EECE5666 (DSP) : Homework-4

Due on March 8, 2022 by 11:59 pm via submission portal.

NAME: McKean, Tyler

Table of Contents

Instructions

1. You are required to complete this assignment using Live Editor.
2. Enter your MATLAB script in the spaces provided. If it contains a plot, the plot will be displayed after the script.
3. All your plots must be properly labeled and should have appropriate titles to get full credit.
4. Use the equation editor to typeset mathematical material such as variables, equations, etc.
5. After completeing this assignment, export this Live script to PDF and submit the PDF file through the provided submission portal.
6. You will have only one attempt to submit your assignment. Make every effort to submit the correct and completed PDF file the first time.
7. Please submit your homework before the due date/time. A late submission after midnight of the due date will result in loss of points at a rate of 10% per hour until 8 am the following day, at which time the solutions will be published.

Default Plot Parameters

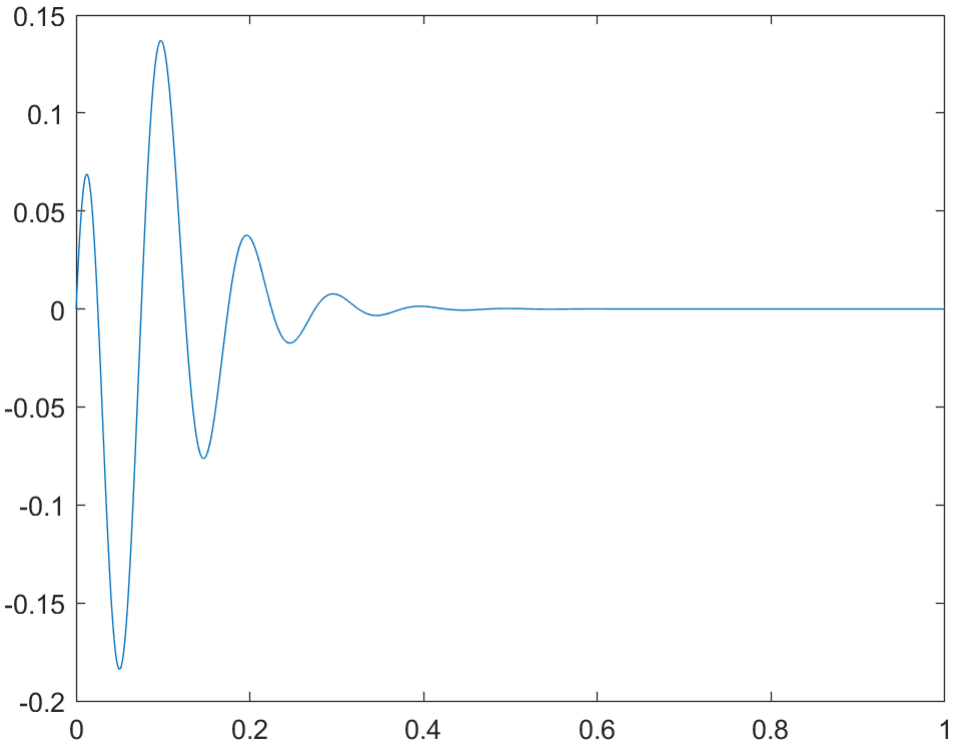
```
set(0,'defaultfigurepaperunits','points','defaultfigureunits','points');
set(0,'defaultaxesfontsize',10); set(0,'defaultaxeslinewidth',1.5);
set(0,'defaultaxestitlefontsize',1.4,'defaultaxeslabelfontsize',1.2);
```

Problem 4.1

Text Problem 7.27 (Page 426)

Let $x_c(t) = 10te^{-20t}\cos(20\pi t)u(t)$.

```
clc; close all; clear;
t = linspace(0,1,1001); xc = 10.*t.*exp(-20.*t).*cos(20*pi.*t);
figure, plot(t,xc)
```



(a) Determine the CTFT $X_c(j2\pi F)$ of $x_c(t)$.

Solution:

The CTFT of $x_c(t)$ is

$$X_c(j2\pi F) = \int_{-\infty}^{\infty} x_c(t)e^{-j2\pi Ft} = \int_0^{\infty} 10te^{-20t}\cos(20\pi t)e^{-j2\pi Ft}$$

We can simplify the integration process by using the Laplace transform and euler's identities

$$\cos(20\pi t) \leftrightarrow \frac{e^{j20\pi t} + e^{-j20\pi t}}{2}$$

$$X_c(j2\pi F) = \int_0^\infty \frac{10}{2} t e^{-20+j2\pi(10)t} e^{-j2\pi Ft} dt + \int_0^\infty \frac{10}{2} t e^{-20-j2\pi(10)t} e^{-j2\pi Ft} dt$$

Then using the Laplace Transform pair, we get:

$$x(t) = te^{at} \rightarrow X(s) = \frac{1}{(s-a)^2}$$

$$X_c(s) = \frac{5}{(s+20-j2\pi(10))^2} + \frac{5}{(s+20+j2\pi(10))^2}$$

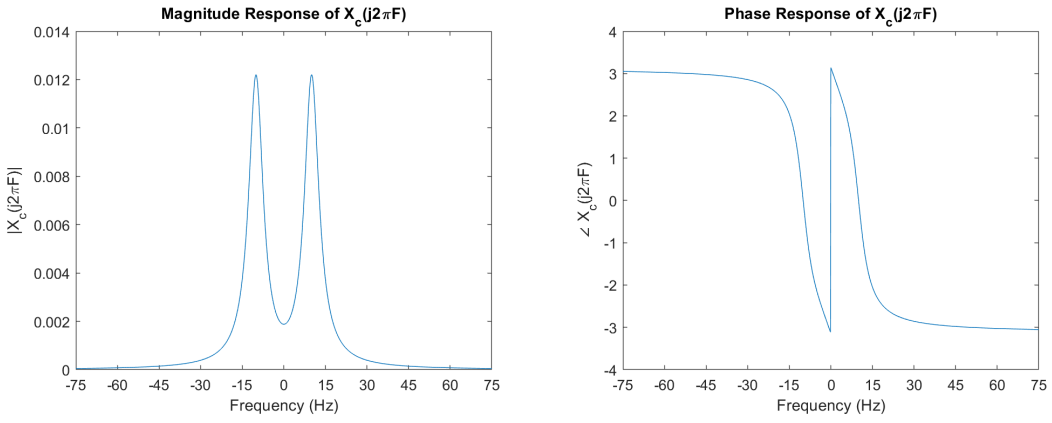
$$X_c(s) = X_c(j2\pi F)|_{s=j2\pi F} = \frac{5}{(j2\pi F+20-j2\pi(10))^2} + \frac{5}{(j2\pi F+20+j2\pi(10))^2}$$

$$X_c(j2\pi F) = \frac{5}{(20+j2\pi(F+10))^2} + \frac{5}{(20+j2\pi(F-10))^2}$$

(b) Plot magnitude and phase of $X_c(j2\pi F)$ over $-75 \leq F \leq 75$ Hz.

