**Signals & Systems**

**EEE-223**

Lab # 04



.

|  |  |
| --- | --- |
| Name | Muhammad Haris Irfan |
| Registration Number | FA18-BCE-090 |
| Class | BCE-4A |
| Instructor’s Name | Muhammad Bilal Qasim |

**Lab 04 - Signal Transformations (Scaling, Shifting and Reversal)**

**Pre-lab Tasks**

* 1. **Transformations of the Time Variable for Continuous-Time Signals:**

In many cases, there are signals related to each other with operations performed in the independent variable, namely, the time. In this lab, we will examine the basic operations that re performed on the independent variable.

**4.1.1 Time Reversal or Reflection:**

The first operation performed is the signal’s reflection. A signal is a reflection or a reflected version of about the interval axis if.

The operation of time reversal is actually an alteration of the signal values between negative and positive time. Assume that  is the vector that denotes the signal  in time . The MATLAB statement that plots the reflected version of  is **plot(-t,x).**

**Example**

Suppose that  Plot the signal.

t=-1:0.1:3;

x=t.\*exp(-t);

subplot(2,1,1)

a=2;b=1/2;

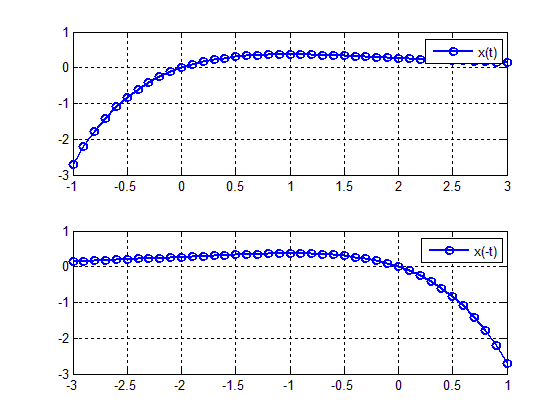
plot(t,x,'-o','Linewidth',2),grid on

legend('x(t)')

subplot(2,1,2)

plot(-t,x,'-o','Linewidth',2),grid on

legend('x(-t)')



**Figure 4.1:** Graphs of original and 

In order to draw, we just to use the command **plot(-t,x).** The graph of is given in Figure 4.1 bottom.

**4.1.2 Time Scaling:**

The second operation discussed is time scaling. A signal is a compressed version of  if . The time compression practically means that the time duration of the signal is reduced by a factor of  On the other hand, a signal is an expanded version of if . In this case, the time duration of the signal is increased by a factor of.

In order to plot in MATLAB a time-scaled version of, namely, a signal of the form, the statement employed is **plot ((1/a)\*t,x).** In contrast to what, someone would expect the vector of time  must be multiplied by  and not 

**Example**

Consider again the continuous time signal. We will plot the signal , which is a time compression of  by a factor of ; and the signal , which is a time expansion of by a factor .

t=-1:0.1:3;

x=t.\*exp(-t);

subplot(3,1,1)

a=2;b=1/2;

plot(t,x,'-o','Linewidth',2),g

rid on

legend('x(t)')

subplot(3,1,2)

