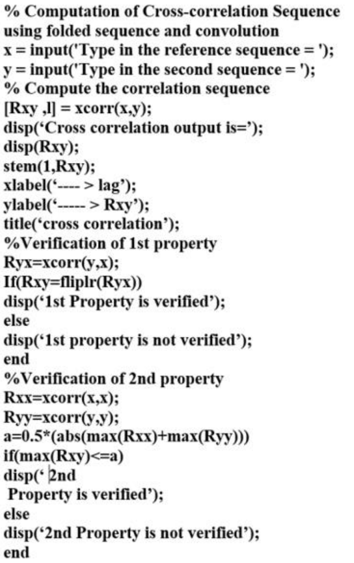
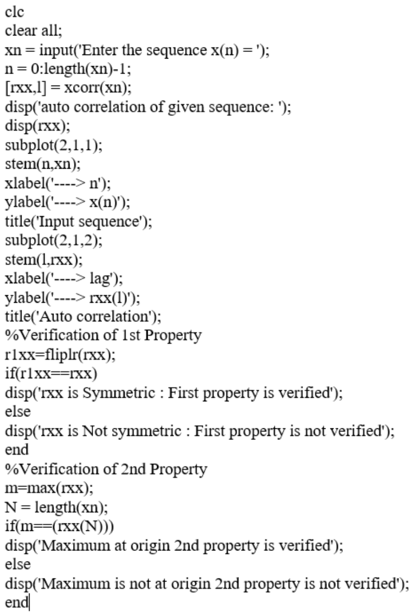
```
tfinal=0.05;
t=0:0.00005: tfinal;
fd=input('Enter analog frequency');
%define analog signal for comparison
xt=sin(2*pi*fd*t);
%simulate condition for under sampling i.e., fs1<2*fd
fs1=1.3*fd;
%define the time vector
n1=0:1/fs1: tfinal;
%Generate the under sampled signal
xn=sin(2*pi*n1*fd);
%plot the analog & sampled signals
 subplot(3,1,1);
plot(t,xt,'b',n1,xn,'r*-');
title('under sampling plot');
%condition for Nyquist plot
fs2=2*fd;
n2=0:1/fs2:tfinal;
xn=sin(2*pi*fd*n2);
subplot(3,1,2);
 plot(t,xt,'b',n2,xn,'r*-');
title('Nyquist plot');
%condition for oversampling
fs3=5*fd;
n3=0:1/fs3:tfinal;
xn=sin(2*pi*fd*n3);
subplot(3,1,3);
plot(t,xt,'b',n3,xn,'r*-');
title('Oversampling plot');
xlabel('time');
ylabel('amplitude');
legend('analog', 'discrete')
```

```
%To find Impulse Response
N=input('Length of response required=');
b=[1]; %x[n] coefficient
a=[1,-1,0.9];   %y coefficients
%impulse input
x=[1,zeros(1,N-1)];
%time vector for plotting
 n=0:1:N-1;
%impulse response
h=filter(b,a,x);

%plot the waveforms
subplot(2,1,1);
stem(n,x);
title('impulse input');
xlabel('----->   n');
ylabel('---->  x(n)');
subplot(2,1,2);
stem (n,h);
title ('impulse response'); xlabel('--->   n');
ylabel('---> h(n)');
```

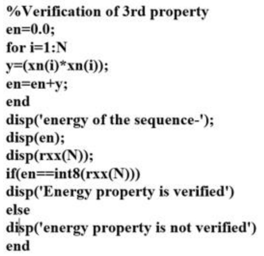


```
% Computation of Cross-correlation Sequence
using folded sequence and convolution
x = input('Type in the reference sequence = ');
y = input('Type in the second sequence = ');
% Compute the correlation sequence
[Rxy ,l] = xcorr(x,y);
disp('Cross correlation output is=');
disp(Rxy);
stem(1,Rxy);
xlabel('---- > lag');
ylabel('----- > Rxy');
title('cross correlation');
%Verification of 1st property
Ryx=xcorr(y,x);
If(Rxy=fliplr(Ryx))
disp('1st Property is verified');
else
disp('1st property is not verified');
end
%Verification of 2nd property
Rxx=xcorr(x,x);
Ryy=xcorr(y,y);
a=0.5*(abs(max(Rxx)+max(Ryy)))
if(max(Rxy)<=a)
disp(' 2nd
 Property is verified');
else
disp('2nd Property is not verified');
end
```

```
%To find step response
clc;
clear;

 N=input('Length of response required=');
 b=[1];              %x[n] coefficient
a=[1,-1,0.9];       %y coefficients
x=[ones(1,N)]; %step input
 n=0:1:N-1;           %time vector for plotting
y=filter(b,a,x);       %step response
 %plot the waveforms
subplot(2,1,1);
stem(n,x);
title('step input');
xlabel('n');
ylabel('u(n)');
 subplot(2,1,2);
stem(n,y);
title('step response');
xlabel('n');
ylabel('y(n)');
Result:
Length=100
```



```
clc
clear all;
xn = input('Enter the sequence x(n) = ');
n = 0:length(xn)-1;
[rxx,l] = xcorr(xn);
disp('auto correlation of given sequence: ');
disp(rxx);
subplot(2,1,1);
stem(n,xn);
xlabel('----> n');
ylabel('----> x(n)');
title('Input sequence');
subplot(2,1,2);
stem(l,rxx);
xlabel('----> lag');
ylabel('----> rxx(l)');
title('Auto correlation');
%Verification of 1st Property
r1xx=fliplr(rxx);
if(r1xx==rxx)
disp('rxx is Symmetric : First property is verified');
else
disp('rxx is Not symmetric : First property is not verified');
end
%Verification of 2nd Property
m=max(rxx);
N = length(xn);
if(m==(rxx(N)))
disp('Maximum at origin 2nd property is verified');
else
disp('Maximum is not at origin 2nd property is not verified');
end
```



```
%Verification of 3rd property
en=0.0;
for i=1:N
y=(xn(i)*xn(i));
en=en+y;
end
disp('energy of the sequence-');
disp(en);
disp(rxx(N));
if(en==int8(rxx(N)))
disp('Energy property is verified')
else
disp('energy property is not verified')
end
```