



Electrical Principles

Student Name: Joshua Smith

Student ID: SMI16836261

Module Code:

Lecture: Dr Rabya Kadhim

Table Of Contents

Table Of Contents	2
Fourier's Analysis	3
What is fourier's analysis	3
Types of Waveforms	3
How To Identify The Waveforms By Shape	3
Fourier's Transform vs Fourier's Series	4
What is fourier's transform ?	4
What is Fourier's Series ?	4
What is the difference ?	4
Why is Fourier Series Useful For Analyzing Electrical Waveforms ?	5
Some Key points	5
Harmonics	6
How Harmonics Relate to The Fourier Series	6
Harmonics in Power Systems and Their Importance	6
Generating and Plotting Rectangular and Sawtooth Waves Using MATLAB	7
Generating Rectangular Waves	7
Analysis	7
Generating Sawtooth Waveforms	8
Analysis	8
Task 3	9
Part A	9
Part B	11
Part C	13
Task 4	14
Part A	14
Part B	15
Part C	16
Part D	17

Fourier's Analysis

What is fourier's analysis

A mathematical technique called Fourier analysis looks for patterns or cycles in a time series data set that has already undergone normalisation. By breaking down complicated or noisy data into a sequence of exponential or trigonometric functions, like sine waves, it specifically aims to simplify the data. The observed data might be reconstructed by adding the sine waves back together, each of which would have a distinct cycle duration, amplitude, and phase relationship with the others.

(Hayes, 2023)

Types of Waveforms

Waveforms can be roughly classified as either periodic or aperiodic. The first class can be separated into simple and complex waveforms and appears to have a pitch connected with them. Impulsive waves, which happen only once, and noise waves, which are random but continuous waveforms, are two subtypes of the aperiodic class.

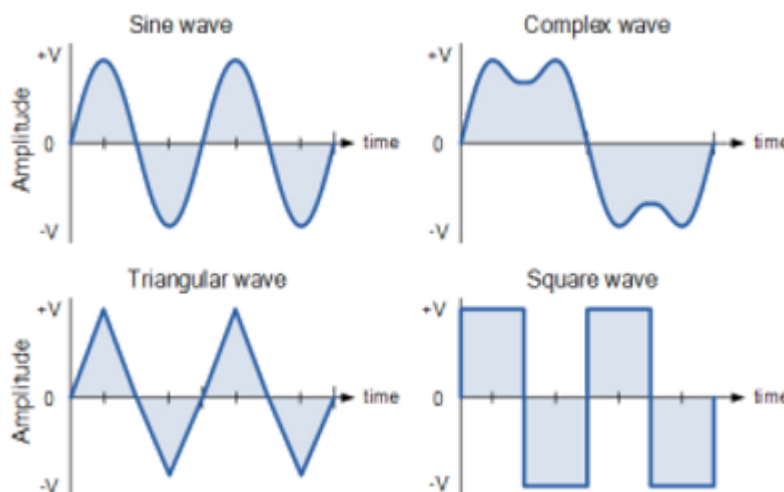
(UCL, 2003)

How To Identify The Waveforms By Shape

The most recognisable waveforms are sine waves, which have a perfect, pure signal. The tuning fork is actually the nearest source of pure waves.

When applied to sound, complex waves, which are made up of two or more sine waves, have the effect of reducing sound quality, albeit the human ear frequently isn't able to perceive this.

Most string instruments produce triangle waves, which are more sharper, as a result of friction changes that occur when the string is moved, whether by plucking or bowing.



(Worthington, 2023)

Fourier's Transform vs Fourier's Series

What is fourier's transform ?

A mathematical procedure known as the Fourier transform establishes the connection between a single's frequency domain and time domain representations. It breaks down a function or signal into oscillatory functions. No information is created or lost during the Fourier transform as we can retrieve the original signal from its transformation.

Differential equations are frequently solved using the Fourier transform. Mathematical characteristics of the Fourier transform include linearity, scaling, duality, convolution, modulation, conjugation, and more. It is employed in various forms of spectroscopy, including nuclear magnetic resonance.

What is Fourier's Series ?

Fourier series is the mathematical process of breaking down a periodic signal into a sum of sines and cosines. The orthogonal relationship between sine and cosine functions is used in Fourier series. A periodic function can be divided into the sum of simple terms using the Fourier series, which can then be utilised to solve a particular problem.

Originally created to solve heat equations, the Fourier series is now used in many different domains, such as electronics, electrical, signal processing, quantum physics, image processing, and more. Harmonic analysis, the study of Fourier series, is a popular method for examining periodic functions.

What is the difference ?

	Fourier Transform	Fourier series
Definition	A mathematical process called the Fourier Transform is used to change a signal's time domain into its frequency domain	. A periodic signal can be broken down into a sum of sine and cosine components using the Fourier series technique.
Type of signal	Applied to only periodic signals	Can be applied to both periodic and aperiodic signals
Application	Fourier series are used in signal processing, electronics, electrical, and quantum mechanics.	Nuclear magnetic resonance and other forms of spectroscopy use the Fourier transform.

