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

4.1 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 $y(t)$ is a reflection or a reflected version of $x(t)$ about the interval axis if $y(t) = x(-t)$.

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

Example

Suppose that $x(t) = te^{-t}, -1 \le t \le 3.$ Plot the signal $x(-t)$.

```
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)')
```

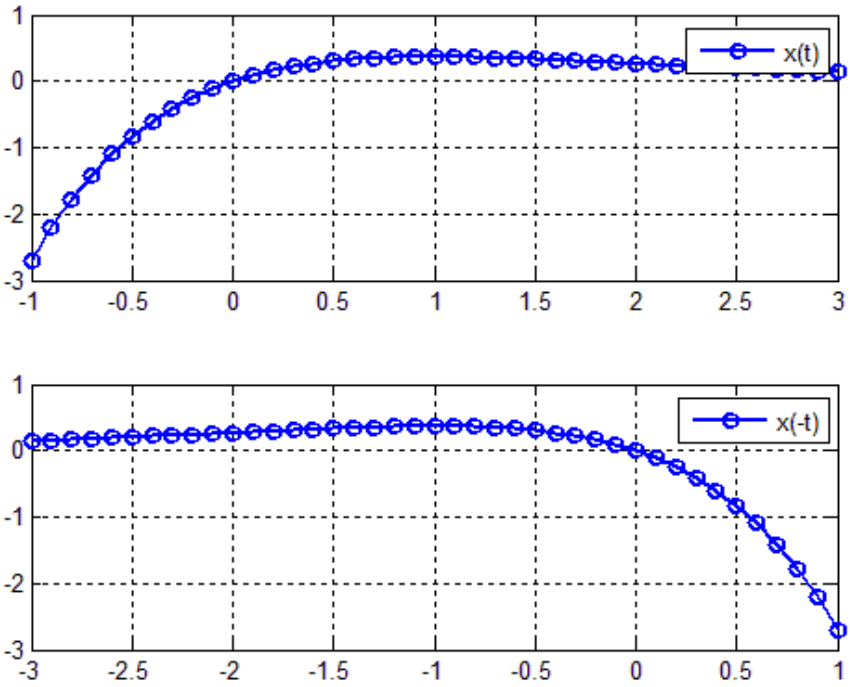


Figure 4.1: Graphs of original $x(t)$ and $x(-t)$

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

4.1.2 Time Scaling:

The second operation discussed is time scaling. A signal $x_1(t)$ is a compressed version of $x(t)$ if $x_1(t) = x(at), a > 1$. The time compression practically means that the time duration of the signal is reduced by a factor of a. On the other hand, a signal $x_2(t)$ is an expanded version of $x(t)$ if $x_2(t) = x(at), 0 < a < 1$. In this case, the time duration of the signal is increased by a factor of $1/a$. In order to plot in MATLAB a time-scaled version of $x(t)$, namely, a signal of the form $y(t) = x(at)$, the statement employed is **plot ((1/a)*t,x).** In contrast to what, someone would expect the vector of time t must be multiplied by $1/a$ and not a.

Example

Consider again the continuous time signal $x(t) = te^{-t}, -1 \le t \le 3$. We will plot the signal $x_1(t) = x(2t)$, which is a time compression of $x(t)$ by a factor of $a = 2$; and the signal $x_2(t) = x(0.5t)$, which is a time expansion of $x(t)$ by a factor $1/a = 2$.

```
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)')
```

