EECE5666 (DSP) : Homework-4 Solutions

Table of Contents

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;
```

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

Solution: From the CTFT table in the "LS_Brief_Review" document, the CTFT of $y_c(t) = e^{-20t}\cos(20\pi t)u(t)$ is

$$Y_c(j\Omega) = \mathscr{F}\left[e^{-20t}\cos(20\pi t)u(t)\right] = \frac{20 + j\Omega}{(20 + j\Omega)^2 + (20\pi)^2}.$$

Since $x_c(t) = 10ty_c(t)$, then using the CTFT property $\mathscr{F}\{ty_c(t)\} = \jmath\dfrac{dY_c(\jmath\Omega)}{d\Omega}$, the CTFT of $x_c(t)$ is $10\jmath\dfrac{dY_c(\jmath\Omega)}{d\Omega}$ or

$$x_c(\jmath\Omega) = 10\jmath\frac{d}{d\Omega}\left(\frac{20+\jmath\Omega}{(20+\jmath\Omega)^2+(20\pi)^2}\right) = \frac{10\left[(20+\jmath\Omega)^2-(20\pi)^2\right]}{\left[(20+\jmath\Omega)^2+(20\pi)^2\right]^2}$$

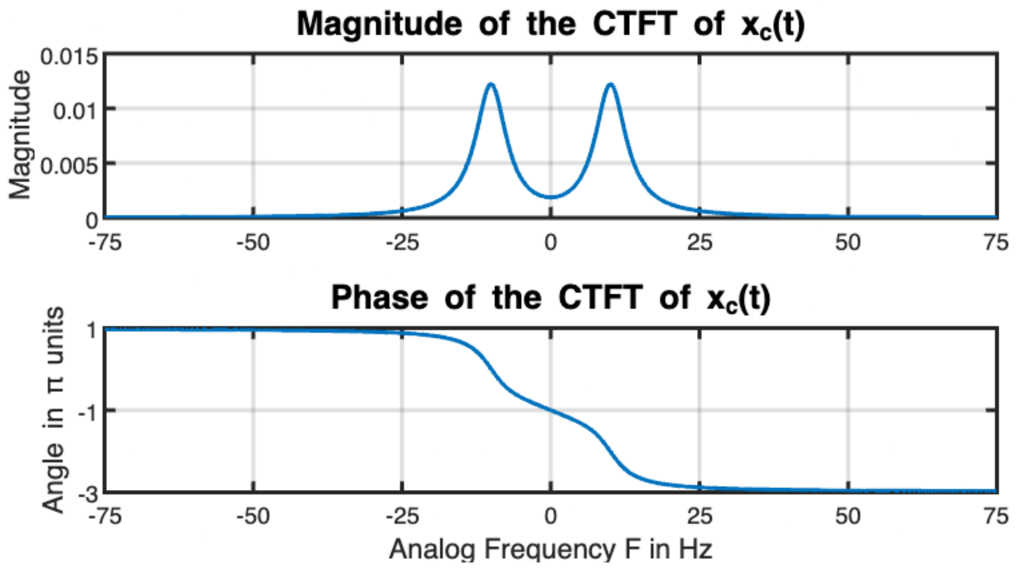
or

$$x_c(\jmath 2\pi F) = \frac{10\left[(20+\jmath 2\pi F)^2-(20\pi)^2\right]}{\left[(20+\jmath 2\pi F)^2+(20\pi)^2\right]^2}.$$

(b) Plot magnitude and phase of $X_c(\jmath 2\pi F)$ over $-75 \le F \le 75$ Hz.

MATLAB script: Computation and plotting of the CTFT is done using the following script.

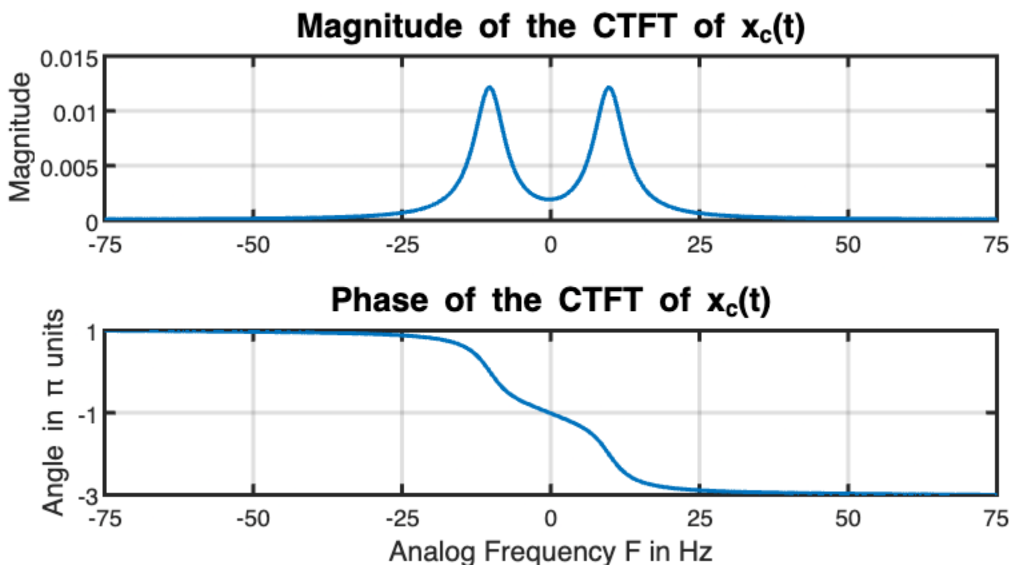
```
F = linspace(-75,75,1001);
Xc = -10*((20*pi)^2-(2j*pi*F+20).^2)./((20*pi)^2+(2j*pi*F+20).^2).^2;
figure('position',[0,0,8,4]*72);
subplot(2,1,1); plot(F,abs(Xc),'linewidth',1.5); axis([-75,75,0,0.015]);
ylabel('Magnitude'); title('Magnitude of the CTFT of x_c(t)');
set(gca,'xtick',(-75:25:75),'ytick',(0:0.005:0.015)); grid;
subplot(2,1,2); plot(F,unwrap(angle(Xc))/pi,'linewidth',1.5);
ylabel('Angle in \pi units'); title('Phase of the CTFT of x_c(t)');
xlabel('Analog Frequency F in Hz'); axis([-75,75,-3,1]);
set(gca,'xtick',(-75:25:75),'ytick',(-3:2:1)); grid;
```



(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: From the above plot, a reasonable value of the bandwidth of $x_c(t)$ is 75 Hz. Thus, we will choose $F_s = 150$ Hz to minimize the frequency-domain aliasing. Also, since e^{-20t} is zero in approximately $5/20 = 0.25$ sec, we will choose time-width of about 2 seconds to capture most of the non-zero signal waveform which will minimize the time-domain aliasing. This will give us about 300 samples. Now we can use FFT to numerically compute the CTFT of $x_c(t)$. These calculations are given in the script below.

