General signal generator

Mini project

Abdalla Magdy Abdelazem Elgohary	20010900
Sohaib Alaraby Ali AbdElhafez	20010751
Abdalla Ashraf Mohamed Melad	20012318
Mohamed Nagy Mabrouk	20012409

Intro:

A signal generator is one of a class of electronic devices that generates electrical signals with set properties of amplitude, frequency, and wave shape. These generated signals are used as a stimulus for electronic measurements, typically used in designing, testing, troubleshooting, and repairing electronic or electroacoustic devices, though it often has artistic uses as well.

Here is a MATLAB code to implement it

Code explanation:

- Ask the user for sampling frequency, start, and end of time scale. Then check that data is valid.

```
%enter the sample freqency
samplingFrequency=input("enter the Sampling frequency\n");
typeOf_SF=class(samplingFrequency);
while(~isnumeric(samplingFrequency)||samplingFrequency<=0)
    samplingFrequency=input("sampling Frequency is positive number try again\n");
    typeOf_SF=class(samplingFrequency);

end
%enter the starting point
starting_time=input('enter the Starting time\n');
typeOf_Starting_time=class(starting_time);
while(~isnumeric(starting_time))
    starting_time = input("starting time is a number try again\n");
    typeOf_Starting_time = class(starting_time);

end
%enter the ending point
ending_time=input('enter the ending time\n');
typeOf_ending_time=class(ending_time);
while(~isnumeric(ending_time))
    ending_time = input("ending time is a number try again\n");
    typeOf_ending_time = class(ending_time);

end
```

- Ask the user the number of break points and their position

```
25
26      %enter the number of breaking points
27 -    numberOfBreakPoints=input('enter the number Of Break Points\n');
28 -    typeOf_breaking_point=class(numberOfBreakPoints);
29 -    while(~isnumeric(numberOfBreakPoints)||numberOfBreakPoints<0)
30 -        numberOfBreakPoints = input("number Of Break Points is a positive number try again\n");
31 -        typeOf_breaking_point = class(numberOfBreakPoints);
32
33 -    end
34      %enter the position of breaking points in seconds
35 -    position=zeros(1,numberOfBreakPoints);
36 -    for i=1:1:numberOfBreakPoints
37 -        l= input("enter the position number "+i+" ");
38 -        while(~isnumeric(l))
39 -                l= input(" position is a number enter position"+i+" ");
40 -        end
41 -        position(i)=l;
42 -    end
43 -    position = [starting_time position ending_time];
```

- Generate the signal: Ask the user the type of signal at each region according to the number of break point.

```
45      %enter the type of signals thromugh each duration
46 -    fprintf("a. DC signal\nb. Ramp signal\nc. General order polynomial\nd. Exponential signal\ne. Sinusoida
47 -    signals=[];
48 -    sizeOfsignals = size(signals);
49 -    figure;
50
51 -    for i=1:1:numberOfBreakPoints+1
52
53 -        c =input("enter type signal from "+position(i)+" second to "+position(i+1)+" ");
54
55 -        while ~(ischar(c))
56 -             c =input("please enter a character which listed\n");
57 -        end
58
59 -        switch c
60 -            case 'a'
61 -                amplitude_DC = input("enter the amplitude of DC signal");
62 -                t=linspace(position(i),position(i+1),(position(i+1)-position(i))* samplingFrequency);
63 -                signal= zeros(1,(position(i+1)-position(i))* samplingFrequency)+ amplitude_DC;
64 -                plot(t,signal,'b-','LineWidth',1)
65 -                hold on;
66 -            case 'b'
67 -                slope     = input("enter slope of the ramp signal ");
68 -                intercept_ramp = input("enter the intercept of the ramp signal ");
69 -                t=linspace(position(i),position(i+1),(position(i+1)-position(i))* samplingFrequency);
70
71 -                signal = slope*t + intercept_ramp;
72
```

```
            plot(t,signal,'b-','LineWidth',1)
            hold on;
        case 'c'
            amplitude_poly = input("enter the amplitude of the General order polynomial ");
            Power = input("enter the Power of the General order polynomial ");
            intercept_poly = input("enter the intercept of the General order polynomia ");
            t=linspace(position(i),position(i+1),(position(i+1)-position(i))* samplingFrequency);

            signal = amplitude_poly * (t.^Power) + intercept_ramp;

            plot(t,signal,'r-','LineWidth',1)
            hold on;
        case 'd'
            amplitude_expo=input("enter the amplitude of the exponential signal ");
            exponent=input("enter the exponent of the exponential signal ");
            t=linspace(position(i),position(i+1),(position(i+1)-position(i))* samplingFrequency);

            signal =amplitude_expo* exp(exponent*t) ;

            plot(t,signal,'g-','LineWidth',1)
            hold on;
        case 'e'
            amplitude_sinusoidal=input("enter the amplitude of the sinusoidal signal ");
            freqency=input("enter the freqency of the sinusoidal signal ");
            phase= input("the phase of the sinusoidal signal ");
            t=linspace(position(i),position(i+1),(position(i+1)-position(i))* samplingFrequency);

            signal = amplitude sinusoidal*sin(2*pi*freqency*t+phase);
```

```

            plot(t,signal,'y-','LineWidth',1)
            hold on;
        case 'f'
            amplitude_sinc = input("enter the amplitude of sinc function ");
            center_shift_sinc = input("enter the center shift of the sinc function ");
            t=linspace(position(i),position(i+1),(position(i+1)-position(i))* samplingFrequency);

            signal = amplitude_sinc*sinc(t-center_shift_sinc);

            plot(t,signal,'g-','LineWidth',1)
            hold on;
        case 'g'
            amplitude_triangle = input("enter the amplitude of the triangle signal ");
            center_shift_triangle = input("enter the center shift of triangle signal ");
            width= input("enter the width of triangle signal ");

            t=linspace(position(i),position(i+1),(position(i+1)-position(i))* samplingFrequency);
            signal = amplitude_triangle *sawtooth(2*pi*(t-center_shift_triangle)/width,0.5);

            plot(t,signal,'g-','LineWidth',1)
            hold on;
    end
    if i== numberOfBreakPoints+1
        hold off;
    end
    f = cat(2,signals,signal);
    signals=f;
    m=f;
```

