Homework 1 with Solution:
Lab 0: MATLAB Basics:

Task 1:

Use the subplot command to provide following three plots in one figure:

- Plot 1: sin(t), sin(3t), sin(5t)
- Plot 2: cos(t), cos(3t), cos(5t)
- Plot 3: tan(t), tan(3t), tan(5t)

For $t\epsilon[-1, 1]$. It is important to use different color schemes to differentiate the harmonics; provide legend, xlabel, ylabel, and title for each plot.

Solution

```
clc; close all; clear;
t = -1:0.001:1;
y11 = sin(t);  y12 = sin(3*t);  y13 = sin(5*t);
y21 = cos(t);  y22 = cos(3*t);  y23 = cos(5*t);
y31 = tan(t);  y32 = tan(3*t);  y33 = tan(5*t);
figure;
subplot(3,1,1);
plot(t, y11, t, y12, t, y13);
xlabel('time, t');
ylabel('Amplitude, y');
title('Sine Plots');
legend('sin(t)', 'sin(3t)', 'sin(5t)');

subplot(3,1,2);
plot(t, y21, t, y22, t, y23);
xlabel('time, t');
ylabel('Amplitude, y');
title('Cosine Plots');
legend('cos(t)', 'cos(3t)', 'cos(5t)');

subplot(3,1,3);
plot(t, y31, t, y32, t, y33);
xlabel('time, t');
ylabel('Amplitude, y');
title('Tan Plots');
legend('tan(t)', 'tan(3t)', 'tan(5t)');
```

Task 2:

Write a MATLAB function, which takes the following two inputs:

A vector (of variable length) representing resistors in an electrical circuit A flag representing whether resistors are in parallel or series The function should return equivalent resistance.

Solution:

```

function Req = EquivalentResistance(R, inParallel)

if inParallel ~=0
    Req = sum(R);
else
    Req = 1/(sum(1./R));
end
end
```

Task 3:

Define the symbolic variable x. Make use of this variable to define the symbolic function sin(1/x). Then, plot the function from 0 to 2. What do you see? Why might the plotting routine be having trouble plotting this function?

Solution:

Plotting symbolic function with plot function is not possible, because, plot function is defined for numerical values. However, plot can be used when we substitute numerical value like:

```
clc; close all; clear;
syms x
f(x) = sin(1/x);
x = 0.001:0.001:2;
y = double(subs(f));
plot(x, y)
```

Task 4:

The natural response of a series RC circuit is given by:

$$v(t) = V_0 e^{-\frac{t}{RC}}$$

Write a MATLAB function that takes two inputs R and C, and plots the natural response of this circuit for $t \epsilon [0,2]$.

Solution:

```
function naturalResponseRC(R, C)

V0 = 10;
t = 0:0.01:2;
v = V0*exp(-t/(R*C));
plot(v,t);
xlabel('time, t (sec)');
ylabel('Voltage, v (volts)');
title('Natural Response of RC Circuit');

end
```

Lab. 1 Task with Solution:
Signals & Systems Properties

Task 1:

Accumulator or summer is a discrete time system with memory, given by:

$$y[n] = \sum_{k=-\infty}^{n} x[k]$$

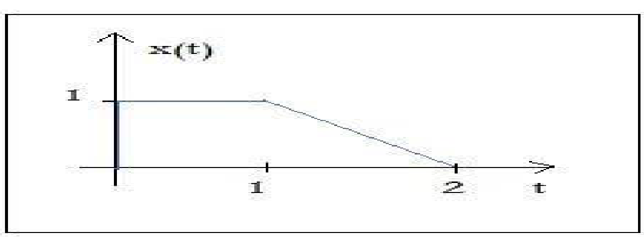
Write a MATLAB Script for the accumulator.

