RYERSON UNIVERSITY

Faculty of Engineering, Architecture and Science

Department of Electrical and Computer Engineering

Course Number	ELE632
Course Title	Signals and Systems II
Semester/Year	Winter 2022

Instructor	Dr. Dimitri Androutsos

ASSIGNMENT No. 4

Assignment Title	Discrete-Time Fourier Transform

Submission Date	March 27, 2022
Due Date	March 27, 2022

Student Name	Reza Aablue
Student ID	500966944
Signature*	R.A.

**By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: www.ryerson.ca/senate/current/pol60.pdf.*

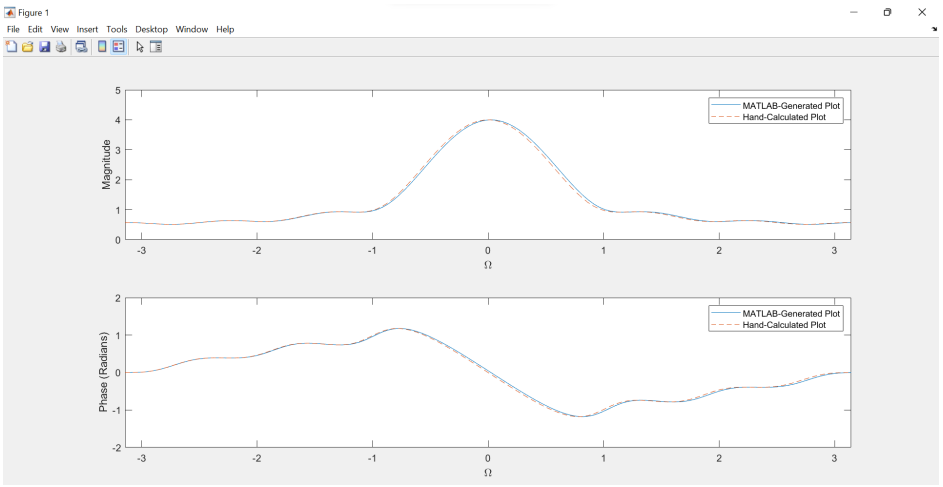
Problems A.1-A.3

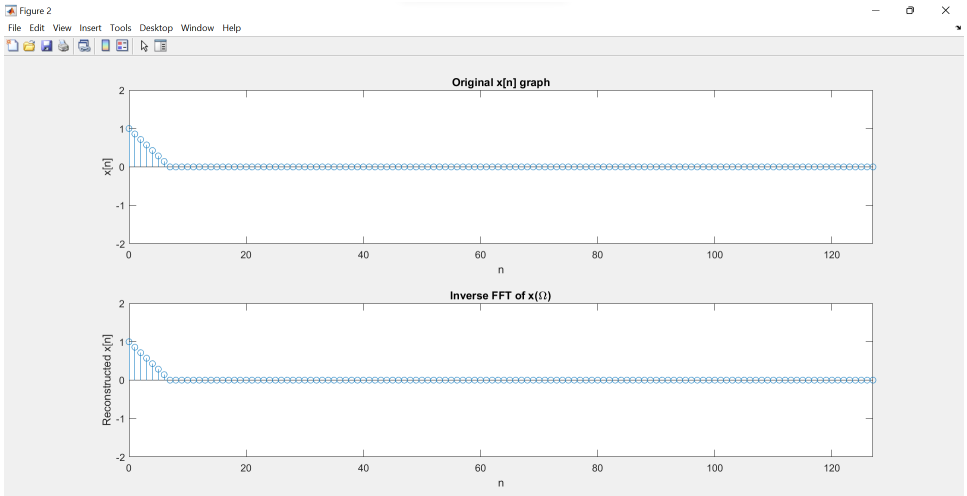
```
% Reza Aablue
% 500966944
% Section 05

% Problem A.1
xofn = zeros (1,128); % Zeros array of 128 pre-allocated points.
n = (0:127);
xofn(1:7) = 1-1/7.*n(1:7); % Equation of the line for n=0 up to n=7.

XofF = fft(xofn); % Fast FT of x[n].
XofF = fftshift(XofF);
Wo=linspace (-pi, pi, 128);
```

```
% Problem A.2
XofFmanual = @(a) (1 + (6/7).*exp(-1.*1i.*a) + (5/7).*exp(-2.*1i.*a) + (4/7).*exp(-3.*1i.*a)
+ (3/7).*exp(-4.*1i.*a) + (2/7).*exp(-5.*1i.*a) + (1/7).*exp(-6.*1i.*a));
```

```
figure (1);
subplot (2,1,1);
plot (Wo, abs (XofF));
hold on;
plot (Wo, abs (XofFmanual(Wo)),'--');
legend ('MATLAB-Generated Plot','Hand-Calculated Plot');
hold off;
axis ([-pi pi -5 5]);
xlabel ('\Omega'); ylabel ('Magnitude');

subplot (2,1,2);
plot (Wo, angle (XofF));
hold on;
plot (Wo, angle (XofFmanual(Wo)),'--');
legend ('MATLAB-Generated Plot','Hand-Calculated Plot');
hold off;
axis ([-pi pi -5 5]);
xlabel ('\Omega'); ylabel ('Phase (Radians)');
```

```
% Problem A.3
XofF=ifftshift(XofF);
z=ifft(XofF);

figure (2);
subplot (2,1,1);
stem (n,xofn);
title('Original x[n] graph');
xlabel('n'); ylabel ('x[n]');
axis ([0 127 -2 2]);

subplot (2,1,2);
stem (n,z);
title ('Inverse FFT of x(\Omega)');
xlabel ('n'); ylabel ('Reconstructed x[n]');
axis ([0 127 -2 2]);
```





