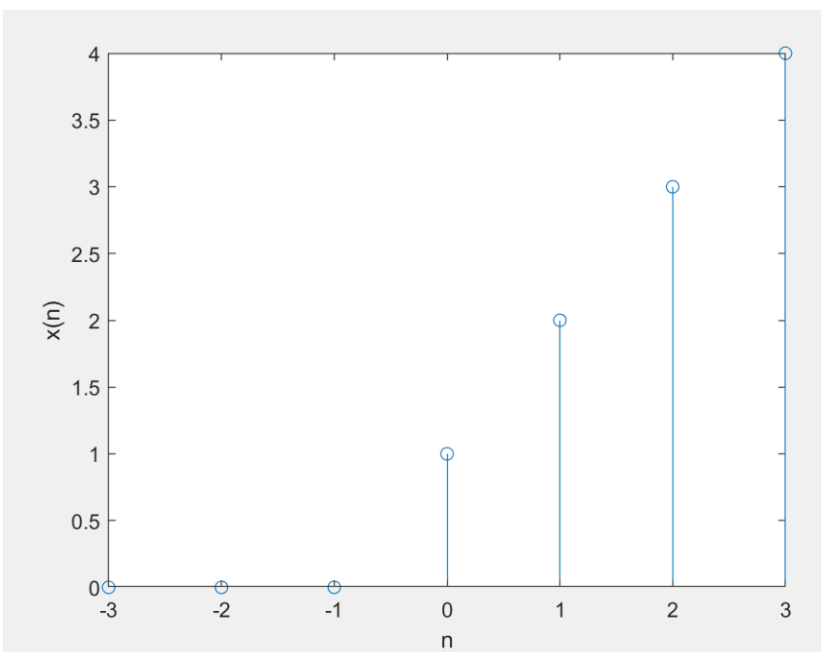
Part A

Exercise A1:

Code:

```
clc;
clearvars;


%generating unit ramp sequence

[n1 n2] = deal(-3, 3);
n = n1:n2;
n0 = -1; %lag = -1
x = (n-n0).*((n-n0)>=0);
stem(n,x); xlabel('n'); ylabel('x(n)');
```

Output:

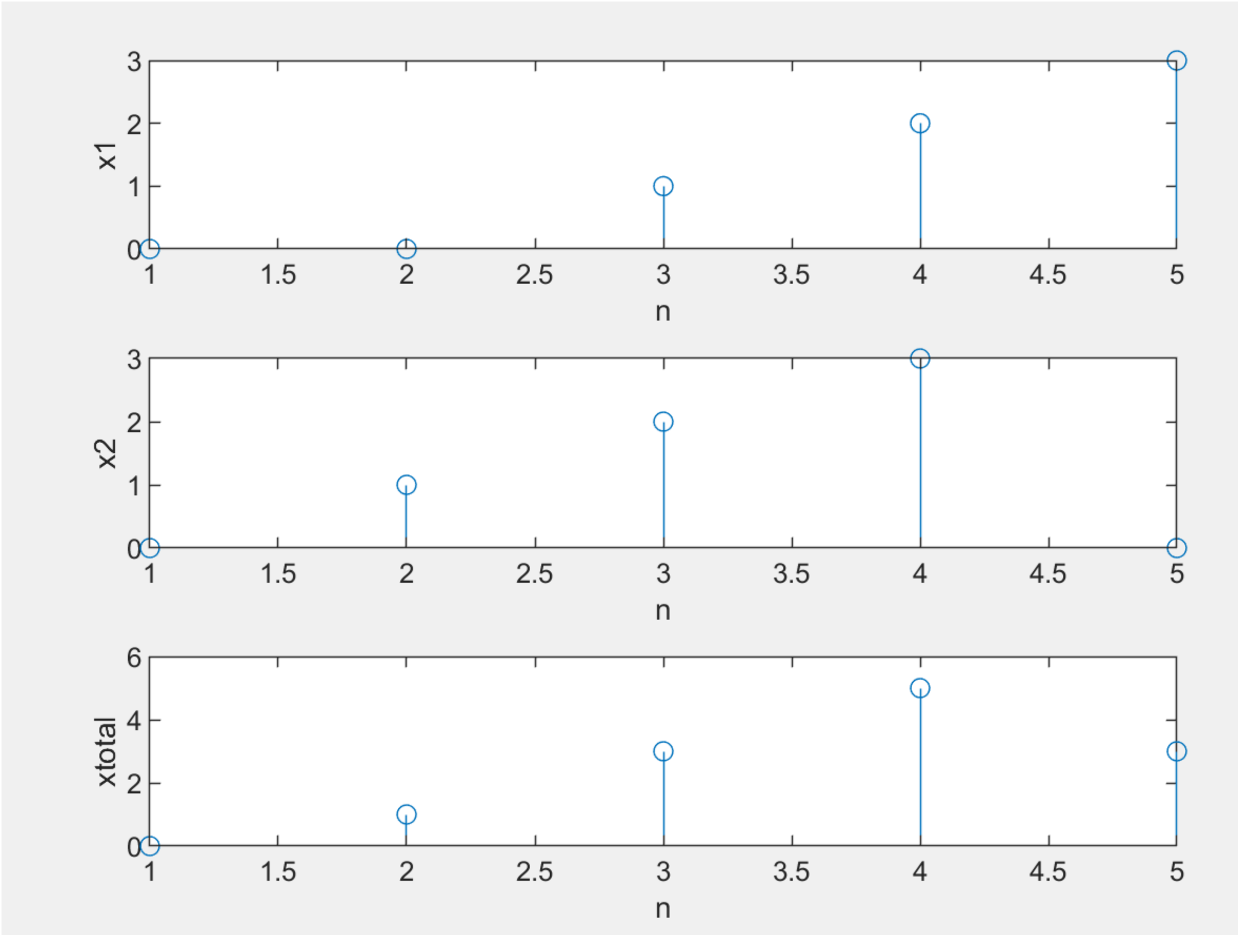


Exercise A2:

Code:

```
clc;
clearvars;

x1 = [ 0 1 2 3];
x2 = [ 0 1 2 3] ;

n1ref = 0; n2ref = 1;


    diffleftx2 = n2ref-1;
    diffleftx1 = n1ref-1;

    diffrightx1 = length(x1)-n1ref;
    diffrightx2 = length(x2)-n2ref;

    if diffleftx2>diffleftx1
        addleft= zeros(1,(diffleftx2-diffleftx1));
        x1 = [addleft x1];
    else
        addleft= zeros(1,(diffleftx2-diffleftx1));
        x2 = [addleft x2];

    end

    if diffrightx2>diffrightx1
        addright = zeros(1, (diffrightx2-diffrightx1));
        x1 = [x1 addright];
    else
         addright = zeros(1, (diffrightx1-diffrightx2));
         x2 = [x2 addright];
    end


 xtotal = x1 + x2;
 subplot(3,1,1)
 stem(x1); xlabel('n'); ylabel('x1');
 subplot(3,1,2);
 stem(x2); xlabel('n'); ylabel('x2');
 subplot(3,1,3);
 stem(xtotal); xlabel('n'); ylabel('xtotal');
```

Output:

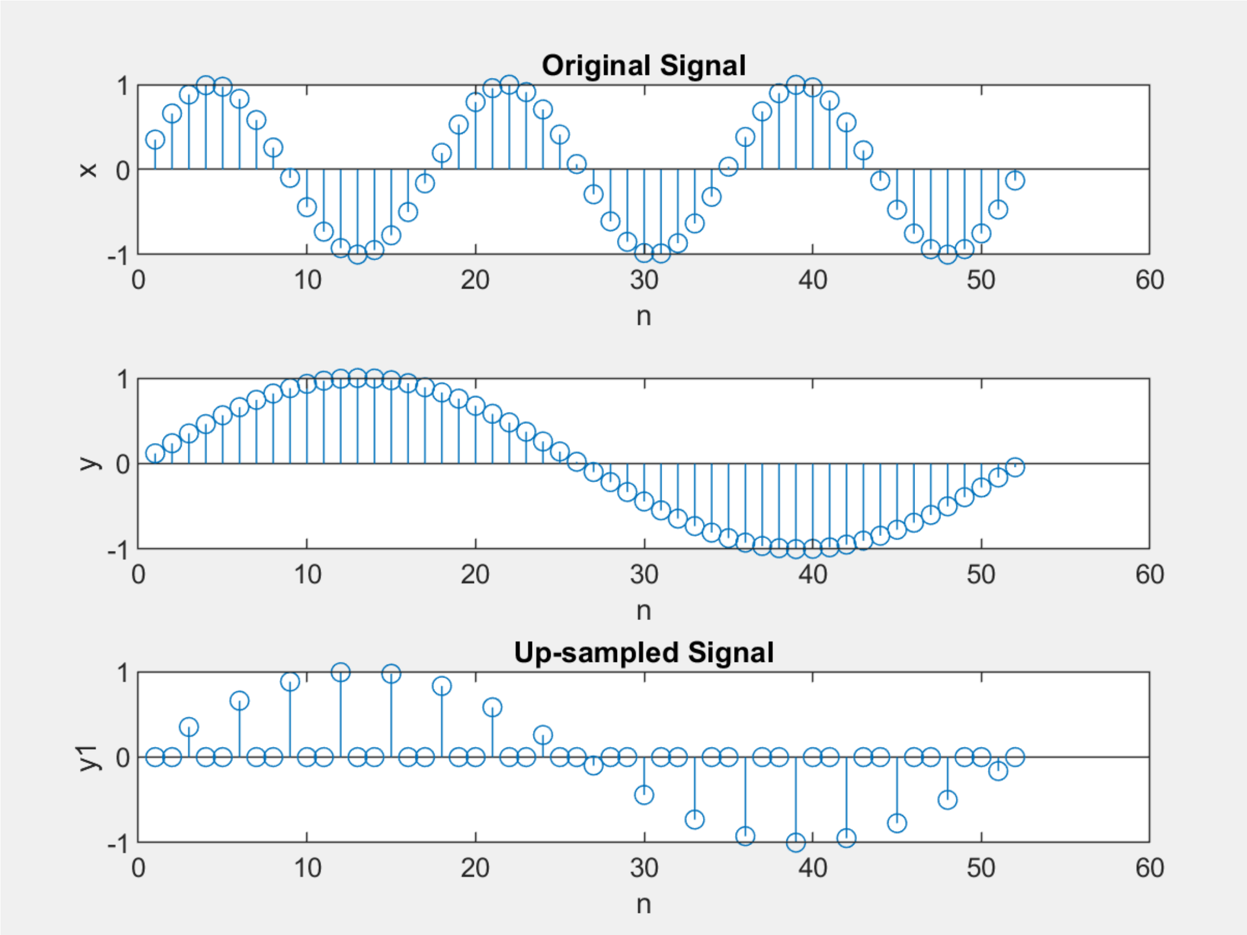


Exercise A3:

