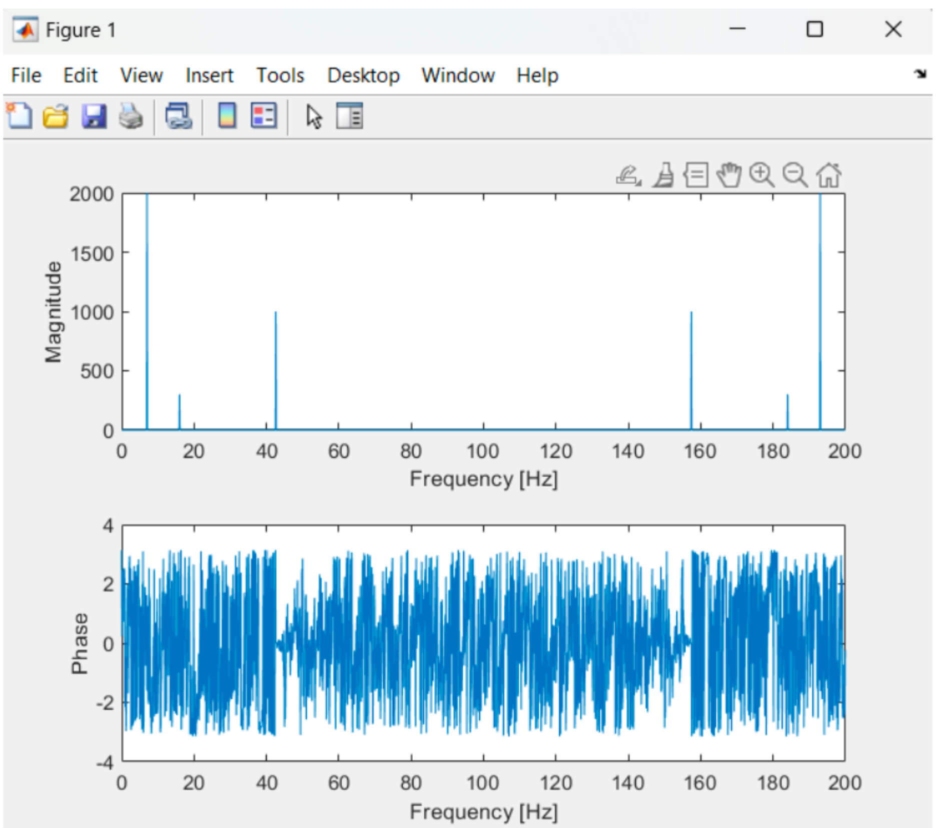
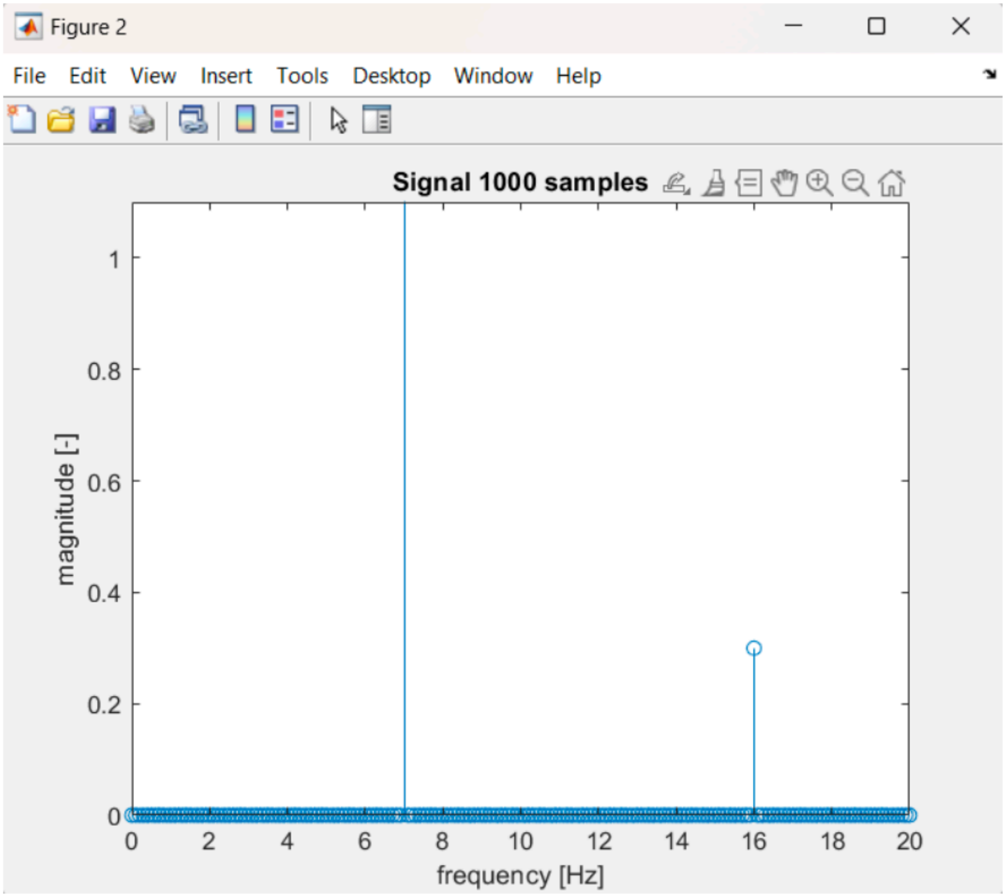
Signals_LAB4_Sonmez_Isil

```
%% Task 1

n=10; % n-number of letters in your name
s=6;  % s-number of letters in your surname
T=10; % total length of time

fs=200; % sampling frequency
dt=1/fs; % time step
t=0:dt:T-dt; % time vector
s=2*sin(2*pi*0.7*n*t)+cos(2*pi*7.1*s*t)+0.3*cos(2*pi*(s+n)*t+0.5*pi);
S=fft(s);

N=length(s);  % number of samples
df=fs/N; % frequency step
f_vec = (0:(N-1))*df; % frequency step
S_amp = abs(S); % Amplitude
S_phase = atan2(imag(S),real(S)); % phase
S_amp_norm = abs(S)/N*2; S_amp_norm(1)=S_amp_norm(1)/2;
S_phase_norm = S_phase; S_phase_norm(S_amp_norm<0.01)=0;

