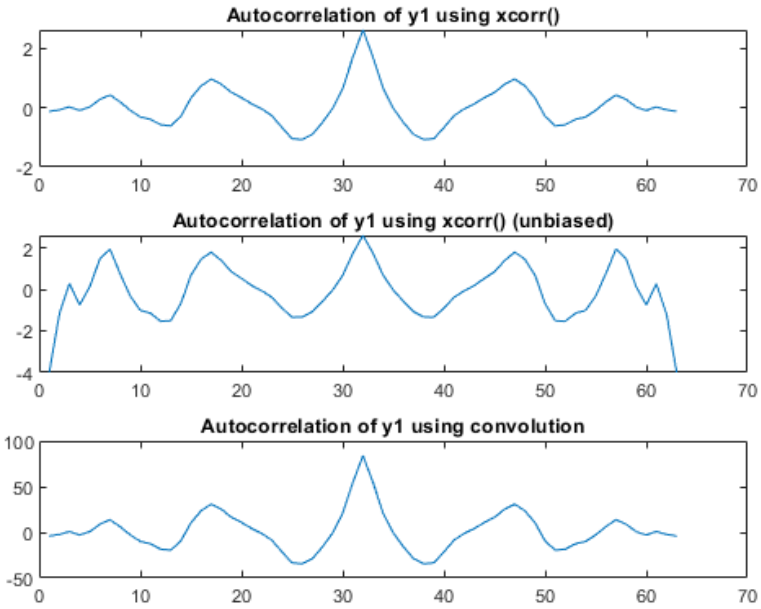
Contents

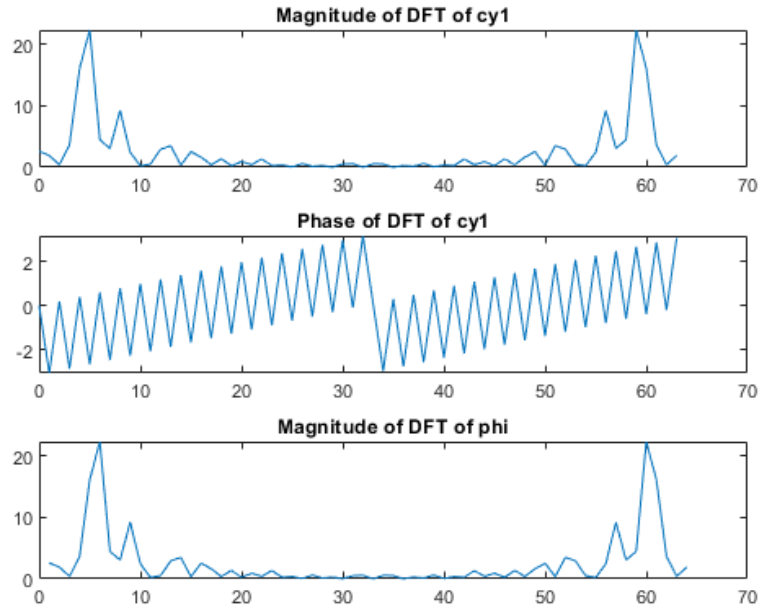

```
clc;
close all;
clear;

load pj2data;
```

