Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Lab Report

Of

“Digital Signal Processing”

[Course Code: COMP 407]

Submitted by:

Ramraj Chimouriya (14)

Submitted to:

Mr. Satyendra Nath Lohani

Department of Computer Science and Engineering

17 November, 2022

Lab 1

Question:

To study important commands of MATLAB software
clc, close, xlabel, ylabel, zlabel, title, figure, subplot, linspace, stem, bar, plot, colon operator.

1. clc
   The function clc will allow you to clear the screen from within Octave programs.

2. close
   This function close the figure. Various arguments like figname, h etc. can be given to the function.

3. xlabel
   The function xlabel(string) specifies the string used to label the x-axis of the current axis.

4. ylabel
   The function ylabel(string) specifies the string used to label the y-axis of the current axis.

5. zlabel
   The function zlabel(string) specifies the string used to label the z-axis of the current axis.

6. title
   This function specifies the string used as a title for the current axis.

7. figure
   This function creates a new figure window for plotting.

8. subplot
   The function subplot(rows, cols, index) sets up a plot grid with rows by cols subwindows and set the current axes for plotting (gca) to the location given by index.

9. linspace
   The function linspace(start, end, n) returns a row vector with n linearly spaced elements between start and end.

10. stem
    This function plots a 2-D stem graph.

11. bar

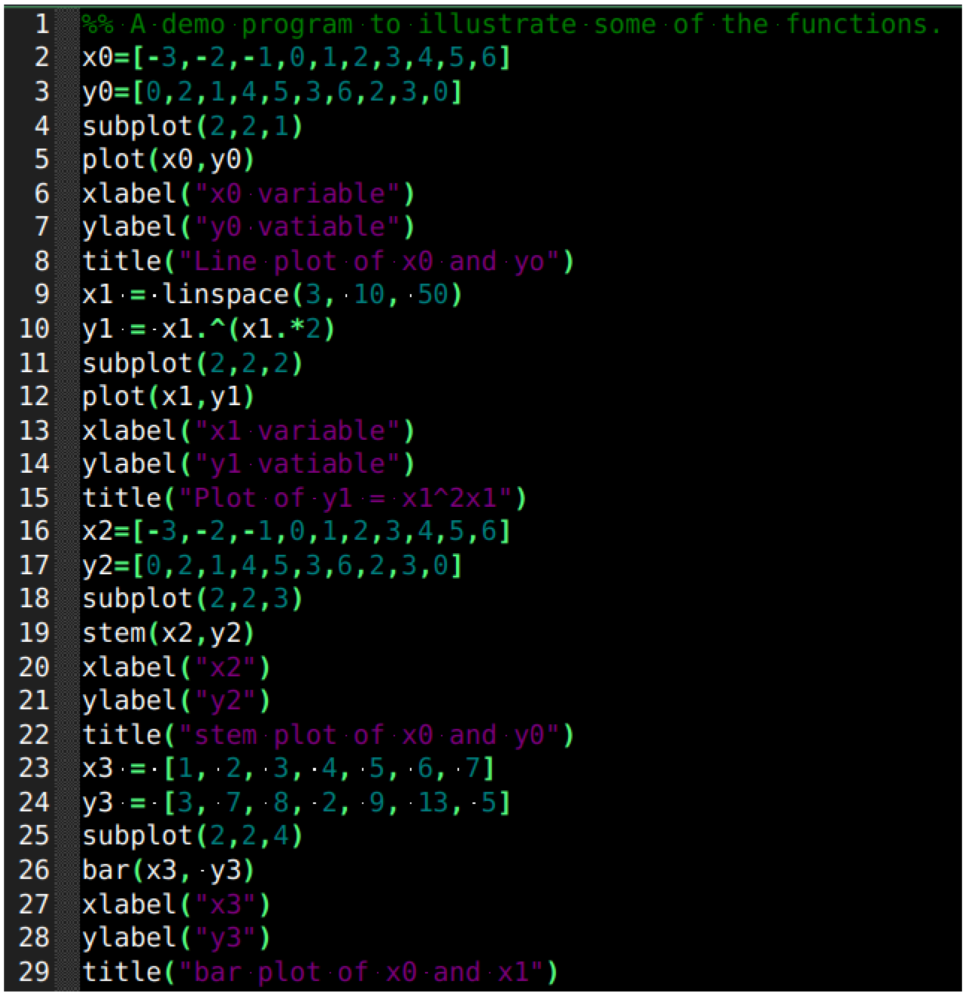
This function produces a bar graph from two vectors of X-Y data.

12. plot

This function produce 2-D plots. This function takes various arguments.

13. colon

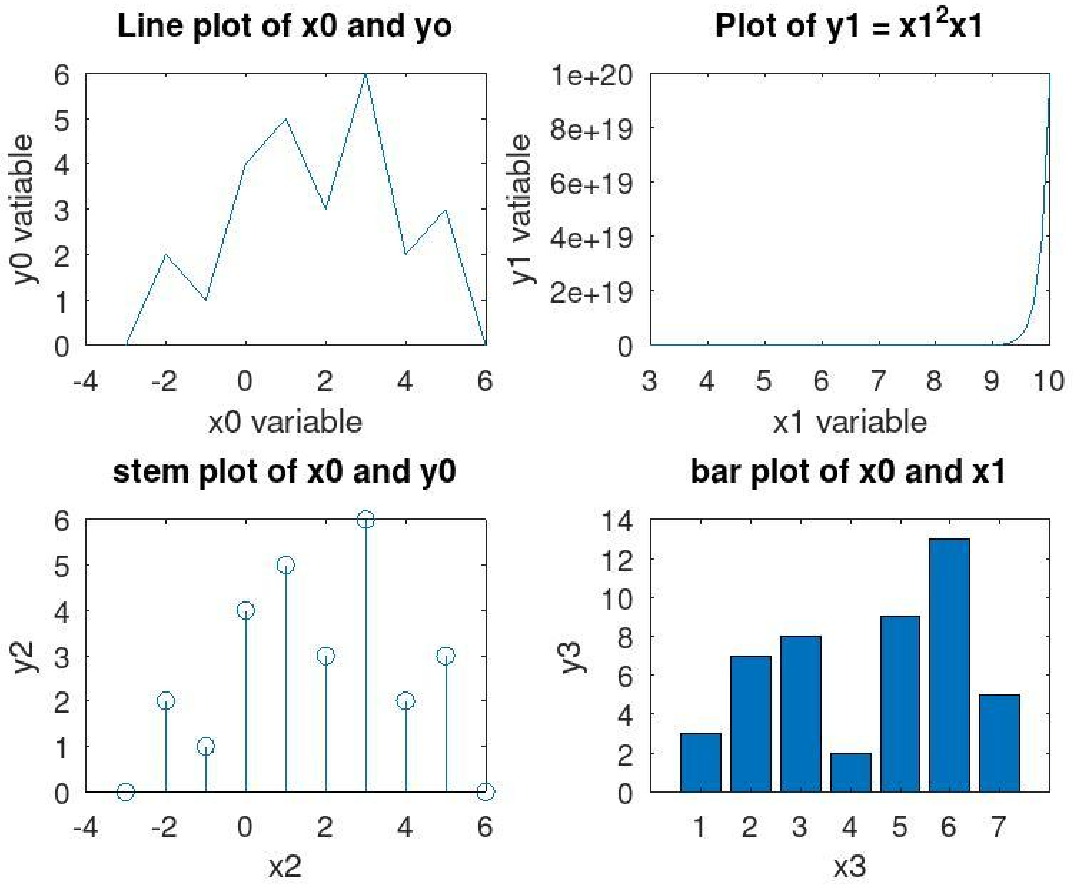
The function colon(base, limit, increment) returns the result of the colon expression corresponding to *base*, *limit*, and optionally, *increment*.



```
%% A demo program to illustrate some of the functions.
x0=[-3,-2,-1,0,1,2,3,4,5,6]
y0=[0,2,1,4,5,3,6,2,3,0]
subplot(2,2,1)
plot(x0,y0)
xlabel("x0 variable")
ylabel("y0 vatiable")
title("Line plot of x0 and yo")
x1 = linspace(3, 10, 50)
y1 = x1.^(x1.*2)
subplot(2,2,2)
plot(x1,y1)
xlabel("x1 variable")
ylabel("y1 vatiable")
title("Plot of y1 = x1^2x1")
x2=[-3,-2,-1,0,1,2,3,4,5,6]
y2=[0,2,1,4,5,3,6,2,3,0]
subplot(2,2,3)
stem(x2,y2)
xlabel("x2")
ylabel("y2")
title("stem plot of x0 and y0")
x3 = [1, 2, 3, 4, 5, 6, 7]
y3 = [3, 7, 8, 2, 9, 13, 5]
subplot(2,2,4)
bar(x3, y3)
xlabel("x3")
ylabel("y3")
title("bar plot of x0 and x1")
```

Figure 1: Demo program to illustrate some above-mentioned function

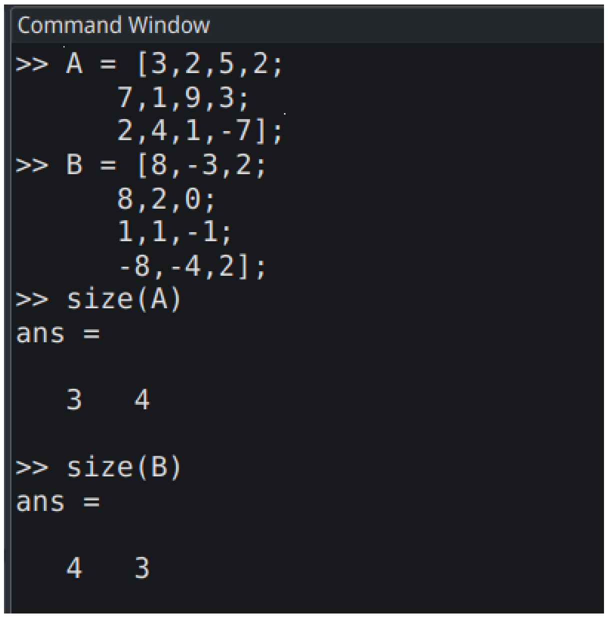
Figure 1 is the screen capture of the program that illustrates some of the described functions. Figure 2 is the output of the code.


Line plot of x0 and yo
y0 vatiable
x0 variable
Plot of $y1 = x1^2 x1$
y1 vatiable
x1 variable
1e+20
8e+19
6e+19
4e+19
2e+19
stem plot of x0 and y0
y2
x2
bar plot of x0 and x1
y3
x3

Question:

Familiarization with MATLAB environment.
Ans:

1. Create a matrix, A of size 3*4. Create another matrix, B of size 4*3.

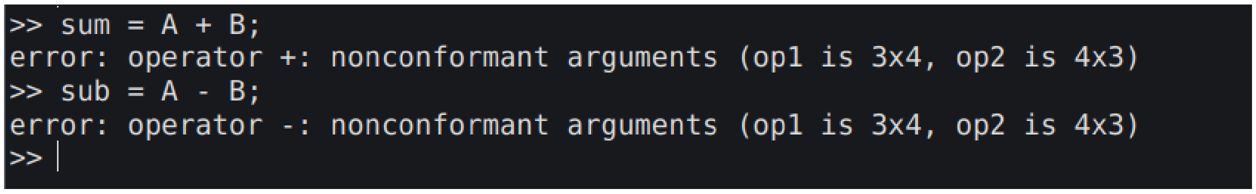


Command Window

```
>> A = [3,2,5,2;
      7,1,9,3;
      2,4,1,-7];
>> B = [8,-3,2;
      8,2,0;
      1,1,-1;
      -8,-4,2];
>> size(A)
ans =

   3   4

>> size(B)
ans =

   4   3
```

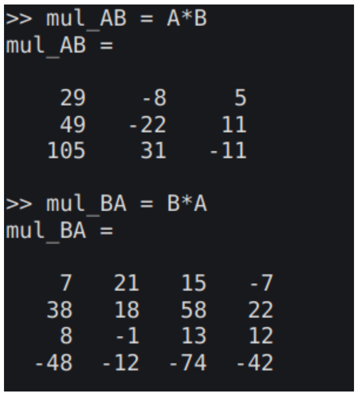
2. Add Matrix A and B. Subtract A from B.

While adding and subtracting A and B, since the sizes are different, error occurs.



```
>> sum = A + B;
error: operator +: nonconformant arguments (op1 is 3x4, op2 is 4x3)
>> sub = A - B;
error: operator -: nonconformant arguments (op1 is 3x4, op2 is 4x3)
>>
```

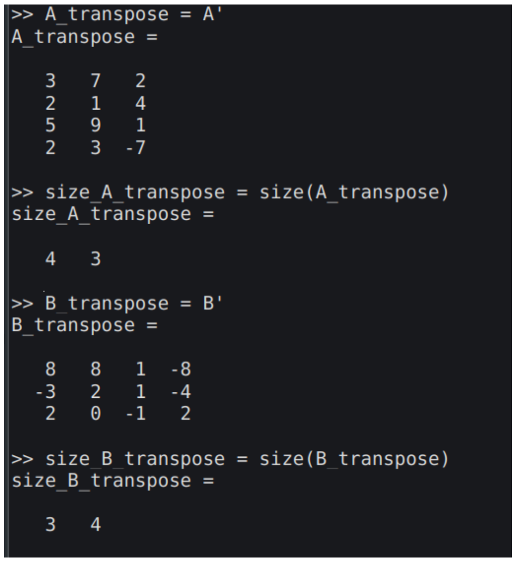
3. Multiply A and B. Multiply B and A [Errors, Reason ?].



```
>> mul_AB = A*B
mul_AB =

    29    -8     5
    49   -22    11
   105    31   -11

>> mul_BA = B*A
mul_BA =

    7   21   15   -7
   38   18   58   22
    8   -1   13   12
  -48  -12  -74  -42
```

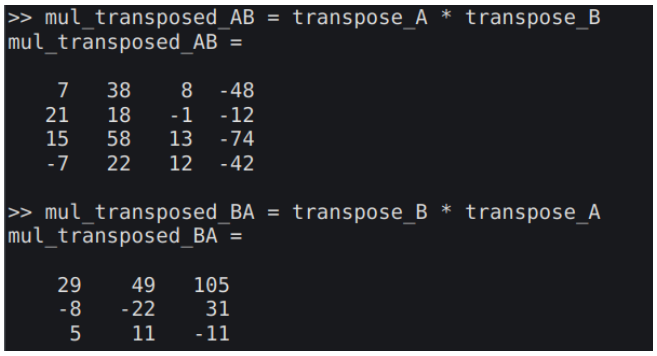
4. Transpose matrix A and B. Multiply the transposed matrices.

