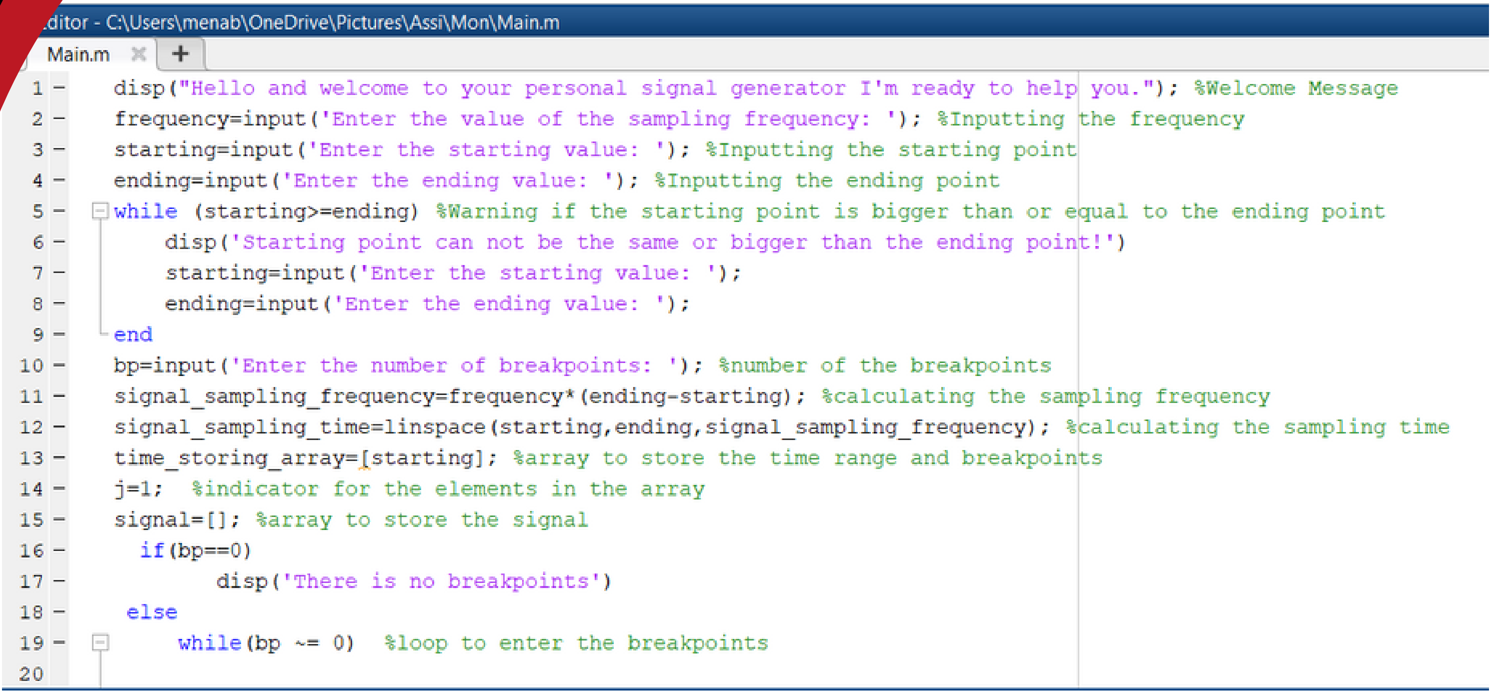
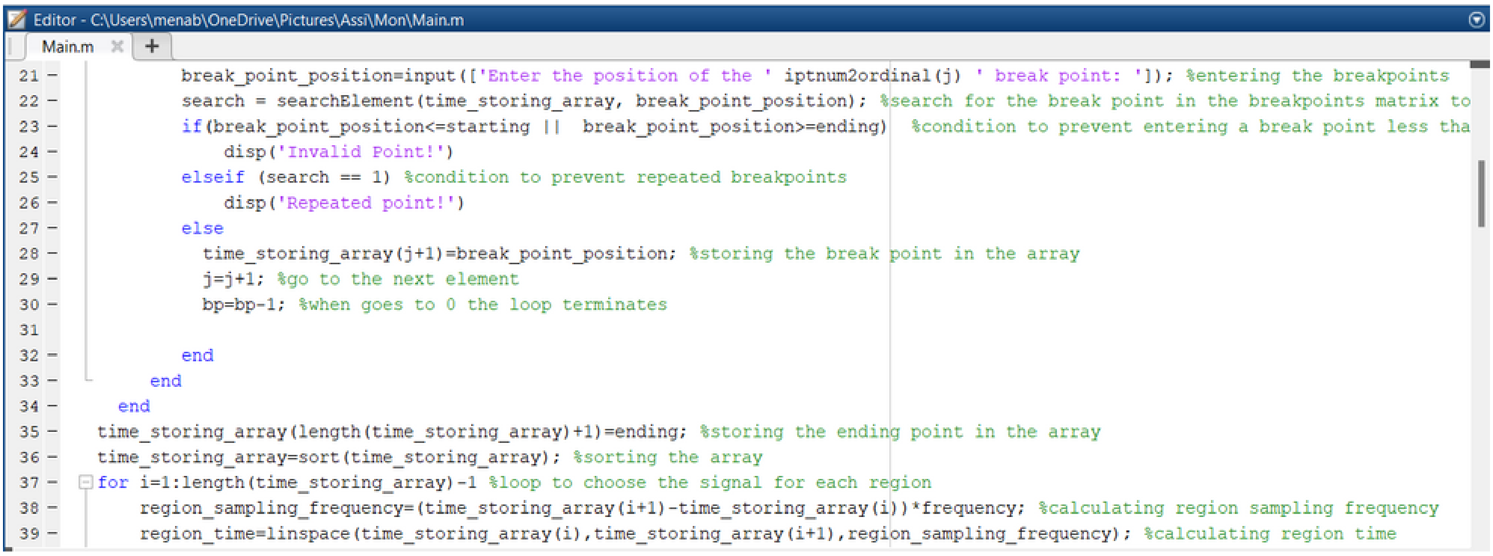
MATLAB MINI PROJECT

Mena Badr Helmy	21011445
Shehab Eldeen Ahmed Gaber	21010667
Mariam Mostafa Mohamed	21011334
Shaimaa Ahmed Mostafa	21010683

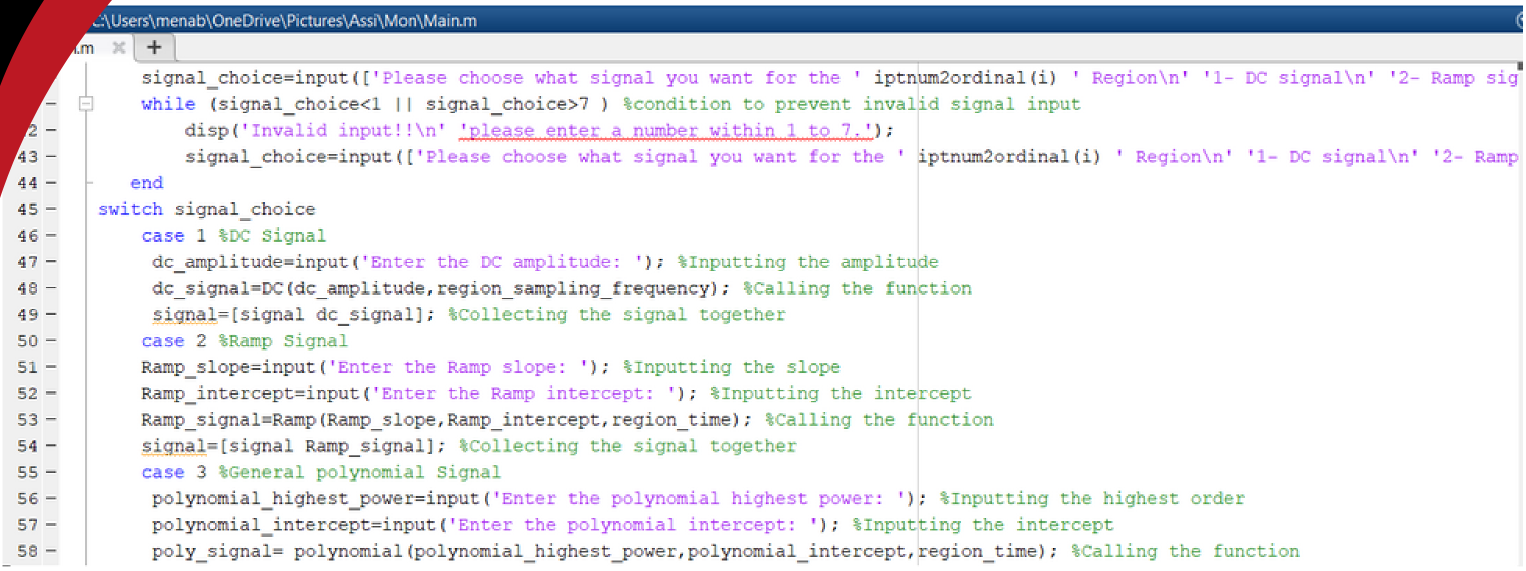
code:



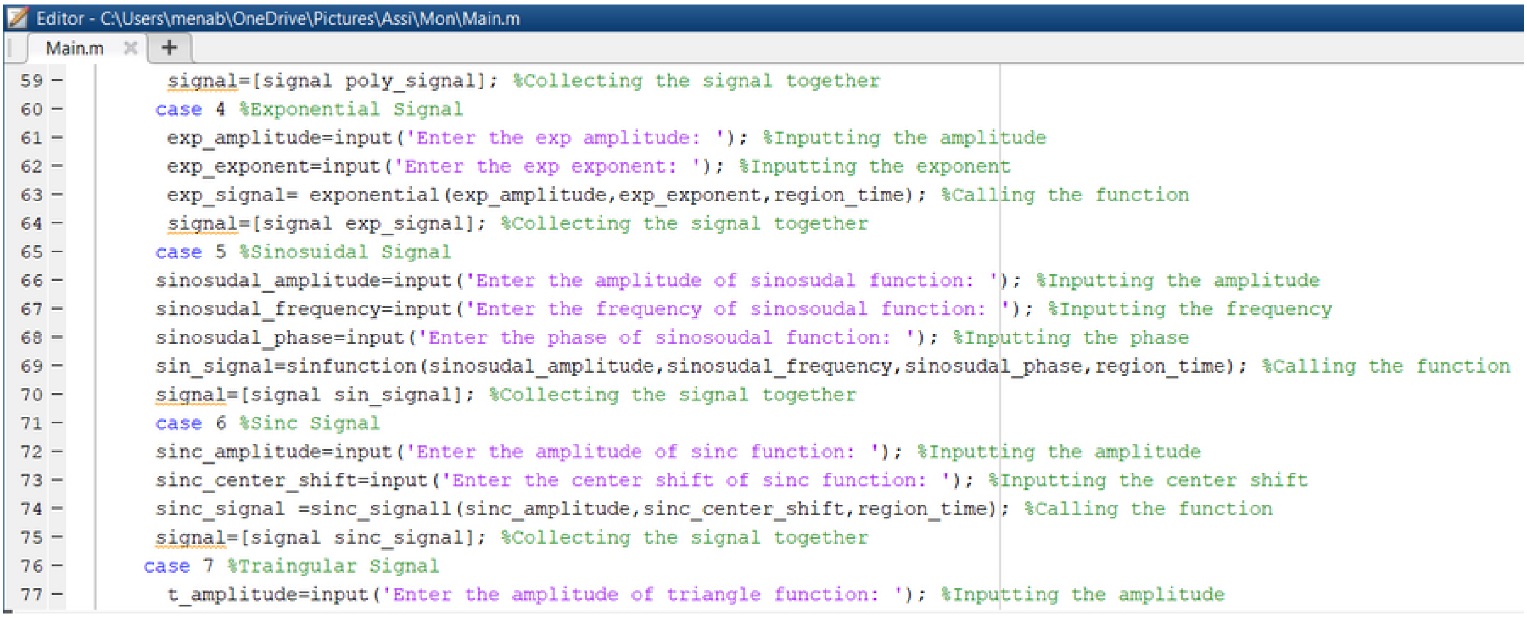
```matlab
disp("Hello and welcome to your personal signal generator I'm ready to help you."); %Welcome Message
frequency=input('Enter the value of the sampling frequency: '); %Inputting the frequency
starting=input('Enter the starting value: '); %Inputting the starting point
ending=input('Enter the ending value: '); %Inputting the ending point
while (starting>=ending) %Warning if the starting point is bigger than or equal to the ending point
    disp('Starting point can not be the same or bigger than the ending point!')
    starting=input('Enter the starting value: ');
    ending=input('Enter the ending value: ');
end
bp=input('Enter the number of breakpoints: '); %number of the breakpoints
signal_sampling_frequency=frequency*(ending-starting); %calculating the sampling frequency
signal_sampling_time=linspace(starting,ending,signal_sampling_frequency); %calculating the sampling time
time_storing_array=[starting]; %array to store the time range and breakpoints
j=1;  %indicator for the elements in the array
signal=[]; %array to store the signal
  if(bp==0)
        disp('There is no breakpoints')
 else
     while(bp ~= 0)  %loop to enter the breakpoints

