**4.3 b & c)**

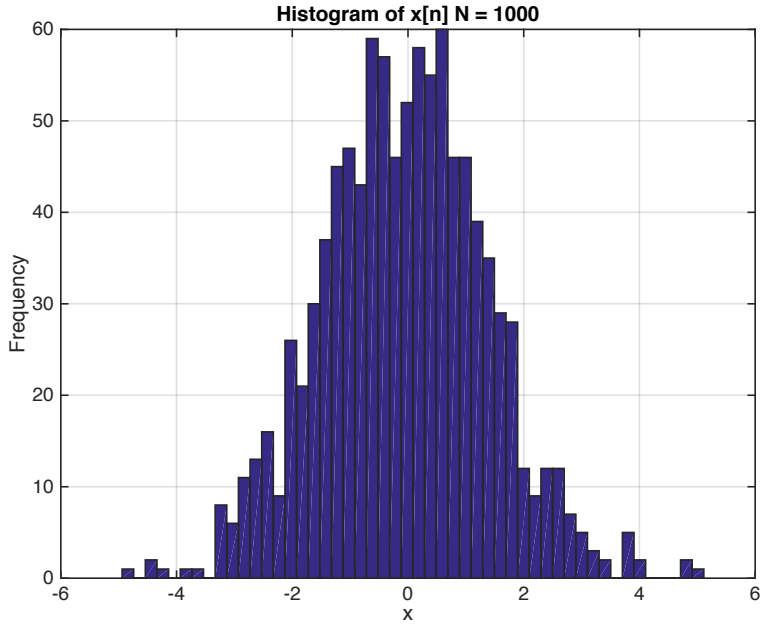


**Fig. 1:** Histogram for N = 1000 realizations of the random process x[n] defined by the difference equation x[n] =1/4 x[n-2] + $(15/8)^{1/2}$ w[n].

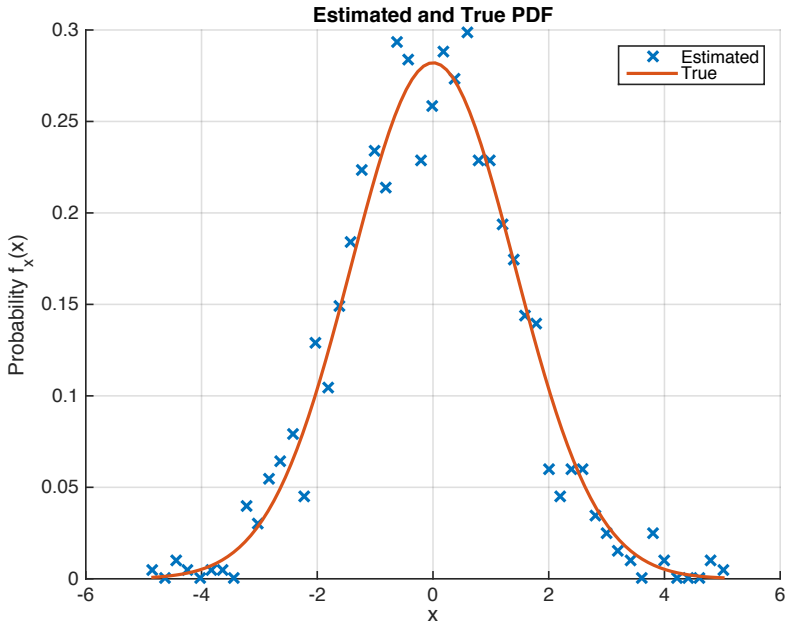


**Fig. 2:** Normalized true probability density function of x[n] given by N(0,2) and estimated probability density function generated from the histogram of the 1000 x[n] realizations. The PDF approximated using the model for x[n] closely matches the theoretical function. Note that the estimated PDF is found by normalizing the histogram values plotted in fig. 1 so that the area under the curve is equal to 1.

