Advanced Digital Signal Processing

Lab Report

Cand No: 137037

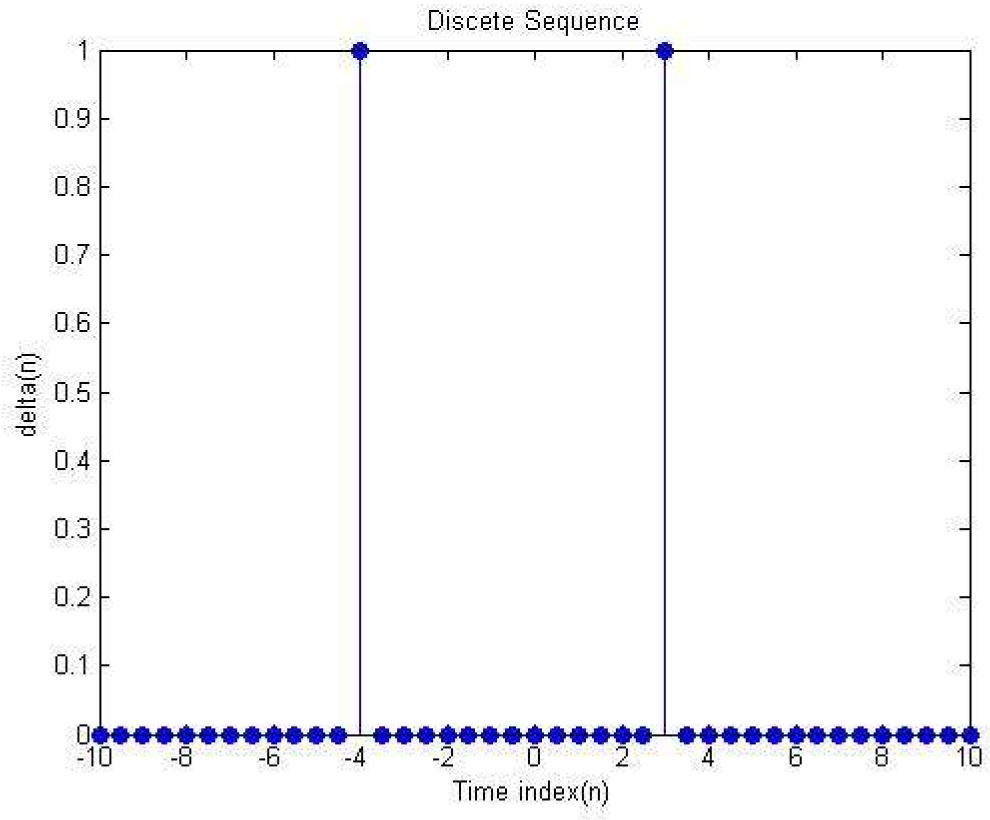
Due: 10/12/2015

Problem 1.a Soln.:

```
%Problem_1a.m
%Candidate No. 137037
%Date Created: 19 November 2015
%Last Modified: 09 December 2015


clear all; close all; clc %Clear all variables, close all figures and clear
the command line.
n=[-10:0.5:10]; %Generate a sequence from -10 to 10 in steps of 0.5
delta=(n==3)+(n==-4);
stem(n,delta,'filled'); %plot n against delta
xlabel('Time index(n)');
ylabel('delta(n)');
title('Discete Sequence')
```

Discussion:

The Dirac delta function is zero everywhere apart from where x=0. As we can see below the code implements the discrete sequence δ(n-3)+ δ(n+4). It achieves by placing all points apart from n=3 and n=-4 to zero.

