5/20/2019

# Final Project Report

## ECTE 363: COMMUNICATION SYSTEMS


**Group B:**

**Umm Kulsoom Emad 5529657**

**Jean Xavier 5567828**

**Ummer Sheriff 5600376**

# Table of Contents

# Introduction

This report is for the ECTE363 Final Project. The project focuses on creating a GUI in which the user enters amplitude, frequency and phase for message signals and carrier signals. This report will show the test results of all the modulation techniques required.

# Explanations and reasoning of ‘coding lines and testing’

- This line of code takes values from the GUI text box whose tag name can be set and stored in a variable.
  ```
  variable=str2num(get(handles.tagname,'string'));
  ```

- The graphs are plotted on the above-mentioned code for the x axis
  The sampling frequency is 6000, therefore, the maximum frequency that the user can enter can be up to 2999. Any value above it will cause a distortion in the graph.
  ```
  fs = 6000; %sampling frequency
  t = 0:1/fs:1-1/fs;
  ```

- To use variables from one push button to other push buttons
  ```
  handles.variable=variable;
  guidata(hObject,handles);
  ```

- To retrieve variables from other push buttons in a push button
  ```
  variable=handles.variable;
  ```

- am= amplitude of the message signal
  fm= frequency of the message signal
  ac= amplitude of the carrier signal
  fc= frequency of the carrier signal

- For different modulation techniques, different values were chosen for the amplitude and frequency of the message and carrier signals so that the graphs are displayed in most accurate way. Also, because some modulation techniques are supposed to have a specific frequency difference between the carrier and message signals.

- The BASK, BPSK, BFSK need the amplitudes of both carrier and message signal to be same therefore any one of those amplitude could be used in the code.

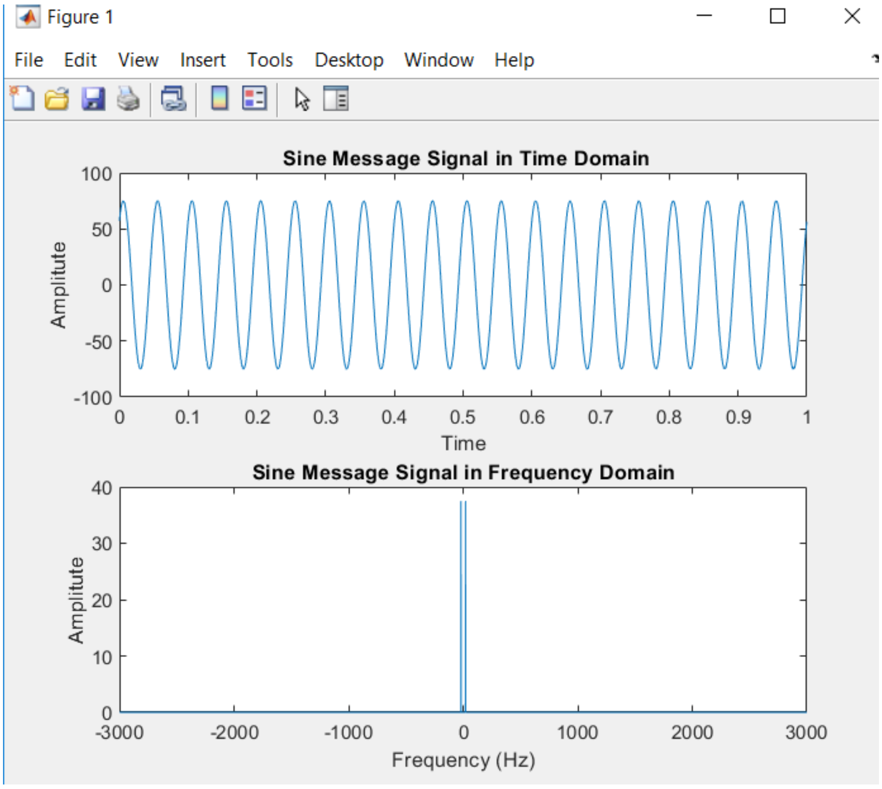
## Part 1 and Part 2

These parts require the user to enter amplitude, frequency and phase and choose a type of message signal

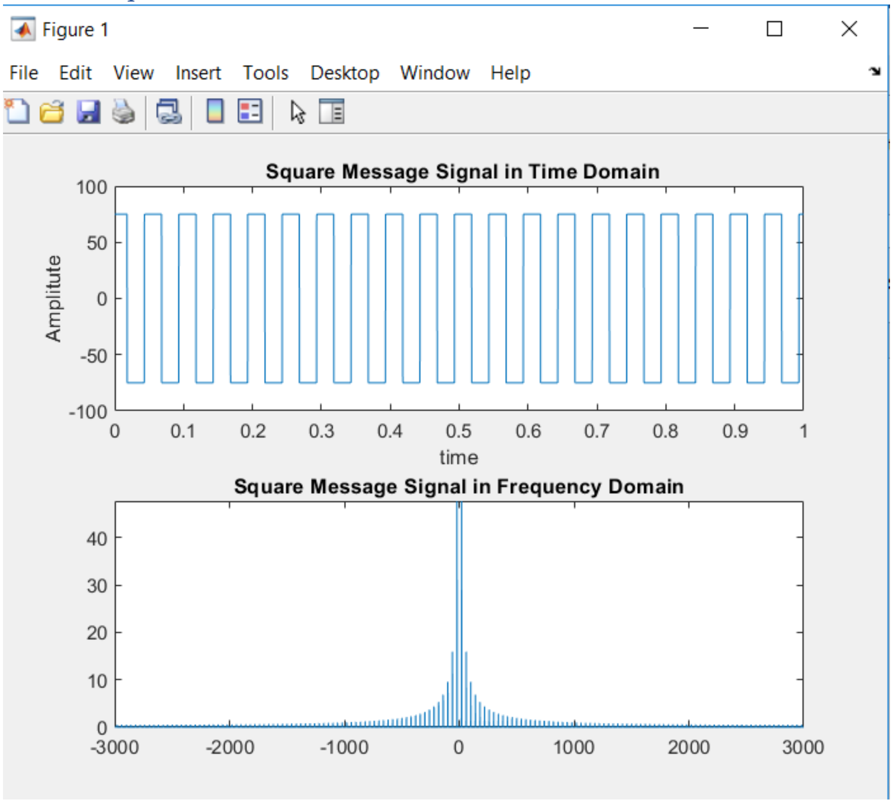
The following inputs were entered in the GUI for all message signals

Amplitude=75, frequency=20, phase=50

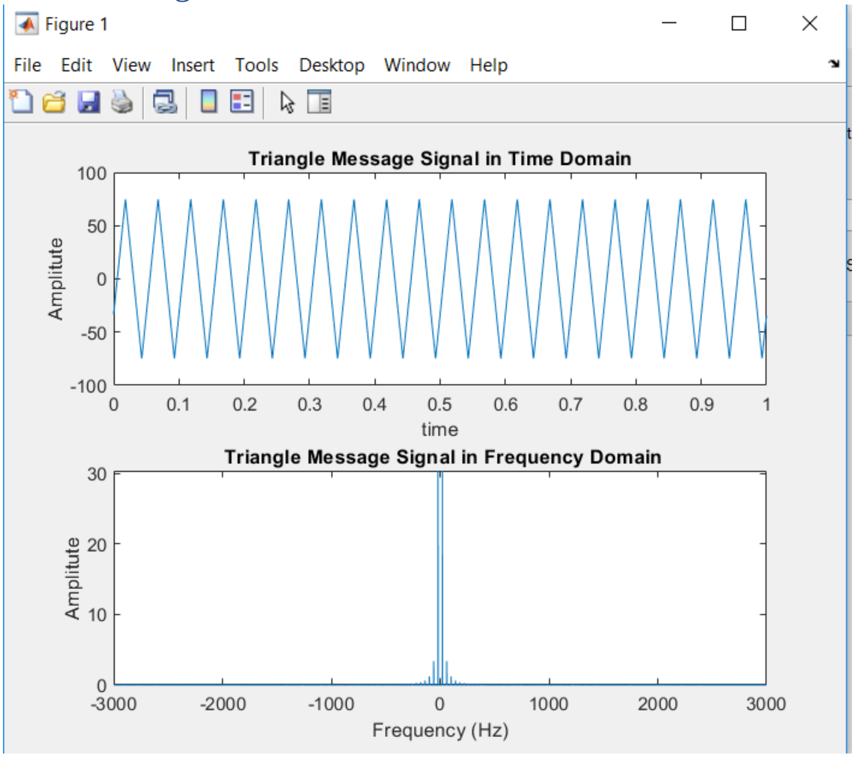
### 1. Sine Wave



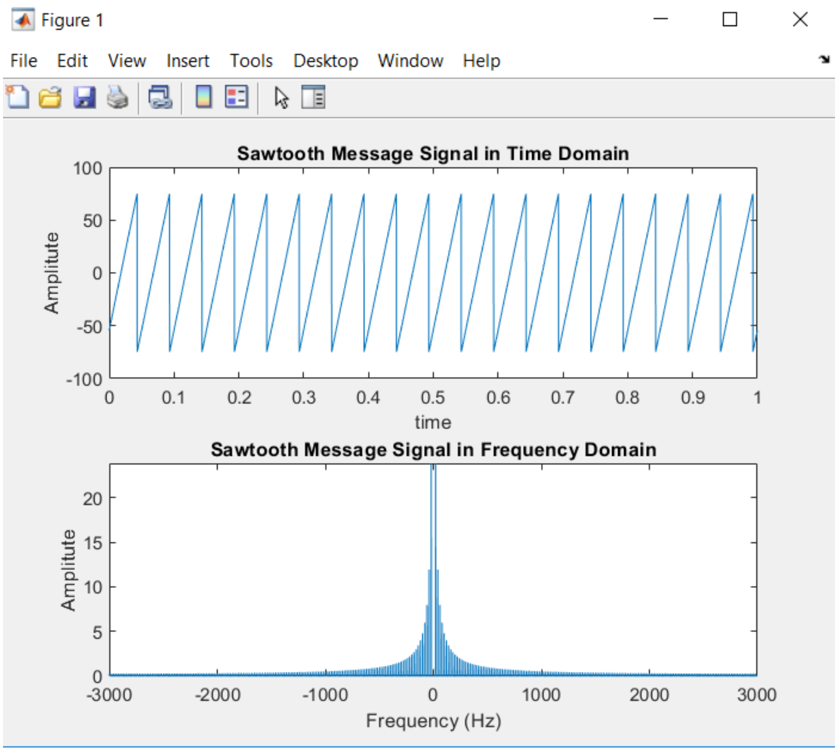
## 2. Square Wave




## 3. Triangle Wave



## 4. Sawtooth Wave

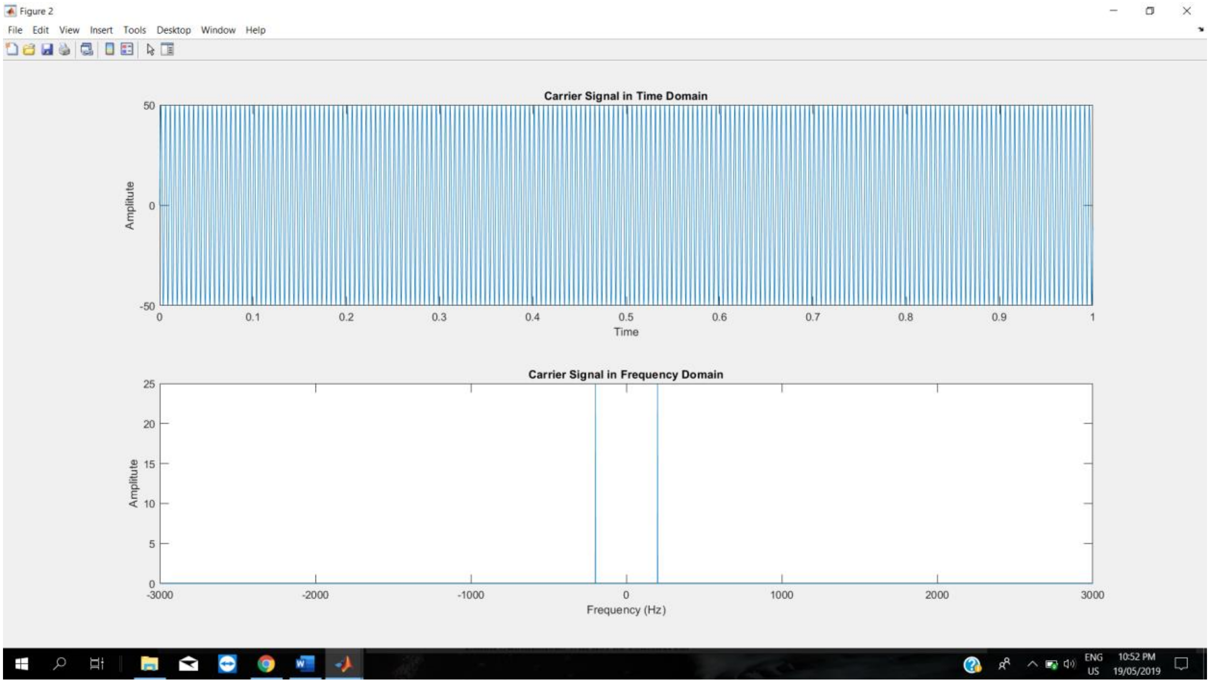



# Part 3

## Carrier Signal

This part requires to choose the amplitude, frequency and phase of the carrier signal, the carrier signal is always sine wave.

Ac=20, fc=200, phase=0



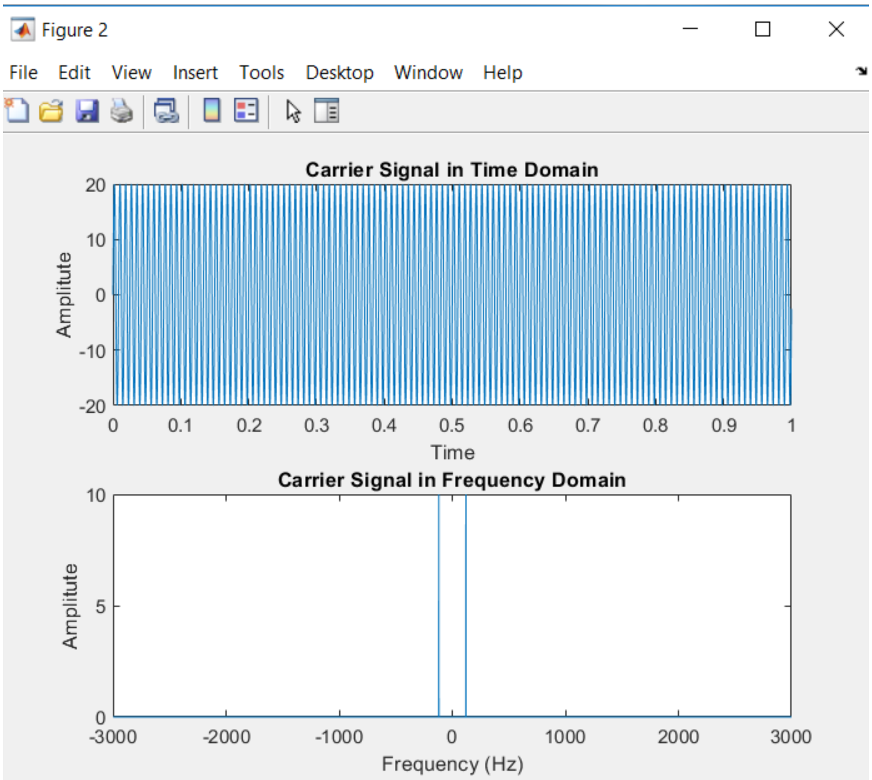
# Part 4

This part is for determining different types of AM modulation, FM modulation and PM modulation to be used.

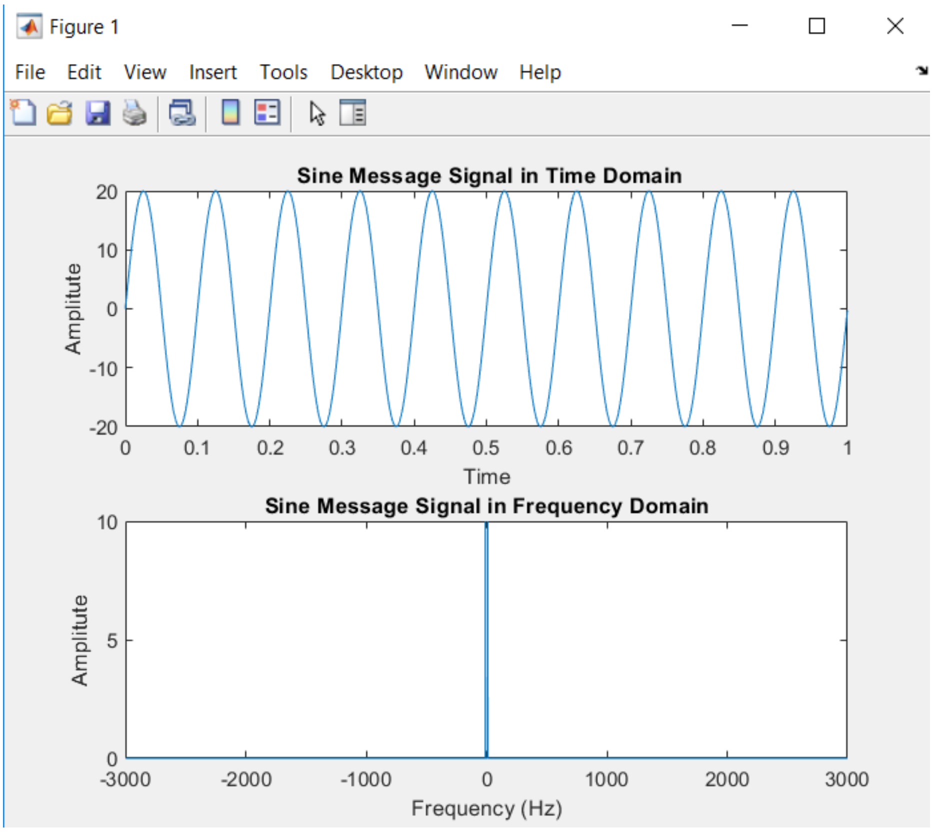
## 4.1 Modulated AM Signal

The following inputs were entered in the GUI for all message signals
Am= 20, fm=10, ac=20, fc=120

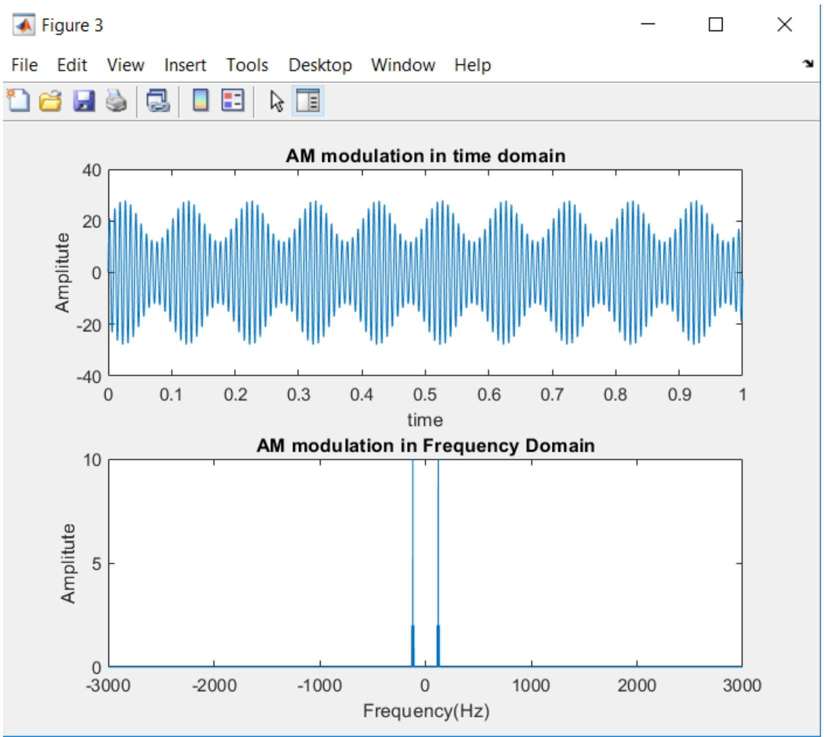
The carrier wave is same for all type of messages with the details mentioned above and figure is below



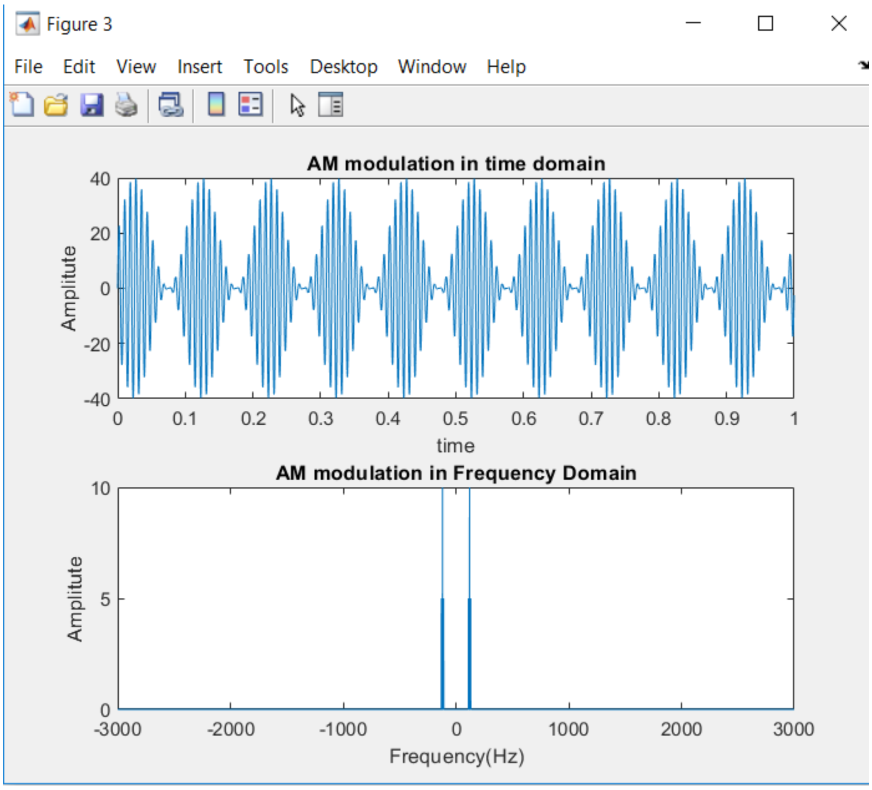
a. Sine Wave



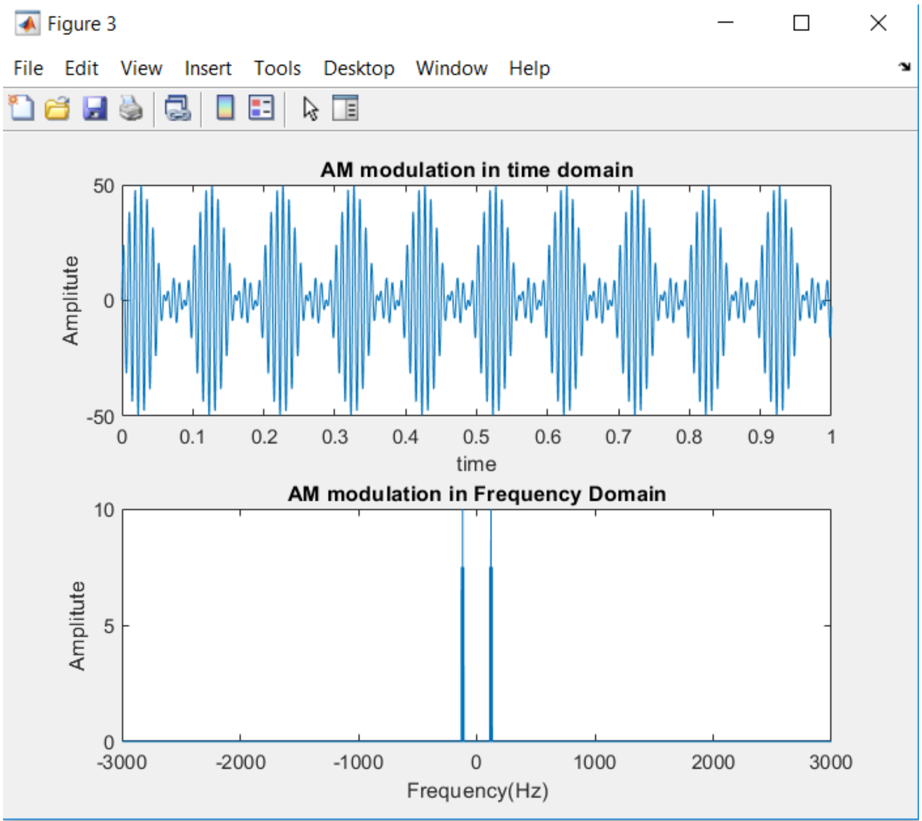

Am modulation when modulation index is 0.4



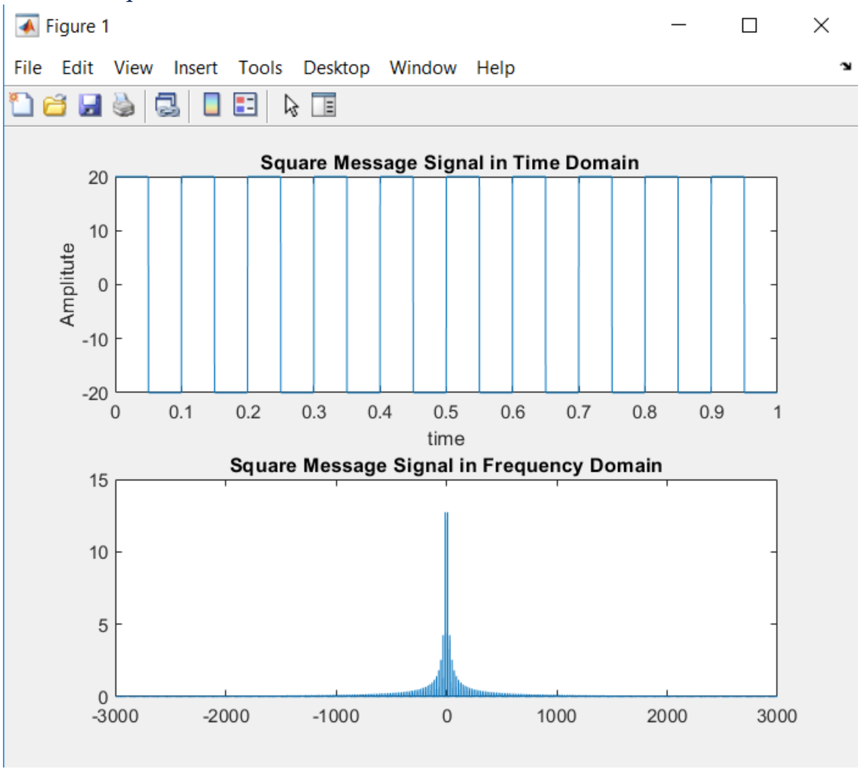
Am modulation when modulation index is 1



