HW5

Nima noorizadeh 40023086

Mahdi Mahjouri Namin 40023125

```
clear; close all; clc;
```

Section 1 – Signal Generation

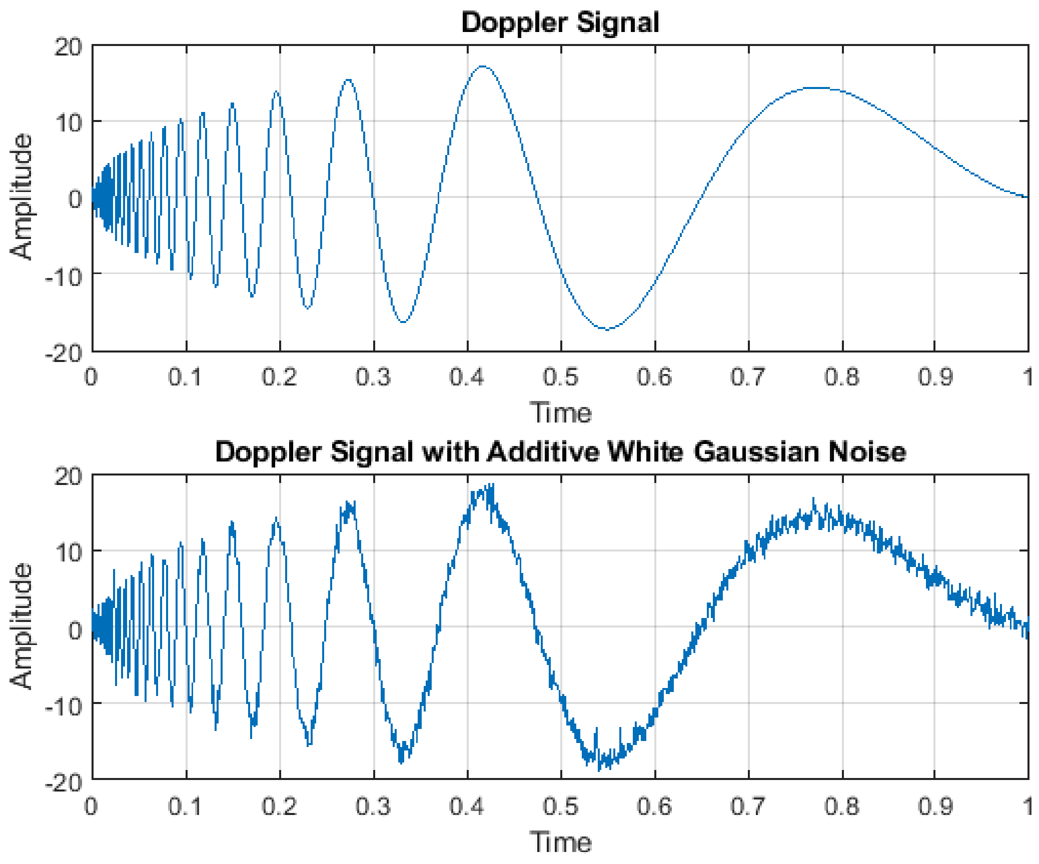
A Doppler signal is generated using the `wnoise` function with 2^{10} samples.

Additive white Gaussian noise is introduced to simulate a realistic measurement environment.

The clean and noisy signals are plotted to visualize the distortion caused by the noise.

```
Npow  = 10;
loc   = linspace(0,1,2^Npow);
[clean, noisy] = wnoise('doppler',Npow,10);

figure(1);
subplot(2,1,1); plot(loc,clean,'LineWidth',1); axis tight; ylim([-20 20]); grid on;
title('Doppler Signal'); xlabel('Time'); ylabel('Amplitude');
subplot(2,1,2); plot(loc,noisy,'LineWidth',1); axis tight; ylim([-20 20]); grid on;
title('Doppler Signal with Additive White Gaussian Noise'); xlabel('Time'); ylabel('Amplitude'
```



```
Ts = loc(2) - loc(1);
fs = 1 / Ts;
```