Transpose:



```
>> A_transpose = A'
A_transpose =

   3   7   2
   2   1   4
   5   9   1
   2   3  -7

>> size_A_transpose = size(A_transpose)
size_A_transpose =

   4   3

>> B_transpose = B'
B_transpose =

   8   8   1  -8
  -3   2   1  -4
   2   0  -1   2

>> size_B_transpose = size(B_transpose)
size_B_transpose =

   3   4
```

Multiply of transposed_matrices:

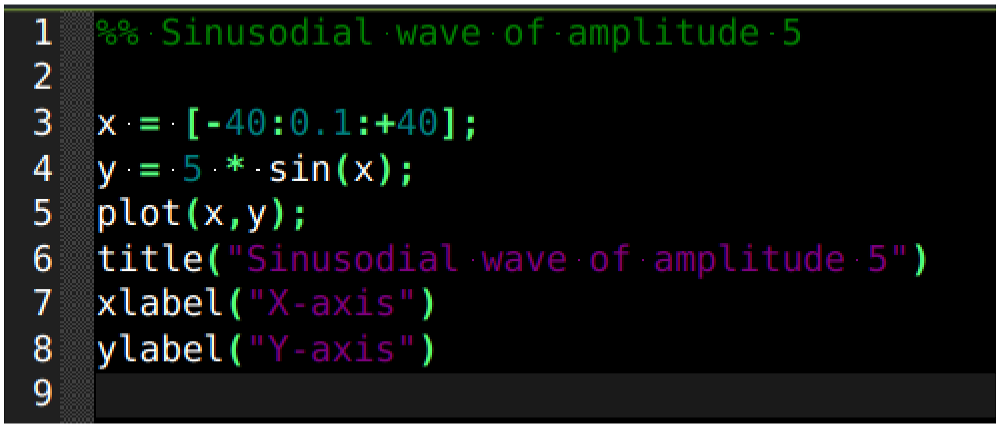


```
>> mul_transposed_AB = transpose_A * transpose_B
mul_transposed_AB =

    7   38    8  -48
   21   18   -1  -12
   15   58   13  -74
   -7   22   12  -42

>> mul_transposed_BA = transpose_B * transpose_A
mul_transposed_BA =

    29    49   105
    -8   -22    31
     5    11   -11
```

Lab 2

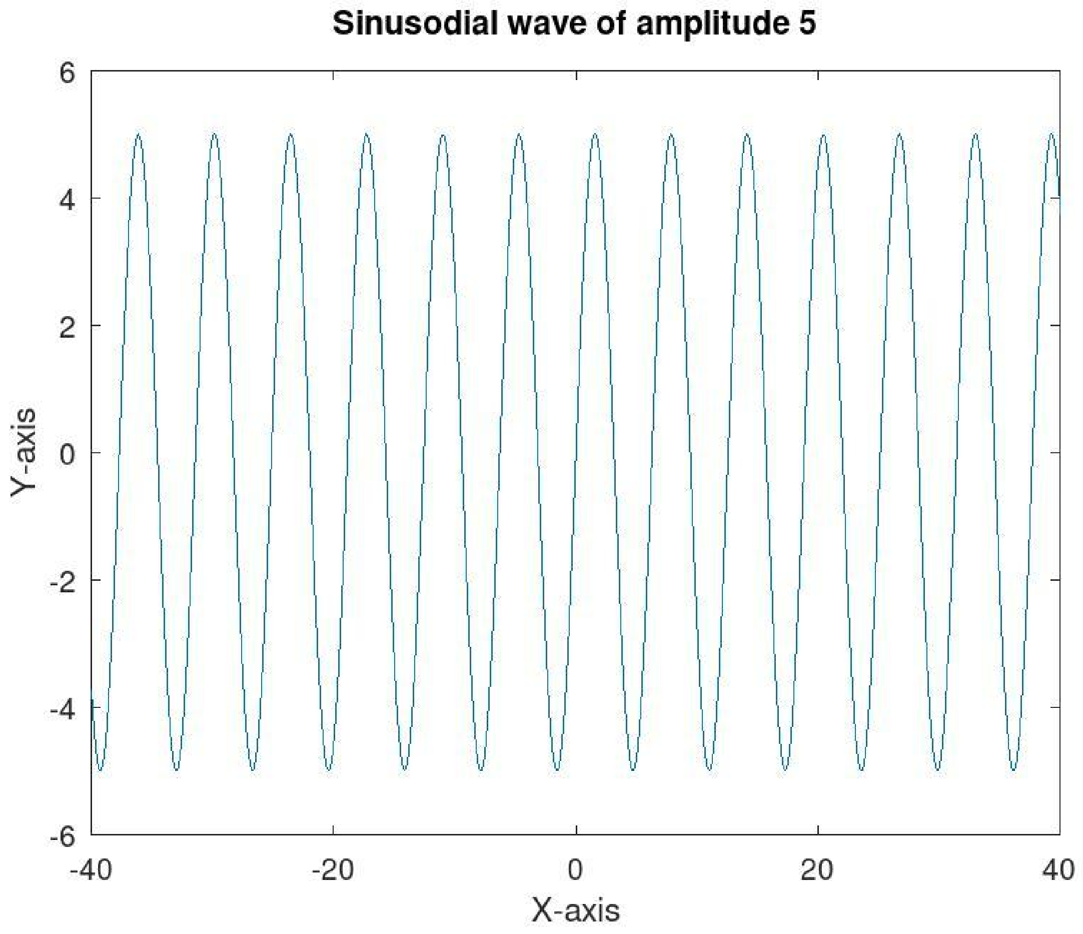
1. Generate a continuous time sinusoidal wave of amplitude 5.

Source code:



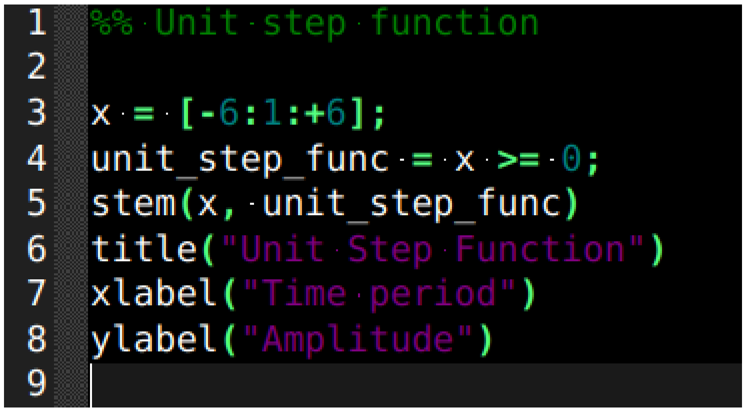
```
%% Sinusodial wave of amplitude 5

x = [-40:0.1:+40];
y = 5 * sin(x);
plot(x,y);
title("Sinusodial wave of amplitude 5")
xlabel("X-axis")
ylabel("Y-axis")
```

Output:



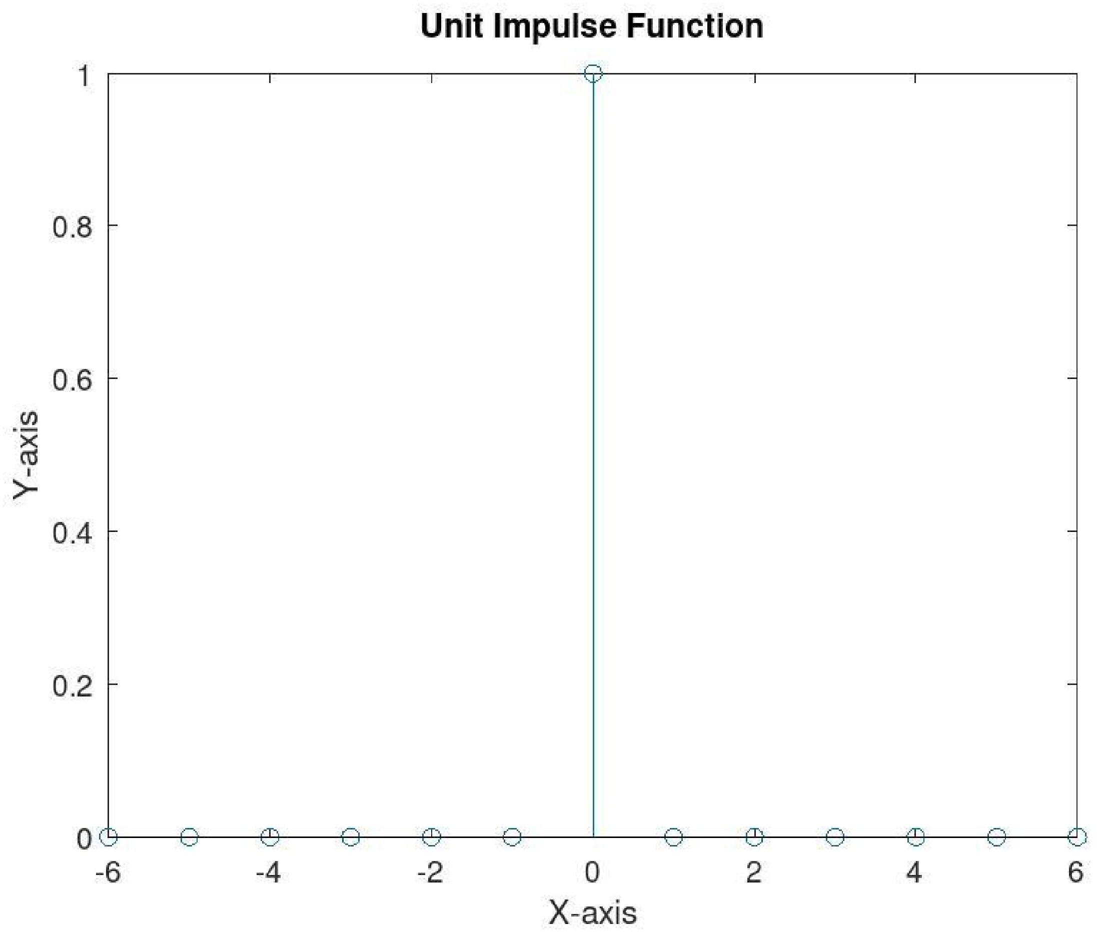
2. Generate a unit impulse function.

Source Code:



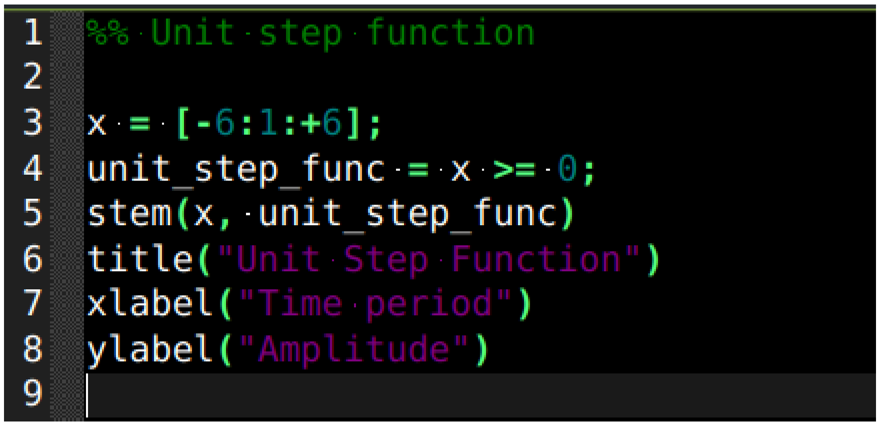
```
%% Unit step function

x = [-6:1:+6];
unit_step_func = x >= 0;
stem(x, unit_step_func)
title("Unit Step Function")
xlabel("Time period")
ylabel("Amplitude")

```

Output:



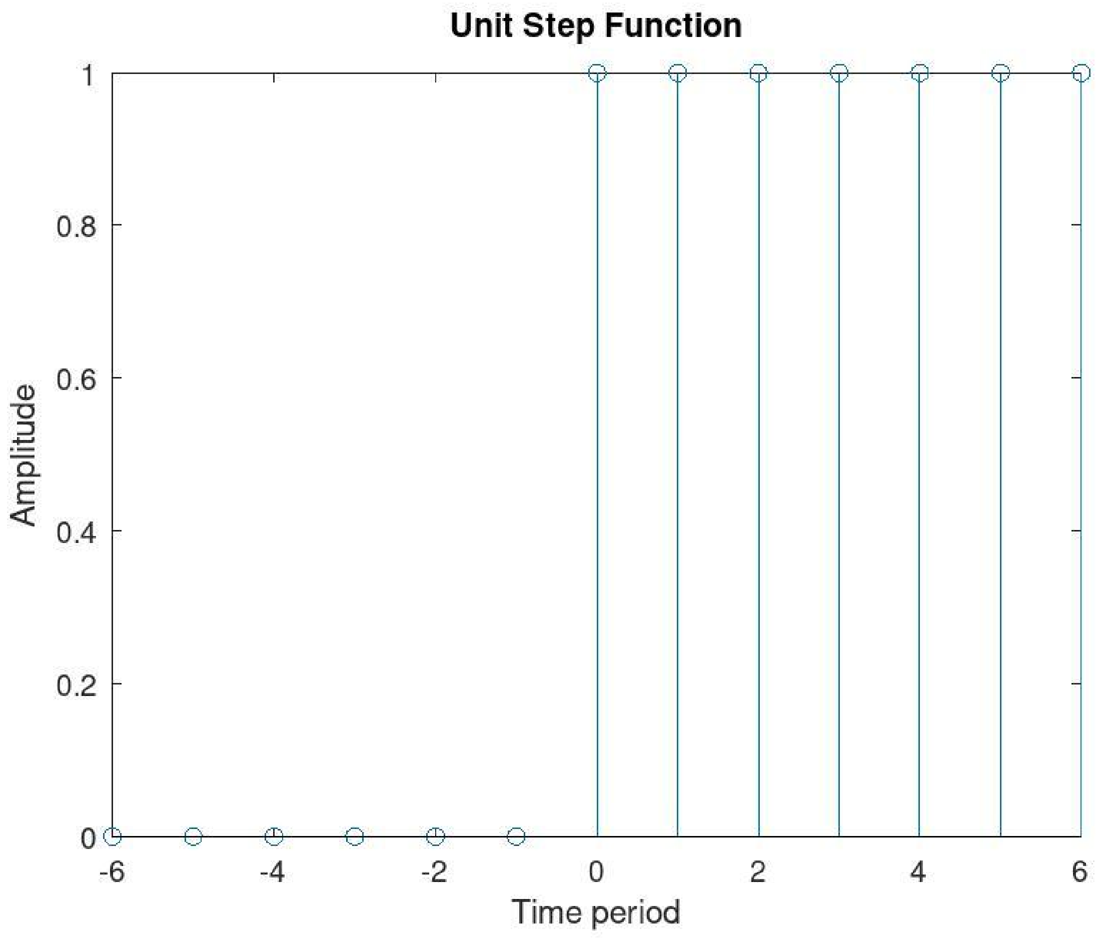
3. Generate a unit step function.