```
Fs = 150; T = 1/Fs; t1 =0; t2 = 2; nT = t1:T:t2; N = length(nT);
xnT = 10*nT.*exp(-20*nT).*cos(20*pi*nT);
X = fftshift(fft(xnT)); Xc_approx = T*[X,X(1)];
om = linspace(-1,1,N+1)*pi; F = om*Fs/(2*pi);
figure('position',[0,0,8,4]*72);
subplot(2,1,1); plot(F,abs(Xc_approx),'linewidth',1.5); axis([-75,75,0,0.015]);
ylabel('Magnitude'); title('Magnitude of the CTFT of x_c(t)');
set(gca,'xtick',(-75:25:75),'ytick',(0:0.005:0.015)); grid;
subplot(2,1,2); plot(F,unwrap(angle(Xc_approx))/pi,'linewidth',1.5);
ylabel('Angle in \pi units'); title('Phase of the CTFT of x_c(t)');
xlabel('Analog Frequency F in Hz'); axis([-75,75,-3,1]);
set(gca,'xtick',(-75:25:75),'ytick',(-3:2:1)); grid;
```



The plots shown above agree with those in part (b).

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 $\tilde{X}(e^{j\omega})$ of $x[n]$.

Solution: Using the z-transform table, we have

$$\widetilde{X}(e^{j\omega}) = \mathcal{Z}\{x[n]\}\big|_{z=e^{j\omega}} = 10\mathcal{Z}\{(0.5)^n \sin(0.1\pi n)u[n]\}\big|_{z=e^{j\omega}}$$

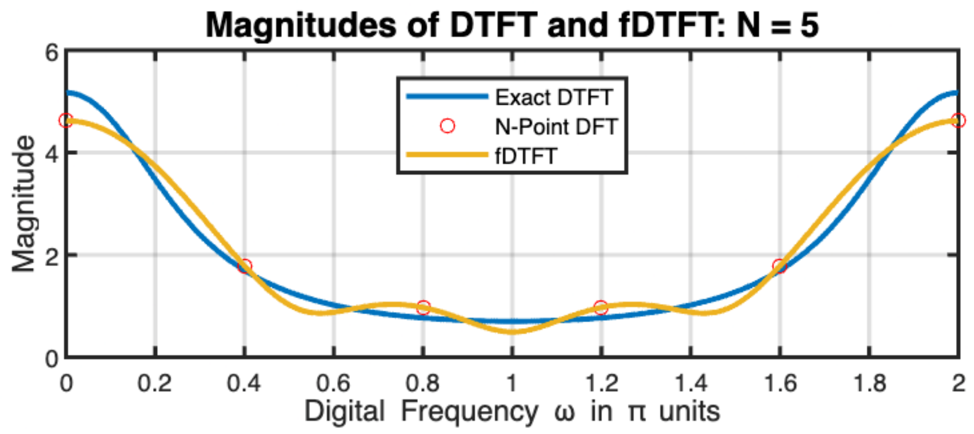
$$= 10\left\{\frac{0.5\sin(0.1\pi)z^{-1}}{1-2(0.5)\cos(0.1\pi)z^{-1}+0.25z^{-2}}\right\}\Bigg|_{z=e^{j\omega}} = \frac{5\sin(0.1\pi)e^{-j\omega}}{1-\cos(0.1\pi)e^{-j\omega}+0.25e^{-j2\omega}}$$

$$= \frac{1.5451e^{-j\omega}}{1-0.9511e^{-j\omega}+0.25e^{-j2\omega}}.$$

(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: Computation and plotting is done using the following script.

