UNIVERSITY OF MORATUWA

Faculty of Engineering

Biosignal Processing Assignment 3

Continuous and Discrete Wavelet Transforms

Course: BM4152 Biosignal Processing
Name: Kavishan G.T.
ID: 210285M
Date: October 4, 2025

Department of Electronic and Telecommunication Engineering

Contents

List of Figures

1. Continuous Wavelet Transform

1.1 Introduction

The continuous wavelet transform is defined by the following equation.

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

where $s =$ scaling factor, $\tau =$ translation, $\psi =$ wavelet function.

1.2 Wavelet properties

Derivation of the Mexican Hat Function

The Mexican Hat (Ricker) wavelet $m(t)$ can be obtained by taking the negative of the second derivative of the Gaussian function $g(t)$.

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

The Gaussian function is defined as

$$g(t) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2\right)$$

where $\mu = 0$ and $\sigma = 1$. Thus,

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

First derivative:

$$\frac{dg(t)}{dt} = -\frac{t}{\sqrt{2\pi}}e^{-t^2/2}.$$

Second derivative:

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

Hence, the Mexican Hat function is

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

Normalization of the Mexican Hat Function

To normalize, the energy E must satisfy

$$E = \int_{-\infty}^{\infty} m^2(t)\, dt = 1.$$

Substitute $m(t)$:

$$E = \int_{-\infty}^{\infty} \left[\frac{1-t^2}{\sqrt{2\pi}} e^{-t^2/2}\right]^2 dt.$$

$$E = \frac{1}{2\pi} \int_{-\infty}^{\infty} (1 - 2t^2 + t^4) e^{-t^2} dt.$$

The Gaussian integral identities are:

$$\int_{-\infty}^{\infty} e^{-t^2} dt = \sqrt{\pi}, \quad \int_{-\infty}^{\infty} t^2 e^{-t^2} dt = \frac{\sqrt{\pi}}{2}, \quad \int_{-\infty}^{\infty} t^4 e^{-t^2} dt = \frac{3\sqrt{\pi}}{4}.$$

Substituting:

$$E = \frac{1}{2\pi} \left[\sqrt{\pi} - 2 \cdot \frac{\sqrt{\pi}}{2} + \frac{3\sqrt{\pi}}{4}\right].$$

$$E = \frac{1}{2\pi} \cdot \frac{3\sqrt{\pi}}{4} = \frac{3}{8\sqrt{\pi}}.$$

Thus, the normalizing factor K is

$$K = \sqrt{\frac{1}{E}} = \sqrt{\frac{8\sqrt{\pi}}{3}}.$$

Normalized Mexican Hat Wavelet

The normalized mother wavelet is therefore

$$\psi(t) = K \cdot m(t) = \sqrt{\frac{8\sqrt{\pi}}{3}} \cdot \frac{1-t^2}{\sqrt{2\pi}} e^{-t^2/2}.$$

Generic Wavelet with Scaling Factor

Including the scaling factor s, the wavelet is

$$\psi_s(t) = \frac{1}{\sqrt{s}}\, \psi\left(\frac{t}{s}\right).$$

Thus, the normalized Mexican Hat daughter wavelet becomes

$$\psi_s(t) = \frac{1}{\sqrt{s}} \cdot \sqrt{\frac{8\sqrt{\pi}}{3}} \cdot \frac{1 - \left(\frac{t}{s}\right)^2}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2s^2}\right).$$

MATLAB Implementation

The provided script `wavelet_construction.m` can be completed to generate daughter wavelets for scaling factors $s = 0.01 : 0.1 : 2$. The time-domain waveforms should be plotted and inspected for compact support.