(Panigrahi, 2022)

Why is Fourier Series Useful For Analyzing Electrical Waveforms ?

A Fourier series is a useful tool for understanding and designing electrical circuits and signal processing systems because it makes it possible to deconstruct complex periodic signals into simpler, easier-to-understand components, such as sine waves, which reveal the various frequencies that contribute to the waveform's overall shape.

Some Key points

- Decomposing Complex signals - It makes it possible to investigate the distinct elements of a complex signal by representing any periodic waveform as a sum of sine and cosine waves at various frequencies (harmonics).
- Frequency domain analysis - Engineers can develop filters, amplifiers, and other signal processing systems by understanding how various frequencies contribute to the overall waveform through the analysis of the amplitudes and phases of these harmonics.
- Harmonic distortion - Power quality analysis and comprehension of non-linear circuit behaviour depend on the ability to quantify harmonic distortion in a signal using Fourier series

(Mayoral et al., 2017)

Harmonics

How Harmonics Relate to The Fourier Series

Since the Fourier series enables us to express any periodic waveform as a sum of sine and cosine functions at various frequencies, harmonics and the series are closely connected concepts. These frequencies are whole number multiples of the signal's fundamental frequency. Harmonics are integer multiples of the fundamental frequency, which is typically 50 Hz (in the UK) in a power system (100 Hz, 150 Hz, etc.).

Harmonics in Power Systems and Their Importance

Non-linear loads in power systems, such as fluorescent lights, computer power supply, and variable frequency drives, produce harmonics. These non-linear devices produce current waveforms with a mixture of sinusoidal components at various frequencies rather than pure sine waves. The harmonics are these elements.

For example, the following harmonics could exist in a 50 Hz power system:

50 Hz is the fundamental, followed by 100 Hz for the second harmonic, 150 Hz for the third, and so forth.

Power systems can detect these harmonics, which can skew the waveforms of voltage and current, by using the Fourier series.

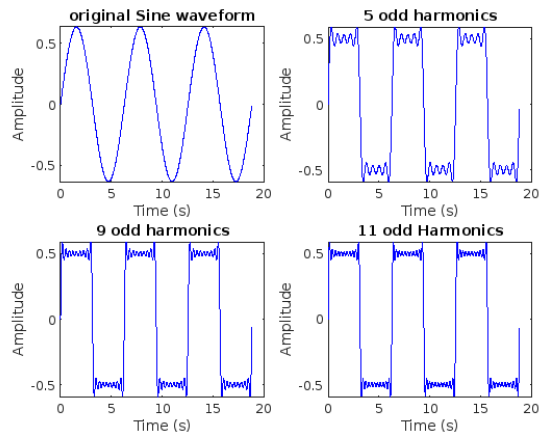
Below are some listed reasons of harmonics importance in power systems;

- Signal Distortion - Pure sinusoidal waveforms, which are suitable for AC systems, are distorted by harmonics. Numerous problems, including equipment failure, overheating, and inefficient power distribution, can result from this distortion. When exposed to distorted waveforms, equipment including motors, transformers, and circuit breakers could not operate as efficiently.
- Power Quality - Harmonics have the potential to deteriorate power quality, leading to voltage swings, decreased effectiveness, and possible malfunctions in delicate equipment. In extreme situations, it may result in resonance occurrences that could harm parts.
- Equipment Losses - Because harmonics contain higher-frequency components of the current, they cause additional heating in transformers, motors, and cables. *Electrical systems become less efficient as a result of increased power losses.*

(Dsp illustrations, 2022)

Generating and Plotting Rectangular and Sawtooth Waves Using MATLAB

Generating Rectangular Waves



Analysis

Adding more harmonics improves the approximation of the desired waveform, the more odd harmonics are added in the more defined the sharp edges become. Also the rectangular waveform has sharp discontinuities.