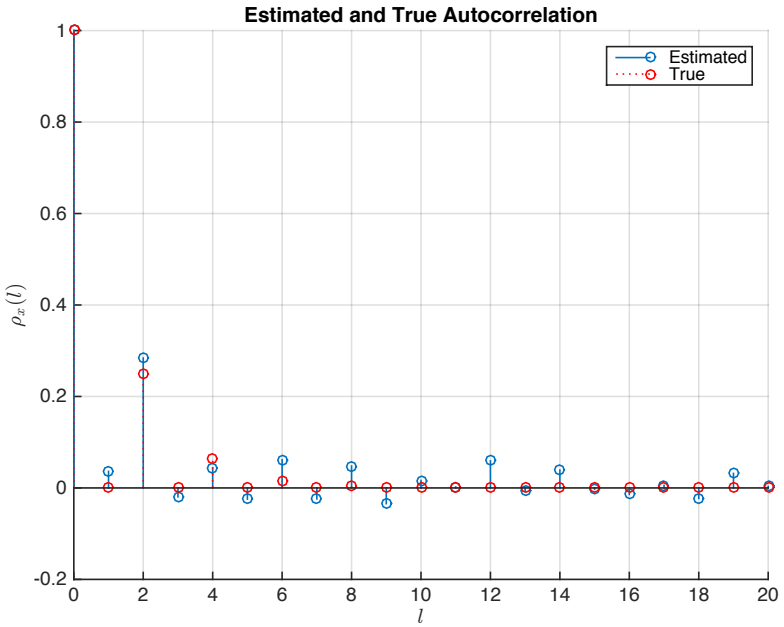


**Fig. 3:** Normalized autocorrelation estimated using Eqn. 1.2.1 plotted with the analytically derived autocorrelation. Autocorrelation values estimated from the AP model for x[n] are observed to accurately reflect the true values. Given the results shown in figures 2 and 3, the difference equation derived in part a) can be considered a valid and appropriate model for the Gaussian process x[n].

```
%% 4.3 part b and c

% generate 1000 realizations of x[n] using difference equation
x = zeros(1,1000);
ni = 0:999;
for n = ni
    if n >=2
    x(n+1) = x(n+1)+(1/4)*x(n+1-2);
    end
    x(n+1) = x(n+1)+sqrt(15/8)*randn;
end

% organize x[n] realizations in histogram to estimate pdf
figure
[counts, centers] = hist(x,50);
hist(x,50)
xlabel('x'), ylabel('Frequency'), title('Histogram of x[n] N = 1000')

% generate true pdf values based on calculate mean and variance of x
varx = 2;
mux = 0;
X = linspace(min(centers),max(centers),100);
Y = normpdf(X,mux,sqrt(varx));

% plot normalized estimated and true pdfs
tmp = diff(centers);
dx = tmp(1);
figure
hold on
% estimated histogram is normalized to ensure AUC = 1
plot(centers,counts./(dx*sum(counts)),'x')
plot(X,Y)
hold off
xlabel('x'), ylabel('Probability f_x(x)')
legend('Estimated','True')
title('Estimated and True PDF')

% estimate the normalized autocorrelation using user defined function
% calcAC, implementing eqn 1.2.1 (see section below)
li = 0:20;
rho_hat = calcAC(x,21);

% calculate the normalized autocorrelation given by the known
expression
r_0 = (1/2).^abs(0)+(-1/2).^abs(0);
rho = ((1/2).^abs(li)+(-1/2).^abs(li))./r_0;

% plot estimated and true rho values
figure
hold on
stem(li, rho_hat)
stem(li, rho, 'r:')
hold off
xlabel('$$l$$','Interpreter','Latex')
ylabel('$$\rho_x(l)$$','Interpreter','Latex')
legend('Estimated','True')
title('Estimated and True Autocorrelation')
```

```
function rho_hat = calcAC(x,nlags)
% calculate the estimated normalized autocorrelation (eqn 1.2.1)
% nlags specifics number of lags to calculate rho (starting from l = 0)
% x is the signal to autocorr whose 1st index corresponds to n = 0

rho_hat = zeros(1,nlags);
for l = 0:nlags-1
    rho_hat(l+1) = 0;
    for n = l:(length(x)-1)
        rho_hat(l+1) = rho_hat(l+1) + x(n+1)*conj(x(n-l+1));
    end
    rho_hat(l+1) = rho_hat(l+1)/sum(x.*conj(x));
end
```

**4.19 a & b)**

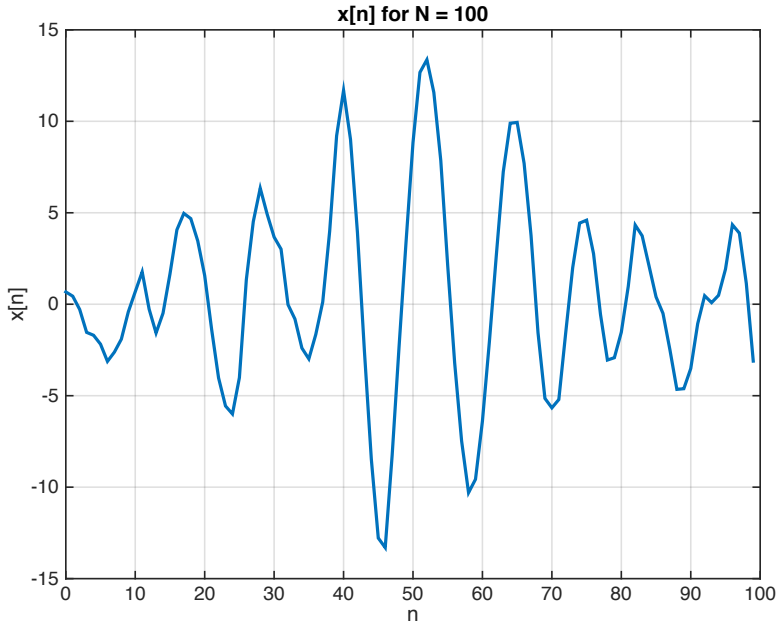


**Fig. 4:** Realization of AR(2) modeled process x[n] consisting of 100 samples.

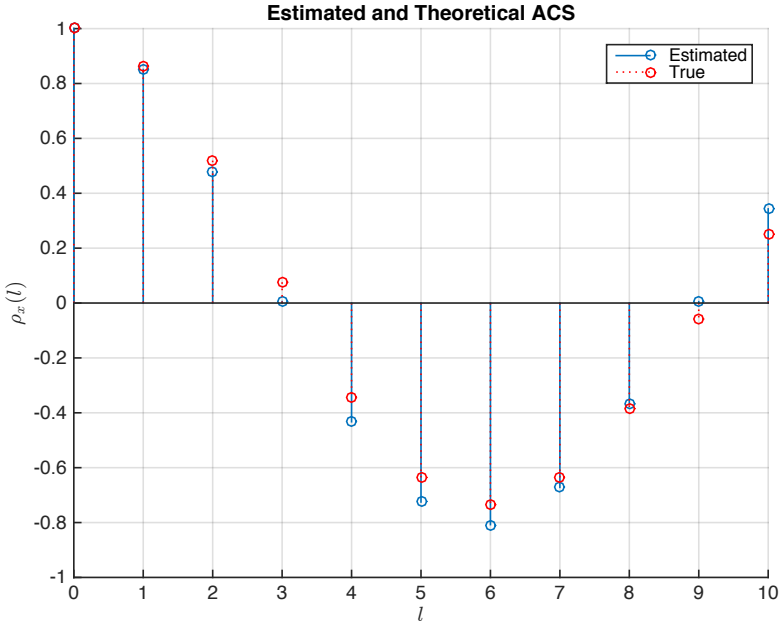


**Fig. 5:** Plot of the estimated ACS generated from a single realization of x[n] and the theoretical ACS for lags from 0 to 10. The ACS are comparable, i.e. the estimated ACS provides a fairly accurate representation of the true ACS defined by the AR(2) model.

**4.19 c)**

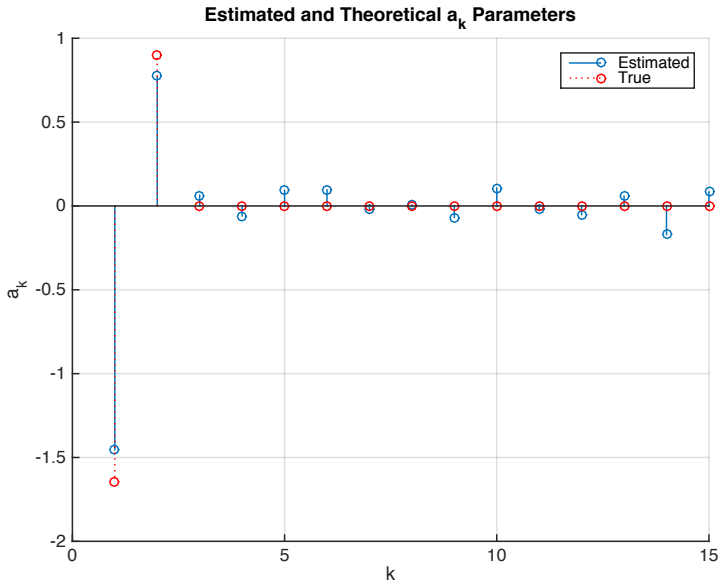


**Fig. 6:** Estimated direct form parameters calculated using the empirically derived values for $\rho_x$ plotted against theoretical $a_k$ parameters. Parameters calculated using the estimated autocorrelations closely match true values for $a_k$. $\hat{a}_1 = -1.454$ and $\hat{a}_2 = 0.7727$ accurately approximate the true parameters $a_1 = -1.6454$ and $a_2 = 0.9025$. Moreover, $\hat{a}_k$ for $k > 2$ approach zero as expected based on the true $a_k$.