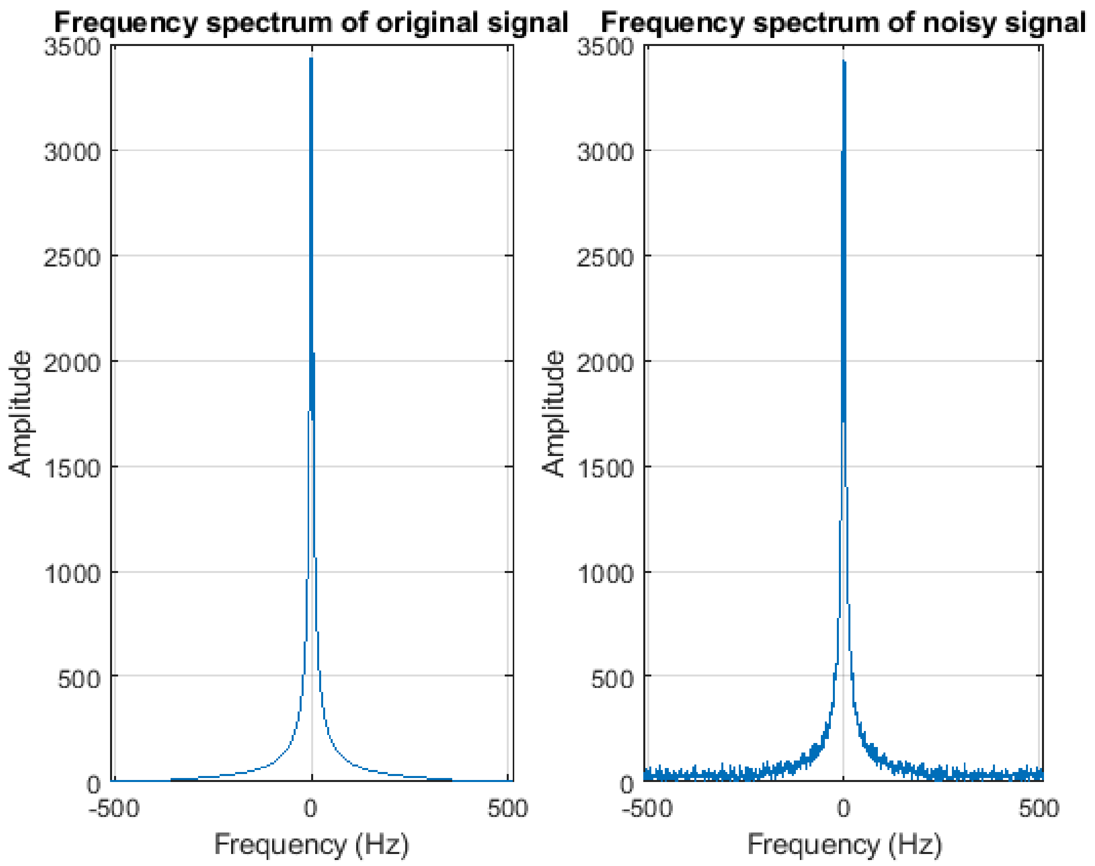
Section 2 – Frequency Analysis

The Fourier transforms of both clean and noisy signals are computed using `fft`.

Frequency spectra reveal that the noise increases energy across all frequencies.

This confirms that the added noise is broadband and not frequency-selective.

```
N = numel(clean); f = (-N/2:N/2-1)/N*N;
Xc = fftshift(fft(clean)); Xn = fftshift(fft(noisy));
figure(2);
subplot(1,2,1); plot(f,abs(Xc)); grid on; title('Frequency spectrum of original signal'); xlabe
subplot(1,2,2); plot(f,abs(Xn)); grid on; title('Frequency spectrum of noisy signal'); xlabel(
```



Section 3 – Single-Level DWT

A one-level discrete wavelet transform (DWT) using the `db2` wavelet separates the noisy signal into approximation and detail components.