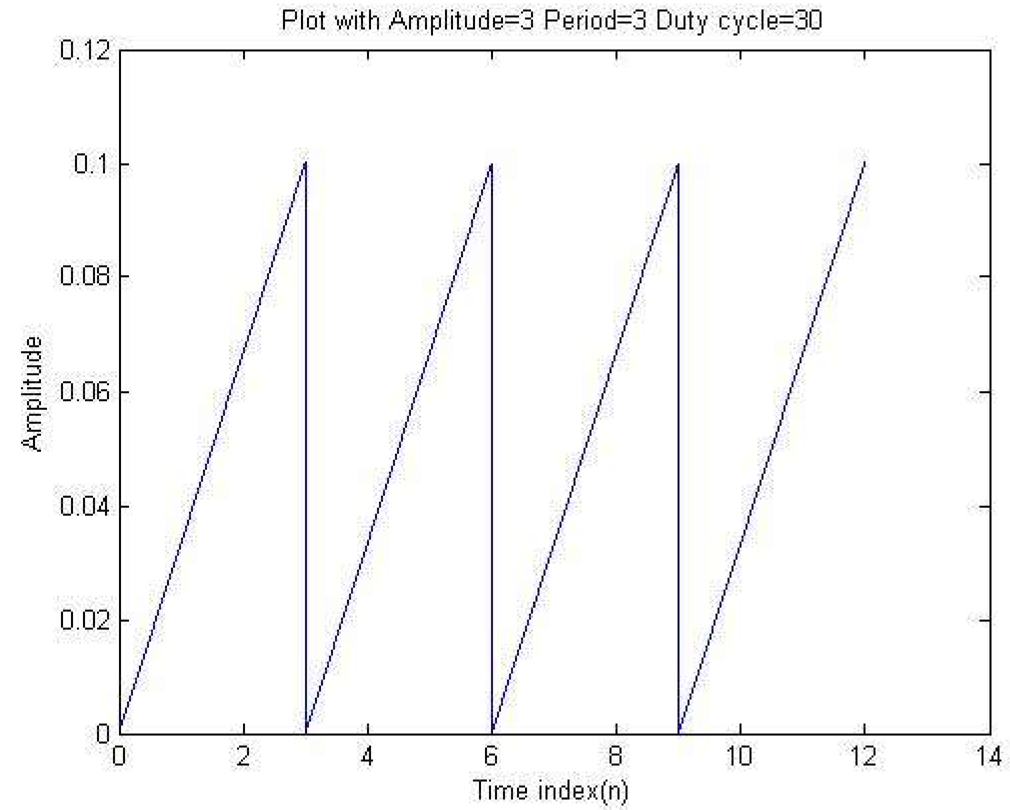


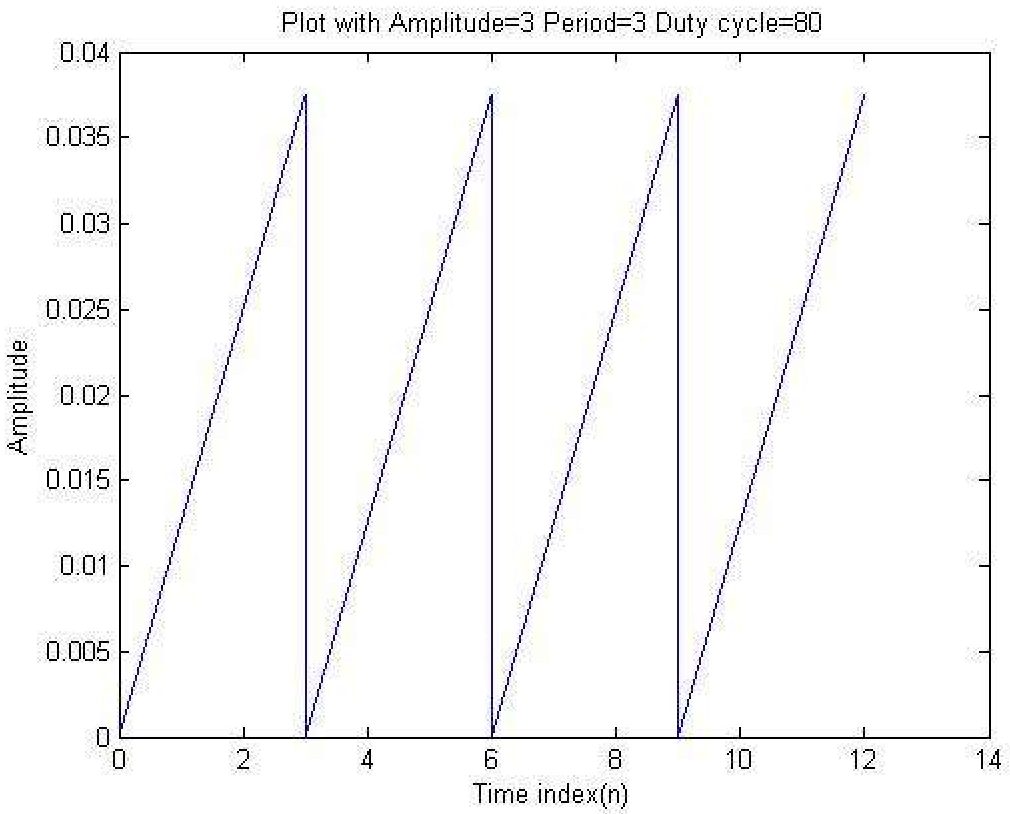
Problem 1.b. Soln.:

```
%Problem_1b.m
%Candidate No. 137037
%Date Created: 27 November 2015
%Last Modified: 09 December 2015


clear all; close all; clc; %Clear all variables, close all figures and clear
the command line.
A=input('Enter amplitude:'); %Specify the amplitude
T=input('Enter the period:'); %Specify the period
D=input('Enter the duty cycle:'); %Specify the duty cycle

tau=T*D; %Time signal is active
t(1)=0;
i=1;
l=0;
C=T;

%Generate Sawtooth shape
for j=1:4
    while t(i)<C
        i=i+1;
        t(i)=t(i-1)+1e-3;
        if t(i)<tau;
            x(i)=A*t(i-l)/(tau-T*(j-1));
        else
            x(i)=0;
        end
    end
    l=i-1;
    C=C+T;
    tau=tau+T;
end
%Plot Sawtooth
str=sprintf('Plot with Amplitude=%d Period=%d Duty cycle=%d',A,T,D);
plot(t,x)
xlabel('Time index(n)')
ylabel('Amplitude')
title(str)
```

Discussion:

Below are shown two graphs which demonstrate how the time active for a sawtooth signal is modified by the duty cycle.


Plot with Amplitude=3 Period=3 Duty cycle=30
Amplitude
Time index(n)
0.12
0.1
0.08
0.06
0.04
0.02
0
0
2
4
6
8
10
12
14


Plot with Amplitude=3 Period=3 Duty cycle=80
Amplitude
Time index(n)
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
2
4
6
8
10
12
14

Problem 2. Soln.:

