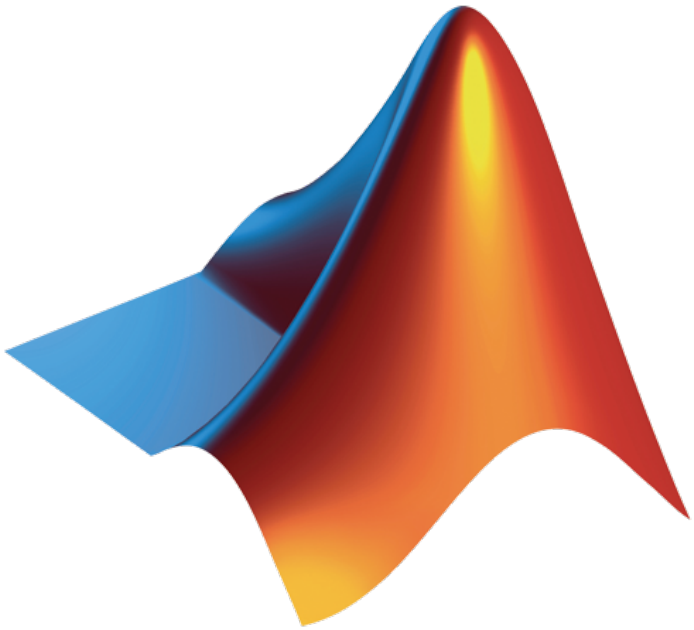
Indian Institute of Information Technology, Surat



MATLAB LAB FILES

Yash Sojitra UI22EC72

Electronics and Communication Engineering

Table of Contents:

Yash Sojitra
UI22EC72

LAB 1

Introduction to MATLAB

· MATLAB is a programming language and interactive environment designed for scientific computing, data analysis, and visualization. MATLAB stands for "matrix laboratory," which reflects its emphasis on matrix and array mathematics.

· MATLAB provides a variety of tools and functions for numerical computation, data analysis, signal processing, image processing, control systems, optimization, and machine learning. It is widely used in academic research, engineering, finance, and other industries where numerical analysis and computation are required.

· MATLAB code is written in a high-level language that is easy to learn and use. The code can be executed interactively in the MATLAB command window or compiled into standalone applications or executables. MATLAB also provides a rich set of visualization tools for creating plots, graphs, and other visual representations of data.

· Overall, MATLAB is a powerful and flexible tool for solving a wide range of numerical problems and analyzing data. Its popularity in academia and industry makes it a valuable skill to have for anyone interested in scientific computing or data analysis.

Yash Sojitra
UI22EC72

layout of MATLAB

· The MATLAB environment consists of several components, including:

1. Command Window: This is the main MATLAB interface where you can enter commands, view results, and interact with the MATLAB environment.
2. Editor Window: This is where you can create, edit, and save MATLAB code files. The editor provides features such as syntax highlighting, code folding, and debugging tools.
3. Workspace Window: This displays information about the variables currently in the MATLAB workspace. You can use this window to view, edit, and manipulate variables.
4. Command History Window: This displays a history of the commands that have been executed in the Command Window.
5. Current Folder Window: This displays the files and folders in the current working directory.
6. Help Browser: This provides access to the MATLAB documentation, including function reference, examples, and tutorials.
7. Plotting Windows: These are used to display graphs, plots, and other visualizations of data.

· In addition to these components, MATLAB provides a wide range of toolboxes and add-ons for specialized applications, such as signal processing, image processing, control systems, and optimization. These toolboxes provide additional functions and capabilities that can be integrated into your MATLAB code.

Yash Sojitra
UI22EC72

LAB 2

Basic MATLAB Commands

· Here are some basic MATLAB commands:

· **help**: Displays help documentation for a particular command or function.
· **clear**: Clears all variables from the workspace.
· **clc**: Clears the Command Window.
· **close**: Closes a figure window or other MATLAB window.
· **who**: Lists the variables in the workspace.
· **whos**: Lists detailed information about the variables in the workspace.
· **save**: Saves variables from the workspace to a file.
· **plot**: Creates a 2D plot of data.
· **subplot**: Divides the current figure into subplots.
· **title**: Adds a title to a plot.
· **xlabel**: Adds a label to the x-axis of a plot.
· **ylabel**: Adds a label to the y-axis of a plot.
· **max** : Returns the maximum value in the input array A.
· **mean**: Computes the mean (average) value of the elements in the input array A.
· **lookfor**: Searches the help documentation for any MATLAB function or topic containing the specified keyword.

