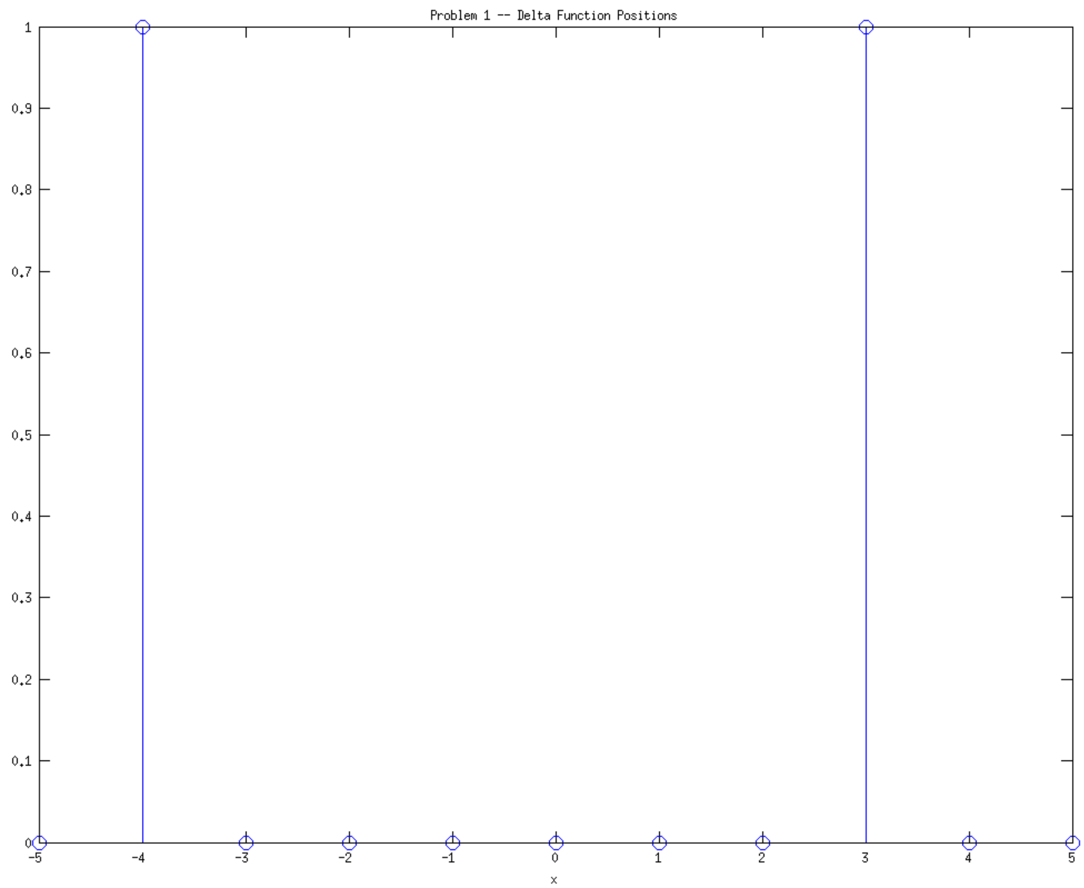
# Advanced Digital Signal Processing Coursework
Candidate Number : 137377