**4.19 d)**

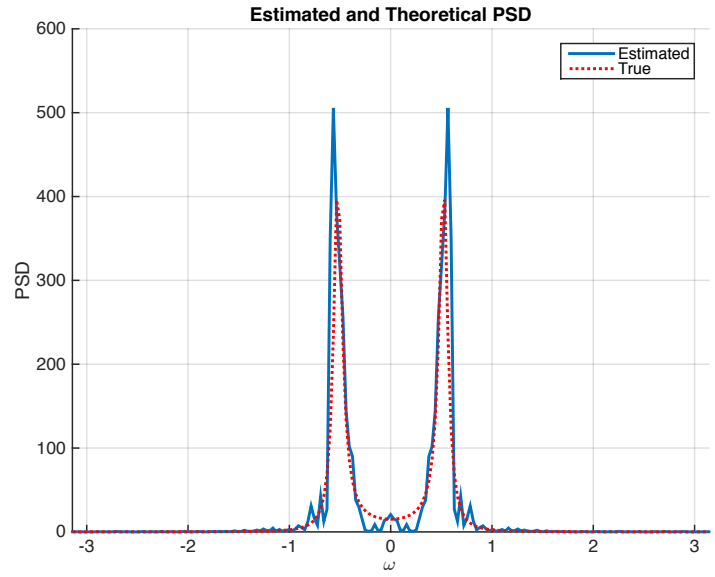


**Fig. 7:** Plots of the power spectral density estimated using direct form parameters along with the analytically derived PSD from $\omega = -\pi$ to $\pi$. The estimated PSD provides a representation of the general form of the true PSD, however does exhibit a degree of noise and error. Note that the PSD is approximated using the variance of the input ($\sigma_w^2$) and the system response $H(e^{j\omega})$ characterized by the $\hat{a}_k$ parameters.

**4.19 e)**

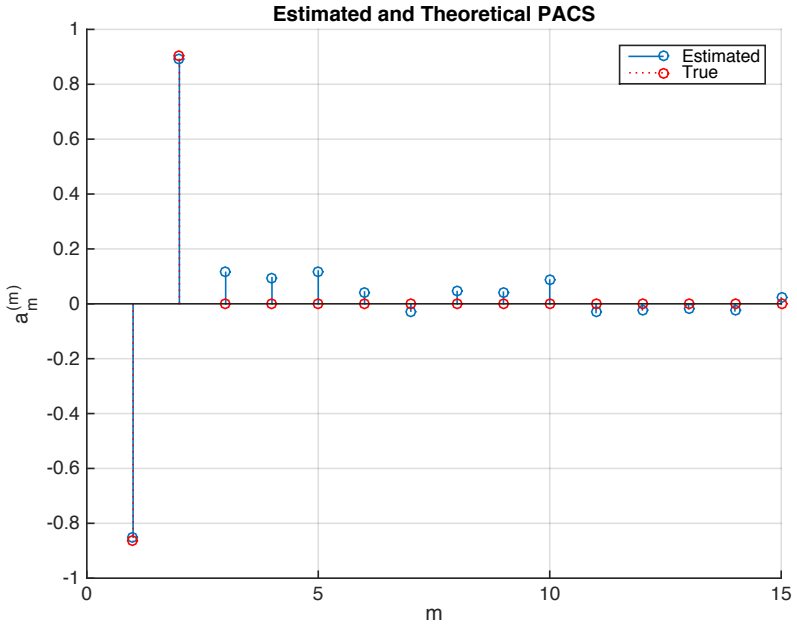


**Fig. 8:** Estimated and theoretical PACS derived from eqn. 4.2.43 using the estimated and true autocorrelation values. Plots of both PACS show strong agreement across all values of m. Given that PACS values for m > 2 are negligible, we can see that it is sufficient to model the system as an AR(2). This is consistent with the form of the original signal model.

```
%% 4.19 part a & b

% Generate 100 samples of x[n] given by the autoregressive model:
% AR(2): x(n) = -a1*x(n-1)-a2*x(n-2)+w(n)
d0 = 1;
a1 = -1.6454;
a2 = 0.9025;
x = zeros(1,100);
ni  = 0:99;
for n = ni;
    if n >= 1
        x(n+1) = -a1*x(n+1-1);
    end
    if n >= 2
        x(n+1) = x(n+1)-a2*x(n+1-2);
    end
    w(n+1) = randn;
    x(n+1) = x(n+1) + w(n+1);
end
figure
plot(ni,x); xlabel('n'), ylabel('x[n]')
title('x[n] for N = 100')

% estimate ACS (eqn 1.2.1)
li = 0:99;
rho_hat = calcAC(x,100);

% calculate poles of H(z) given a1 and a2
p = roots([1 a1 a2]);

% extract angle and magnitude of complex poles to solve true rho values
r = unique(abs(p));
theta = unique(abs(angle(p)));

% calculate theoretical ACS (eqn. 4.2.83)
rho = r.^li.*(sin((li+1).*theta)-r^2*sin((li-1).*theta))...
    ./((1+r^2)*sin(theta));

% plot estimated and theoretical ACS
figure
hold on
stem(li,rho_hat);
stem(li,rho,'r:');
xlabel('$$l$$','Interpreter','Latex')
ylabel('$$\rho_x(l)$$','Interpreter','Latex')
xlim([0 20])
ylim([-1 1])
legend('Estimated','True')
title('Estimated and Theoretical ACS')
hold off
```

```
%% 4.19 part c

% define estimated rho vector for Yule-Walker
rhohat_vec = rho_hat(2:end)';
```

```
% define estimated autocorr matrix for Yule-Walker
r1 = conj(rho_hat(1:end-1)); % first row of toeplitz hermitian matrix
Phat = toeplitz(r1); clear r1

% solve for estimated parameters using the Yule-Walker eqn (eqn 4.2.33)
ahat_vec = -inv(Phat)*rhohat_vec;

% define true rho vector for Yule-Walker
rho_vec = rho(2:end)';

% define true autocorr matrix for Yule-Walker
r1 = conj(rho(1:end-1));
P = toeplitz(r1); clear r1

% solve for true parameters using the Yule-Walker eqn (eqn 4.2.33)
a_vec = -inv(P)*rho_vec;

% plot true and estimated parameter values for comparison
figure
hold on
stem(ahat_vec);
stem(a_vec,'r:')
xlabel('k'); ylabel('a_k')
hold off
legend('Estimated','True')
xlim([0 15])
title('Estimated and Theoretical a_k Parameters')
```