Yash Sojitra
UI22EC72

Declaring Variables

· In MATLAB, you can declare variables by assigning a value to a variable name using the equals sign (=). MATLAB is a dynamically typed language, which means that you do not need to specify the data type of a variable when you declare it. The data type of a variable is determined automatically based on the value you assign to it.

· Eg :- x = 5;

Basic matrix operations

MATLAB is a powerful tool for working with matrices. Here are some basic matrix commands in MATLAB:

1. Creating a matrix:

```
A = [1 2 3; 4 5 6; 7 8 9];
```

This creates a 3x3 matrix A with the elements 1, 2, 3 in the first row, 4, 5, 6 in the second row, and 7, 8, 9 in the third row.

2. Accessing elements of a matrix:

```
A(2,3)
```

This returns the element in the second row and third column of matrix A, which is 6.

Yash Sojitra
UI22EC72

3. Performing matrix addition:

```
B = [4 5 6; 7 8 9; 10 11 12];
C = A + B;
```

This creates a new matrix C that is the sum of matrices A and B.

4. Performing matrix multiplication:

```
D = A * B;
```

This creates a new matrix D that is the product of matrices A and B.

5. Finding the transpose of a matrix:

```
A'
```

This returns the transpose of matrix A.

6. Finding the determinant of a matrix:

```
det(A)
```

This returns the determinant of matrix A.

7. Finding the inverse of a matrix:

```
inv(A)
```

This returns the inverse of matrix A.

LAB 3

Basic MATLAB Commands

MATLAB provides several functions for plotting and visualizing data. Here are some commonly used plotting functions and their features:

· plot(x, y, 'color'): This function plots the data in vectors x and y with the specified color. For example, plot(x, y, 'r') plots the data with a red color. Other color options include b (blue), g (green), k (black), m (magenta), c (cyan), and y (yellow).

· stem(x, y, 'color'): This function plots discrete data as vertical lines at each point with the specified color. For example, stem(x, y, 'r') plots the data with a red color.

· subplot(m, n, p): This function divides the figure window into an m-by-n grid of subplots and activates the p-th subplot for plotting. For example, subplot(2, 2, 1) creates a 2-by-2 grid of subplots and activates the top-left subplot for plotting.

· hold on: This function allows multiple plots to be displayed in the same figure window. Subsequent plots will be added to the same figure instead of replacing the previous plot.

Yash Sojitra
UI22EC72

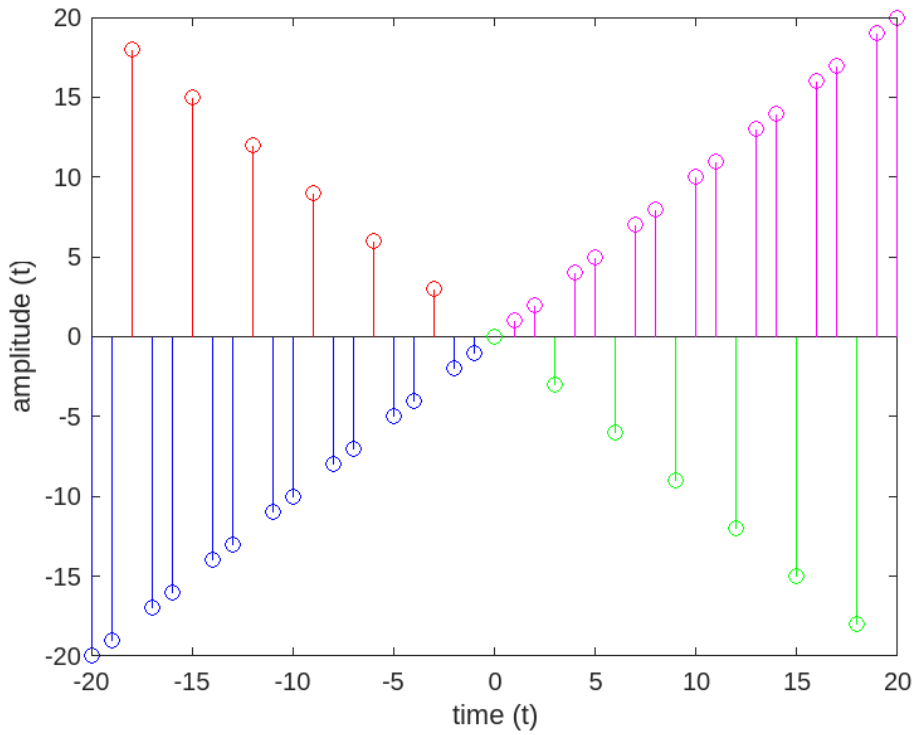
Aim :- To plot the given graph by recognizing its algorithm

