National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering and Computer Science

Faculty Member: Dr. Ahmad Salman Dated: 17/03/2023

Semester: 6th Section: BEE 12C

EE-330 Digital Signal Processing

Lab 7: DFT Spectral Leakage

Group Members

		PLO4 - CLO4		PLO5 - CLO5	PLO8 - CLO6	PLO9 - CLO7
Name	**Reg. No**	**Viva / Quiz / Lab Performance**	**Analysis of data in Lab Report**	**Modern Tool Usage**	**Ethics and Safety**	**Individual and Teamwork**
		5 Marks	**5 Marks**	**5 Marks**	**5 Marks**	**5 Marks**
Danial Ahmad	331388					
Muhammad Umer	345834					
Tariq Umar	334943					

1 Table of Contents

2 Sampling of Audio Signals in MATLAB

2.1 Objectives

- The objective in this lab is to explore basic spectral analysis using DFT and block convolution DFT practical implementation and limitations

2.2 Introduction

The objective of this lab is to explore basic spectral analysis using DFT (Discrete Fourier Transform) and block convolution DFT practical implementation and limitations. Spectral analysis is an important technique for analyzing signals in the frequency domain and can be used to extract information about the frequency content of a signal. The DFT is a powerful tool for spectral analysis and is commonly used in signal processing applications. In this lab, we will use MATLAB to perform spectral analysis on different signals using DFT and block convolution DFT techniques. We will also investigate the limitations of these techniques and explore methods for mitigating them.

2.3 Software

MATLAB is a high-level programming language and numerical computing environment. Developed by MathWorks, it provides an interactive environment for numerical computation, visualization, and programming. MATLAB is widely used in various fields, including engineering, science, and finance, due to its capabilities for matrix and vector operations, implementation of algorithms, and creation of graphical representations of data. The objective of this lab is to provide a hands-on experience with the A-to-D sampling and the D-to-A reconstruction processes that are essential for digital image processing. We will also demonstrate a commonly used method of image zooming (reconstruction) that produces “poor” results, which will help illustrate the importance of understanding the underlying principles of digital image processing.

2.4 Lab Report Instructions

All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

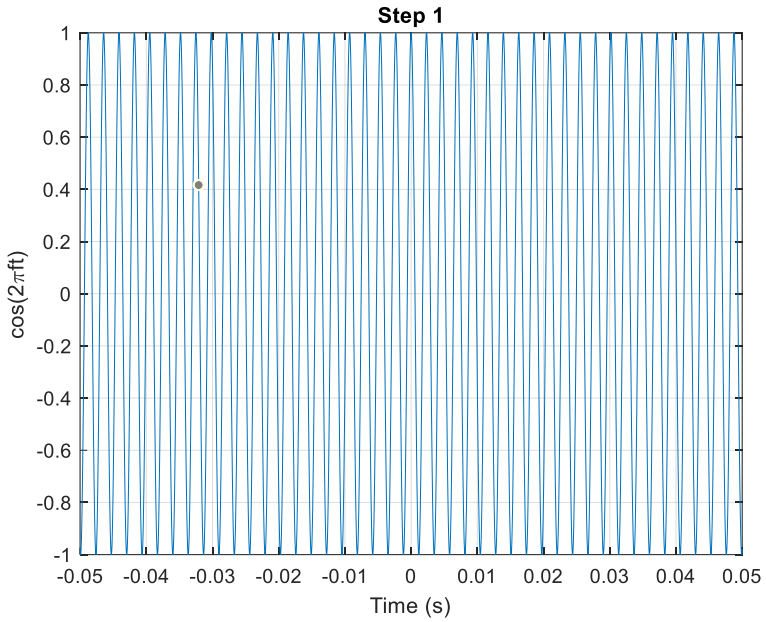
- Lab objectives
- MATLAB codes
- Results (graphs/tables) duly commented and discussed
- Conclusion

3 Lab Procedure

1. Generate a sinusoid of 0.1 second duration sampled at fs = 44.1 KHz. Set the frequency of sinusoid to be f = 10*fs/1024.

