Bangladesh University of Engineering and Technology

Course Number: EEE 312

Course Title: Digital Signal Processing Laboratory

Experiment Number: 02 (d)

Name of the Experiment(s):

Time domain analysis of discrete signals and systems (Correlation)

Date of Performance: 11/01/2023

Date of Submission: 20/01/2023

Prepared by:

Tasmin Khan

Student ID: 1906055

Partner: Tasnimun Razin

Student ID: 1906044

Section: A2

Level:3 Term:1

Department: EEE

Problem 1

1. Write a code segment that will perform cross-correlation between two discrete signals using only the conv( ) function.

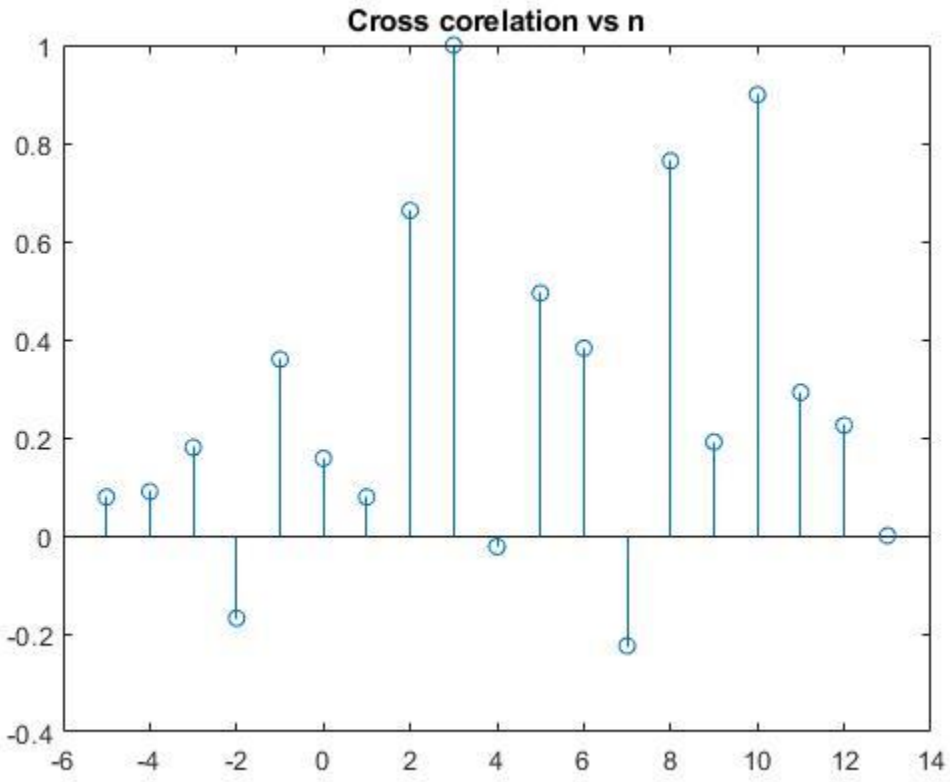
Defining function:

```
function [y,n]=ccr(x1,n1,x2,n2)
x2f=flip(x2);
n2f=-flip(n2);
n=(n1(1)+n2f(1)):(n1(end)+n2f(end));
y=conv(x1,x2f);
end
```

Main code:

```
clc
clear all

%input signal
x1=[ 1 1 2 -3 4 2 1 8 2 4];
x2=[ 0 5 4 8 -9 2 3 1 1 7];
n1=-2:7;
n2=-6:3;

%cross corelation
[rx1x2,n]=ccr(x1,n1,x2,n2);

%plotting
stem(n,rx1x2/max(rx1x2));
title("Cross corelation vs n")
```

Plot:



Discussion:

Here we have taken two discrete signals as x1 and x2.

Correlating two signals can be expressed as:

$$x[n]*y[n] = \sum_{n=-\infty}^{\infty} x[n]y[n-k]$$

And convolution can be expressed as:

$$x[n]*y[n] = \sum_{n=-\infty}^{\infty} x[n]y[k-n]$$

Thus cross correlation can be done using convolution by flipping one signal and passing it through conv function. In terms of expression we can write correlation as,

$$r_{xy} = x[n]*y[-n]$$

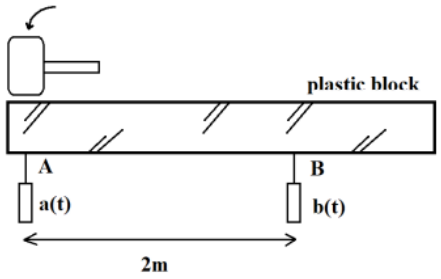
In the plot we see the maximum value occurs at n=3, that is where the correlation is maximum.