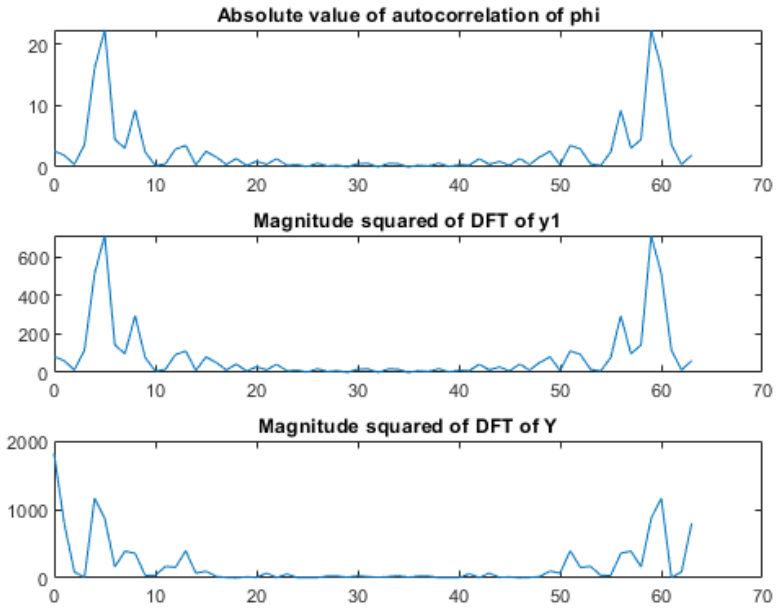
Problem A

```
y1 = y(1:32);
cy1 = xcorr(y1, y1, 'biased');
cy1_unbiased = xcorr(y1, y1, 'unbiased');
convy1 = conv(y1, fliplr(y1));

figure();
subplot(3, 1, 1);
plot(cy1);
title("Autocorrelation of y1 using xcorr()");
subplot(3, 1, 2);
plot(cy1_unbiased);
title("Autocorrelation of y1 using xcorr() (unbiased)");
subplot(3, 1, 3);
plot(convy1);
title("Autocorrelation of y1 using convolution");

% both graphs are the same except for a scaling factor.

% A.1: A unbiased autocorrelation function assumes that on average, the
% error is zero.

% A.2a: The Fourier transform of an autocorrelation function is the power
% spectral density function. A PSD has to be real and positive, as it
% describes average power per frequency, and power can neither be imaginary
% or negative.

% A.2b/c:

cy1dft = fft(cy1, 64);
k6 = 0:1:63;
phi = zeros(1, 32);

for i = 1:32
    phi(i) = sum(y1(1:32-i+1) .* y1(i:32));
end

phi = fft(convy1/32, 64);


figure();
subplot(3, 1, 1);
plot(k6, abs(cy1dft));
title("Magnitude of DFT of cy1");
subplot(3, 1, 2);
plot(k6, angle(cy1dft));
title("Phase of DFT of cy1");
subplot(3, 1, 3);
plot(1:1:64, abs(phi));
title("Magnitude of DFT of phi");

% The DFT of phi and the DFT of cy1 are the same

% A.3a:
figure();
subplot(3, 1, 1);
```

```
plot(k6, abs(phi));
title("Absolute value of autocorrelation of phi");
% A.3b:
subplot(3, 1, 2);
plot(k6, abs(fft(y1, 64)).^2);
title("Magnitude squared of DFT of y1");
% A.3c:
subplot(3, 1, 3);
plot(k6, abs(fft(y(1:64), 64)).^2);
title("Magnitude squared of DFT of Y");

% all of these are the same with the exception of a scaling factor
```