Source code:



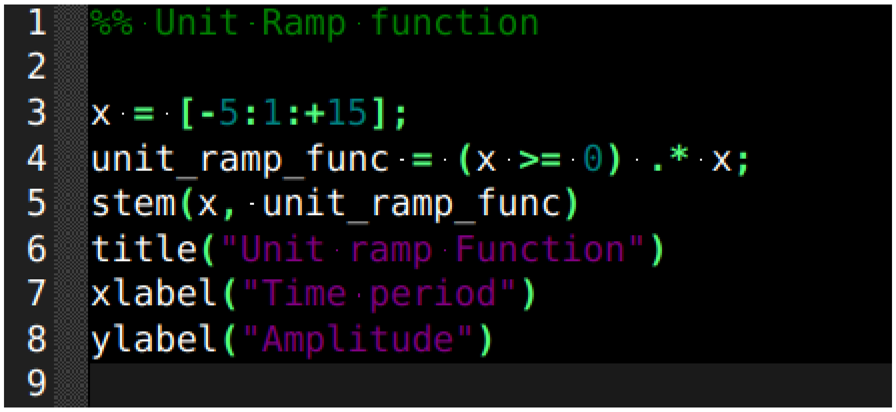
```
%% Unit step function

x = [-6:1:+6];
unit_step_func = x >= 0;
stem(x, unit_step_func)
title("Unit Step Function")
xlabel("Time period")
ylabel("Amplitude")

```

Output:



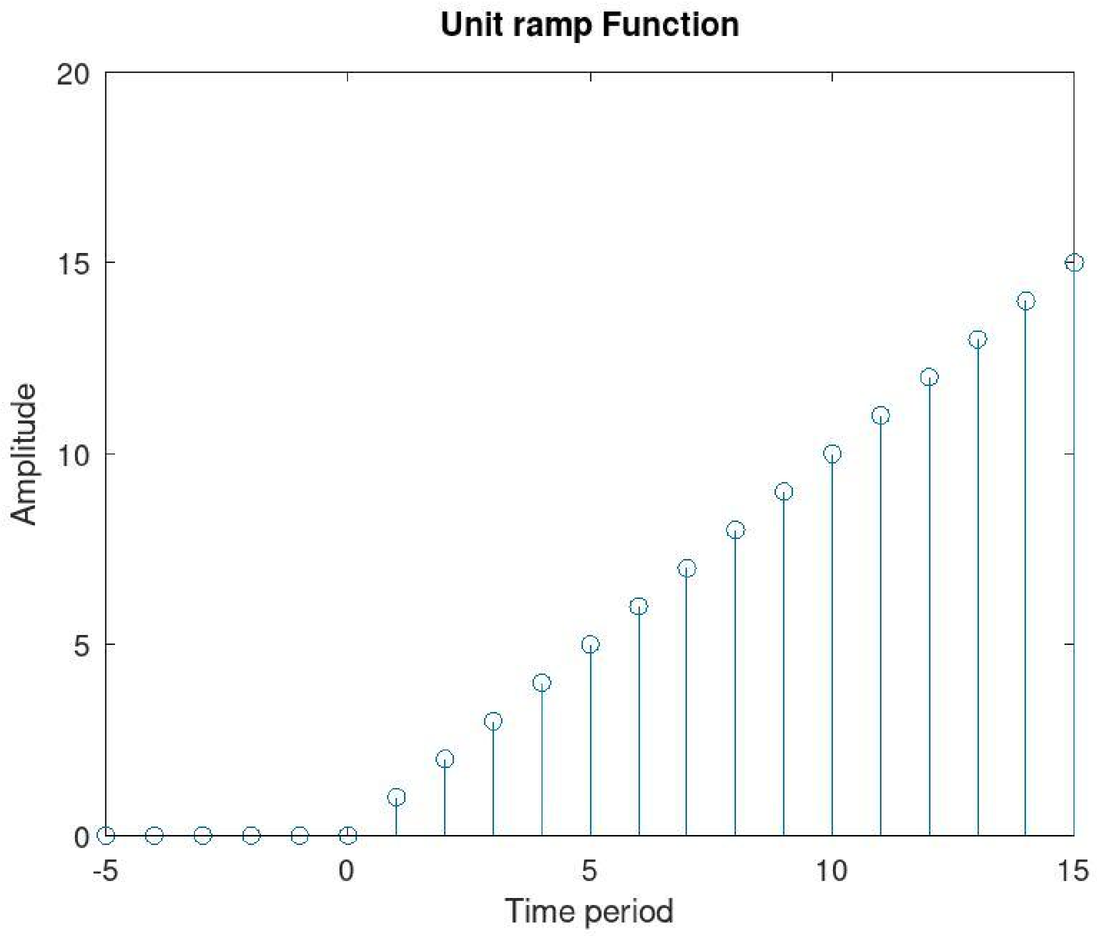
4. Generate a unit ramp function.

Source code:



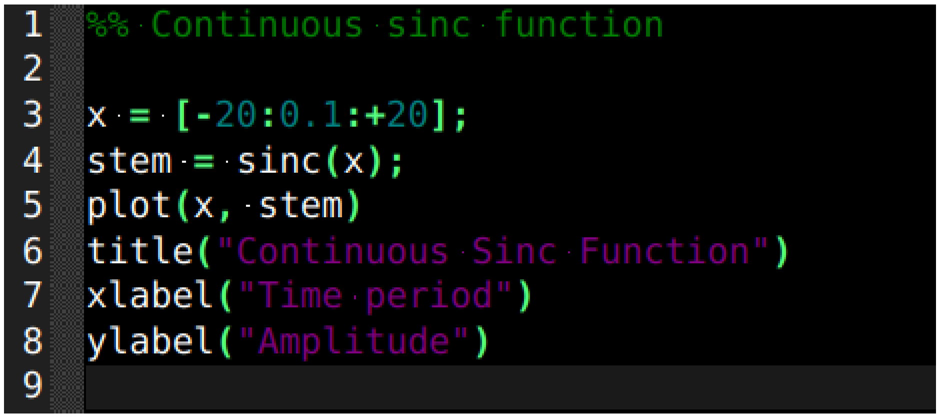
```
%% Unit Ramp function

x = [-5:1:+15];
unit_ramp_func = (x >= 0) .* x;
stem(x, unit_ramp_func)
title("Unit ramp Function")
xlabel("Time period")
ylabel("Amplitude")

```

Output:



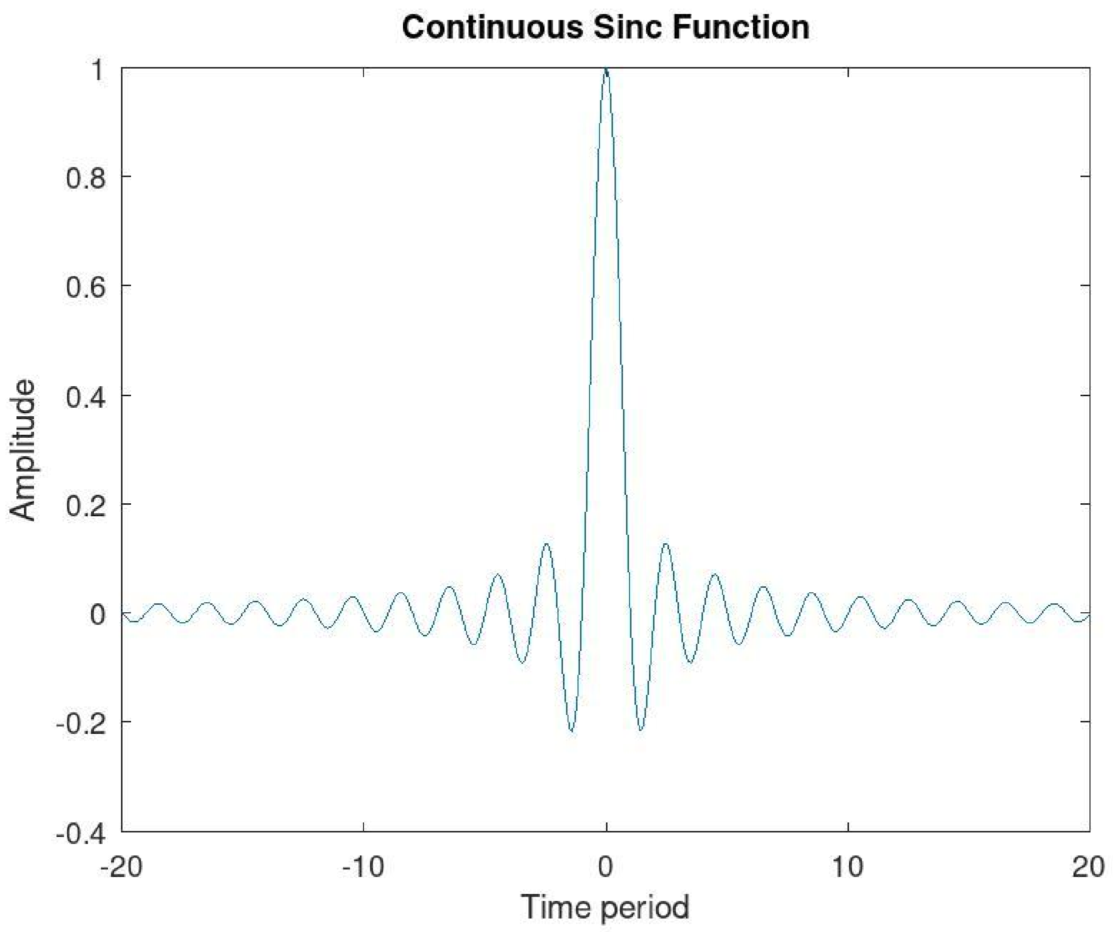
5. Generate a continuous time sinc function

Source Code:



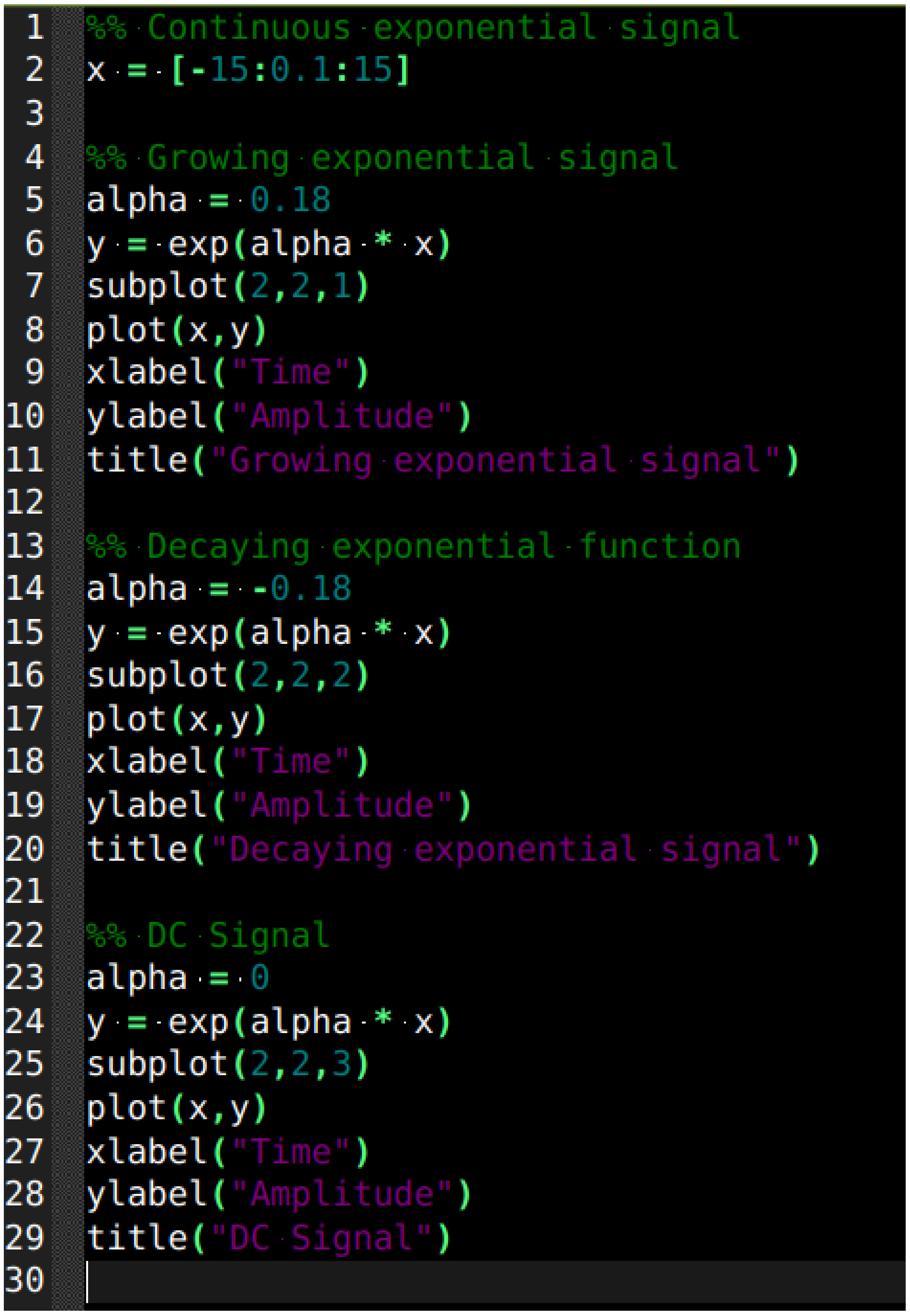
```
%% Continuous sinc function

x = [-20:0.1:+20];
stem = sinc(x);
plot(x, stem)
title("Continuous Sinc Function")
xlabel("Time period")
ylabel("Amplitude")
```

Output:



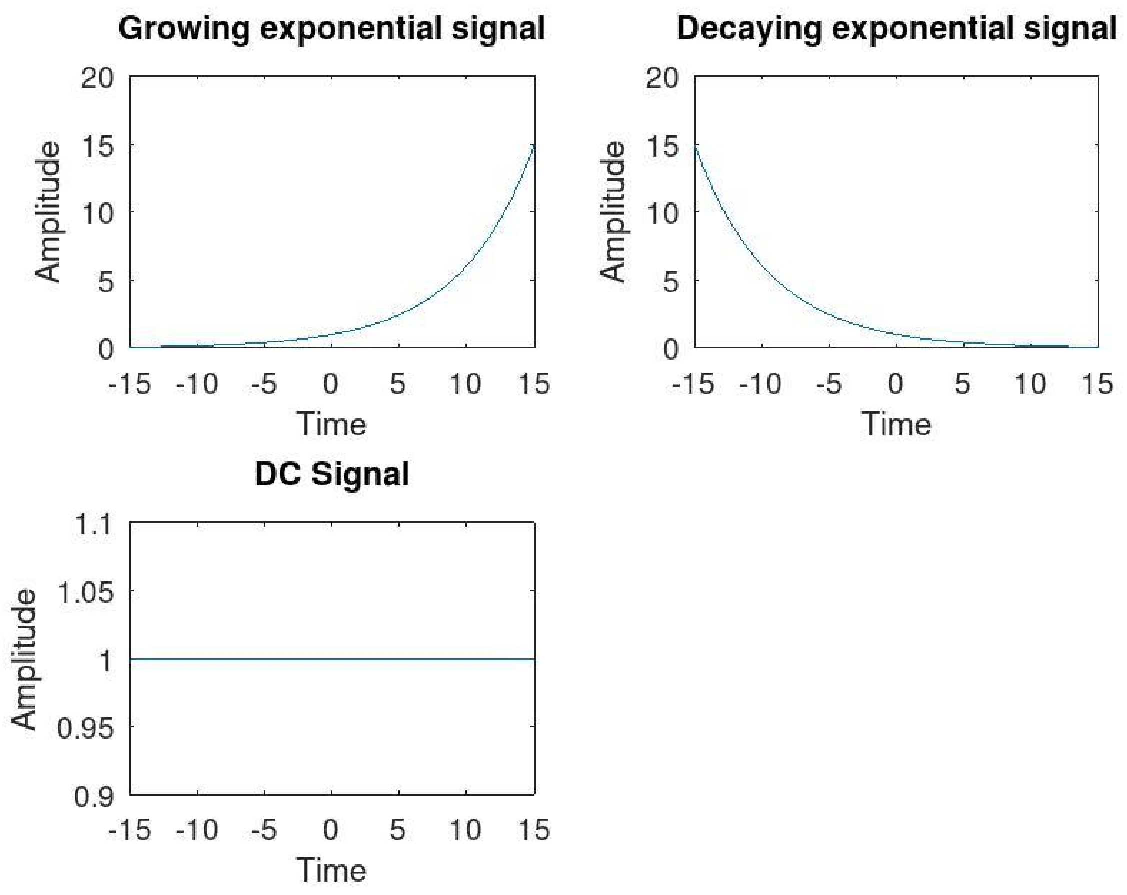
6. Generate a continuous time exponential (growing, decaying, DC signal)

Source Code:



```
%% Continuous exponential signal
x = [-15:0.1:15]

%% Growing exponential signal
alpha = 0.18
y = exp(alpha * x)
subplot(2,2,1)
plot(x,y)
xlabel("Time")
ylabel("Amplitude")
title("Growing exponential signal")

%% Decaying exponential function
alpha = -0.18
y = exp(alpha * x)
subplot(2,2,2)
plot(x,y)
xlabel("Time")
ylabel("Amplitude")
title("Decaying exponential signal")

%% DC Signal
alpha = 0
y = exp(alpha * x)
subplot(2,2,3)
plot(x,y)
xlabel("Time")
ylabel("Amplitude")
title("DC Signal")

```

Output:


Growing exponential signal
Decaying exponential signal
DC Signal
Amplitude
Time
20
15
10
5
0
-15
-10
-5
0
5
10
15
1.1
1.05
1
0.95
0.9

Lab 3

Sampling

The reduction of continuous-time signal to discrete-time signal is called as sampling. A sample is a value of the signal at a point in time and/or space.