```
% Exact DTFT
K = 1024; om = linspace(0,2,K+1)*pi;
X = 5*sin(0.1*pi)*exp(-1j*om)./(1-cos(0.1*pi)*exp(-1j*om)+0.25*exp(-2j*om));
Xmag = abs(X);
% Finite-DTFT using Zero-Padding: N = 5
N = 5; n = 0:N-1; xn = 10*(0.5.^n).*sin(0.1*pi*n);
XN = fft(xn,N); XNmag = abs([XN,XN(1)]); % N-point DFT
k = 0:N; omk = 2*pi*k/N; % Frequency samples for N-point DFT
XK = fft(xn,K); XKmag = abs([XK,XK(1)]); % Zero-padded K-point DFT
figure('position',[0,0,8,3]*72);
plot(om/pi,Xmag,'linewidth',2); axis([0,2,0,6]); hold on; % DTFT
plot(omk/pi,XNmag,'ro','markersize',5); % N-point DFT samples
plot(om/pi,XKmag,'linewidth',2); % f-DTFT
xlabel('Digital Frequency \omega in \pi units');
ylabel('Magnitude'); title('Magnitudes of DTFT and fDTFT: N = 5');
legend('Exact DTFT','N-Point DFT','fDTFT','location','best'); grid;
```

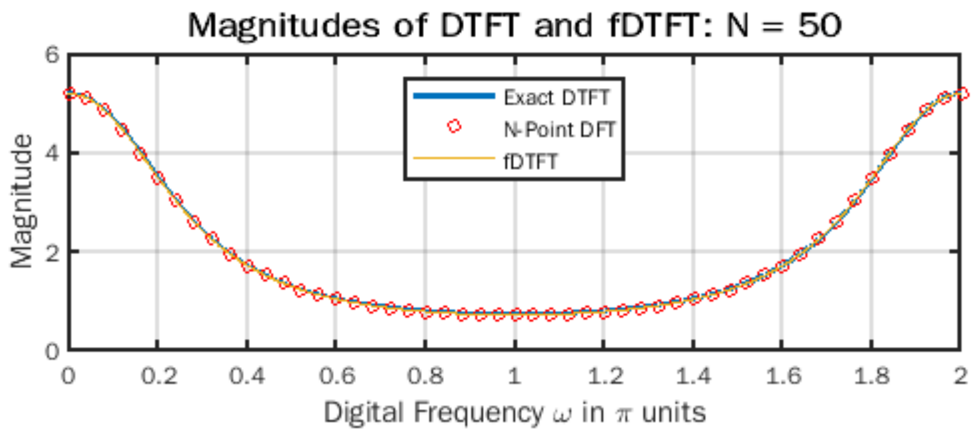


From the plot above, the approximate fDTFT is not the same as the exact DTFT.

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

MATLAB script: Computation and plotting is done using the following script.