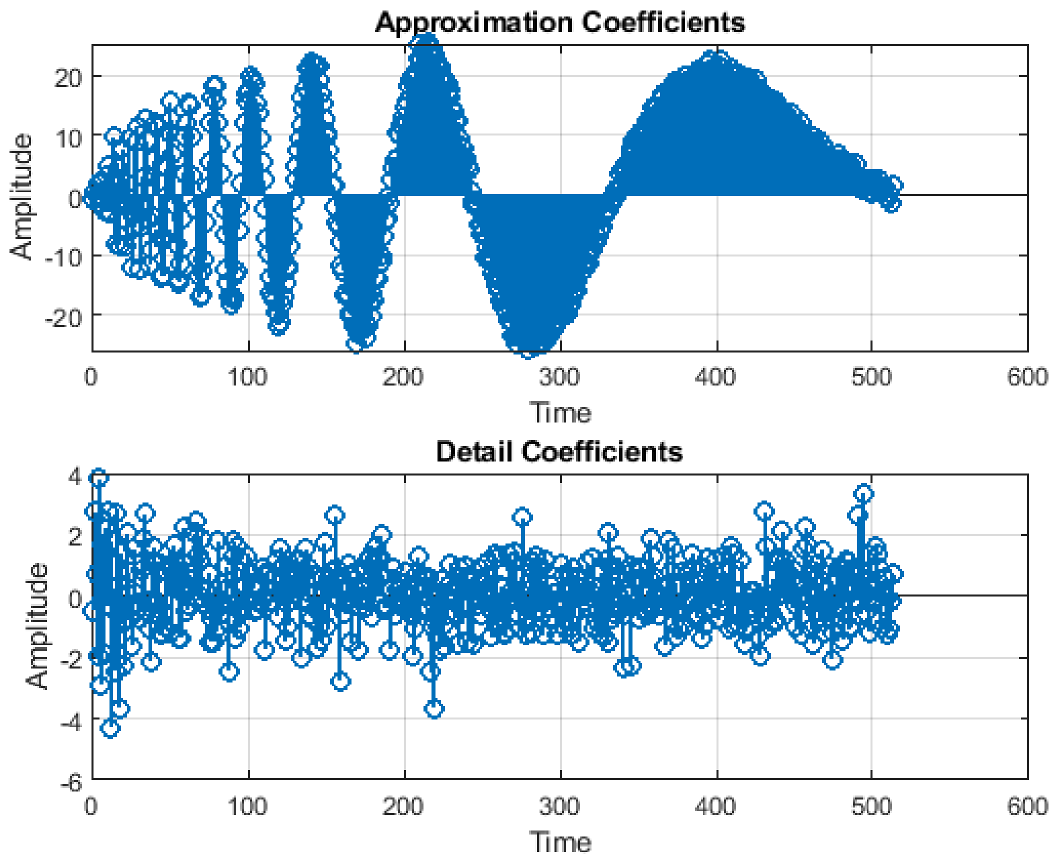
The approximation captures low-frequency trends, while the detail shows high-frequency fluctuations and noise.

Stem plots clearly illustrate the concentration of noise in the detail coefficients.

```
waveletType = 'db2';
[ca, cd] = dwt(noisy, waveletType);

figure(3);
subplot(2,1,1)
stem(ca, 'o','LineWidth', 1.5);
title('Approximation Coefficients'); xlabel('Time'); ylabel('Amplitude'); grid on

subplot(2,1,2)
stem(cd, 'o','LineWidth', 1.5);
title('Detail Coefficients'); xlabel('Time'); ylabel('Amplitude'); grid on
```



Section 4 – Multi-Level Decomposition

Using `wavedec`, the noisy signal is decomposed into 10 levels using the `db8` wavelet.

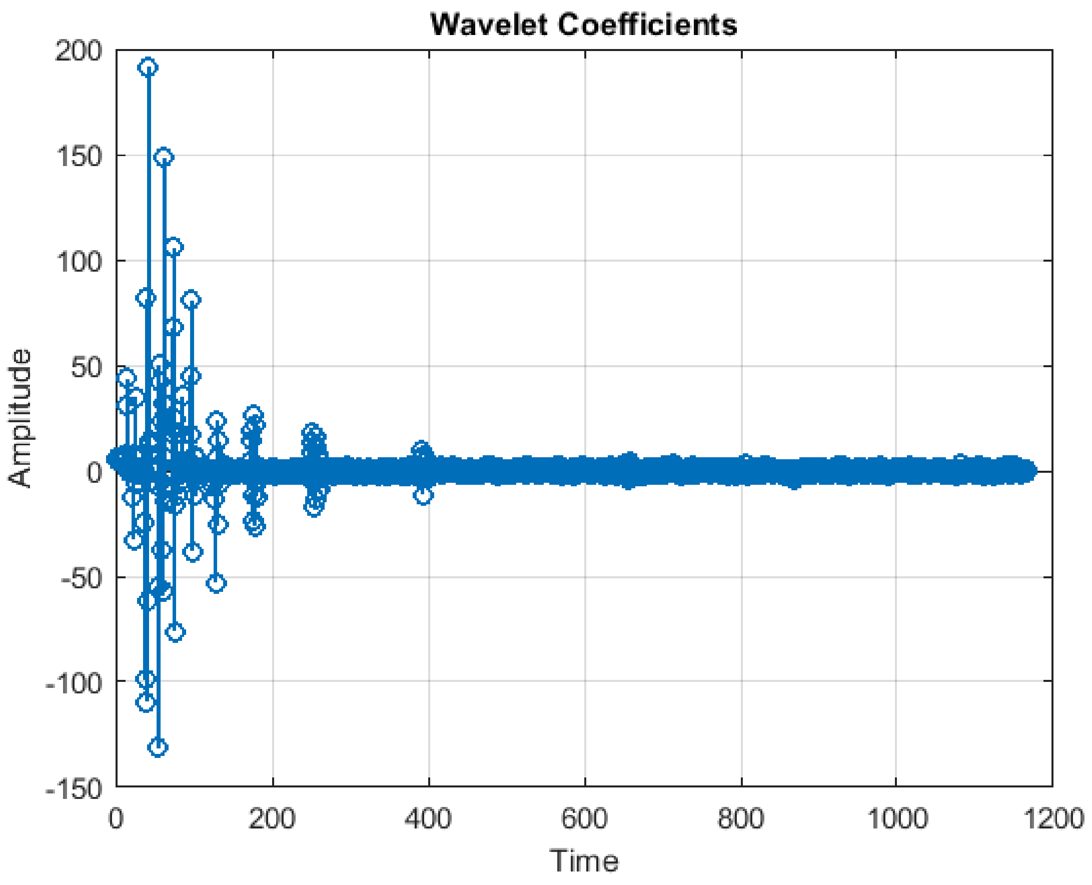
The full wavelet coefficient vector is plotted, showing how energy is distributed across scales.

