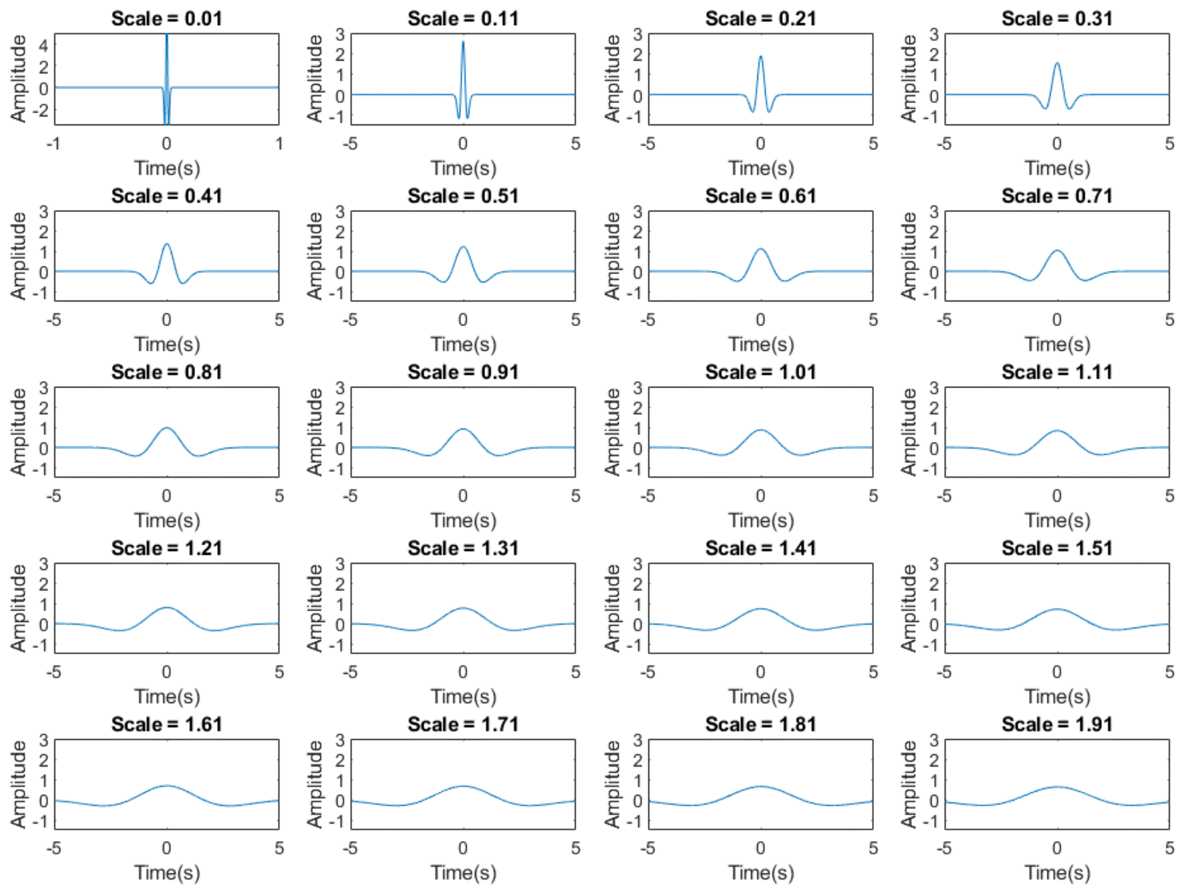

Figure 1.1 Time domain representation in wavelets

According to the above graph, we can observe that when we are increasing the scaling factor, the width of the waveform is became wider and the maximum amplitude is reduced. However, we can clarify the compact support of the daughter wavelets since most of the details are located in a small range of time (limited in the time domain).

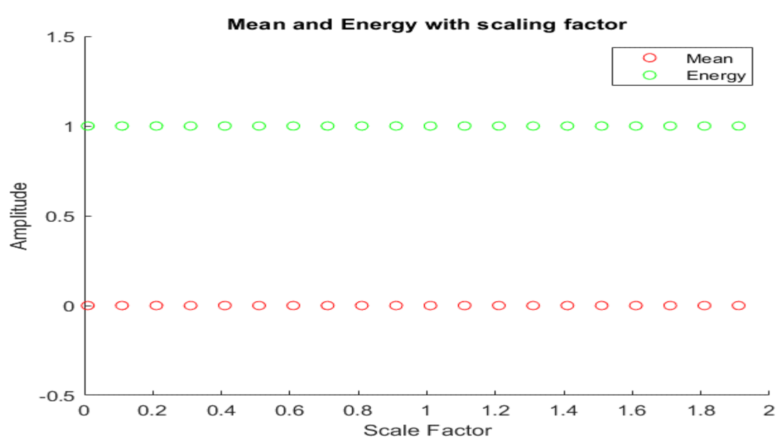

Figure 1.2 Mean and Energy at different scaling factors

According to figure 1.2, at every scaling factor, the mean is 0 and the energy is 1. Therefore, we can conclude that any daughter wavelets have the 0 mean and unity energy.

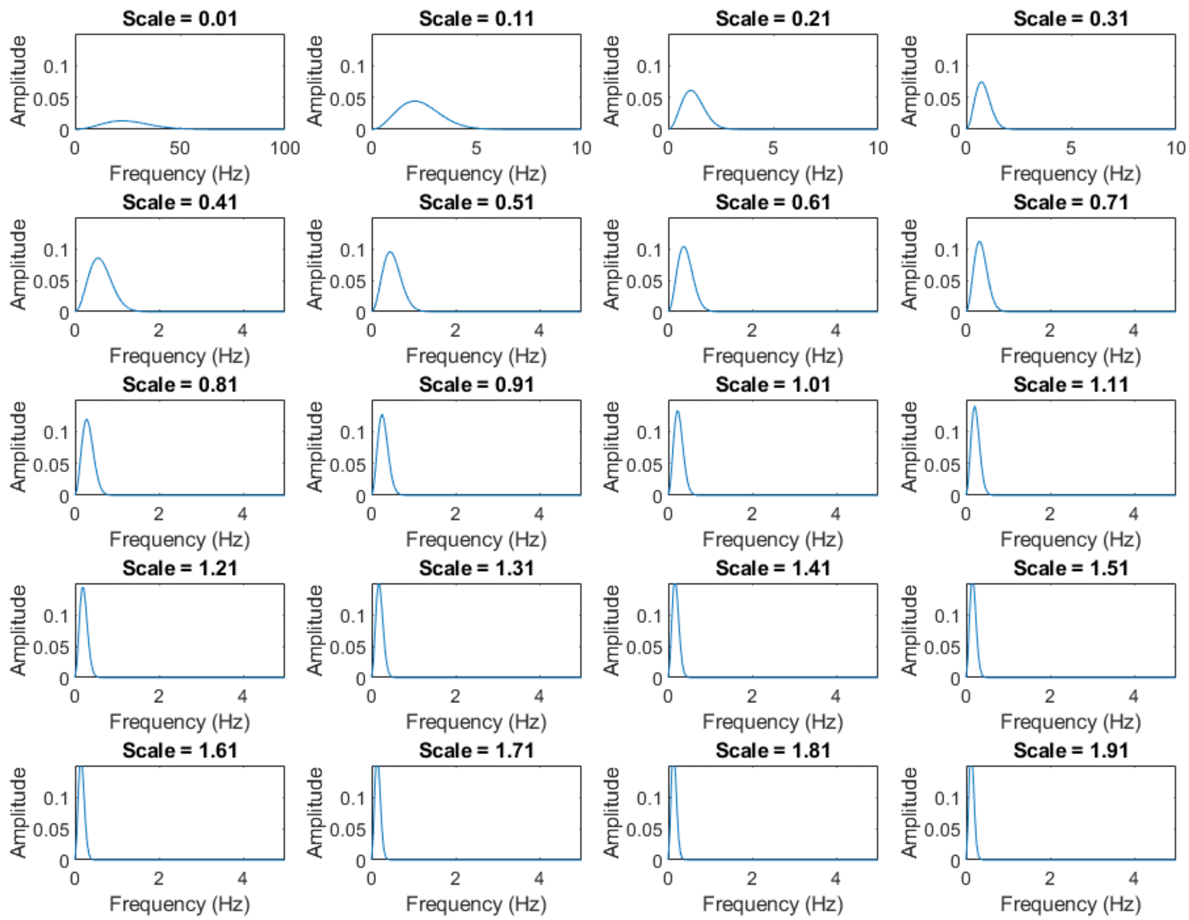

Figure 1.3 Spectrum of wavelets at different scaling factors

By observing the above figure, we can see a bandpass behavior from the wavelets. Also, we can conclude that there is an inversely proportional relationship between the scaling factor and the center frequency since increasing the scaling factor is cause to decrease in the center frequency and vice versa. We can increase the pass band by decreasing the scale so that we can capture higher-frequency components as well.

1.2 Continuous Wavelet Decomposition

We created a waveform on MATLAB as defined below with the following parameters.

$$x(n) = \begin{cases} \sin(0.5\pi n) & 1 \leq n < \frac{3N}{2} \\ \sin(1.5\pi n) & \frac{3N}{2} \leq n < 3N \end{cases}$$

Then we applied the scaled Mexican hat wavelets to $x(n)$ with the following specifications. To achieve translations, for each wavelet scale, we convolved the signal with the constructed wavelet.

Note: increased the scale resolution to 0.01:0.01:2. The input signal and the generated spectrogram are shown in the following figures.

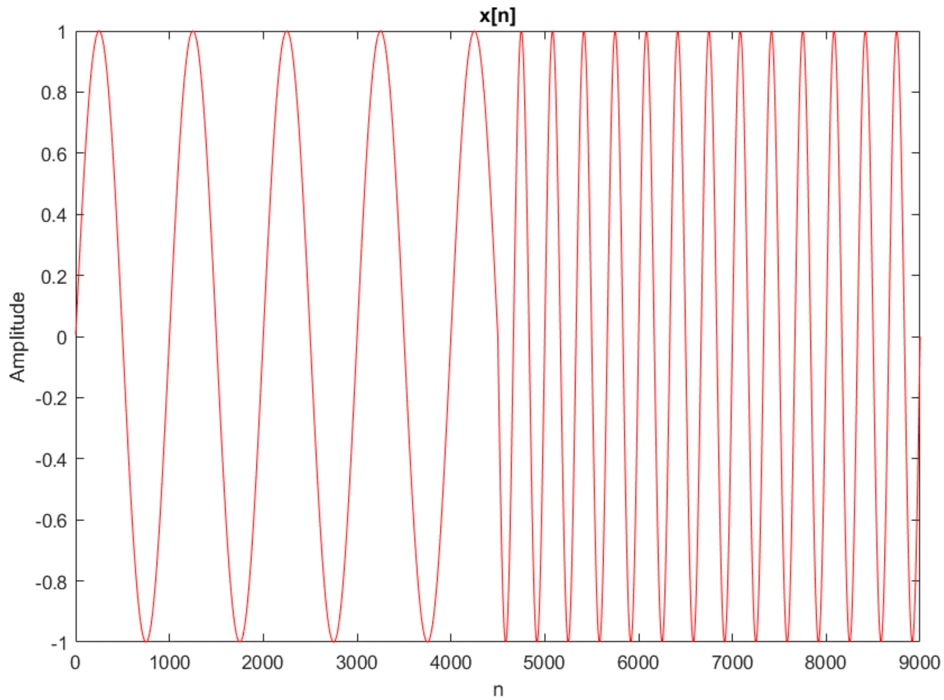

Figure 1.4 input nonstationary signal x[n]

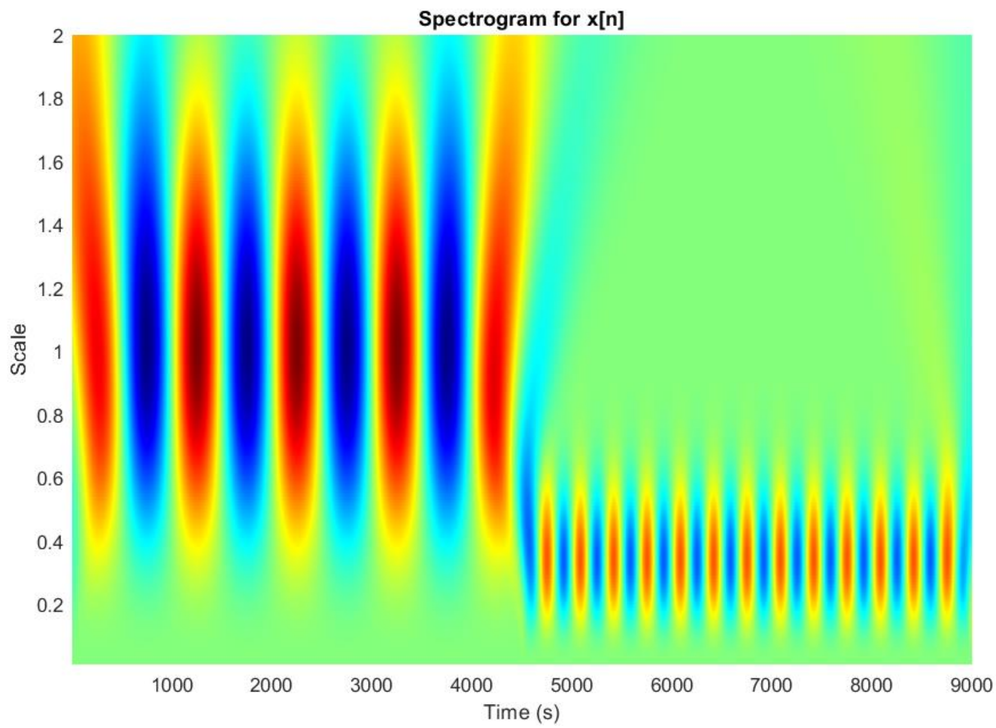

Figure 1.5 Spectrogram of x[n]

If we carefully look at the above spectrogram, we can observe that before the 4500^{th} sample point, most of the highest values are related to the scale value of 1. According to the spectrum of the scaling factor of 1 of the daughter wavelet, the central frequency is around 0.25 Hz ($0.5\pi\ rads^{-1}$). After the 4500^{th} sample point, most of the highest values are related to 0.32. According to the spectrum of the scaling factor of 0.32 of the daughter wavelet, the central frequency is around 0.75 Hz ($1.5\pi\ rads^{-1}$). Therefore, we can conclude that the generated spectrogram shows the coefficient correctly and identified the main frequency components embedded in the signal with respect to time. According to figure 1.3, we noticed by using higher scaling factors we can capture lower frequencies and vice versa. Also, for higher scaling factors we can capture very narrow bandwidth and vice versa. All these facts are clearly shown in this spectrogram. Because this spectrogram represents lower frequencies by locating higher values at higher scaling factors with lower resolution and shows higher frequencies by locating higher values at lower scaling factors with higher resolution. Therefore, this spectrogram correctly identified the both frequencies and the time domain frequency variation details of the signal.