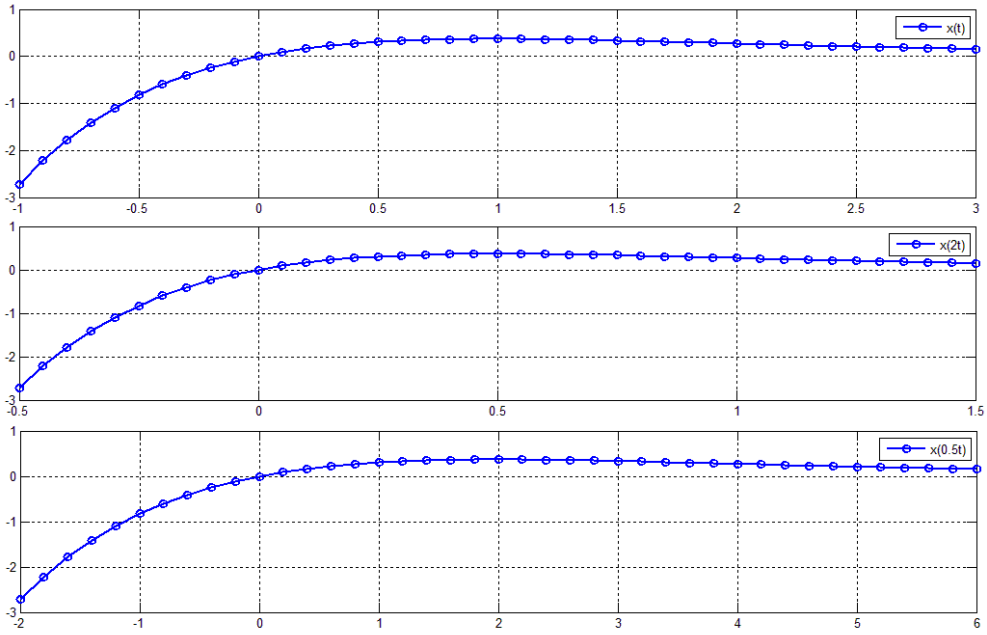


Figure 4.2: Graphs of $x(t), x(2t), x(0.5t)$

4.1.3 Time Shifting:

A third operation performed on time is one of time shifting. A signal $y(t)$ is a time shifted version of $x(t)$ if $y(t) = x(t - t_0)$, where t_0 is the time shift. If $t_0 > 0$, the signal $y(t) = x(t - t_0)$ is shifted by t_0 units to the right (i.e. towards $+\infty$); while if $t_0 < 0$, the signal $y(t) = x(t - t_0)$ is shifted by t_0 units to the left (i.e. towards $-\infty$). The time shifting is implemented in MATLAB in an opposite way to what may be expected. More specifically, in order to plot $x(t - t_0)$ the corresponding MATLAB statement is **plot(t+t0,x).**

Example

The signal $x(t) = te^{-t}, -1 \le t \le 3.$ is again considered. We will plot the signal $x_1(t) = x(t-2)$, that is a shifted version of $x(t)$ by two units to the right (here $t_0 = 2$) and $x_2(t) = x(t+3) = x(t-(-3))$, that is, a shifted version of $x(t)$ by 3 units to the left (here $t_0 = -3$).

```
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)')
```