-The program asks the user the type of signals he wants at each region

-For each type, the program asks the user for the signals specification like amplitude for dc signal or slope for ramp signal

-the number of iterations of for loop = no. of break points+1

- Ask the user if he wants to perform any operation on the signal

  -After choosing the operation, the user enters the c/cs of the operation (like shift value for time shift) and the programs displays the new signal in time domain.

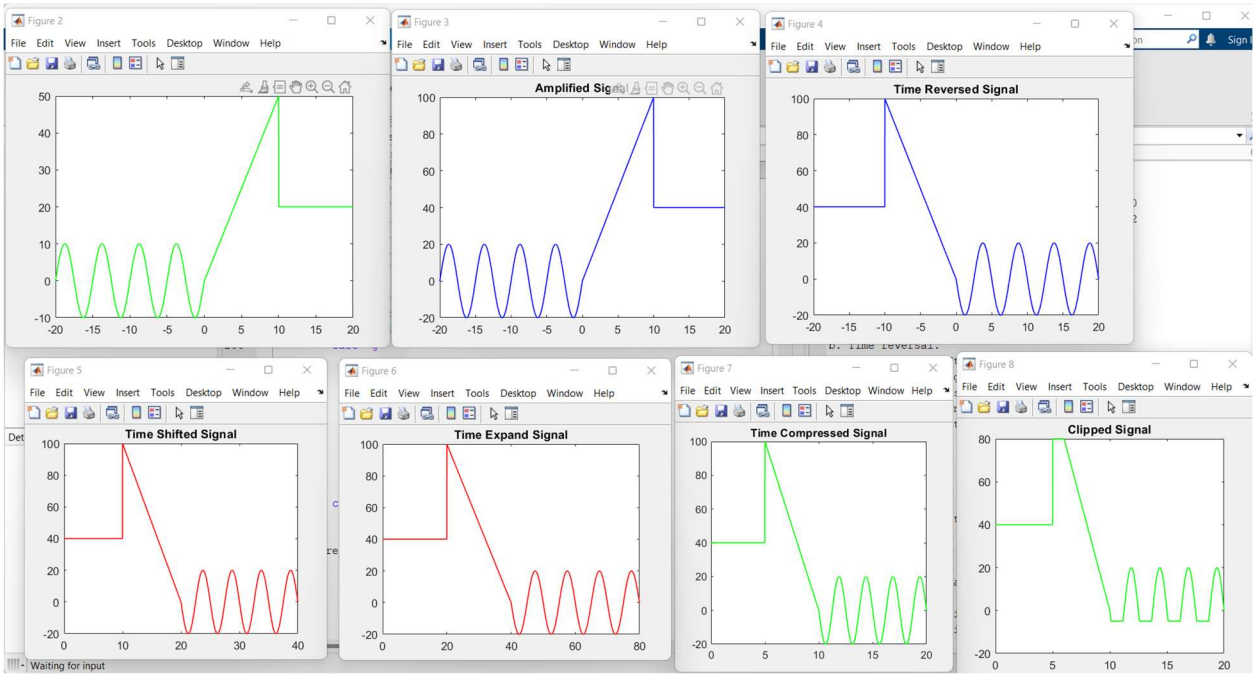
```
129 -   fprintf("a. Amplitude Scaling: scale value. \nb. Time reversal. \nc. Time shift: shift value.\nd. Expanding
130 -   figure_num=3;
131 -   t=linspace(starting_time,(ending_time),(ending_time-starting_time)*samplingFrequency);  %to prevent any chan
132
133 -   while 1
134 -       c =input("which operation you want to perform on signals");
135 -       while ~(ischar(c))
136 -            c =input("please enter a character which listed\n");
137 -       end
138
139 -       switch c
140 -           case 'a'
141 -               amplitude_scaling = input("enter the magnitude of ampltude scale");%amplitude scaling
142 -               t1=linspace(starting_time,(ending_time),(ending_time-starting_time)*samplingFrequency);
143 -               signals=signals * amplitude_scaling;
144 -               figure(figure_num)
145 -               plot(t1,signals,'b-','LineWidth',1)
146 -               title('Amplified Signal')
147               %hold on;
148               %f=signals;
149 -           case 'b'
150 -               signals=flip(signals);%time reverse
151 -               t2=linspace(starting_time,(ending_time),(ending_time-starting_time)*samplingFrequency);
152 -               t2=-1. * t2;
153 -               figure(figure_num)
154 -               signals=signals(end:-1:1);
155 -               plot(t2,signals,'b-','LineWidth',1)
156 -               title('Time Reversed Signal')
157               %temp = starting_time
158               %starting_time = -ending_time
159               %ending_time = -temp
160               %f=signals;
```

```
        case 'c'
           time_shift = input("enter the time shift ");%time shift
           t3=linspace(starting_time,(ending_time),(ending_time-starting_time)*samplingFrequency);
           t3=t3 + time_shift;
           figure(figure_num)
           plot(t3,signals,'r-','LineWidth',1)