Problem 2

2. The figure below depicts a plastic block that has been hit by a hammer on its left side. Two piezoelectric sensors (with some additional circuits) have been placed on points 'A' and 'B'. The vibration caused by the hit enabled the sensors to be faced with the following two signals at those two points:

$$a(t) = 4e^{-6t}\sin(400t)\,u(t)$$

$$b(t) = 2e^{-6t}\sin\left(400t - \frac{\pi}{10}\right)(1 - e^{-1000t})\,u(t)$$



If the sensors are designed to have a sampling frequency of $\mathbf{10kHz}$, estimate the sound velocity created by the vibration in that plastic medium.

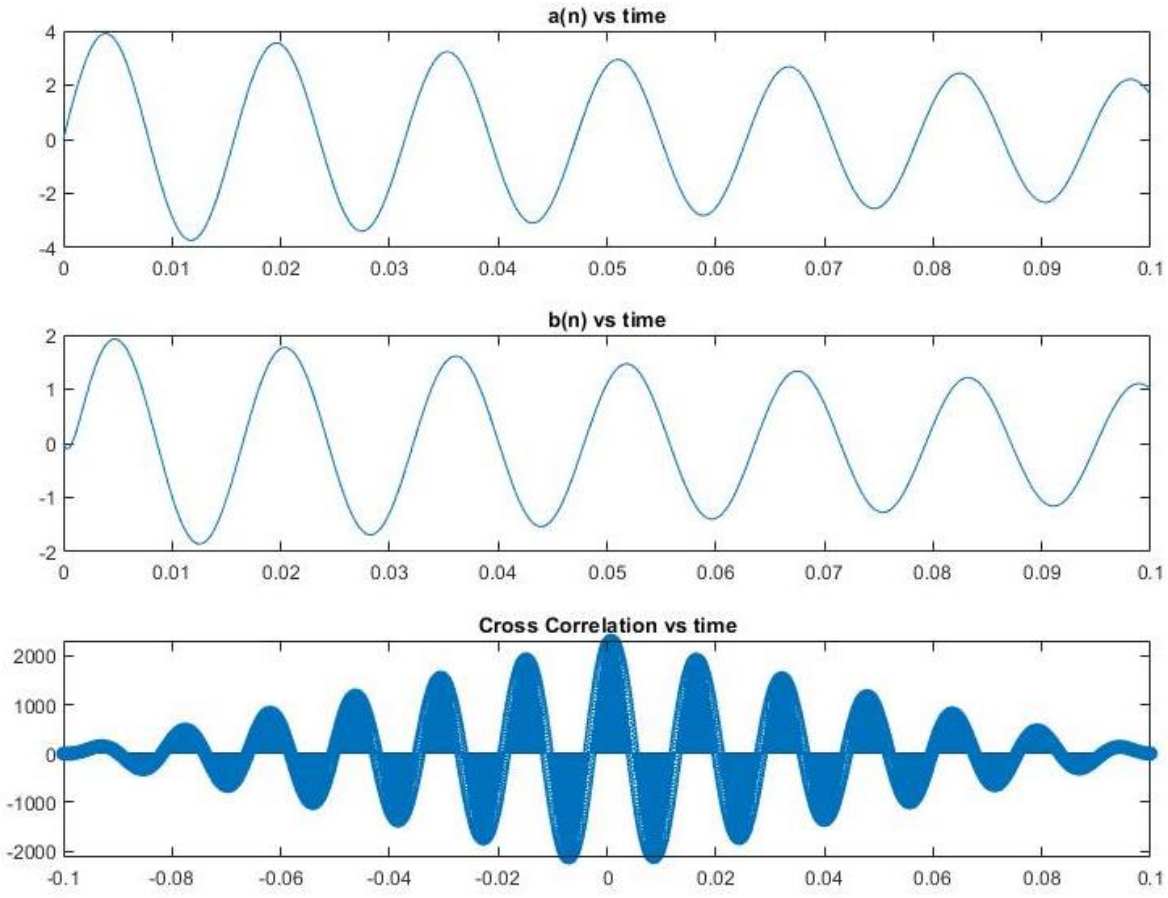
Code:

```
%sampling the inputs
fs=10000;
ts=0:1/fs:0.1;
ns=ts*fs;
u=1*[ts>=0];
an=4*exp(-6*ts).*sin(400*ts).*u;
bn=2*exp(-6*ts).*sin(400*ts-(pi/10)).*(1-exp(-1000*ts)).*u;

%cross corelation
z=xcorr(bn,an);
nf=-flip(ns);
ncorr=(ns(1)+nf(1)):(ns(end)+nf(end));

%plotting
figure(1);
subplot(3,1,1);
plot(ns/fs,an);
title("a(n) vs time");
subplot(3,1,2);
plot(ns/fs,bn);
title("b(n) vs time");
subplot(3,1,3)
stem(ncorr/fs,z);
title("Cross Correlation vs time");

%delay calculation
[m,i]=max(z);
delay=ncorr(i)/fs;
distance=2;
velocity=distance/delay;
fprintf("The velocity of sound in the plastic medium is %f m/s\n",velocity);
```

Command window output:

```
Command Window
The velocity of sound in the plastic medium is 2500.000000 m/s
>>
```

Plot:



Discussion:

The first input signal is collected by the first sensor, which is the immediate vibration from the hammer. The second input signal is a decayed and delayed version of the same vibration collected by the second sensor. As the signal passes through the medium, it goes through a small delay when it is picked up through the second sensor.