Code:

```
clc;
clearvars;


n = 1: 52;
L = 3;
x = sin(0.36*n);
y = sin(0.36*n/L);
y1 = y;
for i= 1:52
    if mod(i, L) ~= 0
        y1(i)= 0;
    end
end

subplot(3,1,1);
stem (n,x); xlabel('n'); ylabel('x');title('Original Signal');
%original signal
subplot(3,1,2);
stem(n,y); xlabel('n'); ylabel('y');%not up sampled yet
subplot(3,1,3);
stem(n,y1); xlabel('n'); ylabel('y1'); title('Up-sampled
Signal');%up sampled signal
```

Output:

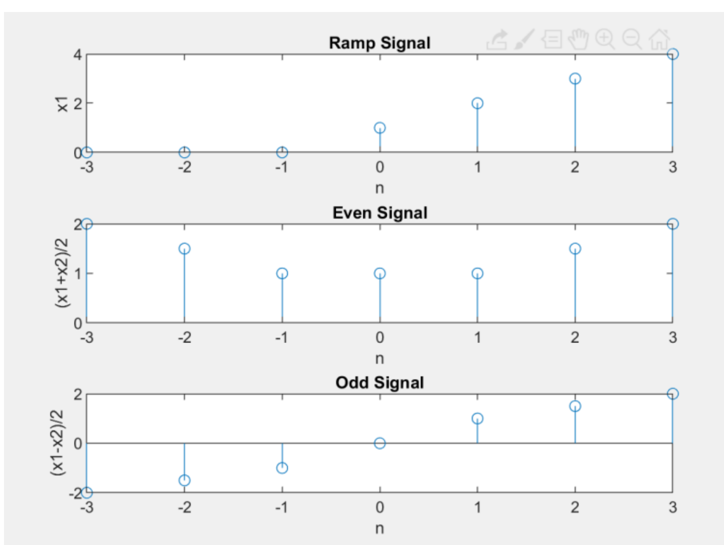


Exercise A4:

Code:

```
clc;
clearvars;

[n1 n2] = deal(-3, 3);
n = n1:n2;
n0 = -1;
x1 = (n-n0).*((n-n0)>=0);


x2= fliplr(x1);

subplot(3,1,1)
stem(n,x1); xlabel('n'); ylabel('x1'); title('Ramp
Signal');%ramp signal
subplot(3,1,2)
stem(n, (x1+x2)/2);  xlabel('n'); ylabel('(x1+x2)/2');
title('Even Signal'); %even part
subplot(3,1,3)
stem(n, (x1-x2)/2); xlabel('n'); ylabel('(x1-x2)/2'); title('Odd
Signal'); %odd part
```

Output:



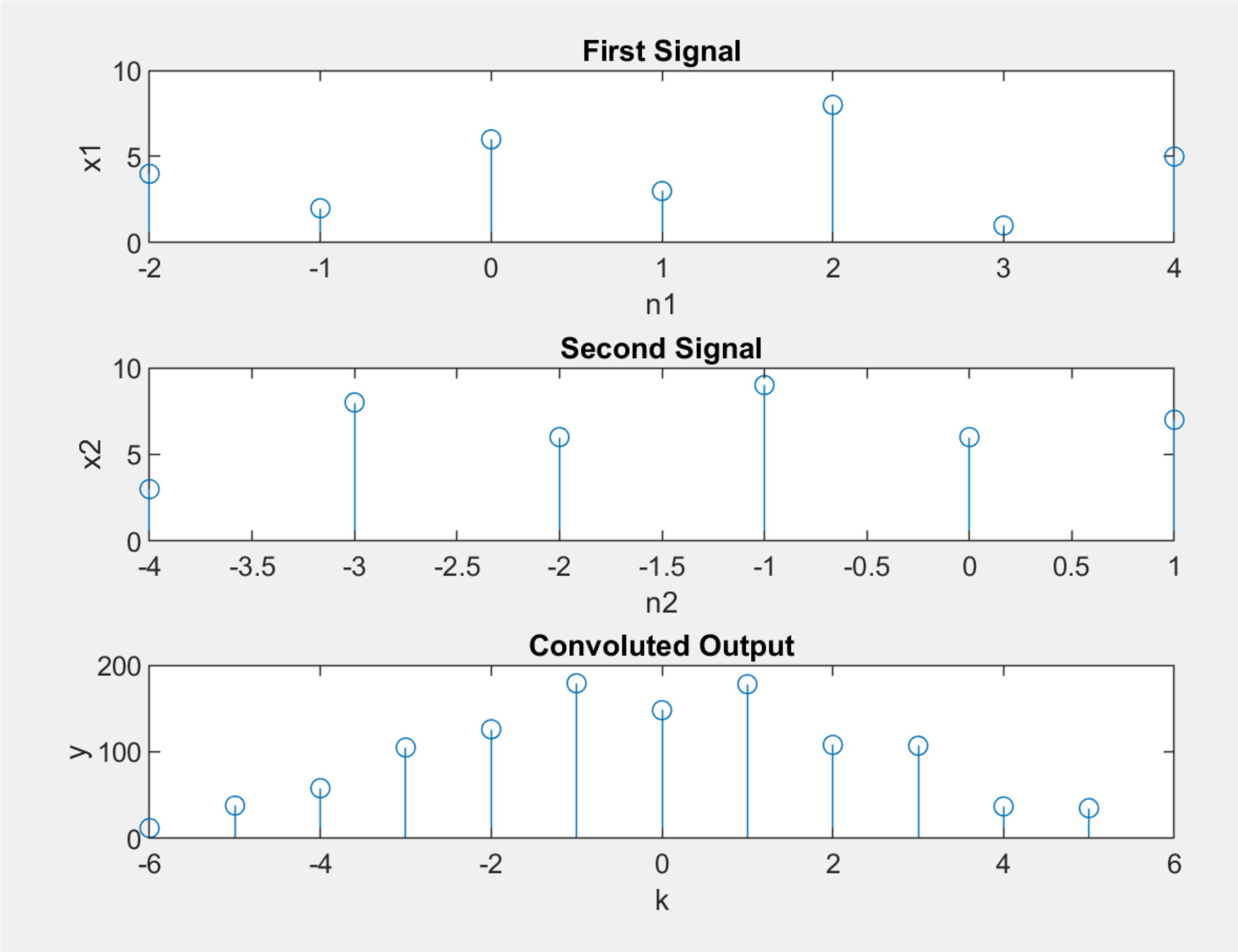
Part B

Example B1:

Code:

```
clc;
clearvars;


x1 = [ 4 2 6 3 8 1 5];
x2 = [3 8 6 9 6 7];

n1 = -2:4;
n2 = -4:1;

kmin = n1(1)+n2(1);
kmax = n1(end)+n2(end);
y = conv(x1,x2);

k = kmin:kmax; %conv index generated

subplot(3,1,1);
stem (n1,x1); xlabel('n1'); ylabel('x1'); title('First Signal');
subplot(3,1,2);
stem(n2,x2); xlabel('n2'); ylabel('x2'); title('Second Signal');
subplot(3,1,3);
stem(k,y); xlabel('k'); ylabel('y'); title('Convoluted Output');
fig= gcf;
WinOnTop(fig, true);
```

Output:

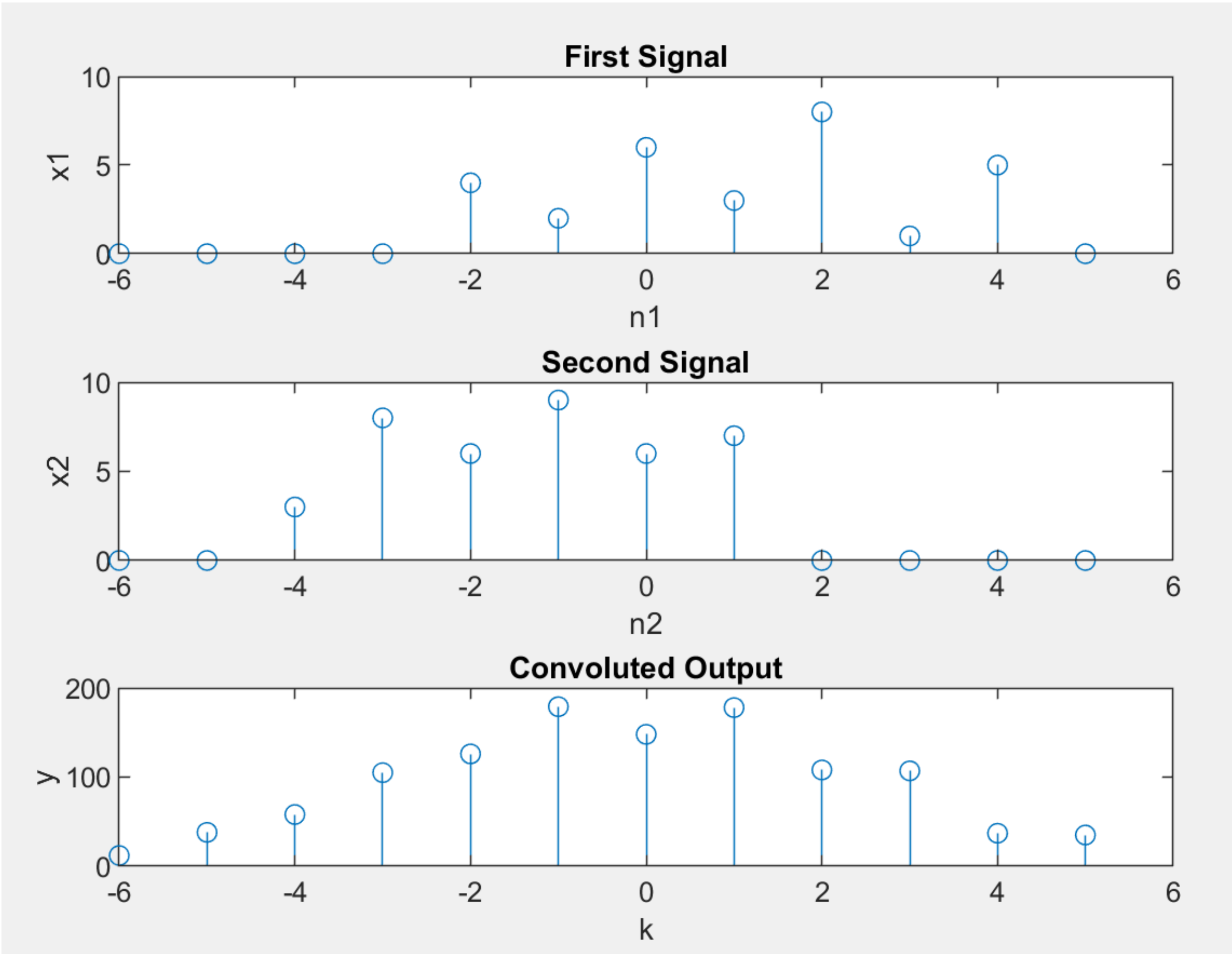

First Signal
x1
10
5
0
-2
-1
0
1
2
3
4
n1
Second Signal
x2
10
5
0
-4
-3.5
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
n2
Convoluted Output
y
200
100
0
-6
-4
-2
0
2
4
6
k

