BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING,
DHAKA-1000, BANGLADESH.

COURSE NO.: EEE 312

List of Experiments

1. Study of Sampling, Quantization and Encoding.
2. Time domain analysis of discrete time signals and systems
3. *Z*-transform and Its Application
4. Frequency domain analysis of DT signals and systems
5. FIR Filter Design

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING,
DHAKA-1000, BANGLADESH.

COURSE NO.: EEE 312

Experiment No. 1

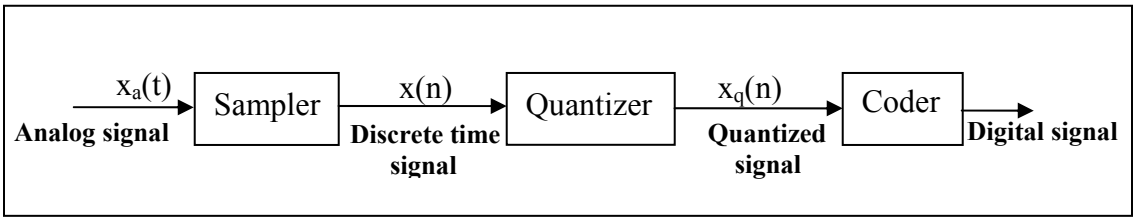
Study of Sampling, Quantization and Encoding

Most signals of practical interest, such as speech, biological signals, communication signals etc. are analog. These signals must be processed for different purposes. Digital signal processing of an analog signal is preferable to processing the signals directly in the analog domain because of its flexibility in reconfiguration, better accuracy in control, better storing capability and cost effectiveness. That's why the analog signals are to be converted into corresponding digital domain for the purpose of processing. The analog signals are converted into digital signals through sampling, quantization and encoding.

PRELAB WORK:

- **Read this laboratory tutorial carefully before coming to the laboratory class, so that you know what is required.**
- Try to follow the lecture notes of EEE 311.
- Familiarize yourself with relevant MATLAB functions and codes necessary for this experiment.
- **Do not bring any prepared MATLAB code in the lab with any portable device.**

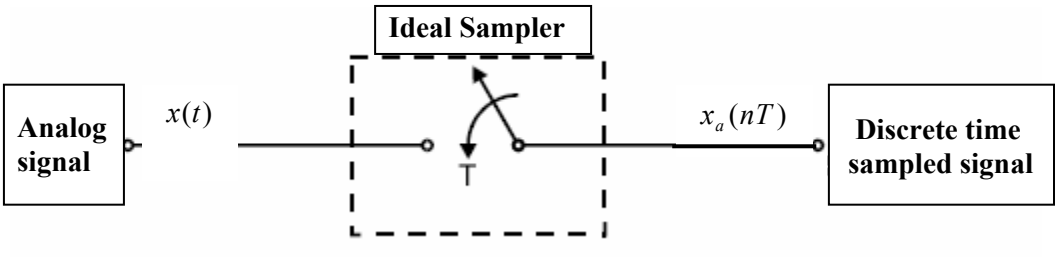
THEORY:



Analog to digital conversion process

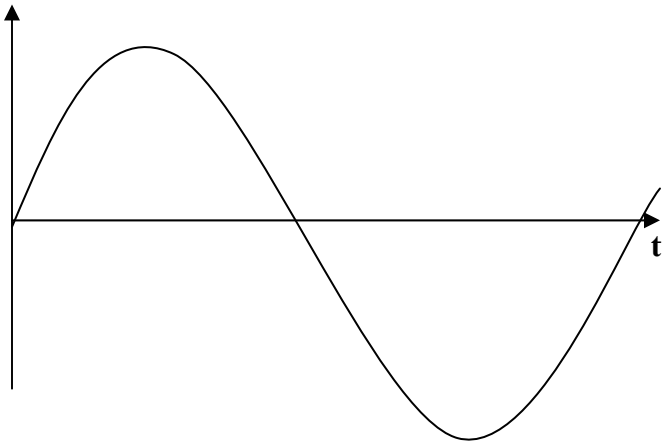
Sampling:

In this section, a continuous time signal is converted into a discrete time signal by taking samples at discrete time instants.

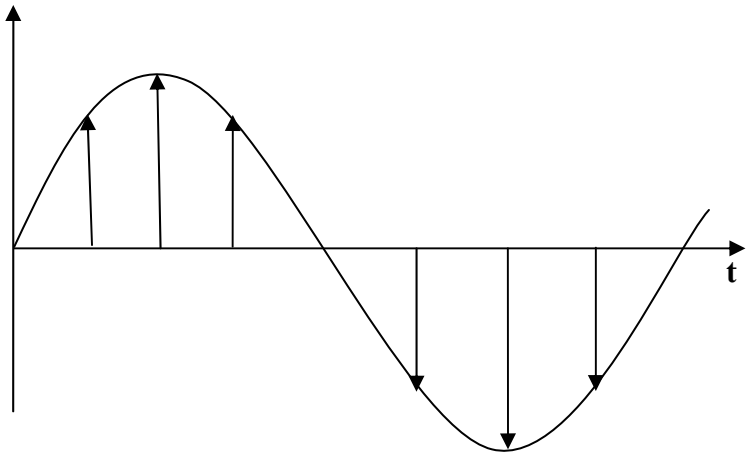


Sampling Process

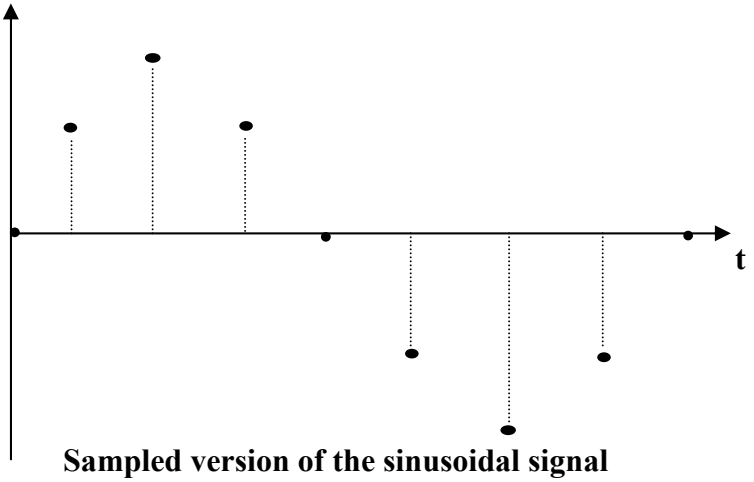
The analog signal is sampled once every T seconds, resulting in a sampled data sequence. The sampler is assumed to be ideal in that the value of the signal at an instant (an infinitely small time) is taken. The most important parameter in the sampling process is the sampling period T, or the sampling frequency or sampling rate f_s, which is defined as $f_s = \frac{1}{T}$. Sampling frequency is given in units of 'samples per second' or 'hertz'.



A sinusoidal Signal



Sampling operation is being performed



Sampled version of the sinusoidal signal

If the sampling is too frequent, then the DSP processor will have to process a large amount of data in a much shorter time frame. If the sampling is too sparse, then important information might be missing in the sampled signal. The choice is governed by sampling theorem.

Sampling Theorem:

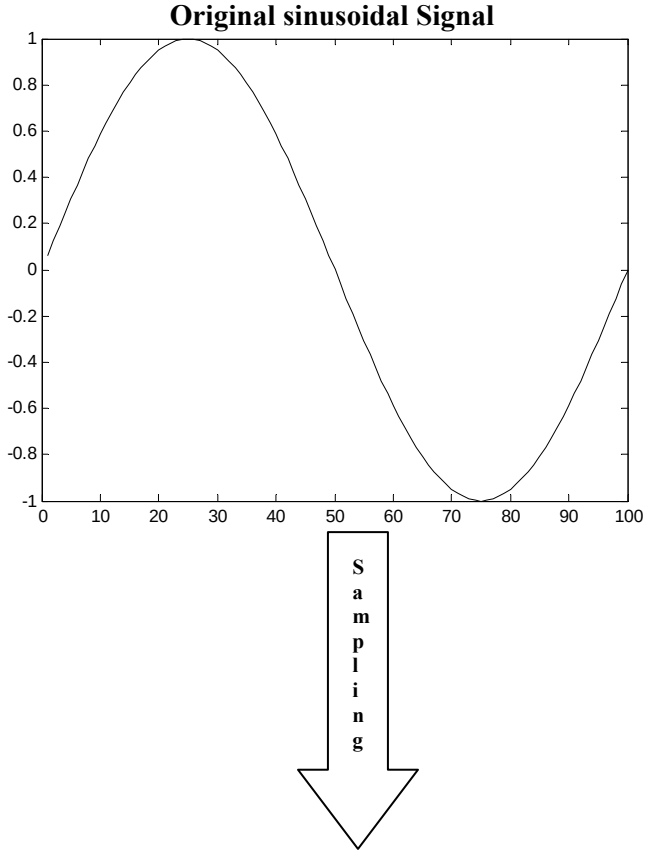
The sampling theorem specifies the minimum-sampling rate at which a continuous-time signal needs to be uniformly sampled so that the original signal can be completely recovered or reconstructed by these samples alone. The reconstruction of a sampled signal is done by simple low pass filtering. For successful reconstruction of the sampled signal, the sampling frequency must be equal to or greater than the twice time of the highest frequency component of the original signal. Otherwise the signal cannot be successfully recovered.

If a continuous time signal contains no frequency components higher than W Hz, then it can be completely determined by uniform samples taken at a rate samples f_s per second

where $f_s \geq 2W$,

or, in terms of sampling period, $T \leq \frac{1}{2W}$.

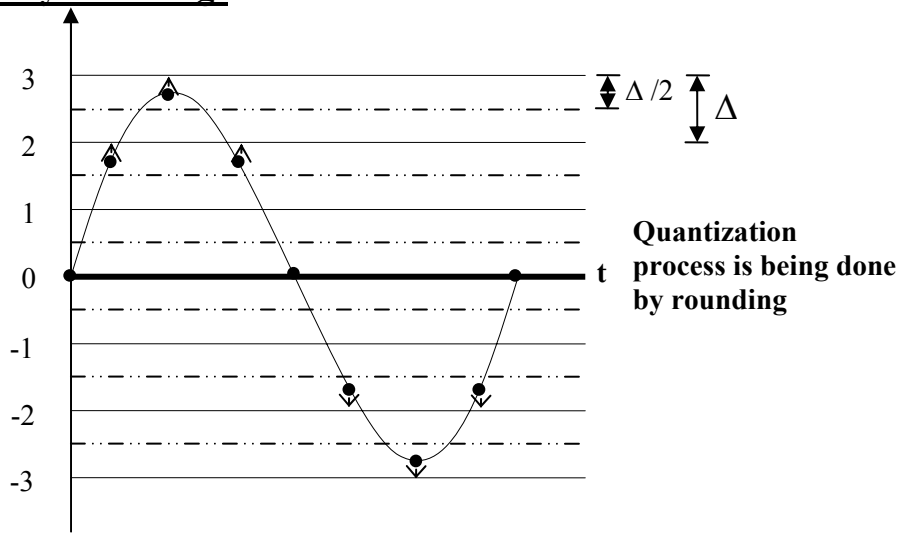
In MATLAB, the signals obtained before and after sampling look like –



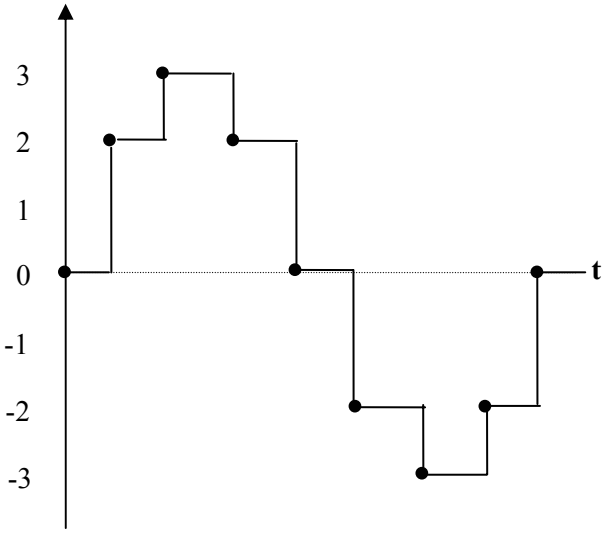
Quantization:

In this section, a discrete time continuous-valued signal is converted into a discrete-time discrete-valued (digital) signal. The value of each signal sample is represented by a value selected from a finite set of possible values. The quantizer assigns each sample of $x(n)$ to the nearest quantization level by either **rounding or truncation.**

Quantization by rounding:

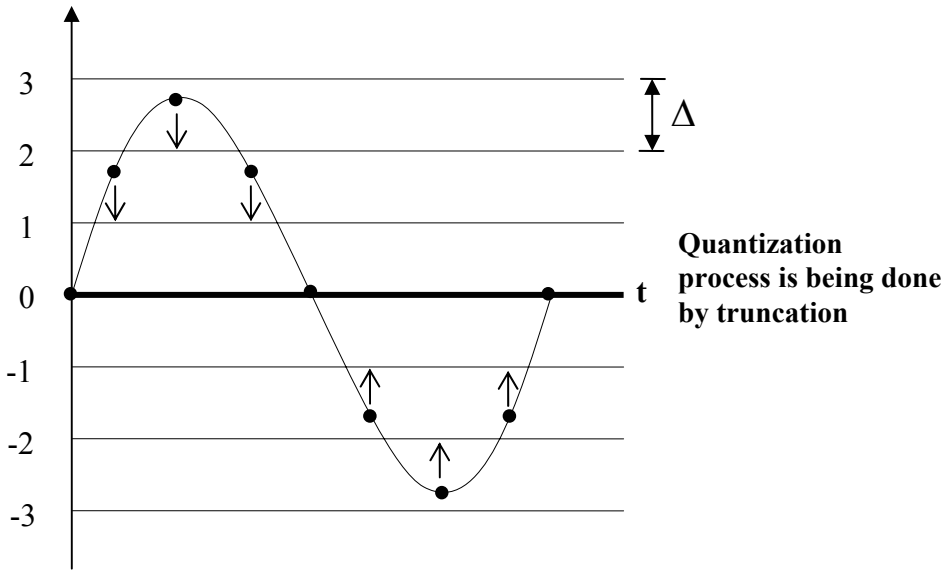


In the case of rounding, the step size is divided into upper and lower halves. The value in the upper half are stepped into the next level and the value in the lower half remains in that level.

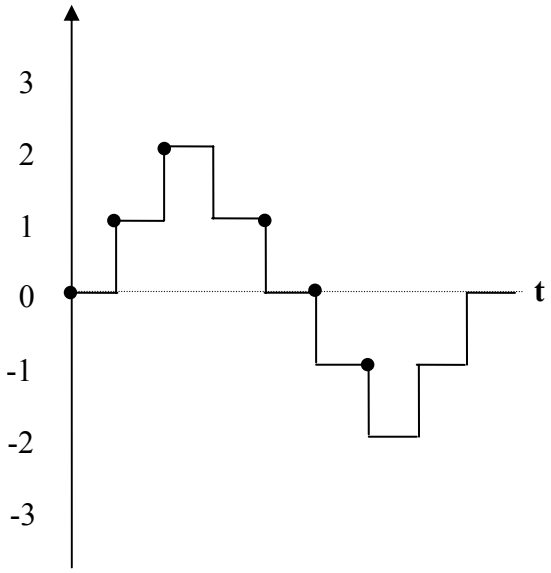


Quantized signal

Quantization by truncation:



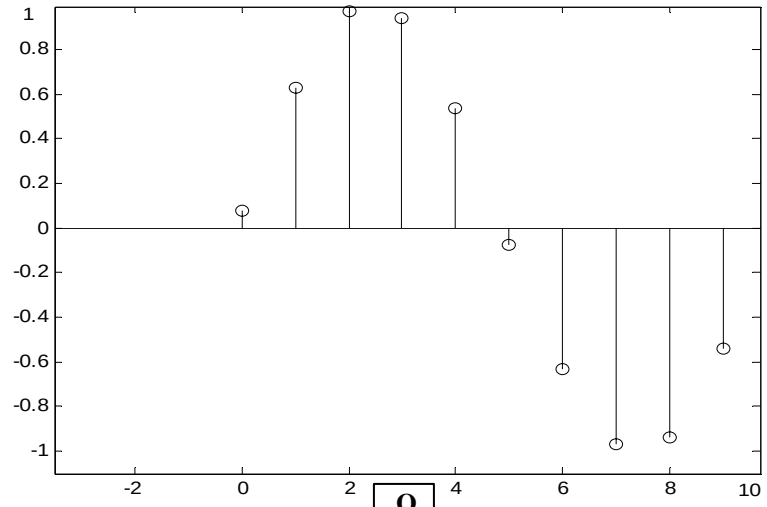
In the case of truncation, the extended value for a particular level is cut down and the sampled point stays at that particular level.



Quantized signal

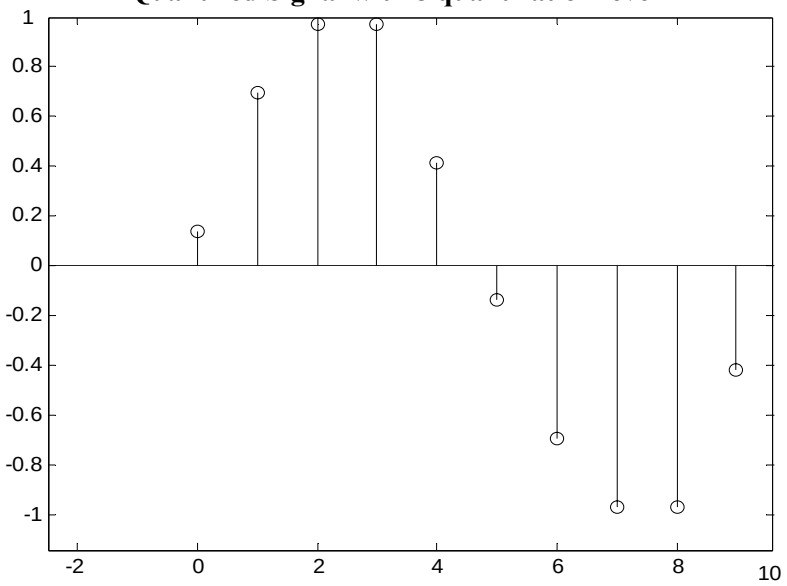
In MATLAB, the sampled signals obtained before and after quantized sampling look like -

Sampled Signal with 100Hz sampling frequency



Quantization

Quantized Signal with 8 quantization level



Performing the quantization operation on $x(n)$, the quantized output $x_q(n)$ is obtained. The difference between unquantized sample $x(n)$ and quantized output $x_q(n)$ is called quantization error, $e_q(n)$. The finite set of possible values are called quantization levels. The distance Δ, between two successive quantization levels is called the quantization step size or resolution.

The quantization error $e_q(n)$ in rounding is limited to

$$-\Delta/2 \text{ to } \Delta/2,$$

$$\text{i.e. } -\Delta/2 \le e_q(n) \le \Delta/2$$

If $x_{\min}$ and $x_{\max}$ represent the minimum and maximum value of $x(n)$ and L is the number of the quantization levels, then

$$\Delta = \frac{x_{\max} - x_{\min}}{L-1}$$

If $x_{\max} = +V$ and $x_{\min} = -V$, and if b is the number of bits then

$$\Delta = \frac{2V}{2^b - 1}$$

The quantization error is then given by –

$$e(n) = x_Q(n) - x(n)$$

The mean-square value of the error is given by –

$$\sigma_e^2 = \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2} e^2(n)de(n) = \frac{\Delta^2}{12}$$

The performance of the A/D converter is characterized by SQNR, defined as

$$SQNR = \frac{\sigma_x^2}{\sigma_e^2} \quad \text{---------- (A)}$$

$$SQNR_{dB} = 10\log_{10} SQNR = 10\log_{10}\sigma_x^2 - 10\log_{10}\sigma_e^2$$

So, $SQNR = \frac{\sigma_x^2}{\sigma_e^2} = \frac{\sigma_x^2}{\Delta^2/12} = \frac{12\sigma_x^2}{\Delta^2} = \frac{12\sigma_x^2(2^b-1)^2}{(2V)^2} = 3\frac{\sigma_x^2}{V^2}(2^b-1)^2$

and $SQNR_{dB} = 10\log_{10} 3 + 10\log_{10}\left(\frac{\sigma_x^2}{V^2}\right) + 10\log_{10}(2^b-1)^2$

For higher values of b like 8, 10 or 12, we can approximate, $2^b - 1 \cong 2^b$

$$SQNR_{dB} \cong 10\log_{10} 3 + 10\log_{10}\left(\frac{\sigma_x^2}{V^2}\right) + 10\log_{10} 2^{2b}$$

$$= 4.77 + 10\log_{10}\left(\frac{\sigma_x^2}{V^2}\right) + 6b$$

Thus for each bit added to the A/D converter, the SQNR is improved by approximately 6 dB. For sine wave input with amplitude = 1 volt,

$$SQNR_{dB} = 4.77 + 10\log_{10}\sigma_x^2 - \cancel{10\log_{10} V^2}^{\,=0} + 6b$$

$$= 1.76 + 6b \qquad \text{---------- (B)}$$

where, b= Number of bits

Coding:

The coding process in an A/D converter assigns a unique binary number to each quantization level. If L is the number of levels, then L levels can be represented by b bits where $2^b = L$ or $b = \log_2 L$.

LAB WORKS :

- Familiarize yourself with the MATLAB commands that will be required for this lab.
- You may require the following MATLAB built-in functions for this experiment –

sin(), interp1(), length(), max(), min(), stem(), plot(),find(), ones(), ceil(), dec2bin()

Part – A

Sampling of an analog signal and reconstruction of the sampled signal.

1. Generate a signal, $x(t) = \sin(2\pi 10t + 0^0)$

2. Take the samples of this signal at a sampling rate of 20 Hz.

3. Reconstruct the analog signal by interpolation. [Use interp1() function]

4. Repeat steps 2 and 3 at a sampling rate of 50 Hz, 100 Hz and 10 Hz.

5. Generate another signal, $y(t) = \sin(2\pi 10t) + \sin(2\pi 50t) + \sin(2\pi 100t)$

6. For successful reconstruction of the signal, what should be its sampling frequency? Verify it though the program.

Comment on the obtained results in step – 4 and 6. Explain the results.

Part – B

Study of quantization and quantization error.

1. Write a MATLAB source code that will realize the uniform quantizer .

2. Take the signal of step – 5 in Part – A and sample it by 200 Hz sampling frequency**.**

3. Quantize the signal by a 3 bit uniform quantizer.

4. Obtain the quantization noise power.

5. Obtain SQNR from equations (A) and (B) which are described in the theory part.

6. Reconstruct the original signal by interpolation. [Use interp1() function]

7. Repeat steps 3 to 6 for 4 and 6 bit uniform quantizer.

Comment on the obtained results in step - 6. Explain the result.

Part C

Encoding of the quantized sequence.

In the next stage of quantization, the sampled data will be converted into binary data. For this, first you have to represent the quantization levels into decimal value and then have to convert it into binary data.

- Obtain a sampled quantized data sequence.
- Represent each level by a definite binary number. The number of digits $b = \log_2 L$, where L is the quantized level.

Home tasks

Theory:

An important application of this experiment is in Pulse Code Modulation (PCM) system. PCM is a method for sampling and quantizing an analog signal for the purpose of transmitting or storing the signal in digital form. It is widely used for speech transmission in telephone communications and for telemetry systems that employ radio transmission.

Speech signals have the characteristics that small signal amplitudes occur more frequently than large signal amplitudes. A uniform quantizer provides the same spacing between successive levels throughout the entire dynamic range of the signal. A better approach is to use a non uniform quantizer, which provides more closely spaced levels at the low signal amplitudes and more widely spaced levels at the large signal amplitudes. A non uniform quantizer characteristic is usually obtained by passing the signal through a non linear device that compresses the signal amplitudes, followed by a uniform quantizer. With the use of non – uniform quantizer prior to a uniform one, quantization noise is greatly reduced.

Non – uniform quantizer are of two types –

- U.S. and Canadian standard
- European standard

U.S. and Canadian telecommunication systems use μ- law compressor, which has the input – output characteristics of the form –

$$y = \frac{\ln(1 + \mu|s|)}{\ln(1 + \mu)} \operatorname{sgn}(s) \qquad |s| \le 1, \quad |y| \le 1$$

where s is the normalized input, y is the normalized output, sgn(.) is the sign function and μ is a parameter that is selected to give the desired compression characteristics. This system adopted μ=255 for encoding of speech waveforms.

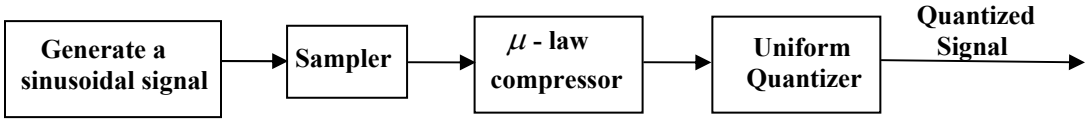
European telecommunication systems use A- law compressor, which has the input – output characteristics of the form –

$$y = \begin{cases} \dfrac{1+\ln(A|s|)}{1+\ln A}\,\mathrm{sgn}(s), & \dfrac{1}{A} \le |s| \le 1 \\ \dfrac{A|s|}{1+\ln A}\,\mathrm{sgn}(s), & 0 \le |s| \le \dfrac{1}{A} \end{cases}$$

where A is chosen as 87.56.

[See Chapter 10 of the 2nd reference]

Simulate the following system with MATLAB –



Process the sinusoidal signal through the above sampler, μ law compressor and uniform quantizer. As a part of processing, you have to do the following -

- **Write MATLAB codes for each section**
- **Show the waveshapes for each section**
- **Compute the signal to quantization-noise ratio (of the quantized signal) with the given formulae(for signal in step 5 part A).**

References:

1) Proakis & Manolakis, **"Digital Signal Processing: Principles, Algorithms and Applications.",** Chapter 1, 3rd Edition , Prentice Hall Ltd.

2) Ingle & Proakis, " Digital Signal Processing using *MATLAB* ", Edition 2000 Thomson-Brooks/Cole Co Ltd.

The laboratory tutorial of this experiment is prepared by –

Imtiaz Ahmed
Lecturer, Dept. of EEE, BUET.

Under the supervision of –

Dr. Md. Kamrul Hasan
Professor, Dept. of EEE, BUET.

June 9, 2007

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING,
DHAKA-1000, BANGLADESH.

COURSE NO.: EEE 312

Experiment No. 2

Time domain analysis of discrete time signals and systems

% "The purpose of computing is insight, not numbers." – R.W. Hamming

In this experiment, mainly time domain properties of discrete time signals and systems will be analyzed. First of all different discrete time sequence generation and synthesis will be performed. Then LTI system response in terms of convolution and difference equation will be observed. Last but not the least, correlation of two signals, its uses in different fields will be analyzed.

Besides some MATLAB based practical pertinent examples will be presented too. Several related exercises and real-time applications are listed for proper comprehension of the theory.