```



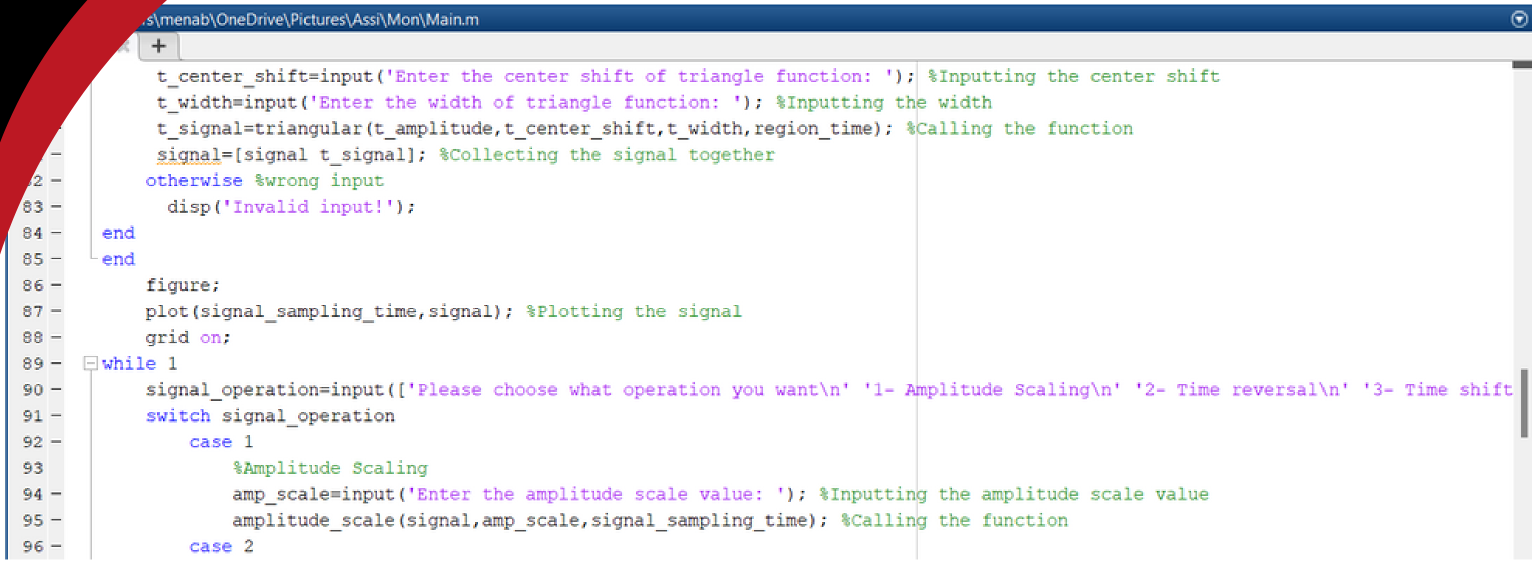
```matlab
        break_point_position=input(['Enter the position of the ' iptnum2ordinal(j) ' break point: ']); %entering the breakpoints
        search = searchElement(time_storing_array, break_point_position); %search for the break point in the breakpoints matrix to
        if(break_point_position<=starting ||  break_point_position>=ending)  %condition to prevent entering a break point less tha
            disp('Invalid Point!')
        elseif (search == 1) %condition to prevent repeated breakpoints
            disp('Repeated point!')
        else
          time_storing_array(j+1)=break_point_position; %storing the break point in the array
          j=j+1; %go to the next element
          bp=bp-1; %when goes to 0 the loop terminates

        end
     end
  end
time_storing_array(length(time_storing_array)+1)=ending; %storing the ending point in the array
time_storing_array=sort(time_storing_array); %sorting the array
for i=1:length(time_storing_array)-1 %loop to choose the signal for each region
    region_sampling_frequency=(time_storing_array(i+1)-time_storing_array(i))*frequency; %calculating region sampling frequency
    region_time=linspace(time_storing_array(i),time_storing_array(i+1),region_sampling_frequency); %calculating region time
```



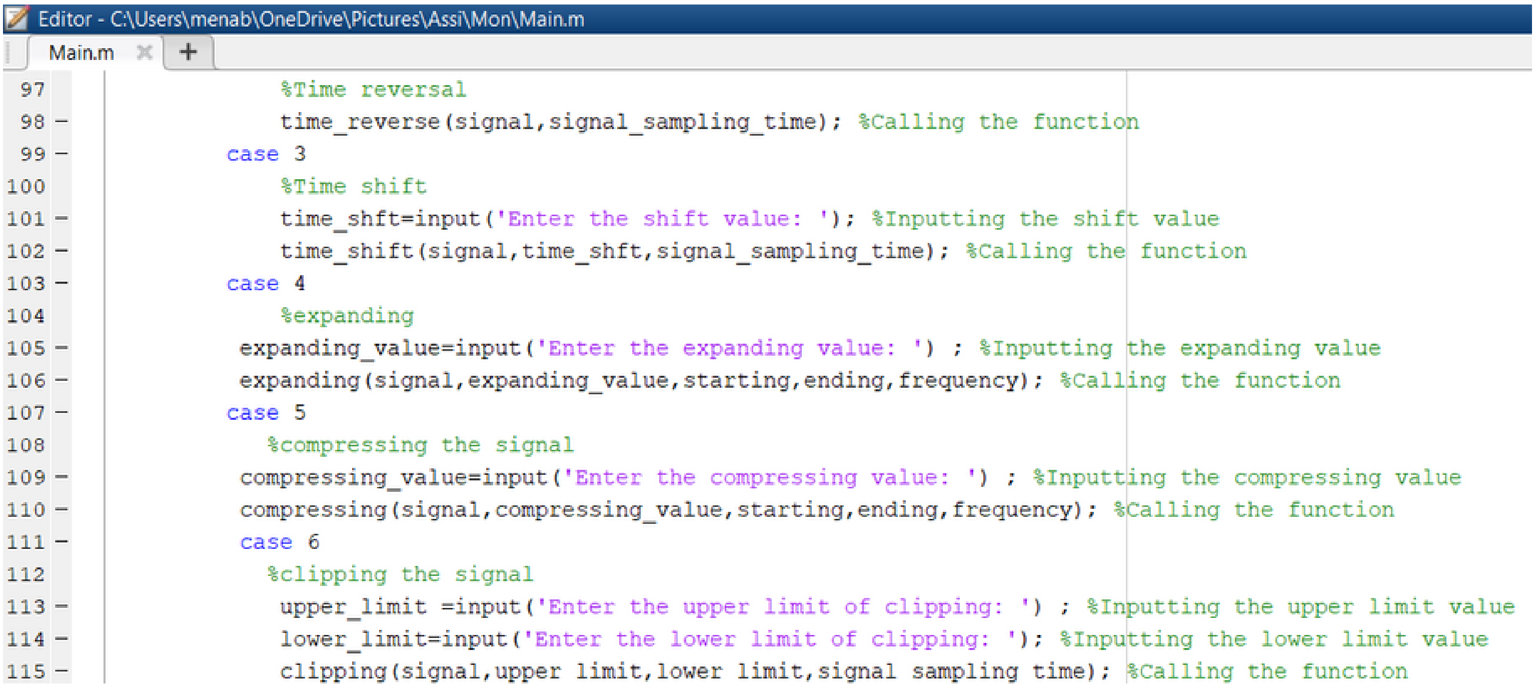
```matlab
        signal_choice=input(['Please choose what signal you want for the ' iptnum2ordinal(i) ' Region\n' '1- DC signal\n' '2- Ramp sig
        while (signal_choice<1 || signal_choice>7 ) %condition to prevent invalid signal input
            disp('Invalid input!!\n' 'please enter a number within 1 to 7.');
            signal_choice=input(['Please choose what signal you want for the ' iptnum2ordinal(i) ' Region\n' '1- DC signal\n' '2- Ramp
       end
    switch signal_choice
        case 1 %DC Signal
         dc_amplitude=input('Enter the DC amplitude: '); %Inputting the amplitude
         dc_signal=DC(dc_amplitude,region_sampling_frequency); %Calling the function
         signal=[signal dc_signal]; %Collecting the signal together
        case 2 %Ramp Signal
        Ramp_slope=input('Enter the Ramp slope: '); %Inputting the slope
        Ramp_intercept=input('Enter the Ramp intercept: '); %Inputting the intercept
        Ramp_signal=Ramp(Ramp_slope,Ramp_intercept,region_time); %Calling the function
        signal=[signal Ramp_signal]; %Collecting the signal together
        case 3 %General polynomial Signal
         polynomial_highest_power=input('Enter the polynomial highest power: '); %Inputting the highest order
         polynomial_intercept=input('Enter the polynomial intercept: '); %Inputting the intercept
         poly_signal= polynomial(polynomial_highest_power,polynomial_intercept,region_time); %Calling the function
```



