Digital Signal Processing Lab

(CS306)

Submitted by

Name – Naman Thapliyal

Registration no. - 2018105181

Semester - VII

Department - CSE

Course In-Charge:

Mr. Salam Nandu

Ms. Farzana

Ms. Kevisino

Index

S. No.	Experiments	Page No.
1.	Generation of basic signals using Matlab	
2.	MATLAB program for linear convolution	
3.	Matlab code for Crosscorrelation	
4.	Matlab code for Autocorrelation	
5.	Matlab code for Sampling Theorem	
6.	Matlab code for Z-tranform	

Experiment 1

Aim: Generation of basic signals using matlab -

(a) Program for the generation of unit impulse signal

(b)Program for the generation of unit step signal

(c)Program for the generation of unit ramp signal

Description-

(a) Unit Impulse signal

A signal, which satisfies the condition, $\delta(t)=\lim_{\epsilon\to\infty} x(t)$ is known as unit impulse signal. This signal tends to infinity when t = 0 and tends to zero when t ≠ 0 such that the area under its curve is always equals to one. The delta function has zero amplitude everywhere.

(b) Unit Step Signal

A signal, which satisfies the following two conditions −

$U(t)=1(when t\geq 0) and$

$U(t)=0(when t<0)$

is known as a unit step signal. It has the property of showing discontinuity at t = 0. At the point of discontinuity, the signal value is given by the average of signal value. This signal has been taken just before and after the point of discontinuity.

(c) Unit ramp signal

A discrete unit ramp function can be defined as −

$$r(n) = \begin{cases} n, & for \quad n \geq 0 \\ 0, & for \quad n < 0 \end{cases}$$