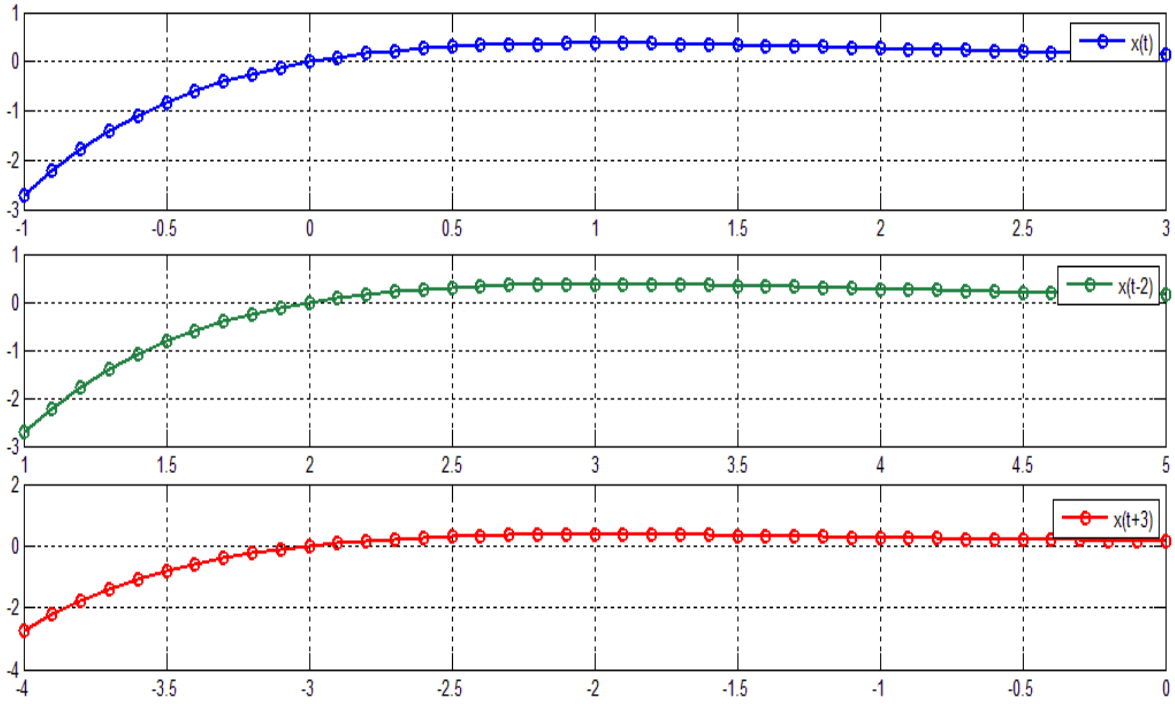


Figure 4.3: Graphs of $x(t), x(t-2), x(t+3)$

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.

4.2 Transformations of the Time Variable for Discrete-Time Signals:

Suppose that $x[n]$ is a discrete time signal defined over the time interval specified by n and n_0 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 $y[n]=x[n-n_0]$ is shifted by n_0 units (or samples) to the right if $n_0>0$, and is shifted by n_0 units to the left if $n_0<0$. 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 $y[n]=x[-n]$, where $y[n]$ is 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 $y[n]=x[an]$. If $a>1$ and $\in \mathbb{Z}$, the time scaling operation is called downsampling or decimation. Notice that a must be an integer as $x[n]$ is defined for fractional values of n. The downsampling operation results in time compression of the signal. Moreover, some samples of $x[n]$ 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 $0<a<1$, the signal $y[n]=x[an]$ is a time expanded version of $x[n]$. In this case, $(1/a)-1$ zeros are inserted between two consecutive samples of $x[n]$. 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 a can take. In case of downsampling a must be a positive integer, and in case of upsampling $1/a$ must be a positive integer.

The downsampling and the upsampling processes are illustrated in the next example for the discrete time signal $x[n]=[1,2,3,4,5,6], 0\le n\le 5$.

Commands	Results	Comments
x=[1 2 3 4 5 6]	x=1 2 3 4 5 6	The discrete time signal $x[n]=[1,2,3,4,5,6], 0\le n\le 5$
a=2; **xds=downsample(x,a)**	xds=1 3 5	The downsampled version of $x[n]$
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 $x[n]$
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 $(1/a)$ .length $(x[n])$

Example

Consider the sequence $x[n]=0.9^n, -10 \le n \le 10.$ Plot the sequences $x[n-10],\ x[n+10],\ x[3n],\ x[n/3],$ and $x[-n]$.