December 9, 2015

# 1 Problem 1a

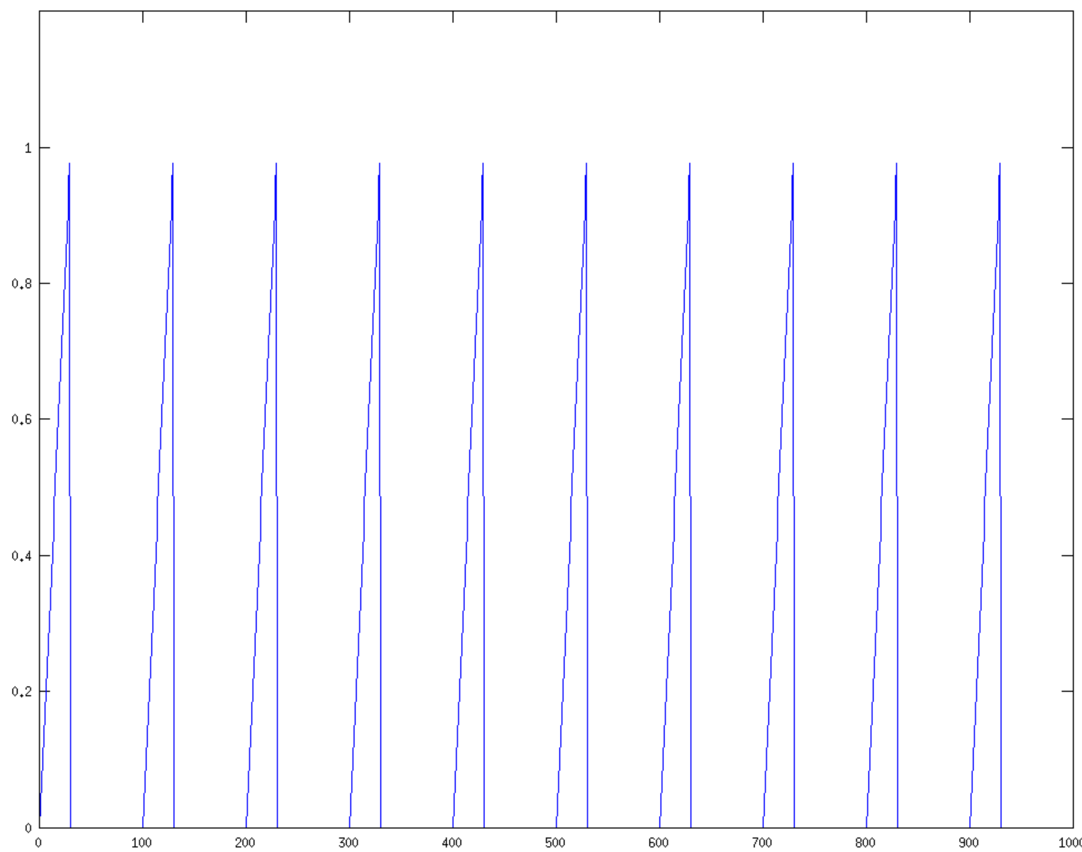
```
% Problem # 1a Soln.:
%------------------------Delta Function Plotting----------------------------
%------------------------
%------------------------Candidate No. 137377 ------------------------------

x=-5:5;
m = zeros(1,size(x,2));
% Delta function at x= -4
m(2) = 1;
% Delta function at x= 3
m(9) = 1;

h = figure;
circshift(m,-4,1);
stem(x,m)
xlabel('x')
title('Problem 1 -- Delta Function Positions')
```

