MATLAB general signal generator

Ahmed Haitham Ahmed, 41

Peter Magdy Fouad, 69

Aliaa Nemir, 132

Kirollos Adel, 157

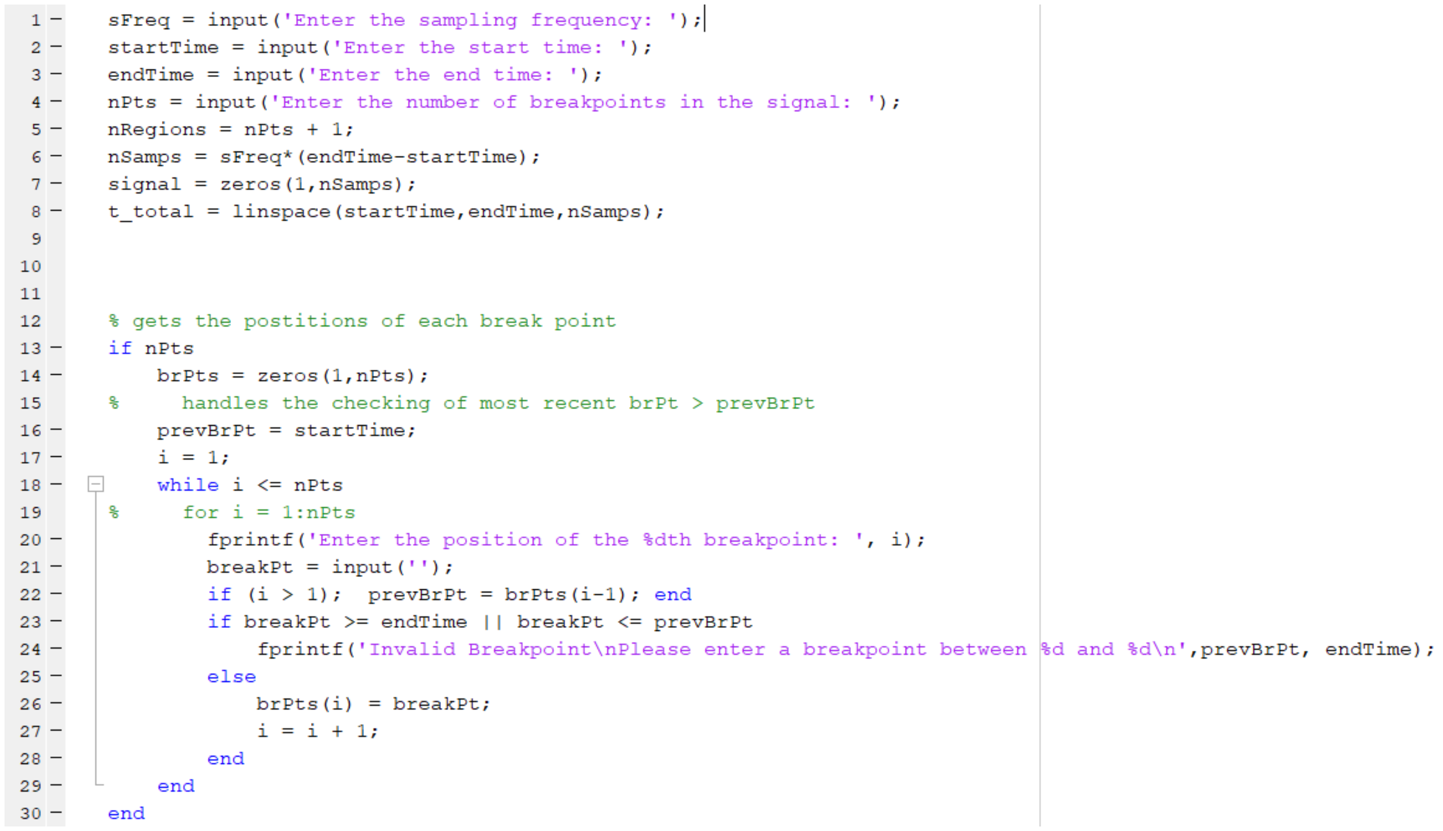
Signal generators:

A signal generator is an electronic device that generate electronic signals and waveform. These electronic signals can either be periodic or non-periodic as required in the different applications.

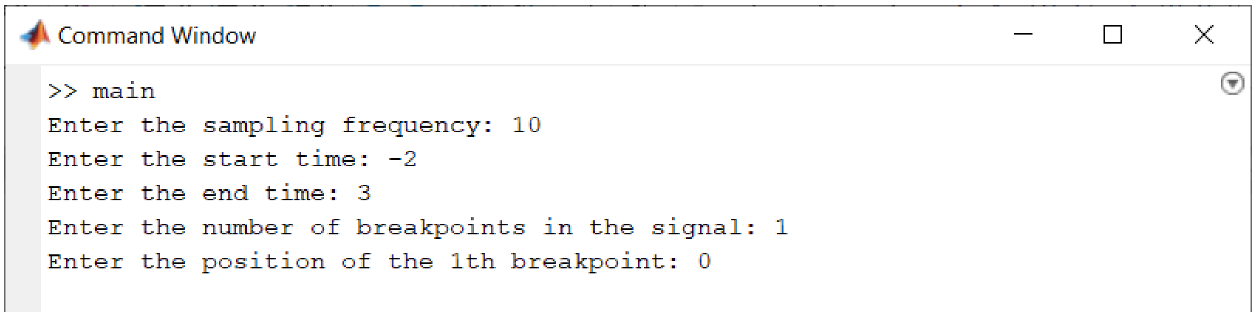
Required, using MATLAB, to create a **general signal generator** that generate a multi-definition continuous signal with as many breakpoints, regions, and types of signal.