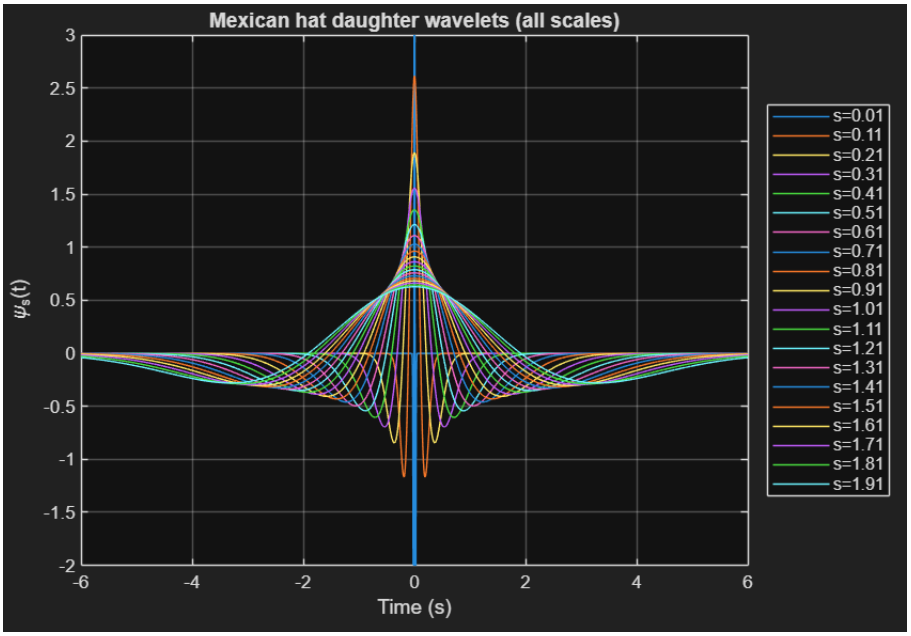

Figure 1: Mexican hat daughter wavelets

Verification of Wavelet Properties

Table 1: Verification of Mexican Hat Wavelet Properties for Different Scales

Scale	Mean	Energy
0.01	-2.0817×10^{-17}	1
0.11	-2.5728×10^{-18}	1
0.21	-5.9030×10^{-18}	1
0.31	-1.0832×10^{-17}	1
0.41	3.0179×10^{-18}	1
0.51	-7.1699×10^{-20}	1
0.61	-1.4494×10^{-17}	1
0.71	-3.4162×10^{-17}	1
0.81	-1.8974×10^{-18}	1
0.91	-1.7998×10^{-17}	1
1.01	-1.9082×10^{-17}	1
1.11	8.6736×10^{-18}	1
1.21	5.8981×10^{-17}	1
1.31	5.8981×10^{-17}	1
1.41	3.3168×10^{-15}	1
1.51	3.2731×10^{-13}	1
1.61	1.4181×10^{-11}	1
1.71	3.2232×10^{-10}	1
1.81	4.4149×10^{-9}	1
1.91	4.0412×10^{-8}	1

- **Zero mean:** Verified numerically as $\int \psi_s(t)\, dt \approx 0$.
- **Unity energy:** Verified by $\int \psi_s^2(t)\, dt = 1$ after normalization.

- **Compact support:** The wavelets decay rapidly outside a finite time window (observable in time-domain plots).

Spectral Analysis

Using the same script, the Fourier transforms of the daughter wavelets are computed and plotted. The spectra confirm the band-pass nature of the Mexican Hat wavelet, with peak frequency shifting as the scaling factor s is varied.

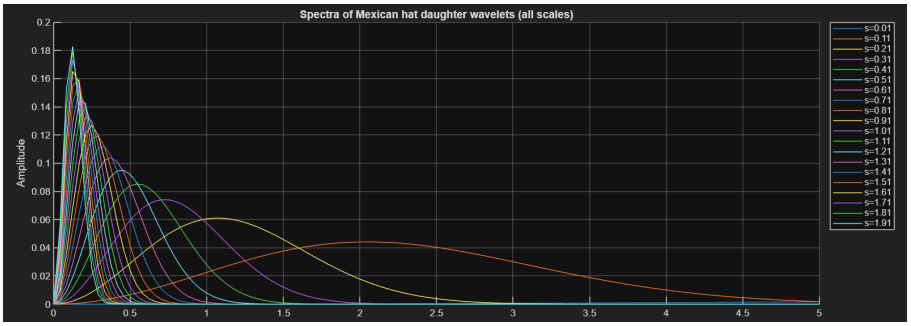

Figure 2: Spectra of Mexican hat daughter wavelets

1.3 Continuous Wavelet Decomposition

Signal Generation

The discrete-time signal $x[n]$ is defined piecewise as:

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

The sampling frequency is $f_s = 250$ Hz. In MATLAB, this can be generated as:

Listing 1: Sine wave generation

```
%% Parameters
fs = 250;    % Sampling frequency
N = 3000;    % Segment length (adjust as needed)
n = 1:3*N;

x = zeros(size(n));
x(n >= 1 & n < (3*N)/2) = sin((1/fs)*0.5 * pi * n(n >= 1 & n < (3*N)/2)
    );
x(n >= (3*N)/2 & n < 3*N) = sin((1/fs)*1.5 * pi * n(n >= (3*N)/2 & n <
    3*N));
```

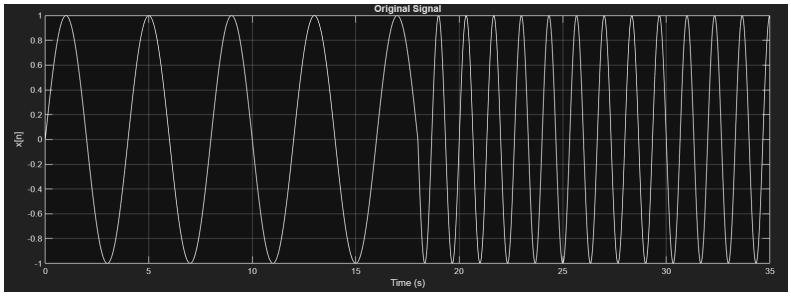

Figure 3: Generated Sine Wave

Continuous Wavelet Transform Using Mexican Hat

The continuous wavelet transform (CWT) of $x[n]$ using the scaled Mexican Hat wavelet $\psi_s(t)$ is computed as:

$$W(s,\tau) = \sum_n x[n]\,\psi_s[n-\tau]$$

where τ represents translation and s the scale factor. To achieve high scale resolution, scales were set from $0.01 : 0.01 : 2$
The magnitude of the wavelet coefficients can be visualized as a spectrogram using `pcolor()`:

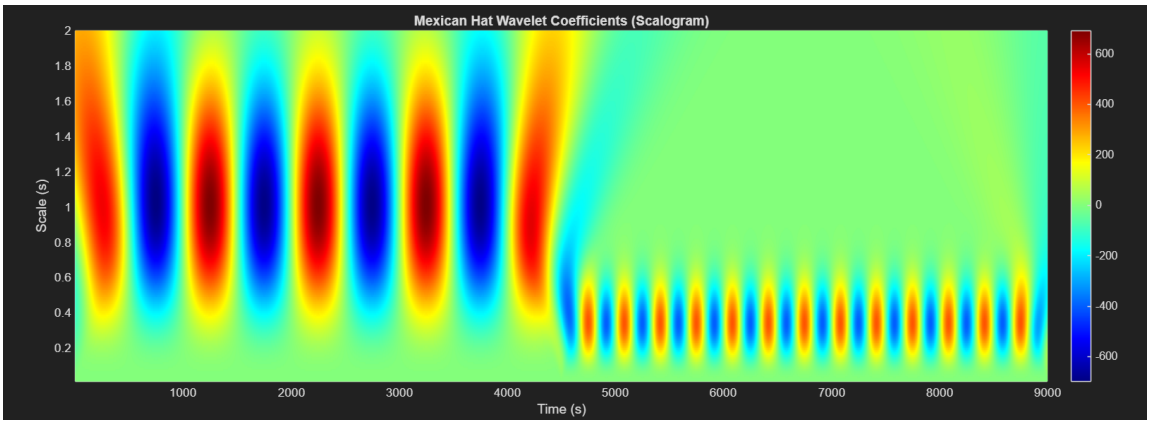

Figure 4: Visualizing the spectrogram

Observation and Comments

- The CWT spectrogram represents the frequency content of $x[n]$ at different times.
- Low scales correspond to high-frequency components, and high scales correspond to low-frequency components.
- In the plot, the first segment ($\sin(0.5\pi n)$) shows energy concentrated at a lower frequency, while the second segment ($\sin(1.5\pi n)$) appears at higher scales corresponding to higher frequency content.
- The spectrogram provides a time-frequency representation, allowing identification of frequency changes over time.

2. Discrete Wavelet Transform (DWT)

Introduction

The Continuous Wavelet Transform (CWT) suffers from redundancy due to continuous scaling and translation, requiring significant computational power and time. To overcome this, the **Discrete Wavelet Transform (DWT)** introduces discretized scaling and translation:

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

where

$$s_0 = \text{scaling step size}, \quad \tau_0 = \text{translation step size}.$$

Typically, $s_0 = 2$ and $\tau_0 = 1$ are chosen for efficient analysis. The integers m and n determine the scale and translation indices.

Applying DWT with the Wavelet Toolbox in MATLAB

Signal Creation

Two signals were defined as follows:

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

$$x_2[n] = \begin{cases} 1, & 0 \le n < 64 \\ -1, & 128 \le n < 512 \\ 2, & 192 \le n < 256 \\ 3, & 512 \le n < 704 \\ 1, & 704 \le n < 960 \\ 0, & \text{otherwise} \end{cases}$$

The sampling frequency is $f_s = 512$ Hz.

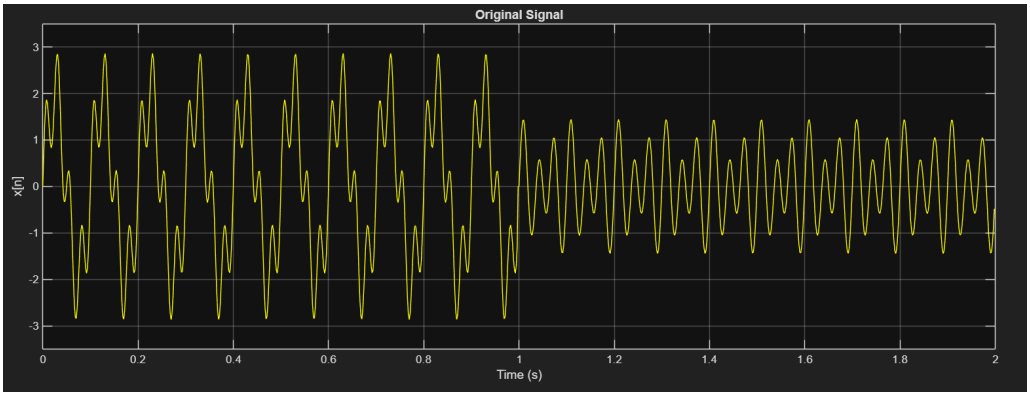

Figure 5: Generated signals $x_1[n]$ (piecewise sinusoidal)

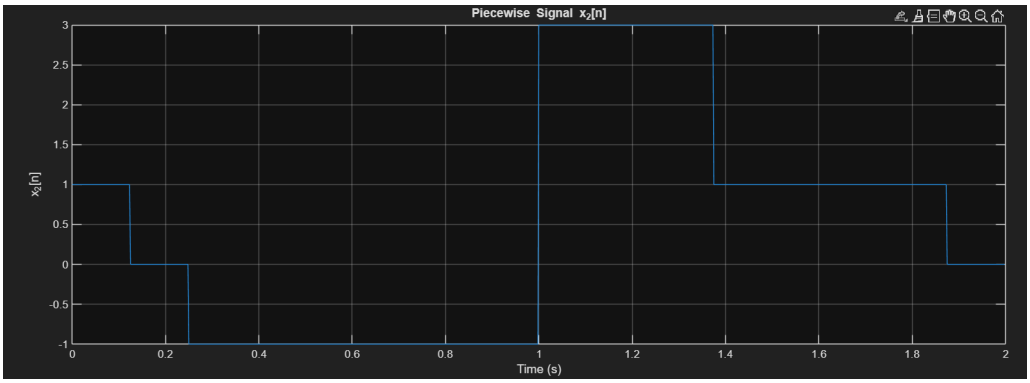

Figure 6: Generated signals $x_2[n]$ (piecewise constant).