           title('Time Shifted Signal')

           %f=signals;
           %hold on;
        case 'd'
            expand=input("enter the expanding value");%time expantion
            %t=expand*t;
            start_expantion=starting_time * expand;
            end_expantion=ending_time * expand;
            t4=linspace(start_expantion,end_expantion,(end_expantion - start_expantion)*samplingFrequency);
            signals=resample(signals,expand,1,'Dimension',2);
            figure(figure_num)
            plot(t4,signals,'r-','LineWidth',1)
            title('Time Expand Signal')
            %f=signals;
            ending_time=end_expantion;
            starting_time=start_expantion;
            %hold on;
        case 'e'
           compress= input("enter compression factor");%time compression
           start_compression=starting_time/compress;
           end_compression=ending_time/compress;
           t5=linspace(start_compression,(end_compression),(end_compression-start_compression)*samplingFrequ
           %t5=t5./compress;
           signals = downsample(signals,compress);
           figure(figure_num);
           plot(t5,signals,'g-','LineWidth',1);
           title('Time Compressed Signal');
           ending_time=end_compression;
           starting_time=start_compression;
           %f=signals;
           %hold on;
        case 'f'
            min = input("enter the minmum clipping  ");%clipping
            max = input("enter the maxmum clipping  ");
            %signals = f;
            t6=linspace(starting_time,(ending_time),(ending_time-starting_time)*samplingFrequency);
            f=signals;
            signals(f < min) = min;
            signals(f > max) = max;
            figure(figure_num)
            plot(t6,signals,'g-','LineWidth',1)
            title('Clipped Signal')
            %f=signals;
            %hold on;
        case 'g'
            t7=linspace(starting_time,(ending_time),(ending_time-starting_time)*samplingFrequency);
            diffsignal=zeros(1,size(signals,2));
            for k=1:(size(t7,2)-1)
                diffsignal(k)=(signals(k+1)-signals(k))/(t7(k+1)-t7(k));
            end
            z=zeros(1,i);
            signal = [diffsignal z];
            signals=signal;
            figure(figure_num)
            plot(t7,signals,'g-','LineWidth',1)
            title('Differntiate Signal')
```