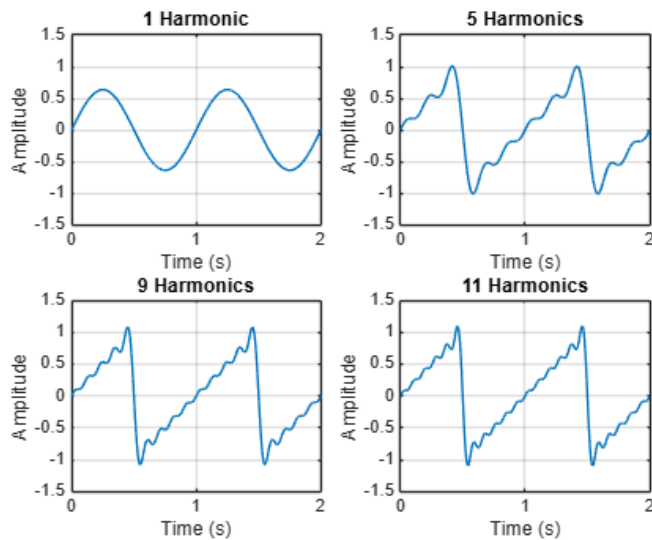
```
clc
% Sine waveform
w=1;
a0=0;
n=1;
t=0:0.01:6*pi;
% Harmonic 1
y=a0+(2/(n*pi))*sin(w*t*n);
subplot(2,2,1)
plot(t,y,'b');
title('original Sine waveform');
ylabel('Amplitude')
xlabel('Time (s)')
```

```
% 5 odd harmonics
w=1;
a0=0;
n=1;
t=0:0.01:6*pi;
y1=a0+(2/(n*pi))*sin(w*t*n);
n=3;
y3=a0+(2/(n*pi))*sin(w*t*n);
n=5;
y5=a0+(2/(n*pi))*sin(w*t*n);
n=7;
y7=a0+(2/(n*pi))*sin(w*t*n);
n=9;
y9=a0+(2/(n*pi))*sin(w*t*n);
z=(y1+y3+y5+y7+y9);
subplot(2,2,2);
plot(t,z,'b');
title('5 odd harmonics');
ylabel('Amplitude')
xlabel('Time (s)')
```

```
% 9 odd harmonics
w=1;
a0=0;
n=1;
t=0:0.01:6*pi;
y1=a0+(2/(n*pi))*sin(w*t*n);
n=3;
y3=a0+(2/(n*pi))*sin(w*t*n);
n=5;
y5=a0+(2/(n*pi))*sin(w*t*n);
n=7;
y7=a0+(2/(n*pi))*sin(w*t*n);
n=9;
y9=a0+(2/(n*pi))*sin(w*t*n);
n=11;
y11=a0+(2/(n*pi))*sin(w*t*n);
n=13;
y13=a0+(2/(n*pi))*sin(w*t*n);
n=15;
y15=a0+(2/(n*pi))*sin(w*t*n);
n=17;
y17=a0+(2/(n*pi))*sin(w*t*n);
z=(y1+y3+y5+y7+y9+y11+y13+y15+y17);
subplot(2,2,3);
plot(t,z,'b');
title('9 odd harmonics');
ylabel('Amplitude')
xlabel('Time (s)')
```

```
% 11 odd harmonics
w=1;
a0=0;
n=1;
t=0:0.01:6*pi;
y1=a0+(2/(n*pi))*sin(w*t*n);
n=3;
y3=a0+(2/(n*pi))*sin(w*t*n);
n=5;
y5=a0+(2/(n*pi))*sin(w*t*n);
n=7;
y7=a0+(2/(n*pi))*sin(w*t*n);
n=9;
y9=a0+(2/(n*pi))*sin(w*t*n);
n=11;
y11=a0+(2/(n*pi))*sin(w*t*n);
n=13;
y13=a0+(2/(n*pi))*sin(w*t*n);
n=15;
y15=a0+(2/(n*pi))*sin(w*t*n);
n=17;
y17=a0+(2/(n*pi))*sin(w*t*n);
n=19;
y19=a0+(2/(n*pi))*sin(w*t*n);
n=21;
y21=a0+(2/(n*pi))*sin(w*t*n);
z=(y1+y3+y5+y7+y9+y11+y13+y15+y17+y19+y21);
subplot(2,2,4);
plot(t,z,'b');
title('11 odd Harmonics');
ylabel('Amplitude')
xlabel('Time (s)')
```

Generating Sawtooth Waveforms



Analysis

As both odd and even harmonics are added the linear slope becomes more accurate, the slope of the sawtooth waveform will remain continuous as harmonics are added

```
fs = 1000;
T = 2;
t = linspace(0, T, T * fs);
f = 1;

harmonic_counts = [1, 5, 9, 11];

waveforms = zeros(length(harmonic_counts), length(t));

for idx = 1:length(harmonic_counts)
    harmonic_limit = harmonic_counts(idx);
    sawtooth_wave = zeros(1, length(t));

    for k = 1:harmonic_limit
        sawtooth_wave = sawtooth_wave + ((-1)^(k + 1) / k) * sin(2 * pi * k * f * t);
    end

    waveforms(idx, :) = (2/pi) * sawtooth_wave;
end

figure;
titles = ["1 Harmonic", "5 Harmonics", "9 Harmonics", "11 Harmonics"];
for i = 1:length(harmonic_counts)
    subplot(2, 2, i);
    plot(t, waveforms(i, :));
    title(titles(i));
    xlabel('Time (s)');
    ylabel('Amplitude');
    grid on;
    ylim([-1.5, 1.5]);
end
```

Task 3

Part A

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi f t} dt$$
$$x(t) = 1 \quad -2 \leq t \leq 2$$
$$X(f) = \int_{-2}^2 e^{-j2\pi f t} dt$$
$$\int e^{-j2\pi f t} dt = \frac{e^{-j2\pi f t}}{-j2\pi f}$$
$$X(f) = \left[\frac{e^{-j2\pi f t}}{-j2\pi f} \right]_{t=-2}^{t=2}$$