This decomposition enables fine-grained signal analysis across multiple frequency bands.

```
waveletType = 'db8';
level = 10;

[C, L] = wavedec(noisy, level, waveletType);

figure(4); clf
stem(C, 'o', 'LineWidth', 1.5);
title('Wavelet Coefficients');
xlabel('Time'); ylabel('Amplitude');
grid on
```



Section 5 – Detail Coefficients by Level

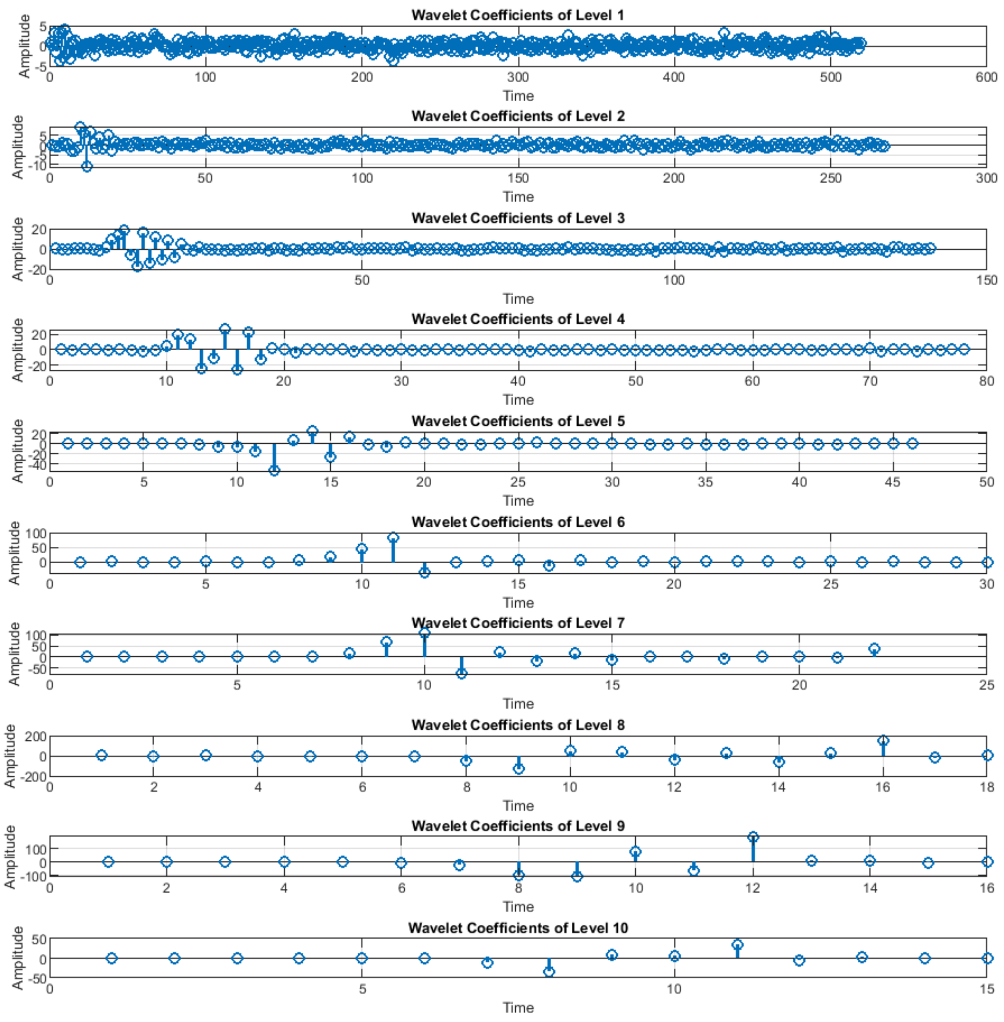
The `detcoef` function is used to extract detail coefficients from each decomposition level.