Problems A.2 and A.3) In problem A.2, it is observed that the MATLAB-generated and hand-calculated DTFT graphs for both magnitude and phase spectra are the same with slight differences due to small rounding errors made by MATLAB. As for problem A.3, the result obtained from the inverse Fourier Transform of the MATLAB-generated DTFT results in the original x[n] function.

Problems B.1-B.6

```
% Reza Aablue
% 500966944
% Section 05

% Problem B.1
n = 0:1000;
u = @(n) (n>=0) * 1.0 .* (mod(n,1)==0); % Unit step function u[n].
xofn = @(n) sin (0.2*n*pi).*(u(n)-u(n-10)); % x[n] expression.
omega = linspace (-pi,pi,1001);
W_omega = exp(-1i).^((0:length(xofn(n))-1)'*omega);
XofF = (xofn(n)*W_omega);

figure (1);
plot (abs(XofF)); title ('X(\Omega) = DTFT of x[n]'); xlabel ('\Omega');

% Problem B.2
h = @(n) (u(n)-u(n-10)); % h[n] function.
Wh = linspace (-pi,pi,1001);
Wh_omega = exp(-1i).^((0:length(h(n))-1)'*Wh);
HofF = (h(n)*Wh_omega);
```

```
figure (2);
plot (abs(HofF)); title ('H(\Omega) = DTFT of h[n]'); xlabel ('\Omega');

% Problem B.3
Y1 = XofF.*HofF;
figure (3); plot (abs(Y1)); hold on;

% Problem B.4
Y2 = conv (xofn(n),h(n));

% Problem B.5
Wy = linspace (-pi,pi,1001);
Wy_omega = exp(-1i).^((0:length(Y2)-1)'*Wy);
Y2ofF = (Y2*Wy_omega);

plot (abs(Y2ofF), '--');
hold off;
title ('Comparison of Y(\Omega) obtained through');
legend ('Y(\Omega)=X(\Omega)H(\Omega)','Y(\Omega) = DTFT of y[n]'); xlabel ('\Omega');
```

