**Name: Lê Văn Sỹ**

**Id: 20119156**

## Section:

Laboratory Exercise 3

**DISCRETE-TIME SIGNALS: FREQUENCY-DOMAIN REPRESENTATIONS**

## DISCRETE-TIME FOURIER TRANSFORM

**Project 3.1 DTFT Computation**

A copy of Program P3\_1 is given below:

% Program P3\_1

% Evaluation of the DTFT clf;

% Compute the frequency samples of the DTFT w = -4\*pi:8\*pi/511:4\*pi;

num = [2 1];den = [1 -0.6];

h = freqz(num, den, w);

% Plot the DTFT subplot(2,1,1) plot(w/pi,real(h));grid

title('Real part of H(e^{j\omega})') xlabel('\omega /\pi'); ylabel('Amplitude');

subplot(2,1,2) plot(w/pi,imag(h));grid

title('Imaginary part of H(e^{j\omega})') xlabel('\omega /\pi'); ylabel('Amplitude');

pause subplot(2,1,1)

plot(w/pi,abs(h));grid

title('Magnitude Spectrum |H(e^{j\omega})|') xlabel('\omega /\pi');

ylabel('Amplitude'); subplot(2,1,2) plot(w/pi,angle(h));grid

title('Phase Spectrum arg[H(e^{j\omega})]') xlabel('\omega /\pi');

ylabel('Phase in radians');

## Answers:

**Q3.1** The expression of the DTFT being evaluated in Program P3\_1 is -

*H* *e j*  

# 2  *z*1

1 0.6*z*1

# The function of the pause command is - to pause execution of a Matlab program. Without arguments, pause waits for the user to type any key. With an argument, pause pauses for a number of seconds specified by the argument.

**Q3.2** The plots generated by running Program P3\_1 are shown below:





The DTFT is a \_periodic\_ function of .

Its period is - 2

The types of symmetries exhibited by the four plots are as follows:

# The real part is 2 periodic and EVEN SYMMETRIC.

* The imaginary part is 2 periodic and ODD SYMMETRIC.

# The magnitude is 2 periodic and EVEN SYMMETRIC.

* The phase is 2 periodic and ODD SYMMETRIC.

**Q3.3** The required modifications to Program P3\_1 to evaluate the given DTFT of Q3.3 are given below:

% Program P3\_1B

% Evaluation of the DTFT clf;

% Compute the frequency samples of the DTFT

% because 0 \leq w \leq pi is the default for "freqz",

% the vector "w" is now an output of freqz instead of an input. N = 512;

num = [0.7 -0.5 0.3 1];

den = [1 0.3 -0.5 0.7];

[h,w] = freqz(num, den, N);

% Plot the DTFT subplot(2,1,1) plot(w/pi,real(h));grid

title('Real part of H(e^{j\omega})') xlabel('\omega /\pi'); ylabel('Amplitude');

subplot(2,1,2) plot(w/pi,imag(h));grid

title('Imaginary part of H(e^{j\omega})') xlabel('\omega /\pi'); ylabel('Amplitude');

pause subplot(2,1,1)

plot(w/pi,abs(h));grid

title('Magnitude Spectrum |H(e^{j\omega})|') xlabel('\omega /\pi');

ylabel('Amplitude'); subplot(2,1,2) plot(w/pi,angle(h));grid

title('Phase Spectrum arg[H(e^{j\omega})]') xlabel('\omega /\pi');

ylabel('Phase in radians');

The plots generated by running the modified Program P3\_1 are shown below:





The DTFT is a \_periodic function of .

Its period is - 2

The jump in the phase spectrum is caused by - a branch cut in the arctan function used by

# angle in computing the phase. “angle” returns the principal branch of arctan.

The phase spectrum evaluated with the jump removed by the command unwrap is as given below:



**Q3.4** The required modifications to Program P3\_1 to evaluate the given DTFT of Q3.4 are given below:

% Program P3\_1D

% Evaluation of the DTFT clf;

% Compute the frequency samples of the DTFT w = -4\*pi:8\*pi/511:4\*pi;

num = [1 3 5 7 9 11 13 15 17];

den = 1;

h = freqz(num, den, w);

% Plot the DTFT subplot(2,1,1) plot(w/pi,real(h));grid

title('Real part of H(e^{j\omega})') xlabel('\omega /\pi'); ylabel('Amplitude');

subplot(2,1,2) plot(w/pi,imag(h));grid

title('Imaginary part of H(e^{j\omega})') xlabel('\omega /\pi'); ylabel('Amplitude');

pause subplot(2,1,1)

plot(w/pi,abs(h));grid

title('Magnitude Spectrum |H(e^{j\omega})|') xlabel('\omega /\pi');

ylabel('Amplitude'); subplot(2,1,2) plot(w/pi,angle(h));grid

title('Phase Spectrum arg[H(e^{j\omega})]') xlabel('\omega /\pi');

ylabel('Phase in radians');

The plots generated by running the modified Program P3\_1 are shown below:





The DTFT is a \_periodic\_ function of .

Its period is - 2

The jump in the phase spectrum is caused by - “angle” returns the principal value of the arc tangent.