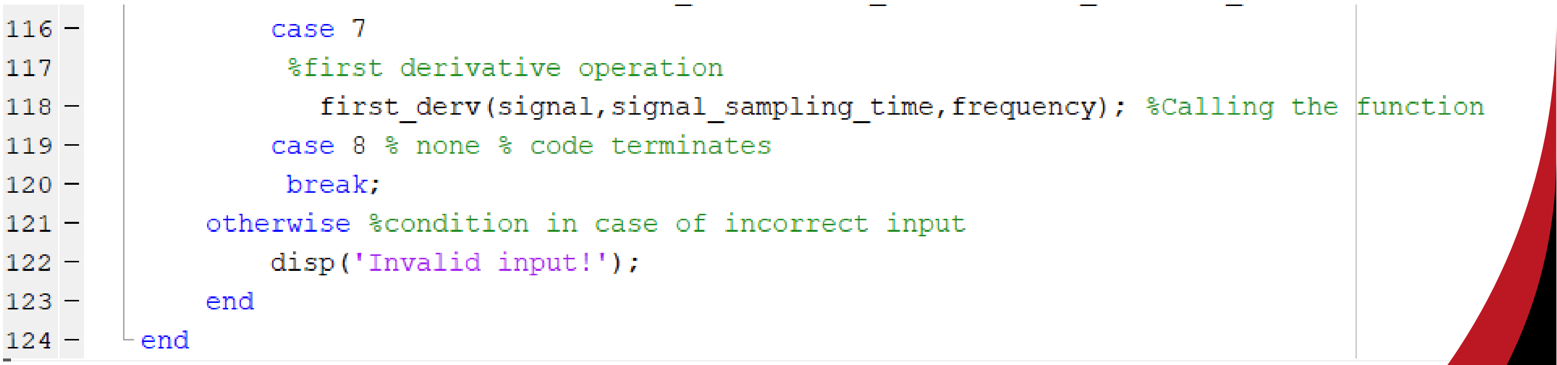
```matlab
         signal=[signal poly_signal]; %Collecting the signal together
        case 4 %Exponential Signal
         exp_amplitude=input('Enter the exp amplitude: '); %Inputting the amplitude
         exp_exponent=input('Enter the exp exponent: '); %Inputting the exponent
         exp_signal= exponential(exp_amplitude,exp_exponent,region_time); %Calling the function
         signal=[signal exp_signal]; %Collecting the signal together
        case 5 %Sinosuidal Signal
        sinosudal_amplitude=input('Enter the amplitude of sinosudal function: '); %Inputting the amplitude
        sinosudal_frequency=input('Enter the frequency of sinosoudal function: '); %Inputting the frequency
        sinosudal_phase=input('Enter the phase of sinosoudal function: '); %Inputting the phase
        sin_signal=sinfunction(sinosudal_amplitude,sinosudal_frequency,sinosudal_phase,region_time); %Calling the function
        signal=[signal sin_signal]; %Collecting the signal together
        case 6 %Sinc Signal
        sinc_amplitude=input('Enter the amplitude of sinc function: '); %Inputting the amplitude
        sinc_center_shift=input('Enter the center shift of sinc function: '); %Inputting the center shift
        sinc_signal =sinc_signall(sinc_amplitude,sinc_center_shift,region_time); %Calling the function
        signal=[signal sinc_signal]; %Collecting the signal together
       case 7 %Traingular Signal
         t_amplitude=input('Enter the amplitude of triangle function: '); %Inputting the amplitude
```



```
     t_center_shift=input('Enter the center shift of triangle function: '); %Inputting the center shift
     t_width=input('Enter the width of triangle function: '); %Inputting the width
     t_signal=triangular(t_amplitude,t_center_shift,t_width,region_time); %Calling the function
     signal=[signal t_signal]; %Collecting the signal together
    otherwise %wrong input
      disp('Invalid input!');
end
end
    figure;
    plot(signal_sampling_time,signal); %Plotting the signal
    grid on;
while 1
    signal_operation=input(['Please choose what operation you want\n' '1- Amplitude Scaling\n' '2- Time reversal\n' '3- Time shift
    switch signal_operation
        case 1
            %Amplitude Scaling
            amp_scale=input('Enter the amplitude scale value: '); %Inputting the amplitude scale value
            amplitude_scale(signal,amp_scale,signal_sampling_time); %Calling the function
        case 2
```



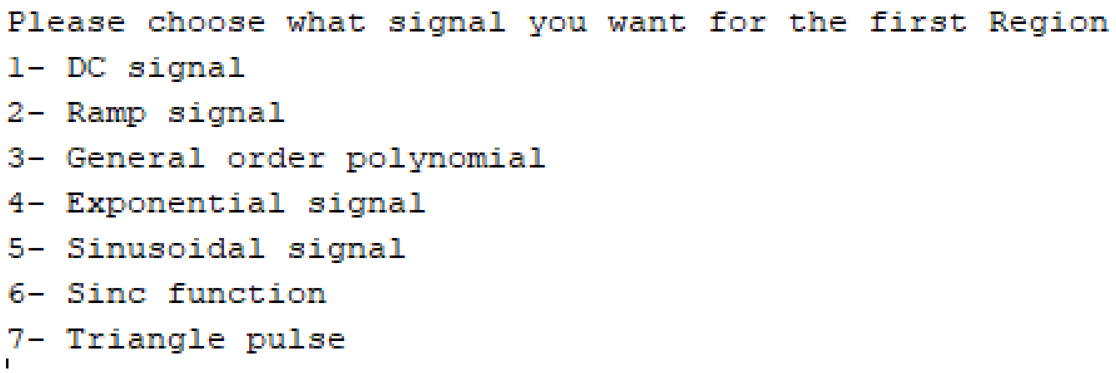
```
            %Time reversal
            time_reverse(signal,signal_sampling_time); %Calling the function
        case 3
            %Time shift
            time_shft=input('Enter the shift value: '); %Inputting the shift value
            time_shift(signal,time_shft,signal_sampling_time); %Calling the function
        case 4
            %expanding
         expanding_value=input('Enter the expanding value: ') ; %Inputting the expanding value
         expanding(signal,expanding_value,starting,ending,frequency); %Calling the function
        case 5
           %compressing the signal
         compressing_value=input('Enter the compressing value: ') ; %Inputting the compressing value
         compressing(signal,compressing_value,starting,ending,frequency); %Calling the function
         case 6
           %clipping the signal
            upper_limit =input('Enter the upper limit of clipping: ') ; %Inputting the upper limit value
            lower_limit=input('Enter the lower limit of clipping: '); %Inputting the lower limit value
            clipping(signal,upper limit,lower limit,signal sampling time); %Calling the function
```



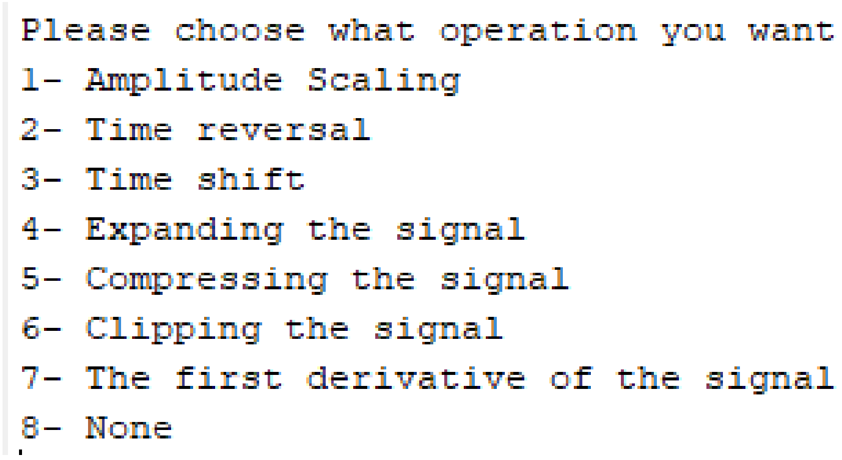
```
        case 7
         %first derivative operation
           first_derv(signal,signal_sampling_time,frequency); %Calling the function
        case 8 % none % code terminates
         break;
    otherwise %condition in case of incorrect input
        disp('Invalid input!');
    end
end
```

signals:

In this project, the user will enter starting and ending points then number of breakpoints then the user will define the breakpoints positions. then he will have to choose the signals he want in each region and then he will choose specific operations for the output signal.

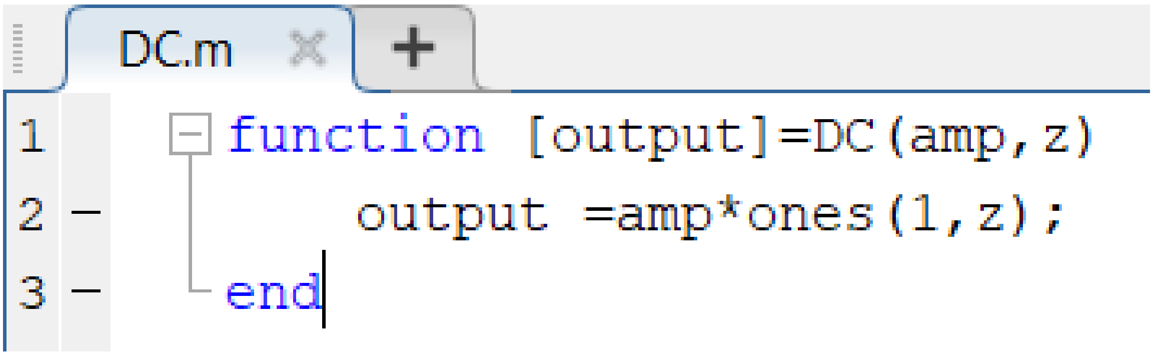


```
Please choose what signal you want for the first Region
1- DC signal
2- Ramp signal
3- General order polynomial
4- Exponential signal
5- Sinusoidal signal
6- Sinc function
7- Triangle pulse
```

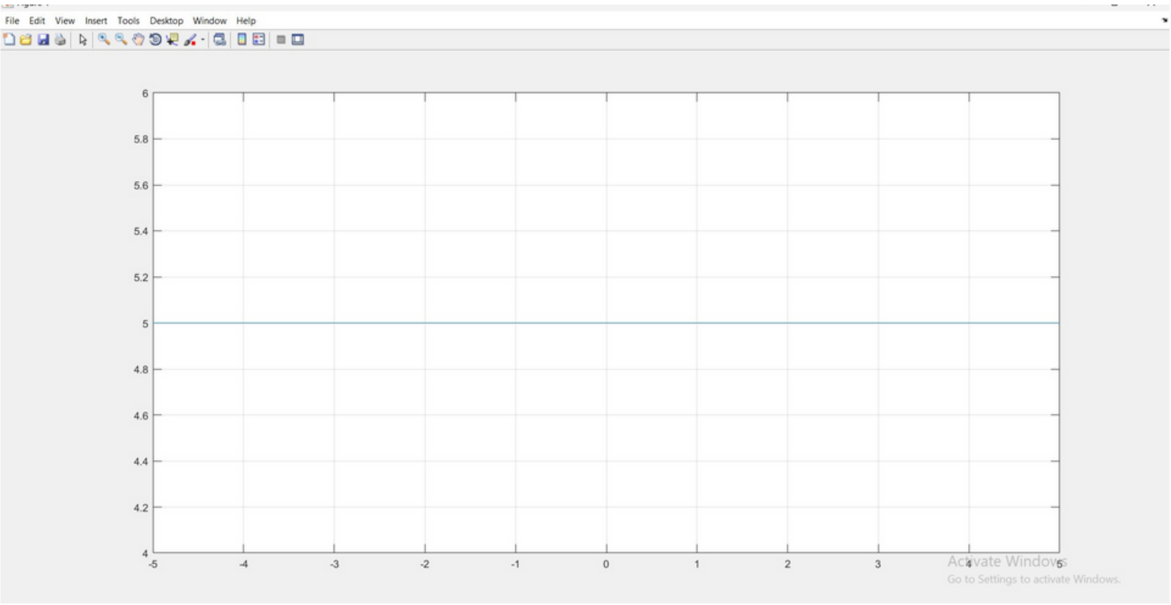


```
Please choose what operation you want
1- Amplitude Scaling
2- Time reversal
3- Time shift
4- Expanding the signal
5- Compressing the signal
6- Clipping the signal
7- The first derivative of the signal
8- None
```

1- DC Signal

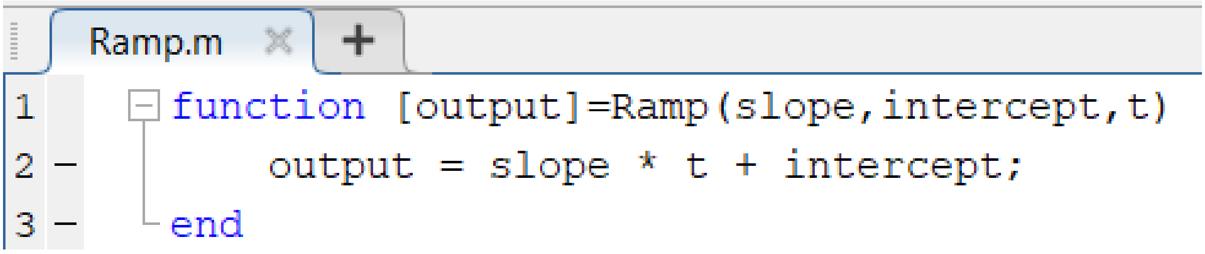


```
function [output]=DC(amp,z)
    output =amp*ones(1,z);
end
```

