

**TRIBHUVAN UNIVERSITY**

**Institute of Engineering**

Central Campus, Pulchowk

**Lab Report On:**

Signal Analysis  
Experiment No. 3

**Submitted To:**

Department of Electronics and Computer Engineering

**Submitted By:**

Rajil Bajracharya (073BEX430)

**Submitted On:**

09-06-019

# **Transformation of Signals**

1. Objectives

* To understand the basic types of signal transformations.
* To apply scaling and shifting transformations to a signal and observe the effect on MATLAB.
* To investigate and analyse the effectiveness of the Fourier-series representation of a signal.
* To understand the Gibbs’ Phenomenon in the Fourier-transform of a square wave/gate function.

1. Theory
   1. Basic Signal Transformations

The most basic types of signal transformations are shifting and scaling. These are discussed below:

* + 1. Shifting Transformation:

Shifting of a signal in the [time] axis involves introducing a phase shift to the function so that the resulting signal is displaced by some amount on the time axis. If be any signal, then shifting this signal by an amount results in the function: The result is that for any value of the signal, say , for the time, , the shifted function will have the same value i.e., at time (i.e., when )

* + 1. Scaling Transformation:

Scaling of a signal in the [time] axis involves shrinking or the expanding of a signal with respect to time. If be any signal, then the signal represents a signal that has been shrunk by times while the signal represents a signal that has been expanded by times along the time-axis.

* 1. Fourier Series Representation

The Fourier Series Representation of any signal was developed by Joseph Fourier. According to him, ‘Any periodic signal can be rewritten as the weighted sum of sines and cosines of different frequencies. Although his theory was not accepted in the mathematical community during his time, it finds widespread usage in modern signal analysis as well as a lot many fields such as image processing, file compression, cryptography, etc. The Fourier series is important because it allows one to model periodic signals as a sum of distinct harmonic components. This allows us to analyse the frequencies present in the harmonics present in the given signal, thereby allowing us to filter/manipulate particular frequency components. Any signal can be represented as the sum of weighted sines and cosines of varying frequencies as follows:

Eqn (2.2.1) is known as the synthesis equation of the Fourier-series while the eqn (2.2.2) is known as the analysis equation.

From eqn (2.2.2), we see that when k = 0, a­­­­k yields a value that is free of the frequency components. Thus, a0 represents the dc component value or simply, the average value of the signal over the given time period.

* 1. Gibbs’ Phenomenon:

The Fourier-integral fails to converge at discontinuities present in the signal. This means that in signals such as the square wave signal, the Fourier series representation does not recover the square wave signal accurately for a finite number of harmonics used. As the number of harmonics used to represent it increases, the weighted sum approximates the square wave to a high degree of accuracy. This phenomenon of distortion of the signal arising at abrupt discontinuities at the peak amplitude and low amplitude while truncating the Fourier-series is known as the Gibbs’ Phenomenon.

1. Code and Outputs:

Fsd = 1;

Fsc = 100;

dtd = 1/Fsd;

dtc = 1/Fsc;

td = -10:dtd:10;

tc = -10:dtc:10;

nd = numel(td);

nc = numel(tc);

yd = zeros(nd);

yc = zeros(nc);

yd((nd + 1)/2) = 1;

yc((nc + 1)/2) = 1;

subplot(2,1,1)

stem(td, yd)

title('Discrete Delta Function')

axis([-10 10 -0.5 1.5])