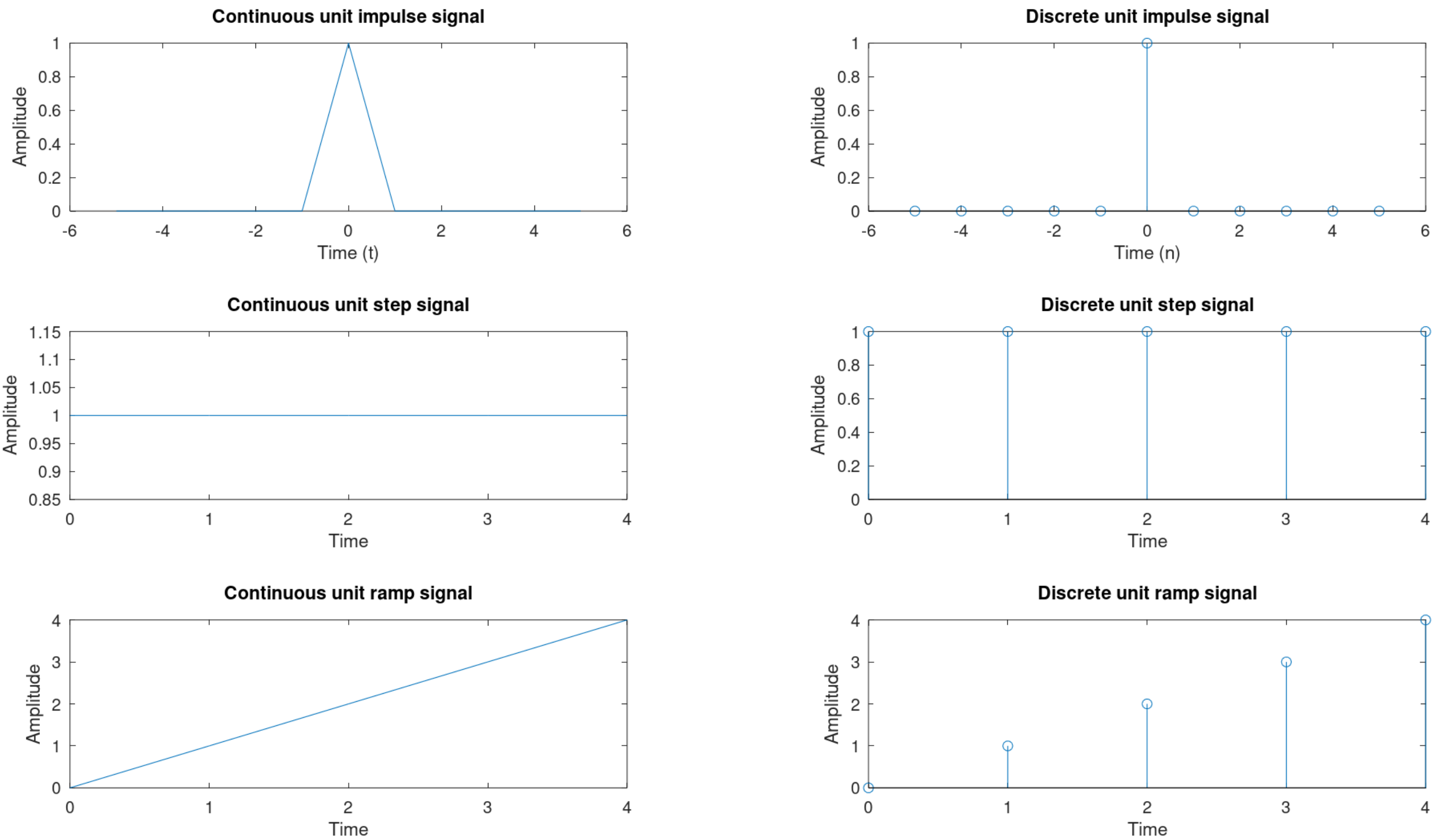
Code-

```
%%% (a)Program for the generation of unit impulse signal

clear all;     %% to clear all the previous data

close all;     %% to close all the dsiplayed figures

clc;           %% to clear all the warning displayed at command window of matlab

N = 5;         %% number of samples at one side of the signal

x = -5:5;      %% Interval of the signal
```

```
y = [zeros(1,N),ones(1,1),zeros(1,N)];  %% Generate Impulse signal

%% Plot for continuous unit impulse signal
subplot(3,2,1);                    %% Plot the subfigure (2 = two rows, 2 = two columns, 1= postion of subfigure)
plot(x,y);                         %% plot continuous signal
ylabel('Amplitude');               %% labeling for y-axis
xlabel('Time (t)');                %% labeling for x-axis
title('Continuous unit impulse signal');%% Giving tilte name of the subfigure
%% Plot for discrete unit impulse signal
subplot(3,2,2);
stem(x,y);                         %% plot discrete signal
ylabel('Amplitude');
xlabel('Time (n)');
title('Discrete unit impulse signal');

%%% (b)Program for the generation of unit step signal
N=5;
t=0:N-1;
y=ones(1,N);
%% Plot for continuous unit step signal
subplot(3,2,3);
plot(t,y);
ylabel('Amplitude');
xlabel('Time');
title('Continuous unit step signal');
%% Plot for discrete unit step signal
subplot(3,2,4);
stem(t,y);
```

```
ylabel('Amplitude');
xlabel('Time');
title('Discrete unit step signal');

%%% (c)Program for the generation of unit ramp signal
N =4;
t=0:N;
%% Plot for continuous unit ramp signal
subplot(3,2,5);
plot(t,t);
ylabel('Amplitude');
xlabel('Time');
title('Continuous unit ramp signal');
%% Plot for discrete unit rampp signal
subplot(3,2,6);
stem(t,t);
ylabel('Amplitude');
xlabel('Time');
title('Discrete unit ramp signal');
```

Output-

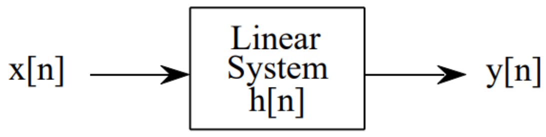


Experiment 2

Aim: MATLAB program for linear convolution

Description:

Convolution is a formal mathematical operation, just as multiplication, addition, and integration. Addition takes two numbers and produces a third number, while convolution takes two signals and produces a third signal. Convolution is used in the mathematics of many fields, such as probability and statistics. In linear systems, convolution is used to describe the relationship between three signals of interest: the input signal, the impulse response, and the output signal.