Corruption with Noise

The signals were corrupted with Additive White Gaussian Noise (AWGN) at 10 dB SNR:

```
y1 = awgn(x1, 10, 'measured');
y2 = awgn(x2, 10, 'measured');
```

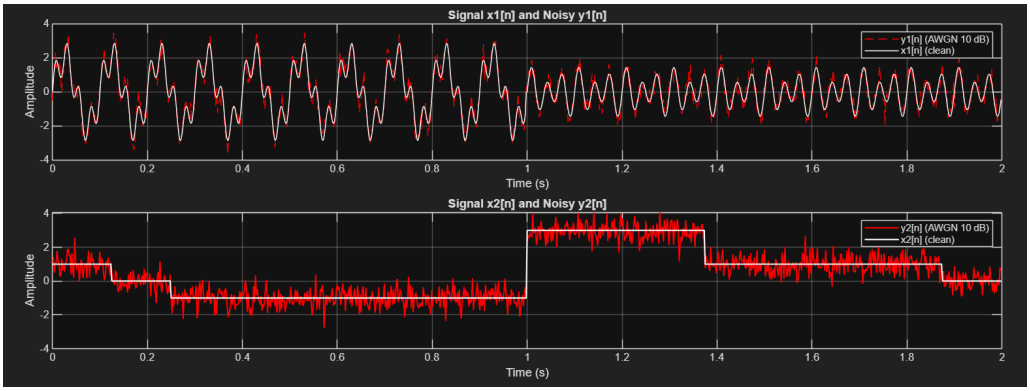

Figure 7: Original and noisy signals. Top: $x_1[n]$ vs $y_1[n]$, Bottom: $x_2[n]$ vs $y_2[n]$.

Wavelet and Scaling Functions

The morphology of the Haar and Daubechies-9 (db9) wavelet and scaling functions were examined using the `wavefun()` command.

```
[phi_haar,psi_haar,x_haar] = wavefun('haar',10); % level = 10 for
    resolution
figure;
subplot(2,1,1);
plot(x_haar, phi_haar, 'y','LineWidth',1.2);
title('Haar Scaling Function \phi');
xlabel('t'); ylabel('\phi(t)'); grid on;

subplot(2,1,2);
plot(x_haar, psi_haar, 'y','LineWidth',1.2);
title('Haar Wavelet Function \psi');
xlabel('t'); ylabel('\psi(t)'); grid on;
```

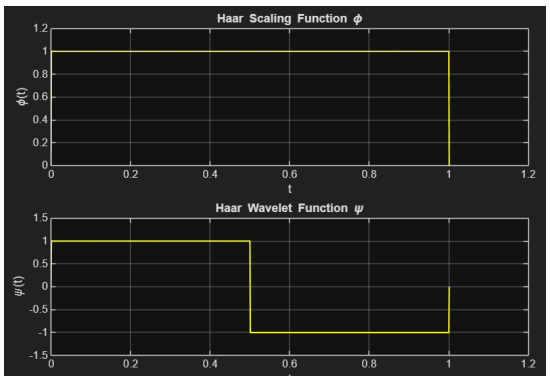

Figure 8: Scaling (ϕ) and wavelet (ψ) functions of Haar.

```
% Daubechies 9 wavelet
[phi_db9,psi_db9,x_db9] = wavefun('db9',10);
figure;
subplot(2,1,1);
plot(x_db9, phi_db9, 'y','LineWidth',1.2);

subplot(2,1,2);
plot(x_db9, psi_db9, 'y','LineWidth',1.2);
```

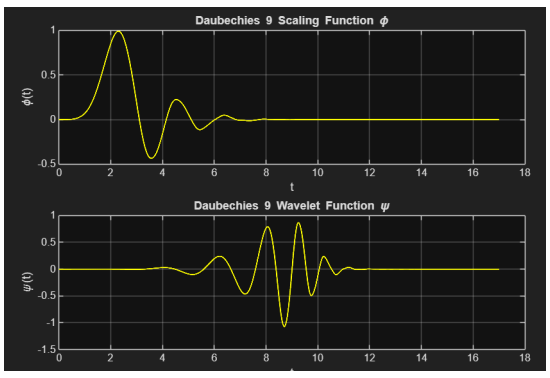

Figure 9: Scaling (ϕ) and wavelet (ψ) functions of db9.

10-Level Wavelet Decomposition

The noisy signals were decomposed using 10-level DWT with both Haar and db9 wavelets:

```
% y1 with haar
[C_y1_haar,L_y1_haar] = wavedec(y1,10,'haar');
```

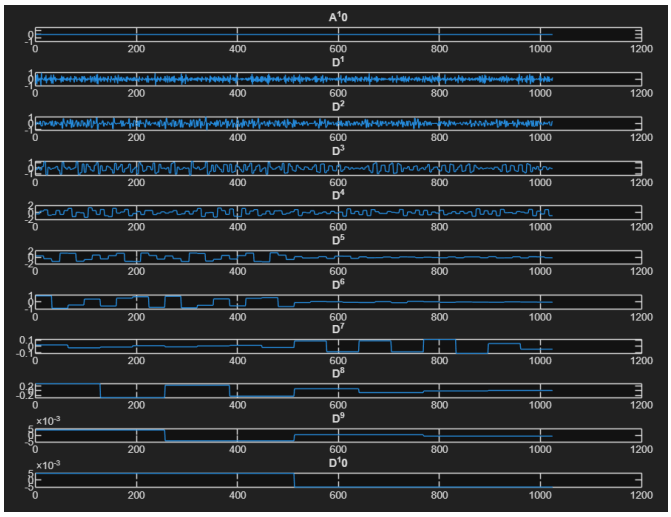

Figure 10: 10-level wavelet decomposition tree of signal $y_1[n]$.

```
% y1 with db9
[C_y1_db9,L_y1_db9] = wavedec(y1,10,'db9');
```