**Q3.5** The required modifications to Program P3\_1 to plot the phase in degrees are indicated below: Only the last paragraph of the code needs to be changed to:

% plot phase in degrees subplot(2,1,2) plot(w/pi,180\*angle(h)/pi);grid

title('Phase Spectrum arg[H(e^{j\omega})]') xlabel('\omega /\pi');

ylabel('Phase in degrees');

## Project 3.2 DTFT Properties Answers:

**Q3.6** The modified Program P3\_2 created by adding appropriate comment statements, and adding program statements for labeling the two axes of each plot being generated by the program is given below:

% Program P3\_2B

% Time-Shifting Properties of DTFT clf;

w = -pi:2\*pi/255:pi; % freqency vector for evaluating DTFT D = 10; % Amount of time shift in samples

num = [1 2 3 4 5 6 7 8 9];

% h1 is the DTFT of original sequence

% h2 is the DTFT of the time shifted sequence h1 = freqz(num, 1, w);

h2 = freqz([zeros(1,D) num], 1, w); subplot(2,2,1)

% plot the DTFT magnitude of the original sequence plot(w/pi,abs(h1));grid

title('Magnitude Spectrum of Original Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Amplitude');

% plot the DTFT magnitude of the shifted sequence subplot(2,2,2)

plot(w/pi,abs(h2));grid

title('Magnitude Spectrum of Time-Shifted Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Amplitude');

% plot the DTFT phase of the original sequence subplot(2,2,3)

plot(w/pi,angle(h1));grid

title('Phase Spectrum of Original Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Phase in radians');

% plot the DTFT phase of the shifted sequence subplot(2,2,4)

plot(w/pi,angle(h2));grid

title('Phase Spectrum of Time-Shifted Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Phase in radians');

The parameter controlling the amount of time-shift is - D

**Q3.7** The plots generated by running the modified program are given below:



# From these plots we make the following observations: the slope of the phase function steeper

**Q3.8** Program P3\_2 was run for the following value of the time-shift – D=5.

The plots generated by running the modified program are given below:



# From these plots we make the following observations: As D=5, the slope of spectrum less stepper than D=10

**Q3.9** Program P3\_2 was run for the following values of the time-shift and for the following values of length for the sequence –

# Length 8, time shift D=3.

* + 1. Length 12, time shift D=9.

The plots generated by running the modified program are given below:





# From these plots we make the following observations: Increasing the length makes the magnitude spectrum more narrow. It also makes the phase steeper

**Q3.10** The modified Program P3\_3 created by adding appropriate comment statements, and adding program statements for labeling the two axes of each plot being generated by the program is given below:

% Program P3\_3B

% Frequency-Shifting Properties of DTFT clf;

w = -pi:2\*pi/255:pi; % freqency vector for evaluating DTFT wo = 0.4\*pi; % Amount of frequency shift in radians

% h1 is the DTFT of the original sequence

% h2 is the DTFT of the frequency shifted sequence num1 = [1 3 5 7 9 11 13 15 17];

L = length(num1);

h1 = freqz(num1, 1, w); n = 0:L-1;

num2 = exp(wo\*i\*n).\*num1; h2 = freqz(num2, 1, w);

% plot the DTFT magnitude of the original sequence subplot(2,2,1)

plot(w/pi,abs(h1));grid

title('Magnitude Spectrum of Original Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Amplitude');

% plot the DTFT magnitude of the freq shifted sequence subplot(2,2,2)

plot(w/pi,abs(h2));grid

title('Magnitude Spectrum of Frequency-Shifted Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Amplitude');

% plot the DTFT phase of the original sequence subplot(2,2,3)

plot(w/pi,angle(h1));grid

title('Phase Spectrum of Original Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Phase in radians');

% plot the DTFT phase of the shifted sequence subplot(2,2,4)

plot(w/pi,angle(h2));grid

title('Phase Spectrum of Frequency-Shifted Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Phase in radians');

The parameter controlling the amount of frequency-shift is - wo.

**Q3.11** The plots generated by running the modified program are given below:



# From these plots we make the following observations: Both the magnitude and the phase spectra are shifted right by wo,

**Q3.12** Program P3\_3 was run for the following value of the frequency-shift – wo = -0.5.

The plots generated by running the modified program are given below:



# From these plots we make the following observations: In this case, the magnitude and phase spectra are shifted left by /2 rad.

**Q3.13** Program P3\_3 was run for the following values of the frequency-shift and for the following values of length for the sequence –

# Length 3, frequency shift wo = -.

1. Length 9, frequency shift wo = 0.3.

The plots generated by running the modified program are given below:





# From these plots we make the following observations: The original sequences have a low pass characteristic.

**Q3.14** The modified Program P3\_4 created by adding appropriate comment statements, and adding program statements for labeling the two axes of each plot being generated by the program is given below:

% Program P3\_4B

% Convolution Property of DTFT clf;

w = -pi:2\*pi/255:pi; % freqency vector for evaluating DTFT x1 = [1 3 5 7 9 11 13 15 17]; % first sequence

x2 = [1 -2 3 -2 1]; % second sequence

y = conv(x1,x2); % time domain convolution of x1 and x2 h1 = freqz(x1, 1, w); % DTFT of sequence x1

h2 = freqz(x2, 1, w); % DTFT of sequence x2

% hp is the pointwise product of the two DTFT's hp = h1.\*h2;

% h3 is the DTFT of the time domain convolution;

% it should be the same as hp h3 = freqz(y,1,w);