```
N = 50; n = 0:N-1; xn = 10*(0.5.^n).*sin(0.1*pi*n);
XN = fft(xn,N); XNmag = abs([XN,XN(1)]); % N-point DFT
k = 0:N; omk = 2*pi*k/N; % Frequency samples for N-point DFT
XK = fft(xn,K); XKmag = abs([XK,XK(1)]); % Zero-padded K-point DFT
figure('position',[0,0,8,3]*72);
plot(om/pi,Xmag,'linewidth',2); axis([0,2,0,6]); hold on; % DTFT
plot(omk/pi,XNmag,'ro','markersize',5); % N-point DFT samples
plot(om/pi,XKmag,'linewidth',1); % f-DTFT
xlabel('Digital Frequency \omega in \pi units');
ylabel('Magnitude'); title('Magnitudes of DTFT and fDTFT: N = 50');
legend('Exact DTFT','N-Point DFT','fDTFT','location','best'); grid;
```

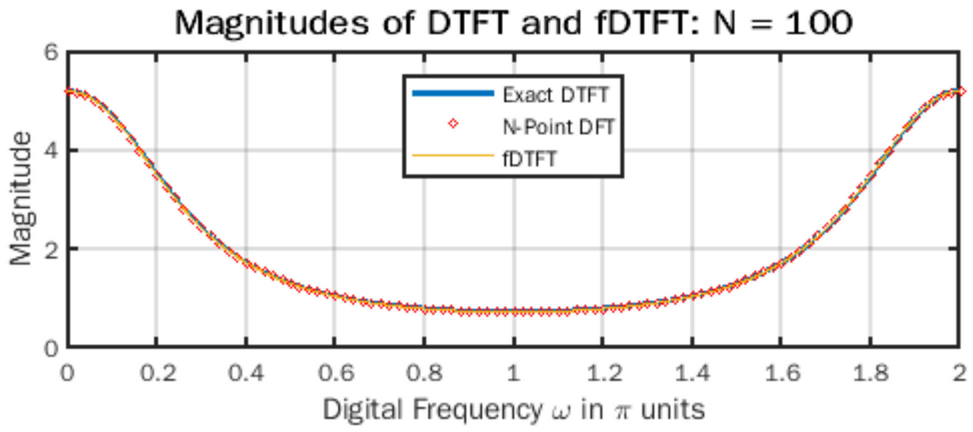


The exact DTFT and approximate fDTFT are almost the same.

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

MATLAB script: Computation and plotting is done using the following script.

```
N = 100; n = 0:N-1; xn = 10*(0.5.^n).*sin(0.1*pi*n);
XN = fft(xn,N); XNmag = abs([XN,XN(1)]); % N-point DFT
k = 0:N; omk = 2*pi*k/N; % Frequency samples for N-point DFT
XK = fft(xn,K); XKmag = abs([XK,XK(1)]); % Zero-padded K-point DFT
figure('position',[0,0,8,3]*72);
plot(om/pi,Xmag,'linewidth',2); axis([0,2,0,6]); hold on; % DTFT
plot(omk/pi,XNmag,'ro','markersize',3); % N-point DFT samples
plot(om/pi,XKmag,'linewidth',1); % f-DTFT
xlabel('Digital Frequency \omega in \pi units');
ylabel('Magnitude'); title('Magnitudes of DTFT and fDTFT: N = 100');
legend('Exact DTFT','N-Point DFT','fDTFT','location','best'); grid;
```



The exact DTFT and approximate fDTFT are practically exact.

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 the 7-point circular convolution, $x_1[n]⑦x_2[n]$, using hand calculations.

Solution: To compute the 7-point circular convolution, let us consider the two sequences as 7-point sequences using zero-padding. Let $x_1[n] = \{\underset{\uparrow}{-2}, 1, -3, -5, 6, 8, 0\}$ and $x_2[n] = \{\underset{\uparrow}{1}, 2, 3, 4, 0, 0, 0\}$. Let $x_3[n]$ be the 7-point circular convolution between $x_1[n]$ and $x_2[n]$. It is given by

$$x_3[n] = \sum_{k=0}^{6} x_1[n] x_2[\langle n-k\rangle_7]$$

Then

$$x_3[0] = \sum_{k=0}^{6} x_1[k] x_2\langle -k\rangle_7] = \sum[\{\underset{\uparrow}{-2}, 1, -3, -5, 6, 8, 0\} \times \{\underset{\uparrow}{1}, 0, 0, 0, 4, 3, 2\}]$$
$$x_3[0] = \sum[-2, 0, 0, 0, 24, 24, 0] = 46.$$

Next,

$$x_3[1] = \sum_{k=0}^{6} x_1[k] x_2\langle 1-k\rangle_7] = \sum[\{\underset{\uparrow}{-2}, 1, -3, -5, 6, 8, 0\} \times \{\underset{\uparrow}{2}, 1, 0, 0, 0, 4, 3\}]$$
$$x_3[1] = \sum[-4, 1, 0, 0, 0, 32, 0] = 29.$$

Similarly,

$$x_3[2] = \sum[\{\underset{\uparrow}{-2}, 1, -3, -5, 6, 8, 0\} \times \{\underset{\uparrow}{3}, 2, 1, 0, 0, 0, 4\}] = \sum[-6, 2, -3, 0, 0, 0, 0] = -7,$$

$$x_3[3] = \sum[\{\underset{\uparrow}{-2}, 1, -3, -5, 6, 8, 0\} \times \{\underset{\uparrow}{4}, 3, 2, 1, 0, 0, 0\}] = \sum[-8, 3, -6, -5, 0, 0, 0] = -16,$$

$$x_3[4] = \sum[\{\underset{\uparrow}{-2}, 1, -3, -5, 6, 8, 0\} \times \{\underset{\uparrow}{0}, 4, 3, 2, 1, 0, 0\}] = \sum[0, 4, -9, -10, 6, 0, 0] = -9,$$

$$x_3[5] = \sum[\{\underset{\uparrow}{-2}, 1, -3, -5, 6, 8, 0\} \times \{\underset{\uparrow}{0}, 0, 4, 3, 2, 1, 0\}] = \sum[0, 0, 4, -12, -15, 12, 8, 0] = -7,$$

$$x_3[6] = \sum[\{\underset{\uparrow}{-2}, 1, -3, -5, 6, 8, 0\} \times \{\underset{\uparrow}{0}, 0, 0, 4, 3, 2, 1\}] = \sum[0, 0, 0, -20, 18, 16, 0] = 14.$$

Thus $x_3[n] = \{\underset{\uparrow}{46}, 29, -7, -16, -9, -7, 14\}$.

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

MATLAB script: Computation is done using the following script.