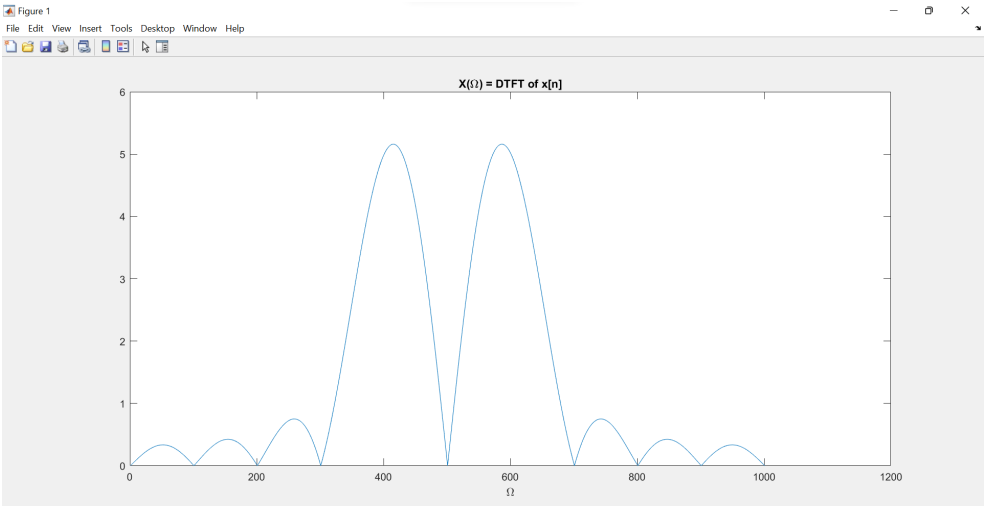


Figure 1
File Edit View Insert Tools Desktop Window Help
X(Ω) = DTFT of x[n]
0 1 2 3 4 5 6
0 200 400 600 800 1000 1200
Ω

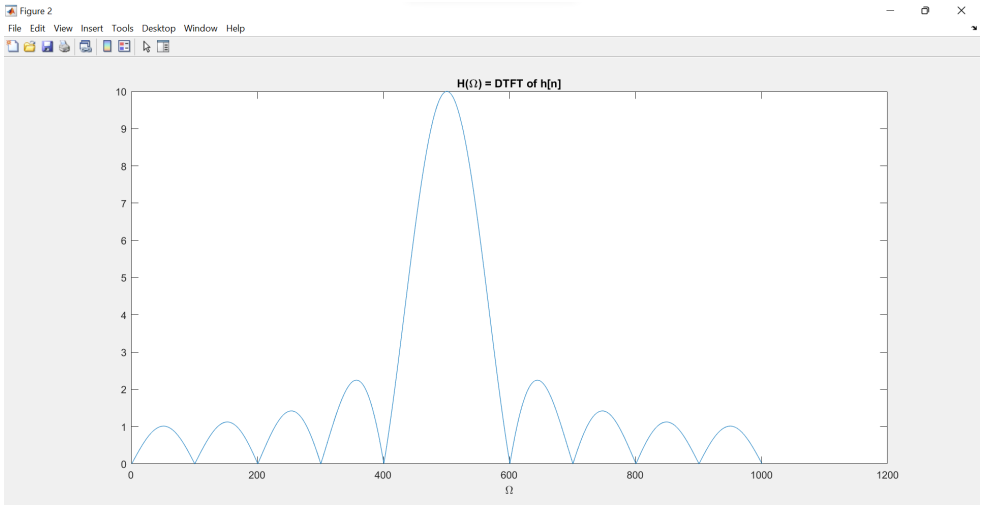


Figure 2
File Edit View Insert Tools Desktop Window Help
H(Ω) = DTFT of h[n]
0 1 2 3 4 5 6 7 8 9 10
0 200 400 600 800 1000 1200
Ω

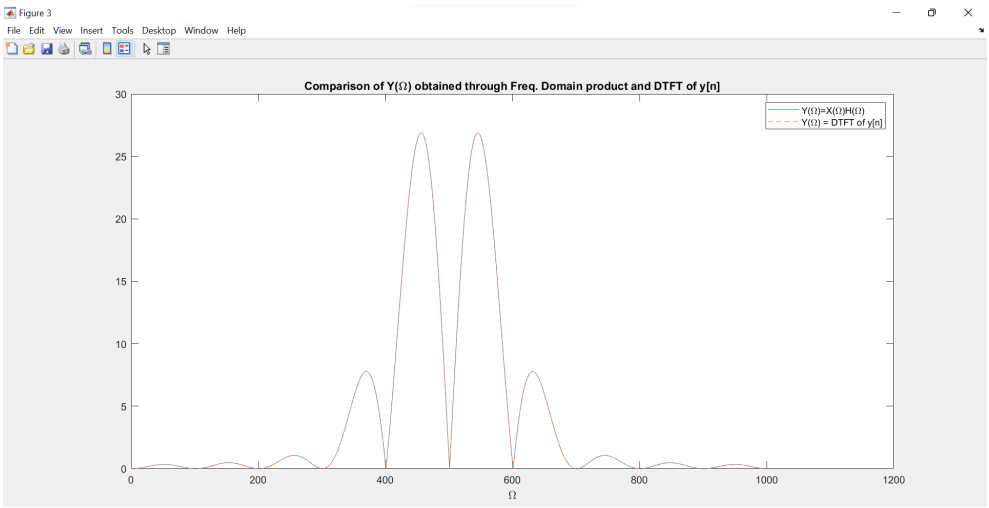


Figure 3
File Edit View Insert Tools Desktop Window Help
Comparison of Y(Ω) obtained through Freq. Domain product and DTFT of y[n]
Y(Ω)=X(Ω)H(Ω)
Y(Ω) = DTFT of y[n]
0 5 10 15 20 25 30
0 200 400 600 800 1000 1200
Ω

