# Project Report for ECE 351

*Lab 09 - Fast Fourier Transform*

Skyler Corrigan

November 1, 2022

ECE351 Code Repository:
$https://github.com/ElfinPeach/ECE351_Code.git$

ECE351 Report Repository:
$https://github.com/ElfinPeach/ECE351_Report.git$

# Contents

# 1 Objective

The objective of this lab is to make Fast Fourier Transfer (FFT) functions. It involves three different functions unique to this lab and the function from Lab08. The unique functions are as follows:

$$cos(2\pi t)$$
$$5sin(2\pi t)$$
$$2cos((2\pi \cdot 2t) - 2) + sin^2((6\pi \cdot 6t) + 3)$$

# 2 Deliverables

## 2.1 Code

Below is the entirety of the code I used for this lab.

*Lab09 Code*

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.fftpack as fft

"""
User defined fast fourier transform funnction.
INPUTS:
    X is a function array
    fs is a frequency of sampleing rate
OUTPUTS:
    output, an array containig various information
    output[0], fourier transformation of the input
    output[1], same as previous, but zero frequency is
        the center of spectrum
    output[2], array of fourier outputs
    output[3], array of fourier magnatudes
    output[4], array of fourier angles
"""
def FFT(X, fs):

    #Length of input array
    n = len(X)

    #Fast fourier transorm
    X_fft = fft.fft(X)
```

```
    #shift zero frequency to center of the spectrium
    X_fft_shift = fft.fftshift(X_fft)

    # Calculate frequnecies for output using fs as a sampling frequency
    freqX = np.arange(-n/2, n/2) * fs / n

    #Calculate magnitude and phase
    magX = np.abs(X_fft_shift)/n
    angX = np.angle(X_fft_shift)

    output = [X_fft, X_fft_shift, freqX, magX, angX]
    return output

"""
CleanFFT
CleanFFT is like the user defined FFT functio, but if a given magnitude is
    less than 1e-10 the coresponding phase angle will be set to zero.
    This function depends of FFT function.
INPUTS:
    The inputs are the same as the previous function
OUTPUTS:
    output is a cleaner version fo the previous array
"""
def FFTClean(X, fs):
    XArry = FFT(X, fs)
    useableArry = [XArry[2], XArry[3], XArry[4]]
    for i in range(0, len(useableArry)-1):
        if (useableArry[1][i] <= 0.000000001):
            useableArry[2][i] = 0

    return useableArry


#----------Fourer transformations----------------

# Finding b_n for fourier estimation given an n
def b_n(n):
    b = (-2/((n)*np.pi)) * (np.cos((n) * np.pi) - 1)
    return b



def W(period):
    return ((2*np.pi)/period)
```

```
def xFourier(t, period, n):
    x_t = 0
    for i in np.arange(1, n+1):
        x_t += (np.sin(i * W(period) * t) * b_n(i))

    return x_t


#--function to plot stuff because I'm lazy
        and don't want to keep making them.

def plot_fft(title, x, X_mag, X_phi, freq, t, zmInt):

    #Calculate the zoomed in data for magnatude and frequency
    zm_mag = [];
    zm_mag_freq = [];
    for i in range(0, len(freq)-1):
        if ((freq[i]>=-zmInt) and (freq[i]<=zmInt)):
            zm_mag.append(X_mag[i])
            zm_mag_freq.append(freq[i])

    zm_ang = [];
    zm_ang_freq = [];
    for i in range(0, len(freq)-1):
        if ((freq[i]>=-zmInt) and (freq[i]<=zmInt)):
            zm_ang.append(X_phi[i])
            zm_ang_freq.append(freq[i])

    fig3 = plt.figure(constrained_layout=True)
    gs = fig3.add_gridspec(3, 2)
    f3_ax1 = fig3.add_subplot(gs[0, :])
    f3_ax1.set_title('User-Defined FFT of '+ title)
    f3_ax1.set_xlabel('time (s)')
    f3_ax1.plot(t,x)
    plt.grid()


    f3_ax2 = fig3.add_subplot(gs[1, 0])
    f3_ax2.set_title('|X(f)|')
    f3_ax2.set_ylabel("Magnitude")
    f3_ax2.stem(freq, X_mag)
    plt.grid()