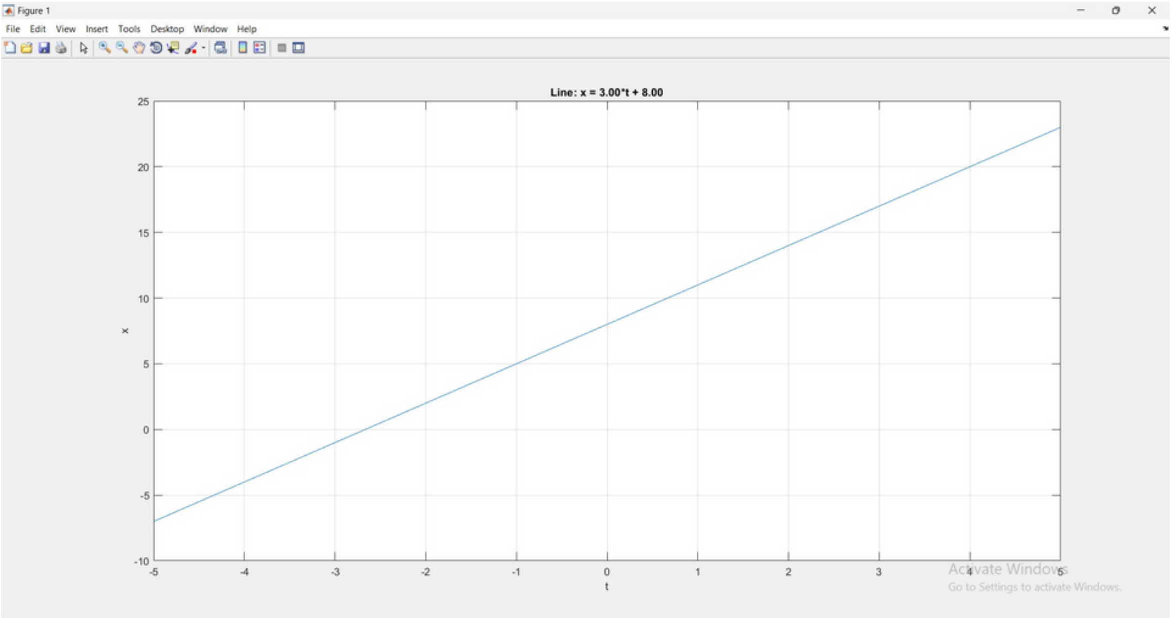


DC with amplitude=5

2- Ramp Signal

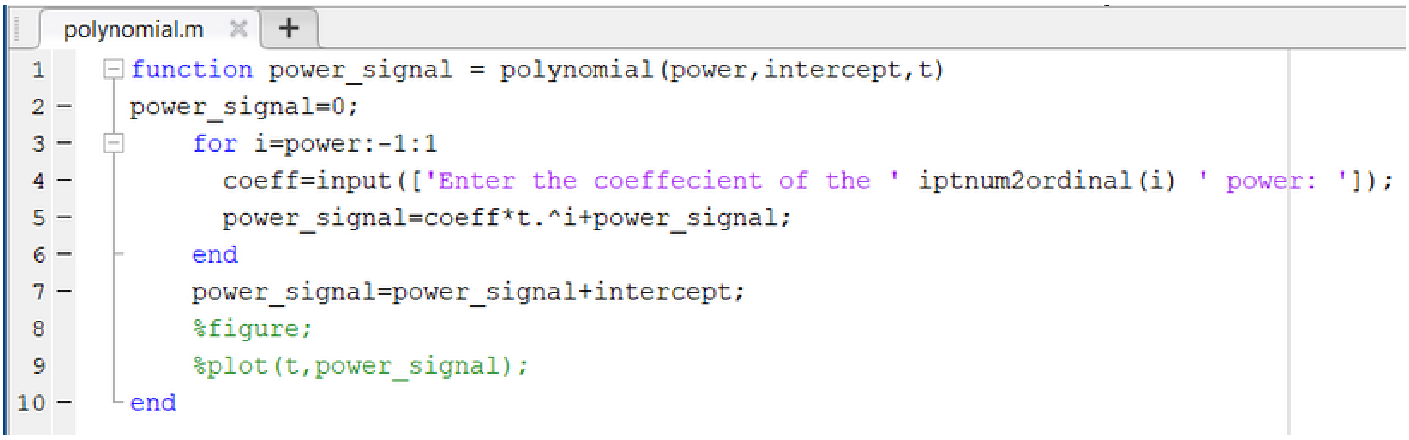


```
function [output]=Ramp(slope,intercept,t)
    output = slope * t + intercept;
end
```

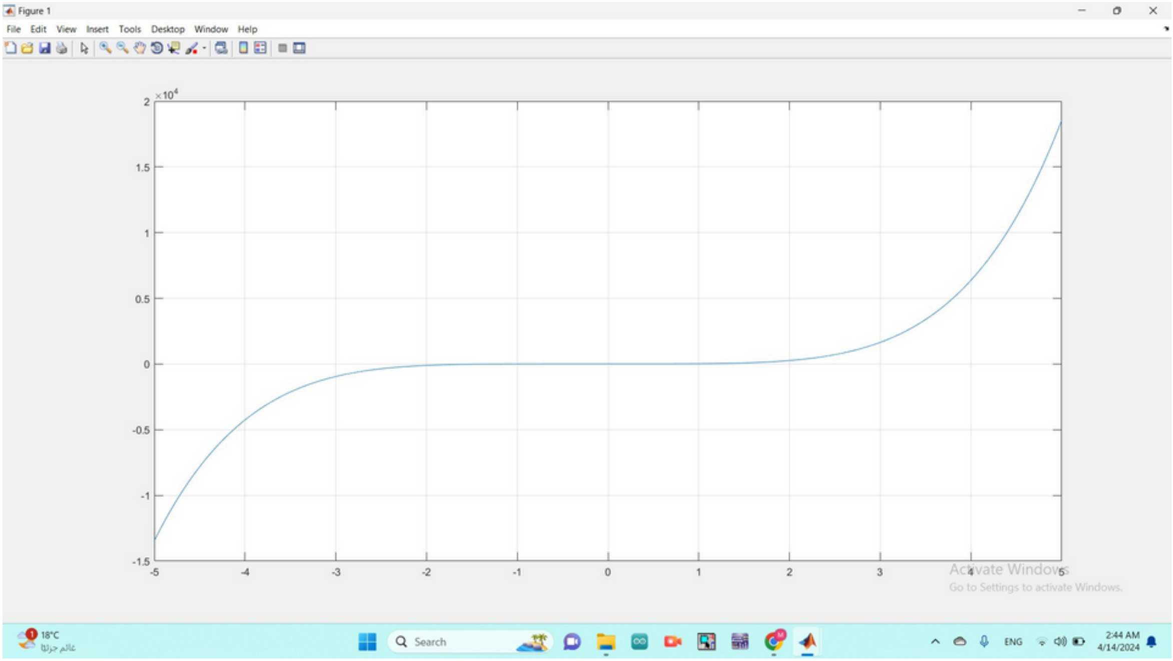


Ramp with equation: x=3t+8

3- General Order Polynomial Signal

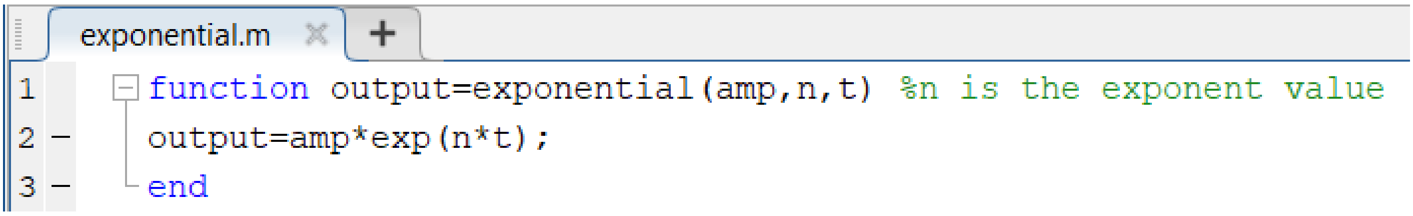


```
function power_signal = polynomial(power,intercept,t)
power_signal=0;
    for i=power:-1:1
      coeff=input(['Enter the coeffecient of the ' iptnum2ordinal(i) ' power: ']);
      power_signal=coeff*t.^i+power_signal;
    end
    power_signal=power_signal+intercept;
    %figure;
    %plot(t,power_signal);
end
```



General order polynomial of line : $5x^5 + 4x^4 + 3x^3 + 2x^2 + x + 6$

4- Exponential Signal



```
function output=exponential(amp,n,t) %n is the exponent value
output=amp*exp(n*t);
end
```

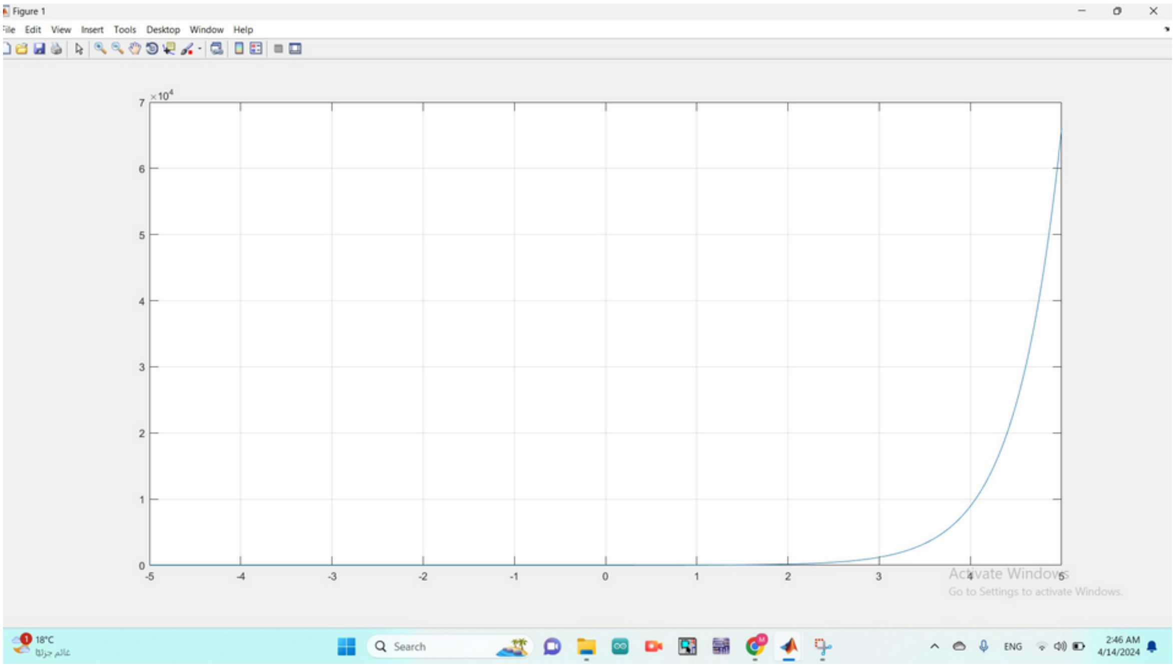
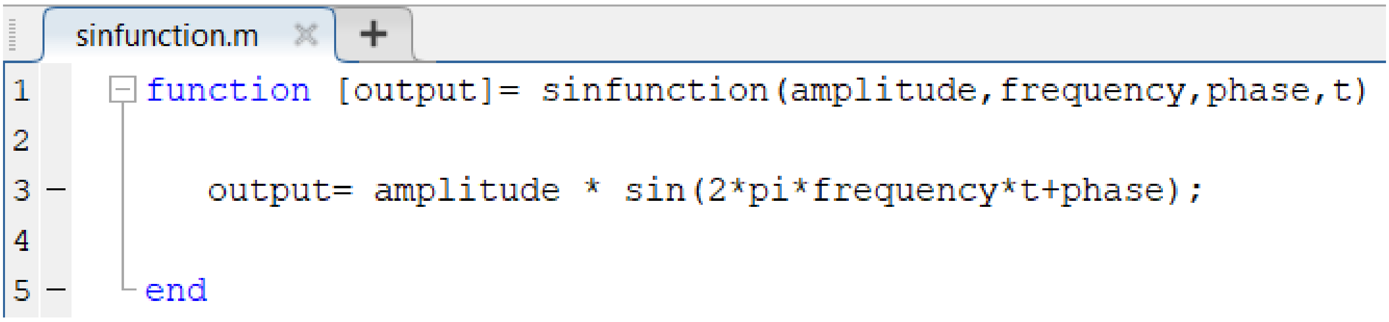