figure(1), subplot(2,1,1), plot(f_vec, S_amp) % plot amplitude spectrum
xlabel('Frequency [Hz]'), ylabel('Magnitude')
subplot(2,1,2), plot(f_vec, S_phase)
xlabel('Frequency [Hz]'), ylabel('Phase')
```



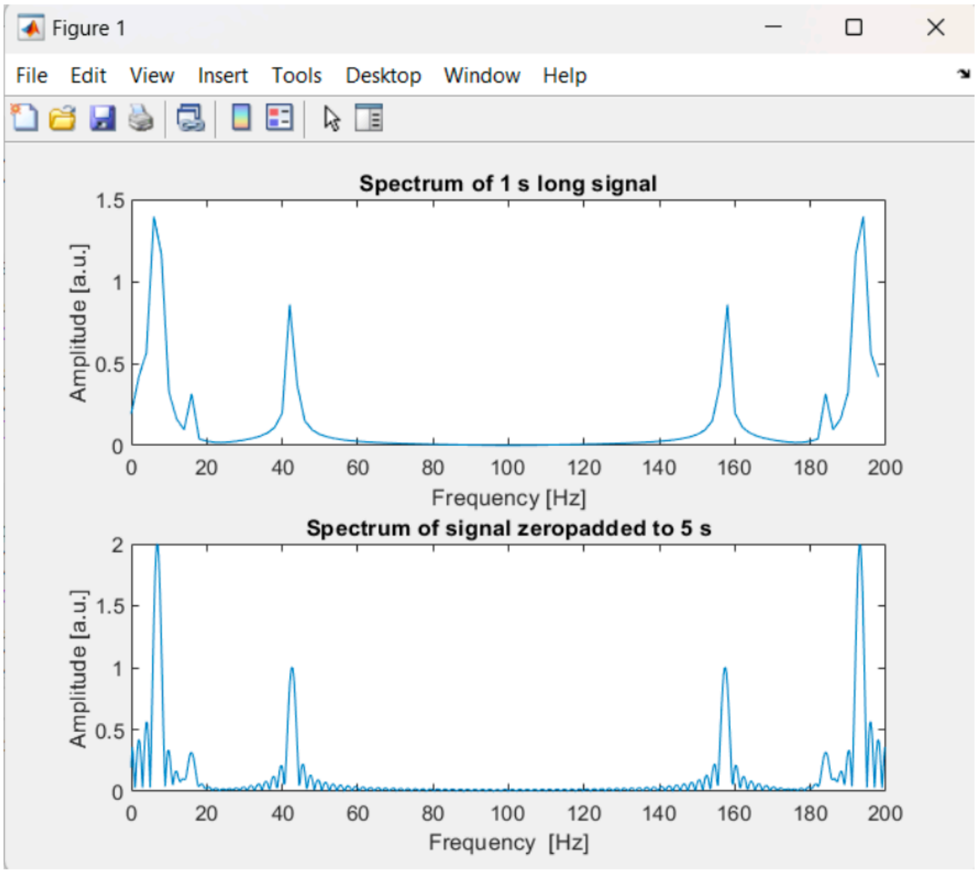
The code's results depends on looking at the signal's amplitude and phase. They show what frequencies are in the signal and how strong they are. The total time of the signal affects how clear these frequencies are. A longer time gives better detail, helping separate close frequencies. But with a shorter time, it's harder to see each frequency clearly and they might overlap. So, choosing the right time helps us see the signal's frequencies more accurately.