---

```
%% 4.19 part d

clear tmp

w = -pi:pi/100:pi;
% calculate the PSD given the estimated a_k parameters by definition of
Rx
% Rx = w_var * |H(z)|^2 where H(z) = 1/(1+a1*z^-1+a2*z^-2 ...)
%   z = e^jw and w_var = 1 (p. 165)
for wi = 1:length(w)
    tmp = exp(-1i.*w(wi).*(1:length(ahat_vec)));
    R_hat(wi) =
1./((1+sum(ahat_vec'.*tmp))*conj(1+sum(ahat_vec'.*tmp)));
    clear tmp
end

% using eqn 4.2.88, we solve for the true PSD using the poles of the AP
system
% with a1 and a2 given by the model
R = d0^2./((1-2.*r.*cos(w-theta)+r^2).*(1-2.*r.*cos(w+theta)+r^2));

figure;
hold on
plot(w,R_hat)
plot(w,R,'r:')
hold off
legend('Estimated','True')
xlim([-pi pi])
```

```
xlabel('\omega')
ylabel('PSD')
title('Estimated and Theoretical PSD')
```

---

```
%% 4.19 part e

% calculate the true and estimated PACS/lattice structures from the rho
% values through application of the Yule-Walker equation given by eqn.
% 4.2.43

for m = 1:15
    % calculate PACS using estimated rho
    % evaluate Yule-Walker using a portion of autocorrelation values
where
    % m is the maximum index
    rhohat_vec = rho_hat(2:m+1)';
    r1 = conj(rho_hat(1:m));
    Phat = toeplitz(r1);
    tmp = -inv(Phat)*rhohat_vec;

    % extract the mth value which defines the lattice/PACS parameter
    khat(m) = tmp(end);
    clear r1 c1 tmp

    % repeat procedure using true rho values
    rho_vec = rho(2:m+1)';
    r1 = conj(rho(1:m));
    P = toeplitz(r1);
    tmp = -inv(P)*rho_vec;
    k(m) = tmp(end);
    clear r1 c1 tmp
end

figure
hold on
stem(khat);
stem(k,'r:')
xlabel('m'); ylabel('a^(^m^)_m')
hold off
legend('Estimated','True')
xlim([0 15])
title('Estimated and Theoretical PACS')
```

**4.32 a)**

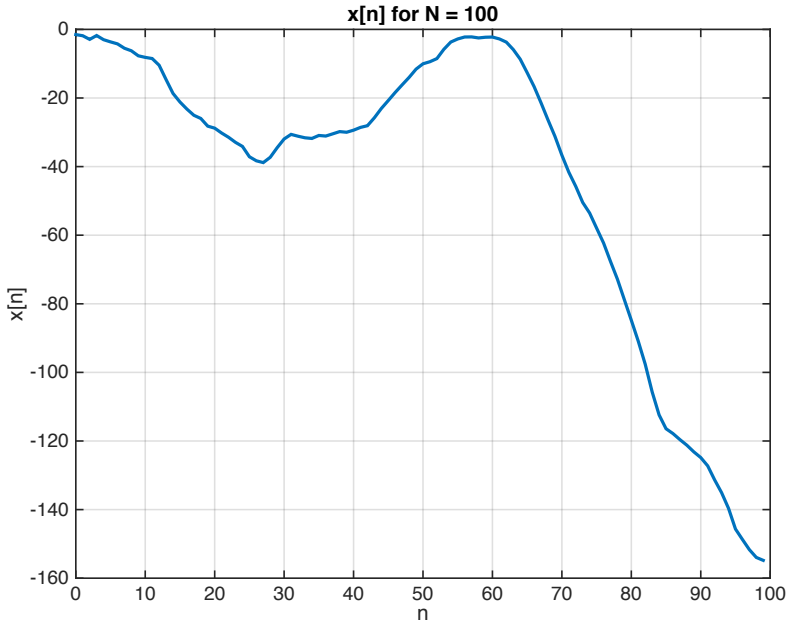


**Fig. 9:** Realization of AP modeled process x[n] consisting of 100 samples.

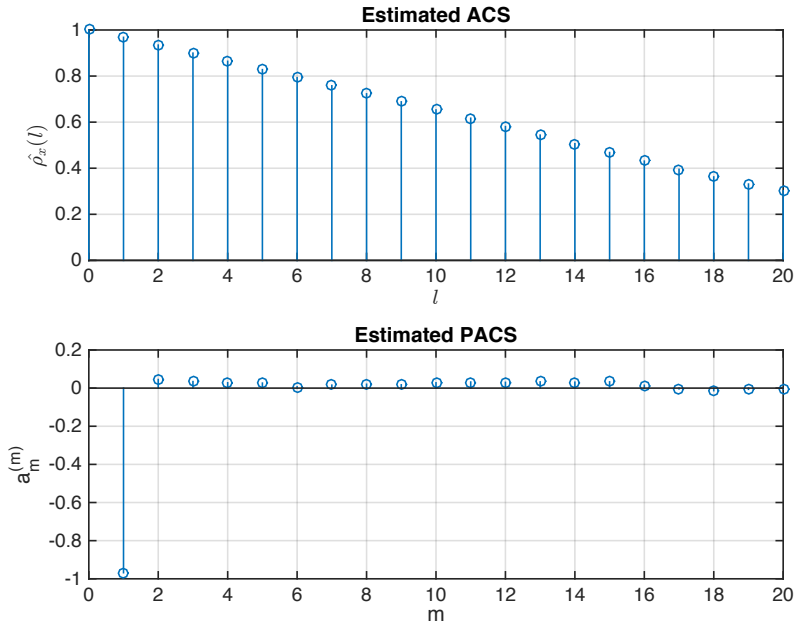


**Fig. 10:** Values for ACS ($l$ = 0 to 20) and PACS (m = 1 to 20) approximated using 100 samples of x[n]. Given that PAC values for m > 1 approach zero, it may be sufficient to model the system as a 1st order AP.

