Homework2_3

Programmers

Mohammad Mahdi Elyasi - 9823007

Moein Nasiri - 9823093

Clear Workspace

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

Homework_1

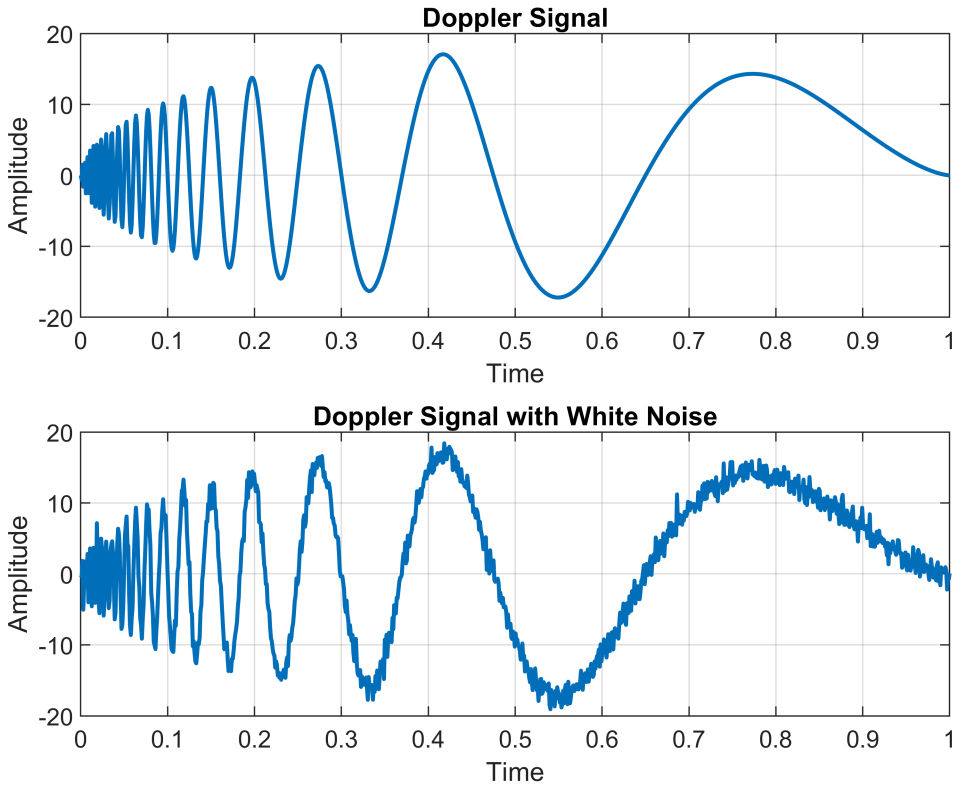
In this Homework we play with wavelet functions and how it effects on signal

Here we declare some basic variables to solve the first question

```
N = 10;
loc = linspace(0, 1, 2 ^ N);
loc_f = linspace(-500, 500, 2 ^ N);
sqrtsnr = 10;
```

Here we want to add white noise to our doppler signal and plot them

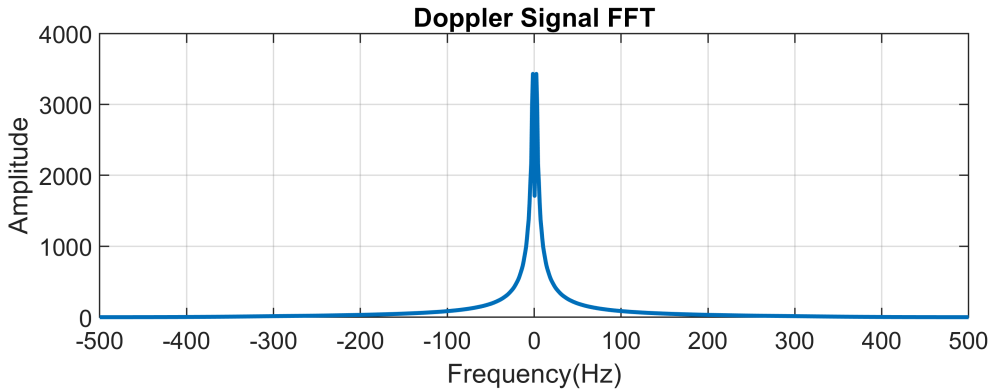
```
[x, noisy_x] = wnoise('doppler', N, sqrtsnr);
figure('Name', 'Doppler Signal vs Doppler Signal with White Noise');
subplot(211);
plot(loc, x, 'LineWidth', 1.5);
title('Doppler Signal');
xlabel('Time');
ylabel('Amplitude');
grid on;
subplot(2, 1, 2);
plot(loc, noisy_x, 'LineWidth', 1.5);
title('Doppler Signal with White Noise');
xlabel('Time');
ylabel('Amplitude');
grid on;
```