2 Discrete Wavelet Transform

The drawbacks of CWT include highly redundant computations which lead to the requirement of additional computational power and time consumption. To avoid this, in discrete wavelet transform (DWT), the scaling and translation are performed in a discrete manner.

For DWT, the equation for CWT is modified as follows.

$$\psi_{m,n}(t) = \frac{1}{\sqrt{S_0^m}} \psi\left(\frac{t - n\tau_0 s_0^m}{S_0^m}\right)$$

Where, $\mathrm{s}_0 = \text{scaling step size}, \tau_0 = \text{translation step size},$ m & n are corresponding multiplier integers.

Usually, $s_0 = 2$ and $\tau_0 = 1$ are used for efficient analysis.

2.1 Applying DWT with the Wavelet Toolbox in MATLAB

First, we created the following waveforms on MATLAB.

$$x_1[n] = \begin{cases} 2\sin(20\pi n) + \sin(80\pi n) & 0 \le n < 512 \\ 0.5\sin(40\pi n) + \sin(60\pi n) & 512 \le n < 1024 \end{cases}$$

$$x_2[n] = \begin{cases} 1 & 0 \le n < 64 \\ 2 & 192 \le n < 256 \\ -1 & 256 \le n < 512 \\ 3 & 512 \le n < 704 \\ 1 & 704 \le n < 960 \\ 0 & otherwise \end{cases}$$

Note: **The assignment has a wrong range (128 – 512) for which the signal has the value of -1. In this case, we consider 256 for meaningfulness**.

The noise-added signals and the ideal signals are shown in the following figures.

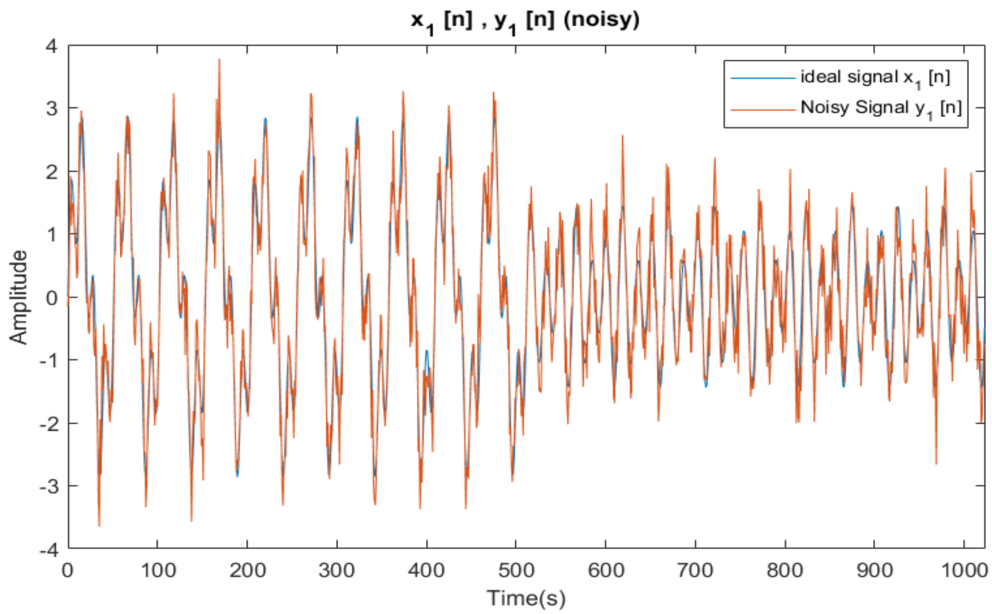

Figure 2.1 noisy y_1 and ideal x_1 in time domain

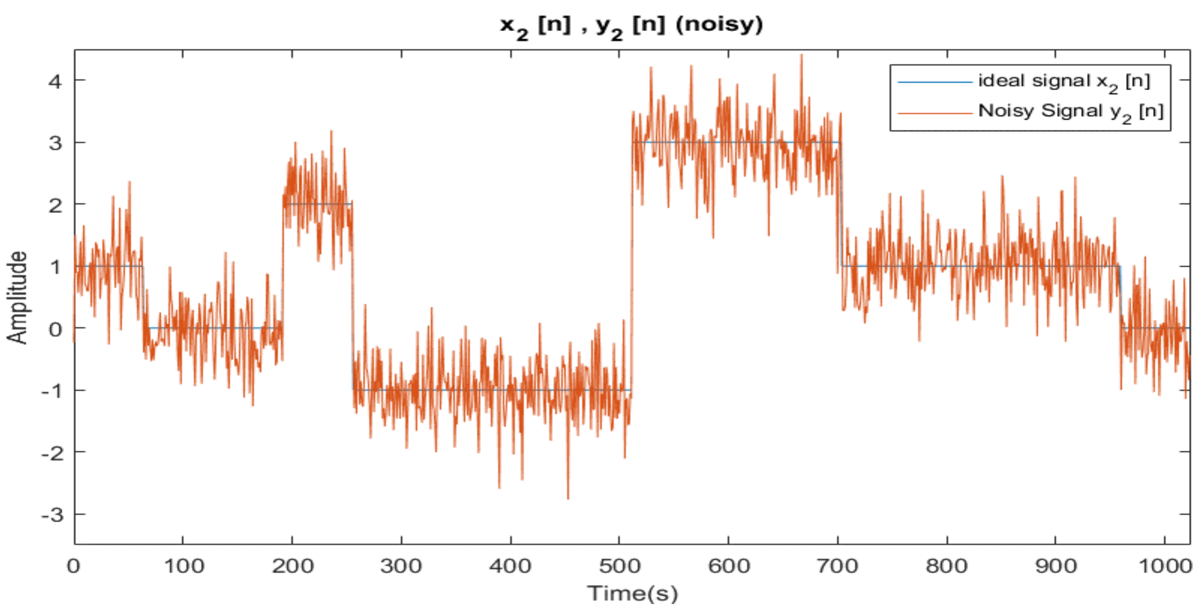

Figure 2.2 noisy y_2 and ideal x_2 in time domain

Let's look at the morphology of the wavelet and scaling functions of 'Haar' and Daubechies tap 9.

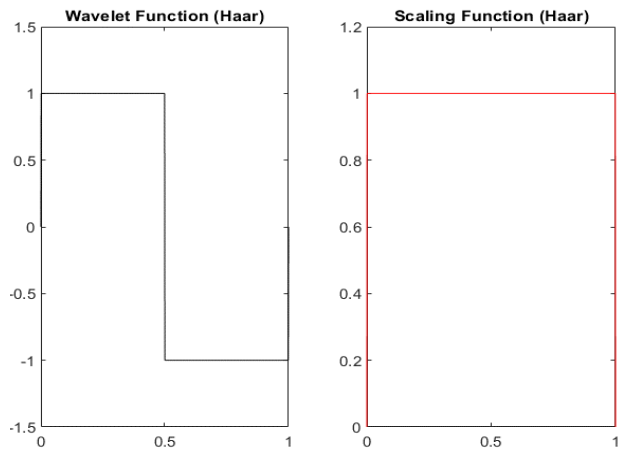

Figure 2.4 wavelet and scaling function of 'haar' wavelet

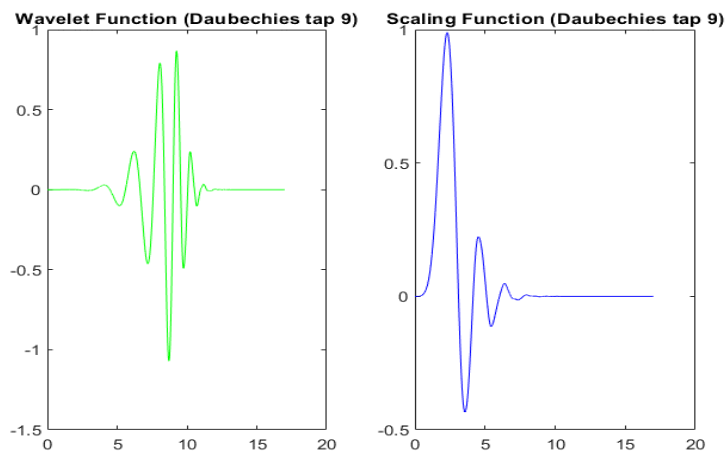

Figure 2.3 wavelet and scaling function of 'db9' wavelet

Let's use 'waveletAnalyzer GUI' to observe the wavelet characteristics.

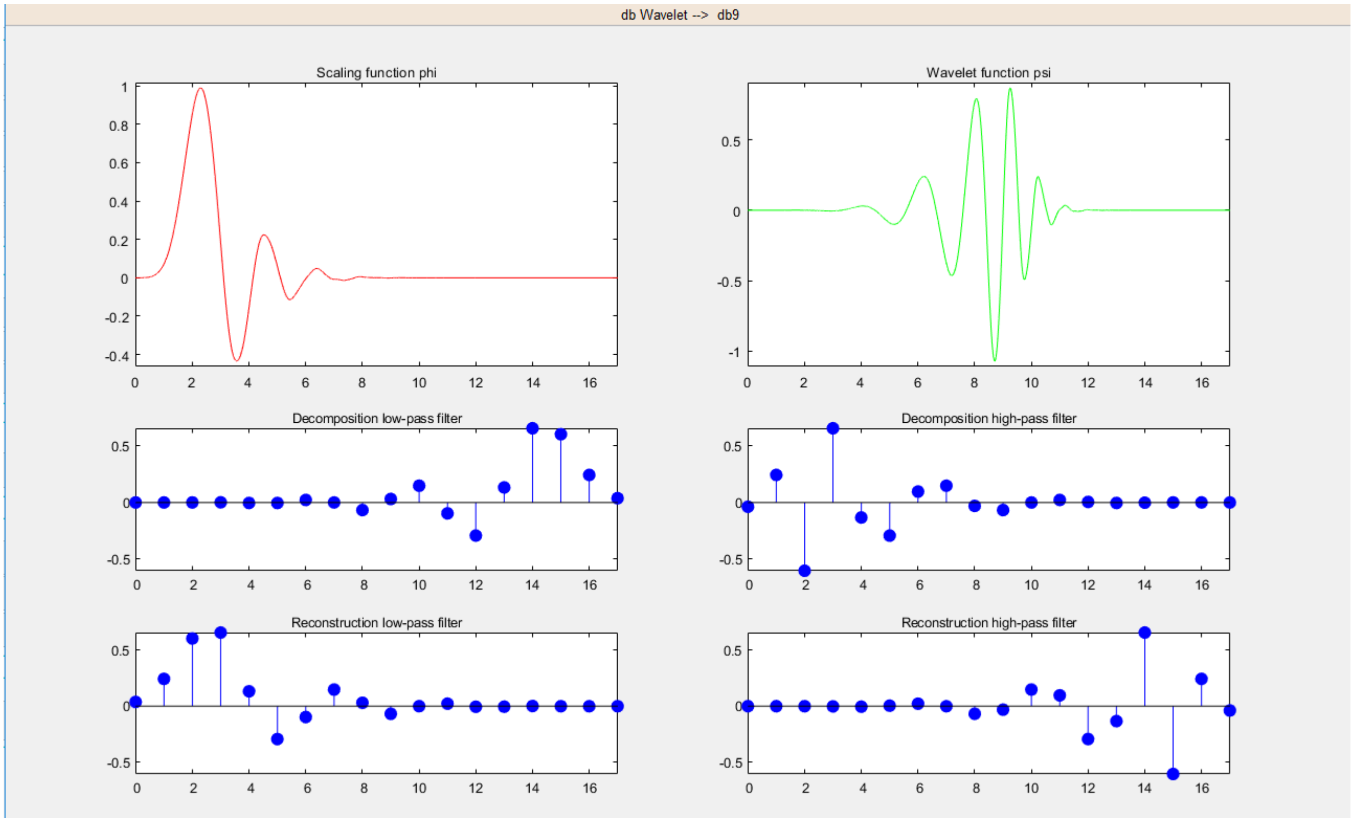

Figure 2.5 'db9' wavelet properties

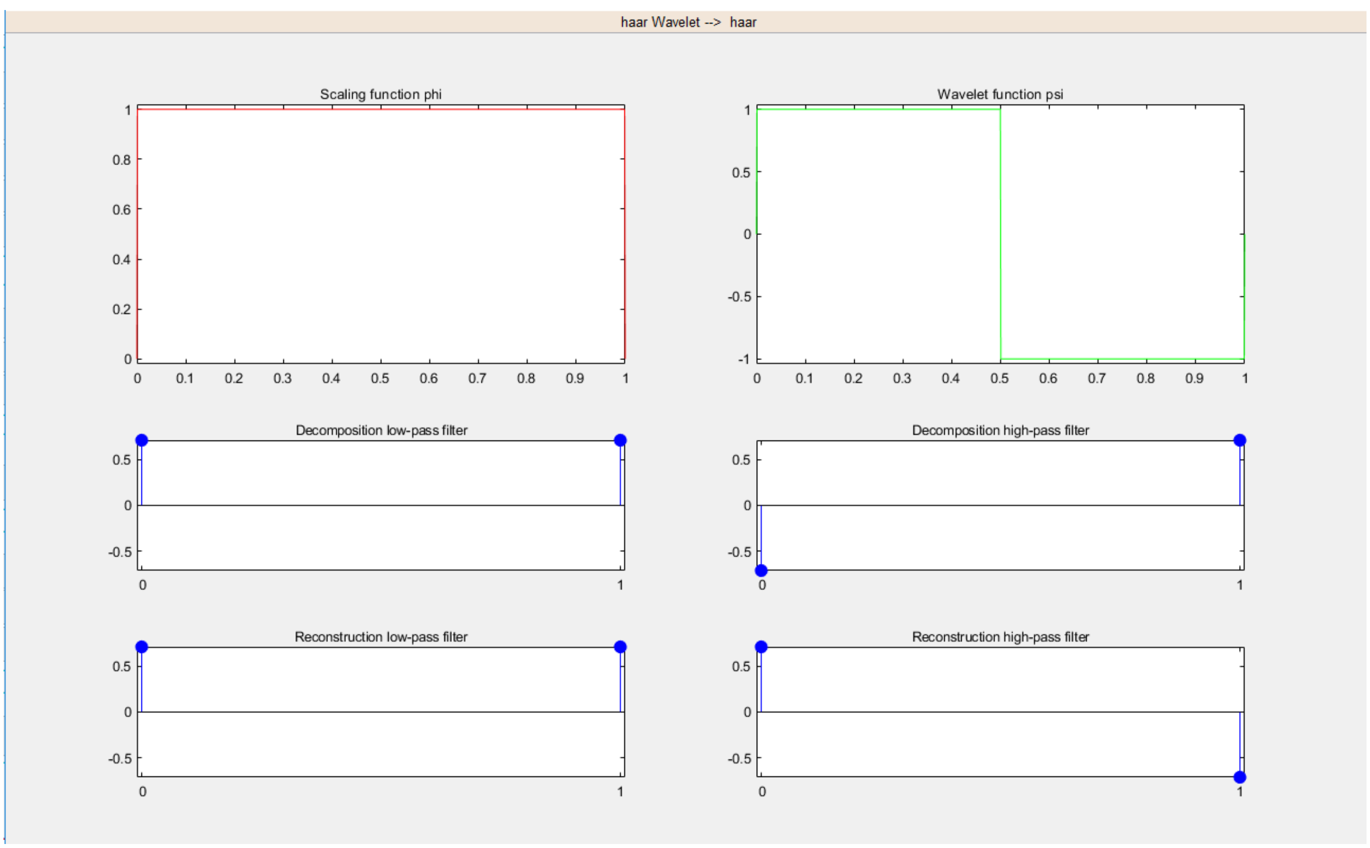

Figure 2.6 'haar' wavelet properties

We calculated the 10-level wavelet decomposition of the signal using wavelets ‘db9’ and ‘haar’ by using the command ‘wavedec()’. The following figures are shows all the approximation and detail coefficients at each signal applied with the ‘haar’ and ‘db9’ wavelets.