$x[n] * h[n] = y[n]$

Code:

```
%% MATLAB program for linear convolution

close all;

clear all;

clc;

x = [1,2,3,4]; %% sequence of input x(n) %% this line can

            %% be replaced by x=input('enter the input sequence')

            %% if you want to enter input sequence from command window



h = [1,2,1,2,5]; %% sequence of input h(n) %% impluse response

n1 = length(x);  %% length means number of sample of x(n)

n2 = length(h);

N = n1+n1-1;   %% length of linear convolved sequence or output signal/sequence

x = [x,zeros(1,(N-n1))]; %% append zero at end of x(n) to make same length with h(n)

h = [h,zeros(1,(N-n2))];

y = zeros(1,N)   %% y is the open sequence/signal or convolved signal or system response

for i = 1:N

   for k = 1:i

      y(i) = y(i)+x(k)*h(i-k+1)   %%% here we are using convolution formula

   end

end
```

Output:

Command Window

```
y =

     0     0     0     0     0     0     0

y =

     1     0     0     0     0     0     0

y =

     1     2     0     0     0     0     0

y =

     1     4     0     0     0     0     0

y =

     1     4     1     0     0     0     0

y =

     1     4     5     0     0     0     0

y =

     1     4     8     0     0     0     0

y =

     1     4     8     2     0     0     0

y =

     1     4     8     4     0     0     0

y =

     1     4     8    10     0     0     0

y =

     1     4     8    14     0     0     0

y =

     1     4     8    14     5     0     0

y =

     1     4     8    14     9     0     0

y =

     1     4     8    14    12     0     0

y =
```

Command Window | Documentation | Variable Editor | Editor

Command Window

```
     1     4     8    14    20     0     0

y =

     1     4     8    14    20     0     0

y =

     1     4     8    14    20     0     0

y =

     1     4     8    14    20    10     0

y =

     1     4     8    14    20    16     0

y =

     1     4     8    14    20    20     0

y =

     1     4     8    14    20    20     0

y =

     1     4     8    14    20    20     0

y =

     1     4     8    14    20    20     0

y =

     1     4     8    14    20    20     0

y =

     1     4     8    14    20    20    15

y =

     1     4     8    14    20    20    23

y =

     1     4     8    14    20    20    23

y =

     1     4     8    14    20    20    23

y =

     1     4     8    14    20    20    23
```

Command Window | Documentation | Variable Editor | Editor

Experiment 3

Aim: Matlab code for Crosscorrelation. Crosscorrelation check how much similarlity between two different, signals with time delay in one of the signals x(n) and h(n-3).

Description:

Correlation is a mathematical operation that is very similar to convolution. Just as with convolution, correlation uses two signals to produce a third signal. This third signal is called the cross-correlation of the two input signals.