The first step in the equation is to use the fourier's transform and and substitute the values for $x(t)$ which are -2 and 2, once that step has been completed, The integral of 'e' needs to be evaluated and the limits then need to be applied outside the brackets

$$X(f) = \frac{e^{-j2\pi f(2)}}{-j2\pi f} - \frac{e^{-j2\pi f(-2)}}{-j2\pi f}$$
$$X(f) = \frac{e^{-j4\pi f}}{-j2\pi f} - \frac{e^{j4\pi f}}{-j2\pi f}$$
$$X(f) = \frac{1}{-j2\pi f} (e^{-j4\pi f} - e^{j4\pi f})$$

The next step is to substitute the limits '2 and -2' into the equation, then the equation can be simplified before factoring out $\frac{1}{-j2\pi f}$.

$$\text{Euler's formula} = e^{-j\theta} - e^{j\theta} = -2j \sin(\theta)$$
$$X(f) = \frac{1}{-j2\pi f} (-2j \sin(4\pi f))$$
$$X(f) = \frac{-2j}{-j2\pi f} \sin(4\pi f)$$
$$= \frac{2}{2\pi f} \sin(4\pi f)$$
$$= 4 \text{sinc}(4\pi f)$$

After that it can be simplified further using euler's formula, by first cancelling the 'j' terms and then simplifying the like terms which will simplify down to $4\text{sinc}(4\pi f)$

```

1  f = linspace(-5, 5, 1000);
2  X_f = 4 * sinc(4 * pi * f);
3
4  figure;
5  subplot(3, 1, 1);
6  plot(f, real(X_f));
7  title('Real part of X(jf)');
8  xlabel('Frequency (Hz)');
9  ylabel('Amplitude');
10
11
12  subplot(3, 1, 2);
13  plot(f, abs(X_f));
14  title('Amplitude Spectrum of X(jf)');
15  xlabel('Frequency (Hz)');
16  ylabel('Magnitude');
17
18
19  subplot(3, 1, 3);
20  plot(f, angle(X_f));
21  title('Phase Spectrum of X(jf)');
22  xlabel('Frequency (Hz)');
23  ylabel('Phase (radians)');
24

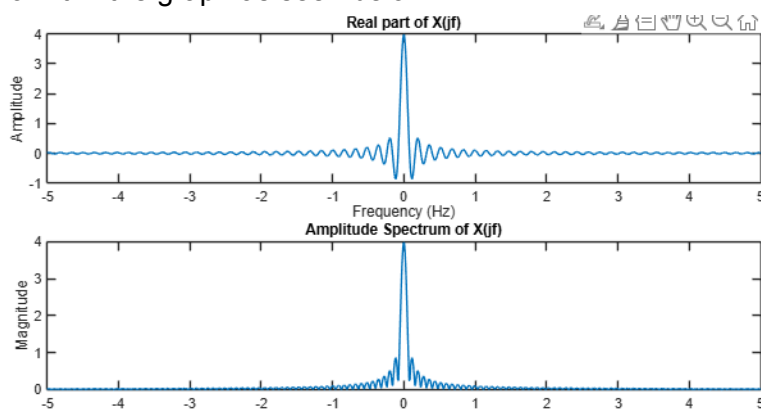
```

Now that the fourier transform of 'x(t)' have been derived, The value can be input into MATLAB in order to find the 'real part', the 'Amplitude spectrum', and the 'phase spectrum' of X(jf)

The MATLAB code works by using a frequency range using the variable "f" which represents the frequency axis the line of code 'linspace(-5, 5, 1000)' will allow MATLAB to generate 1000 equal spaces between -5 and 5Hz ensuring a smooth plots'.

The next line of code simply calculates the function $4\text{sinc}(4\pi f)$,

Below that are the subplots that help plot the real part and the amplitude spectrum. And show it in the graph as seen below.



Part B

Part (B)

$$Y(jf) = \int_{-\infty}^{\infty} y(t) e^{-j2\pi f t} dt$$

$$y(t) = e^{-7t} \quad t \geq 0 \quad t < 0$$

$$Y(jf) = \int_{-\infty}^0 e^{7t} e^{-j2\pi f t} dt + \int_0^{\infty} e^{-7t} e^{-j2\pi f t} dt$$

The first step in the equation is to use the Fourier's transform and substitute the values for 't' and the integral for 'y(t)', then the two integrals can be split into two parts.

$$\begin{aligned} \int_0^{\infty} e^{-7t} e^{-j2\pi f t} dt &= \int_0^{\infty} e^{-(7+j2\pi f)t} dt \\ \int_0^{\infty} e^{-at} dt &= \frac{1}{a} \quad \text{for } \Re(a) > 0 \\ \int_0^{\infty} e^{-(7+j2\pi f)t} dt &= \frac{1}{7+j2\pi f} \\ t < 0 \\ \int_{-\infty}^0 e^{7t} e^{-j2\pi f t} dt &= \int_{-\infty}^0 e^{(7-j2\pi f)t} dt \\ \int_{-\infty}^0 e^{(7-j2\pi f)t} dt &= \frac{1}{7-j2\pi f} \end{aligned}$$

Once it has been split into two parts each integral can be solved separately, this is done by factoring out the exponent and using the formula $\int e^{-at} dt = \frac{1}{a}$ for $\Re(a) \geq 0$. This is how both parts are calculated.