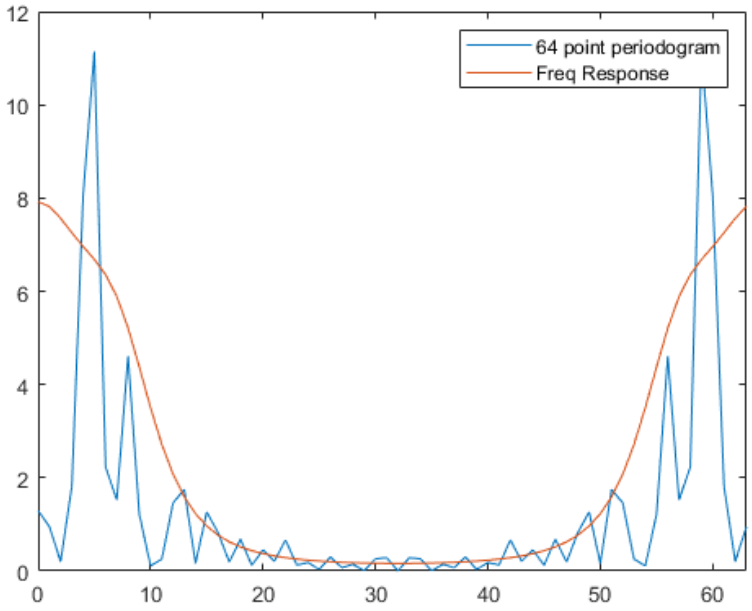


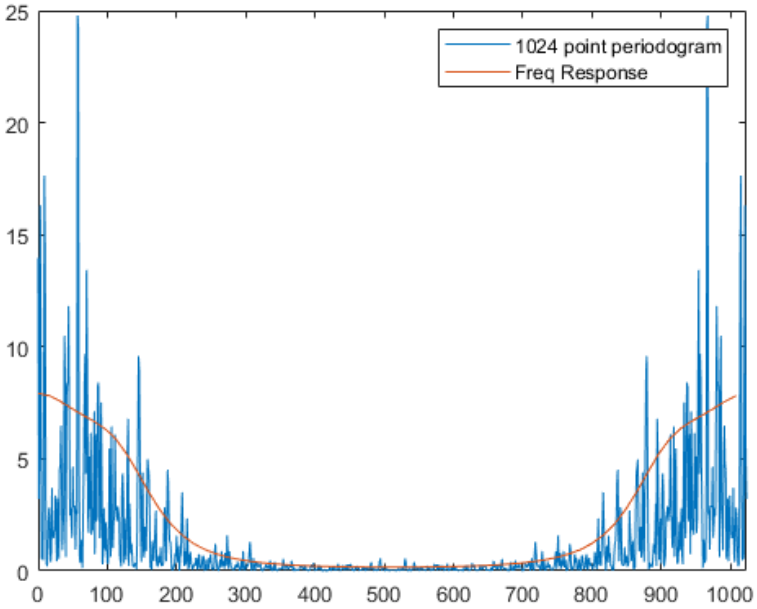
Problem B

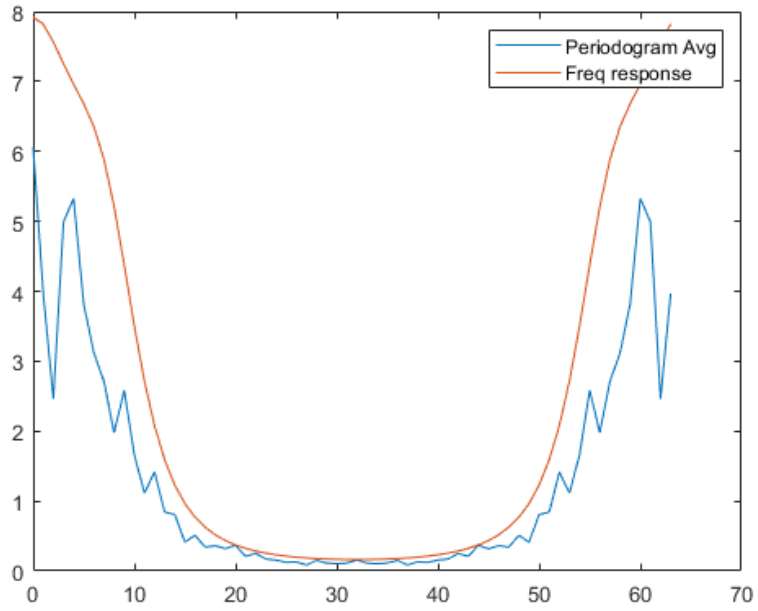
```
% B.1:
Hejw2_down = downsample(Hejw2, 8);
p64 = abs(fft(y1, 64).^2/64);
figure();
plot(k6,p64, k6, Hejw2_down);
xlim([min(k6) max(k6)]);
legend("64 point periodogram", "Freq Response")

err1 = sum(abs(p64 - Hejw2_down).^2)/64;

%B.2
p1024 = abs(fft(y,1024)).^2/1024;
k10 = 0:1023;

figure();
plot(k10, p1024, 16.*k6, Hejw2_down);
xlim([min(k10) max(k10)]);
legend("1024 point periodogram", "Freq Response")
err2 = sum(abs((downsample(p1024,16)-Hejw2_down)).^2)/64;

% B.3
y64 = zeros(1, 64);

for i = 1:16
    yind = 1 + (i-1) * 32;
    yw = y(yind : yind + 31);
    y64 = y64 + abs(fft(yw, 64)).^2;
end

y64 = y64/1024;

figure();
plot(k6, y64, k6, Hejw2_down);
legend('Periodogram Avg','Freq response');
err3 = sum(abs(y64-Hejw2_down).^2)/64;

% This is the only time the Frequency response is always lower than the
% periodogram.

% B.4

ycorr = xcorr(y, y, 'unbiased');
```