% plot the magnitude of the product of the two original spectra subplot(2,2,1)

plot(w/pi,abs(hp));grid

title('Product of Magnitude Spectra','FontSize',8) xlabel('\omega /\pi');

ylabel('Amplitude');

% plot the magnitude spectrum of the time domain convolution subplot(2,2,2)

plot(w/pi,abs(h3));grid

title('Magnitude Spectrum of Convolved Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Amplitude');

% plot the phase of the product of the two original spectra subplot(2,2,3)

plot(w/pi,angle(hp));grid

title('Sum of Phase Spectra','FontSize',8) xlabel('\omega /\pi');

ylabel('Phase in radians');

% plot the phase spectrum of the time domain convolution subplot(2,2,4)

plot(w/pi,angle(h3));grid

title('Phase Spectrum of Convolved Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Phase in radians');

**Q3.15** The plots generated by running the modified program are given below:



# From these plots we make the following observations: The phase and the magnitude of the DTFT is the same

**Q3.16** Program P3\_4 was run for the following two different sets of sequences of varying lengths –

# 1. Length of x1 = 8

# Length of x2 = 5

2. Length of x1 = 8

Length of x2 = 16

The plots generated by running the modified program are given below:





From these plots we make the following observations: The convolution of the DTFT is the same in both cases.

**Q3.17** The modified Program P3\_5 created by adding appropriate comment statements, and adding program statements for labeling the two axes of each plot being generated by the program is given below:

% Program P3\_5B

% Modulation Property of DTFT clf;

w = -pi:2\*pi/255:pi; % freqency vector for evaluating DTFT x1 = [1 3 5 7 9 11 13 15 17]; % first sequence

x2 = [1 -1 1 -1 1 -1 1 -1 1]; % second sequence

% y is the time domain pointwise product of x1 and x2 y = x1.\*x2;

h1 = freqz(x1, 1, w); % DTFT of sequence x1 h2 = freqz(x2, 1, w); % DTFT of sequence x2 h3 = freqz(y,1,w); % DTFT of sequence y

% plot the magnitude spectrum of x1 subplot(3,1,1) plot(w/pi,abs(h1));grid

title('Magnitude Spectrum of First Sequence') xlabel('\omega /\pi');

ylabel('Amplitude');

% plot the magnitude spectrum of x2 subplot(3,1,2) plot(w/pi,abs(h2));grid

title('Magnitude Spectrum of Second Sequence') xlabel('\omega /\pi');

ylabel('Amplitude');

% plot the magnitude spectrum of y

% it should be 1/2pi times the convolution of the DTFT's

% of the two original sequences. subplot(3,1,3) plot(w/pi,abs(h3));grid

title('Magnitude Spectrum of Product Sequence') xlabel('\omega /\pi');

ylabel('Amplitude');

**Q3.18** The plots generated by running the modified program are given below:



# From these plots we make the following observations: The DTFT of the product sequence y is 1/2 times the convolution of the DTFT’s of the two sequences x1 and x2, as expected.

**Q3.19** Program P3\_5 was run for the following two different sets of sequences of varying lengths –

# 1. Length of x1 = 8

Length of x2 = 8

# 2. Length of x1 = 16

# Length of x2 = 16





From these plots we make the following observations: In the first example, both x1 and x2 are low pass sequences.

**Q3.20** The modified Program P3\_6 created by adding appropriate comment statements, and adding program statements for labeling the two axes of each plot being generated by the program is given below:

% Program P3\_6B

% Time Reversal Property of DTFT clf;

w = -pi:2\*pi/255:pi; % freqency vector for evaluating DTFT

% original ramp sequence

% note: num is nonzero for 0 <= n <= 3. num = [1 2 3 4];

L = length(num)-1;

h1 = freqz(num, 1, w); % DTFT of original ramp sequence

% h2 contains the sample values of h1 in reverse order, but

% it is NOT the time reversed version of h1. The time

% reversed version must be nonzero for -3 <= n <= 0. However,

% h2 is nonzero for 0 <= n <= 3. So, to get the time reversed

% version of h1, we still need to time SHIFT h2 to the left.

% This is accomplished in the frequency domain using the time

% shift property of the DTFT. Thus, h3, which IS the time

% reversed version of h1, is obtained by multiplying h2 times

% a linear phase term to accomplish the required time shift. h2 = freqz(fliplr(num), 1, w);

h3 = exp(w\*L\*i).\*h2;

% plot the magnitude spectrum of the original ramp sequence subplot(2,2,1)

plot(w/pi,abs(h1));grid

title('Magnitude Spectrum of Original Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Amplitude');

% plot the magnitude spectrum of the time reversed ramp sequence subplot(2,2,2)

plot(w/pi,abs(h3));grid

title('Magnitude Spectrum of Time-Reversed Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Amplitude');

% plot the phase spectrum of the original ramp sequence subplot(2,2,3)

plot(w/pi,angle(h1));grid

title('Phase Spectrum of Original Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Phase in radians');

% plot the phase spectrum of the time reversed ramp sequence subplot(2,2,4)

plot(w/pi,angle(h3));grid

title('Phase Spectrum of Time-Reversed Sequence','FontSize',8) xlabel('\omega /\pi');

ylabel('Phase in radians');

# The program implements the time-reversal operation as follows – The original ramp sequence is nonzero for 0 ≤ *n* ≤ 3. A new sequence is formed by using fliplr; this new sequence contains the samples of the original ramp sequence in time reversed order. However, the new sequence is still nonzero for 0 ≤ *n* ≤ 3, whereas the time reversed ramp sequence must be nonzero for -3 ≤ *n* ≤ 0. This required left shift in time is accomplished in the frequency domain using the time shift property of the DTFT as follows. First, freqz is called to set h2 equal to the DTFT of the new sequence obtained from calling fliplr on the original ramp sequence. Finally, h3 is set equal to the DTFT of the time reversed ramp by multiplying h2 times a linear phase term to implement the required left shift in the time domain.