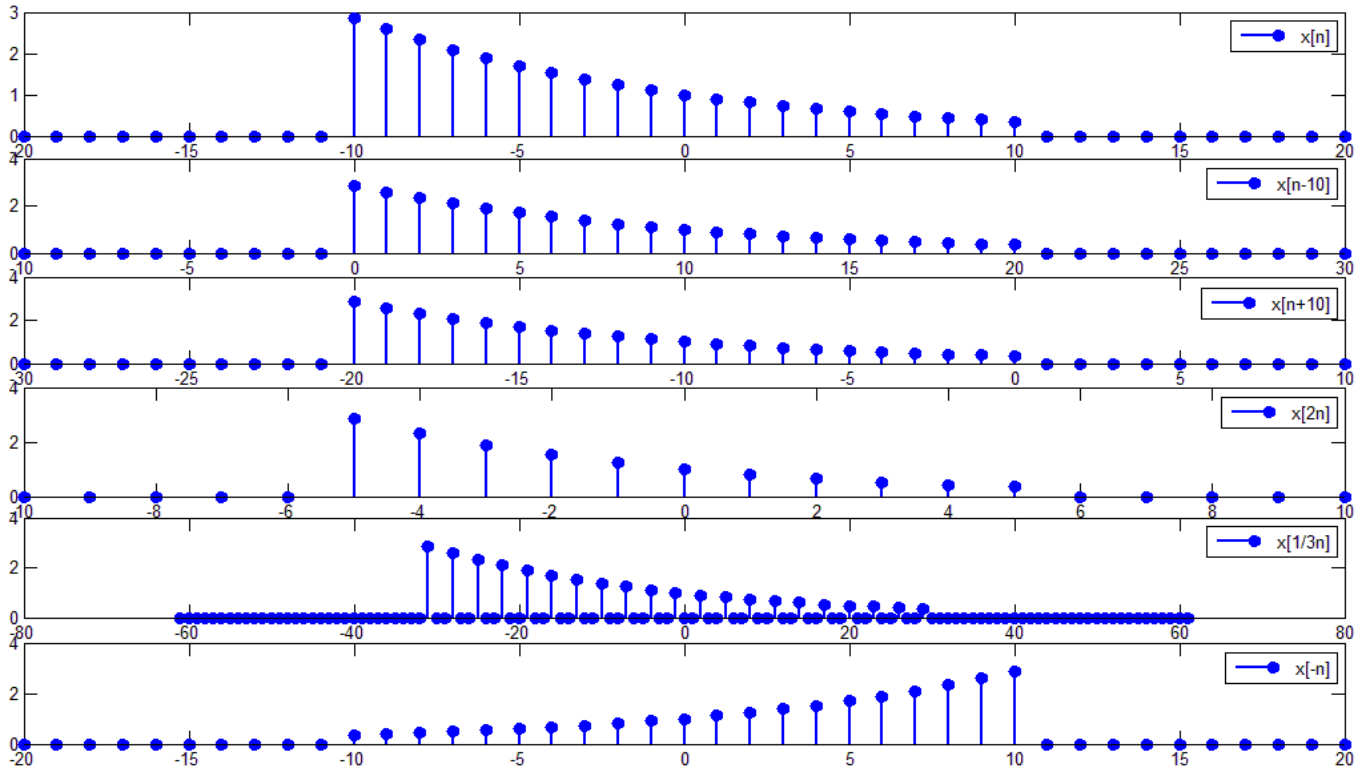


Figure 4.4: Graphs of $x[n], x[n-10], x[n+10], x[3n], x\left[n/3\right]$, and $x[-n]$