$$\begin{aligned} Y(jf) &= \frac{1}{7-j2\pi f} + \frac{1}{7+j2\pi f} \\ Y(jf) &= \frac{(7+j2\pi f) + (7-j2\pi f)}{(7-j2\pi f)(7+j2\pi f)} \\ \text{numerator} &= (7+j2\pi f) + (7-j2\pi f) = 14 \\ \text{denominator} &= (7-j2\pi f)(7+j2\pi f) = 7^2 + (2\pi f)^2 \\ &= 49 + (2\pi f)^2 \\ Y(jf) &= \frac{14}{49 + (2\pi f)^2} \end{aligned}$$

Now what's left is to add both of the integrals together, then find the common denominator then simplify it doing the same with the numerator once they have been simplified recombine the results to get the final Fourier transform

```

1  t = linspace(-10, 10, 1000);
2  f = linspace(-5, 5, 500);
3
4  Y_f = zeros(size(f));
5
6  for k = 1:length(f)
7      y_t = exp(-7 * abs(t));
8      exp_factor = exp(-1j * 2 * pi * f(k) * t);
9      Y_f(k) = trapz(t, y_t .* exp_factor);
10 end
11
12 figure;
13 plot(f, real(Y_f));
14 title('Real part of Y(jf) using Numerical Approximation');
15 xlabel('Frequency (Hz)');
16 ylabel('Amplitude');

```

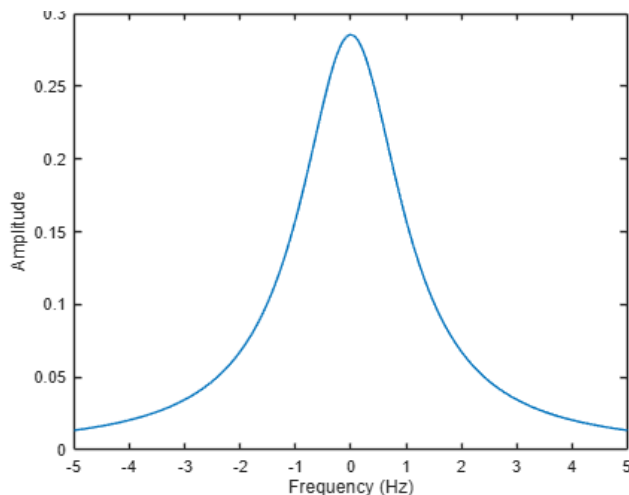
Now that the final fourier Transform of $y(t)$ has been calculated, the value can be input into MATLAB and use the TRAPZ feature which will approximate the fourier transform and help plot the real part, over a range of frequencies.

The code works by using a time and frequency range, the time vector used was -10, 10 with 1000 points, this insures that the resolution can provide an accurate integration and the frequency used was -5, 5hz with 500 points to do this the linspace line of code is used.

The next line of code ' $Y_f = \text{zeros}(\text{size}(f))$ ' creates 0's that are the same size as f in order to store the fourier transforms.

The next few lines of code is a loop which consists of three lines of code, the first line ' $y_t = \exp(-7 * \text{abs}(t))$,' will evaluate the time domain at each instant, the second line of code ' $\text{exp_factor} = \exp(-1j * 2 * \pi * f(k) * t)$,' computes the complex exponential of each frequency And the third line of code ' $Y_f(k) = \text{trapz}(t, y_t .* \text{exp_factor})$,' uses the trapezoidal rule to approximate the function by calculating the numerical integration.

All of this has been plotted in the graph below:

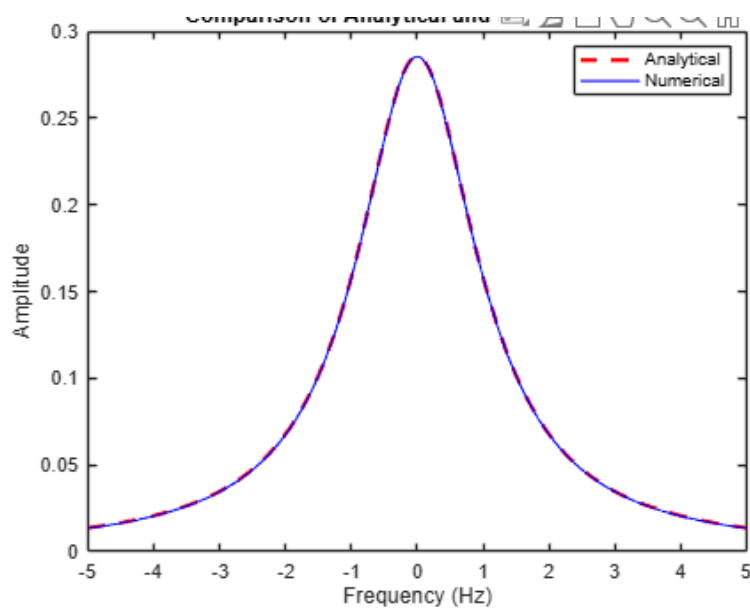


```

1  t = linspace(-10, 10, 1000);
2  f = linspace(-5, 5, 500);
3
4  Y_f_numerical = zeros(size(f));
5
6  for k = 1:length(f)
7      y_t = exp(-7 * abs(t));
8      exp_factor = exp(-1j * 2 * pi * f(k) * t);
9      Y_f_numerical(k) = trapz(t, y_t .* exp_factor);
10 end
11
12 Y_f_analytical = 14 ./ (49 + (2 * pi * f).^2);
13
14
15 figure;
16 plot(f, real(Y_f_analytical), 'r--', 'Linewidth', 2);
17 hold on;
18 plot(f, real(Y_f_numerical), 'b', 'Linewidth', 1);
19 title('Comparison of Analytical and Numerical Fourier Transform Y(jf)');
20 xlabel('Frequency (Hz)');
21 ylabel('Amplitude');
22 legend('Analytical', 'Numerical');
23 grid on;