Solution

```
clc; close all; clear;

x = [1 2 3 4 5 4 3 2 1];
y = zeros(size(x));
for n = 1:length(x)
    y(n) = sum(x(1:n));
end

figure;
subplot(2,1,1);
stem(x);
xlabel('n');
ylabel('x[n]');
title('Input to Accumulator');

subplot(2,1,2);
stem(y);
xlabel('n');
ylabel('y[n]');
title('Output of Accumulator');
```

Task 2:

Consider the following figure:



Plot the following in MATLAB:

a) $x(t+1)$
b) $x(-t+1)$
c) $x\left(\frac{3}{2}t\right)$
d) $x\left(\frac{3}{2}t-1\right)$

Solution:

```
clc; close all; clear;

t = 0:0.01:2;
x = 1*(t<=1) + (2-t).*(t>1);

figure;
plot(t, x);
xlabel('t');
ylabel('x');
title('Original, x(t)')

figure;
plot(t-1, x);
xlabel('t');
ylabel('x');
title('x(t+1)')

figure;
plot(-t+1, x);
xlabel('t');
ylabel('x');
title('x(-t+1)')

figure;
plot(t/(3/2), x);
xlabel('t');
ylabel('x');
title('x(3t/2)')

figure;
plot(t/(3/2)+1, x);
xlabel('t');
ylabel('x');
title('x(3t/2 - 1)')

```

Task 3:

Analyze each of the following continuous-time systems in terms of the six basic system properties. In addition, summarize your conclusions in the table below:

a) $S_1(t) = x(t-2) + x(2-t)$
b) $S_2(t) = \cos(3t)x(t)$
c) $S_3(t) = \begin{cases} 0 & t < 0 \\ x_1(t) + x(t-2) & t \geq 0 \end{cases}$

Solution:

Systems	**Memory**	**Causality**	**Invertibility**	**Stability**	**Time invariance**	**Linearity**
S_1	Yes	No	Yes	No	Yes	Yes
S_2	No	Yes	Yes	Yes	No	No
S_3	Yes	Yes	No	Yes	Yes	No

Task 4:

Record your own sound saying the sentence, “The quick brown fox jumps over the lazy dog”. Play this recording in MATLAB with and without time reversal, also comment on the observations.

Solution:

```
clc; close all; clear;

recObj = audiorecorder

disp('Start speaking.');
recordblocking(recObj, 5);
disp('End of Recording.');

play(recObj);

y = getaudiodata(recObj);

subplot(2,1,1)
plot(y);
title('Voice without Time Reversal')

subplot(2,1,2)
```

```
plot(y(end:-1:1));
title('Voice with Time Reversal')
```

Lab. 2 Tasks with Solution: Signals, Periodicity and Harmonics

Task 1:

Compute Odd and Even part of a step function for $n\epsilon[-5,5]$. A step function is given by:

$$u[n] = \begin{cases} 0 & n < 0 \\ 1 & n \geq 0 \end{cases}$$

Soution:

```
clc; close all; clear;

n = -5:5;
u = double(n>=0);

u_ = u(end:-1:1);

u_even = 1/2*(u + u_);
u_odd = 1/2*(u - u_);

figure;
subplot(3,1,1)
stem(n, u);
xlabel('n');
ylabel('u');
title('Step Function');

subplot(3,1,2)
stem(n, u_even);
xlabel('n');
ylabel('u_even');
title('Even Component of Step Function');

subplot(3,1,3)
stem(n, u_odd);
xlabel('n');
ylabel('u_odd');
title('Odd Component of Step Function');
```

Task 2:

Plot 5 odd harmonics of cosine for $t\epsilon[-10,10]$ given by:

$$x(t) = \frac{4}{pi} \times \left\{ \cos t - \frac{1}{3}\cos 3t + \frac{1}{5}\cos 5t - \frac{1}{7}\cos 7t + \frac{1}{9}\cos 9t \right\}$$

Solution:

```
clc; close all; clear;

t = -10:0.01:10;
y1 = cos(t);
y2 = -1/3*cos(3*t);
y3 = 1/5*cos(5*t);
y4 = -1/7*cos(7*t);
y5 = 1/9*cos(9*t);

y = (y1+y2+y3+y4+y5)*4/pi;
plot(t,y);
xlabel('t');
ylabel('y');
title('5 Odd Harmonics of Cosine');
```