**4.32 b)**

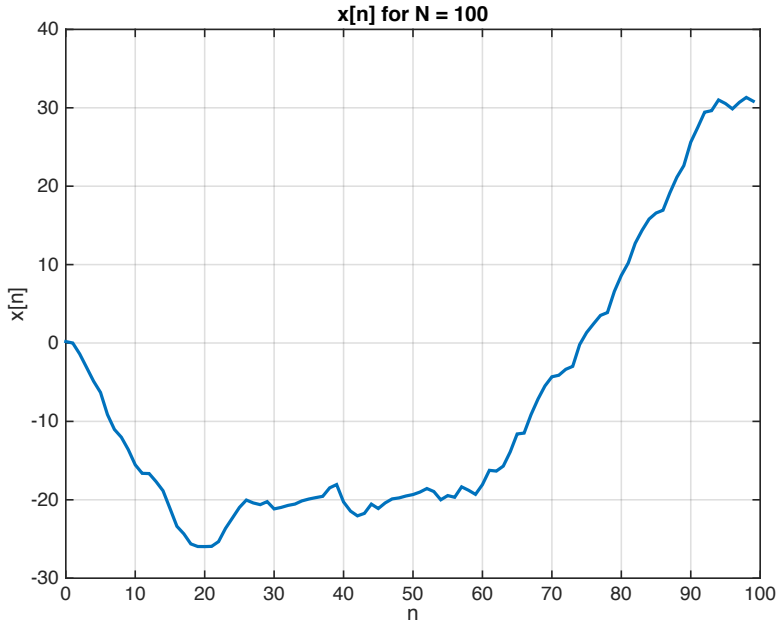


**Fig. 11:** Realization of PZ modeled process x[n] consisting of 100 samples.

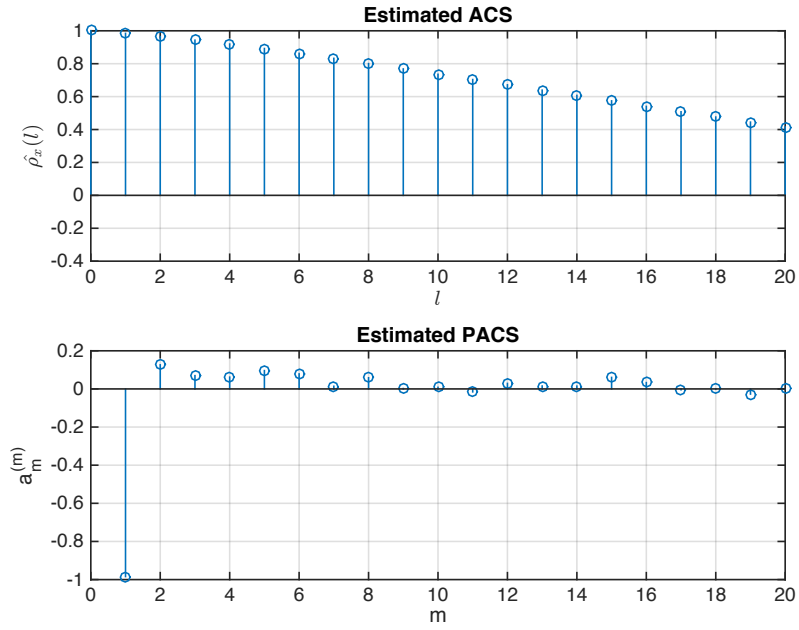


**Fig. 12:** Values for ACS ($l$ = 0 to 20) and PACS (m = 1 to 20) approximated using 100 samples of x[n]. Given that PAC values for m > 1 approach zero, it may be sufficient to model the system as a 1st order AP.

```
%% 4.32 part a
close all; clear all;

% Generate 100 samples of x[n] in part a) based on difference equation
% defined by the system function H(z)
ni = 0:99;
x = zeros(1,length(ni));
for n = ni
    if n >= 1
        x(n+1) = x(n+1)+1.9*x(n+1-1);
    end
    if n >= 2
        x(n+1) = x(n+1)-0.9*x(n+1-2);
    end
    x(n+1) = x(n+1) + randn;
end
figure
plot(ni,x); xlabel('n'),ylabel('x[n]')
title('x[n] for N = 100')

% estimate the normalized autocorrelation using user defined function
% calcAC, implementing eqn 1.2.1
li = 0:99;
rho_hat = calcAC(x,length(li));

figure
subplot(211)
stem(li, rho_hat);
xlabel('$$l$$','Interpreter','Latex')
ylabel('$$\hat{\rho_x}(l)$$','Interpreter','Latex')
xlim([0 20])
title('Estimated ACS')

% calculate the estimated PACS/lattice structures from the rho
% values through application of the Yule-Walker equation given by eqn.
% 4.2.43
for m = 1:20
    rhohat_vec = rho_hat(2:m+1)';
    r1 = conj(rho_hat(1:m));
    Phat = toeplitz(r1);
    tmp = -inv(Phat)*rhohat_vec;
    khat(m) = tmp(end);
end

subplot(212)
stem(khat);
xlabel('m'); ylabel('a^(^m^)_m')
xlim([0 20])
title('Estimated PACS')
```

```
%% 4.32 part b
close all; clear all

% Generate 100 samples of x[n] in part b) based on difference equation
% defined by the system function H(z)
ni = 0:99;
x = zeros(1,length(ni));
```

```
for n = ni
    if n >= 1
        x(n+1) = x(n+1)+1.9*x(n+1-1)-0.5*W(n+1-1);
    end
    if n >= 2
        x(n+1) = x(n+1)-0.9*x(n+1-2);
    end
    W(n+1) = randn; % note that the system is PZ
    x(n+1) = x(n+1) + W(n+1);
end
figure
plot(ni,x); xlabel('n'),ylabel('x[n]')
title('x[n] for N = 100')

% estimate the normalized autocorrelation using user defined function
% calcAC, implementing eqn 1.2.1
li = 0:99;
rho_hat = calcAC(x,length(li));

figure
subplot(211)
stem(li, rho_hat);
xlabel('$$l$$','Interpreter','Latex')
ylabel('$$\hat{\rho_x}(l)$$','Interpreter','Latex')
xlim([0 20])
title('Estimated ACS')


% calculate the estimated PACS/lattice structures from the rho
% values through application of the Yule-Walker equation given by eqn.
% 4.2.43.
for m = 1:20
    rhohat_vec = rho_hat(2:m+1)';
    r1 = conj(rho_hat(1:m));
    Phat = toeplitz(r1);
    tmp = -inv(Phat)*rhohat_vec;
    khat(m) = tmp(end);
end

subplot(212)
stem(khat);
xlabel('m'); ylabel('a^(^m^)_m')
xlim([0 20])
title('Estimated PACS')
```