Stem plots of these coefficients highlight how noise and signal energy are distributed.

Lower levels capture high-frequency noise, while higher levels reflect smoother signal features.

```
waveletType = 'db8';
level = 10;
[C, L] = wavedec(noisy, level, waveletType);

detailed_coeff = detcoef(C, L, 1:level);
 figure('Name', 'Wavelet Coefficients of Different Levels');
 set(gcf, 'Position', [100, 100, 1000, 1000]);
 for i = 1:level
    subplot(level, 1, i);
    stem(detailed_coeff{i}, 'LineWidth', 1.5);
    title(['Wavelet Coefficients of Level ', num2str(i)]);
    xlabel('Time');
    ylabel('Amplitude');
    grid on;
 end
```



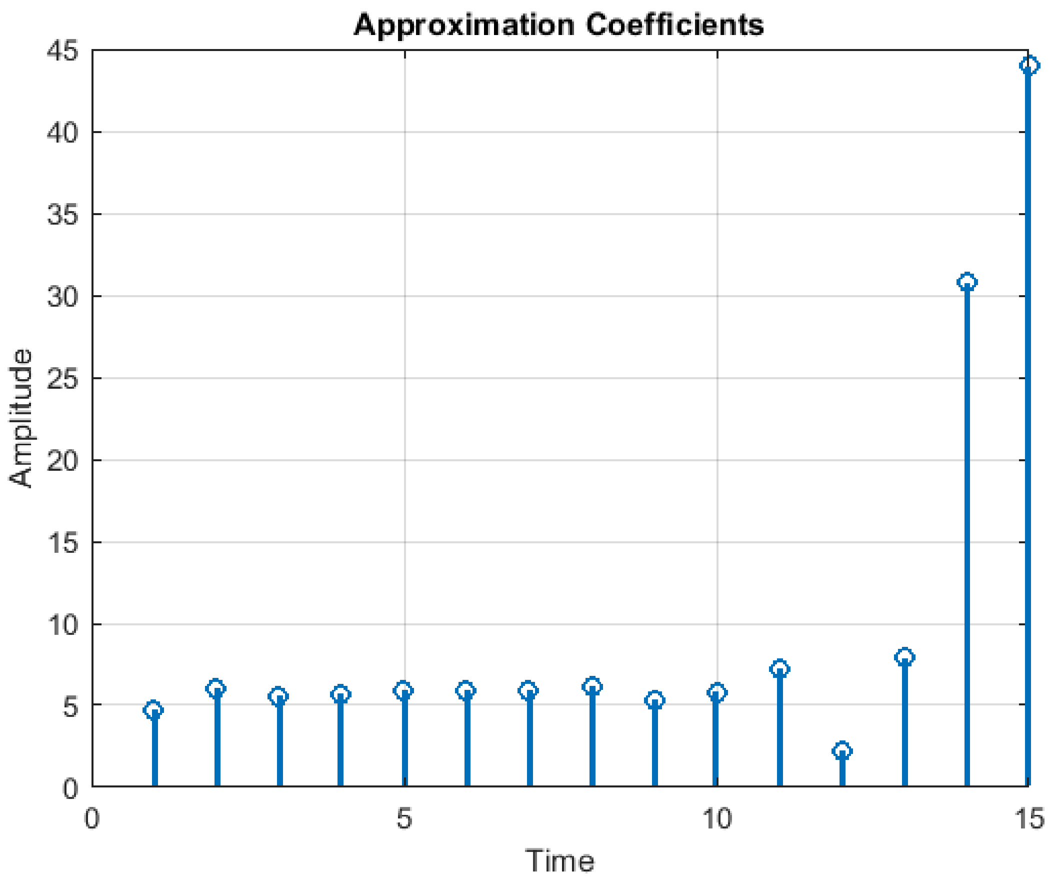
Section 6 – Approximation Coefficients at Deepest Level

`appcoef` is used to extract the approximation coefficients from level 10.