Pre-lab Work:

1. Familiarize yourself with the experiment manual before attending the lab.
2. Practice the examples and exercises listed in this experiment at home for better class performance.
3. DO NOT bring any relevant MATLAB codes, neither in any portable device nor in written form.

Important MATLAB functions used in this experiment:

stem(), **fliplr**(), **min**(), **max**(), **conv**(), **filter**(), **xcorr**(), **randn**(), **rand**()

Part A

Generating basic sequences:

a) To implement unit impulse sequence

$$\delta(n-n_0)=\begin{cases}1, & n=n_0\\ 0, & n\neq n_0\end{cases} \quad \text{over} \quad n_1 \le n_0 \le n_2 \tag{1}$$

% Generate impulse sequence
% Say, n_1= -3 and n_2=3 (7 point sequence). Consider lag, n_o = -1
>>n_1= -3;
>>n_2= 3;
>>n=n_1:n_2;
>>n_o= -1;
>>x_1=[(n-n_o) = = 0]; % using logical argument
>>stem(n,x_1);

b) To implement unit step sequence

$$u(n-n_0)=\begin{cases}1, & n\geq n_0\\ 0, & n< n_0\end{cases} \quad \text{over} \quad n_1 \le n_0 \le n_2 \tag{2}$$

just set x_2 = [(n-n_o)>=0] in the above program.
Sample results are shown in Figure 1.

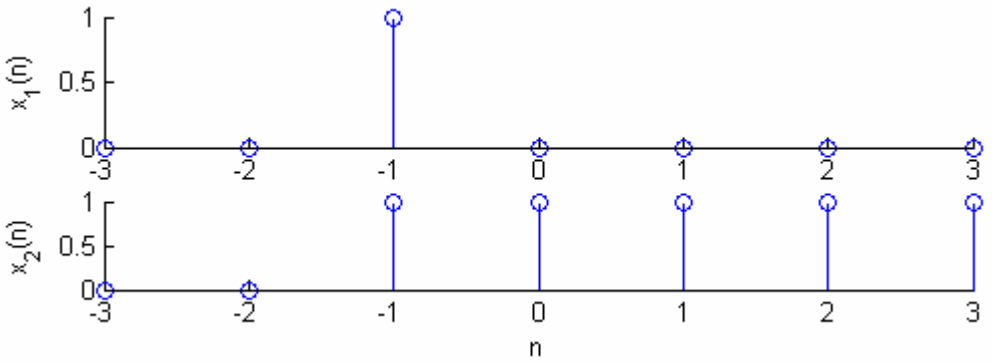


Figure 1: $x_1(n) = \delta(n+1)$ and $x_2(n) = u(n+1)$

Lab Exercise : A.1

Generate a ramp sequence (shown in Figure 2) following the above program.(length and lag must be variable).

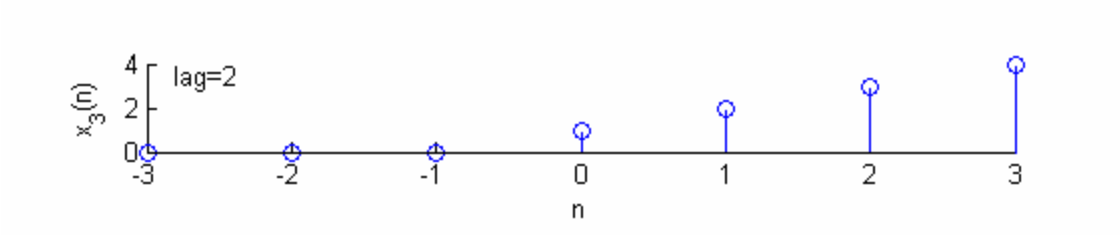


Figure 2: Result of Lab Exercise: A.1

Lab Exercise: A.2

Let n_1 denotes the time index for $x_1(n)$ and n_2 represent the time index for $x_2(n)$.Write a general program to find $x(n)=x_1(n)+x_2(n)$.The index of x(n) will start with the minimum of n_1 and n_2 and end with the maximum of n_1 and n_2.

Given, $x_1(n)=\{\underset{\uparrow}{0}, 1, 2, 3\}$ and $x_2(n)=\{0, \underset{\uparrow}{1}, 2, 3\}$.

Lab Exercise : A.3

The up-sampler is a discrete-time system defined by the input-output relation

$$y(n) = \begin{cases} x(\frac{n}{L}), & n = 0, \pm L, \pm 2L, \ldots \\ 0, & otherwise \end{cases} \quad (3)$$

L is called the up-sampling factor which is a positive integer greater than 1. L-1 equidistant zero samples are inserted by the up-sampler between two input samples. It finds applications in sampling rate alteration process. Say input is a sine wave with frequency 0.36 rad/sec and L = 3 . Take time index, n = 1 to 52. Observe the up-sampled signal up to 52 samples. (Result shown in Figure 3)

Lab Exercise : A.4

Find the even and odd parts of the ramp signal obtained from Exercise 1.1
(Result shown in Figure 4)

Some helpful m-file routines:

```
function [y,n]=sigshift(x,m,n0) % For signal
shifting
n=m+n0;
y=x;

function [y,n]=sigfold(x,n) % For signal folding
y=fliplr(x);
n=-fliplr(n);
```

```
function [y,n]=sigadd(x1,n1,x2,n2) % For adding
%two signals
n=min(n1(1),n2(1)):max(n1(end),n2(end));
y1=zeros(1,length(n));
y2=y1;
y1(find((n>=n1(1))&(n<=n1(end))))=x1;
y2(find((n>=n2(1))&(n<=n2(end))))=x2;
y=y1+y2;
```

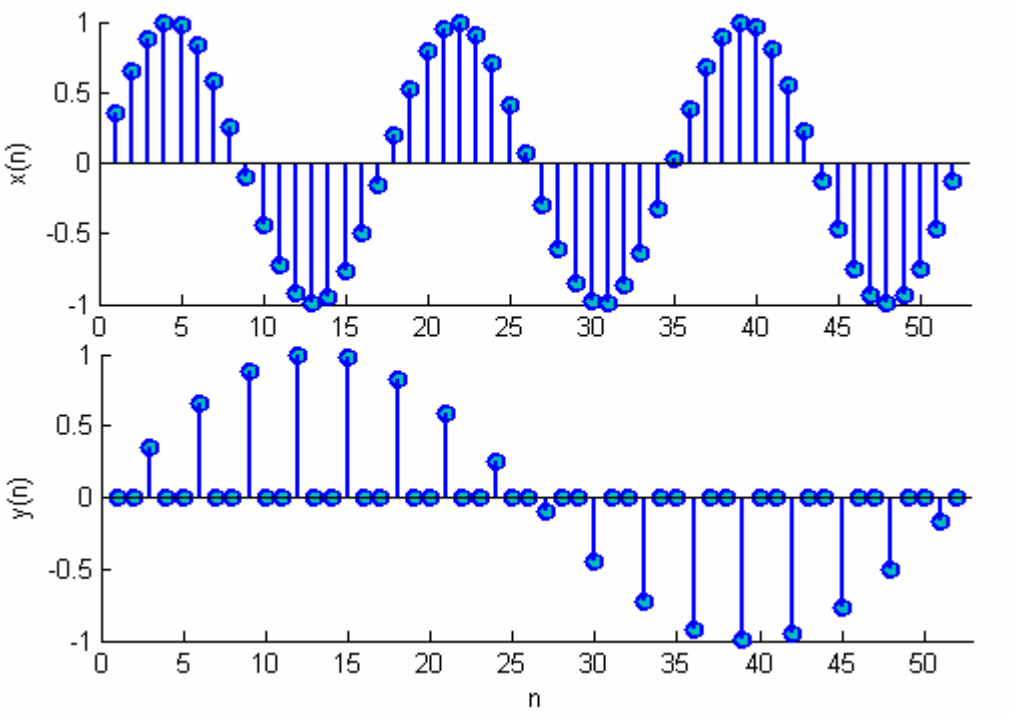


Figure 3 : Result of Lab Exercise : A.3

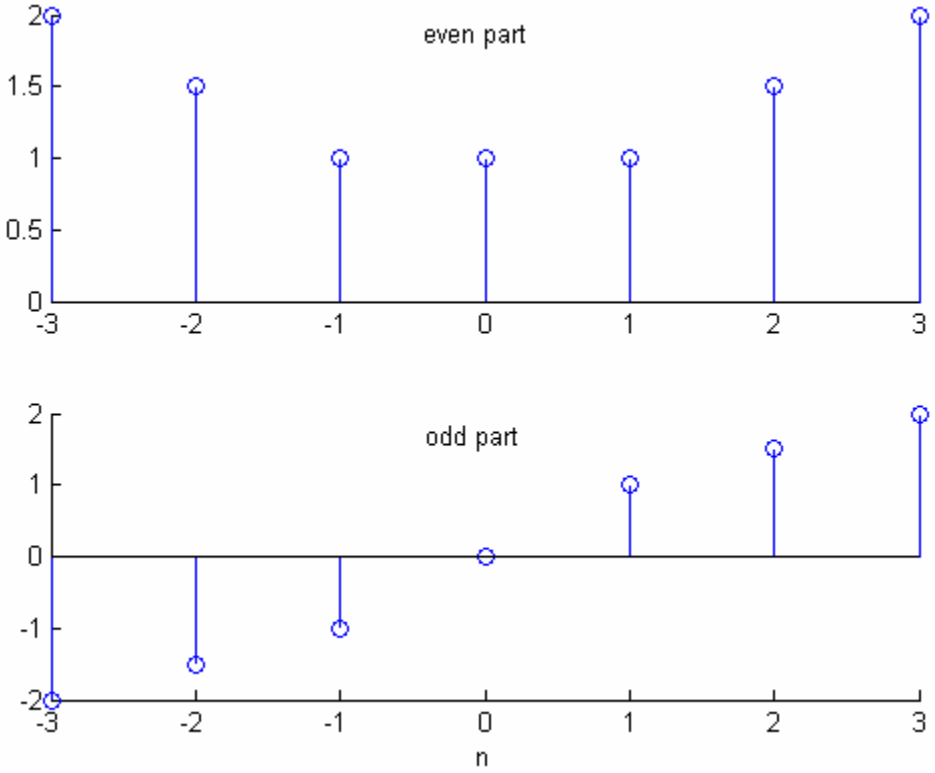


Figure 4 : Result of Lab Exercise : A.4

Part B

Response of LTI Systems to Arbitrary inputs: Convolution

For a relaxed LTI system, the response *y(n)* to a given input signal *x(n)* can be obtained if we know the system impulse response *h(n)* . Then response *y(n)* is given by the following relation

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = x(n) * h(n) \quad (4)$$

which is known as the convolution sum.

Example:

Using **conv**() function , we can compute the convolution sum of two discrete time signals. But the outcome of conv() function does not reveal the timing information. So time index of the sum should be derived from the signals to be convolved. Recall that the lowest time index of the convolution sum is the sum of the lowest time indices of the two signals to be convolved. Same thing applies for highest time index of the sum. Let

$\mathbf{x}_1$=[4 2 6 3 8 1 5]; $\mathbf{x}_2$=[3 8 6 9 6 7];

```
>>x1=[4 2 6 3 8 1 5];
>>n1=[-2:4];% generating index
>>x2=[3 8 6 9 6 7];
>>n2=[-4:1];
>>kmin=n1(1)+n2(1);% left edge of convolved result
>>kmax=n1(end)+n2(end); % right edge of convolved result
>>y=conv(x1,x2);
>>k=kmin:kmax; % generating index of the result
>>subplot(311),stem(n1,x1)
>>subplot(312),stem(n2,x2)
>>subplot(313),stem(k,y)
```

Lab Exercise: B.1

In the previous example, $x_1(n)$, $x_2(n)$ and y(n) are all of different lengths and scale, and this is far from ideal (Figure 5). Write a general program to bring three plots into same horizontal scale.(Figure 6)

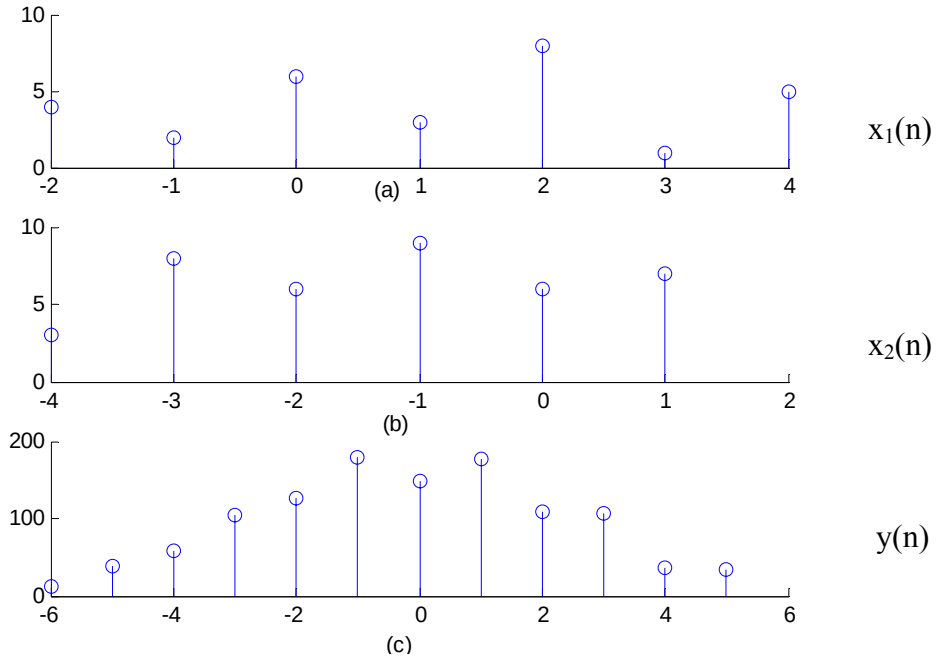


Figure 5 : With different horizontal scale

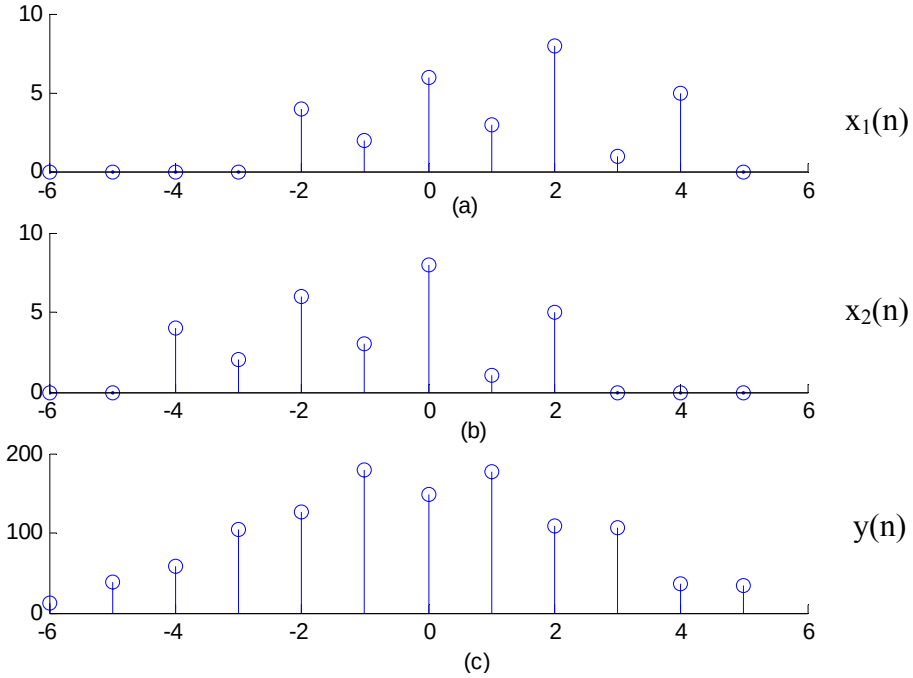


Figure 6 : With same horizontal scale

m-file which returns both convolved result and index :

```
function [y ny]=conv_m(x,nx,h,nh)
nyb=nx(1)+nh(1);
nye=nx(length(x))+nh(length(h));
ny=[nyb:nye];
y=conv(x,h)
```

Part C

System Response from difference equations:

An important subclass of LTI discrete time (DT) systems is characterized by a linear constant co-efficient difference equation of the form

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k) \quad (5)$$

where x(n) and y(n) are, respectively, the input and output of the system. (N≥M assumed). $\{a_k\}$ and $\{b_k\}$ are the coefficients. We can modify the equation as

$y(n) = -\sum_{k=1}^{N} \frac{a_k}{a_0} y(n-k) + \sum_{k=0}^{M} \frac{b_k}{a_0} x(n-k)$. Taking a_0 inside a_k and b_k,we get

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \quad (6)$$

Based on the above equation, a causal LTI system can be simulated in MATLAB using **filter**() function. Its general form is

y = filter(b,a,x); % remember co-efficient a_0 must be non-zero.
If a_0 is not equal to 1, filter() normalizes the system coefficients by a_0

Lab Exercise: C.1

Consider a system described by the following equation

y(n) + 0.6y(n-1)= x(n)

Use MATLAB to find impulse response, step response and sinusoidal response for x(n)=0.5sin(n)u(n) in the range -10≤n≤20. Verify the result using conv() function for step and sinusoidal responses.

Lab Exercise C.2:

Now consider

y(n) + 0.6y(n-1)= x(n)+x(n-2)

Obtain step response for this system applying superposition theorem. DO NOT use filter() function in that case. Verify your result using filter() function later on.

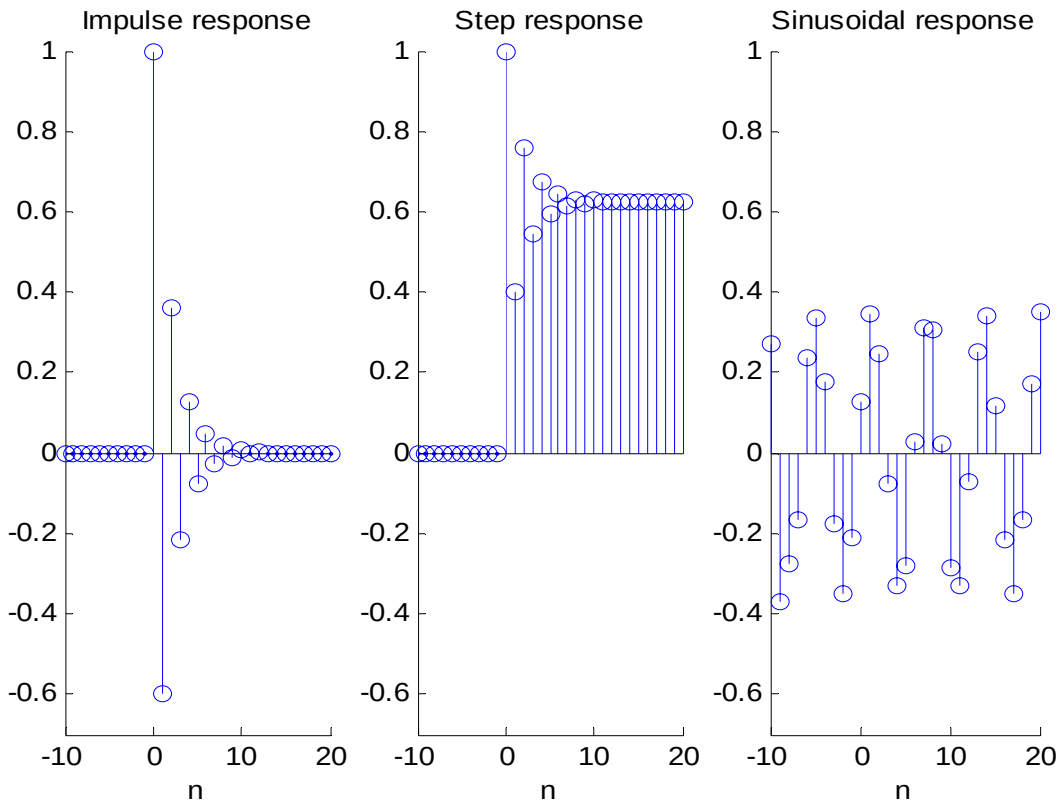


Figure 7 : Result of Exercise C.1

Part D

Correlation:

The cross-correlation of two deterministic real discrete-time finite energy sequences x(n) and y(n) is a third sequence $r_{xy}(l)$ defined as

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n - l)\,. \tag{7}$$

In dealing finite duration sequences where x(n)=y(n)=0 for n<0 and n>N-1

$$r_{xy}(l) = \sum_{n=i}^{N-|k|-1} x(n)y(n - l) \tag{8}$$

where $i = l$, $k = 0$ for $l \geq 0$ and $i = 0$,$k = l$ for $l < 0$(See Appendix for clarification)

A distinct peak in the CCF indicates that the two signals are matched for that particular time shift. This has important applications in signal detection and system identification.

Example:

Consider,

y(n)=x(n-2) where x(n)=[3 11 7 0 -1 4 2]. Then cross-correlating x(n) with y(n) we get

In the following figure, it is apparent by inspection that the CCF is very small if k=0 but substantial if k=2. This is because y(n) lags behind x(n) by 2 samples and a match therefore occurs at 2 samples delay.

```
>>x = [3,11,7,0,-1,4,2];
>>n = -3:3;
>>[y,ny] = sigshift(x,n,2)
>>[x,nx] = sigfold(x,n);
>>[rxy,nxy] = conv_m(x,nx,y,ny);
>>stem(nxy,rxy/max(rxy))
```

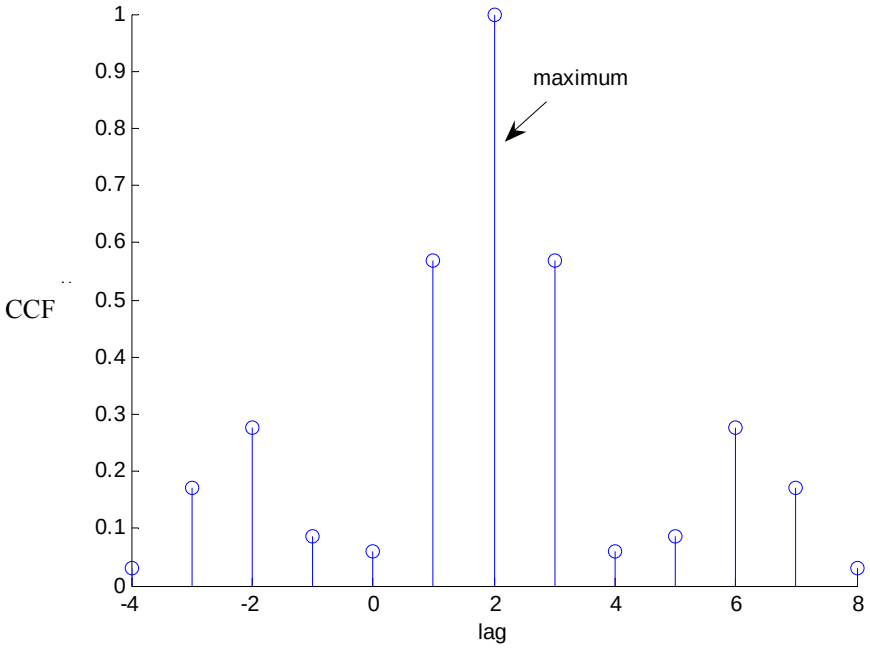


Figure 8 : Result of cross-correlation between x(n) and y(n)

There is a connection between convolution and correlation.

For convolution, we set $x(n) * y(n) = \sum_{n=-\infty}^{\infty} x(n)y(k-n)$

Then if we set, $x(n) * y(-n) = \sum_{n=-\infty}^{\infty} x(n)y(n-k)$ we get the desired result for CCF.

$$r_{xy}(k) = \sum_{n=-\infty}^{\infty} x(n)y(n-k) = x(n) * y(-n) \quad (9)$$

So we can compute CCF by **conv**() function too.

r_{xy} = conv(x,fliplr(y));% Here **fliplr**() function folds y.

If the signals are ergodic then CCF is defined as

$$\hat{r}_{xy}(k) = \lim_{M \to \infty} \frac{1}{2M + 1} \sum_{n=-M}^{M} x(n)y(n - k) \quad (10)$$

For random signals we have to calculate $r_{xy}(k)$ as E[x(n)y(n-k)] to be exact. But to calculate an expected value accurately we need infinite data points. In practice data is finite so we take the estimated CCF as averaged result.

For *N* point periodic signals a simpler alternative is

$$\hat{r}_{xy}(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n)y(n - k) \quad (11)$$

Similarly auto-correlation function can be defined. Just replace y(n) with x(n).

Note: ACF of a periodic signal is periodic.

Exercise: D.1

a) Determine the ACF of a sine wave sin(w_ot) analytically.(Homework)

b) Take period, T=2ms, tstep as T/100.

1. Take t=tstep:tstep:T ;
   Find ACF for x=2*sin(2*pi*t/T).
2. Repeat 1 for t=tstep:tstep:3*T

Explain for (b.2) why ACF magnitude decreases for successive periods? What happens if we set t(end)=5T , 10T etc.?

c) Observe the ACF of a random white noise sequence.