    f3_ax3 = fig3.add_subplot(gs[1, 1])
    f3_ax3.set_title('Nicer |X(f)|')
```

```
    f3_ax3.stem(zm_mag_freq, zm_mag)
    plt.grid()

    f3_ax4 = fig3.add_subplot(gs[2, 0])
    f3_ax4.set_title('/_ X(f)')
    f3_ax4.set_xlabel('f (Hz)')
    f3_ax4.set_ylabel('Angle (rad)')
    f3_ax4.stem(freq, X_phi)
    plt.grid()

    f3_ax5 = fig3.add_subplot(gs[2, 1])
    f3_ax5.set_title('Nicer /_ X(f)')
    f3_ax5.set_xlabel('f (Hz)')
    f3_ax5.stem(zm_ang_freq, zm_ang)
    plt.grid()

    #Define step size
steps = 1e-2

    #t for part 1
start = 0
stop = 2
    #Define a range of t. Start at 0 and go to 20 (+a step)
t = np.arange(start, stop, steps)

# Sampling frquency for lab
fs = 100

# Task 1 input function, FFT, FFTClean
task1Func = np.cos(2* np.pi * t)
FFT_task1Func = FFT(task1Func, fs)
FFTCleanTask1 = FFTClean(task1Func, fs)

# Task 2 input function, FFT, FFTClean
task2Func = 5 * np.sin(2 * np.pi * t)
FFT_task2Func = FFT(task2Func, fs)
FFTCleanTask2 = FFTClean(task2Func, fs)

# Task 3  input function, FFT, FFTClean
task3Func = 2* np.cos((4*np.pi*t) - 2) + (np.sin((12*np.pi*t) + 3) )**2
FFT_task3Func = FFT(task3Func, fs)
FFTCleanTask3 = FFTClean(task3Func, fs)

#Fourier plot of the previous signal from lab 8
t2 = np.arange(0, 16, steps)
```

```
x_15 = xFourier(t2, 8, 15)

FFT_Lab8 = FFT(x_15, fs)

#Make the plots using the function!!!
plot_fft("Task 1 Dirty", task1Func, FFT_task1Func[3],
    FFT_task1Func[4], FFT_task1Func[2], t, 2)
plot_fft("Task 2 Dirty", task2Func, FFT_task2Func[3],
    FFT_task2Func[4], FFT_task2Func[2], t, 2)
plot_fft("Task 3 Dirty", task3Func, FFT_task3Func[3],
    FFT_task3Func[4], FFT_task3Func[2], t, 15)

# Plot the noise reduced versions of the functions
plot_fft("Task 1 Clean", task1Func, FFTCleanTask1[1],
    FFTCleanTask1[2], FFTCleanTask1[0], t, 2)
plot_fft("Task 2 Clean", task2Func, FFTCleanTask2[1],
    FFTCleanTask2[2], FFTCleanTask2[0], t, 2)
plot_fft("Task 3 Clean", task3Func, FFTCleanTask3[1],
    FFTCleanTask3[2], FFTCleanTask3[0], t, 15)