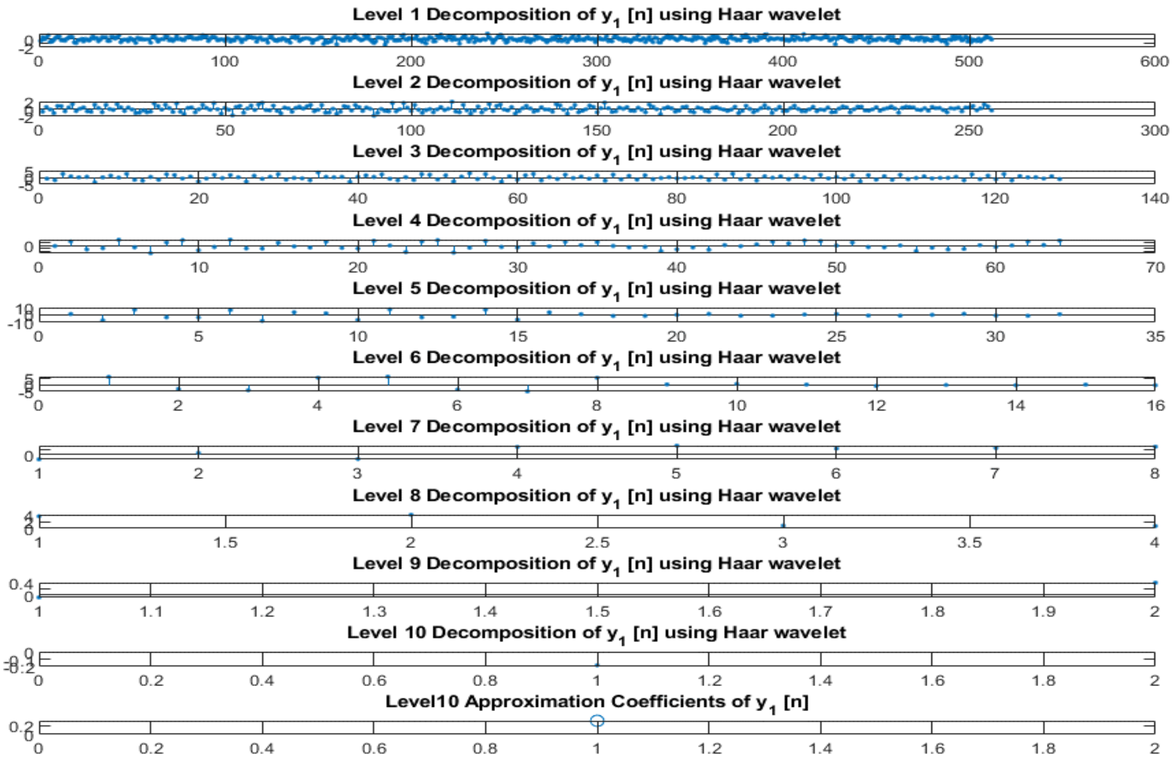

Figure 2.7 Y1[n] decomposition using ‘Haar’ wavelet

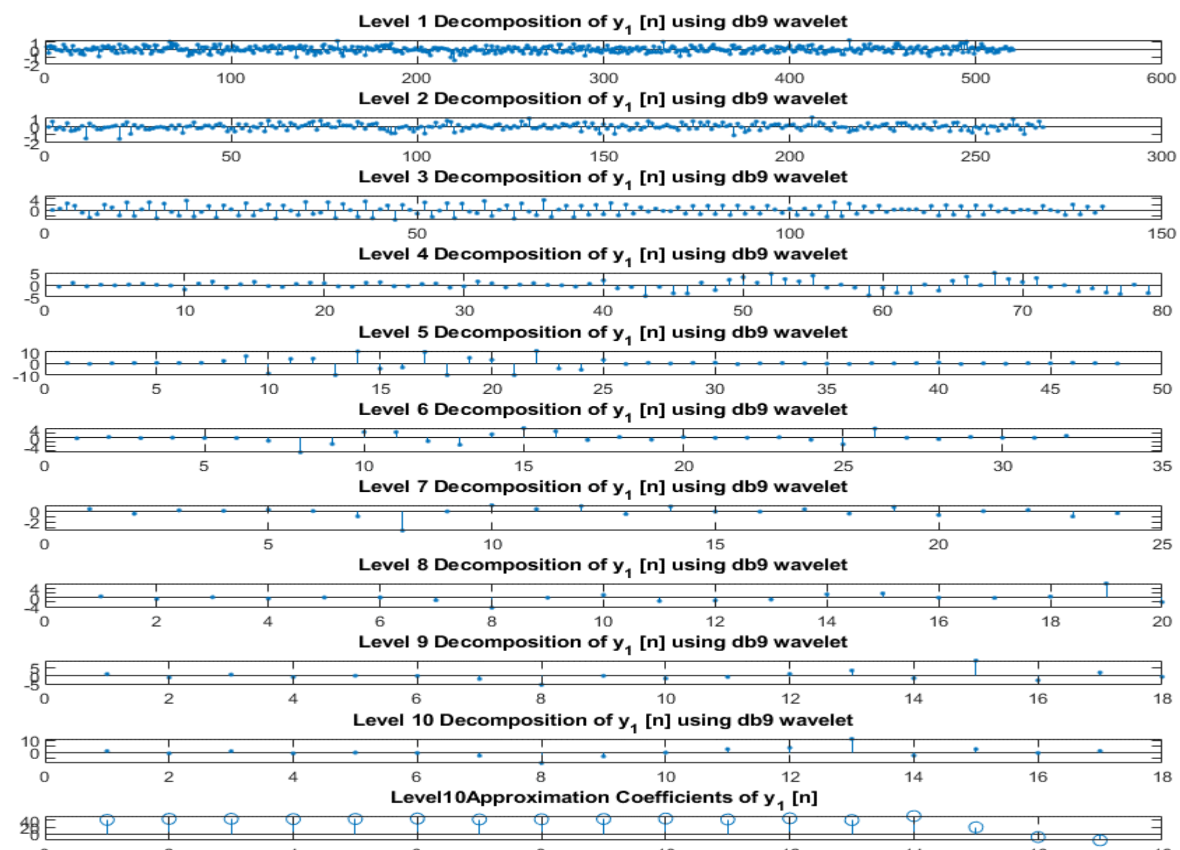

Figure 2.8 $Y_1[n]$ decomposition using ‘db9’ wavelet

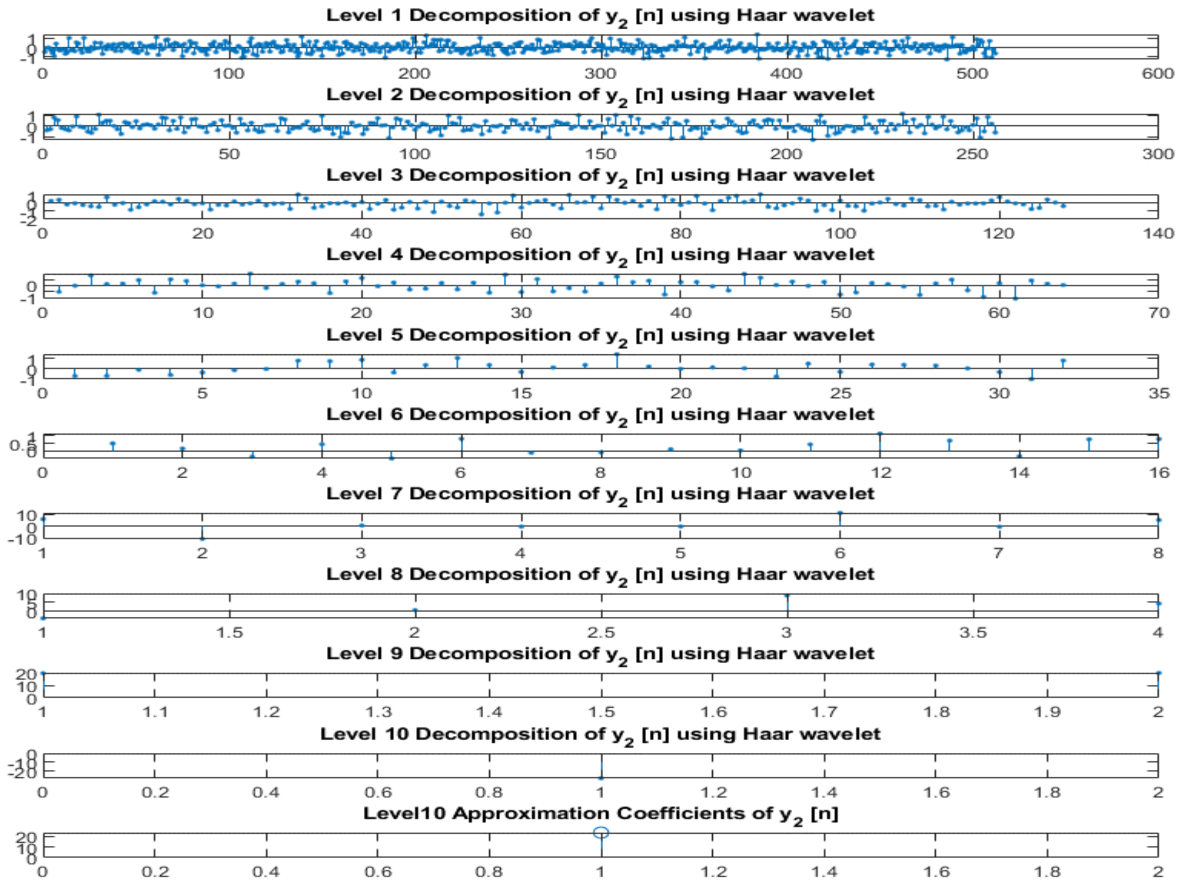

Figure 2.9 $Y_2[n]$ decomposition using 'Haar' wavelet

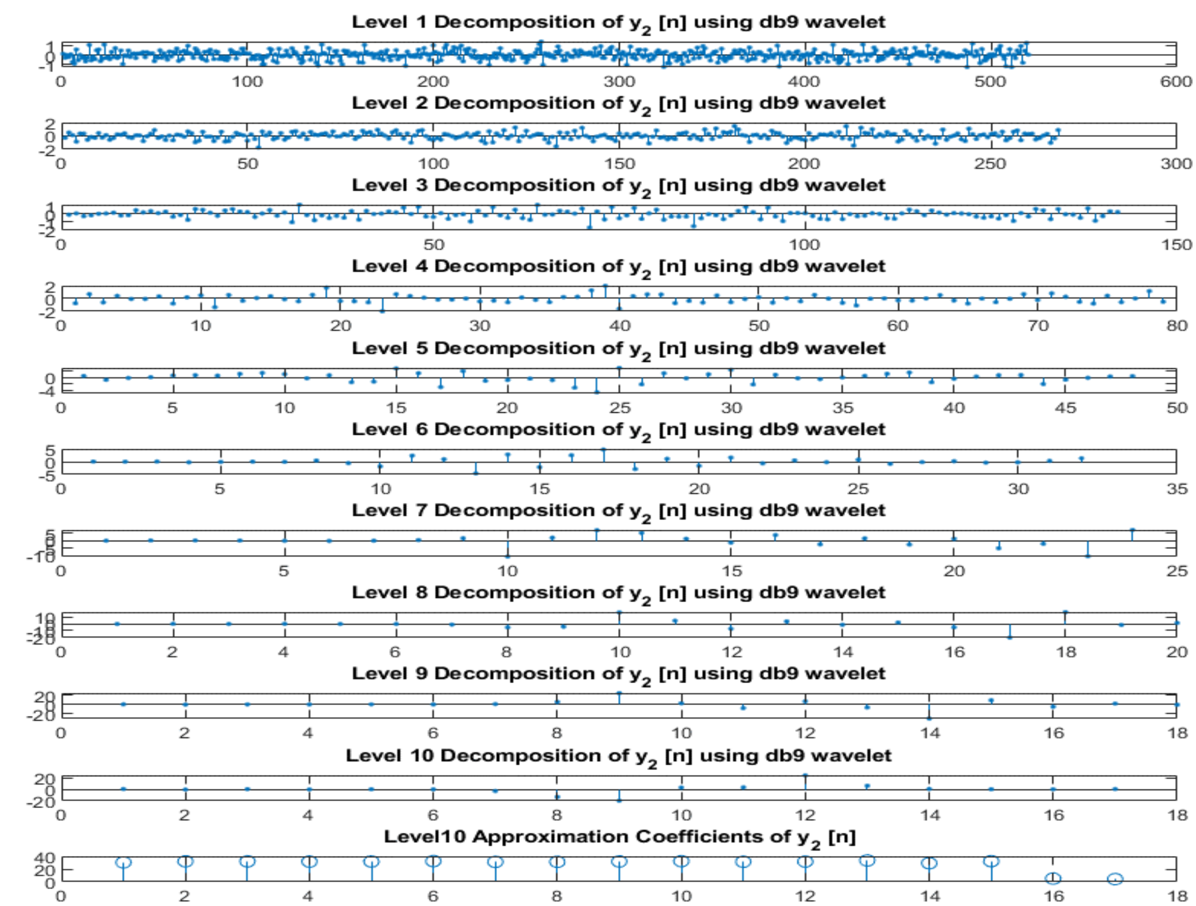

Figure 2.10 $Y_2[n]$ decomposition using 'db9' wavelet

We used inverse DWT to reconstruct the signal using A10, D10, D9, …., D2, D1, and $y = \Sigma\, D\, i + A$. We reconstructed the signal and that is very similar to the original signal. We used a recursive calculation method to reach the final signal. ((A10 , D10)=>A9 || (A9 , D9) => A8 || ….|| (A1 , D1) => original signal). The following figures show the reconstructed signals in the time domain.