Applications of ACF/CCF:

1. Detecting a periodic input corrupted by AWGN

This is an example (See Figure 9-10) that illustrates the use of ACF to identify a hidden periodicity in an observed signal. This example will demonstrate how white noise gets suppressed in auto-correlation domain. For instance, you can observe the ACF of a white noise sequence which is characterized by an impulse (i.e noise power) at zero lag and nearly zero at other locations. The more the data points you take, the more the result matches an impulse.

```
%observe the result for different SNR(0,-5,-10 dB)
>>T=2e-3; % period=2ms
>>tstep=T/100;
>>t=tstep:tstep:3*T; % taking time index upto 3 periods
>>x=2*sin(2*pi*t/T); % Input
>>Px=sum(x.^2)/length(x); % Input power
>>SNR= -10; % in dB
>>Py=Px/10^(SNR/10);
>>n=sqrt(Py)*randn(1,length(t));% generate white noise
>>y=x+n; % Corrupted input
>>ACF_x=normalize(xcorr(x)); % Normalizing the peak to 1
>>ACF_n=normalize(xcorr(n));
>>ACF_y=normalize(xcorr(y));
>>ACF_y(length(x))=.4*max(ACF_y); %You can enable this line for better
%understanding. Scaling value should be decreased for lower SNR
>>figure(1)
>>subplot(211),plot(t,x)
>>subplot(212),plot(t,n)
>>figure(2)
>>subplot(221),plot(tstep*(1:length(ACF_x)),ACF_x)% showing ACF w.r.t. time
>>subplot(222),plot(tstep*(1:length(ACF_y)),ACF_y)% showing ACF w.r.t. time
% hold on

function R=normalize(x)
R=x/max(x);
```

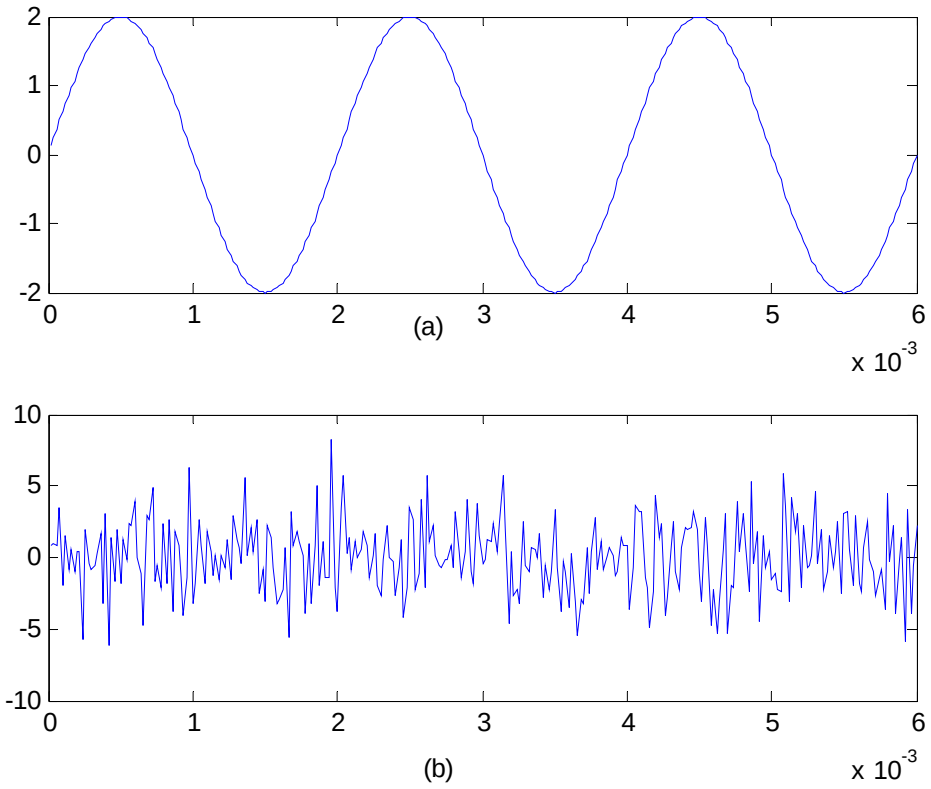


Figure 9: (a) Input periodic wave (b) Input corrupted by AWGN for -5dB SNR

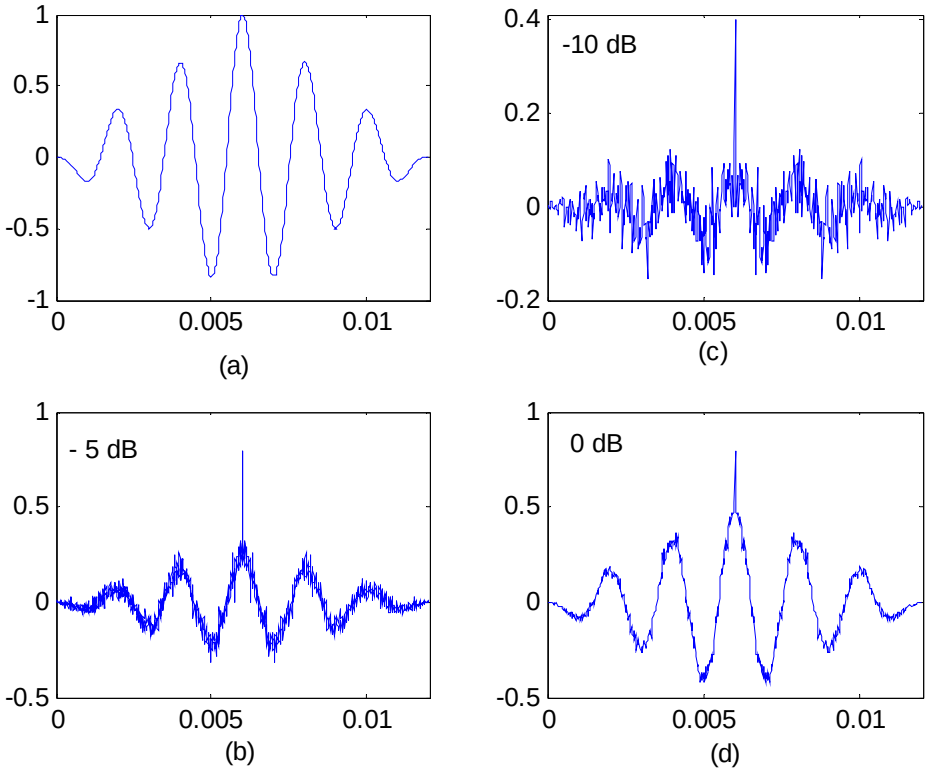


Figure 10: (a) ACF of input (b,c,d) ACF of noisy sequence.

Lab Exercise D.2

Derive a technique to obtain the original periodic signal if you have ACF of the noisy sequence. Test your technique for the cases displayed above. If the signal has a phase shift, could you recover it from ACF? Explain.

2. Estimation of impulse response:

Output of a relaxed system is given by y(n)=h(n) * x(n). If we cross-correlate the output of the system with a noisy input with normal pdf , then the cross-correlation

$$R_{yr}(k) = y(n) * r(-n) = [r(n) * h(n)] * r(-n) = r(n) * r(-n) * h(n) = R_{rr}(k) * h(n) \qquad (12)$$

where r(n) is a white noise input, $R_{rr}(k)$ is the auto-correlation of the noise input. As the auto-correlation of white noise sequence is like impulse sequence (In other words, the input noise has a much greater bandwidth than that of the system).we can write

$$R_{yr}(k) \approx h(n) \qquad (13)$$

Therefore, we say that if we correlate the white noise input with the output of the system, we will have an approximation of the impulse response of the system (See Figure 11).

Consider a system:

$$y(n) + 0.6y(n-1) = x(n)$$

```
%estimating system impulse response using cross-correlation
>>N=500;
>>nr=0:499;
>>ny=nr;
>>r=randn(1,N);
>>y=zeros(size(r));
>>for n=2:500
        y(n)=r(n)-0.6*y(n-1);
   end
>>rr=fliplr(r);
>>nrr=-fliplr(nr);
>>Ryr=conv(y,rr); % Calculating correlation
>>kmin=ny(1)+nrr(1);
>>kmax=ny(length(ny))+nrr(length(nrr));
>>k=kmin:kmax; % generating index
>>subplot(211),stem(k,Ryr/Ryr(N))
>>title('Approximate Impulse Response');
>>num=[1 0];
>>den=[1 0.5];
```

```
>>n=0:499;
>>x=zeros(size(n));
>>x(1)=1;
>>yy=filter(num,den,x);
>>subplot(212),stem(n,yy)
>>title('Actual Impulse Response')
```

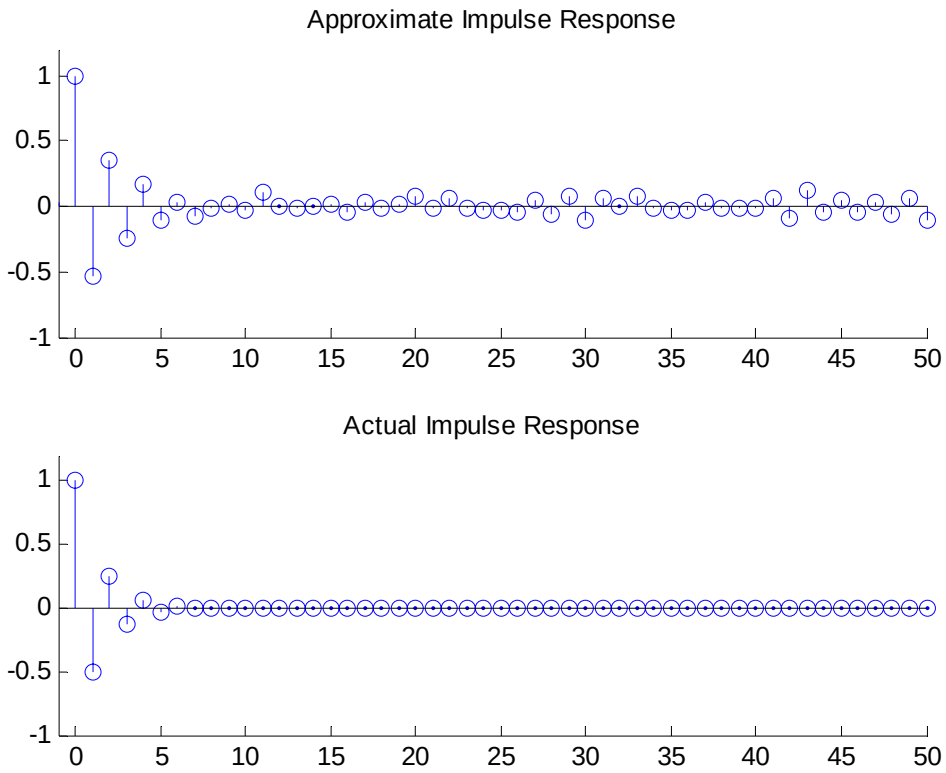


Figure 11: Actual and Estimated Impulse Response.

End of Experiment Exercises:

1. Write a general program to compute CCF of two DT signals. Signals can have different time scale. DO NOT use conv() or xcorr() function. Say two inputs are

x(n)={ ….0,1,1,3,5,7,2,0….} and y(n)={ ….0,5,5,5,3,3,1,1,0….}

You can take insight from (8).This problem is just for proper understanding about what is happening inside conv() and xcorr() functions.

2. Detection of signals in noise by auto-correlation

Consider a radar transmitting a short tone burst of EM energy(Figure 12-a) and a weak echo from a distant target(Figure 12-b).In the absence of noise weak echo can be amplified and there is no problem detecting it. If there is background of noise whose amplitude exceeds that of the echo, the echo will be masked and not detectable((Figure 12-c),echo+noise).As we know noise is suppressed in ACF domain, then write a general program to detect the echo using correlation (Figure 12-d). If the tone burst has a gradual increase in frequency, what will happen? Is it advantageous for detection? Explain with necessary simulated figures.

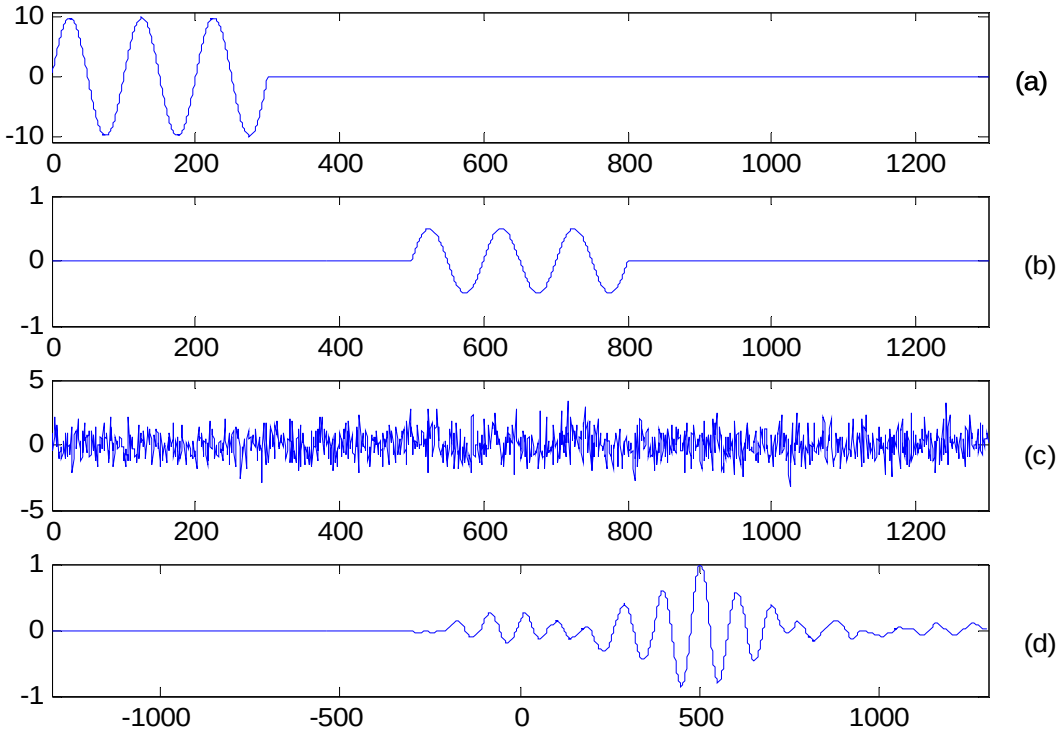


Figure 12:(a) Transmitted tone burst (b)Received weak echo(c)Received echo buried into background noise(d)CCF between (a) and (c) to locate a weak echo. It shows that after 500 units delay an echo arrives (location of the peak).

3. Detection of a transmitted sequence

Let in the transmitter, to transmit zero (**0**) we send $x_0(n)$ for $0 \le n \le$ L-1 and to transmit one (**1**) we send $x_1(n)$ for $0 \le n \le$ L-1 where $x_1(n) = - x_0(n)$. The signal received by the receiver

$$y(n) = x_i(n) + w(n) \quad i = 0,1 \text{ and } 0 \le n \le \text{L-1} \tag{14}$$

$w(n)$ is additive white noise.

Present a technique to detect the sequence transmitted from $y(n)$.Assume that particular receiver knows $x_0(n)$ and $x_1(n)$.Write a general MATLAB program for this purpose.

4. Signal smoothing by a moving average (MA) system:

From (6) if we set $\{a_k\}=0$ for $k = 1,\ldots,N$ then (6) turns into

$$y(n) = \sum_{k=0}^{M} b_k x(n - k) \quad (15)$$

which is a non-recursive linear time-invariant system. This system takes most recent M+1 points and add them after weighting. This type of system is called moving average (MA) system.

Simple working MA system can be expressed as

$$y(n) = \frac{1}{M} \sum_{k=0}^{M-1} x(n - k). \quad (16)$$

Such a system is often used in smoothing random variations in data.

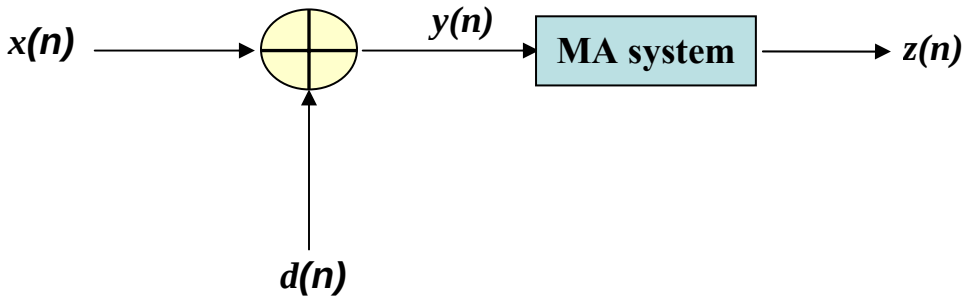


Figure 13: Signal smoothing by MA system

Consider, $x(n) = 2n(0.9)^n$. Take 50 point signal.
Noise, d = rand(1,50)-0.5; % to make noise bipolar
Take M=3.Write a sample program for the whole system shown (See Figure 14). Plot the M versus normalized error curve.

Normalized error, $e(n) = \left| \frac{x(n) - z(n)}{x(n)} \right|$

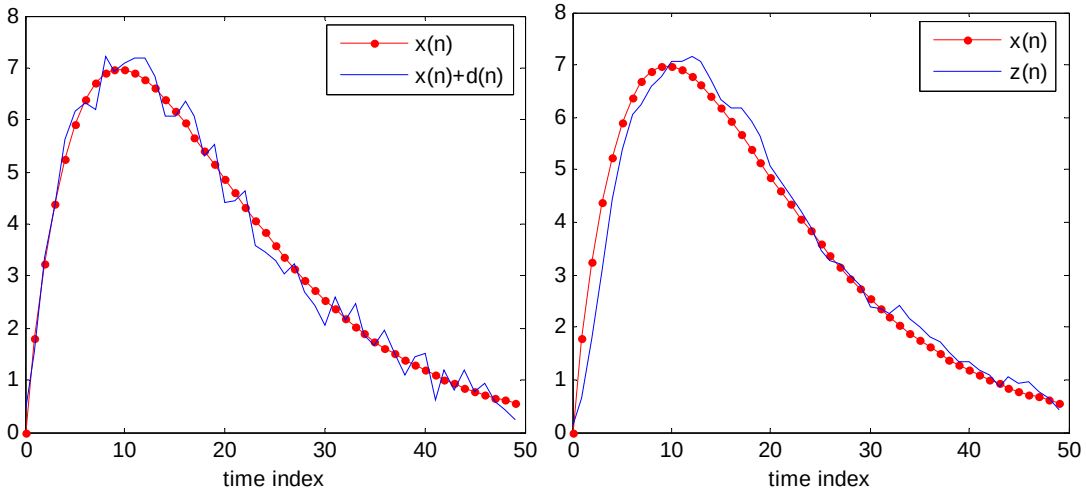
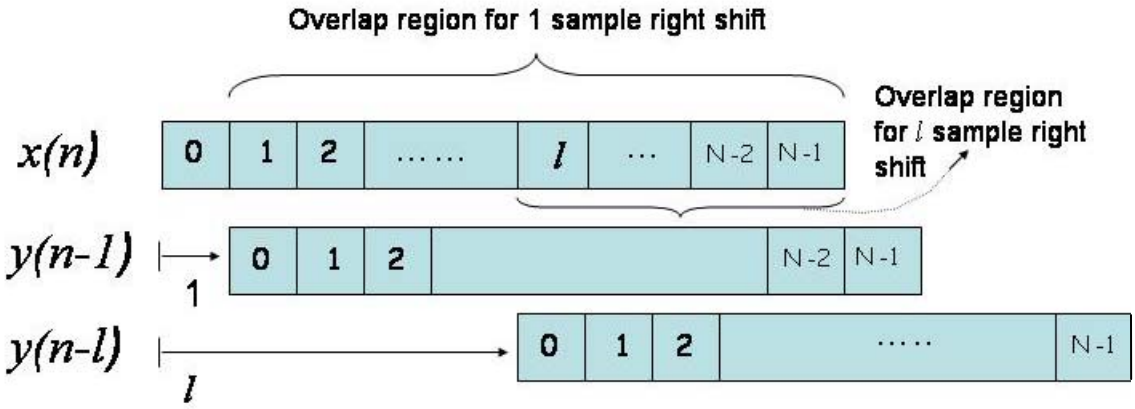


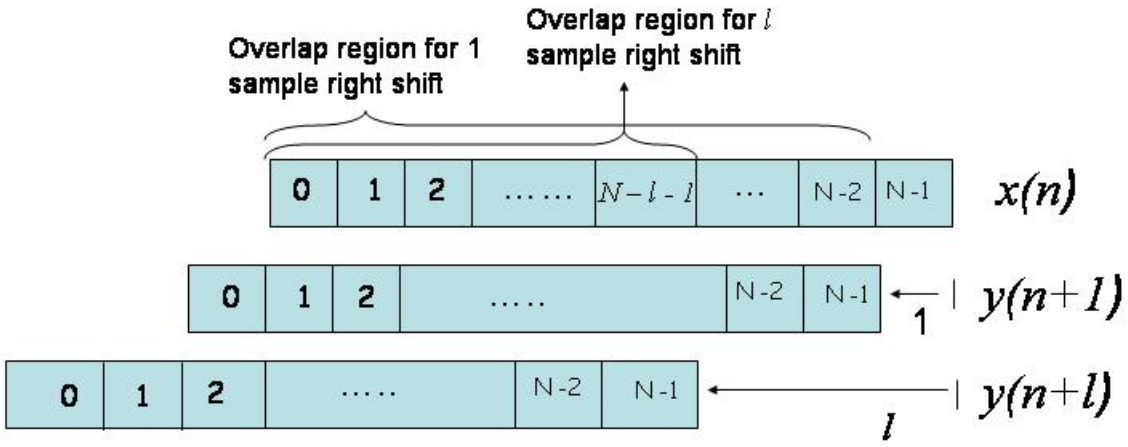
Figure 14: (left) *x(n)*=original sequence and *y(n)=x(n)+d(n),*corrupted sequence. (right) | *z(n)* ,output of the MA system and *x(n)*

Appendix :



$$r_{xy}(l) = \sum_{n=i}^{N-|k|-1} x(n)y(n-l) \qquad \text{where } i = l, k = 0 \text{ for } l \geq 0$$

(a)



$$r_{xy}(l) = \sum_{n=i}^{N-|k|-1} x(n)y(n-l) \qquad \text{where } i=0, k=l \text{ for } l<0$$

(b)

Figure 15: (a , b)illustrates cross-correlation of two finite N point sequences (equation 8).

References:

1) Proakis & Manolakis,"Digital Signal Processing:Principles,Algorithms and Applications.",Chapter 2, 3rd Edition , Prentice Hall Ltd.
2) Mitra," Digital Signal Processing: A Computer Based Approach" Chapter 2, Edition 1998, Tata McGraw -Hill Co. Ltd.
3) Denbigh, "System Analysis and Signal Processing" Chapter (2,16), Edition 1998, Addison-Wesley.
4) Elali, "Discrete Systems and Digital Signal Processing with MATLAB® " Chapter 2,Edition 2004, CRC Press
5) Ingle & Proakis, "Digital Signal Processing using *MATLAB*® " Chapter 2,Edition 2000 Thomson-Brooks/Cole Co Ltd.

Laboratory manual for this experiment is prepared by

TOUFIQUL ISLAM
Lecturer, Dept of EEE, BUET

and supervised by

Dr. Md. KAMRUL HASAN
Professor, Dept of EEE, BUET

June 9, 2007

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING,
DHAKA-1000, BANGLADESH.

COURSE NO.: EEE 312

Experiment No. - 3

Z-transform and Its Application

The Z-transform plays the same role in the analysis of discrete time signals and LTI systems as the Laplace transform does in the analysis of continuous-time signals and LTI systems. The use of z-transform techniques permits simple algebraic manipulations. The Z-transform has become an important tool in the analysis and design of digital filters. In this experiment, different features of Z-transform have been discussed through various examples and pre-lab work exercises. At the end of this experiment, one will be able to perform Z transformation, inverse Z transform, system analysis through stability and causality.

Pre Lab Concerns:

- **Read this lab description carefully before coming to the laboratory class, so that you will know what is required.**
- Try to follow the lecture notes of EEE 311.
- **Try to practice the Pre Lab exercises before coming to class.**
- **The problems that will be given in the class may differ from the given ones but they will certainly cover the syllabus.**
- You will be able to solve the problems in lab if you do practice before coming to class.
- **Do not bring any prepared MATLAB code in the lab with any portable device.**

Used Functions:

ztrans(), iztrans(), conv(), residuez(), impz(), filter(), zplane(),zp2sos(),tf2zp()

&

SPTOOL

Part – A

Z - transform

The Z-transform of a discrete time signal $x(n)$ is –

$$X(z) = \sum_{n=-\infty}^{n=\infty} x(n) z^{-n}$$

where, z is a complex variable. Since z-transform is an infinite series, it exists only for those values of z for which this series converges. The region of convergence (ROC) of X(z) is the set of all values of z for which X(z) attains a finite value.

Example:

x(n) is a finite duration signal like -

$$x(n) = \{1,2,5,7,0,1\}.$$
$$\uparrow$$

Its z transformed signal, $X(z) = 1 + 2z^{-1} + 5z^{-2} + 7z^{-3} + z^{-5}$, its ROC is entire Z-plane except z=0.

For $x(n) = \{1,2,5,7,0,1\}$, its z transformed signal is $X(z) = z^2 + 2z + 5 + 7z^{-1} + z^{-3}$, here its ROC is entire Z-plane except z=0 and z=∞.

Z transform of different functions:

Sequence	Transform	ROC
$\delta[n]$	1	all z
u[n]	z/(z-1)	|z|>1
-u[-n-1]	z/(z-1)	|z|<1
δ [n-m]	z^{-m}	all z except 0 if m>0 or if m<0
$a^n u[n]$	z/(z-a)	|z|>|a|
$-a^n u[-n-1]$	z/(z-a)	|z|<|a|
$na^n u[n]$	$az/(z-a)^2$	|z|>|a|
$-na^n u[-n-1]$	$az/(z-a)^2$	|z|<|a|
$[\cos\omega_0 n]u[n]$	$(z^2-[\cos\omega_0]z)/(z^2-[2\cos\omega_0]z+1)$	|z|>1
$[\sin\omega_0 n]u[n]$	$[\sin\omega_0]z)/(z^2-[2\cos\omega_0]z+1)$	|z|>1
$[r^n\cos\omega_0 n]u[n]$	$(z^2-[r\cos\omega_0]z)/(z^2-[2r\cos\omega_0]z+r^2)$	|z|>r
$[r^n\sin\omega_0 n]u[n]$	$[r\sin\omega_0]z)/(z^2-[2r\cos\omega_0]z+r^2)$	|z|>r
$a^n u[n] - a^n u[n-N]$	$(z^N-a^N)/z^{N-1}(z-a)$	|z|>0

MATLAB representation:

For representing a signal like this $X(z) = 1 + 2z^{-1} + 5z^{-2} + 7z^{-3} + z^{-5}$ in MATLAB, we just have to put the coefficients of the polynomial like -

x=[1,2,5,7,0,1];

For converting the time domain signal into equivalent Z domain **ztrans()** function can be used which is part of the symbolic toolbox. It evaluates signals of the form $x[n]u[n]$, i.e. for non-negative values of n.

Example

To find a Z transform of a function, for example, $a^n u[n]$, we have to write following codes –

```
>> syms a n f;
>> f=a^n;
>> ztrans(f)
```

Note that there is also the **iztrans()** function which performs inverse Z transform.

Pre-Lab Exercise – A.1:

Find the Z transform for the signals listed in the above table and verify with the given results.

Properties of Z transform:

Properties

Uniqueness	$\{x_{(n)}\} = \{y_{(n)}\} \Leftrightarrow X(z) = Y(z)$
Homogeneity	$\mathcal{Z}(K\{x_{(n)}\}) = KX(z)$ $\mathcal{Z}^{-1}(KX(z)) = K\{x_{(n)}\}$
Additivity	$\mathcal{Z}(\{x_{(n)}\} + \{y_{(n)}\}) = X(z) + Y(z)$ $\mathcal{Z}^{-1}(X(z) + Y(z)) = \{x_{(n)}\} + \{y_{(n)}\}$
Shifting	$\mathcal{Z}\{x_{(n-\mu)}\} = z^{-\mu}X(z)$ $\mathcal{Z}^{-1}z^{-\mu}X(z) = \{x_{(n-\mu)}\}$
Convolution	$\mathcal{Z}\{x_{(n)}\} * \{h_{(n)}\} = \mathcal{Z}\{h_{(n)}\} * \{x_{(n)}\} = X(z)H(z)$ $\mathcal{Z}^{-1}(X(z)H(z)) = \{x_{(n)}\} * \{h_{(n)}\} = \{h_{(n)}\} * \{x_{(n)}\}$

Among different properties of z-transform, convolution property is an important one. It removes the complexity of doing convolution, because this property transforms the time domain convolution into a multiplication of two functions.