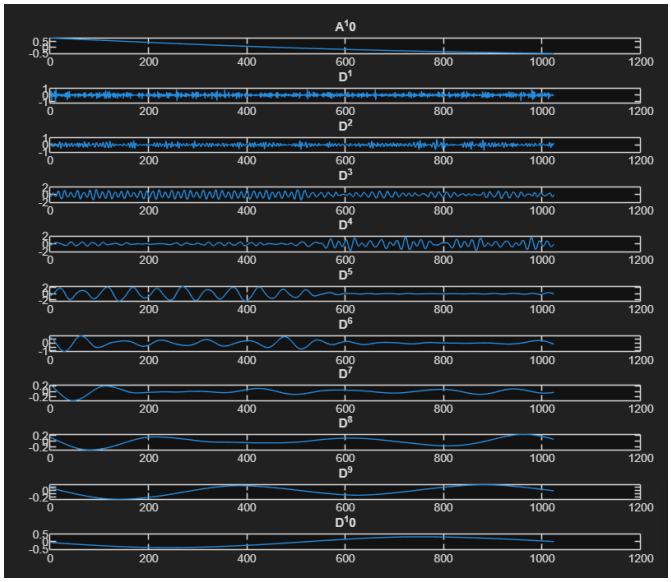

Figure 11: 10-level wavelet decomposition tree of signal $y_1[n]$.

```
% y2 with haar
[C_y2_haar,L_y2_haar] = wavedec(y2,10,'haar');
```

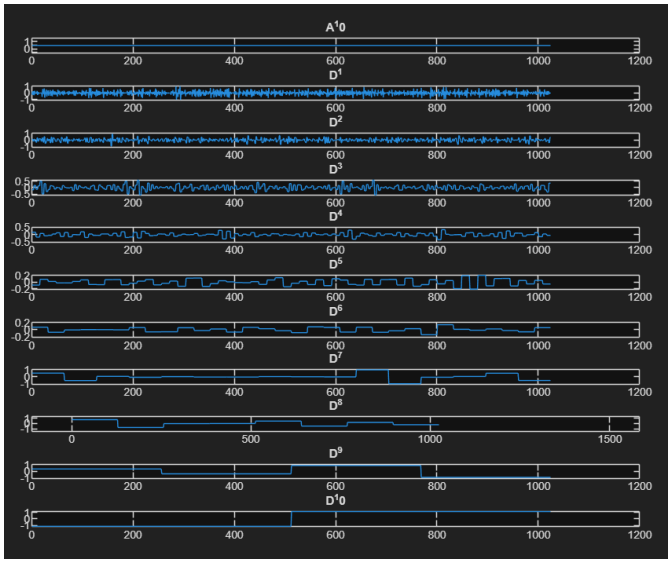

Figure 12: 10-level wavelet decomposition tree of signal $y_2[n]$.

```
% y2 with db9
[C_y2_db9,L_y2_db9] = wavedec(y2,10,'db9');
```

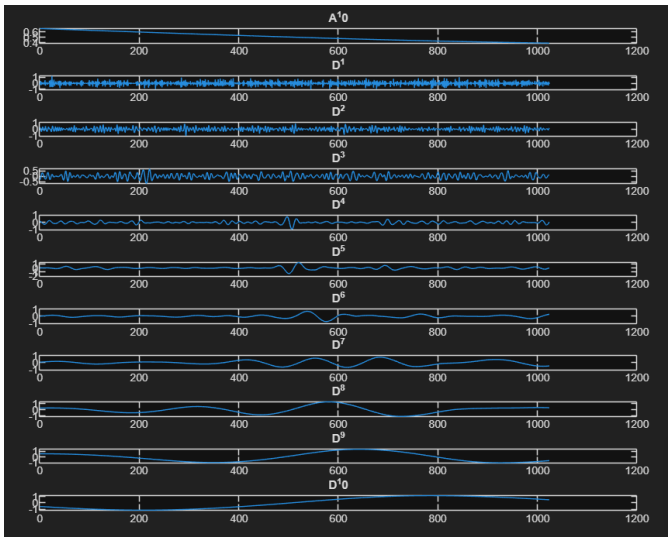

Figure 13: 10-level wavelet decomposition tree of signal $y_2[n]$.

Reconstruction via Inverse DWT

The approximation A_{10} and detail coefficients $D_{10}, D_9, \ldots, D_1$ were reconstructed using the inverse DWT:

```
for i = 1:10
    D_y1_haar{i} = wrcoef('d', C_y1_haar, L_y1_haar, 'haar', i);
end
A_y1_haar{1} = wrcoef('a', C_y1_haar, L_y1_haar, 'haar', 10);

figure('Name', 'y1_haar_det_approx_fn', 'NumberTitle', 'off');
```

```
subplot(11, 1, 1);
plot(A_y1_haar{1});
title(['A^' num2str(10)]);

for i = 1:10
    subplot(11, 1, i+1);
    plot(D_y1_haar{i});
    title(['D^' num2str(i)]);
end

y1_haar = 0;
for i = 1:10
    y1_haar = y1_haar + D_y1_haar{i};
end
y1_haar = y1_haar + A_y1_haar{1};


energy_diff_y1_haar = sum(y1.^2)- sum(y1_haar.^2);
fprintf('energy differnece between y1 and reconstructed y1 using haar:
   %.14f\n', abs(energy_diff_y1_haar));
```

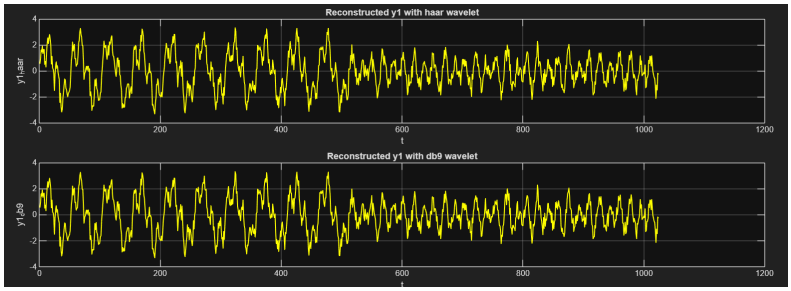

Figure 14: Reconstructed signal $y_{\text{reconstructed}}[n]$ compared to original noisy signal $y1[n]$.