Code :-

```
clc
clear all;
close all;
x=-20:20; %defining x values
y = zeros(1,40);

for i=1:length(x)
    if x(i)<0
        if mod(x(i),3)==0
            y(i)=-x(i);
            figure(1),stem(x(i),y(i),'r'),hold on
        else
            y(i)=x(i);
            figure(1),stem(x(i),y(i),'b'),hold on
        end
    else
        if mod(x(i),3)==0
            y(i)=-x(i);
            figure(1),stem(x(i),y(i),'g'),hold on
        else
            y(i)=x(i);
            figure(1),stem(x(i),y(i),'m'),hold on
        end
    end
end
xlabel("time (t)");
ylabel("amplitude (t)");
```

Output :-



Yash Sojitra
UI22EC72

LAB 4

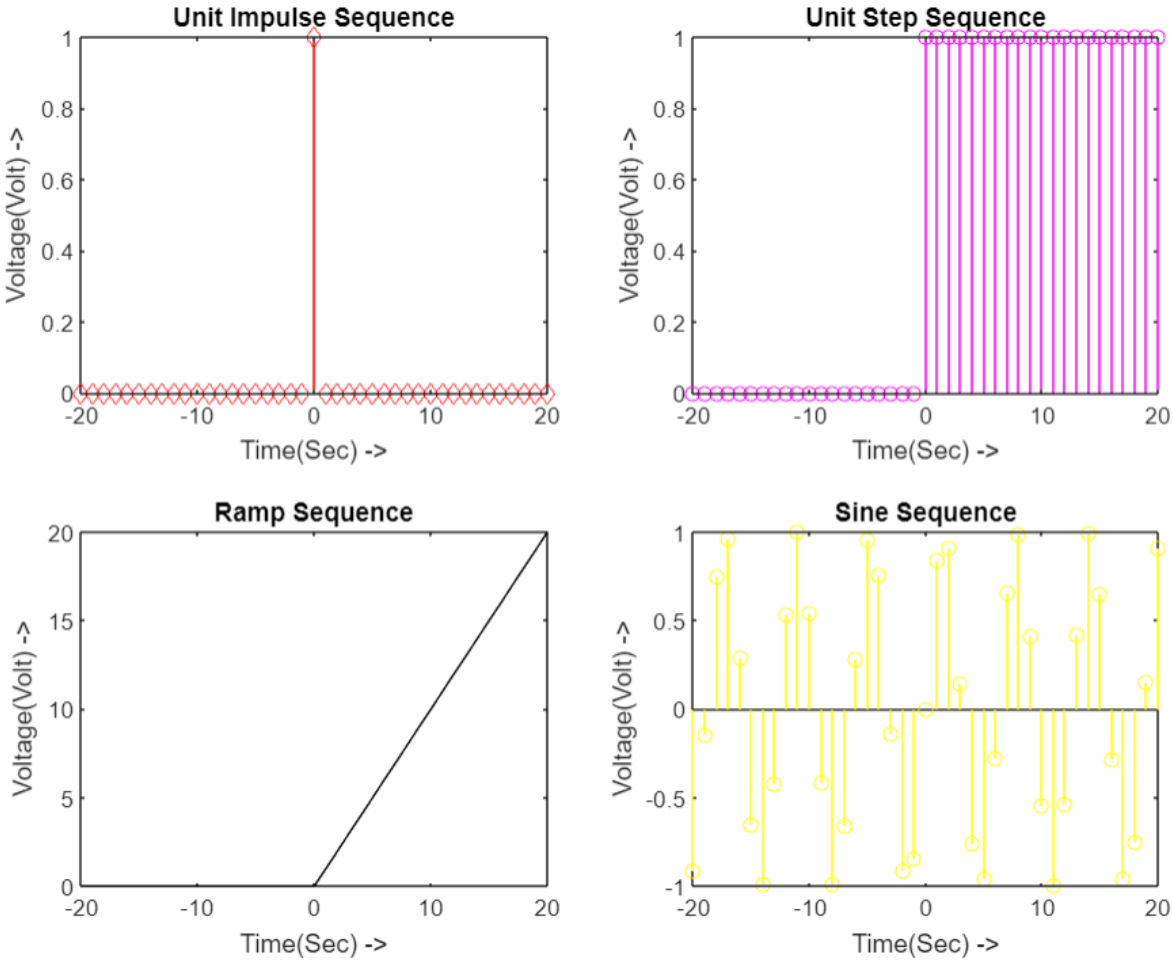
Aim :- To plot the given sub-graphs by recognizing its algorithm