Pre-Lab Exercise – A.2:

Let $X_1(z) = 2 + 3z^{-1} + 4z^{-2}$ and $X_2(z) = 3 + 4z^{-1} + 5z^{-2} + 6z^{-3}$. Determine $X_3(z) = X_1(z)X_2(z)$. Use **conv()** function to obtain the result and verify the result theoretically.

Part – B

Inverse Z-transform

The inverse Z-transform is the conversion of Z-domain signal into time domain signal. The inversion can be done by Cauchy's Integral theorem, long division process, partial fraction expansion etc.

I) Partial fraction expansion:

Suppose a function X(z) is given in Z-domain.

$$X(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots\cdots + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots\cdots + a_N z^{-N}} = \frac{B(z)}{A(z)}$$

To convert it in the time domain, it is expressed as

$$X(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots\cdots + b_{N-1} z^{-(N-1)}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots\cdots + a_N z^{-N}} + \sum_{k=0}^{M-N} C_k z^{-k} \text{ , where, } M \geq N$$

Then performing partial fraction expansion on the proper rational part of X(z) to obtain

$$X(z) = \sum_{k=1}^{N} \frac{R_k}{1 - p_k z^{-1}} + \sum_{k=0}^{M-N} C_k z^{-k}$$

Now the time domain signal x(n) is –

$$x(n) = \sum_{k=1}^{N} R_k Z^{-1}\left[\frac{1}{1 - p_k z^{-1}}\right] + \sum_{k=0}^{M-N} C_k \delta(n-k)$$

MATLAB representation:

A MATLAB function **residuez()** is available to compute the residue part and the direct terms of a rational part in z^{-1}. [R, p, C] = residuez(b,a) finds the residues, poles and direct terms of X(z) in which two polynomial B(z) and A(z) are given in two vectors b and a.

Pre-Lab Exercise – B.1:

1. A function X(z) is given below.

$$X(z) = \frac{z}{3z^2 - 4z + 1}$$

Find the residue, poles and direct terms of it from the function and hence determine the inverse of X(z) manually. Verify your result using **iztrans().**

II) Long Division:

For causal sequences, the Z-transform X(z) can be expanded into power series in z^{-1}. In series expansion, the coefficient multiplying by z^{-n} is then the n[th] sample x[n]. For a rational X(z), a convenient way to determine the power series is to express the numerator and the denominator as polynomials in z^{-1}, and then obtain the power series expansion by long division.

MATLAB representation:

By using **impz()** function we can get the sampled values in time domain from an analogous Z-domain function. Same process can be done by using a **filter()** function, where the input will be an impulse function.

Pre-Lab Exercise – B.2:

1. The transfer function of a causal system is given by

$$H(z) = \frac{1 + 2.0z^{-1}}{1 + 0.4z^{-1} - 0.12z^{-2}}$$

Determine the first 10 coefficients of the impulse response of this system.

Part – C

Pole – zero plot & ROC for different cases

The poles are those values for which the system transfer function becomes infinite. These are the roots of the denominator of a transfer function.

The zeros are those values for which the system transfer function becomes zero. These are the roots of the numerator of a transfer function.

The pole zero plot of a system provides information about the system behavior.

MATLAB representation:

With the function **zplane()**, the pole zero plot can be obtained from which we can get the information of system behavior.

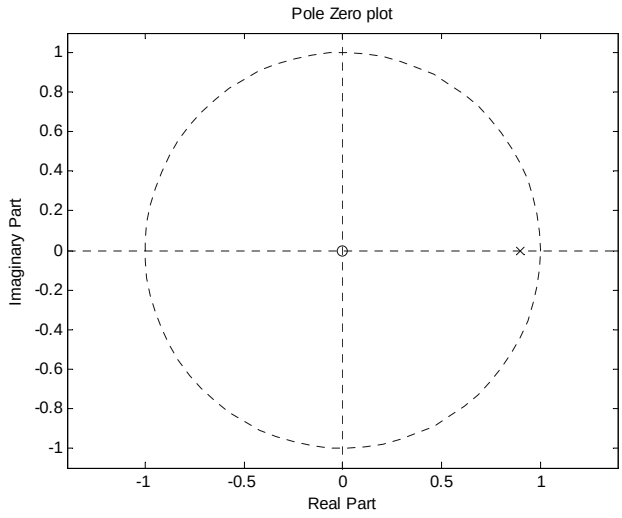
Example:

For a transfer function of $H(z) = \frac{1}{1 - 0.9z^{-1}}$,

if we write MATLAB code like –

```
>> b=[1 0];
>> a=[1 -0.9];
>> zplane(b,a)
```

the pole zero plot will be –



ROC :

- The ROC of a causal signal is the exterior of a circle of radius $|\alpha|$.
- The ROC of an anticausal signal is the interior of a circle $|\beta|$.
- The ROC of a noncausal signal is a ring (annular region) in the Z-plane.

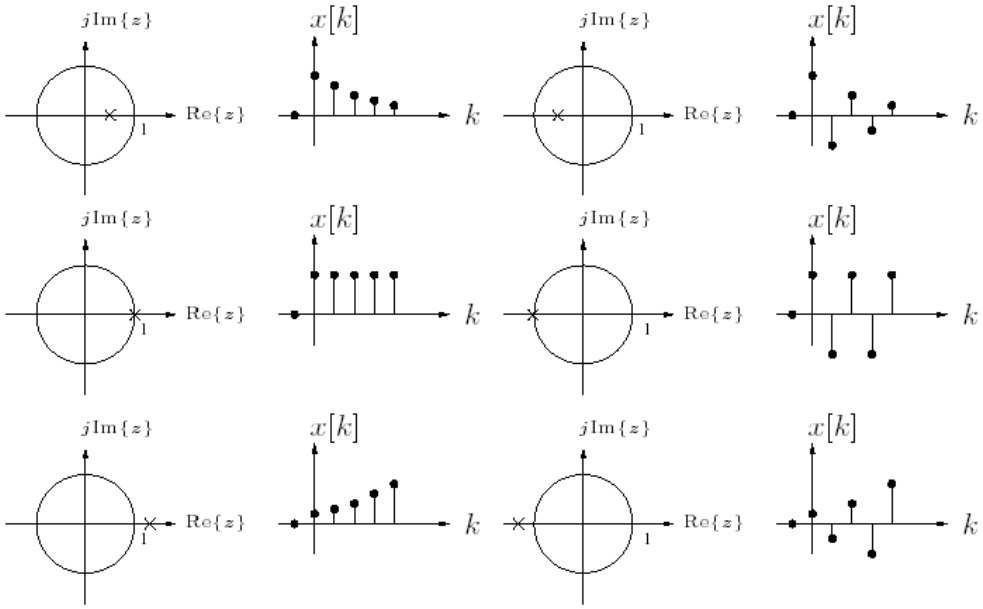
Pole location and time domain behavior for causal signals

Causal real signals with real poles or simple complex conjugate pair of poles, which are inside or on the unit circle are always bounded in amplitude. On the other hand, if they are outside the unit circle then the signals become unbounded.

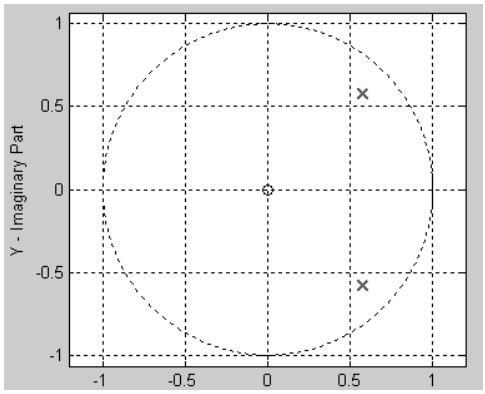
Effect of single pole:

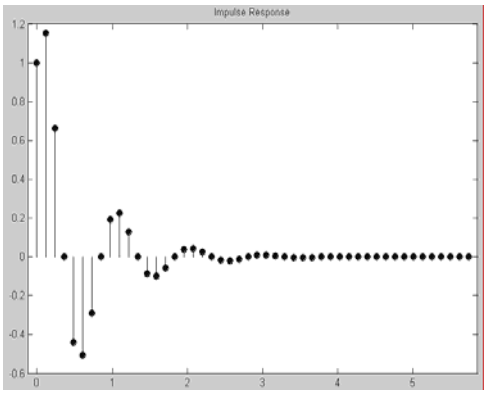
Effect of real poles:

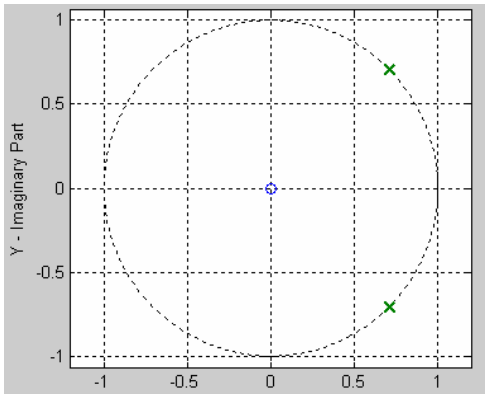
$$x[k] = a^k \epsilon[k] \circ\!\!-\!\!\bullet X(z) = \frac{z}{z-a} \qquad \text{ROC } |z| > |a|$$

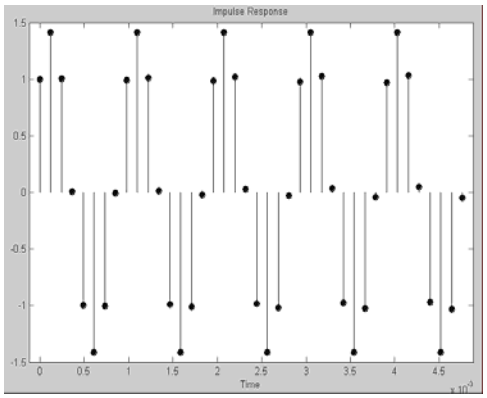


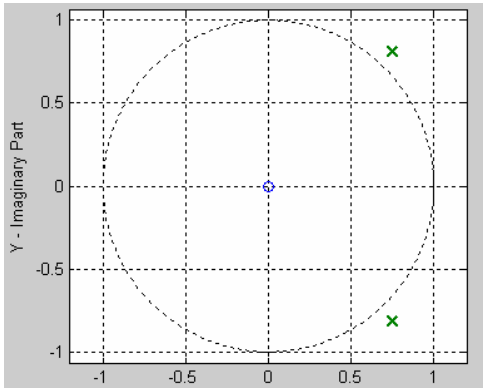
Effect of complex conjugate pair of poles:

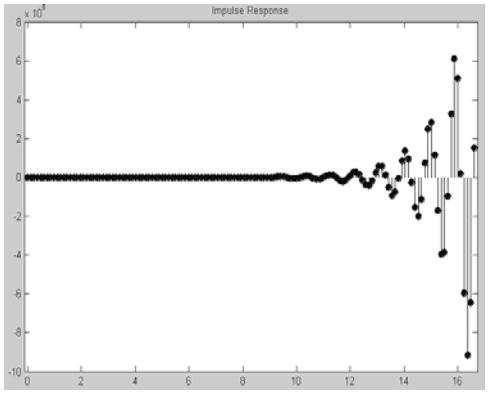






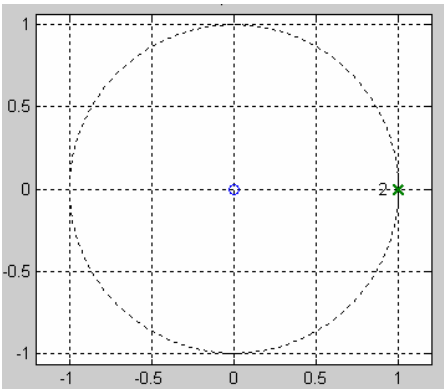


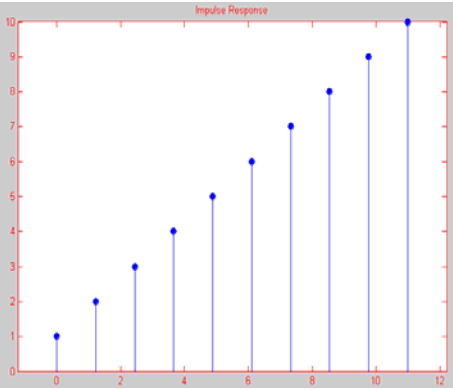




Effect of multiple poles:

Multiple real poles:



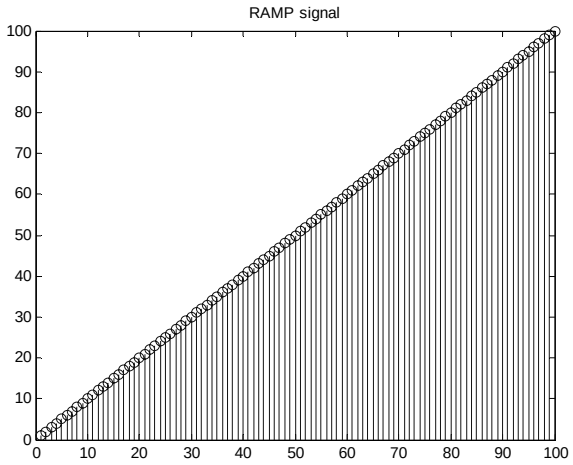


MATLAB representation:

From the above discussion it is known that different systems provide different impulse responses depending on their pole zero location. That's why different signals can be generated from those defined systems.

For a ramp signal we see that the signal should have 2 positive real poles on the unit circle. So define a system transfer function which will contain 2 positive real poles. Then obtain the impulse response of that system.

```
>> num=[1 0 0];
>> den=[1 -2 1];
>> [h,t]=impz(num,den,100);
>> stem(h)
```



Using *sptool* for system design:

On the MATLAB command prompt type **sptool** to start the filter design tool. Click **New *Design***, set ***Algorithm*** to **Pole/Zero Editor**, and delete all current poles and zeros by pressing **Delete All**. To be able to watch the frequency response simultaneously

while adding/moving/deleting poles/zeros, you should also open the **View window** from the small parent window ("Filters" specification is in the middle column). After adding a set of poles/zeros you can also observe the impulse response and step response for that particular design.

Pre-Lab Exercises – C.1:

Design a system whose impulse response gives a digital oscillation.

Pre-Lab Exercises – C.2:

Design a simple low pass and a high pass filter.

Causality and stability:

- **An LTI system is causal if and only if the ROC of a system transfer function is the exterior of a circle of radius,** $r < |\infty|$**.**
- **An LTI system is BIBO stable if and only if the ROC of the system transfer function includes the unit circle.**

So a causal LTI system is BIBO stable if and only if all the poles of H(z) are inside the unit circle.

MATLAB representation:

The inversion process can be done by any of the above described methods. The function **zp2sos()** can be used to convert the z-domain transfer function into the factored form whereas **tf2zp()** can be used to obtain poles, zeros and gain constant.

Pre-Lab Exercise – C.3:

Determine the system function and the response for a) unit step b) unit impulse input described by the difference equation- (assume a causal LTI system)

$$y(n) = y(n-1) + x(n)$$

Looking only the pole-zero plot of Y(Z), state whether the system will be stable.
Comment on ROC.

Pre-Lab Exercise – C.4:

Determine the system function and the unit sample response for a stable LTI system described by the difference equation-

$$y(n) = \frac{1}{2}y(n-1) + x(n) + \frac{1}{3}x(n-1)$$

Comment on ROC.

Pre-Lab Exercise – C.5:

An LTI system is characterized by the system transfer function

$$H(z) = \frac{3 - 4z^{-1}}{1 - 3.5z^{-1} + 1.5z^{-2}}$$

Specify the ROC of H(z) and determine h(n) for the following conditions:

(a) The system is causal, (b) The system is anticausal, & (c) The system is noncausal.

For which case, the system is stable?

Pre-Lab Exercise – C.6:

Express the following z – transform in factored form, find its poles and zeros, plot the poles, zeros and then determine its ROCs for causal cases, anticausal case and noncausal case. (Here take the coefficients as the inputs from the keyboard.)

$$G(z) = \frac{2z^4 + 16z^3 + 44z^2 + 56z + 32}{3z^4 + 3z^3 - 15z^2 + 18z - 12}$$

Pole zero cancellation:

When a Z-transform has a pole that is at the same location as a zero, the pole is cancelled by zero, consequently, the term containing that pole in the inverse z term vanishes. This pole zero cancellation can occur either in the system function itself or in the product of the system transfer function with the z-transform of the input signal.

Pre-Lab Exercise – C.7:

Determine the system function and the output response for the causal system described by the difference equation-

$$y(n) = \frac{5}{6}y(n-1) - \frac{1}{6}y(n-2) + x(n)$$

where, $$x(n) = \delta(n) - \frac{1}{3}\delta(n-1)$$

Is it possible to reconstruct the original system response from the output response?

Pre-Lab Exercise – C.8:

An LTI system is given by

$$\frac{z^2}{z^2 - 2.5z + 1}$$

Make the system stable assuming the system (a) causal and (b) anticausal.

Part – D

Higher order stability testing

For testing the stability of a higher order transfer function, a famous method called Schür Cohn stability test is used.

The transfer function of a system is –

$$H(z) = \frac{B(z)}{A(z)}$$

The denominator polynomial of the system transfer function is –

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots\cdots\cdots + a_N z^{-N}$$

By denoting this denominator with a polynomial of degree m –

$$A_m(z) = \sum_{k=0}^{m} a_m(k) z^{-k} \text{ and } a_m(0) = 1$$

The reciprocal or reverse polynomial of $B_m(z)$ are the same as those of $A_m(z)$, but in the reverse order. In the Schür Cohn stability test, to determine a set of coefficients, called reflection coefficients, $K_1, K_2, \cdots\cdots K_N$ from the polynomials $A_m(z)$. First, we set

$$A_N(z) = A(z)$$

and

$$K_N = a_N(N)$$

Then we compute the lower-degree polynomials $A_m(z)$, m = N, N-1, N-2, ……., 1, according to the recursive equation

$$A_{m-1}(z) = \frac{A_m(z) - K_m B_m(z)}{1 - K_m^2}$$

where the coefficients K_m are defined as

$$K_m = a_m(m)$$

The Schür Cohn stability test states that the polynomial $A(z)$ has all its roots inside the unit circle if and only if the coefficients K_m satisfy the condition $|K_m| < 1$ for all m = 1, 2, ….., N.

Pre-Lab Exercises – D:

Write a generalized program in MATLAB which takes the coefficients of the denominator of a system transfer function as input and tests the system stability by Schür Cohn stability test method.

Algorithm:

- Take the inputs from the user.
- Set $a_N(k) = a_k$ for $k = 1,2,\cdots\cdots, N$
- Set $K_N = a_N(N)$
- Then for $m = N, N-1, \cdots\cdots, 1$, compute $K_m = a_m(m)$ where $a_{m-1}(0) = 1$
- Then compute $a_{m-1}(k) = \dfrac{a_m(k) - K_m b_m(k)}{1 - K_m^2}$, $k = 1,2,\cdots\cdots, m-1$

  where $b_m(k) = a_m(m-k)$, $k = 0,1,\cdots\cdots, m$

References:

3) Proakis & Manolakis, **"Digital Signal Processing: Principles, Algorithms and Applications.",** Chapter 1, 3rd Edition , Prentice Hall Ltd.

4) Ingle & Proakis, " **Digital Signal Processing using *MATLAB*** ", Edition 2000 Thomson-Brooks/Cole Co Ltd.

5) Sanjit K. Mitra, "**Digital Signal Processing – A Computer based approach**" , TATA McGraw –Hill Edition.

The laboratory tutorial of this experiment is prepared by –

Imtiaz Ahmed
Lecturer, Dept. of EEE, BUET.
&
Shankhanaad Mallick
Lecturer, Dept. of EEE, BUET.

Under the supervision of –

Dr. Md. Kamrul Hasan
Professor, Dept. of EEE, BUET.

June 30, 2007

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING,
DHAKA-1000, BANGLADESH.

COURSE NO.: EEE 312

Experiment No. – 4
Phase I

Frequency domain analysis of DT signals and systems

Frequency analysis of signal involves the resolution of the signal into its frequency (sinusoidal) components. For the class of periodic signals, such decomposition is called a Fourier series. For the class of finite energy (aperiodic) signals, the decomposition is called the Fourier transform.These decompositions are extremely important in the analysis of LTI system because the response of an LTI system to a sinusoidal input is a sinusoid of same frequency but of different amplitude and phase. Furthermore, the linearity property of LTI system implies that a linear sum of sinusoidal components at the input produces a similar linear sum of sinusoidal components at the output, which differ only in the amplitudes and phases from the input sinusoids.

Pre Lab Concerns:

- **Read this lab description carefully before coming to the laboratory class, so that you will know what is required.**
- Try to follow the lecture notes of EEE 311.
- **The problems that will be given in the class may differ from the given ones but they will certainly cover the syllabus.**
- **Students must not use the function fft() unless instructed to do so. It is desired that students write codes to calculate DTFS coefficients, DTFT, and DFT coefficients from first principles.**
- You will be able to solve the problems in lab if you do practice before coming to class.
- **Do not bring any prepared MATLAB code in the lab with any portable device.**

New Functions:

freqz(), fft(), fftshift(), dftmtx()

Part – A

Introducing DTFS, DTFT, DFT

DISCRETE TIME FOURIER SERIES (DTFS)

Let us consider a periodic sequence *x(n)* with period N, that is *x(n) = x(n+N)* for all *n*. The Fourier series representation of *x(n)* consists of *N* harmonically related xponential functions

$e^{\frac{j2\pi kn}{N}}, k = 0,1,..., N-1$

And expressed as

$$x(n) = \sum_{k=0}^{N-1} c_k e^{\frac{j2\pi kn}{N}}$$

This equation is often called the discrete-time Fourier series (DTFS). Fourier co-efficients $\{c_k\}$, k=0,1,….,N-1 provide the description of *x(n)* in frequency domain.
$\{c_k\}$ can be computed as

$$c_k = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}}$$

Note that $c_{k+N} = c_k$.
That is, $\{c_k\}$ is a periodic sequence with fundamental period N. Thus the spectrum of a signal x(n), which is periodic with period with period N, is a periodic sequence with period N.

Average power can be given as

$$P_x = \sum_{k=0}^{N-1} |c_k|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2$$

The sequence $|c_k|^2$ for k=0,1,……,N-1 is the distribution of power as a function of frequency and is called the power density spectrum of the periodic signal.

If the signal x(n) is real [$x^*(n)=x(n)$],then, it can be shown that

$c_k^* = c_{-k}$. Again the following symmetry relationship holds

$|c_k| = |c_{N-k}|$ and $\angle c_k = -\angle c_{N-k}$

$\left|c_{\frac{N}{2}}\right| = \left|c_{\frac{N}{2}}\right|$ and $\angle c_{N/2} = 0$ if N is even

$\left|c_{(N-1)/2}\right| = \left|c_{(N+1)/2}\right|$ and $\angle c_{(N-1)/2} = -\angle c_{(N+1)/2}$ if N is odd

CALCULATION OF FOURIER SERIES COEFFICIENTS

Let us consider a continuous rectangular pulse sequence of 1 ms period and pulse width 0.1 ms. The signal is sampled at 100kHz and sampled discrete with *n* its index is shown in figure 1(b).

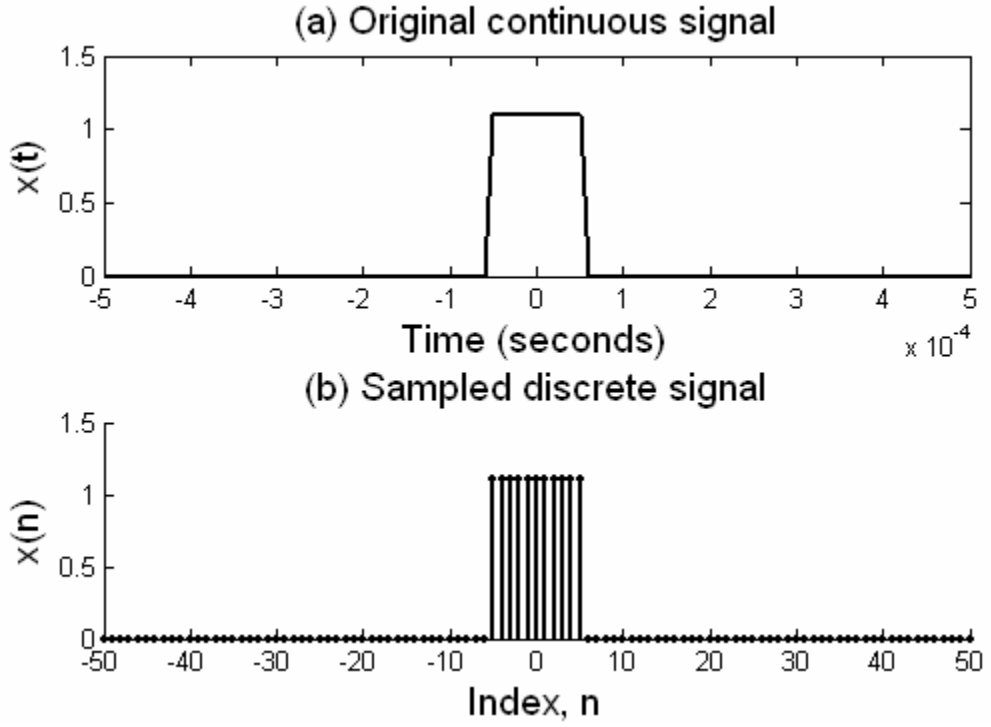


Figure 1. A rectangular pulse in continuous domain and its sampled version.