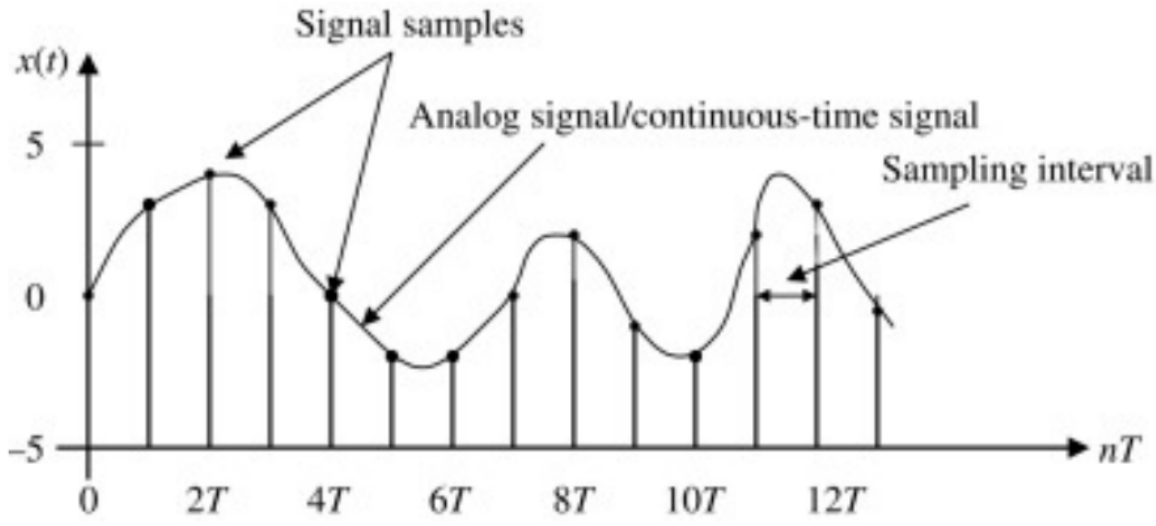
Some sampling terms:

1. Sampling period or sampling interval
   Let x(t) be a signal to be sampled, then the value of the continuous time signal is measured at every interval T, which is called as sampling period.

2. Sampling frequency or sampling rate
   The average number of samples obtained in each second is called as sampling frequency, f_s.

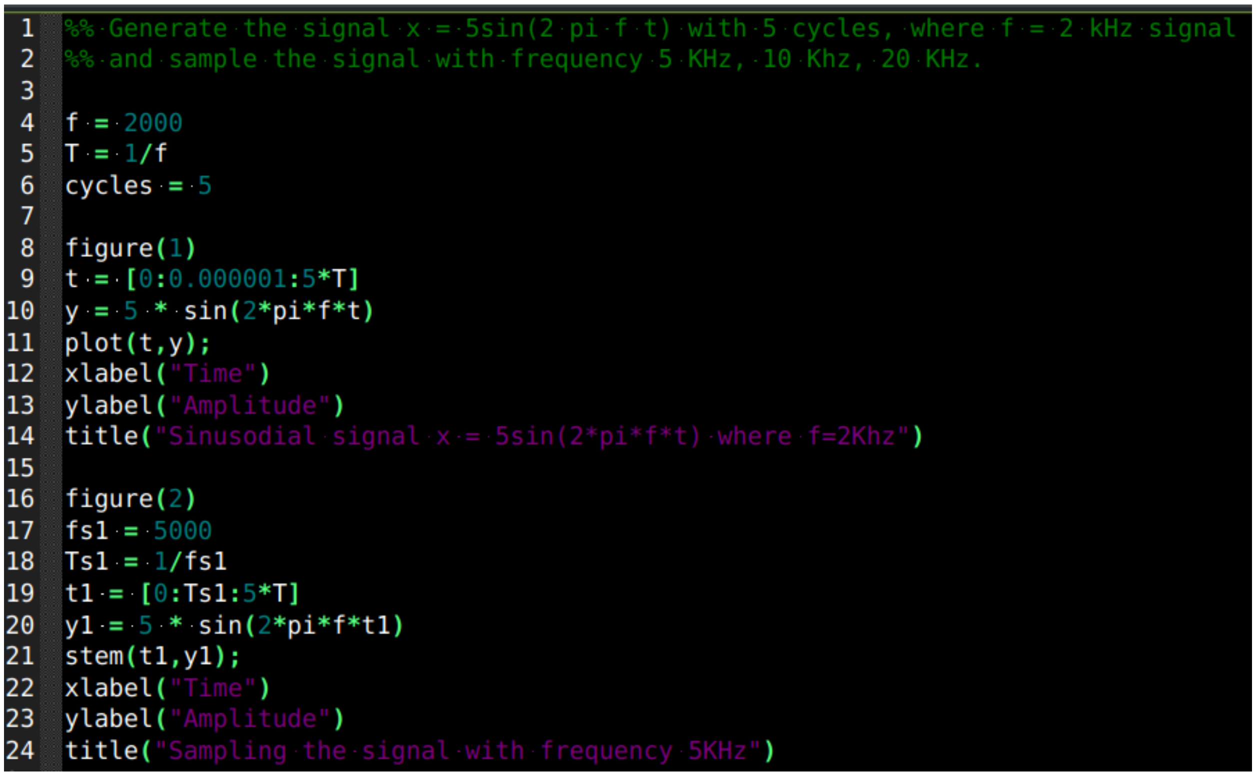
3. Aliasing
   Aliasing is an effect that causes different signals to become indistinguishable when sampled. Aliasing occurs because of undersampling.

4. Nyquist Sampling Theorem
   It states that to sample a signal without aliasing, the sampling rate must be at least twice the frequency of the signal.

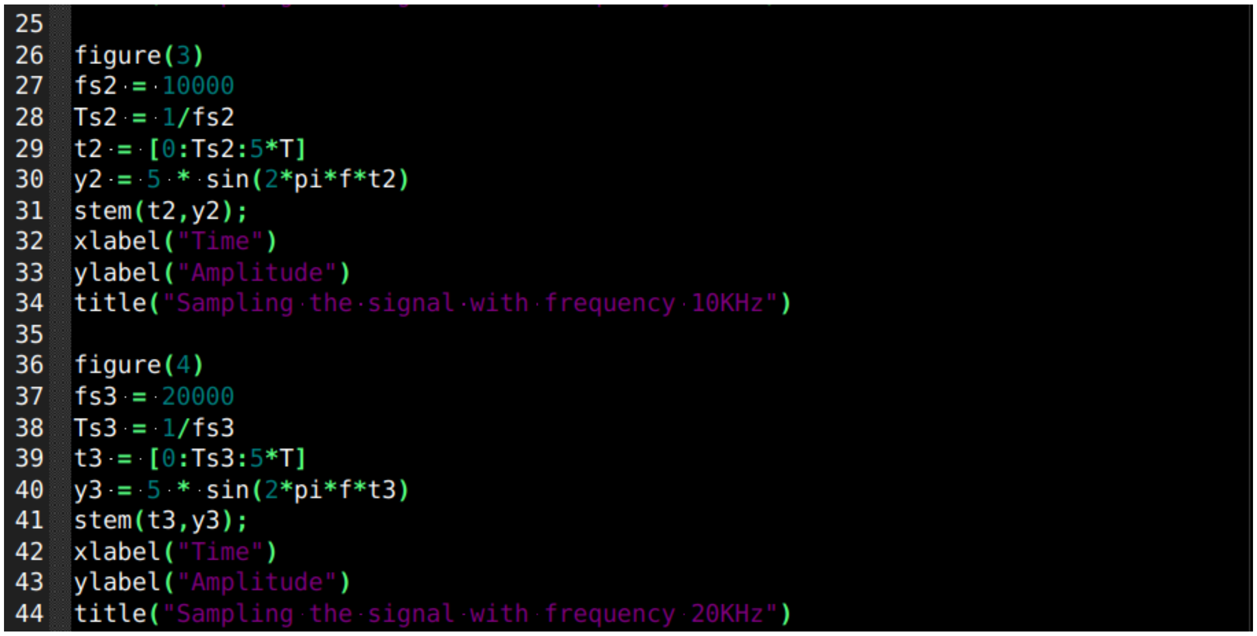


Question 1:
Generate the signal x = 5sin(2 pi f t) with 5 cycles, where f = 2 kHz signal and sample the signal with frequency 5 KHz, 10 Khz, 20 KHz. (Title and label each figure)

Source Code:

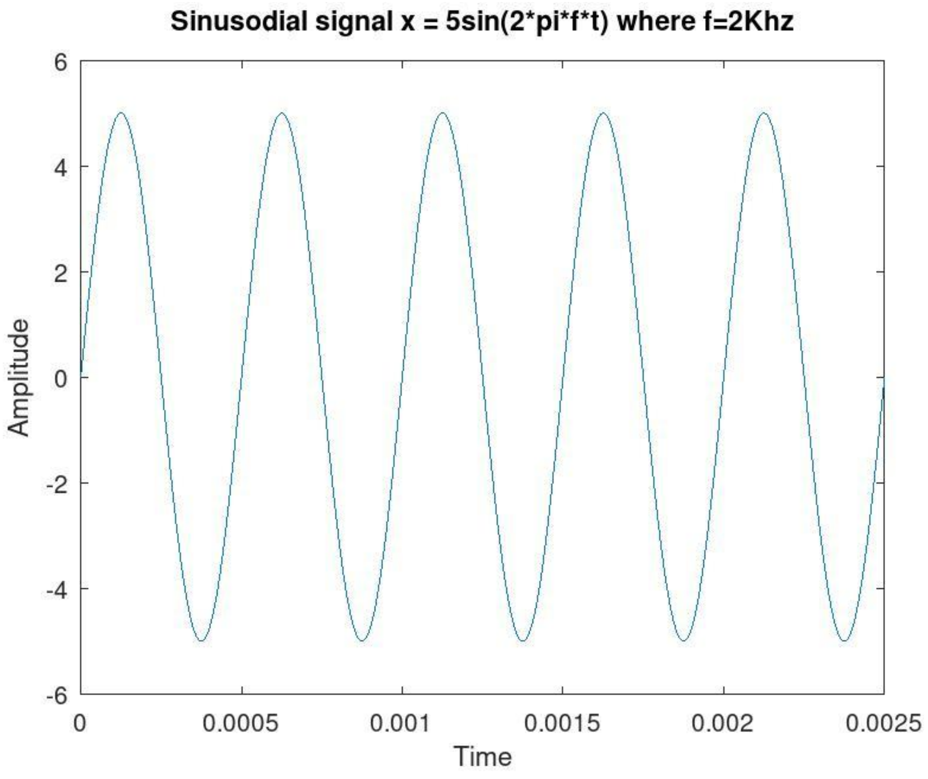


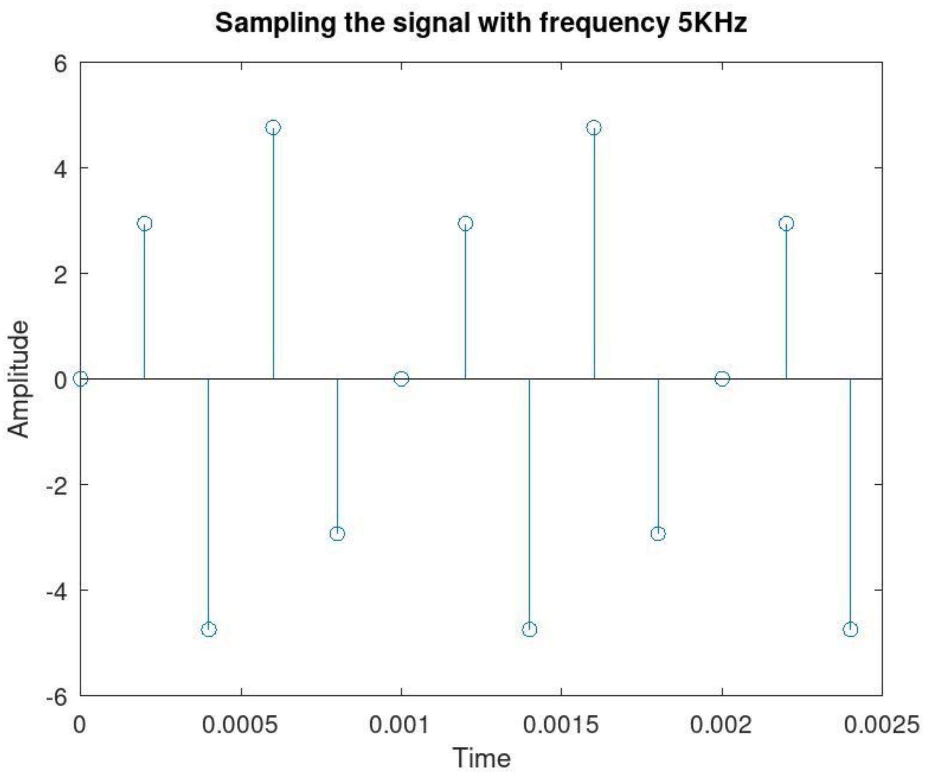
```
%% Generate the signal x = 5sin(2 pi f t) with 5 cycles, where f = 2 kHz signal
%% and sample the signal with frequency 5 KHz, 10 Khz, 20 KHz.

f = 2000
T = 1/f
cycles = 5

figure(1)
t = [0:0.000001:5*T]
y = 5 * sin(2*pi*f*t)
plot(t,y);
xlabel("Time")
ylabel("Amplitude")
title("Sinusodial signal x = 5sin(2*pi*f*t) where f=2Khz")

figure(2)
fs1 = 5000
Ts1 = 1/fs1
t1 = [0:Ts1:5*T]
y1 = 5 * sin(2*pi*f*t1)
stem(t1,y1);
xlabel("Time")
ylabel("Amplitude")
title("Sampling the signal with frequency 5KHz")
```