```
figure(2), subplot(1,1,1), stem(f_vec,abs(fft(s))/N*2),
xlim([0 20]), ylim([0 1.1]), ylabel('magnitude [-]')
xlabel('frequency [Hz]'); title('Signal 1000 samples'),
```



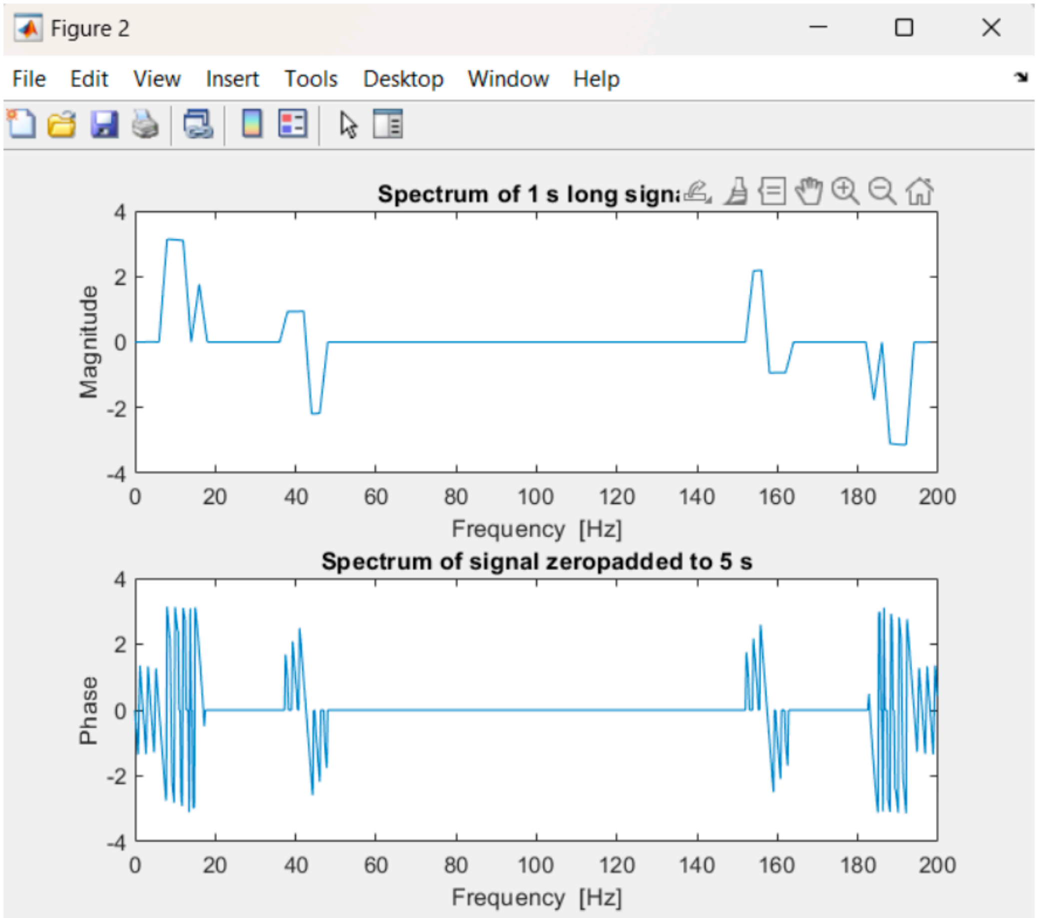

```
%% Task 2

n=10; % n-number of letters in your name
s=6;  % s-number of letters in your surname
T=0.5; % total length of time

fs=200; % sampling frequency
dt=1/fs; % time step
t=0:dt:T-dt; % time vector

x=2*sin(2*pi*0.7*n*t)+cos(2*pi*7.1*s*t)+0.3*cos(2*pi*(s+n)*t+0.5*pi);
N=length(x); % number of samples
df=fs/N; %frequency vector
fv = (0:(N-1))*df;  % frequency step
y=fft(x);
amp=2*abs(y)/N;
amp(1)=amp(1)/2;
ph=atan2(imag(y),real(y)); ph(amp<0.1)=0;
N2=4*fs;

df2=fs/N2; fv2=(0:N2-1)*df2;
y=fft(x,N2);
amp2=2*abs(y)/N;

amp2(1)=amp2(1)/2;
ph2=atan2(imag(y),real(y)); ph2(amp2<0.1)=0;
```

```
figure(1),
subplot(2,1,1), plot(fv,amp), xlim([0 fs/2]);
xlabel('Frequency [Hz]'), ylabel('Amplitude [a.u.]');xlim([0 200]);
title('Spectrum of 1 s long signal')
subplot(2,1,2), plot(fv2,amp2), xlim([0 fs/2]);
xlabel('Frequency  [Hz]'), ylabel('Amplitude [a.u.]');xlim([0 200]);