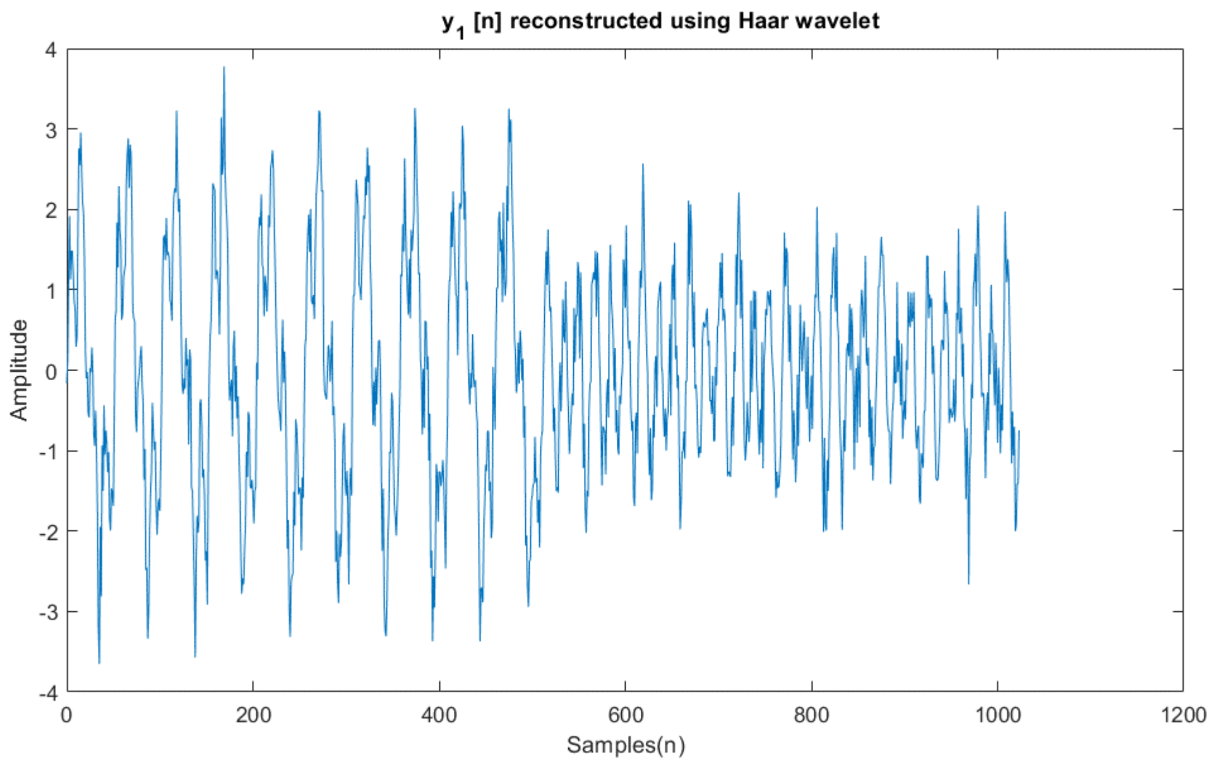

Figure 2.11 $Y_1[n]$ reconstructed | 'haar' wavelet

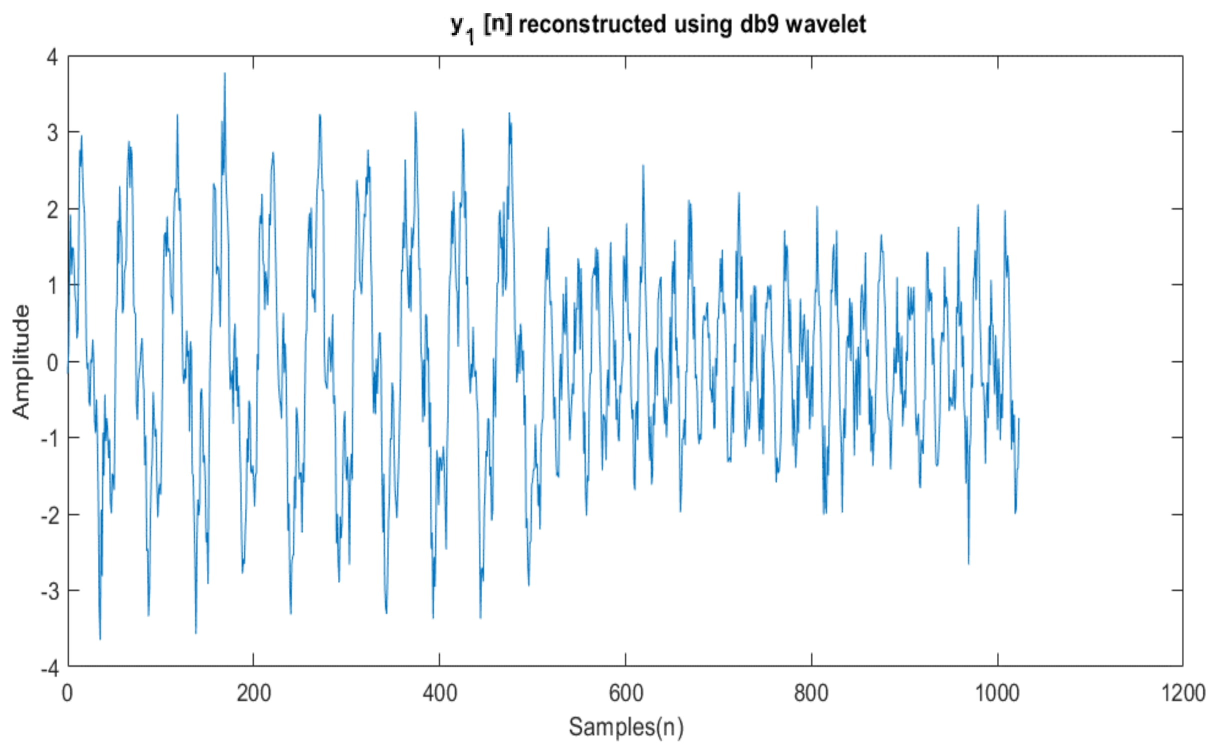

Figure 2.12 $Y_1[n]$ reconstructed | 'db9' wavelet

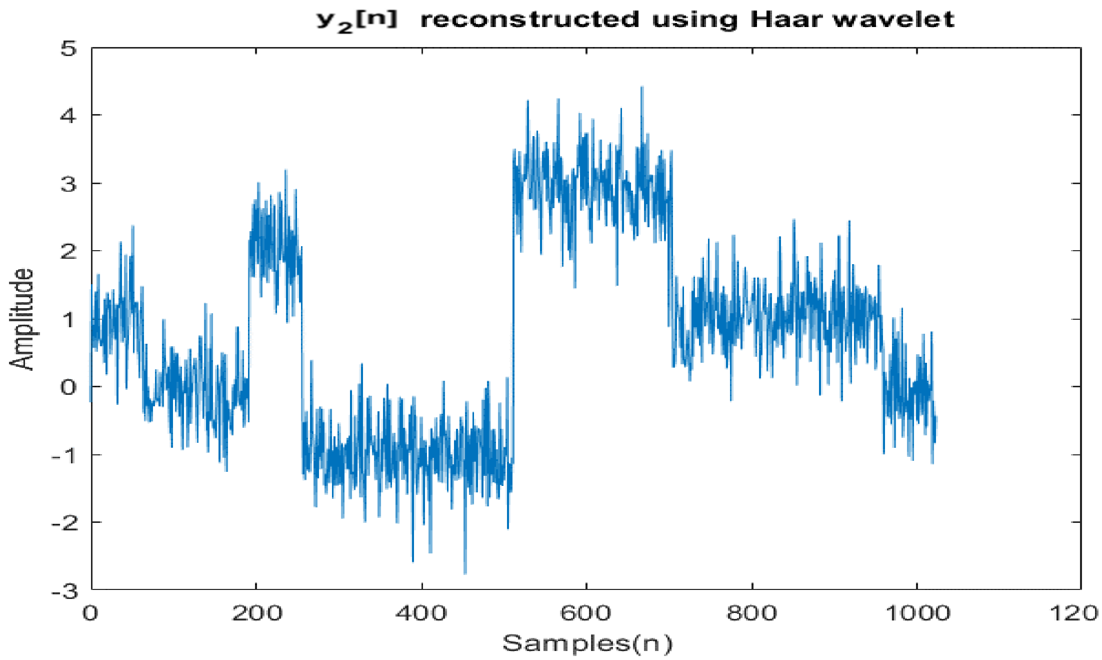

Figure 2.13 $Y_2[n]$ reconstructed |' haar' wavelet

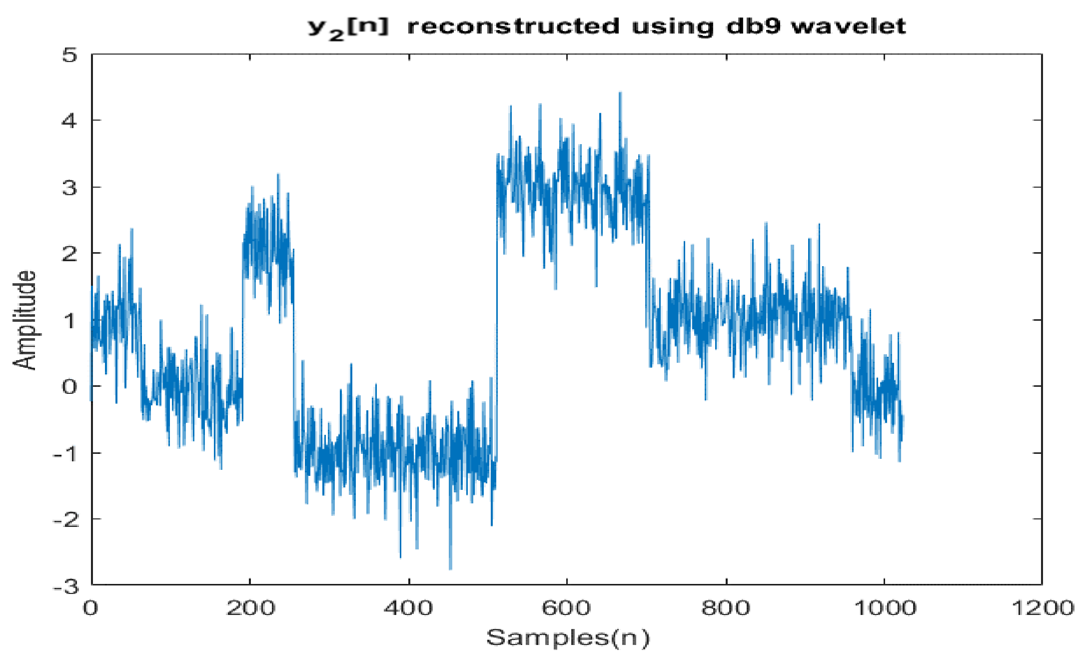

Figure 2.14 $Y_2[n]$ reconstructed | 'db9' wavelet

The calculated energy values of each reconstructed and original signal are as follows.

```
Command Window
E(y_1) = 1792.1993
E(Reconstructed y_1 | haar wavelet) = 1792.1993
E(Reconstructed y_1 | db9 wavelet) = 1792.1993

E(y_2) = 2866.2236
E(Reconstructed y_2 | haar wavelet) = 2866.2236
E(Reconstructed y_2 | db9 wavelet) = 2866.2236
```

We can observe that both reconstructed signals (using the 'haar' wavelet and the 'db9' wavelet) and the original signal have the same energy in each case. Therefore, $y = \Sigma\, D\ i + A$ can be used to reconstruct the original signal with minimum loss. In this case, the energy loss is almost 0.

2.2 Signal Denoising with DWT

We can consider a noisy signal has two components which are the original non-noisy signal and the noise part. In this case, we assume that the Energy of the non-noisy signal is captured by larger coefficients (greater than a threshold λ) whereas the noise-related coefficients are smaller ($< \lambda$). Therefore by changing coefficients $< \lambda$ to zero, a reduced noise estimation of the original signal can be reconstructed.

The following figure shows the magnitude of wavelet coefficients (stem plot) in descending order in the case of using the 'haar' wavelet with noisy signal y_1.

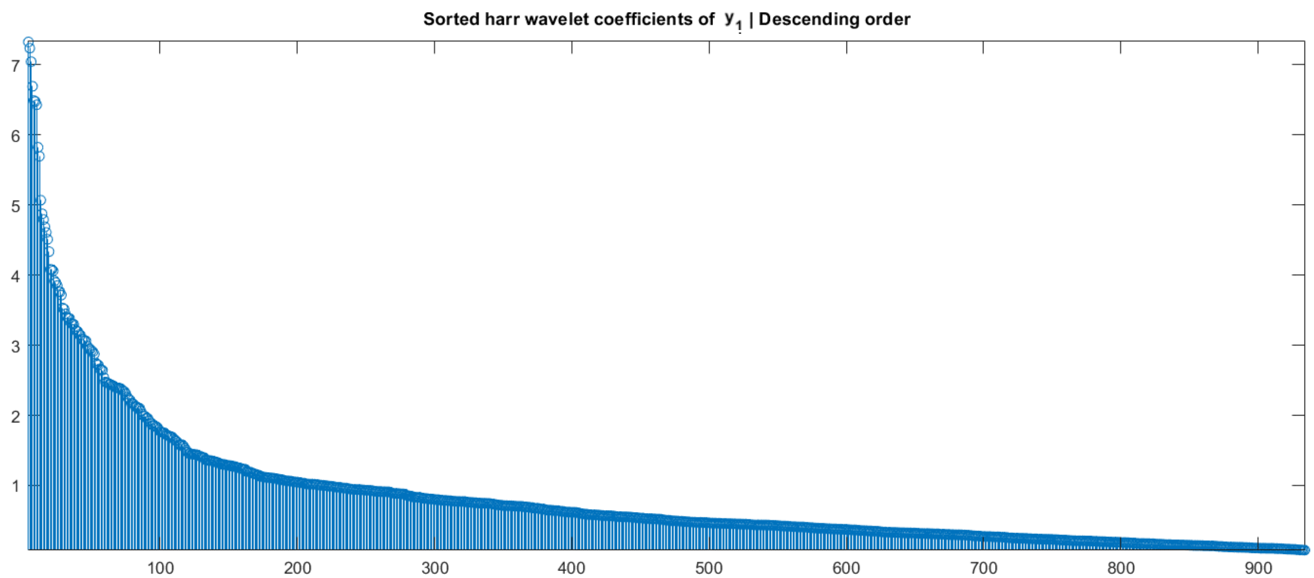

Figure 2.15 sorted haar wavelet coeff. (y_1)

By observing the above graph and by doing several trials we selected the threshold as 1. The reconstructed signal and the original signal are comparatively shown in the following figures.

