جامعة الأمير مقرن بن عبد العزيز
University of Prince Mugrin

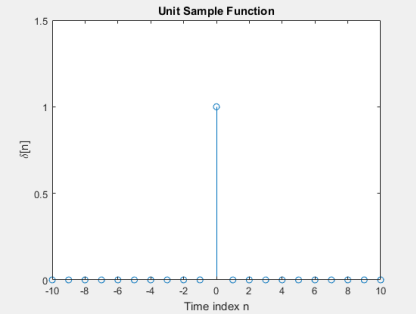
December 22, 2022
Signals & Systems Project

"Hamza Alashi – 4010339
Marwan Bitar – 4110259"
Dr. Doha Hamza

Task-1

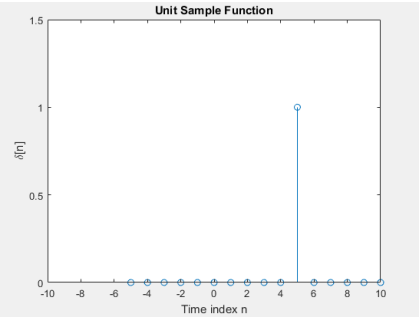
(a) Create an m-file with the following code and run it. (See page 1).

```
n = -10:10;
delta_n = [zeros(1,10) 1 zeros(1,10)];
stem(n,delta_n);
axis([-10 10 0 1.5]);
title('Unit Sample Function');
xlabel('Time index n');
ylabel('\delta[n]');
```



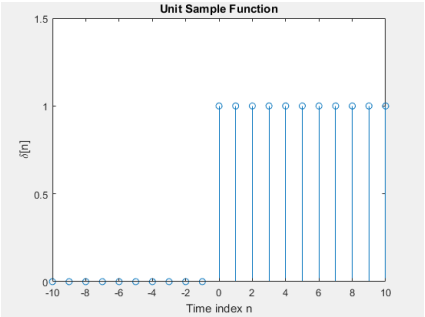
(b) Modify the code above to generate and plot δ[n – 5] for –10 ≤ n ≤ 10.

```
n = -10:10;
delta_n = [zeros(1,10) 1 zeros(1,10)];
stem((n+5),delta_n);
axis([-10 10 0 1.5]);
title('Unit Sample Function');
xlabel('Time index n');
ylabel('\delta[n]');
```



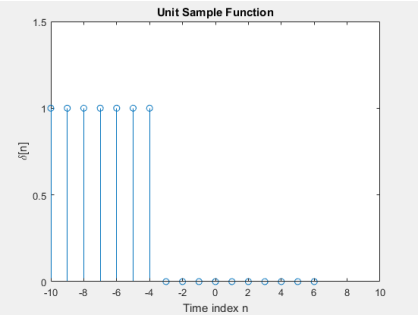
(c) Use the MATLAB functions ones and zeros to generate and plot the signal u[n] for –10 ≤ n ≤ 10.

```
n = -10:10;
delta_n = [zeros(1,10) 1 ones(1,10)]
stem(n,delta_n);
axis([-10 10 0 1.5]);
title('Unit Sample Function');
xlabel('Time index n');
ylabel('\delta[n]');
```



(d) Generate and plot u[–n – 4] for –10 ≤ n ≤ 10.

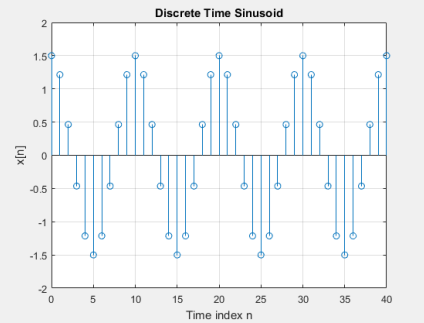
```
n = -10:10;
delta_n = [zeros(1,10) 1 ones(1,10)];
stem(-(n+4),delta_n);
axis([-10 10 0 1.5]);
title('Unit Sample Function');
xlabel('Time index n');
ylabel('\delta[n]');
```



Task-2

(a) Type in this code and run it.

```
n = 0:40;
w = 0.1*2*pi;
phi = 0;
A = 1.5;
xn = A * cos(w*n - phi);
stem(n,xn);
axis([0 40 -2 2]);
grid;
title('Discrete Time Sinusoid');
xlabel('Time index n');
ylabel('x[n]');
```



(b) What is the length of the signal x[n]?

"It is 41 samples"

(c) What is the fundamental period of x[n]?

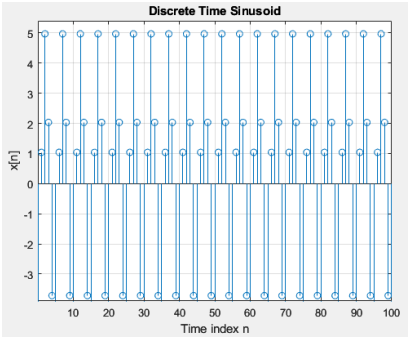
"N=10"

(d) What is the purpose of the grid command?

"To display or hide axes grid lines."

(e) Modify the code so that it generates a sinusoid with length 100, frequency 0.2×2π radians per sample, amplitude 5, and a phase offset of π/3 radians. Run the modified code to generate and plot another discrete-time sinusoid.

```
n = 0:100;
w = 0.2*2*pi;
phi = pi/3;
A = 5;
xn = A * sin(w*n - phi);
stem(n,xn);
axis([0 40 -2 2]);
grid;
title('Discrete Time Sinusoid');
xlabel('Time index n');
ylabel('x[n]');
```



Task-3

(a) Use MATLAB to generate and plot the discrete-time signal x[n] = sin(ω_on) for the following values of ω_o:

$$\frac{-29\pi}{8}, \frac{-3\pi}{8}, \frac{-\pi}{8}, \frac{\pi}{8}, \frac{3\pi}{8}, \frac{5\pi}{8}, \frac{7\pi}{8}, \frac{9\pi}{8}, \frac{13\pi}{8}, \frac{15\pi}{8}, \frac{33\pi}{8}, \text{and } \frac{21\pi}{8}.$$