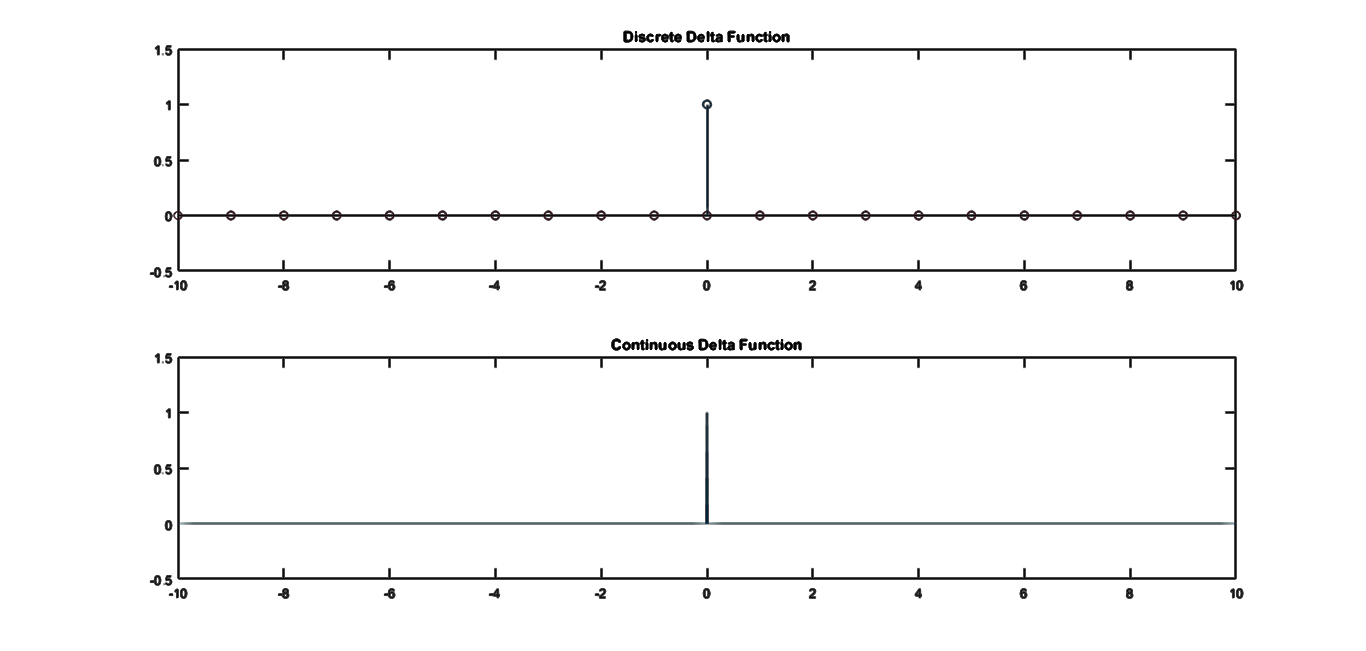
subplot(2,1,2)

plot(tc, yc)

title('Continuous Delta Function')

axis([-10 10 -0.5 1.5])

**OUTPUT:**

****

1.3.2

Fsd = 1;

Fsc = 100;

dtd = 1/Fsd;

dtc = 1/Fsc;

td = -10:dtd:10;

tc = -10:dtc:10;

nd = numel(td);

nc = numel(tc);

yd = zeros(nd);

yc = zeros(nc);

for i = 1:nd

if(td(i) >= 0)

yd(i) = 1;

end

end

for j = 1:nc

if(tc(j) > 0)

yc(j) = 1;

end

end

subplot(2,1,1)

stem(td, yd)

title('Discrete Step Function')

axis([-10 10 -0.5 1.5])