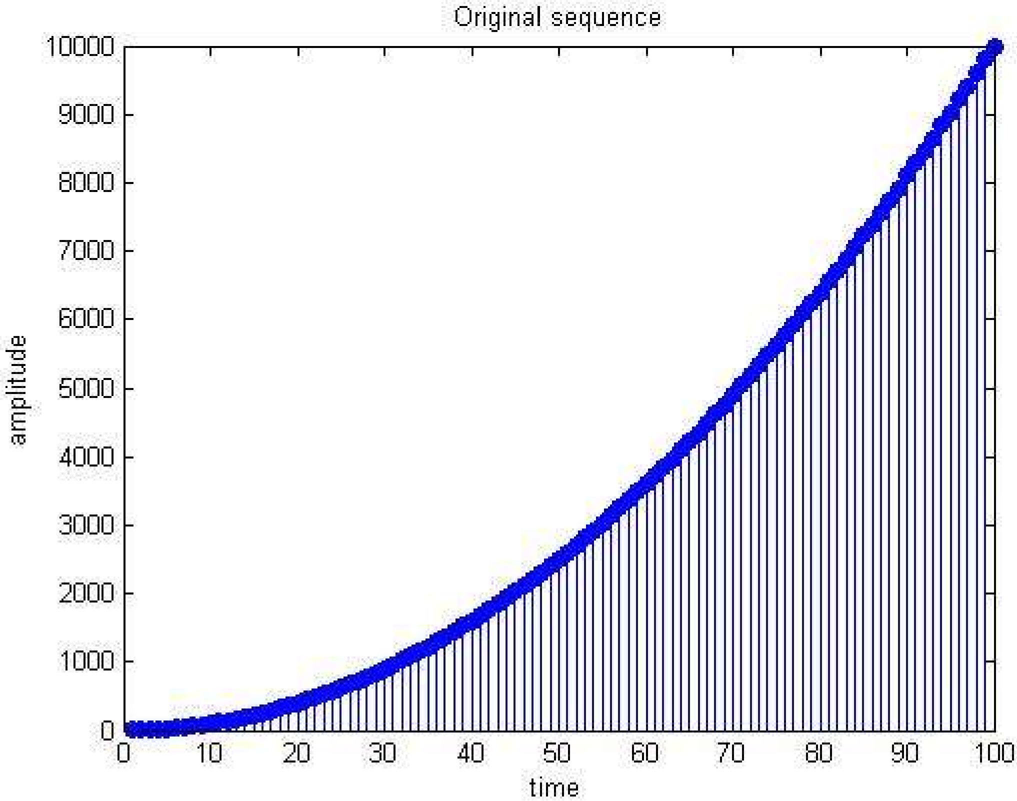
```
%autocorrelate
%Candidate No. 137037
%Date Created: 27 November 2015
%Last Modified: 09 December 2015


function [AC] = autocorrelate(x);


lengthx=length(x); %Calculate length of sequence
x_rev= fliplr(x); %flip the vector x
AC=zeros(1,2*lengthx-1); %define variable AC
for n=1:lengthx %autocorrelation loop
    for k=1:lengthx
        AC(n+k-1)=AC(n+k-1)+x(n)*conj(x_rev(k))/(lengthx);
    end
end

%plot original sequence
figure
stem(real(x),'filled')
title('Original sequence')
xlabel('time')
ylabel('amplitude')


%Plot auto correlation
figure
stem(real(AC),'filled')
title('Autocorrelation')
xlabel('time')
ylabel('amplitude')
```

Discussion:

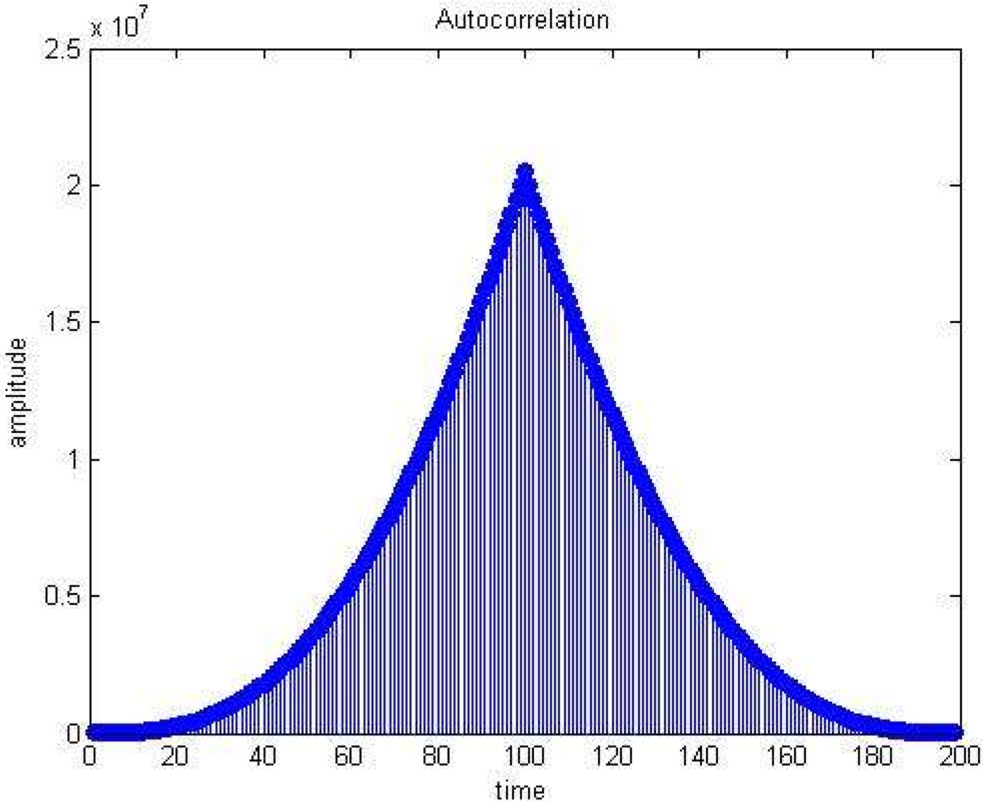
The sequence x which was input into the function was generated as follows:

```
for i=1:100; %for loop to create a sequence
    x(i)=(i)^2;
end

input value x
```

From this input we were able to generate the autocorrelation of the sequence. Both the autocorrelation and the original sequence are shown plotted below.