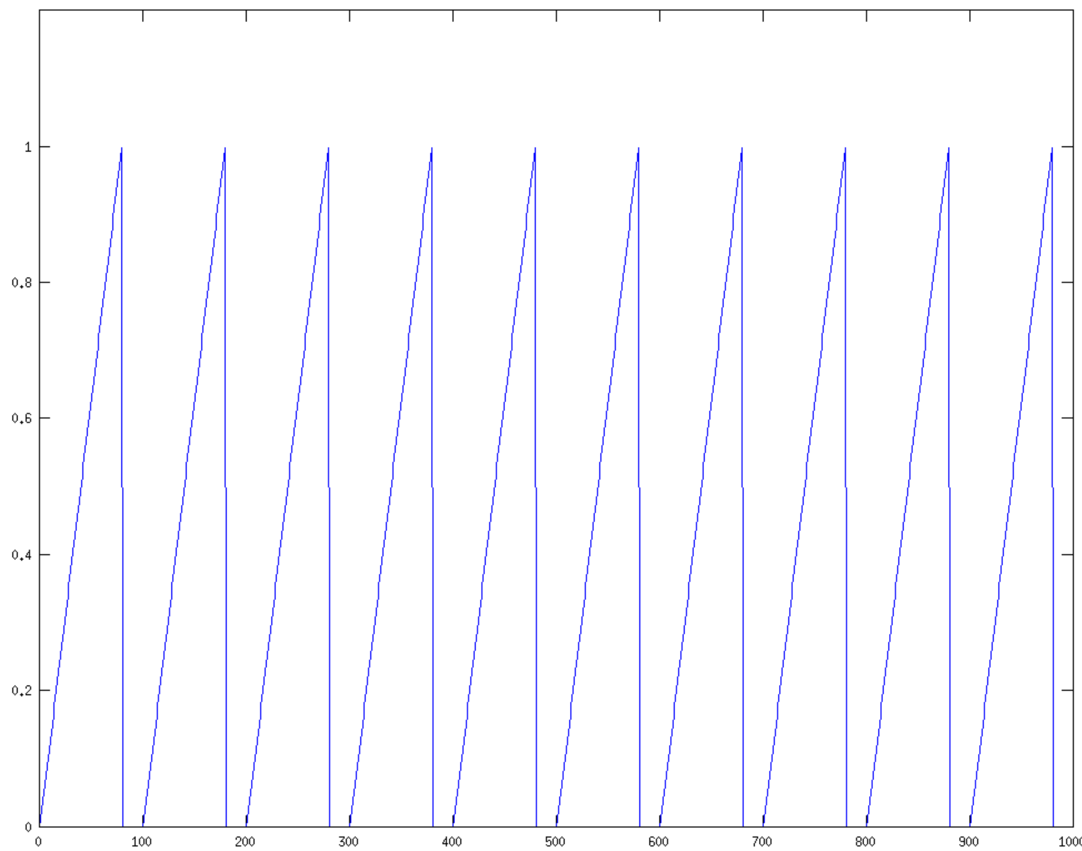

Problem 1 -- Delta Function Positions
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
-5
-4
-3
-2
-1
0
1
2
3
4
5
x

## 2 Problem 1b

```
% Problem # 1b (i & ii) Soln.:
%-------------Sawtooth Function Plotting--------------
%-------------
%-------------Candidate No. 137377 ----------------
%Generate a sawtooth waveform with a chosen duty cycle:
clear all; close all;

%Get the duty cycle percentage. Should be less than 100%.
duty = mod(ceil(input('Enter desired Duty Cycle (%) : '))
    ,100);

%Generate the sawtooth shape
f = @(x) [x/duty.*(0<=x & x<=duty)];
x = linspace(0,100);
y = f(x);

%Number of times to repeat sawtooth in plot
repetitions = 10;
yy = y;
for i=1:repetitions;
yy = cat(2,yy,y);
end

% Plot
h = figure(1);
plot(1:length(yy),yy)
ylim([0 1.2]);
xlim([0 repetitions*100]);
```

Showing output for 30% and 80% duty cycles respectively.