The program starts by asking the user to enter

- The sampling frequency, which will be used to plot the continuous signal.
- The start & end time scale, within which the signal will be plotted.
- The number of breakpoints (nPts), which divides the signal to (nPts+1) regions each contains a specific type of signal.
- The position of breakpoints, within the time scale.

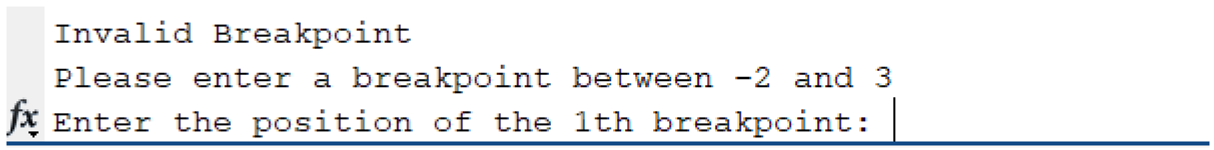


```
sFreq = input('Enter the sampling frequency: ');
startTime = input('Enter the start time: ');
endTime = input('Enter the end time: ');
nPts = input('Enter the number of breakpoints in the signal: ');
nRegions = nPts + 1;
nSamps = sFreq*(endTime-startTime);
signal = zeros(1,nSamps);
t_total = linspace(startTime,endTime,nSamps);



% gets the postitions of each break point
if nPts
    brPts = zeros(1,nPts);
%     handles the checking of most recent brPt > prevBrPt
    prevBrPt = startTime;
    i = 1;
    while i <= nPts
%     for i = 1:nPts
        fprintf('Enter the position of the %dth breakpoint: ', i);
        breakPt = input('');
        if (i > 1);  prevBrPt = brPts(i-1); end
        if breakPt >= endTime || breakPt <= prevBrPt
            fprintf('Invalid Breakpoint\nPlease enter a breakpoint between %d and %d\n',prevBrPt, endTime);
        else
            brPts(i) = breakPt;
            i = i + 1;
        end
    end
end
```



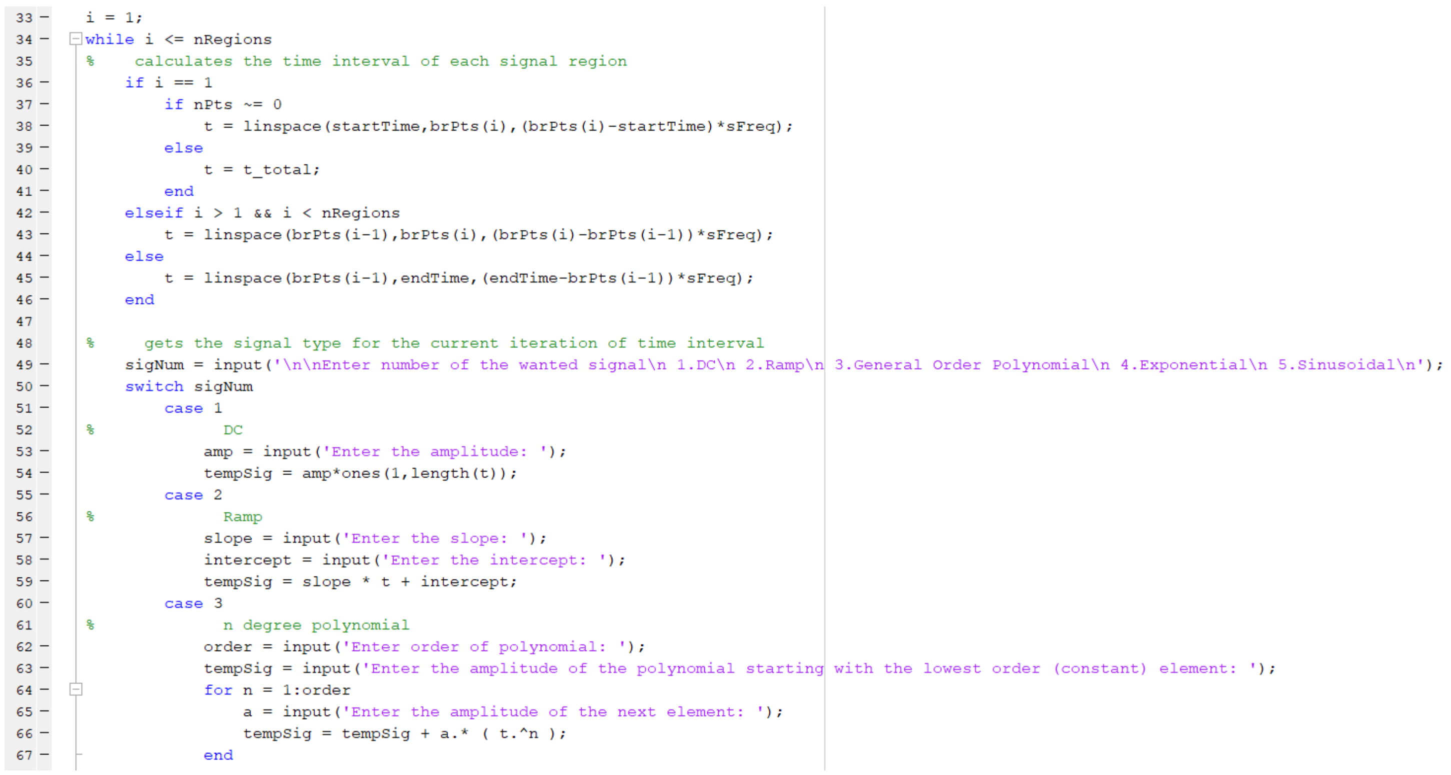
```
>> main
Enter the sampling frequency: 10
Enter the start time: -2
Enter the end time: 3
Enter the number of breakpoints in the signal: 1
Enter the position of the 1th breakpoint: 0
```