```

In the plot section, the analytical and numerical comparison was accomplished using the red dotted line as the analytical and the blue solid line as the Numerical Fourier transform. As shown in the graph below.



Task 4

Part A

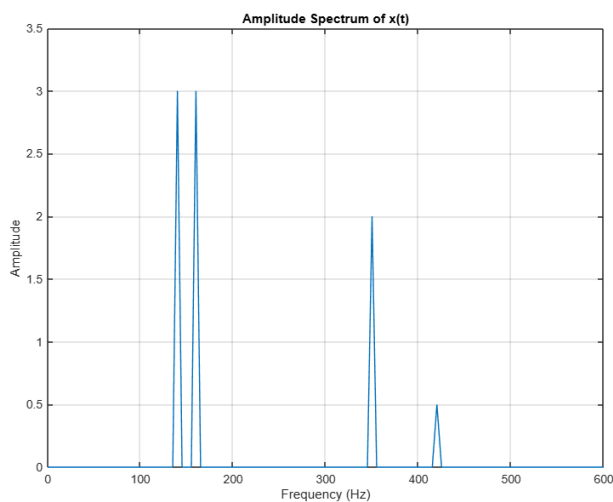
```
1 fs = 1/0.0002;  
2 T = 0.2;  
3 N = 1000;  
4 t = (0:N-1) * (1/fs);  
5  
6 x = 3*cos(2*pi*140*t) + 3*cos(2*pi*160*t) + 2*cos(2*pi*350*t) + 0.5*cos(2*pi*420*t);  
7  
8 X = fft(x);  
9  
10 f = (0:N/2-1) * (fs/N);  
11  
12 X_magnitude = abs(X(1:N/2))/N;  
13  
14 figure;  
15 plot(f, 2*X_magnitude);  
16 xlabel('Frequency (Hz)');  
17 ylabel('Amplitude');  
18 title('Amplitude Spectrum of x(t)');  
19 grid on;  
20 ylim([0 3.5]);  
21 xlim([0 600]);
```

The first line of code 'fs = 1/0.0002' is what defines the sampling frequency, the sampling frequency is 5000 which is sampled every 0.00002 seconds the next line T= 0.2 is the total duration which is 0.2 seconds 'N= 1000' is the number of samples and the line of code, 't = (0:N-1) * (1/fs)' is what creates the time vector.

The line of code 'x = 3*cos(2*pi*140*t) + 3*cos(2*pi*160*t) + 2*cos(2*pi*350*t) + 0.5*cos(2*pi*420*t);' This defines the original signal x(t), which is a sum of cosine waves at different frequencies.

The line of code 'X = fft(x)' this code is a fast fourier transform which converts the time-domain signal into frequency domain.

When this code is ran it will plot the amplitude spectrum which is shown in the graph below



Part B

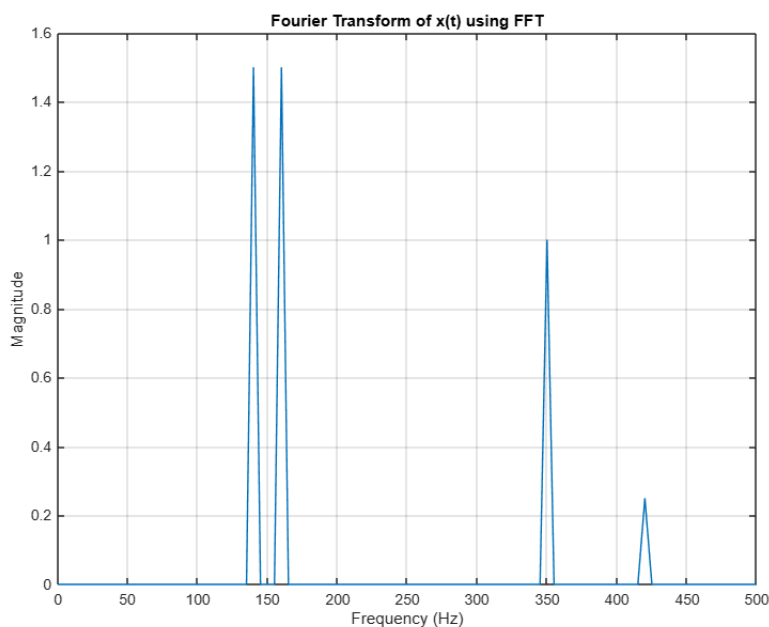
```
1 fs = 1/0.0002;
2 T = 0.2;
3 N = 1000;
4 t = (0:N-1) * (1/fs);
5
6 x = 3*cos(2*pi*140*t) + 3*cos(2*pi*160*t) + 2*cos(2*pi*350*t) + 0.5*cos(2*pi*420*t);
7 X = fft(x);
8
9 f = (0:N-1) * (fs/N);
10 X_magnitude = abs(X)/N;
11
12 figure;
13 plot(f, X_magnitude);
14 xlabel('Frequency (Hz)');
15 ylabel('Magnitude');
16 title('Fourier Transform of x(t) using FFT');
17 grid on;
18
19
20 xlim([0 500]);
21 ylim ([0 1.6]);
22
```