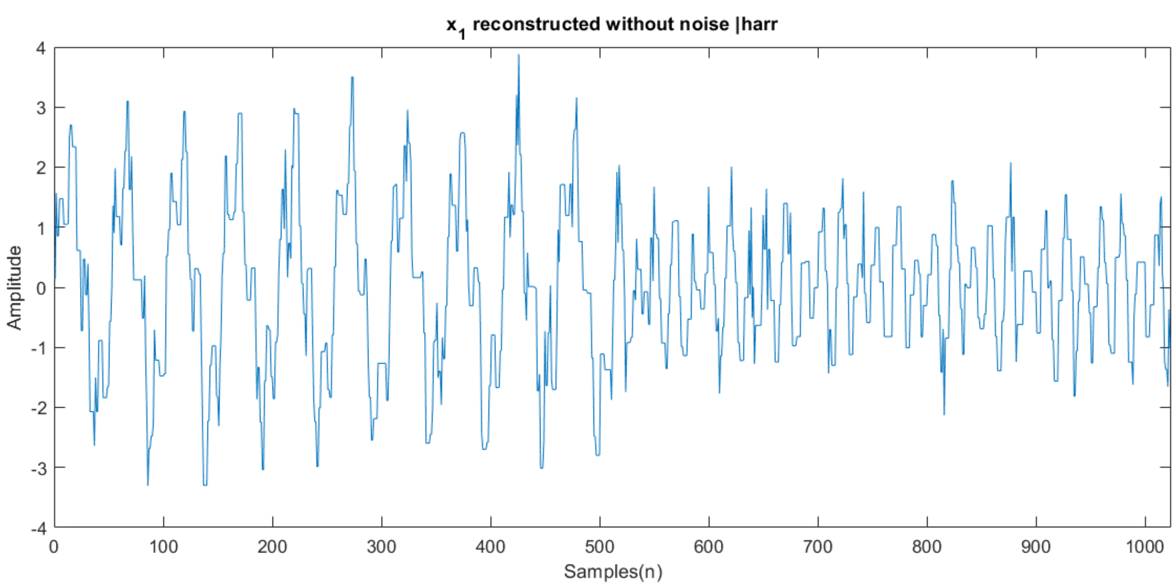

Figure 2.16 constructed x_1 | harr wavelet

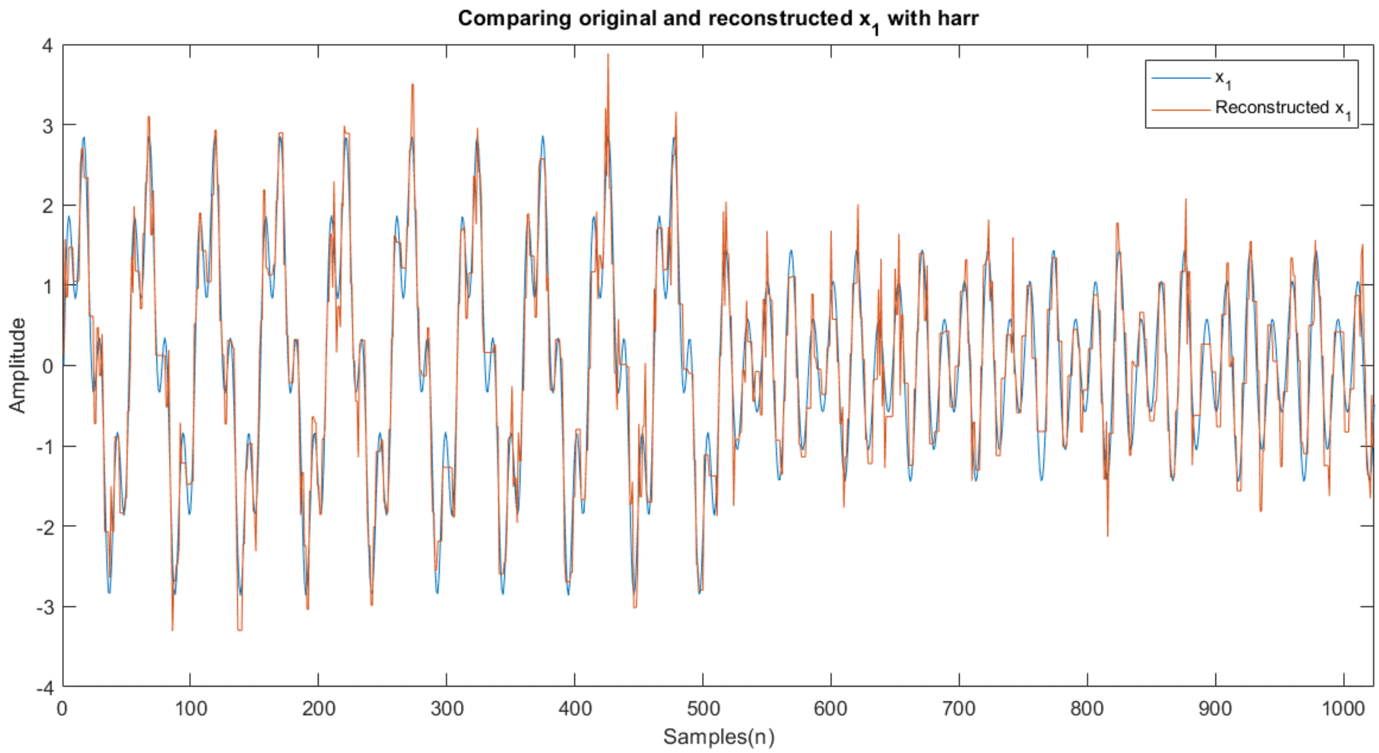

Figure 2.17 Comparison between original x_1 and reconstructed x_1

RMSE (x_1 reconstructed | haar wavelet) = 0.39431

The following figure shows the magnitude of wavelet coefficients (stem plot) in descending order in the case of using the 'db9' wavelet with noisy signal y_1.

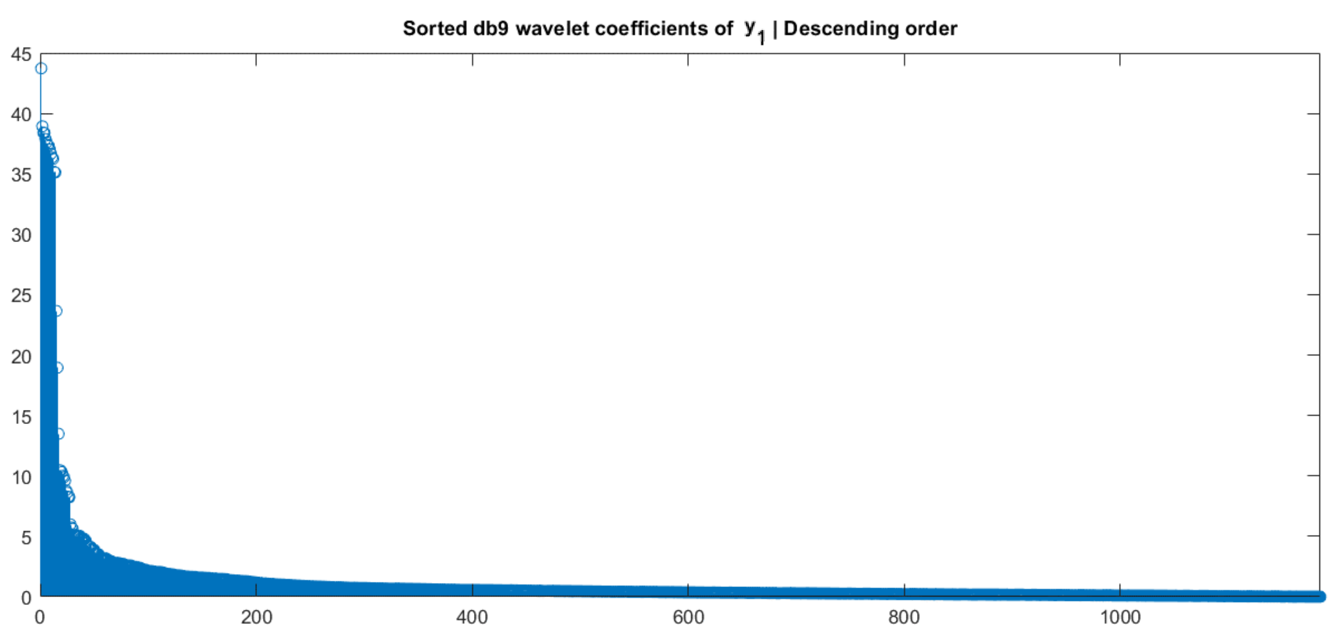

Figure 2.18 sorted 'haar' wavelet coeff. (x_1)

By observing the above graph and by doing several trials we selected the threshold as 1. The reconstructed signal and the original signal are comparatively shown in the following figures.

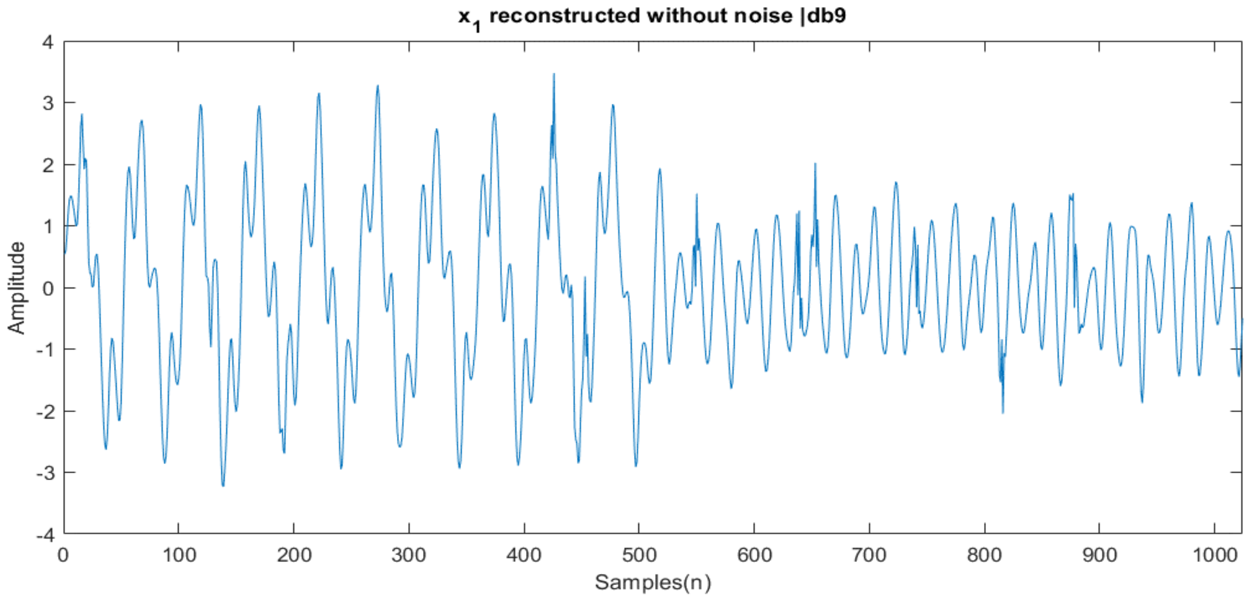

Figure 2.19 constructed x_1 | 'db9' wavelet

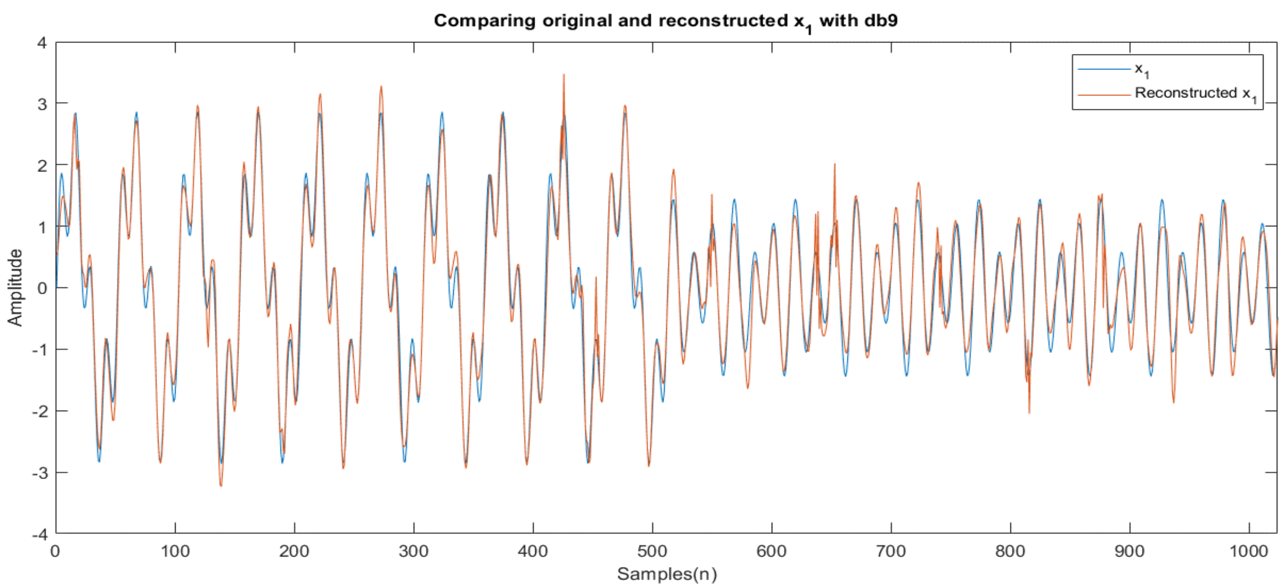

Figure 2.20 Comparison between original x_1 and reconstructed x_1

RMSE (x_1 reconstructed | db9 wavelet) = 0.25644

In this case, the root means square (RMSE) value is lower in the case of using the 'db9' wavelet compared to the case of the 'haar' wavelet. Since the haar wallet has rapid transition and 'db9' have smooth transitions it is preferable to use 'db9' to filter the noises of smooth signals (sinusoidal variations). This fact is further proven by visual inspection of the above comparison plots (figure 2.20 and figure 2.17). From that figure, we can observe that the reconstructed signal in the case of the 'db9' wavelet is more similar to the original x_1 compared to the other case.

The following figure shows the magnitude of wavelet coefficients (stem plot) in descending order in the case of using the 'haar' wavelet with noisy signal y_2.