Now the following MATLAB code calculates the Fourier series coefficients from the first principle and from the Fourier series coefficients, again reconstructs the original periodic sequence.

```
%Example 1
%CALCULATION OF FOURIER SERIES COEFFICIENTS

clear all;
Fs=100e3;
dt=1/Fs;

%GENERATING THE RECTANGULAR PULSE TRAIN
T=1e-3;                    %PERIOD OF THE PULSE TRAIN
D=.1;                      %DUTY CYCLE
PW=D*T;                    %PULSE WIDTH
f=1/T;                     %ANALOG FREQUENCY
t=-T/2:dt:T/2;             %TIME INTERVAL FOR A PERIOD
```

```
n=t/dt;                          %INDEX FOR DATA POINTS IN A PERIOD
L=PW/dt;                         %DATA POINTS IN THE THE HIGH TIME
x=zeros(1,length(t));            %INITIALIZING A SINGLE RECTANGULAR PULSE
x(find(abs(n)<=L/2))=1.1;        %GENERATION OF A SINGLE RECTANGULAR PULSE
%END OF RECTANGULAR PULSE TRAIN

figure(20),  subplot(211), plot(t, x, 'k','linewidth', 2), xlabel('Time (seconds)', 'fontsize', 14), ylabel('x(t)',
'fontsize', 14), title('(a) Original continuous signal', 'fontsize', 14);
subplot(212), stem(n, x, '.k', 'linewidth', 2);xlabel('Index, n', 'fontsize', 14), ylabel('x(n)', 'fontsize', 14),
title('(b) Sampled discrete signal', 'fontsize', 14);


N=length(x);    %TOTAL NO DATA POINTS IN A PERIOD
Nc=N; ← %TOTAL NO OF COEFFICIENTS
if mod(Nc,2)==0,
   k=-Nc/2:Nc/2-1;
else
   k=-(Nc-1)/2:(Nc-1)/2;
end

c=zeros(1,length(k));%INITIALIZING FOURIER COEFFICIENTS

for i1=1:length(k),
   for i2=1:length(x),

      c(i1)=c(i1)+1/N*x(i2)*exp(-i*2*pi*k(i1)*n(i2)/N); ←
   end
end

figure(2), subplot(211), stem(k,abs(c), '.k', 'linewidth', 2); xlabel('k', 'fontsize', 14), ylabel('|c_k|',
'fontsize', 14), title('Fourier series coefficients c_k', 'fontsize', 14);
subplot(212), stem(k,angle(c)*180/pi, '.k', 'linewidth', 2);xlabel('k', 'fontsize', 14), ylabel('angle(c_k)',
'fontsize', 14)

figure(3), stem(k*f,c, '.k', 'linewidth', 2); xlabel('Frequency (Hz)', 'fontsize', 14), ylabel('c_k', 'fontsize',
14), title('Fourier series coefficients c_k', 'fontsize', 14);

%START OF RECONSTRUCTION OF SIGNAL
t_remax=T/2; ←
t_re=-t_remax:dt:t_remax;
n_re=t_re/dt;
x_re=zeros(1,length(n_re));

for i1=1:length(k),
   for i2=1:length(x_re),

      x_re(i2)=x_re(i2)+c(i1)*exp(i*2*pi*k(i1)*n_re(i2)/N); ←
   end
end
%END OF RECONSTRUCTION OF SIGNAL

figure(4),subplot(211), stem(n_re, x_re,'.k',  'linewidth',2);xlabel('n', 'fontsize', 14),
ylabel('x_{reconstructed}', 'fontsize', 14), title('Reconstructed signal', 'fontsize', 14);
subplot(212),plot(t_re, x_re,'k',0,0,  'linewidth',2);xlabel('t', 'fontsize', 14), ylabel('x_{reconstructed}',
'fontsize', 14), title('Reconstructed signal', 'fontsize', 14);
```

$$c_k = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-\frac{j2\pi kn}{N}}$$

$$x(n) = \sum_{k=0}^{N-1} c_k e^{\frac{j2\pi kn}{N}}$$

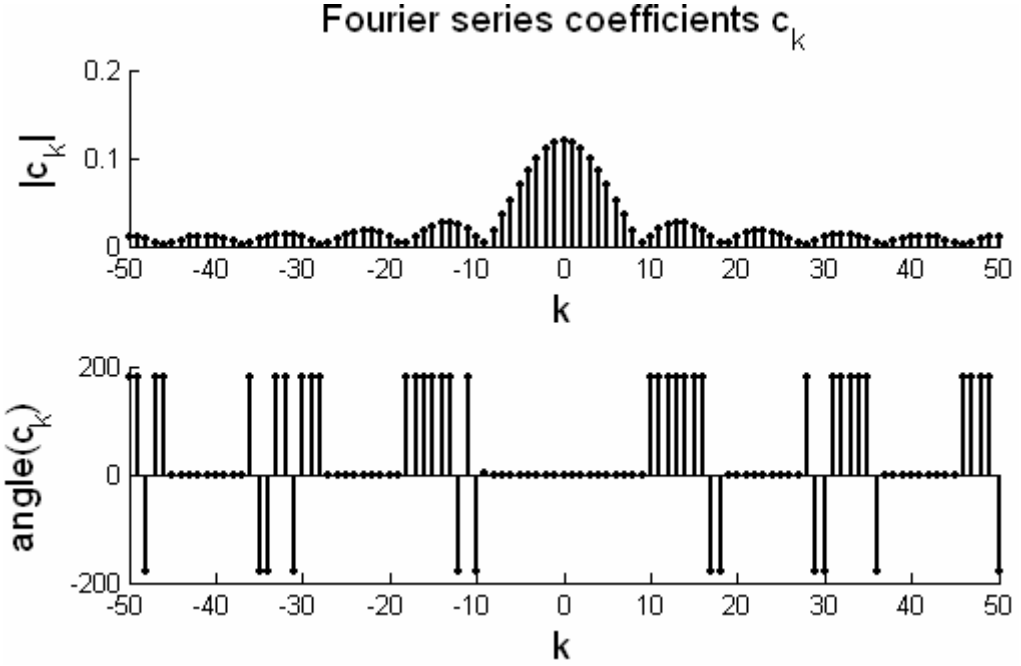


Figure 2: Magnitude and phase spectrum of the discrete time signal. Note that the phase spectrum is not skew-symmetric with respect to k=0. It is an artifact of MATLAB and θ= $+180^o$ and -180^o are in fact synonymous.

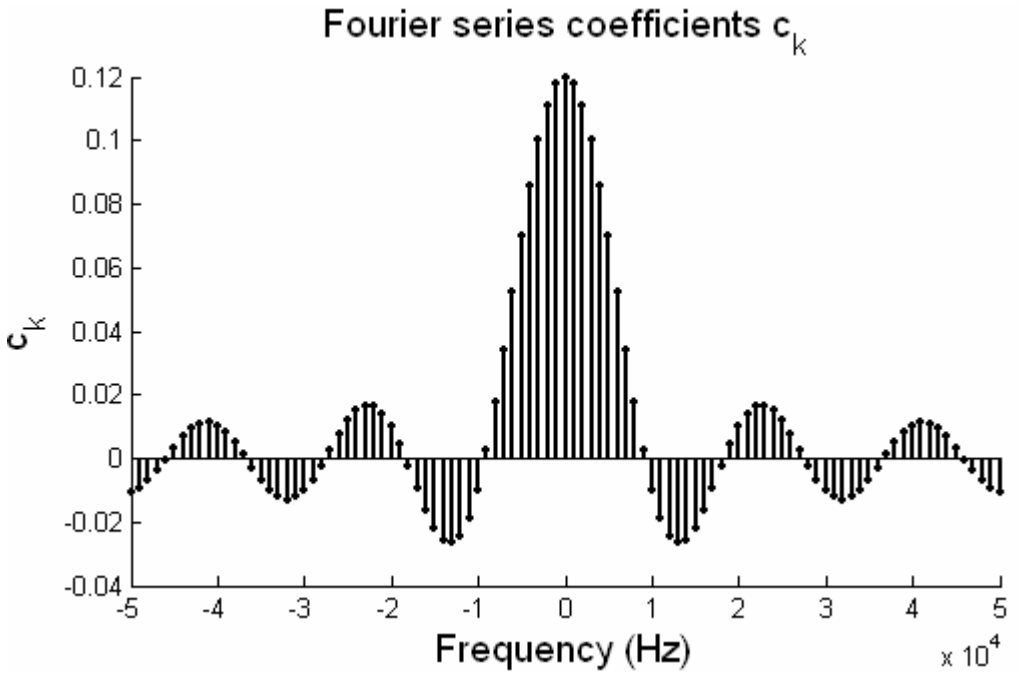


Figure 3. Fourier series coefficients of the continuous time signal and indexed with respect to the analog frequency.

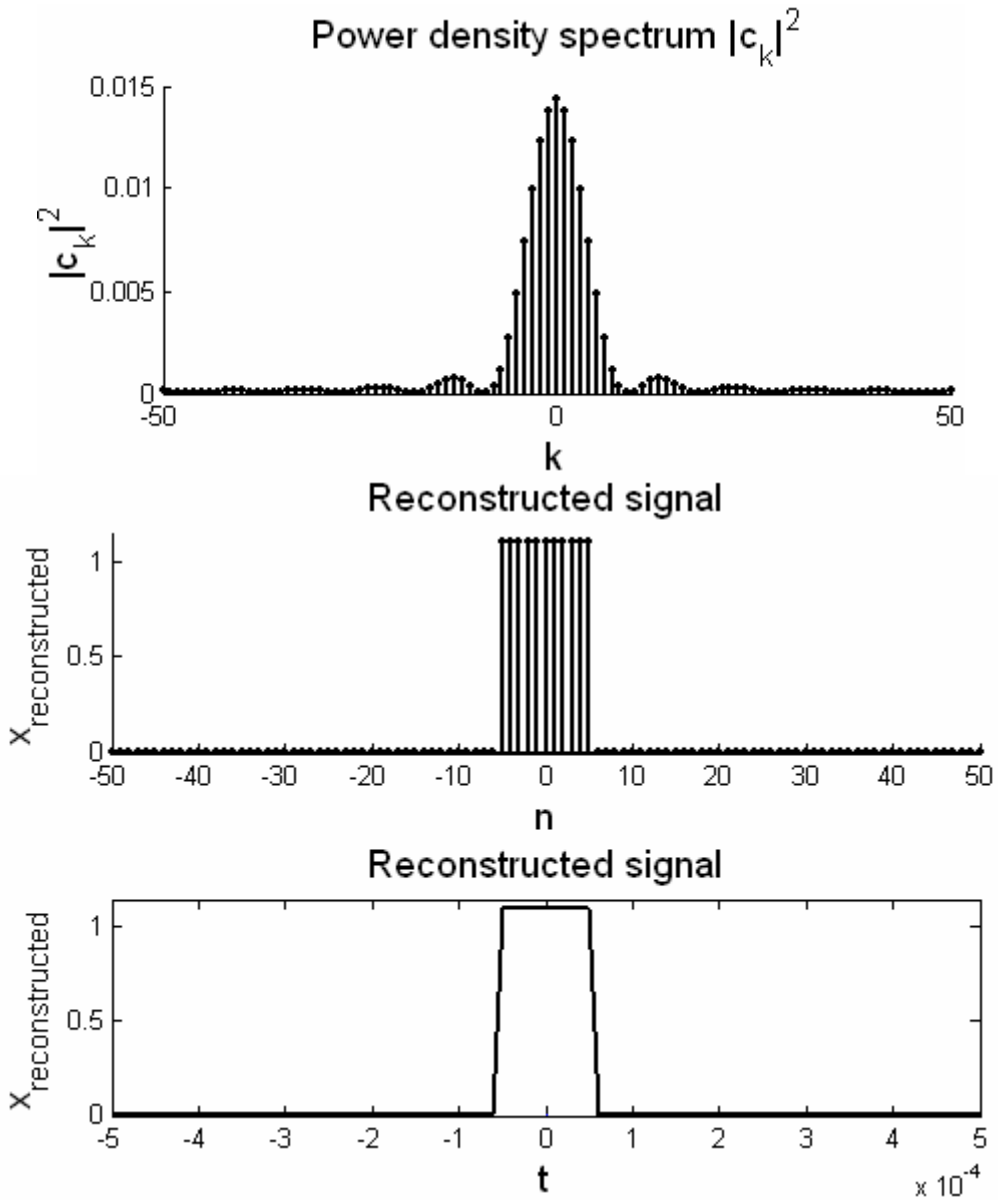


Figure 4. Reconstructed signal.

Now vary **t_remax (marked with ←in the code)** in the code and observe that a periodic repetition is obtained.

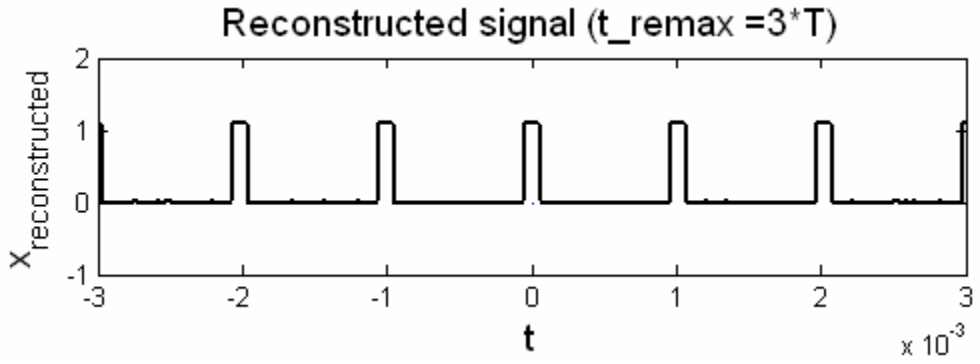


Figure 5. Periodic repetition of reconstructed signal.

Now change the total number of coefficients **Nc (marked with ←in the code)** to **3*N** and observe the periodicity of c_k with respect to n.

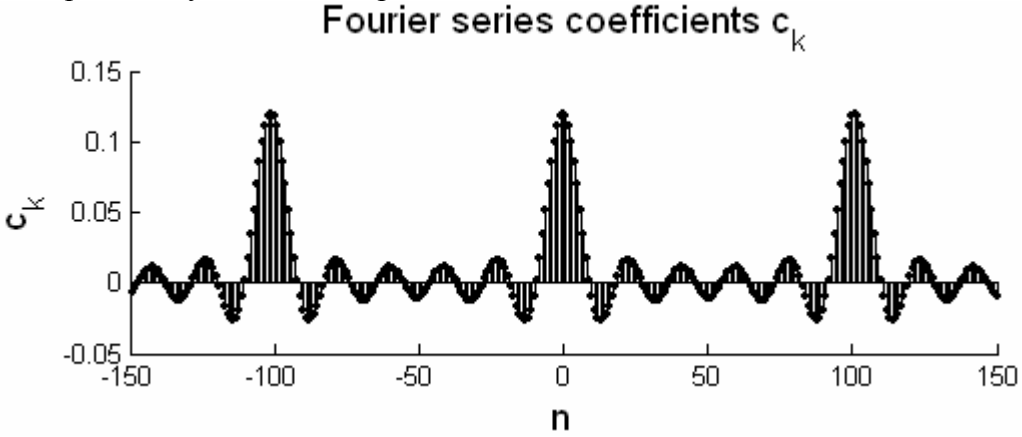


Figure 6. Periodicity of Fourier coefficients.

Now observe when we set the total number of coefficients to less than **N**.

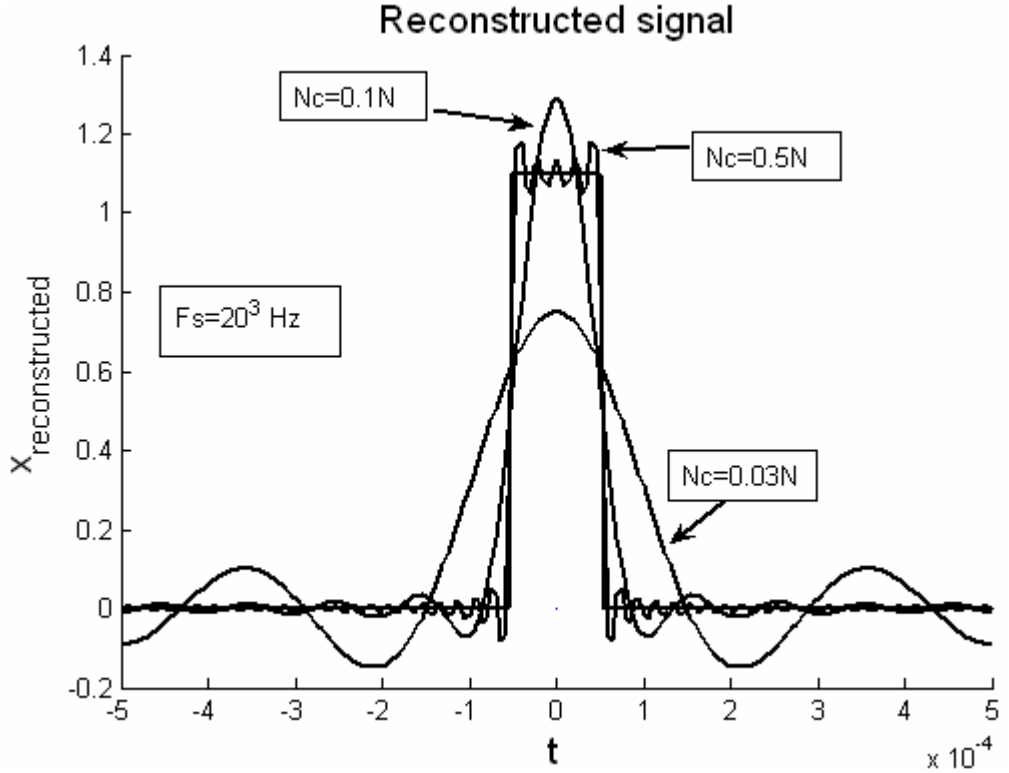


Figure 7. Effect of taking less no of FS coefficient in reconstructing the signal. Note that the higher number of coefficients we take the better the reconstructed signal matches with the original signal

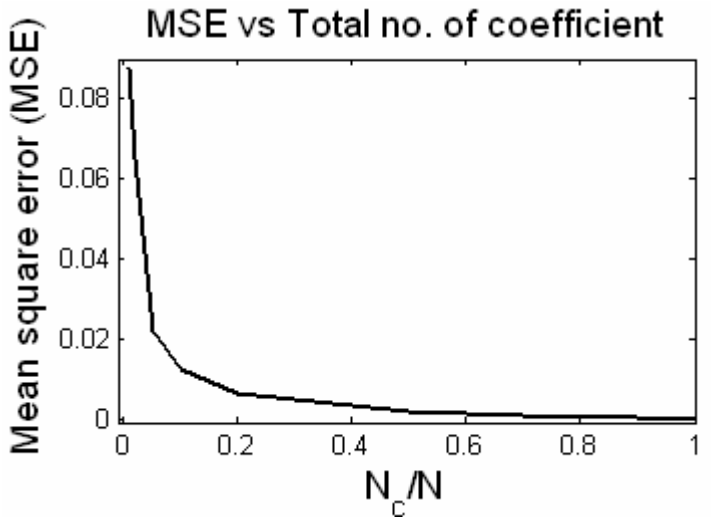
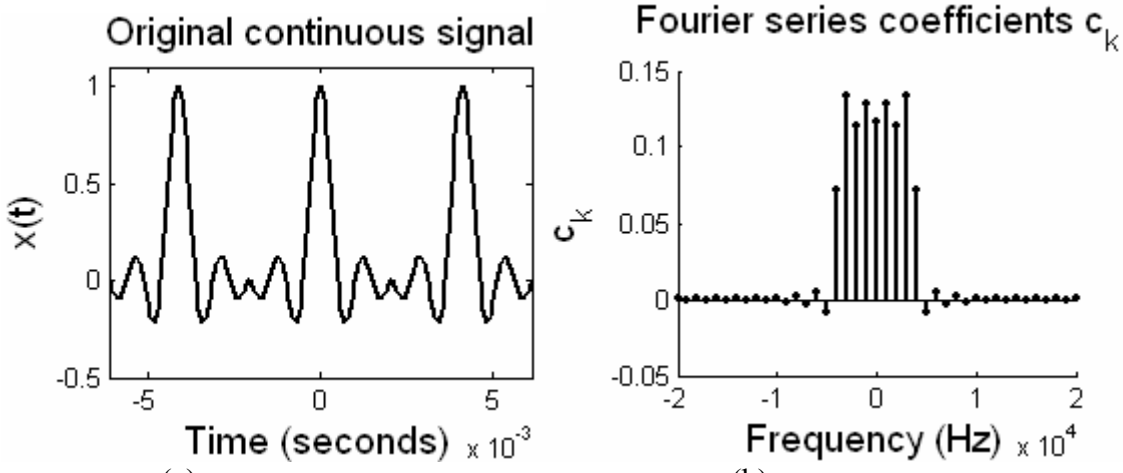


Figure 8. Change of mean square error with the increase of total number of coefficient. Note that the higher no of coefficient we take the smaller the MSE and MSE=0 when *Nc=N*.

Now generate a sinc function using a the following codes and examine its Fourier series coefficients. The results are shown in figure 5.

```
%GENERATING A SINC PULSE
T=1e-3;
f=1/T;
t=-2*T:dt:2*T;
n=t/dt;
x=sinc(2*f*t);
N=length(t);
%END OF SINC PULSE
```



(a) (b)

Figure 6. Periodic sinc pulse sequence and its Fourier series coefficients.

Exercise:

1. Note that in the example 1, the time interval is defined from –T/2 to T/2, Change the interval to 0 to T and to –T/4 to 3T/4. Do you observe any change in the magnitude spectrum? Try to explain the observation. Change the interval to –T to T. Explain the observation.
2. We have stated that $c_{k+N} = c_k$. Modify the code for example 1 and see whether you a periodic repetition of c_k.

```
N=length(x);

if mod(N,2)==0,
   k=-N:N;
else
   k=-(N-1):(N-1);
end
```

3. Take a sine wave and determine in spectrum.
4. Derive the power density spectrum for a rectangular pulse train.

DISCRETE TIME FOURIER TRANSFORM (DTFT)

If x(n) is absolutely summable that is $\sum_{-\infty}^{\infty} |x(n)| < \infty$ then its discrete time Fourier transform is given by

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} \qquad (1)$$

Now ω is a real variable between $-\infty$ to $+\infty$ but one important property of $X(\omega)$ is its periodicity in ω with period 2л. So we only need one period of $X(\omega)$ i.e. ω є [0,2л] or [-л, л].

The inverse DTFT equation, in other words the equation for reconstructing the signal from *X(ω)* is given by

$$x(n) = \frac{1}{2\pi}\int_{2\pi} X(\omega)e^{j\omega n}d\omega$$

CALCULATION OF FOURIER TRANSFORM FROM FIRST PRINCIPLE

Since the frequency is continuous variable, DTFT cannot be implemented in digital hardward in rigorous sense. By defining a fine grid of frequency in the range [-π, π]
the frequency variable is usually handled in DTFT implementation. In MATLAB we evaluate $X(\omega)$ at equidistant frequencies between [0,2л] and then interpolate using **plot**() function.
Say we want M equidistant samples between [0,2л]

Then $\omega_k = \frac{2\pi k}{M}$, k=0,1,…..,M-1 . Then (1) van be written as

$$X(\omega_k) = \sum_{n=n_1}^{n_N} e^{-j\frac{2\pi kn}{M}} x(n) \quad \text{where } k=0,1,\ldots..,M\text{-}1 \qquad (2)$$

Although matrix method may be an easier approach for DTFT in some cases, we first examine calculation DTFT of an aperiodic signal from first principles in the following example

```
%Example 2
%CALCULATION OF FOURIER TRANSFORM FROM FIRST PRINCIPLE

clear all;

%GENERATING A SINC PULSE
f_c=1/8;      %DEFINING FREQUENCY VARIABLE FOR SINC PULSE
n=-40:40;     %DEFINING THE INDEX FOR SINC PULSE
x=sinc(2*f_c*n);
%END OF SINC PULSE

figure(1), stem(n, x, '.k');xlabel('n', 'fontsize', 14), ylabel('x(n)', 'fontsize', 14), title('Discrete time signal',
'fontsize', 14);

M=101;    %NUMBER OF POINTS IN DIGITAL FREQUENCY GRID
w=linspace(-pi, pi, M);     %DEFINING THE DIGITAL FREQUENCY GRID
dw=w(2)-w(1);               %RESOLUTION OF DIGITAL FREQUENCY
X=zeros(1,M);               %INITIALIZING THE DTFT OF x(n)

for i1=1:M,
   for i2=1:length(x),
      X(i1)=X(i1)+x(i2)*exp(-i*w(i1)*n(i2));
   end
end

figure(2), plot(w,abs(X), 'k', 'linewidth', 2); xlabel('Frequency (rad/sec)', 'fontsize', 14), ylabel('X(w)',
'fontsize', 14), title('FREQUENCY SPECTRA', 'fontsize', 14);

%RECONSTRUCTION OF SIGNAL
n_re=-80:80;
x_re=zeros(1,length(n_re));    %INITIALIZING RECONSTRUCTED SIGNAL

for i1=1:M,
   for i2=1:length(x_re),
      x_re(i2)=x_re(i2)+1/(2*pi)*X(i1)*exp(-i*w(i1)*n_re(i2))*dw;
   end
end
figure(3), stem(n_re, x_re,'.k',  'linewidth',1);xlabel('t', 'fontsize', 14), ylabel('x_{reconstructed}',
'fontsize', 14), title('Reconstructed signal', 'fontsize', 14);
```

$\leftarrow \boxed{X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}}$

$\leftarrow \boxed{x(n) = \frac{1}{2\pi}\int_{2\pi} X(\omega)e^{j\omega n}d\omega}$

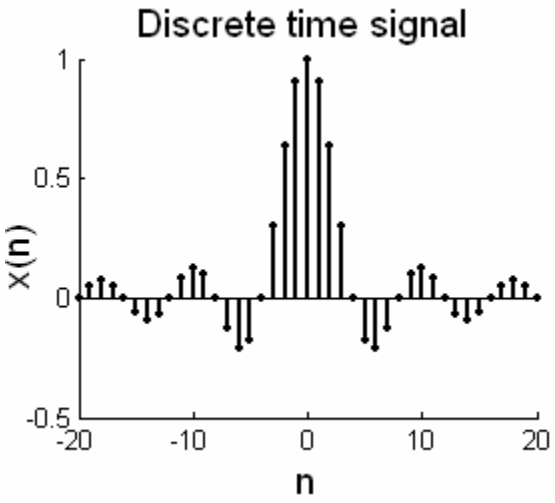


Figure 6. A sinc pulse (discrete)

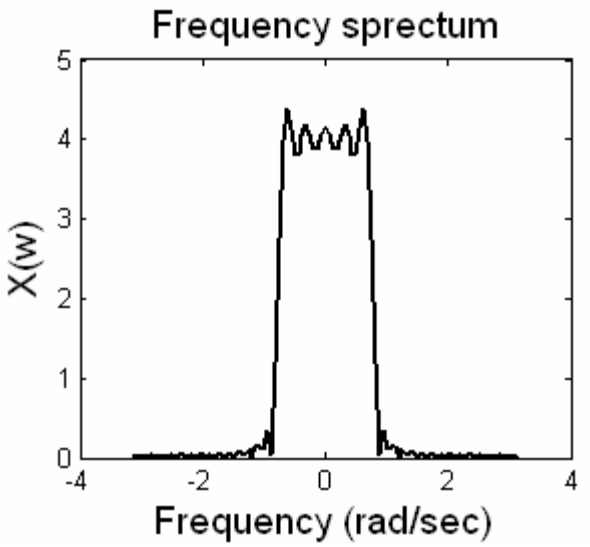


Figure 7. Magnitude spectrum of the signal

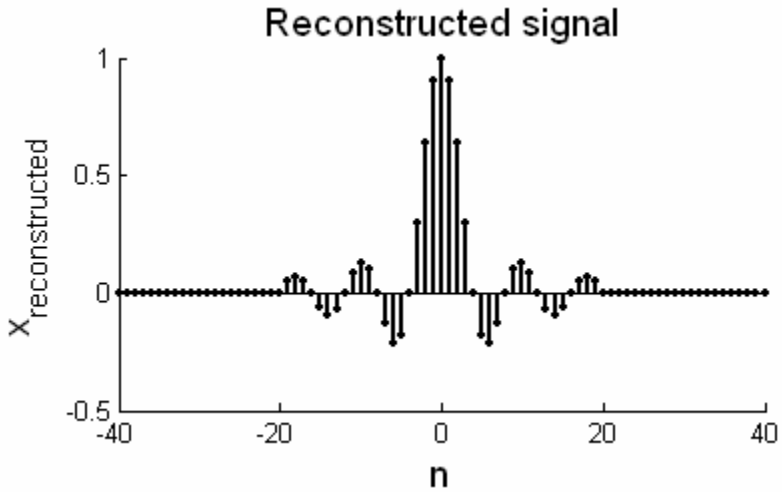
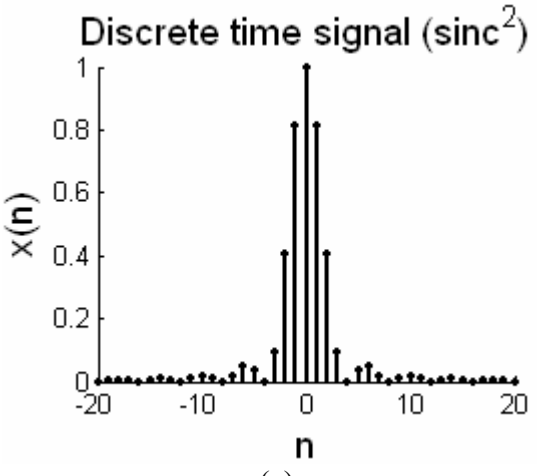
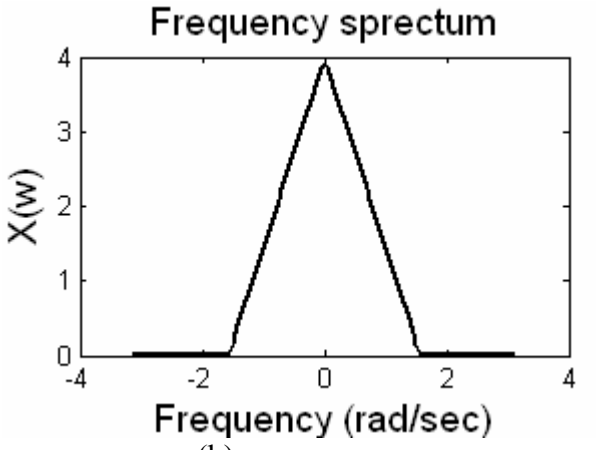


Figure 8. Sinc pulse reconstructed using inverse DTFT. Note the differences in original and reconstructed signals.