We used correlation between the two signals to determine the time delay that occurred. For maximum value of correlation, the signals overlap each other with maximum match, and that instant is our desired delay.

If there had been no delay, we would see the maximum value at time index zero.

Finally, we use the delay and distance between the two sensors to calculate the velocity.

Problem 3

3. Use the `audioread()` function to read a voice clip of yourself (or somebody you know) in an .mp3 or a .wav format. `audioread()` returns a signal and a sampling frequency:

```
[y,Fs] = audioread("sample.mp3");
```

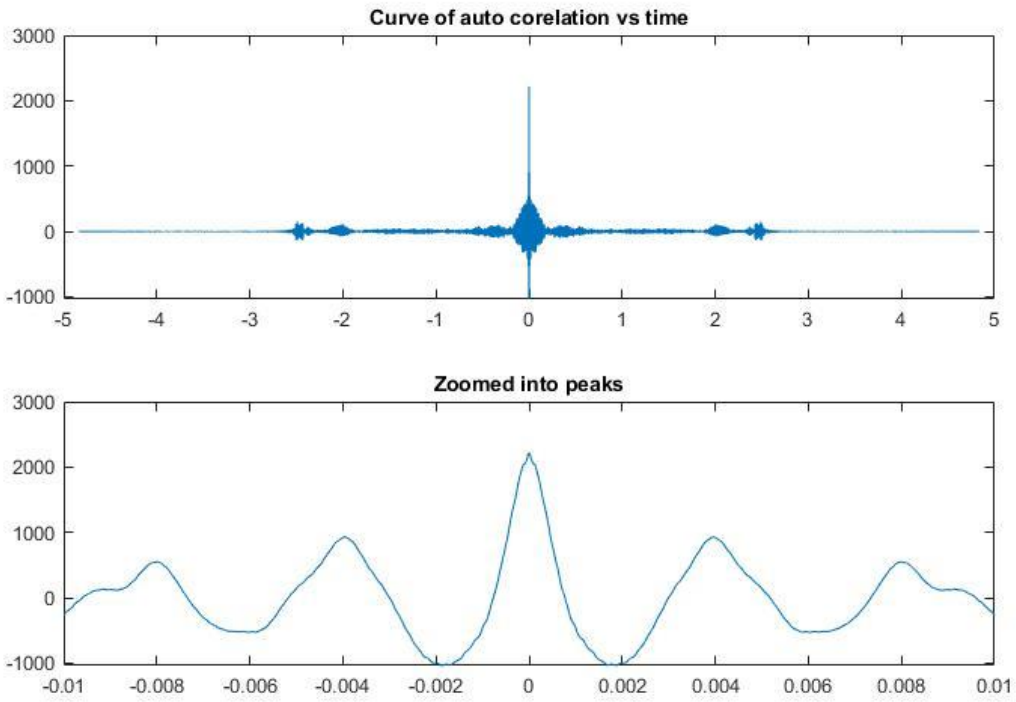
Considering the principles of **correlation**, write a MATLAB code to estimate the voice pitch (fundamental frequency of that voice) from the voice clip signal.

Code:

```
clc
clear all
close all

%audio input
[y,fs]=audioread("DSP audio signal.mp3");
y=y(:,1);
ys=y';

%auto correlation of y
ryy=(xcorr(ys));
Lyy=length(ryy);
n=(Lyy-1)/2;
nyy=-n:n;
tyy=nyy/fs;

%plotting
subplot(2,1,1)
plot(nyy/fs,ryy);
title("Curve of auto corelation vs time");
subplot(2,1,2)
plot(tyy,ryy);
title("Zoomed into peaks");
xlim([-0.01 0.01]);

%defining searching range
izero=n+1;
range=0.1;
irange=(izero+range*fs);

%searching for second hightest peak
k=1;
for j=izero+1:irange
    if ryy(j)>ryy(j-1) && ryy(j)>ryy(j+1)
        maximas(k)=ryy(j);
        imaximas(k)=j;
        k=k+1;
    end
end
```

```
[peak2,imax]=max(maximas);
imax=imaximas(imax);

%pitch calculation
delt=tyy(imax);
pitch=1/delt;
fprintf("\nThe pitch of the input voice is  %f Hz\n",pitch);
```

Plot:



Command window output:

Command Window

```
The pitch of the input voice is  252.631579 Hz
>>
```

Discussion:

Our target was to calculate the pitch of the input audio signal. We used autocorrelation for this purpose. Since autocorrelation gives maximum peak at time = 0, we needed to find the time for only the second highest peak.

To find the second highest peak, we defined a range of 0.1 second. We know audible sound range starts from 20Hz, hence taking 0.05 second was enough. We took a bigger range to be safe and built a set of maxima.