An alternative way to obtain the signal $x[-n]$ is by using the command fliplr. This command flips the order of the vector elements. To demonstrate its use, the signal $x[n]=0.9^n, -2 \le n \le 4$ 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]')
```

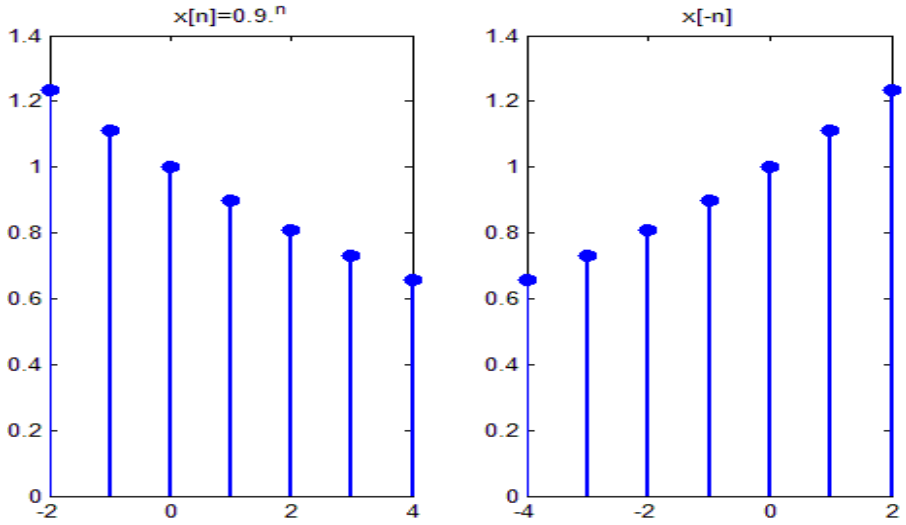
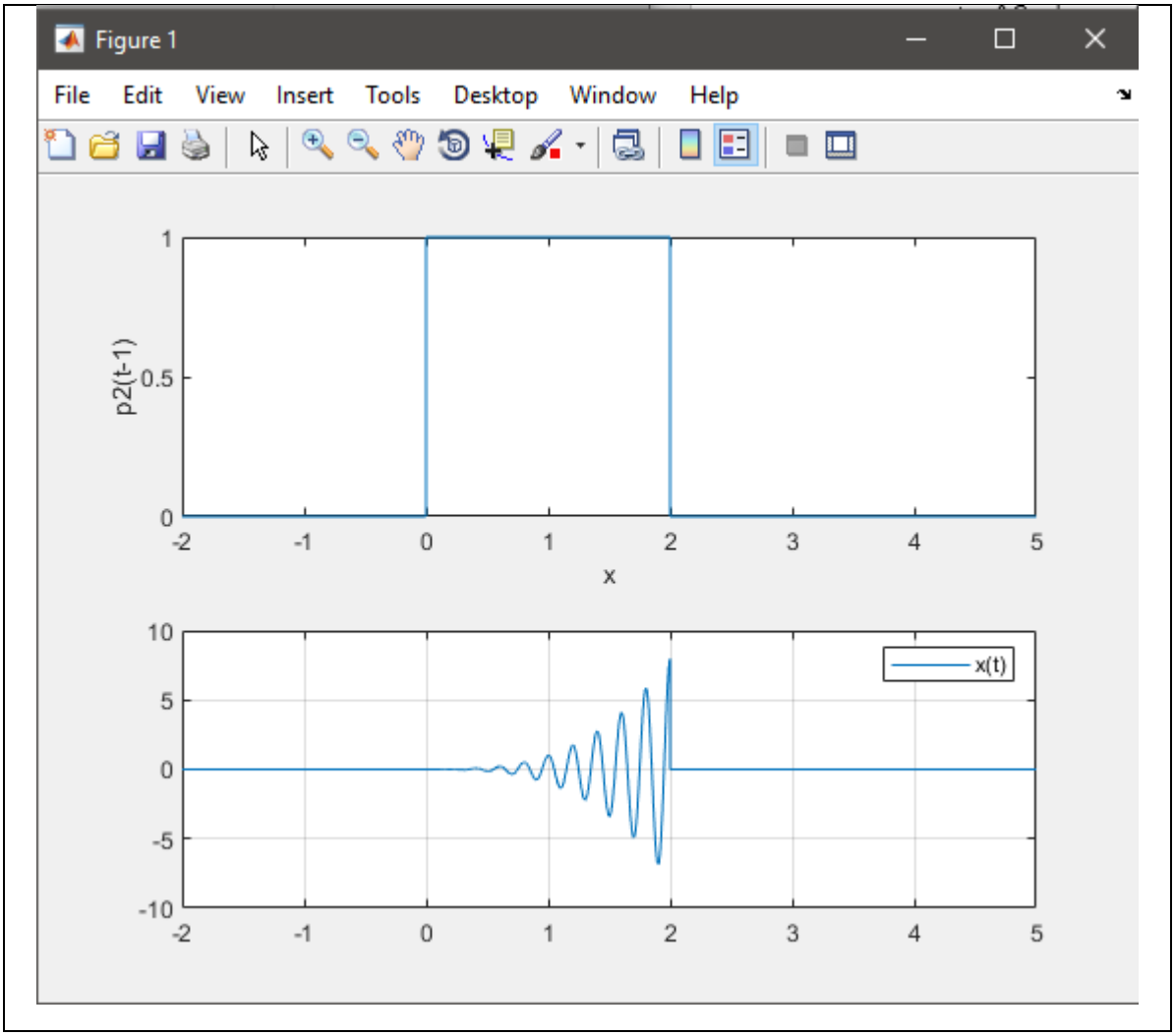


Figure 4.5: Graphs of $x[n]$ and $x[-n]$

In-lab Tasks

Task 01: Plot the signal $x(t) = t^3 \cos(10\pi t)\, p2(t-1)$ **for** $-2 \le 5$**, where** $pT(t)$ **is a rectangular pulse of duration T, denoted by** $p2(t-1) = u(t-1+2/2) - u(t-1-2/2) = u(t) - u(t-2)$**.**