Homework 2 Task with Solution

Homework 2:

Plot 100 Odd harmonics of Cosine using:

$$x(t) = \frac{4}{pi}\sum_{k=1}^{\infty}\frac{(-1)^{k+1}\cos\big((2k-1)t\big)}{2k-1}$$

For $t\epsilon[-10, 10]$

Solution:

```
clc; close all; clear;

t = -10:0.01:10;
y = zeros(size(t));
for k = 1:100
    y = y + (-1)^(k+1)*cos((2*k-1)*t)/(2*k - 1);
end
plot(t,y);
xlabel('t');
ylabel('y');
title('Sum of 100 Odd Harmonics of Cosine');
```

Lab 3 Task with Solution: Graphical User Interface (GUI)

Task 1:

Design a MATLAB GUI that takes two numbers and prints the sum when a pushbutton is pressed. Give meaningful names and tags to GUI component.

Solution

In the following code, add_Callback() function (line: 138-148) correspond to the pushbutton with Tag 'add'. 'handles.n1' and 'handles.n2' correspond to the edit fields while 'handles.sum' corresponds the static text field where final sum is shown to the user.

Please consider the highlighted piece of code.

```
1  function varargout = myAdder(varargin)
2  % MYADDER MATLAB code for myAdder.fig
3  %      MYADDER, by itself, creates a new MYADDER or raises the
4  existing
5  %      singleton*.
6  %
7  %      H = MYADDER returns the handle to a new MYADDER or the handle
8  to
9  %      the existing singleton*.
10 %
11 %      MYADDER('CALLBACK',hObject,eventData,handles,...) calls the
12 local
13 %      function named CALLBACK in MYADDER.M with the given input
14 arguments.
15 %
16 %      MYADDER('Property','Value',...) creates a new MYADDER or raises
17 the
18 %      existing singleton*.  Starting from the left, property value
19 pairs are
20 %      applied to the GUI before myAdder_OpeningFcn gets called.  An
21 %      unrecognized property name or invalid value makes property
22 application
23 %      stop.  All inputs are passed to myAdder_OpeningFcn via
24 varargin.
25 %
```

```
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help myAdder

% Last Modified by GUIDE v2.5 04-Nov-2021 03:12:22

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @myAdder_OpeningFcn, ...
                   'gui_OutputFcn',  @myAdder_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before myAdder is made visible.
function myAdder_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to myAdder (see VARARGIN)

% Choose default command line output for myAdder
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

```

```
% UIWAIT makes myAdder wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = myAdder_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function n1_Callback(hObject, eventdata, handles)
% hObject    handle to n1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n1 as text
%        str2double(get(hObject,'String')) returns contents of n1 as a
double


% --- Executes during object creation, after setting all properties.
function n1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function n2_Callback(hObject, eventdata, handles)
% hObject    handle to n2 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n2 as text
%        str2double(get(hObject,'String')) returns contents of n2 as a
double


% --- Executes during object creation, after setting all properties.
function n2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in add.
function add_Callback(hObject, eventdata, handles)
% hObject    handle to add (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
n1 = str2double(get(handles.n1, 'String'));
n2 = str2double(get(handles.n2, 'String'));
disp(n1)
disp(n2)
sm = n1+n2;
set(handles.sum, 'String', num2str(sm))
```

Task 2:

Write a MATLAB Code that takes n and m from the user and plots n numbers of Odd Cosine Harmonics for the time 0 to m seconds when pushbutton 'PLOT' is pressed.

n: No. of Harmonics

m: No. of Seconds

Solution:

Fs

```
function varargout = myHarmonicPlot(varargin)
% MYHARMONICPLOT MATLAB code for myHarmonicPlot.fig
%      MYHARMONICPLOT, by itself, creates a new MYHARMONICPLOT or
raises the existing
%      singleton*.
%
%      H = MYHARMONICPLOT returns the handle to a new MYHARMONICPLOT
or the handle to
%      the existing singleton*.
%
%      MYHARMONICPLOT('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in MYHARMONICPLOT.M with the given
input arguments.
%
%      MYHARMONICPLOT('Property','Value',...) creates a new
MYHARMONICPLOT or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before myHarmonicPlot_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to myHarmonicPlot_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help myHarmonicPlot

```

```
% Last Modified by GUIDE v2.5 04-Nov-2021 04:04:10

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @myHarmonicPlot_OpeningFcn, ...
                   'gui_OutputFcn',  @myHarmonicPlot_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before myHarmonicPlot is made visible.
function myHarmonicPlot_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to myHarmonicPlot (see VARARGIN)

% Choose default command line output for myHarmonicPlot
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes myHarmonicPlot wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = myHarmonicPlot_OutputFcn(hObject, eventdata,
handles)
```

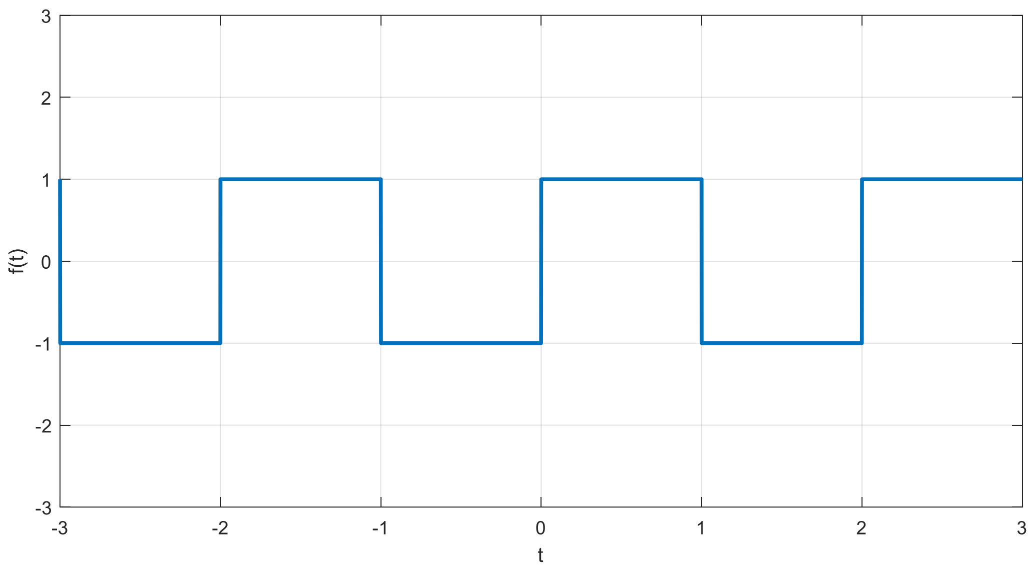
```
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function m_Callback(hObject, eventdata, handles)
% hObject    handle to m (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of m as text
%        str2double(get(hObject,'String')) returns contents of m as a
double


% --- Executes during object creation, after setting all properties.
function m_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function n_Callback(hObject, eventdata, handles)
% hObject    handle to n (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n as text
%        str2double(get(hObject,'String')) returns contents of n as a
double
```

```
% --- Executes during object creation, after setting all properties.
function n_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in plot.
function plot_Callback(hObject, eventdata, handles)
% hObject    handle to plot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
n = str2double(get(handles.n, 'String'));
m = str2double(get(handles.m, 'String'));

t = 0:0.01:m;
y = zeros(size(t));

for k = 1:n
    y = y + (-1)^(k+1)*cos((2*k-1)*t)/(2*k - 1);
end
plot(handles.axes1,t,y);
xlabel(handles.axes1,'t');
ylabel(handles.axes1,'y');
```

Lab 4 Tasks & Solution:
Fourier Series

Task 1:

Plot f(t) for 100 harmonics using Fourier Series:



where

$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n \cos(nw_0 t) + b_n \sin(nw_0 t)$$

$$a_0 = \int_{-\frac{T_0}{2}}^{+\frac{T_0}{2}} f(t)dt$$

$$a_n = \int_{-\frac{T_0}{2}}^{+\frac{T_0}{2}} f(t)\cos(nw_0 t)\,dt$$

$$b_n = \int_{-\frac{T_0}{2}}^{+\frac{T_0}{2}} f(t)\sin(nw_0 t)\,dt$$

$$w_0 = \frac{2\pi}{T_0}$$

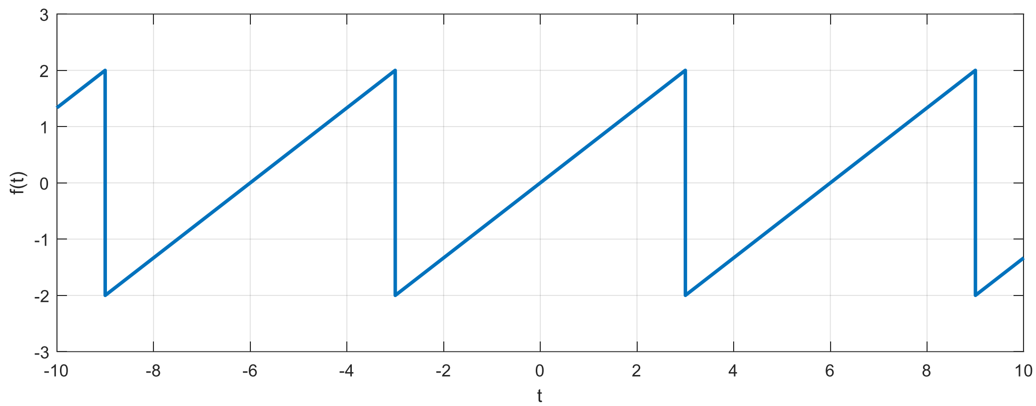
Solution:

In this case, we have $T_0 = 2 \;\Rightarrow\; w_0 = \pi$. Consider MATLAB code below:

```
clc; close all; clear;

T0 = 2;
w0 = 2*pi/T0;

syms f1(t) f2(t) n t;

f1(t) = -1;
f2(t) =  1;
a0 = int(f1, [-T0/2 0]) + int(f2, [0 T0/2]);
an = int(f1*cos(n*w0*t), [-T0/2 0]) + int(f2*cos(n*w0*t), [0 T0/2]);
bn = int(f1*sin(n*w0*t), [-T0/2 0]) + int(f2*sin(n*w0*t), [0 T0/2]);

% syms is only used for calculating integral.
% converting from syms to numerical, as:

t = -5:0.001:5;
n = 1:100;

an = double(subs(an));
bn = double(subs(bn));

if isscalar(an)
    an = an*ones(size(t));
end
if isscalar(bn)
    bn=bn*ones(size(t));
end


y = zeros(size(t));

for idx = n
    y = y + an(idx)*cos(idx*w0*t) + bn(idx)*sin(idx*w0*t);
end
y = a0 + y;

plot(t, y);
xlabel('t');
ylabel('f(t)');
title('Sum of First 100 Harmonics of f(t)');
```

Task 2:

Plot f(t) for 100 harmonics using Fourier Series:



$$f(t) = \frac{2}{3}t \quad \text{for} \quad -3 < t \leq 3$$

Solution:

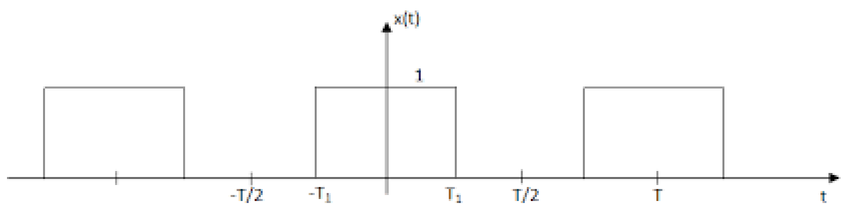
```
clc; close all; clear;

T0 = 6;
w0 = 2*pi/T0;

syms f(t) n t;

f(t) = 2/3*t;
a0 = 1/T0*int(f, [-T0/2 T0/2]);
an = 2/T0*int(f*cos(n*w0*t), [-T0/2 T0/2]);
bn = 2/T0*int(f*sin(n*w0*t), [-T0/2 T0/2]);

% syms is only used for calculating integral.
% converting from syms to numerical, as:

t = -5:0.001:5;
n = 1:100;

an = double(subs(an));
bn = double(subs(bn));

if isscalar(an)
```

```
    an = an*ones(size(t));
end
if isscalar(bn)
    bn=bn*ones(size(t));
end


y = zeros(size(t));

for idx = n
    y = y + an(idx)*cos(idx*w0*t) + bn(idx)*sin(idx*w0*t);
end
y = a0 + y;

plot(t, y);
xlabel('t');
ylabel('f(t)');
title('Sum of First 100 Harmonics of f(t)');
```

Lab 5 Tasks & Solution:
Fourier Series & Fourier Transform:

Task 1:

Find the effect of time period on Fourier Series Coefficients a_0, a_n & b_n for the following figure having $T_1 = 5$ and observe the effect when $T_0 \to \infty$.



Solution:

Suppose we observe 5th harmonic:

```
clc; close all; clear;

T1 = 5;
syms f(t) n t T0;
f(t) = 1;
a0 = 1/T0*int(f, [-T1 T1]);
an = 2/T0*int(f*cos(n*2*pi*t/T0), [-T1 T1]);
bn = 2/T0*int(f*sin(n*2*pi*t/T0), [-T1 T1]);

% for fifth harmonic, put = 5
n = 5;

T0 = T1+0.01:0.01:50;
plot(T0, subs(an));
xlabel('T_0');
ylabel('a_n');
title('a_n vs T_0');

figure, plot(T0, subs(a0));
xlabel('T_0');
ylabel('a_0');
title('a_0 vs T_0');

figure, plot(T0, subs(bn));
xlabel('T_0');
```

```
ylabel('b_n');
title('b_n vs T_0');
```

Task 2:

Write Fourier Transform (Magnitude and Phase plots) of the following signal for the following discrete time signal.

i. $x[n] = [1,2,3,3,2,1]$
ii. $x(t) = e^{-2t}u(t)$ for $-10 \leq t \leq 10$

Solution:

```
clc; close all; clear;
% Part i:
x = [1 2 3 3 2 1];
X = fftshift(fft(x));
figure, subplot(121)
plot(abs(X))
xlabel('DFT Samples')
ylabel('Magnitude')
title('Magnitude plot')
subplot(122)
plot(angle(X))
xlabel('DFT Samples')
ylabel('Phase (Radians)')
title('Phase plot')
% Part ii:
t = -10:0.01:10;
x = exp(-2*t);
x(t<=0) = 0;
X = fftshift(fft(x));
figure, subplot(121)
plot(abs(X))
xlabel('DFT Samples')
ylabel('Magnitude')
title('Magnitude plot')
subplot(122)
plot(angle(X))
xlabel('DFT Samples')
ylabel('Phase (Radians)')
title('Phase plot')
```

Lab 6 Tasks & Solution:
Fourier Transform:

Task 1:

Record 'Hello World' speech, evaluate and plot speech signal in frequency domain and then find the range of human voice frequency.

Note: Convert the audio into monophonic if it is a stereo sound.