```
x1 = [-2,1,-3,-5,6,8,0]'; x2 = [1,2,3,4,0,0,0]';
x3 = circonv(x1,x2); x3 = x3'; display(x3);
```

```
x3 = 1×7
    46    29    -7   -16    -9    -7    14
```

which agrees with hand calculations in (a).

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

MATLAB script: Computation is done using the following script.

```
X1 = fft(x1,7); X2 = fft(x2,7); X3 = X1.*X2;
x3 = real(ifft(X3,7)); x3 = round(x3(:)')
```

```
x3 = 1×7
    46    29    -7   -16    -9    -7    14
```

which also agrees with hand calculations in (a).

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: The DFT and IDFT pairs are given by

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \qquad (7.21)$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-nk} \qquad (7.22)$$

Now (7.22) can be expressed as

$$x[n] = \frac{1}{N}\left(\sum_{k=0}^{N-1} X^*[k] W_N^{nk}\right)^*$$

Note that the expression inside the parenthesis is a DFT of $X^*[k]$. Thus the sequence of operations are

$$X[k] \xrightarrow{\text{Conjugate}} X^*[k] \xrightarrow{\text{DFT}} \text{DFT}\left[X^*[k]\right] = y[n] \xrightarrow{\text{Conjugate}} y^*[n] \xrightarrow{\text{Divide by } N} x[n]$$

(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 x = myifft(X)
% Computes IDFT using FFT function only
% Enter your code below

N = length(X);
x = fft(X')'/N; % Uses "'" for conjugate operation twice
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.

Solution: Verification of the **`myifft`** function is done using the following script.

```
n = 0:9; xn = sin(0.1*pi*n); X = fft(xn);
xncheck = myifft(X);
error = max(abs(xn-xncheck))
```