MATLAB script:

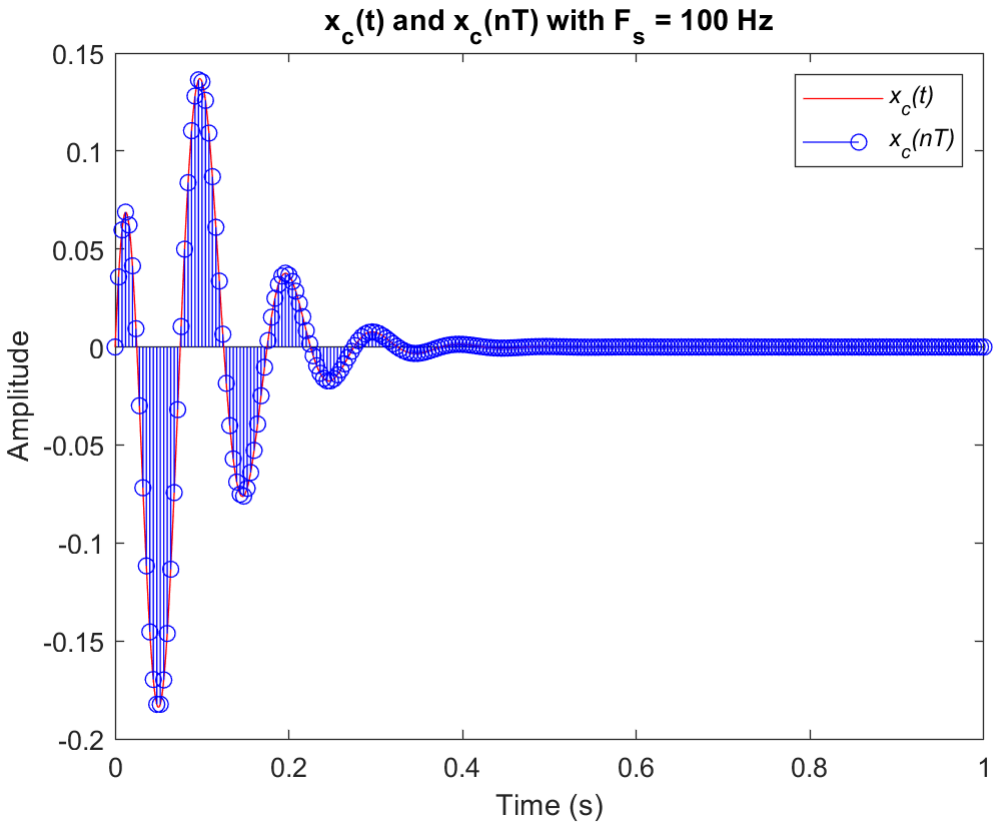
```
f = linspace(-75,75,1001);
Xc = (5./((20+1j*2*pi*(f+10)).^2))+(5./((20+1j*2*pi*(f-10)).^2));
Mag = abs(Xc);
Pha = angle(Xc);
figure('Units','inches','Position',[0,0,12,4])
subplot(1,2,1);
plot(f,Mag); xlim([-75 75]), xticks(-75:15:75), xlabel('Frequency (Hz)'), ylabel('|X_c(j2\piF)|
title('Magnitude Response of X_c(j2\piF)');
subplot(1,2,2);
plot(f,Pha); xlim([-75 75]), xticks(-75:15:75), xlabel('Frequency (Hz)'), ylabel('\angle X_c(j2
title('Phase Response of X_c(j2\piF)');
```



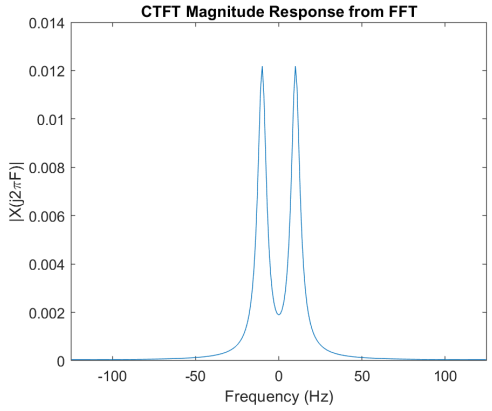
(c) Use the **`fft`** function to approximate CTFT computations. Choose sampling rate to minimize aliasing and the number of samples to capture most of the signal waveform. Plot magnitude and phase of your approximation and compare it with the plot in (b) above.

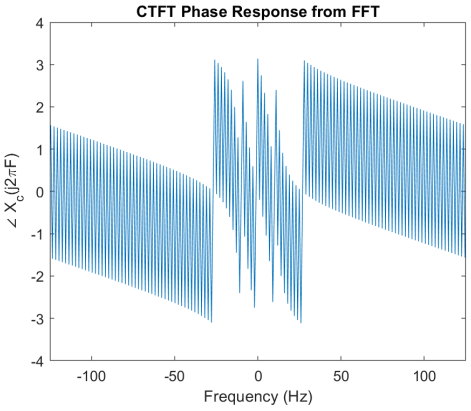
Solution:

```
t = 0:1/1000:0.5; xc = 10.*t.*exp(-20.*t).*cos(20*pi.*t);
Fs = 250; Ts = 1/Fs; N = 251; n = 0:N-1; nTs = n*Ts;
xnT = 10.*nTs.*exp(-20.*nTs).*cos(20.*pi.*nTs);
figure, plot(t,xc,'r'), hold on, stem(nTs,xnT,'b'), legend('\it{x_c(t)}','\it{x_c(nT)}')
xlabel('Time (s)'), ylabel('Amplitude'), title('x_c(t) and x_c(nT) with F_s = 100 Hz'), hold of
```




```
T1 = -0.5; T2 = 0.5;
Fs = 250; T = 1/Fs;
nTs = 0:T:1; xn = 10.*nTs.*exp(-20.*nTs).*cos(20.*pi.*nTs);
N = length(nTs); k = -(N-1)/2:(N-1)/2; Fk = k*Fs/N;
XcF = fftshift(T*fft(ifftshift(xn)));
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1), plot(Fk,abs(XcF)), xlim([-125 125]), title('CTFT Magnitude Response from FFT'),
subplot(1,2,2), plot(Fk,angle(XcF)), xlim([-125 125]), title('CTFT Phase Response from FFT'), x
```





Comparison with the plot in (b):

We can observe that the generated plots for the fft() function, do resemble the plots in part (b).The original signal $x_c(t)$ decays to zero around 0.5 seconds which factored into the choosing an appropriate sample frequency of $F_s = 250Hz$. This then gave us a sample period of $T_s = \frac{1}{F_s} = 0.004$ secs, and I was able to calculate the appropriate amount of samples for $N = (0.5 - (-0.5))/T + 1 \rightarrow 1/0.004 + 1 = 251$ samples. Then when we sample the continuous-time signal and use the fft() function, we see that the magnitude response is approximatedly identical to the computed CTFT in part b). However, the phase response had been affected by the fft and fftshift operations and is very jagged and rapid for variating frequencies.