title('Spectrum of signal zeropadded to 5 s')
```

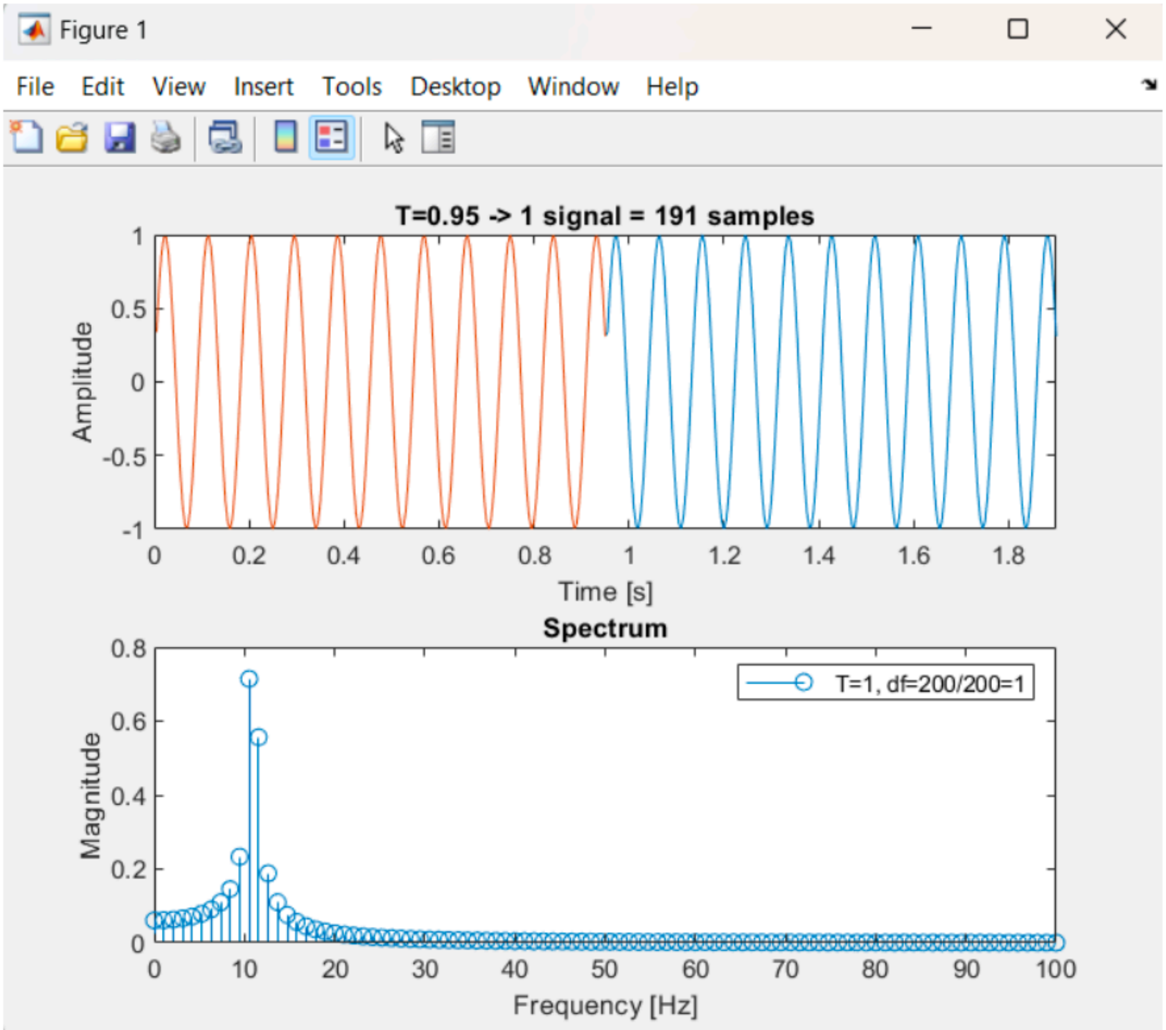



```
figure(2)
subplot(2,1,1), plot(fv,ph), xlim([0 fs/2]);
xlabel('Frequency  [Hz]'), ylabel('Magnitude');xlim([0 200]);
title('Spectrum of 1 s long signal')
subplot(2,1,2), plot(fv2,ph2), xlim([0 fs/2]);
xlabel('Frequency  [Hz]'), ylabel('Phase');xlim([0 200]);
title('Spectrum of signal zeropadded to 5 s')
```

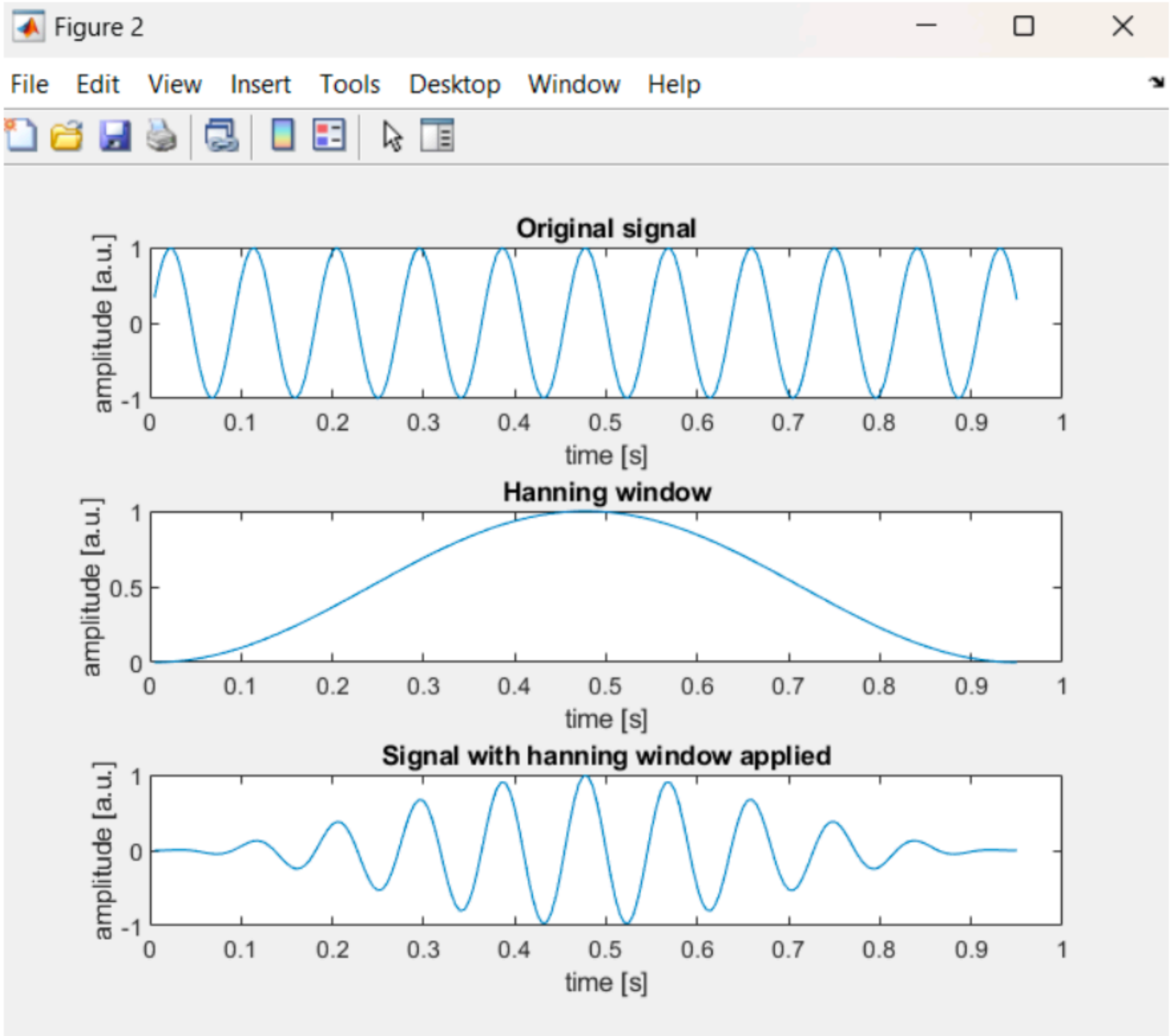
The code compares how the frequency spectrum of a signal changes when we pad it with zeros to make it longer. When the signal is short (0.5 seconds), it's like hard to see the details. But when we do it to 5 seconds, it's like making the details clearer. So, adding zeros helps us see the signal's frequencies better



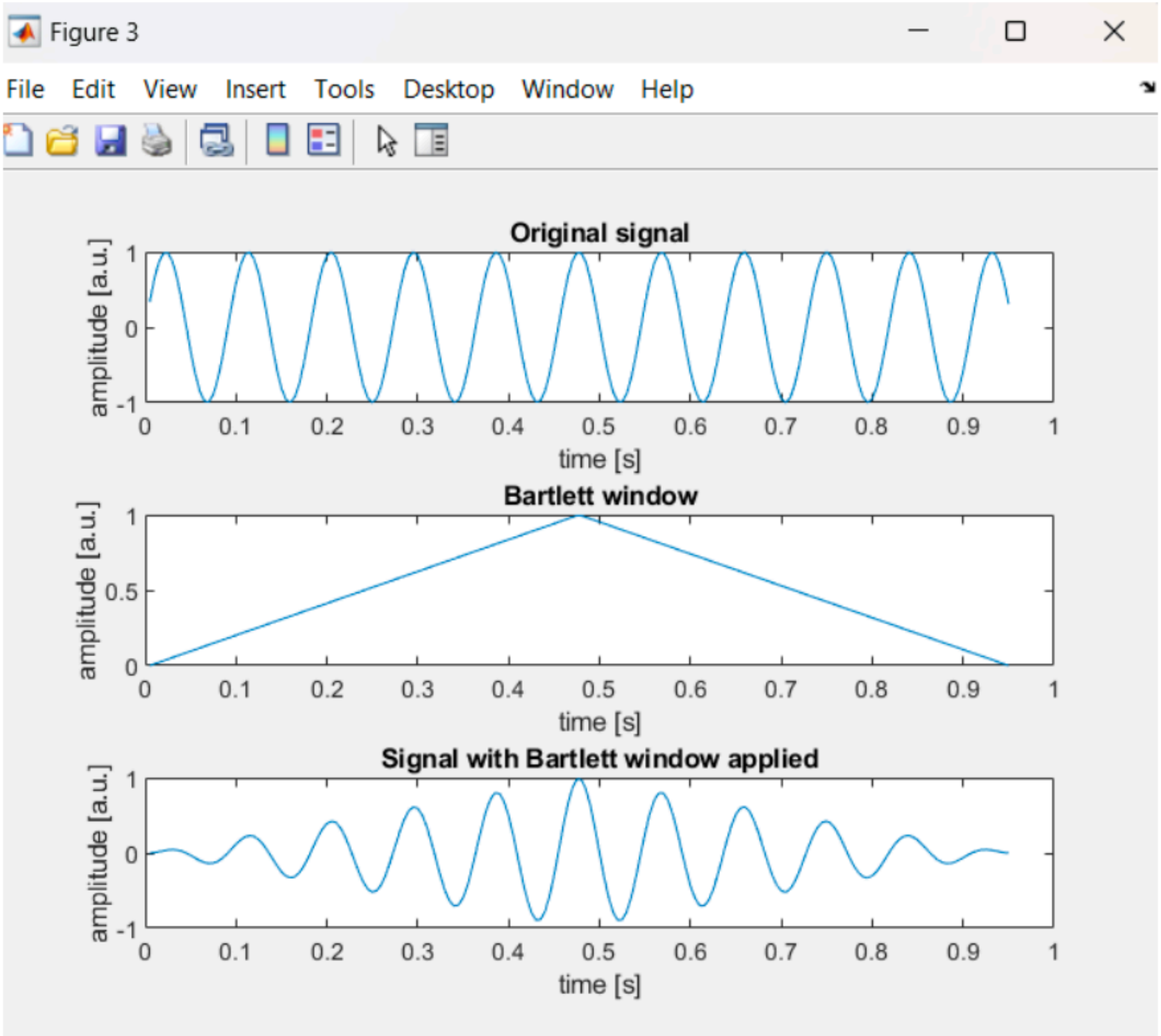

```
%% Task 3
clc;
clear all;
close all;
fs=200;
dt=1/fs;
t=dt:dt:0.95;
x=sin(2*pi*11*t);
T=dt:dt:length(t)*2*dt;
X=[x x];
y=abs(fft(x))/length(x)*2;
f=(0:length(y)-1)*fs/length(y);

figure(1)
subplot(2,1,1), plot(T,X), hold on, plot(t,x); hold off;
xlim([0 max(T)]); xlabel('Time [s]'), ylabel('Amplitude'),