plot_fft("Lab 8 signal", x_15, FFT_Lab8[1], FFT_Lab8[2],
    FFT_Lab8[0], t2, 1e-15)
```

## 2.2 Figures

Below are the figures for each equation.

*Figure 1*

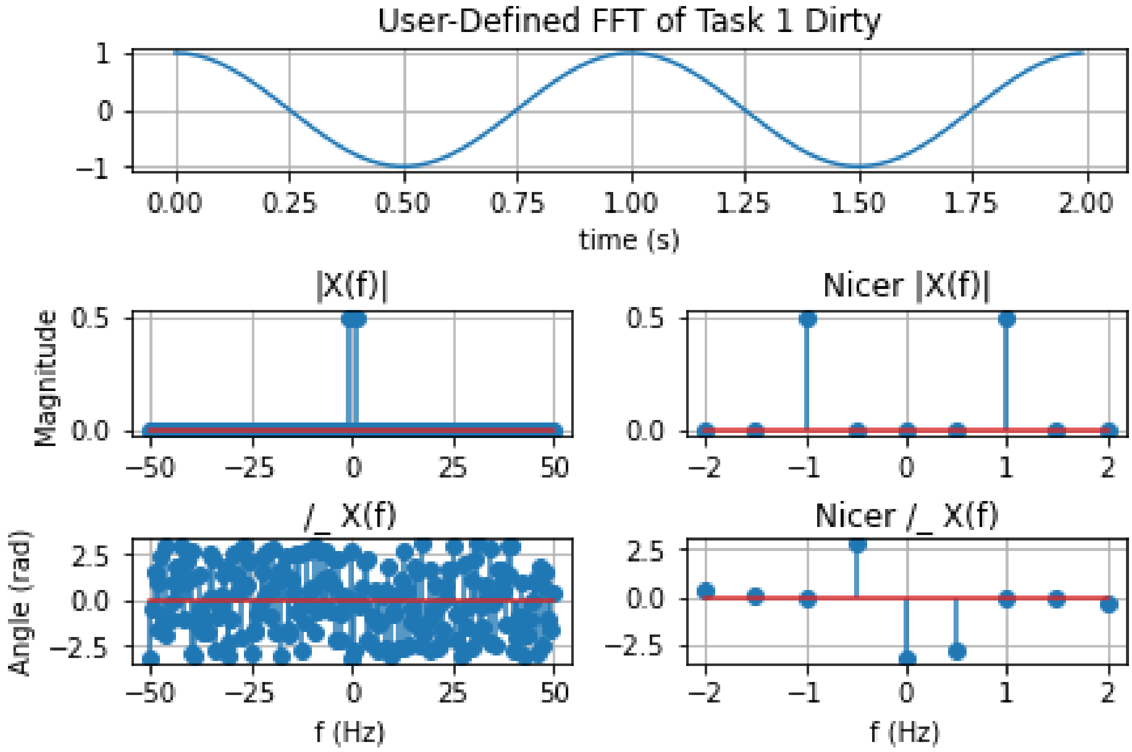



*Figure 2*

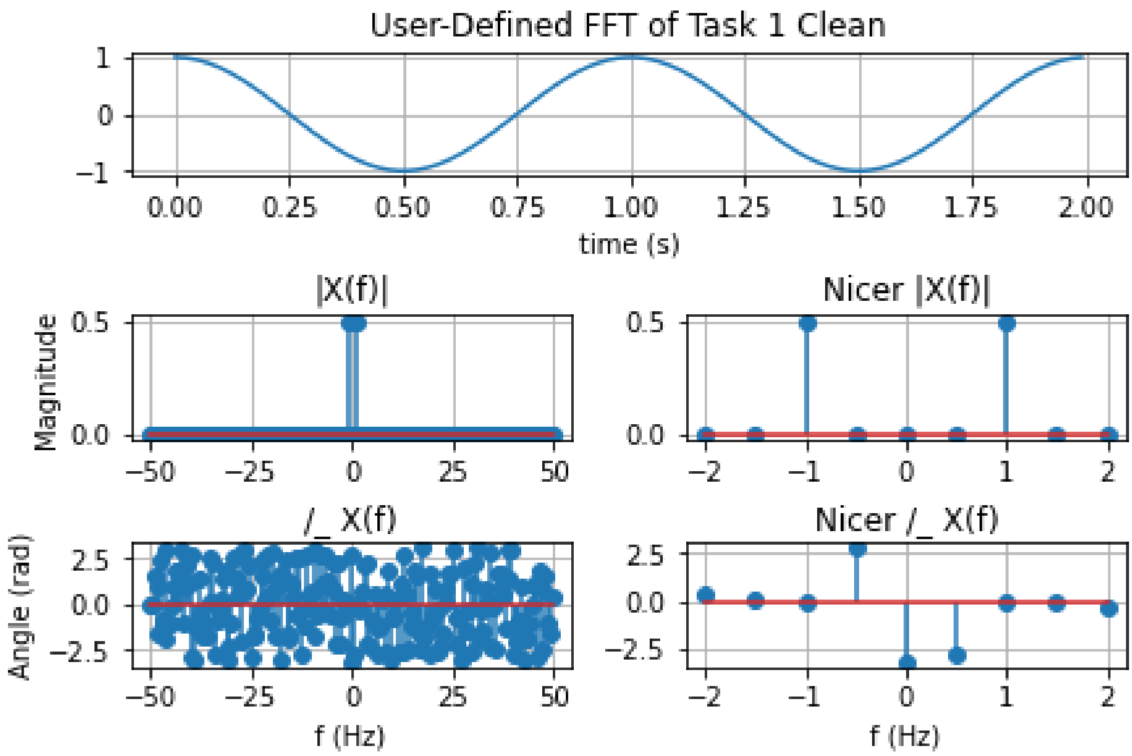


*Figure 3*

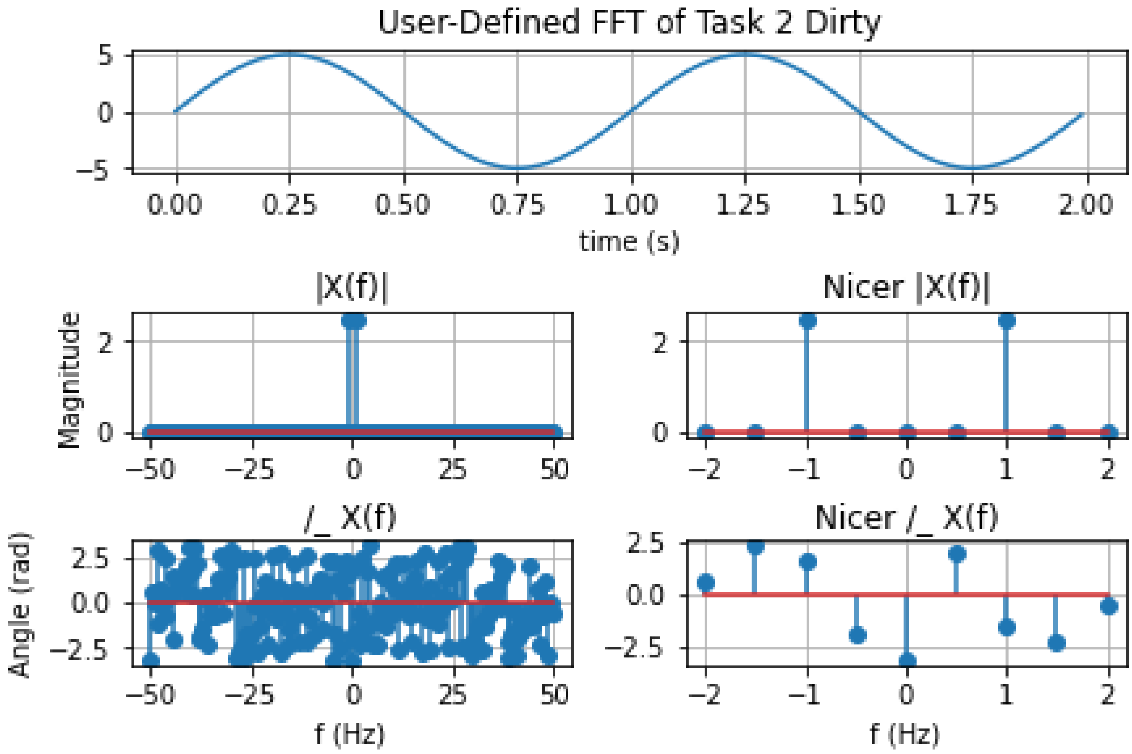



*Figure 4*

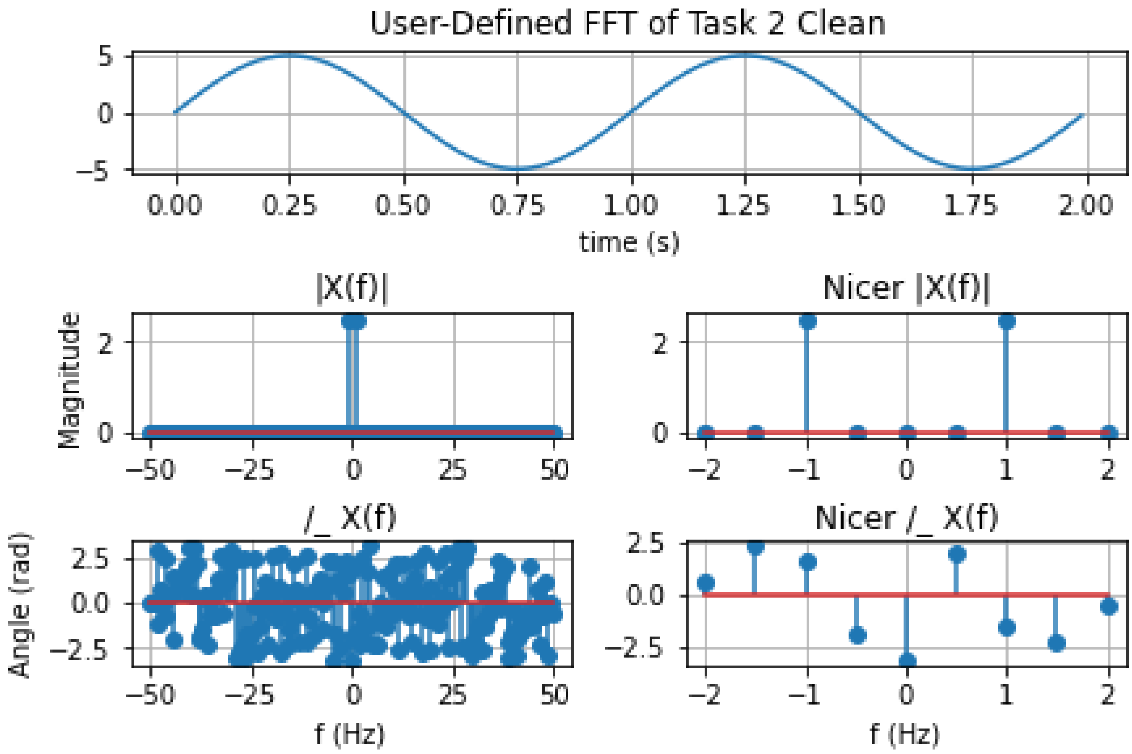