Energy verification:

$$E_{\text{haar}} = \sum |y1[n]|^2 - \sum |y1_{\text{reconstructed}}[n]|^2 \approx 0.00000000000159$$

$$E_{\text{db9}} = \sum |y1[n]|^2 - \sum |y1_{\text{reconstructed}}[n]|^2 \approx 0.00000029993907$$

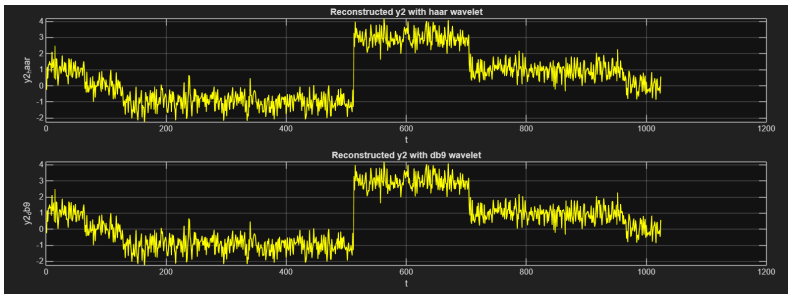

Figure 15: Reconstructed signal $y_{\text{reconstructed}}[n]$ compared to original noisy signal $y2[n]$.

Energy verification:

$$E_{\text{haar}} = \sum |y2[n]|^2 - \sum |y2_{\text{reconstructed}}[n]|^2 \approx 0.00000000000591$$

$$E_{\text{db9}} = \sum |y2[n]|^2 - \sum |y2_{\text{reconstructed}}[n]|^2 \approx 0.00000046159448$$

Signal Denoising with DWT

Wavelet Coefficient Analysis

The magnitude of the wavelet coefficients can be used to separate signal from noise. We first obtain the 10-level wavelet decomposition of the noisy signal $y_1[n]$ and $y_2[n]$ using the haar and the db9 wavelet:

```
%y1_haar
[C_y1_haar, L_y1_haar] = wavedec(y1, 10, 'haar');
coeffs_y1_haar = abs(C_y1_haar);
coeffs_sorted_y1_haar = sort(coeffs_y1_haar,'descend');

%y2_haar
[C_y2_haar, L_y2_haar] = wavedec(y2, 10, 'haar');
coeffs_y2_haar = abs(C_y2_haar);
coeffs_sorted_y2_haar = sort(coeffs_y2_haar,'descend');

%y1_db9
[C_y1_db9, L_y1_db9] = wavedec(y1, 10, 'db9');
coeffs_y1_db9 = abs(C_y1_db9);
coeffs_sorted_y1_db9 = sort(coeffs_y1_db9,'descend');

%y2_db9
[C_y2_db9, L_y2_db9] = wavedec(y2, 10, 'db9');
coeffs_y2_db9 = abs(C_y2_db9);
coeffs_sorted_y2_db9 = sort(coeffs_y2_db9,'descend');
```

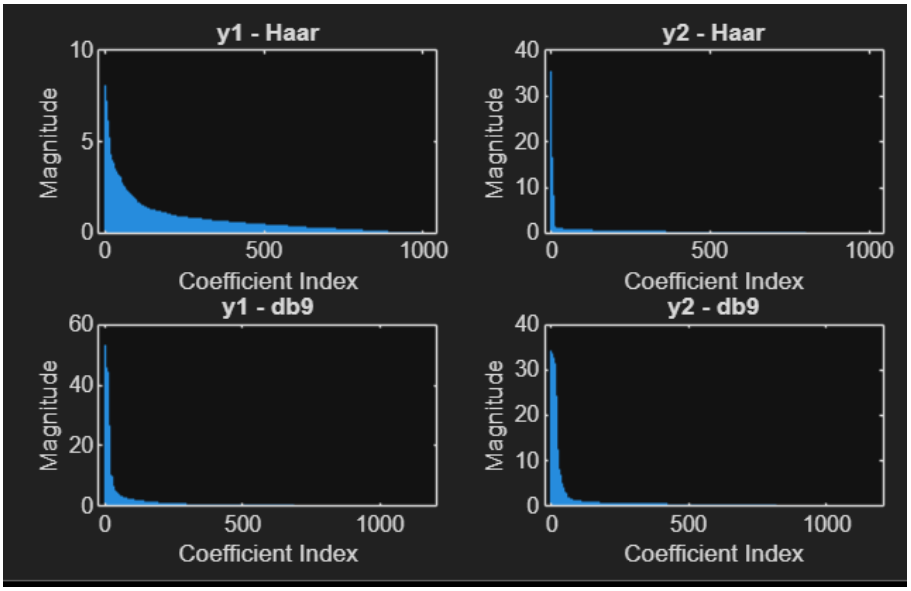

Figure 16: Magnitude of wavelet coefficients sorted in descending order. High-magnitude coefficients correspond to signal, low-magnitude to noise.

Thresholding and Signal Reconstruction

We select a threshold θ to suppress low-magnitude coefficients. A simple hard-thresholding method is used:

$$C_i' = \begin{cases} C_i, & |C_i| \geq \theta \\ 0, & |C_i| < \theta \end{cases}$$

```
%y1_haar
theta_y1_haar = 0.2 * max(coeffs_y1_haar); % threshold chosen by
    observation
fprintf('threshold for y1_haar: %.2f\n', theta_y1_haar);

C_thresh_y1_haar = C_y1_haar;
C_thresh_y1_haar(abs(C_thresh_y1_haar)<theta_y1_haar) = 0;

y_denoised_haar = waverec(C_thresh_y1_haar,L_y1_haar,'haar');
```

Original and denoised time domain waveforms for x1 signal.