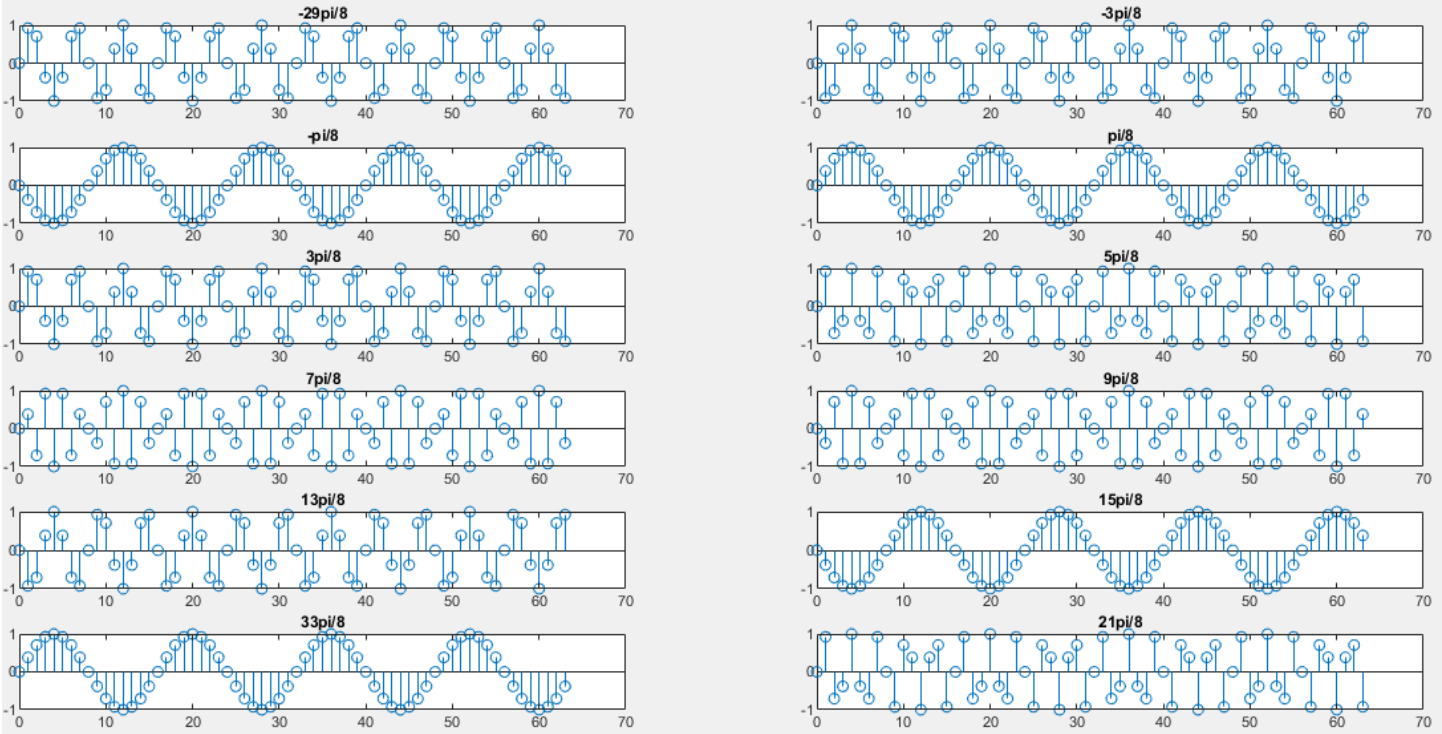
- Plot each signal for 0 ≤ n ≤ 63.
- Label each graph with the frequency.
- Use the subplot function to plot four graphs per figure.

%for w0 = -29pi/8
 	
n = 0:63;
k = -29;
w = k * pi/8;
xn = sin(w*n);
subplot(6,2,1);
stem(n,xn);
title('-29pi/8');
	%for w0 = 3pi/8
	
	n = 0:63;
	k = 3;
	w = k * pi/8;
	xn = sin(w*n);
	subplot(6,2,5);
	stem(n,xn);
	title('3pi/8');
		%for w0 = 13pi/8
		
		n = 0:63;
		k = 13;
		w = k * pi/8;
		xn = sin(w*n);
		subplot(6,2,9);
		stem(n,xn);
		title('13pi/8');

%for w0 = -3pi/8
 	
n = 0:63;
k = -3;
w = k * pi/8;
xn = sin(w*n);
subplot(6,2,2);
stem(n,xn);
title('-3pi/8');
	% for w0 = 5pi/8
	
	n = 0:63;
	k = 5;
	w = k * pi/8;
	xn = sin(w*n);
	subplot(6,2,6);
	stem(n,xn);
	title('5pi/8');
		%for w0 = 15pi/8
		
		n = 0:63;
		k = 15;
		w = k * pi/8;
		xn = sin(w*n);
		subplot(6,2,10);
		stem(n,xn);
		title('15pi/8');

%for w0 = -pi/8
 	
n = 0:63;
k = -1;
w = k * pi/8;
xn = sin(w*n);
subplot(6,2,3);
stem(n,xn);
title('-pi/8');
	%for w0 = 7pi/8
	
	n = 0:63;
	k = 7;
	w = k * pi/8;
	xn = sin(w*n);
	subplot(6,2,7);
	stem(n,xn);
	title('7pi/8');
		%for w0 = 33pi/8
		
		n = 0:63;
		k = 33;
		w = k * pi/8;
		xn = sin(w*n);
		subplot(6,2,11);
		stem(n,xn);
		title('33pi/8');

%for w0 = pi/8
 	
n = 0:63;
k = 1;
w = k * pi/8;
xn = sin(w*n);
subplot(6,2,4);
stem(n,xn);
title('pi/8');
	%for w0 = 9pi/8
	
	n = 0:63;
	k = 9;
	w = k * pi/8;
	xn = sin(w*n);
	subplot(6,2,8);
	stem(n,xn);
	title('9pi/8');
		%for w0 = 21pi/8
		
		n = 0:63;
		k = 21;
		w = k * pi/8;
		xn = sin(w*n);
		subplot(6,2,12);
		stem(n,xn);
		title('21pi/8');



The Plots for Part (a)

(b) Are any of the graphs from part (a) identical to one another? Explain.

Yes, there are some functions identical to others. The functions that have the following w_0 are equal:

- -29pi/8 & 3pi/8
- -pi/8 & 15pi/8
- pi/8 & 33pi/8
- -3pi/8 & 13pi/8
- 5pi/8 & 21pi/8

If we took the first two similar w0: we know all frequencies are divided by the same factor, so -29pi/8 is equal to some insignificant value with a reminder of -5pi/8, and this angle is equal to +3pi/8. The same reason applies to all the rest of the functions. Also, we do not forget that the sine function is odd.