title('T=0.95 -> 1 signal = 191 samples');
subplot(2,1,2), stem(f,y); hold off;
xlim([0 fs/2]); xlabel('Frequency [Hz]'), ylabel('Magnitude'),
title('Spectrum');
legend('T=1, df=200/200=1','T=0.95, df=200/191=1.0471');
```




```
window1=hann(length(x)); % window definition
x1=x.*window1.'; % resulting signal is the product of
                 % original signal and time window

window2=bartlett(length(x)); % window definition
x2=x.*window2.'; % resulting signal is the product of

window3=chebwin(length(x)); % window definition
x3=x.*window3.'; % resulting signal is the product of

figure(2),
subplot(3,1,1), plot(t,x); xlabel('time [s]'),
ylabel('amplitude [a.u.]'), title('Original signal')
subplot(3,1,2), plot(t,window1); xlabel('time [s]'),
ylabel('amplitude [a.u.]'), title('Hanning window')
subplot(3,1,3), plot(t,x1); xlabel('time [s]'),
ylabel('amplitude [a.u.]'), title('Signal with hanning window applied')
```




```
figure(3)
subplot(3,1,1), plot(t,x); xlabel('time [s]'),
ylabel('amplitude [a.u.]'), title('Original signal')
subplot(3,1,2), plot(t,window2); xlabel('time [s]'),