Now consider a *sinc*^{2} pulse and observe its frequency spectrum.



(a)



(b)

Figure 9. A *sinc*^{2}pulse (discrete) (a) and its frequency spectrum.

Now let us examine what happens if reconstruct the signal considering only a limited range of frequencies.

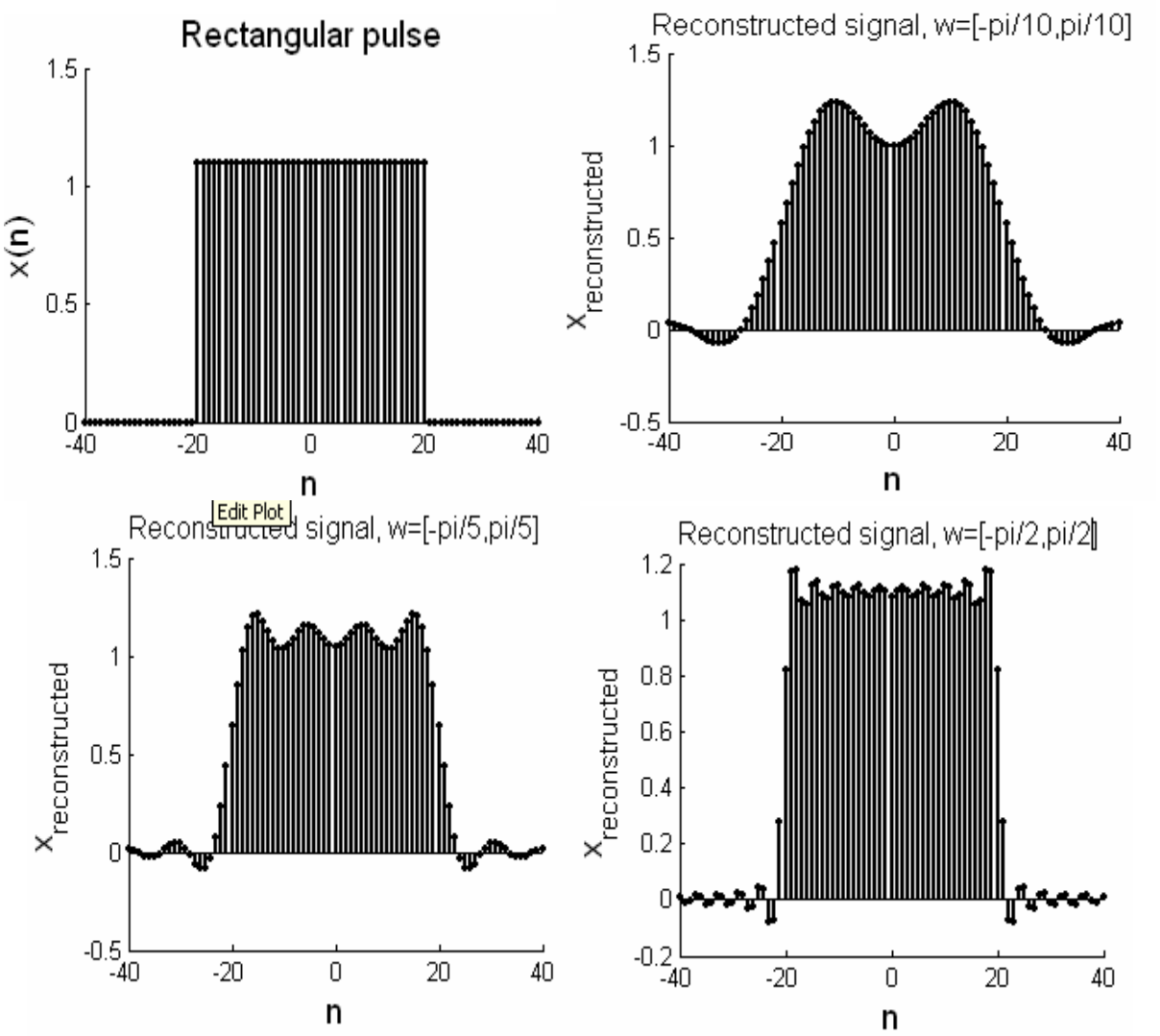


Figure 10. Effect of taking a limited number of frequencies in reconstructing the signal for DTFT.

DTFT EQUATIONS IN MATRIX FORM:

We can arrange DTFT equations in Matrix form.

Let **w** be the vector containing M, frequency points between [0,2л]
x be the data sequence having N data points
n be the time index vector

$$X = Wx^T \qquad (3)$$

where W is an M×N matrix defined as,

$$\mathbf{W}= \begin{bmatrix} e^{-j\omega(0)n(0)} & e^{-j\omega(0)n(1)} & e^{-j\omega(0)n(2)} & \dots & e^{-j\omega(0)n(N-1)} \\ e^{-j\omega(1)n(0)} & e^{-j\omega(1)n(1)} & e^{-j\omega(1)n(2)} & \dots & e^{-j\omega(1)n(N-1)} \\ e^{-j\omega(2)n(0)} & e^{-j\omega(2)n(1)} & e^{-j\omega(2)n(2)} & \dots & e^{-j\omega(2)n(N-1)} \\ . & . & . & .... & . \\ e^{-j\omega(M-1)n(0)} & e^{-j\omega(M-1)n(1)} & e^{-j\omega(M-1)n(2)} & \dots & e^{-j\omega(M-1)n(N-1)} \end{bmatrix}$$

The following example calculates the DTFT of a sequence x=[1, 3 , -9, 5, 10];

```
%Example
%MATRIX IMPLEMENTATION OF DTFT
x=[1, 3 , -9, 5, 10];
n1=-1;  %DEFINING THE INDEX OF FIRST ELEMENT OF x
n2=3;   %DEFINING THE INDEX OF THE LAST ELEMENT OF x
n=n1:n2;%INDEX OF x
M=500;  %TOTAL NUMBER OF POINTS IN THE FREQUENCY RANGE
w=(-M/2:M/2)*2*pi/M; %FREQUENCY GRID
W=exp(-j*w'*n); % Matrix formation
X=W*x';
figure(1), subplot(211),plot(w/(2*pi),abs(X),'k',  'linewidth', 2), xlabel('Digital frequency, f', 'fontsize',
14),  ylabel('|X(f)|',  'fontsize', 14), title('Magnitude Spectrum', 'fontsize', 14);
subplot(212),plot(w/(2*pi),angle(X)*180/pi, 'k', 'linewidth', 2), xlabel('Digital frequency, f', 'fontsize', 14),
ylabel('angle(X(f))',  'fontsize', 14), title('Phase Spectrum', 'fontsize', 14);
```

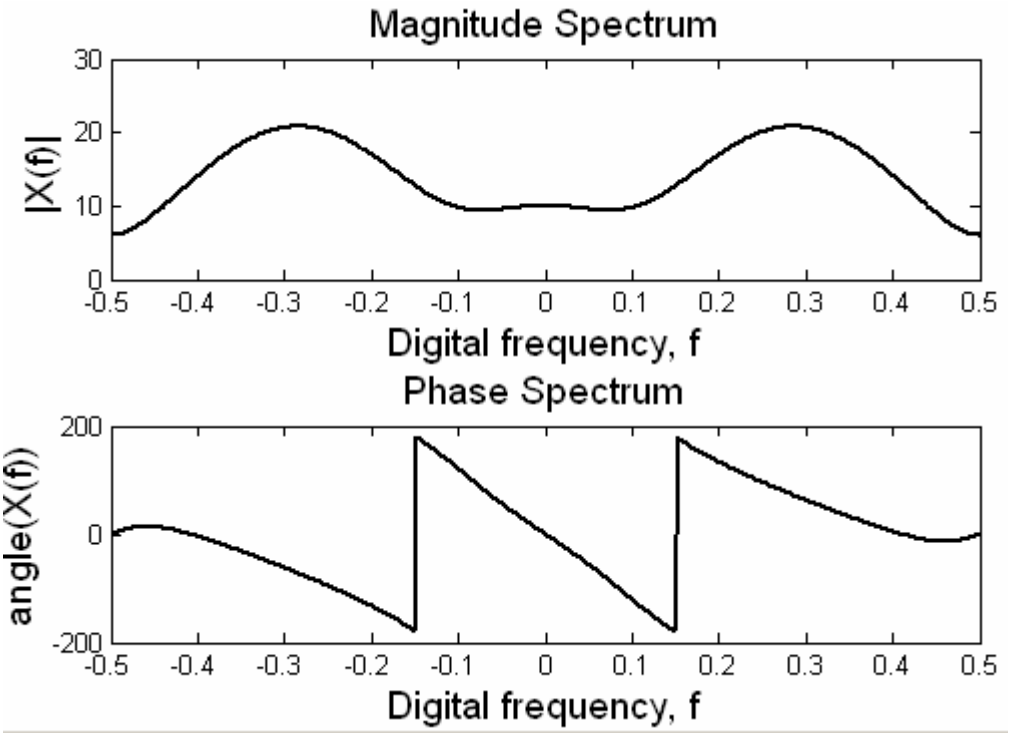


Figure 11. DTFT of sequence calculated by matrix method.

Problems (1-4 for Example 2):

1. Increase the range of the frequency grid to [0, 2π]. Observe and explain the effects.
2. Change the range of the frequency range to [-2π, 2π]. Obsevre and exaplain the effects.
3. Verify the effect of changing index (n) to -40:120, -40:80. In each case observe the reconstructed signal.
4. If you reconstruct the signal in n_re=-240:240, why do you get a repetition of the aperiodic signal?
5. Consider the continuous time signal shown in figure 12. Sample it and then find out the Fourier transform in terms of the analog frequency.
6. Verify the duality property of DTFT with rectangular pulse and sinc function.

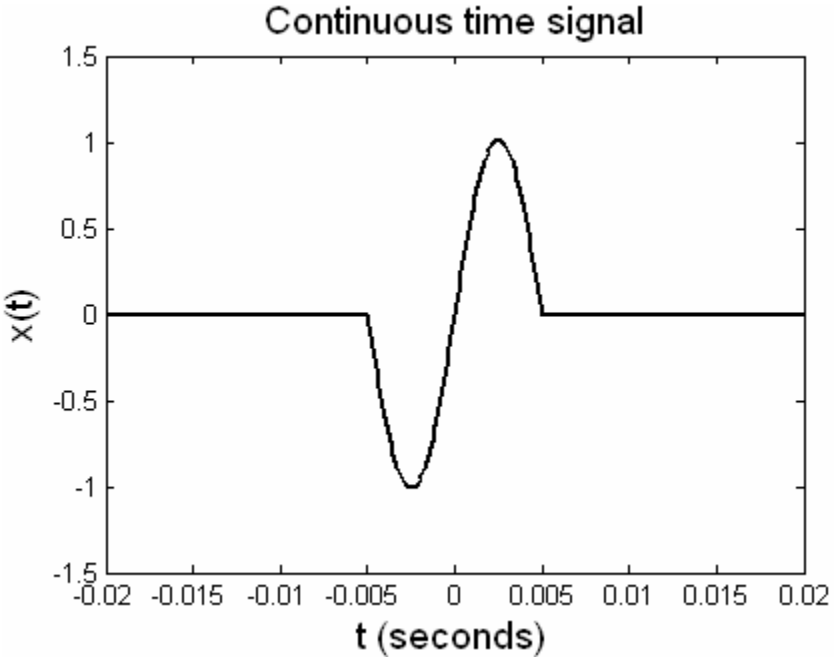


Figure 12. Figure for problem 4.

DTFT & Z-TRANSFORM:

Fourier transform can be viewed as the z-transform of the sequence evaluated on the unit circle.

$$X(z)\Big|_{z=e^{j\omega}} \equiv X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}$$

If X(z) does not converge in the region $|z| = 1$ i.e if the unit circle is not contained in the region of convergence of X(z), the Fourier transform X(ω) does not exist.

Example:

Determine frequency response H(ω) of a system characterized by $h(n)=(0.1)^n u(n)$.Plot the magnitude and phase responses.

Now, $H(z) = \dfrac{1}{1-0.1z^{-1}}$ ROC: $|z| > 0.1$ (as causal)

Clearly ROC includes the unit circle hence DTFT exists. So

$$H(\omega) = \frac{1}{1-0.1e^{-j\omega}}$$

CALCULATING THE DTFT OF DT SIGNAL/SYSTEM DEFINED IN Z-DOMAIN

MATLAB function **freqz**(num,den,w) returns the frequency response(DTFT) vector H calculated at the frequencies (in radians per sample) supplied by the vector w of the DT signal or system defined in Z-domain by the Z-transform formed by (num, den). The following example calculates the DTFT of function which is defined by its Z-tranform.

```
%Example
%RELATIONSHIP BETWEEN Z-TRANSFORM & DTFT
k=256;
w=-pi:pi/k:pi;
num=1;
den=[1 -0.1];
H=freqz(num,den,w);
figure(1), subplot(211),plot(w,abs(H),'k',  'linewidth', 2), xlabel('w (rad)', 'fontsize', 14),  ylabel('|X(w)|',
'fontsize', 14), title('Magnitude Spectrum', 'fontsize', 14);
subplot(212),plot(w,angle(H)*180/pi, 'k', 'linewidth', 2), xlabel('w (rad)', 'fontsize', 14),
ylabel('angle(X(w))',  'fontsize', 14), title('Phase Spectrum', 'fontsize', 14);
```

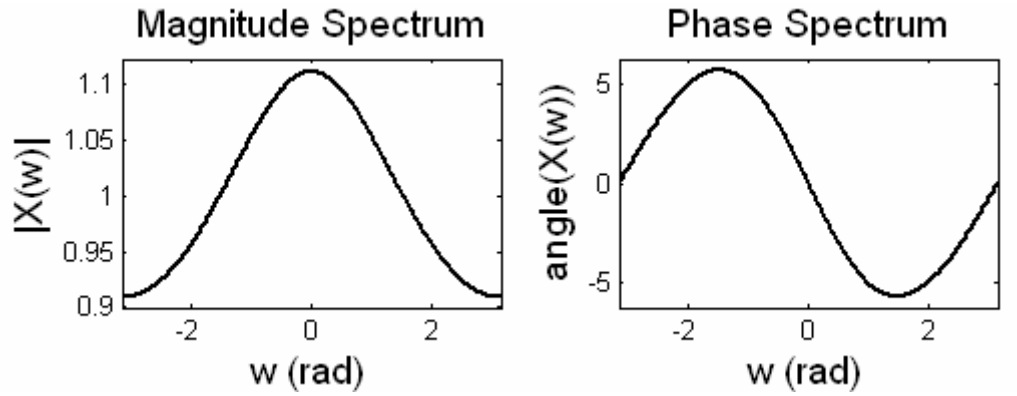


Figure 13. DTFT of the signal having the Z-transform, $H(\omega) = \dfrac{1}{1-0.1e^{-j\omega}}$

DISCRETE FOURIER TRANSFORM (DFT)

We know that aperiodic finite energy signals have continuous spectra (DTFT).

$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}$ .In case of a finite length sequence x(n), 0≤n≤L-1., Then only L values of X(ω) over its period, called the frequency samples, are sufficient to determine x(n) and hence X(ω). This leads to the concept of discrete Fourier transform (DFT) which is obtained by periodic sampling of $X(\omega)$ (DTFT).

We often compute a higher point (N point) DFT where N > L . This is because padding the sequence x(n) with N-L zeros and computing an N point DFT results in a "better display" of the Fourier transform X(ω).

To summarize, the formulas are (for causal sequence)

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-\frac{j2\pi kn}{N}} = \sum_{n=0}^{L-1} x(n)e^{-\frac{j2\pi kn}{N}} (DFT),\ k = 0,1,....,\ N-1$$

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{\frac{j2\pi kn}{N}} (IDFT),\ n = 0,1,.....N-1$$

In fact (3) is nothing but general DFT matrix equation.

$\boldsymbol{X}=\boldsymbol{W}_N\boldsymbol{x}$ DFT equation

$\boldsymbol{X}=\frac{1}{N}\boldsymbol{W}_N^{*}\boldsymbol{X}$ IDFT equation

Where $W_N=e^{-j2\pi/N}$

$\mathbf{x}=[x(0)\ x(1)\ \ldots\ x(N-1)]^T$

$\mathbf{X}=[X(0)\ X(1)\ \ldots\ X(N-1)]^T$

$$\mathbf{W}_N = \begin{array}{c} \xrightarrow{\quad n \quad} \\ \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & W_N & W_N^{2} & \ldots & W_N^{N-1} \\ 1 & W_N^{2} & W_N^{4} & \ldots & W_N^{2(N-1)} \\ . & . & . & .... & . \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \ldots & W_N^{(N-1)^2} \end{bmatrix} \end{array} \begin{array}{l} \\ \\ \\ \\ \\ \downarrow\ k \end{array}$$

Note : W_N matrix can be generated by dftmtx**()** function.

The following example calculates the 5 point DFT of the sequence, x(n) =[1 1 0 0 1].

```
%Example
%CALCULATION OF 5 POINT DFT
clear all;
x=[1,1,0, 0, 1];
N=5;←    %NO OF FREQUENCY SAMPLES
L=length(x);
x=[x zeros(1,N-L)]; %ZERO PADDING
n=length(x);
n=0:N-1;    %INDEX OF DATA SEQUENCE
k=0:N-1;    %INDEX OF FREQUENCY SAMPLE
Wn=exp(-j*2*pi/N);
WN=Wn.^(n'*k);
X=WN*x';     %DFT
figure(1), subplot(211),stem(k/N,abs(X),'k',  'linewidth', 2), xlabel('Digital frequency, f (Hz)', 'fontsize',
14),  ylabel('|X(f)|',  'fontsize', 14), title('Magnitude Spectrum', 'fontsize', 14);
subplot(212),stem(k/N,angle(X)*180/pi, 'k', 'linewidth', 2), xlabel('Digital frequency, f (Hz)', 'fontsize',
14),  ylabel('angle(X(f))',  'fontsize', 14), title('Phase Spectrum', 'fontsize', 14);
 abs(X)
angle(X)*180/pi
```

```
OUTPUT OF EXAMPLE
abs(X) =
3.0000    1.6180    0.6180    0.6180    1.6180
angle(X)*180/pi=
0         0 -180.0000 -180.0000    0.0000
```

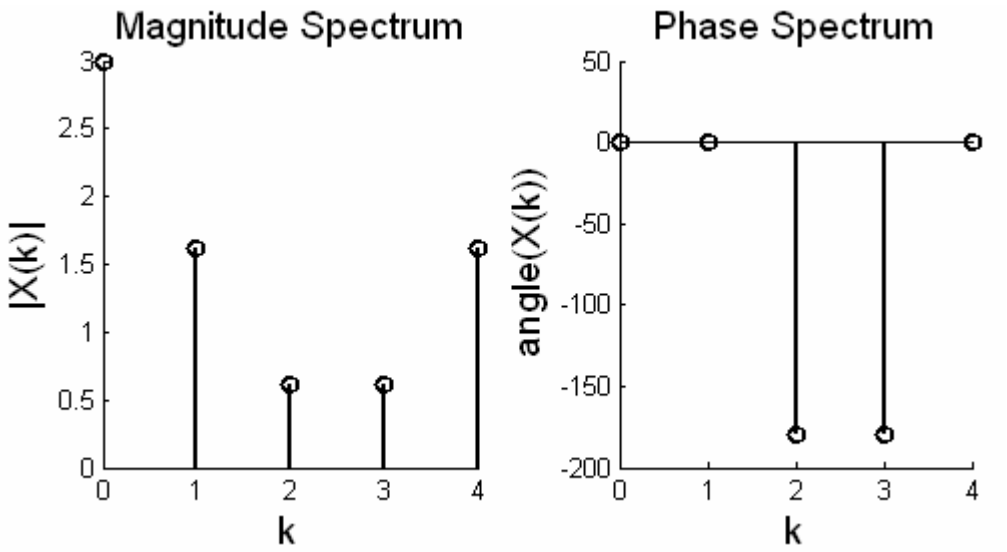


Figure 14. Plot of DFT

In order to understand the effects of zero padding, we set N=1000, plot the coefficients with respect to digital frequency, f=k/N. Then same plot the coefficients of 5-point DFT with respect to f. You should appreciate the fact that that the coefficients for 5-point DFT are actually samples from the DTFT.

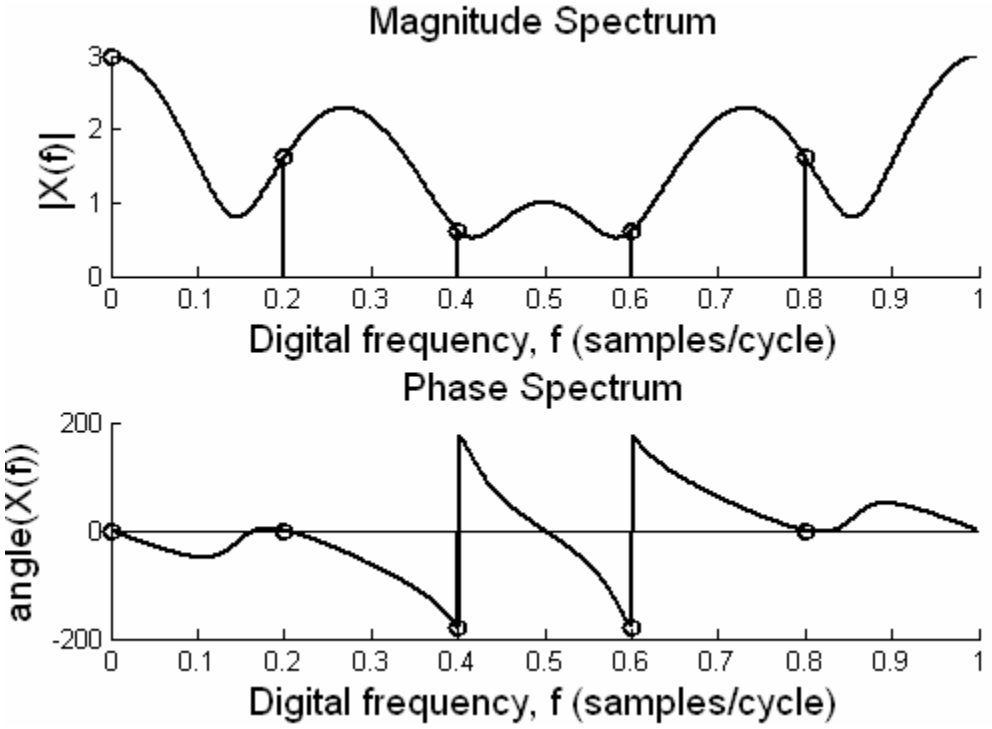


Figure 15. Effect of zero padding.