These represent the low-frequency, smoothed version of the signal.

Plotting them provides insight into the signal’s coarse structure.

```
approx_coeff = appcoef(C,L, 'db8', 10);
figure('Name', 'Approximation Coefficients');
stem(approx_coeff, 'o','LineWidth', 1.5);
title('Approximation Coefficients');
xlabel('Time');
ylabel('Amplitude');
grid on;
```



Section 7 – Coefficient Reconstruction and Error

The full wavelet coefficient vector is reconstructed from approximation and detail parts.

This is compared with the original coefficients, and the error is computed.

A zero error confirms perfect lossless reconstruction via `wavedec` and `appcoef/detcoef`.

```
approx_coeff_deepest_layer = appcoef( C, L,'db8',10);

wavelet_coefficients_regenerated = ...
   [approx_coeff_deepest_layer, detailed_coeff{10:-1:1}];
fprintf("\nError: %d\n", ...
   sum(abs(wavelet_coefficients_regenerated - C)));
```

```
Error: 0
```

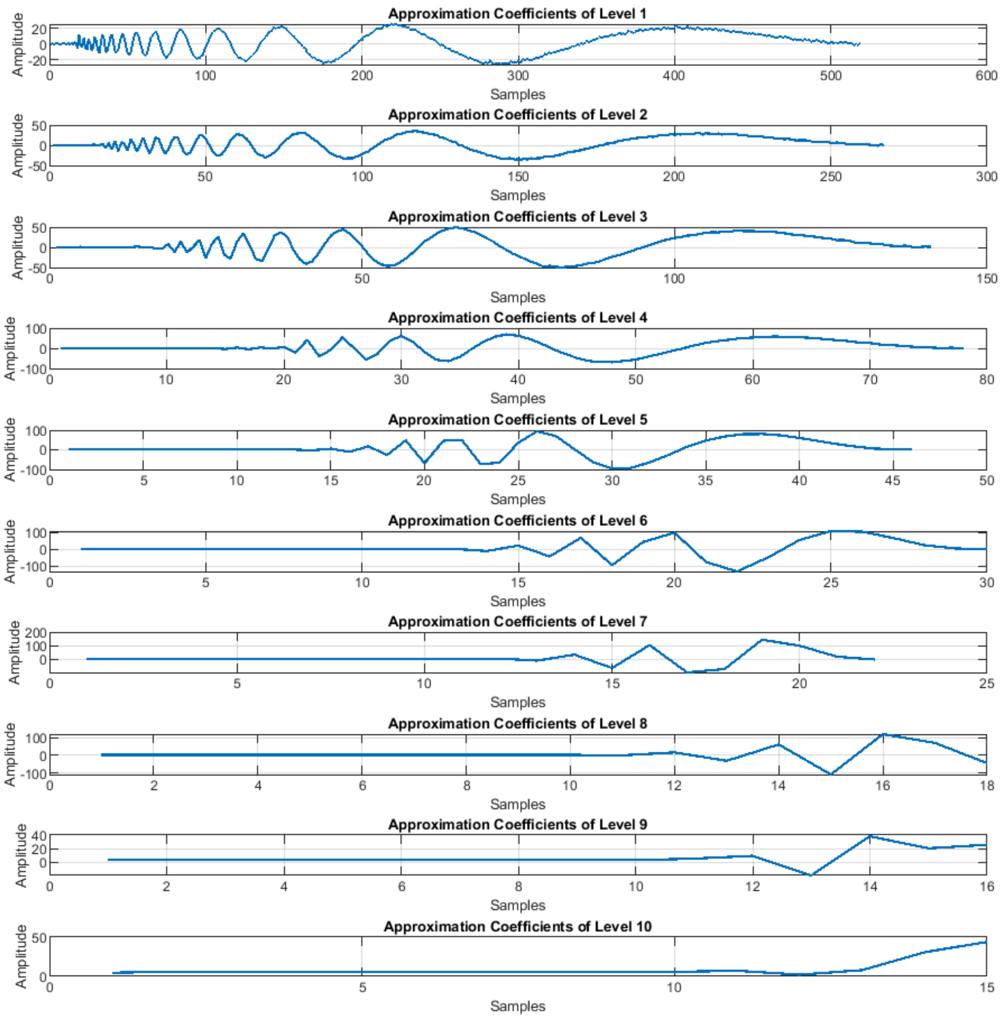
Section 8 – Approximation Across All Levels

Approximation coefficients from all 10 levels are extracted using `appcoef`.