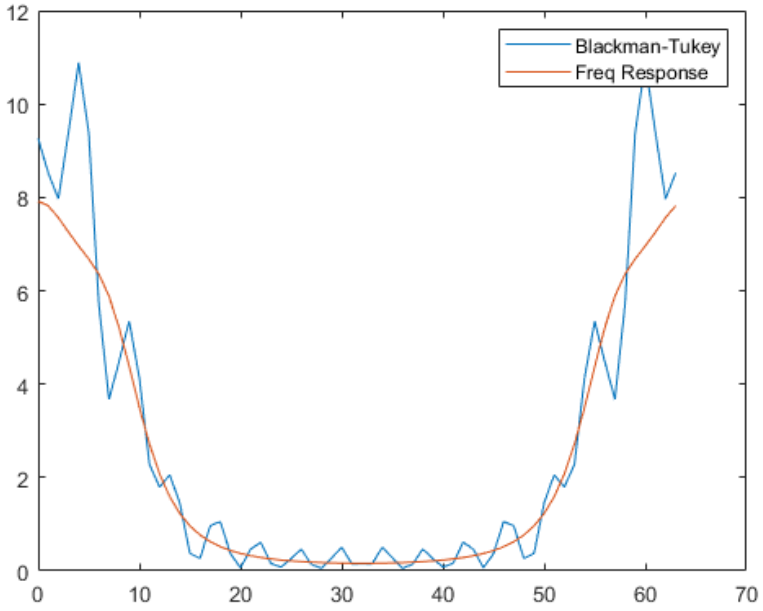
```
bt = abs(fft(ycorr(512-15 : 512+15), 64));

figure();
plot(k6, bt, k6, Hejw2_down);
legend("Blackman-Tukey", "Freq Response");

err4 = sum(abs(bt-Hejw2_down).^2)/64;

% B.5
bt_tri = abs(fft(ycorr(512-15 : 512+15) .* triang(31)', 64));
figure();
plot(k6, bt_tri, k6, Hejw2_down);
legend("Blackman-Tukey (Triangular Window)", "Freq Response");
err5 = sum(abs(bt_tri-Hejw2_down).^2)/64;

T = table([err1 ; err2 ; err3 ; err4 ; err5],'VariableNames',{'Error'},'RowName',{'64 pt', '1024 pt', 'Avg', 'BT', 'BT Triangular'});
disp(T)

% Blackman Tukey with a triangle window performed the best, as it had the
% smallest estimation error. It performs well because insted of averaging
% each sample, it actually smooths out the signal because it uses a window.
% The triangular window seems to perform way better than the rectangular
% window.
```

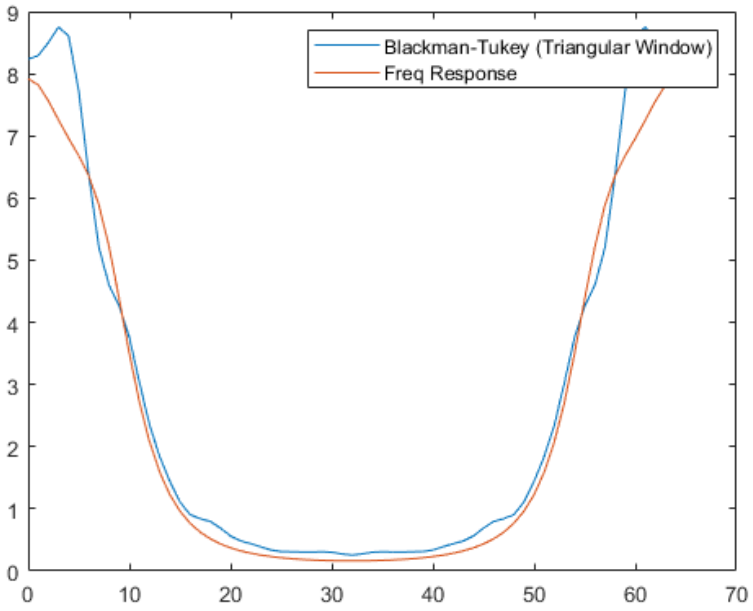
```
                  Error
                 _______

    64 pt          7.5039
    1024 pt        5.9197
    Avg            3.1403
    BT             1.1899
    BT Triangular 0.27679
```


64 point periodogram
Freq Response
0
2
4
6
8
10
12
0
10
20
30
40
50
60


1024 point periodogram
Freq Response
0
5
10
15
20
25
0
100
200
300
400
500
600
700
800
900
1000


Periodogram Avg
Freq response
0
1
2
3
4
5
6
7
8
0
10
20
30
40
50
60
70


Blackman-Tukey
Freq Response
0
2
4
6
8
10
12
0
10
20
30
40
50
60
70



Published with MATLAB® R2020b