Solution:

```
clc; close all; clear;

[y, fs] = audioread('myfile.mp4');

% Check if the audio is stereophonic, then convert into monophonic
If size(y,2) == 2
    Y = mean(y,2);
end

Y = fftshift(fft(t));
f = fs*linspace(-0.5,0.5,length(Y));
% Convert the frequency vector into column vector
f = f(:)';
figure, subplot(121)
plot(f, abs(Y))
xlabel('frequency (Hz)')
ylabel('Magnitude')
title('Magnitude plot')
subplot(122)
plot(f, angle(Y))
xlabel('frequency (Hz)')
ylabel('Phase (Radians)')
title('Phase plot')
```

Task 2:

Do Task 1 for normalized frequency i.e. range $[-\pi, \pi]$ rad/sample.

Solution:

```
clc; close all; clear;
[y, fs] = audioread('myfile.mp4');

% Check if the audio is stereophonic, then convert into monophonic
```

```
If size(y,2) == 2
    Y = mean(y,2);
end

Y = fftshift(fft(t));
f = fs*linspace(-pi, pi, length(Y));
% Convert the frequency vector into column vector
f = f(:)';
figure, subplot(121)
plot(f, abs(Y))
xlabel('frequency (rad/sample)')
ylabel('Magnitude')
title('Magnitude plot')
subplot(122)
plot(f, angle(Y))
xlabel('frequency (rad/sample)')
ylabel('Phase (Radians)')
title('Phase plot')
```

Lab 7 Tasks & Solution:
Nyquist's Criterion & Aliasing:

Task 1:

Define a time vector for a sinusoid, keep changing the sampling frequency and observe the spectrum for Nyquist's Criterion violation that results aliasing.

Solution:

```
clc; close all; clear;
f = 10;  % in Hz
fs = 80; % in samples/sec
t = 0:1/fs:10;
x = sin(2*pi*f*t);
X = fftshift(fft(x));
f = fs*linspace(-0.5,0.5,length(X))
figure, plot(f, abs(X));
xlabel('f (Hz)');
ylabel('Magnitude');
f = 10;  % in Hz
fs = 30; % in samples/sec
t = 0:1/fs:10;
x = sin(2*pi*f*t);
X = fftshift(fft(x));
f = fs*linspace(-0.5,0.5,length(X))
figure, plot(f, abs(X));
xlabel('f (Hz)');
ylabel('Magnitude');
f = 10;  % in Hz
fs = 15; % in samples/sec
t = 0:1/fs:10;
x = sin(2*pi*f*t);
X = fftshift(fft(x));
f = fs*linspace(-0.5,0.5,length(X))
figure, plot(f, abs(X));
xlabel('f (Hz)');
ylabel('Magnitude');
% in Figure 3, folding/aliasing in the spectrum is obvious
% as it gives peak on 5 Hz instead of 10Hz
```

Task 2:

Read a sound file and store the sampling rate f_s, convert into monophonic if required, keep every fourth sample in time domain. After doing so, the sampling rate becomes $\frac{1}{4}f_s$.

Play the original sound with sampling rate f_s and the resampled sound with sampling rate $\frac{1}{4}f_s$ and notice the difference. Also, visualize the magnitude spectrum plot and see if there is aliasing effect.

Solution:

```
clc; close all; clear;
load handel.mat;
yres = y(1:4:end);
Y = fftshift(fft(y));
Yres = fftshift(fft(yres));
f = Fs*linspace(-0.5,0.5, length(Y));
fres = Fs*linspace(-0.5,0.5, length(Yres))/4;

f = f(:)';
fres = fres(:)';

sound(y, Fs);
pause(9);
sound(yres, Fs/4)

figure, plot(f, abs(Y))
xlabel('freq (Hz)');
ylabel('Magnitude');
title('Original - Magnitude Plot');

figure, plot(fres, abs(Yres))
xlabel('freq (Hz)');
ylabel('Magnitude');
title('Resampled - Magnitude Plot');

% Magnitude plot are different as well the heard sounds are also
different,
% so aliasing is detected
```

Lab 8 Tasks & Solution:

Images & Sounds:

Task1:

Read the default image 'peppers.png' and display the RGB channels in its respective color.