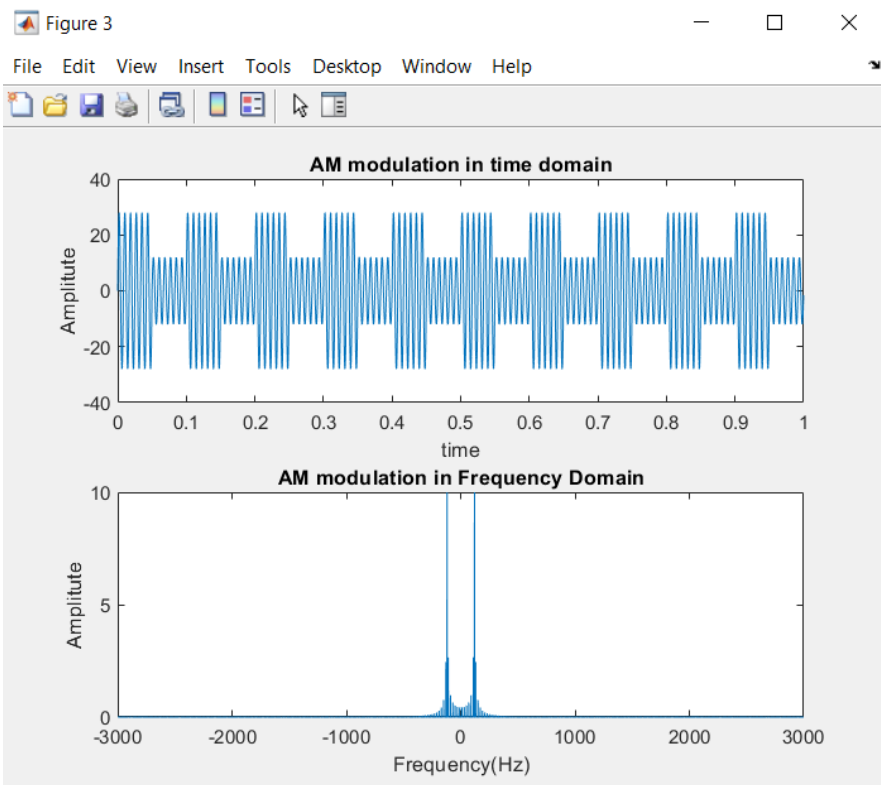

Am modulation when modulation index is 1.5



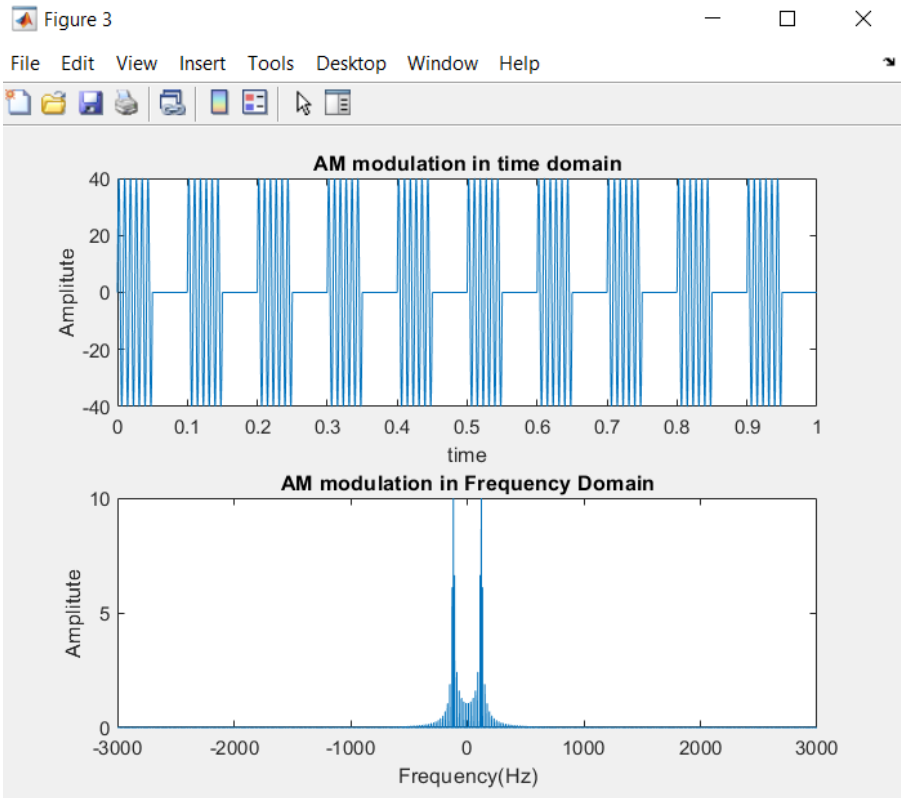
b. Square Wave



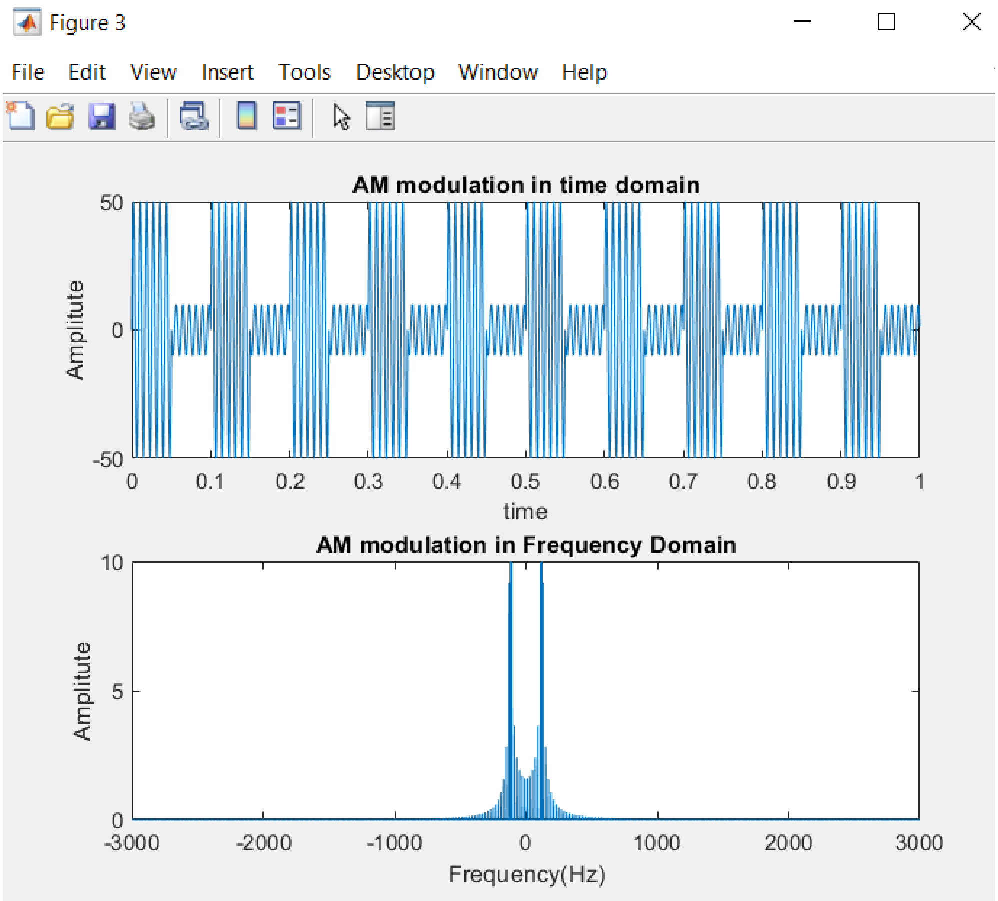

Am modulation when modulation index is 0.4



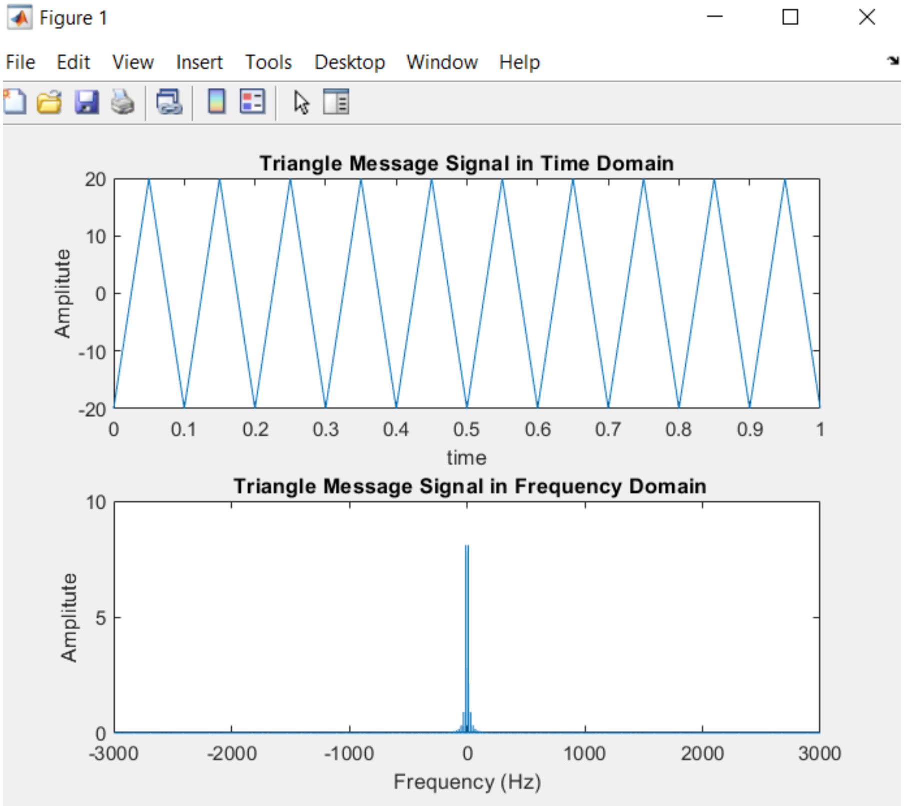
Am modulation when modulation index is 1



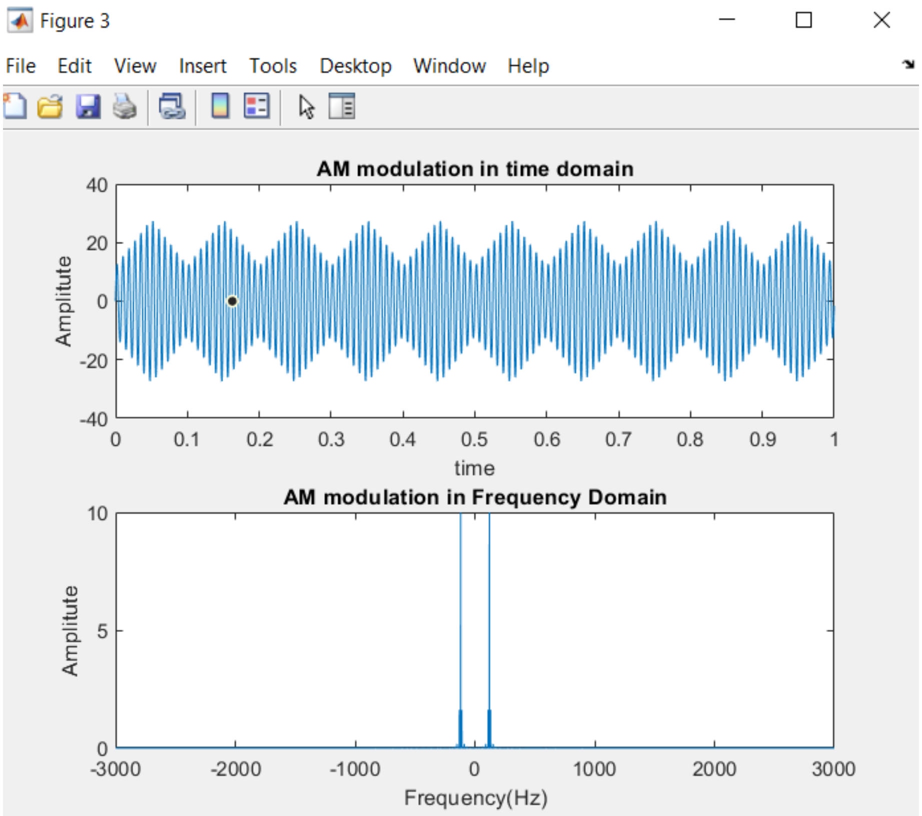

Am modulation when modulation index is 1.5



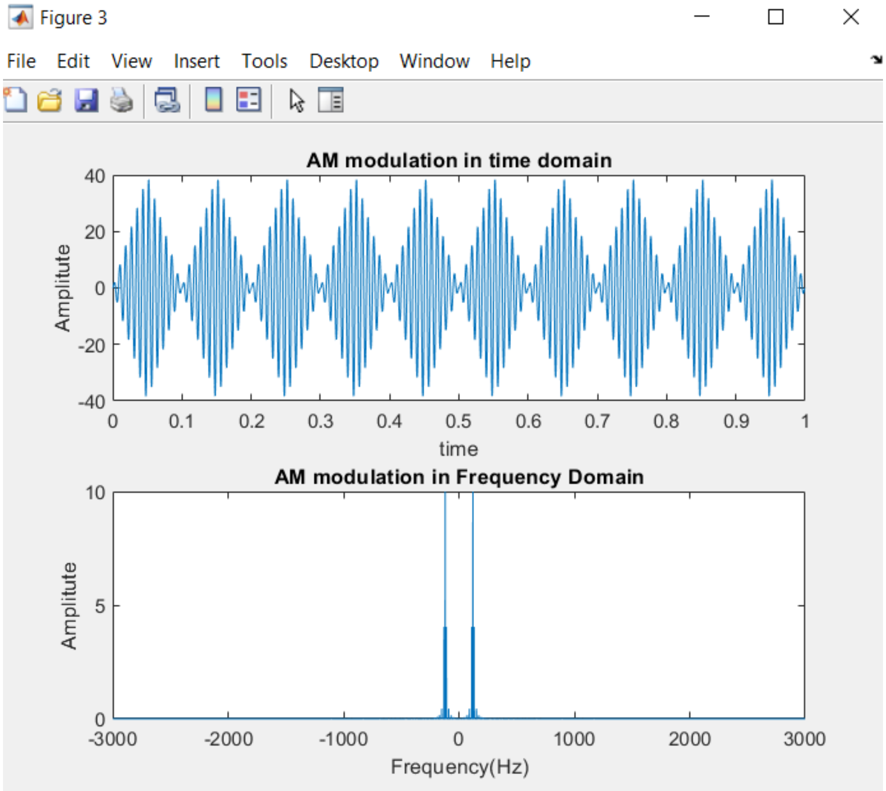
c. Triangle Wave



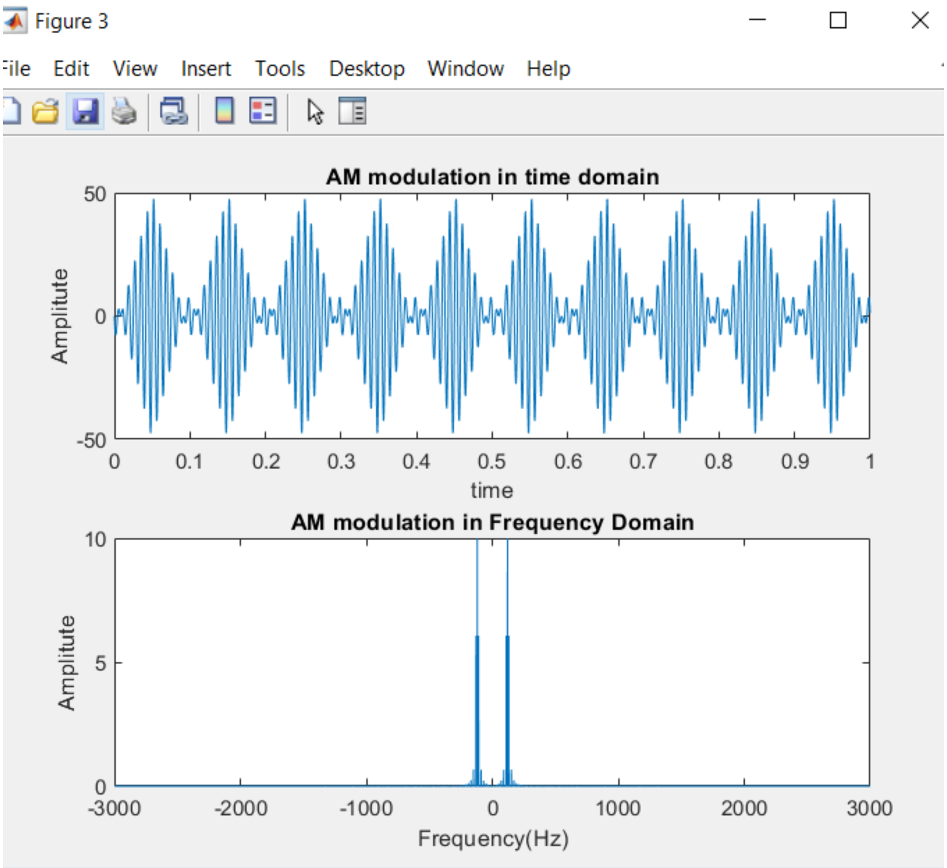

Am modulation when modulation index is 0.4



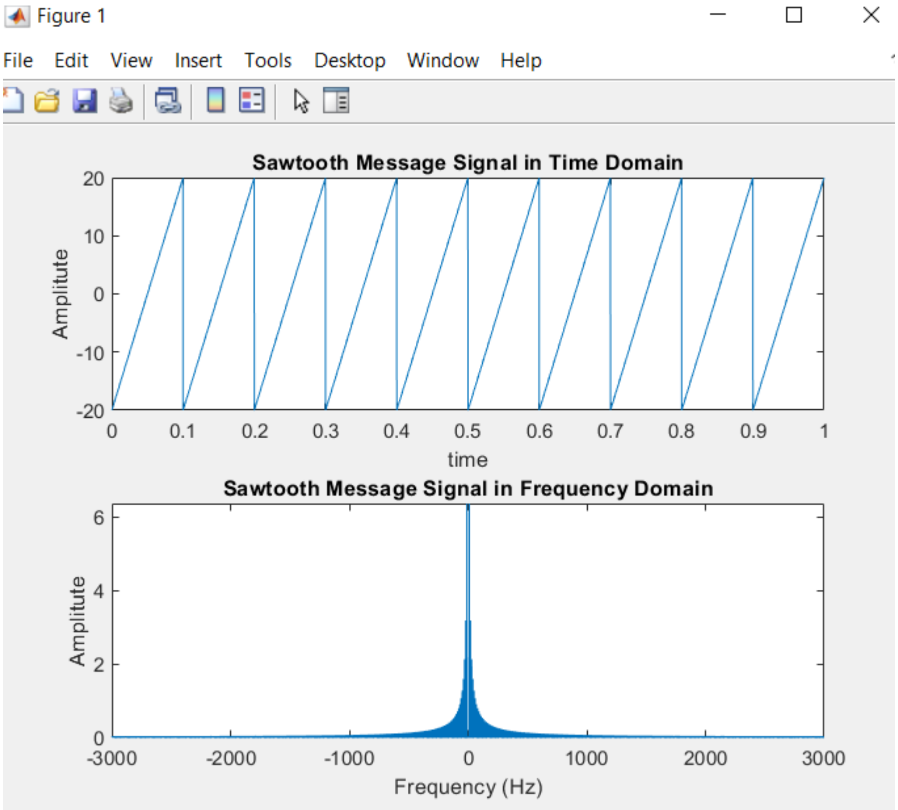
Am modulation when modulation index is 1



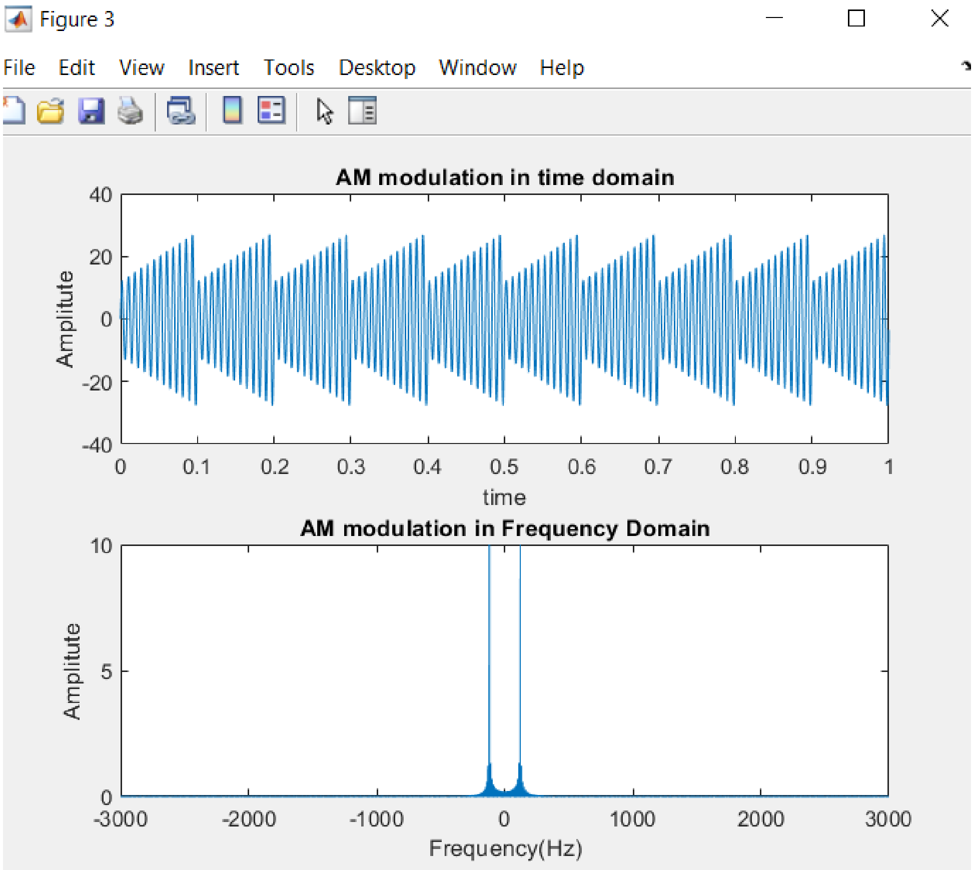

Am modulation when modulation index is 1.5



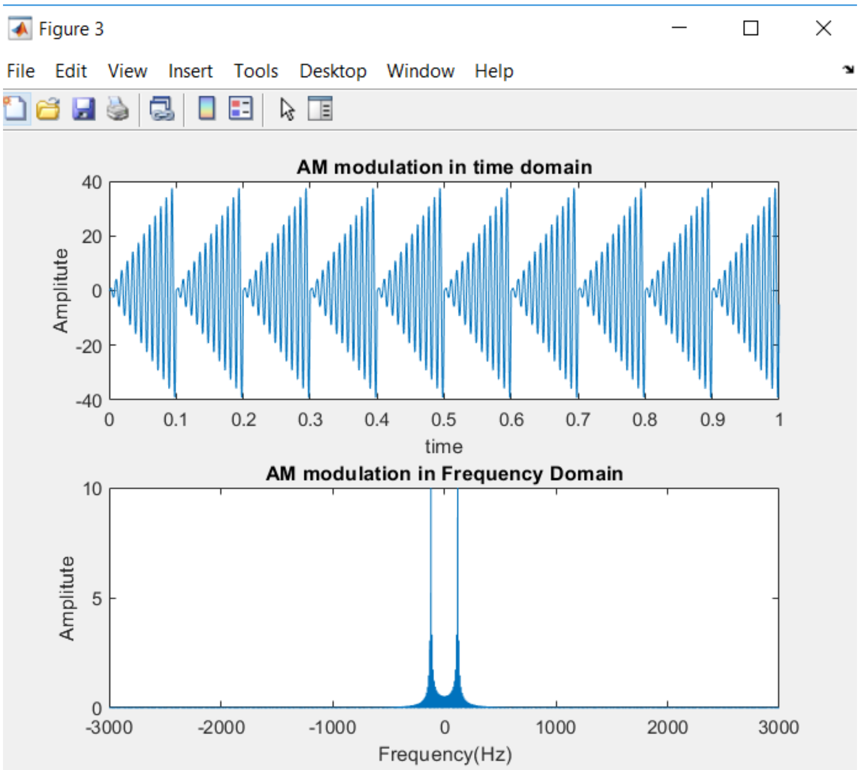
d. Sawtooth Wave



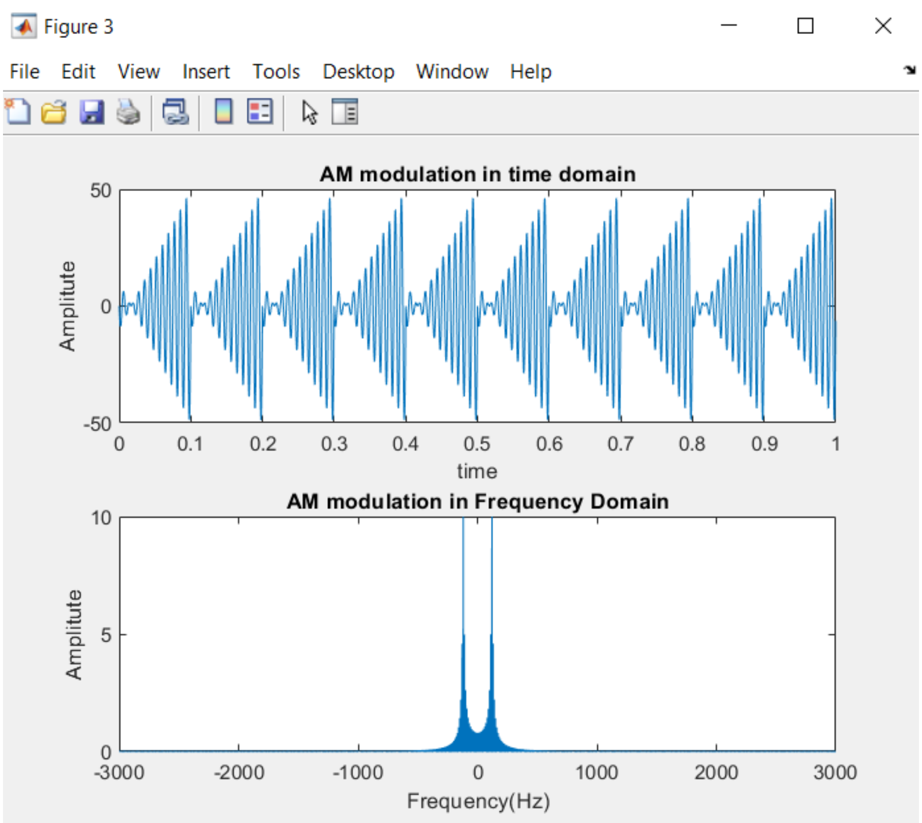

Am modulation when modulation index is 0.4



Am modulation when modulation index is 1



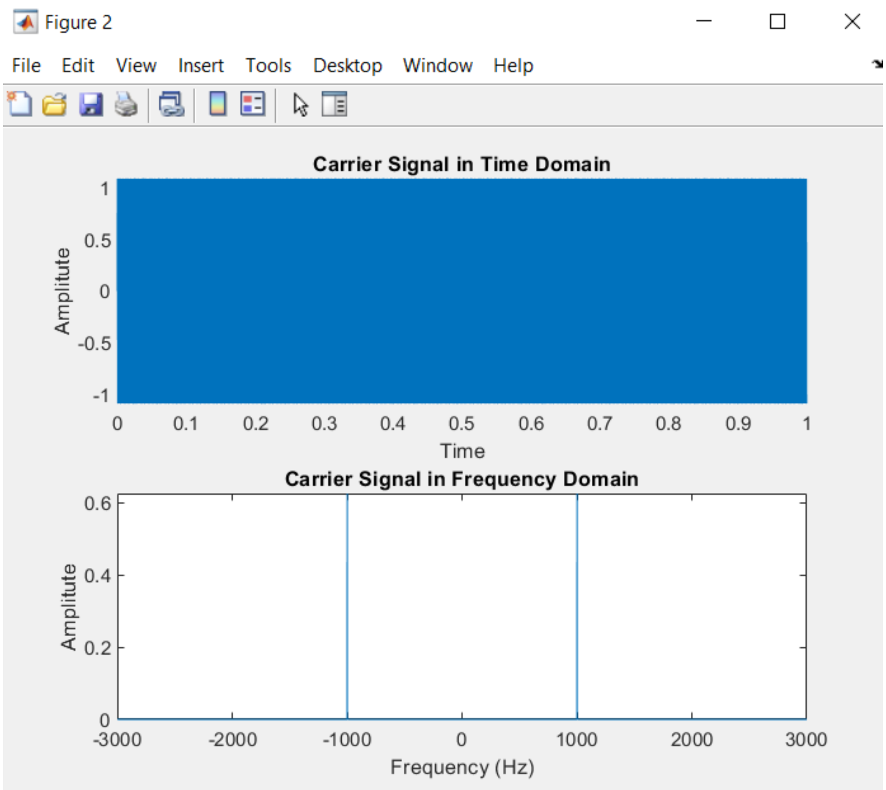

Am modulation when modulation index is 1.5



## 4.2 Double Sideband Full Carrier

The following inputs were entered in the GUI for all message signals
Am= 1, fm=50, ac=1.25, fc=1000 and modulation index to be 0.8

The carrier signal is same for all message types which is shown in the figure below.