Each plot shows how the signal appears at increasingly coarser resolutions.

This multiscale analysis reveals the progressive loss of high-frequency detail.

```
figure('Name', 'Approximation Coefficients of Different Levels');
 for level = 1:10
    subplot(10, 1, level);
    set(gcf, 'Position', [100, 100, 1000, 1000]);
    approx = appcoef(C, L, 'db8', level);
    plot(approx, 'LineWidth', 1.5);
    title(['Approximation Coefficients of Level ', num2str(level)]);
    xlabel('Samples');
    ylabel('Amplitude');
    grid on;
 end
```



Section 9

Zeroing the first three wavelet detail levels deletes genuine high-frequency Doppler content together with the noise, over-smoothing the waveform.

Low-frequency noise remains in the retained bands, so overall SNR improves only marginally.

The abrupt band removal also distorts time-localised features, introducing phase shifts and amplitude attenuation.

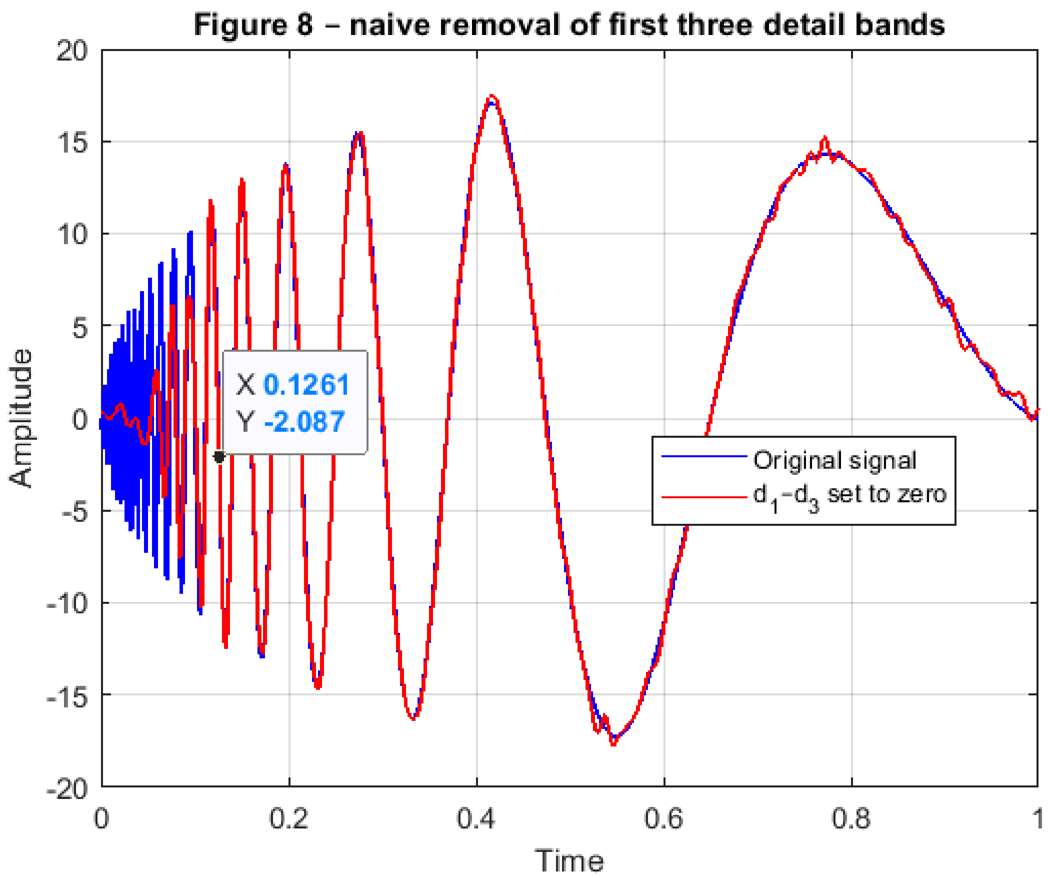
Adaptive thresholding (e.g., `wdenoise`) is preferred: it attenuates small noise-dominated coefficients while preserving large signal-bearing ones.

```
waveletType = 'db8';          % X ≥ 8
level       = 10;
[C,L]       = wavedec(noisy, level, waveletType);

C_mod = wthcoef('d', C, L, [1 2 3]);

y_wrong = waverec(C_mod, L, waveletType);

t = linspace(0,1,numel(noisy));
figure(8); clf
plot(t, clean,   'b', 'LineWidth',1.2); hold on
plot(t, y_wrong, 'r', 'LineWidth',1.2);
legend('Original signal', 'd_1-d_3 set to zero', 'Location','best');
title('Figure 8 – naive removal of first three detail bands');
xlabel('Time'); ylabel('Amplitude'); grid on
```



Section 10 – Proper Denoising using `wdenoise`

`wdenoise` performs adaptive thresholding of detail coefficients across all levels.