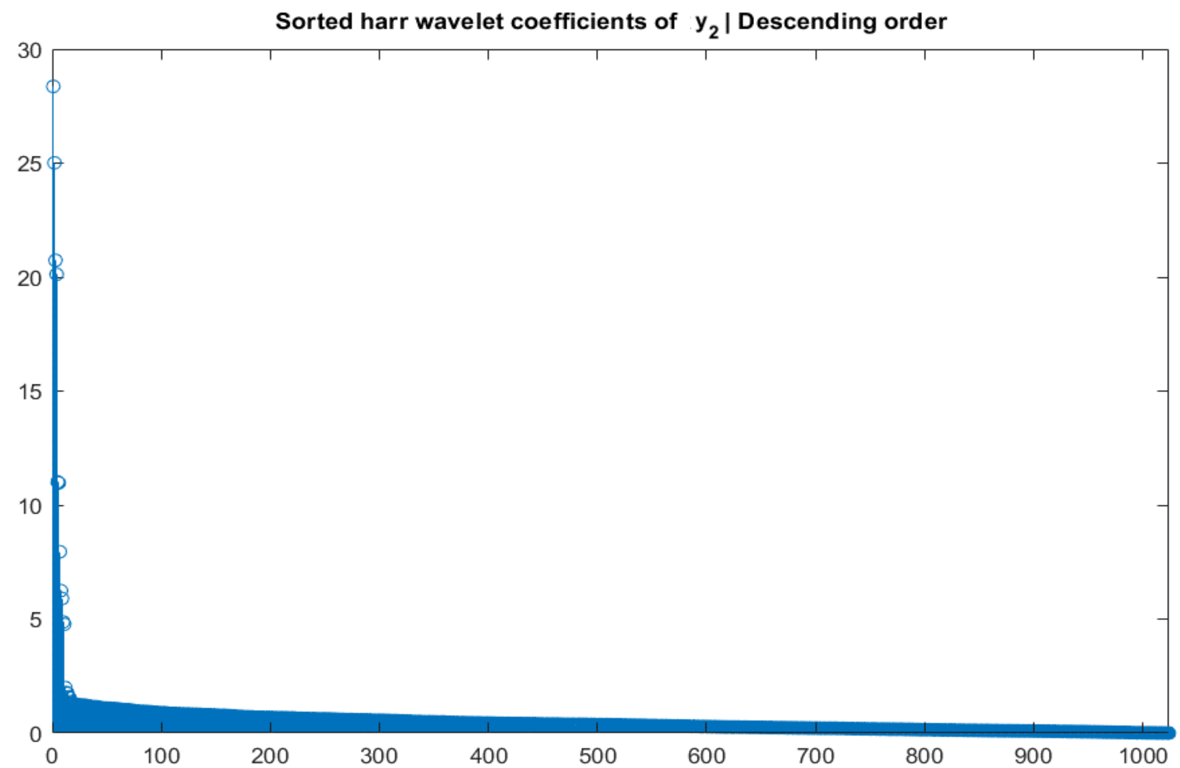

Figure 2.21 sorted 'haar' wavelet coeff. (y_2)

By observing the above graph and by doing several trials we selected the threshold as 2. The reconstructed signal and the original signal are comparatively shown in the following figures.

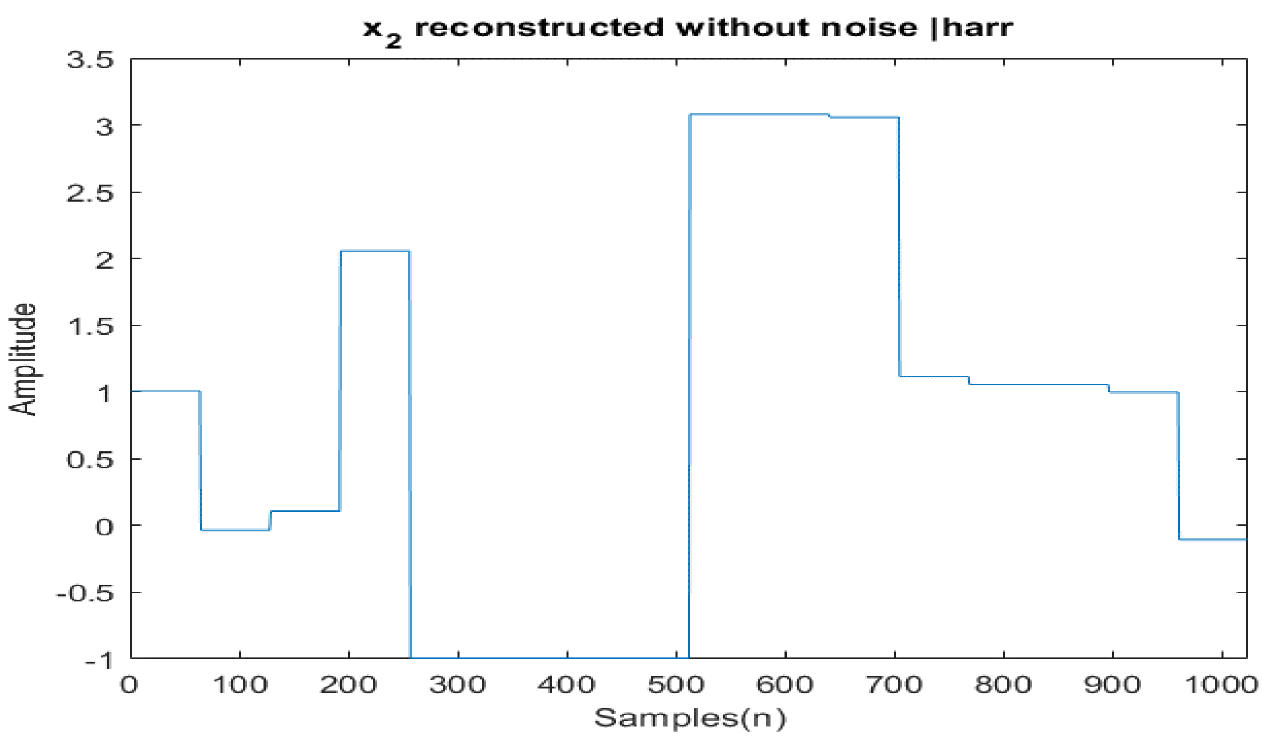

Figure 2.22 constructed x_2 | 'harr' wavelet

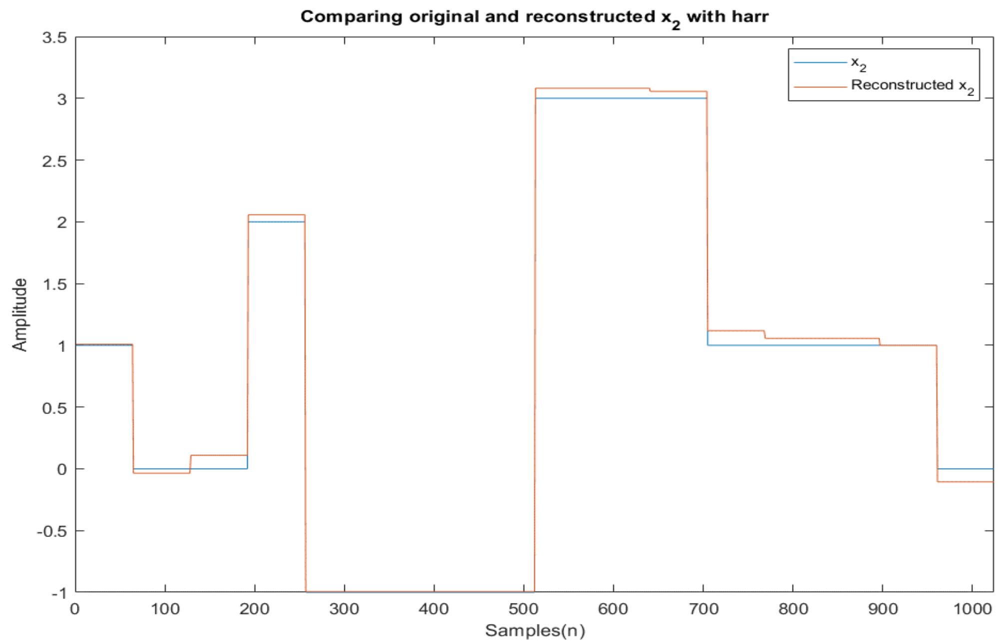

Figure 2.23 Comparison between original x2 and reconstructed x2 |' haar'

RMSE (x_2 reconstructed | haar wavelet) = 0.063365

The following figure shows the magnitude of wavelet coefficients (stem plot) in descending order in the case of using the 'haar' wavelet and with noisy signal y_2.

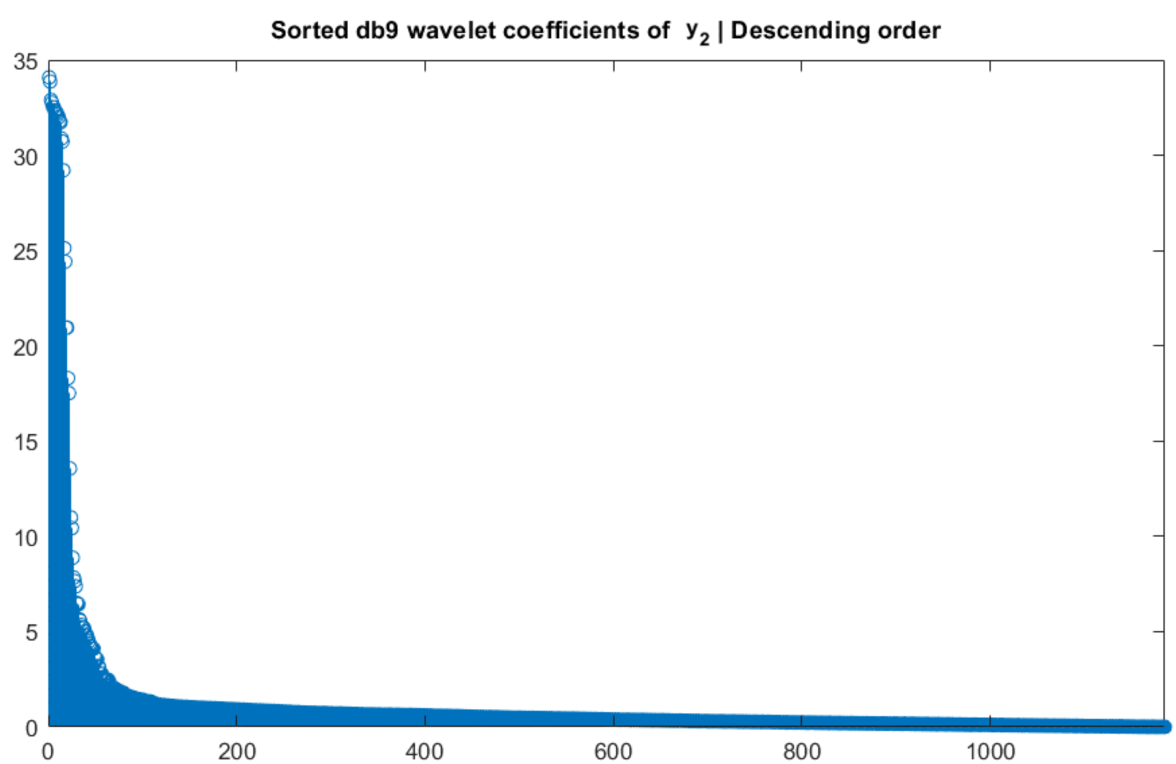

Figure 2.24 sorted 'db9' wavelet coeff.

By observing the above graph and by doing several trials we selected the threshold as 2. The reconstructed signal and the original signal are comparatively shown in the following figures.

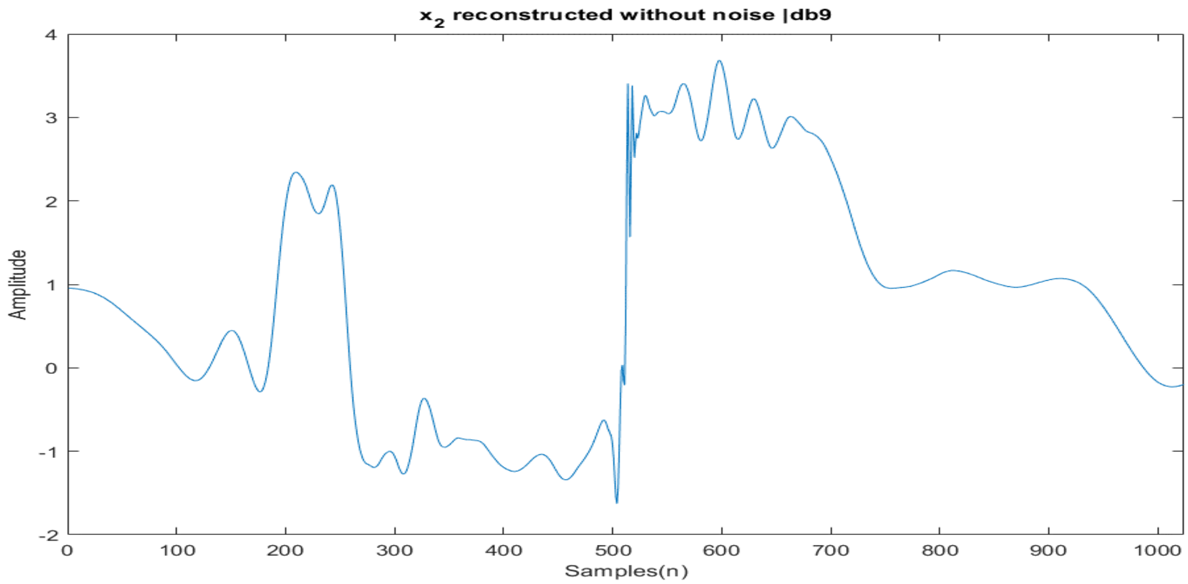

Figure 2.25 constructed x_2 | 'db9' wavelet

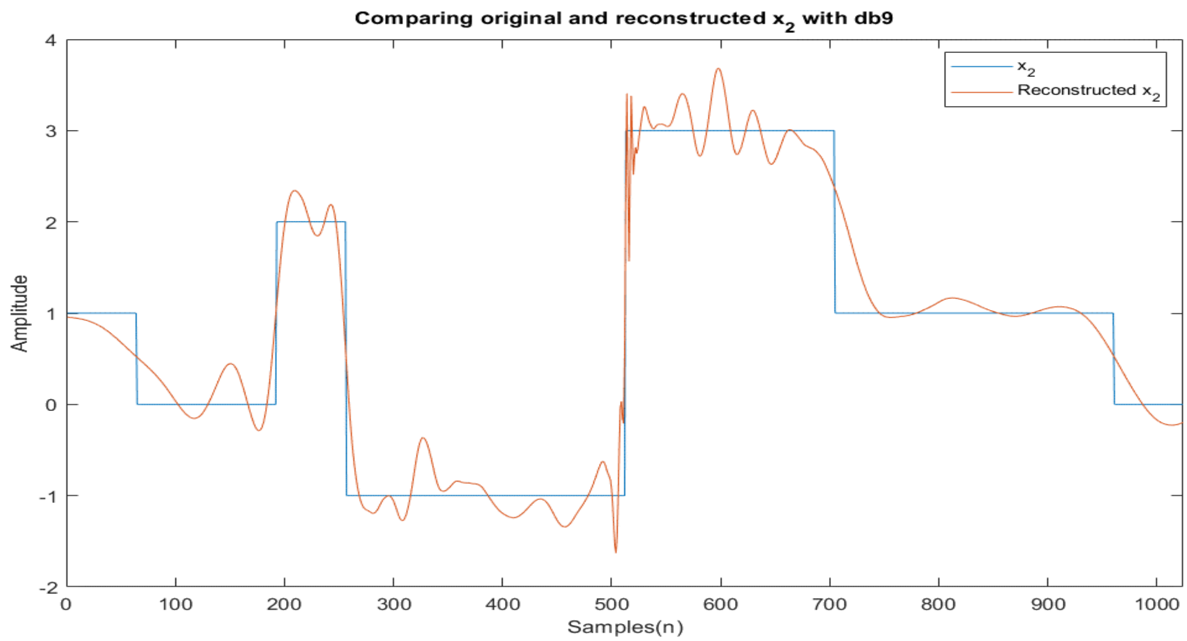

Figure 2.26 Comparison between original x2 and reconstructed x2 | 'db9'

RMSE (x_2 reconstructed | db9 wavelet) = 0.30474

In this case, the root means square (RMSE) value is lower in the case of using a haar wavelet compared to the case of a 'db9' wavelet. Since the haar wavelet has rapid transition and 'db9' have smooth transitions it is preferred to use the 'Haar' wavelet to filter the noises of rapid transition signals with noise(y_2). This fact is further proven by visual inspection of the above comparison plots (figure 2.26 and figure 2.23). From that figure, we can observe that the reconstructed signal in the case of the 'haar' wavelet is more similar to the original x_1 compared to the other case.