Problem 4.2

Text Problem 7.29 (Page 427)

Let $x[n] = 10(0.5)^n \sin(0.1\pi n)u[n]$.

```
clc; close all; clear;
```

(a) Determine the DTFT $\widetilde{X}(e^{j\omega})$ of $x[n]$.

Solution:

The DTFT of the sequence $x[n]$ is

$$\widetilde{X}(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} = \sum_{n=-\infty}^{\infty} 10(0.5)^n \sin(0.1\pi n)u[n]e^{-j\omega n} = \sum_{n=0}^{N-1} 10(0.5)^n \sin(0.1\pi n)e^{-j\omega n}$$

Using the following Transform Pairs:

$$10(0.5)^n \sin(0.1\pi n)u[n] \leftrightarrow \frac{10(0.5)\sin(0.1\pi)e^{-j\omega}}{1-(2)(0.5)\cos(0.1\pi)e^{-j\omega}+(0.5)^2e^{-2j\omega}}$$

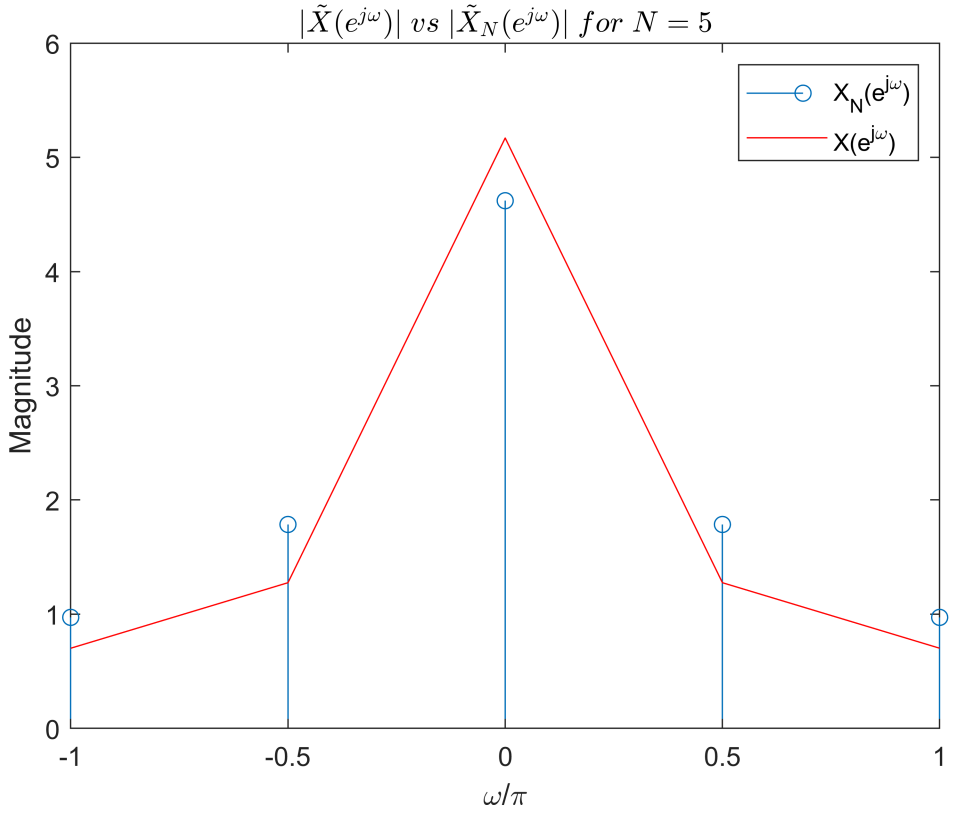
Then the DTFT of the sequence is

$$\widetilde{X}(e^{j\omega}) = \frac{1.5451e^{-j\omega}}{1 - 0.9511e^{-j\omega} + 0.25e^{-2j\omega}}$$

(b) Choose first $N = 5$ samples of $x[n]$ and compute the approximate DTFT $\widetilde{X}_N(e^{j\omega})$ using the **`fft`** function. Plot magnitudes of $\widetilde{X}(e^{j\omega})$ and $\widetilde{X}_N(e^{j\omega})$ in one plot and compare your results.

MATLAB script:

```
N = 5; n = 0:N-1; omega = linspace(-pi,pi,N);
xn = 10*(0.5.^n).*sin(0.1*pi.*n);
XN = fft(xn,N);
figure;
stem(omega/pi,abs(fftshift(XN)));
X = (1.5451*exp(-1i.*omega))./(1 - 0.9511*exp(-1i.*omega)+0.25*exp(-1i*2.*omega));
hold on;
plot(omega/pi,abs(X),'r');
xlabel('\omega/\pi'); xlim([-1 1])
ylabel('Magnitude'); legend('X_N(e^{j\omega})', 'X(e^{j\omega})')
title('$|\tilde{X}(e^{j\omega})|\; vs \; |\tilde{X}_N(e^{j\omega})| \; for \; N = 5$', 'Interpr
```



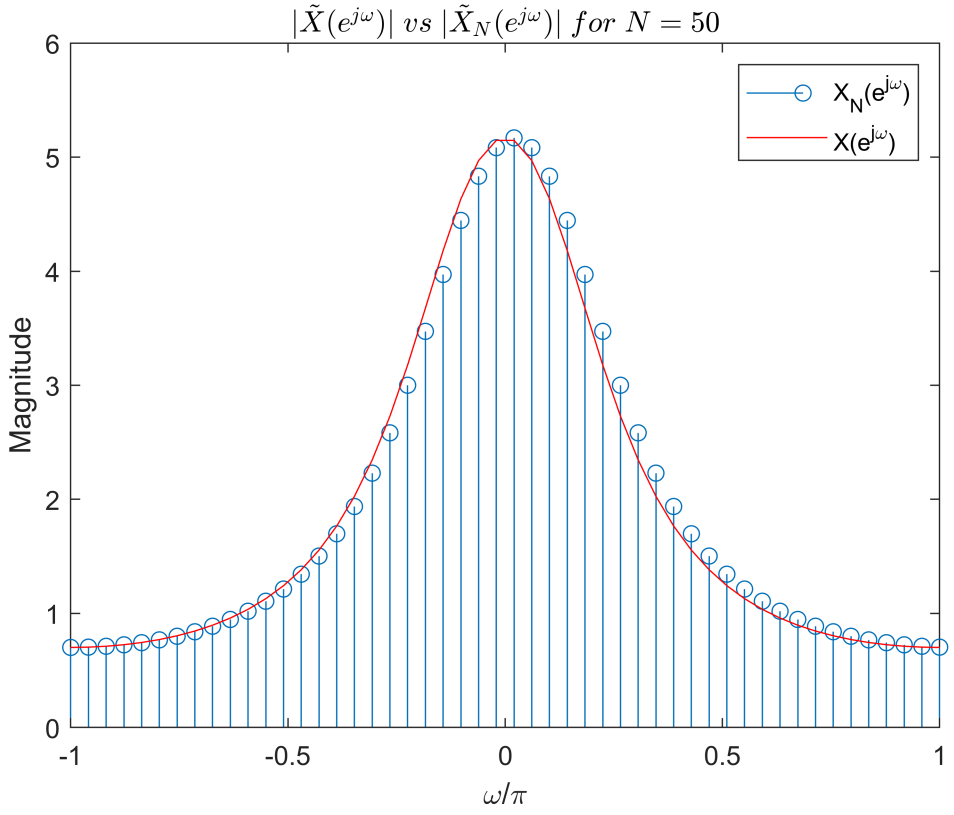
Comparison:

When N = 5, the resolution in the frequency response is not a smooth signal and very jagged, hence the triangular look to the magnitude responses. Sampling the time domain signal with $N = 5$ samples and then computing the DFT shows an approximation of the frequency information, but because our value of $N$was a small quantity of sample, the spectrum $\widetilde{X}_N(e^{j\omega})$ has aliasing in it's values compared to $\widetilde{X}(e^{j\omega})$.

(c) Repeat part (b) using $N = 50$.

MATLAB script:

```
N = 50; n = 0:N-1; omega = linspace(-pi,pi,N); u = (n>=0);
xn = 10*(0.5.^n).*sin(0.1*pi.*n).*u;
XN = fft(xn,N);
figure;
stem(omega/pi,abs(fftshift(XN)));
X = (1.5451*exp(-1i.*omega))./(1 - 0.9511*exp(-1i.*omega)+0.25*exp(-1i*2.*omega));
hold on;
plot(omega/pi,abs(X),'r');
xlabel('\omega/\pi'); xlim([-1 1])
ylabel('Magnitude');legend('X_N(e^{j\omega})', 'X(e^{j\omega})')
title('$|\tilde{X}(e^{j\omega})|\; vs \; |\tilde{X}_N(e^{j\omega})| \; for \; N = 50$', 'Interp
```



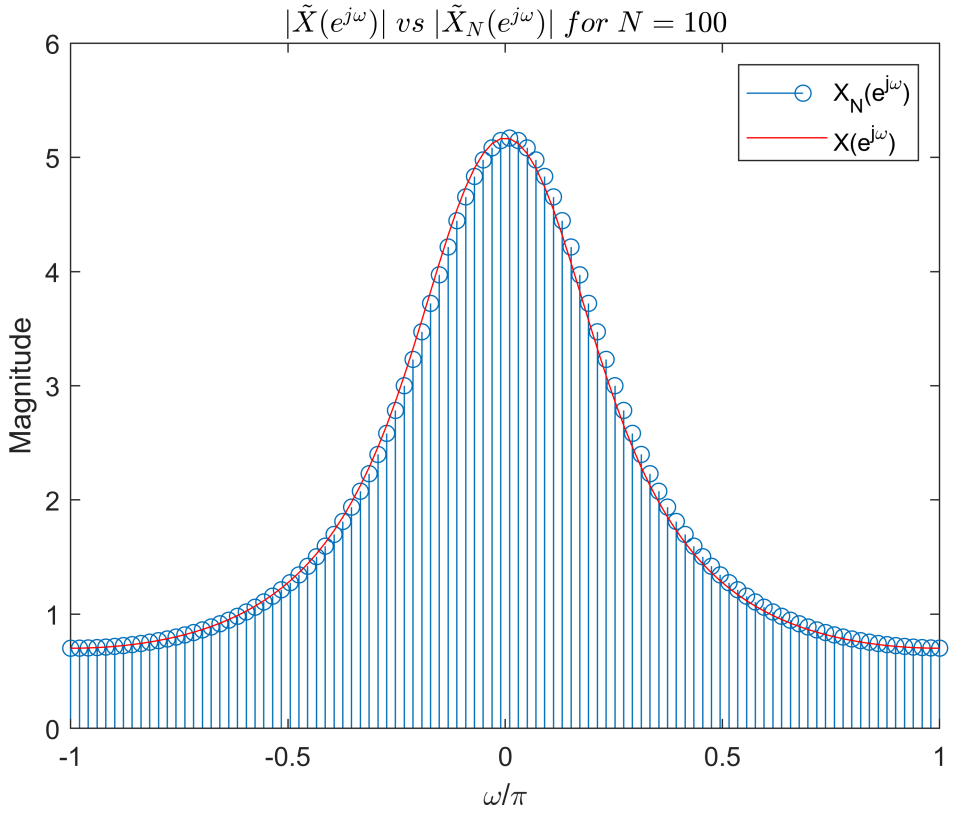
Comparison:

When $N = 50$ we now have a closer approximation of $\widetilde{X}_N(e^{j\omega})$ compared to $\widetilde{X}(e^{j\omega})$ given the shape of the magnitude response being a smoother signal than the previous plot. However, the values of $\widetilde{X}_N(e^{j\omega})$ are not exact, as the values are shifted slightly to the right of $\widetilde{X}(e^{j\omega})$.

(d) Repeat part (b) using $N = 100$.

MATLAB script:

```
N = 100; n = 0:N-1; omega = linspace(-pi,pi,N); u = (n>=0);
xn = 10*(0.5.^n).*sin(0.1*pi.*n).*u;
XN = fft(xn,N);
figure;
stem(omega/pi,abs(fftshift(XN)));
X = (1.5451*exp(-1i.*omega))./(1 - 0.9511*exp(-1i.*omega)+0.25*exp(-1i*2.*omega));
hold on;
plot(omega/pi,abs(X),'r');
xlabel('\omega/\pi'); xlim([-1 1])
ylabel('Magnitude');legend('X_N(e^{j\omega})', 'X(e^{j\omega})')
title('$|\tilde{X}(e^{j\omega})|\; vs \; |\tilde{X}_N(e^{j\omega})| \; for \; N = 100$', 'Inter
```



Comparison:

We can see now, when $N = 100$ the magnitude response of both $\widetilde{X}_N(e^{j\omega})$ and $\widetilde{X}(e^{j\omega})$ are almost identical showing that when $N = 100$ samples, the DFT of $x[n]$ closely resembles the magnitude response of $\widetilde{X}(e^{j\omega})$.