(c) How are the graphs of $x[n] = \sin(\omega_0 n)$ for $\omega_0 = \frac{7\pi}{8}$ and $\omega_0 = \frac{9\pi}{8}$ related? Explain.

Each one looks like a reflected version of the other about the n-axis. This because the Sine function is an odd function, and 7pi/8 as an angle is equal to -pi/8, and 9pi/8 is equal to pi/8. So, sin(pi/8) = -sin(-pi/8).

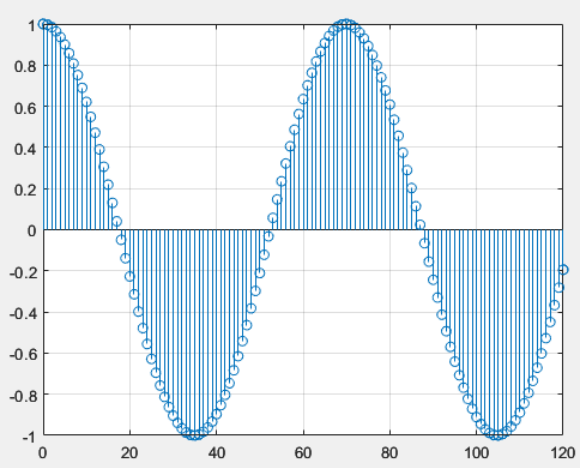
Task-4

(a) Use MATLAB to generate and plot the discrete-time signal

$$x[n] = \cos(0.09n)$$

for $0 \le n \le 120$.
For your plot, turn the grid on and scale the axes using the MATLAB statements
axis([0 120 -1.0 1.0]);
grid;

```
%P4
clc
clear
% Define the range of n values
n = 0:120;

% Generate the signal x[n]
xn = cos(0.09*n);

% Plot the signal x[n]
plot(n, xn);

% Turn on the grid and scale the axes
axis([0 120 -1.0 1.0]);
grid;
```



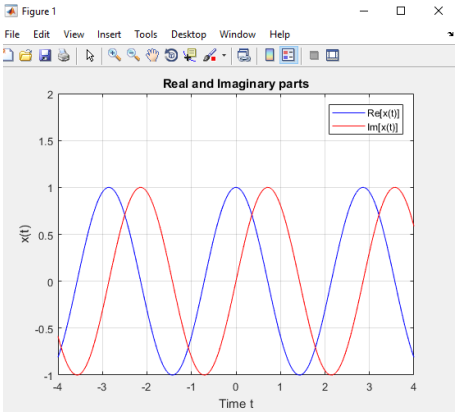
(b) Is this signal periodic? Explain.

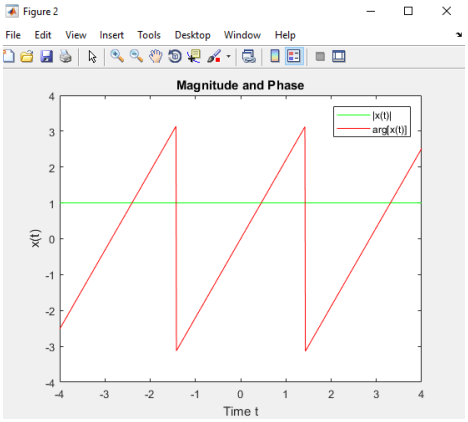
No, it is not. It needs to have same value of x[n] after N samples, while here, by inspection: When we look at the graph between n= 60 & n= 80, we do not get a value of x[n]= 1 as when x[0]=1.

Task-5

(a) Type in and run this code.

```
t = -4:0.01:4;
w = 2.2;
xt = exp(1i*w*t);
xtR = real(xt);
xtI = imag(xt);

figure(1); %  make Figure 1 active
plot(t,xtR,'-b'); % '-b' = means solid blue line
axis([-4 4 -1.0 2.0]);
grid;
hold on; % add more curves to the same graph
plot(t,xtI,'-r'); % 'r' = solid red
title('Real and Imaginary parts');
xlabel('Time t');
ylabel('x(t)');
legend('Re[x(t)]','Im[x(t)]');
hold off;
mag = abs(xt);
phase = angle(xt);

figure(2); % make Figure 2 active
plot(t,mag,'-g'); % '-' = solid line and 'g' = green grid
hold on; % add more curves to the graph
plot(t,phase,'-r');  % 'r' = red
title('Magnitude and Phase');
legend('|x(t)|','arg[x(t)]');
xlabel('Time t');
ylabel('x(t)');
hold off;
```



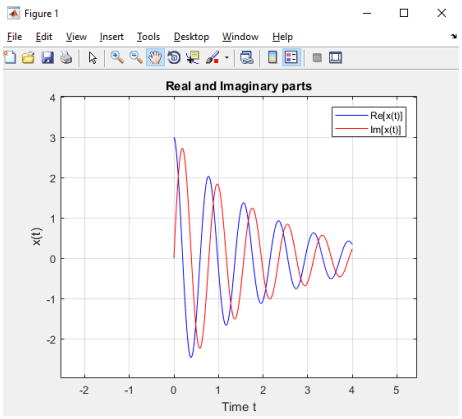


(b) Use similar MATLAB statements to generate the continuous time damped exponential signal

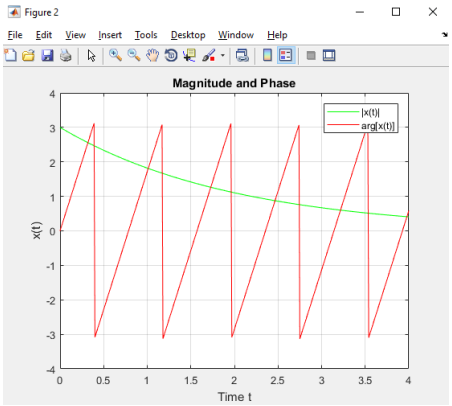
$$x(t) = 3e^{-t/2}e^{j8t}$$

for 0 ≤ t ≤ 4. Plot the real part, imaginary part, magnitude, and phase.

```
t = 0:0.01:4;
w = 8;
xt = 3*exp(-t/2).*exp(1i*w*t);
xtR = real(xt);
xtI = imag(xt);
figure(1); % make Figure 1 active
plot(t,xtR,'-b'); % '-b' means 'solid blue line'
axis([-4 4 -3.0 4.0]);
grid;
hold on; % add more curves to the same graph
plot(t,xtI,'-r'); % 'r' = red
title('Real and Imaginary parts');
xlabel('Time t');
ylabel('x(t)');
legend('Re[x(t)]','Im[x(t)]');
hold off;
mag = abs(xt);
phase = angle(xt);
```