Problem B.6) In part B.3, the computation of y[n] = x[n]*h[n] is performed in frequency domain by Y(Ω) = X(Ω) H (Ω). Similarly, in part B.5, the time-domain convolution is converted into frequency domain multiplication through the Discrete-Time Fourier Transform. From the third graph, it is observed that both methods yield the same results as they are effectively doing the same thing, just in different ways.

Problems C.1-C.5

```
% Reza Aablue
% 500966944
% Section 05

% Problem C.1 (N=35)
N = 35; W = linspace (0,2*pi*(1-1/N),N);
H_d_35 = @ (W) (mod(W,2*pi)>(2/3)*pi).*(mod(W,2*pi)<2*pi-(2/3)*pi);
H_35 = H_d_35 (W);
H_35 = H_35.*exp(-1i*W*((N-1)/2));
H_35_real = real (ifft(H_35));

figure (1); stem (H_35_real);
title ('h[n] with filter length N=35 and Wo = 2*pi/3'); xlabel('n');

% Problem C.2 (H(\Omega), N=35)
H_35_ofF = freqz (H_35_real,1,0:2*pi/1001:2*pi);
```

```
% Problem C.4 (h[n], N=71)
N2=71; W2 = linspace (0,2*pi*(1-1/N2),N2);
H_d_71 = @ (W2) (mod(W2,2*pi)>(2/3)*pi).*(mod(W2,2*pi)<2*pi-(2/3)*pi);
H_71 = H_d_71 (W2);
H_71 = H_71.*exp(-1i*W2*((N2-1)/2));
H_71_real = real (ifft(H_71));

figure (2);
stem (H_71_real); title ('h[n] with filter length N=71 and Wo = 2*pi/3');
xlabel ('n');

% Problems C.2 and C.4 (H(\Omega), N=35 and N=71)
H_71_ofF = freqz (H_71_real,1,0:2*pi/1001:2*pi);

figure(3); plot (abs (H_35_ofF)); hold on;
plot (abs (H_71_ofF)); hold off;
title ('Comparison of H(\Omega) with filter lengths N=35, N=71');
xlabel ('\Omega'); legend ('N=35','N=71');
```

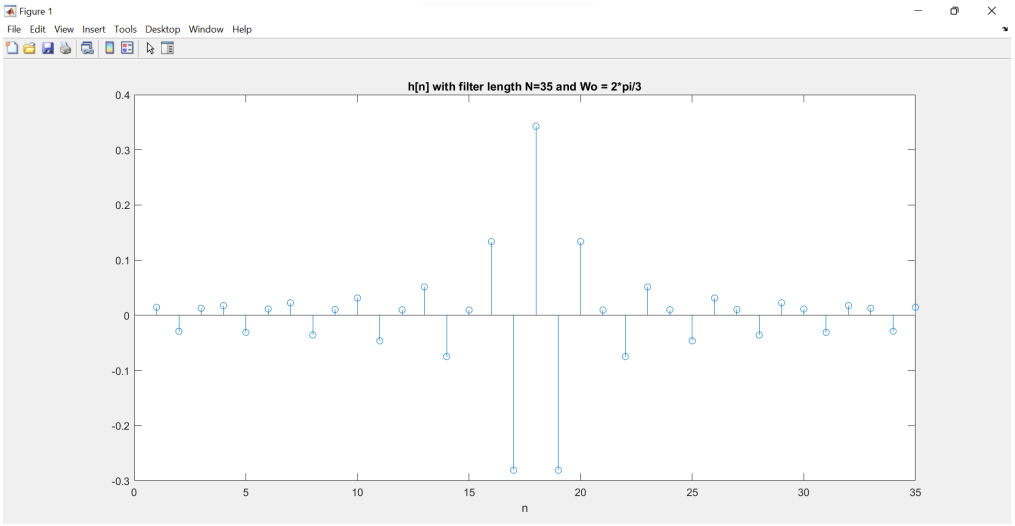


Figure 1
File Edit View Insert Tools Desktop Window Help
h[n] with filter length N=35 and Wo = 2*pi/3
n