Problem 4.3

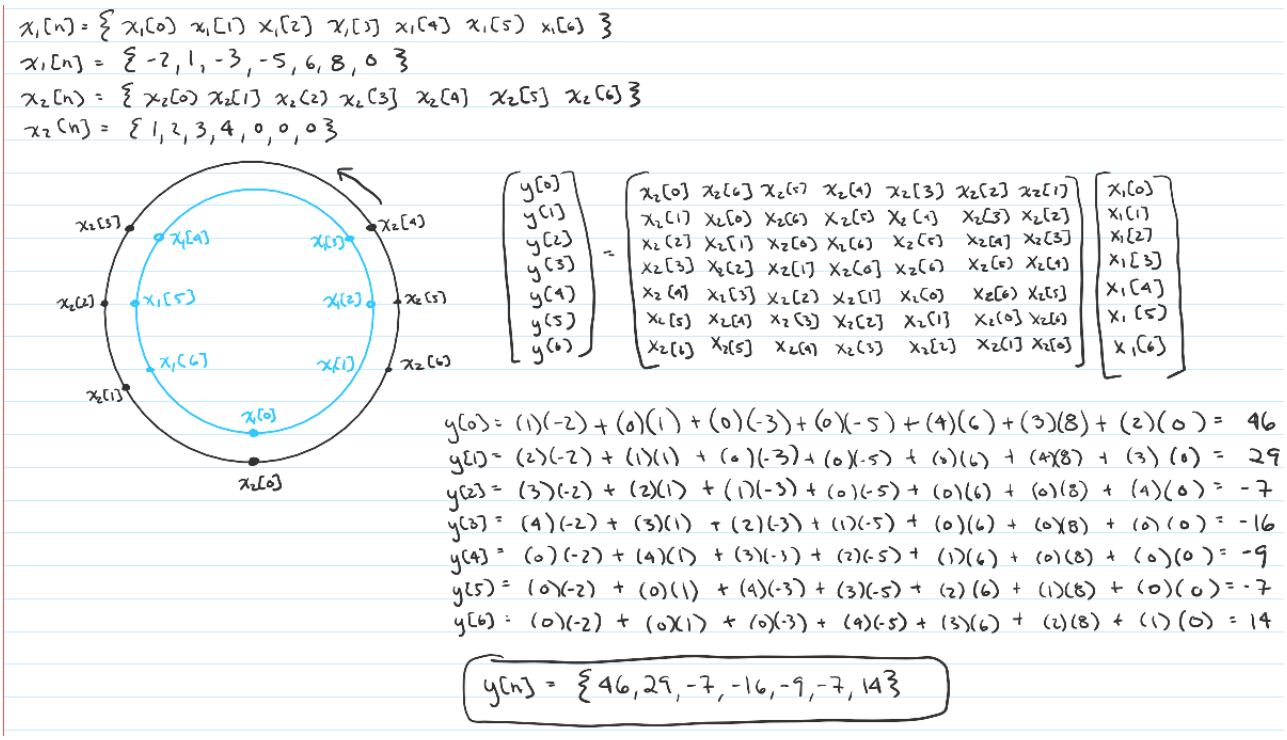
Text Problem 7.41 (Page 428)

Let $x_1[n] = \{\underset{\uparrow}{-2}, 1, -3, -5, 6, 8\}$ be a 6-point sequence and let $x_2[n] = \{\underset{\uparrow}{1}, 2, 3, 4\}$ be a 4-point sequence.

```
clc; close all;clear;
```

(a) Determine $x_1[n](7)x_2[n]$ using hand calculations.

Solution:



(b) Verify your calculations in (a) using the **`circonv`** function.

MATLAB script:

```
x1 = [-2 1 -3 -5 6 8 0]; x2 = [1 2 3 4 0 0 0];
y = circonv(x1,x2)'
```

```
y = 1×7
```

```
    46    29    -7   -16    -9    -7    14
```

(c) Verify your calculations in (a) by computing the DFTs and IDFT.

MATLAB script:

```
y = floor(ifft(fft(x1).*fft(x2)))
```

```
y = 1×7
    46    29    -7   -16   -10    -8    14
```

Problem 4.4

Text Problem 7.47 (Page 430)

The DFT in the Text equation (7.21) and the IDFT in (7.22) share the same complex-exponential term W_N^{nk} but with different signs.

```
clc; close all; clear;
```

(a) Show that it is possible to express the (7.22) as a DFT operation with additional processing steps. This approach can be used to obtain both transforms using one software function.

Solution:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{-kn}$$

$$Nx[n] = \sum_{k=0}^{N-1} X[k]W_N^{-kn}$$

We can perform a sign change in both $x[n]$ and W_N^{-kn} into

$$Nx[-n] = \sum_{k=0}^{N-1} X[k]W_N^{kn}$$

Then if we interchange the summation from k to n we can represent $X[k]$ as

$$X[k] \leftrightarrow Nx^*[\langle -n \rangle_N]$$

Thus, our Ncoefficients of the DFT can be represented by the complex conjugate of our original discrete sequence after being folded and mod by N

Which means, $x[n]$ can also be recovered by taking the fft of the complex conjugate $X^*[k]$

(b) Write a MATLAB function **`x = myifft(X)`** that implements your procedure in part (a) above. Use **`fft`** to implement the DFT computations.

MATLAB function: Enter your function code below after the comments for the TA to evaluate and grade. Create your function at the end of this file for it to execute properly.

```
function xn = myifft(X)
% Computes IDFT using FFT function only
% Enter your code below
Xconj = conj(X); % Complex Conjugate of X
N = length(X); % Length of X
xn = 1/N * conj(fft(Xconj)); % Derive xn from conjugate of X with fft function
end
```

(c) Let $x[n] = \sin(0.1\pi n), 0 \le n \le 9.$ Determine its DFT using the **`fft`** function and then its IDFT using **`myifft`** function to verify its accuracy.

MATLAB script:

```
N = 9; n = 0:N-1; xn = sin(0.1*pi.*n)
```

```
xn = 1×9
         0    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090 ···
```

```
X = fft(xn); x_ifft = abs(real(myifft(X)))
```

```
x_ifft = 1×9
    0.0000    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090 ···
```

The results of using the myifft() function is identical to the original discrete sequence.

Problem 4.5

Text Problem 7.50 (Page 430)