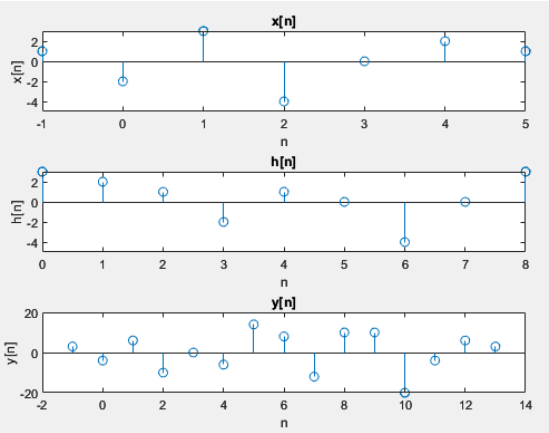
```
figure(2); % make Figure 2 active
plot(t,mag,'-g'); % '-' = solid line and 'g' = green
grid;
hold on; % add more curves to the graph
plot(t,phase,'-r'); % 'r' = red
title('Magnitude and Phase');
legend('|x(t)|','arg[x(t)]');
xlabel('Time t');
ylabel('x(t)');
hold off;
```



Task-6

(a) Type in this code into an m-file and run it.

```
% compute and plot a discrete convolution
h = [3 2 1 -2 1 0 -4 0 3]; % Impulse response
x = [1 -2 3 -4 0 2 1]; % Input signal
y = conv(x,h); % y[n] = x[n] * h[n]
subplot(3,1,1); % 3x1 array of graphs
stem(-1:5,x); % plot x[n]
title('x[n]');
xlabel('n');
ylabel('x[n]');
subplot(3,1,2); % make 2nd graph active
stem(0:8,h); % plot h[n]
title('h[n]');
xlabel('n');
ylabel('h[n]');
subplot(3,1,3); % make 3rd graph active
stem(-1:13,y); % plot y[n]
title('y[n]');
xlabel('n');
ylabel('y[n]');
```

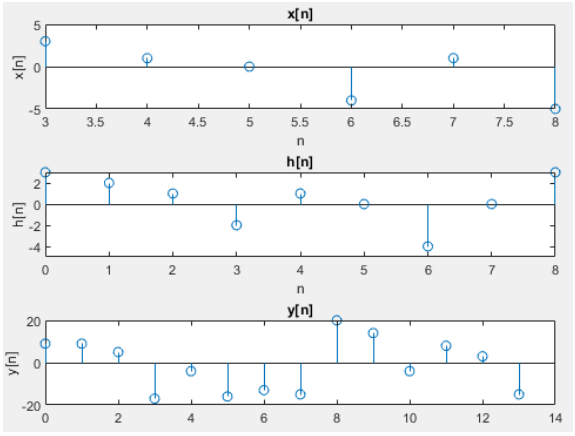


(b) Modify this code to compute and plot the convolution of h[n] with the new input signal

$$x[n] = 3\delta[n-3] + \delta[n-4] - 4\delta[n-6] + \delta[n-7] - 5\delta[n-8].$$

Make sure to briefly explain your calculations for the starting and stopping times of y[n].

```
% compute and plot a discrete convolution
h = [3 2 1 -2 1 0 -4 0 3]; % Impulse response
x = [3 1 0 -4 1 -5]; % Input signal
y = conv(x,h); % y[n] = x[n] * h[n]
subplot(3,1,1); % 3x1 array of graphs
stem(3:8,x); % plot x[n]
title('x[n]');
xlabel('n');
ylabel('x[n]');
subplot(3,1,2); % make 2nd graph active
stem(0:8,h); % plot h[n]
title('h[n]');
xlabel('n');
ylabel('h[n]');
subplot(3,1,3); % make 3rd graph active
stem(0:13,y); % plot y[n]
title('y[n]');
xlabel('n');
ylabel('y[n]');
```



The changes are highlighted.

Now: x[n] starts at n = 3 and ends at n = 8. So, the length of x[n] is 8 − 3 + 1 = 6. h[n] starts at n = 0 and ends at n = 8. So, the length of h[n] is 8 − 0 + 1 = 9. Therefore, conv will return the convolution result in an array of length 6 + 9 − 1 = 14. And y[n] starts from 0 because the convolution starts from there (the first sample is for h[n] at 0), and it ends at 13 to have a resultant of 14 samples.

Task-7

(a) Now we're going to use MATLAB to help us work out convolution using the Fourier transform. We are given an LTI system with impulse response h(t) = e −4tu(t) and input x(t) = te−2tu(t). We are asked to find the system output y(t).

```
%Using MATLAB to determine the convolution using the Fourier transform:
numer = [1]; %numerator coefficient
denom = [1 8 20 16]; %denominator coefficient
% r = represents the values of numerator , p =  represents the value of denominator
[r p k] = residue(numer,denom)
```

The results will be:

r =	p =	k =
0.2500 -0.2500 0.5000	-4.0000 -2.0000 -2.0000	[]

The partial fraction expansion for $Y(\omega)$:

$$Y(\omega) = \frac{\frac{1}{4}}{j\omega + 4} - \frac{\frac{1}{4}}{j\omega + 2} + \frac{\frac{1}{2}}{(j\omega + 2)^2}$$

From the Fourier Transform tables, it is then easy to write down the final expression for $y(t)$:

$$y(t) = \left[\frac{1}{4}e^{-4t} - \frac{1}{4}e^{-2t} + \frac{1}{2}te^{-2t}\right] u(t)$$

(b)

$$\text{(d)}\quad X(j\omega) = \frac{-(j\omega)^2 - 4j\omega - 6}{((j\omega)^2 + 3j\omega + 2)(j\omega + 4)}$$

Multiply the brackets in the denominator:

$$= \frac{-(j\omega)^2 - 4j\omega - 6}{(j\omega)^3 + 7j\omega^2 + 14j\omega + 8}$$

```
%Using MATLAB to determine the convolution using the Fourier transform:
numer = [-1 -4 -6]; %numerator coefficient
denom = [1 7 14 8]; %denominator coefficient
[r p k] = residue(numer,denom)
```

The results will be:

r =	p =	k =
-1.0000 1.0000 -1.0000	-4.0000 -2.0000 -1.0000	[]