If any point had higher values than the previous and the past values, it was stored as one of the maxima. After, we picked out the maximum from this set and that was our second highest peak.

Indexing was easier in this case since audio signal is only right sided and auto correlation gives symmetric output.

Finally, we calculated the time gap between these the highest peak at time=zero and our newfound second highest peak and used it to get the pitch.

For the input audio, it came out to be around **252.63 HZ.**

Problem 4

4. A clean Radar-signal is being sent around a transmission-tower (TX). A bird happened to get into its range. One of the Radar signals hit the bird and came back to the TX-tower. The retrieved (RX) signal, however, was not as clean as the transmitted one. It was very cold $(-40℃)$ that day making environmental sound velocity $305\ m/s$.

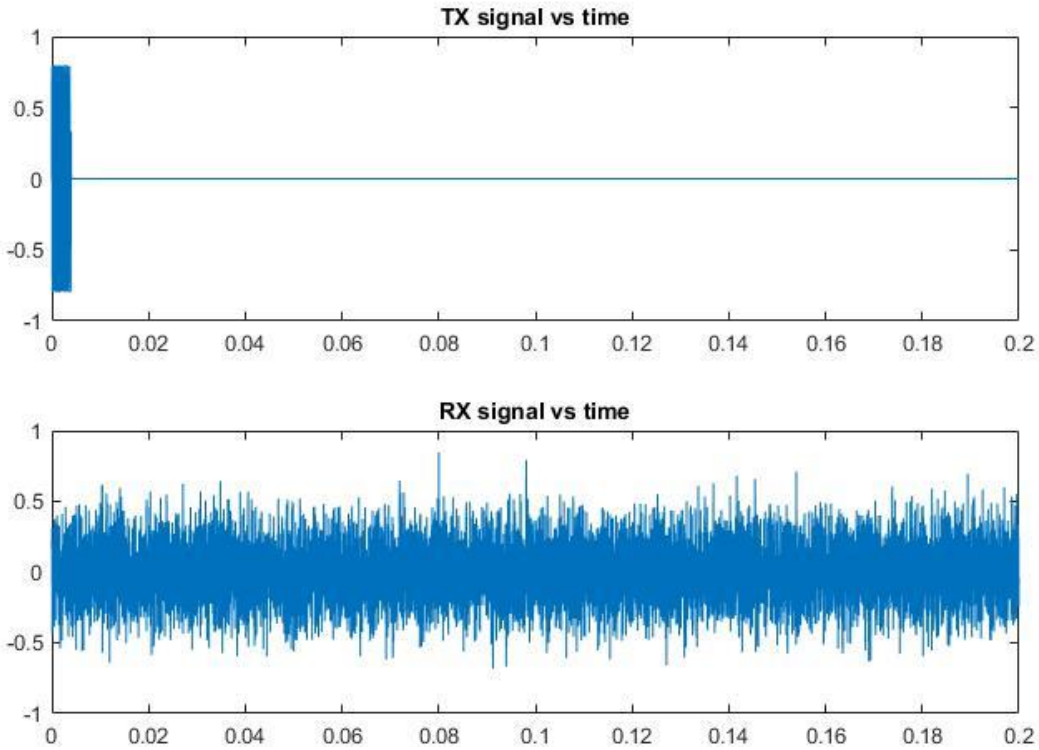
(a) Plot/Stem both TX and RX signals on the time span mentioned.
(b) Zoom into the non-zero part of the TX signal.
(c) How far away was the bird from the tower?
(d) Determine the noise power within the RX signal in dB.

Code:

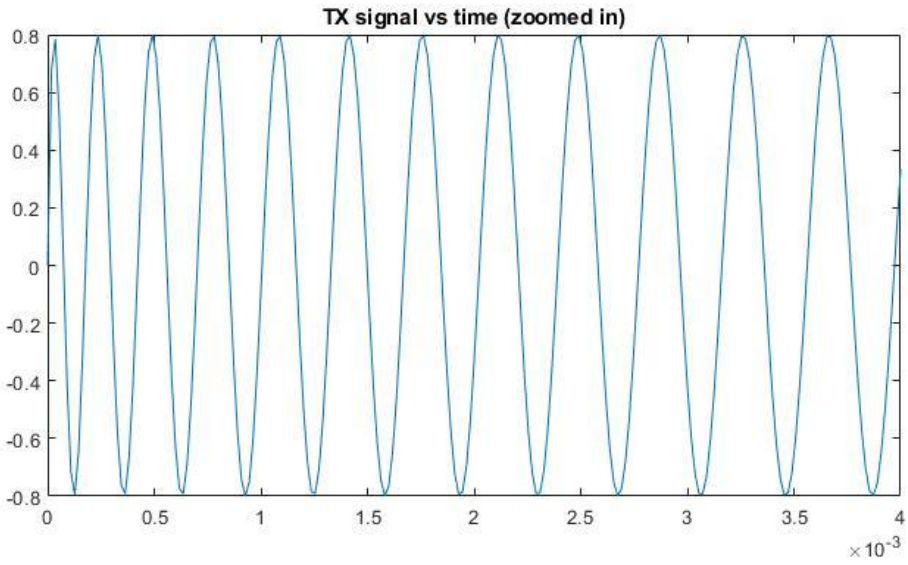
```
clc
clear all
%audio input
[pulseclean, Fs]=audioread("TX signal.wav");
[pulsenoise]=audioread("RX signal.wav");
pulseclean=reshape(pulseclean,[1,length(pulseclean)]);
pulsenoise=reshape(pulsenoise,[1,length(pulsenoise)]);

ns=0:(length(pulseclean)-1);
ts=ns/Fs;

%% problem a
%plotting
figure(1);
subplot(2,1,1);
plot(ts,pulseclean);
title("TX signal vs time");
subplot(2,1,2);
plot(ts,pulsenoise);
title("RX signal vs time");

%% problem b
figure(2);
plot(ts,pulseclean);
xlim([0 0.004]);
title("TX signal vs time (zoomed in)");

%% problem c
%cross co relation
rnc=xcorr(pulsenoise,pulseclean);
Lnc=length(rnc);
n=(Lnc-1)/2;
nnc=-n:n;
```

```
%plotting
figure(3);
plot(nnc/Fs,rnc);
ttile("Cross corelation vs time")

%delay calculation
[max,ind]=max(rnc);
delay=nnc(ind)/Fs;
velocity=305;
distance=0.5*velocity*delay;
fprintf("The distance of the bird is %f meter\n",distance);

%% problem d
SNR=-22;
sigPower=sum(pulseclean.^2)/length(pulseclean);
noisePower=sigPower/(10^(SNR/10));
noisePowerdB=10*log10(noisePower);
fprintf("The Noise Power in dB is %f dB\n",noisePowerdB);
```

Plot for problem a:



Plot for problem b:

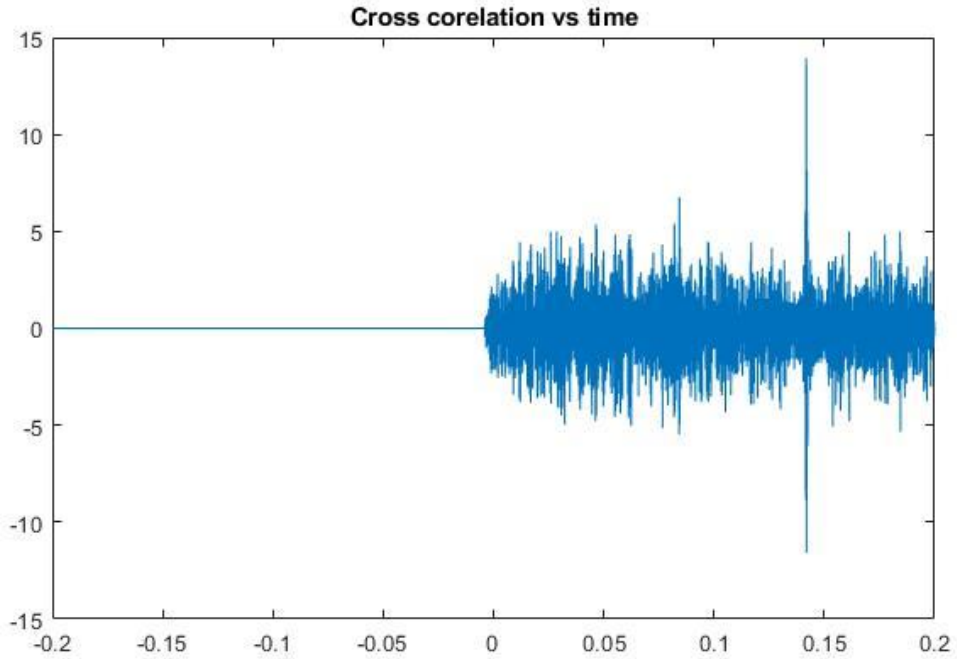


Discussion:

Here, we first plotted the whole of TX and RX signals and then plotted for the nonzero time duration of TX signal using the function xlim.

Xlim limits the value on x-axis in the plot to the defined value.

Plot for problem c:



Command window output:

```
The distance of the bird is 21.621727 meter
>>
```

Discussion:

Similar to the problem in number 2, we used cross correlation between TX and RX signals to get the time delay between two. We can see the max value is very high in the plot for a certain time. That is our delay. Using the delay and the given sound velocity for the day, we could get the distance of the bird easily.

Command window output for Problem d:

```
The Noise Power in dB is 0.041006 dB
>>
```

Discussion:

We know that SNR = Signal power/Noise power

We were given the value of SNR. Then we used the TX signal to calculate signal power.

Using these two we could easily calculated noise power and express it in dB later.