Let the DTFT $\widetilde{X}(e^{j\omega})$ of a sequence $x[n]$ be given by

$$\widetilde{X}(e^{j\omega}) = \frac{3}{5 - 4\cos(\omega)}. \qquad (4.5.1)$$

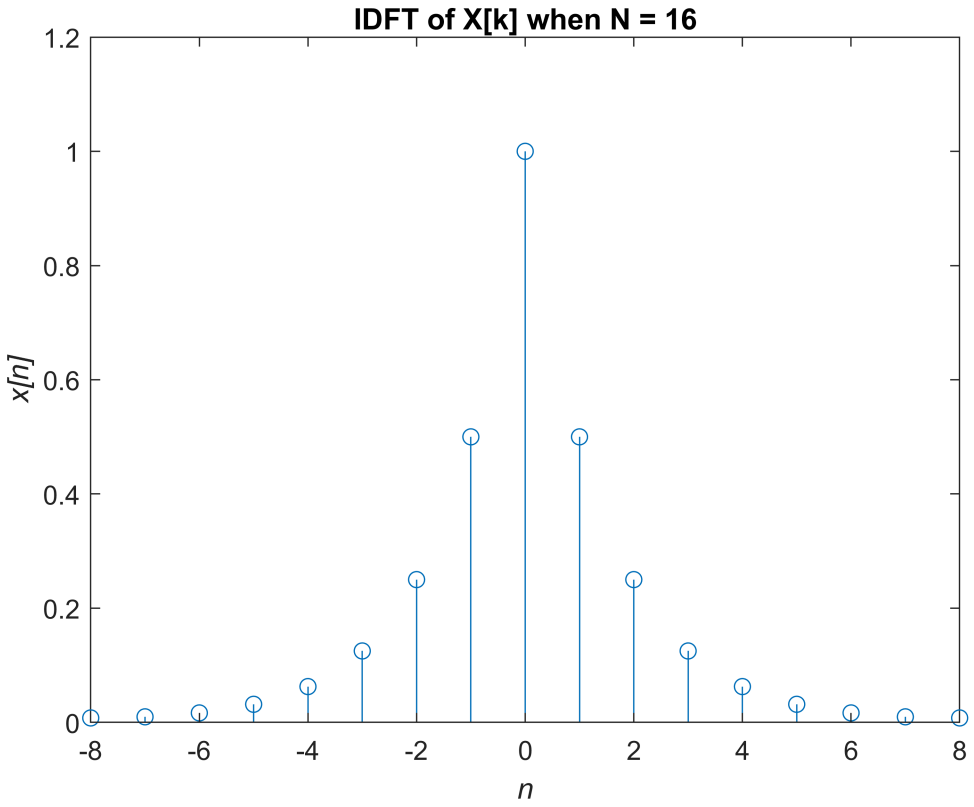
It is sampled at N equally-spaced frequencies to obtain $X[k] = \widetilde{X}(e^{j2\pi k/N})$ for $0 \le k \le N-1.$

```
clc; close all; clear;
```

(a) For $N = 16$ determine and **`stem`**-plot the sequence $x_1[n]$ from $-8 \le n \le 8$ by taking the IDFT of $X[k]$.

MATLAB script:

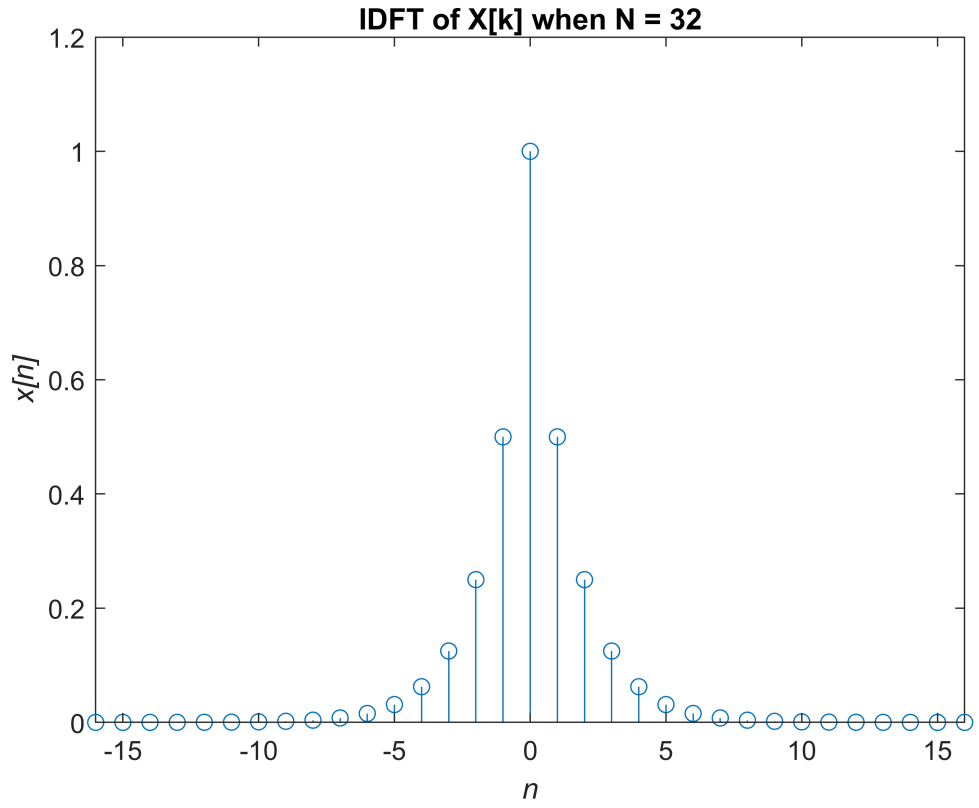
```
N = 16; k = 0:N-1; omega = (2*pi.*k)/N;  X = 3./(5 - 4*cos(omega));
x = ifftshift(real(ifft(X,N))); n = -N/2:N/2; x = [x, x(1)];
figure; stem(n,x), title('IDFT of X[k] when N = 16'), xlabel('\it{n}'), ylabel('\it{x[n]}')
```



(b) For $N = 32$ determine and **stem**-plot the sequence $x_2[n]$ from $-16 \leq n \leq 16$ by taking the IDFT of $X[k]$.

MATLAB script:

```
N = 32; k = 0:N-1; omega = (2*pi.*k)/N; X = 3./(5 - 4*cos(omega));
x = ifftshift(real(ifft(X,N))); n = -N/2:N/2; x = [x, x(1)];
figure, stem(n,x), title('IDFT of X[k] when N = 32'), xlabel('\it{n}'), ylabel('\it{x[n]}'), xl
```



(c) From your observations of the plots in (a) and (b) above, what do you think the original sequence $x[n]$ is?

Verify your answer by computing its DTFT and comparing it with the given $\widetilde{X}(e^{j\omega})$.

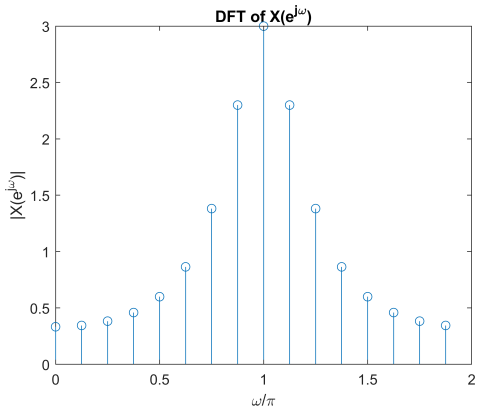
Solution:

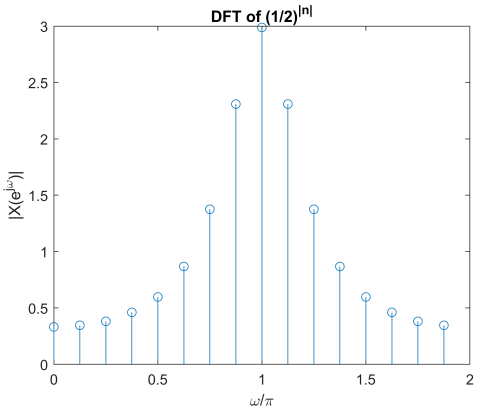
The shape from the plots in (a) and (b) appear to be some expontential raised to the absolute value of n.