Gives:

$$Y(w) = \frac{-1}{(jw+4)} + \frac{1}{jw+2} + \frac{-1}{(jw+1)}$$

And the final answer:

$$\boldsymbol{y(t) = (-e^{-4t} + e^{-2t} - e^{-t})u(t)}$$

Task-8

To determine the frequency response and impulse response for this exercise, we will use the syms function. The goal of employing syms in this exercise is to symbolically represent the variables.

$$x(t) = e^{-3t}u(t), \quad y(t) = e^{-3(t-2)}u(t-2)$$

These two equations will be represented by two variables, and we will apply the Fourier transform function to both equations. The frequency response will be (H(w) = Y(w) / X(w)). Additionally, the impulse response will then be determined using the invers Fourier transform.

```
% P8
%Using the Matlab Symbolic to Calculate the frequency response and impulse response:
clear
clc
syms w t
xt = exp(-3*t)*heaviside(t) ;
yt = exp(-3.*(t-2))* heaviside(t-2) ;
Xw = fourier(xt,t,w) ;
Yw = fourier(yt,t,w) ;
Hw = Yw / Xw %identify the frequency response. (Transfer function);
ht = ifourier(Hw,w,t)%the invers Fourier transform
```

The result will be:

```
Hw =

exp(-w*2i)


ans =

dirac(t - 2)
```

Task-9

(a) Find the Inverse Laplace transform of the following functions:

$$X1(s) = \frac{4s^2 + 8s + 10}{(s+2)(s^2+2s+5)} = \frac{4s^2 + 8s + 10}{s^3 + 4s^2 + 9s + 10}$$

$$X2(s) = \frac{3s^2 + 10s + 10}{(s+2)(s^2+6s+10)} = \frac{3s^2 + 10s + 10}{s^3 + 8s^2 + 22s + 20}$$

Then we take the coefficients of the <u>numerator</u> and <u>denominator</u>:

- ✓ **Numerator_coefficients = [4 8 10]**
- ✓ **Denominator_coefficients = [1 4 9 10]**

Here we are using residue method, we can write the partial fractional form of the transfer function:

$$X1(s) = \frac{1 + 0.5i}{s + 1 - 2i} + \frac{1 - 0.5i}{s + 1 + 2i} + \frac{2}{s+2}$$

$$X2(s) = \frac{1 + 3i}{s + 3 - i} + \frac{1 - 3i}{s + 3 + i} + \frac{1}{s+2}$$

We can manually use the following inverse Laplace transform table to get x1(t) & x2(t):

Pair Number	Nature of Roots	$F(s)$	$f(t)$
1	Distinct real	$\frac{K}{s+a}$	$Ke^{-at}u(t)$
2	Repeated real	$\frac{K}{(s+a)^2}$	$Kte^{-at}u(t)$
3	Distinct complex	$\frac{K}{s+\alpha-j\beta} + \frac{K^*}{s+\alpha+j\beta}$	$2\lvert K\rvert e^{-\alpha t}\cos(\beta t + \theta)u(t)$
4	Repeated complex	$\frac{K}{(s+\alpha-j\beta)^2} + \frac{K^*}{(s+\alpha+j\beta)^2}$	$2t\lvert K\rvert e^{-\alpha t}\cos(\beta t + \theta)u(t)$

$$\boldsymbol{x1(t) = \sqrt{5}e^{-t}cos(2t + 63°)u(t) + 2e^{-2t}u(t)}$$

$$\boldsymbol{x2(t) = 2\sqrt{10}e^{-3t}cos(t + 18°)u(t) + e^{-2t}u(t)}$$

The Code:

```
%Using the MATLAB to generate the transfer function:

% For X1(s):-
numer1 = [4 8 10]; % Coefficient of the numerator
denom1 = [1 4 9 10]; % Coefficient of the denominator
[r1,p1,k1] = residue(numer1,denom1)

% For X2(s):-
numer2 = [3 10 10]; % Coefficient of the numerator
denom2 = [1 8 22 20]; % Coefficient of the denominator
[r2,p2,k2] = residue(numer2,denom2)
```

The results will be:

r1 =	p1 =	k1 =
1.0000 + 0.5000i 1.0000 - 0.5000i 2.0000 + 0.0000i	-1.0000 + 2.0000i -1.0000 - 2.0000i -2.0000 + 0.0000i	[]
r2 =	p2 =	k2 =
1.0000 + 3.0000i 1.0000 - 3.0000i 1.0000 + 0.0000i	-3.0000 + 1.0000i -3.0000 - 1.0000i -2.0000 + 0.0000i	[]

Task-10

Get **the transfer function** and get **the poles and zeros** of:

(a)

$$\frac{d^2}{dt^2}y(t) + 5\frac{d}{dt}y(t) + 6y(t) = \frac{d^2}{dt^2}x(t) + 8\frac{d}{dt}x(t) + 13x(t)$$

1- Enter both x & y coefficients into MATLAB:

```
x_coeff = [ 1 8 13 ]
y_coeff = [ 1 5 6 ]
```

2- Use "tf" function to get the transfer function of part a:

```
H_a = tf(x_coeff, y_coeff)
```

We get:

```
Ha =

  s^2 + 8 s + 13
  --------------
  s^2 + 5 s + 6

Continuous-time transfer function.
```

3- Find poles and zeros, use "zero" for zeros, and use "pole" for poles:

```
z_a = zero(H_a)
p_a = pole(H_a)
```

We get:

```
z_a =

   -5.7321
   -2.2679


p_a =

   -3.0000
   -2.0000
```

And here is the hole code:

```
% P10
% part (a)
% 1-   Enter both x & y coefficients into MATLAB:
x_coeff = [ 1 8 13 ];
y_coeff = [ 1 5 6 ];
% 2-   Use "tf" function to get the transfer function of part a:
H_a = tf(x_coeff, y_coeff)
% 3-    Find poles and zeros, use "zero" for zeros, and use "pole" for poles:
z_a = zero(H_a)
p_a = pole(H_a)
```

For part (b):

(b)

$$\frac{d^2}{dt^2}y(t) - 2\frac{d}{dt}y(t) + 10y(t) = x(t) + 2\frac{d}{dt}x(t)$$

Using the same code:

```
% P10
% part (b)
% 1-   Enter both x2 & y2 coefficients into MATLAB:
x2_coeff = [ 2 1 ];
y2_coeff = [ 1 -2 10 ];
% 2-   Use "tf" function to get the transfer function of part b:
H_b = tf(x2_coeff, y2_coeff)
% 3-    Find poles and zeros, use "zero" for zeros, and use "pole" for poles:
z_b = zero(H_b)
p_b = pole(H_b)
```