The program checks if the breakpoint is within the time scale or not if not, it prints ‘Invalid Breakpoint’ and asks the user to enter another one.

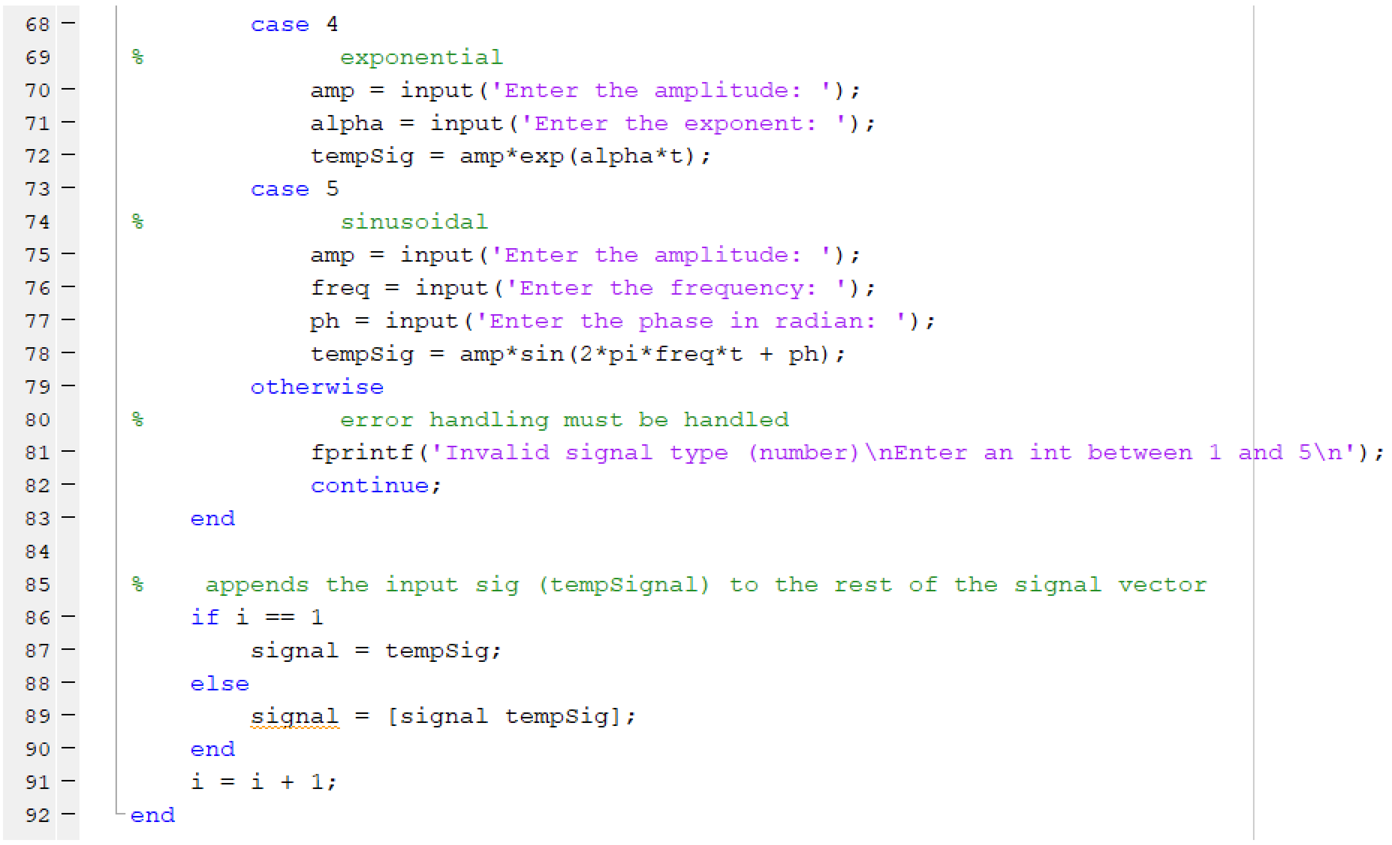


```
Invalid Breakpoint
Please enter a breakpoint between -2 and 3
Enter the position of the 1th breakpoint:
```