After analyzing the discrete values after taking the IDFT, the arrays appear to be exponential multiples of $\frac{1}{2}$, thus our original sequence is

$$x[n] = \left(\frac{1}{2}\right)^{|n|}$$

```
N = 16; k = 0:N-1; omega = (2*pi.*k)/N;  Xtilde = 3./(5 - 4*cos(omega));
n = -N/2:N/2; xn = (1/2).^(abs(n)); XN = fft(xn,N);
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1), stem(omega/pi,fftshift(Xtilde)), title('DFT of X(e^{j\omega})'), xlabel('\omega
subplot(1,2,2), stem(omega/pi,abs(fftshift(XN))); xlabel('\omega/\pi'), ylabel('|X(e^{j\omega}))
```





Observing both $\widetilde{X}(e^{j\omega})$ and the DFT of sequence $x[n] = \left(\frac{1}{2}\right)^{|n|}$, both frequency responses have identical shapes and values, thus the original sequence

$$x[n] = \left(\frac{1}{2}\right)^{|n|}$$

Problem 4.6

Analysis of error between linear and circular convolutions

Let $x_1[n]$, $0 \le n < N_1$, be an N_1-point sequence. Let $x_2[n]$, $0 \le n < N_2$, be an N_2-point sequence. Let $x_3[n] \triangleq x_1[n] * x_2[n]$ be the **\textitlinear** convolution. Let $x_4[n] \triangleq x_1[n](N)x_2[n]$ be an N-point **\textitcircular** convolution where $N \ge \max(N_1, N_2)$.

(a) Show that, in general, the circular convolution is an aliased version of the linear convolution, that is,

$$x_4[n] = \left(\sum_{r=-\infty}^{\infty} x_3[n - rN]\right) p_N[n] \qquad (4.6.1)$$

where $p_N \triangleq u[n] - u[n - N]$ is the rectangular window.

Proof:

If we examine the definitions of both sequences, $x_3[n]$ and $x_4[n]$:

$$x_3[n] = x_1[n] * x_2[n] = \sum_{k=-\infty}^{\infty} x_1[k]x_2[n-k] = \sum_{k=0}^{N-1} x_1[k]x_2[n-k]$$

$$x_4[n] = \left[\sum_{m=0}^{N-1} x_1[m]x_2[(n-m)]_N\right] p_N[n]$$

$$x_4[n] = \left[\sum_{m=0}^{N-1} x_1[m] \sum_{r=\infty}^{\infty} x_2(n-m-rN)\right] p_N[n]$$

$$x_4[n] = \left[\sum_{r=\infty}^{\infty} \sum_{m=0}^{N-1} x_1(m) x_2(n-m-rN)\right] p_N[n]$$

Then recognizing the inner argument of the summation is equal to the linear convolution, $x_3[n]$ being periodically extended by rN, we get:

$$x_4[n] = \left[\sum_{r=\infty}^{\infty} x_3(n-rN)\right] p_N[n]$$

Thus, the circular convolution will be an aliased version of the linear convolution.

(b) Let $N \geq L \triangleq (N_1 + N_2 - 1)$ which is the length of the linear convolution. From $(4.6.1)$ show that

$$x_4[n] = x_3[n], \quad 0 \leq n \leq (N-1), \qquad (4.6.2)$$

which means that the circular convolution is same as the linear convolution over the given interval.

Solution:

$$x_4[n] = \left[\sum_{r=\infty}^{\infty} x_3(n-rN)\right] p_N[n]$$

When the length is greater than or equal to the length, $L \triangleq N_1 + N_2 - 1$, our window can be removed and we examine when $r = 0$

$$x_4[n] = \sum_{r=0} x_3(n-rN) = x_3(n-(0)N)$$

Thus,

$$x_4[n] = x_3[n], \qquad 0 \leq n \leq N-1$$

(c) Let $\max(N_1, N_2) \leq N < L$ and let $e_N[n] \triangleq x_4[n] - x_3[n]$, $0 \leq n \leq N-1$ be an error between the circular and the linear convolutions. From $(4.6.1)$ show that

$$e_N[n] = x_3[n+N], \quad 0 \leq n \leq (N-1). \qquad (4.6.3)$$

Thus, under the proper conditions, the error at each n is given by the linear convolution N samples away.

Solution:

With the error defined as:

$$e_N[n] \triangleq x_4[n] - x_3[n] = \left[\sum_{r\neq 0} x_3(n - rN)\right] p_N[n]$$

Since $N \geq max(N_1, N_2)$, only two terms will corresponding to $r \pm 1$ will remain, giving

$$e[n] = [x_3(n - N) + x_3(n + N)]p_N[n]$$

Then assuming the orignal signals were causal, their convolution sum will also be causal:

$$x_3(n - N) = 0; \qquad 0 \leq n \leq N - 1$$

Thus,

$$e_N[n] = x_3(n + N), \qquad 0 \leq n \leq N - 1$$

Which implies that when $\max(N_1, N_2) \leq N < L$ the error value at n is the same as the linear convolution value computed N samples away.

(d) Let $N = \max(N_1, N_2)$ and $M = \min(N_1, N_2)$. Then, using $(4.6.3)$, show that

$$e_N(n) = 0 \text{ for } n \geq (M - 1), \qquad (4.6.6)$$

that is, that the first $M - 1$ samples in $x_3[n]$ are in error but the remaining samples are the correct linear convolution samples $x_4[n]$, $(M - 1) \leq n \leq (N - 1)$.

Solution:

$$e_N[n] = x_3[n + N], \quad 0 \leq n \leq (N - 1).$$

substituting $e_N[n] = x_4 - x_3$ and the fact that $x_4 = x_3[n + N] + x_3[n - N]$:

$$e_N[n] = x_3[n + N] + x_3[n - N] - x_3$$

Again, assuming the original sequences are causal and $M = min(N_1, N_2)$:

$$e_N[n] = x_3[n + M] - x_3$$

Then we observe that the first $M - 1$ sample will be shifted to the left and the remaining terms in the error calculation will result in zeros. Thus,

$$e_N[n] = 0, \qquad n \geq (M - 1)$$

Problem 4.7

This problem numerically verifies equations (4.6.3) and (4.6.6) from Problem 4.6 above.

For the following sequences compute (i) the linear convolution $x_3[n] = x_1[n] * x_2[n]$, (ii) the N-point circular convolution $x_3[n] = x_1[n] (N) x_2[n]$, and (iii) the error sequence $e_N[n]$ in $(4.6.3)$.