Code :-

```
clc
clear
close all;
x = -20:20;
y1 = zeros(1,length(x));
y2 = zeros(1,length(x));
y3 = zeros(1,length(x));
subplot(2,2,1);
for i = 1:length(x)
    if x(i) == 0
        y1(i) = 1;
    else
        y1(i) = 0;
    end
end
stem(x,y1,'rd');
title("Unit Impulse Sequence");
xlabel('Time(Sec) ->');
ylabel('Voltage(Volt) ->');
subplot(2,2,2);
for i = 1:length(x)
    if x(i) >= 0
        y2(i) = 1;
    else
        y2(i) = 0;
    end
end
stem(x,y2,'m');
title("Unit Step Sequence");
xlabel('Time(Sec) ->');
ylabel('Voltage(Volt) ->');
subplot(2,2,3);
for i = 1:length(x)
    if x(i) >=  0
        y3(i) = x(i);
    else
        y3(i) = 0;
    end
end
plot(x,y3,'k');
title("Ramp Sequence");
xlabel('Time(Sec) ->');
ylabel('Voltage(Volt) ->');
subplot(2,2,4);
y4 = sin(x);
stem(x,y4,'y');
title("Sine Sequence");
xlabel('Time(Sec) ->');
ylabel('Voltage(Volt) ->');
```

Yash Sojitra
UI22EC72

Output :-



Yash Sojitra
UI22EC72

LAB 5

If statements and for loops in MATLAB

In MATLAB, if-else statements are used to execute a set of statements if a certain condition is true, and another set of statements if the condition is false. The general syntax for an if-else statement in MATLAB is as follows:

```
if condition
    % statements to execute if condition is true
else
    % statements to execute if condition is false
End
```

Multiple elseif statements can be used to check for multiple conditions. The general syntax for an if-elseif-else statement in MATLAB is as follows:

```
if condition1
    % statements to execute if condition1 is true
elseif condition2
    % statements to execute if condition2 is true
else
    % statements to execute if both condition1
    % and condition2 are false
End
```

Yash Sojitra
UI22EC72

In MATLAB, for loops are used to execute a set of statements a specific number of times. The general syntax for a for loop in MATLAB is as follows:

```
for i = 1:5
disp(i);
end
```

Exercise Question:

plot the following sequence h versus n at r = 0.1 and m = 4 for the n range from -25 to 25.