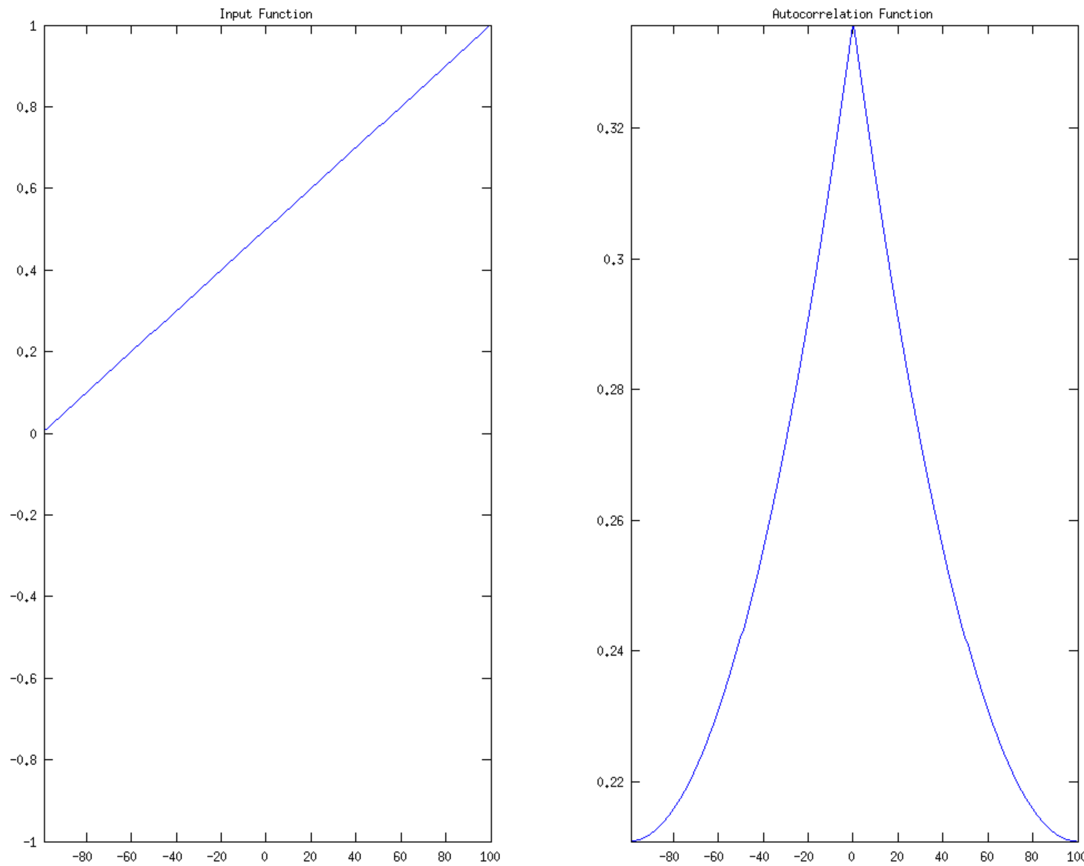




1
0.8
0.6
0.4
0.2
0
0
100
200
300
400
500
600
700
800
900
1000

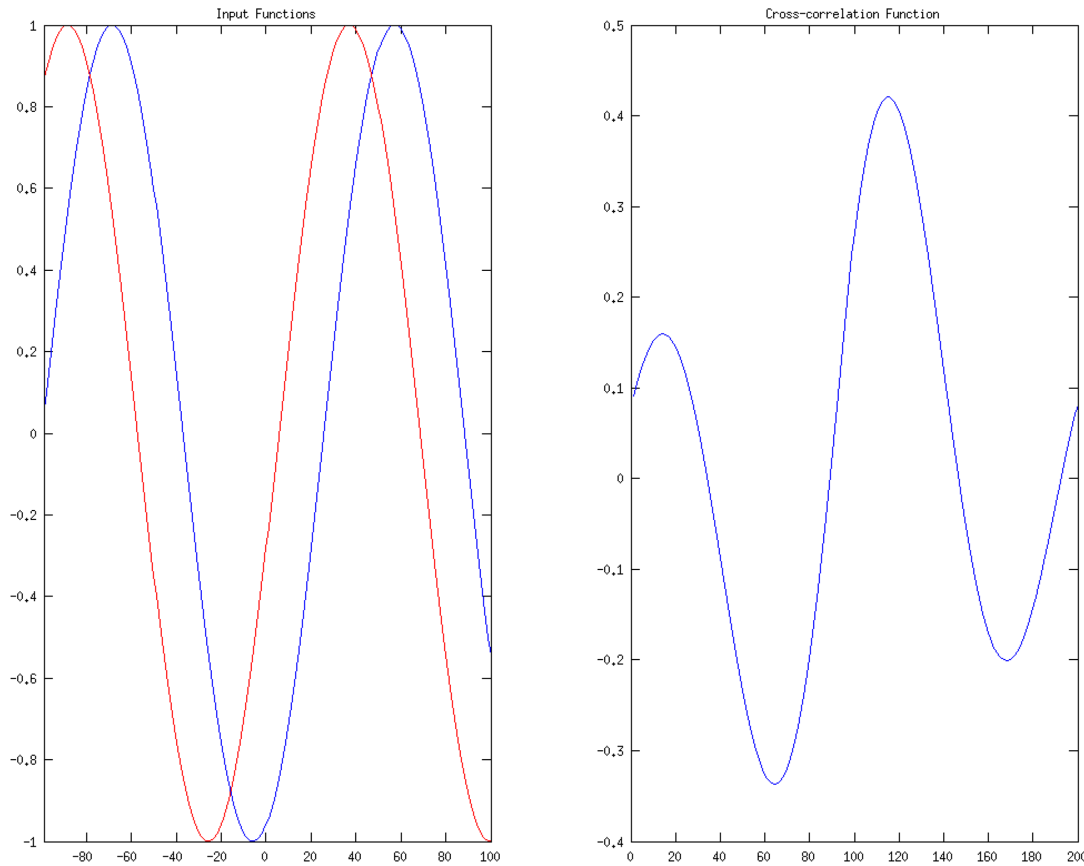
## 3 Problem 2(i)

```
% Problem # 2(i) Soln.:
%--------------Autocorrelation----------------
%--------------
%--------------Candidate No. 137377 ------------------
%Auto-correlate a sequence with itself
clear all; close all;

N = 200;
% Simple Line Input
x = (1:200)/N;


% Initialize array to hold the autocorrelation data
psi = x*0;

for i = 1:N
    xshift = circshift(x,[0,(i + N/2)]);
    psi(i) = N^-1 * sum(x .* xshift);
end

h= figure;
subplot(1,2,1)
plot(-floor(N/2)+1:floor(N/2),x);
title('Input Function');
axis([-floor(N/2)+1 floor(N/2) -1 1]);
subplot(1,2,2)
plot(-floor(N/2)+1:floor(N/2),psi);
title('Autocorrelation Function');
axis([-floor(N/2)+1 floor(N/2) min(psi) max(psi)])
```