Original sequence
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
amplitude
0
10
20
30
40
50
60
70
80
90
100
time



Autocorrelation
x 10^7
2.5
2
1.5
1
0.5
0
amplitude
0
20
40
60
80
100
120
140
160
180
200
time

Problem 2.ii Soln.:

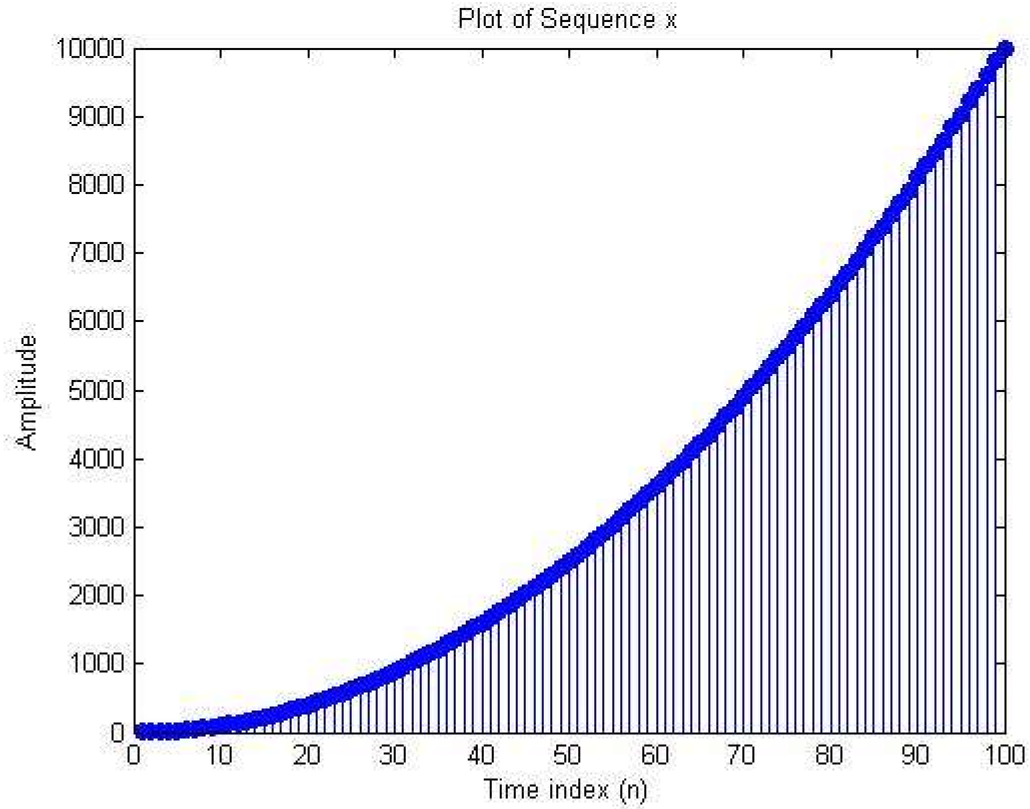
```
%crosscorrelate
%Candidate No. 137037
%Date Created: 27 November 2015
%Last Modified: 09 December 2015

function [CC] = crosscorrelate[x,y]

close all; clc; %Close all figures and clear the command line.

leng_x=length(x); leng_y=length(y);
y_rev= fliplr(y);
CC=zeros(1,leng_x+leng_y-1);% Cross-correlation vector %
for n=1:leng_x
    for k=1:leng_y
     CC(n+k-1)=CC(n+k-1)+x(n)*conj(y_rev(k))/leng_x;
    end
end

% Plot of sequence 1
figure
stem(real(x),'filled');
title('plot of Sequence x');
xlabel('Time index (n)');
ylabel('Amplitude');

% Plot of sequence 2
figure;
stem(real(y),'filled');
title('Plot of Sequence y');
xlabel('Time index (n)');
ylabel('Amplitude');

%Plot cross-correlation sequence
figure;
stem(real(CC),'filled');
title('Cross-Correlation of Sequence x WITH Sequence y');
xlabel('Time index (n)');
ylabel('Amplitude');
```

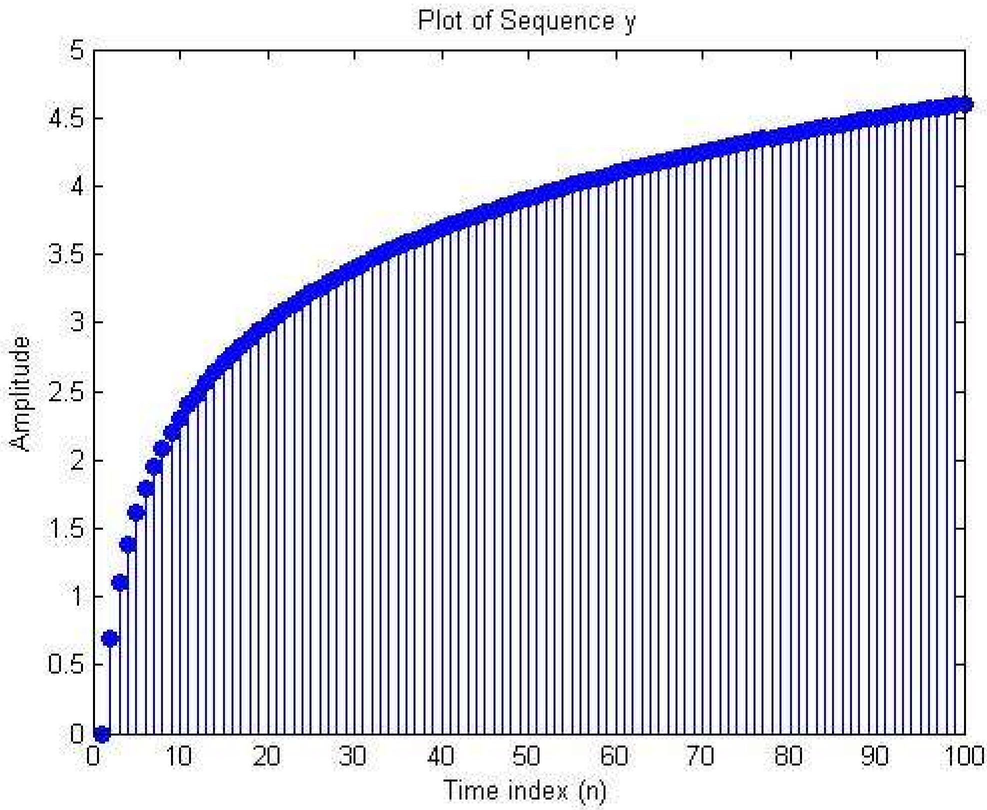
Discussion:

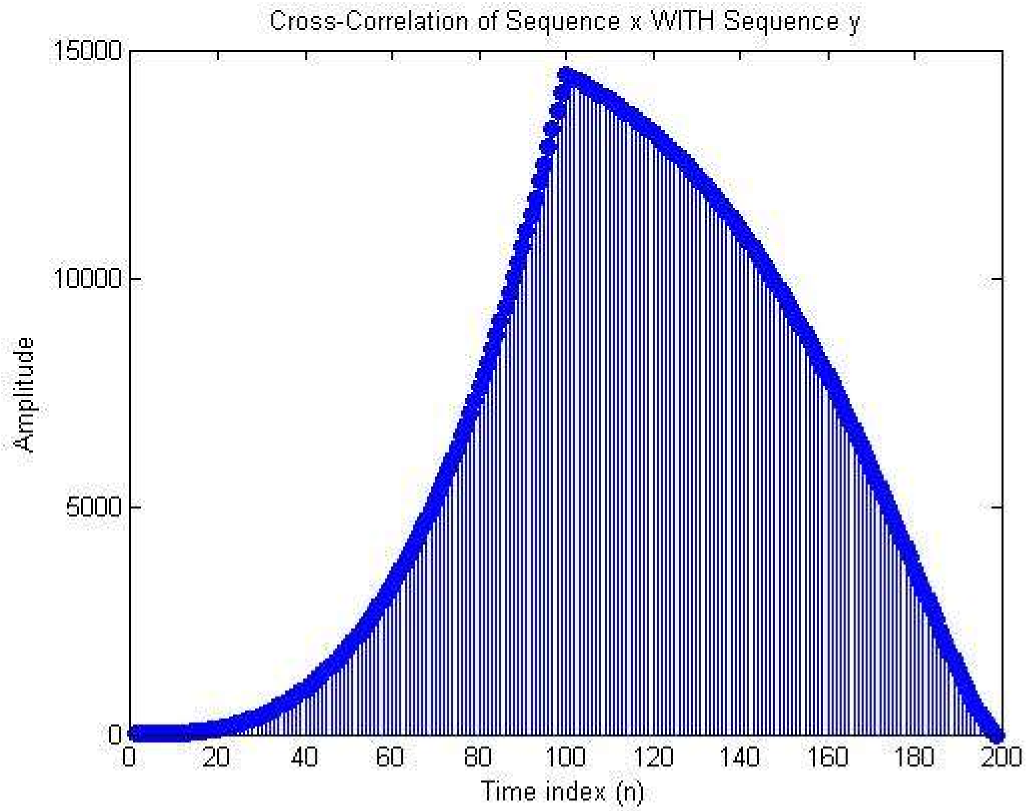
The sequence x which was input into the function was generated as follows:

```
for i=1:100; %for loop to create x sequence
    x(i)=(i^2);
end


for i=1:100; %for loop to create y sequence
    y(i)=sin(i);
end
```

From this input we were able to generate the crosscorrelation of the sequences x and y. Both the crosscorrelation and the original sequences are shown plotted below.




Plot of Sequence y
Amplitude
Time index (n)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
10
20
30
40
50
60
70
80
90
100


Cross-Correlation of Sequence x WITH Sequence y
Amplitude
Time index (n)
0
5000
10000
15000
0
20
40
60
80
100
120
140
160
180
200

Problem 3 Soln.:

```
%conv
%Candidate No. 137037
%Date Created: 27 November 2015
%Last Modified: 09 December 2015

function [conv1,conv2]=conv(x,y);
close all; clc;
format long;
clear conv1 conv2 conv3
lx=length(x);
ly=length(y);
n=lx+ly-1;


x_f = fft(x);
y_f = fft(y);
%Linear Correlation
conv1=ifft(((x_f)).*y_f);
%Circular Correlation
xz=fft([x zeros(1,ly-1)]);
yz=fft([zeros(1,lx-1) y]);
conv2=ifft(xz.*yz);

conv3 = conv2(1:100)+conv2(100:end);
%plot sequence 1
stem(x,'filled')
ylabel('amplitude')
xlabel('time')
title('Plot sequence x')


%plot sequence 2
figure;
stem(y,'filled')
ylabel('amplitude')
xlabel('time')
title('Plot sequence y')

%Plot Correlated Sequence without Zero Supplementation (Linear)
figure;
stem(conv1,'filled')
ylabel('amplitude')
xlabel('time')
title('Plot of Correlated Sequence without Zero Supplementation (Linear)')

%Plot Correlated Sequence with Zero Supplementation (Circular)
figure;
stem(conv2,'filled')
ylabel('amplitude')
xlabel('time')
title('Plot of Correlated Sequence with Zero Supplementation (Circular)')

%Plot of summed circular correlation
figure;
plot(1:100,conv3)
hold on
plot(1:100,conv1,'r')
```