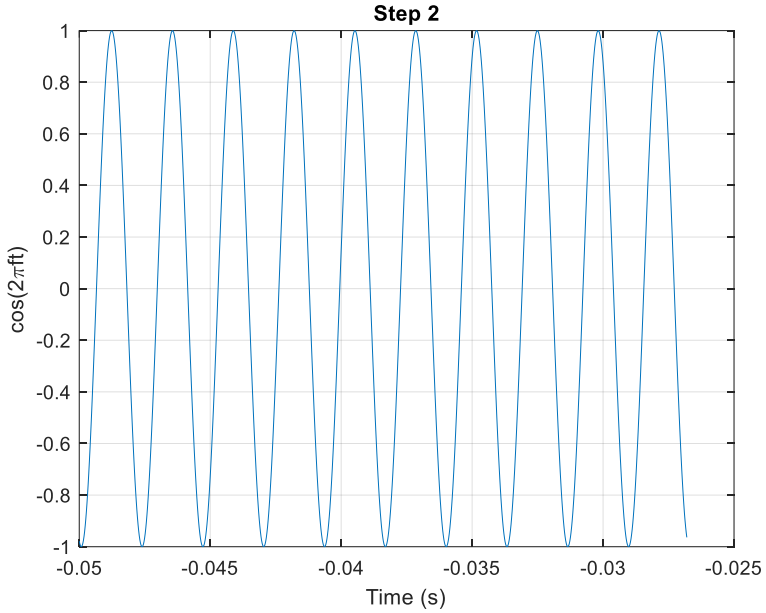
$$x1 = cos(2 * pi * f * t)$$

```
close all;
Fs = 44100;
t = -0.1/2:1 / Fs:0.1/2;
f = 10 * Fs / 1024;
x1 = cos(2 * pi * f * t);

figure
plot(t, x1)
title('Step 1: Original Signal')
xlabel("Time (s)")
ylabel('cos(2\pift)')
```



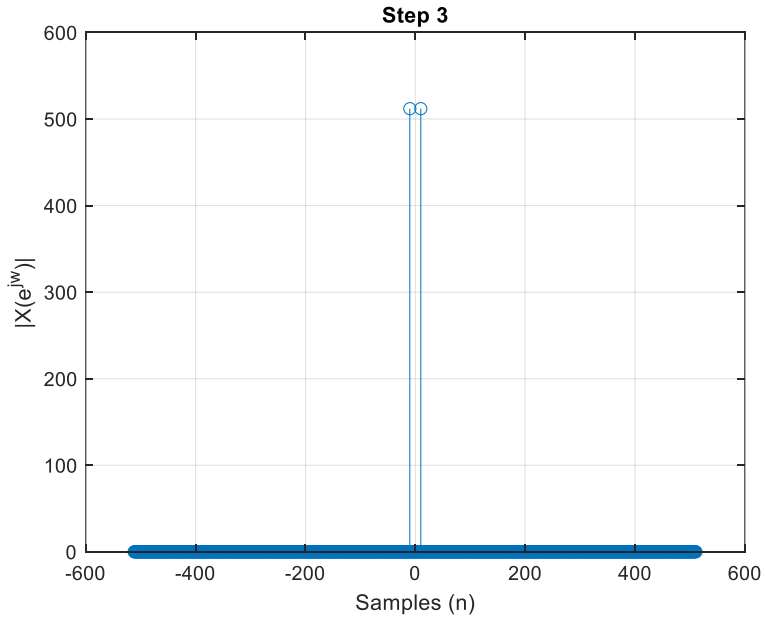
2. Display the first 1024 samples of this sinusoid as a function of time in seconds.

```
figure
plot(t(1:1024), x1(1:1024))
title('Step 2: Initial 1024 Points')
xlabel('Time (s)')
ylabel('cos(2\pift)')
```