$$h = \frac{1}{m} + \frac{r}{m + (\frac{4}{\pi} - 1)}, if n = 0$$

$$h = \frac{r}{m} * [2\cos\left(\frac{\pi(1+r)}{4r}\right) - \cos\left(\frac{\pi(1-r)}{4r}\right)], if \ \ n = \pm\frac{m}{4r}$$

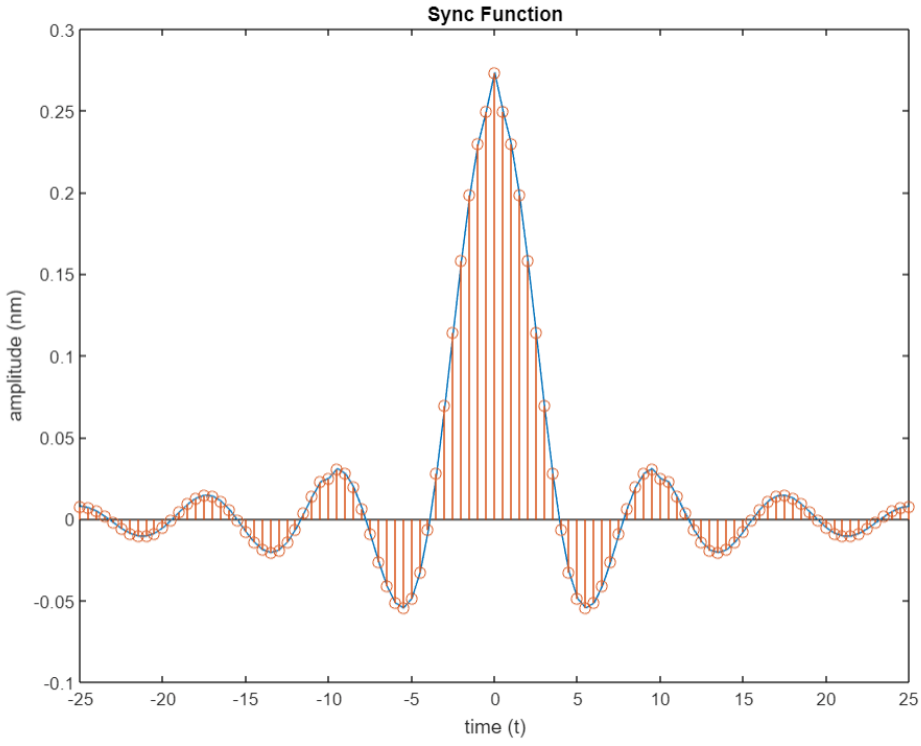
$$h = \frac{\left(4rn\cos\left(\frac{n\pi(1+r)}{m}\right)\right) + \left(m\sin\left(\frac{n\pi(1-r)}{m}\right)\right)}{\left(1 - \left(\frac{4rn}{m}\right)^2\right)nm\pi}$$

Yash Sojitra
UI22EC72

Code :-

```
clc
clear
close all;
n = -25:0.5:25;
r = 0.1;
m = 4;
mid = m/(4*r);
h = zeros(1,length(n));
for i = 1:length(n)
    if n(i)==0
        h(i) = (1/m) + (r/(m+((4/pi)-1)));
    elseif abs(n(i)) == mid
        h(i) = (r/m) * (2*cos((pi*(1+r))/4*r) - cos((pi*(1-r))/4*r));
    else
        h(i) = (4*r*n(i)*cos((n(i)*pi*(1+r)/m)) + m*sin((n(i)*pi*(1-r)/m)))
            / ((1-((4*r*n(i))/m)^2)*n(i)*m*pi);
    end
end
plot(n,h); hold on
stem(n,h);
xlabel("time (t)");
ylabel("amplitude (nm)");
title("Sync Function");
```