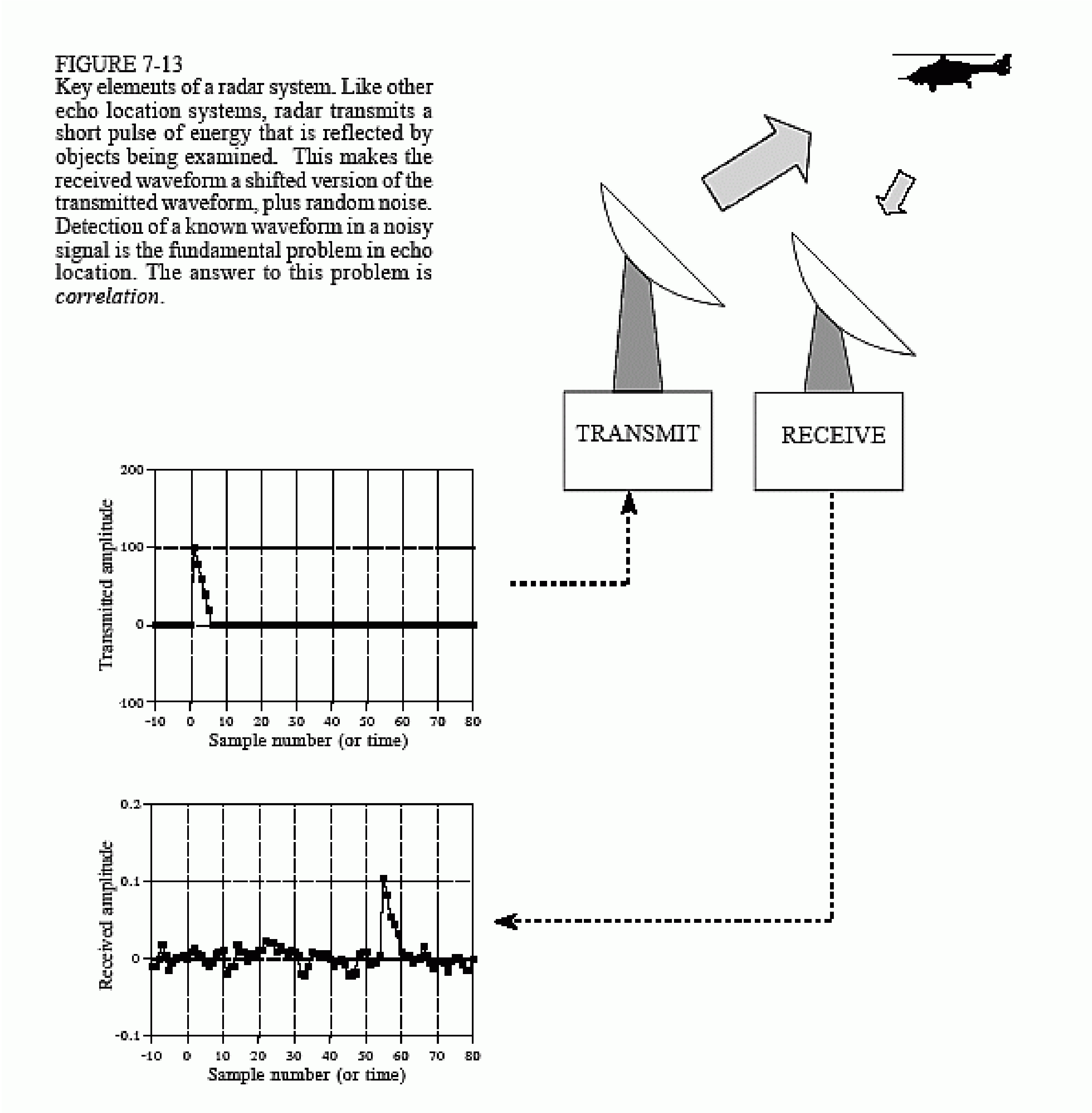


FIGURE 7-13
Key elements of a radar system. Like other echo location systems, radar transmits a short pulse of energy that is reflected by objects being examined. This makes the received waveform a shifted version of the transmitted waveform, plus random noise. Detection of a known waveform in a noisy signal is the fundamental problem in echo location. The answer to this problem is *correlation*.

Code:

```
%%% Matlab code for Crosscorrelation

%%% Crosscorrelation check how much similarlity between two different

%%% signals with time delay in one of the signals  x(n) and h(n-3)

clc;  % clear screen

clear all;  % clear workspace

close all;  % close all figure windows

x = [1,2,3,4];

h = [1,2,5,4];

h= fliplr(h);    %% flipped left to right %% example h = 1,2,5,4 becomes 4,5,2,1

              %% we do fliplr here because fliplr of linear convolution is same as
correlation

              %%% This code is exactly same as linear convolution except

              %%% we fliplr

n1 = length(x);  %% length means number of sample of x(n)

n2 = length(h);

N = n1+n2-1;   %% length of linear convolved sequence or output
signal/sequence


x = [x,zeros(1,(N-n1))]; %% append zero at end of x(n) to make same length
with h(n)

h = [h,zeros(1,(N-n2))];

y = zeros(1,N)   %% y is the open sequence/signal or convolved signal


for i = 1:N

   for k = 1:i

      y(i) = y(i)+x(k)*h(i-k+1)   %%% here we are using convolution formula

   end

end
```

Output:

Command Window

```
y =

     0     0     0     0     0     0     0

y =

     4     0     0     0     0     0     0

y =

     4     5     0     0     0     0     0

y =

     4    13     0     0     0     0     0

y =

     4    13     2     0     0     0     0

y =

     4    13    12     0     0     0     0

y =

     4    13    24     0     0     0     0

y =

     4    13    24     1     0     0     0

y =

     4    13    24     5     0     0     0

y =

     4    13    24    20     0     0     0

y =

     4    13    24    36     0     0     0

y =

     4    13    24    36     0     0     0

y =

     4    13    24    36     2     0     0

y =

     4    13    24    36     8     0     0

y =
```

Command Window | Documentation | Variable Editor | Editor

Command Window

```
y =

     4    13    24    36     0     0     0

y =

     4    13    24    36     2     0     0

y =

     4    13    24    36     8     0     0

y =

     4    13    24    36    28     0     0

y =

     4    13    24    36    28     0     0

y =

     4    13    24    36    28     0     0

y =

     4    13    24    36    28     0     0

y =

     4    13    24    36    28     3     0

y =

     4    13    24    36    28    11     0

y =

     4    13    24    36    28    11     0

y =

     4    13    24    36    28    11     0

y =

     4    13    24    36    28    11     0

y =

     4    13    24    36    28    11     0

y =

     4    13    24    36    28    11     0

y =
```

Command Window | Documentation | Variable Editor | Editor

```
y =

     4    13    24    36    28    11     4

y =

     4    13    24    36    28    11     4

y =

     4    13    24    36    28    11     4

y =

     4    13    24    36    28    11     4

>>
```

Command Window | Documentation | Variable Editor | Editor

Experiment 4

Aim: Matlab code for Autocorrelation. Autocorrelation check how much similarlity in one signal and signal with its time delay- x(n) and x(n-3).