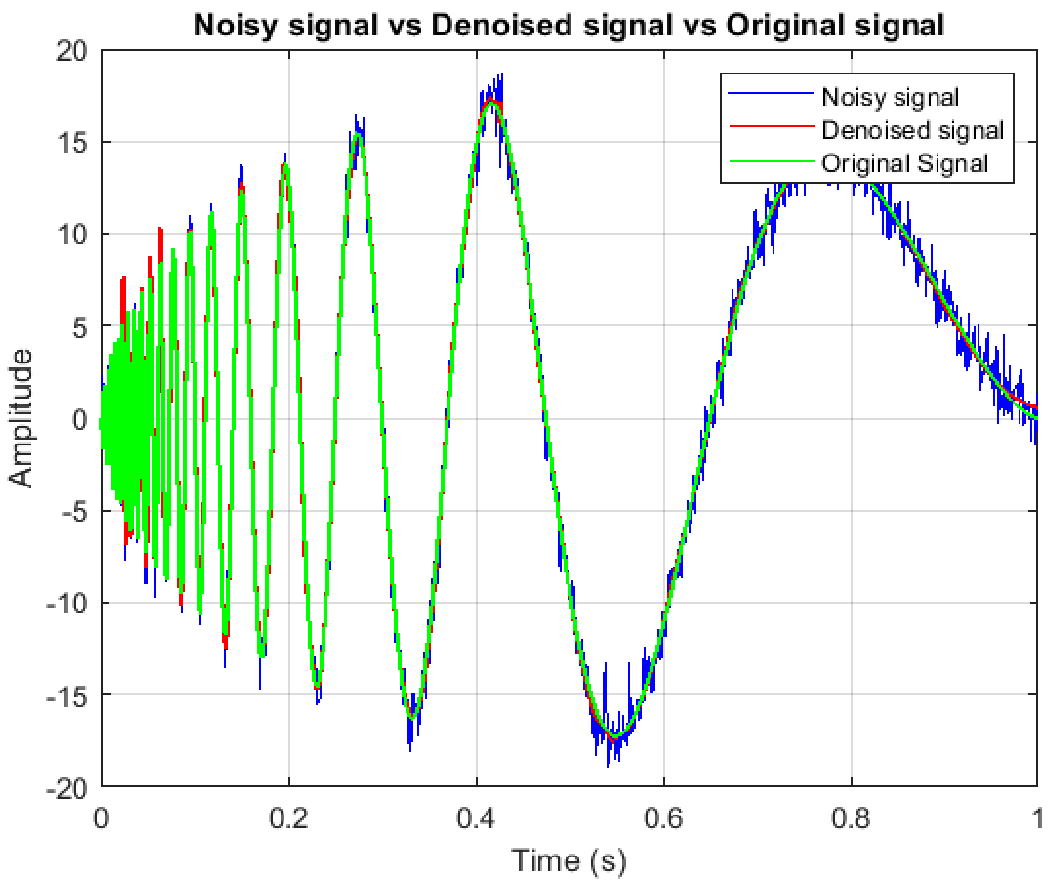
The denoised signal preserves important features while effectively suppressing noise.

Compared to Section 9, this method achieves superior SNR and signal quality

```
denoised = wdenoise(noisy, 10, 'Wavelet', 'db8');
loc = linspace(0, 1, length(noisy));

figure(9); clf
plot(loc, noisy, 'b', 'LineWidth', 0.5); hold on;
plot(loc, denoised, 'r', 'LineWidth', 1.2);
plot(loc, clean, 'g', 'LineWidth', 1.2);

title('Noisy signal vs Denoised signal vs Original signal');
xlabel('Time (s)'); ylabel('Amplitude'); grid on;
legend('Noisy signal', 'Denoised signal', 'Original Signal');
```


Noisy signal vs Denoised signal vs Original signal
Noisy signal
Denoised signal
Original Signal
Amplitude
Time (s)
20
15
10
5
0
-5
-10
-15
-20
0
0.2
0.4
0.6
0.8
1