*Figure 5*

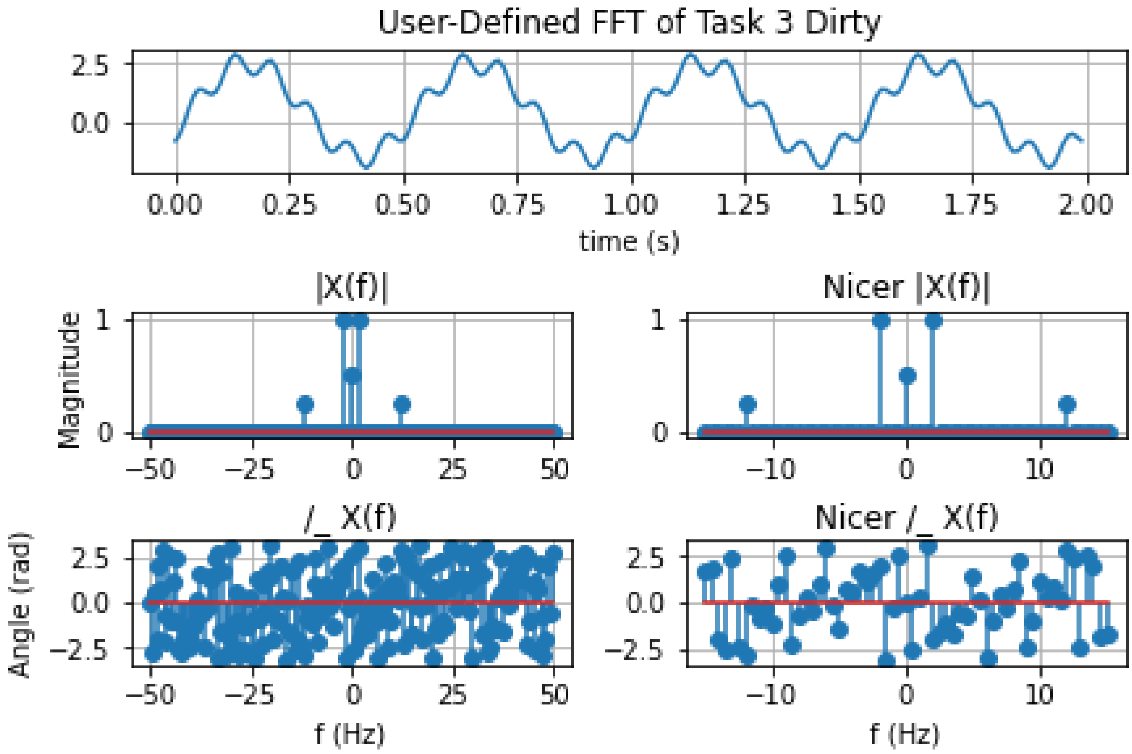



*Figure 6*

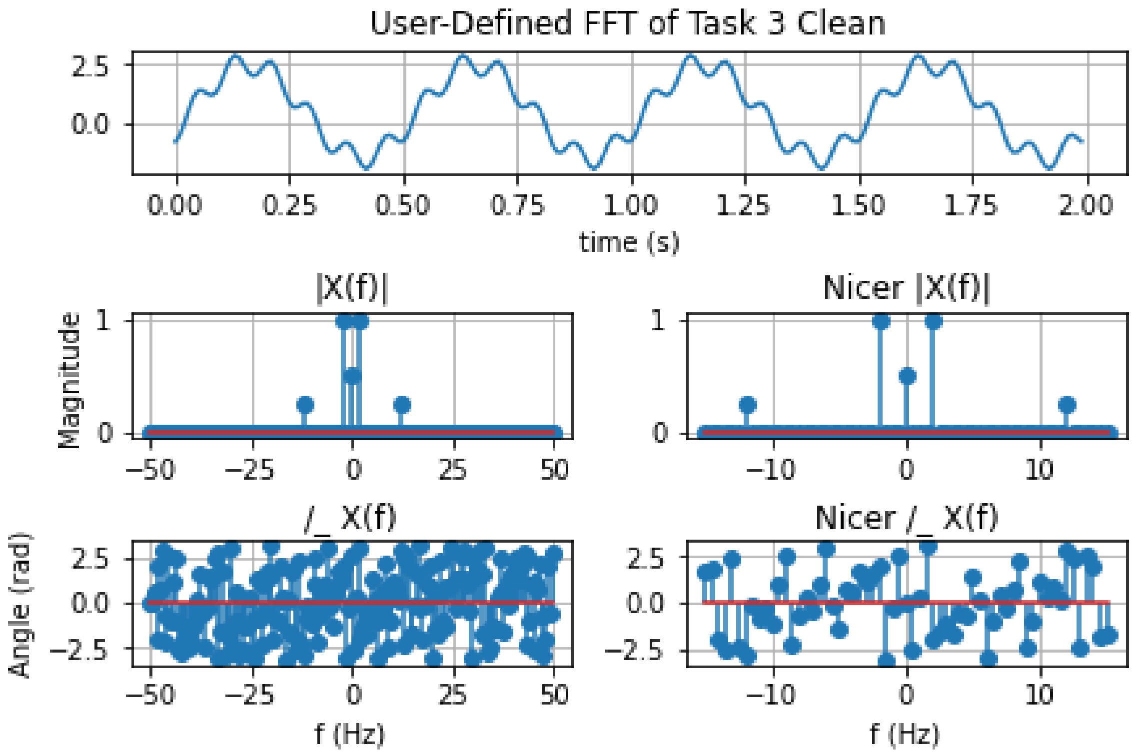
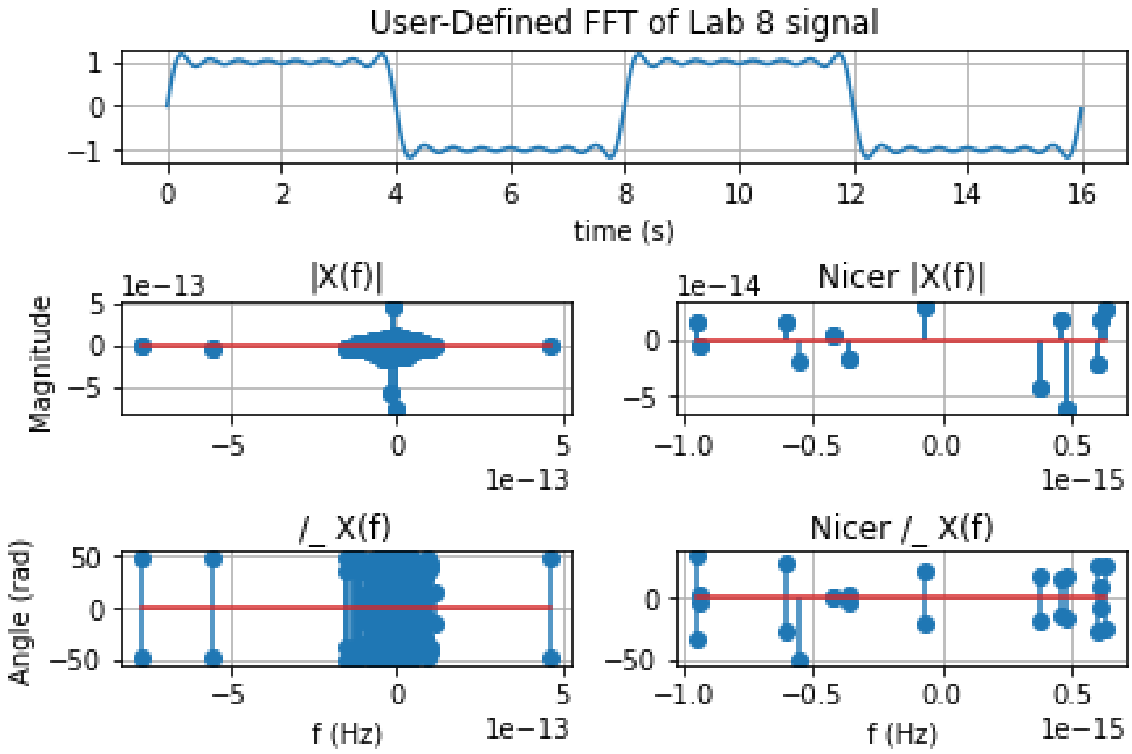


*Figure 7*


User-Defined FFT of Lab 8 signal
time (s)
|X(f)|
Nicer |X(f)|
Magnitude
1e−13
1e−14
1e−15
/_ X(f)
Nicer /_ X(f)
Angle (rad)
f (Hz)

# 3 Question

## 3.1 Question 1

What happens if fs is lower? If it is higher? fs in your report must span a few orders of magnitude.

fs directly correlates to the resolution of the plots. Having a higher fs increases the resolution, and having a lower fs has a worse resolution.

## 3.2 Question 2

What difference does eliminating the small phase magnitudes make?

Doing so decreases the amount of noise in the plots.

## 3.3 Question 3

Verify your results from Tasks 1 and 2 using the Fourier transforms of cosine and sine. Explain why your results are correct. You will need the transforms in terms of Hz, not rad/s. For example, the Fourier transform of cosine (in Hz) is:

$$\mathscr{F}\{cos(2\pi f_0 t\} = \frac{1}{2}[\delta(f - f_0) + \delta(f + f_0)]$$

The Fourier transform for $cos(2\pi t)$ is:

$$\mathscr{F}\{cos(2\pi t\} = \frac{1}{2}[\delta(\omega - \omega_0) + \delta(\omega + \omega_0)]$$

The Fourier transform for $5sin(2\pi t)$ is:

$$\mathscr{F}\{5sin(2\pi t\} = \frac{5j}{2}[\delta(\omega - \omega_0) + \delta(\omega + \omega_0)]$$

These would generate a spike roughly every $n \cdot \omega$ frequency. This correlates with what's seen on the plots.