The message signal along with their modulation are shown by parts

a. Sine Wave

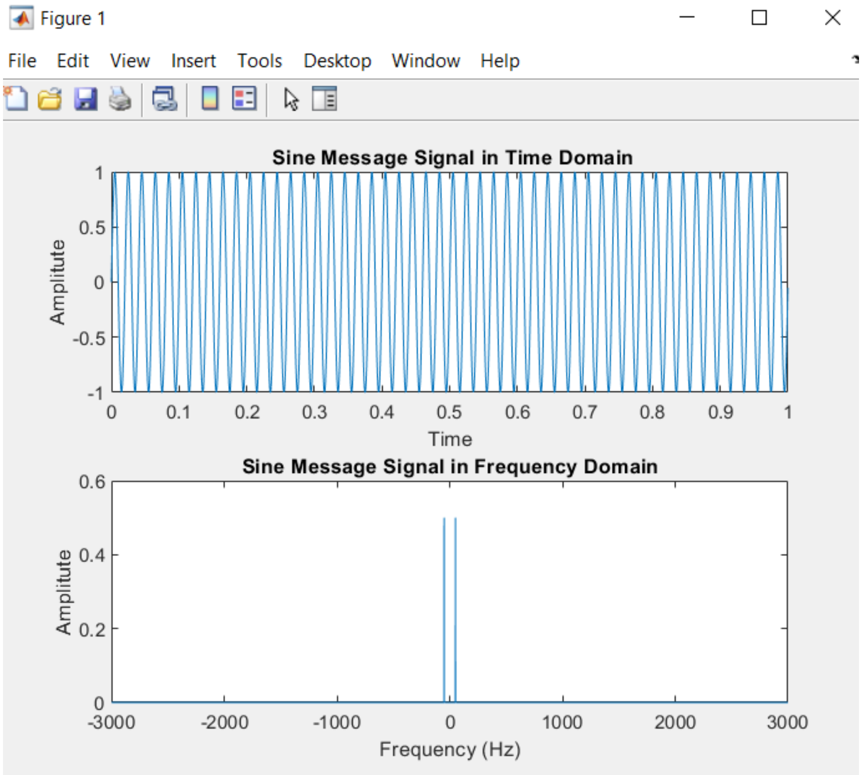


Figure 1
File Edit View Insert Tools Desktop Window Help
Sine Message Signal in Time Domain
Amplitute
Time
Sine Message Signal in Frequency Domain
Amplitute
Frequency (Hz)

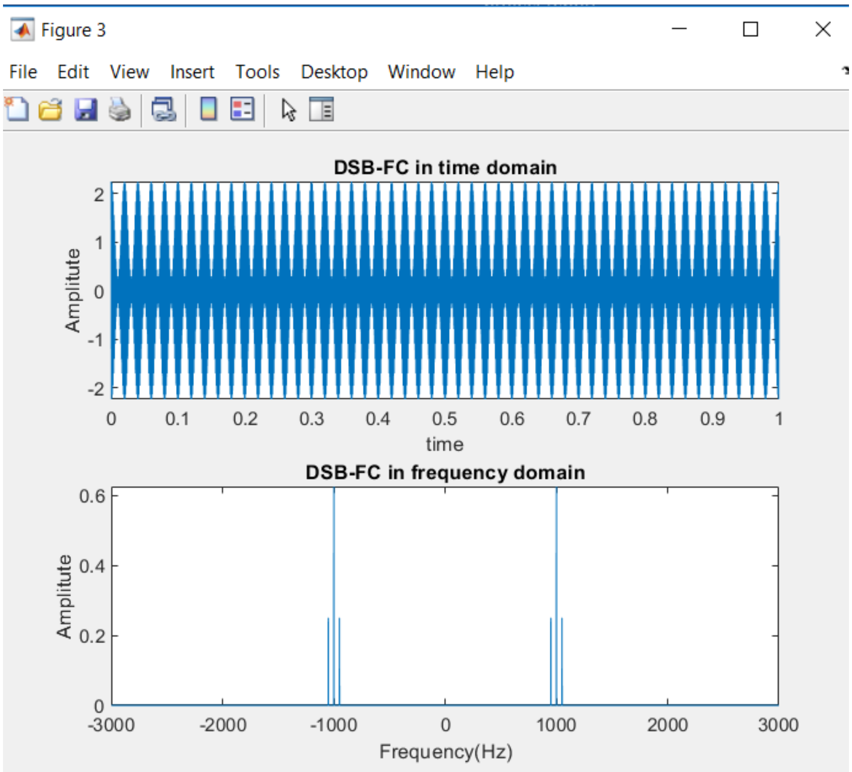
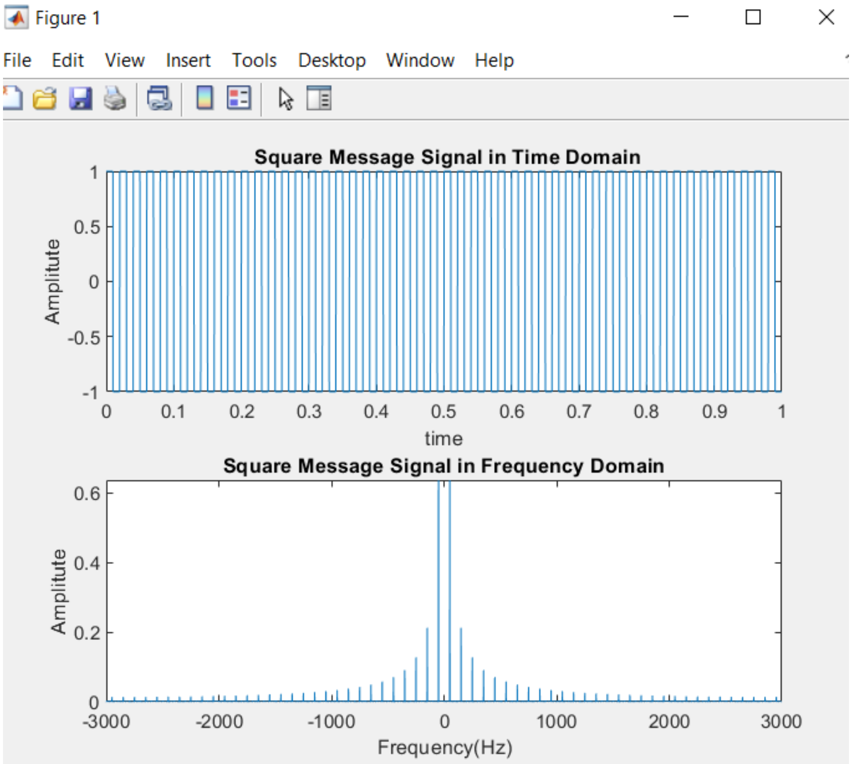
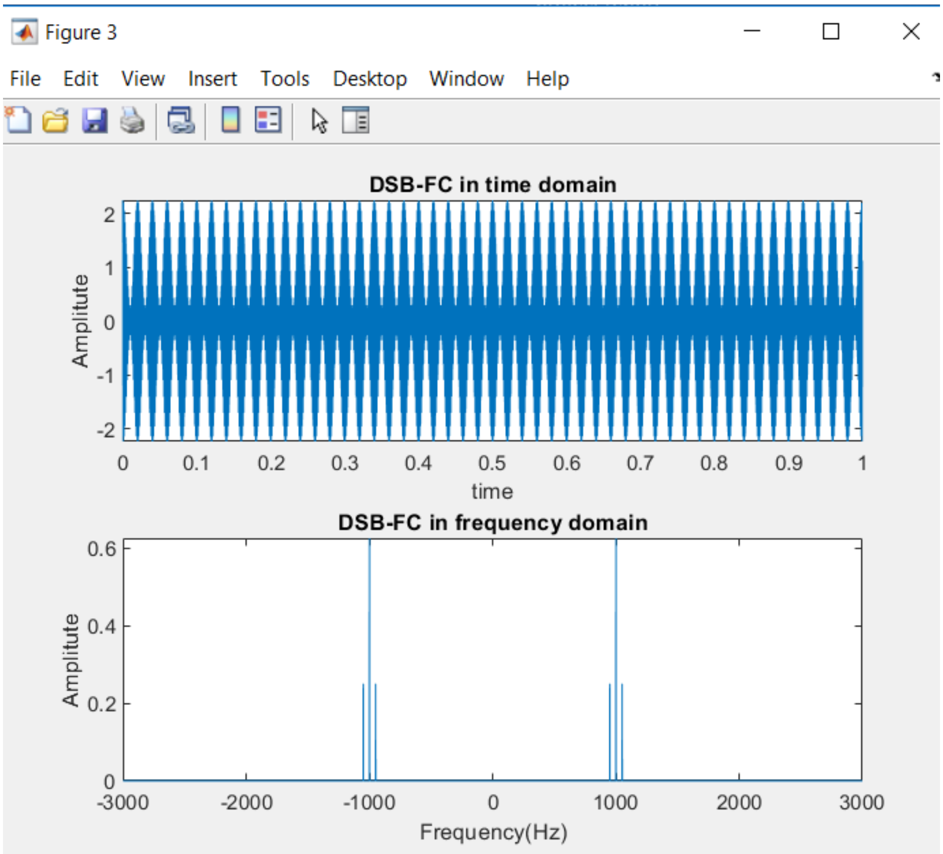



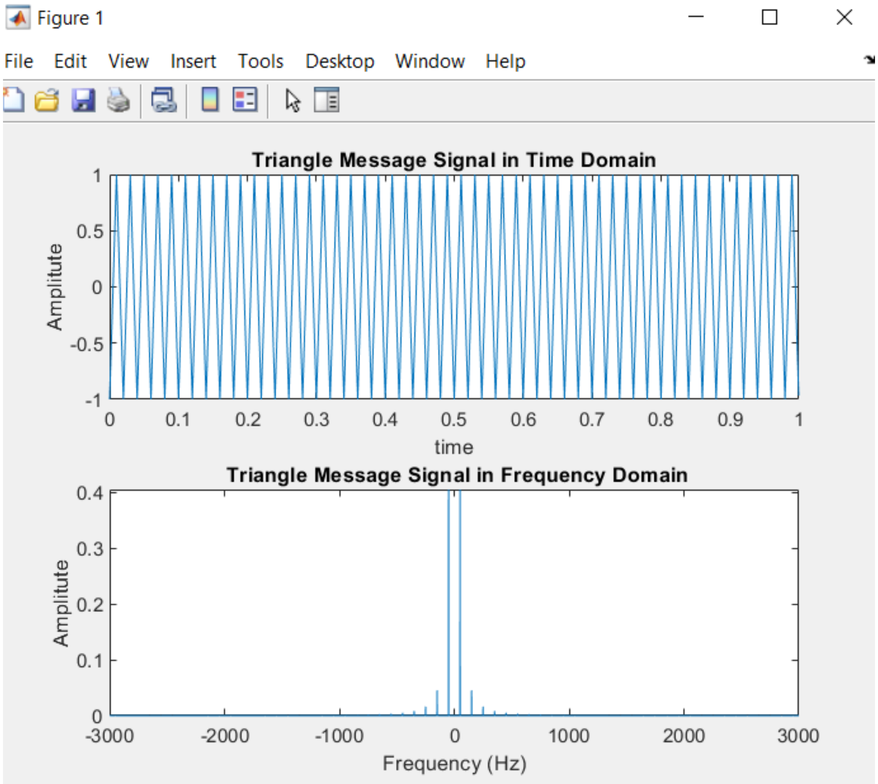
Figure 3
File Edit View Insert Tools Desktop Window Help
DSB-FC in time domain
Amplitute
time
DSB-FC in frequency domain
Amplitute
Frequency(Hz)

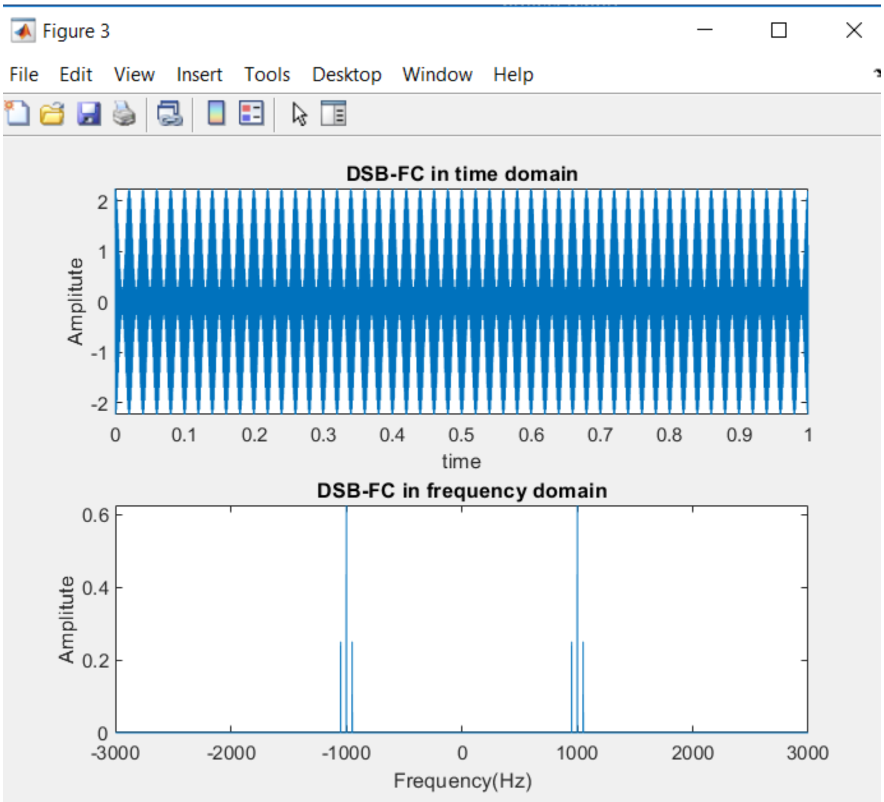
b. Square Wave



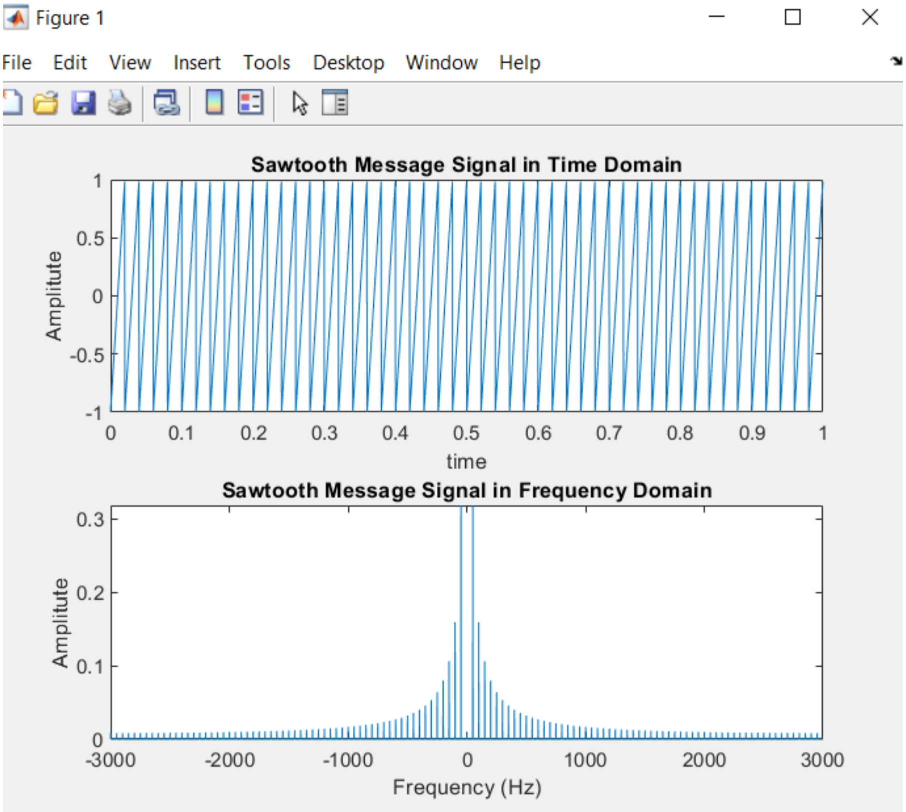



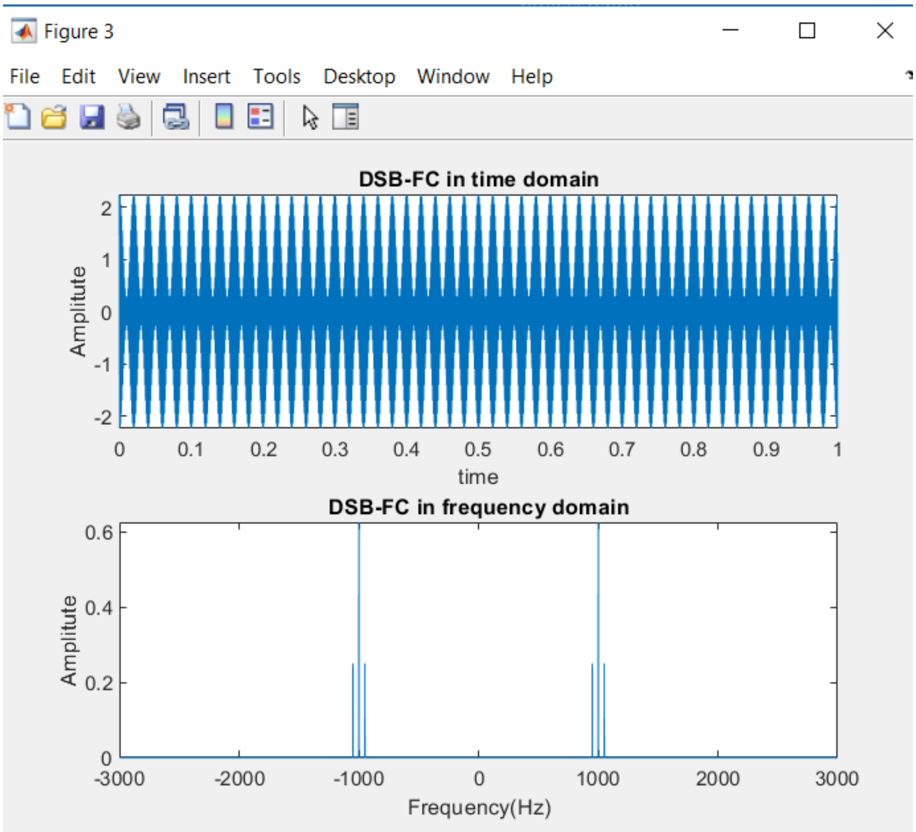
c. Triangle Wave



d. Sawtooth Wave

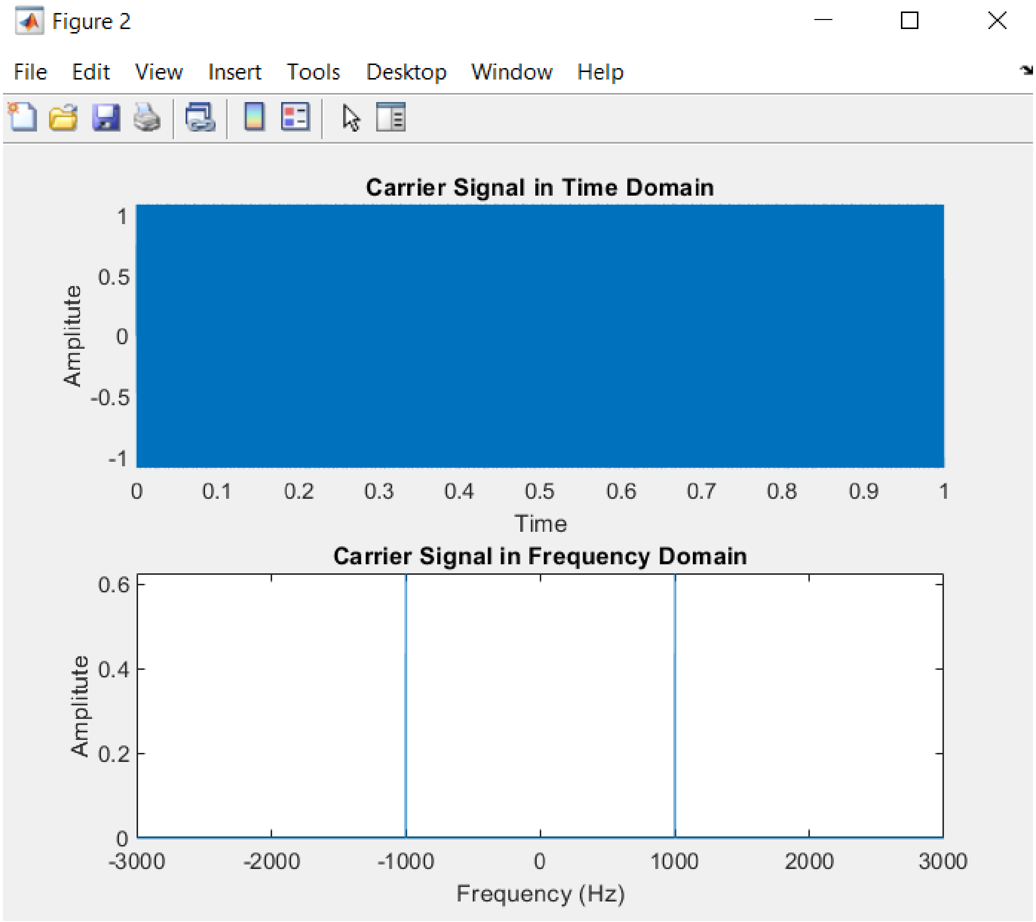


## 4.3 Double Sideband Suppressed Carrier

The following inputs were entered in the GUI for all message signals
Am= 1, fm=50, ac=1.25, fc=1000 and modulation index is 0.8

The carrier wave is same for all type of messages with the details mentioned above and figure is below




The message signal along with their modulation are shown by parts

a. Sine Wave

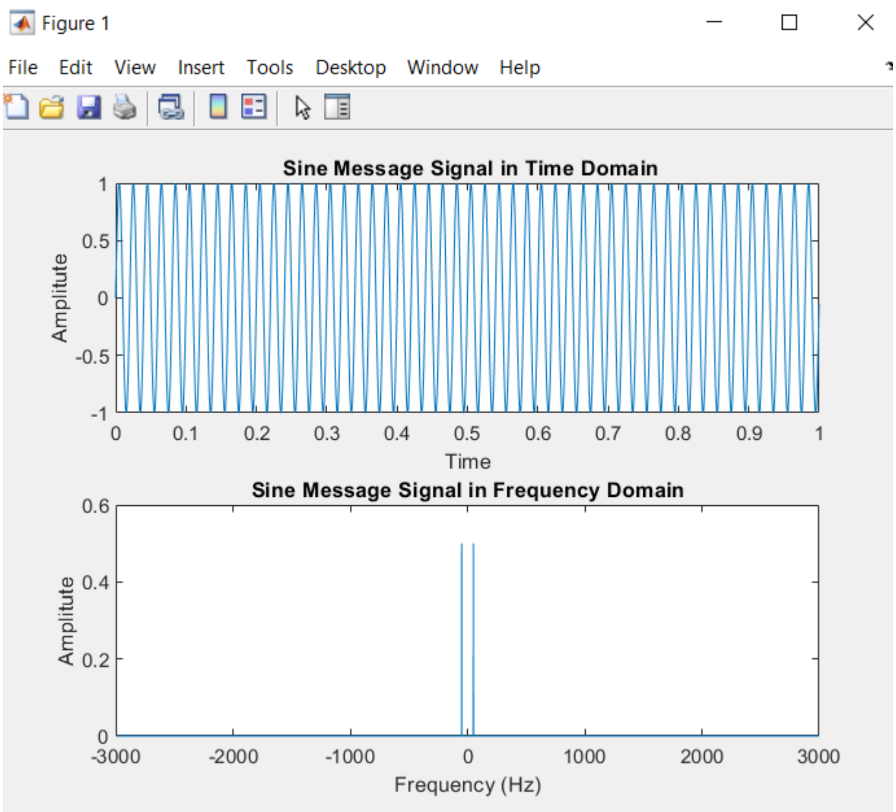


Figure 1
File Edit View Insert Tools Desktop Window Help
Sine Message Signal in Time Domain
Amplitute
Time
Sine Message Signal in Frequency Domain
Amplitute
Frequency (Hz)