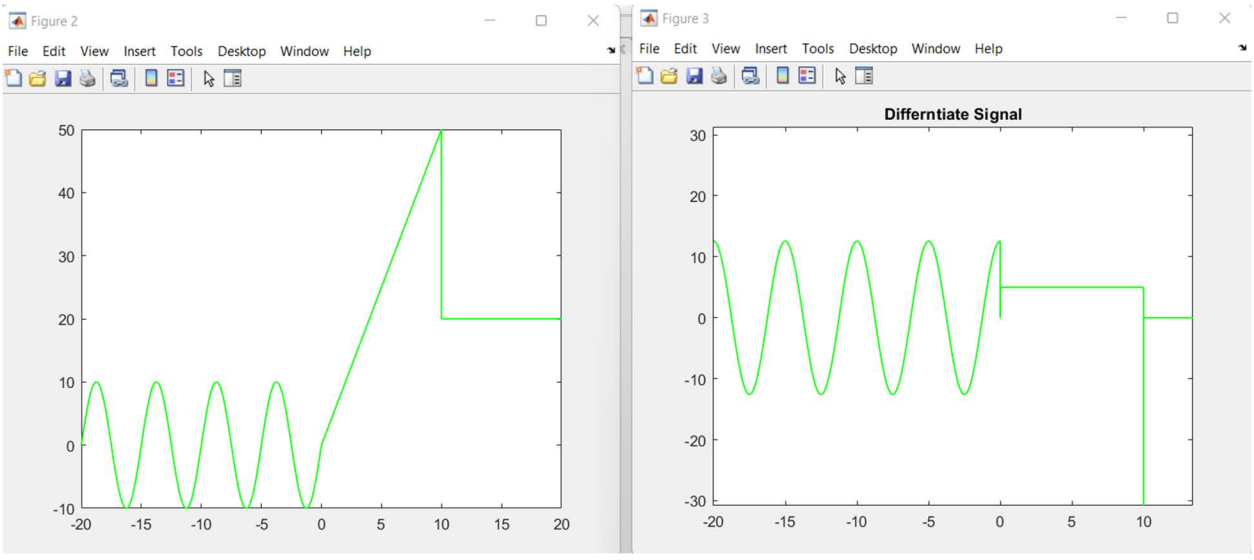
```
            %f=signals;
        case 'h'
            signals= m;
            figure(figure_num)
            plot(t,signals,'g-','LineWidth',1)
            title('nothing')
            ending_time=e;
            starting_time=s;
            fprintf("%d",starting_time);
            fprintf("%d",ending_time);
     end
    figure_num=figure_num+1;

end
```

Screenshots of output:

```
>> mini_project
enter the Sampling frequency
1000
enter the Starting time
-20
enter the ending time
20
enter the number Of Break Points
2
enter the position number 1 0
enter the position number 2 10
a. DC signal
b. Ramp signal
c. General order polynomial
d. Exponential signal
e. Sinusoidal signal
f. Sinc function
g. Triangle pulse
enter type signal from -20 second to 0 'e'
enter the amplitude of the sinusoidal signal 10
enter the freqency of the sinusoidal signal 0.2
the phase of the sinusoidal signal 0
enter type signal from 0 second to 10 'b'
enter slope of the ramp signal 5
enter the intercept of the ramp signal 0
enter type signal from 10 second to 20 'a'
enter the amplitude of DC signal20
a. Amplitude Scaling: scale value.
b. Time reversal.
c. Time shift: shift value.
d. Expanding the signal: expanding value
e. Compressing the signal: compressing value
f. Clipping the signal: upper and Lower clipping values
g. The first derivative of the signal.
h. None.
which operation you want to perform on signals'a'
enter the magnitude of ampltude scale2
which operation you want to perform on signals'b'
which operation you want to perform on signals'c'
enter the time shift 20
which operation you want to perform on signals'd'
enter the expanding value2
which operation you want to perform on signals'e'
enter compression factor4
which operation you want to perform on signals'f'
enter the minmum clipping  -5
enter the maxmum clipping  80
which operation you want to perform on signals
```



-differentiate signal:



Note that the function is Dirichlet at t=10