Exercise B1:

Code:

```
clc;
clearvars;


x1 = [ 4 2 6 3 8 1 5];
x2 = [3 8 6 9 6 7];

n1 = -2:4;
n2 = -4:1;


kmin = n1(1)+n2(1);
kmax = n1(end)+n2(end);
y = conv(x1,x2);

k = kmin:kmax;
%scaling operation
y1 = [ zeros(1, (abs(k(1)-n1(1)))) x1 zeros(1, abs(k(end)-
n1(end)))];
y2 = [ zeros(1, abs(k(1)-n2(1))) x2 zeros(1, abs(k(end)-
n2(end)))];

subplot(3,1,1);
stem (k,y1); xlabel('n1'); ylabel('x1'); title('First Signal');
subplot(3,1,2);
stem(k,y2); xlabel('n2'); ylabel('x2'); title('Second Signal');
subplot(3,1,3);
stem(k,y); xlabel('k'); ylabel('y'); title('Convoluted Output');
```

Output:



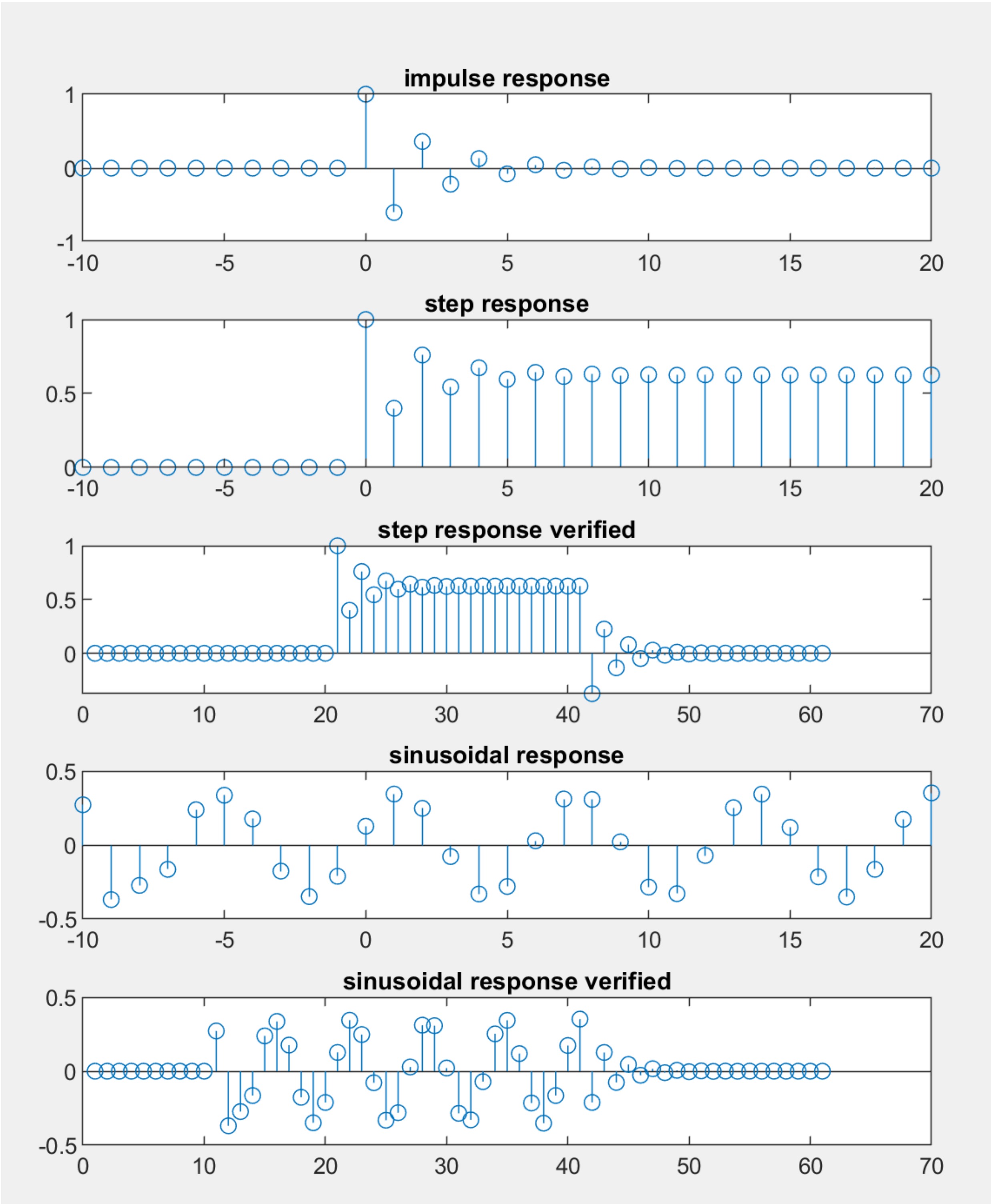
Part C

Exercise C1:

Code:

```
clc;
clearvars;

a = [1 0.6];
b = [1 0];

n = -10:20;

u = [(n)>=0];
r = n .* (n>=0);
s = 0.5 * sin(n);
i = [n==0];

x = 0.5 .* u;
impulse_response = filter(b,a,i);
step_response = filter(b,a,u);
sinusoidal_response = filter(b,a, s);

subplot(5,1,1)
stem(n,impulse_response); title('impulse_response');

subplot(5,1,2)
stem(n,step_response); title('step_response');
%verification 1
step_response_verified = conv(impulse_response, u);
subplot(5,1,3)
stem(step_response_verified); title('step_response_verified');

subplot(5,1,4)
stem(n,sinusoidal_response); title('sinusoidal_response');
%verification 2
sinusoidal_response_verified = conv(impulse_response, s);
subplot(5,1,5)
stem(sinusoidal_response_verified);
title('sinusoidal_response_verified');
```

Output:

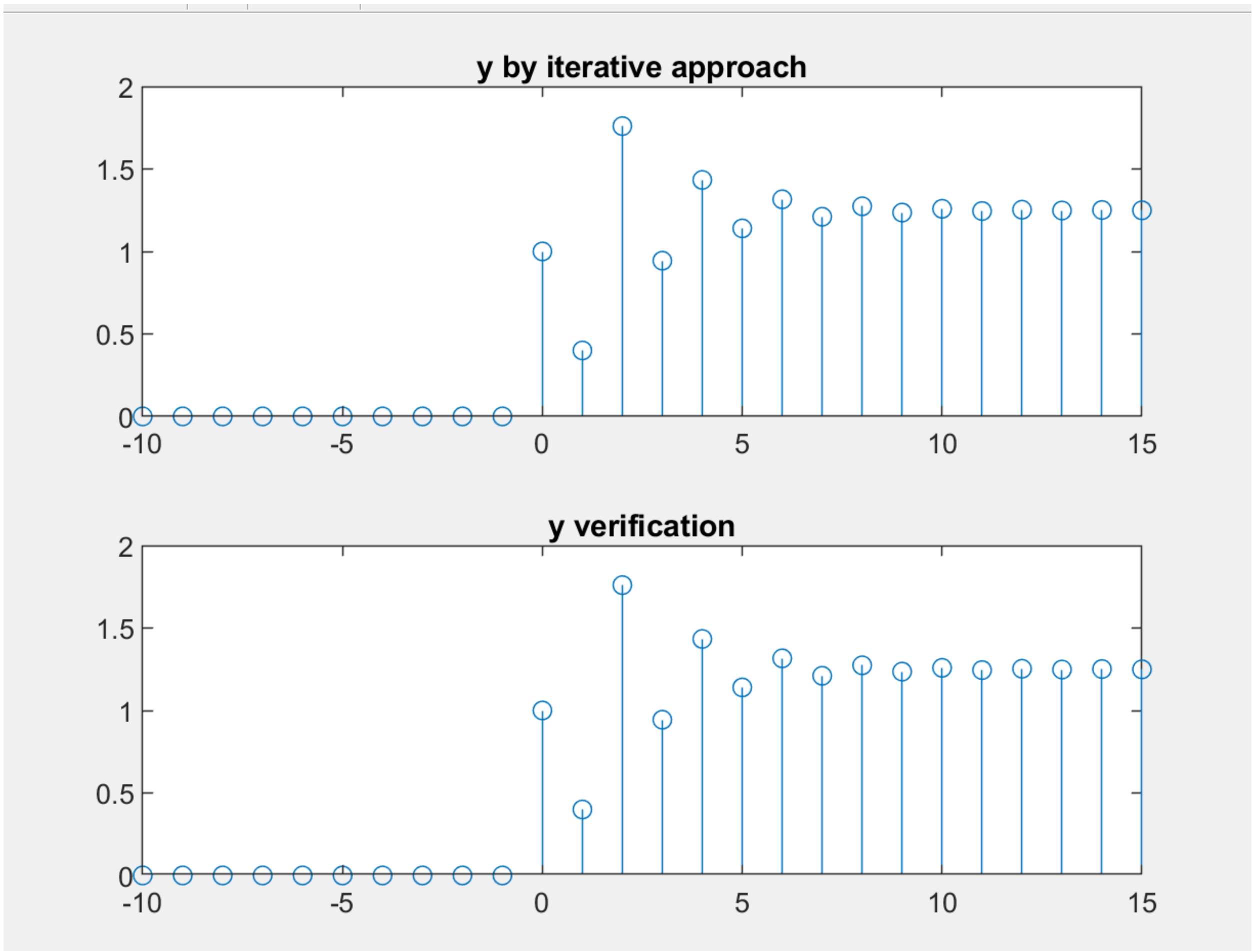

impulse response
step response
step response verified
sinusoidal response
sinusoidal response verified