We get:

```
H_b =

     2 s + 1
  --------------
  s^2 - 2 s + 10

Continuous-time transfer function.


z_b =

   -0.5000


p_b =

   1.0000 + 3.0000i
   1.0000 - 3.0000i
```

Task-11

Test your knowledge on the following. Find the differential equation corresponding to the following transfer functions:

(a)

$$H(s) = \frac{s^2 - 2}{s^3 - 3s + 1}$$

(b)

$$H(s) = \frac{2(s+1)(s-1)}{s(s+2)(s+1)} = \frac{2s^2 + 2}{s^3 + 3s^2 + 2s}$$

There were some errors using the provided method in the provided project tasks document. So, the doctor has told us to use residue function to get the following results:

the code:

```
% P11
% part (a)
clear
clc
syms s t

numerator = [1 0 -2];
denominator = [1 0 -3 1];
[r,p,k] = residue(numerator, denominator);
r1 = r(1,1);
r2 = r(2,1);
r3 = r(3,1);
p1 = p(1,1);
p2 = p(2,1);
p3 = p(3,1);
ph1 = ilaplace(r1/(s-p1));
ph2 = ilaplace(r2/(s-p2));
ph3 = ilaplace(r3/(s-p3));
h = ph1+ph2+ph3;
pretty(h)
```

The result: (this is h1(t))

```
   / 1724978734289935 t \                    /   8463998673628445 t \                    / 6256334945874825 t \
exp| ------------------ | 96742124222621   exp| - ------------------ | 7266637939040409   exp| ------------------ | 3208301388099961
   \  1125899906842624  /                     \    4503599627370496  /                     \  18014398509481984 /
-------------------------------------- + ---------------------------------------- + ----------------------------------------
            1125899906842624                          36028797018963968                          4503599627370496
```

The code for part b:

```
% part (b)
```

```
clear
syms s t
numerator2 = [2 0 -2];
denominator2 = [1 3 2 0];
[r2,p2,k2] = residue(numerator2, denominator2);
r1b = r2(1,1);
r2b = r2(2,1);
r3b = r2(3,1);
p1b = p2(1,1);
p2b = p2(2,1);
p3b = p2(3,1);
ph1b = ilaplace(r1b/(s-p1b));
ph2b = ilaplace(r2b/(s-p2b));
ph3b = ilaplace(r3b/(s-p3b));
hb = ph1b+ph2b+ph3b;
pretty(hb)
```

The result: (this is h2(t))

```
exp(-2 t) 3 + 3
```

- For getting the differential equation, we can get it manually by inspection (part a & b):

Part a:

$$\frac{d^3}{dt^3}y(t) - 3\frac{d}{dt}y(t) + y(t) = \frac{d^2}{dt^2}x(t) - 2x(t)$$

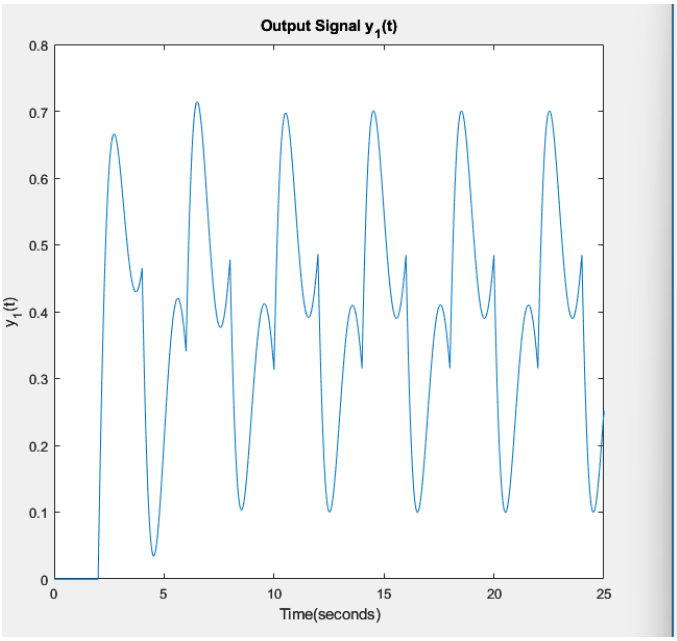
Part b:

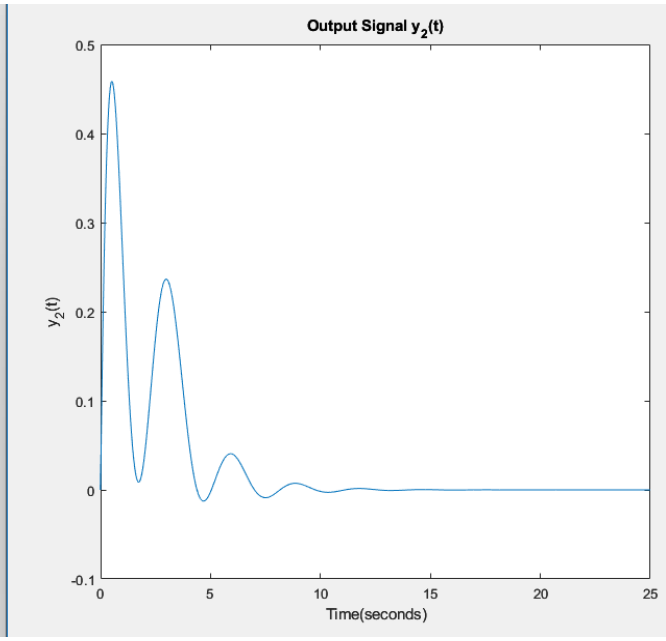
$$\frac{d^3}{dt^3}y(t) + 3\frac{d^2}{dt^2}y(t) + 2\frac{d}{dt}y(t) = 2\frac{d^2}{dt^2}x(t) + 2x(t)$$

Task-12

(a) Save this code as an m-file and run it.