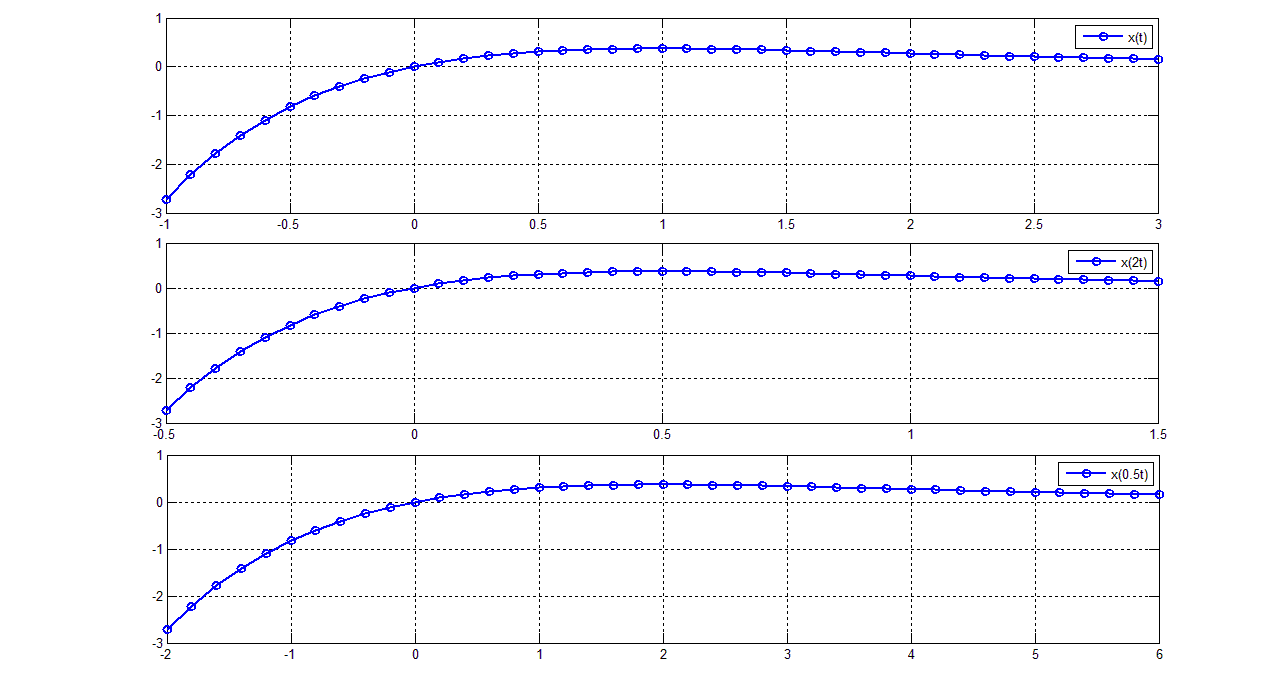
plot((1/a)\*t,x,'-o','Linewidth',2),grid on

legend('x(2t)')

subplot(3,1,3)

plot((1/b)\*t,x,'-o','Linewidth',2),grid on

legend('x(0.5t)')



**Figure 4.2:** Graphs of 

**4.1.3 Time Shifting:**

A third operation performed on time is one of time shifting. A signal is a time shifted version ofif, where is the time shift. If, the signal is shifted by  units to the right (i.e. towards); while if, the signal is shifted by  units to the left (i.e. towards). The time shifting is implemented in MATLAB in an opposite way to what may be expected. More specifically, in order to plot the corresponding MATLAB statement is **plot(t+t0,x).**

**Example**

The signal is again considered. We will plot the signal, that is a shifted version of by two units to the right (here) and, that is, a shifted version ofby 3 units to the left (here).

t=-1:0.1:3;

x=t.\*exp(-t);

t0=2;

t1=-3;

subplot(3,1,1)

plot(t,x,'-o','Linewidth',2),grid on

legend('x(t)')

subplot(3,1,2)