The first line of code 'fs = 1/0.0002' is what defines the sampling frequency, the sampling frequency is 5000 which is sampled every 0.00002 seconds the next line T= 0.2 is the total duration which is 0.2 seconds 'N= 1000' is the number of samples and the line of code, 't = (0:N-1) * (1/fs)' is what creates the time vector.

The line of code 'x = 3*cos(2*pi*140*t) + 3*cos(2*pi*160*t) + 2*cos(2*pi*350*t) + 0.5*cos(2*pi*420*t),' This defines the original signal x(t), which is a sum of cosine waves at different frequencies.

The line of code 'X = fft(x)' this code is a fast fourier transform which converts the time-domain signal into frequency domain.

When this code is ran it will plot the fourier Transform of x(t) which is shown in the graph below



Part C

```
fs = 1/0.0002;
T = 0.2;
N = 1000;
t = (0:N-1) * (1/fs);

x = 3*cos(2*pi*140*t) + 3*cos(2*pi*160*t) + 2*cos(2*pi*350*t) + 0.5*cos(2*pi*420*t);
X = fft(x);
x_recovered = ifft(X);

figure;
subplot(2,1,1);
plot(t, x);
title('Original Signal x(t)');
xlabel('Time (seconds)');
ylabel('Amplitude');
grid on;

subplot(2,1,2);
plot(t, real(x_recovered));
title('Recovered Signal from IFFT');
xlabel('Time (seconds)');
ylabel('Amplitude');
grid on;

error = norm(x - real(x_recovered));
disp(['Reconstruction Error (L2 norm): ', num2str(error)]);
```

The first line of code ' $fs = 1/0.0002$ ' is what defines the sampling frequency, the sampling frequency is 5000 which is sampled every 0.00002 seconds the next line $T = 0.2$ is the total duration which is 0.2 seconds ' $N = 1000$ ' is the number of samples and the line of code, ' $t = (0:N-1) * (1/fs)$ ' is what creates the time vector.

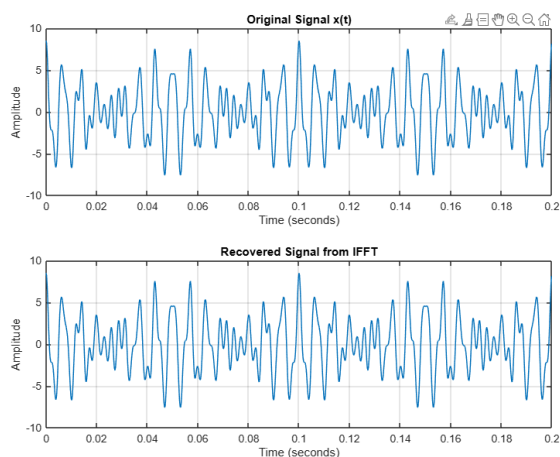
The line of code ' $x = 3*\cos(2*\pi*140*t) + 3*\cos(2*\pi*160*t) + 2*\cos(2*\pi*350*t) + 0.5*\cos(2*\pi*420*t)$,' This defines the original signal $x(t)$, which is a sum of cosine waves at different frequencies.

The line of code ' $X = \text{fft}(x)$ ' this code is a fast fourier transform which converts the time-domain signal into frequency domain.

When this code is ran it will plot the fourier Transform of $x(t)$ which is shown in the graph below

In order to plot the two graphs you need to have two subplots, the first will show the original time domain the second subplot shows the recovered signal,

The Error between the original waveform and the recovered waveform will be calculated and shown in the graphs below.