Output :-



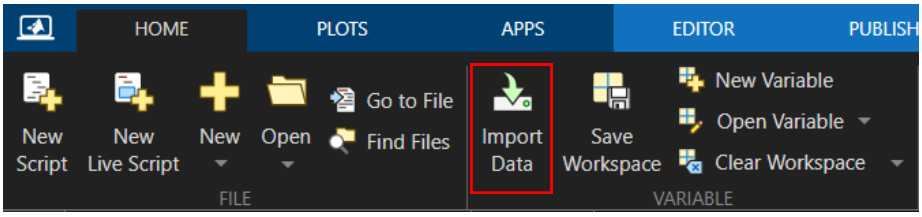
Yash Sojitra
UI22EC72

LAB 6

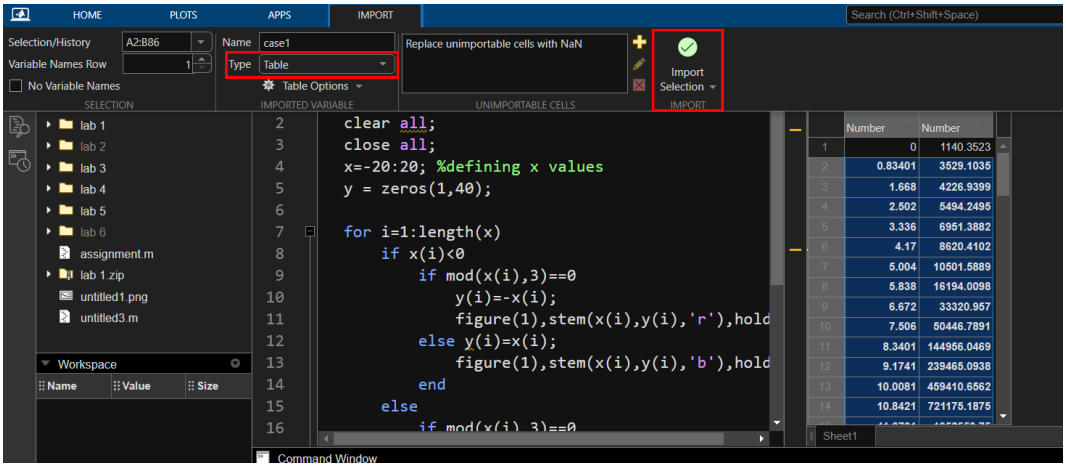
Importing Data into MATLAB:

You can import data into MATLAB via many ways, but we this is the most easiest way.

Step 1: click on import data option. select appropriate data file.



Step 2: Select how you want to import data ane import it.



Step 3: now the data will be available in the workspace

Yash Sojitra
UI22EC72

Introduction to CFTool :

The cftool is a graphical user interface (GUI) tool in MATLAB that allows you to interactively fit curves and surfaces to your data. It provides a convenient way to explore different curve-fitting techniques and customize the fitting options.

1. Launching cftool

   - Open MATLAB.
   - In the Command Window, type cftool and press Enter.
   - The cftool GUI window will open, providing an interactive environment for curve fitting.

2. Importing data

   - In the cftool window, click on the "Import Data" button.
   - Select the data file you want to use for curve fitting.
   - The data will be loaded into the cftool environment.

3. Selecting Fitting type

   - cftool supports various fitting types such as linear, exponential, polynomial, custom equations, and more.
   - Choose the desired fitting type from the drop-down menu in the "Fitting" section.

Yash Sojitra
UI22EC72

4. Adjusting the fit

    - Use the sliders and options in the "Fitting Options" section to customize the fit.
    - You can modify parameters like the degree of the polynomial, initial guesses for the fit, and constraints.

5. Plotting and analyzing the fit

    - Click the "Fit" button to apply the selected fit to the data.
    - The fitted curve or surface will be displayed on the plot along with the original data.
    - Analyze the quality of the fit by examining the goodness-of-fit metrics, residuals, and confidence intervals.

6. Exporting the fit

    - Once you are satisfied with the fit, you can export the fitted curve or surface to the MATLAB workspace.
    - Click the "Export" button and select the appropriate option to save the fit as a function or as MATLAB code.

Assignment 1:

Plot the following functions for t = -0.1 to 0.1 seconds; (given values are F_c = 10, F_m = 50)

plot the fuctions for following set of A_c and A_m.