ylabel('amplitude [a.u.]'), title('Bartlett window')
subplot(3,1,3), plot(t,x2); xlabel('time [s]'),
ylabel('amplitude [a.u.]'), title('Signal with Bartlett window applied')
```




```
figure(4)
subplot(3,1,1), plot(t,x); xlabel('time [s]'),
ylabel('amplitude [a.u.]'), title('Original signal')
subplot(3,1,2), plot(t,window3); xlabel('time [s]'),
ylabel('amplitude [a.u.]'), title('Chebyshev window')
subplot(3,1,3), plot(t,x3); xlabel('time [s]'),
ylabel('amplitude [a.u.]'), title('Signal with Chebyshev window applied')
```

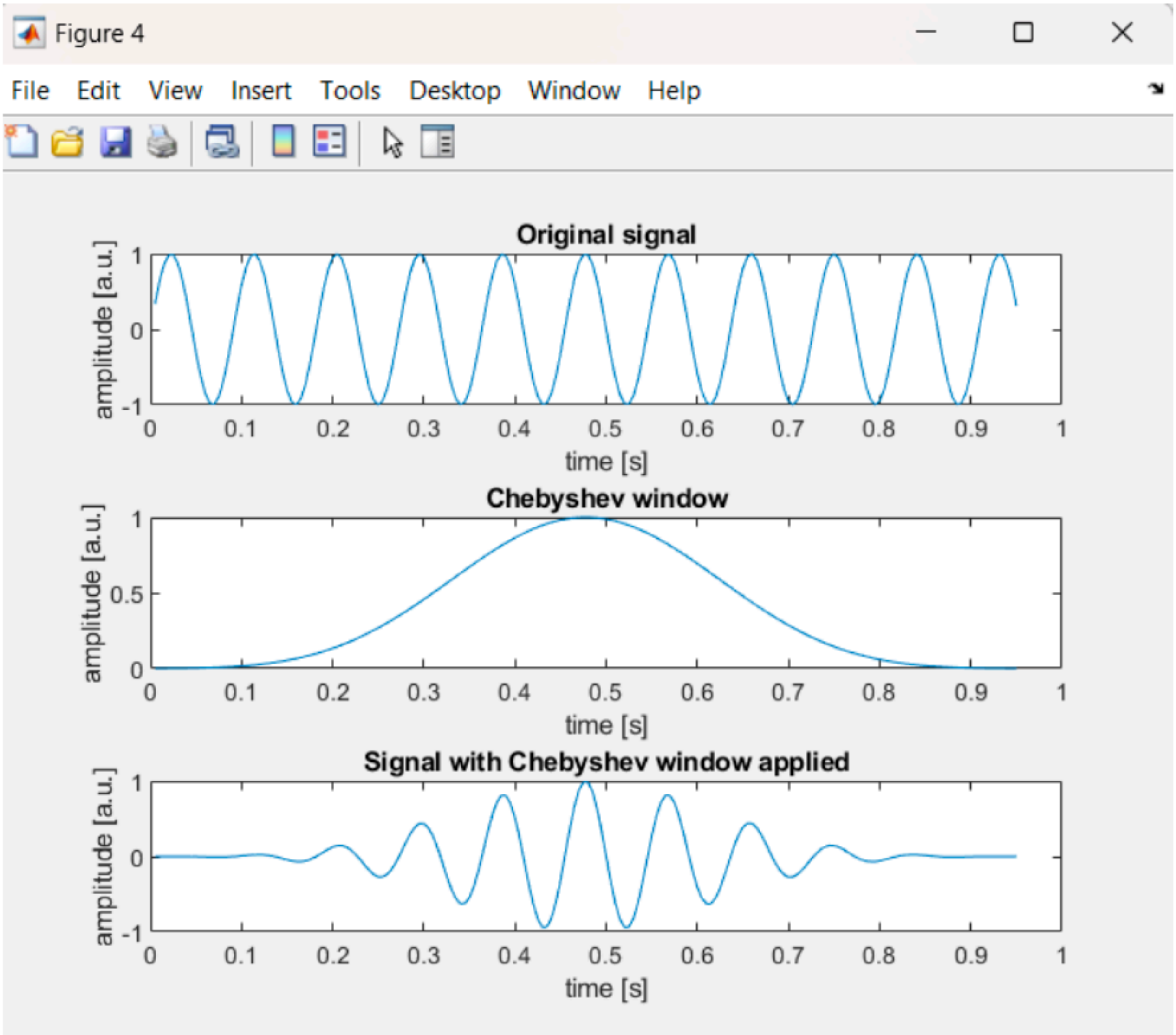


Figure 4
File Edit View Insert Tools Desktop Window Help
Original signal
amplitude [a.u.]
time [s]
Chebyshev window
amplitude [a.u.]
time [s]
Signal with Chebyshev window applied
amplitude [a.u.]
time [s]

```
clc;
clear all;
close all;

fs = 200;
dt = 1/fs;
t = dt:dt:0.95;
x = sin(2*pi*11*t);
T = dt:dt:length(t)*2*dt;
X = [x x];
y_original = abs(fft(x))/length(x)*2;
f_original = (0:length(y_original)-1)*fs/length(y_original);

% Different types of windows
window_types = {@rectwin, @bartlett, @hann, @hamming, @blackman};
window_labels = {'Rectangular', 'Bartlett', 'Hann', 'Hamming', 'Blackman'};
% Additional window types
window_types_extra = {@hanning, @chebwin, @bartlett};
window_labels_extra = {'Hanning', 'Chebyshev', 'Bartlett'};

figure;

for i = 1:length(window_types_extra)
    window_func = window_types_extra{i};
    window_label = window_labels_extra{i};

    % Apply windowing function
    window_length = length(x);
    window = window_func(window_length);
    x_windowed = x .* window';

    % Calculate FFT of original and windowed signals
    y_original = abs(fft(x))/length(x)*2;
    y_windowed = abs(fft(x_windowed))/length(x_windowed)*2;

    % Plot frequency spectra
    subplot(length(window_types_extra), 1, i);
    plot(f_original, y_original, 'b'); hold on;
    plot(f_original, y_windowed, 'r'); hold off;
    xlim([0 fs/2]);
    xlabel('Frequency [Hz]'), ylabel('Magnitude'),
    title(['Frequency spectrum with ' window_label ' window']);
    legend('Original Signal', 'Windowed Signal');
end
```

Conclusion

```
%Windowing functions reduce spectral leakage by smoothing the signal's edges.