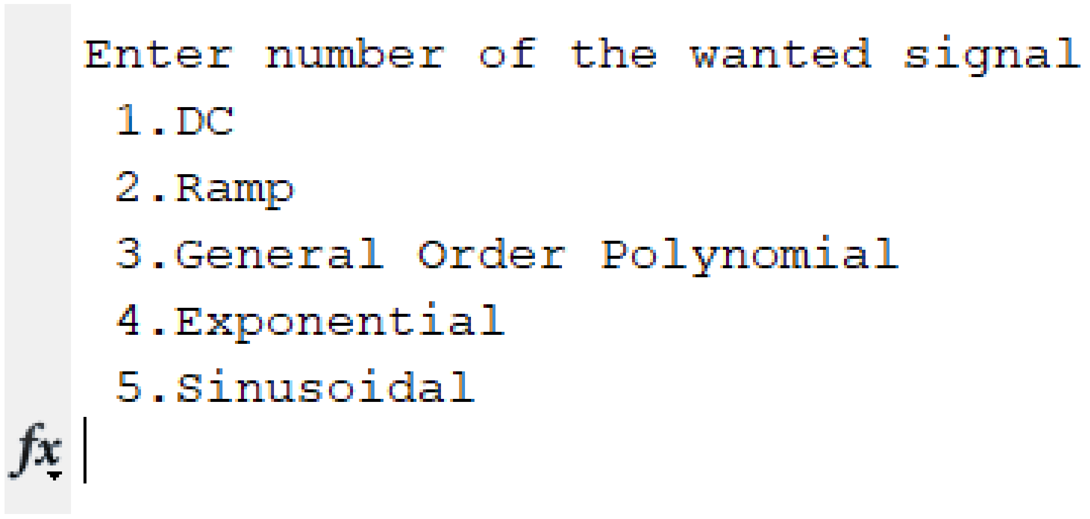
Next, the program asks to enter the type of signal wanted in each region. It can be DC, Ramp, General order polynomial, Exponential or Sinusoidal signal. Then depending on each type, it asks different parameters that helps generating the signal (i.e., for DC, it asks the amplitude).



```
i = 1;
while i <= nRegions
%    calculates the time interval of each signal region
    if i == 1
        if nPts ~= 0
            t = linspace(startTime,brPts(i),(brPts(i)-startTime)*sFreq);
        else
            t = t_total;
        end
    elseif i > 1 && i < nRegions
        t = linspace(brPts(i-1),brPts(i),(brPts(i)-brPts(i-1))*sFreq);
    else
        t = linspace(brPts(i-1),endTime,(endTime-brPts(i-1))*sFreq);
    end

%     gets the signal type for the current iteration of time interval
    sigNum = input('\n\nEnter number of the wanted signal\n 1.DC\n 2.Ramp\n 3.General Order Polynomial\n 4.Exponential\n 5.Sinusoidal\n');
    switch sigNum
        case 1
%             DC
            amp = input('Enter the amplitude: ');
            tempSig = amp*ones(1,length(t));
        case 2
%             Ramp
            slope = input('Enter the slope: ');
            intercept = input('Enter the intercept: ');
            tempSig = slope * t + intercept;
        case 3
%             n degree polynomial
            order = input('Enter order of polynomial: ');
            tempSig = input('Enter the amplitude of the polynomial starting with the lowest order (constant) element: ');
            for n = 1:order
                a = input('Enter the amplitude of the next element: ');
                tempSig = tempSig + a.* ( t.^n );
            end
```



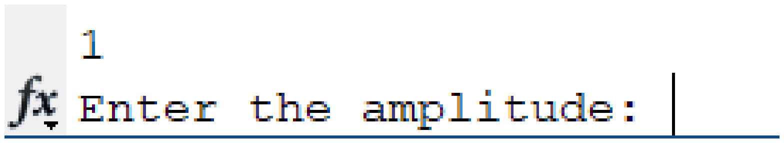
```
        case 4
%             exponential
            amp = input('Enter the amplitude: ');
            alpha = input('Enter the exponent: ');
            tempSig = amp*exp(alpha*t);
        case 5
%             sinusoidal
            amp = input('Enter the amplitude: ');
            freq = input('Enter the frequency: ');
            ph = input('Enter the phase in radian: ');
            tempSig = amp*sin(2*pi*freq*t + ph);
        otherwise
%             error handling must be handled
            fprintf('Invalid signal type (number)\nEnter an int between 1 and 5\n');
            continue;
    end

%    appends the input sig (tempSignal) to the rest of the signal vector
    if i == 1
        signal = tempSig;
    else
        signal = [signal tempSig];
    end
    i = i + 1;
end
```



```
Enter number of the wanted signal
 1.DC
 2.Ramp
 3.General Order Polynomial
 4.Exponential
 5.Sinusoidal
fx
```