For $\mathbf{y_1}$ => 'db9' wavelet

For $\mathbf{y_2}$ => 'haar' wavelet

2.3 Signal Compression with DWT

For the compression process, we followed the following 7 steps.

- Step 1: Perform a wavelet transform of the signal.
- Step 2: Arrange the coefficients in decreasing order of magnitude. In this case, we use a maximum number of dyadic wavelet decompositions ($\log_2(2570) \approx 12$) for proper compression.
- Step 3: Choose a threshold value such that a significant amount of energy is retained using the accumulation of energy components
- Step 4: Set all values of the wavelet transform which are insignificant to 0. i.e., which lie below a threshold.
- Step 5: Save only the significant, non-zero values of the transform obtained from Step 2.
- Step 6: To reconstruct the signal, perform the inverse wavelet transform of the data saved. This produces an estimation of the original signal

In this part we have the aVR lead of an ECG sampled at 257 Hz in 'ECGsig.mat'. The following figures show the time domain representations and the wavelength decomposition coefficients for both 'haar' and 'db9' wavelet cases.

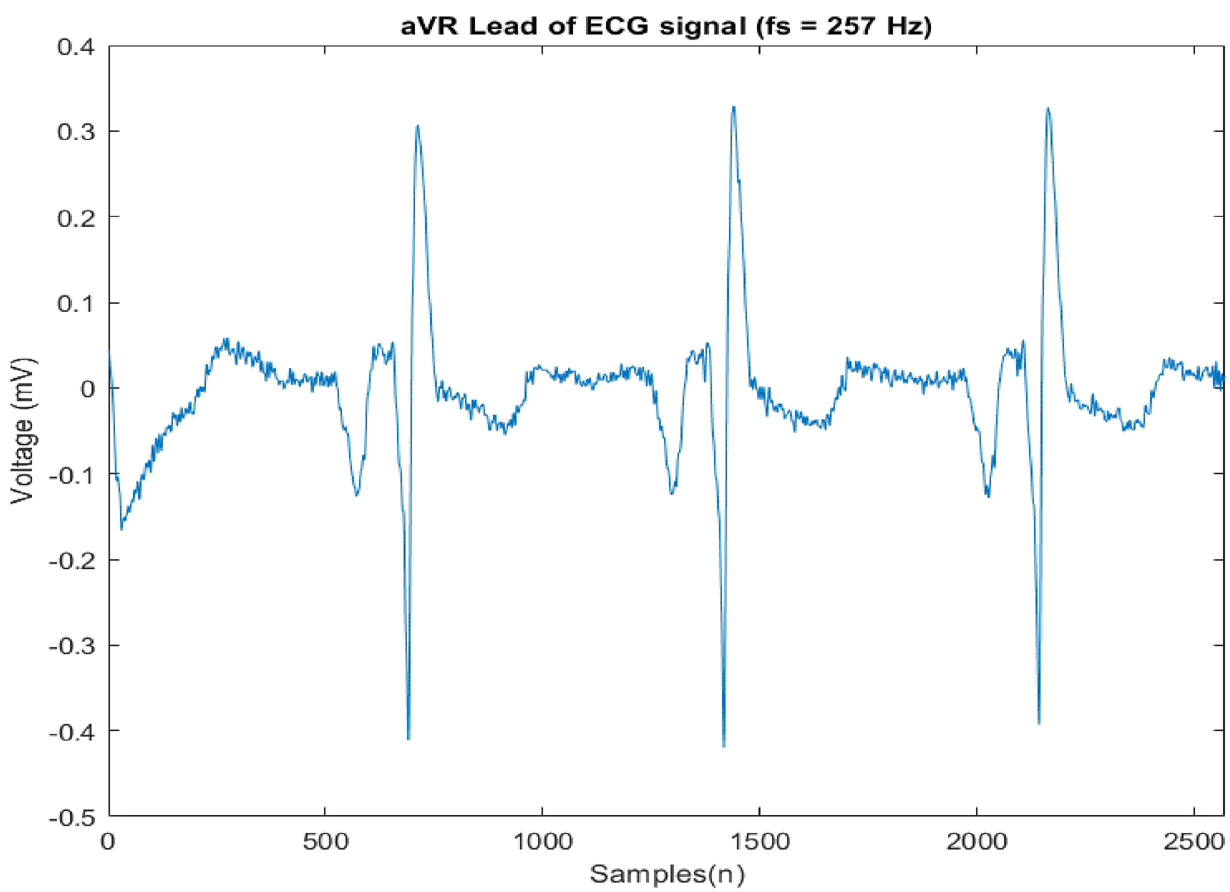

Figure 2.27 aVR signal in time domain

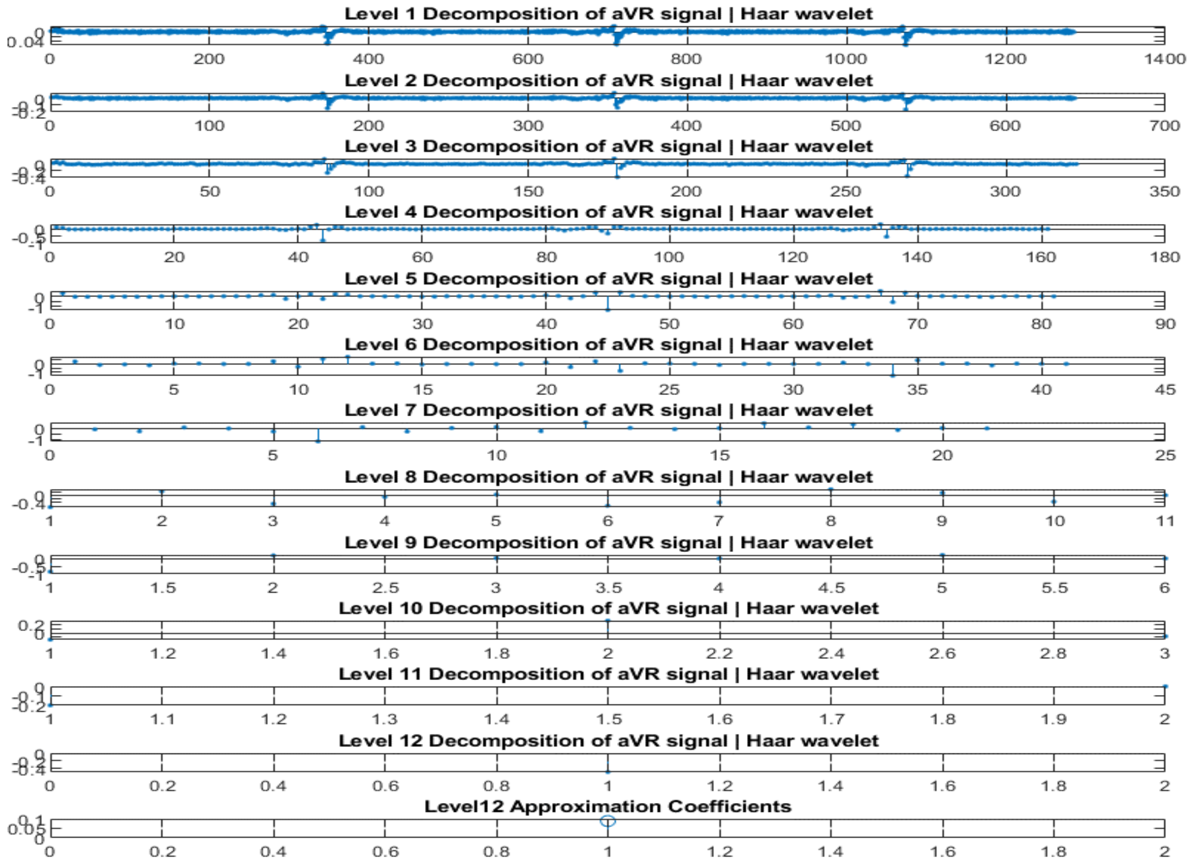

Figure 2.28 aVR decomposition using 'haar' wavelet

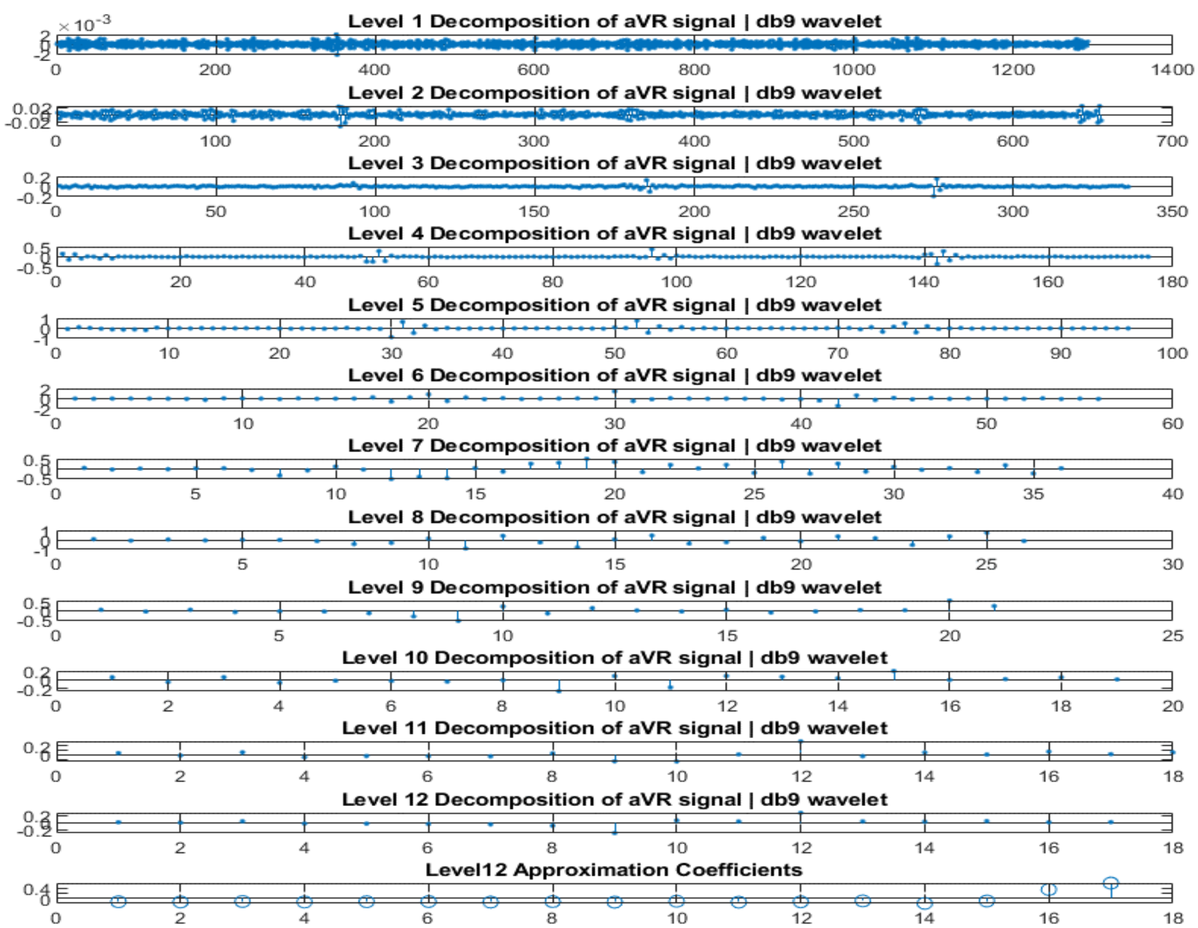

Figure 2.29 aVR decomposition using 'db9' wavelet

The following figures show the sorted coefficients, compressed signal, and original signal comparatively for both cases (‘haar’, ‘db9’).