**Q3.21** The plots generated by running the modified program are given below:



# From these plots we make the following observations: Both the original and time reversed ramp sequences are real-valued.

**Q3.22** Program P3\_6 was run for the following two different sets of sequences of varying lengths –

# Length of num = 8

1. Length of num = 16

The plots generated by running the modified program are given below:





# From these plots we make the following observations: the phase spectrum of the time reversed sequence is a frequency reversed version of the phase spectrum of the original sequence.

## DISCRETE FOURIER TRANSFORM Project 3.3 DFT and IDFT Computations Answers:

**Q3.23** The MATLAB program to compute and plot the L-point DFT X[k] of a length-N sequence

x[n] with L  N and then to compute and plot the IDFT of X[k] is given below:

% Program P3\_3DFT

% Compute and plot the L-point DFT of an N-point signal, L >= N. clf;

%Initialize

N=200; % length of signal L=256; % length of DFT

nn = [0:N-1];

kk = [0:L-1];

% the signal x

xR = [0.1\*(1:100) zeros(1,N-100)]; % real part xI = [zeros(1,N)]; % imag part

x = xR + i\*xI;

% DFT

XF = fft(x,L);

% plot xR and xI subplot(3,2,1);grid; plot(nn,xR);grid; title('Re\{x[n]\}'); xlabel('Time index n'); ylabel('Amplitude'); subplot(3,2,2); plot(nn,xI);grid; title('Im\{x[n]\}'); xlabel('Time index n'); ylabel('Amplitude');

% plot real and imag parts of DFT subplot(3,2,3);

plot(kk,real(XF));grid;

title('Re\{X[k]\}'); xlabel('Frequency index k'); ylabel('Amplitude'); subplot(3,2,4);

plot(kk,imag(XF));grid;

title('Im\{X[k]\}'); xlabel('Frequency index k'); ylabel('Amplitude');

% IDFT

xx = ifft(XF,L);

% plot real and imaginary parts of the IDFT subplot(3,2,5);

plot(kk,real(xx));grid;

title('Real part of IDFT\{X[k]\}'); xlabel('Time index n'); ylabel('Amplitude'); subplot(3,2,6);

plot(kk,imag(xx));grid;

title('Imag part of IDFT\{X[k]\}'); xlabel('Time index n'); ylabel('Amplitude');

The DFT and the IDFT pairs generated by running the program for sequences of different lengths N and for different values of the DFT length L are shown below:





# From these plots we make the following observations: With N=100 and L= 200, the real number in figure 2 is become linear

**Q3.24** The MATLAB program to compute the N-point DFT of two length-N real sequences using a single N-point DFT and compare the result by computing directly the two N-point DFTs is given below:

% Program Q3\_24

% Use a single N-point DFT to compute simultaneously the N-point

% DFT's of two real-valued N-point sequences. clf;

%Initialize

N=256; % length of signal nn = [0:N-1];

ntime = [-N/2:N/2-1];

g = (0.75).^abs(ntime); % signal g h = (-0.9).^ntime; % signal h

% DFT's of g and h GF = fft(g);

HF = fft(h);

x = g + i\*h; % the composite signal x

% DFT of composite signal XF = fft(x);

% DFT of g derived from composite DFT XF XFstar = conj(XF);

XFstarmod = [XFstar(1) fliplr(XFstar(2:N))]; GF2 = 0.5\*(XF + XFstarmod);

HF2 = -i\*0.5\*(XF - XFstarmod);

abs(max(GF-GF2))

abs(max(HF-HF2))

% plot real and imag parts of direct computation of GF figure(1);clf;

subplot(2,2,1);grid; plot(nn,real(GF));grid;

title('Two N-point DFT''s'); xlabel('Frequency index k'); ylabel('Re\{G[k]\}'); subplot(2,2,2);

plot(nn,imag(GF));grid;

title('Two N-point DFT''s'); xlabel('Frequency index k'); ylabel('Im\{G[k]\}');

% plot real and imag parts of composite computation of GF subplot(2,2,3);grid;

plot(nn,real(GF2));grid; title('Single N-point DFT'); xlabel('Frequency index k'); ylabel('Re\{G[k]\}'); subplot(2,2,4); plot(nn,imag(GF2));grid; title('Single N-point DFT'); xlabel('Frequency index k'); ylabel('Im\{G[k]\}');

% plot real and imag parts of direct computation of HF figure(2);clf;

subplot(2,2,1);grid; plot(nn,real(HF));grid;

title('Two N-point DFT''s'); xlabel('Freq index k'); ylabel('Re\{H[k]\}'); subplot(2,2,2);

plot(nn,imag(HF));grid;

title('Two N-point DFT''s'); xlabel('Freq index k'); ylabel('Im\{H[k]\}');