1. A_c = 5, A_m = 8
2. A_c = 6, A_m = 6
3. A_c = 8, A_m = 5

fucnctions are:

$$M = A_m \cos(2\pi f_m t)$$

$$C = A_c \cos(2\pi f_c t)$$

$$S = [A_c + A_m \cos(2\pi f_m t)] \cos(2\pi f_c t)$$

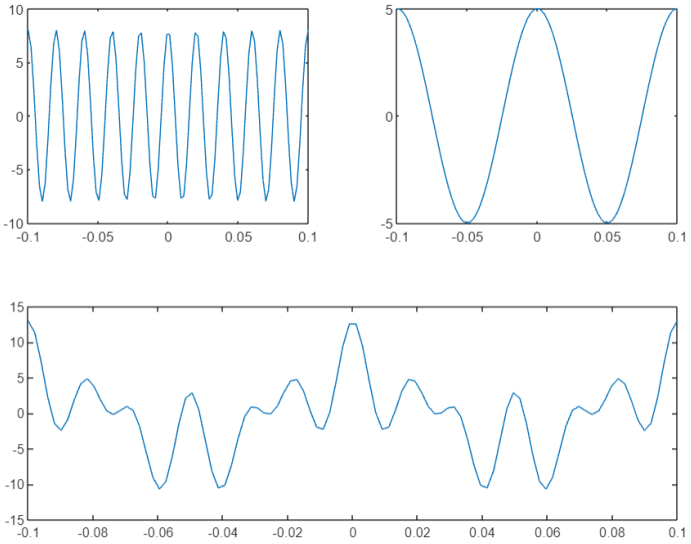
Code:

```
clc
close all;
Ac = 6; %(5,6,8)
Am = 6; %(8,6,5)
fc = 10;
fm = 50;
t = linspace(-0.1,0.1);
M = Am*cos(2*pi*fm*t);
C= Ac*cos(2*pi*fc*t);
S=(Ac+Am*cos(2*pi*fm*t)).*cos(2*pi*fc*t);
subplot(2,2,1);
plot(t, M);
subplot(2,2,2);
plot(t, C);
subplot(2,2,[3,4]);
plot(t, S);
```

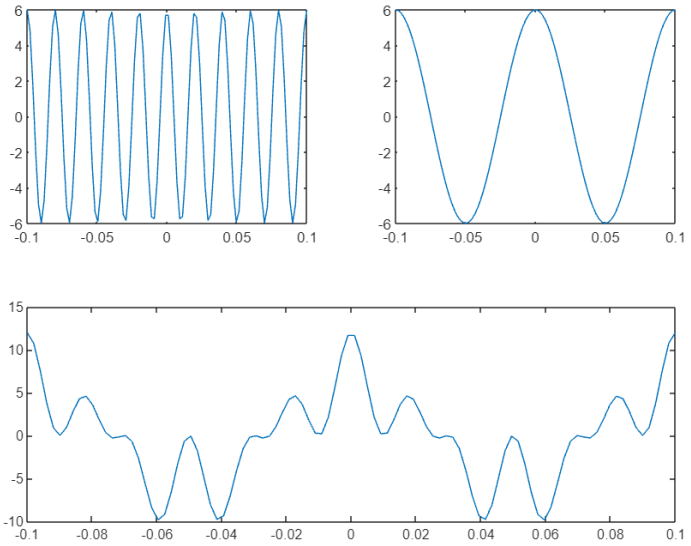
Yash Sojitra
UI22EC72

Output:

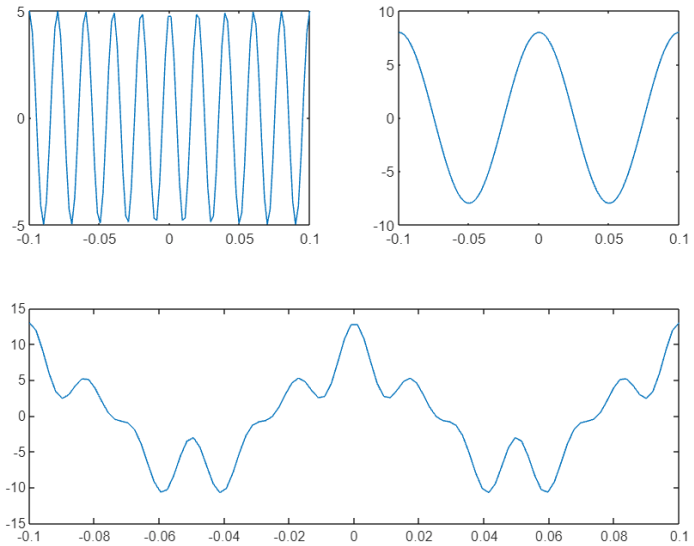
1.