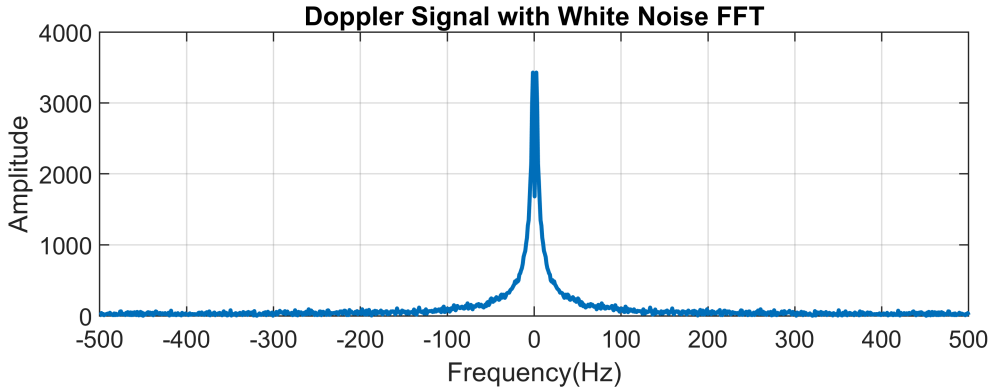


Part 2:

Now we plot FFT of noisy signal and original signal

```
figure('Name', 'Doppler Signal vs Doppler Signal with White Noise FFT');
subplot(211);
plot(loc_f, abs(fftshift(fft(x))), 'LineWidth', 1.5);
title('Doppler Signal FFT');
xlabel('Frequency(Hz)');
ylabel('Amplitude');
grid on;
subplot(2, 1, 2);
plot(loc_f, abs(fftshift(fft(noisy_x))), 'LineWidth', 1.5);
title('Doppler Signal with White Noise FFT');
xlabel('Frequency(Hz)');
ylabel('Amplitude');
grid on;
```