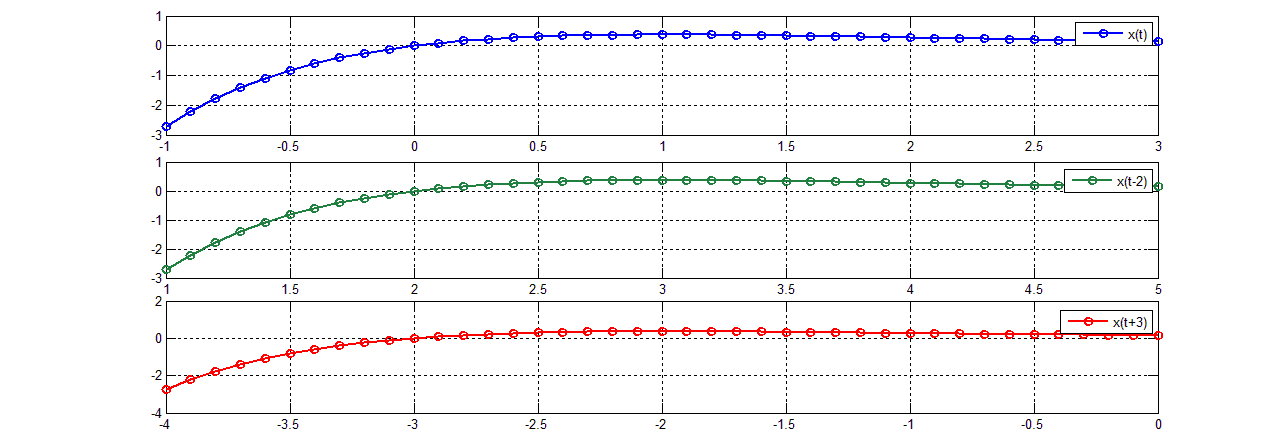
plot(t+t0,x,'-o','Linewidth',2),grid on

legend('x(t-2)')

subplot(3,1,3)

plot(t+t1,x,'-o','Linewidth',2),grid on

legend('x(t+3)')



**Figure 4.3:** Graphs of 

The order (shifting-scaling-reversal) must be strictly followed in order to correctly plot a signal. The fact that entire expression (which is the first argument of the command plot) is multiplied by scaling factor and afterward reversed is easily explained if we consider that it represents the time interval in which the signal is plotted.

* 1. **Transformations of the Time Variable for Discrete-Time Signals:**

Suppose that is a discrete time signal defined over the time interval specified byand is an integer number. The three transformations for the time variable for discrete time signals are the following:

* Time Shifting. This operation is similar to the continuous time operation. More specifically, the signal is shifted by units (or samples) to the right if , and is shifted by  units to the left if . The associated MATLAB statement is **stem(n+n0,x).**
* Time Reversal is also similar to operation performed for the continuous-time signals. It is described by the relationship, whereis the reflected (about the vertical axis) signal. Time reversal is implemented in MATLAB by typing **stem(-n,x).**
* Time scaling. This transformation however is somehow different from the one described in the continuous time case. The relationship the time scaling operation is If  and , the time scaling operation is called downsampling or decimation. Notice that must be an integer as  is defined for fractional values of . The downsampling operation results in time compression of the signal. Moreover, some samples of  are lost. The downsampling operation is implemented in MATLAB by using the command y=downsample(x,a)or the statement y=x(1:a:end). The variable end represents the last index in an indexing expression. If , the signal is a time expanded version of . In this case,  zeros are inserted between two consecutive samples of . The time expression operation is called upsampling, and is implemented in MATLAB command y=upsample(x,1/a) or with the statement y(1:1/a:end)=x.