Description:

The cross-correlation of a signal with itself gives its autocorrelation:

$$\boxed{\hat{r}_x(l) \triangleq \frac{1}{N}(x \star x)(l) \triangleq \frac{1}{N}\sum_{n=0}^{N-1} \overline{x(n)}x(n+l)}$$

The autocorrelation function is Hermitian:

$$\hat{r}_x(-l) = \overline{\hat{r}_x(l)}$$

When x is real, its autocorrelation is real and even (symmetric about lag zero).

Code:

```
%%% Matlab code for Autocorrelation

%%% Autocorrelation check how much similarlity in one signal

%%% signal with its time delay     %%  x(n) and x(n-3)


clc;  % clear screen

clear all;  % clear workspace

close all;  % close all figure window

x = [4, 7, 8, 10]; %% this code is exactly same as crosscorrelation except here
we use only one input

x= fliplr(x);    %% flipped left to right %% example h = 1,2,5,4 becomes 4,5,2,1


               %% we do fliplr here because fliplr of linear convolution is same as
correlation

               %%% This code is exactly same as linear convolution except

               %%% we fliplr

n1 = length(x);  %% length means number of sample of x(n)
```

```
N = 2*n1-1;   %% length of linear convolved sequence or output signal/sequence

x = [x,zeros(1,(N-n1))]; %% append zero at end of x(n) to make same length with h(n)

y = zeros(1,N)   %% y is the open sequence/signal or convolved signal

for i = 1:N

   for k = 1:i

      y(i) = y(i)+x(k)*x(i-k+1)   %%% here we are using convolution formula

   end

end
```

Output:

```
Command Window
Command Window
y =
   0   0   0   0   0   0   0
y =
   100     0     0     0     0     0     0
y =
   100    80     0     0     0     0     0
y =
   100   160     0     0     0     0     0
y =
   100   160    70     0     0     0     0
y =
   100   160   134     0     0     0     0
y =
   100   160   204     0     0     0     0
y =
   100   160   204    40     0     0     0
y =
   100   160   204    96     0     0     0
y =
   100   160   204   152     0     0     0
y =
   100   160   204   192     0     0     0
y =
   100   160   204   192     0     0     0
y =
   100   160   204   192    32     0     0
y =
   100   160   204   192    81     0     0
y =
   100   160   204   192   113     0     0
y =
   100   160   204   192   113     0     0
```

Command Window

Command Window

```
y =

   100   160   204   192   113     0     0

y =

   100   160   204   192   113     0     0

y =

   100   160   204   192   113     0     0

y =

   100   160   204   192   113     0     0

y =

   100   160   204   192   113    28     0

y =

   100   160   204   192   113    56     0

y =

   100   160   204   192   113    56     0

y =

   100   160   204   192   113    56     0

y =

   100   160   204   192   113    56     0

y =

   100   160   204   192   113    56     0

y =

   100   160   204   192   113    56     0

y =

   100   160   204   192   113    56    16

y =

   100   160   204   192   113    56    16

y =

   100   160   204   192   113    56    16

y =

   100   160   204   192   113    56    16

>> 
```

Experiment 5

Aim: Matlab code for Sampling Theorem.

Description:

Sampling theorem states that “continues form of a time-variant signal can be represented in the discrete form of a signal with help of samples and the sampled (discrete) signal can be recovered to original form when the sampling signal frequency Fs having the greater frequency value than or equal to the input signal frequency Fm.

$$Fs \geq 2Fm$$

If the sampling frequency (Fs) equals twice the input signal frequency (Fm), then such a condition is called the Nyquist Criteria for sampling. When sampling frequency equals twice the input signal frequency is known as “Nyquist rate”.

$$Fs=2Fm$$

If the sampling frequency (Fs) is less than twice the input signal frequency, such criteria called an Aliasing effect.

$$Fs<2Fm$$

So, there are three conditions that are possible from the sampling frequency criteria. They are sampling, Nyquist and aliasing states.