Problem 5

5. Complete the following tasks from the lab-sheet:

   (i) Estimation of impulse response
   (ii) Signal smoothing by a moving average (MA) system
   (iii) Detection of a transmitted sequence

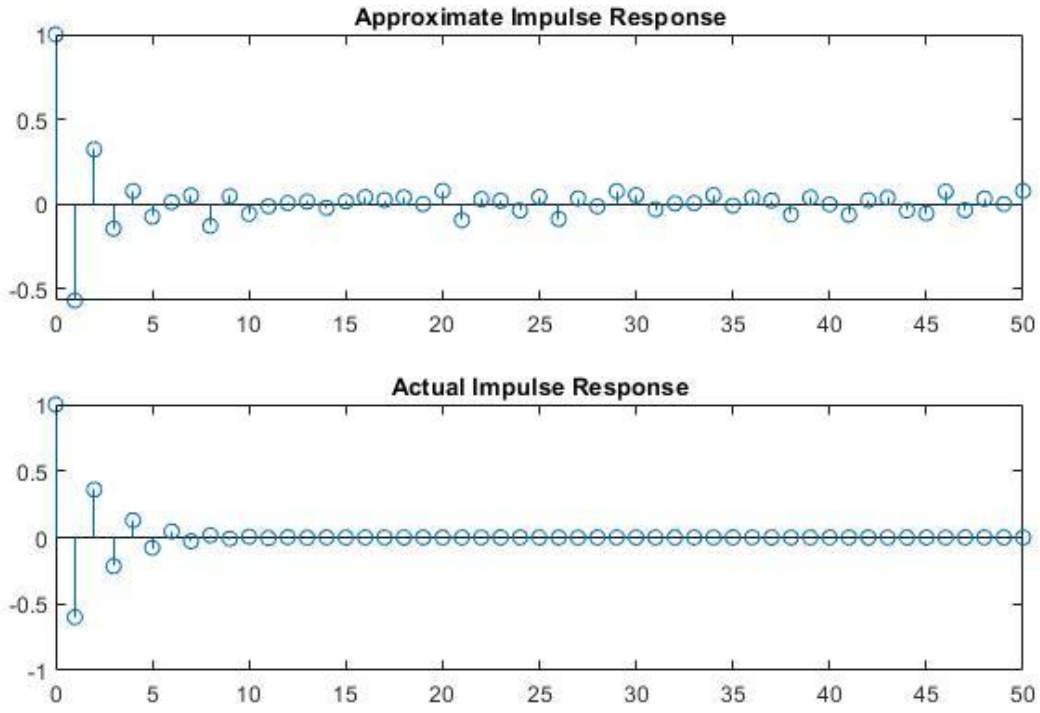
Comment on how you can estimate the response in (i) and smoothen the signal in (ii) better.

Part a

Code:

```
clc
clear all

N=500;
nr=0:499;
ny=nr;
r=randn(1,N);
y=zeros(size(r));
for n=2:500;
    y(n)=r(n)-0.6*y(n-1);
end
rr=fliplr(r);
nrr=-fliplr(nr);
Ryr=conv(y,rr);
kmin=(ny(1)+nrr(1));
kmax=(ny(length(ny))+nrr(length(nrr)));
k=kmin:kmax;

%plotting
subplot(211)
stem(k,Ryr/Ryr(N));
title("Approximate Impulse Response");
xlim([0 50]);

b=[1 0];
a=[1 0.6];
n=0:499;
x=zeros(size(n));
x(1)=1;
yy=filter(b,a,x);
subplot(212);
stem(n,yy);
title("Actual Impulse Response");
xlim([0 50]);
```

Plot:



Discussion:

We can get the impulse response of a given system by correlation process. If we correlate the white noise input with the output of the system corresponding to white noise, we get an approximation of the impulse response of the system.

When we cross correlate the noise with the output y(n) ,

then y(n)*r(-n) = r(n)*x(n)*r(-n)

= r(n)*r(-n)*x(n) ;cumulative property of convulation

= del(n)*x(n)

= h(n) ; which is the impulse response of the system

We used the randn function to generate the noise for us. It is evident from the plot that using this system to get the impulse response gives a slightly deviated result from the actual impulse response.