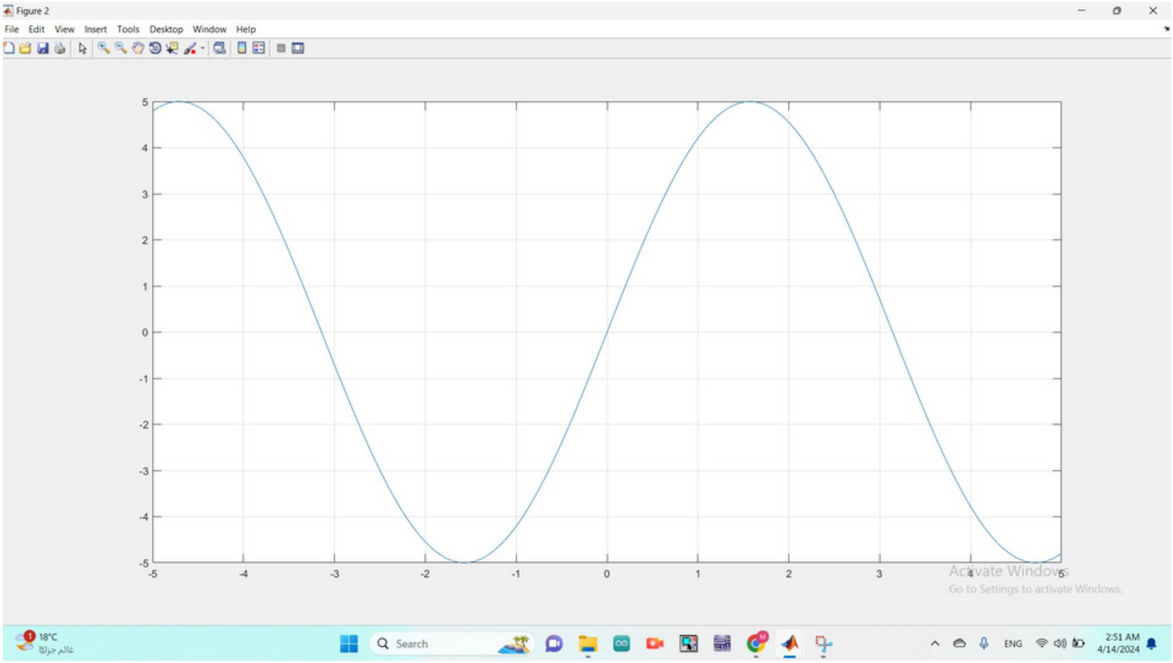
Figure 1
File Edit View Insert Tools Desktop Window Help
×10^4
7
6
5
4
3
2
1
0
-5
-4
-3
-2
-1
0
1
2
3
4
5
Activate Windows
Go to Settings to activate Windows.
Search
ENG
2:46 AM
4/14/2024

Exponential with line: X=$3e^{2t}$

5- Sinusoidal Signal



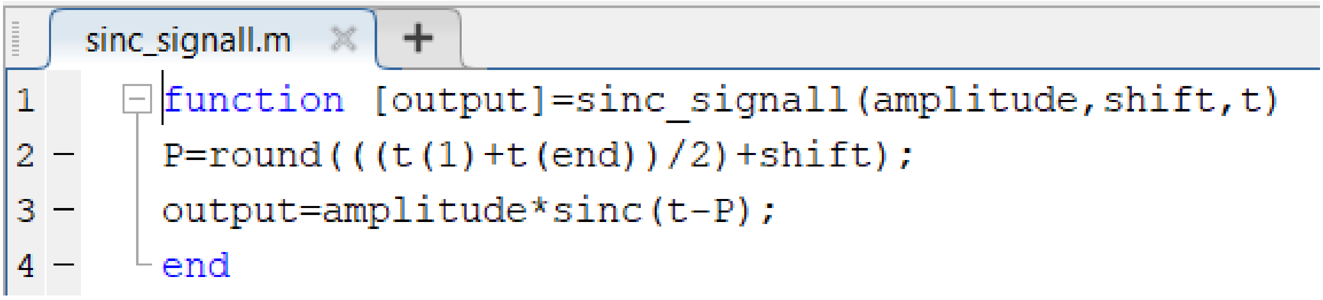
```
function [output]= sinfunction(amplitude,frequency,phase,t)

    output= amplitude * sin(2*pi*frequency*t+phase);

end
```



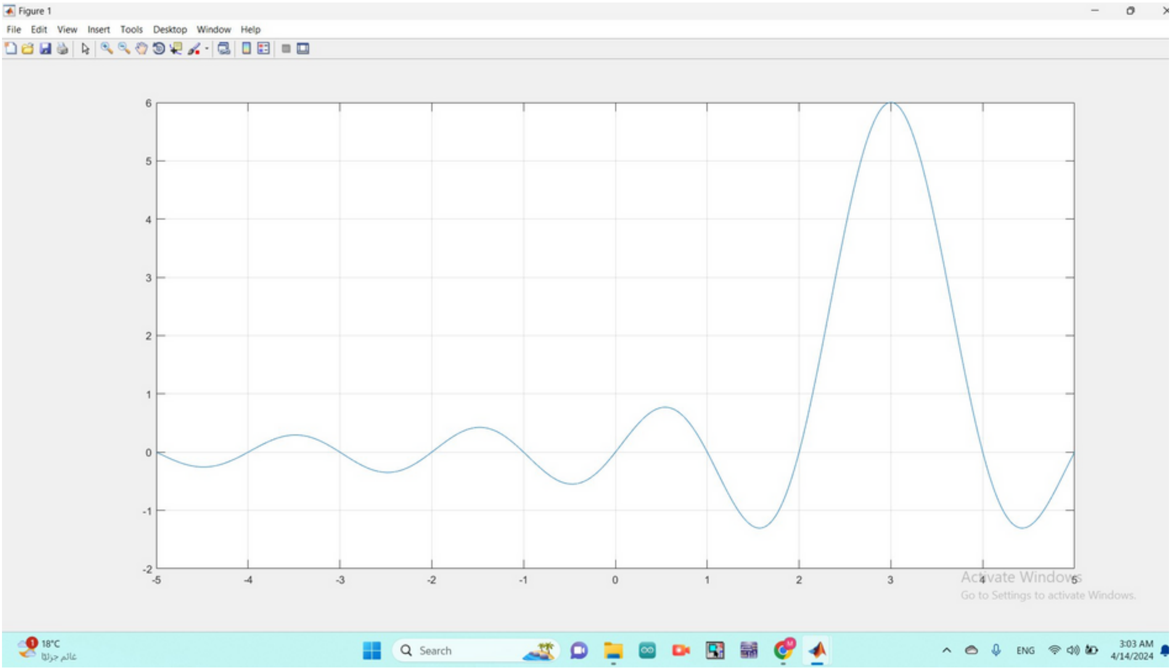
Sinosudal with amplitude 5 ,frequency =1/(2*pi) and phase =2*pi :

$$x=5\,sin\left(\frac{t}{2\pi}+2\pi\right)$$

6- Sinc Signal

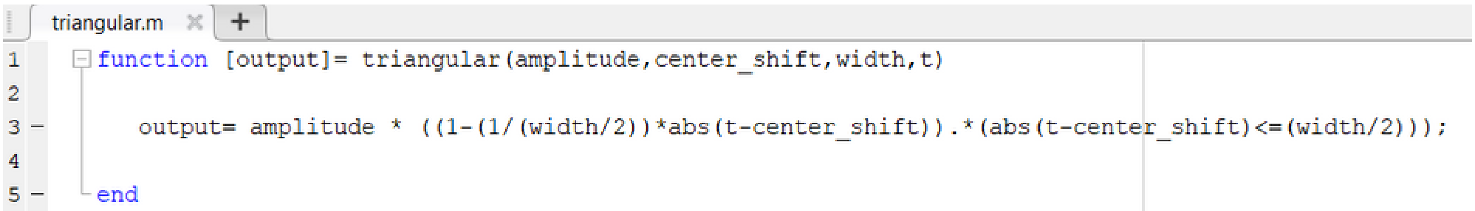


```
function [output]=sinc_signall(amplitude,shift,t)
P=round(((t(1)+t(end))/2)+shift);
output=amplitude*sinc(t-P);
end
```

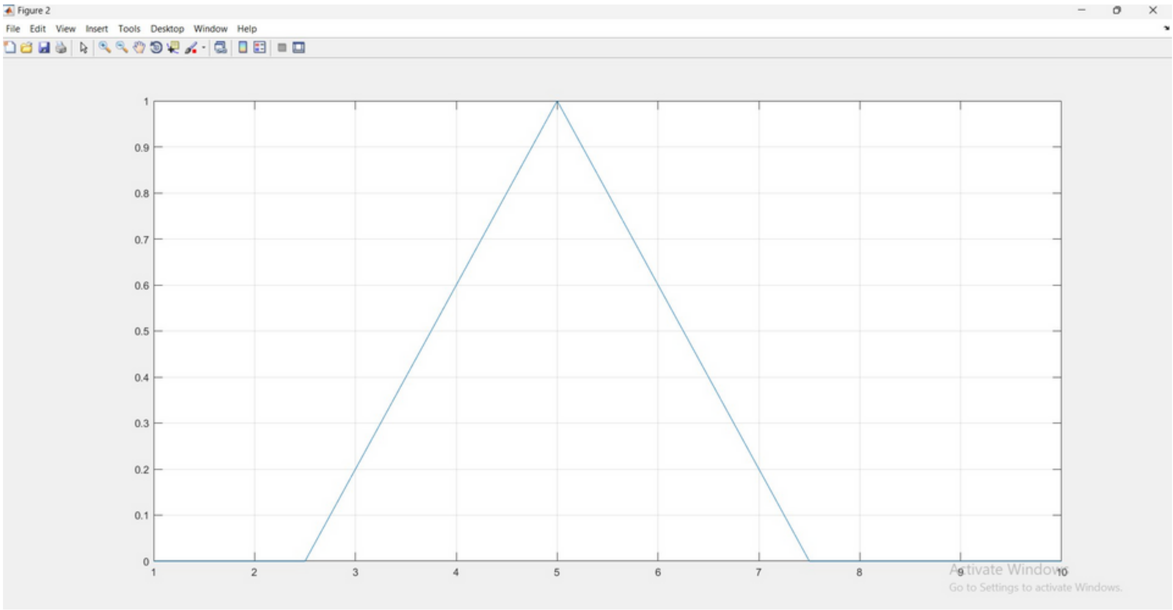


Sinc function with amplitude 6 and center shift 3 : $x = 6\frac{sin(t-3)}{t-3}$

7- Traingular Signal



```
function [output]= triangular(amplitude,center_shift,width,t)

    output= amplitude * ((1-(1/(width/2))*abs(t-center_shift)).*(abs(t-center_shift)<=(width/2)));

end
```



Triangular signal with amplitude 1 , center shift 5 and width 5

Example on signals:

We have a signal of starting point=1 , ending point=10 and two breakpoints =4,6 containing 3 regions : sinusoidal , dc and sinc siganls .