```
error = 2.2204e-16
```

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 $0 \le k \le N-1$.

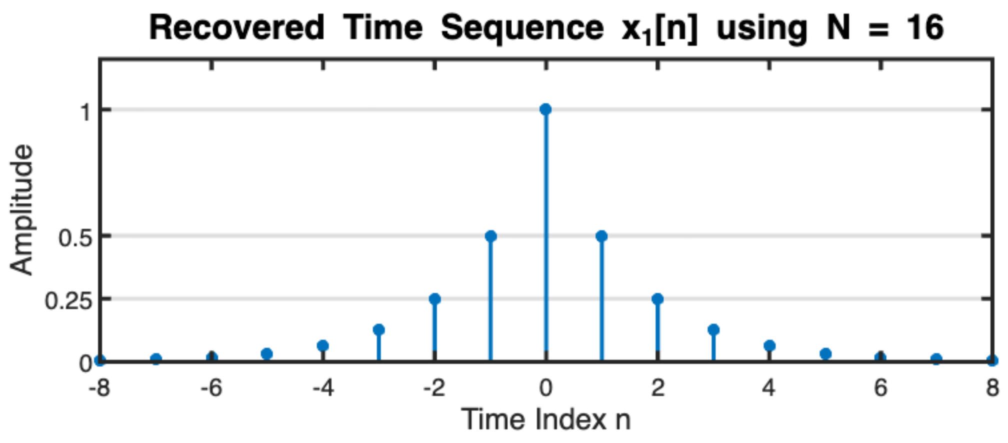
```
clc; close all; clear;
```

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

$$X[k] = \widetilde{X}(e^{j2\pi k/N}) = \frac{3}{5 - 4\cos(2\pi k/N)}, \quad k = 0, 1, \ldots, N-1.$$

After computing the IDFT of $X[k]$, we will rearrange the resulting sequence using **`ifftshift`** for plotting purposes.

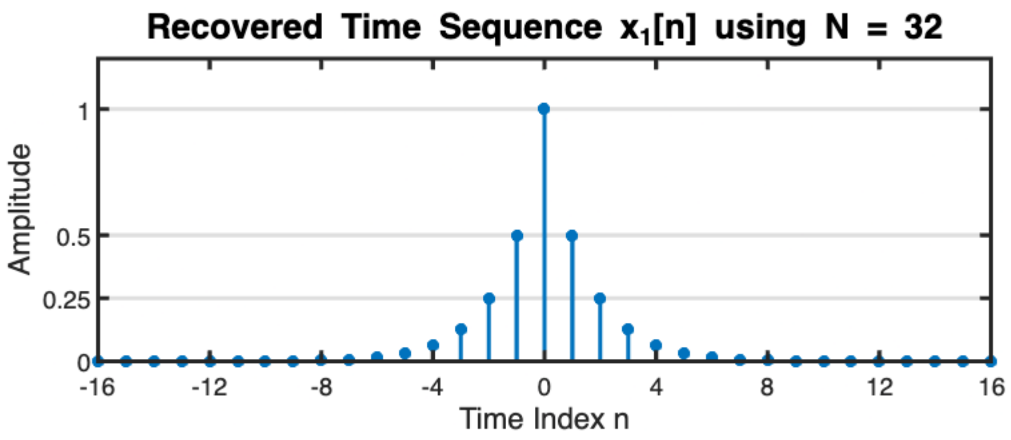
```
N = 16; n = -N/2:N/2; % For plotting x1[n]
k = 0:N-1; Xk = 3./(5-4*cos(2*pi*k/N));
x1n = ifft(Xk); x1n = real(ifftshift(x1n)); x1n = [x1n,x1n(1)];
figure('position',[0,0,8,3]*72);
stem(n,x1n,'filled','linewidth',1.5,'markersize',3); axis([-N/2,N/2,0,1.2]);
xlabel('Time Index n'); ylabel('Amplitude');
title('Recovered Time Sequence x_1[n] using N = 16');
set(gca,'ytick',[0,0.25,0.5,1],'ygrid','on');
```



(b) For $N = 32$ determine and **`stem`**-plot the sequence $x_2[n]$ from $-16 \le n \le 16$ by taking the IDFT of $X[k]$.
Solution: The following script computes and plots $x_2[n]$.