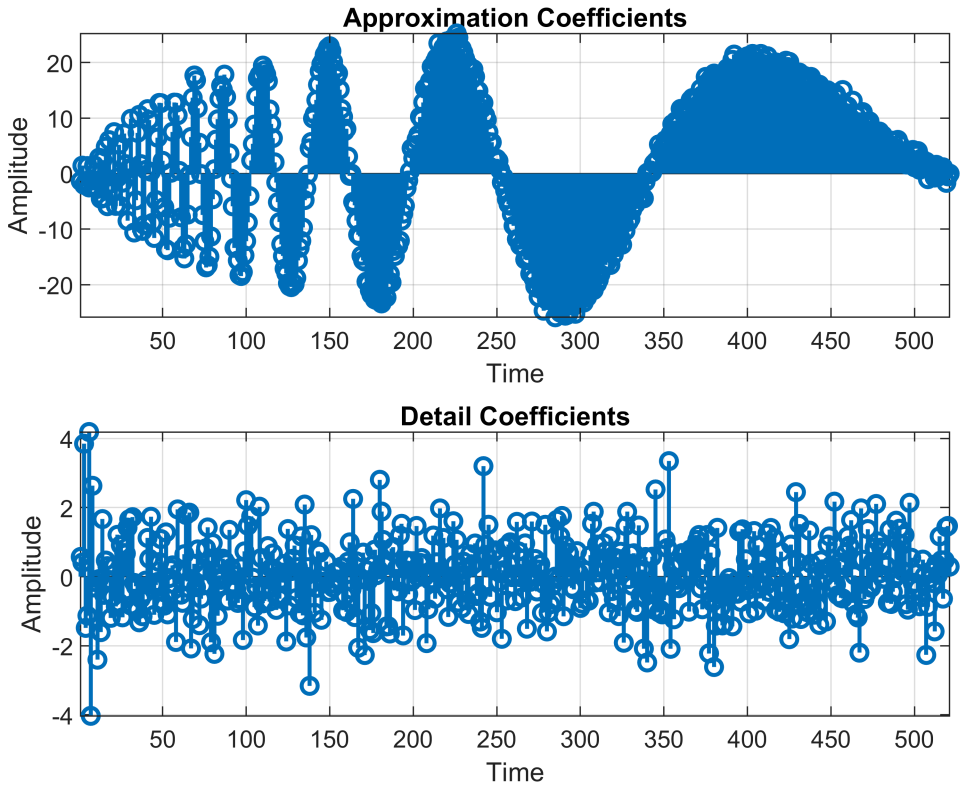




Part 3:

Here we want to extract Coefficients of our noisy signal and then plot them

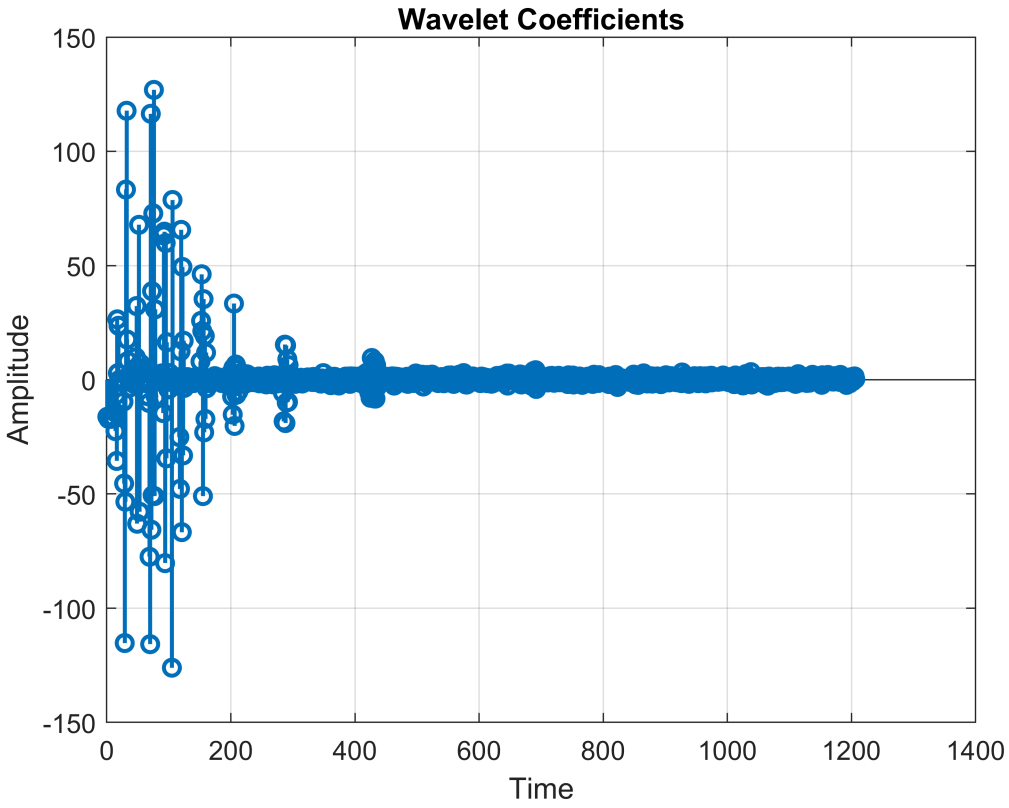
```
[cA, cD] = dwt(noisy_x, 'db10');
figure('Name', 'Approximation and Detail Coefficients');
subplot(211);
stem(cA, 'LineWidth', 1.5);
title('Approximation Coefficients');
xlabel('Time');
ylabel('Amplitude');
grid on;
axis tight;
subplot(212);
stem(cD, 'LineWidth', 1.5);
title('Detail Coefficients');
xlabel('Time');
ylabel('Amplitude');
grid on;
axis tight;
```