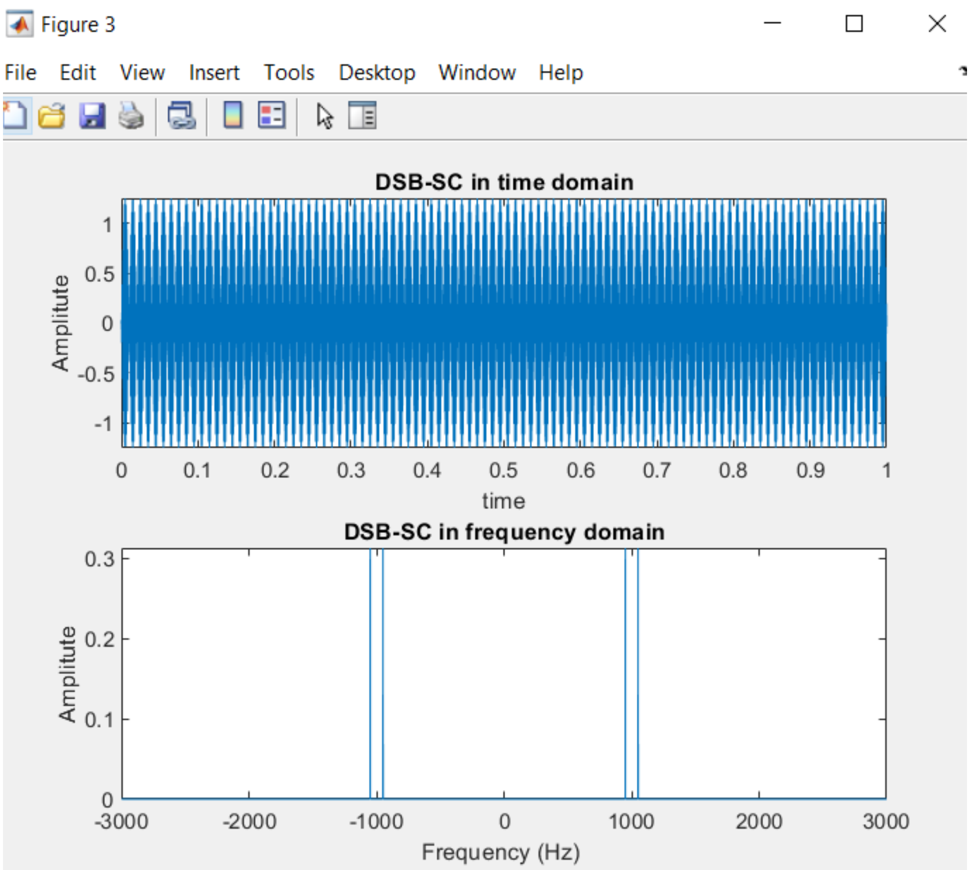
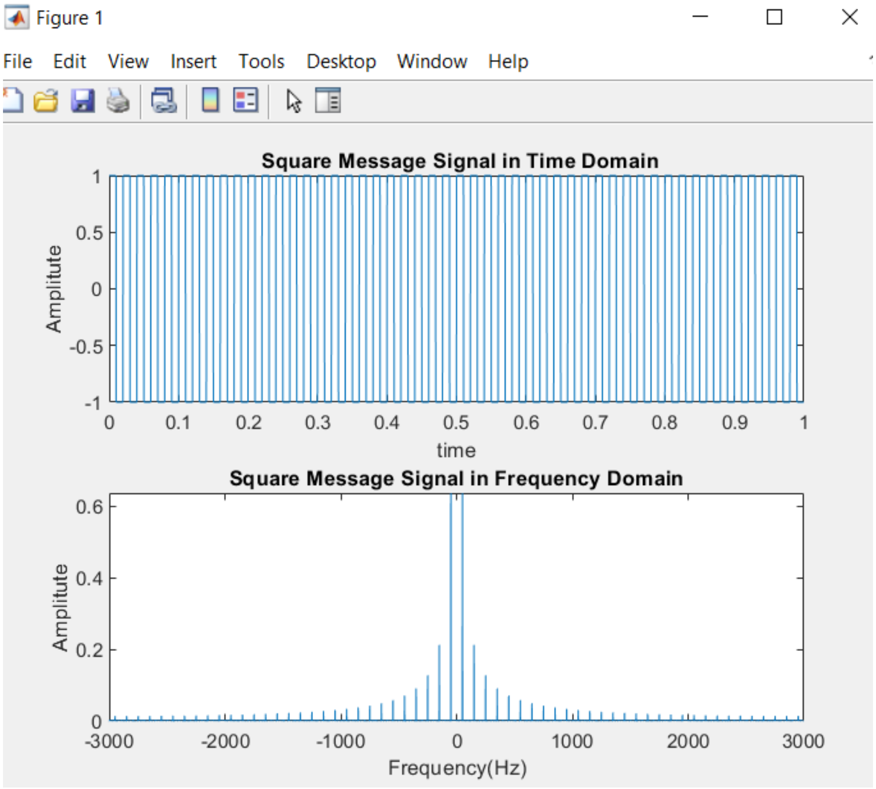
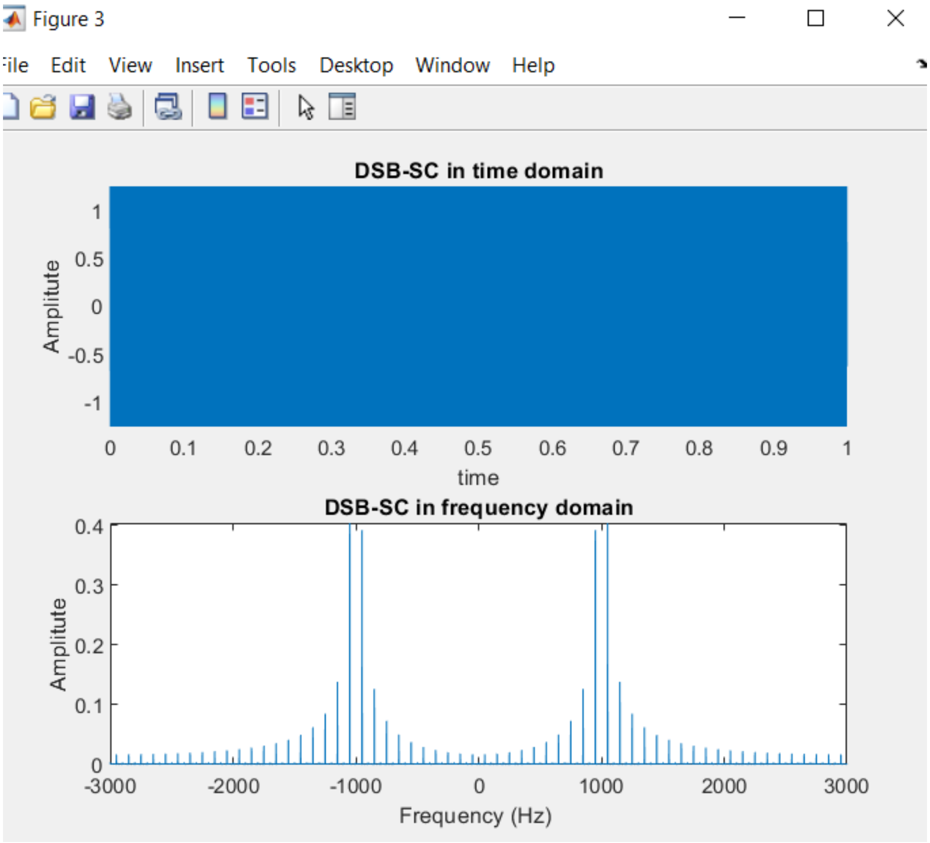



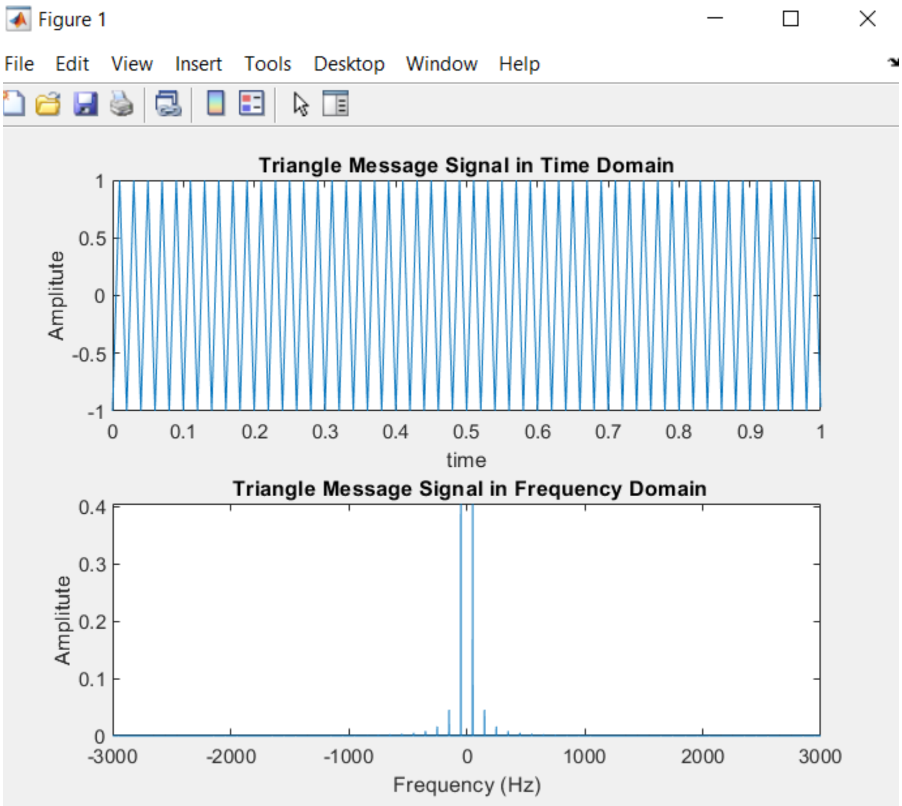
Figure 3
File Edit View Insert Tools Desktop Window Help
DSB-SC in time domain
Amplitute
time
DSB-SC in frequency domain
Amplitute
Frequency (Hz)

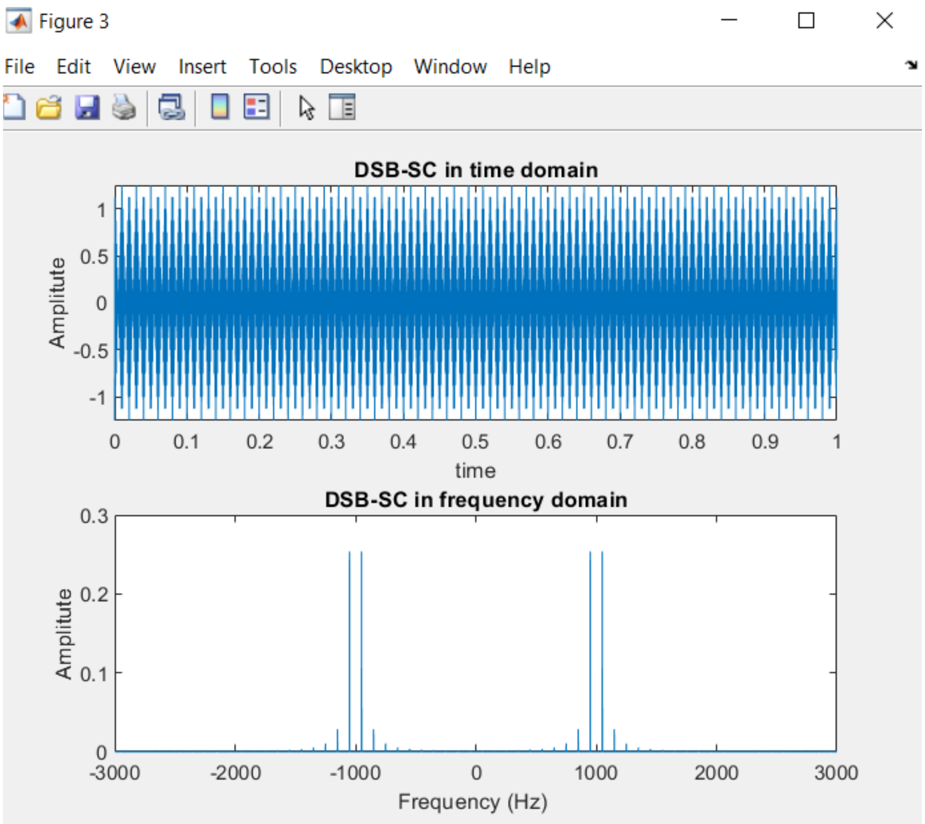
b. Square Wave



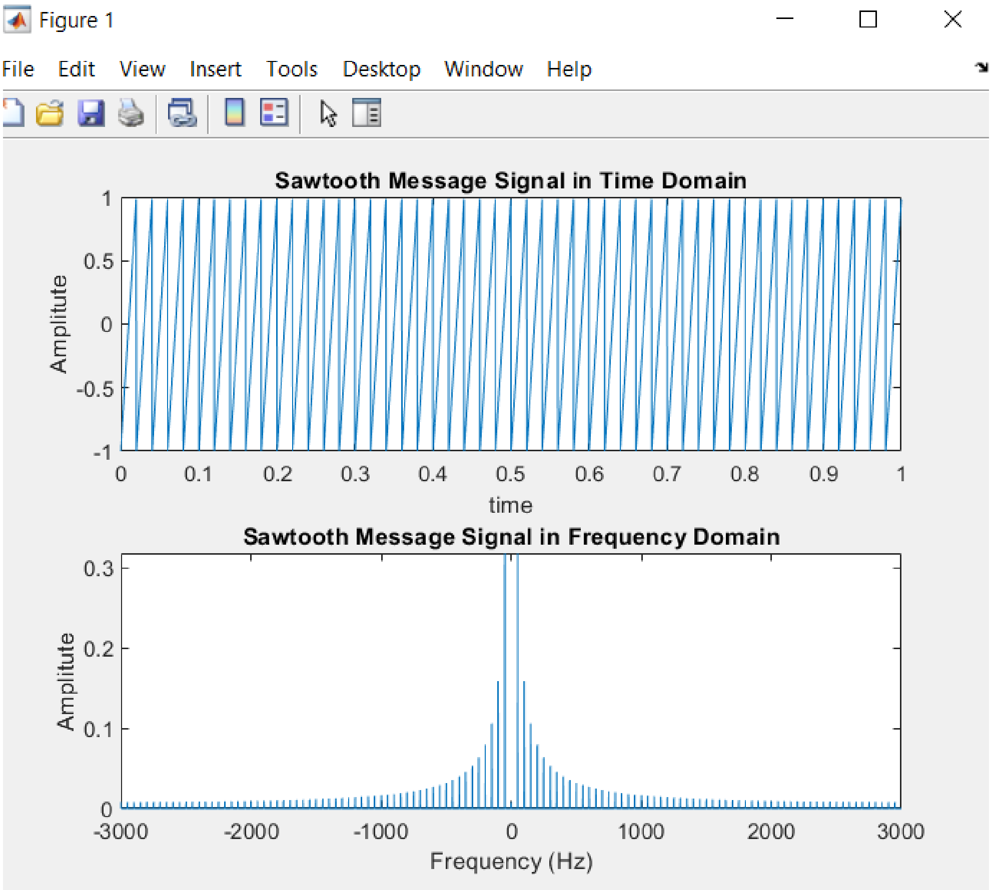



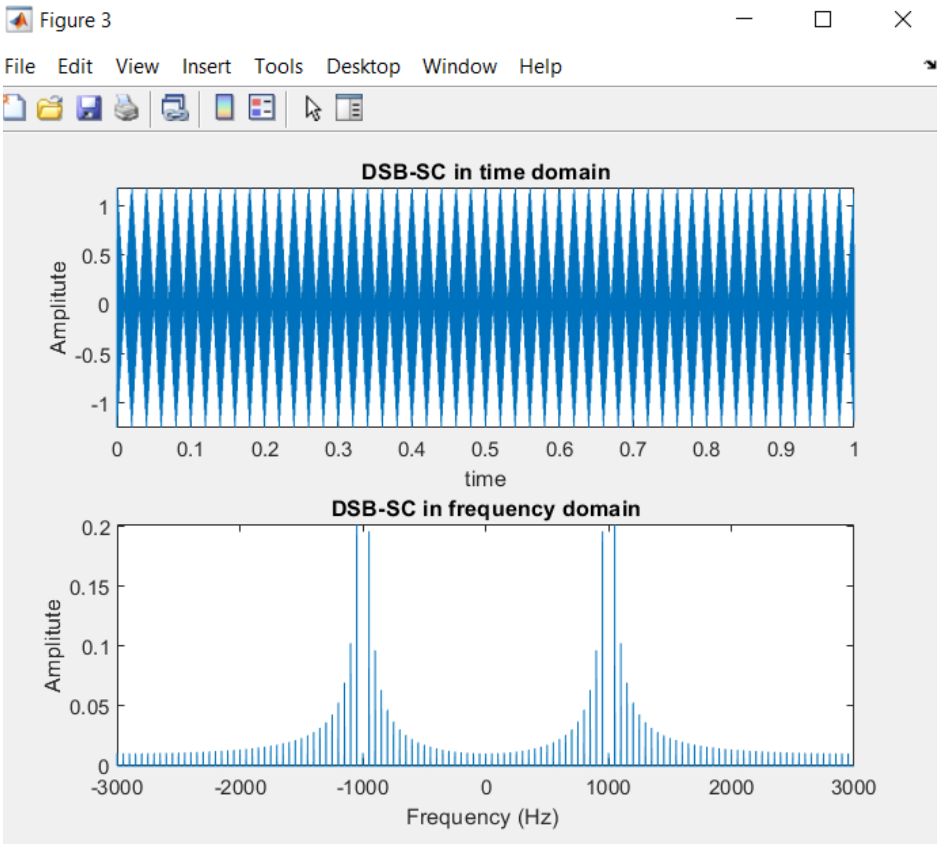
c. Triangle Wave



d. Sawtooth Wave

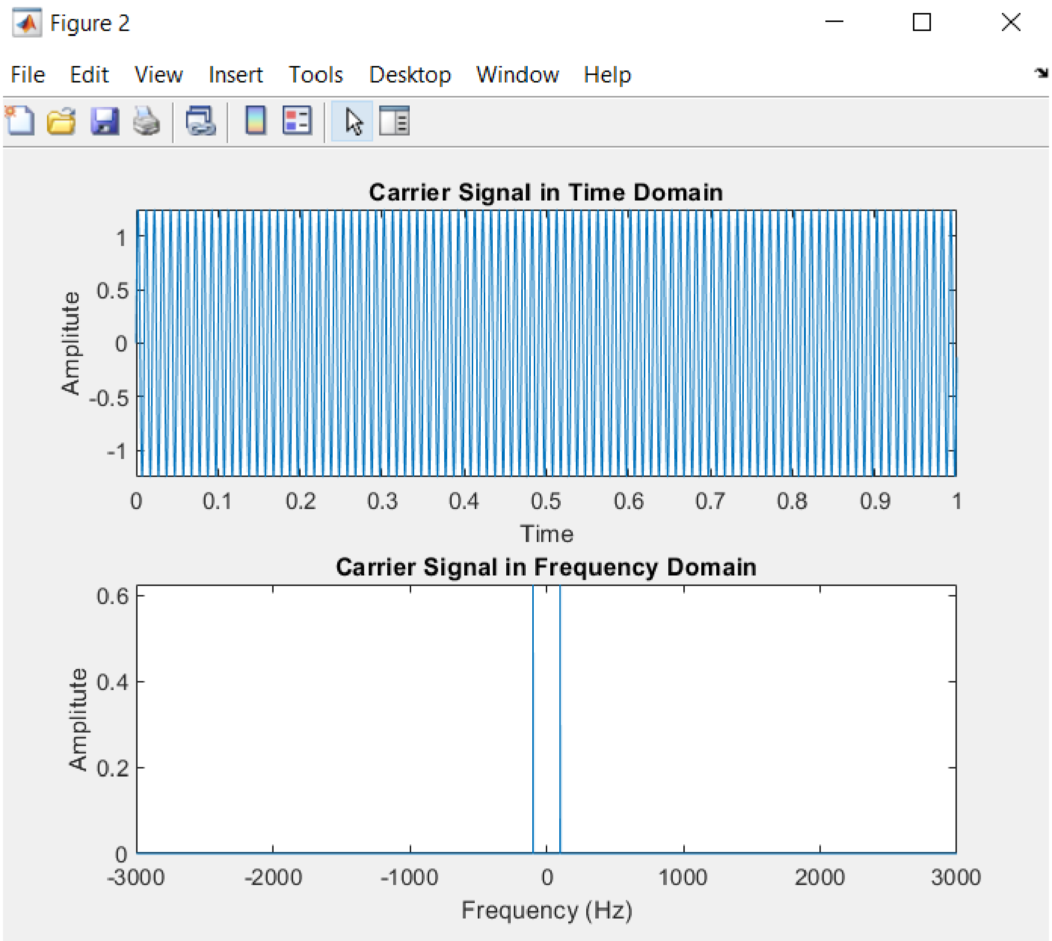


## 4.4 Single Sideband Upper and Lower Modulation

Am=1, fm=5, ac=1.25, fc=100, modulation index= 0.8

The carrier wave is same for all type of messages with the details mentioned above and figure is below




The message signal along with their modulation are shown by parts

## a. Sine Wave

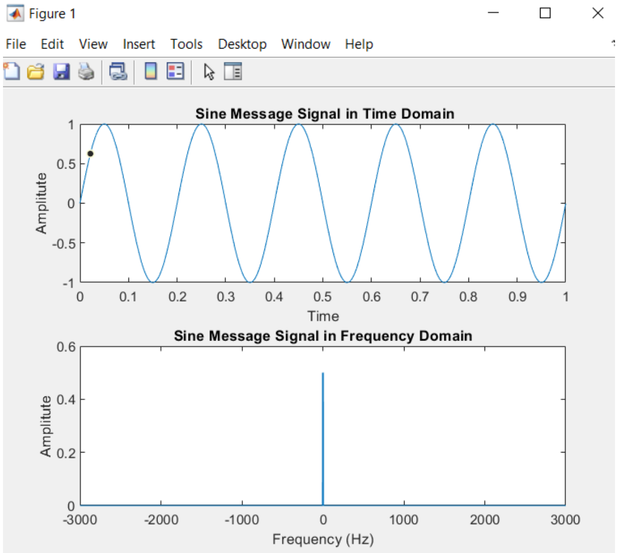


Figure 1
File Edit View Insert Tools Desktop Window Help
Sine Message Signal in Time Domain
Amplitute
Time
Sine Message Signal in Frequency Domain
Amplitute
Frequency (Hz)

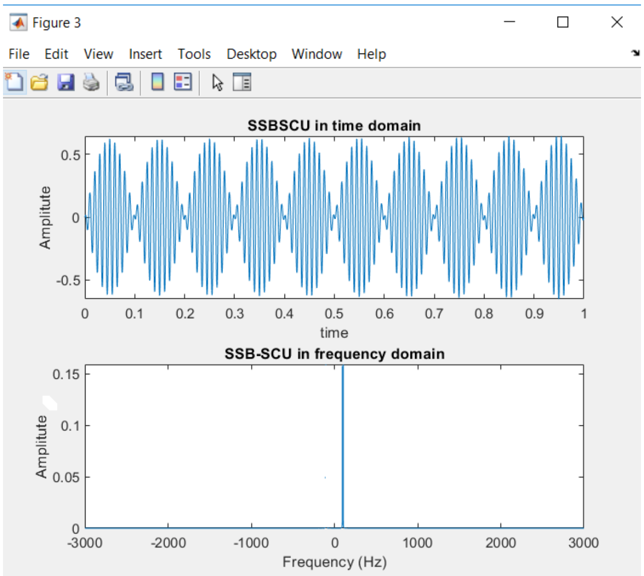



Figure 3
File Edit View Insert Tools Desktop Window Help
SSBSCU in time domain
Amplitute
time
SSB-SCU in frequency domain
Amplitute
Frequency (Hz)

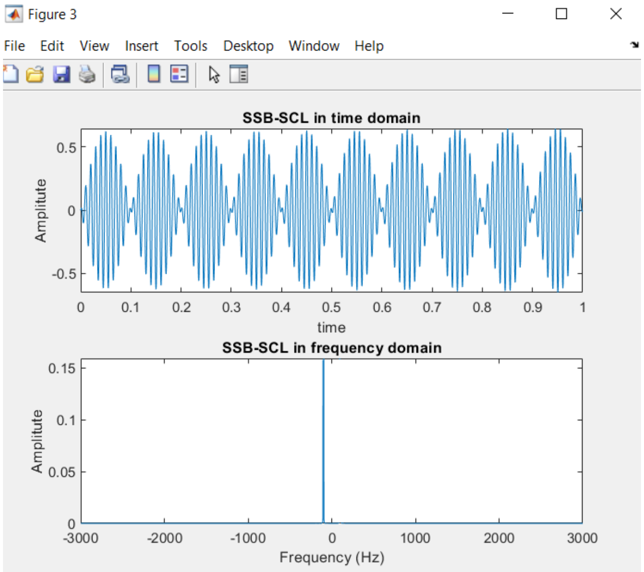
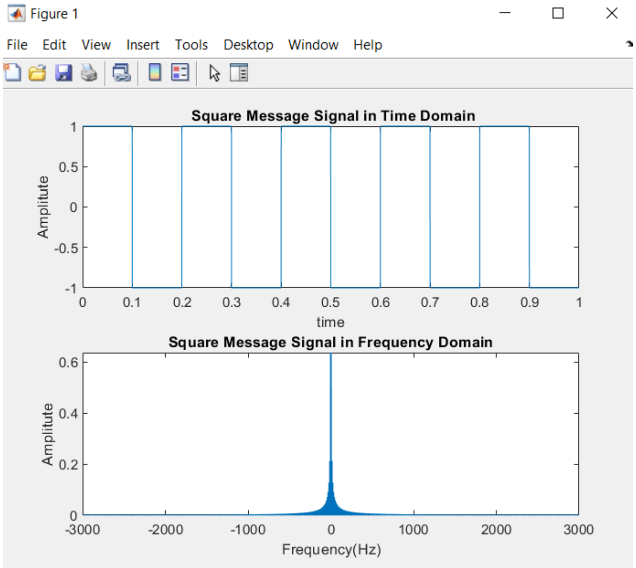
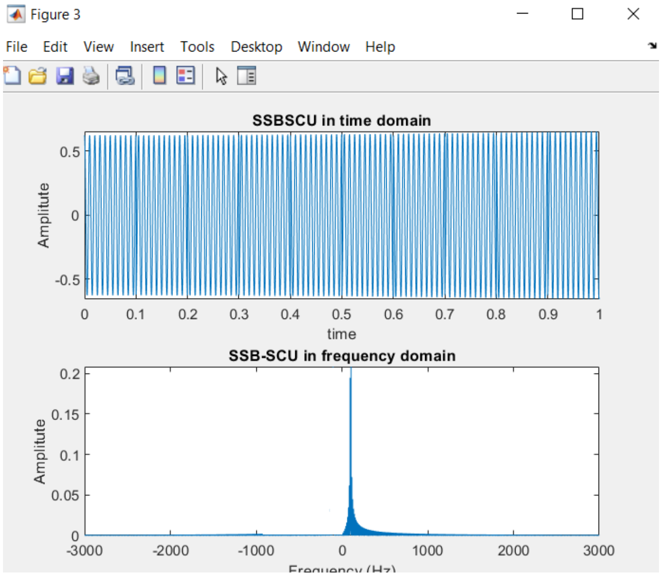
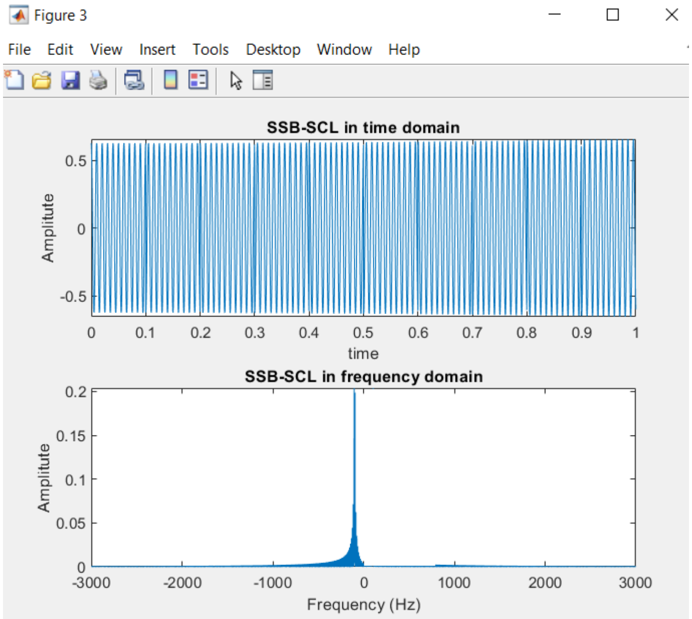



Figure 3
File Edit View Insert Tools Desktop Window Help
SSB-SCL in time domain
Amplitute
time
SSB-SCL in frequency domain
Amplitute
Frequency (Hz)