```
N = 32; n = -N/2:N/2; % For plotting x1[n]
```

```
k = 0:N-1; Xk = 3./(5-4*cos(2*pi*k/N));
x1n = ifft(Xk); x1n = real(ifftshift(x1n)); x1n = [x1n,x1n(1)];
figure('position',[0,0,8,3]*72);
stem(n,x1n,'filled','linewidth',1.5,'markersize',3); axis([-N/2,N/2,0,1.2]);
xlabel('Time Index n'); ylabel('Amplitude');
title('Recovered Time Sequence x_1[n] using N = 32');
set(gca,'ytick',[0,0.25,0.5,1],'ygrid','on','xtick',(-16:4:16));
```



(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: From the close examination of the above two plots, it appears that the original sequence $x[n]$ is:

$$x[n] = (0.5)^{|n|} \text{ for all } n.$$

To verify this conjecture, the DTFT of $x[n]$ is

$$\begin{aligned}\widetilde{X}(e^{j\omega}) &= \sum_{n=-\infty}^{\infty}\left(\frac{1}{2}\right)^{|n|}e^{-j\omega n} = 1 + \sum_{n=-\infty}^{-1}\left(\frac{1}{2}\right)^{-n}e^{-j\omega n} + \sum_{n=1}^{\infty}\left(\frac{1}{2}\right)^{n}e^{-j\omega n}\\ &= 1 + \sum_{n=1}^{\infty}\left(\frac{1}{2}\right)^{n}e^{j\omega n} + \sum_{n=1}^{\infty}\left(\frac{1}{2}\right)^{n}e^{-j\omega n} = 1 + \frac{\frac{1}{2}e^{j\omega}}{1-\frac{1}{2}e^{j\omega}} + \frac{\frac{1}{2}e^{-j\omega}}{1-\frac{1}{2}e^{-j\omega}}\\ &= \frac{3}{5-4\cos\omega}.\end{aligned}$$

which agrees with the DTFT given in (4.5.1).

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 ***linear*** convolution. Let $x_4[n] \triangleq x_1[n](N)x_2[n]$ be an N-point ***circular*** 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: Consider the circular convolution

$$\begin{aligned} x_4[n] &= x_1[n](N)x_2[n] = \left[\sum_{m=0}^{N-1} x_1[m] x_2[\langle n-m\rangle_N]\right] p_N[n] \\ &= \left[\sum_{m=0}^{N-1} x_1[m] \sum_{r=-\infty}^{\infty} x_2[n-m-rN]\right] p_N[n] \\ &= \left[\sum_{r=-\infty}^{\infty} \underbrace{\left(\sum_{m=0}^{N-1} x_1[m] x_2[(n-rN)-m]\right)}_{\text{Linear convolution } x_3[n-rN]}\right] p_N[n] \\ &= \left(\sum_{r=-\infty}^{\infty} x_3[n-rN]\right) p_N[n]. \end{aligned}$$

This proves $(4.6.1)$ which means that, in general, the circular convolution is an aliased version of the linear convolution.

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

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

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

Solution: Since $x_3[n]$ is an $N = (N_1 + N_2 - 1)$-point sequence, there is no overlap between shifted replicas of $x_3[n]$ in the aliasing sum in $(4.6.1)$. Hence $(4.6.2)$ follows.

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

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

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

Solution: Using the definition of the error sequence $e[n]$ and $(4.6.1)$ we have

$$\begin{aligned} e_N[n] &= x_4[n] - x_3[n] = \left(\sum_{r=-\infty}^{\infty} x_3[n-rN]\right) - \underbrace{x_3[n]}_{r=0 \text{ term}}, \quad 0 \le n \le N-1 \\ &= \sum_{r\neq 0} x_3[n-rN], \quad 0 \le n \le N-1. \qquad (4.6.4) \end{aligned}$$

Since $N \ge \max(N_1, N_2)$, only the first left $(r=-1)$ and the first right $(r=1)$ aliases (or images) remain in $(4.6.4)$ over the $0 \le n \le (N-1)$ interval. Hence

$$e[n] = x_3[n+N] + x_3[n-N], \quad 0 \le n \le N-1. \qquad (4.6.5)$$

Finally, since $x_1[n]$ and $x_2[n]$ are causal sequences, $x_3[n]$ is also causal. Then the second term in $(4.6.5)$ is zero, proving $(4.6.3)$.

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

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

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 $x_4[n]$, $(M-1) \le n \le (N-1)$.

Solution: Without loss of generality, let $N_1 = \max(N_1, N_2) = N$ which means that $M = N_2$. Now the linear convolution is $L = (N_1 + N_2 - 1)$-point long. Then from $(4.6.3)$, the error sequence is zero after $n + N \ge L$ or $n + N_1 \ge N_1 + N_2 - 1$ or $n \ge N_2 - 1 = M - 1$ proving $(4.6.6)$.

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: The convolutions and the corresponding error sequences are computed using the following \ml script.