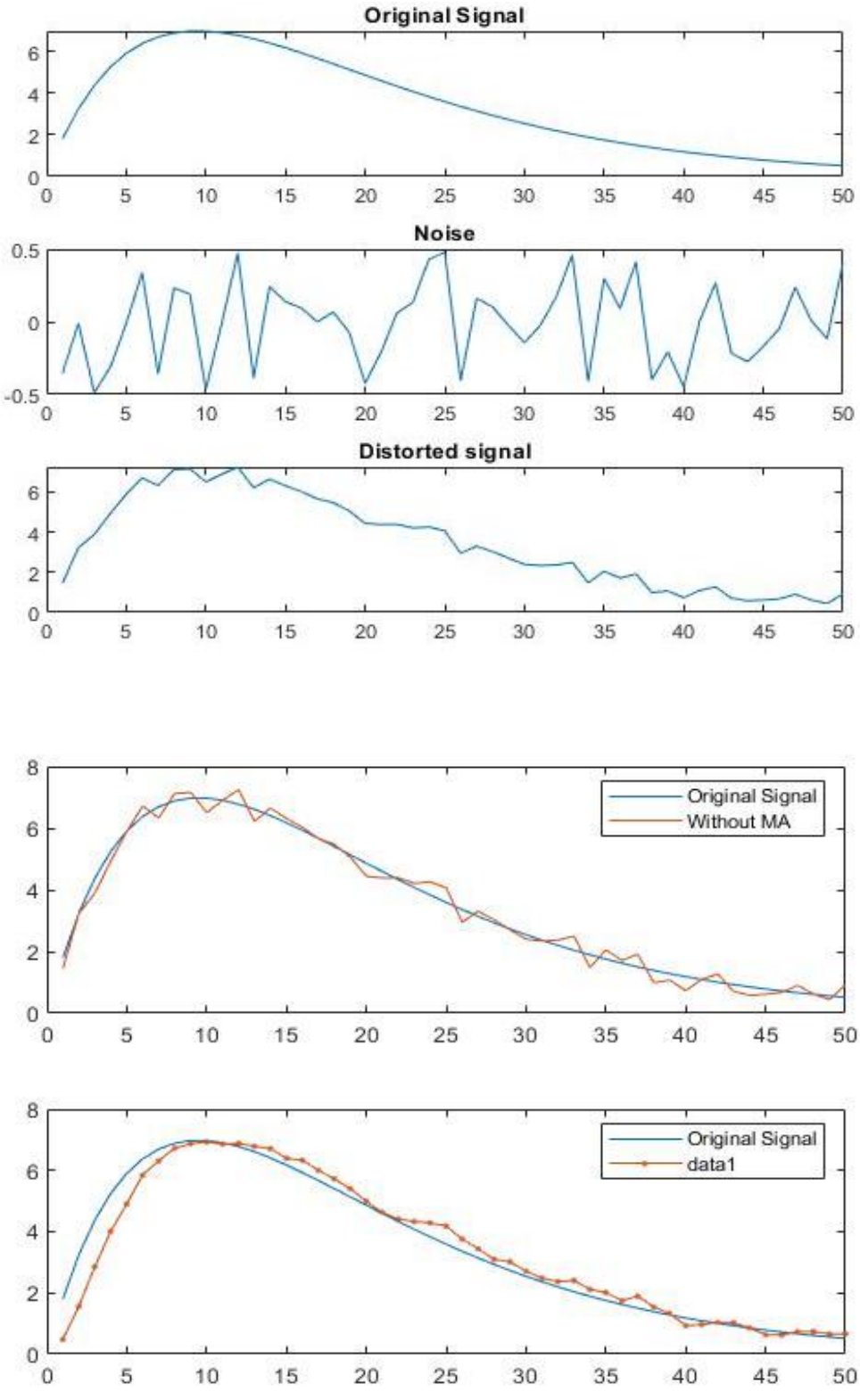
We can overcome the problem of the deviation by simply increasing the number of samples. If we take large number of samples like 3000, the deviation will get insignificant.

Part b

Code:

```
clc
clear all
N=50;
n=1:N;
x=2.*n.*(0.9).^n;
noise=rand(1,N)-0.5;
xnoise=x+noise;
x1=xnoise;

subplot(311)
plot(n,x)
title('Original Signal')
subplot(312)
plot(n,noise)
title('Noise')
subplot(313)
plot(n,xnoise)
title('Distorted signal')

%%
E=3;
z=zeros(1,N-1);
for i=0:(E-1)
    z=[z 0]+x1;
    x1=[0 x1];
end
z=z/E;

%%
subplot(211)
plot(n,x,'DisplayName','Original Signal')
hold on
plot(n,xnoise,'DisplayName','Without MA')
legend
subplot(212)
plot(n,x,'DisplayName','Original Signal')
hold on
plot(n,z(1:length(n)),'.-')
legend
```

Plot:


Original Signal
Noise
Distorted signal
Original Signal
Without MA
Original Signal
data1

Discussion:

In this problem, we used a moving sum system to smoothen our signal. Moving sum generates the average value for a certain index using some other adjacent values. We used previous three values to achieve the moving sum.

We defined a separated matrix to store the sums. As we shifted the signal matrix by left padding it with a zero, we summed it with the z. We right shifted as three times since we wanted MA for previous three values. Completing this using a loop, we then averaged the whole z matrix and got our expected moving sum average value. Moving sum generates a smoother plot since it compensates for the random variations created by the noise.

We can further improve our MA system by taking the average summation of both previous and later elements of any particular value. That way the noise will be suppressed down more.