**Remarks**

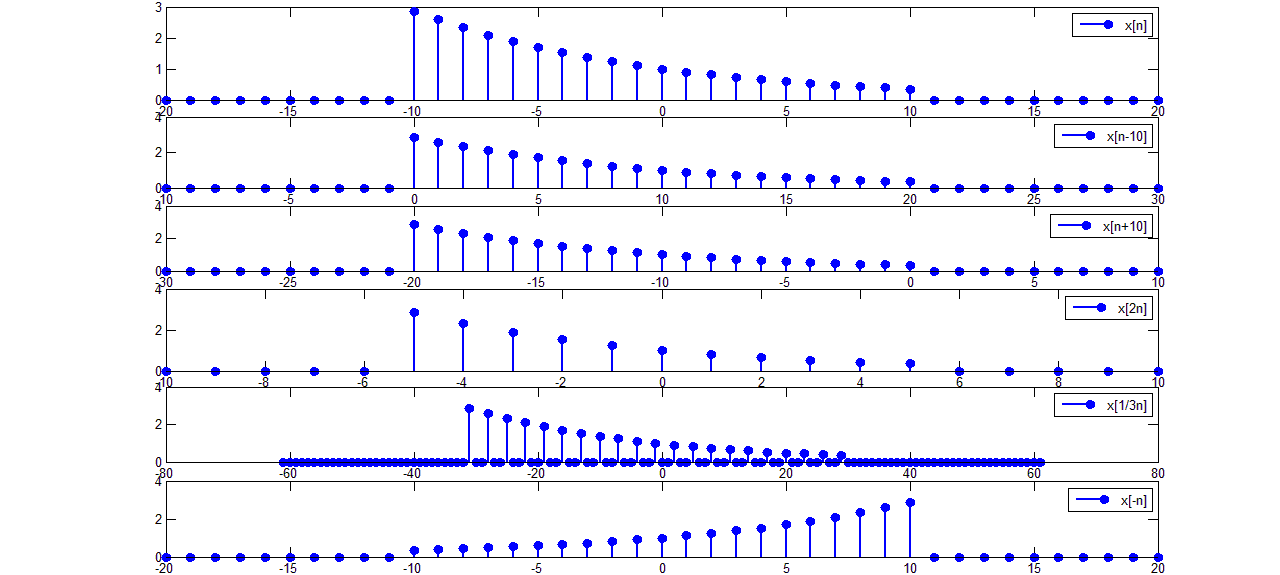
There are some limitations on the values that can take. In case of downsampling  must be a positive integer, and in case of upsampling must be a positive integer.

The downsampling and the upsampling processes are illustrated in the next example for the discrete time signal .

|  |  |  |
| --- | --- | --- |
| **Commands** | **Results** | **Comments** |
| **x=[1 2 3 4 5 6]** | x=1 2 3 4 5 6 | The discrete time signal |
| **a=2;**  **xds=downsample(x,a)** | xds=1 3 5 | The downsampled version of |
| **xds=x(1:a:end)** | xds=1 3 5 | Alternative computation of the downsampled signal. |
| **a=1/2;**  **xups=upsample(x,1/a)** | xups=1 0 2 0 3 0 4 0 5 0 6 0 | Upsampling operation on |
| **xups=zeros(1,1/a\*length(x))**  **xups(1:1/a:end)=x** | xups=1 0 2 0 3 0 4 0 5 0 6 0 | Upsampling operation performed in an alternative way. Notice that the variable in which the upsampled signal is stored must be first defined as a vector of zeros with length.length |

**Example**

Consider the sequence  Plot the sequences  and.



**Figure 4.4:** Graphs of and 

An alternative way to obtain the signal  is by using the command fliplr. This command flips the order of the vector elements. To demonstrate its use, the signal is considered.

n=-2:4; %n= -2 -1 0 1 2 3 4

n1=-fliplr(n); %n1=-4 -3 -2 -1 0 1 2

x=0.9.^n; %x= 1.23 1.11 1.00 0.90 0.81 0.73 0.66

x1=fliplr(x); %x1=0.66 0.73 0.81 0.90 1.00 1.11 1.23

subplot(1,2,1)

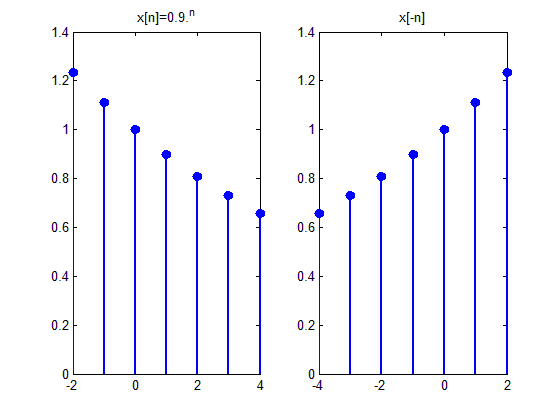
stem(n,x,'fill','linewidth',2)

title('x[n]=0.9.^n')

subplot(1,2,2)

stem(n1,x1,'fill','linewidth',2)

title('x[-n]')