```
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
```



Task 02: Suppose that $x(t) = t\cos(2\pi t), 0 \le t \le 5.$ **Plot the signals**

(a) $x(t)$

(b) $x(-t)$

(c) $x(t/5)$

(d) $x(1+3t)$

(e) $x(-1-3t)$

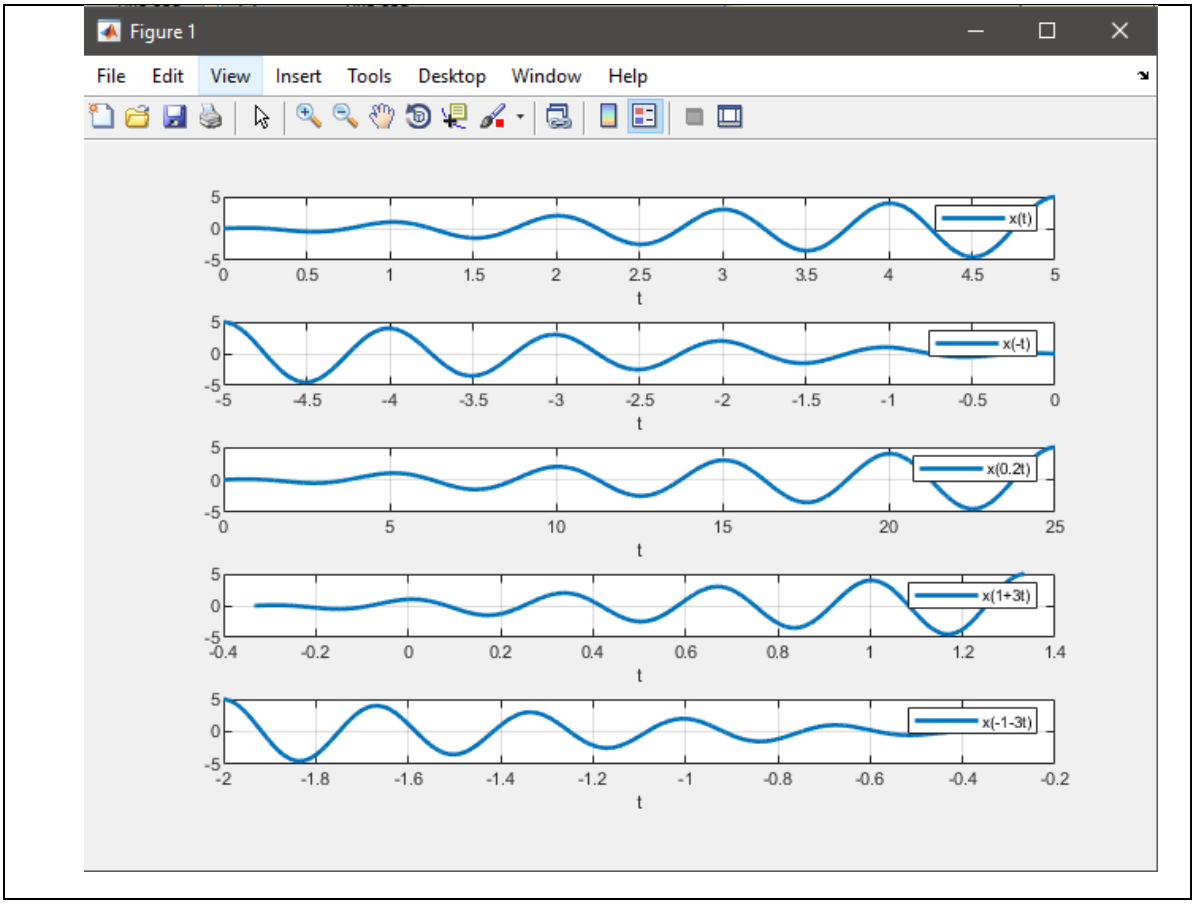
```
t = 0:0.0001:5;
x = t.*cos(2*pi*t);
subplot(5,1,1)
plot(t,x,'LineWidth',2)
grid on
xlabel('t');
legend('x(t)')