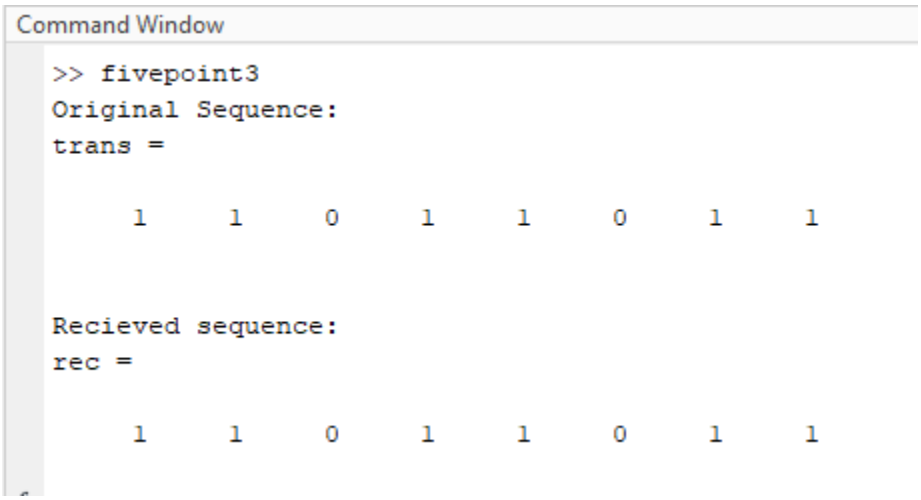
Part c

Code:

```
clear all
trans=[1 1 0 1 1 0 1 1];
unit=ones(1,length(trans));
uf=-unit;
noise=rand(1,length(unit))-0.5;
y=zeros(length(trans),length(unit));
rcorr=zeros(length(trans),2*length(unit)-1);
rec=zeros(1,length(trans));
%%
for i=1:length(trans)
    if(trans(i)==0)
        y(i,:)=unit+noise;
    else
        y(i,:)=uf+noise;
    end
end

for i=1:length(trans)
    rcorr(i,:)=conv(y(i,:),flip(unit));
    if (abs(max(rcorr(i,:)))<abs(min(rcorr(i,:))))
        rec(i)=1;
    else
        rec(i)=0;
    end
end
```

```
%%
n=1:length(trans)
plot(n,rcorr(:,1),'displayName','Correlation');
hold on
plot(n,trans,'DisplayName','Transmitted signal');
hold on
plot(n,rec,'DisplayName','Received Signal');
legend

fprintf('Original Sequence: ');
trans
fprintf('\nRecieved sequence: ');
rec
```

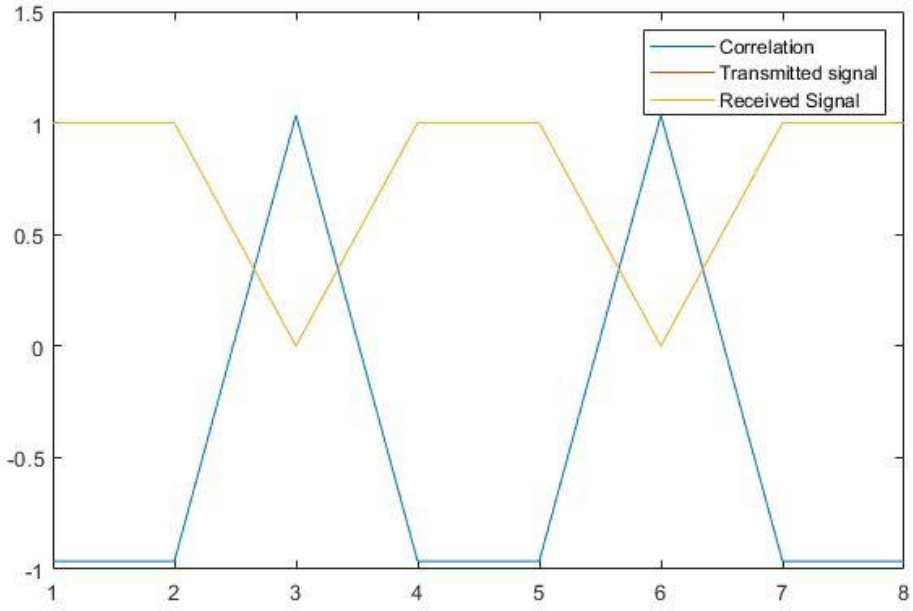
Output:



Discussion:

Here we used correlation to detect a digital transmitted signal. We transmitted a simple pulse sequence with random 0 and 1 values.

In the receiver end, we correlated the received function with known zero and one pulse to identify the signal. For zero pulse, the cross correlation will be of low value in the negative range with one, and of high value in the positive range with zero. Thus, we can identify whether it was originally a zero or one.



As we can see the transmitted and received signals overlap perfectly. Correlation is low for one pulse and high for zero pulse.

Conclusion:

In this experiment, we learnt about correlation and its uses. It can be used in variety of applications and also can be done by the convolution operation. Correlating sequences make it easier to deal with unwanted noise. Thus, this is indeed a very powerful tool.