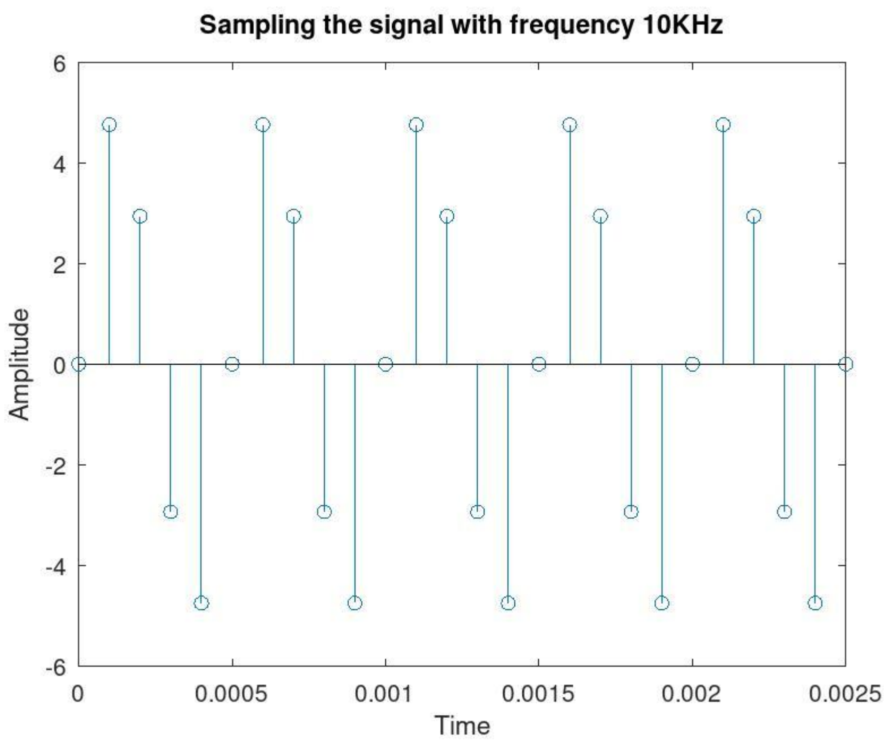


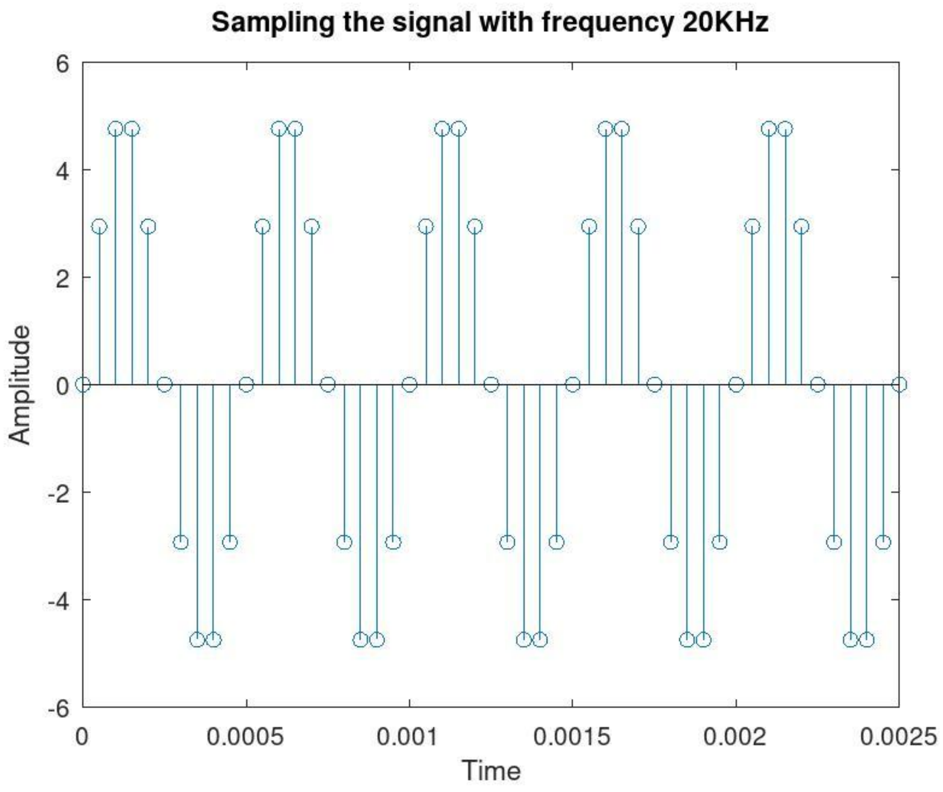
```

figure(3)
fs2 = 10000
Ts2 = 1/fs2
t2 = [0:Ts2:5*T]
y2 = 5 * sin(2*pi*f*t2)
stem(t2,y2);
xlabel("Time")
ylabel("Amplitude")
title("Sampling the signal with frequency 10KHz")

figure(4)
fs3 = 20000
Ts3 = 1/fs3
t3 = [0:Ts3:5*T]
y3 = 5 * sin(2*pi*f*t3)
stem(t3,y3);
xlabel("Time")
ylabel("Amplitude")
title("Sampling the signal with frequency 20KHz")
```

Output


Sinusodial signal x = 5sin(2*pi*f*t) where f=2Khz
6
4
2
0
-2
-4
-6
Amplitude
0
0.0005
0.001
0.0015
0.002
0.0025
Time

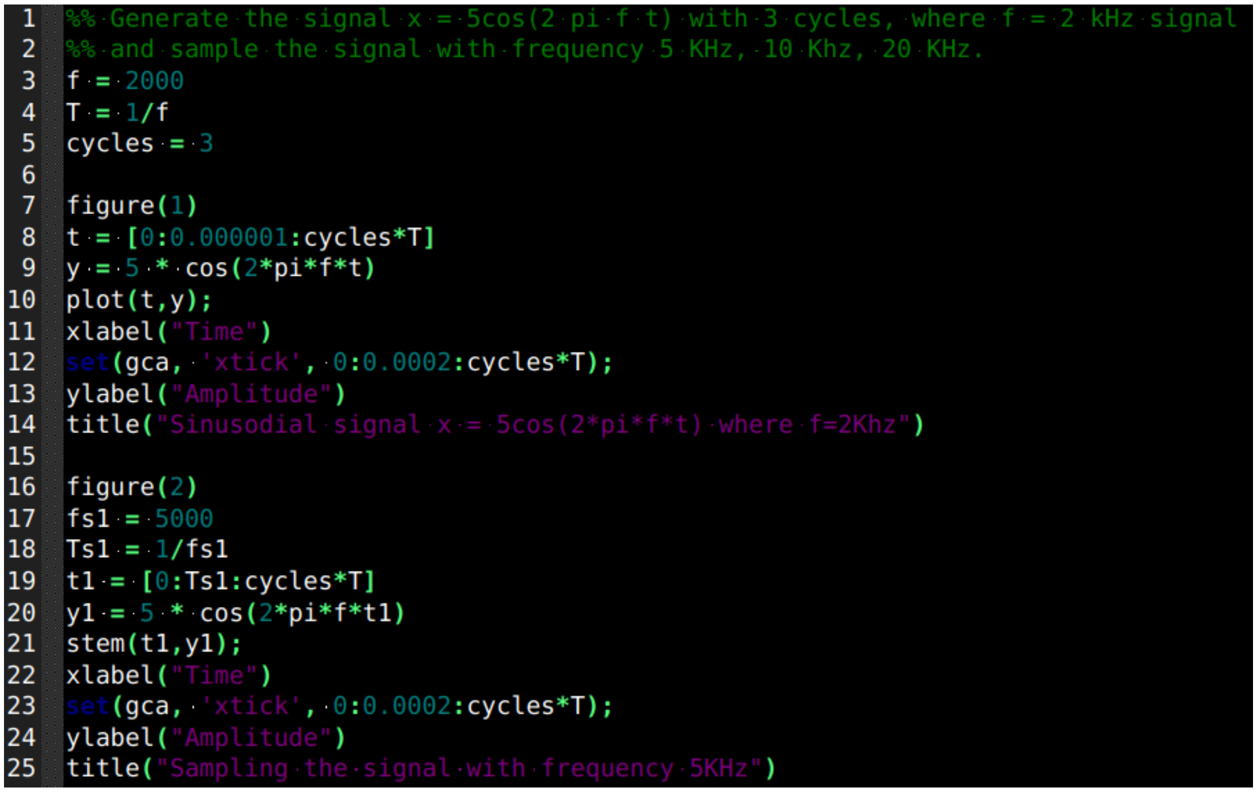

Sampling the signal with frequency 5KHz
6
4
2
0
-2
-4
-6
Amplitude
0
0.0005
0.001
0.0015
0.002
0.0025
Time


Sampling the signal with frequency 10KHz
Amplitude
Time
6
4
2
0
-2
-4
-6
0
0.0005
0.001
0.0015
0.002
0.0025

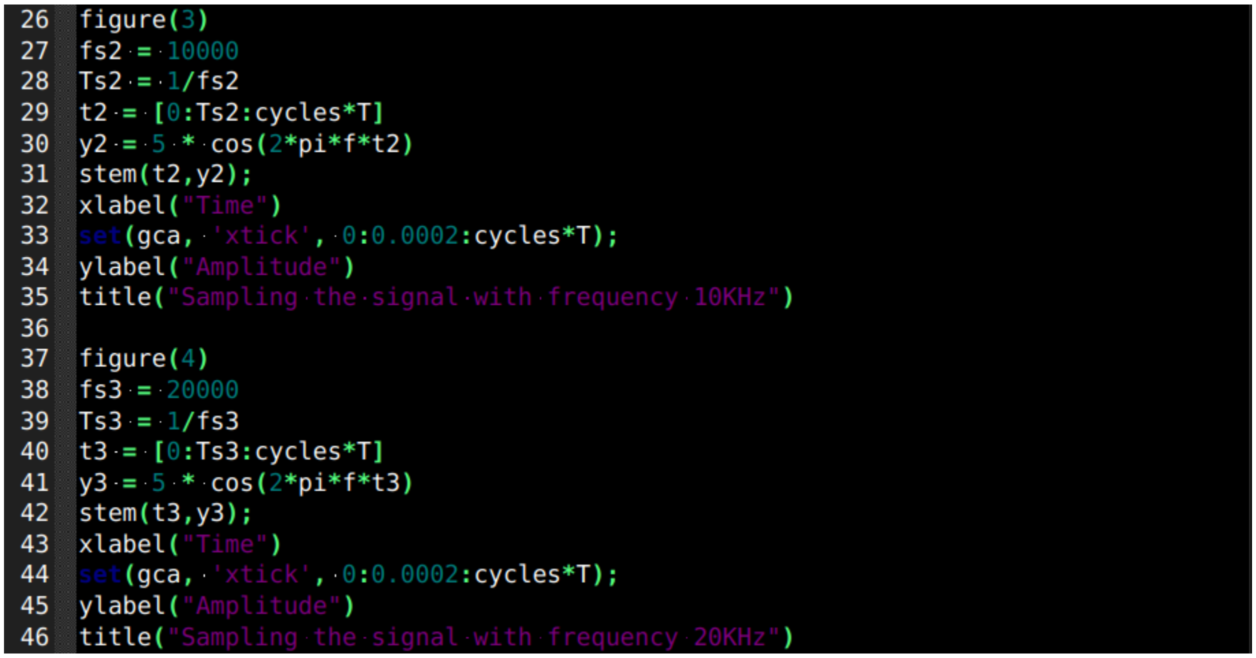

Sampling the signal with frequency 20KHz
Amplitude
Time
6
4
2
0
-2
-4
-6
0
0.0005
0.001
0.0015
0.002
0.0025

Question 2:
Generate the signal x = 5cos(2 pi f t) with 3 cycles, where f = 2 kHz signal and sample the signal with frequency 5 KHz, 10 Khz, 20 KHz. (Title and label each figure)

Source Code:

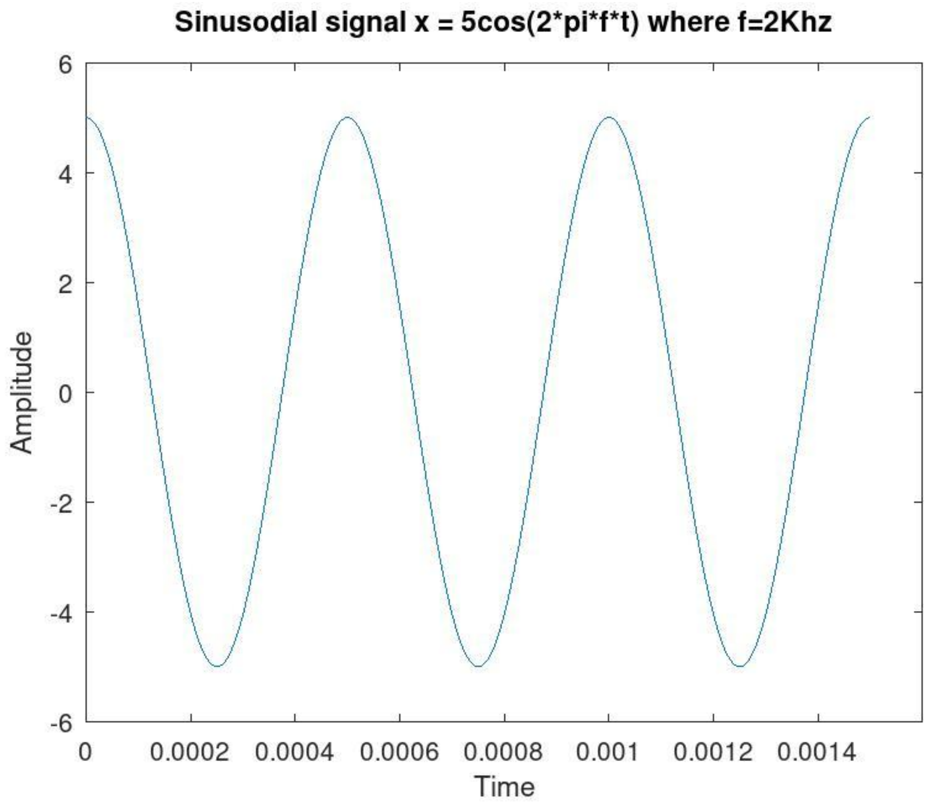


```
%% Generate the signal x = 5cos(2 pi f t) with 3 cycles, where f = 2 kHz signal
%% and sample the signal with frequency 5 KHz, 10 Khz, 20 KHz.
f = 2000
T = 1/f
cycles = 3

figure(1)
t = [0:0.000001:cycles*T]
y = 5 * cos(2*pi*f*t)
plot(t,y);
xlabel("Time")
set(gca, 'xtick', 0:0.0002:cycles*T);
ylabel("Amplitude")
title("Sinusodial signal x = 5cos(2*pi*f*t) where f=2Khz")

figure(2)
fs1 = 5000
Ts1 = 1/fs1
t1 = [0:Ts1:cycles*T]
y1 = 5 * cos(2*pi*f*t1)
stem(t1,y1);
xlabel("Time")
set(gca, 'xtick', 0:0.0002:cycles*T);
ylabel("Amplitude")
title("Sampling the signal with frequency 5KHz")
```