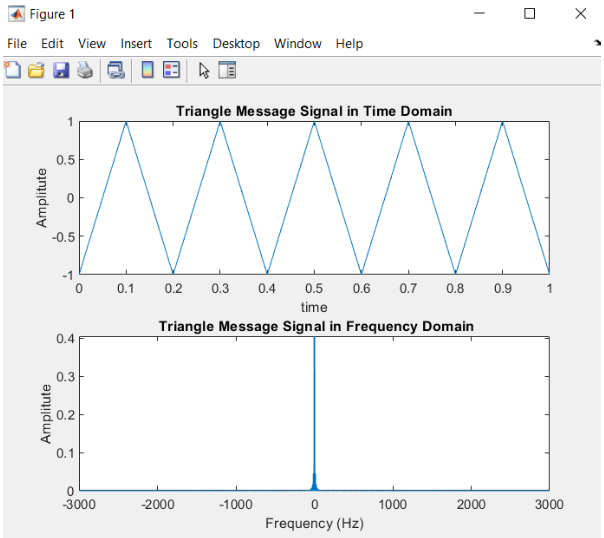
b. Square Wave

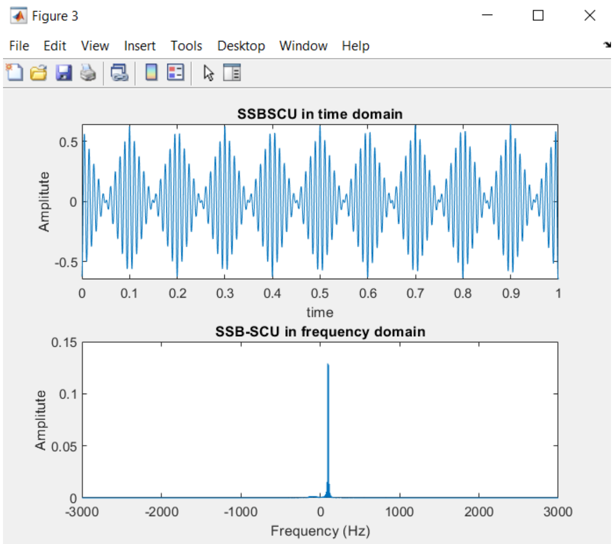


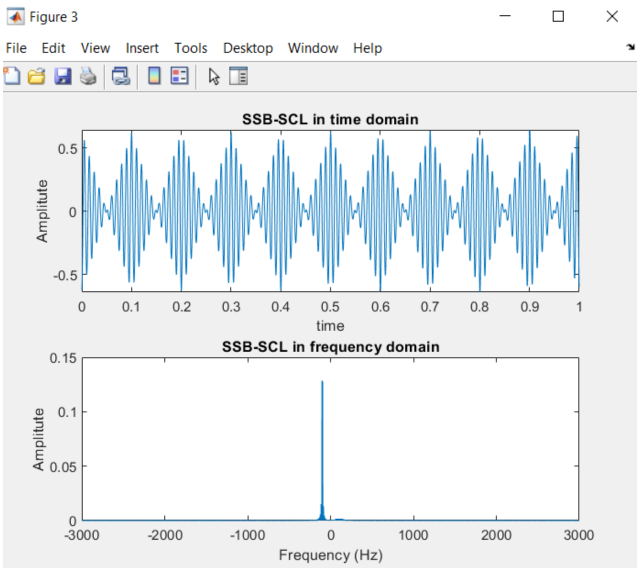



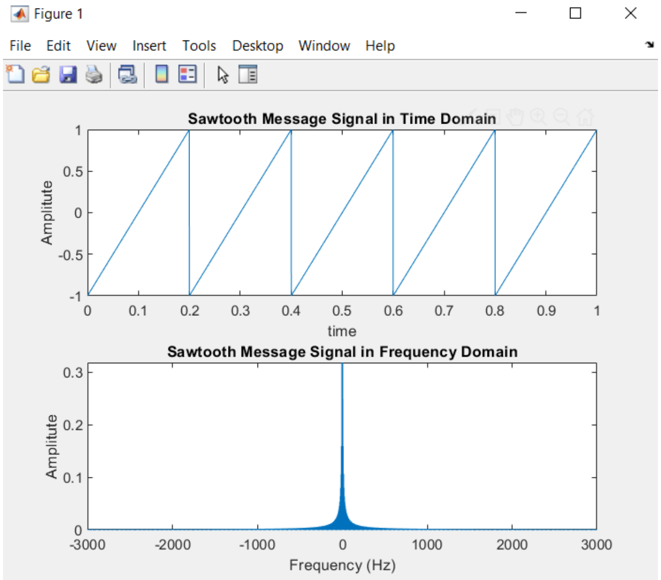
## c. Triangle Wave

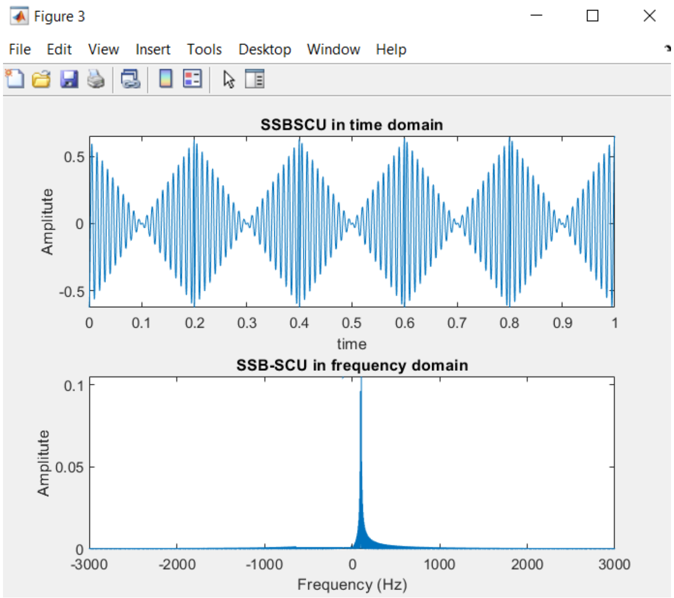


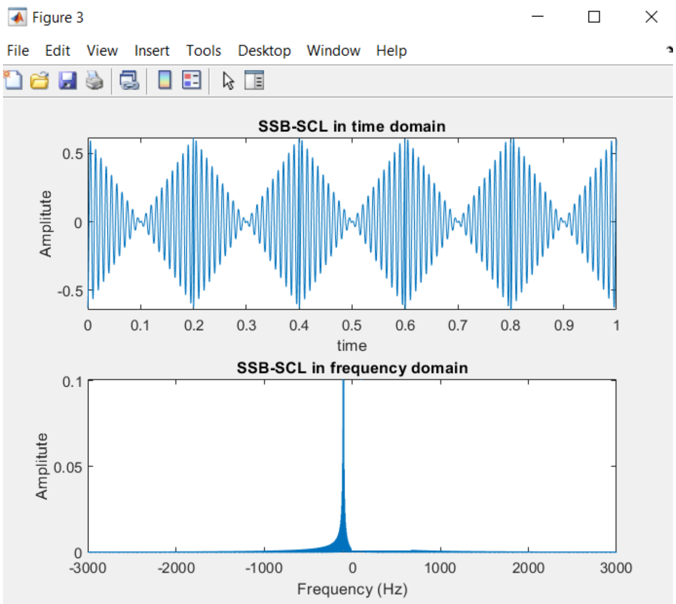



## d. Sawtooth Wave



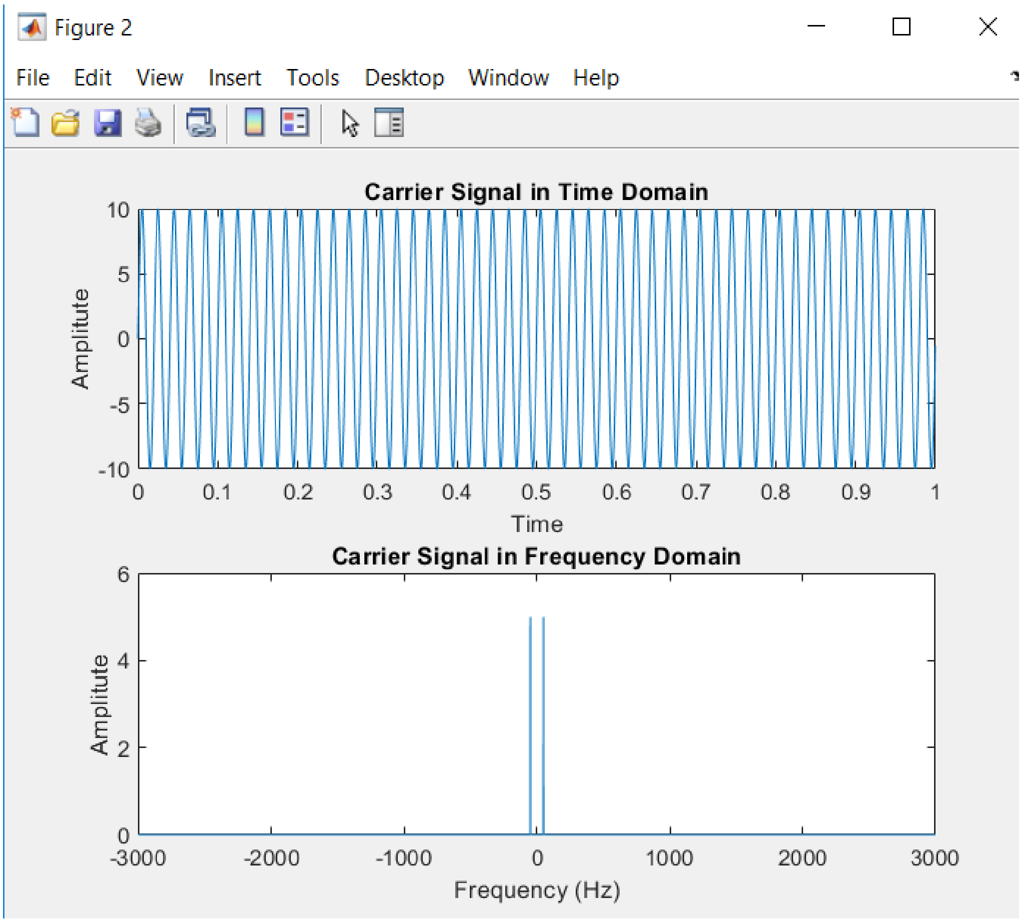



## 4.5 Frequency Modulation

Am=10, fm=10, ac=10, fc=50, modulation index= 3

The carrier wave is same for all type of messages with the details mentioned above and figure is below




The message signal along with their modulation are shown by parts

a. Sine Wave

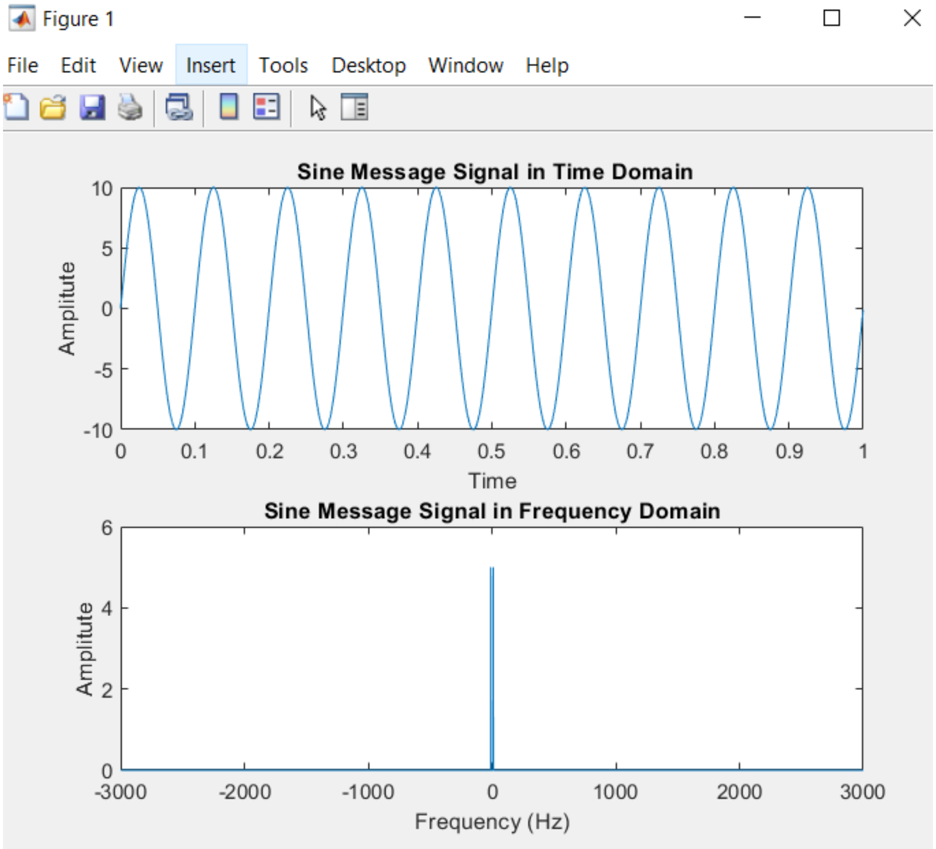


Figure 1
File Edit View Insert Tools Desktop Window Help
Sine Message Signal in Time Domain
Amplitute
Time
Sine Message Signal in Frequency Domain
Amplitute
Frequency (Hz)

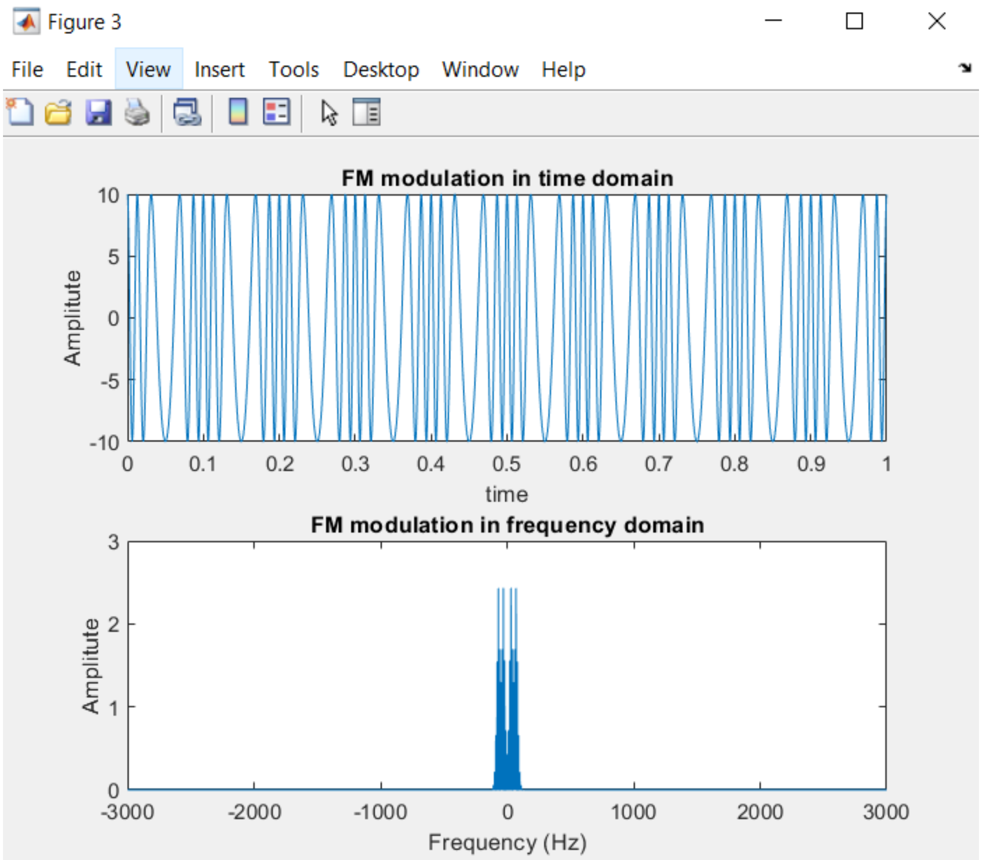
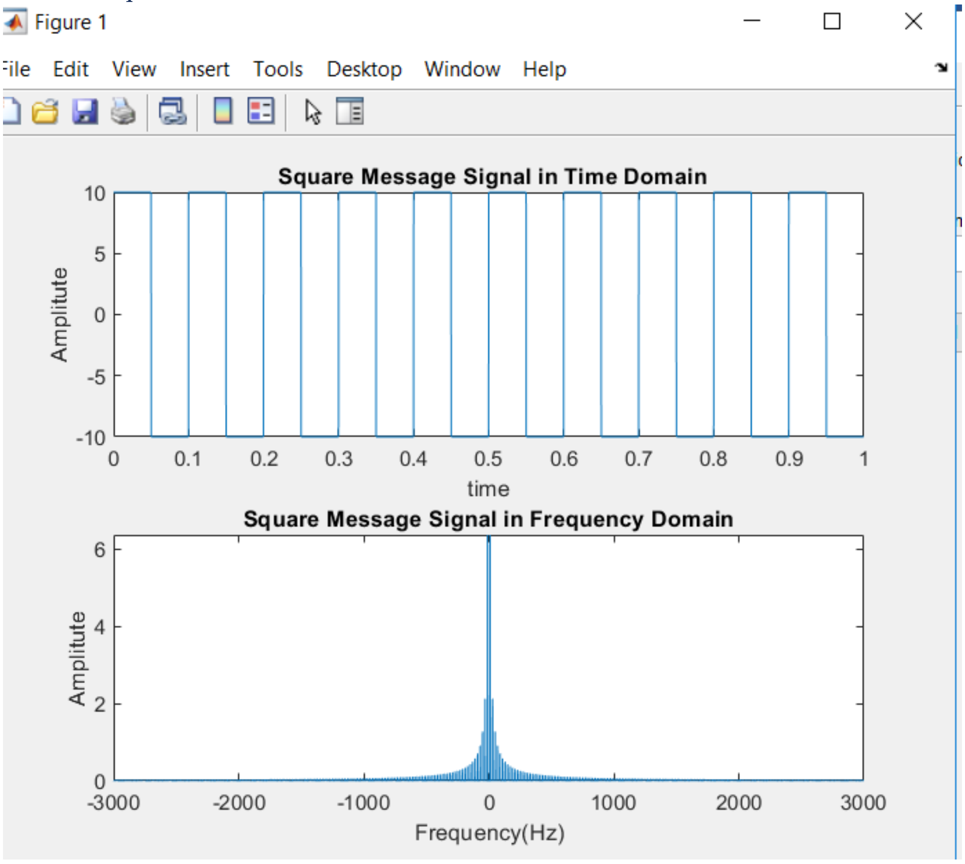
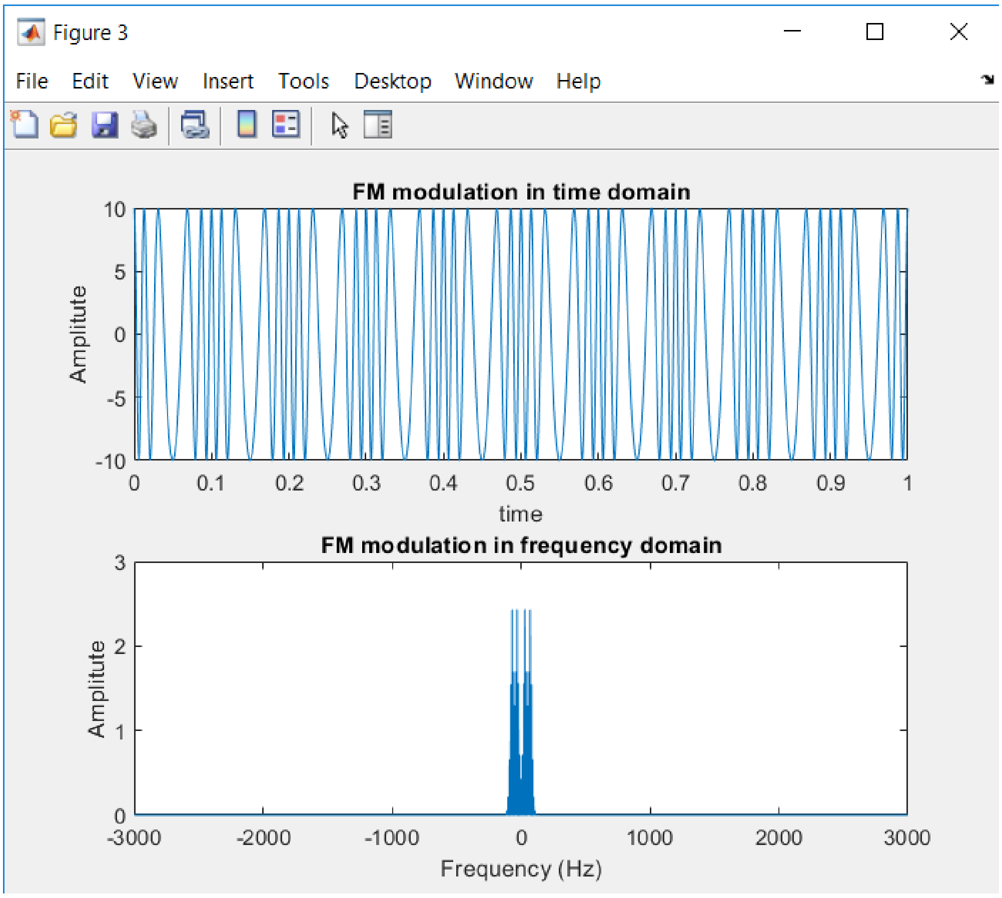


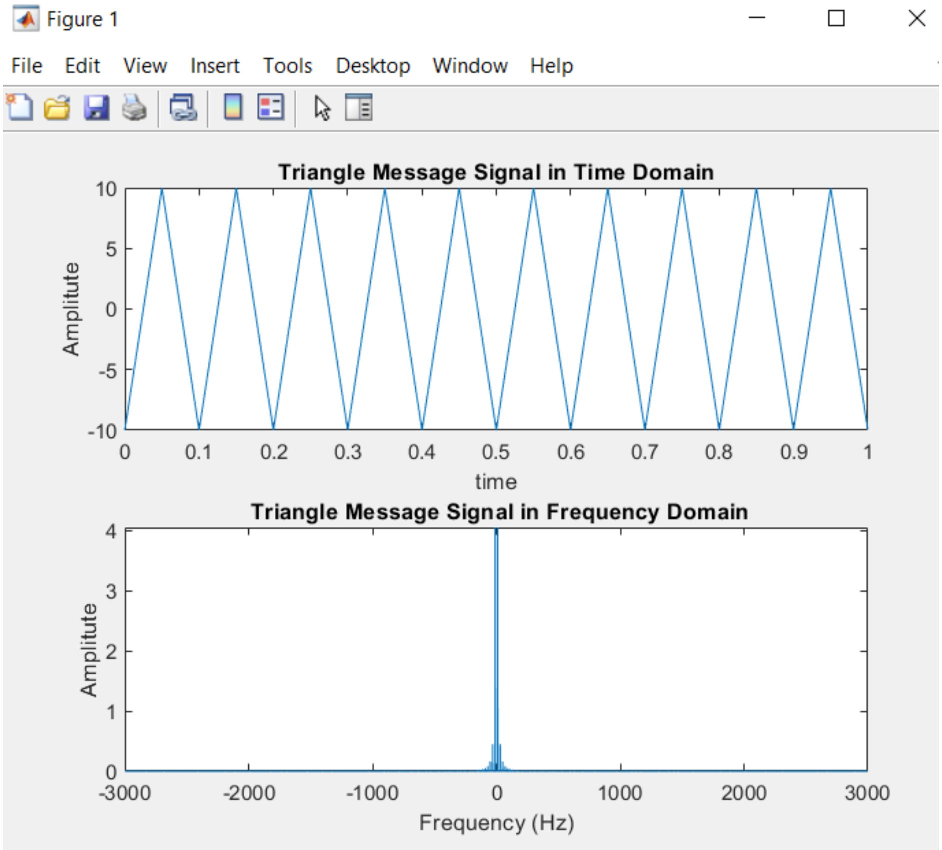
Figure 3
File Edit View Insert Tools Desktop Window Help
FM modulation in time domain
Amplitute
time
FM modulation in frequency domain
Amplitute
Frequency (Hz)