Exercise C2:

Code:

```
clc;
clearvars;

clc;
clearvars;

a = [1 0.6 0];
b = [1 0 1];

n = -10:15;
y1 = zeros(1, length(n));
y2 = y1;

u = [n>=0];
%itierative approach
for i = 3:length(n)
    y1(i) = u(i) - 0.6 * y1(i-1);
    y2(i) = u(i-2)- 0.6 * y2(i-1);
end

y = y1+y2; %superposition
y_verification = filter(b,a,u);
subplot(2,1,1);
stem(n, y); title('y by iterative approach');
subplot(2,1,2)
stem(n, y_verification); title('y verification');
```

Output:



y by iterative approach
2
1.5
1
0.5
0
-10
-5
0
5
10
15
y verification
2
1.5
1
0.5
0
-10
-5
0
5
10
15

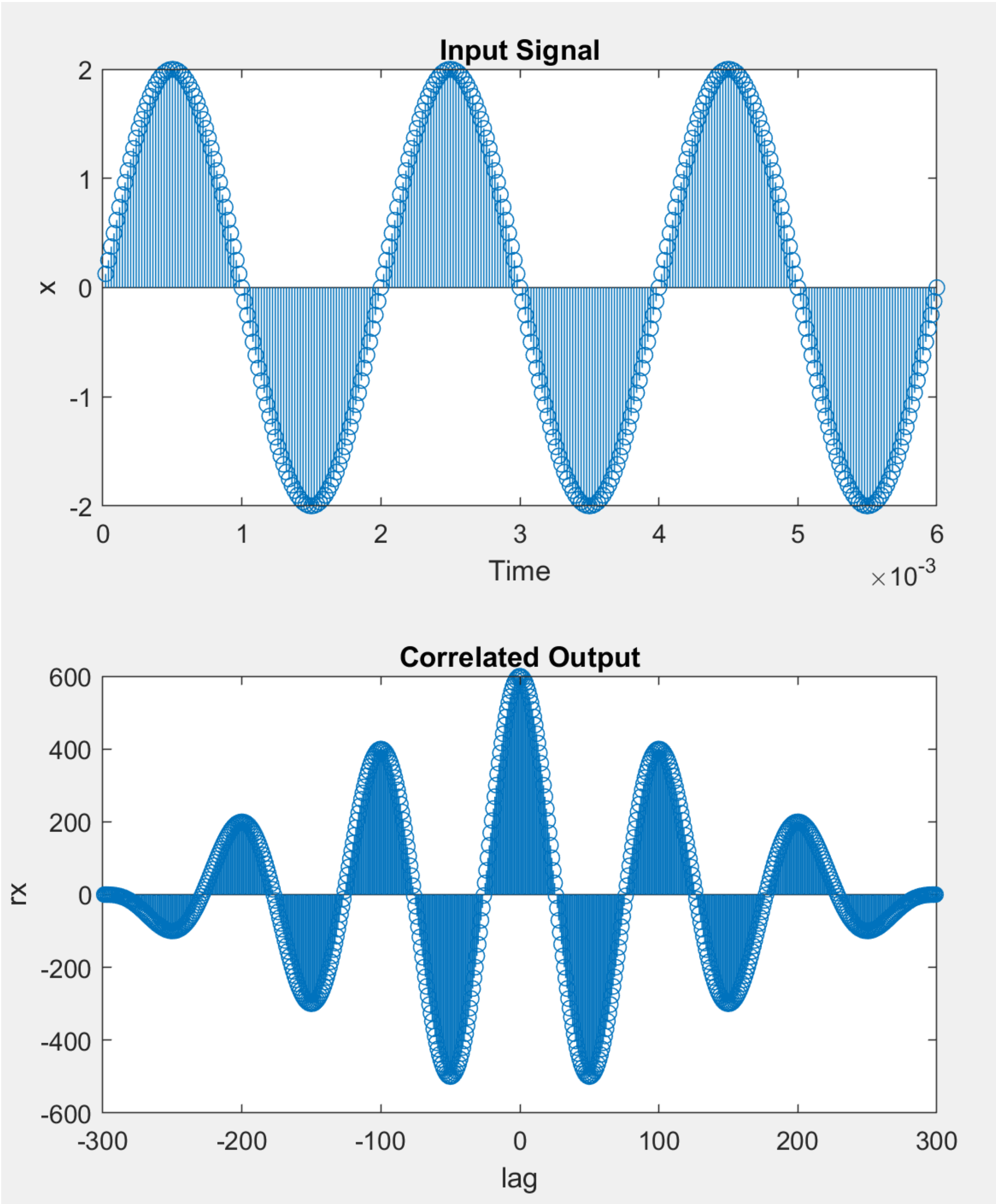
Part D

Exercise D1:

```
clc;
clearvars;

%Exercise D1 B

T = 2*10^(-3);
tstep = T/100;
t = tstep:tstep:3*T;

x = 2* sin(2*pi*t/T)

[rx,lag] = xcorr(x);
subplot(2,1,1)
stem(t,x); xlabel('Time'); ylabel('x'); title('Input Signal');
subplot(2,1,2)
stem(lag, rx); xlabel('lag'); ylabel('rx'); title('Correlated
Output');
```

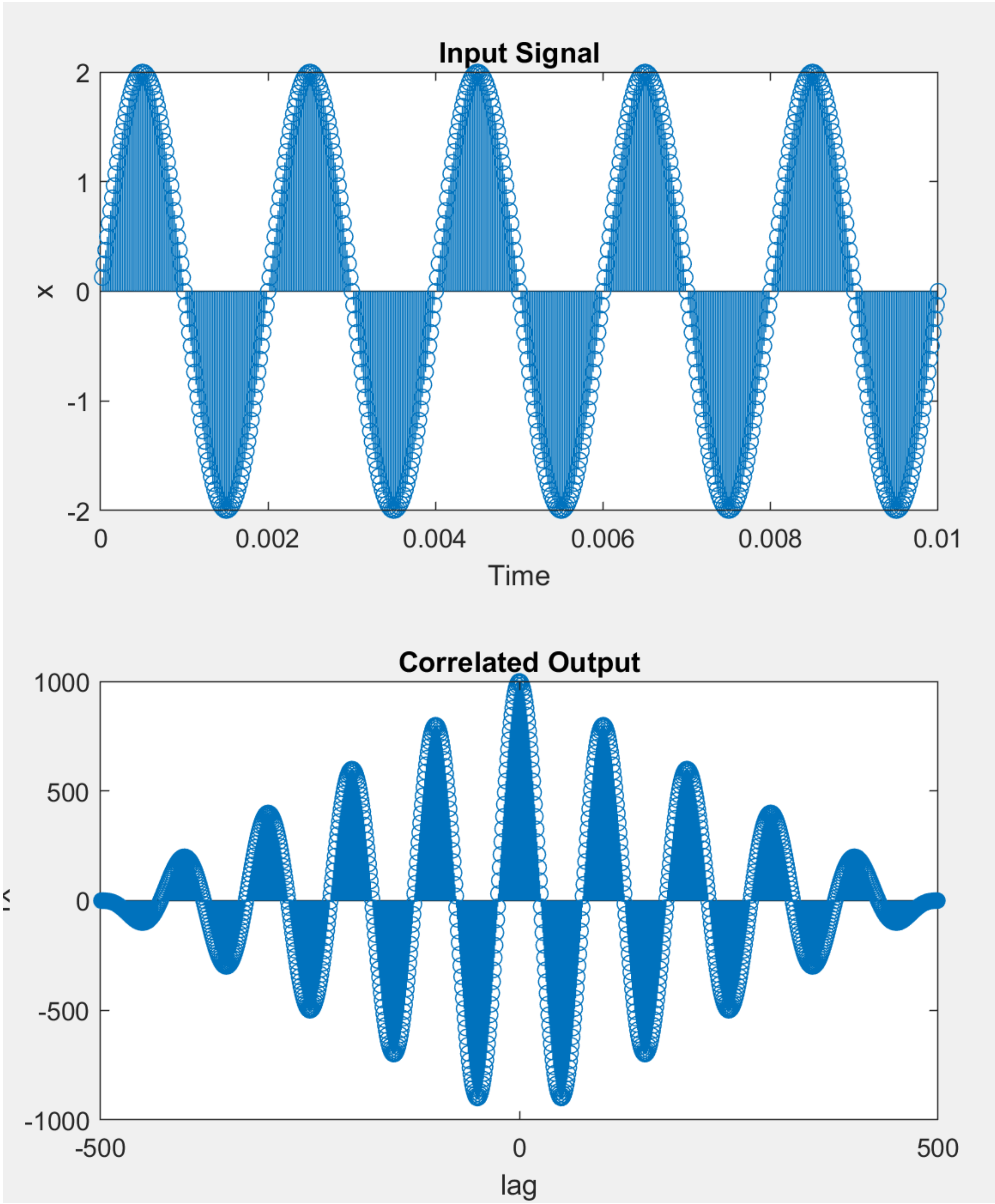
Output:

For 3T:



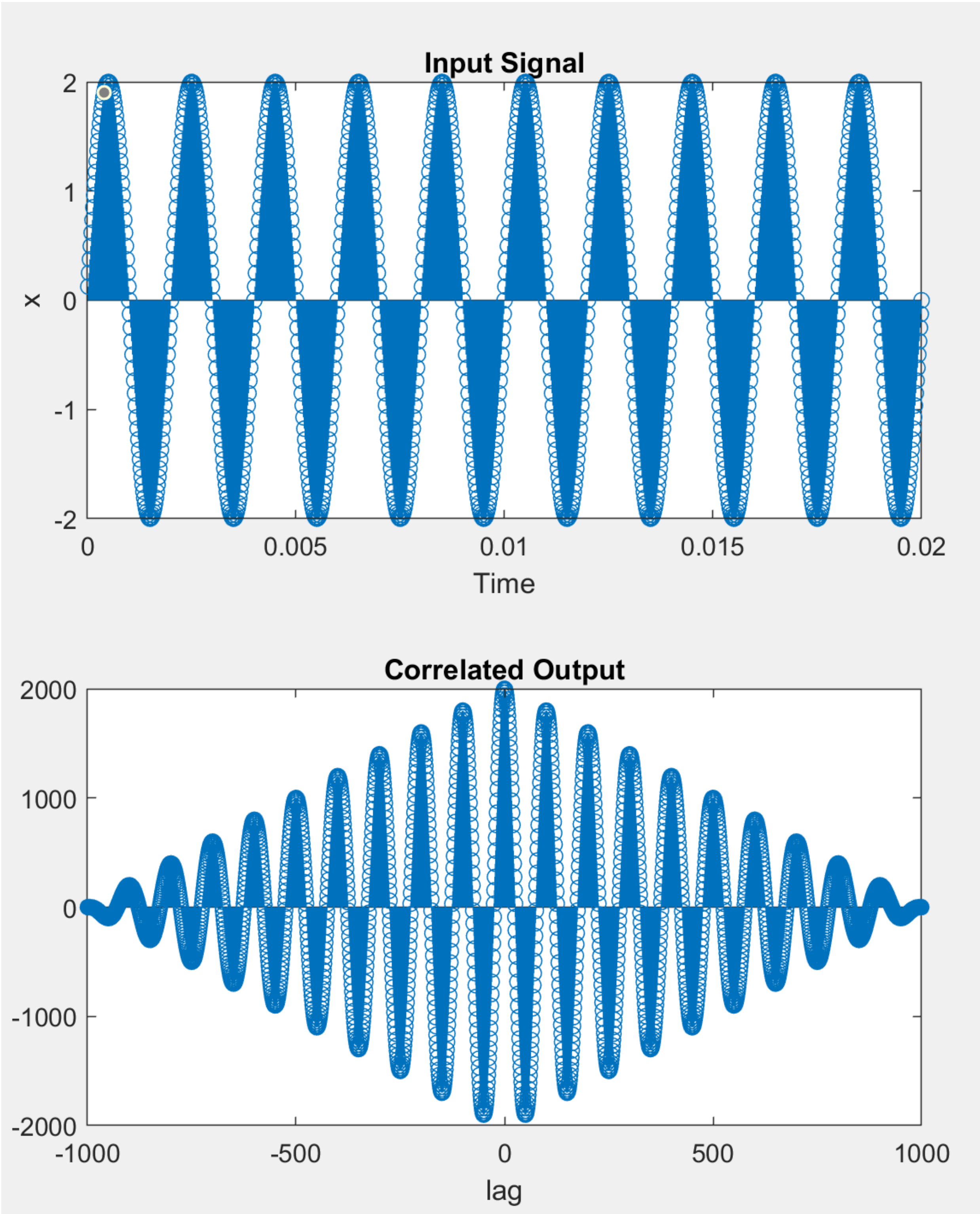
Input Signal
2
1
0
-1
-2
x
0
1
2
3
4
5
6
Time
×10^-3
Correlated Output
600
400
200
0
-200
-400
-600
rx
-300
-200
-100
0
100
200
300
lag

For 5T:



Input Signal
2
1
0
-1
-2
x
0
0.002
0.004
0.006
0.008
0.01
Time
Correlated Output
1000
500
0
-500
-1000
-500
0
500
lag

For 10T:



Input Signal
2
1
0
-1
-2
x
0
0.005
0.01
0.015
0.02
Time
Correlated Output
2000
1000
0
-1000
-2000
-1000
-500
0
500
1000
lag

Comment:

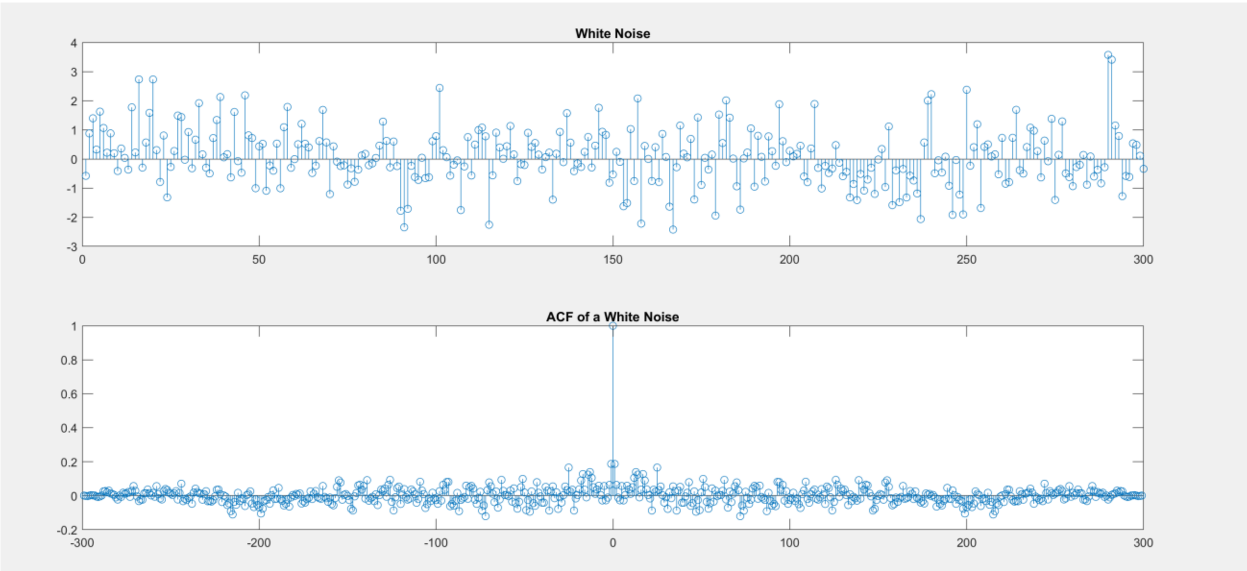
The output graph is getting denser, and there are more peaks occurring as the time period increases. Conceptually as the signals are getting multiplied followed by a summation process, so higher time period is resulting in a higher number of peaks.

Exercise D1 C:

Code:

```
clc;
clearvars;

%Exercise D1 c

%observe ACF of a random white noise

white_noise = randn(1, 300);
[rxx lag] = xcorr(white_noise);
rxx = rxx/max(rxx);
subplot(2,1,1);
stem(white_noise); title('White Noise');
subplot(2,1,2);
stem(lag, rxx); title('ACF of a White Noise');
```

Output:

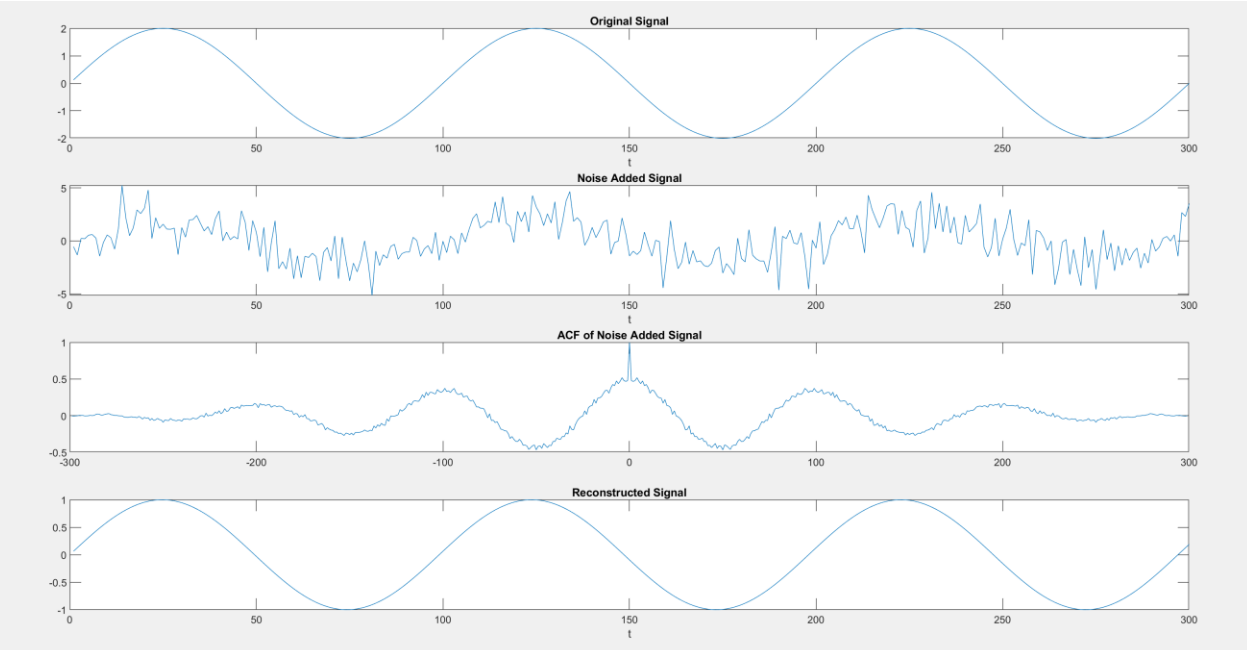


Exercise D2:

Code:

```
clc;
clearvars;


T = 2*10^(-3); % period = 2 ms
N = 100; % number of points
tstep = T/N; % time step

t = tstep:tstep:3*T;
x = 2*sin(2*pi*t/T);

Px = sum(x.^2)/length(x);
SNR = 0; %dB scale
P_noise = Px/10^(SNR/10);
noise_sig = sqrt(P_noise) * randn(1, length(t)); %final Noise
Signal

noise_added_sig = x + noise_sig; %corrupted input

[ACF_x lag_x] = xcorr(x);
ACF_x = normalize(ACF_x);


[ACF_noise_added_sig lag_noise_added_sig] =
xcorr(noise_added_sig);
ACF_noise_added_sig = normalize(ACF_noise_added_sig);

[peaks locs] = findpeaks(ACF_noise_added_sig,
'MinPeakProminence', 0.5);
time_period = tstep * mean(diff(locs))

subplot(4,1,1)
plot(x); title('Original Signal'); xlabel('t');
subplot(4,1,2)
plot(noise_added_sig); title('Noise Added Signal'); xlabel('t');
subplot(4,1,3)
plot(lag_noise_added_sig, ACF_noise_added_sig); title('ACF of
Noise Added Signal');
subplot(4,1,4)
plot(sin(2*pi*t/time_period)); title('Reconstructed Signal');
xlabel('t');
```

Output:



Comment:

We have plotted the reconstructed signal by retrieving the information of time period from the prominent peaks. However, we cannot retrieve any information of the phase shift as the auto correlation diminishes the information of the phase from the signal.