(a) $x_1[n] = (0.8)^n p_{10}[n], \quad x_2[n] = (-0.8)^n p_{10}[n]; \quad N = 15$. Do this part using MATLAB.

MATLAB script:

```
clc; close all; clear;
N = 15; n = 0:N-1; u = ((n>=0)&(n<10));
x1 = (0.8.^n).*u; x2 = (-0.8.^n).*u;
x3 = conv(x1,x2); x4 = cconv(x1,x2,N); L1 = length(x3), L2 = length(x4), L = L2-L1, x4 = [x4 ze
```

```
L1 = 29
L2 = 15
L = -14
```

```
en = abs(x4 - x3); display(en(1:N))
```

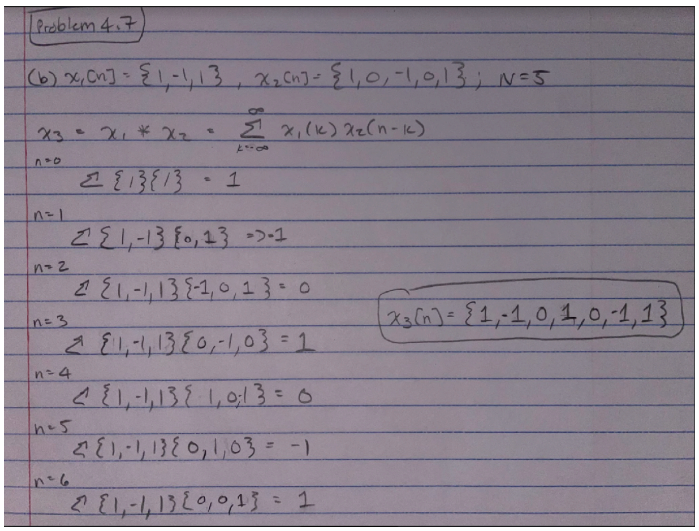
```
    0.1407    0.0844    0.0450    0.0180         0         0    0.0000         0         0    0.0000    0.0000
```

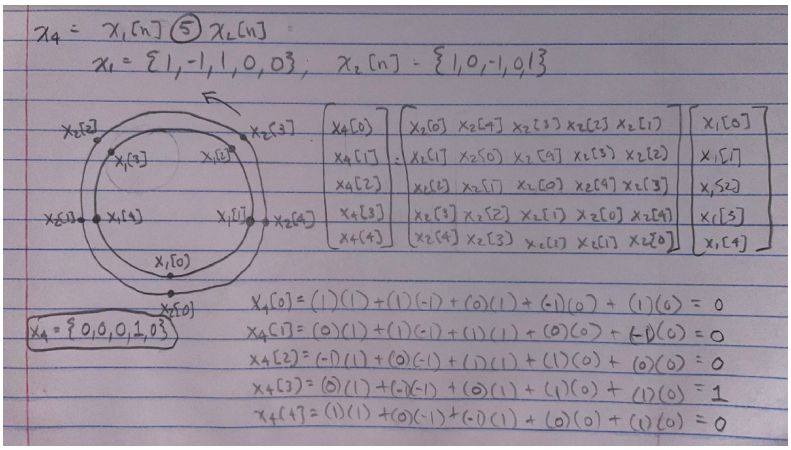
Thus, the first 4 samples are in error and these samples repeat from our linear convolution sequence for

$e_N[n] = x_3(n + 15), \qquad 0 \le n \le 14$

(b) $x_1[n] = \{1, -1, 1\}, \quad x_2[n] = \{1, 0, -1, 0, 1\}; \quad N = 5$. Do this part using hand calculations.

Solution:





Thus,

$x_3[n] = \{1, -1, 0, 1, 0, -1, 1\}$ and $x_4[n] = \{0, 0, 0, 1, 0\}$

This means the error is

$$e_N[n] = x_4[n] - x_3[n]$$

Observing that the first two samples in $x_4[n]$ are in error, our error sequence can be defined as:

$$e_N[n] = x_3(n+5)$$

Problem 4.8

Text Problem 7.43 (Page 429)

Let $x_1[n]$ be an N_1-point sequence and let $x_2[n]$ be an N_2-point sequence. Let $N \geq \max(N_1, N_2)$. Their N-point circular convolution is shown to be equal to the aliased version of their linear convolution in Problem 4.6 above. This result can be used to compute the circular convolution via the linear convolution.

(a) Develop a MATLAB function **`y = lin2circonv(x,h)`** that implements this approach.

Solution:

MATLAB function: Enter your function code below after the comments for the TA to evaluate and grade. Create your function at the end of this file for it to execute properly.

```
function [y] = lin2circonv(x,h)
%LIN2CIRCONV Linear to Circular Convolution using Aliasing Formula
%   y = lin2circonv(x,h)
%   x and h are assumed to be causal sequences
% Enter your code below
N1 = length(x); N2 = length(h); % Determine Lengths of sequences
N = max(N1,N2); y_lin = conv(x,h); % Compute Max length and the linear convolution
en = y_lin(N+1:end); en = [en zeros(1,N - length(en))]; % Compute the error and zero pad
y_lin = y_lin(1:N); % Only take N samples from Linear Convolution
```

```
y = y_lin + en; % Circular Convolution is equal to the Linear Convolution plus the Error
end
```

(b) For $x[n] = \{1, 2, 3, 4\}$ and $h[n] = \{1, -1, 1, -1\}$, determine their 4-point circular convolution using the **`lin2circonv`** function and verify using the **`circonv`** function

MATLAB script:

```
clc; close all; clear;
x = [1 2 3 4]; h = [1 -1 1 -1];
y = lin2circonv(x,h), y1 = circonv(x,h)'
```

```
y = 1×4
    -2     2    -2     2
y1 = 1×4
    -2     2    -2     2
```

Values for both the **lin2circonv()** and **circonv()** function are identical, which means the user-defined function successfully computes the circular convolution using the linear convolution of the input sequences.

Create your MATLAB functions below.

```
function xn = myifft(X)
% Computes IDFT using FFT function only
% Enter your code below
Xconj = conj(X); % Complex Conjugate of X
N = length(X); % Length of X
xn = 1/N * conj(fft(Xconj)); % Derive xn from conjugate of X with fft function
end

function [y] = lin2circonv(x,h)
%LIN2CIRCONV Linear to Circular Convolution using Aliasing Formula
%   y = lin2circonv(x,h)
%   x and h are assumed to be causal sequences
% Enter your code below
N1 = length(x); N2 = length(h); % Determine Lengths of sequences
N = max(N1,N2); y_lin = conv(x,h); % Compute Max length and the linear convolution
en = y_lin(N+1:end); en = [en zeros(1,N - length(en))]; % Compute the error and zero pad
y_lin = y_lin(1:N); % Only take N samples from Linear Convolution
y = y_lin + en; % Circular Convolution is equal to the Linear Convolution plus the Error
end
```