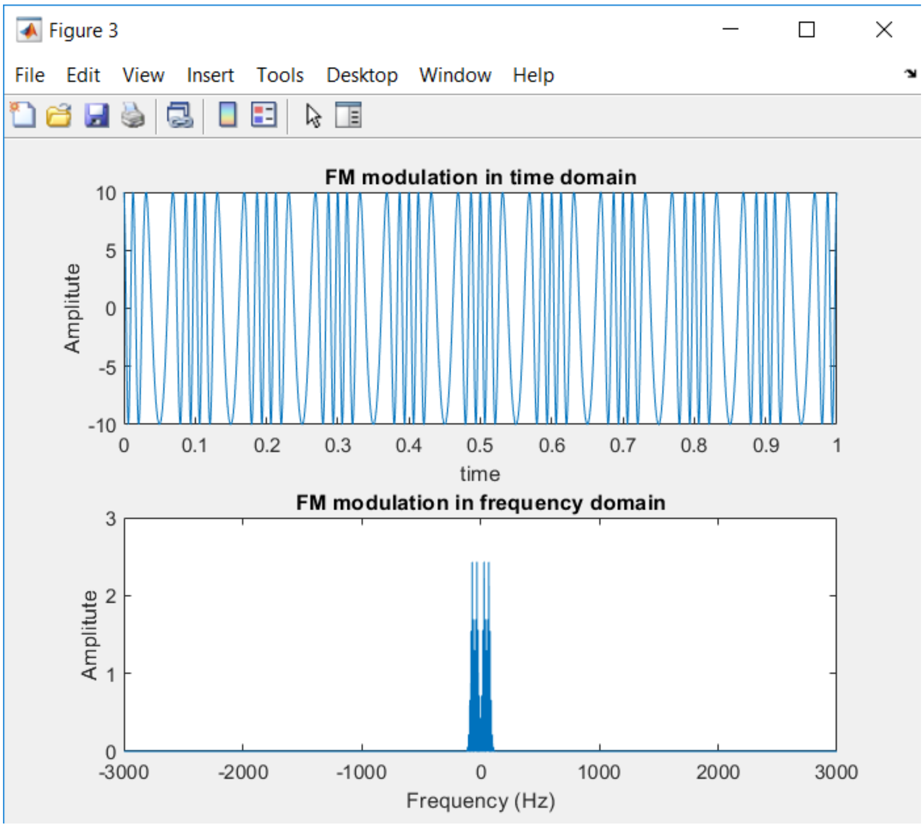
b. Square Wave



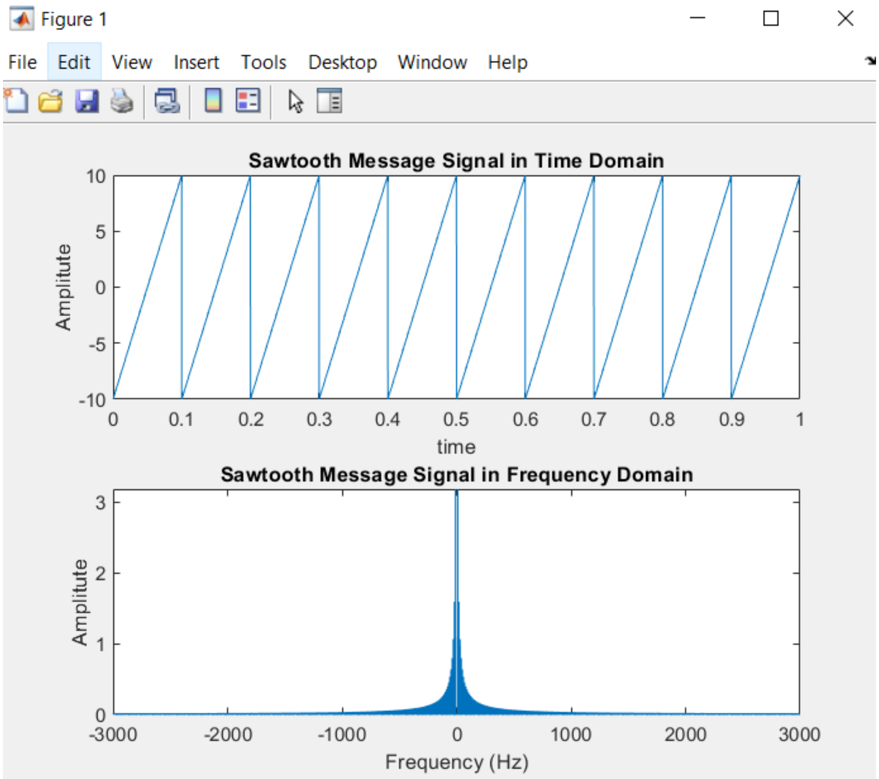



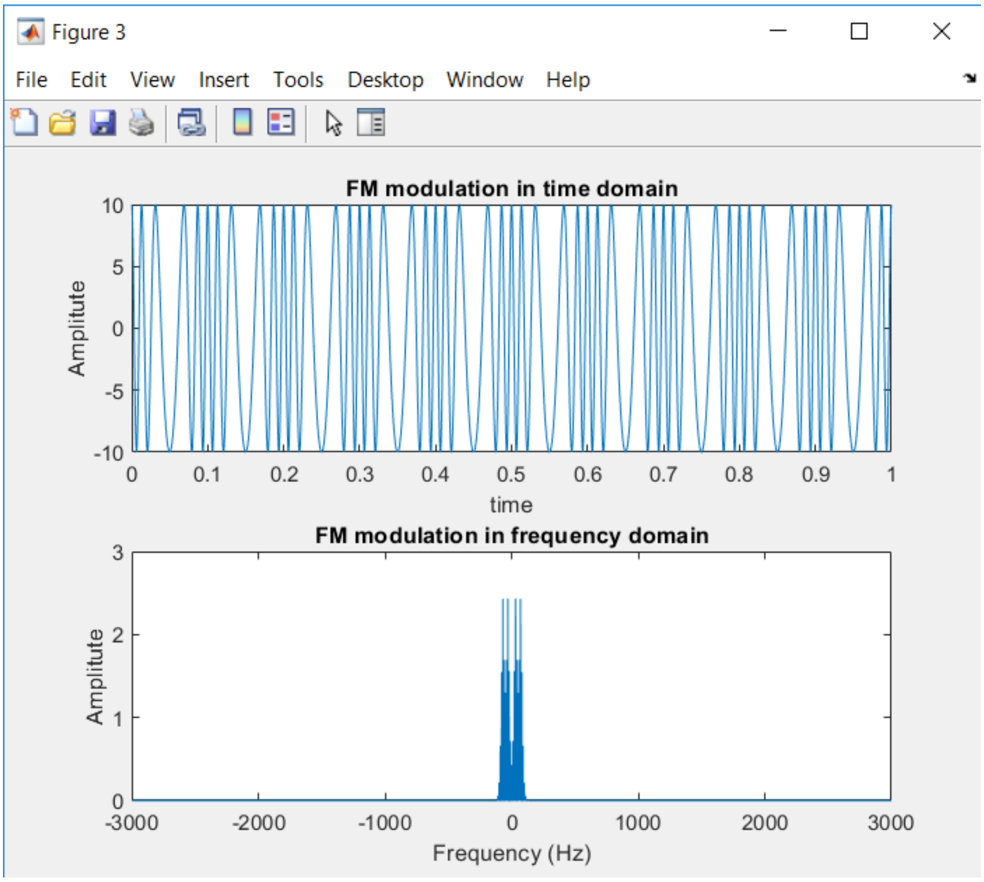
## c. Triangle Wave



d. Sawtooth Wave

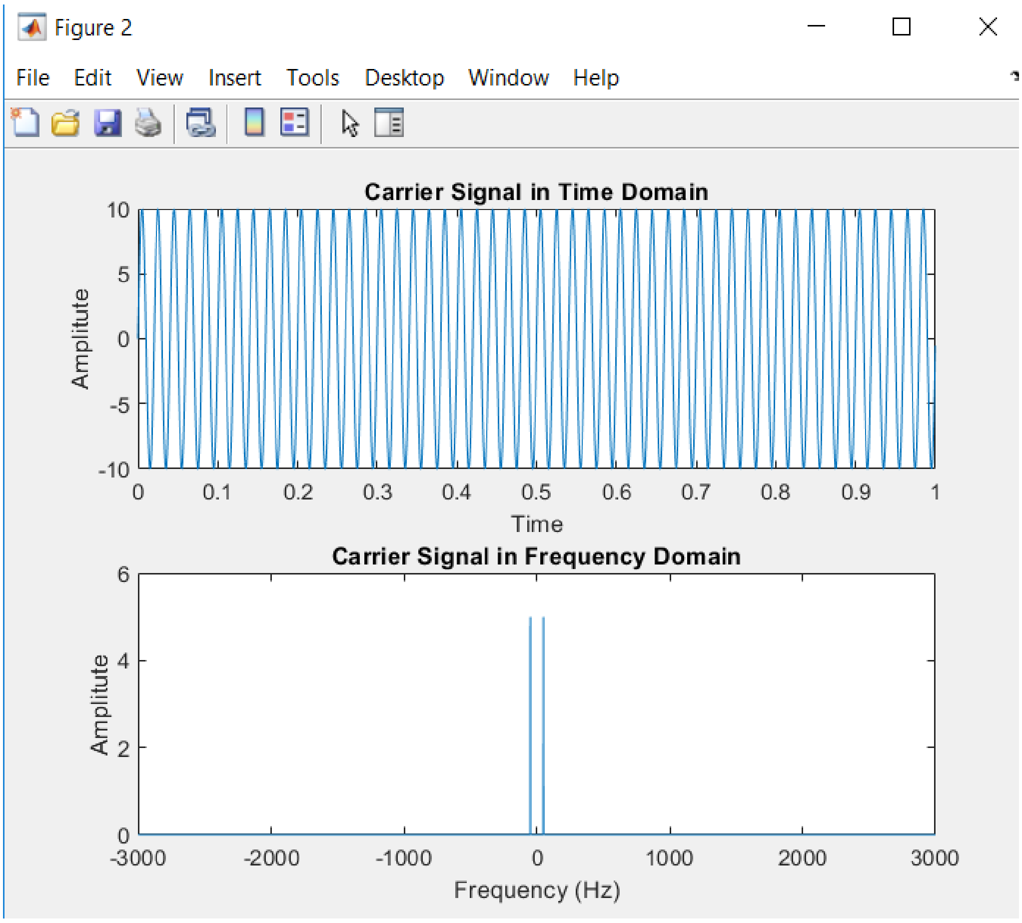


## 4.6 Phase Modulation

Am=10, fm=10, ac=10, fc=50, modulation index= 2

The carrier wave is same for all type of messages with the details mentioned above and figure is below




The message signal along with their modulation are shown by parts.

a. Sine Wave

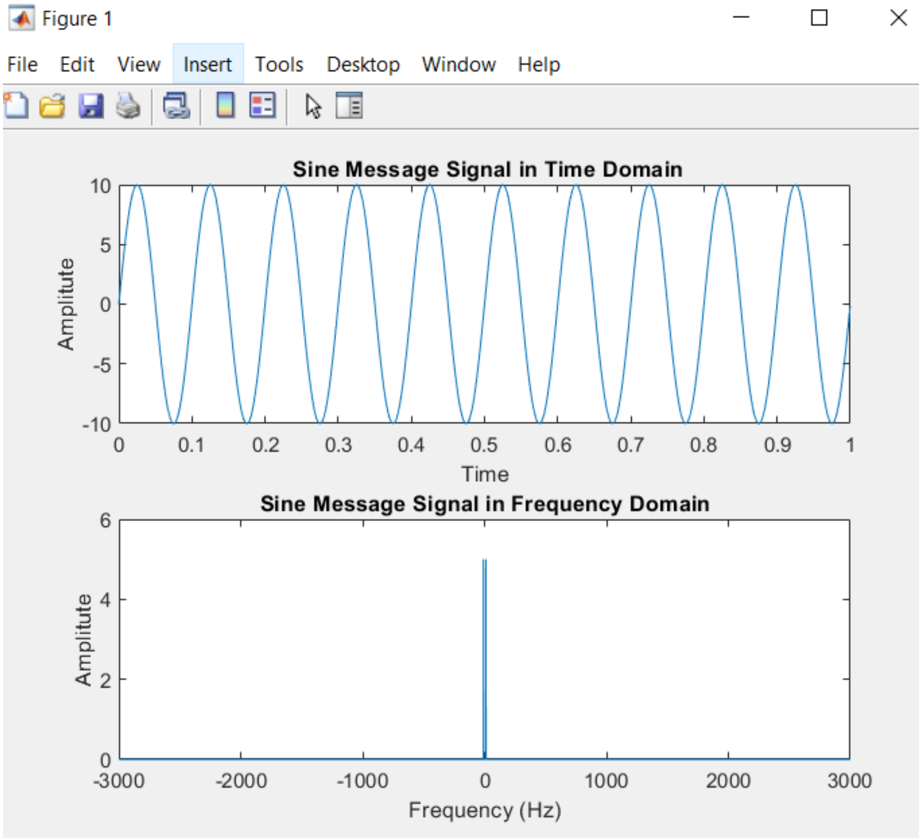


Figure 1
File Edit View Insert Tools Desktop Window Help
Sine Message Signal in Time Domain
Amplitute
Time
Sine Message Signal in Frequency Domain
Amplitute
Frequency (Hz)

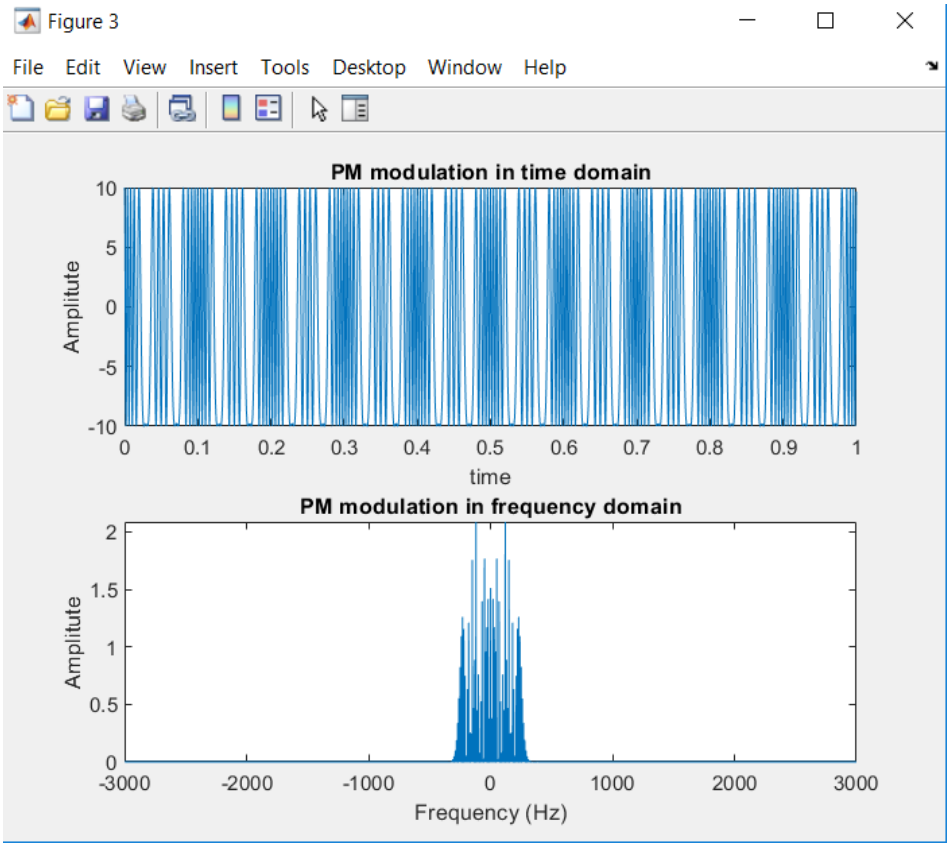
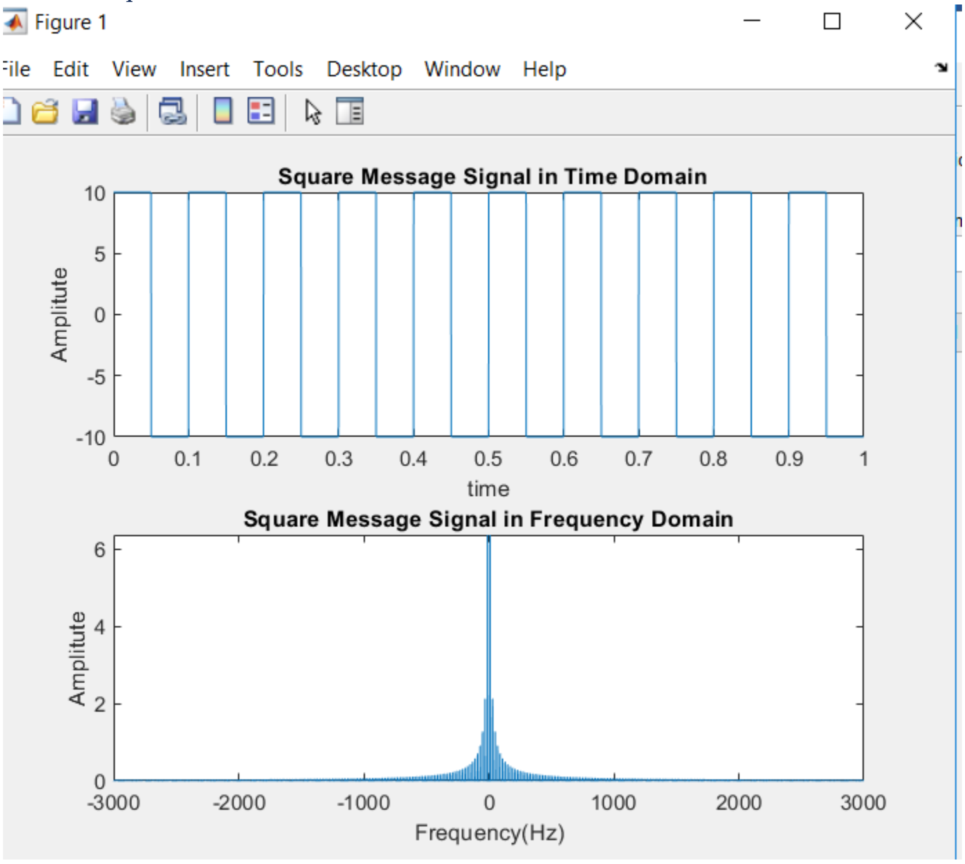
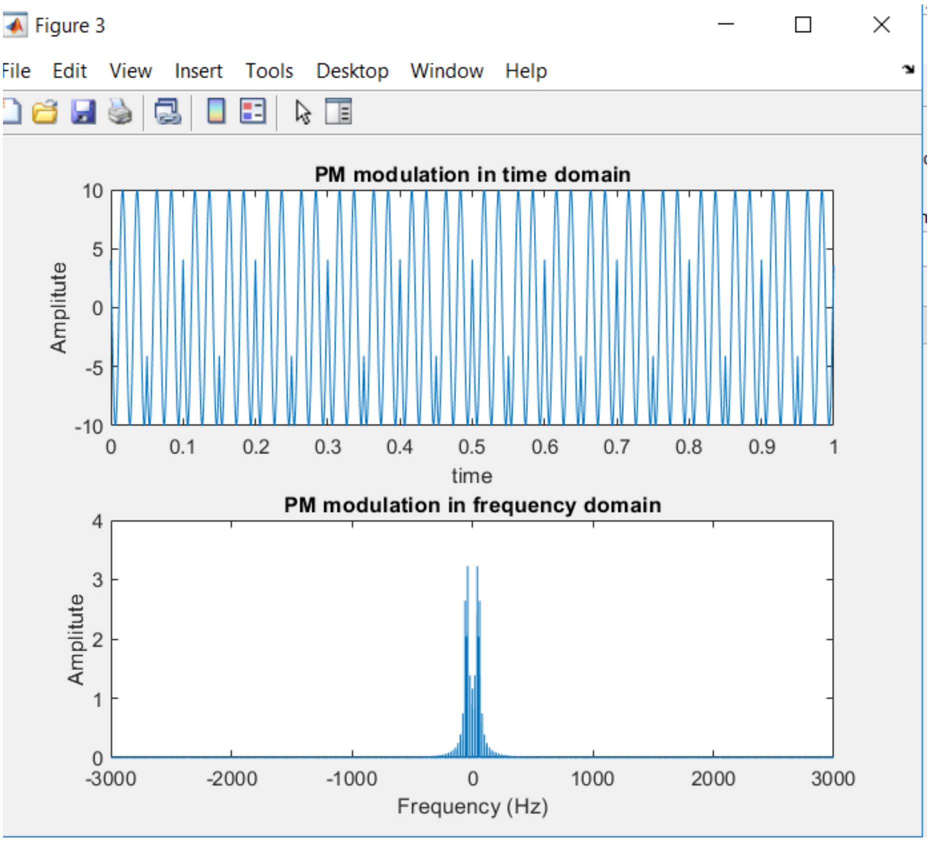


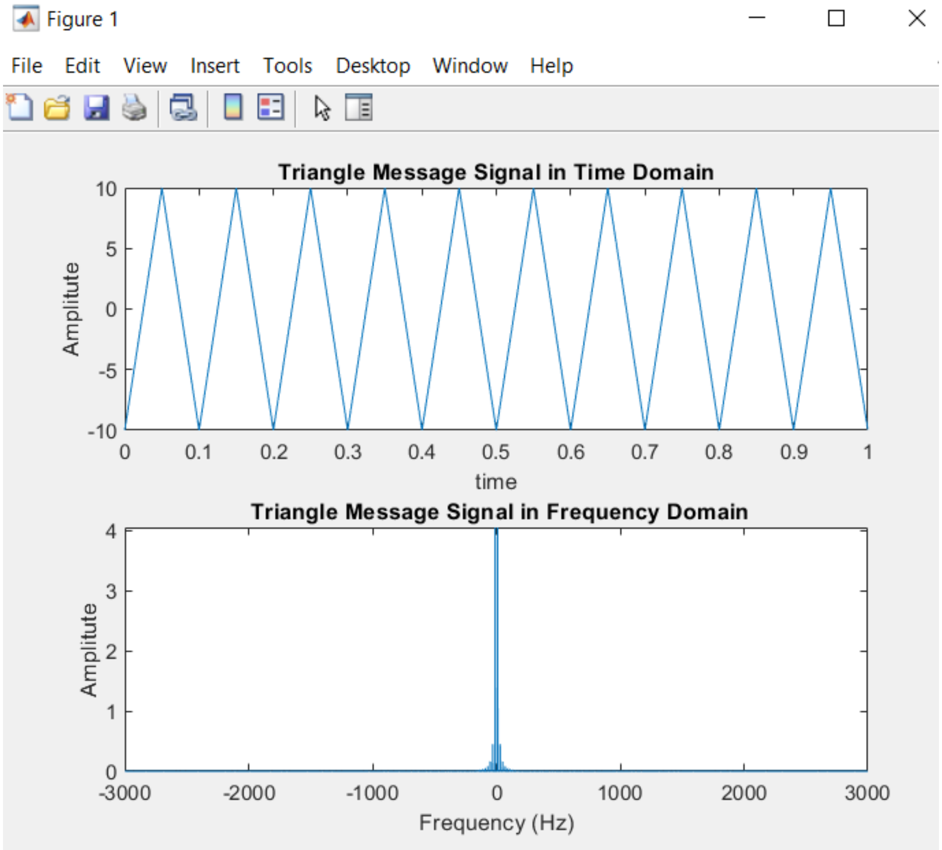
Figure 3
File Edit View Insert Tools Desktop Window Help
PM modulation in time domain
Amplitute
time
PM modulation in frequency domain
Amplitute
Frequency (Hz)

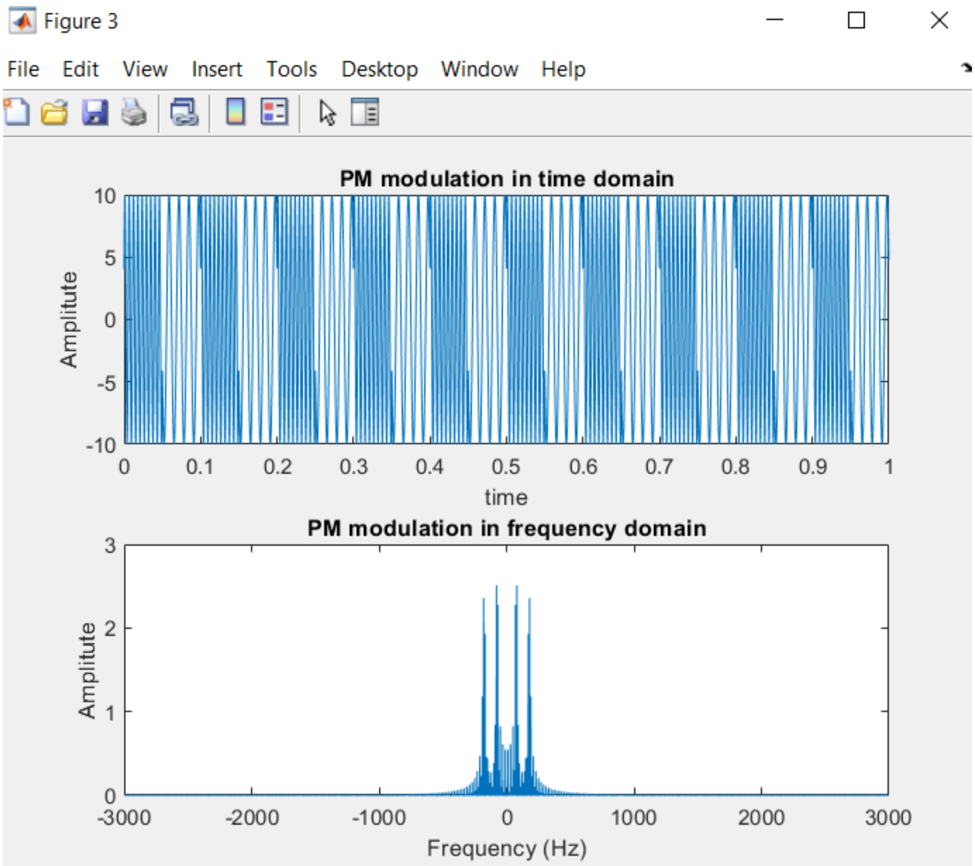
b. Square Wave



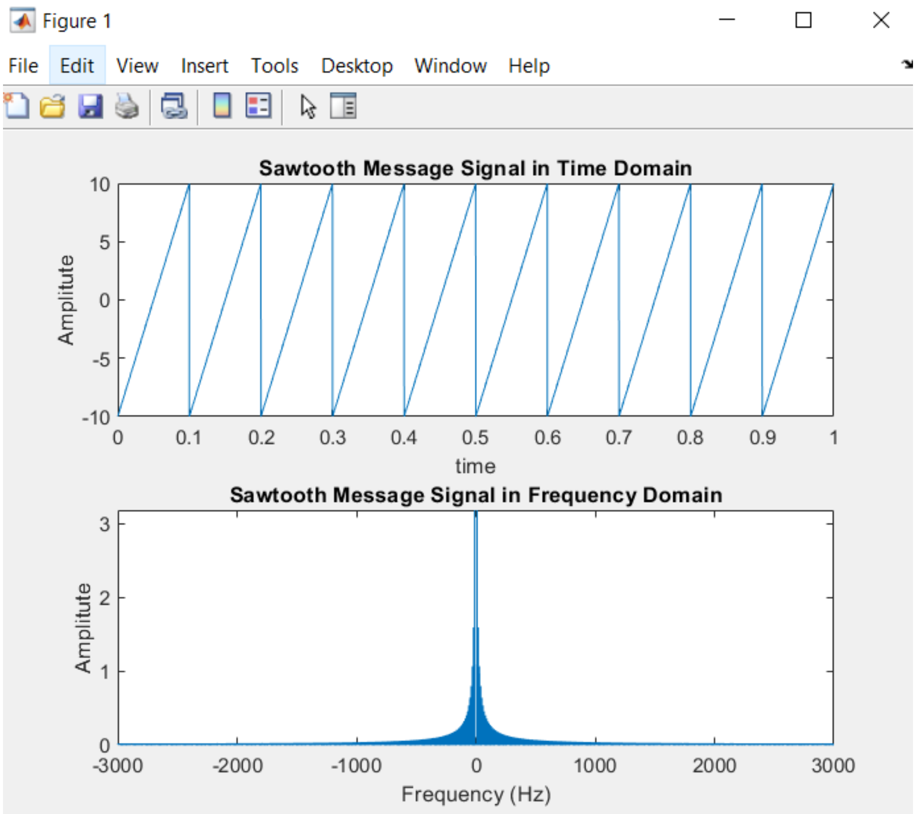



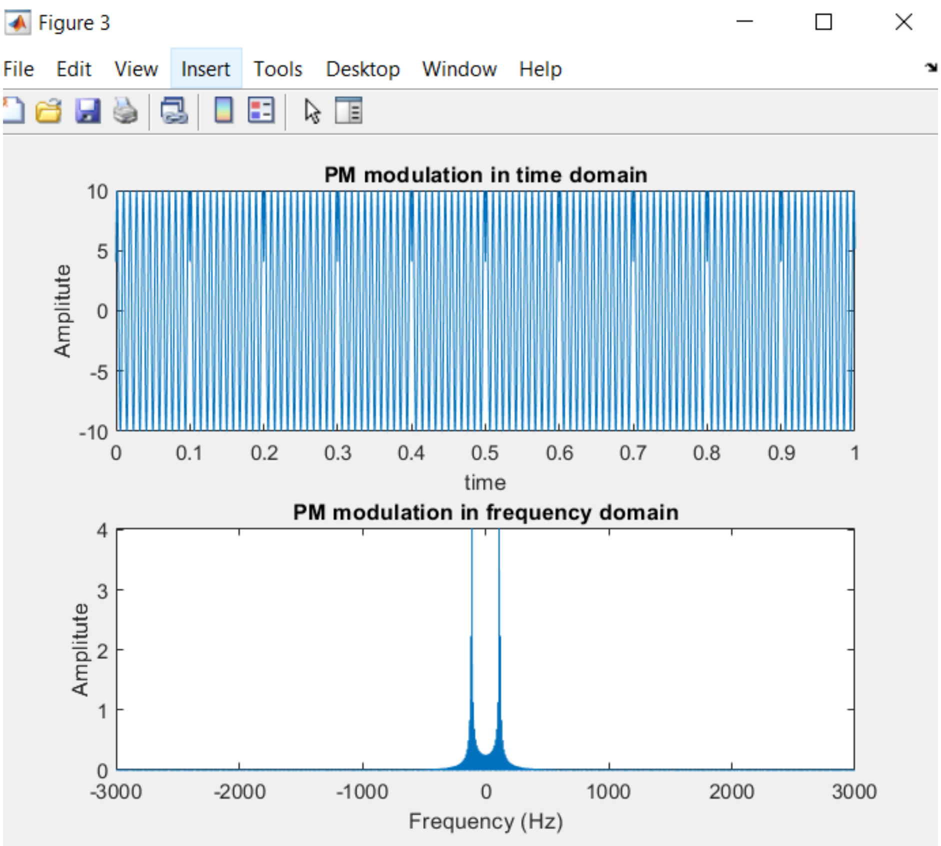
c. Triangle Wave



d. Sawtooth Wave



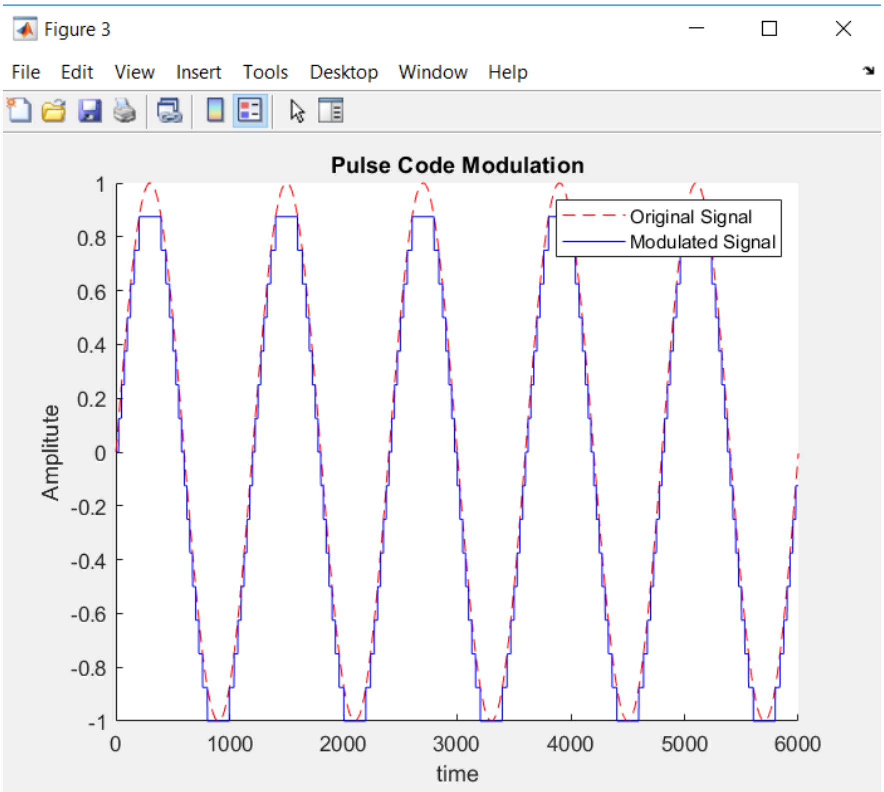
## Part 5

This part required to do Pulse Code Modulation for each type of message signal by letting the user enter the number of quantization level.