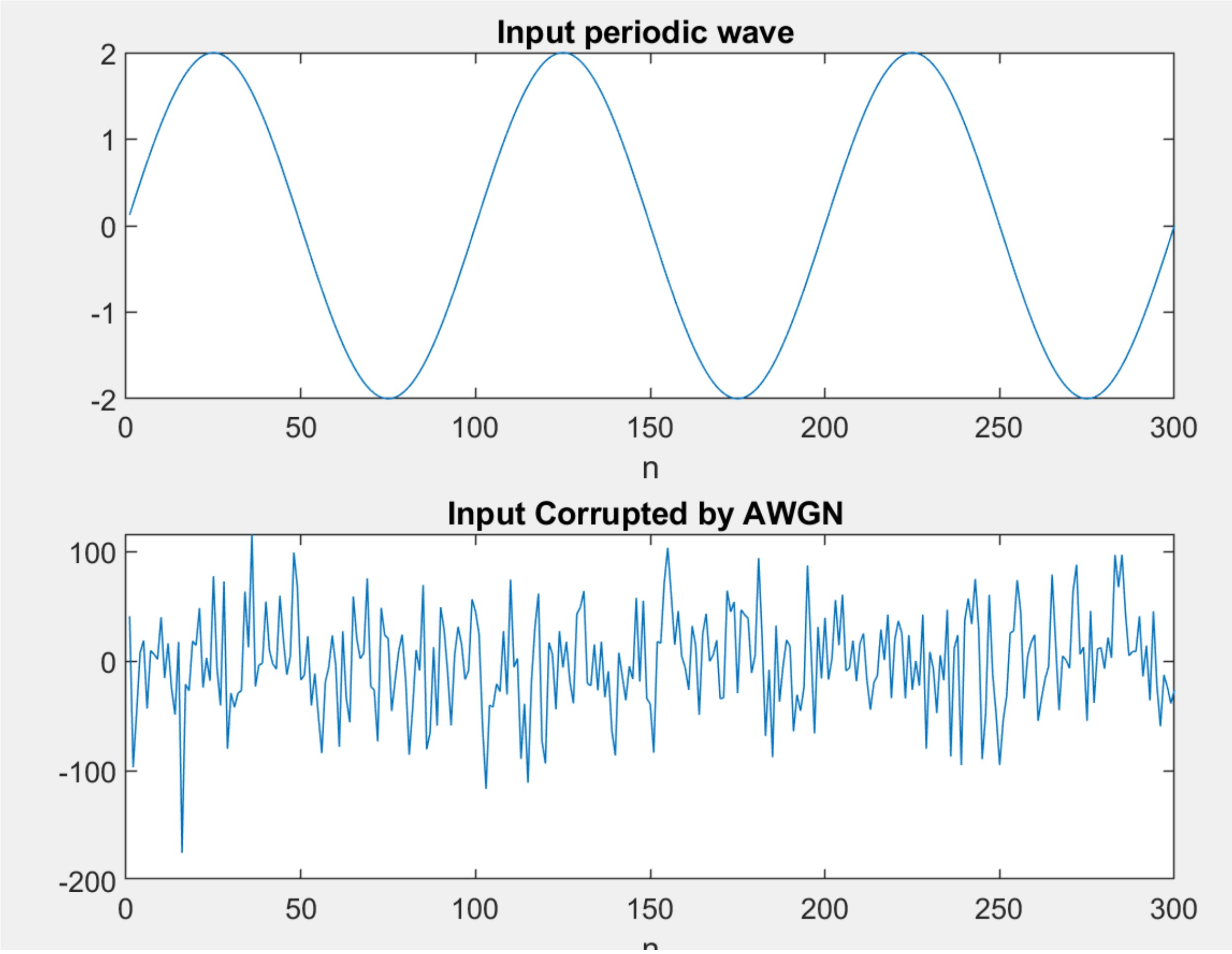
PartD: 1. Detecting a periodic Input Corrupted by AWGN

Code:

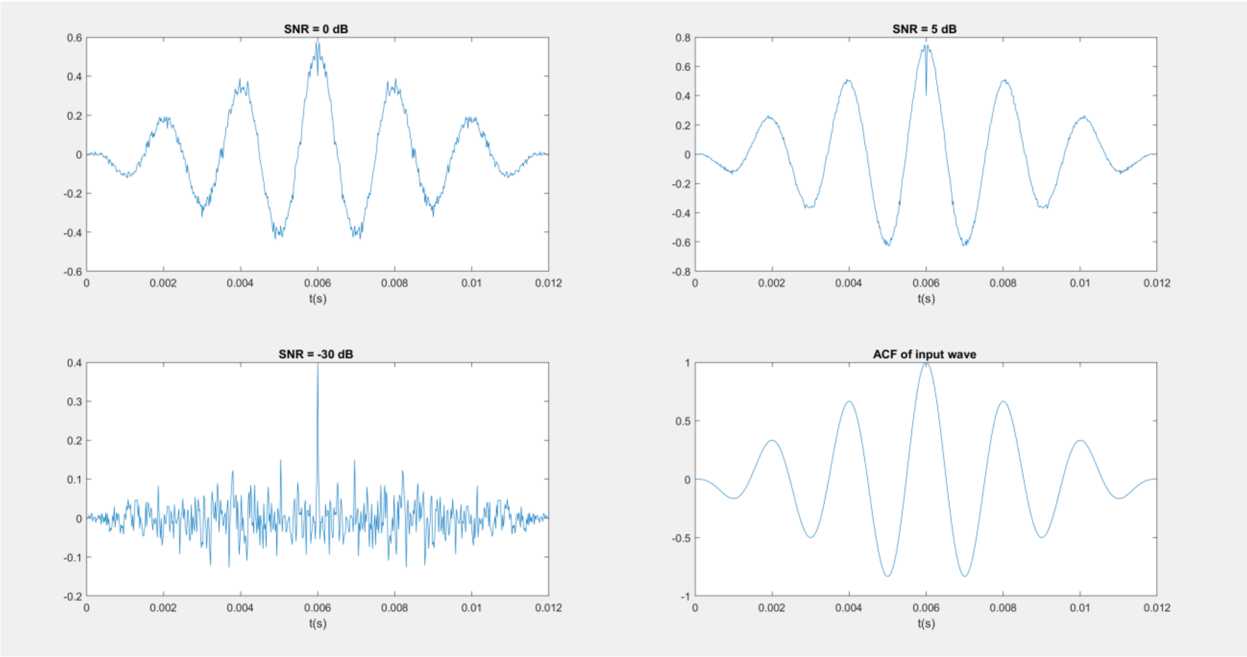
```
clc;
clearvars;

 T = 2*10^(-3); % period =2 ms
 N = 100; % number of points
 tstep = T/N; % time step

 t = tstep : tstep :3*T; % taking time index upto 3 periods
 x = 2* sin (2* pi*t/T); % Input signal
 ACF_x = normalize ( xcorr (x)); % Normalizing the peak to 1

 Px = sum(x .^2) / length (x); % Input power
 SNR = [0, 5, -30]; % in dB

 figure (1) ;
 for i = 1: length (SNR)
 Py = Px /10^( SNR (i) /10) ; % Noise power
 n = sqrt (Py)* randn (1, length (t)); % generating white noise
 %general randn function creates values with zero mean and 1
Standard
 %Deviation
 %however power denotes variance, so we multiplied sqrt(py)
 y = x + n; % Corrupted input

 ACF_x = normalize ( xcorr (x));
 ACF_n = normalize ( xcorr (n));
 ACF_y = normalize ( xcorr (y));
 ACF_y ( length (x)) = 0.4* max( ACF_y );
 %You can enable this line for  -better
 % understanding . Scaling value  -should
 %be decreased for lower SNR

 subplot (2 ,2 ,i)

 plot ( tstep *(1: length ( ACF_y )),ACF_y ) % showing ACF
w.r.t. time
 title ( sprintf ('SNR = %d dB ', SNR(i))); xlabel ('t(s)');
 end
```

```
 %for higher snr, you can understand the value of periodicity
from checking
 %the peaks of the ACF graph, however, if the noise power is too
large, you
 %actually cant

 subplot (2 ,2 ,4);
 plot ( tstep *(1: length ( ACF_x )),ACF_x ) % showing ACF
w.r.t. time
 title ('ACF of input wave '); xlabel ('t(s)');

 figure (2)
 subplot (211) ,plot (x), title ('Input periodic wave '); xlabel
('n');
 subplot (212) ,plot (n), title ('Input Corrupted by AWGN ');
xlabel ('n');
```

Output:



Input periodic wave
n
Input Corrupted by AWGN
n



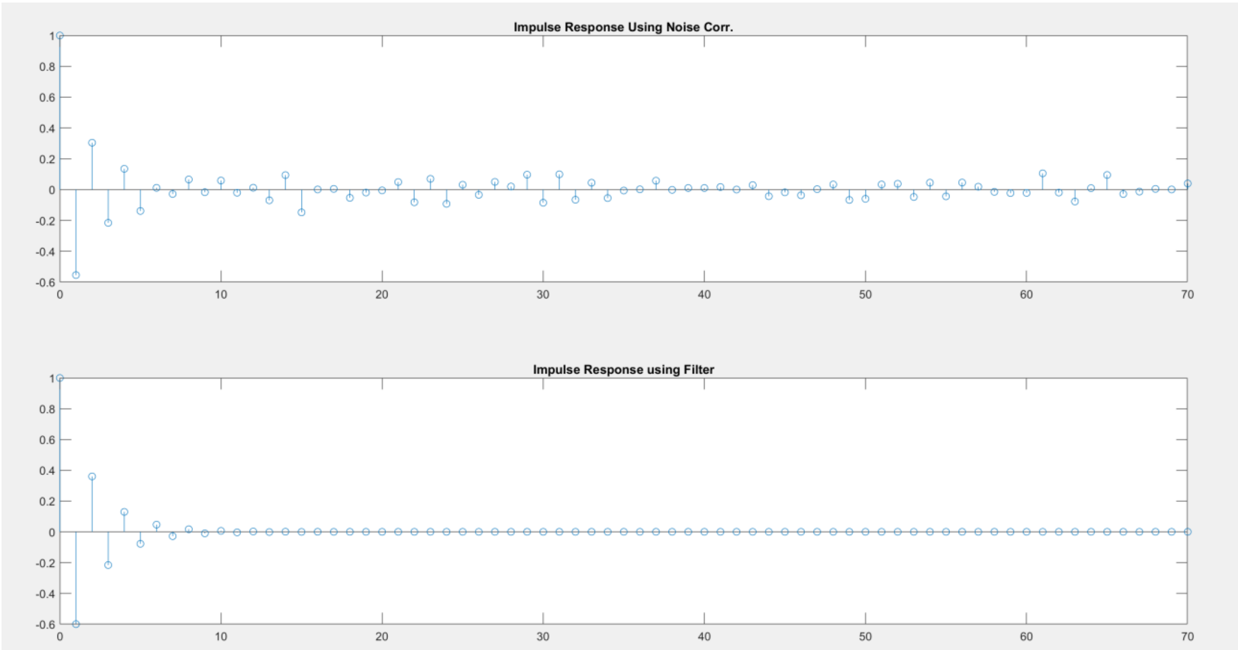
SNR = 0 dB
t(s)
SNR = 5 dB
t(s)
SNR = -30 dB
t(s)
ACF of input wave
t(s)

PartD: 2. Estimation of Impulse Response

Code:

```
clc;
clearvars;

%consider a system y(n) + 0.6 y(n-1) = x(n)
%now estimate the impulse response of the system using noise
autocorr


N = 500;
nr = 0:499;
ny = nr;

r = randn(1,N);
y = zeros(size(r));
for n = 2:N %itierative approach
    y(n) = r(n) - 0.6* y(n-1);
end

[rr nrr] = sigfold(r,nr);

[Ryr k] = conv_m(y, ny, rr, nrr);

subplot(2,1,1)
stem(k, Ryr/max(Ryr)); title('Impulse Response Using Noise
Corr.') %max(Ryr)= Ryr(N)
xlim([0 70]); %limiting the values of X axis

%verification
impulse = [nr==0];
a = [1 0.6];
b = [1 0];

impulse_response = filter(b, a, impulse);
subplot(2,1,2)
stem(nr, impulse_response); title('Impulse Response using
Filter')
xlim([0 70]); %limiting the values of X axis
```

Output:



Impulse Response Using Noise Corr.
Impulse Response using Filter

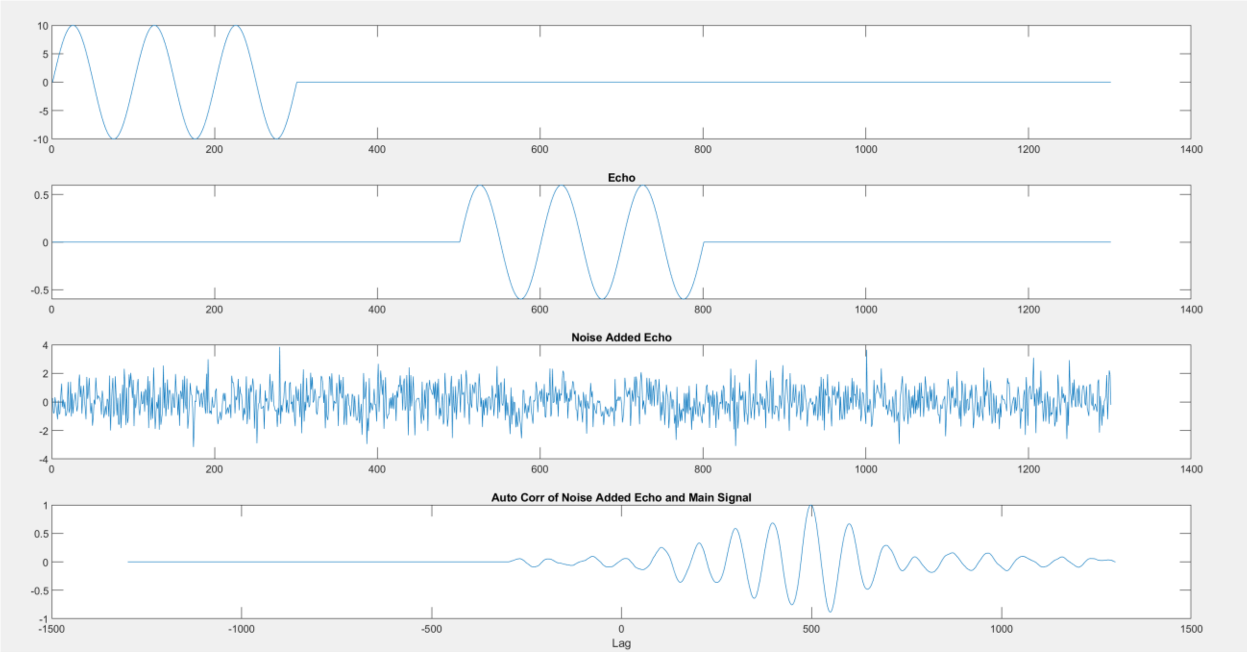
End of Experiments:

1. Detection of Signals in Noise by Auto Correlation:

Code(Single Frequency):

```
clc;
clearvars;

T = 100;
t = 0:3*T;
x1 = 10*sin(2*pi*t/T); %single Frequency
x = [ x1 zeros(1,1000)];

echo = [zeros(1,500) 0.06.*x1 zeros(1,500)];



noise = randn(1, length(echo));
noise_add_sig = echo + noise;

[Rxr lag] = xcorr(noise_add_sig, x);
Rxr = normalize(Rxr);

subplot(4,1,1)
plot(x); ('Transmitted Original Signal')
subplot(4,1,2)
plot(echo); title('Echo')
subplot(4,1,3)
plot(noise_add_sig); title('Noise Added Echo');
subplot(4,1,4)
plot(lag, Rxr); xlabel('Lag'); title('Auto Corr of Noise Added
Echo and Main Signal')
```

Output:



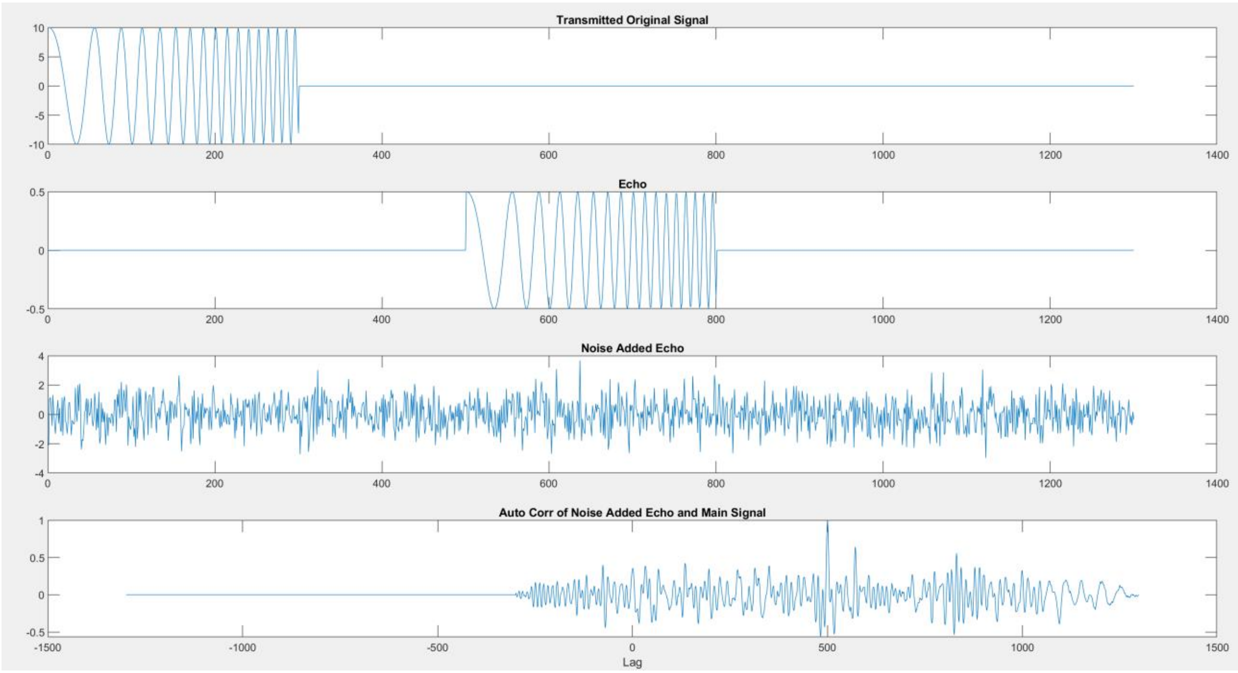
Comment:

We can observe the maximum peak of the correlated graph comes at a 500 lag which is the starting point of the echo. So, the footprint of the echo is successfully recovered.

Code(Multiple Freq):

```
clc;
clearvars;

T = 100;
t = 0:3*T;
x1 = 10*chirp(0:299,0.01,t(end), 0.1);
%be careful with chirp thing
%understand the value thing
x = [ x1 zeros(1,1000)];

echo = [zeros(1,500) 0.05.*x1 zeros(1,500)];



noise = randn(1, length(echo));
noise_add_sig = echo + noise;

[Rxr lag] = xcorr(noise_add_sig, x);
Rxr = normalize(Rxr);
%multifreq burst is better single freq burst
subplot(4,1,1)
plot(x); ('Transmitted Original Signal')
subplot(4,1,2)
plot(echo); title('Echo')
subplot(4,1,3)
plot(noise_add_sig); title('Noise Added Echo');
subplot(4,1,4)
plot(lag, Rxr); xlabel('Lag'); title('Auto Corr of Noise Added
Echo and Main Signal')
```

Output:



Comment:

So, multiple frequency burst is convenient for the echo footprint detection as the peak prominence is higher in the correlated signal.