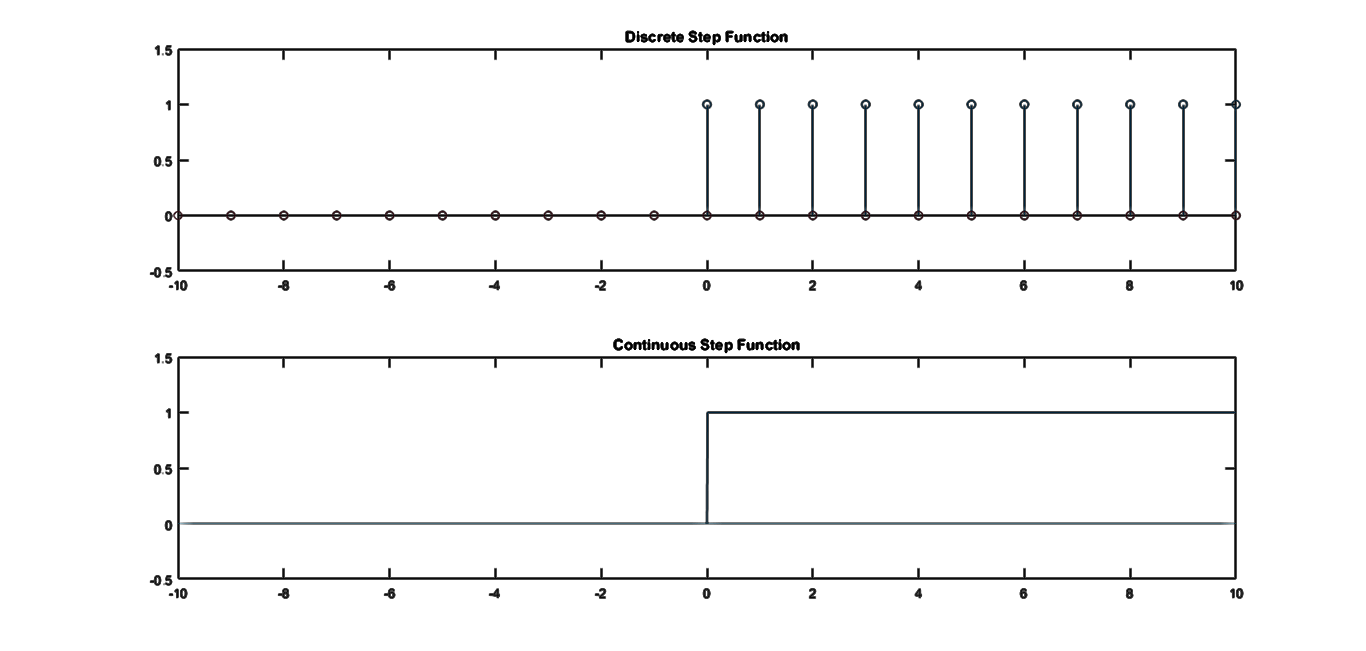
subplot(2,1,2)

plot(tc, yc)

title('Continuous Step Function')

axis([-10 10 -0.5 1.5])

**OUTPUT:**

****

1.3.3

Fsd = 1;

Fsc = 100;

dtd = 1/Fsd;

dtc = 1/Fsc;

k = 1;

td = -10:dtd:10;

tc = -10:dtc:10;

nd = numel(td);

nc = numel(tc);

yd = k \* td;

yc = k \* tc;

subplot(2,1,1)

stem(td, yd)

title('Discrete Ramp Function, k = 1')

axis([-10.5 10.5 -10.5 10.5])

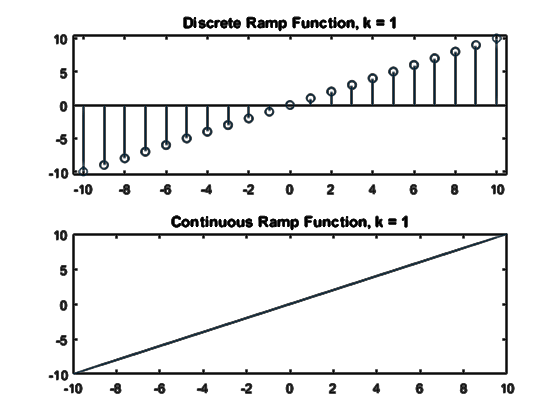
subplot(2,1,2)

plot(tc, yc)

title('Continuous Ramp Function, k = 1')

axis([-10 10 -10 10])

**OUTPUT:**

****

1.3.4

Fsd = 1;

Fsc = 100;

dtd = 1/Fsd;

dtc = 1/Fsc;

td = -10:dtd:10;

tc = -10:dtc:10;

nd = numel(td);

nc = numel(tc);

yd = zeros(nd);

yc = zeros(nc);

for i = 1:nd

if(td(i) >= -2 && td(i) <= 3)

yd(i) = 1;

end

end

for j = 1:nc

if(tc(j) >= -2 && tc(j) <= 3)

yc(j) = 1;

end

end

subplot(2,1,1)

stem(td, yd)

title('Discrete Gate Function')

axis([-10 10 -0.5 1.5])