% plot real and imag parts of composite computation of HF subplot(2,2,3);grid;

plot(nn,real(HF2));grid; title('Single N-point DFT'); xlabel('Freq index k'); ylabel('Re\{H[k]\}'); subplot(2,2,4); plot(nn,imag(HF2));grid; title('Single N-point DFT'); xlabel('Freq index k'); ylabel('Im\{H[k]\}');

The DFTs generated by running the program for sequences of different lengths N are shown below:





# 

# 

From these plots we make the following observations: With N=128, the frequency of this DFT is sparser than N = 25

**Q3.25** The MATLAB program to compute the 2N-point DFT of a length-2N real sequence using a single N-point DFT and compare the result by computing directly the 2N-point DFT is shown below:

% Program Q3\_25A

% Use a single N-point complex-valued DFT to compute the 2N-point

% DFT of a 2N-point real-valued sequence.

%

clf;

%Initialize constants

N = 128; % length of the complex-valued DFT TwoN = 2\*N; % length of the real-valued sequence W2N = exp(-i\*pi/N);

k = [0:TwoN-1];

% create 2N-point signal v[n] v = (-0.7.^k);

% create two N-point signals

g = downsample(v,2); % g[n] = v[2n] h = downsample(v,2,1); % h[n] = v[2n+1]

% N-point complex-valued composite signal x[n] x = g + i\*h;

% Use one N-point complex DFT to compute simultaneously

% G[k] and H[k] as in Q3.24.

XF = fft(x); % N-point complex DFT of x[n]

% DFT's G[k] and H[k] derived from composite DFT X[k] XFstar = conj(XF);

XFstarmod = [XFstar(1) fliplr(XFstar(2:N))]; GF = 0.5\*(XF + XFstarmod);

HF = -i\*0.5\*(XF - XFstarmod);

% 2N-point DFT V[k]

VF = [GF GF] + (W2N.^k).\*[HF HF];

% For Comparison, compute directly the 2N-point DFT V[k] VF2 = fft(v);

% Print Sanity Check abs(max(VF-VF2))

% plot real and imag parts of V[k] computed by complex N-point DFT subplot(2,2,1);

plot(k,real(VF));grid; title('Complex N-point DFT'); xlabel('Frequency index k'); ylabel('Re\{V[k]\}'); subplot(2,2,2);

plot(k,imag(VF));grid; title('Complex N-point DFT'); xlabel('Frequency index k'); ylabel('Im\{V[k]\}');

% plot real and imag parts of V[k] computed by 2N-point DFT subplot(2,2,3);

plot(k,real(VF2));grid; title('Real 2N-point DFT'); xlabel('Frequency index k'); ylabel('Re\{V[k]\}'); subplot(2,2,4);

plot(k,imag(VF2));grid; title('Real 2N-point DFT'); xlabel('Frequency index k'); ylabel('Im\{V[k]\}');

The DFTs generated by running the program for sequences of different lengths 2N are shown below:



From these plots we make the following observations:

## Project 3.4 DFT Properties Answers:

**Q3.26** The purpose of the command **rem** in the function circshift is – rem(x,y) is the remainder after x is divided by y.

# **Q3.27** The function **circshift** operates as follows: The input sequence x is circularly shifted left by M positions. If M > 0, then circshift removes the leftmost M elements from the vector x and appends them on the right side of the remaining elements to obtain the circularly shifted sequence. If If M < 0, then circshift first complements M by the length of x, i.e., the rightmost length(x)-M samples are removed from x and appended on the right of the remaining M samples to obtain the circularly shifted sequence.

**Q3.28** The purpose of the operator ~= in the function circonv is – This is the binary relational NOT EQUAL operator. A ~= B returns the value 1 if A and B are unequal and the value 0 if A and B are equal.

# **Q3.29** The function **circonv** operates as follows: The input is two vectors x1 and x2 of equal length L. To understand how circonv works, it is useful to think in terms of the periodic extension of x2. Let x2p be the infinite-length periodic extension of x2. Conceptually, the routine time reverses x2p and sets x2tr equal to elements 1 through L of the time reversed version of x2p. Elements 1 through L of the output vector y are then obtained by taking the inner product between x1 and a length L vector sh obtained by circularly shifting right the time reversed vector x2tr. For the output sample y[n], 1

≤ n ≤ L, the amount of the right circular shift is n-1 positions.

**Q3.30** The modified Program P3\_7 created by adding appropriate comment statements, and adding program statements for labeling each plot being generated by the program is given below:

% Program P3\_7B

% Illustration of Circular Shift of a Sequence clf;

% initialize shift amount M M = 6;

% initialize sequence a to be shifted a = [0 1 2 3 4 5 6 7 8 9];

b = circshift(a,M); % perform the circular shift L = length(a)-1;

% plot the original sequence a and the circularly shifted sequence b n = 0:L;

subplot(2,1,1); stem(n,a);axis([0,L,min(a),max(a)]); title('Original Sequence'); xlabel('time index n'); ylabel('a[n]');

subplot(2,1,2); stem(n,b);axis([0,L,min(a),max(a)]);

title(['Sequence Obtained by Circularly Shifting by ',num2str(M),' Samples']);

xlabel('time index n'); ylabel('b[n]');

The parameter determining the amount of time-shifting is - M

If the amount of time-shift is greater than the sequence length then – The circular shift actually implemented is rem(M,length(a)) positions left, which is equivalent to circularly shifting by M positions (more than once around) and also to shifting left by M the periodic extension of the sequence.