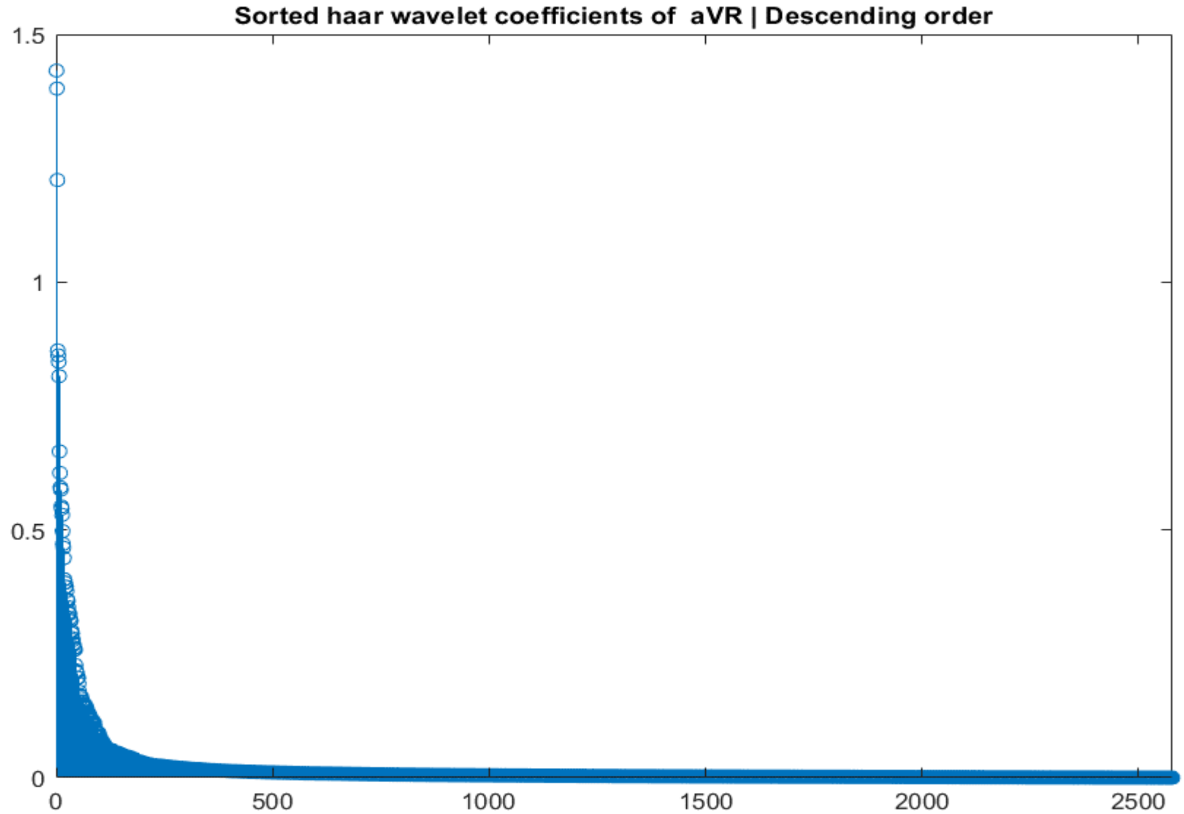

Figure 2.30 sorted coefficients(‘haar’) for aVR

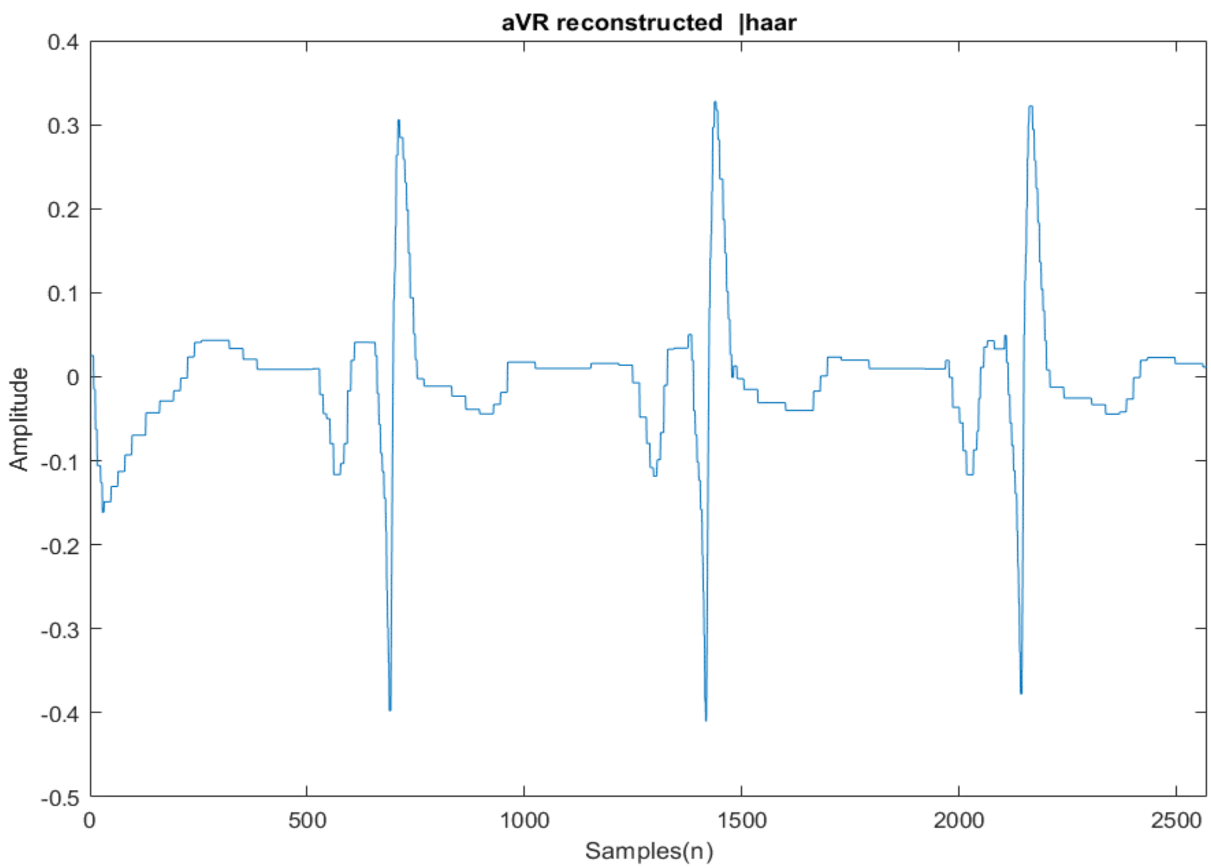

Figure 2.31 constructed aVR signal using compressed data | ‘haar’

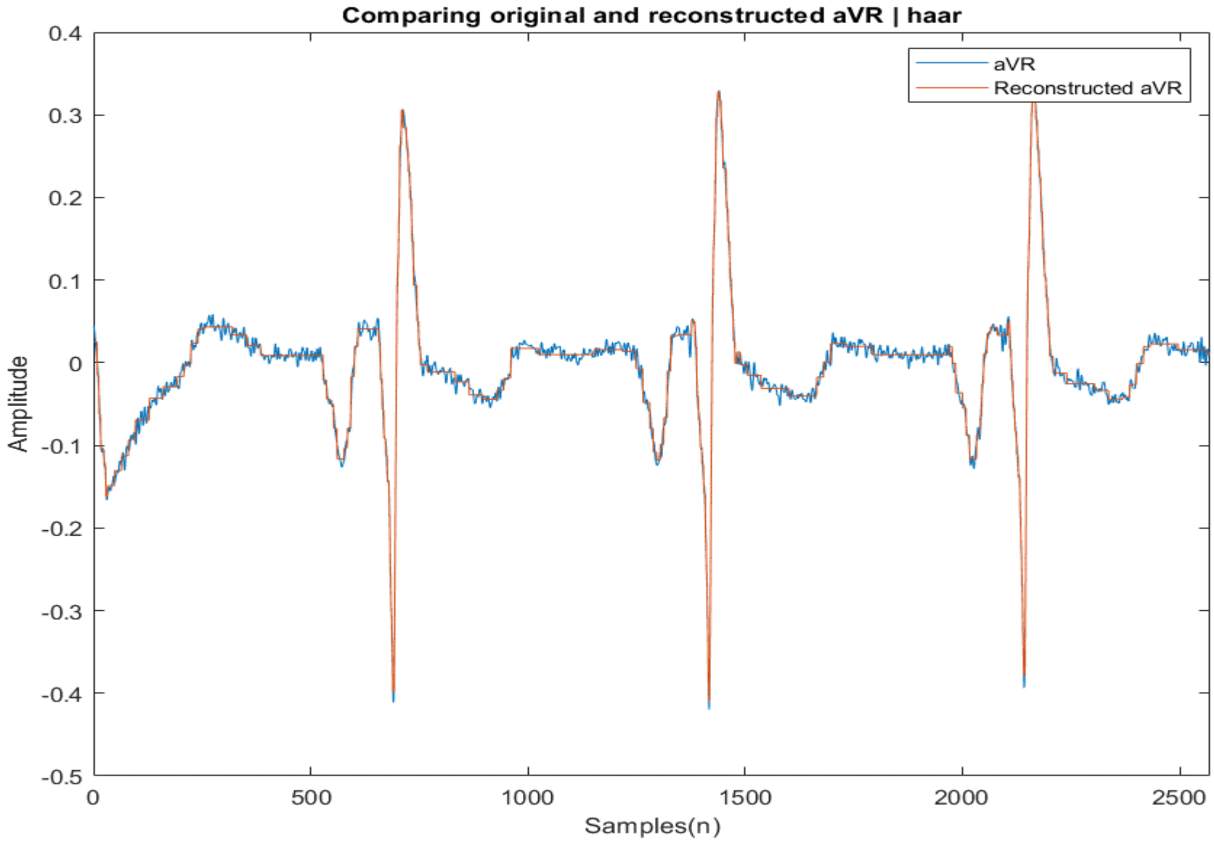

Figure 2.32 aVR signal(original) and constructed using compressed data | 'haar'

Number coefficients required for 99% of the energy of the signal = 177

Compression Ratio = 14.5198 : 1

RMSE (aVR reconstructed | 'haar' wavelet) = 0.0078202

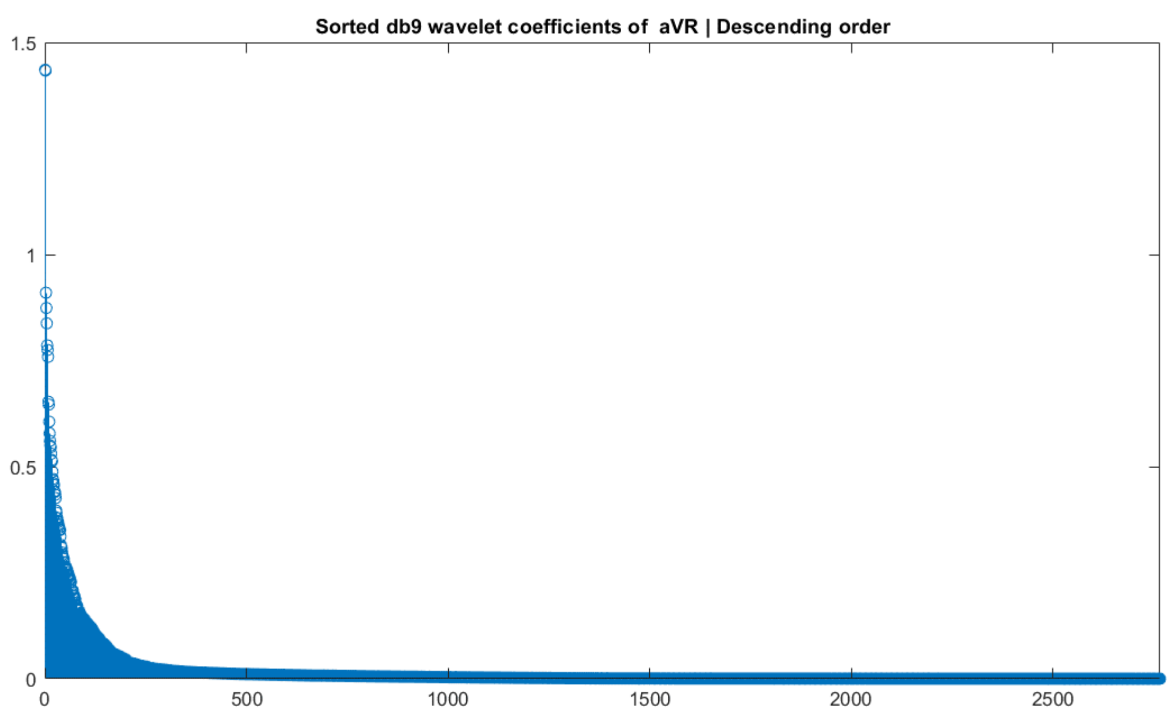

Figure 2.33 sorted coefficients('db9') for aVR

Figure 2.34 constructed aVR signal using compressed data | 'db9'

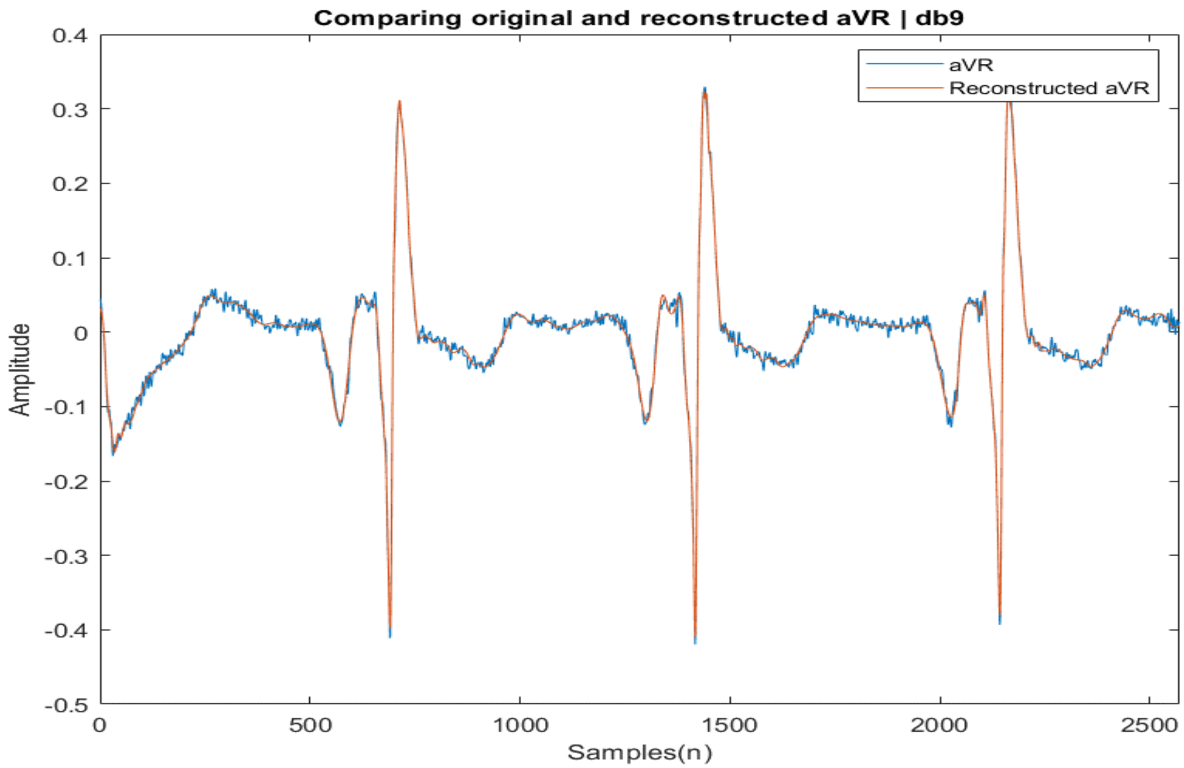

Figure 2.35 aVR signal(original) and constructed using compressed data | 'db9'

Number coefficients required for 99% of the energy of the signal = 189

Compression Ratio = 13.5979 : 1

RMSE aVR reconstructed | 'db9' wavelet = 0.0068967

We can observe that the RMSE value of the reconstructed signal from the case of ‘db9’ has a lower value compared to the case of the ‘haar’ wavelet. So, the compressed signal from the case of ‘db9’ is more similar to the original signal. Because of the sharp edges of the ‘haar’ wavelet, it is impossible to recover the original signal. In both compression cases, the compression process removes the very lower coefficients which correspond to the lower energies. Since most noise components have lower energies those higher-frequency noise components are removed. So that we have smoothened signal in both compression cases. Both cases have close compression ratios. But the case of ‘haar’ has a higher compression rate compared to the case of the ‘db9 case.

Note: When finding the number of coefficients required to represent 99% of energy, we didn’t use any rounding when calculating cumulative energy. Instead, we use an exact number of components at least needed to represent 99% of energy (loop break condition: (cum_energy/total_energy) >= 0.99). If we use rounded cumulative energy then the number of components required to represent 99% of energy will be lower than the values we used here. If we assumed that cum_energy rounded for two decimal points then,

For ‘haar’ case => no of components:138 , Compression ratio : 18.6232 : 1

For ‘db9’ case => no of components:161 , Compression ratio : 15.9627 : 1