Parseval's Theorem:

It states that if $x(n) \xleftrightarrow[N]{DFT} X(k)$ then

$$\sum_{n=0}^{N-1} x(n)x^*(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)X^*(k)$$

$$i.e \sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N}\sum_{k=0}^{N-1} |X(k)|^2$$

Hence energy is conserved in each domain.

```
%Example
%VERIFICATION OF PERSEVAL'S THEOREM
x=ones(1,10);
Et=sum(x.*x);
X=fft(x,128);
Ef=sum(X.*conj(X))/128
```

```
RESULTS
Et =
     10

Ef

Ef =      10.0000
```

IMPORTANT PROPERTIES OF DFT:

1. Periodicity: If x(n) and X(k) are an N-point DFT pair, then
X(n+N)= x(n) for all n
X(k+N)=X(k) for all k

2. Linearity:
If
$x_1(n) \xleftrightarrow{N-DFT} X_1(k)$
And
$x_2(n) \xleftrightarrow{N-DFT} X_2(k)$

Then for any real valued or complex valued constant a_1 and a_2

$$a_1 x_1(n) + a_s x_s(n) \xleftrightarrow{N-DFT} a_1 X_1(k) + a_2 X_2(k)$$

3. Time reversal of a sequence:

$$x((-n))_N = x(N-n) \xleftrightarrow{N-DFT} X((-k))_N = X(N-k)$$

4. Circular time shift reversal of a sequence:

$$x((n-l))_N \xleftrightarrow{N-DFT} X(k)e^{-i2\pi kl/N}$$

5. Circular frequency shift :

$$x(n)e^{i2\pi kl/N} \xleftrightarrow{N-DFT} X((k-l))_N$$

The following example illustrates the circular frequency shift property.

```
%Example
%CIRCULAR FREQUENCY SHIFT
clear all;
x=[1, 0,1, 0, 1];
N=10;    %NO OF FREQUENCY SAMPLES
L=length(x);
x=[x zeros(1,N-L)]; %ZERO PADDING
n=length(x);
n=0:N-1;    %INDEX OF DATA SEQUENCE
k=0:N-1;    %INDEX OF FREQUENCY SAMPLE
l=3;    %SHIFT
for ii=1:N,
  x(ii)=x(ii)*exp(i*2*pi*ii*l/N);
end
Wn=exp(-j*2*pi/N);
WN=Wn.^(n'*k);
X=WN*x';     %DFT

figure(1), subplot(313),stem(k,abs(X),'k', 'linewidth', 2), ylabel('|X(k)|', 'fontsize', 14), title('Magnitude
Spectrum', 'fontsize', 14)%, xlabel('k', 'fontsize', 14);
```

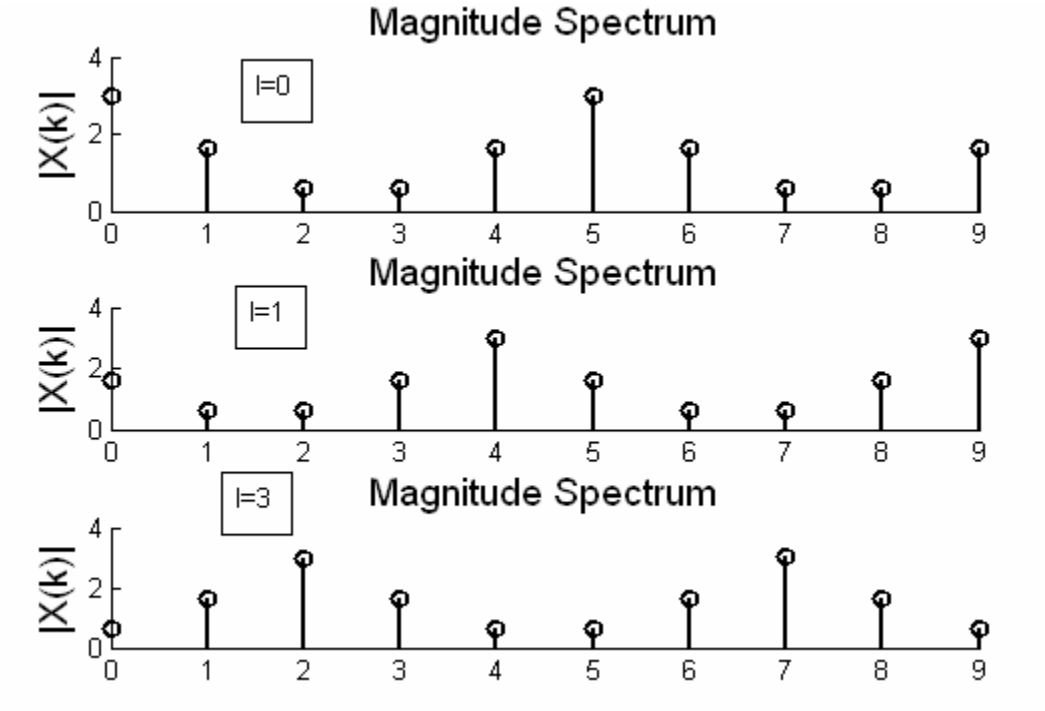


Figure 16. Circular frequency shift.

Sampling theorem revisited:

For example, sampling $x_a(t)$ periodically at T seconds, we get

$$x(n) = x_a(nT) \quad -\infty < n < \infty$$

If $x_a(t)$ is an aperiodic signal with finite energy, then its spectrum can be given as

$$X_a(F) = \int_{-\infty}^{\infty} x_a(t) e^{-j2\pi Ft} dt$$

Spectrum of discrete time signal is given by

$$X(f) = \sum_{n=-\infty}^{\infty} x(n) e^{-j2\pi fn}$$

The relationship of spectra between continuous and discrete time signal can be obtained as

$$X(f) = F_s \sum_{k=-\infty}^{\infty} X_a[(f-k)F_s] \quad [\text{ Recall that, } f = F/F_s]$$

[See Proakis for proof.]

Example :

Consider, the aperiodic finite energy signal

$$m_2(t) = \begin{cases} \sin c^2(100t), & |t| \le t_o \\ 0, & otherwise \end{cases}$$

where, $t_o = 0.1s$. Baseband BW of the signal is 100 Hz. Show the spectra of the sampled signal for $F_s = 4BW$, $F_s = 2BW$, $F_s = 1.25BW$ and $F_s = BW$.

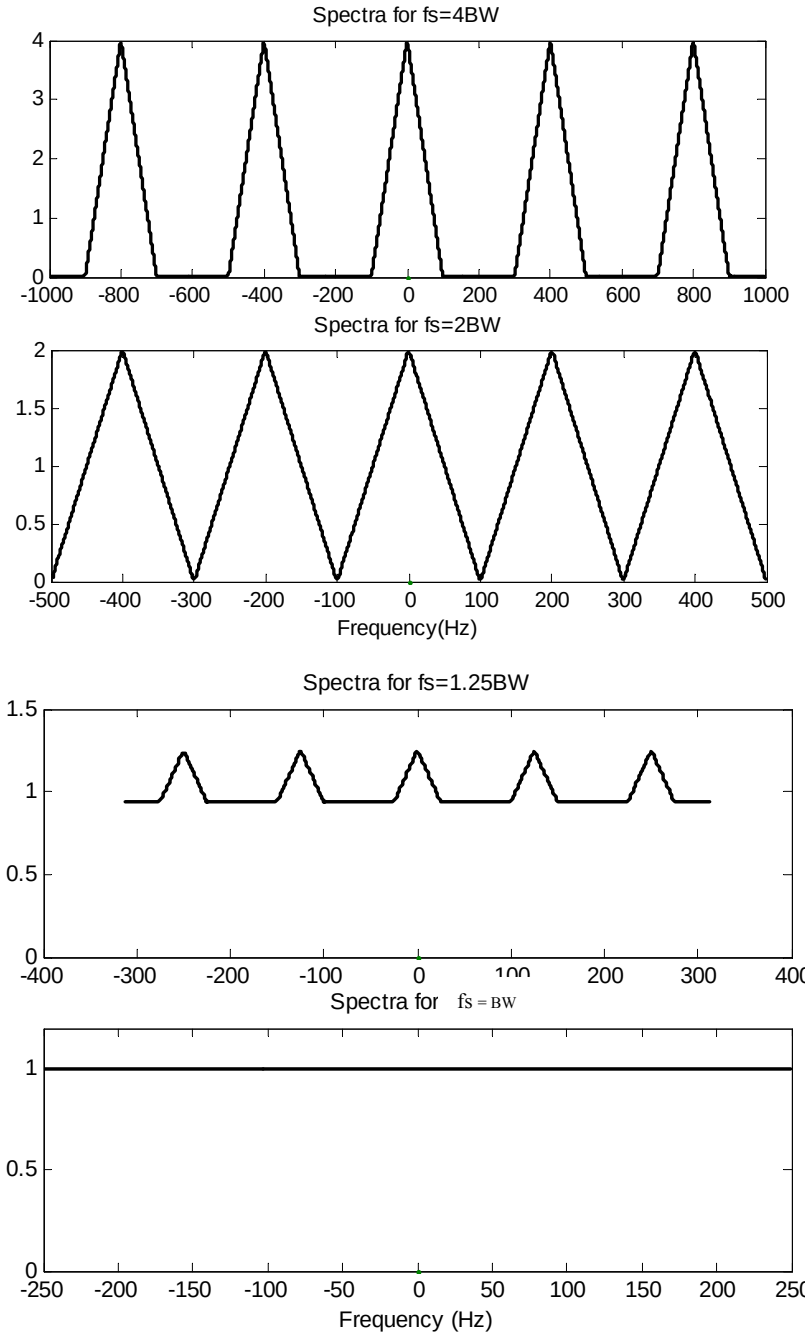


Figure 17: Spectra of signals sampled at different rate

The MATLAB codes for these figures are given below.

```
clear all
close all

%fs=4*BW
to=.1;
ts=.0025;
fs=1/ts;
nup=round(to/ts);
nlo=-nup;% generate time index
n=nlo:nup;
t=n*ts;
m=(sinc(100*t)).^2;
N=1024;% FFT bin size ,after zero padding
fn = [0:1/N:5-1/N]*fs-5*fs/2;
k=0:5*N-1;
WN=exp(-j*2*pi/N);
nk=n'*k;
W=WN.^nk;
M=m*W;
%M=fft(m,1024);
subplot(211),plot(fn,abs(fftshift(M)),0,0)%

%fs=2*BW
ts=.005;
fs=1/ts;
nup=round(to/ts);
nlo=-nup;% generate time index
n=nlo:nup;
t=n*ts;
m=(sinc(100*t)).^2;
N=1024;% FFT bin size ,after zero padding
fn = [0:1/N:5-1/N]*fs-5*fs/2;
k=0:5*N-1;
WN=exp(-j*2*pi/N);
nk=n'*k;
W=WN.^nk;
M=m*W;
%M=fft(m,1024);
subplot(212),plot(fn,abs(fftshift(M)),0,0)%
```

```
%fs=1.25*BW

ts=.008;
fs=1/ts;
nup=round(to/ts);
nlo=-nup;% generate time index
n=nlo:nup;
t=n*ts;
m=(sinc(100*t)).^2;
N=1024;% FFT bin size ,after zero padding
fn = [0:1/N:5-1/N]*fs-5*fs/2;
k=0:5*N-1;
WN=exp(-j*2*pi/N);
nk=n'*k;
W=WN.^nk;
M=m*W;
%M=fft(m,1024);
figure(2)
subplot(211),plot(fn,abs(fftshift(M)),0,0)%

%fs=BW

ts=.01;
fs=1/ts;
nup=round(to/ts);
nlo=-nup;% generate time index
n=nlo:nup;
t=n*ts;
m=(sinc(100*t)).^2;
N=1024;% FFT bin size ,after zero padding
fn = [0:1/N:5-1/N]*fs-5*fs/2;
k=0:5*N-1;
WN=exp(-j*2*pi/N);
nk=n'*k;
W=WN.^nk;
M=m*W;
%M=fft(m,1024);
subplot(212),plot(fn,abs(fftshift(M)),0,0)
```

This laboratory manual is prepared by

Toufiqul Islam & Asif Islam Khan

and supervised by

Prof. Md. Kamrul Hasan

August 1, 2007

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING,
DHAKA-1000, BANGLADESH.

COURSE NO.: EEE 312

Experiment No. – 4
Phase II

Frequency domain analysis of DT signals and systems

% Example teaches better than precept – S.Smiles

In this phase, we will concentrate on major DFT applications such as linear and circular convolutions, estimation of correlations, frequency analysis, modulation and demodulation last but not the least applications in communications such as FDM.

Pre Lab Concerns:

- Read this lab description carefully before coming to the laboratory class, so that you will know what is required.
- Try to follow the lecture notes of EEE 311.
- The problems that will be given in the class may differ from the given ones but they will certainly cover the syllabus.
- Students must not use the function fft() unless instructed to do so. It is desired that students write their own codes to calculate DFT coefficients from matrix equations.
- Theoretical derivations are avoided in the manual. You can get the proofs in your textbook.
- Do not bring any prepared MATLAB code in the lab neither in written form nor in any portable device**.**

New functions used :

mod ( ), gallery ( ), ifft ( ), convmtx ( ), hanning ( ), fir2 ( )

1. Review of DFT :

The Fourier transform of a sequence is readily obtained from its z-transform simply by setting the complex variable **z** equal to exp(j**ω**). When the sequence of interest has a finite duration, we may go one step further and develop a Fourier representation for it by defining the *discrete Fourier transform (DFT)*. The DFT is itself made up of a sequence of samples, uniformly spaced in frequency. The DFT has established itself as a powerful tool in digital signal processing by virtue of the fact that there exits efficient algorithms for its numerical computation such as FFT.

Consider a finite duration causal sequence x(n), assumed of length N, then N-point DFT is given as

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}}, \qquad k = 0,1,....,N-1$$

Note : *Limit of summation will change for anticausal and non-causal sequences.*

The N-point inverse DFT (IDFT) of *X(k)* is defined by

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{\frac{j2\pi kn}{N}}, \qquad n = 0,1,.....N-1$$

The DFT has an interesting interpretation in terms of the z-transform: the DFT of a finite-duration sequence may be obtained by evaluating the z-transform of that same sequence at *N* points uniformly spaced on the unit circle in the z-plane. This sampling process is illustrated in the following figure (for N=8).

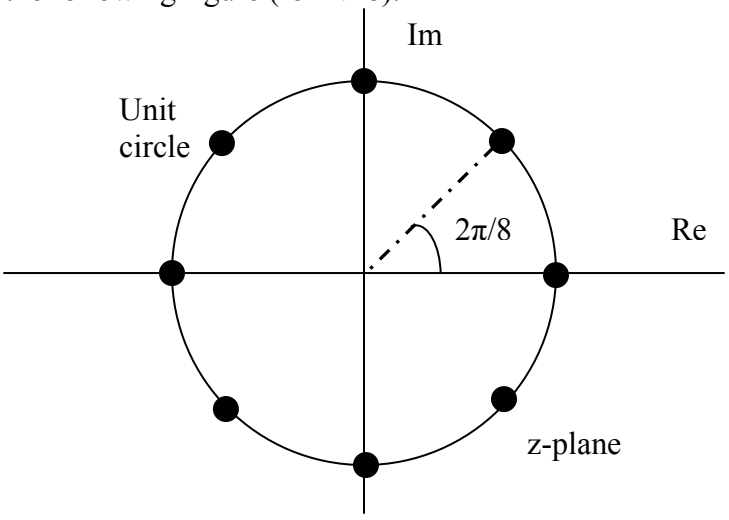


Figure 1: A set of 8 uniformly spaced points on the unit circle in the z-plane.

Though the sequences x(n) and X(k) are defined as finite-length sequences, in reality they both represent a single period of their respective periodic sequences. This double periodicity is the direct consequence of sampling a continuous time signal as well as its continuous Fourier transform.

2. Implementing convolutions using the DFT:

2.1 Circular Convolution:

If an N point sequence is shifted in either direction, then the result is no longer between $0 \le n \le N-1$. Therefore we first convert x(n) into its periodic extension $x_p(n)$ and then shift by m samples to obtain $x_p(n-m)$. Then finite duration sequence

$$x'(n) = \begin{cases} x_p(n-m) & \text{for } 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases}$$

$= x((n-m))_N$ (Represented as modulo-N index) [See Proakis for details]

Example:

x =[1 2 3 4 5];

Circularly shifting **x** by 2 samples will yield

x $'$ = [4 5 1 2 3];

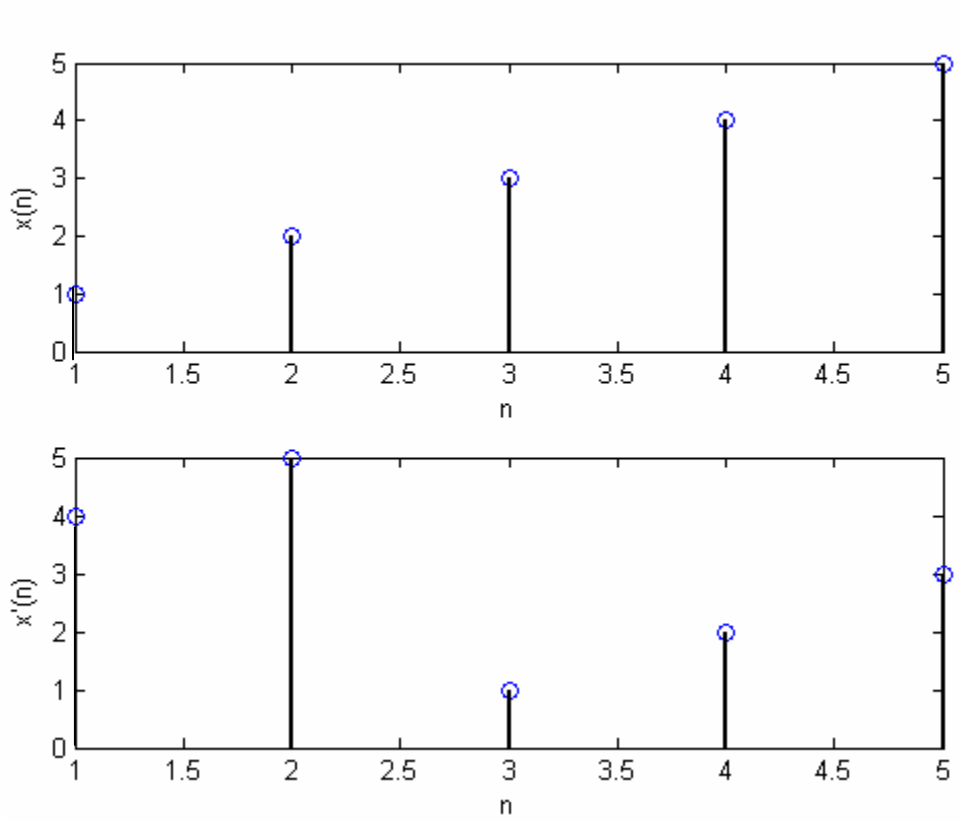


Figure 2 : **Two** sample circular right shift.

```
function [y,x]=cirshft(x,m,N) % for circular shifting
x=[x zeros(1,N-length(x))];
n=0:N-1;
n=mod(n-m,N);
y=x(n+1);
```

Circular convolution of $x_1(n)$ and $x_2(n)$ can be given as

$y(n)=\sum_{n=0}^{N-1} x_1(n)x_2((m-n))_N$ (N be the length of longest signal vector)

[See Proakis for illustration]

For two 4 point sequence $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, above equation can be written in matrix form

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix} = \begin{bmatrix} x_2(0) & x_2(3) & x_2(2) & x_2(1) \\ x_2(1) & x_2(0) & x_2(3) & x_2(2) \\ x_2(2) & x_2(1) & x_2(0) & x_2(3) \\ x_2(3) & x_2(2) & x_2(1) & x_2(0) \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix}$$

where,

$x_1=[x_1(0) \quad x_1(1) \quad x_1(2) \quad x_1(3)];$
$x_2=[x_2(0) \quad x_2(1) \quad x_2(2) \quad x_2(3)];$
$y=[y(0) \quad y(1) \quad y(2) \quad y(3)];$

In MATLAB , circular convolution can be obtained easily by matrix operation

y = gallery('circul',x2)'*x1'; % gallery function returns some basic test matrices.

gallery('circul',x2)' gives the circulant matrix of x_2

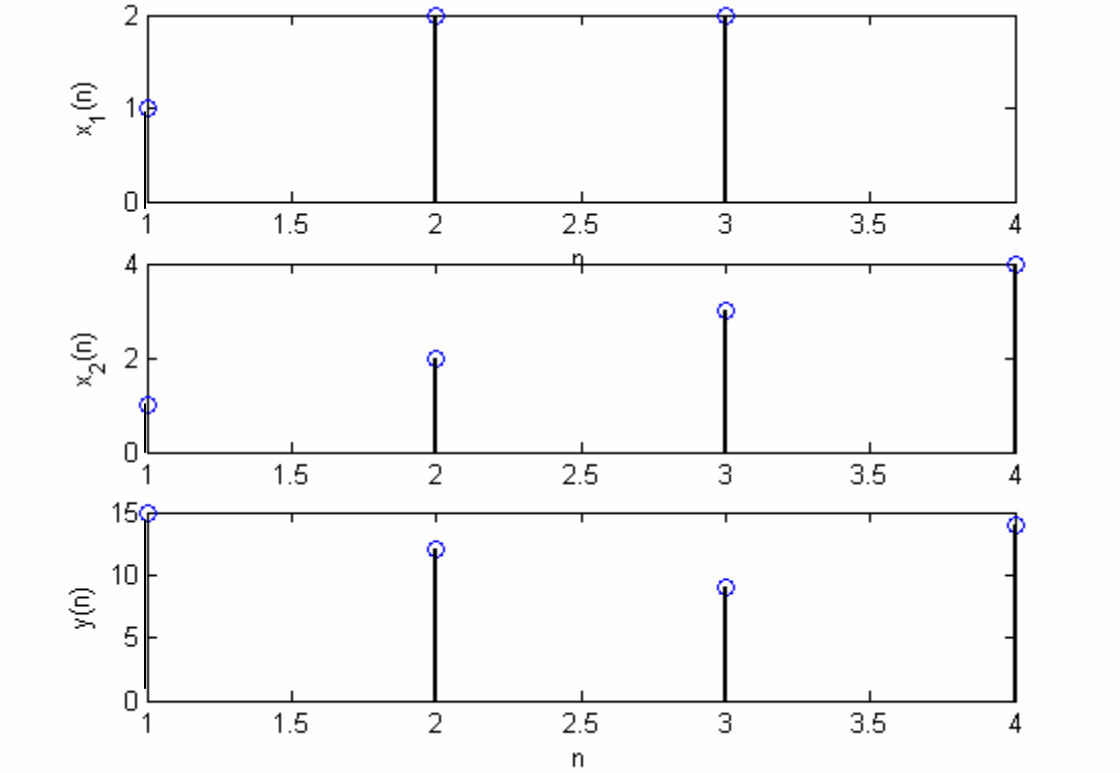


Figure 3: Showing sample circular convolution, y(n) of $x_1(n)$ and $x_2(n)$

2.2 Multiplication of two DFTs and Circular Convolution:

Suppose that we have two finite duration sequences of length *N*, $x_1(n)$ and $x_2(n)$. Their respective *N*-point DFTs are

$$X_1(k) = \sum_{n=0}^{N-1} x_1(n) e^{-\frac{2\pi nk}{N}} \quad k = 0,1,...,N-1$$

$$X_2(k) = \sum_{n=0}^{N-1} x_2(n) e^{-\frac{2\pi nk}{N}} \quad k = 0,1,...,N-1$$

It can be shown that multiplication of the DFTs of two sequences is equivalent to the circular convolution of the two sequences in the time domain. [See Proakis for proof]

$$y(n) = \sum_{n=0}^{N-1} x_1(n) x_2((m-n))_N$$
$$= x_1(n) \otimes x_2(n) \xrightarrow{DFT} Y(k) = X_1(k) X_2(k)$$

where, $\otimes$ denotes circular convolution.

If x_1 and x_2 are column vectors then circular convolution by DFT can be obtained in MATLAB as

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

2.3 Linear Convolution:

Suppose that we have input, x(n) of length L and filter response,h(n) of length M i.e

$x(n) = 0 \qquad n < 0 \text{ and } n \geq L$
$h(n) = 0 \qquad n < 0 \text{ and } n \geq M$

Then output response, *y(n)* can be given as

$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k)$, which is a linear convolution equation.

This equation can be expressed in matrix format,

For L = 4 and M = 3, N = L+M-1 (length of convolved result, *y*)

$$\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(4) \\ y(5) \end{bmatrix} = \underbrace{\begin{bmatrix} x(0) & 0 & 0 \\ x(1) & x(0) & 0 \\ x(2) & x(1) & x(0) \\ x(3) & x(2) & x(1) \\ 0 & x(3) & x(2) \\ 0 & 0 & x(3) \end{bmatrix}}_{\text{Convolution matrix of } \mathbf{x}} \begin{bmatrix} h(0) \\ h(1) \\ h(2) \end{bmatrix}$$

Very interesting technique, isn't it?

In signal processing, the better you handle equations in matrix format, the less it takes for MATLAB to compute.(and the more you excel as well !!)

For example, **h** = [1 2 3] and **x** = [1 2 2 1];

Then convolution matrix of **x** in MATLAB is given by

>> convmtx(x,3)

ans =

```
1   0   0
2   1   0
2   2   1
1   2   2
0   1   2
0   0   1
```

Hence, result of linear convolution can be obtained as

```
>> x=[1 2 2 1]';
>> h=[1 2 3]';
>> y=convmtx(x,3)*h
y =

   1
   4
   9
  11
   8
   3
```

2.4 Linear Convolution using DFT:

In previous section, we see that length of *y(n)* is N=L+M-1 . If the sequence *y(n)* is to be represented uniquely in the frequency domain by samples of its spectrum Y(*w*) at a set of discrete frequencies, the number of distinct samples must equal or exceed L+M-1. Therefore, a DFT of size N ≥ L+M-1 is required to represent {y(n)} in the frequency domain.

$Y(k) = X(k)H(k) \quad , \quad k=0,1,\ldots..,N-1$

{X(k)} and {H(k)} are the *N*-point DFTs of the corresponding sequences *x(n)* and *h(n)*. Since the sequences have a duration less than *N,* we simply pad these sequences with zeros to increase their length to *N.* Obviously zero padding does not alter the spectra.

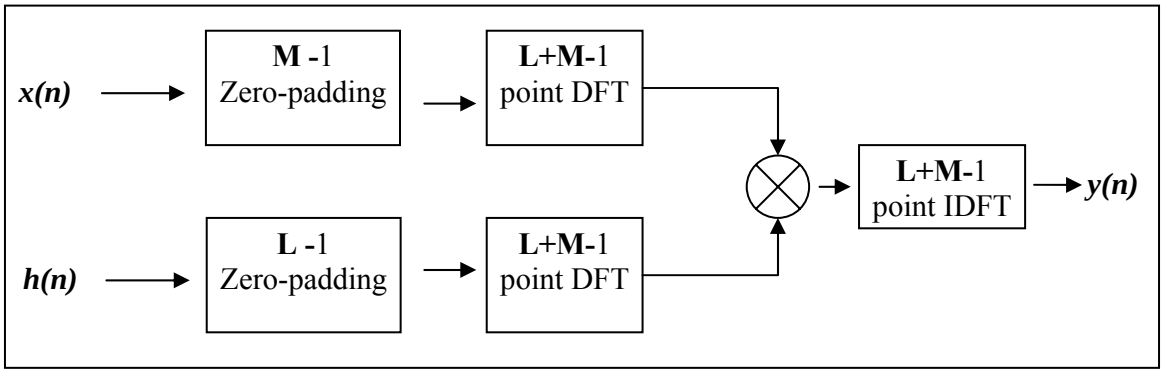


Figure 4: DFT based implementation of linear convolution.

Continuing from previous example,

h = [1 2 3] and **x** = [1 2 2 1];

Here N = 6 then

h=[1 2 3 0 0 0]; and x=[1 2 2 1 0 0]; Then in MATLAB , we can obtain the result as

```
y = ifft( fft(x) .* fft(h) );% by DFT approach
```

3. Correlation:

Remember that the cross-correlation equation can be written as

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-l) = x(n) * y(-n)$$

From properties of DFT, we know that

$r_{xy}(l) \xrightarrow{DFT} X(k)Y(-k) = X(k)Y^*(k)$ (Cross-energy spectral density.)

For ergodic or power signal

$$\hat{r}_{xy}(l) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)y(n-l) \xrightarrow{DFT} \frac{1}{N} X(k)Y^*(k)$$

Similarly ACF can be given as

$\hat{r}_{xx}(l) \xrightarrow{DFT} \frac{1}{N} X(k)X^*(k) = \frac{1}{N}|X(k)|^2$ which is the discrete power spectrum of the signal.

Hence one important application of autocorrelation is the estimation of the energy/power spectrum density of the signal.