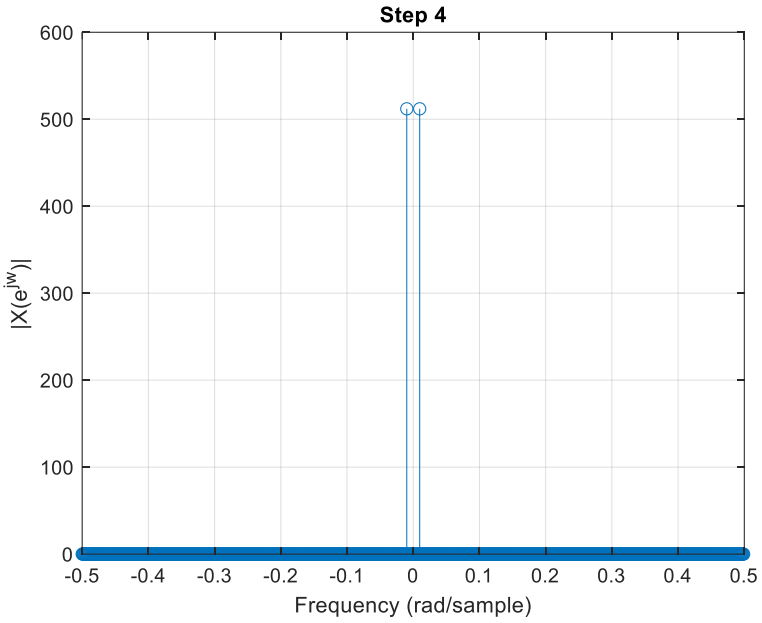
3. Compute the DFT of the signal x1 using FFT of size N = 1024 (refer to help fft). Plot the magnitude spectrum as a function of 'k', the bin nu number (use stem function). Explain the peaks obtained in different bins by linking them to the original sinusoid.

```
n = -512:(512 - 1);
N = 1024;
X1 = fftshift(fft(x1, N));

figure
stem(n, abs(X1))
title('Step 3: Sampled FFT')
xlabel('Samples (n)')
ylabel('|X(e^{jw})|')
```



4. Plot the magnitude spectrum again but now as a function of normalized (digital) frequency. For this purpose, define the frequency axis from - π to π (radians/sample) or -0.5 to 0.5 (cycles/sample). Recall the definition of this frequency axis from class notes.

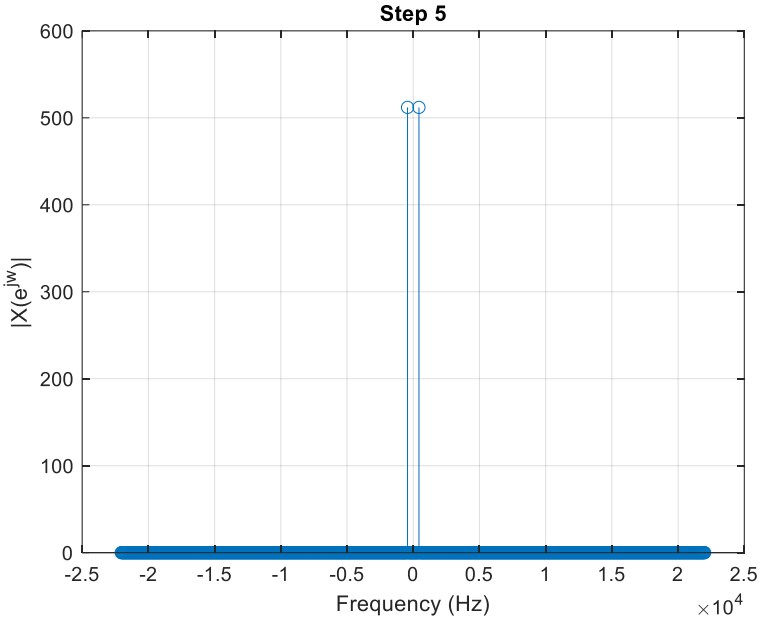
```
df = Fs / N;
index = -N / 2:(N / 2) - 1;
f_axis = (df * index);

figure
stem(f_axis / Fs, abs(X1))
title('Step 4: Normalized FFT')
xlabel('Frequency (rad/sample)')
ylabel('|X(e^{jw})|')
```



5. Redraw the same magnitude spectrum as a function of frequency in Hz and verify if you are able to identify the sinusoid that was input to the system. Is spectral leakage manifested in this case? Why/Why not?

```
df = Fs / N;
index = -N / 2:(N / 2) - 1;
f_axis = (df * index);

figure
stem(f_axis, abs(X1))
title('Step 5: Hertz FFT')
xlabel('Frequency (Hz)')
ylabel('|X(e^{jw})|')
```