2.



3.

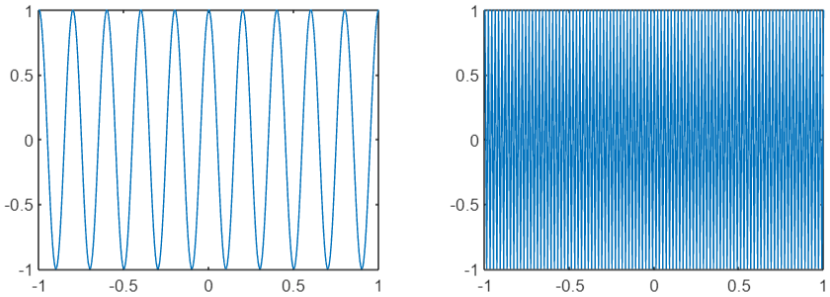


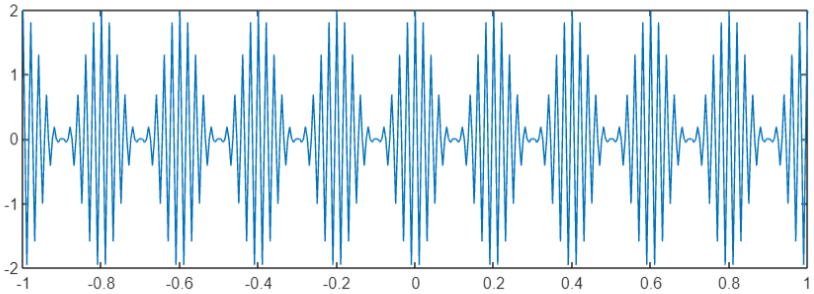
Yash Sojitra
UI22EC72

Assignment 2 :

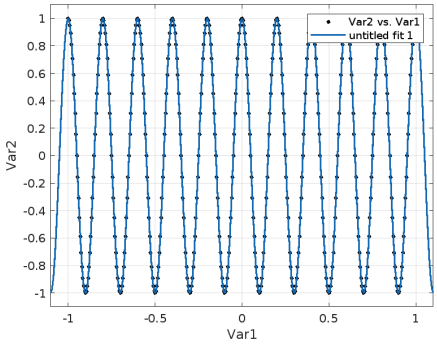
Fit the following the data and also compare the given data plot with plot obtained from CFTOOL.

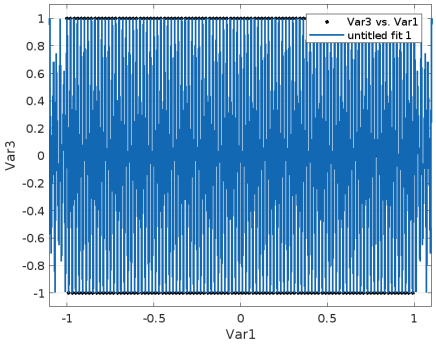
Imported data plot :

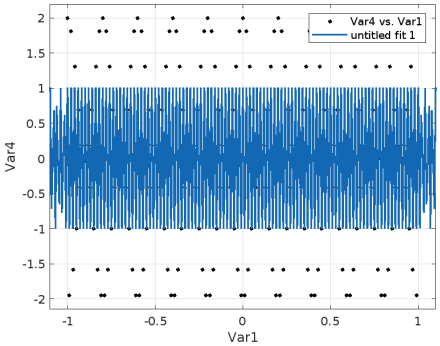




Curve-Fitter data plot:







Yash Sojitra
UI22EC72