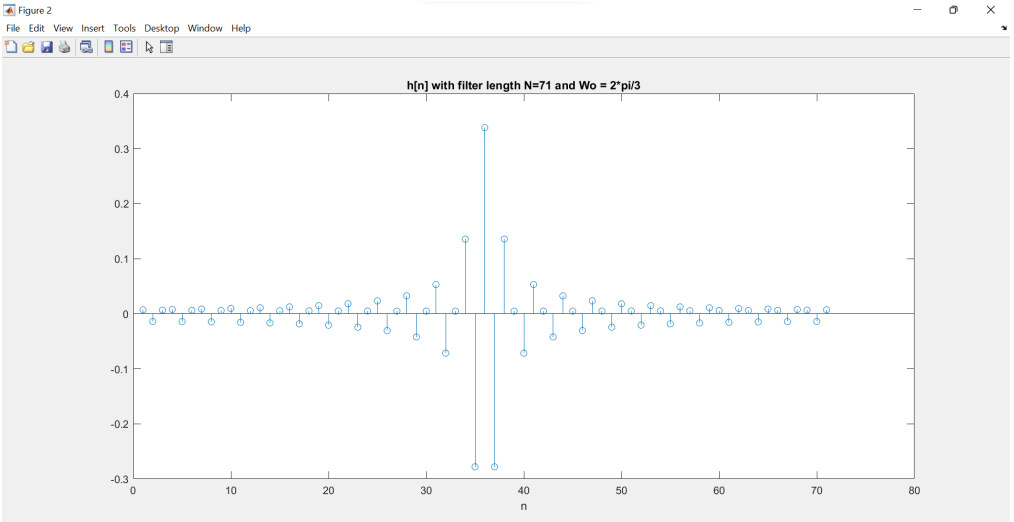


Figure 2
File Edit View Insert Tools Desktop Window Help
h[n] with filter length N=71 and Wo = 2*pi/3
n

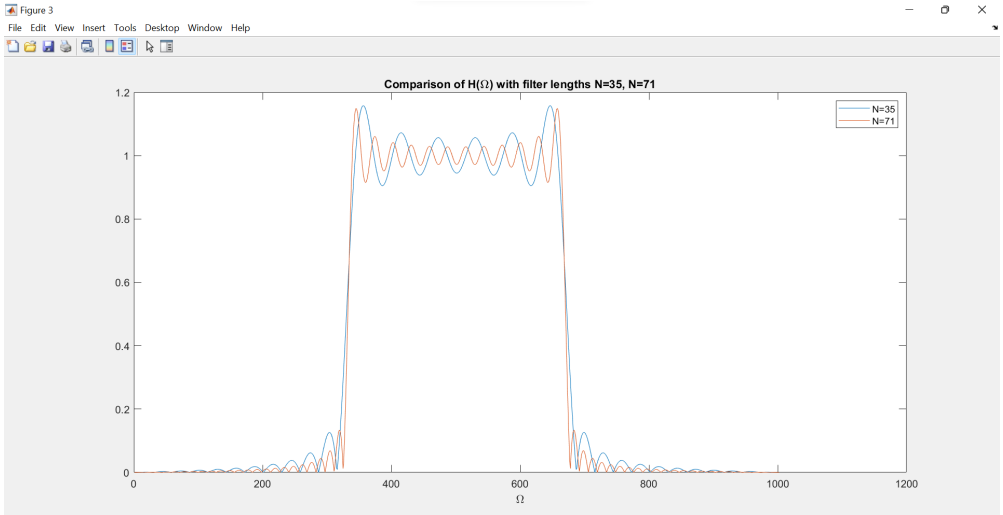


Figure 3
File Edit View Insert Tools Desktop Window Help
Comparison of H(Ω) with filter lengths N=35, N=71
N=35
N=71
Ω

Problem C.3) The ideal and perfect filter accounts for no noise around the edges (bandwidth length), and incorporates an idealistic approach to “blocking out” unwanted noise. The regular filter, however, would have noise around its edges and the noise would affect the input and output signals.

Problem C.5) The filter that has length N=71 is more accurate than that of length N=35 as it contains more samples of the input signal. Therefore, with increasing the filter length, the accuracy of sampling improves as well.