The original signal consists of multiple frequency components when the 'IFFT,' is applied it recovered the exact signal because all the Frequency components were captured correctly in the 'FFT' Process the original and recovered signal should match closely with only a small numerical difference between the two

Part D

```
1 fs = 1/0.0002;
2 T = 0.2;
3 N = 1000;
4 t = (0:N-1) * (1/fs);
5
6 x = 3*cos(2*pi*140*t) + 3*cos(2*pi*160*t) + 2*cos(2*pi*350*t) + 0.5*cos(2*pi*420*t);
7 X = fft(x);
8 X_filtered = X;
9 X_filtered(350:end-350+1) = 0;
10 x_recovered_low_freq = ifft(X_filtered);
11
12 figure;
13
14 subplot(2,1,1);
15 plot(t, x);
16 title('Original Signal x(t)');
17 xlabel('Time (seconds)');
18 ylabel('Amplitude');
19 grid on;
20
21
22 subplot(2,1,2);
23 plot(t, real(x_recovered_low_freq));
24 title('Recovered Signal (Low Frequencies Only)');
25 xlabel('Time (seconds)');
26 ylabel('Amplitude');
27 grid on;
28
29 error = norm(x - real(x_recovered_low_freq));
30 disp(['Reconstruction Error (L2 norm) with Low Frequencies: ', num2str(error)]);
```

The first line of code ' $fs = 1/0.0002$ ' is what defines the sampling frequency, the sampling frequency is 5000 which is sampled every 0.00002 seconds the next line $T = 0.2$ is the total duration which is 0.2 seconds ' $N = 1000$ ' is the number of samples and the line of code, ' $t = (0:N-1) * (1/fs)$ ' is what creates the time vector.

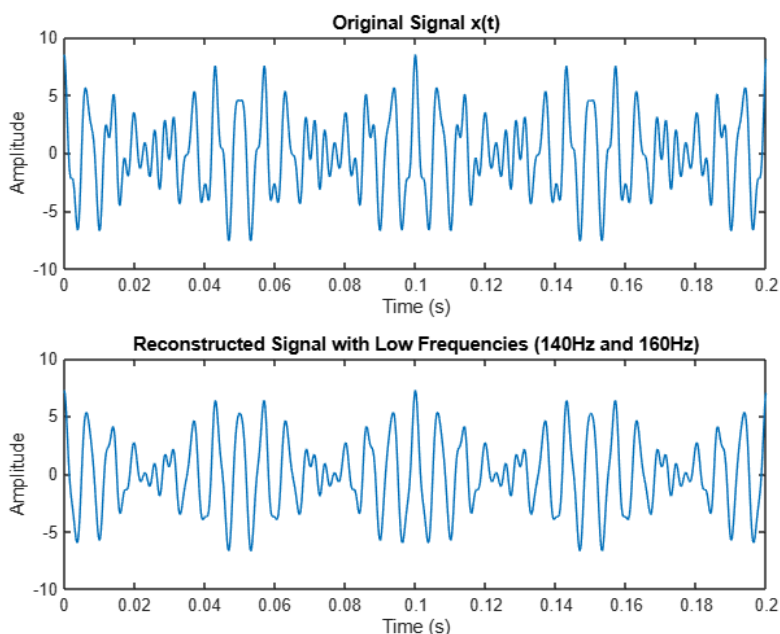
The line of code ' $x = 3*\cos(2*\pi*140*t) + 3*\cos(2*\pi*160*t) + 2*\cos(2*\pi*350*t) + 0.5*\cos(2*\pi*420*t)$,' This defines the original signal $x(t)$, which is a sum of cosine waves at different frequencies.

The line of code ' $X = \text{fft}(x)$ ' this code is a fast fourier transform which converts the time-domain signal into frequency domain.

When this code is ran it will plot the fourier Transform of $x(t)$ which is shown in the graph below

In order to plot the two graphs you need to have two subplots, the first will show the original time domain the second subplot shows the recovered signal,

The Error between the original waveform and the recovered waveform will be calculated and shown in the graphs below.



References

Dsp illustrations (2022) *The fourier series and harmonic approximation¶¶, Fourier Series and Harmonic Approximation - DSPIllustrations.com*. Available at: <https://dspillustrations.com/pages/posts/misc/fourier-series-and-harmonic-approximation.html> (Accessed: 18 March 2025).

Hayes, A. (no date) *Fourier analysis: What it means, how it works, Investopedia*. Available at: <https://www.investopedia.com/terms/f/fourieranalysis.asp> (Accessed: 18 February 2025).

Mayoral, E.H. et al. (2017) *Fourier analysis for harmonic signals in Electrical Power Systems, IntechOpen*. Available at: <https://www.intechopen.com/chapters/53524> (Accessed: 18 February 2025).

Panigrahi, K.K. (2022) *Difference between fourier series and Fourier transform, Tutorialspoint*. Available at: <https://www.tutorialspoint.com/difference-between-fourier-series-and-fourier-transform> (Accessed: 18 March 2025).

UCL (2003) *Lecture 2-4:Harmonic Analysis/Synthesis, Harmonic Analysis/Synthesis*. Available at: <https://www.phon.ucl.ac.uk/courses/spsci/acoustics/week2-4.pdf> (Accessed: 18 March 2025).

Worthington, J. (2023) *What are periodic waves?, iLearn Engineering®*. Available at: <https://www.ilearnengineering.com/electrical-and-electronic/what-are-periodic-waves#:~:text=Complex%20waves%20consist%20of%20two,plucking%20or%20using%20a%20bow> (Accessed: 18 March 2025).