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)')
```



Task 03: Suppose that $x(t)=\begin{cases} t & 0\le t\le 2 \\ 4-t & 2<t\le 4 \end{cases}$**. Plot the signals**

(f) $x(t)$
(g) $x(-t)$
(h) $x(t/2)$
(i) $x(2+4t)$
(j) $x(-2-4t)$

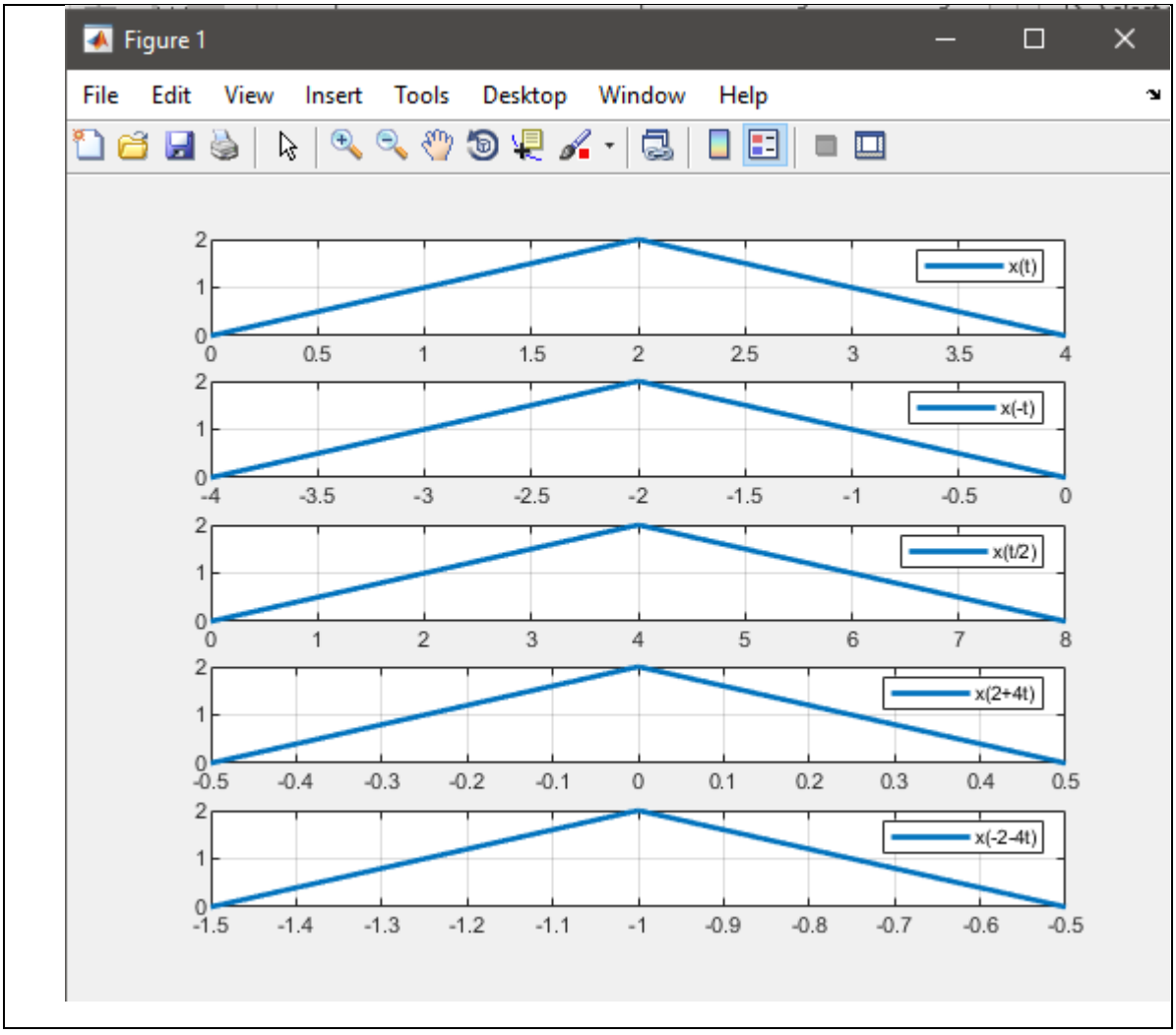
```
t = 0:0.01:4;
x = t.*(t<=2) + (4-t).*(t>2);
subplot(5,1,1)
plot(t,x,'LineWidth',2)
grid on
legend('x(t)')


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)')
```



Task 04: Write a function that accepts a sequence $x[n]$, the discrete time n and a number n_0, a, b as input arguments, and returns the signals $x[n-n_0], x[-n], x[an]$ and $x[bn]$. Where $x[an]$ represents the time compressed version of $x[n]$ and $x[bn]$ is the time expanded version of $x[n]$.