Region 1 : (sinusoidal)

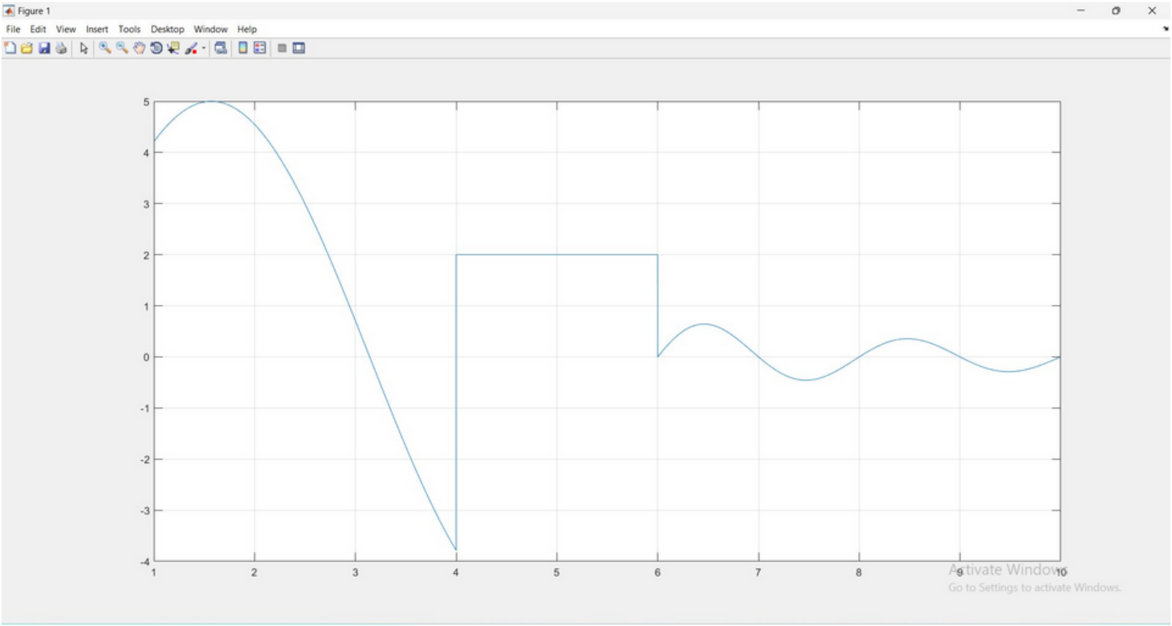
Amplitude 5 ,frequency =1/(2*pi) and phase =2*pi

Region 2: (DC)

Amplitude=2

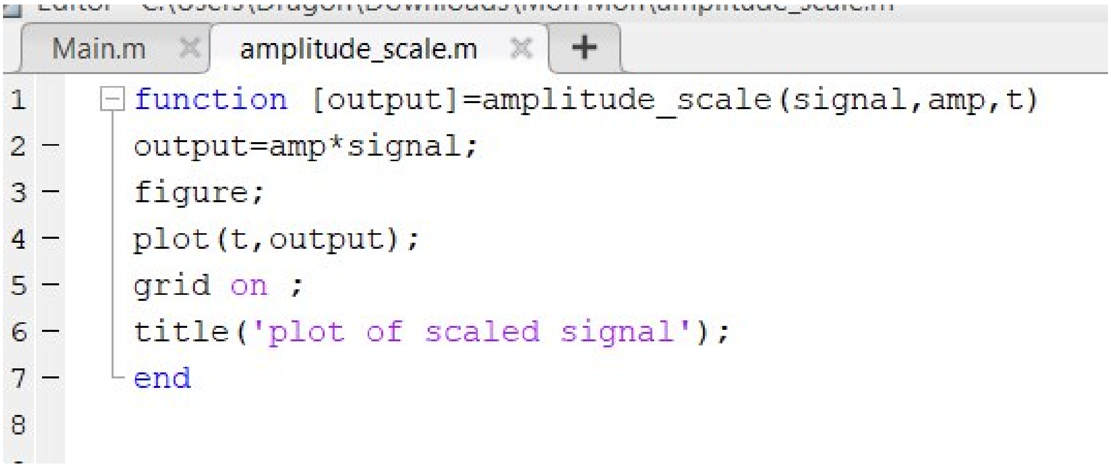
Region 3: (sinc)

Amplitude 5 and center shift 4

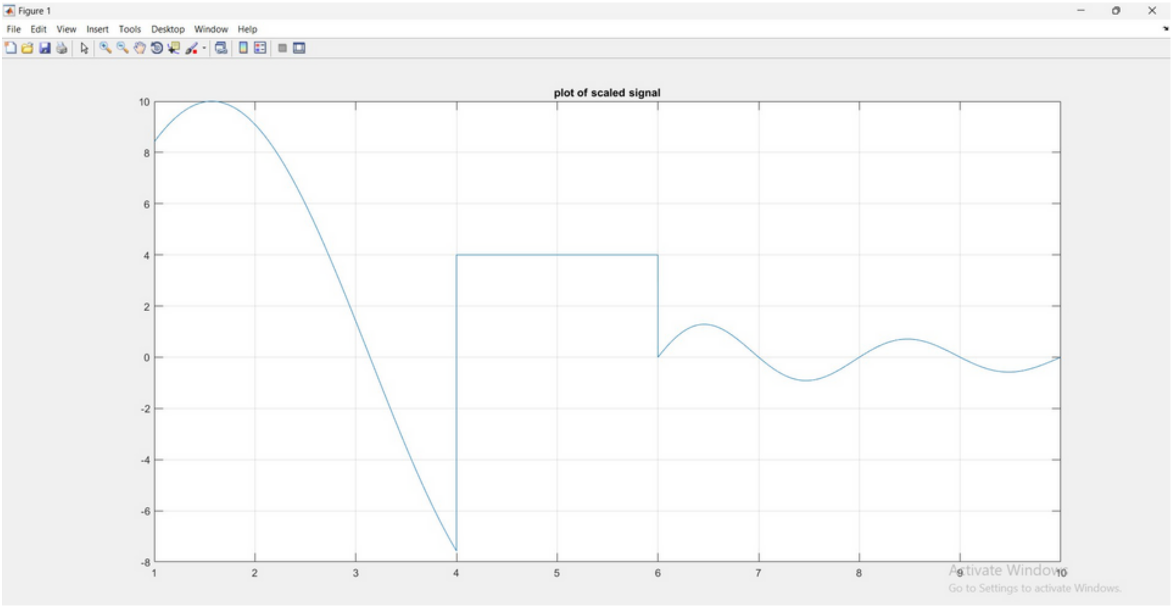


Operations on signals:

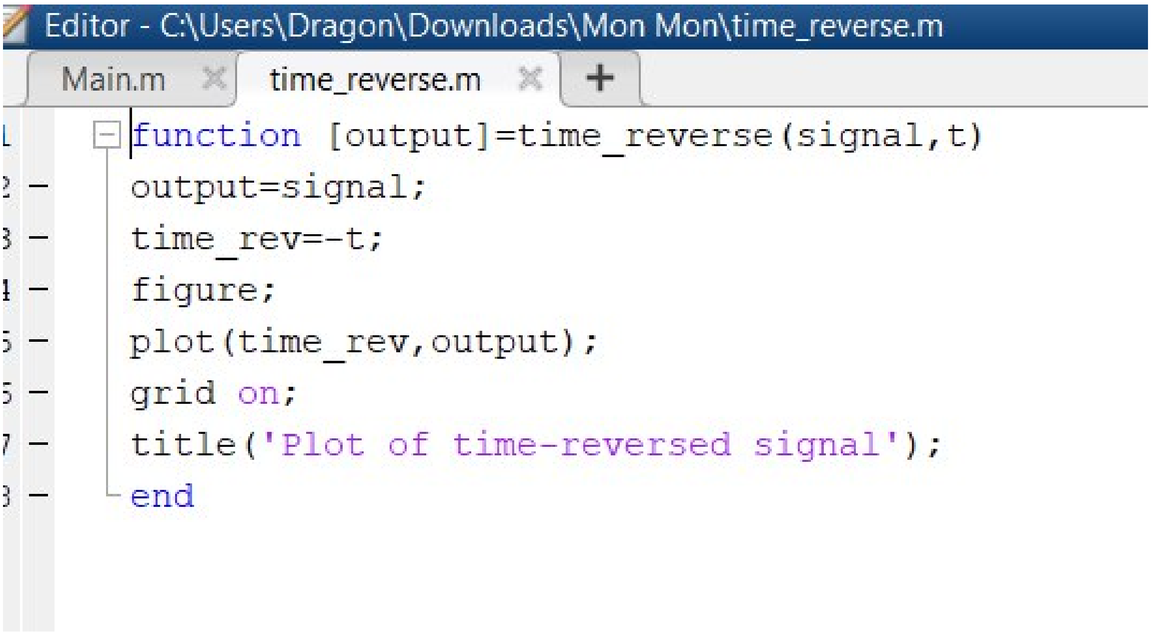
Now , we will try different operation on the previous example.



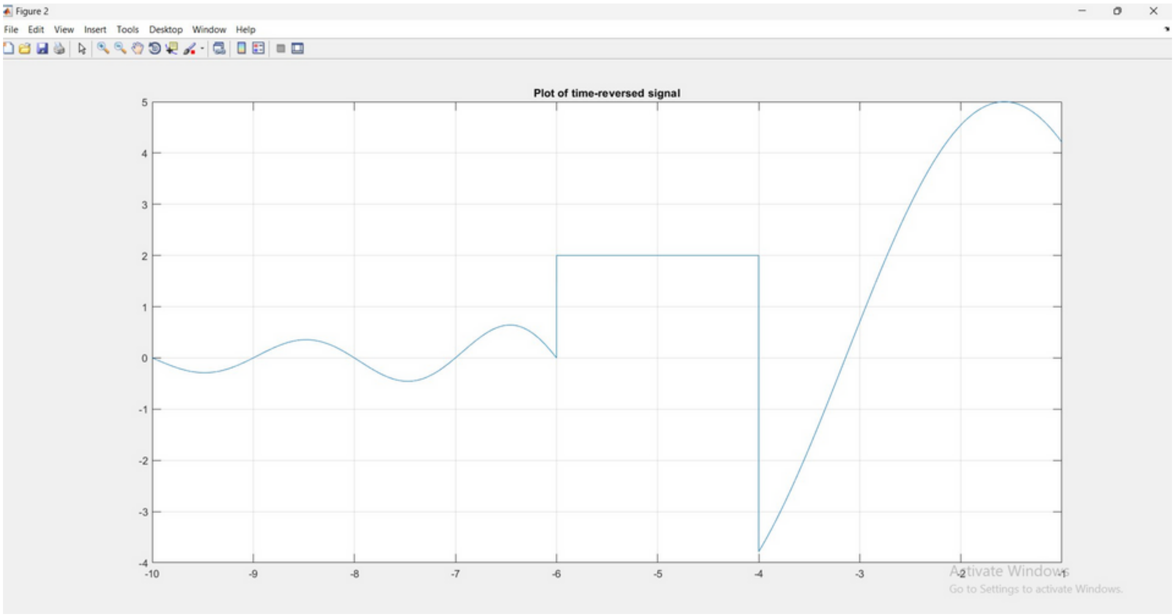
```
function [output]=amplitude_scale(signal,amp,t)
output=amp*signal;
figure;
plot(t,output);
grid on ;
title('plot of scaled signal');
end
```