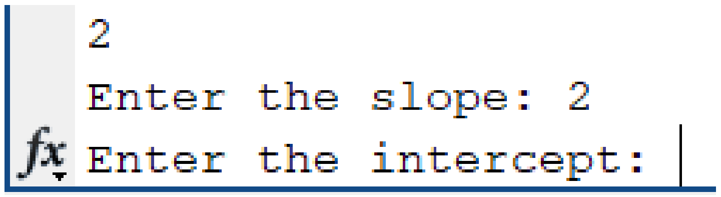
As mentioned before, each signal has its different parameters that the program asks the user to enter.

- For DC signal; it asks the amplitude.



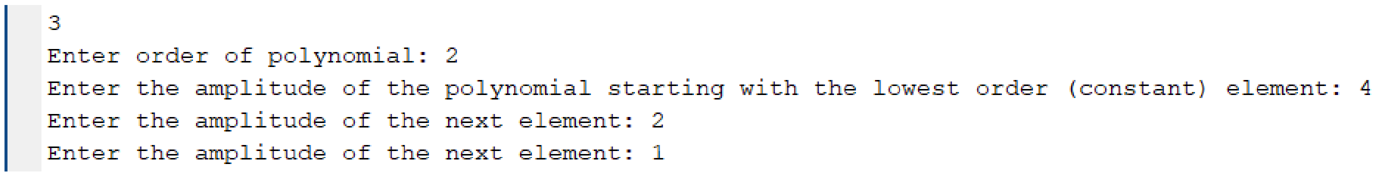
```
1
Enter the amplitude: 
```

- For Ramp signal; slope and intercept.



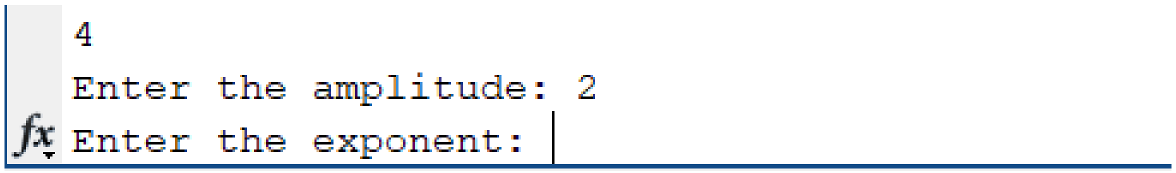
```
2
Enter the slope: 2
Enter the intercept: 
```

- For General order polynomial; Amplitude, power and intercept.



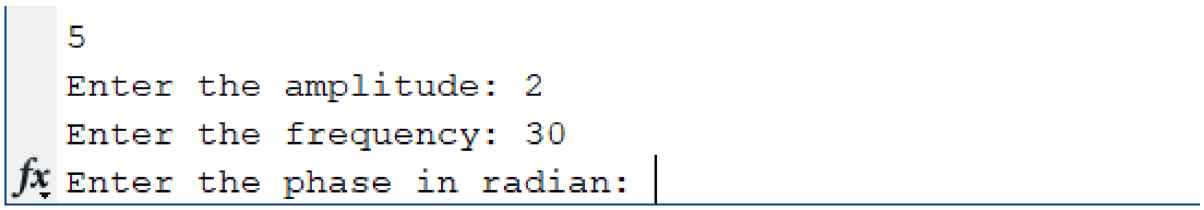
```
3
Enter order of polynomial: 2
Enter the amplitude of the polynomial starting with the lowest order (constant) element: 4
Enter the amplitude of the next element: 2
Enter the amplitude of the next element: 1
```

- For Exponential signal; amplitude and exponent.



```
4
Enter the amplitude: 2
Enter the exponent: 
```

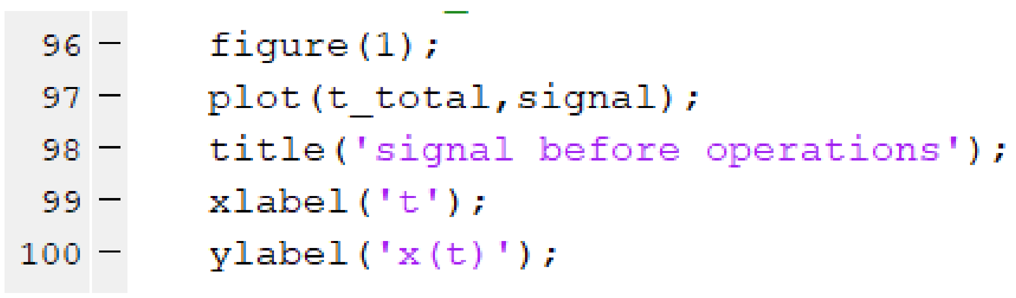
- For Sinusoidal signal; amplitude, frequency and phase.