This prevents signal energy from spreading into neighboring frequency bins.
Different windows offer varying levels of leakage reduction, balancing
factors like main lobe width and side lobe levels. Choosing the right window
function is about finding the best compromise for your signal and analysis
goals.
```

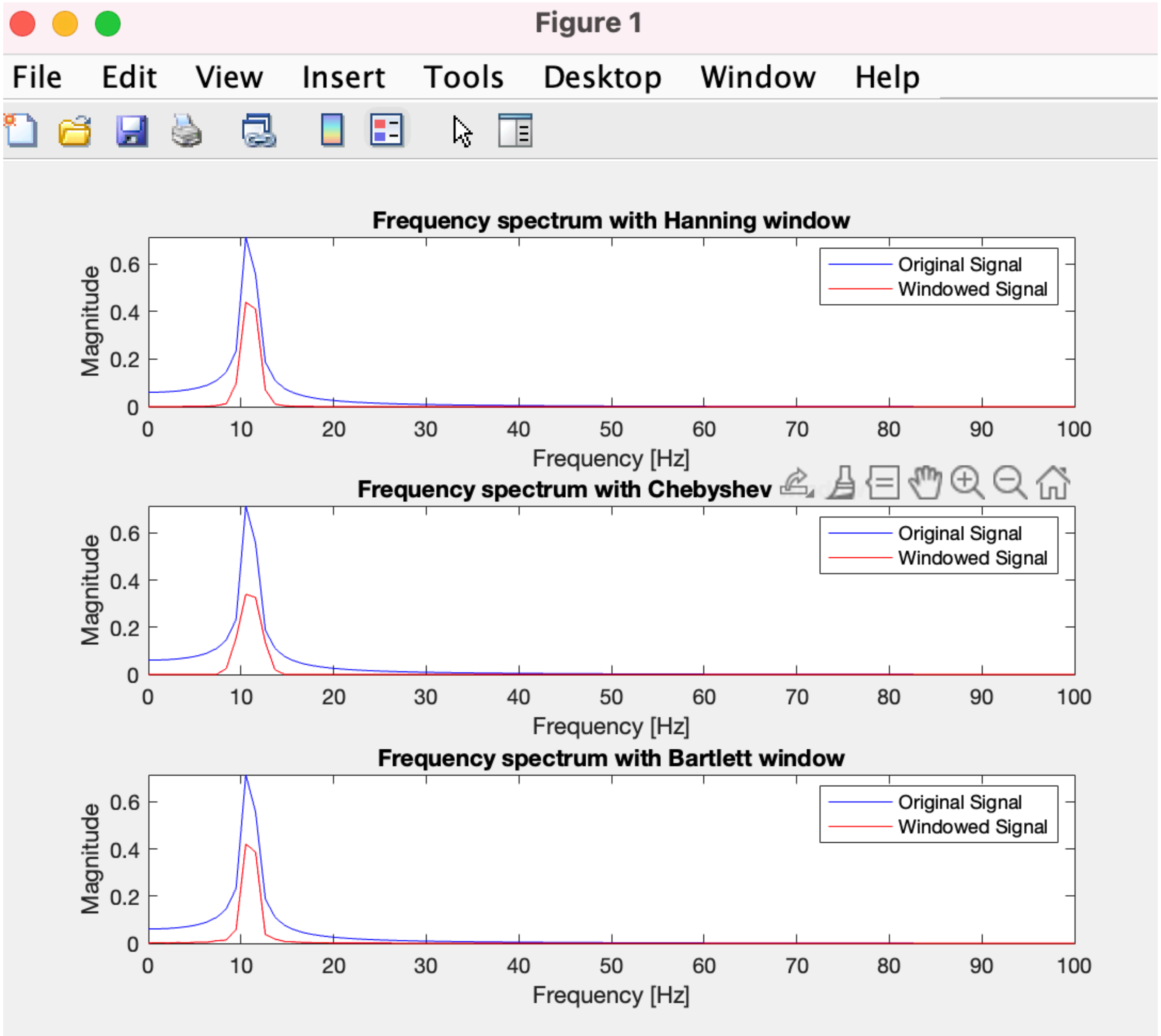


Figure 1
File Edit View Insert Tools Desktop Window Help
Frequency spectrum with Hanning window
Magnitude
Frequency [Hz]
Original Signal
Windowed Signal
Frequency spectrum with Chebyshev
Magnitude
Frequency [Hz]
Original Signal
Windowed Signal
Frequency spectrum with Bartlett window
Magnitude
Frequency [Hz]
Original Signal
Windowed Signal