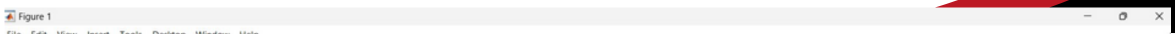
Amplitude scale by 2

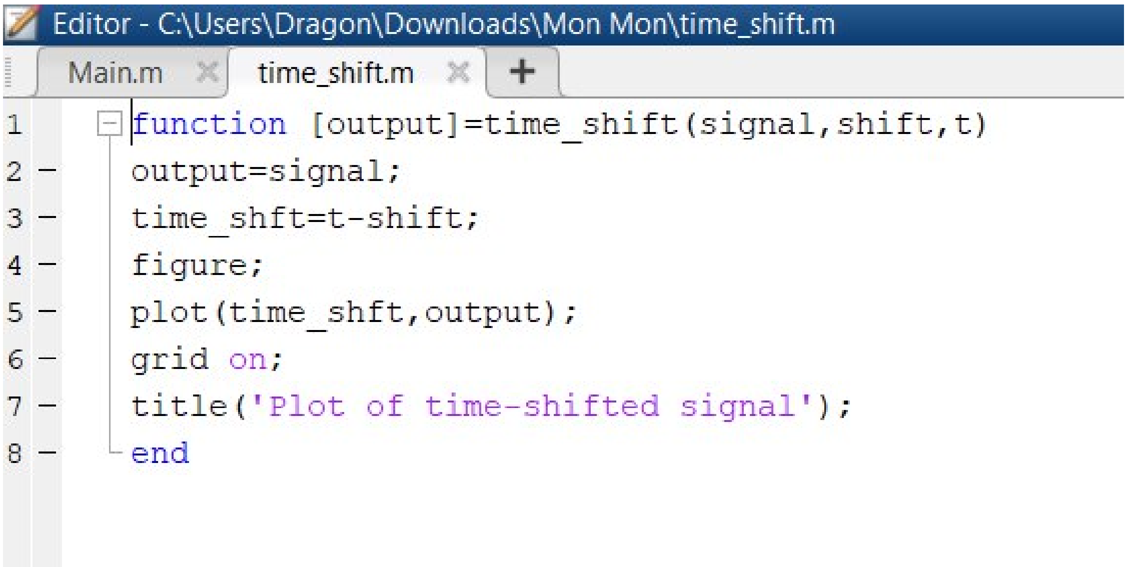


```
function [output]=time_reverse(signal,t)
output=signal;
time_rev=-t;
figure;
plot(time_rev,output);
grid on;
title('Plot of time-reversed signal');
end
```

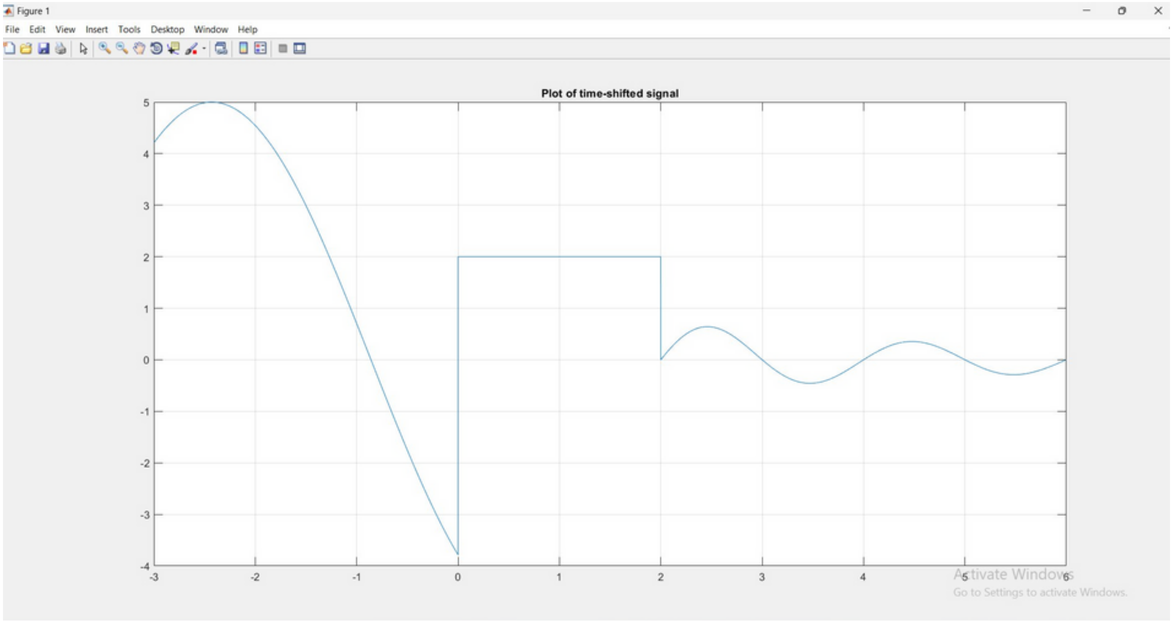


Reversed

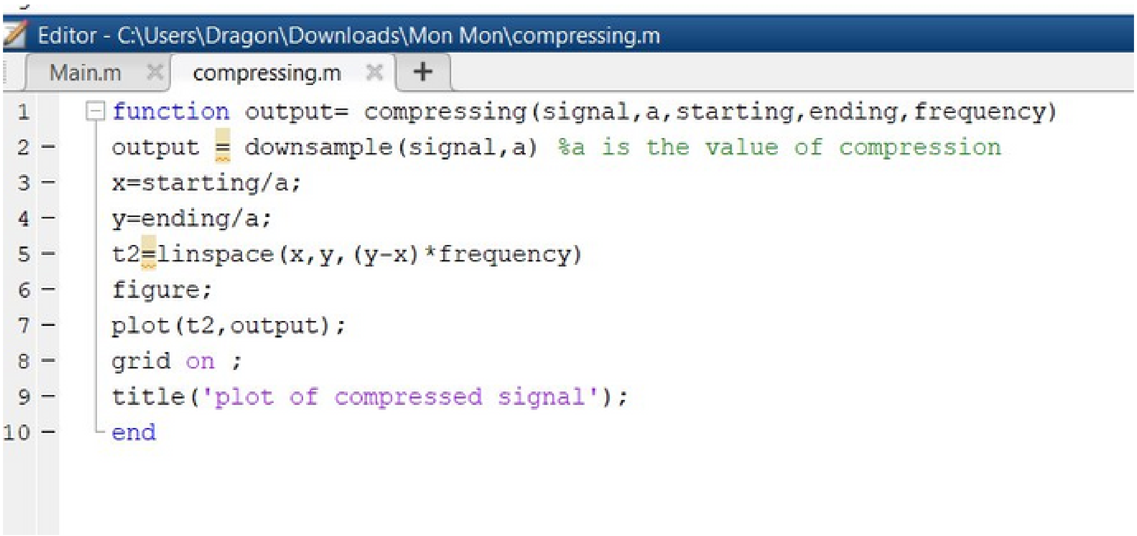




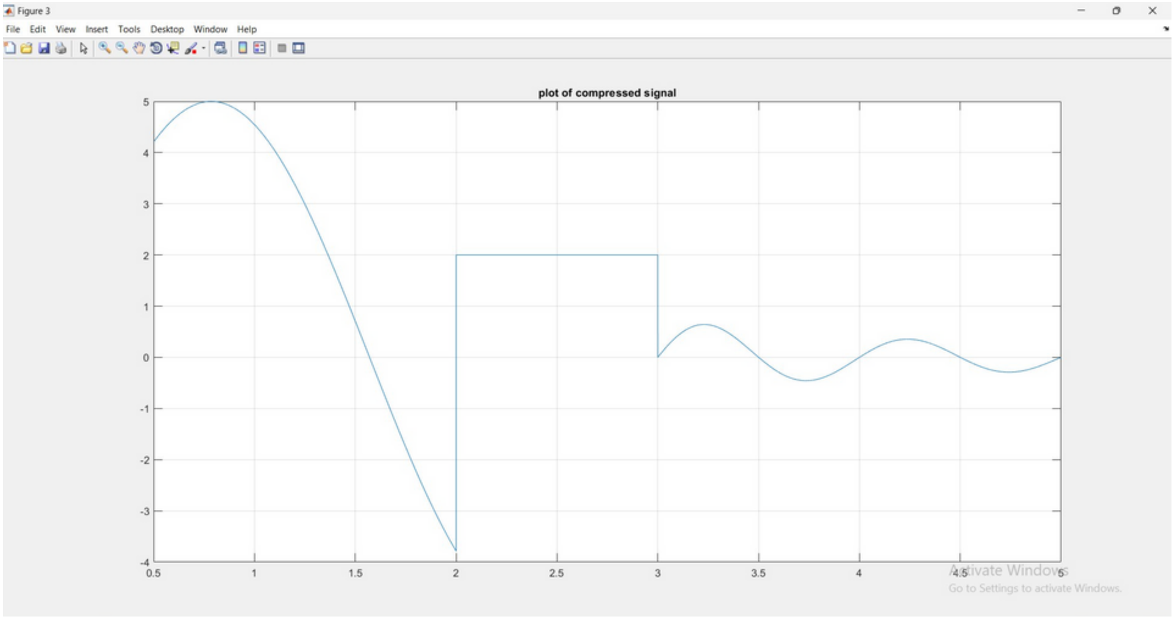
```
function [output]=time_shift(signal,shift,t)
output=signal;
time_shft=t-shift;
figure;
plot(time_shft,output);
grid on;
title('Plot of time-shifted signal');
end
```