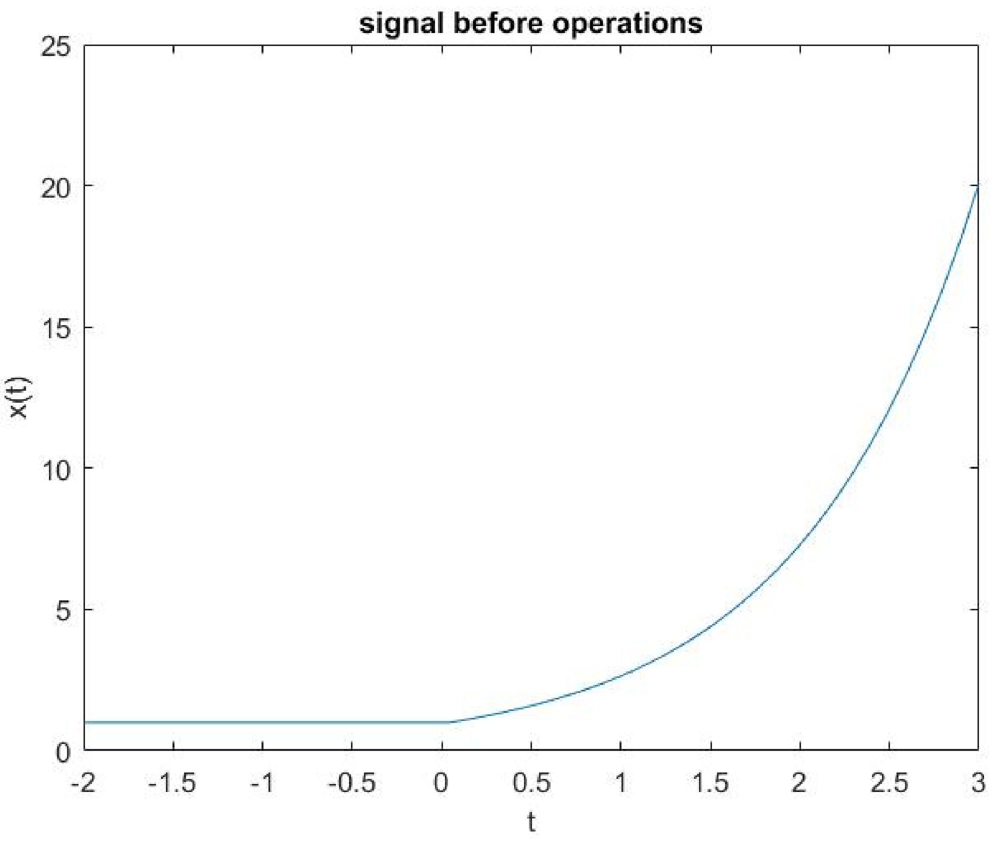
```
5
Enter the amplitude: 2
Enter the frequency: 30
Enter the phase in radian: 
```

After taking the necessary parameters the program generates the signal and plots it in the time domain.