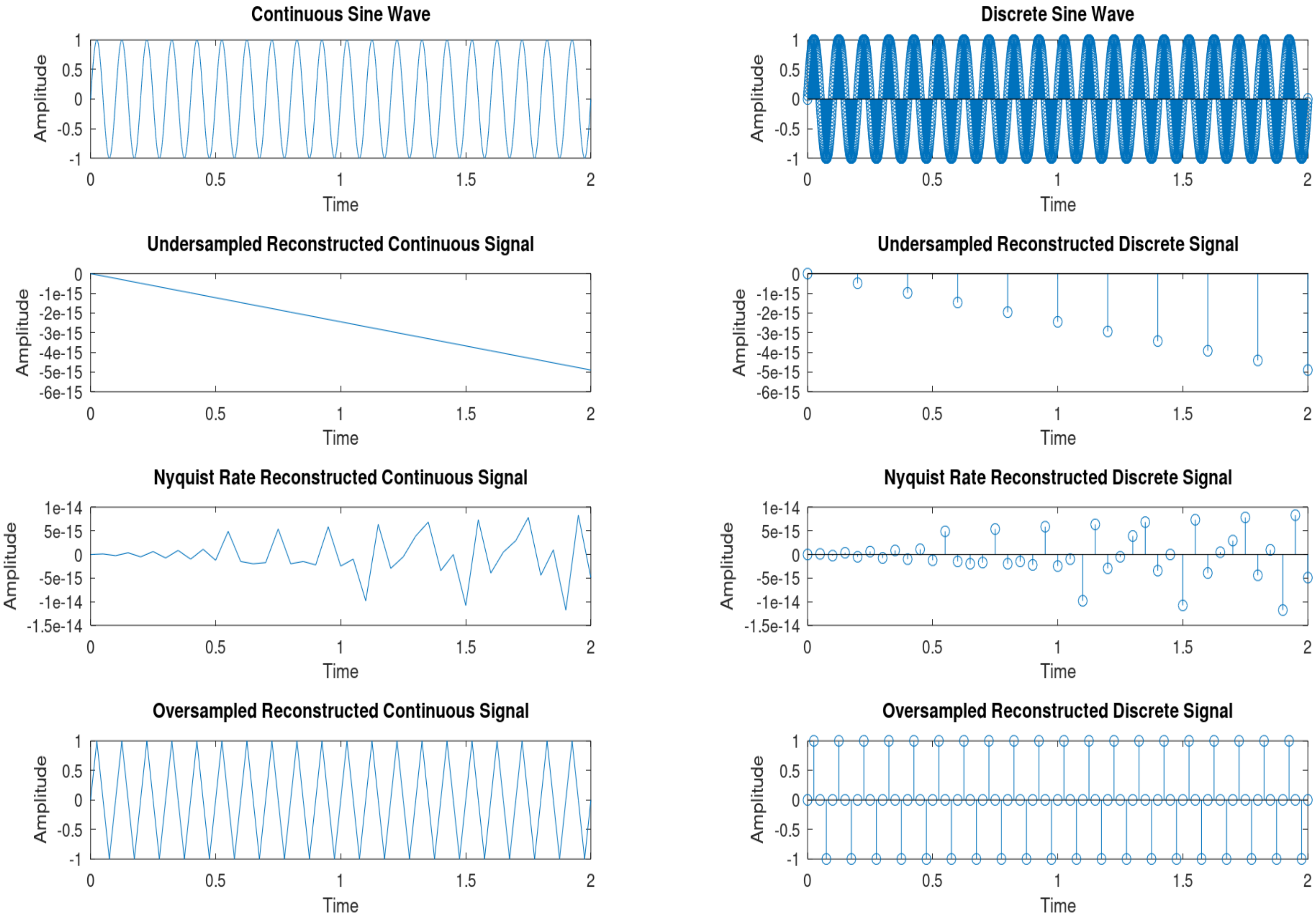
Code:

```
%% exp.5 Matlab code for Sampling Theorem

clc;

close all;

clear all;

fm = 10; %   max frequency   %%%   fm = input('Enter frequency of message signal: ');

fs1 =  5; %%    fs = sampling frequency fs1 < 2fm  %%%%  fs1 = input('Enter frequency of sampling frequency(<2fm): ');

fs2 = 20; %%%    fs2 = 2fm    %%%%   fs2 =input('Enter frequency of sampling frequency(=2fm): ');

fs3 = 40;   %%%%  fs3 > 2fm    %%%  fs3 = input('Enter frequency of sampling frequency(>2fm): ');
```

```
%% Plot for continuous signal or message signal or input signal
t = 0:0.001:2;
x = sin(2*pi*fm*t);
subplot(4,2,1);
plot(t,x);
xlabel('Time');
ylabel('Amplitude');
title('Continuous Sine Wave');
subplot(4,2,2);
stem(t,x);
xlabel('Time');
ylabel('Amplitude');
title('Discrete Sine Wave');

%%% Undersampled Reconstructed Signal %%% fs1 < 2fm
t1 = 0:1/fs1:2;
x1 = sin(2*pi*fm*t1);
subplot(4,2,3);
plot(t1,x1);
xlabel('Time');
ylabel('Amplitude');
title('Undersampled Reconstructed Continuous Signal');
subplot(4,2,4);
stem(t1,x1);
xlabel('Time');
ylabel('Amplitude');
title('Undersampled Reconstructed Discrete Signal');
```

```
%%% Nyquist rate Signal %%% fs2=2fm

t2 = 0:1/fs2:2;

x2 = sin(2*pi*fm*t2);

subplot(4,2,5);

plot(t2,x2);

xlabel('Time');

ylabel('Amplitude');

title('Nyquist Rate Reconstructed Continuous Signal');

subplot(4,2,6);

stem(t2,x2);

xlabel('Time');

ylabel('Amplitude');

title('Nyquist Rate Reconstructed Discrete Signal');


%%% Oversampled Reconstructed Signal

t3 = 0:1/fs3:2;

x3 = sin(2*pi*fm*t3);

subplot(4,2,7);

plot(t3,x3);

xlabel('Time');

ylabel('Amplitude');

title('Oversampled Reconstructed Continuous Signal');

subplot(4,2,8);

stem(t3,x3);

xlabel('Time');

ylabel('Amplitude');

title('Oversampled Reconstructed Discrete Signal');
```