Solution:

```
clc; close all; clear all;

img = imread('peppers.png');
img = double(img)/255;

imgR = zeros(size(img));
imgG = zeros(size(img));
imgB = zeros(size(img));

imgR(:,:,1) = img(:,:,1);
imgG(:,:,2) = img(:,:,2);
imgB(:,:,3) = img(:,:,3);

figure, subplot(221);
imshow(img);
title('Original');

subplot(222), imshow(imgR);
title('Red Channel');

subplot(223), imshow(imgG);
title('Green Channel');

subplot(224), imshow(imgB);
title('Blue Channel');
```

Task 2:

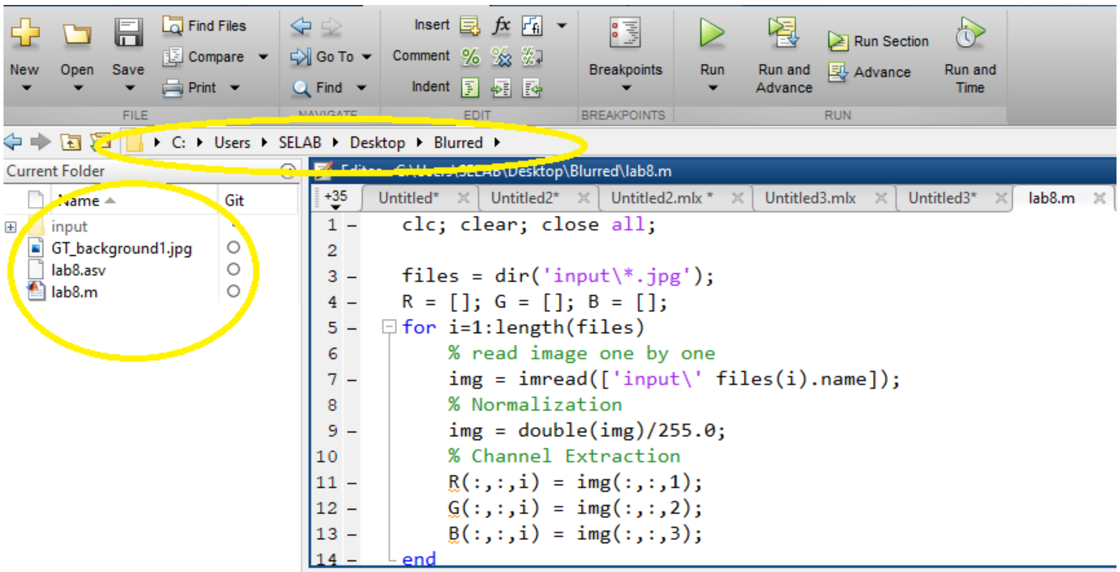
You need to implement Temporal Average Filter (TAF) using the provided images in the directory. The provided directory having n images having $p \times q$ pixels each image.

- Read the directory of the jpg files in MATLAB.
- Read and normalize each image to the range [0,1].
- Extract R, G and B channels and concatenate each channel into a separate array.
- Find the mean of n channels corresponding to R,G and B.

- Concatenate the means to obtain RGB image, and show it.

Solution:

Set MATLAB path according to the following snapshot and put the provided data in the current directory accordingly.




```
clc; clear; close all;
files = dir('input\*.jpg');
R = []; G = []; B = [];
for i=1:length(files)
    % read image one by one
    img = imread(['input\' files(i).name]);
    % Normalization
    img = double(img)/255.0;
    % Channel Extraction
    R(:,:,i) = img(:,:,1);
    G(:,:,i) = img(:,:,2);
    B(:,:,i) = img(:,:,3);
end
Rmean = mean(R, 3); Gmean = mean(G, 3); Bmean = mean(B, 3);
imgMean = cat(3, Rmean, Gmean, Bmean);
imshow(imgMean);
title('The Mean of all images');

% In the mean image you will hardly observed the moving objects,
% TAF is used for Background Estimation in image processing..
% The data is taken from Scene Background Challenge – 2017 – observe
% each variable size in workspace.
```