**Q3.31** The plots generated by running the modified program are given below:



From these plots we make the following observations: Here, the length of the sequence is 10 samples and we have M=12.

**Q3.32** The modified Program P3\_8 created by adding appropriate comment statements, and adding program statements for labeling each plot being generated by the program is given below:

% Program P3\_8B

% Circular Time-Shifting Property of DFT clf;

x = [0 2 4 6 8 10 12 14 16]; % original sequence x N = length(x)-1; n = 0:N; % time index vector

% set y equal to the circular shift left of x y = circshift(x,5);

XF = fft(x); % DFT of x

YF = fft(y); % DFT of y subplot(2,2,1);

% plot the spectral magnitudes of the original and shifted sequences stem(n,abs(XF));grid;

title('Magnitude of DFT of Original Sequence'); xlabel('Frequency index k');

ylabel('|X[k]|');

subplot(2,2,2);

stem(n,abs(YF));grid;

title('Magnitude of DFT of Circularly Shifted Sequence'); xlabel('Frequency index k');

ylabel('|Y[k]|');

% plot the spectral phases of the original and shifted sequences subplot(2,2,3);

stem(n,angle(XF));grid;

title('Phase of DFT of Original Sequence'); xlabel('Frequency index k'); ylabel('arg(X[k])');

subplot(2,2,4);

stem(n,angle(YF));grid;

title('Phase of DFT of Circularly Shifted Sequence'); xlabel('Frequency index k');

ylabel('arg(Y[k])');

The amount of time-shift is - hard coded in this program at 5 samples to the left.

**Q3.33** The plots generated by running the modified program are given below:



From these plots we make the following observations: The length of the sequence is N=8 and the time shift is an advance by five samples to the left

**Q3.34** The plots generated by running the modified program for the following two different amounts of time-shifts, with the amount of shift indicated, are shown below:

# M=3



# M=-3



From these plots we make the following observations: In all cases, the spectral magnitude is not affected by the shift.

**Q3.35** The plots generated by running the modified program for the following two different sequences of different lengths, with the lengths indicated, are shown below:

# Length = 24

# 

# Length = 15

# 

From these plots we make the following observations:

**Q3.36** A copy of Program P3\_9 is given below along with the plots generated by running this program:

% Program P3\_9

% Circular Convolution Property of DFT

g1 = [1 2 3 4 5 6]; g2 = [1 -2 3 3 -2 1];

ycir = cconv(g1,g2);

disp('Result of circular convolution = ');disp(ycir) G1 = fft(g1); G2 = fft(g2);

yc = real(ifft(G1.\*G2));

disp('Result of IDFT of the DFT products = ');disp(yc)

Result of circular convolution = 1 0 2 7 10 14 11 28 12 -7 6

Result of IDFT of the DFT products = 12 28 14 0 16 14

From these plots we make the following observations: The DFT of a circular convolution is the pointwise products of the DFT’s.

**Q3.37** Program P3\_9 was run again for the following two different sets of equal-length sequences: The plots generated are shown below:

Result of circular convolution = 1.0000 0.0000 2.0000 7.0000 10.0000 14.0000 18.0000 14.0000 33.0000 14.0000 -8.0000 7.0000 -0.0000

Result of IDFT of the DFT products = 15.0000 33.0000 16.0000 -1.0000 17.0000 14.0000 18.0000

From these plots we make the following observations: The circular convolution property of the DFT seems to hold.

**Q3.38** A copy of Program P3\_10 is given below along with the plots generated by running this program:

% Program P3\_10

% Linear Convolution via Circular Convolution g1 = [1 2 3 4 5];g2 = [2 2 0 1 1];

g1e = [g1 zeros(1,length(g2)-1)]; g2e = [g2 zeros(1,length(g1)-1)]; ylin = cconv(g1e,g2e);

disp('Linear convolution via circular convolution = ');disp(ylin); y = conv(g1, g2);

disp('Direct linear convolution = ');disp(y)

Linear convolution via circular convolution = 2 6 10 15 21 15 7 9 5

Direct linear convolution = 2 6 10 15 21 15 7 9 5

From these plots we make the following observations: zero padding to the appropriate length does indeed make it possible to implement linear convolution using circular convolution.

**Q3.39** Program P3\_10 was run again for the following two different sets of sequences of unequal lengths:

g1 = [3 1 4 1 0 0 0 3];g2 = [2 2 0 1 1];

Linear convolution via circular convolution =

Columns 1 through 13

6.0000 8.0000 10.0000 13.0000 6.0000 5.0000 5.0000 7.0000 6.0000 0.0000 3.0000 3.0000 0.0000

Columns 14 through 23

0.0000 0.0000 0 -0.0000 0.0000 0.0000 -0.0000 0 0 0.0000

Direct linear convolution =

6 8 10 13 6 5 5 7 6 0 3 3

g1 = [9 6 1 5 0 1];g2 = [ 0 1 2 3 5];

Linear convolution via circular convolution =

Columns 1 through 13

-0.0000 9.0000 24.0000 40.0000 70.0000 43.0000 21.0000 27.0000 3.0000 5.0000 0.0000 0.0000 0.0000

Columns 14 through 19

0.0000 0 0.0000 0.0000 0.0000 0.0000

Direct linear convolution =

0 9 24 40 70 43 21 27 3 5

From these plots we make the following observations: You can implement the linear convolution of two sequences by zero padding them to the sum of their lengths less one and then invoking circular convolution on the zero padded sequences.