**5.2 a & b)**

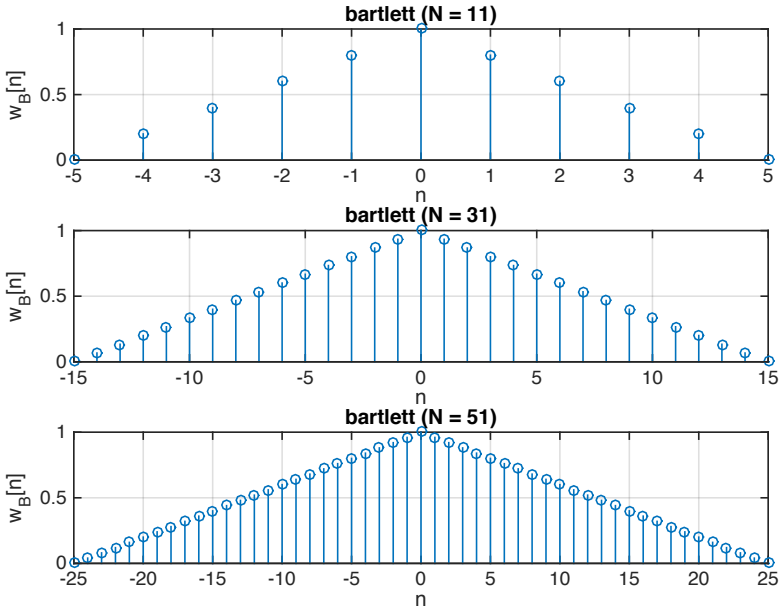


**Fig. 13:** N = 11, 31, and 51 length windows generated by the MATLAB `bartlett` function.

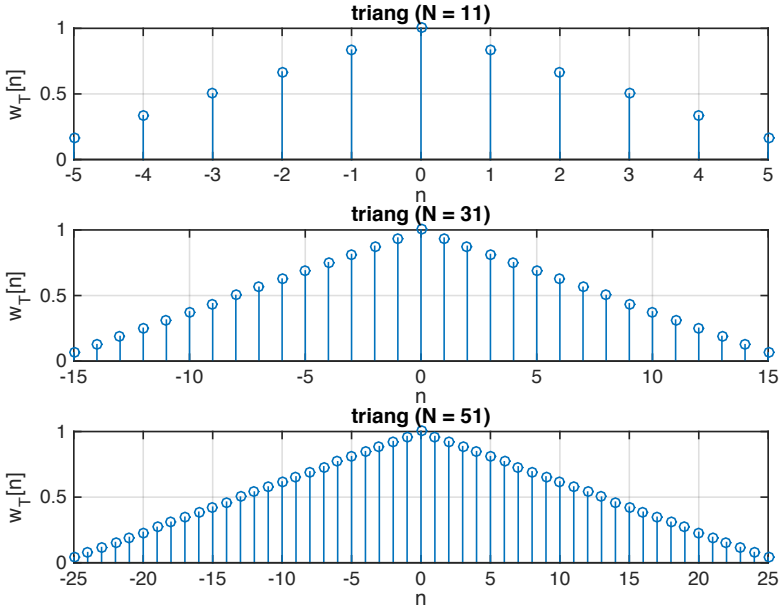


**Fig. 14:** N = 11, 31, and 51 length windows generated by the MATLAB `triang` function.

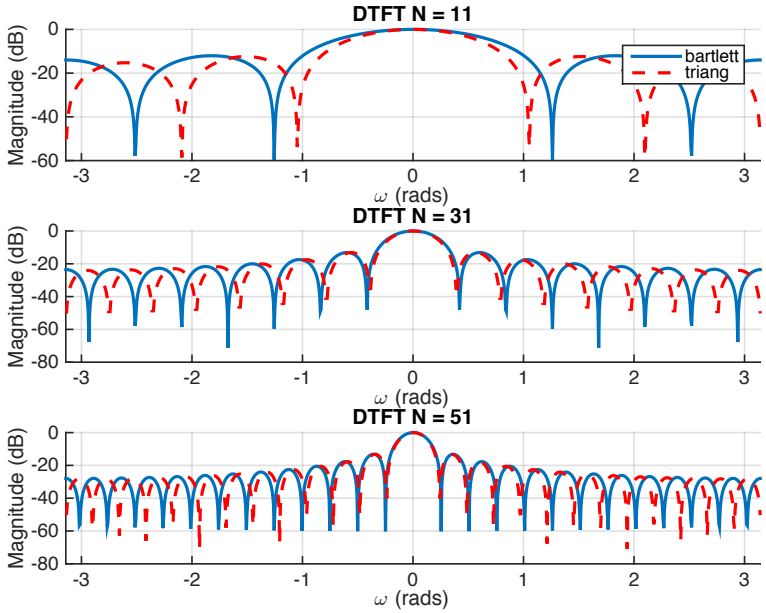


**Fig. 15:** DTFTs $W_B(e^{j\omega})$ and $W_T(e^{j\omega})$ for N = 11, 31, and 51 length windows in log-scale from $\omega = -\pi$ to $\pi$. Comparison of the spectra show that the mainlobe of windows generated by `triang` are narrower than the Bartlett windows for a given length N. This is further illustrated in fig. 16, which provides a direct comparison of the null-to-null mainlobe width as a function of N.