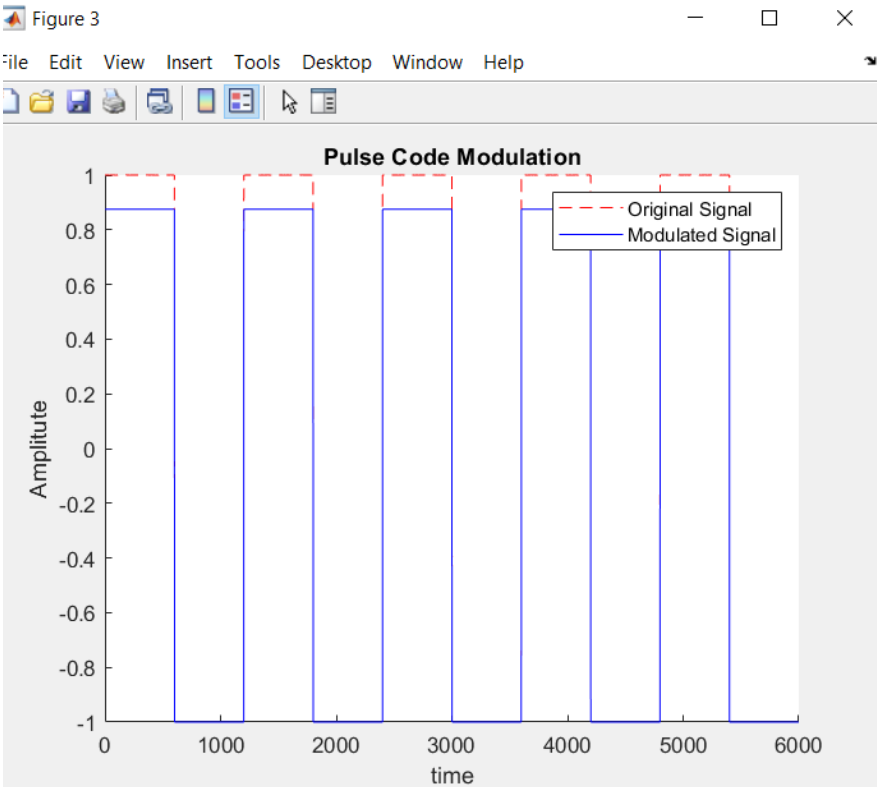
The Quantization level is chosen to be 4.

For each type of message signal, the PCM is given below

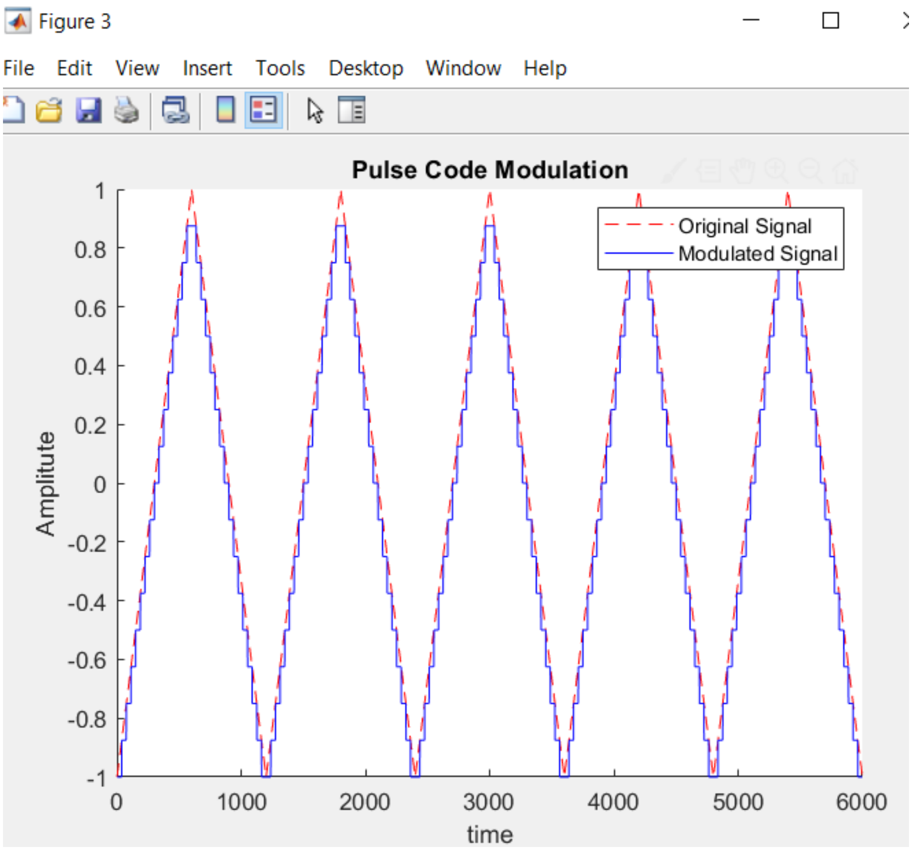
### a. Sine Wave



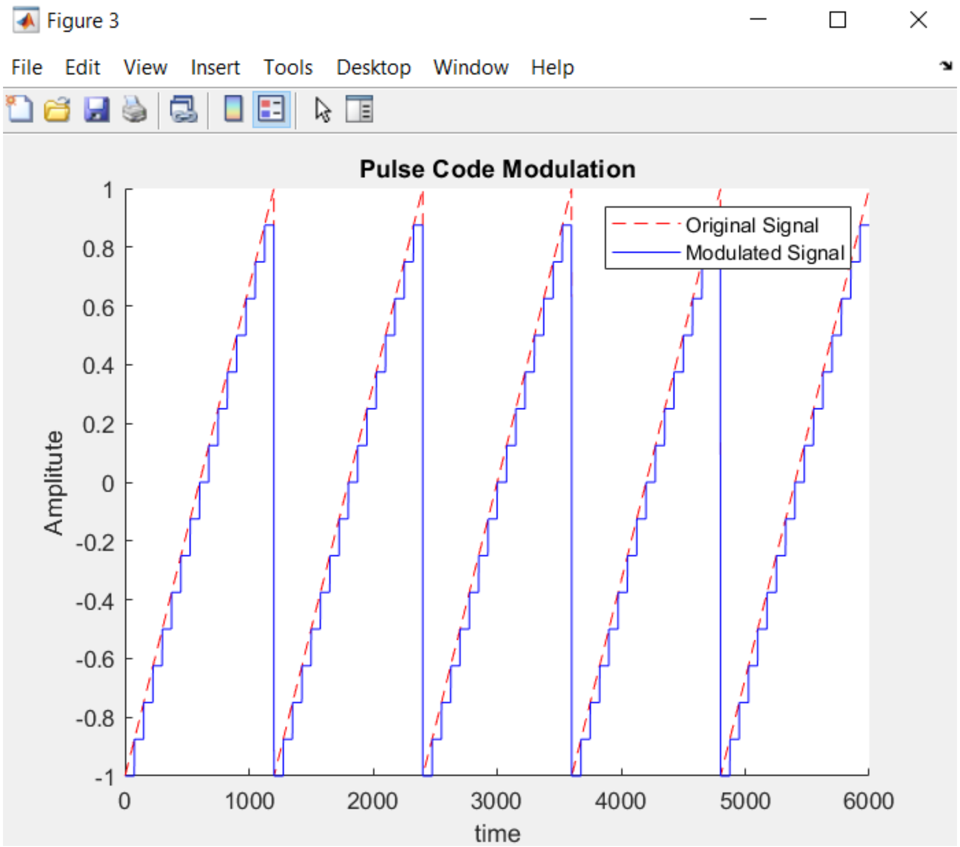
b. Square Wave




c. Triangle Wave

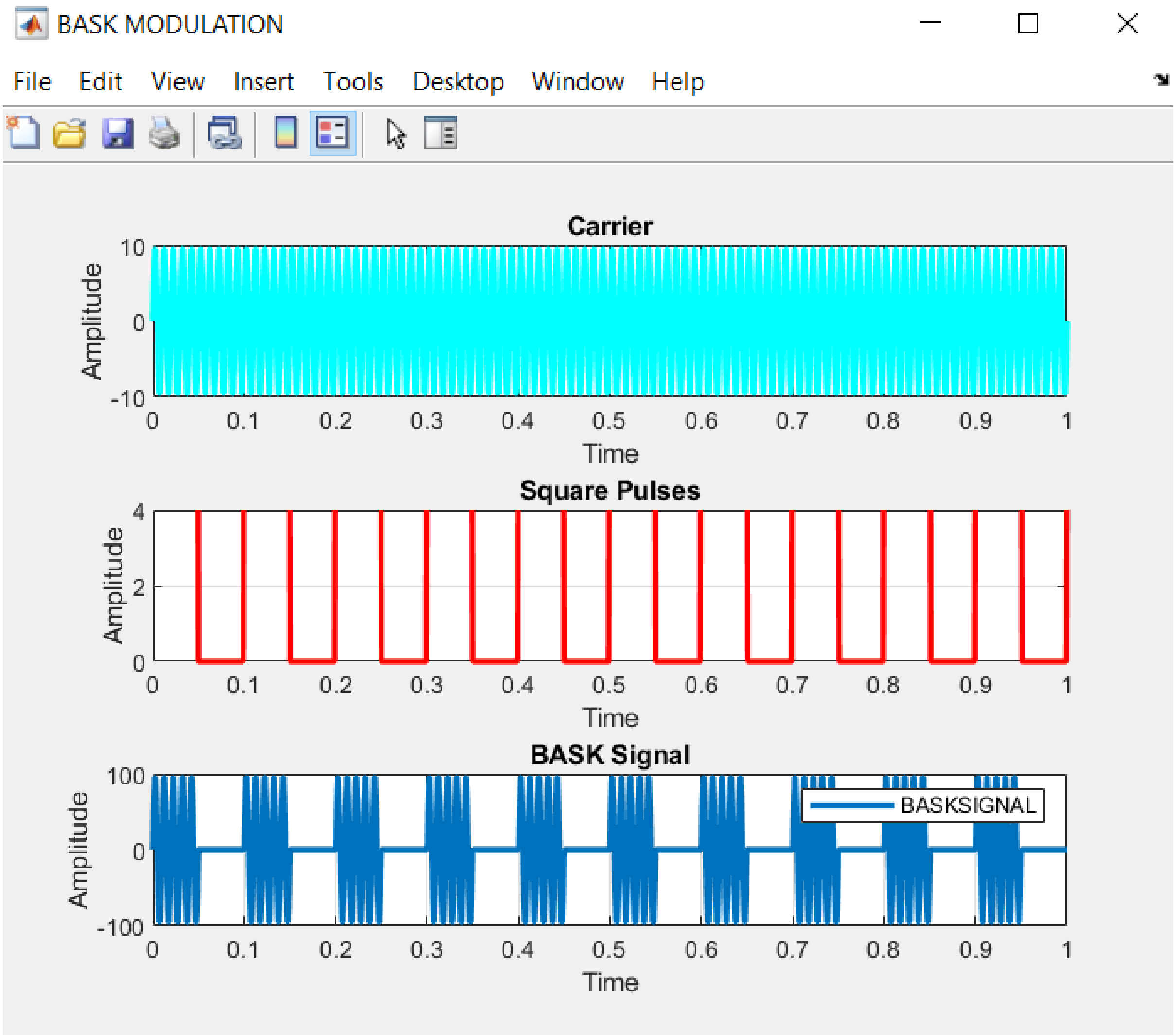


d. Sawtooth Wave



# Part 6

## 6.1 BASK



## 6.2 BFSK

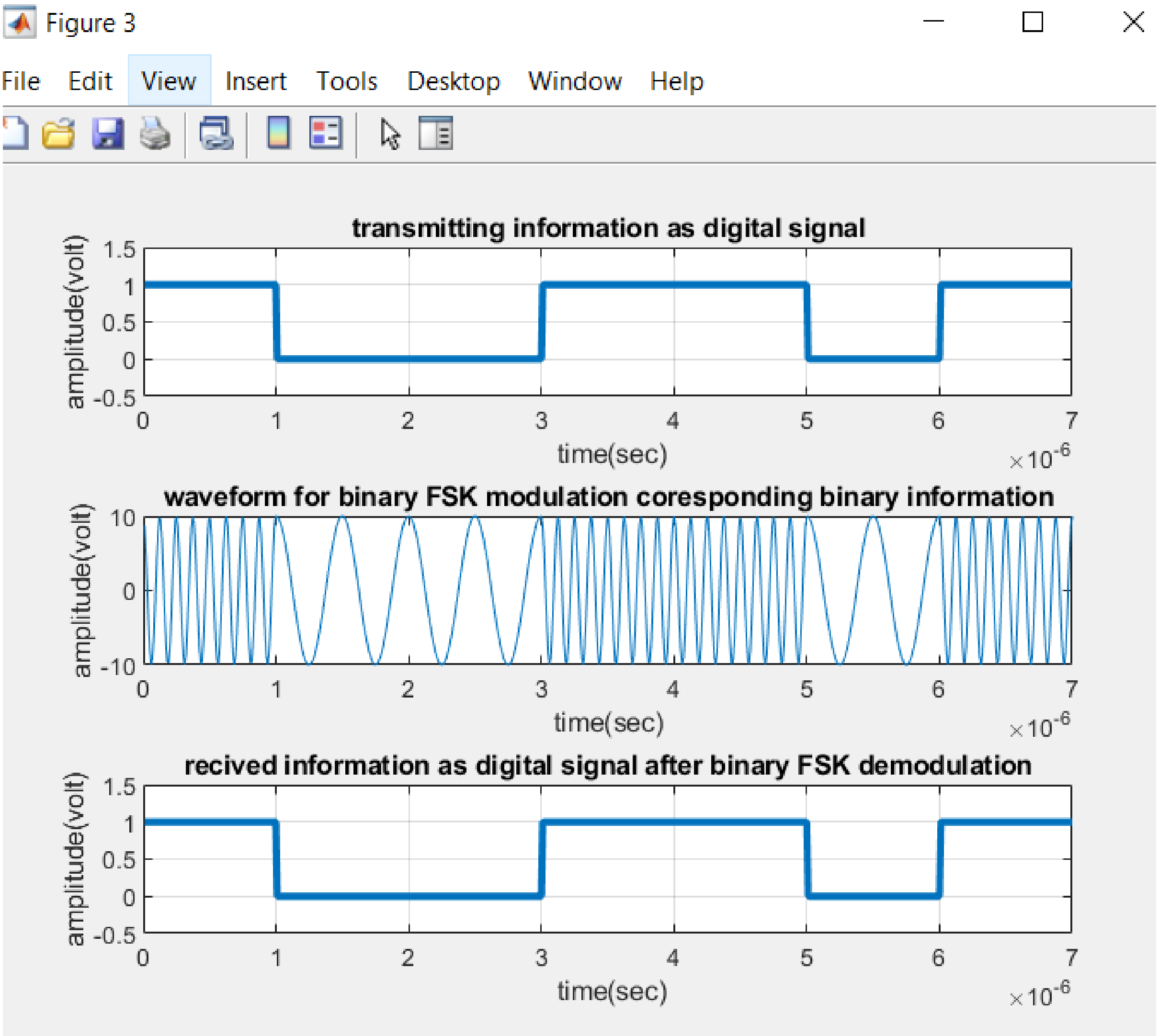


Figure 3
File Edit View Insert Tools Desktop Window Help
transmitting information as digital signal
amplitude(volt)
time(sec)
$\times 10^{-6}$
waveform for binary FSK modulation coresponding binary information
amplitude(volt)
time(sec)
$\times 10^{-6}$
recived information as digital signal after binary FSK demodulation
amplitude(volt)
time(sec)
$\times 10^{-6}$

## 6.3 BPSK

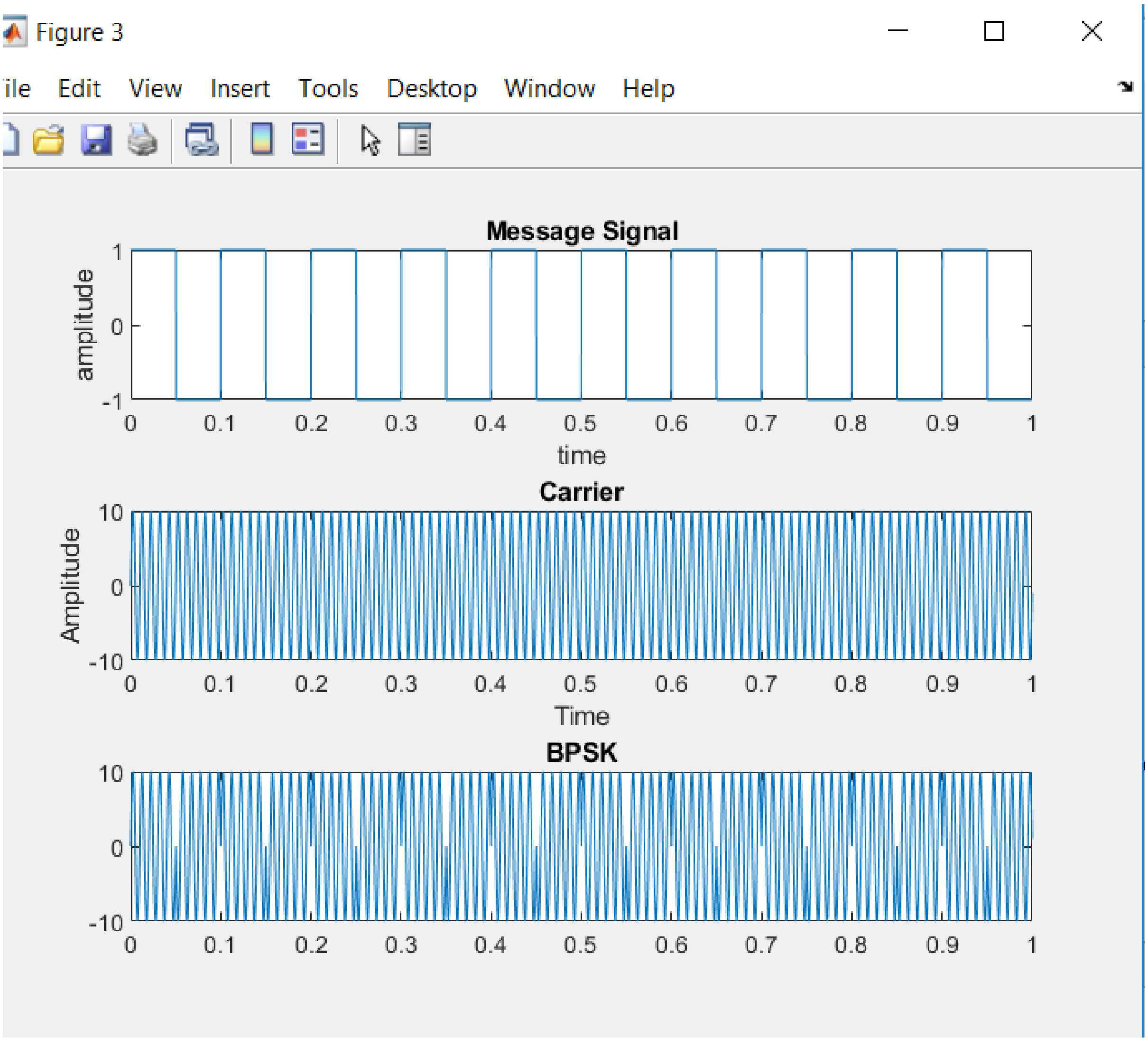


Figure 3
File Edit View Insert Tools Desktop Window Help
Message Signal
amplitude
1
0
-1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
time
Carrier
Amplitude
10
0
-10
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Time
BPSK
10
0
-10
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

## 6.4 QAM

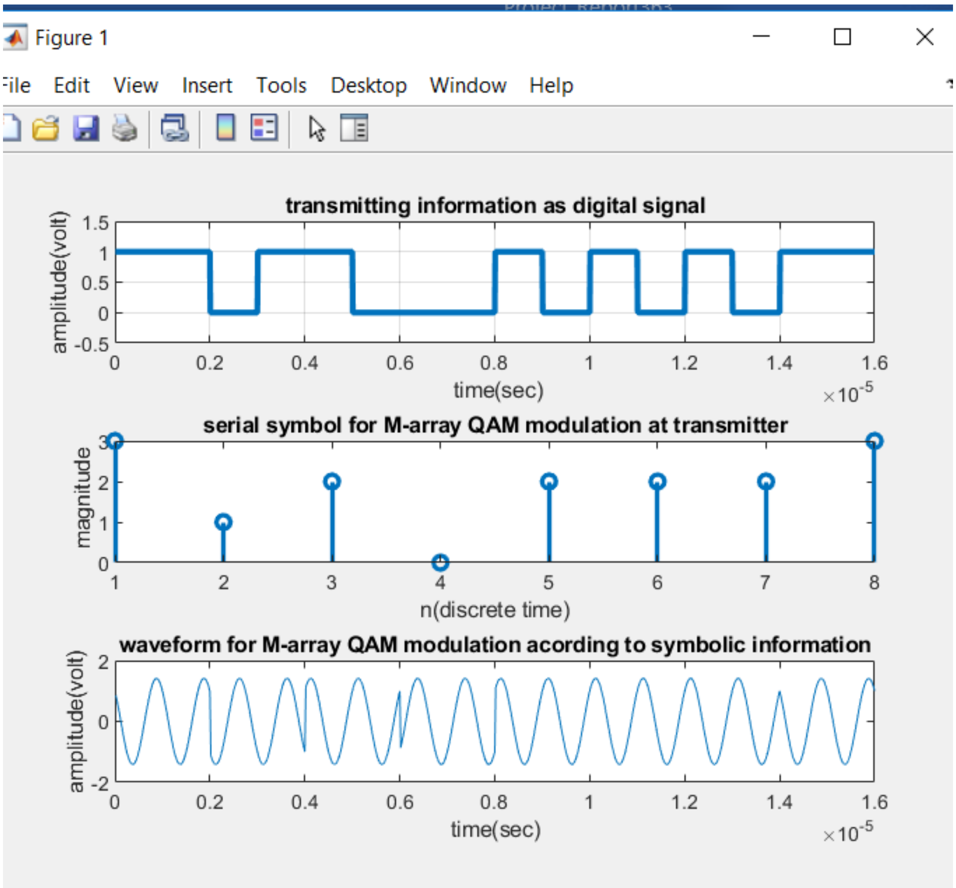


Figure 1
File Edit View Insert Tools Desktop Window Help
transmitting information as digital signal
amplitude(volt)
time(sec)
serial symbol for M-array QAM modulation at transmitter
magnitude
n(discrete time)
waveform for M-array QAM modulation acording to symbolic information
amplitude(volt)
time(sec)

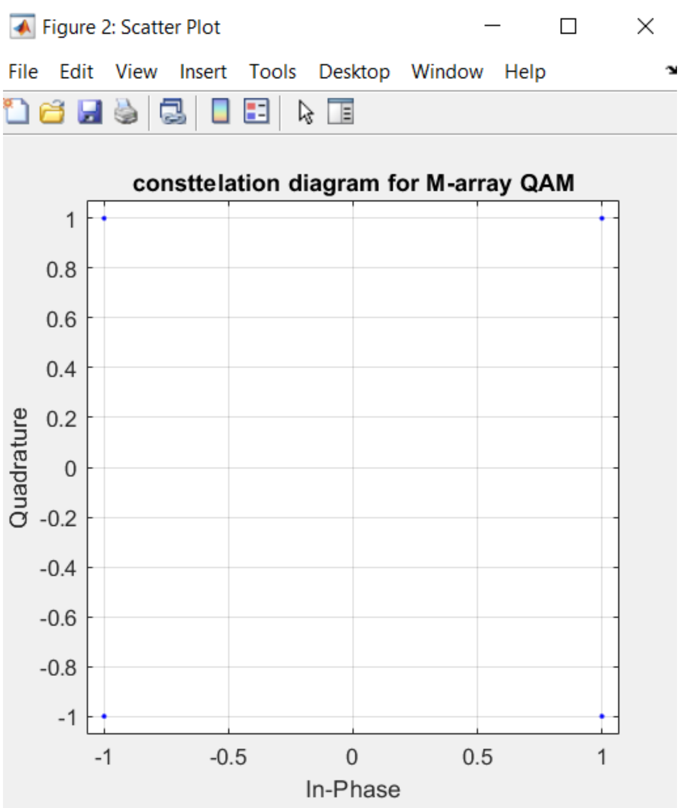



Figure 2: Scatter Plot
File Edit View Insert Tools Desktop Window Help
consttelation diagram for M-array QAM
Quadrature
In-Phase