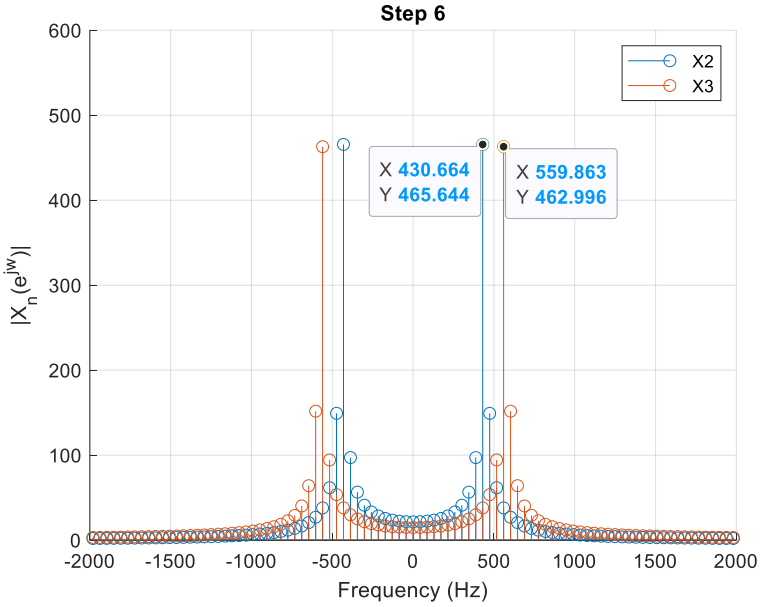
Yes, we are able to identify the original sinusoid as a sinusoidal signal has two impulses centered around its frequency components. There isn't any spectral leakage as the f_m of sinusoid is an integer multiple of the sampling frequency.



6. Generate two other sinusoids x2 and x3 with frequencies at 10.25 and 13.25 times 44100/1024 and display the first 1024 samples of each sinusoid. Compute the 1024-point DFT of x2 and x3 and display the magnitude spectra (as function of frequency in Hz) and try to identify their frequencies from the graphs. What is the change with respect to point (5) above? Elaborate.

```
f2 = 10.25 * Fs / 1024; f3 = 13.25 * Fs / 1024;
x2 = cos(2 * pi * f2 * t);
x3 = cos(2 * pi * f3 * t);
X2 = fftshift(fft(x2, N));
X3 = fftshift(fft(x3, N));

figure
hold on
stem(f_axis, abs(X2))
stem(f_axis, abs(X3))
title('Step 6: Spectral Leakage')
xlabel('Frequency (Hz)')
ylabel('|X(e^{jw})|')
legend('X2', 'X3')
axis([-2000 2000 0 600])
```

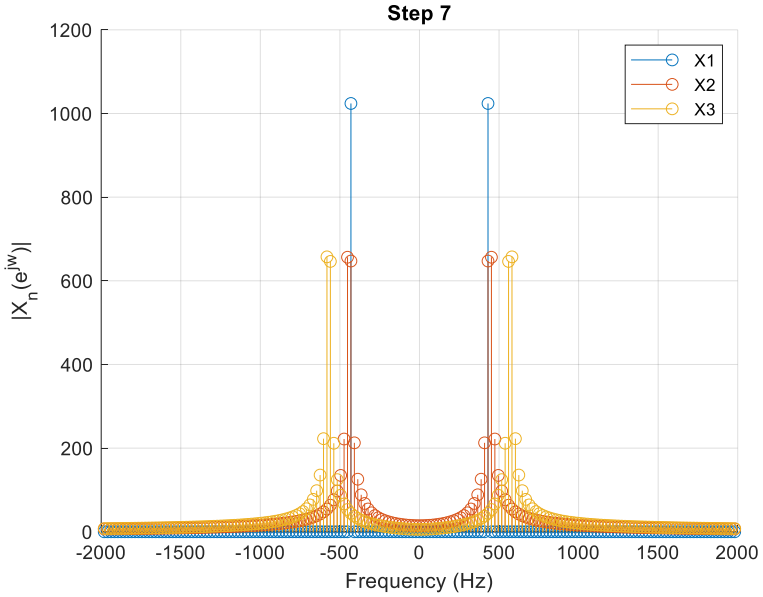
There will be spectral leakage as the f_m of sinusoid is not an integer multiple of the sampling frequency in either $x2$ or $x3$. It is further verified through the following figures, and that we have additional frequency components rather than a single impulse centered on f_m of respective sinusoids.



7. Repeat the DFT evaluations done in part 5 and part 6 but using a 2048-point FFT. (Recall that applying a 2048-point FFT to a 1024 length data buffer has the effect of appending (2048-1024) zeros to the end of the data, which increases the number of points at which the spectrum is evaluated. Compare the resultant spectra with those obtained in part 5 and part 6 and explain the differences.