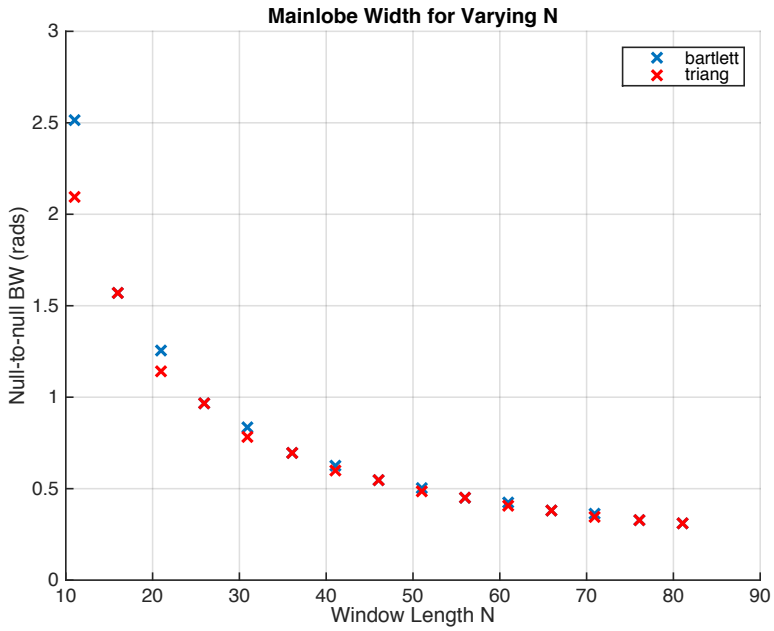


**Fig. 16:** Mainlobe width given by null-to-null bandwidth for windows generated by `bartlett` and `triang` for varying lengths N. Again, we see that mainlobe width is generally larger for Bartlett windows especially for smaller values of N. This separation in width diminishes with increased N.

As can be seen from the previous figures, window type and length govern a number of properties e.g. mainlobe width, side-lobe amplitude, etc. Consequently, for windowing nonzero samples, there are a number trade-offs to consider and the selection of a specific window depends greatly on the properties of the windowed signal and the features to be extracted.

Given that the multiplication of a signal by a window in the time domain corresponds to convolution of the signal's spectrum to the spectrum of the window, we can infer a number of trends related to windowing and spectral estimation based on the DTFT of the windowing function. When convolving the signal spectrum with a narrow function, we expect better recovery of the original spectrum, i.e. higher spectral resolution. Contrarily, when convolving the signal spectrum with a wide function, we expect some write in of adjacent frequency values and smoothing of the original signal spectrum. Along this reasoning, windowing functions with narrow mainlobes and low amplitude sidelobes are thus particularly useful for spectra consisting of distinct, closely spaced frequency components. On the other hand, in the case of a smooth and more uniform specta, windows with wider mainlobes can help to eliminate and smooth out noise in the spectra.

In regards to our results, we observe a decrease in the mainlobe width for both `bartlett` and `triang` windows with increasing window size N (fig. 16). Furthermore, mainlobe widths for the `triang` function are in general observed to be narrower than the mainlobe widths of Bartlett windows. Therefore, if high spectral resolution is needed, a `triang` window with large N is more appropriate.

Shorter windows with lower N and windows generated using the `bartlett` function exhibit larger mainlobe widths. In this case, if a smooth spectrum is expected, the `bartlett` function can provide greater blurring and smoothing of the frequency components to better estimate the desired spectrum.

Note that additional considerations should also be taken in regards to the window size and averaging between the windowed data. For example, the use of larger windows limits the number of spectra which can be averaged making the estimates less accurate. On the other hand, the use of smaller windows allows for more averaging of realizations and higher accuracy, however, this also limits the spectral resolution.

**5.2 c)**

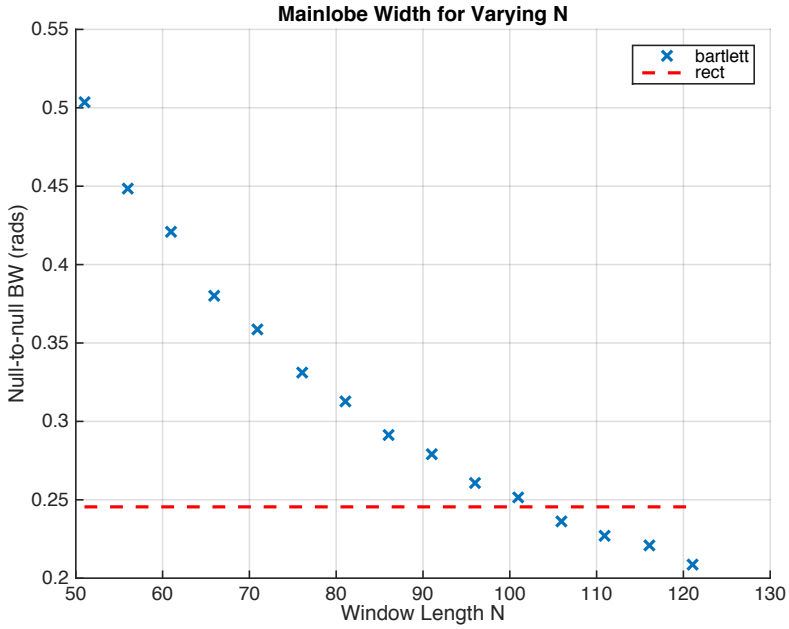


**Fig. 17:** Null-to-null mainlobe width of Bartlett windows with varying N plotted in reference to the mainlobe width of a rectangular window of N = 51. As shown in the figure, a Bartlett window of approximately N = 101 is required to produce a mainlobe of similar width to that of a 51-point rectangular window.