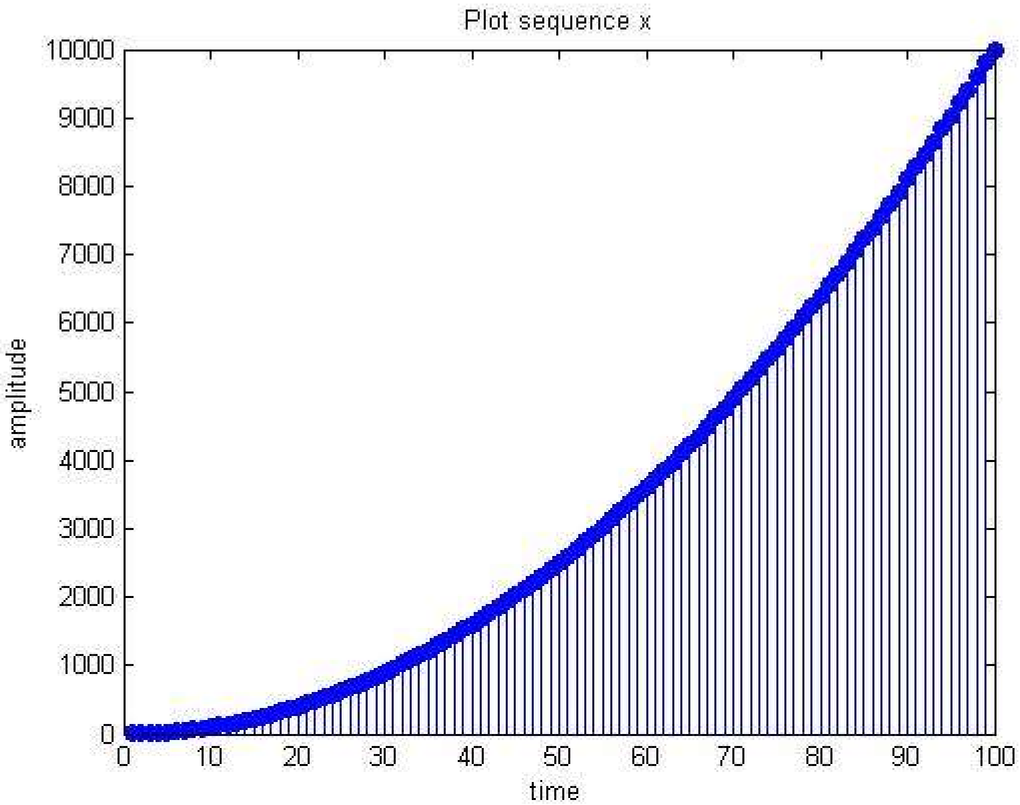
```
ylabel('amplitude')
xlabel('time')
title('Plot of summed circular correlation and linear correlation')
```

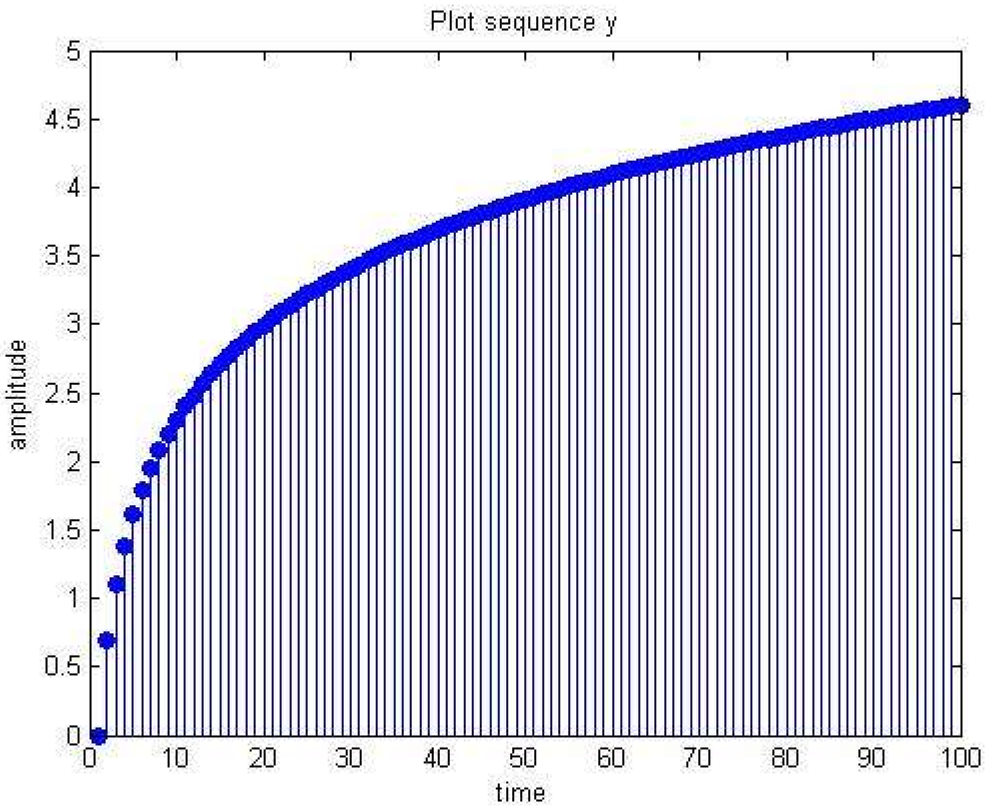
Discussion:

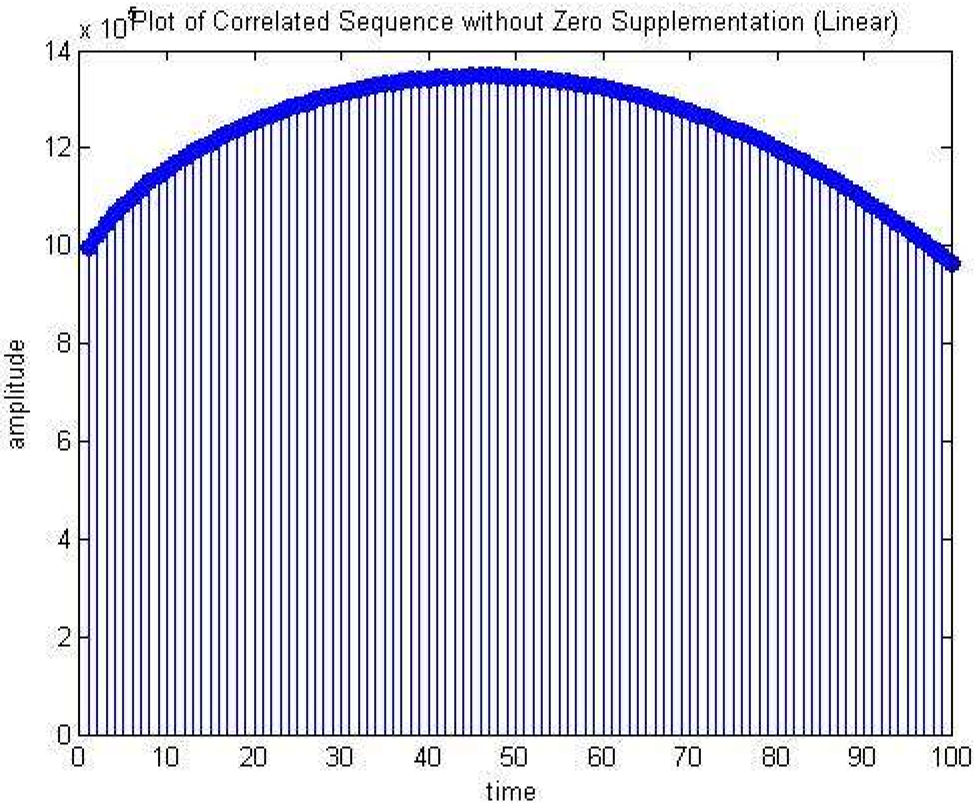
The sequence x which was input into the function was generated as follows:

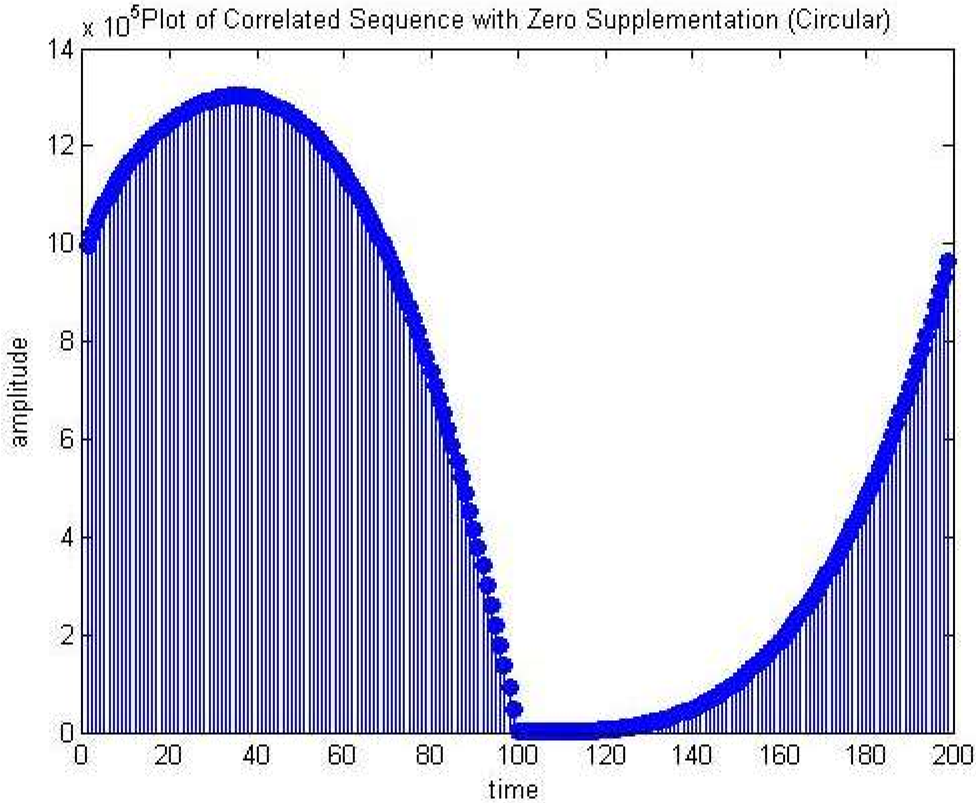
```
for i=1:100; %for loop to create x sequence
    x(i)=(i^2);
end


for i=1:100; %for loop to create y sequence
    y(i)=log(i);
end
```

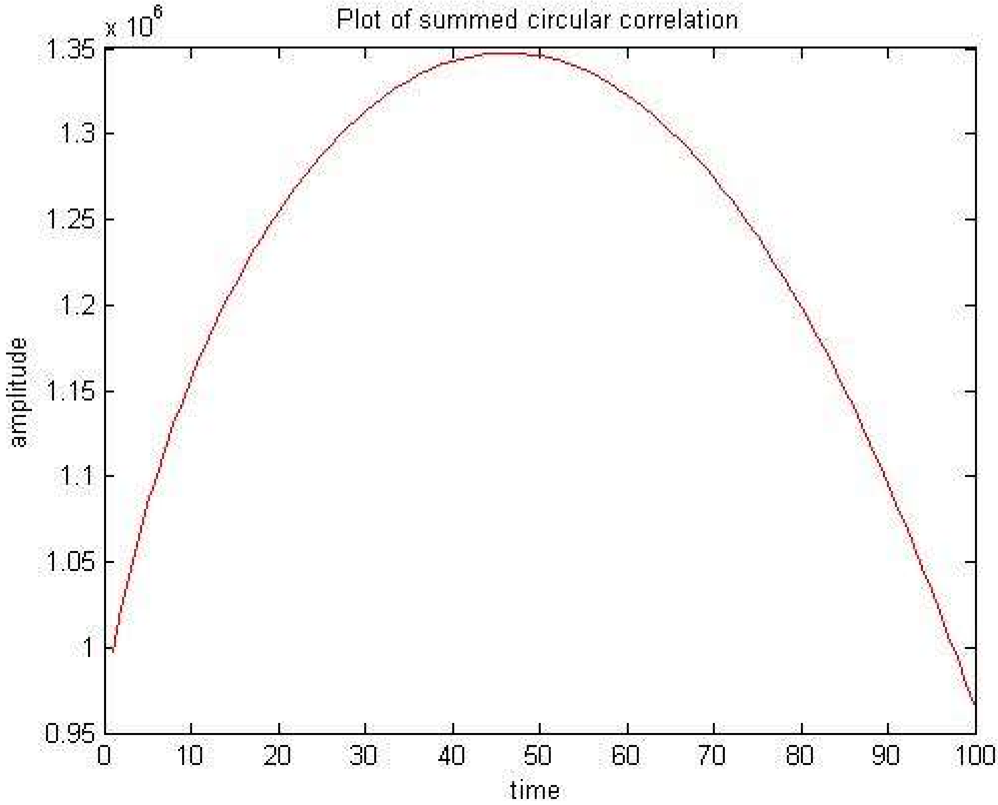
From these inputs we were able to plot the Circular and Zero supplemented DFT to be implemented. These are shown below. Also plotted are the original sequences. The final plot shows a comparison between the summed circular correlation and linear correlation and shows them both to be the same. Circular correlation is as if we put the two sequences on two concentric circles shifting from each other. Multiplying term by term and making a summation to perform the correlation. In the linear case we increase the sampling period to prevent an overlap between sequences correlated and the original sequences. Therefore both achieve the same results.