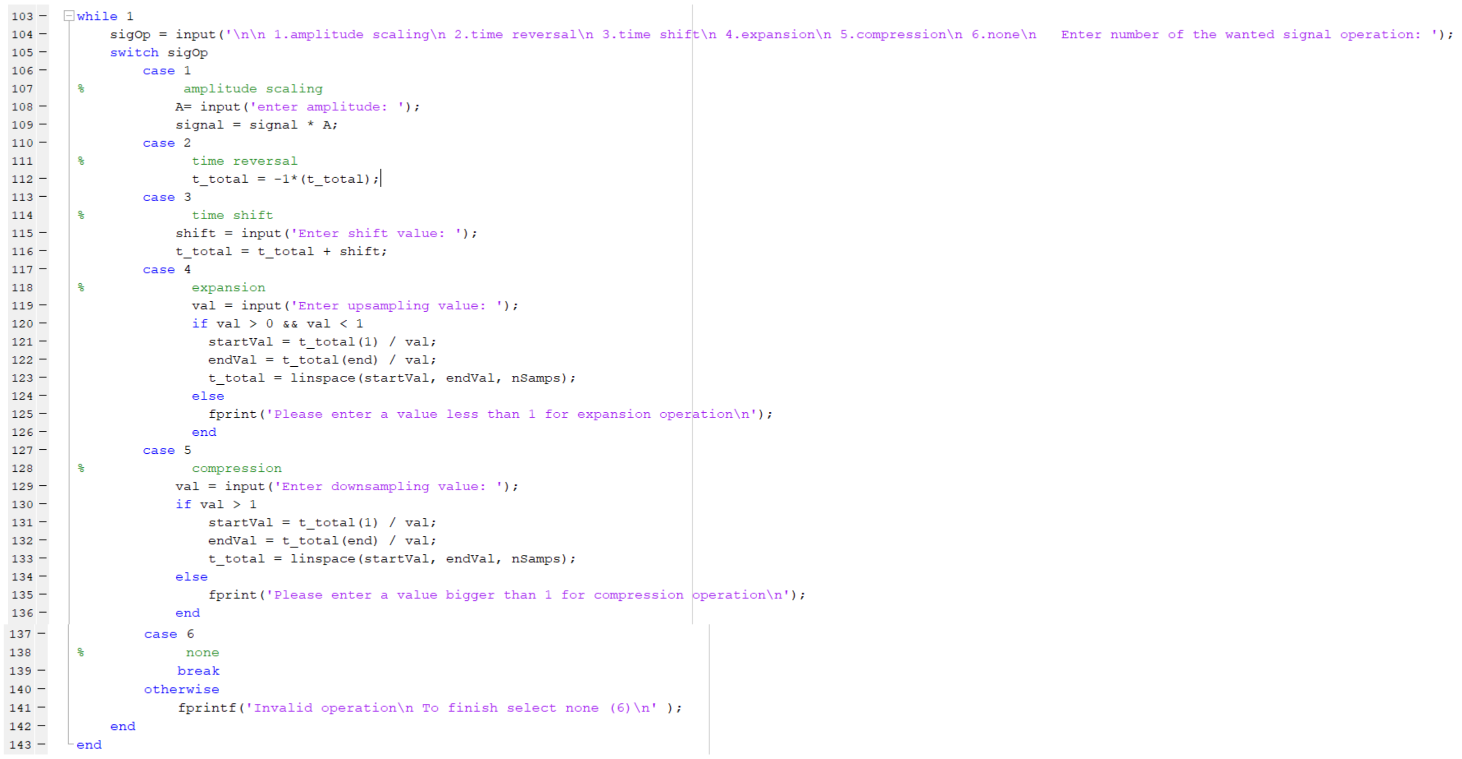
Ex: A signal with 1 breakpoint at zero, DC from -2 to 0, and exponential from 0 to 3.



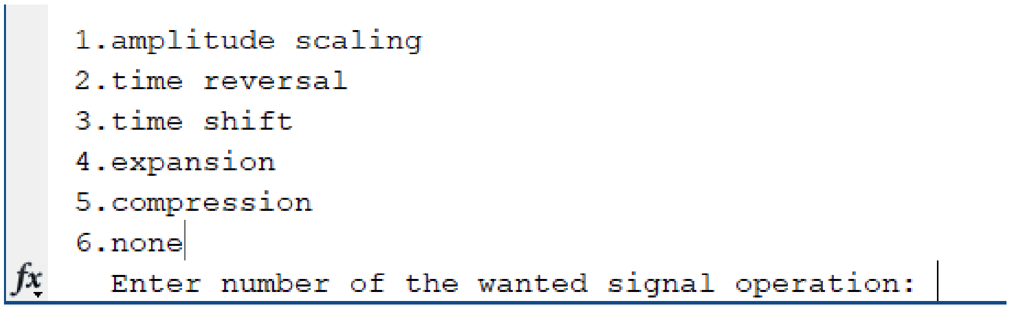
```
figure(1);
plot(t_total,signal);
title('signal before operations');
xlabel('t');
ylabel('x(t)');
```



To perform any operation on the signal, the user enters a number from 1 to 5 to choose the desired operation whether it is **amplitude Scaling**, **time reversal**, **time shift**, **expanding the signal** or **compressing the signal** as displayed in the following figure.



```matlab
while 1
    sigOp = input('\n\n 1.amplitude scaling\n 2.time reversal\n 3.time shift\n 4.expansion\n 5.compression\n 6.none\n   Enter number of the wanted signal operation: ');
    switch sigOp
        case 1
%            amplitude scaling
            A= input('enter amplitude: ');
            signal = signal * A;
        case 2
%             time reversal
              t_total = -1*(t_total);
        case 3
%             time shift
            shift = input('Enter shift value: ');
            t_total = t_total + shift;
        case 4
%             expansion
              val = input('Enter upsampling value: ');
              if val > 0 && val < 1
                startVal = t_total(1) / val;
                endVal = t_total(end) / val;
                t_total = linspace(startVal, endVal, nSamps);
              else
                fprint('Please enter a value less than 1 for expansion operation\n');
              end
        case 5
%             compression
            val = input('Enter downsampling value: ');
            if val > 1
                startVal = t_total(1) / val;
                endVal = t_total(end) / val;
                t_total = linspace(startVal, endVal, nSamps);
            else
                fprint('Please enter a value bigger than 1 for compression operation\n');
            end
        case 6
%            none
            break
        otherwise
            fprintf('Invalid operation\n To finish select none (6)\n' );
    end
end
```