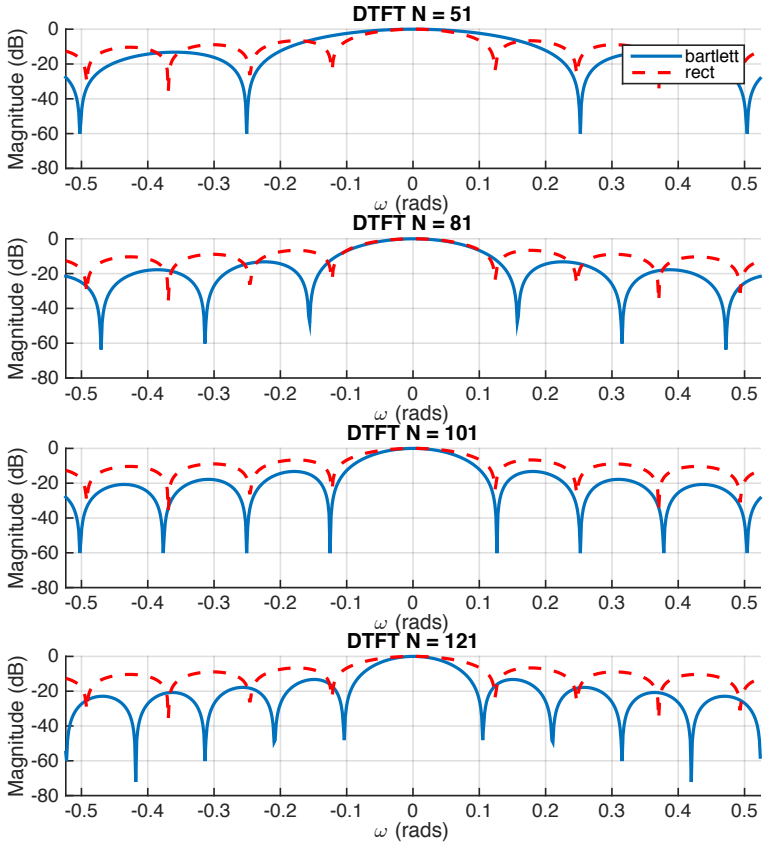


**Fig. 18:** DTFTs of Bartlett windows for varying N and rectangular window of N = 51. Consistent with the results presented in fig. 18, the mainlobe widths of the 51-point rectangular and 101-point Bartlett windows are roughly equivalent.

```
%% 5.2 part a & b
clear all; clc; close all

% generate and plot bartlett windows of varying length N
N = [11 31 51];
figure
for nn = 1:length(N)
    subplot(length(N),1,nn)
    stem(-floor(N(nn)/2):floor(N(nn)/2),bartlett(N(nn)))
    xlabel('n')
    ylabel('w_B[n]')
    title(sprintf('bartlett (N = %d)',N(nn)))
end

% generate and plot triang windows of varying length N
figure
for nn = 1:length(N)
    subplot(length(N),1,nn)
    stem(-floor(N(nn)/2):floor(N(nn)/2),triang(N(nn)))
    xlabel('n')
    ylabel('w_T[n]')
    title(sprintf('triang (N = %d)',N(nn)))
end

% generate plots of dtft of bartlett and triang windows of varying N
Ndtft = 4*1024;
figure
for nn = 1:length(N)
    % calculate the dtft of the bartlett windows in dB with zero pad
    tmp = fft(bartlett(N(nn)),Ndtft);
    WB(nn,:) = 10.*log10(abs(tmp)./max(abs(tmp)));
    clear tmp

    % calculate the dtft of the triang windows in dB with zero pad
    tmp = fft(triang(N(nn)),Ndtft);
    WT(nn,:) = 10.*log10(abs(tmp)./max(abs(tmp)));
    clear tmp

    % plot the dtft of windows in dB
    subplot(length(N),1,nn)
    hold on
    plot(linspace(-pi,pi,Ndtft),fftshift(WB(nn,:)));
    plot(linspace(-pi,pi,Ndtft),fftshift(WT(nn,:)),'r--');
    hold off
    if nn == 1, legend('bartlett','triang'); end;
    xlim([-pi,pi])
    xlabel('\omega (rads)')
    ylabel('Magnitude (dB)')
    title(sprintf('DTFT N = %d',N(nn)))
end

% generate plot of mainlobe width as a function of window length
N = 11:5:81
for nn = 1:length(N)
    % calculate dtfts of windows for varying N
    tmp = fft(bartlett(N(nn)),Ndtft);
    WB(nn,:) = 10.*log10(abs(tmp)./max(abs(tmp)));
```

```
    clear tmp

    tmp = fft(triang(N(nn)),Ndtft);
    WT(nn,:) = 10.*log10(abs(tmp)./max(abs(tmp)));
    clear tmp
    w = linspace(0,2*pi,Ndtft);

    % find the frequency value corresponding to full width half max of
    % bartlett and triang windows of varying N (where db = -3)
    fwhmB(nn) = 2*w(find(WB(nn,:)<=-3,1));
    fwhmT(nn) = 2*w(find(WT(nn,:)<=-3,1));

    % find the frequency value corresponding to null to null main lobe
    % width for bartlett and triang windows of varying N
    [~, itmp] = findpeaks(-WB(nn,:));
    bwB(nn) = 2*w(itmp(1));
    [~, itmp] = findpeaks(-WT(nn,:));
    bwT(nn) =  2*w(itmp(1));
end
figure; hold on
plot(N,bwB,'x')
plot(N,bwT,'rx')
hold off
xlabel('Window Length N')
ylabel('Null-to-null BW (rads)')
title('Mainlobe Width for Varying N')
legend('bartlett','triang')
```

---

```
%% 5.2 part c

clear WB bwB
% calculate and plot the mainlobe width for series of bartlett windows
of
% varying N and compare to mainlobe width of 51-point rect
N = 51:5:121
for nn = 1:length(N)
    % calculate dtfts of windows for varying N
    tmp = fft(bartlett(N(nn)),Ndtft);
    WB(nn,:) = 10.*log10(abs(tmp)./max(abs(tmp)));
    clear tmp

    tmp = fft(ones(1,51),Ndtft);
    WR(nn,:) = 10.*log10(abs(tmp)./max(abs(tmp)));
    clear tmp
    w = linspace(0,2*pi,Ndtft);

    % find the frequency value corresponding to null to null main lobe
    % width for bartlett and rect windows of varying N
    [~, itmp] = findpeaks(-WB(nn,:));
    bwB(nn) = 2*w(itmp(1));
    [~, itmp] = findpeaks(-WR(nn,:));
    bwR(nn) =  2*w(itmp(1));
end
figure; hold on
plot(N,bwB,'x')
plot(N,bwR,'r--')
hold off
```

```
xlabel('Window Length N')
ylabel('Null-to-null BW (rads)')
title('Mainlobe Width for Varying N')
legend('bartlett','rect')

clear WB bwB
% compare dtft of bartlett of varying lengths to rect window with N =
51
figure
N = [51 81 101 121];
for nn = 1:length(N)

    % calculate dtft of rect and bartlett windows
    WR = fft(ones(1,51),Ndtft);
    WB = fft(bartlett(N(nn)),Ndtft);

    subplot(length(N),1,nn)
    hold on
    plot(linspace(-
pi,pi,Ndtft),fftshift(10.*log10(abs(WB)./max(abs(WB)))))
    plot(linspace(-
pi,pi,Ndtft),fftshift(10.*log10(abs(WR)./max(abs(WR)))),'r--')
    hold off

    if nn == 1, legend('bartlett','rect'); end;

    xlim([-pi/6,pi/6])
    xlabel('\omega (rads)')
    ylabel('Magnitude (dB)')
    title(sprintf('DTFT N = %d',N(nn)))
end
```