Plot sequence x
amplitude
time
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
10
20
30
40
50
60
70
80
90
100


Plot sequence y
amplitude
time
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
10
20
30
40
50
60
70
80
90
100


x 10^5
Plot of Correlated Sequence without Zero Supplementation (Linear)
amplitude
time
0
2
4
6
8
10
12
14
0
10
20
30
40
50
60
70
80
90
100


x 10^5
Plot of Correlated Sequence with Zero Supplementation (Circular)
amplitude
time
0
2
4
6
8
10
12
14
0
20
40
60
80
100
120
140
160
180
200

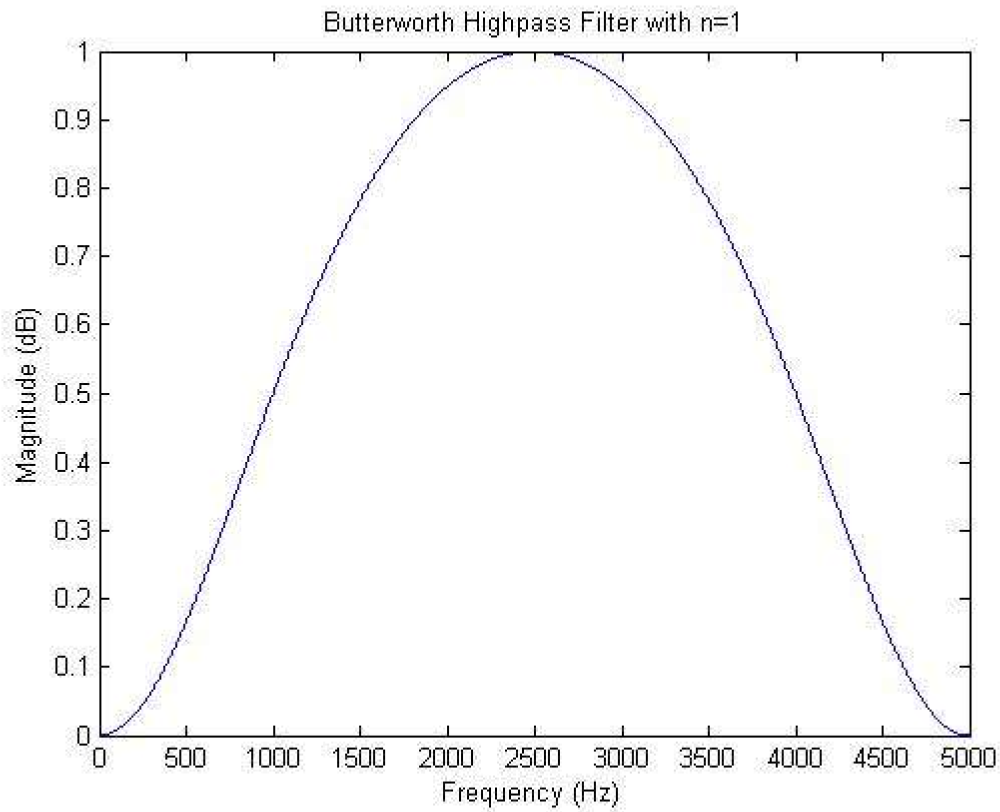

Plot of summed circular correlation
x 10^6
1.35
1.3
1.25
1.2
1.15
1.1
1.05
1
0.95
amplitude
0
10
20
30
40
50
60
70
80
90
100
time

Problem 4 Soln.:

```
%ButterworthHP
%Candidate No. 137037
%Date Created: 27 November 2015
%Last Modified: 09 December 2015


function                                                              [G]=
ButterworthHP(CuttoffFreq,Stopbandedge,Stopbandattenuation,Samplingfreq)

close all; clc;

wsam=Samplingfreq;
T=1/wsam;
wc=(2/T)*tan((T*(CuttoffFreq*2*pi))/2);
wst=(2/T)*tan((T*(Stopbandedge*2*pi))/2);
xdb=(Stopbandattenuation);
n=ceil(log10(sqrt((10^(xdb/10))-1))/log10(wc/wst));


w=(0:wsam);
for Z=1:wsam+1;
    z = exp(1i*2*pi*w(Z)*T);
    sbt=(2/T)*((z-1)/(z+1)); %Bilinear transform
    sn=(sbt^n)/((sbt^n)+((-1)^n)*wc^n);  %Apply filter order
    G(Z)=sn;
    G2(Z)=(0.579*(z-1))/(z-0.1584);

end


str=sprintf('Butterworth Highpass Filter with n=%d',n);
plot(w,abs(G).^2);
title(str)
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
```

Discussion:
The Input values for the filter are shown below. From these we able to calculate the filter order (demonstrated in the code above). By using the bilinear approach we must take into account warping of the frequencies. Therefore we must consider the relationship between the analog system frequency and the digital system frequency. The plot shows the response of the Butterworth Highpass filter. With these choice of input parameters, the response obtained by my code is able to reproduce the response of the signal processing toolbox of MatLab.

Input:
CuttoffFreq=1000Hz
Stopbandedge=350Hz
Stopbandattenuation=10dB
Samplingfreq=5000Hz


Butterworth Highpass Filter with n=1
Magnitude (dB)
Frequency (Hz)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000