# CODE OF THE SYSTEM

```
function varargout = starting_O(varargin)
% STARTING_O MATLAB code for starting_O.fig
%      STARTING_O, by itself, creates a new STARTING_O or raises the 
existing
%        singleton*.
%
%      H = STARTING_O returns the handle to a new STARTING_O or the handle 
to
%      the existing singleton*.
%
%      STARTING_O('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in STARTING_O.M with the given input 
arguments.
%
%      STARTING_O('Property','Value',...) creates a new STARTING_O or 
raises the
%      existing singleton*.  Starting from the left, property value pairs 
are
%      applied to the GUI before starting_O_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property 
application
%      stop.  All inputs are passed to starting_O_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 starting_O

% Last Modified by GUIDE v2.5 20-May-2019 20:53:12

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @starting_O_OpeningFcn, ...
                   'gui_OutputFcn',  @starting_O_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 starting_O is made visible.
function starting_O_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 starting_O (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = starting_O_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 amp_m_Callback(hObject, eventdata, handles)
% hObject    handle to amp_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 amp_m as text
%        str2double(get(hObject,'String')) returns contents of amp_m as a
double


% --- Executes during object creation, after setting all properties.
function amp_m_CreateFcn(hObject, eventdata, handles)
% hObject    handle to amp_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 freq_m_Callback(hObject, eventdata, handles)
% hObject    handle to freq_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 freq_m as text
%        str2double(get(hObject,'String')) returns contents of freq_m as a
double


% --- Executes during object creation, after setting all properties.
function freq_m_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to freq_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 phase_m_Callback(hObject, eventdata, handles)
% hObject    handle to phase_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 phase_m as text
%        str2double(get(hObject,'String')) returns contents of phase_m as a
double


% --- Executes during object creation, after setting all properties.
function phase_m_CreateFcn(hObject, eventdata, handles)
% hObject    handle to phase_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 amp_c_Callback(hObject, eventdata, handles)
% hObject    handle to amp_c (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 amp_c as text
%        str2double(get(hObject,'String')) returns contents of amp_c as a
double


% --- Executes during object creation, after setting all properties.
function amp_c_CreateFcn(hObject, eventdata, handles)
% hObject    handle to amp_c (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 freq_c_Callback(hObject, eventdata, handles)
% hObject    handle to freq_c (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 freq_c as text
%        str2double(get(hObject,'String')) returns contents of freq_c as a
double


% --- Executes during object creation, after setting all properties.
function freq_c_CreateFcn(hObject, eventdata, handles)
% hObject    handle to freq_c (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 phase_c_Callback(hObject, eventdata, handles)
% hObject    handle to phase_c (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 phase_c as text
%        str2double(get(hObject,'String')) returns contents of phase_c as a
double


% --- Executes during object creation, after setting all properties.
function phase_c_CreateFcn(hObject, eventdata, handles)
% hObject    handle to phase_c (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 d_carrier.
function d_carrier_Callback(hObject, eventdata, handles)
% hObject    handle to d_carrier (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


%extracting values from GUI using tag names
%and storing in variables to be used in the code
ac=str2num(get(handles.amp_c,'string')); % ac = carrier amp
fc=str2num(get(handles.freq_c,'string')); % fc = carrier freq
```

```
pc=str2num(get(handles.phase_c,'string')); % pc = carrier phase
prc=deg2rad(pc);

fs = 6000; %sampling freq
tc = 0:1/fs:1-1/fs;

%formula for carrier signal
carrier=ac*sin((2*pi*fc*tc)+prc);

%axes(handles.axes3);
figure(2);
subplot(211);
plot(tc,carrier);
title('Carrier Signal in Time Domain');
xlabel('Time'); ylabel('Amplitute');

%for displaying graphs in frequency domain
N=length(carrier);
f_carrier=linspace(-fs/2+1,fs/2,fs);
fft_carrier=fftshift(abs(fft(carrier)))/N;

%axes(handles.axes4);
subplot(212);
plot(f_carrier,fft_carrier);
title('Carrier Signal in Frequency Domain');
xlabel('Frequency (Hz)'); ylabel('Amplitute');

%to use carrier signal in other push buttons
handles.carrier=carrier;
guidata(hObject,handles);

% --- Executes on button press in sine.
function sine_Callback(hObject, eventdata, handles)
% hObject    handle to sine (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%extracting values from GUI using tag names
%and storing in variables to be used in the code
am=str2num(get(handles.amp_m,'string')); % am= message signal's amp
fm=str2num(get(handles.freq_m,'string')); % fm= message signal's freq
pm=str2num(get(handles.phase_m,'string')); % pm= message signal's phase
prm=deg2rad(pm);

fs = 6000; %sampling freq
tm = 0:1/fs:1-1/fs; %distribution of signal over this x axis

ms = am*sin((2*pi*fm*tm)+prm); %equation for sine as message signal

%axes(handles.axes1);
figure(1);
subplot(211);
plot(tm,ms);
title('Message Signal in Time Domain');
xlabel('Time'); ylabel('Amplitute');

%for displaying graphs in frequency domain
N=length(ms);
f_ms=linspace(-fs/2+1,fs/2,fs);
fft_ms=fftshift(abs(fft(ms)))/N;
```

```
%axes(handles.axes2);
subplot(212);
plot(f_ms,fft_ms);
title('Message Signal in Frequency Domain');
xlabel('Frequency (Hz)'); ylabel('Amplitute');

%to use message signal in other push buttons
handles.ms=ms;
guidata(hObject,handles);



% --- Executes on button press in square.
function square_Callback(hObject, eventdata, handles)
% hObject    handle to square (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%extracting values from GUI using tag names
%and storing in variables to be used in the code
am=str2num(get(handles.amp_m,'string')); % am= message signal's amp
fm=str2num(get(handles.freq_m,'string')); % fm= message signal's freq
pm=str2num(get(handles.phase_m,'string')); % pm= message signal's phase
prm=deg2rad(pm);

fs = 6000; %sampling freq
tm = 0:1/fs:1-1/fs; %distribution of signal over this x axis

ms=am*square((2*pi*fm*tm)+prm); %equation for square as message signal

%axes(handles.axes1);
figure(1);

subplot(211);
plot(tm,ms);
title('Message Signal in Time Domain');
xlabel('time'); ylabel('Amplitute');

%for displaying graphs in frequency domain
N=length(ms);
f_ms=linspace(-fs/2+1,fs/2,fs);
fft_ms=fftshift(abs(fft(ms)))/N;

%axes(handles.axes2);
subplot(212);
plot(f_ms,fft_ms);
title('Message Signal in Frequency Domain');

%to use message signal in other push buttons
handles.ms=ms;
guidata(hObject,handles);



% --- Executes on button press in triangle.
function triangle_Callback(hObject, eventdata, handles)
% hObject    handle to triangle (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%extracting values from GUI using tag names
```

```
%and storing in variables to be used in the code
am=str2num(get(handles.amp_m,'string')); % am= message signal's amp
fm=str2num(get(handles.freq_m,'string')); % fm= message signal's freq
pm=str2num(get(handles.phase_m,'string')); % pm= message signal's phase
prm=deg2rad(pm);

fs = 6000; %sampling freq
tm = 0:1/fs:1-1/fs; %distribution of signal over this x axis

ms=am*sawtooth(((2*pi*fm*tm)+prm),0.5); %equation for triangle as message
signal

%axes(handles.axes1);
figure(1);
subplot(211);
plot(tm,ms);
title('Message Signal in Time Domain');
xlabel('time'); ylabel('Amplitute');

%for displaying graphs in frequency domain
N=length(ms);
f_ms=linspace(-fs/2+1,fs/2,fs);
fft_ms=fftshift(abs(fft(ms)))/N;

%axes(handles.axes2);
%figure(2);
subplot(211);
plot(f_ms,fft_ms);
title('Message Signal in Frequency Domain');
xlabel('Frequency (Hz)'); ylabel('Amplitute');

%to use message signal in other push buttons
handles.ms=ms;
guidata(hObject,handles);



% --- Executes on button press in sawtooth.
function sawtooth_Callback(hObject, eventdata, handles)
% hObject    handle to sawtooth (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%extracting values from GUI using tag names
%and storing in variables to be used in the code
am=str2num(get(handles.amp_m,'string')); % am= message signal's amp
fm=str2num(get(handles.freq_m,'string')); % fm= message signal's freq
pm=str2num(get(handles.phase_m,'string')); % pm= message signal's phase
prm=deg2rad(pm);

fs = 6000; %sampling freq
tm = 0:1/fs:1-1/fs; %distribution of signal over this x axis

ms=am*sawtooth((2*pi*fm*tm)+ prm); %equation for sawtooth as message signal

%axes(handles.axes1);
figure(1);

subplot(211);
plot(tm,ms);
title('Message Signal in Time Domain');
```

```
xlabel('time'); ylabel('Amplitute');

%for displaying graphs in frequency domain
N=length(ms);
f_ms=linspace(-fs/2+1,fs/2,fs);
fft_ms=fftshift(abs(fft(ms)))/N;

%axes(handles.axes2);
%figure(2);
subplot(212);
plot(f_ms,fft_ms);
title('Message Signal in Frequency Domain');
xlabel('Frequency (Hz)'); ylabel('Amplitute');

%to use message signal in other push buttons
handles.ms=ms;
guidata(hObject,handles);


function a_mi_Callback(hObject, eventdata, handles)
% hObject    handle to a_mi (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 a_mi as text
%        str2double(get(hObject,'String')) returns contents of a_mi as a
double


% --- Executes during object creation, after setting all properties.
function a_mi_CreateFcn(hObject, eventdata, handles)
% hObject    handle to a_mi (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 kf_Callback(hObject, eventdata, handles)
% hObject    handle to kf (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 kf as text
%        str2double(get(hObject,'String')) returns contents of kf as a
double


% --- Executes during object creation, after setting all properties.
function kf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to kf (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 p_kp_Callback(hObject, eventdata, handles)
% hObject    handle to p_kp (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 p_kp as text
%        str2double(get(hObject,'String')) returns contents of p_kp as a
double


% --- Executes during object creation, after setting all properties.
function p_kp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to p_kp (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 a_modulated.
function a_modulated_Callback(hObject, eventdata, handles)
% hObject    handle to a_modulated (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%extracting values from GUI using tag names
%and storing in variables to be used in the code
ac=str2num(get(handles.amp_c,'string')); %ac= carrier's amplitude
am=str2num(get(handles.amp_m,'string')); %am= message signal's amplitude
fc=str2num(get(handles.freq_c,'string')); %fc= carrier's frequency
modind=str2num(get(handles.a_mi,'string')); %modind= modulation index for
AM

ka=modind/am; %relationship between modulation index and ka

fs = 6000; %sampling freq
t = 0:1/fs:1-1/fs; %distribution of signal over this x axis

%retrieving msg signal from other push button
ms=handles.ms;
mt=ms;

%retrieving carrier signal from other push button
carrier=handles.carrier;
ct=carrier;

amp_mod=ct.*(1+ka*mt); %equation for amplitude modulation

figure(3);
subplot(211);
```

```
plot(t,amp_mod);
title('AM modulation in time domain');
xlabel('time'); ylabel('Amplitute');

%for displaying graphs in frequency domain
N=length(amp_mod);
f_amod=linspace(-fs/2+1,fs/2,fs);
fft_amod=fftshift(abs(fft(amp_mod)))/N;


%figure(2);
subplot(212);
plot(f_amod,fft_amod);
title('AM modulation in Frequency Domain');


% --- Executes on button press in dsb_fc.
function dsb_fc_Callback(hObject, eventdata, handles)
% hObject    handle to dsb_fc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%extracting values from GUI using tag names
%and storing in variables to be used in the code
ac=str2num(get(handles.amp_c,'string')); %ac= carrier's amplitude
fc=str2num(get(handles.freq_c,'string')); %fc= carrier's freq
fm=str2num(get(handles.freq_m,'string')); %fm= message signal's freq
modind=str2num(get(handles.a_mi,'string')); %modind= modulation index for
AM

fs = 6000; %sampling freq
t = 0:1/fs:1-1/fs; %distribution of signal over this x axis

%eqaution for double sideband full carrier
dsbfc=ac*(1+(modind*cos(2*pi*fm*t))).*cos(2*pi*fc*t);

figure(4);
subplot(211);
plot(t,dsbfc);
title('DSB-FC in time domain');
xlabel('time'); ylabel('Amplitute');

%for displaying graphs in frequency domain
N=length(dsbfc);
f_dsbfc=linspace(-fs/2+1,fs/2,fs);
fft_dsbfc=fftshift(abs(fft(dsbfc)))/N;

subplot(212);
plot(f_dsbfc,fft_dsbfc);
title('DSB-FC in frequency domain');

% --- Executes on button press in dsb_sc.
function dsb_sc_Callback(hObject, eventdata, handles)
% hObject    handle to dsb_sc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%extracting values from GUI using tag names
%and storing in variables to be used in the code
ac=str2num(get(handles.amp_c,'string'));  %ac= carrier's amplitude
fc=str2num(get(handles.freq_c,'string')); %fc= carrier's freq
```

```
fs = 6000; %sampling freq
t = 0:1/fs:1-1/fs; %distribution of signal over this x axis

%retrieve msg signal
ms=handles.ms
mt=ms;

%eqaution for double sideband suppressed carrier
dsbsc=(ac*mt).*cos(2*pi*fc*t);

figure(5);
subplot(211);
plot(t,dsbsc);
title('DSB-SC in time domain');
xlabel('time'); ylabel('Amplitute');

%for displaying graphs in frequency domain
N=length(dsbsc);
f_dsbsc=linspace(-fs/2+1,fs/2,fs);
fft_dsbsc=fftshift(abs(fft(dsbsc)))/N;

subplot(212);
plot(f_dsbsc,fft_dsbsc);
title('DSB-SC in frequency domain');


% --- Executes on button press in ssb_scu.
function ssb_scu_Callback(hObject, eventdata, handles)
% hObject    handle to ssb_scu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%extracting values from GUI using tag names
%and storing in variables to be used in the code
ac=str2num(get(handles.amp_c,'string'));  %ac= carrier's amplitude
fc=str2num(get(handles.freq_c,'string')); %fc= carrier's freq

fs = 6000; %sampling freq
t = 0:1/fs:1-1/fs; %distribution of signal over this x axis

%retrieving msg signal
ms=handles.ms;
mt=ms;

mth=mt.*(1/pi*t); %hilbert transform of msg signal

%equation for single sideband suppressed carrier (upper)
ssbscu=((0.5*ac*mt).*cos(2*pi*fc*t)) - ((0.5*ac*mth).*sin(2*pi*fc*t));

figure(6);
subplot(211);
plot(t,ssbscu);
title('SSBSCU in time domain');
xlabel('time'); ylabel('Amplitute');

%for displaying graphs in frequency domain
N=length(ssbscu);
f_ssbscu=linspace(-fs/2+1,fs/2,fs);
fft_ssbscu=fftshift(abs(fft(ssbscu)))/N;
```

```
%figure;
subplot(212);
plot(f_ssbscu,fft_ssbscu);
title('SSB-SCU in frequency domain');


% --- Executes on button press in ssb_scl.
function ssb_scl_Callback(hObject, eventdata, handles)
% hObject    handle to ssb_scl (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%extracting values from GUI using tag names
%and storing in variables to be used in the code
ac=str2num(get(handles.amp_c,'string'));  %ac= carrier's amplitude
fc=str2num(get(handles.freq_c,'string')); %fc= carrier's freq

fs = 6000; %sampling freq
t = 0:1/fs:1-1/fs; %distribution of signal over this x axis;

%retrieving msg signal
ms=handles.ms;
mt=ms;

mth=mt.*(1/pi*t); %hilbert transform of msg signal

%equation for single sideband suppressed carrier (lower)
ssbscl=((0.5*ac*mt).*cos(2*pi*fc*t)) + ((0.5*ac*mth).*sin(2*pi*fc*t));

figure(7);
subplot(211);
plot(t,ssbscl);
title('SSB-SCL in time domain');
xlabel('time'); ylabel('Amplitute');

%for displaying graphs in frequency domain
N=length(ssbscl);
f_ssbscl=linspace(-fs/2+1,fs/2,fs);
fft_ssbscl=fftshift(abs(fft(ssbscl)))/N;

%figure;
subplot(212);
plot(f_ssbscl,fft_ssbscl);
title('SSB-SCL in frequency domain');


% --- Executes on button press in p_modulated.
function p_modulated_Callback(hObject, eventdata, handles)
% hObject    handle to p_modulated (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%extracting values from GUI using tag names
%and storing in variables to be used in the code
ac=str2num(get(handles.amp_c,'string'));  %ac= carrier's amplitude
fc=str2num(get(handles.freq_c,'string')); %fc= carrier's freq
kp=str2num(get(handles.p_kp,'string')); %sensitivity for the phase
modulation

fs = 6000; %sampling freq
t = 0:1/fs:1-1/fs; %distribution of signal over this x axis
```

```
%retrieving msg signal
ms=handles.ms;
mt=ms;

phase_mod= ac.*cos((2*pi*fc*t)+(kp*mt)); %equation for phase modulation

figure;
subplot(211);
plot(t,phase_mod);
title('PM modulation in time domain');

%for displaying graphs in frequency domain
N=length(phase_mod);
f_phase_mod=linspace(-fs/2+1,fs/2,fs);
fft_phase_mod=fftshift(abs(fft(phase_mod)))/N;

subplot(212);
plot(f_phase_mod,fft_phase_mod);
title('PM modulation in frequency domain');


% --- Executes on button press in fm_sine.
function fm_sine_Callback(hObject, eventdata, handles)
% hObject    handle to fm_sine (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function pcm_level_Callback(hObject, eventdata, handles)
% hObject    handle to pcm_level (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 pcm_level as text
%        str2double(get(hObject,'String')) returns contents of pcm_level as
a double


% --- Executes during object creation, after setting all properties.
function pcm_level_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pcm_level (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 bask_carrier_Callback(hObject, eventdata, handles)
% hObject    handle to bask_carrier (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 bask_carrier as text
%        str2double(get(hObject,'String')) returns contents of bask_carrier
as a double


% --- Executes during object creation, after setting all properties.
function bask_carrier_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bask_carrier (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 bask_message_Callback(hObject, eventdata, handles)
% hObject    handle to bask_message (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 bask_message as text
%        str2double(get(hObject,'String')) returns contents of bask_message
as a double


% --- Executes during object creation, after setting all properties.
function bask_message_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bask_message (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 bask_amplitude_Callback(hObject, eventdata, handles)
% hObject    handle to bask_amplitude (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 bask_amplitude as text
%        str2double(get(hObject,'String')) returns contents of
bask_amplitude as a double


% --- Executes during object creation, after setting all properties.
function bask_amplitude_CreateFcn(hObject, eventdata, handles)
% hObject    handle to bask_amplitude (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 BASK_buttn.
function BASK_buttn_Callback(hObject, eventdata, handles)
% hObject    handle to BASK_buttn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ac=str2num(get(handles.amp_c,'string'));  % Amplitude
A=ac;

fc=str2num(get(handles.freq_c,'string'));  % Amplitude
F1=fc;


fm=str2num(get(handles.freq_m,'string'));  % Amplitude
F2=fm;

t=0:0.001:1;            %Sampling Interval time 0 ? t ? 1
x=A.*sin(2*pi*F1*t);    %Carrier Sine wave A sin(2?fct) fc:frequency of
carrier
u=A/2.*square(2*pi*F2*t)+(A/2); %Square wave message with peak of amplitude
A and and peak of zero (binary data)
v=x.*u;      %Modulation Process by multiply message with carrier v(t) = A
u(t) sin 2?fct
%Plot Carrier Signal
figure('name','BASK MODULATION','numbertitle','off');
subplot(3,1,1);
plot(t,x,'c','linewidth',2);
xlabel('Time');
ylabel('Amplitude');
title('Carrier');
grid on;
%Plot Message Singal
subplot(3,1,2);
plot(t,u,'r','linewidth',2);
xlabel('Time');
ylabel('Amplitude');
title('Square Pulses');
axis([0 1 0 4]);
grid on;
%Plot Modulated Signal
subplot(3,1,3);
plot(t,v,'linewidth',2);
xlabel('Time');
ylabel('Amplitude');
title('BASK Signal');
legend('BASKSIGNAL');
grid on;



% --- Executes on button press in bfsk_btn.
function bfsk_btn_Callback(hObject, eventdata, handles)
% hObject    handle to bfsk_btn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

ac=str2num(get(handles.amp_c,'string'));  % Amplitude
A=ac; %the amplitude of msg and carrier should be same in this technique

x=[ 1 0 0 1 1 0 1];                                      % Binary Information
bp=.000001;                                                      % bit period

%XX representation of transmitting binary information as digital signal XXX
bit=[];
for n=1:1:length(x)
    if x(n)==1;
       se=ones(1,100);
    else x(n)==0;
        se=zeros(1,100);
    end
     bit=[bit se];
end
t1=bp/100:bp/100:100*length(x)*(bp/100);
figure;
subplot(3,1,1);
plot(t1,bit,'lineWidth',2.5);grid on;
axis([ 0 bp*length(x) -.5 1.5]);
ylabel('amplitude(volt)');
xlabel(' time(sec)');
title('transmitting information as digital signal');
%XXXXXXXXXXXXXXXXXXXXXXX Binary-FSK modulation XXXXXXXXXXXXXXXXXXXXXXXXXXX%
br=1/bp;                                                           % bit rate
f1=br*8;                           % carrier frequency for information as 1
f2=br*2;                           % carrier frequency for information as 0
t2=bp/99:bp/99:bp;
ss=length(t2);
m=[];
for (i=1:1:length(x))
    if (x(i)==1)
        y=A*cos(2*pi*f1*t2);
    else
        y=A*cos(2*pi*f2*t2);
    end
    m=[m y];
end
t3=bp/99:bp/99:bp*length(x);
subplot(3,1,2);
plot(t3,m);
xlabel('time(sec)');
ylabel('amplitude(volt)');
title('waveform for binary FSK modulation coresponding binary
information');
%XXXXXXXXXXXXXXXXXXXX Binary FSK demodulation XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
mn=[];
for n=ss:ss:length(m)
  t=bp/99:bp/99:bp;
  y1=cos(2*pi*f1*t);                      % carrier siignal for information 1
  y2=cos(2*pi*f2*t);                      % carrier siignal for information 0
  mm=y1.*m((n-(ss-1)):n);
  mmm=y2.*m((n-(ss-1)):n);
  t4=bp/99:bp/99:bp;
  z1=trapz(t4,mm)                                                 % intregation
  z2=trapz(t4,mmm)                                                % intregation
  zz1=round(2*z1/bp)
  zz2= round(2*z2/bp)
```

```
  if(zz1>A/2)      % logic lavel= (0+A)/2 or (A+0)/2 or 2.5 ( in this case)
    a=1;
  else(zz2>A/2)
    a=0;
  end
  mn=[mn a];
end
disp(' Binary information at Reciver :');
disp(mn);
%XXXXX Representation of binary information as digital signal which achived
%after demodulation XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
bit=[];
for n=1:length(mn);
    if mn(n)==1;
       se=ones(1,100);
    else mn(n)==0;
        se=zeros(1,100);
    end
     bit=[bit se];
end
t4=bp/100:bp/100:100*length(mn)*(bp/100);
subplot(3,1,3)
plot(t4,bit,'LineWidth',2.5);grid on;
axis([ 0 bp*length(mn) -.5 1.5]);
ylabel('amplitude(volt)');
xlabel(' time(sec)');
title('recived information as digital signal after binary FSK
demodulation');




% --- Executes on button press in bpsk_btn.
function bpsk_btn_Callback(hObject, eventdata, handles)
% hObject    handle to bpsk_btn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ac=str2num(get(handles.amp_c,'string'));  % Amplitude
fc=str2num(get(handles.freq_c,'string'));  % Amplitude
fm=str2num(get(handles.freq_m,'string'));  % Amplitude


fs=6000;
t=0:1/fs:1-1/fs;

amp=ac; %for this modulation technique both msg and carrier amplitudes
should be same

%retrieving carrier signal from other push button
carrier=handles.carrier;
ct=carrier;

figure(3);
subplot(312) %For Plotting The Carrier wave
plot(t,ct)
xlabel('Time')
ylabel('Amplitude')
title('Carrier')

% For Plotting binary signal
bwave=square(2*pi*fm*t);
```

```
subplot(311)
plot(t,bwave)
xlabel('time')
ylabel('amplitude')
title('binary Signal')

% carrier wave multiplied with msg wave in order to generate PSK
bpsk=ct.*bwave;

subplot(313) % For Plotting BPSK (Phase Shift Keyed) signal
plot(t,bpsk)
title('BPSK')



function qam_n_Callback(hObject, eventdata, handles)
% hObject    handle to qam_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 qam_n as text
%        str2double(get(hObject,'String')) returns contents of qam_n as a
double


% --- Executes during object creation, after setting all properties.
function qam_n_CreateFcn(hObject, eventdata, handles)
% hObject    handle to qam_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 qam_btn.
function qam_btn_Callback(hObject, eventdata, handles)
% hObject    handle to qam_btn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
M=str2num(get(handles.qam_n,'string'));  %N bits

%%Ld=log2(M);
%%ds=ceil(Ld);
%%dif=ds-Ld;
%%if(dif~=0)
%%%%%  error('the value of M is only acceptable if log2(M)is an integer');
%%end

%%%%%XXXXXXXXXXXXXXXXXXX binary Information Generation
XXXXXXXXXXXXXXXXXXXXXXXX
nbit=16;                                          %number of information bits
msg=round(rand(nbit,1));             % information generation as binary form
disp(' binary information at transmitter ');
disp(msg);
fprintf('\n\n');
%XX representation of transmitting binary information as digital signal XXX
```

```
x=msg;
bp=.000001;                                                    % bit period
bit=[];
for n=1:1:length(x)
    if x(n)==1;
       se=ones(1,100);
    else x(n)==0;
        se=zeros(1,100);
    end
     bit=[bit se];
end
t1=bp/100:bp/100:100*length(x)*(bp/100);
figure(1)
subplot(3,1,1);
plot(t1,bit,'lineWidth',2.5);grid on;
axis([ 0 bp*length(x) -.5 1.5]);
ylabel('amplitude(volt)');
xlabel(' time(sec)');
title('transmitting information as digital signal');
% binary information convert into symbolic form for M-array QAM modulation
M=M;                                                % order of QAM modulation
msg_reshape=reshape(msg,log2(M),nbit/log2(M))';
disp(' information are reshaped for convert symbolic form');
disp(msg_reshape);
fprintf('\n\n');
size(msg_reshape);
for(j=1:1:nbit/log2(M))
   for(i=1:1:log2(M))
       a(j,i)=num2str(msg_reshape(j,i));
   end
end
as=bin2dec(a);
ass=as';
figure(1)
subplot(3,1,2);
stem(ass,'Linewidth',2.0);
title('serial symbol for M-array QAM modulation at transmitter');
xlabel('n(discrete time)');
ylabel(' magnitude');
disp('symbolic form information for M-array QAM ');
disp(ass);
fprintf('\n\n');
%XXXXXXXXXXXXXX Mapping for M-array QAM modulation XXXXXXXXXXXXXXXXXXXXXXXX
M=M;                                                  %order of QAM modulation
x1=[0:M-1];
p=qammod(ass,M)      %constalation design for M-array QAM acording to symbol
sym=0:1:M-1;      % considerable symbol of M-array QAM, just for scatterplot
pp=qammod(sym,M);                     %constalation diagram for M-array QAM
scatterplot(pp),grid on;
title('consttelation diagram for M-array QAM');
%XXXXXXXXXXXXXXXXXXXXXXX  M-array QAM modulation XXXXXXXXXXXXXXXXXXXXXXXXXXX
RR=real(p)
II=imag(p)
sp=bp*2;                                        %symbol period for M-array QAM
sr=1/sp;                                                       % symbol rate
f=sr*2;
t=sp/100:sp/100:sp;
ss=length(t);
m=[];
for(k=1:1:length(RR))
    yr=RR(k)*cos(2*pi*f*t);                      % inphase or real component
```

```
    yim=II(k)*sin(2*pi*f*t);            % Quadrature or imagenary component
    y=yr+yim;
    m=[m y];
end
tt=sp/100:sp/100:sp*length(RR);
figure(1);
subplot(3,1,3);
plot(tt,m);
title('waveform for M-array QAM modulation acording to symbolic
information');
xlabel('time(sec)');
ylabel('amplitude(volt)');
%XXXXXXXXXXXXXXXXXXXX M-array QAM demodulation XXXXXXXXXXXXXXXXXXXXXXXXXXX
m1=[];
m2=[];
for n=ss:ss:length(m)
  t=sp/100:sp/100:sp;
  y1=cos(2*pi*f*t);                                   % inphase component
  y2=sin(2*pi*f*t);                                % quadrature component
  mm1=y1.*m((n-(ss-1)):n);
  mm2=y2.*m((n-(ss-1)):n);
  z1=trapz(t,mm1)                                           % integration
  z2=trapz(t,mm2)                                           % integration
  zz1=round(2*z1/sp)
  zz2=round(2*z2/sp)
  m1=[m1 zz1]
  m2=[m2 zz2]
end
%XXXXXXXXXXXXXXXXXXX de-mapping for M-array QAM modulation XXXXXXXXXXXXXXXX
clear i;
clear j;
for (k=1:1:length(m1))
gt(k)=m1(k)+j*m2(k);
end
gt
ax=qamdemod(gt,M);
figure(3);
subplot(2,1,1);
stem(ax,'linewidth',2);
title(' re-obtain symbol after M-array QAM demodulation ');
xlabel('n(discrete time)');
ylabel(' magnitude');
disp('re-obtain symbol after M-array QAM demodulation ');
disp(ax);
fprintf('\n\n');
bi_in=dec2bin(ax);
[row col]=size(bi_in);
p=1;
 for(i=1:1:row)
     for(j=1:1:col)
         re_bi_in(p)=str2num(bi_in(i,j));
         p=p+1;
     end
 end
disp('re-obtain binary information after M-array QAM demodulation');
disp(re_bi_in')
fprintf('\n\n');
%XX representation of receiving binary information as digital signal XXXXXX
x=re_bi_in;
bp=.000001;                                                  % bit period
bit=[];
```

```
for n=1:1:length(x)
    if x(n)==1;
       se=ones(1,100);
    else x(n)==0;
        se=zeros(1,100);
    end
     bit=[bit se];
end
t1=bp/100:bp/100:100*length(x)*(bp/100);
figure(3)
subplot(2,1,2);
plot(t1,bit,'lineWidth',2.5);grid on;
axis([ 0 bp*length(x) -.5 1.5]);
ylabel('amplitude(volt)');
xlabel(' time(sec)');
title('receiving information as digital signal after M-array QAM
demoduation');




% --- Executes on button press in pcm_mod.
function pcm_mod_Callback(hObject, eventdata, handles)
% hObject    handle to pcm_mod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

qlevel=str2num(get(handles.pcm_level,'string')); %getting number of
quantization levels from GUI


fs = 6000; %sampling freq
t = 0:1/fs:1-1/fs; %distribution of signal over this x axis

%retrieving msg signal
ms=handles.ms;
mt=ms;

e1 = uencode(mt,qlevel);
d1 = udecode(e1,qlevel);

figure(3) %%in this case the step size is constrant so tracking
hold on
plot(mt,'--red');
plot(d1,'blue');
title('Pulse Code Modulation');
xlabel('time'); ylabel('Amplitute');

hleg = legend('Original Signal','Modulated Signal')
hold off
```

# CONTRIBUTION SHEET

## Assignments - Contributions Sheet

<u>This sheet must be attached (electronically scanned) to your group report submission.</u>

There have been a number of instances where particular group members have not engaged in the project to an adequate level (sometimes making no contribution).

To take account of this, each group is required to list the percentage contribution of each group member (totalling 100%).

Each group member must sign this sheet in acknowledgment that they accept the contributions listed against them.

In the case where there are differences in the level of contribution by group members, the final individual group mark will be adjusted up or down based on the procedure explained on page 2 of this document.

**Group number:** Team B

| | Group Member Name | Percentage Contribution | Signature |
|---|---|---|---|
| **1** | Umm Kulsoom Emad 5529657 | 33.33% | |
| **2** | Jean Xavier 5567828 | 33.33% | |
| **3** | Ummer Sheriff 5600376 | 33.33% | |
| **4** | | | |
| **5** | | | |