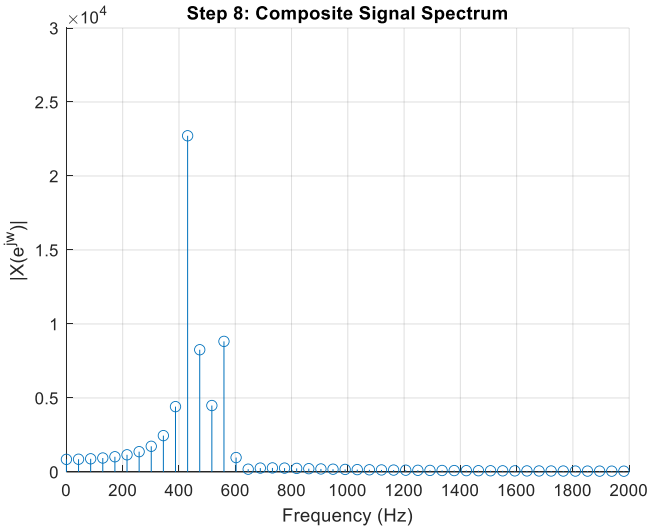
```
N = 2048;
X1 = fftshift(fft(x1, N));
X2 = fftshift(fft(x2, N));
X3 = fftshift(fft(x3, N));
df = Fs / N;
index = -N / 2:(N / 2) - 1;
f_axis = (df * index);

figure
hold on
stem(f_axis, abs(X1))
stem(f_axis, abs(X2))
stem(f_axis, abs(X3))
title('Step 7: N=2048 FFT')
xlabel('Frequency (Hz)')
ylabel('|X(e^{jw})|')
legend('X1', 'X2', 'X3')
axis([-2e3 2e3 0 1.2e3])
```

We observe that the zero-padding, which is essentially increasing the DFFT points, in this case instead worsened the spectral leakage of the signals $x2$ and $x3$, without having any meaningful effect on the original spectrum $x1$.



8. Construct a composite signal and its spectrum by adding up the sinusoids used in part 6 as follows: $xc = 50x2 + 15x3$

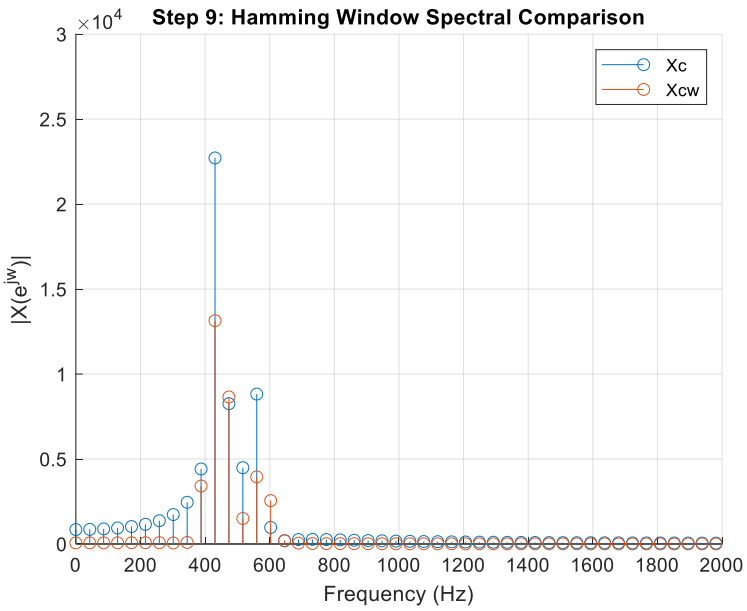
```
xc = 50 * x2 + 15 * x3;
N = 1024;
Xc = fftshift(fft(xc, N));
df = Fs / N;
index = -N / 2:(N / 2) - 1;
f_axis = (df * index);

figure
hold on
stem(f_axis, abs(Xc))
title('Step 8: Composite Signal Spectrum')
xlabel('Frequency (Hz)')
ylabel('|X(e^{jw})|')
axis([0 2e3 0 3e4])
```



No, it is not possible to clearly identify the peaks from the background frequencies, as apparent from the above attacked figure.