**Figure 4.5:** Graphs of and 

**In-lab Tasks**

**Task 01: Plot the signal  for , where is a rectangular pulse of duration T, denoted by .**

|  |
| --- |
| t = -2:0.001:5;  u1(t>=0) = 1;  u2(t>=2) = 1;  u = u1 - u2;  subplot(2,1,1)  plot(t,u)  xlabel('x');  ylabel('p2(t-1)');  x = t.^3.\*cos(10\*pi\*t).\*u;  subplot(2,1,2)  plot(t,x)  legend('x(t)');  grid on  Chart  Description automatically generated |

**Task 02: Suppose that  Plot the signals**

1. ****
2. 
3. 
4. 
5. 

|  |
| --- |
| t = 0:0.0001:5;  x = t.\*cos(2\*pi\*t);  subplot(5,1,1)  plot(t,x,'LineWidth',2)  grid on  xlabel('t');  legend('x(t)')    subplot(5,1,2)  plot(-t,x,'LineWidth',2)  grid on  xlabel('t');  legend('x(-t)')    a = 1/5;  subplot(5,1,3)  plot((1./a).\*t,x,'LineWidth',2)  grid on  xlabel('t');  legend('x(0.2t)')    subplot(5,1,4)  plot((1/3).\*(t-1),x,'LineWidth',2)  grid on  xlabel('t');  legend('x(1+3t)')    subplot(5,1,5)  plot((-1/3).\*(t+1),x,'LineWidth',2)  grid on  xlabel('t');  legend('x(-1-3t)')  A picture containing graphical user interface  Description automatically generated |

**Task 03: Suppose that . Plot the signals**

1. 
2. 
3. 
4. 
5. 

|  |
| --- |
| t = 0:0.01:4;  x = t.\*(t<=2) + (4-t).\*(t>2);  subplot(5,1,1)  plot(t,x,'LineWidth',2)  grid on  legend('x(t)')    subplot(5,1,2)  plot(-t,x,'LineWidth',2)  grid on  legend('x(-t)')    subplot(5,1,3)  a = 1/2;  plot((1/a).\*t,x,'LineWidth',2)  grid on  legend('x(t/2)')    subplot(5,1,4)  b = 4;  plot((1/b).\*(t-2),x,'LineWidth',2)  grid on  legend('x(2+4t)')    subplot(5,1,5)  c = -4;  plot((1/c).\*(t+2),x,'LineWidth',2)  grid on  legend('x(-2-4t)')  Chart, line chart  Description automatically generated |

**Task 04: Write a function that accepts a sequence, the discrete time n and a number n0, a, b as input arguments, and returns the signals and. Whererepresents the time compressed version of  andis the time expanded version of **.

|  |
| --- |
| function [ ] = sequence(x,n,n0,a,b)    subplot(5,1,1);  stem(n,x,'LineWidth',2);  grid on  legend ('x[n]');    subplot(5,1,2);  stem(n+n0, x,'LineWidth', 2);  grid on  legend('x[n-n0]')    subplot(5,1,3);  stem(-n , x,'LineWidth', 2);  grid on  legend('x[-n]')    x1=downsample(x,a);  n1=n(1:a:end);  subplot(5,1,4);  stem(n1,x1,'LineWidth',2);  grid on  legend('x[an])');    x2=upsample(x,b);  i=(length(x2)-1)/2;  n2=-i:i;  subplot(5,1,5);  stem(n2,x2,'LineWidth',2);  grid on  legend('x[bn]');  >> n = -3:3;  >> x = (n>=0);  >> sequence(x,n,1,2,3)  A picture containing diagram  Description automatically generated |

**Post-lab Task**

**Critical Analysis / Conclusion**

|  |
| --- |
| In this lab, we learned to perform operations which includes reversing, shifting, and scaling on both continuous and discrete signals. We also learned how to influence a single function by sending a complete wave signal through it and generating a series of signals. |

|  |  |  |
| --- | --- | --- |
| **Lab Assessment** | | |
| **Pre-Lab** | **/1** | **/10** |
| **In-Lab** | **/5** |
| **Critical Analysis** | **/4** |
| **Instructor Signature and Comments** | | |