Input Function
Autocorrelation Function
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0.32
0.3
0.28
0.26
0.24
0.22
-80
-60
-40
-20
0
20
40
60
80
100

## 4 Problem 2(ii)

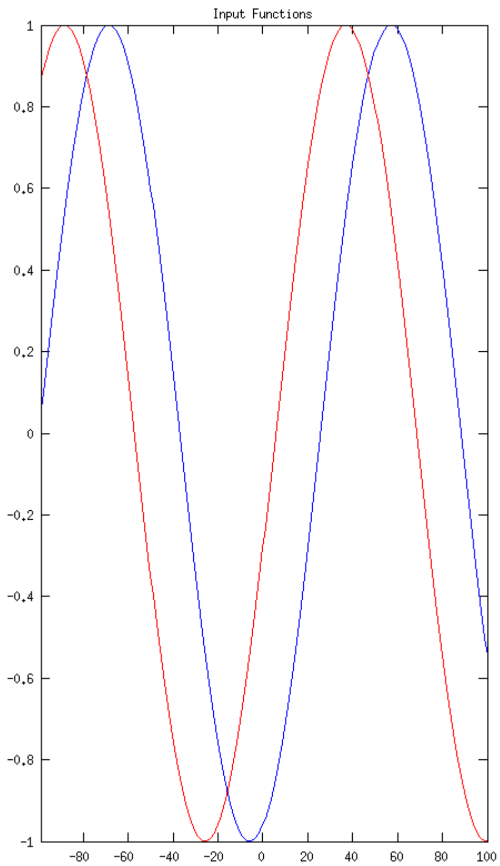
```
% Problem # 2(ii) Soln.:
%------------Cross Correlation--------------
%------------
%------------Candidate No. 137377 ----------------
% Cross correlation of two sequences.
clear all; close all;

N = 200;
% sin-shaped signal -- slowly varying
x(1:N) = sin((1:N) / 20);
y(1:N) = sin(((1:N)+20) / 20);

% Initialize array to hold the correlation data
psi = x*0;

% Apply shift to cross-correlate
for i = 1:N
    yshift = circshift(y,[0,(i + N/2)]);
    psi(i) = N^-1 * sum(x .* yshift);
end

h = figure
subplot(1,2,1)
plot(-floor(N/2)+1:floor(N/2),x,'b');
title('Input Functions');
axis([-floor(N/2)+1 floor(N/2) -1 1]);
hold on;
plot(-floor(N/2)+1:floor(N/2),y,'r');
subplot(1,2,2)
plot(1:size(psi,2),psi);
title('Cross-correlation Function');
```