9. Now create a new signal "xcw" and its spectrum by applying a hamming window of length 1024 to the first 1024 samples of "xc" as: $xcw = xc(1:1024).* hamming(1024)'$

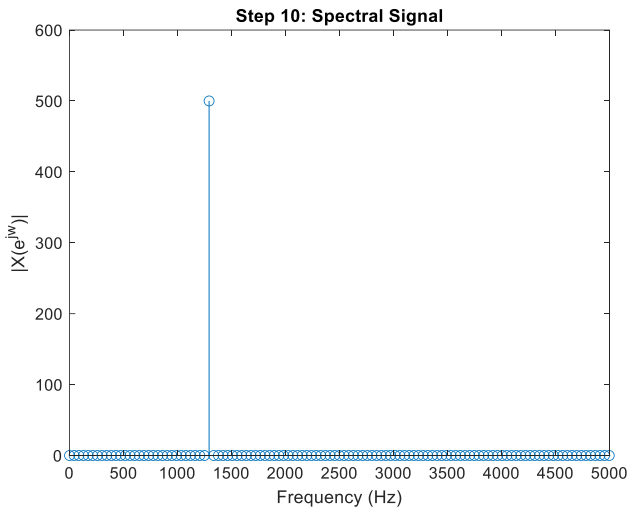
```
xcw = xc(1:1024) .* hamming(1024)';
Xcw = fftshift(fft(xcw, N));

figure
hold on
stem(f_axis, abs(Xc))
stem(f_axis, abs(Xcw))
title('Step 9: Hamming Window Spectral Comparison')
xlabel('Frequency (Hz)')
ylabel('|X(e^{jw})|')
axis([0 2e3 0 3e4])
legend('Xc', 'Xcw')
```

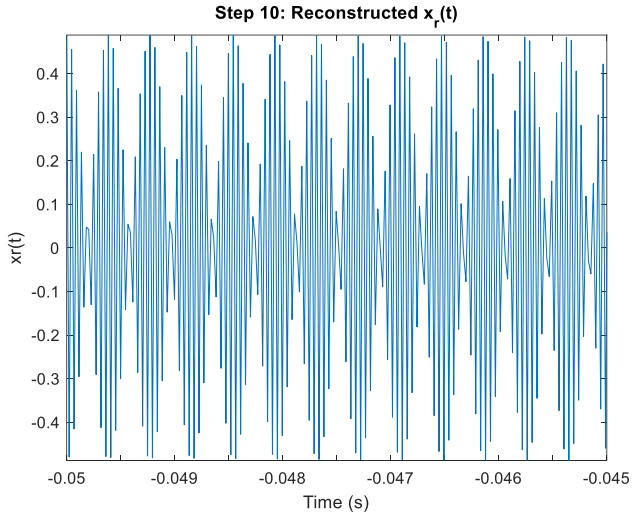


10. This part demonstrates the inverse DFT using the function ifft (). Generate a signal in the spectral domain by setting up a variable "Xr" of length 1024 (corresponding to the number of FFT points) and initialize it to zero. Now the desired time domain signal is a sinusoid of frequency 1291.99 Hz sampled at 44.1 kHz. Set the appropriate bins by putting the value N/2 = 512 in them, note which bins to set should be determined from the given specifications above. Take the inverse DFT "xr" of this signal "Xr". Plot the original spectrum as well as the obtained time domain signal "xr". Read out the frequency of the sinusoid from the time domain graph and compare it with the one from the spectrum.

```
N = 1024;
Fs = 44100;
t = -0.1/2:1 / Fs:0.1/2;
Xr = zeros(1, N);
```

```
f = 1291.99;
set_idx = find(min(abs(f_axis - f)) == abs(f_axis - f));
Xr(set_idx) = 500; % to have enough amplitude of x_r(t)
xr = ifft(Xr, N);

figure
stem(f_axis, abs(Xr))
title('Step 10: Spectral Signal')
xlabel('Frequency (Hz)')
ylabel('|X(e^{jw})|')
axis([0 5e3 0 600])

figure
plot(t(1:1024), real(xr))
title('Step 10: Reconstructed x_r(t)')
xlabel('Time (s)')
ylabel('xr(t)')
axis([-0.05 -0.045 -inf inf])
```





4 Conclusion

In conclusion, this lab introduced basic spectral analysis using DFT and block convolution DFT. We investigated the practical implementation of these techniques using MATLAB and explored their limitations. We learned that the choice of window function and block size can have a significant impact on the accuracy of the spectral analysis and that careful consideration must be given to these factors when performing spectral analysis. We also learned that increasing the DFT length can improve the frequency resolution but may also increase spectral leakage and computational complexity. Overall, this lab provided a valuable hands-on experience with spectral analysis techniques and gave us a deeper understanding of their practical implementation and limitations.