- Threshold for haar wavelet: 1.5
- Threshold for db9 wavelet: 0.9

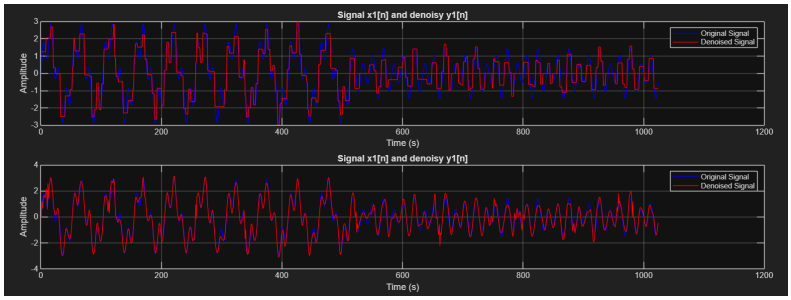

Figure 17: Original noisy signal $y_1[n]$ and denoised signal using haar and db9 wavelet.

Original and denoised time domain waveforms for x2 signal.

- Threshold for haar wavelet: 3.0
- Threshold for db9 wavelet: 1.2

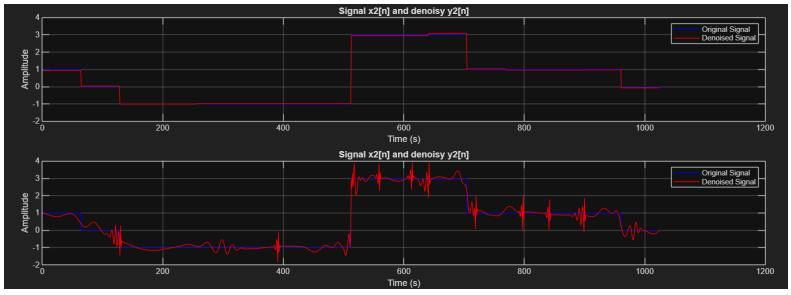

Figure 18: Original noisy signal $y_2[n]$ and denoised signal using haar and db9 wavelet.

RMSE Calculation

The root mean square error (RMSE) quantifies the reconstruction accuracy:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}\left(y_1[n]_{\text{original}} - y_1[n]_{\text{denoised}}\right)^2}$$

```
RMSE_db9 = sqrt(mean((x1 - y_denoised).^2));
fprintf('RMSE (db9): %.5f\n', RMSE);
```

- The RMSE value for $y_1[n]$ signal when denoised using "DB9" wavelet is 0.259596892.
- The RMSE value for $y_2[n]$ signal when denoised using "DB9" wavelet is 0.247202901.

Repeating that method for haar wavelet

- The RMSE value for $y_1[n]$ signal when denoised using "haar" wavelet is 0.520073135.
- The RMSE value for $y_2[n]$ signal when denoised using "haar" wavelet is 0.041337460.

Comparison and Discussion

- **RMSE comparison:** db9 usually achieves lower RMSE than Haar due to smoother filters capturing the signal structure better.
- **Morphology:** db9 preserves smooth sinusoidal features, whereas Haar introduces blocky artifacts due to its piecewise constant nature.
- **Suitability:**
  - db9 $\rightarrow$ Suitable for smooth signals like $x_1[n]$; better frequency representation.
  - Haar $\rightarrow$ Better for signals with sharp edges or discontinuities, but less accurate for smooth oscillations.
- Thresholding effectively removes low-magnitude noise coefficients while retaining signal features.

Signal Compression with DWT

Wavelet Decomposition of ECG Signal

We use the aVR lead of an ECG sampled at $f_s = 257$ Hz. The 10-level discrete wavelet decomposition is performed using both `db9` and `Haar` wavelets:

```
% Load ECG signal
load('aVR_ECG.mat'); % variable: ecg_signal

% Decomposition
[C_db9,L_db9] = wavedec(aVR,10,'db9');
[C_haar,L_haar] = wavedec(aVR,10,'haar');
```

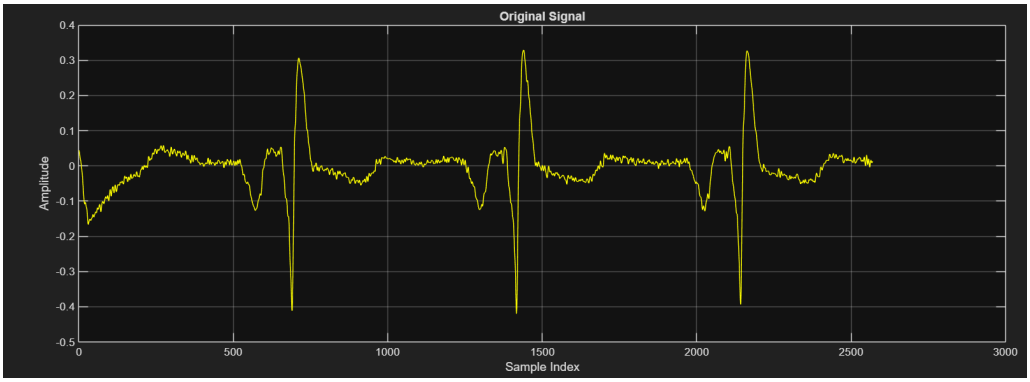

Figure 19: Original aVR ECG signal.

```
% Sort coefficients by magnitude
coeffs_db9 = sort(abs(C_db9),'descend');
coeffs_haar = sort(abs(C_haar),'descend');
```