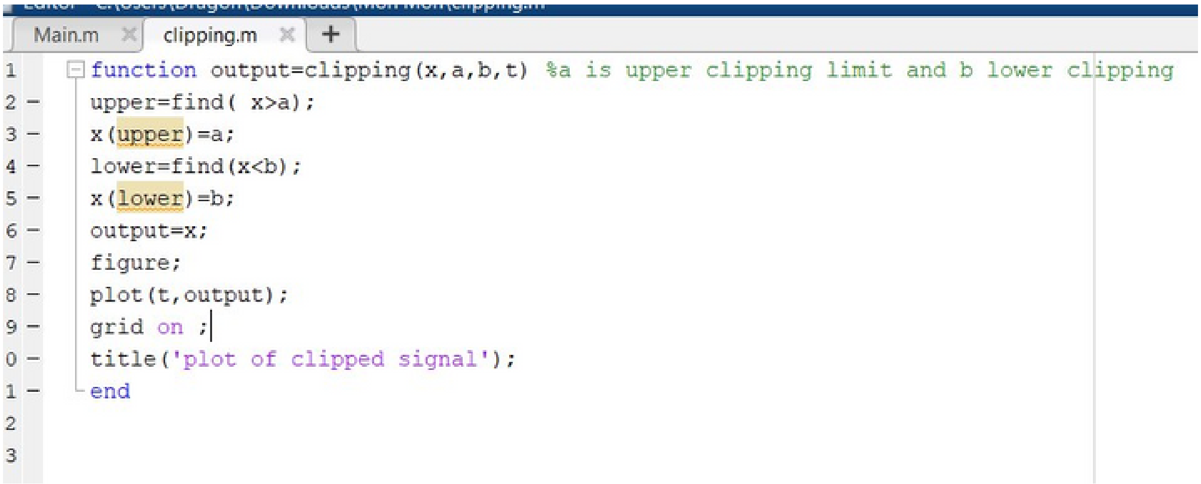
shift by 4



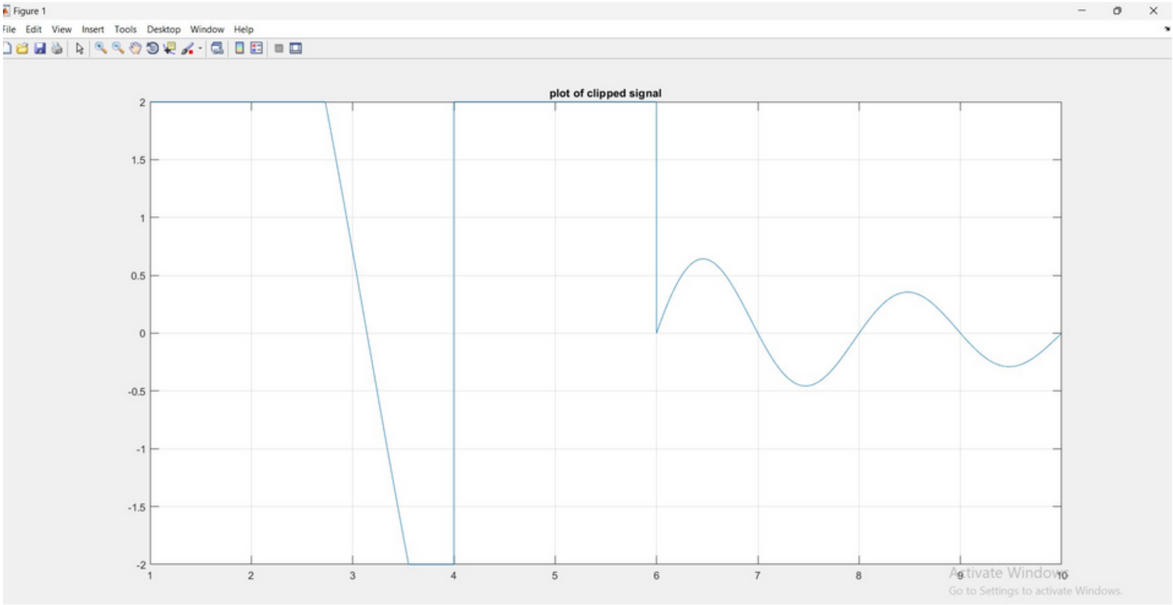
```
function output= compressing(signal,a,starting,ending,frequency)
output = downsample(signal,a) %a is the value of compression
x=starting/a;
y=ending/a;
t2=linspace(x,y,(y-x)*frequency)
figure;
plot(t2,output);
grid on ;
title('plot of compressed signal');
end
```



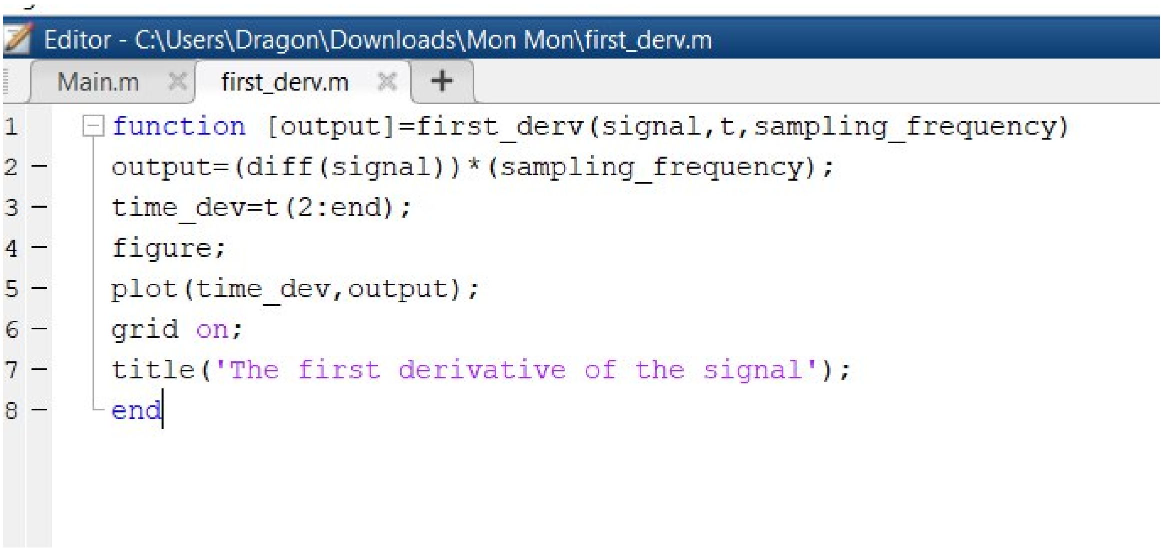
Compression by 2



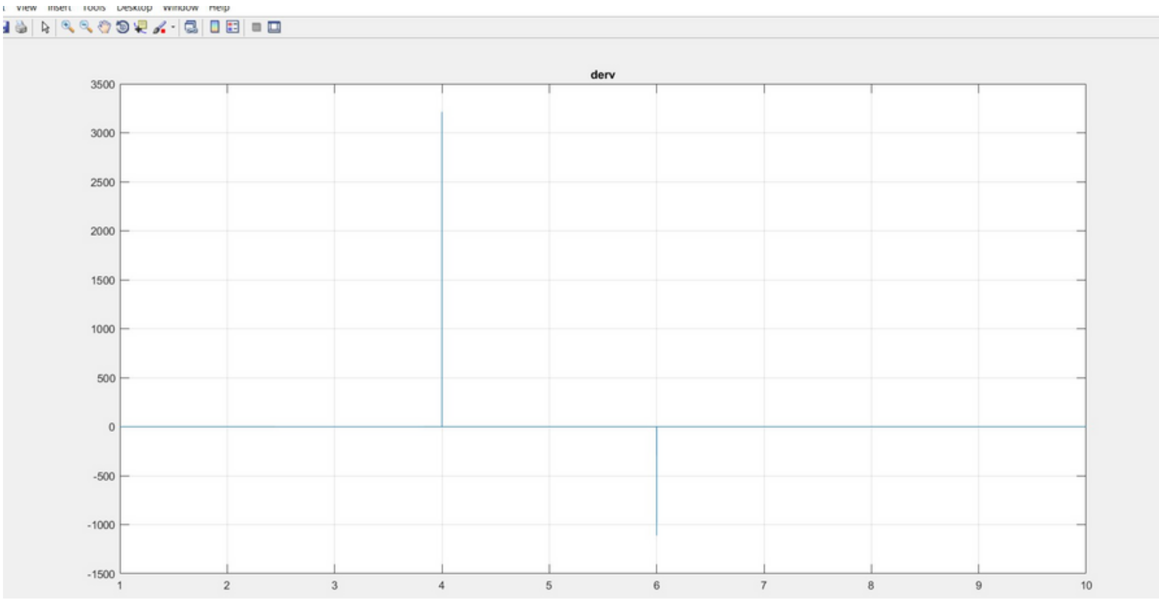
```
function output=clipping(x,a,b,t) %a is upper clipping limit and b lower clipping
upper=find( x>a);
x(upper)=a;
lower=find(x<b);
x(lower)=b;
output=x;
figure;
plot(t,output);
grid on ;
title('plot of clipped signal');
end
```



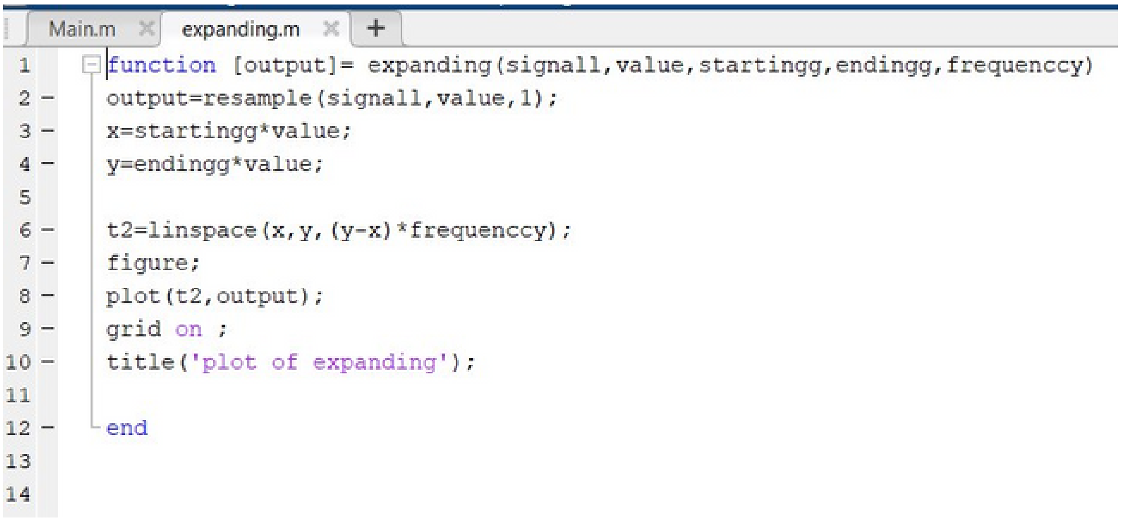
Clipping with upper limit 2 and lower limit -2



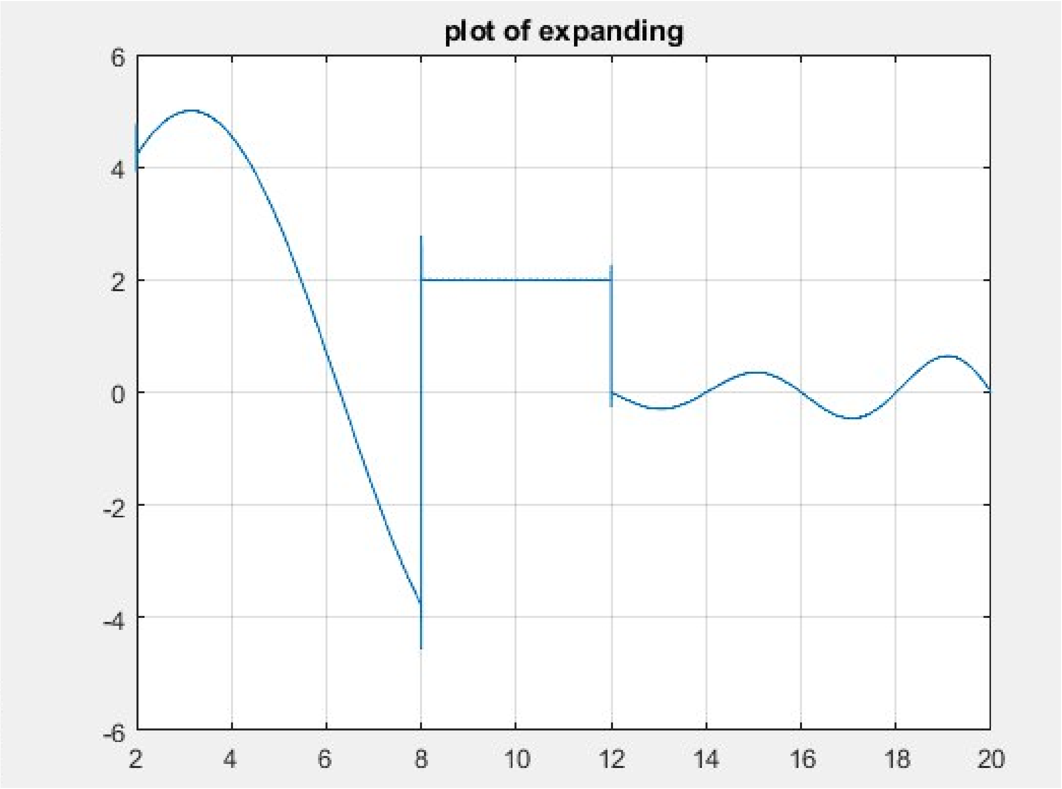
```
function [output]=first_derv(signal,t,sampling_frequency)
output=(diff(signal))*(sampling_frequency);
time_dev=t(2:end);
figure;
plot(time_dev,output);
grid on;
title('The first derivative of the signal');
end
```



First derivative



```
function [output]= expanding(signall,value,startingg,endingg,frequenccy)
output=resample(signall,value,1);
x=startingg*value;
y=endingg*value;

t2=linspace(x,y,(y-x)*frequenccy);
figure;
plot(t2,output);
grid on ;
title('plot of expanding');

end
```



Expanssion by 2

when you press 8 (None) , operations on signal will stop.