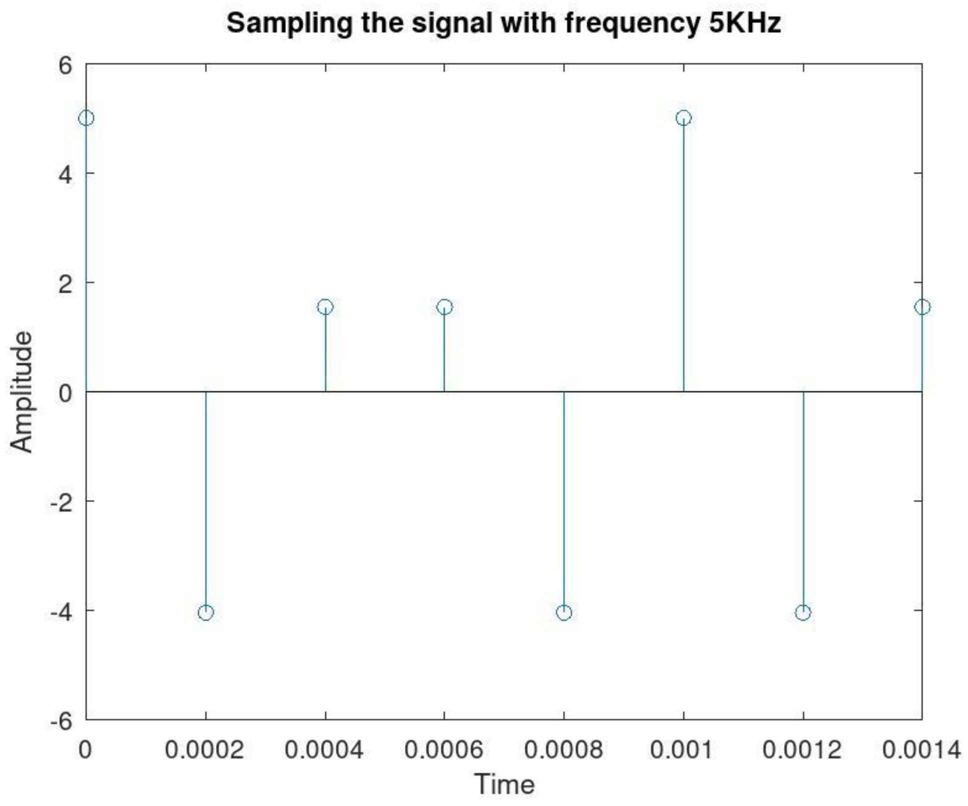


```
figure(3)
fs2 = 10000
Ts2 = 1/fs2
t2 = [0:Ts2:cycles*T]
y2 = 5 * cos(2*pi*f*t2)
stem(t2,y2);
xlabel("Time")
set(gca, 'xtick', 0:0.0002:cycles*T);
ylabel("Amplitude")
title("Sampling the signal with frequency 10KHz")

figure(4)
fs3 = 20000
Ts3 = 1/fs3
t3 = [0:Ts3:cycles*T]
y3 = 5 * cos(2*pi*f*t3)
stem(t3,y3);
xlabel("Time")
set(gca, 'xtick', 0:0.0002:cycles*T);
ylabel("Amplitude")
title("Sampling the signal with frequency 20KHz")
```

Output:

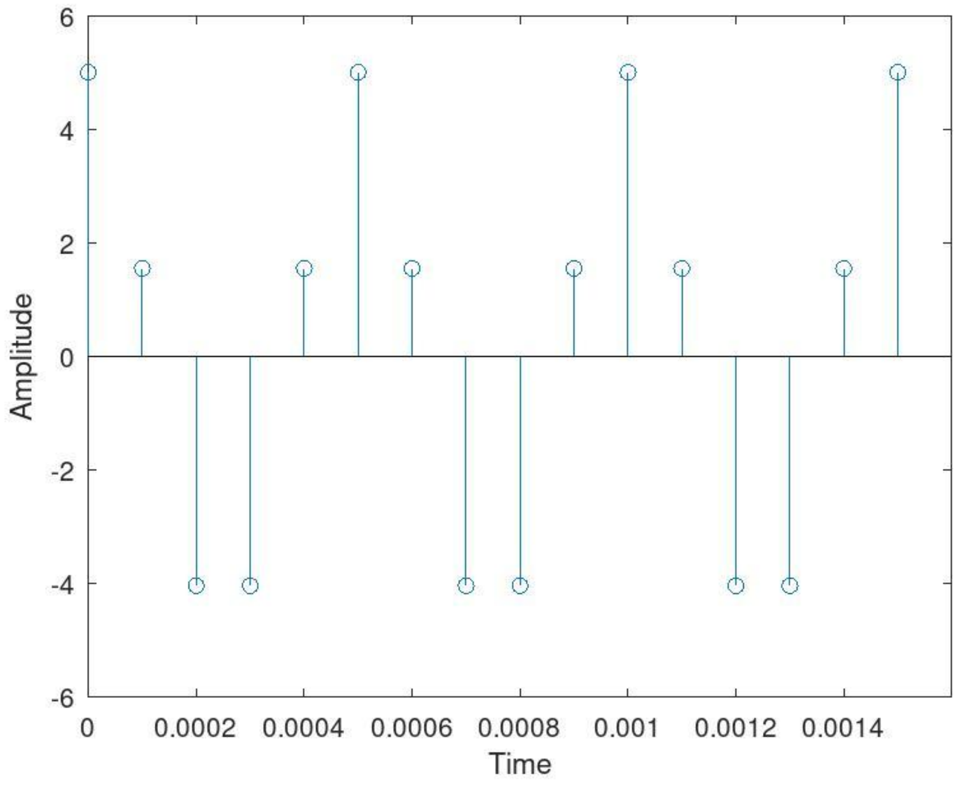


Sinusodial signal x = 5cos(2*pi*f*t) where f=2Khz
Amplitude
Time
6
4
2
0
-2
-4
-6
0
0.0002
0.0004
0.0006
0.0008
0.001
0.0012
0.0014

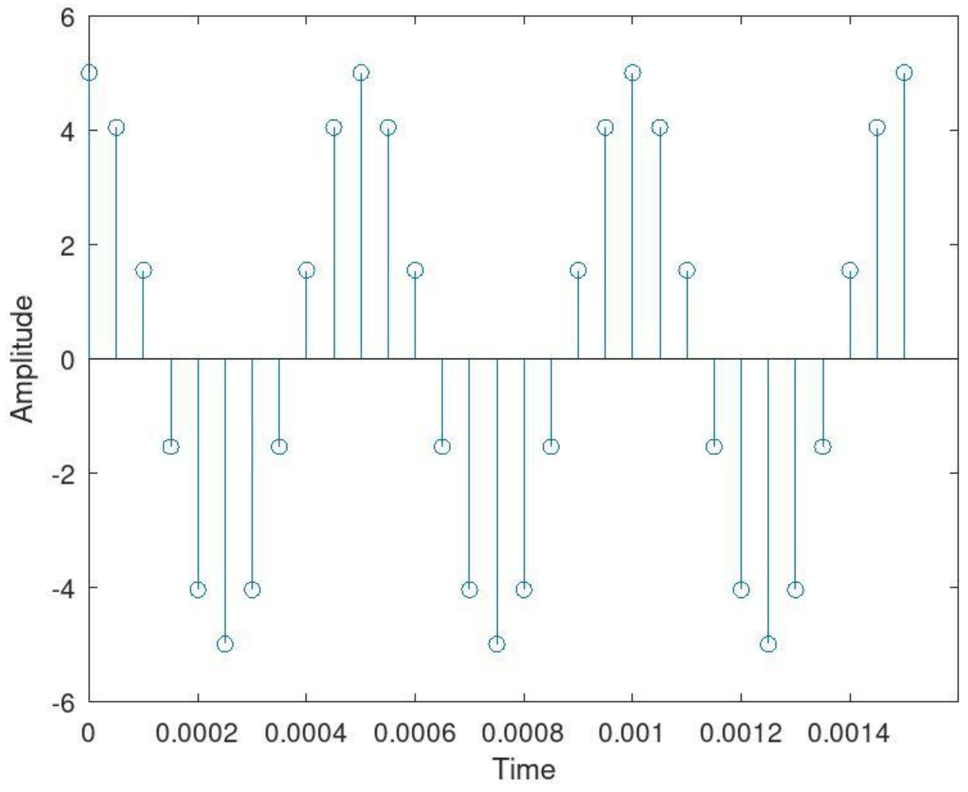


Sampling the signal with frequency 5KHz
Amplitude
Time
6
4
2
0
-2
-4
-6
0
0.0002
0.0004
0.0006
0.0008
0.001
0.0012
0.0014

Sampling the signal with frequency 10KHz



Sampling the signal with frequency 20KHz



Lab 4

Question:

Discuss Fourier Series and perform the following operation.

Fourier series is a linear combination of harmonically related complex exponential. It is used for the spectrum analysis of periodic signals. It models a periodic signal as a sum of distinct harmonic components. It decomposes a function into frequency components.

Spectrum analysis of continuous time periodic signal is generally called continuous time Fourier series. Similarly, spectrum analysis of discrete time periodic signal is called discrete time Fourier series.

Fourier Series is a function that breaks down any periodic function into a simple series of sine & cosine waves.

The generalized equation of Fourier series of an equation $x(t)$.

$$x(t) = a_0 + \sum_{n=1}^{\infty} [a_n cos(n\omega_0 t) + b_n sin(n\omega_0 t)]$$

Where,

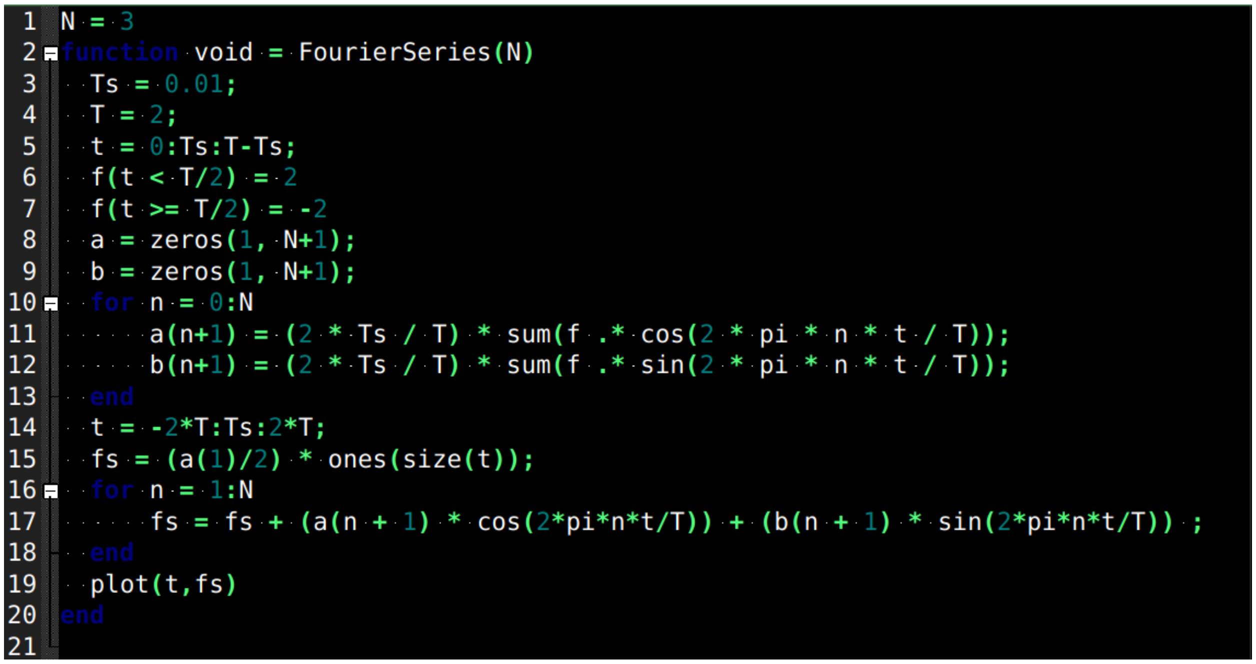
$$a_0 = \frac{1}{T} \int_T x(t)\, dt$$

$$a_n = \frac{2}{T} \int_T x(t)\, cos(n\omega_0 t)\, dt$$

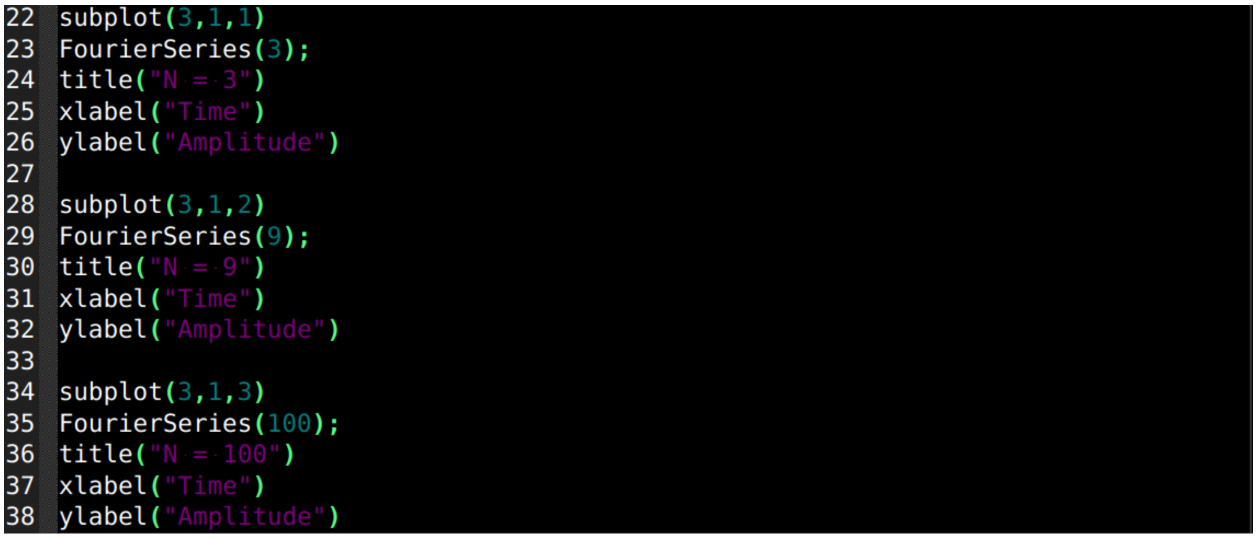
$$b_n = \frac{2}{T} \int_T x(t)\, sin(n\omega_0 t)\, dt$$

1. Fourier series expansion of odd signal for different N.(N= 3, 9, 100).

Source code:

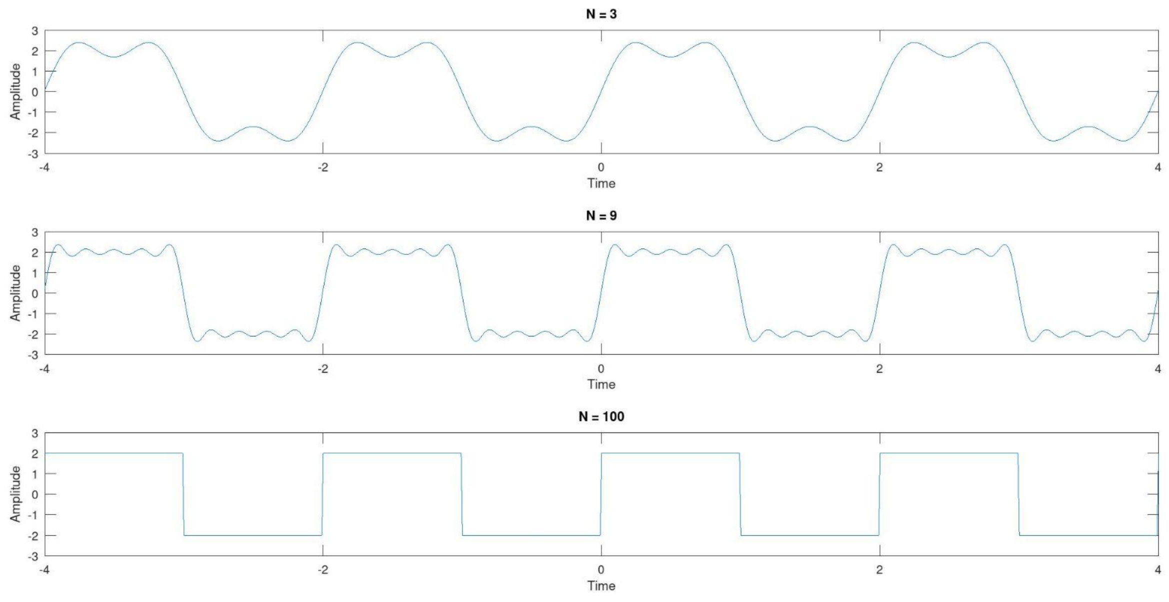


```
N = 3
function void = FourierSeries(N)
  Ts = 0.01;
  T = 2;
  t = 0:Ts:T-Ts;
  f(t < T/2) = 2
  f(t >= T/2) = -2
  a = zeros(1, N+1);
  b = zeros(1, N+1);
  for n = 0:N
      a(n+1) = (2 * Ts / T) * sum(f .* cos(2 * pi * n * t / T));
      b(n+1) = (2 * Ts / T) * sum(f .* sin(2 * pi * n * t / T));
  end
  t = -2*T:Ts:2*T;
  fs = (a(1)/2) * ones(size(t));
  for n = 1:N
      fs = fs + (a(n + 1) * cos(2*pi*n*t/T)) + (b(n + 1) * sin(2*pi*n*t/T)) ;
  end
  plot(t,fs)
end

```



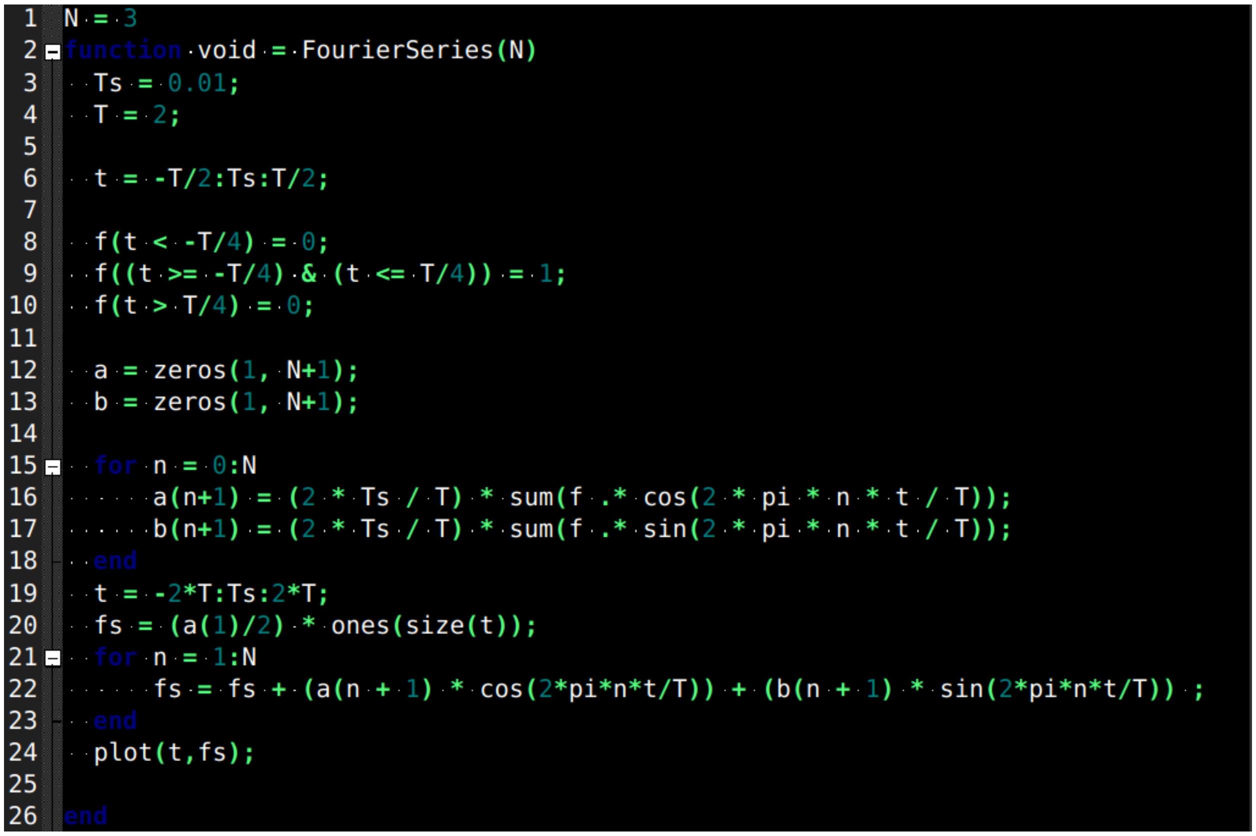
```
subplot(3,1,1)
FourierSeries(3);
title("N = 3")
xlabel("Time")
ylabel("Amplitude")

subplot(3,1,2)
FourierSeries(9);
title("N = 9")
xlabel("Time")
ylabel("Amplitude")

subplot(3,1,3)
FourierSeries(100);
title("N = 100")
xlabel("Time")
ylabel("Amplitude")
```

Output:

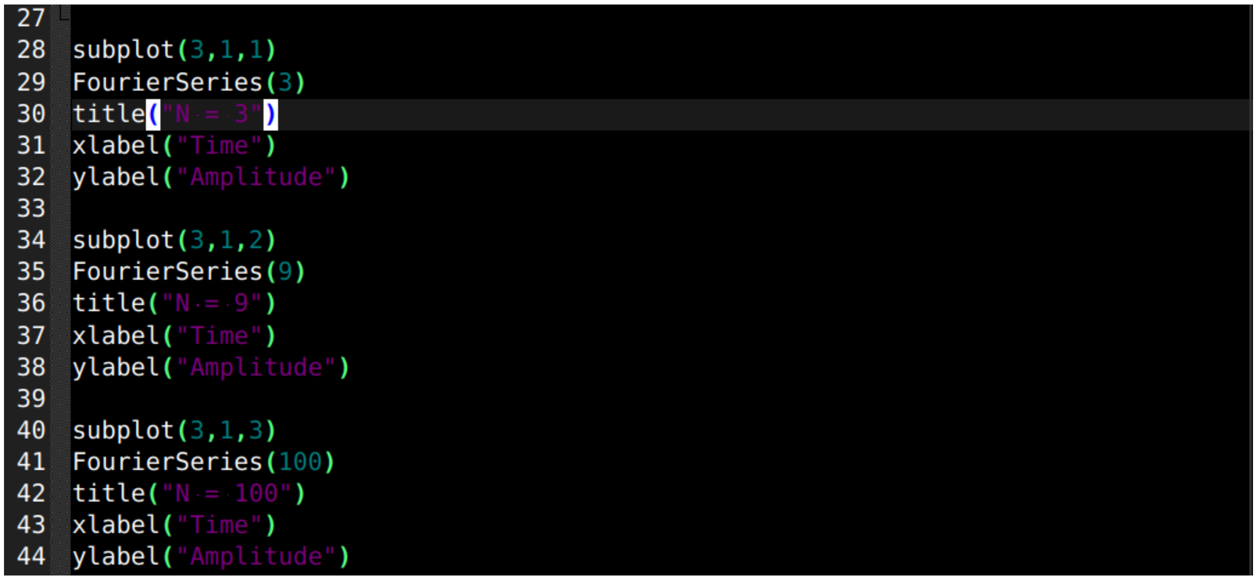


2. Fourier series expansion of even signal for different N. (N=3,9,100).

Source Code:

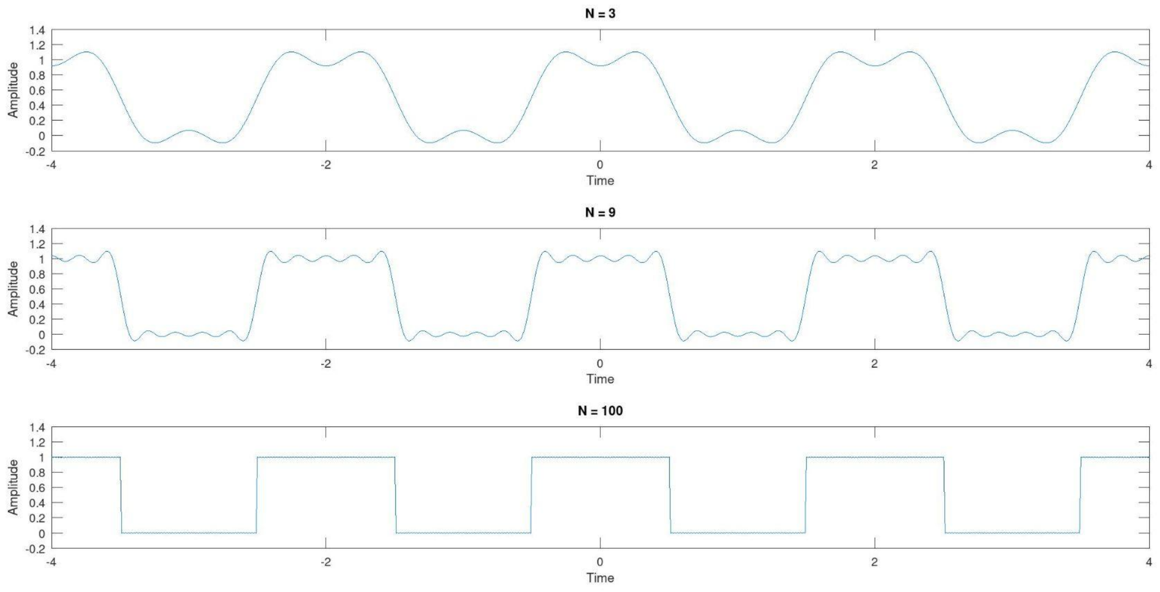


```
N = 3
function void = FourierSeries(N)
  Ts = 0.01;
  T = 2;

  t = -T/2:Ts:T/2;

  f(t < -T/4) = 0;
  f((t >= -T/4) & (t <= T/4)) = 1;
  f(t > T/4) = 0;

  a = zeros(1, N+1);
  b = zeros(1, N+1);

  for n = 0:N
      a(n+1) = (2 * Ts / T) * sum(f .* cos(2 * pi * n * t / T));
      b(n+1) = (2 * Ts / T) * sum(f .* sin(2 * pi * n * t / T));
  end
  t = -2*T:Ts:2*T;
  fs = (a(1)/2) * ones(size(t));
  for n = 1:N
      fs = fs + (a(n + 1) * cos(2*pi*n*t/T)) + (b(n + 1) * sin(2*pi*n*t/T)) ;
  end
  plot(t,fs);

end
```



```

subplot(3,1,1)
FourierSeries(3)
title("N = 3")
xlabel("Time")
ylabel("Amplitude")

subplot(3,1,2)
FourierSeries(9)
title("N = 9")
xlabel("Time")
ylabel("Amplitude")

subplot(3,1,3)
FourierSeries(100)
title("N = 100")
xlabel("Time")
ylabel("Amplitude")
```

Output:



Lab 5 Linear and Circular Convolution

Question
Write theory about both convolutions.

Linear Convolution

Linear convolution is a mathematical operation done to calculate the output of any Linear-Time Invariant (LTI) system given its input and impulse response.

We can represent Linear Convolution as
y(n)=x(n)*h(n)
Here, y(n) is the output (also known as convolution sum). x(n) is the input signal, and h(n) is the impulse response of the LTI system.

In linear convolution, both the sequences (input and impulse response) may or may not be of equal sizes. That is, they may or may not have the same number of samples. Thus, the output, too, may or may not have the same number of samples as any of the inputs.

Circular Convolution

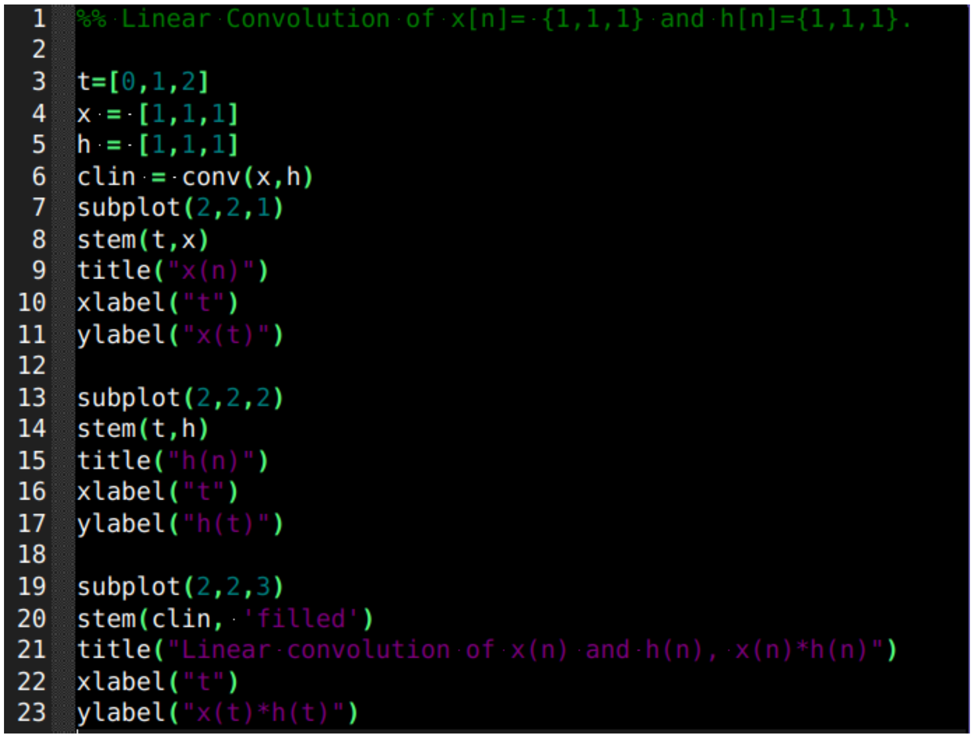
Circular convolution is essentially the same process as linear convolution. Just like linear convolution, it involves the operation of folding a sequence, shifting it, multiplying it with another sequence, and summing the resulting products. However, in circular convolution, the signals are all periodic. Thus, the shifting can be thought of as actually being a rotation. Since the values keep repeating because of the periodicity. Hence, it is known as circular convolution.

We can represent Circular Convolution as
y(n)=x(n)⊕h(n)
Here, y(n) is a periodic output, x(n) is a periodic input, and h(n) is the periodic impulse response of the LTI system.

In circular convolution, both the sequences (input and impulse response) must be of equal sizes. They must have the same number of samples. Thus the output of a circular convolution has the same number of samples as the two inputs.