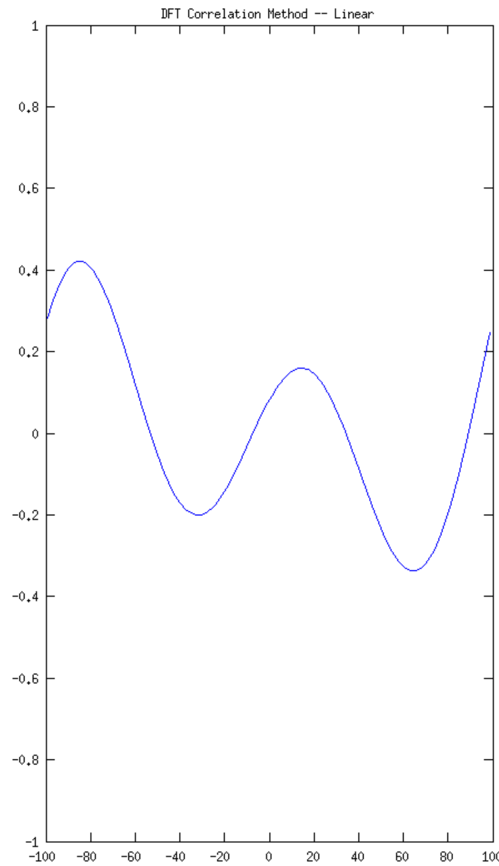

Input Functions
Cross-correlation Function

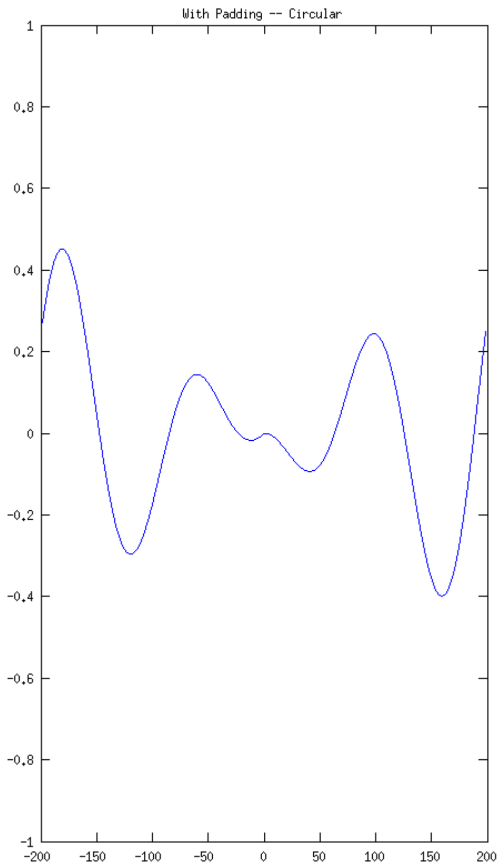
## 5 Problem 3

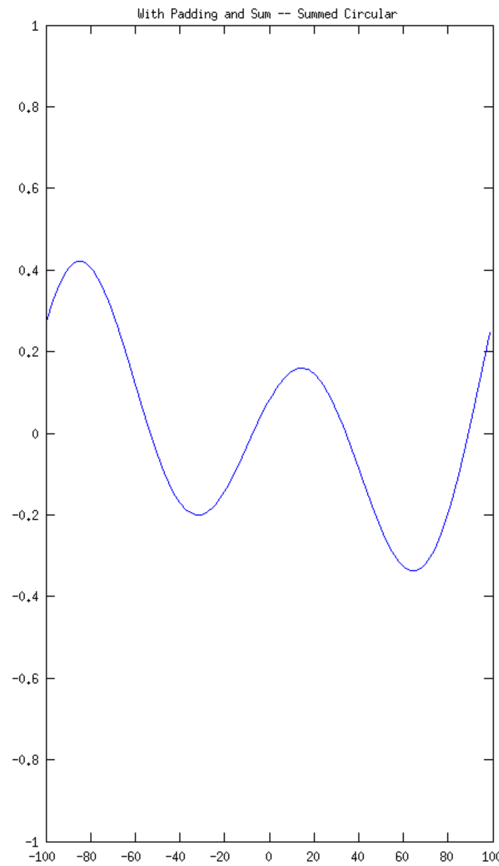
```
% Problem # 3 Soln.:
%-----------------Comparison of Correlation Methods-
%-----------------
%-----------------Candidate No. 137377 ----------------------
%
% Graph 1 shows the input functions, Graph 2 shows the linear correlation, Graph 3 shows the circular correlation, Graph 4 shows the summed version of Graph 3 to show that it is the same as Graph 2.
% Ultimately, the different methods produce the same output in Graphs 2 and
% 4. However, the zero-padding circular method produces an output that is
% periodic and must be summed to produce the final result.

clear all; close all;

N = 200;
% sin-shaped signal -- slowly varying
x(1:N) = sin((1:N)/20);
x_f = fft(fftshift(x));
y(1:N) = sin(((1:N)+20) / 20);
y_f = fft(fftshift(y));

% Correlation without Padding
% Do the multiplication in frequency space
psi_f = x_f .* conj(y_f);
psi_1 = (ifft(psi_f))/N;

% Correlation with Zero Padding
% Pad arrays to size N >= N_1 + N_2 - 1
pad_size = size(x,2) + size(y,2) - 1;
x_padded = zeros(1,pad_size); x_padded(size(x,2):end) = x(1:end);
y_padded = zeros(1,pad_size); y_padded(1:size(y,2)) = y(1:end);
x_padded_f = fft(fftshift(x_padded));
y_padded_f = fft(fftshift(y_padded));

y_save = x_padded_f .* conj(y_padded_f);
y_save = ifftshift(ifft(y_save));