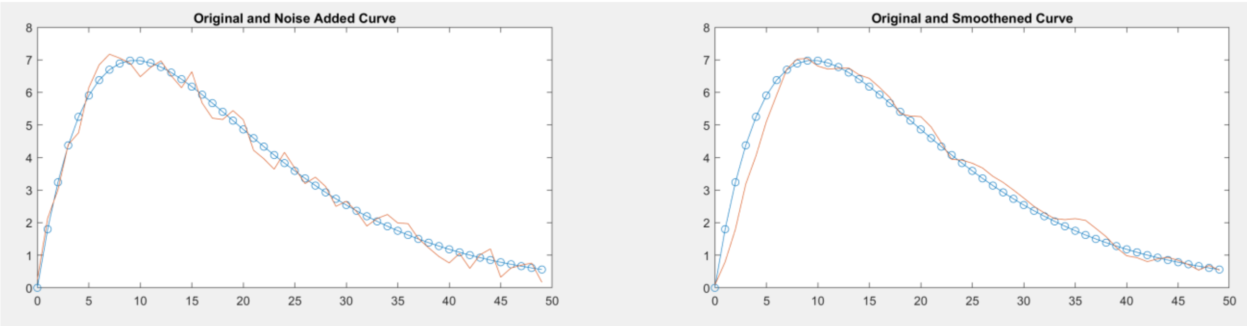
2. Signal Smoothing by a moving average system:

Code:

```
clc;
clearvars;

M =3;
temp = 0;
n = 0:49;
x = 2.*n.*(0.9.^(n));
subplot(2,2,1)
plot(n,x,'-o'); title('Original and Noise Added Curve')
hold on
noise = rand(1,50)-.5; %rand usually gives random numbers from 0
to 1
%now it will give random numbers of 0.5 to -0.5
y = x + noise;
plot(n,y);
hold off

z = zeros(1,length(n));
%for initial values
for i = 1:M-1
    z(i) = y(i)/M + temp;
    temp = z(i);
end

temp = 0;
for i = M:length(n)
    for j = 0:M-1
    z(i) = y(i-j)/M + temp;
    temp = z(i);
    end
    temp = 0;
end

error = zeros(1, length(x));
for i = 2:length(n)
error(i) = abs((x(i)-z(i))/x(i));
end



subplot(2,2,2)
```

```
plot(n,x,'-o'); title('Original and Smoothened Curve');
hold on
plot(n,z)
hold off
```

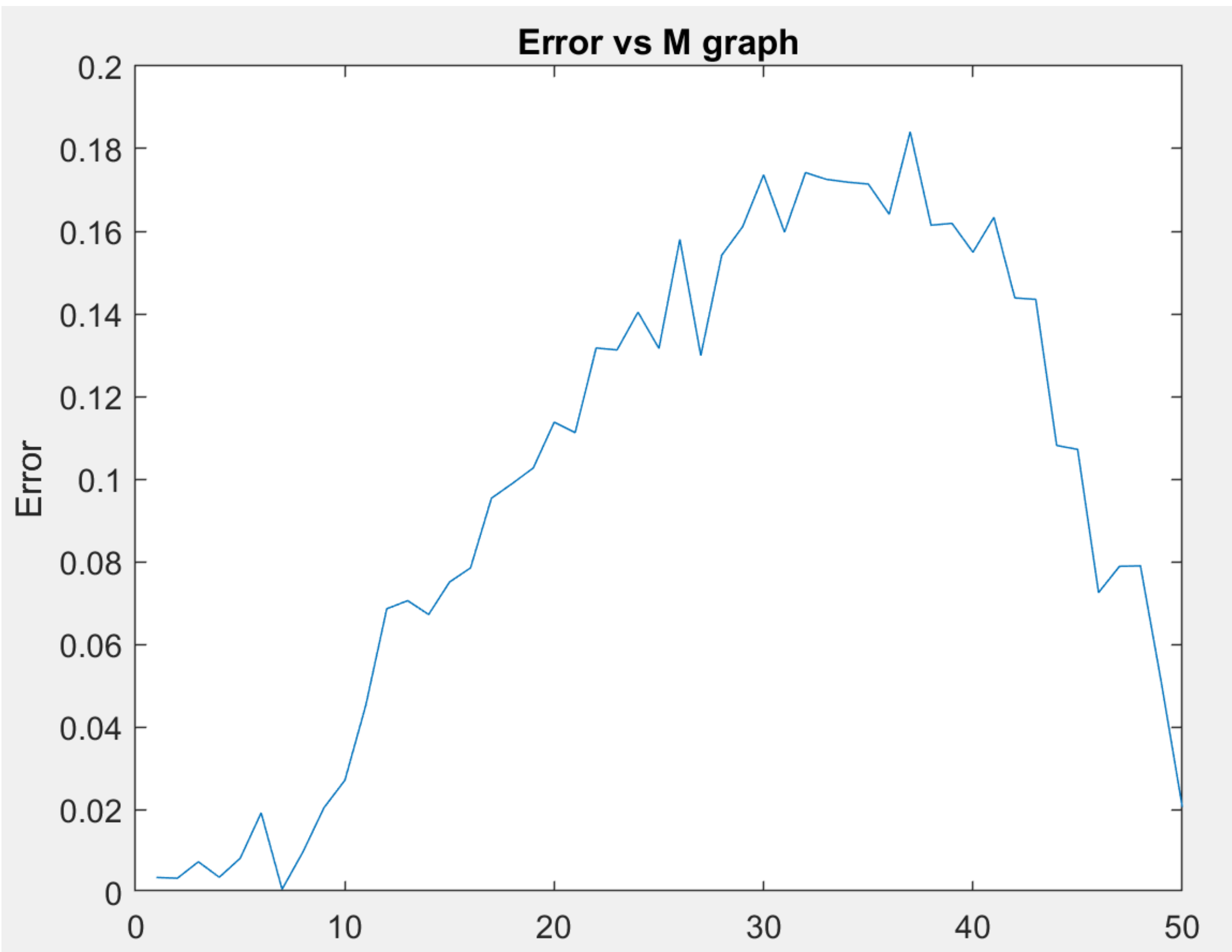
Output:



Code for Error vs M graph:

```
clc;
clearvars;

n = 0:49;
x = 2.*n.*(0.9.^(n));
M = 1;
for m = 1: length(n)
temp = 0;
noise = rand(1,50)-.5; %rand usually gives random numbers from 0
to 1
%now it will give random numbers of 0.5 to -0.5
y = x + noise;

z = zeros(1,length(n));
%for initial values
for i = 1:M-1
    z(i) = y(i)/M + temp;
    temp = z(i);
end

temp = 0;
for i = M:length(n)
    for j = 0:M-1
```

```
    z(i) = y(i-j)/M + temp;
    temp = z(i);
    end
    temp = 0;
end
error(m) = sum(abs((x-z)/z))
M = M+1;
end

plot(error); xlabel('M'); ylabel('Error'); title('Error vs M
graph')
```

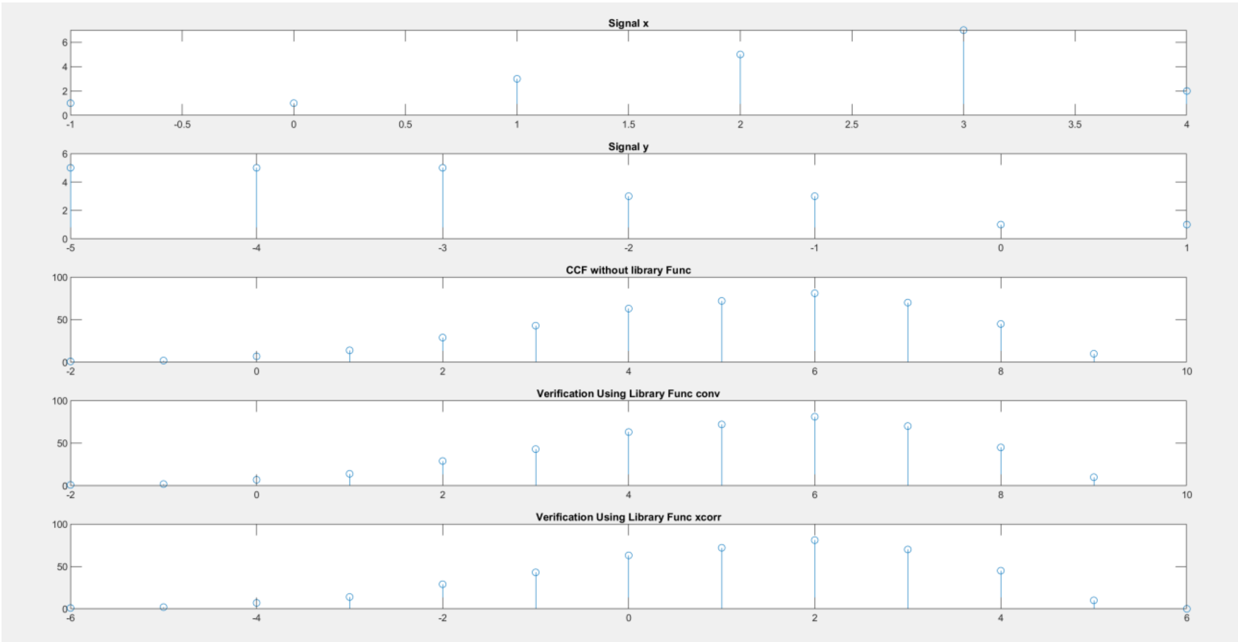


3. Manual Corr and Conv Use:

Code:

```
clc;
clearvars;

x = [1 1 3 5 7 2];
n_x = -1:4;
y = [5 5 5 3 3 1 1];
n_y = -5:1;

nz = (n_x(1)-n_y(end)):(n_x(end)-n_y(1));

z = zeros(1,length(nz));
X = [zeros(1,length(y)-1),x,zeros(1,length(y)-1)];
for i = 1:length(nz)
Y = [zeros(1, i - 1), y, zeros(1, length(X) - length(y) - i +
1)];
z(i) = sum(X.*Y);
end
Rxy = conv(x,fliplr(y));
[Rxy2 lags] = xcorr(x,y);


subplot(5,1,1)
stem(n_x,x); title('Signal x');
subplot(5,1,2)
stem(n_y,y); title('Signal y');
subplot(5,1,3)
stem(nz,z); title('CCF without library Func')
subplot(5,1,4)
stem(nz,Rxy); title('Verification Using Library Func conv')
subplot(5,1,5)
stem(lags,Rxy2); title('Verification Using Library Func xcorr')
```

Output:


Signal x
Signal y
CCF without library Func
Verification Using Library Func conv
Verification Using Library Func xcorr