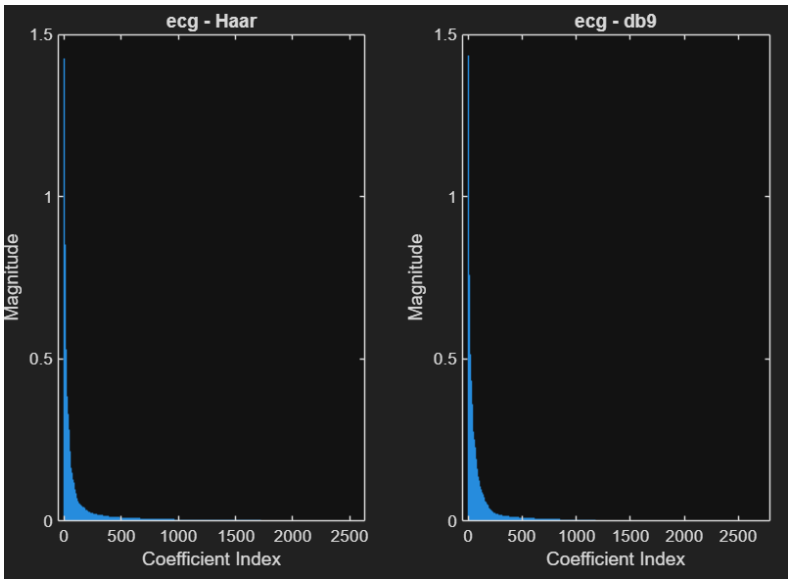

Figure 20: Decomposed coefficients for db9 and haar wavelets.

Energy-based Coefficient Selection

The wavelet coefficients are sorted by magnitude in descending order. We select the minimum number of coefficients needed to retain 99% of the signal energy.

$$E_{\text{total}} = \sum_{i} |C_i|^2, \quad E_{\text{cum}} = \sum_{i=1}^{k} |C_i|^2$$

Choose smallest k such that $E_{\text{cum}}/E_{\text{total}} \geq 0.99$.

```
% Sort coefficients by magnitude
coeffs_db9 = sort(abs(C_db9),'descend');
E_total = sum(coeffs_db9.^2);
E_cum = cumsum(coeffs_db9.^2);

```

```
% Find number of coefficients for 99% energy
k_db9 = find(E_cum >= 0.99*E_total, 1);
fprintf('db9: %d coefficients for 99%% energy\n', k_db9);

coeffs_haar = sort(abs(C_haar),'descend');
E_total_h = sum(coeffs_haar.^2);
E_cum_h = cumsum(coeffs_haar.^2);
k_haar = find(E_cum_h >= 0.99*E_total_h, 1);
fprintf('Haar: %d coefficients for 99%% energy\n', k_haar);
```

- **Haar:** is using 177 coefficients for 0.99 energy
- **db9:** is using 170 coefficients for 0.99 energy

Signal Compression and Reconstruction

Coefficients below the selected threshold are set to zero, and the signal is reconstructed:

```
% Compression using db9
C_db9_comp = C_db9;
C_db9_comp(abs(C_db9) < coeffs_db9(k_db9)) = 0;
y_compressed_db9 = waverec(C_db9_comp,L_db9,'db9');

% Compression using Haar
C_haar_comp = C_haar;
C_haar_comp(abs(C_haar) < coeffs_haar(k_haar)) = 0;
y_compressed_haar = waverec(C_haar_comp,L_haar,'haar');

% Compression ratio
CR_db9 = length(C_db9)/k_db9;
CR_haar = length(C_haar)/k_haar;
fprintf('Compression ratio db9: %.2f\n', CR_db9);
fprintf('Compression ratio Haar: %.2f\n', CR_haar);
```

- Compression ratio for Haar wavelet: **14.56**
- Compression ratio for db9 wavelet: **16.08**

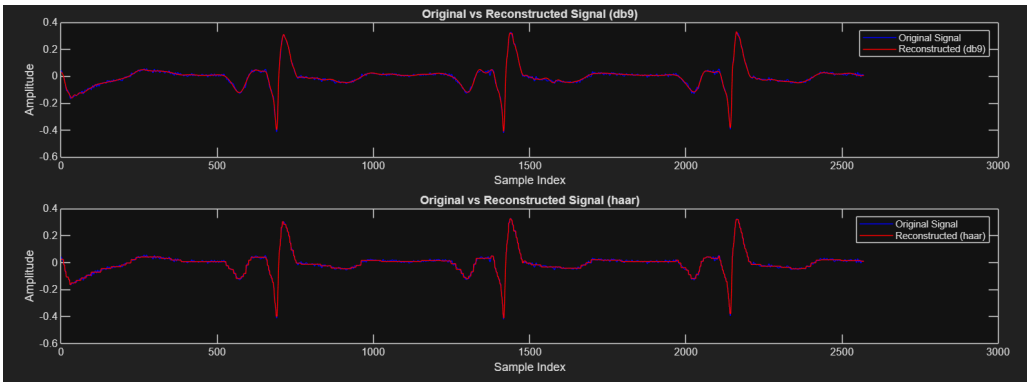

Figure 21: Reconstructed ECG signals after compression. Top: db9, Bottom: Haar.

- **Energy compaction:** db9 wavelet achieves 99% energy with fewer coefficients than Haar, resulting in a higher compression ratio.
- **Morphology preservation:** db9 retains the smooth QRS complex and P/Q/T waves accurately. Haar introduces blocky artifacts due to piecewise constant basis.
- **Compression ratio:** Higher for db9 because of more efficient coefficient representation; lower for Haar.
- **Conclusion:** db9 is more suitable for ECG compression when both energy preservation and waveform fidelity are important.