**Q3.40** The MATLAB program to develop the linear convolution of two sequences via the DFT of each is given below:

% Program Q3\_40

% Linear Convolution via Circular Convolution g1 = [1 2 3 4 5];

g2 = [2 2 0 1 1];

g1e = [g1 zeros(1,length(g2)-1)]; g2e = [g2 zeros(1,length(g1)-1)]; G1EF = fft(g1e);

G2EF = fft(g2e);

ylin = real(ifft(G1EF.\*G2EF));

disp('Linear convolution via DFT = ');disp(ylin);

The plots generated by running this program for the sequences of Q3.38 are shown below: Linear convolution via DFT =

2.0000 6.0000 10.0000 15.0000 21.0000 15.0000 7.0000 9.0000 5.0000

From these plots we make the following observations: The result is the same as before in Q3.38; in other words, it works as advertised

**Q3.41** A copy of Program P3\_11 is given below:

% Program P3\_11

% Relations between the DFTs of the Periodic Even

% and Odd Parts of a Real Sequence

x = [1 2 4 2 6 32 6 4 2 zeros(1,247)]; x1 = [x(1) x(256:-1:2)];

xe = 0.5 \*(x + x1);

XF = fft(x);

XEF = fft(xe); clf;

k = 0:255;

subplot(2,2,1); plot(k/128,real(XF)); grid; ylabel('Amplitude'); title('Re(DFT\{x[n]\})'); subplot(2,2,2); plot(k/128,imag(XF)); grid; ylabel('Amplitude'); title('Im(DFT\{x[n]\})'); subplot(2,2,3); plot(k/128,real(XEF)); grid;

xlabel('Time index n');ylabel('Amplitude'); title('Re(DFT\{x\_{e}[n]\})'); subplot(2,2,4);

plot(k/128,imag(XEF)); grid;

xlabel('Time index n');

ylabel('Amplitude');

title('Im(DFT\{x\_{e}[n]\})');

The relation between the sequence x1[n] and x[n] is – x1[n] is a periodically time reversed version of x[n].

**Q3.42** The plots generated by running Program P3\_11 are given below:



The imaginary part of XEF is equal to zero to within floating point precision. This result can be explained as follows: The real part of the transform of x[n] is the transform of the periodically even part of x[n]. Therefore, the DFT of the periodically even part of x[n] has a real part that is precisely the real part of X[k] and an imaginary part that is zero.

**Q3.43** The required modifications to Program P3\_11 to verify the relation between the DFT of the periodic odd part and the imaginary part of XEF are given below along with the plots generated by running this program:

% Program P3\_11B

% Relations between the DFTs of the Periodic Even

% and Odd Parts of a Real Sequence

x = [1 2 4 2 6 32 6 4 2 zeros(1,247)]; x1 = [x(1) x(256:-1:2)];

xo = 0.5 \*(x - x1);

XF = fft(x);

XOF = fft(xo); clf;

k = 0:255;

subplot(2,2,1); plot(k/128,real(XF)); grid; ylabel('Amplitude'); title('Re(DFT\{x[n]\})'); subplot(2,2,2); plot(k/128,imag(XF)); grid; ylabel('Amplitude'); title('Im(DFT\{x[n]\})'); subplot(2,2,3); plot(k/128,real(XOF)); grid;

xlabel('Time index n');ylabel('Amplitude'); title('Re(DFT\{x\_{o}[n]\})'); subplot(2,2,4);

plot(k/128,imag(XOF)); grid;

xlabel('Time index n');ylabel('Amplitude'); title('Im(DFT\{x\_{o}[n]\})');



From these plots we make the following observations: The DFT of the periodically odd part of x[n] is precisely the imaginary part of the DFT of x[n]. Therefore, the DFT of the periodically odd part of x[n] has a real part that is zero to within floating point precision and an imaginary part that is precisely the imaginary part of the DFT of x[n].

**Q3.44** A copy of Program P3\_12 is given below:

% Program P3\_12

% Parseval's Relation

x = [(1:128) (128:-1:1)];

XF = fft(x); a = sum(x.\*x)

b = round(sum(abs(XF).^2)/256)

The values for a and b we get by running this program are –

a =1414528

b =1414528

**Q3.45** The required modifications to Program P3\_11 are given below:

% Program P3\_12B

% Parseval's Relation

x = [(1:128) (128:-1:1)];

XF = fft(x); a = sum(x.\*x)

b = round(sum(XF.\*conj(XF))/256)

## z-TRANSFORM

**Project 3.5 Analysis of z-Transforms Answers:**

**Q3.46** The frequency response of the z-transform obtained using Program P3\_1 is plotted below:

**Q3.47** The MATLAB program to compute and display the poles and zeros, to compute and display the factored form, and to generate the pole-zero plot of a rational z-transform is given below:

% Program Q3\_47

% Given numerator and denominator coefficient vectors for G(z),

% - compute and display poles and zeros

% - compute and display factored form of G(z)