Part 4:

Here we want to extract wavelet coefficients and plot

```
numberOfLevels = 10;
[waveletCoefficients, waveletLevels] = wavedec(noisy_x, 10, 'db10');
figure('Name', 'Wavelet Coefficients');
stem(waveletCoefficients, 'LineWidth', 1.5);
title('Wavelet Coefficients');
xlabel('Time');
ylabel('Amplitude');
grid on;
```

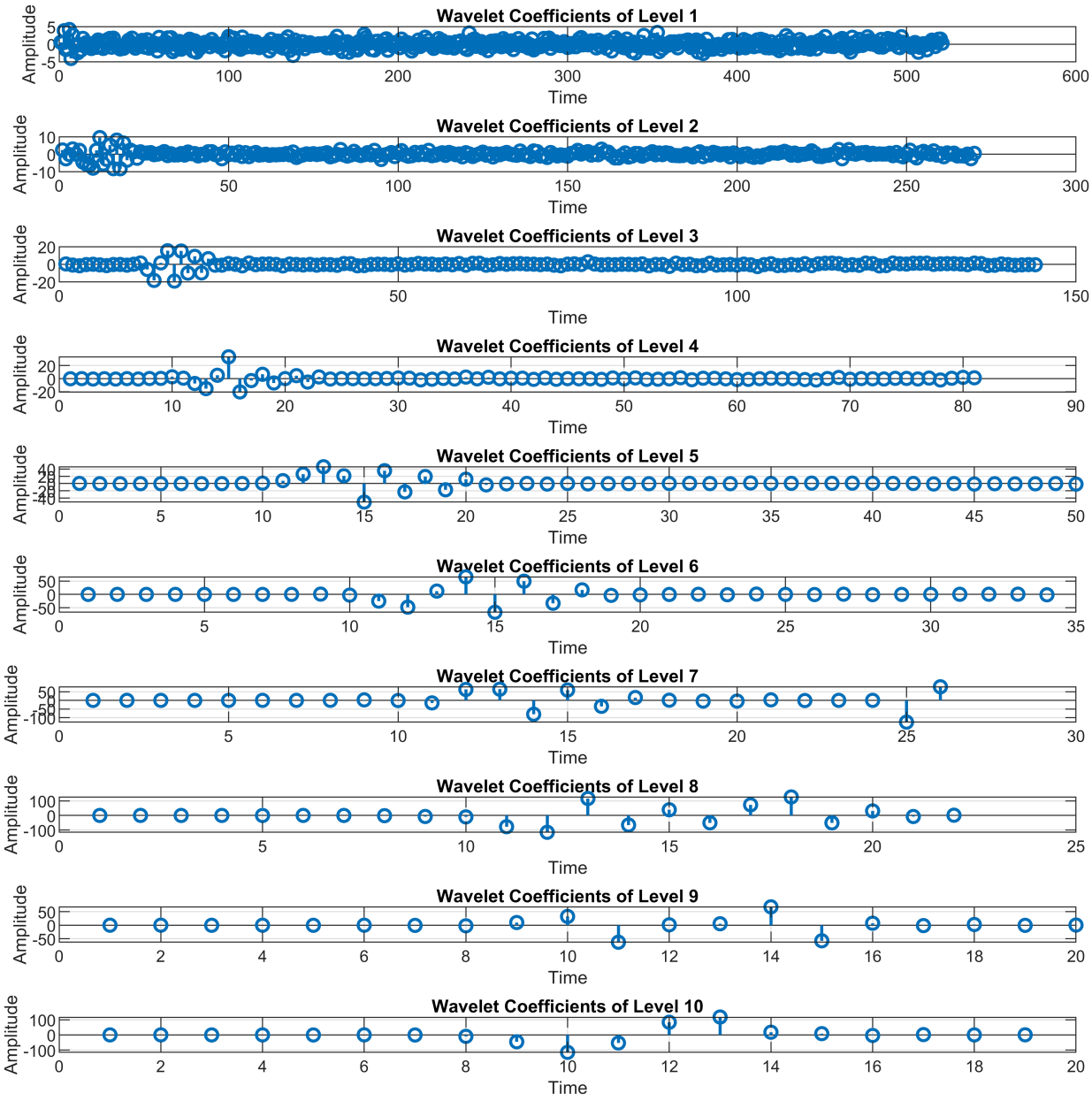


Part 5:

Here we want to extract detailed coefficient of wavelet coefficient and plot them.

```
detailed_coeff = detcoef(waveletCoefficients, waveletLevels, 1:numberOfLevels);
figure('Name', 'Wavelet Coefficients of Different Levels');
set(gcf, 'Position', [100, 100, 1000, 1000]);

for i = 1:numberOfLevels
    subplot(numberOfLevels, 1, i);
    stem(detailed_coeff{i}, 'LineWidth', 1.5);
    title(['Wavelet Coefficients of Level ', num2str(i)]);
    xlabel('Time');
    ylabel('Amplitude');
    grid on;
end
```

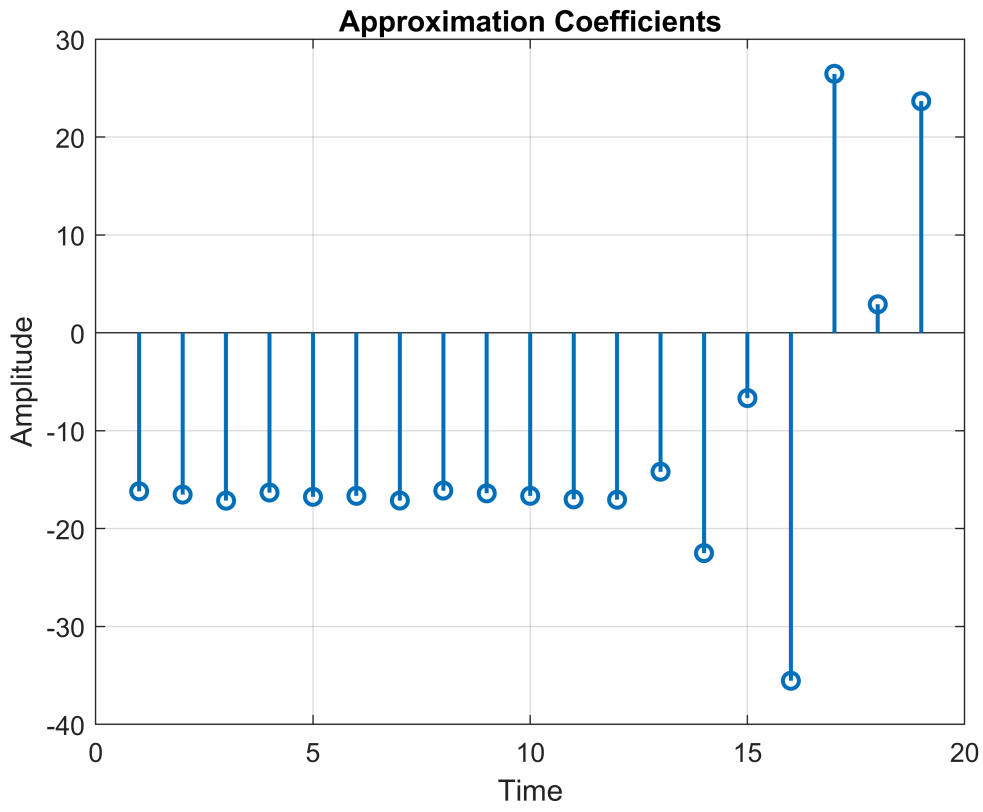


Part 6:

Here we want to extract the deepest approximated layer and we want to plot them

```
approx_coeff = appcoef(waveletCoefficients, waveletLevels, 'db10', 10);
figure('Name', 'Approximation Coefficients');
stem(approx_coeff, 'LineWidth', 1.5);
title('Approximation Coefficients');
xlabel('Time');
```

```
ylabel('Amplitude');
grid on;
```



Part 7:

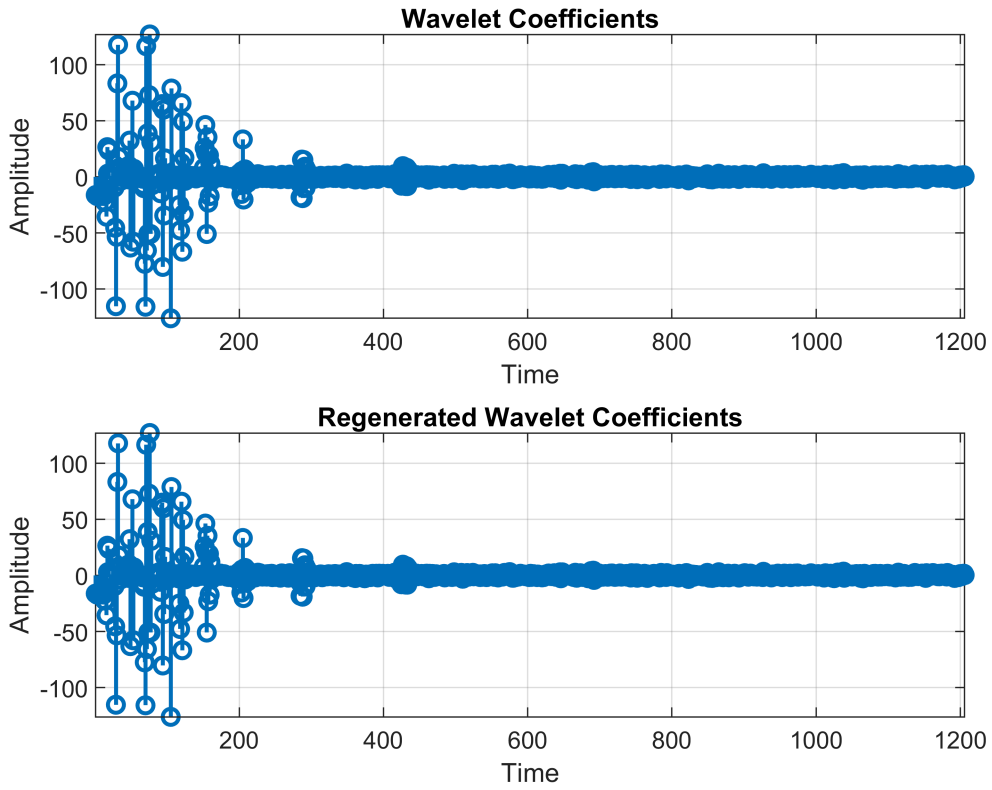
Here we want to extract approximated coefficient from wavelet coefficient and then we want to reconstruct signal from approximated and detailed coefficient and then see how much is the error

```
approx_coeff_deepest_layer = appcoef( ...
    waveletCoefficients, ...
    waveletLevels, ...
    'db10', ...
    numberOfLevels ...
);
wavelet_coefficients_regenerated = ...
    [approx_coeff_deepest_layer, detailed_coeff{numberOfLevels:-1:1}];
fprintf("\nError: %d\n", ...
    sum(abs(wavelet_coefficients_regenerated - waveletCoefficients)));
```

```
Error: 0
```

```
figure('Name', 'Wavelet Coefficients vs Regenerated Wavelet Coefficients');
subplot(211);
stem(waveletCoefficients, 'LineWidth', 1.5);
title('Wavelet Coefficients');
xlabel('Time');
ylabel('Amplitude');
grid on;
axis tight;
```

```
subplot(212);
stem(wavelet_coefficients_regenerated, 'LineWidth', 1.5);
title('Regenerated Wavelet Coefficients');
xlabel('Time');
ylabel('Amplitude');
grid on;
axis tight;
```