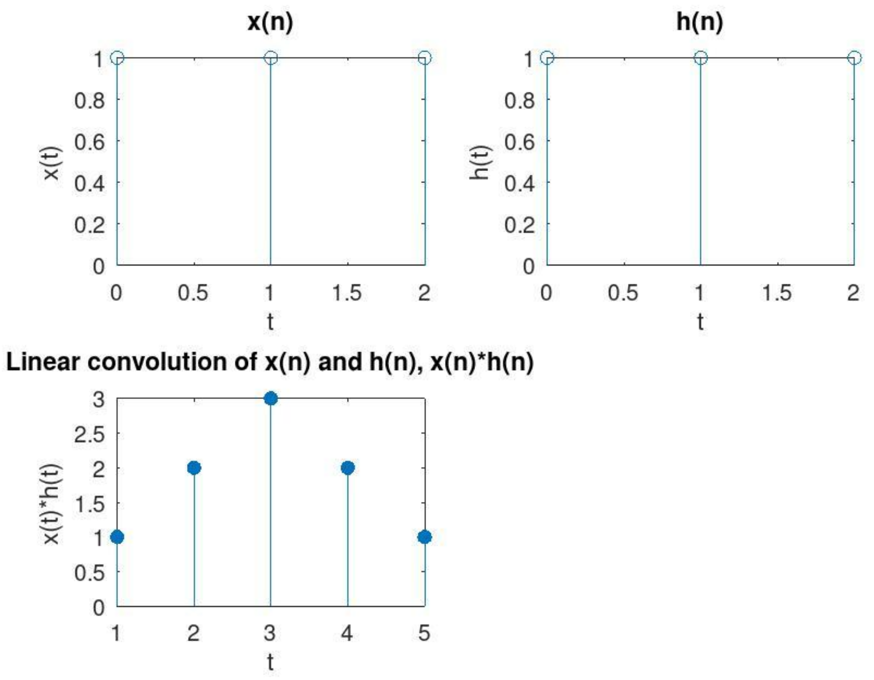
Perform Linear Convolution:
1. x[n]= {1,1,1} and h[n]={1,1,1}.

Source Code:



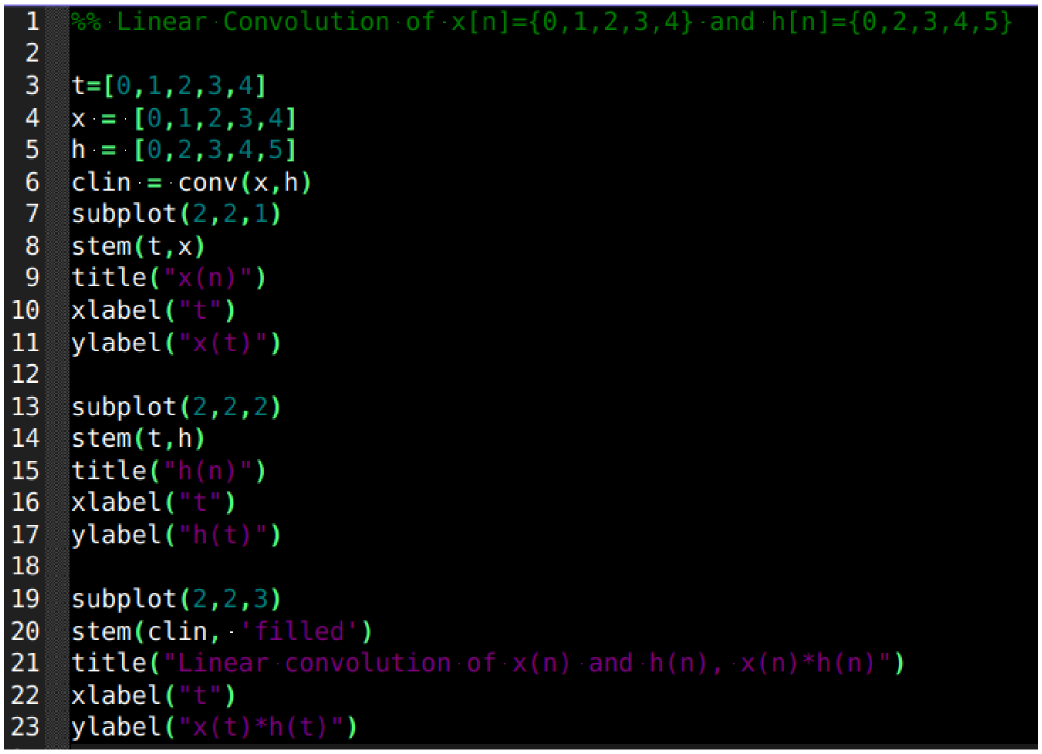
```
%% Linear Convolution of x[n]= {1,1,1} and h[n]={1,1,1}.

t=[0,1,2]
x = [1,1,1]
h = [1,1,1]
clin = conv(x,h)
subplot(2,2,1)
stem(t,x)
title("x(n)")
xlabel("t")
ylabel("x(t)")

subplot(2,2,2)
stem(t,h)
title("h(n)")
xlabel("t")
ylabel("h(t)")

subplot(2,2,3)
stem(clin, 'filled')
title("Linear convolution of x(n) and h(n), x(n)*h(n)")
xlabel("t")
ylabel("x(t)*h(t)")
```

Output:



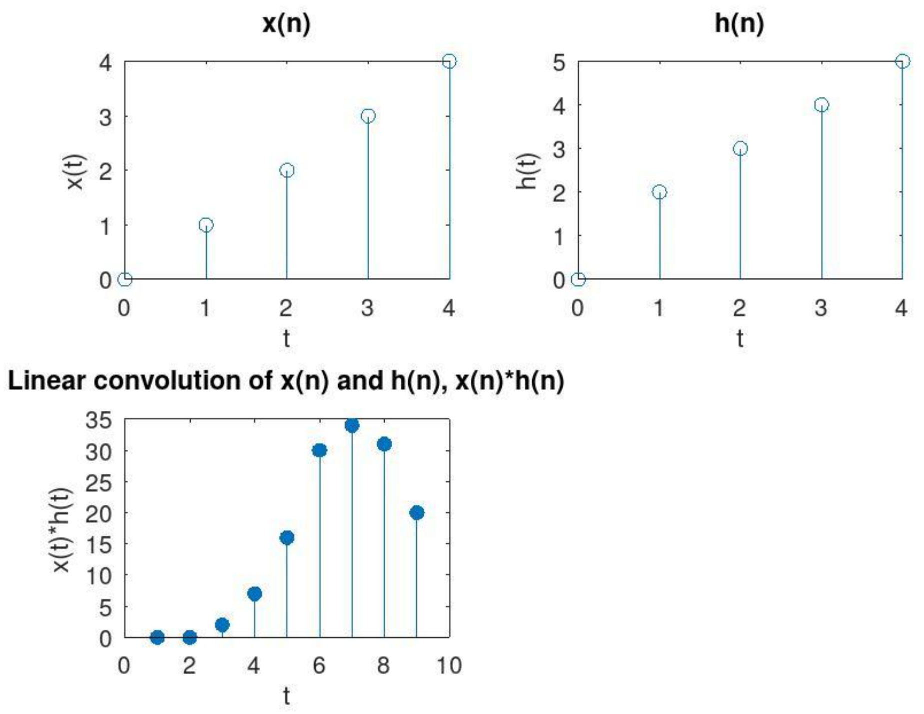
2. x[n]={0,1,2,3,4} and h[n]={0,2,3,4,5}

Source Code



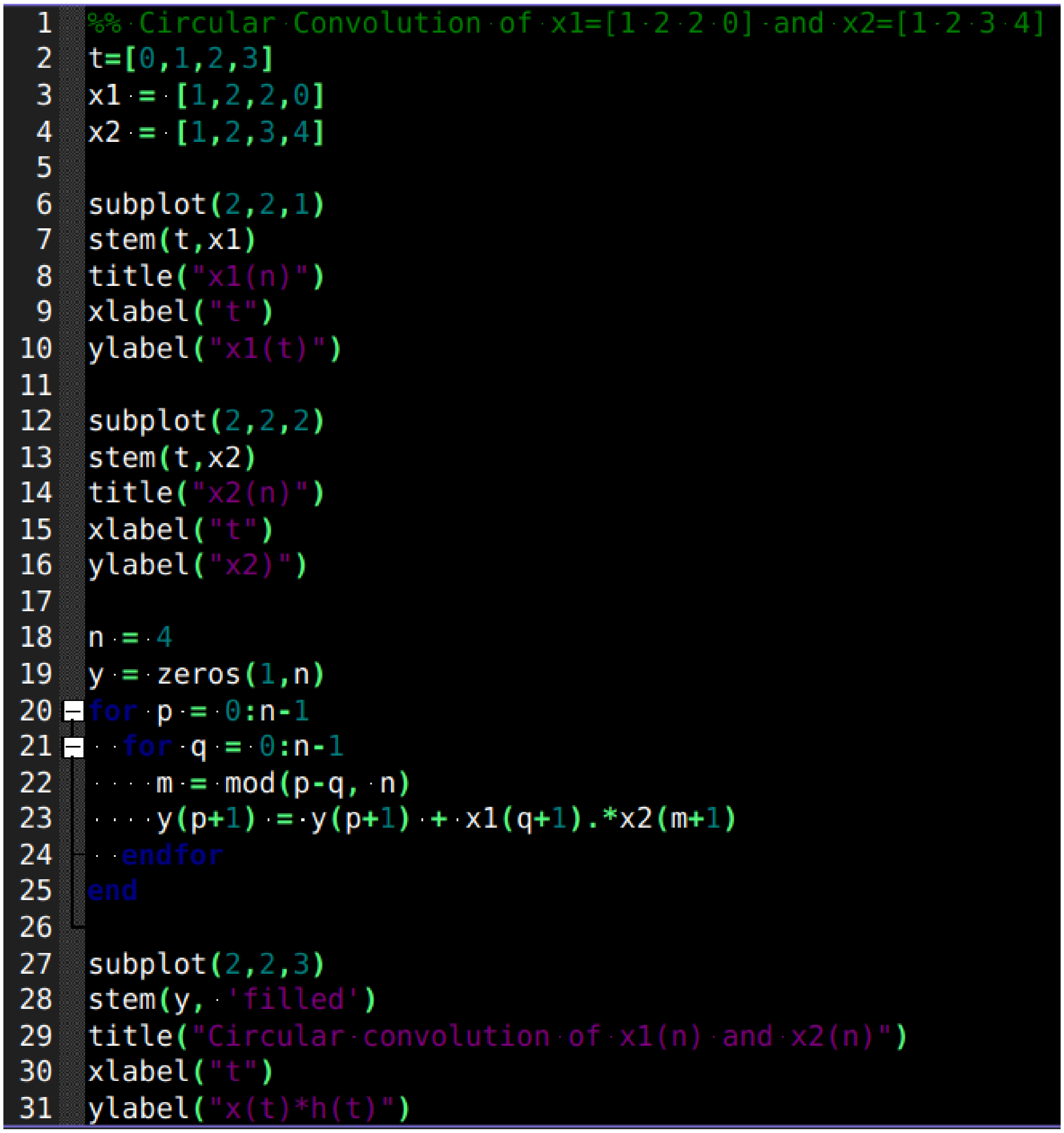
```
%% Linear Convolution of x[n]={0,1,2,3,4} and h[n]={0,2,3,4,5}

t=[0,1,2,3,4]
x = [0,1,2,3,4]
h = [0,2,3,4,5]
clin = conv(x,h)
subplot(2,2,1)
stem(t,x)
title("x(n)")
xlabel("t")
ylabel("x(t)")

subplot(2,2,2)
stem(t,h)
title("h(n)")
xlabel("t")
ylabel("h(t)")

subplot(2,2,3)
stem(clin, 'filled')
title("Linear convolution of x(n) and h(n), x(n)*h(n)")
xlabel("t")
ylabel("x(t)*h(t)")
```

Output



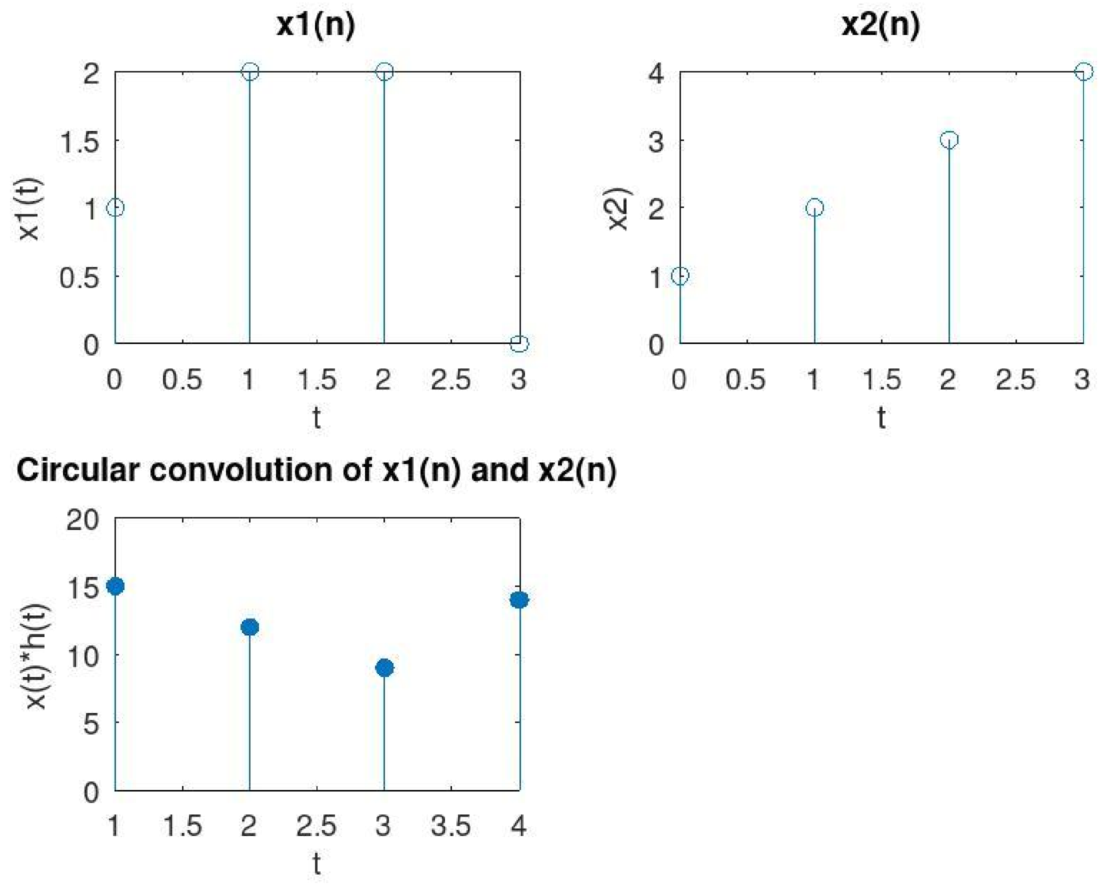
Perform Circular Convolution:
x1=[1 2 2 0] and x2=[1 2 3 4]

Source Code:



```
%% Circular Convolution of x1=[1 2 2 0] and x2=[1 2 3 4]
t=[0,1,2,3]
x1 = [1,2,2,0]
x2 = [1,2,3,4]

subplot(2,2,1)
stem(t,x1)
title("x1(n)")
xlabel("t")
ylabel("x1(t)")

subplot(2,2,2)
stem(t,x2)
title("x2(n)")
xlabel("t")
ylabel("x2)")

n = 4
y = zeros(1,n)
for p = 0:n-1
  for q = 0:n-1
    m = mod(p-q, n)
    y(p+1) = y(p+1) + x1(q+1).*x2(m+1)
  endfor
end

subplot(2,2,3)
stem(y, 'filled')
title("Circular convolution of x1(n) and x2(n)")
xlabel("t")
ylabel("x(t)*h(t)")
```

Output


x1(n)
x1(t)
t
x2(n)
x2)
t
Circular convolution of x1(n) and x2(n)
x(t)*h(t)
t