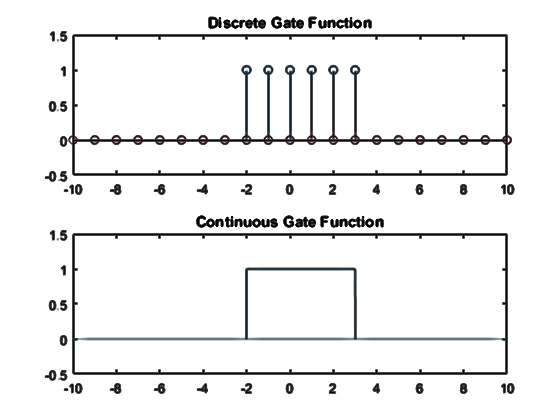
subplot(2,1,2)

plot(tc, yc)

title('Continuous Gate Function')

axis([-10 10 -0.5 1.5])

**OUTPUT:**

****

1.3.5

Fs = 100;

dt = 1/Fs;

t = -30:dt:30;

n = numel(t);

x1 = 2 \* cos(pi \* t/3); %period = 6

x2 = 3 \* cos(2 \* pi \* t / 9); %period = 9

x3 = x1 + x2; %period = -(-14.41) + 4.045

x4 = x1 .\* x2; %period = 18

subplot(4,1,1)

plot(t, x1)

title('x1 = 2 \* cos(pi\*t/3)')

axis([-30 30 -7 7])

subplot(4,1,2)

plot(t, x2)

title('x2 = 3 \* cos(2 \* pi\*t/9)')

axis([-30 30 -7 7])

subplot(4,1,3)

plot(t, x3)

title('x3 = x1 + x2')

axis([-30 30 -7 7])

grid on;

subplot(4,1,4)

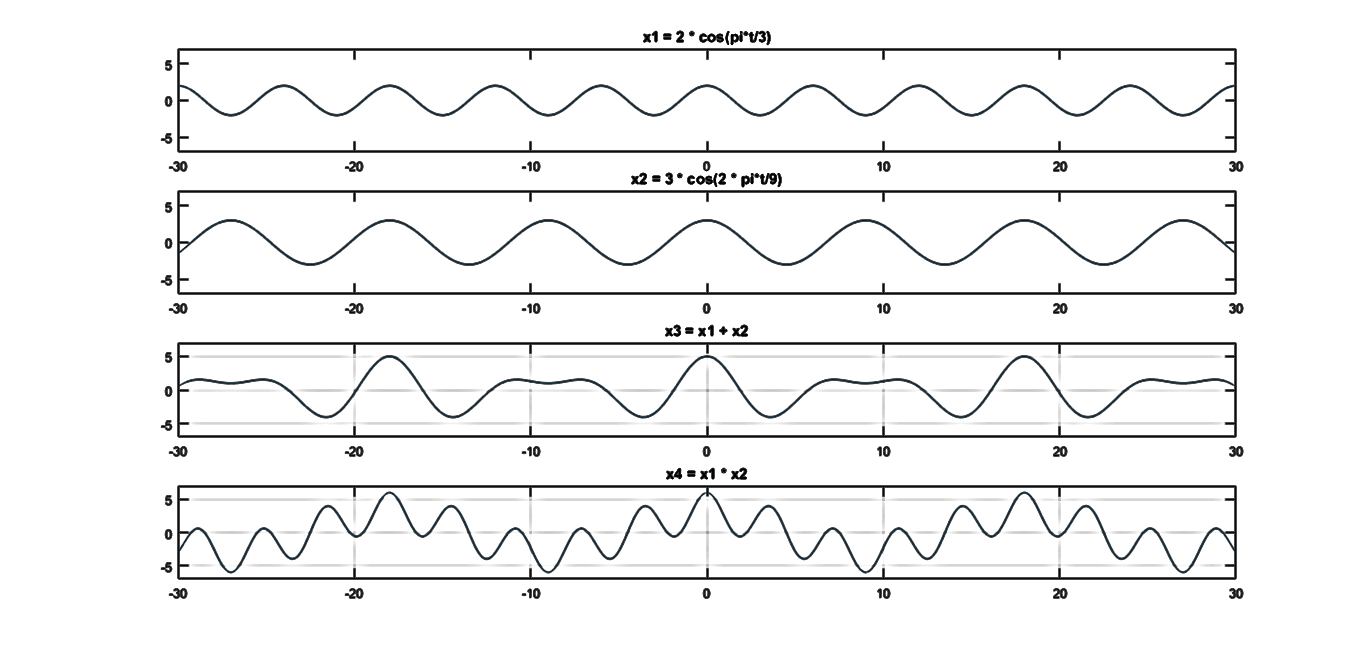
plot(t, x4)

title('x4 = x1 \* x2')

axis([-30 30 -7 7])

grid on;