Output:



Experiment 6

Aim: Matlab code for Z-tranform

Description:

The Z-transform of a sequence x[n]x[n] is given by

$$X(z) = \sum_{n=\infty}^{\infty} x[n]z^{-n}$$

For the Z-transform to be meaningful the infinite summation has to converge to a finite value. Convergence is not guaranteed for all values of zz. In fact the *region of convergence* (ROC) defines all values for which the Z-transform converges (we say the Z-transform exists for those values of zz). Determining the ROC is not a simple task. In these lecture notes we will most often only state the ROC without proof or use simple rules for systems/signals for which the ROC is known.
The Z-transform is such an important transform in practice because:

- Discrete convolution in the time domain corresponds with multiplication in the zz-domain. The Z-transform shares this property with the Fourier transform.
- Any difference equation relating input and output of an LTI system is turned into an algebraic equation in zz and the Z-transform of both input and output signal.

Especcially the last property is important in digital signal processing. Difference equations are the important characterizations of digital linear filters.

The above definition of the Z-transform is the *bilateral* Z-transform. It integrates from −∞−∞ to +∞+∞. The *unilateral* Z-transform integrates from 00 to +∞+∞. In signal processing this is a common definition for the Z-transform. In signal processing we are most often interested in causal systems and signals and for such systems both definitions of the Z-transform coincicde.
The inverse Z-transform is a mathematically complex transform. It is given as a line integral:

$$x[n] = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz$$

where CC is a closed path containing the origin and entirely in the ROC. Don’t be too much bothered in case you find it hard to see what is going on here. In (digital) signal processing we mostly use Z-transform pairs x[n]⟷X(z)x[n]⟷X(z) of simple functions without too much use of the definition of the inverse transform.

Code:

```
%%% Matlab code for Z-tranform

clc;
```

```
close all;

clear all;

x = [6 7 8 9] %input('Enter the input sequence: ');

b = 0;

n = length(x);

y = sym('z');

for i = 1:n

   b = b+x(i)*y^(1-i); %% z-transform formula

end

display(x)

display('z Transform of the input sequence: ');

display(sum(b))
```

Output:

Command Window

```
Command Window
x =

    6     7     8     9

z Transform of the input sequence:
(sym)

        7     8     9
  6 +  -  +  --  +  --
        z      2      3
              z      z

>> |
```

-----------------------------------The End--