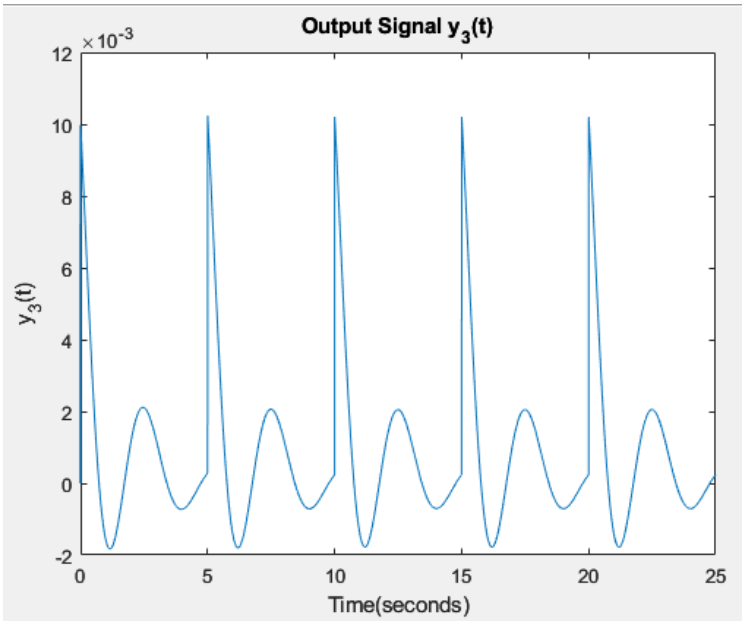
```
%-------------------------------------------------------------
% P12
%
% Use lsim to simulate an LTI system for two different input % signals.
%
close all; % close any open figure windows
clear
clc
% Make the system model
%
numer = [2 1 4];
denom = [1 2 6 5];
H = tf(numer,denom);
%
% Make input x1 a periodic square wave
%
[x1,t1] = gensig('square',4,25,0.01);
%
% Simulate system and plot the output signal
y1 = lsim(H,x1,t1);
figure(1);
plot(t1,y1);
title('Output Signal y_1(t)'); xlabel('Time(seconds)'); ylabel('y_1(t)'); %
% Make input x2 a one-sided decaying exponential
%
t2 = 0:0.01:25;
x2 = exp(-t2);
%
% Simulate system and plot the output signal
%
y2 = lsim(H,x2,t2);
figure(2);
plot(t2,y2);
title('Output Signal y_2(t)'); xlabel('Time(seconds)'); ylabel('y_2(t)');
```





(b) Modify the code to simulate the system and plot the output for a periodic pulse input signal x3(t) that starts at t = 0 sec, ends at t = 25 sec, has period 5 sec, and is sampled 200 times per second. Use the gensig command to make x3(t).

```
%-----------------------------------------------------------------
% P12
% part b
% Use lsim to simulate an LTI system for two different input % signals.
%
close all; % close any open figure windows
clear
clc
% Make the system model
%
numer = [2 1 4];
denom = [1 2 6 5];
H = tf(numer,denom);
%
% Make input x3 a periodic pulse
%
[x3,t3] = gensig('pulse', 5, 25, 0.0.005);
%
% Simulate system and plot the output signal
y3 = lsim(H,x3,t3);
figure(3);
plot(t3,y3);
title('Output Signal y_3(t)');
xlabel('Time(seconds)');
ylabel('y_3(t)');
```