**OUTPUT:**



1.3.6

Fs = 100;

dt = 1/Fs;

t = -5:dt:5;

c = 1;

a = [0.5 -0.5];

x1 = c \* exp(a(1) \* t);

x2 = c \* exp(a(2) \* t);

subplot(2, 1, 1)

plot(t, real(x1));

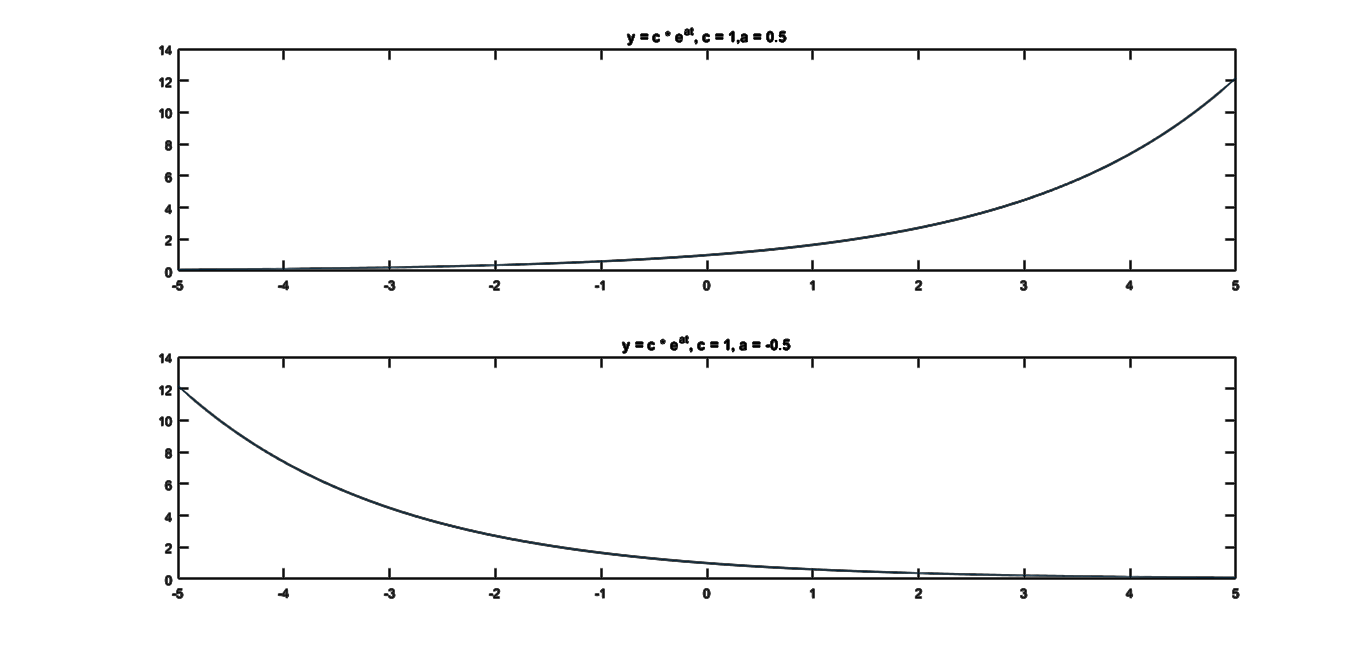
title(‘y=c\*e^{at},c=1,a=0.5')

subplot(2,1,2)

plot(t, real(x2));

title('y=c\*e^{at},c=1,a=-0.5')

**OUTPUT:**

****

1.3.7

Fs = 100;

dt = 1/Fs;

t = -10:dt:10;

c = 15;

a = [complex(0, 0.5) complex(0, pi/3)];

x1 = c \* exp(a(1) \* t);

x2 = c \* exp(a(2) \* t);

subplot(2, 2, 1)

plot(t, real(x1));

title('realy=c\*e^{at},c=15,a=0.5j')

subplot(2, 2, 2)

plot(t, imag(x1));

title('imagy=c\*e^{at},c=15,a=0.5j')

subplot(2,2,3)

plot(t, real(x2));