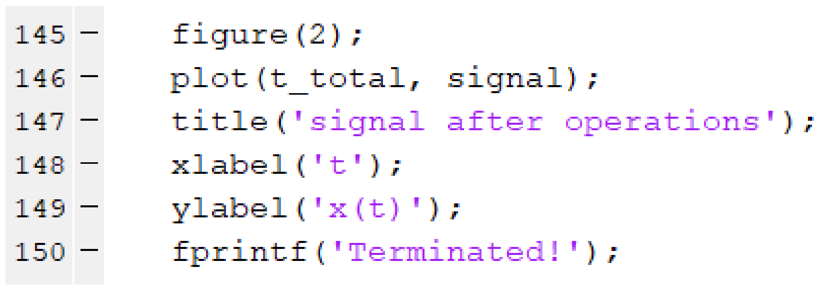
```
1.amplitude scaling
2.time reversal
3.time shift
4.expansion
5.compression
6.none
   Enter number of the wanted signal operation:
```

If no operation desired the user can enter '6' and the program will end.

There are several values to be taken in consideration when choosing an operation to be done on the signal.

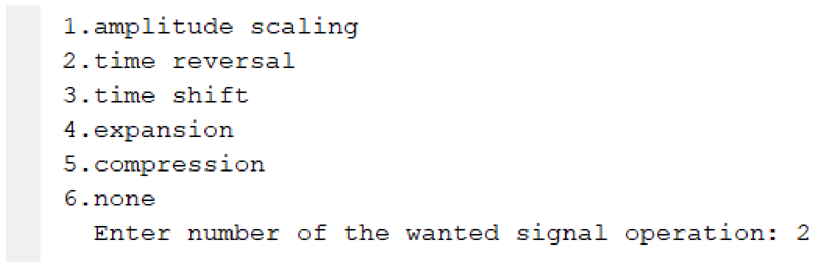
In case of **amplitude scaling**, *the scale value*; **time shift**, *the shift value*; **expanding and compressing the signal**, the corresponding *expanding and compressing values.*

After choosing the desired operations, the program then plots the new signal against time.

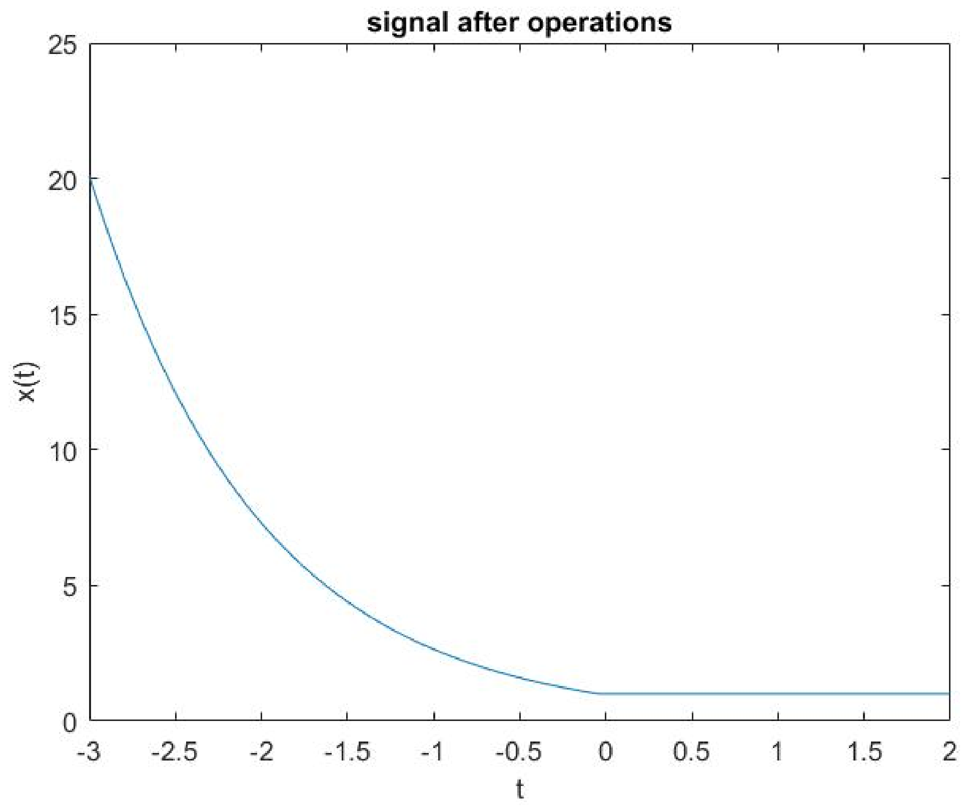


```
figure(2);
plot(t_total, signal);
title('signal after operations');
xlabel('t');
ylabel('x(t)');
fprintf('Terminated!');
```

Shown is the time reversal operation of the signal generated before and plotted in time domain.



```
1.amplitude scaling
2.time reversal
3.time shift
4.expansion
5.compression
6.none
  Enter number of the wanted signal operation: 2
```


signal after operations
x(t)
t