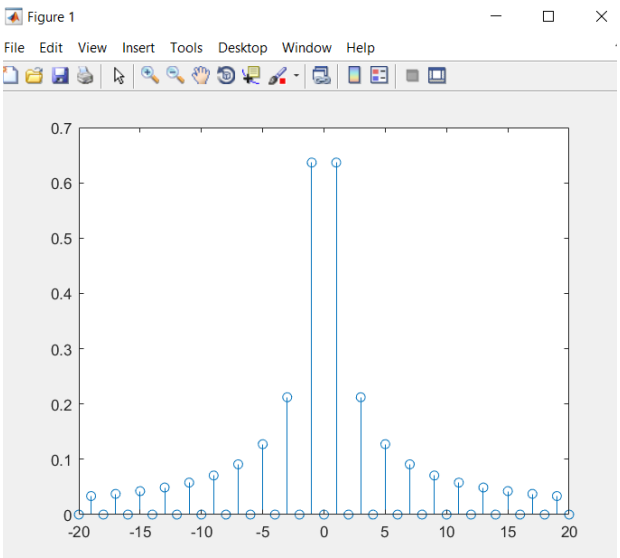


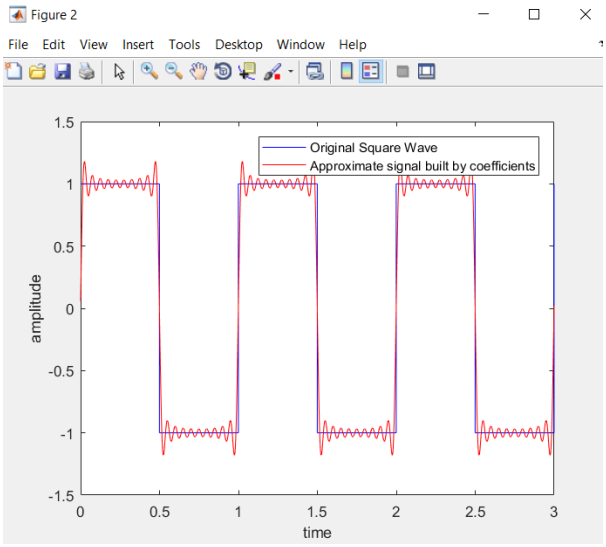
Task-13

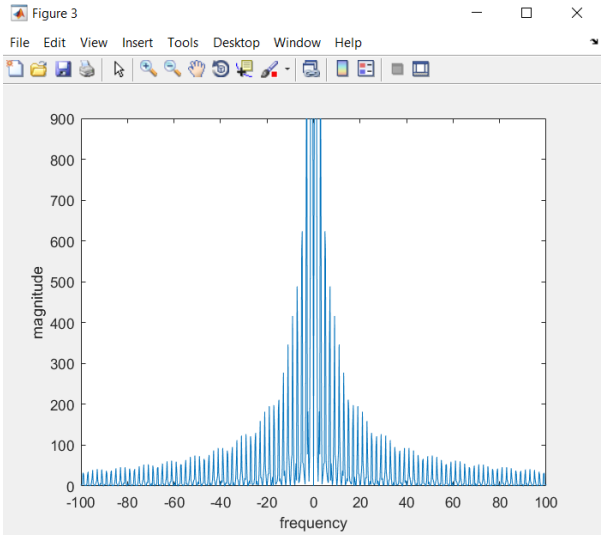
The square wave is analyzed for the Fourier Series coefficients using the following MATLAB script:

```
%generate periodic square wave.
period = 1;
ts = 0.0005; %sampling time.
[xt,t] = gensig('square', period, 3.5,ts);
xt = xt *2 - 1;
xt = xt(t>0.5);
t = t(t<t(end)-0.5) ;
plot(t,xt)
axis([0 3.5 0 1.2])

%calculate the coefficients.
idx = find (t < period); % we only need to evaluate the coefficients over 1 period.
teff = t(idx); %take the pertinent time instants
xteff = xt(idx).'; %take the pertinent signal component.
max_harmonics = min(20); %maximum number of harmonics considered, need to be bounded to less then
(1/ts -1 )because of discrete nature.
xx = 1:max_harmonics;
z = -flip(xx);
xx = [z 0 xx];
xxx = repmat(xx',1, length(teff));
yyy = repmat(teff',2*max_harmonics+1,1);
harmonics = xxx.*yyy;
harmonicss = exp(-1i*2*pi*harmonics);
xtefff = repmat(xteff,2*max_harmonics+1,1);

coo = xtefff.*harmonicss;
coeffs = ts*sum(coo, 2) ; %evaluate the coefficients theoretical Coeff
stem(xx, abs(coeffs));
hold on
%stem (xx, theoretical_coeff,'r')
hold off

%calculate the approximate square signal
coeffsrep = repmat(coeffs,1,length(t));
xxx = repmat(xx',1, length(t));
yyy = repmat(t',2*max_harmonics+1,1);
harmonics = xxx.*yyy;
harmonics_plus = exp(1i*2*pi*harmonics);
xt_approx = sum (coeffsrep.*harmonics_plus);
figure
plot(t,xt,'-b')
hold on
plot(t,xt_approx,'r')
legend('Original Square Wave', 'Approximate signal built by coefficients')
xlabel('time')
ylabel('amplitude')
%Plot the spectrum of the signal
Lfft=length(t); %defining DFT (or FFT) size
Lfft=2^ceil(log2(Lfft)) % making Lfft a power of 2 since this makes the fft algorithm work fast.
freqm=(-Lfft/2:Lfft/2-1)/(Lfft*ts) % Defining the frequency axis for the frequency domain DSB
modulated signal
XF=fftshift(fft(xt,Lfft)) % i.e. calculating the frequency domain picture of xt,
% fft algorithm calculates points from ) to Lfft-1, hence we use fftshift on this
% result to order samples from -Lfft/2 to Lfft/2 -1
Frange=[-100 100 0 900];
figure
plot(freqm,abs(XF))
xlabel('frequency')
ylabel('magnitude')
axis(Frange)
```

Run the above code in MATLAB after saving it as an m-file. Report your results:



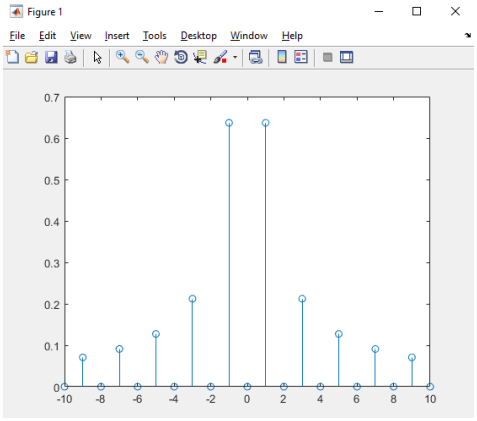


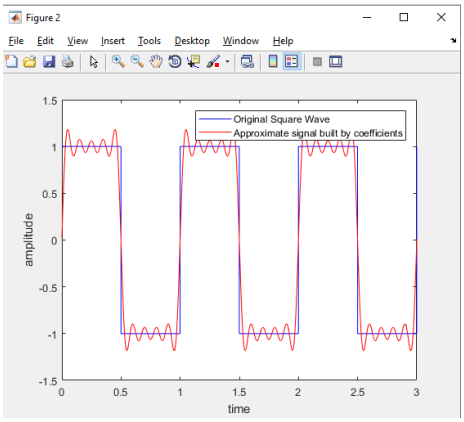


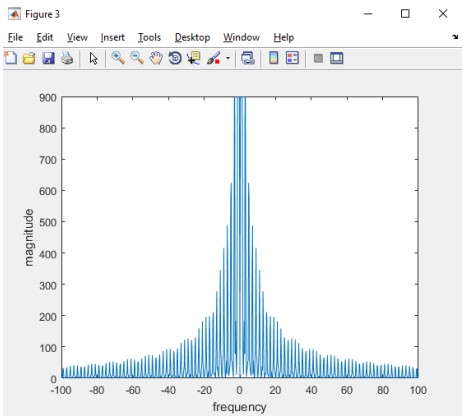
What happens when you change the max_harmonics number to 50 (it was 20), do the results improve?

Yes, it will be a better approximation of the original square wave signal, since it will have more terms and therefore more accurately capture the features of the original signal. However, increasing the number of harmonics also increases the computational cost of calculating the Fourier coefficients and evaluating the Fourier series.

What happens if you decrease the number of max_harmonics to 10? Report your results and comment on this.







Since it will have fewer terms and so capture less features of the original signal, it will be a less accurate approximation of the original square wave signal. However, fewer harmonics also results in a lower processing cost for determining the Fourier coefficients and analyzing the Fourier series.

What did you understand about the Fourier series from this experiment? How are the Fourier series coefficients related to the spectrum of x(t) obtained via the FFT operation in MATLAB?

The Fourier series is a way of representing a periodic function as a sum of sinusoidal functions. It can be used to approximate a given periodic signal by capturing its main frequency components.

By changing the number of harmonics used in the Fourier series, you can control the accuracy of the approximation and the computational cost of the calculation. Using a larger number of harmonics generally results in a better approximation of the original signal, but also increases the computational cost. Using a smaller number of harmonics reduces the computational cost but may result in a less accurate approximation of the original signal.

It's important to note that the Fourier series only works for periodic signals, and that the periodic signal must be exactly representable as a sum of sinusoidal functions (i.e., it must be a "nice" function) for the Fourier series to converge to the original signal. However, for many practical purposes, the Fourier series can be used to obtain a good approximation of a periodic signal even if the signal is not exactly representable as a sum of sinusoidal functions.

The Fourier coefficients are making an approximation function of x(t). So they are very helpful when we need to find an approximation to a function that we do not have its value, and all we have is just the Fourier coefficients of it.