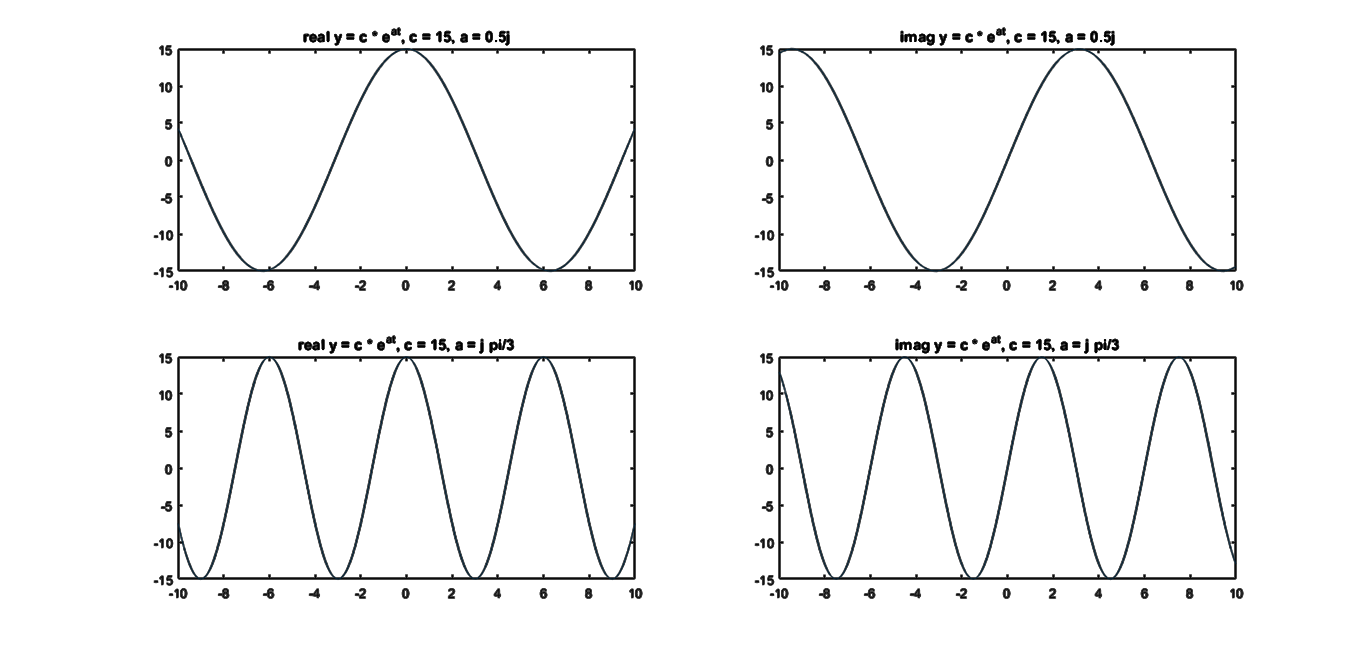
title('realy=c\*e^{at},c=15,a=j pi/3')

subplot(2,2,4);

plot(t, imag(x2));

title('imagy=c\*e^{at},c=15,a=j pi/3')

**OUTPUT:**

****

1. Discussion and Conclusion

In this way, we used MATLAB to plot various types of basic signals such as Dirac-delta function, unit step function, ramp function, harmonic functions, complex exponential signal as well as the functions formed by the addition and multiplication of two out-of-phase harmonic (cosine) signals. The graphical nature of these signals was studied. The period of the composite signals was found through MATLAB itself and a phase difference of was seen between the real and imaginary components of the complex exponential signals. This is due to the fact that a complex exponential signal can be represented by a real cosine part and an imaginary sine part from the Euler’s identity: and these harmonic components have a phase difference. It was also seen that while plotting these signals, the discrete and continuous form of the various signals can be obtained simply by varying the sampling frequency (Fs) to an appropriate value and using the stem() and plot() functions for discrete and continuous signals respectively.