```
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)
```

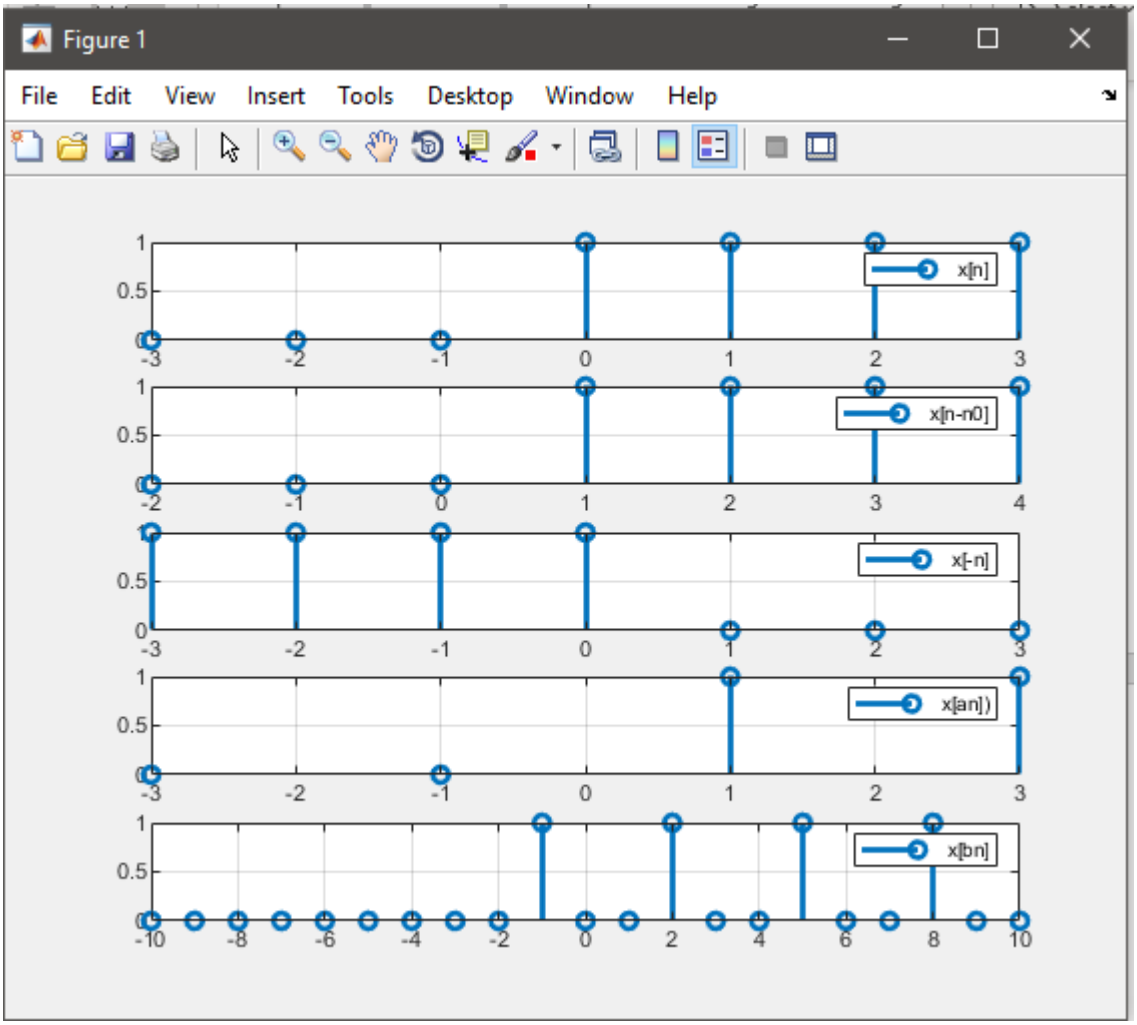


Figure 1
File Edit View Insert Tools Desktop Window Help
x[n]
x[n-n0]
x[-n]
x[an])
x[bn]

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		