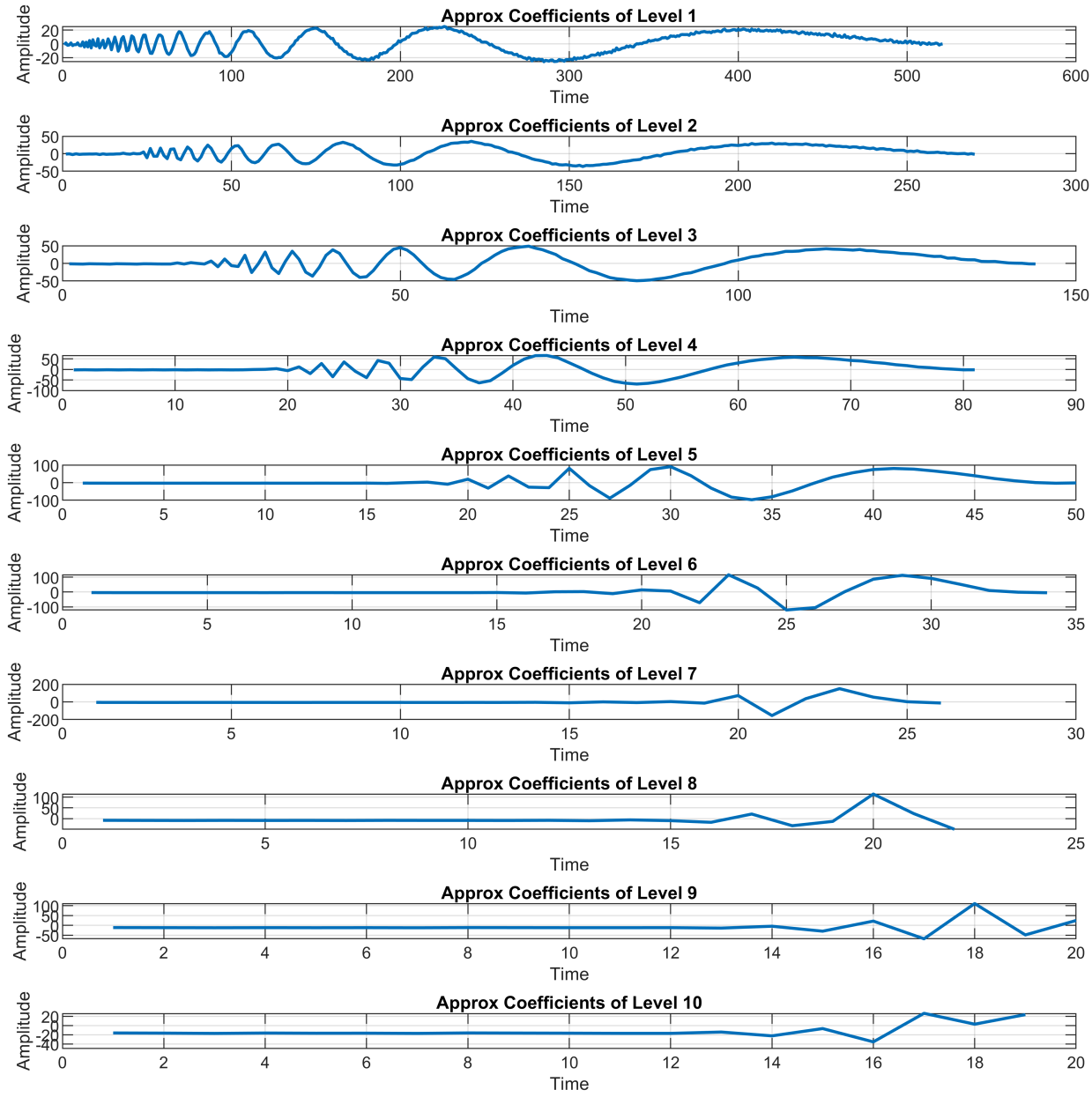


Part 8:

We want to calculate approximated coefficient of each level

```
figure('Name', 'Wavelet Coefficients of Different Levels');
set(gcf, 'Position', [100, 100, 1000, 1000]);

for i = 1:numberOfLevels
    approx_coeff_1 = appcoef(waveletCoefficients, waveletLevels, 'db10', i);
    subplot(numberOfLevels, 1, i);
    plot(approx_coeff_1, 'LineWidth', 1.5);
    title(['Approx Coefficients of Level ', num2str(i)]);
    xlabel('Time');
    ylabel('Amplitude');
    grid on;
end
```



Part 9:

Here we want to reconstruct signal with substitute first 3 layer of detailed coefficient with 0

and then plot it with noisy signal

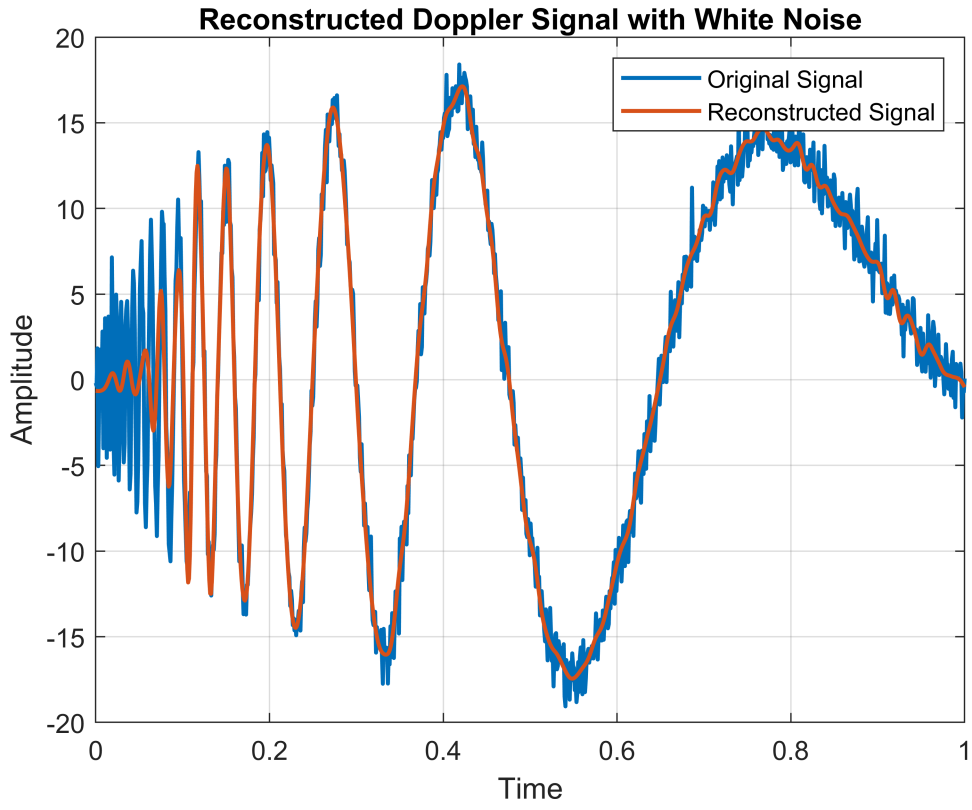
```
numberOfLevels1 = 3;
approx_coeff_deepest = appcoef( ...
    waveletCoefficients, ...
    waveletLevels, ...
```

```
    'db10', ...
    numberOfLevels1 ...
);
wavelet_coefficients_regenerated = ...
    [approx_coeff_deepest, detailed_coeff{numberOfLevels1:-1:1}];

x_rec = waverec(waveletCoefficients, waveletLevels, 'db10');
length1 = idwt(approx_coeff_deepest, zeros(1, 144), 'db10');
length2 = idwt(length1, zeros(1, 270), 'db10');
length3 = idwt(length2, zeros(1, 522), 'db10');

loc_re = linspace(0, 1, 2 ^ N + 2);

figure('Name', 'Reconstructed Doppler Signal with White Noise');
plot(loc, x_rec, 'LineWidth', 1.5);
title('Reconstructed Doppler Signal with White Noise');
xlabel('Time');
ylabel('Amplitude');
grid on;
hold on;
plot(loc_re, length3, 'LineWidth', 1.5);
legend('Original Signal', 'Reconstructed Signal');
```



Part 10:

Here we want to denoise the signal and compare it with original and noisy signal

```
denoised_x = wdenoise(noisy_x, 10);
```

```
figure('Name', 'Doppler Signal with White Noise vs Denoised Heavy Sine Signal');
plot(loc, noisy_x, 'LineWidth', 1.5);
title('Doppler Signal with White Noise');
xlabel('Time');
ylabel('Amplitude');
grid on;
hold on;
title('Denoised Doppler Signal');
xlabel('Time');
ylabel('Amplitude');
plot(loc, denoised_x, 'LineWidth', 1.5);
grid on;
hold on;
title('Doppler Signal');
xlabel('Time');
ylabel('Amplitude');
plot(loc, x, 'LineWidth', 1.5);
grid on;
hold on;
legend('Noisy Signal', 'Denoised Signal', 'Original Signal');
```