```

```
psi_2 = y_save / N;
psi_2_sum = (y_save(1:N) + y_save(N:end)) / N;


h = figure
subplot(1,4,1)
plot(-floor(N/2)+1:floor(N/2),x,'b');
title('Input Functions');
axis([-floor(N/2)+1 floor(N/2) -1 1]);
hold on;
plot(-floor(N/2)+1:floor(N/2),y,'r');
subplot(1,4,2)
plot(-N/2:N/2-1,psi_1);
title('DFT Correlation Method -- Linear');
axis([-N/2 N/2 -1 1])
subplot(1,4,3)
plot(-N+1:N-1,psi_2); hold on;
title('With Padding -- Circular')
axis([-N N -1 1])
subplot(1,4,4)
plot(-N/2:N/2-1,psi_2_sum);
title('With Padding and Sum -- Summed Circular')
axis([-N/2 N/2 -1 1])
```


Input Functions
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-80
-60
-40
-20
0
20
40
60
80
100


DFT Correlation Method -- Linear
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-100
-80
-60
-40
-20
0
20
40
60
80
100


With Padding -- Circular
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-200
-150
-100
-50
0
50
100
150
200


With Padding and Sum -- Summed Circular
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-100
-80
-60
-40
-20
0
20
40
60
80
100

## 6 Problem 4

```
% Problem # 4 Soln.:
%-------------IIR Filter Design--------------
%------------
%-------------Candidate No. 137377 ----------------
% Computes the frequency response through the bilinear transform method of
% a high-pass IIR filter.
% See report for the derivation of G.

clear all; close all;

% point_1_f = input('Enter desired -3dB cut-off frequency : ')
% point_2 = input('Enter next attenuation point (dB): ')
% point_2_f = input('Enter desired frequency at the next attenuation point : ')
% sample_rate = input('Enter sampling rate : ')

% Inputs used for graph in report
point_1_f = 1000;
point_2 = 10;
point_2_f = 350;
sample_rate = 5000;

T = 1/sample_rate;
w = 2*pi*(0:sample_rate/2);
w_ac = (2/T) * tan(2*pi * point_1_f * T / 2);
w_att = (2/T) * tan(2*pi * point_2_f * T /2);

filter_order = round(log10(10^(point_2/10) - 1) / (2*log10(w_ac/w_att)));
disp(['Filter has order: ', num2str(filter_order)]);

% Generate range of frequencies based on sample rate
z = exp(1i*w*T);
% Apply substitution for s for the type of filter, and the bilinear
% transform
s = ((2/T) .* ((z-1)./(z+1))).^filter_order;
G = s ./ (s + (-1 ^ filter_order) * (w_ac)^filter_order);

% High-pass filter:

```

```
% Plot up until operating range of filter at z=-1
h = figure;
plot(w./(2*pi),abs(G).^2);
xlabel('Frequency');
ylabel('|G(f)|^2');
title('Frequency Response of Filter')
```