% - generate pole-zero plot

% NOTE: the lab book says to use tf2zp. For a rational function

% in z^-1, it's better to use tf2zpk. clf;

% initialize

num = [2 5 9 5 3];

den = [5 45 2 1 1];

% compute poles and zeros and display [z p k] = tf2zpk(num,den); disp('Zeros:');

disp(z); disp('Poles:'); disp(p);

input('Hit <return> to continue...');

% compute and display factored form of G(z) [sos k] = zp2sos(z,p,k)

input('Hit <return> to continue...');

% generate pole-zero plot zplane(z,p);

Using this program we obtain the following results on the z-transform G(z) of Q3.46:

# Zeros:

-1.0000 + 1.4142i

# -1.0000 - 1.4142i

-0.2500 + 0.6614i

# -0.2500 - 0.6614i

Poles:

# -8.9576

-0.2718

# 0.1147 + 0.2627i

0.1147 - 0.2627i

# sos =

|  |  |  |
| --- | --- | --- |
| 1.0000 | 2.0000 | 3.0000 1.0000 9.2293 2.4344 |
| 1.0000 | 0.5000 | 0.5000 1.0000 -0.2293 0.0822 |

k =

# 0.4000

1 2*z*1  3*z*2

# 1 0.5*z*1  0.5*z*2

*G*(*z*)  0.41 9.2293*z*1  2.4344*z*2 1 0.2293*z*1  0.0822*z*2

****

**Q3.48** From the pole-zero plot generated in Question Q3.47, the number of regions of convergence (ROC) of G(z) are - FOUR. note: the magnitude of the complex conjugate poles inside the unit circle is 0.2866.

All possible ROCs of this z-transform are sketched below:

*R*1 : | *z* | 

# 0.2718

(left-sided, not stable)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *R*2 : | 0.2718  | *z* | |  | 0.2866 | (two-sided, not stable) |
| *R*3 : | 0.2866  | *z* | |  | 8.9576 | (two-sided, stable) |
| *R*4 : | | *z* |  8.9576 |  |  | (right-sided, not stable) |

From the pole-zero plot it can be seen that the DTFT – You cannot tell if the DTFT exists from the pole zero plot alone. In order to know this, the region of convergence must be

# specified. The DTFT does exist for the sequence obtained by using the ROC above. This would be a stable system with a two-sided impulse response.

*R*3 shown

**Q3.49** The MATLAB program to compute and display the rational z-transform from its zeros, poles and gain constant is given below:

% Program Q3\_49

% Given the poles and zeros of G(z), compute and display the rational

% z-transform. clf;

% initialize

z = [0.3 2.5 -0.2+i\*0.4 -0.2-i\*0.4]';

p = [0.5 -0.75 0.6+i\*0.7 0.6-i\*0.7]';

k = 3.9;

% find numerator and denominator polynomial coefficients [num den] = zp2tf(z,p,k)

The rational form of a z-transform with the given poles, zeros, and gain is found to be –

# num =

3.9000 -9.3600 -0.6630 -1.0140 0.5850

# den =

1.0000 -0.9500 0.1750 0.6625 -0.3187

# 3.9  9.36*z*1  0.663*z*2 1.014*z*3  0.585*z*4

*G*(*z*)  1 0.95*z*1  0.175*z*2  0.6625*z*3  0.3187*z*4

## Project 3.6 Inverse z-Transform Answers:

**Q3.50** The MATLAB program to compute the first L samples of the inverse of a rational z-transform is given below:

% Program Q3\_50

% Given numerator and denominator coefficient vectors for G(z),

% find and plot the first L samples of the impulse response, where

% the parameter L is input by the user.

%

clf;

% initialize

num = [2 5 9 5 3];

den = [5 45 2 1 1];

% Query user for parameter L

L = input('Enter the number of samples L: ');

% find impulse response [g t] = impz(num,den,L);

%plot the impulse response stem(t,g);

title(['First ',num2str(L),' samples of impulse response']); xlabel('Time Index n');

ylabel('h[n]');

The plot of the first 50 samples of the inverse of G(z) of Q3.46 obtained using this program is sketched below:



**Q3.51** The MATLAB program to determine the partial-fraction expansion of a rational z-transform is given below:

% Program Q3\_51

% Given numerator and denominator coefficient vectors for G(z),

% find and plot the first L samples of the impulse response, where

% the parameter L is input by the user.

%

clf;

% initialize

num = [2 5 9 5 3];

den = [5 45 2 1 1];

% partial fraction expansion [r p k] = residuez(num,den)

The partial-fraction expansion of G(z) of Q3.46 obtained using this program is shown below:

r =

0.3109 + 0.0000i

-1.0254 - 0.3547i

-1.0254 + 0.3547i

-0.8601 + 0.0000i

p =

-8.9576 + 0.0000i

0.1147 + 0.2627i

0.1147 - 0.2627i

-0.2718 + 0.0000i

k =

# 3.00003

# G(z)= 0.3109/(1 + 8.9576) + ( -1.0254 - 0.3547j)/(1-(0.1147 + 0.2627j )+ (-1.0254 + 0.3547j)/(1-(0.1147 - 0.2627j)) -0.8601/(1+ -0.2718) +3

From the above partial-fraction expansion we arrive at the inverse z-transform g[n] as shown below:

**Date: 5 December 2023 Signature: Sy**