```
clc; close all; clear;
N1 = 10; n1 = 0:N1-1; x1 = 0.8.^n1;             % Sequence x1[n]
N2 = 10; n2 = 0:N2-1; x2 = (-0.8).^n2;          % Sequence x2[n]
L = N1+N2-1; n3 = 0:L-1; x3 = conv(x1,x2);      % Linear conv x3[n]
N = 15; x4 = real(ifft(fft(x1,N).*fft(x2,N),N)); % Circular conv x4[n]
```

```
en = x4 - x3(1:N); en(1:7)                                % Error sequence e[n]
```

```
ans = 1×7
    -0.0000   -0.0281    0.0000   -0.0180   -0.0000    0.0000   -0.0000
```

```
x3N = x3(N+1:19), % Sequence x3[n+N]
```

```
x3N = 1×4
          0   -0.0281         0   -0.0180
```

(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: We will use matrix computational approach to do these hand calculations. For linear convolution, we will convert $x_2[n]$ into a banded Toeplitz matrix and $x_1[n]$ into a column vector, Then $x_3[n]$ is obtained as a column vector using

$$\begin{bmatrix} x_3[0] \\ x_3[1] \\ x_3[2] \\ x_3[3] \\ x_3[4] \\ x_3[5] \\ x_3[6] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 1 \\ 0 \\ -1 \\ 1 \end{bmatrix}$$

or $x_3[n] = \{1, -1, 0, 1, 0, -1, 1\}$.

For circular convolution, we will pad $x_1[n]$ with two zeros to make it a 5-point sequence and convert $x_2[n]$ into a 5×5 circulant matrix to obtain $x_4[n]$ as a column vector

$$\begin{bmatrix} x_4[0] \\ x_4[1] \\ x_4[2] \\ x_4[3] \\ x_4[4] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 & -1 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & 1 & 1 \\ 1 & 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

or $x_4[n] = \{0, 0, 0, 1, 0\}$. Hence

$$e[n] = \big(x_4[n] - x_3[n]\big)\, p_5[n] = \{\,-1, 1, 0, 0, 0\} = x_3[n+5]$$

which satisfies $(4.6.3)$. Also note that the first $M - 1 = 3 - 1 = 2$ samples in the circular convolution are in error which verifies $(4.6.6)$.

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: Let $x_3[n]$ be the linear convolution and let $x_4[n]$ be the circular convolution between $x_1[n]$ and $x_2[n]$, respectively. Then the circular convolution aliasing formula is given by (4.6.1) and reproduced here

$$x_4[n] = \left(\sum_{r=-\infty}^{\infty} x_3[n - rN]\right) p_N[n] \qquad (4.8.1)$$

Now for $N \geq \max(N_1, N_2)$, at most two shifted replicas or images of $x_3[n]$ overlap within the $0 \leq n \leq (N-1)$ interval, one from the left of and the other from the right of $x_3[n]$. These replicas are given by $x_3[n-N]$ (the left one) and $x_3[n+N]$ (the right one). Since $x_1[n]$ and $x_2[n]$ are causal sequences (because they are given as N_1- and N_2-point sequences), the $x[n-N]$ replica contributes only zeros to the aliasing sum in $(4.8.1)$. Considering this fact, we obtain

$$x_4[n] = x_3[n] + x_3[n+N], \quad 0 \leq n \leq N-1 \text{ and } N \geq \max(N_1, N_2). \qquad (4.8.2)$$

This means that we must linearly shift the last $N \leq n \leq L-1$ samples of $x_3[n]$ to align with the first N samples of $x_3[n]$ and then add the two halves, where $L = N_1 + N_2 - 1$. These steps are given in the following MATLAB function.

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); N = max(N1,N2); L = N1+N2-1;
y1 = conv(x(:).',h(:).'); % y1 is now a row sequence
y1 = [y1,zeros(1,2*N-L)]; % Pad with zeros to make 2N-point seq
y = reshape(y1,[N,2]);    % Shift last N samples to origin
y = sum(y,2); y = y.';    % Add two halves to create aliasing
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;
xn = 1:4; hn = [1,-1,1,-1];
y1 = lin2circonv(xn,hn)
```

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

```
y2 = (circonv(xn',hn'))', % Check
```

```
y2 = 1×4
    -2      2     -2      2
```

Create your MATLAB functions below.

```
function [x] = myifft(X)
% Computes IDFT using FFT function only
% Enter your code below

N = length(X);
x = fft(X')'/N; % Uses "'" for conjugate operation twice
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

N1 = length(x); N2 = length(h); N = max(N1,N2); L = N1+N2-1;
y1 = conv(x(:).',h(:).'); % y1 is now a row sequence
y1 = [y1,zeros(1,2*N-L)]; % Pad with zeros to make 2N-point seq
y = reshape(y1,[N,2]);    % Shift last N samples to origin
y = sum(y,2); y = y.';    % Add two halves to create aliasing
end
```