The following were derived for use in the program:

$$\text{Substitution for s in the high-pass filter case}: \ s = \frac{\omega_a}{s} \tag{1}$$

$$G(s) = \frac{1}{1 + (-1)^n(\frac{\omega_{ac}^n}{s^n})} \tag{2}$$

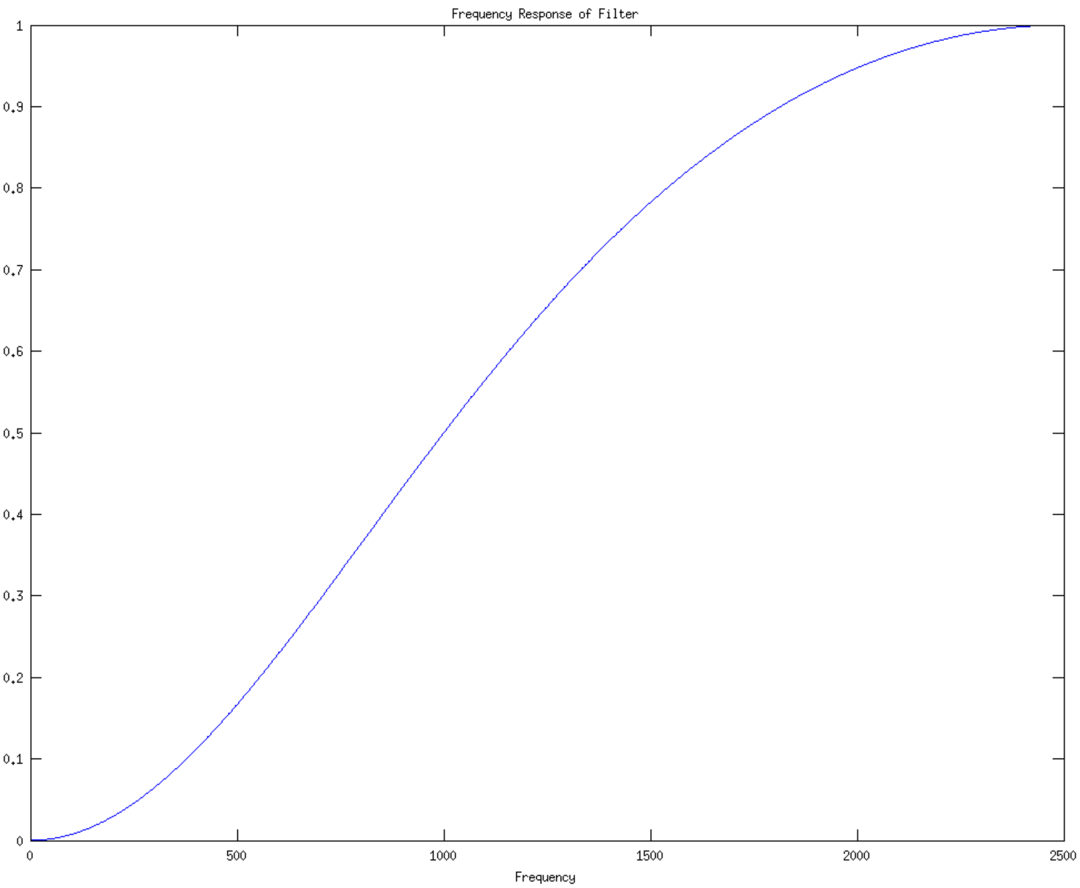
$$G(s) = \frac{s^n}{s^n + (-1)^n(\omega_{ac}^n)} \tag{3}$$

$$\text{Substitution for bilinear transform}: \ s = \frac{2}{T}\frac{z-1}{z+1} \tag{4}$$

$$G(z) = \frac{(\frac{2}{T}\frac{z-1}{z+1})^n}{((\frac{2}{T}\frac{z-1}{z+1})^n + (-1)^n(w_{ac})^n} \tag{5}$$

$$\tag{6}$$

Graph of the frequency response of the filter. With a -3dB cut-off frequency of 1 kHz, attenuation of 10 dB at 350 Hz, and a sampling frequency of 5 KHz.


Frequency Response of Filter
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
2500
Frequency