Example:

Find ACF of x(n) defined as follows:

x=[1 1 0 1]**;** We take length of **x** to be 4+4-1=7 samples by padding 3 zeros.

```
x=[1 1 0 1];
x1=[x zeros(1,length(x)-1)];
X1=fft(x1);
X1conj=conj(X1);
Rx=fftshift(ifft(X1.*X1conj)); %  Calculation from Frequency domain
rx=xcorr(x); % Calculation in time domain
```

```
Results:
Rx =
   1.0000    1.0000    1.0000    3.0000    1.0000    1.0000    1.0000
rx =
    1    1    1    3    1    1    1
```

Exercise 3.1 :

Say, x_1=rand(1,51) % random signal

$x_2(n)$=sin(2л(1000)n/10000); % 1000 Hz signal

Find PSD of x(n) where $x(n)=x_1(n)+x_2(n)$; There from find ACF of x(n). Verify the result by calculating ACF in the time domain.

4. Frequency analysis using DFT:

The finite observational interval (say, T_o=LT, where L = number of samples and T = sample interval) for the signal places a limit on the frequency resolution; that is it limits our ability to distinguish two frequency components that are separated by less than $1/T_o = 1/(LT)$ in frequency.

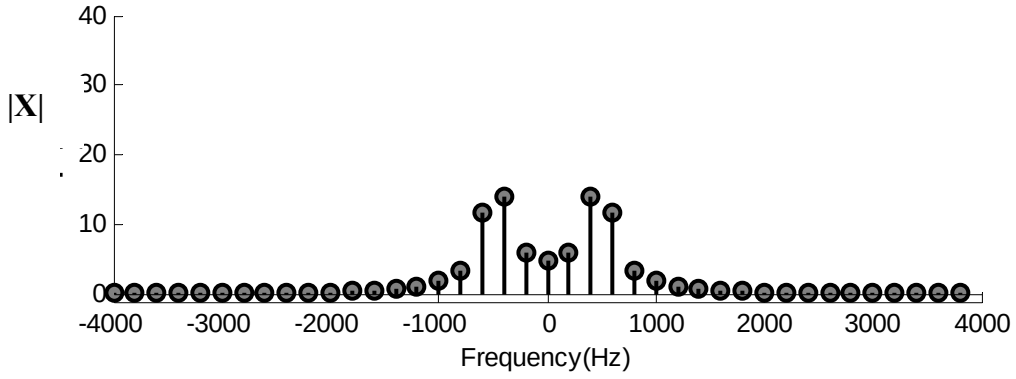
For example,

$$x(t) = \sin(2\pi f_1 t) + 0.05 \sin(2\pi f_2 t)$$

where,

f_1 = 500 Hz and f_2 = 1000 Hz . Let f_s = 8000 Hz

Case 1 : Take length(x), N = 40
Case 2 : Take length(x), N = 32



(a)

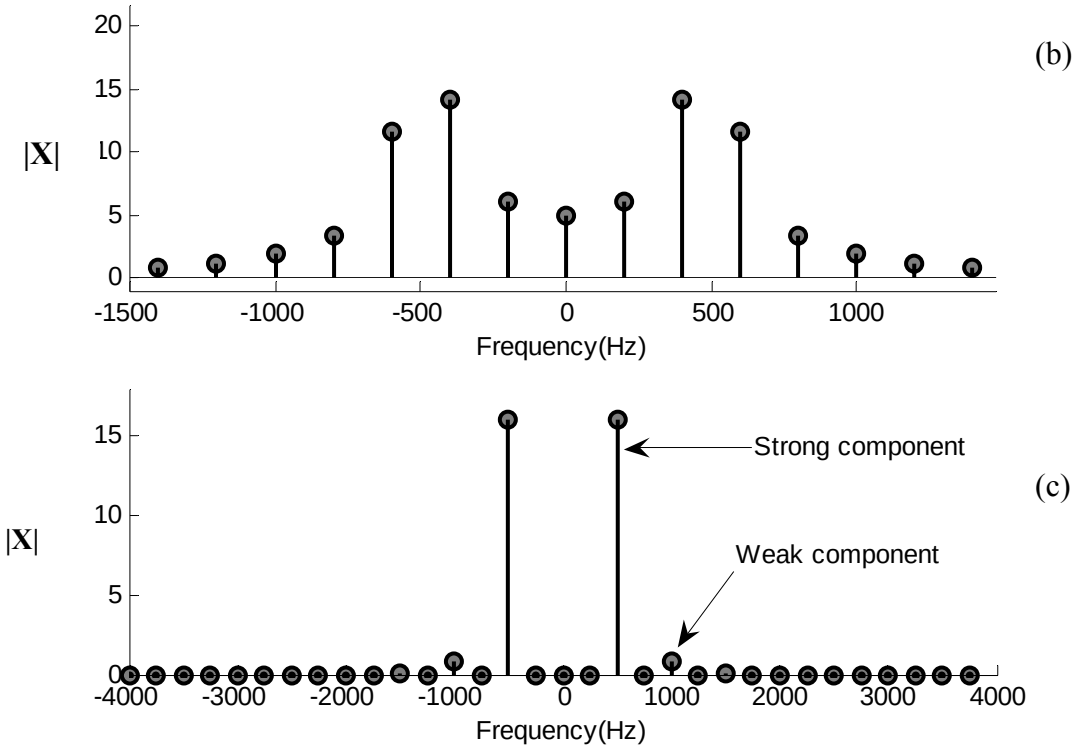


Figure 5: (a) - Magnitude Spectrum when N=40, (b) - Enlarged view for N = 40,
(c) - Magnitude Spectrum when N = 32.

```
% figure 5
f1=500;
f2=1000;
fs=8000;
ts=1/fs;
t=(0:4*fs/f2-1)*ts;
x=sin(2*pi*f1*t)+0.05*sin(2*pi*f2*t);
f=(0:length(t)-1)/(length(t)*ts)-1/(2*ts);
subplot(211),stem(t,x)
xlabel('Time(seconds)'),ylabel('x[n]')
subplot(212),stem(f,abs(fftshift(fft(x))))
xlabel('Frequency(Hz)'),ylabel('|X|')
axis([-4000,4000,0,40])
```

Several points are worthy to note:

a) Spectrum is not localized to f_1 and f_2. (for N = 40).
b) The frequency components occur at integer multiples of $\Delta f = f_s/N$ i.e at multiples of 200 Hz. That means frequency of the stronger sinusoid is exactly between co-efficients at 400 and 600 Hz. However, a visual interpolation of these frequency co-efficients does suggest a peak midway between 400 and 600 Hz that is stronger than either individually.
c) Leakage from the main spectral peak is readily apparent which causes the weak sinusoid to be masked.
d) In later case, f_1 and f_2 are exactly multiple of Δf (Here $\Delta f = 250$ Hz). that is, two sinusoids exactly fit in this 32 point signal for integer number of cycles. So in that case, weak component is not masked rather f_1 and f_2 spectral peaks are distinct. ($f_1 = 2\Delta f$ and $f_2 = 5\Delta f$)

4.1 DFT of a simple sinusoid :

Consider , $x(n) = \cos(w_x nT)$ where we choose $w_x = 2\pi(F_s / 4)$ with $F_s = 1$. We are using signal length N to be a power of 2 for fastest results. Here , $N = 64$.

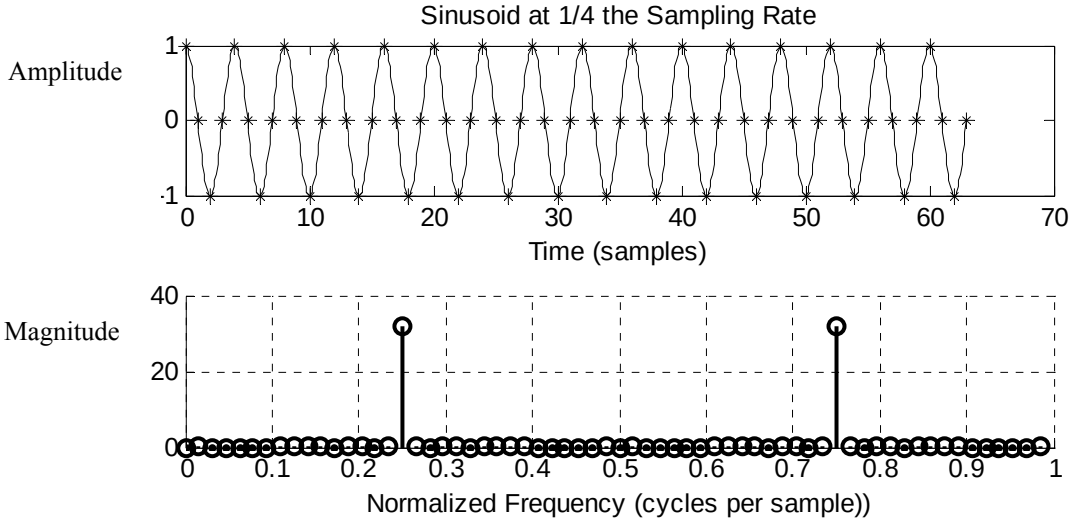


Figure 6: Spectrum when F(analog frequency) can be localized

Here, normalized frequencies are 0.25 (F/F_s) and 0.75 or ($0.75 - 1 = -0.25$, $-F/F_s$).

4.2 DFT of a not-so-simple sinusoid:

Here, set $w_x = 2\pi(\frac{F_s}{4} + 0.5\ \frac{1}{N})$. i.e. raising the frequency half-bin than before.

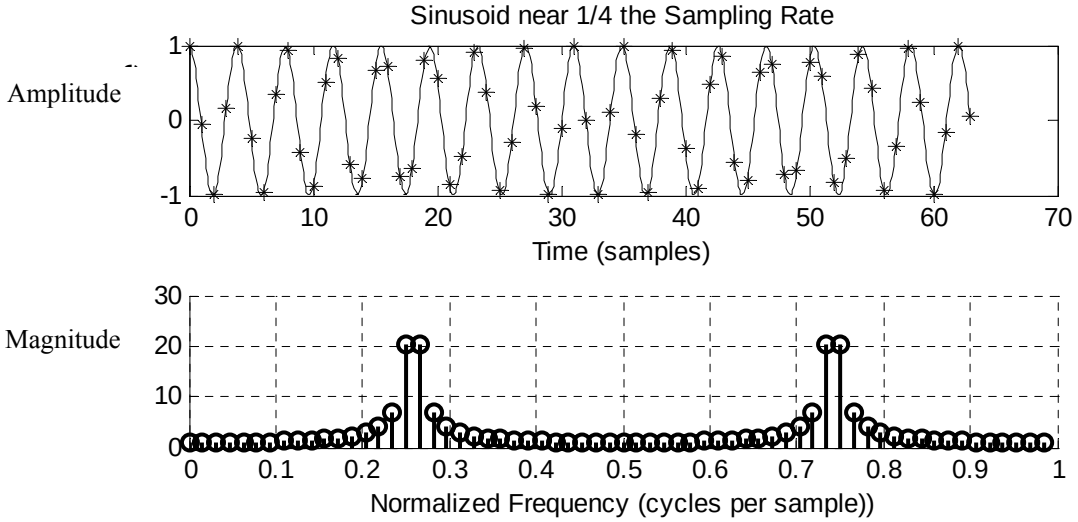


Figure 7: Spectrum when F cannot be localized.

In that case, sinusoid does not have integer number of cycles within N. So frequency of the sinusoid is no longer integer multiple of the bin-size (1/N). Hence, it is not localized in the DFT-bins. In fact, power, which was supposed to be confined in a single frequency component, spreads

into the entire frequency range. We say that the power has 'leaked out' into the entire frequency range. Consequently, this phenomenon is known as *leakage.* The following figure illustrates the occurrence of leakage. Here, signal is zero-padded to get a fine spectral view.

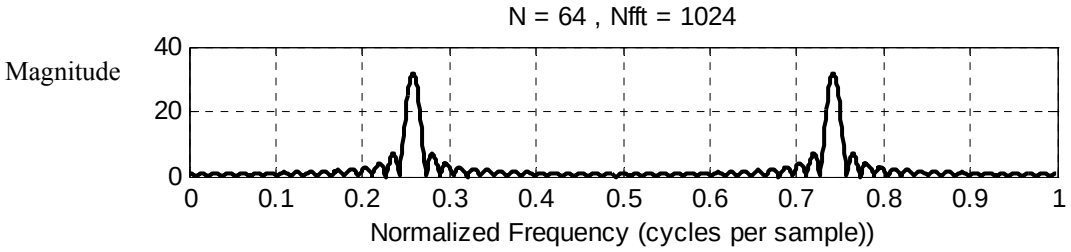


Figure 8: Interpolated spectrum after zero-padding.(DTFT)

4.3 Windowing to improve spectral estimation

The unavoidable act of finite data duration is equivalent to multiplying *x(n)* by a rectangular window *w(n)* of length L.

$$\text{where} \quad w(n) = \begin{cases} 1, & 0 \le n \le L-1 \\ 0, & otherwise \end{cases}$$

Suppose, $x(n) = \cos w_o n$ then

$$\hat{X}(\omega) = \frac{1}{2}\left(W(\omega - \omega_o) + W(\omega + \omega_o)\right)$$

where, W(w) is Fourier transform of the window.

Rectangular window function causes (see figure 8) false ripple or leakage in a spectral estimate. In order to this effect it is beneficial to use window functions whose Fourier transforms have lower sidelobes.

For this purpose, we may choose Hann window which has a 1st sidelobe -32dB relative to the main lobe where as rectangular window has only -13dB relative distance.(See following figure)

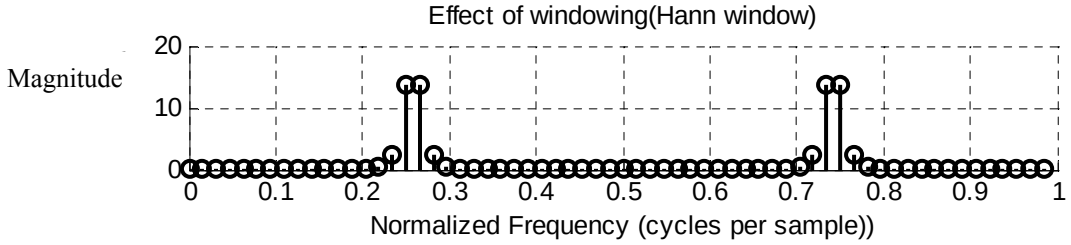


Figure 9: Hann windowed spectrum

Note : Obviously Hann windowing does not give you the true weight of spectral components, it just suppresses false spectral peaks as a result of rectangular windowing. So purpose of windowing here is to get idea of spectral content only.

Exercise 4 (4.1 – 4.3):

1. 64 samples 125μs apart are available for x(t) = sin (2лf$_1$t) + 0.05sin (2 лf$_2$t), where f$_1$ and f$_2$ are 1062.5 Hz and 1625 Hz respectively.
   a) Plot the data sequence and magnitude spectrum.(use **stem**())
   b) Perform windowing (Hann) and again plot the data sequence and magnitude spectrum Comment on your result.

2. "Windowing not only distorts the spectral estimate due to the leakage effects, it also reduces spectral resolution." ---Justify this claim by explanation and simulated example.[See Proakis]

3. "Ability to resolve closely spaced spectral lines of different frequencies in limited by the window main lobe width" --- Justify this statement by proper explanation and results of the following problem.

Let, $x(n) = \cos w_o n + \cos w_1 n + \cos w_2 n$ where $w_o = 0.2л$, $w_1 = 0.22л$ and $w_2 = 0.6л$.
Consider, window lengths L = 25,50,100. Comment on your result.

5. Modulation Theorem :

If $x(n) \xleftrightarrow{DTFT} X(\omega)$ then $x(n) \cos \omega_o n \leftrightarrow \frac{1}{2}[X(\omega + \omega_o) + X(\omega - \omega_o)]$

Consider the following example for DSB-SC modulation of an almost band limited signal, *m(t)*.

$$m(t) = \begin{cases} \sin c(100t), |t| \le t_o \\ 0, \quad otherwise \end{cases}$$

where, t$_o$ = 0.1s . Let, carrier, $c(t) = \cos(2\pi f_c t)$ where f_c = 250 Hz. Let, f_s = 1250 Hz.

```
%figure 10
to=.1;
ts=.001;
fc=250;
fs=1/ts;
t=[-to:ts:to];
m=sinc(100*t);
c=cos(2*pi*fc*t);
u=m.*c;% DSB -AM modulated Signal
N=1024;% FFT bin size ,after zero padding
M=fft(m,1024);
M=M/fs;%scaling
U=fft(u,1024);
U=U/fs;%scaling
fn = [0:1/N:1-1/N]*fs-fs/2;
subplot(211),plot(fn,abs(fftshift(M)))
subplot(212),plot(fn,abs(fftshift(U)))
```

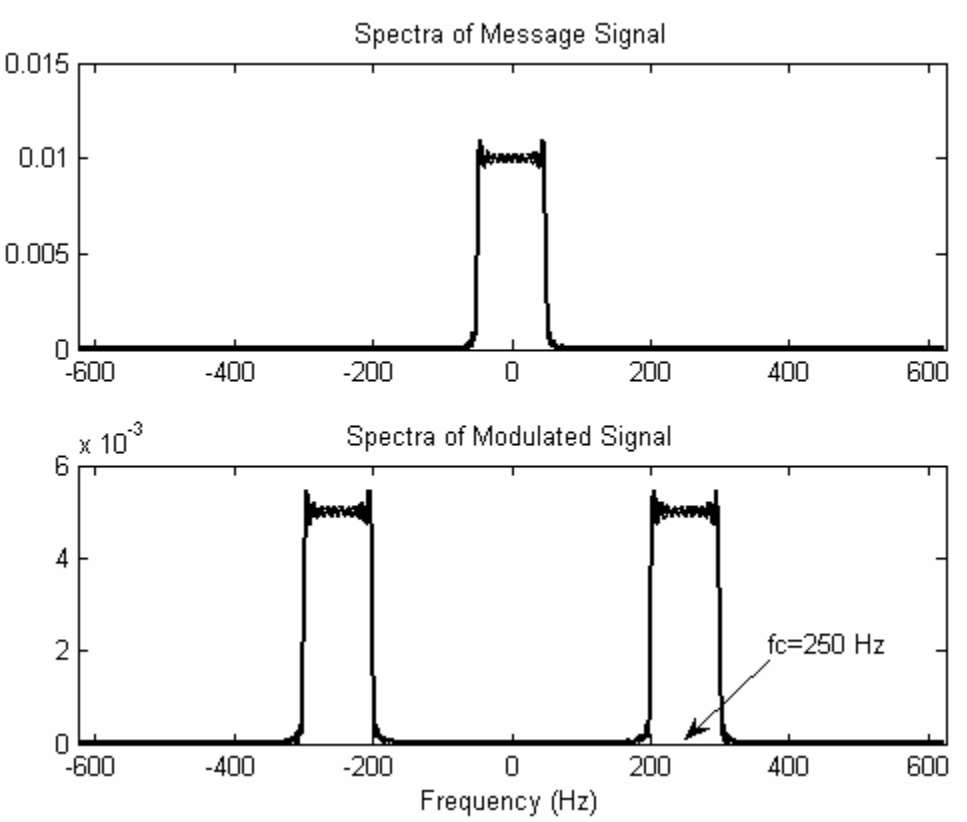


Figure 10: Spectra of message signal is shifted to the carrier frequency and magnitude is halved.

Exercise 5.1:

What happens if the duration of the message signal t_o changes; in particular; what is the effect of having large t_o's and small t_o's? What is the effect on the bandwidth? (for the example given, Figure 10)

Now consider the demodulation of the DSB-SC AM signal.

$$x(n)\cos^2 \omega_o n \leftrightarrow \frac{1}{2}X(\omega) + \frac{1}{4}[X(\omega + 2\omega_o) + X(\omega - 2\omega_o)]$$

Then after proper low-pass filtering we get the following result.(Figure 11(b))

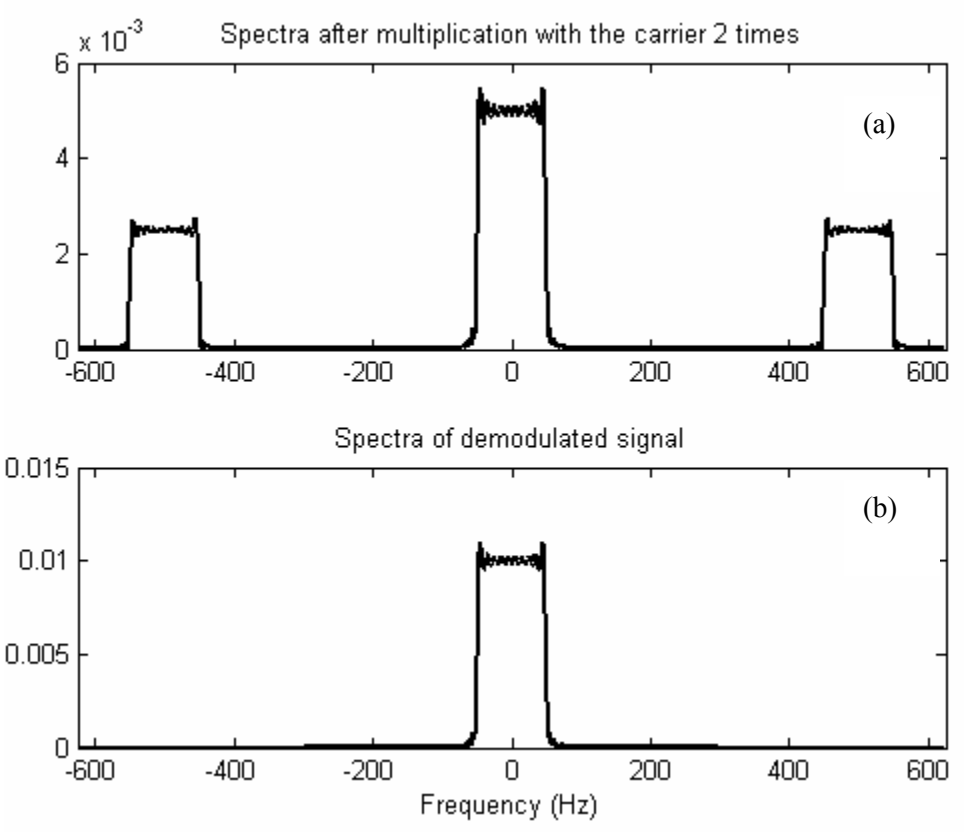


Figure 11: Spectra for demodulation

```
% figure 11
y=u.*c;
Y=fft(y,1024);
Y=Y/fs;
figure(2)
subplot(211),plot(fn,abs(fftshift(Y)))
fcut=200;
ncut=floor(fcut*fs/N);
H=zeros(1,N);
H(1:ncut)=2*ones(1,ncut);
H(N-ncut+1:N)=2*ones(1,ncut);
Uprime=Y.*H;
subplot(212),plot(fn,abs(fftshift(Uprime)))
```

Exercise 5.2:

Consider the following FDM system.

$$m_1(t) = \begin{cases} \sin c(100t), |t| \le t_o \\ 0, \quad otherwise \end{cases}, \qquad m_2(t) = \begin{cases} \sin c^2(100t), |t| \le t_o \\ 0, \quad otherwise \end{cases}$$

where, $t_o = 0.1$s . Let, carrier, $c_1(t) = \cos(2\pi f_{c_1} t)$ where f_{c1}= 250 Hz. Let, f_s = 2000 Hz. and carrier, $c_2(t) = \cos(2\pi f_{c_2} t)$ where f_{c2} = 750 Hz. Obtain magnitude spectra at different stages of the system.

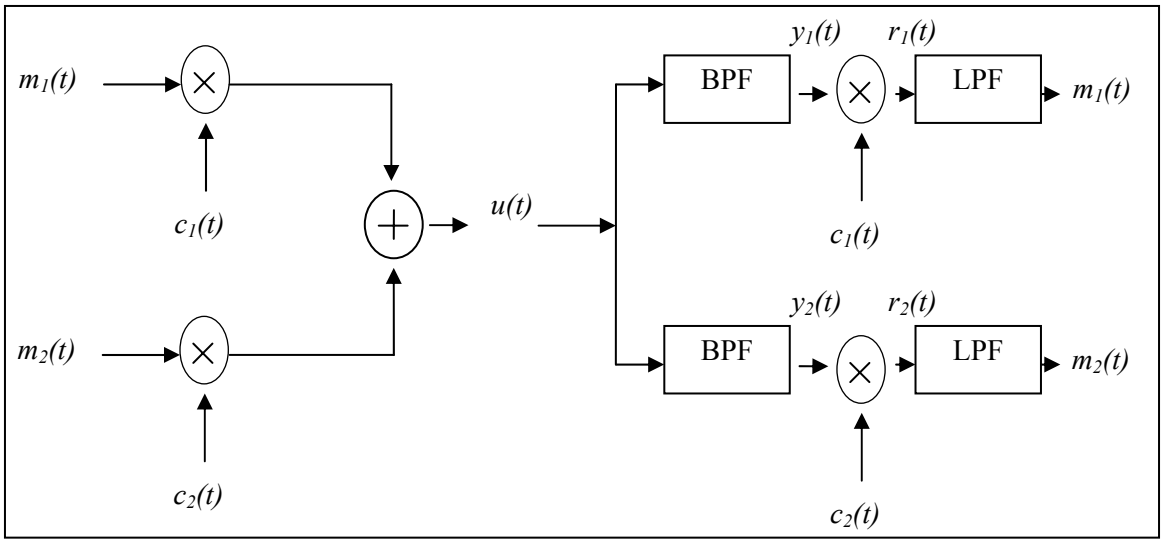


Figure 11: Simple FDM system

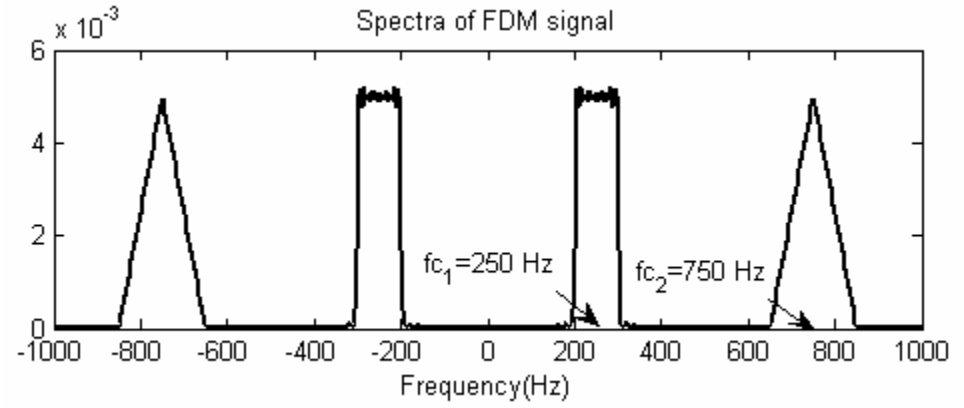


Figure 12: Spectra of FDM signal.

Appendix :

Generating arbitrary bandlimited input sequence for test purpose:

M-function *fir2*() can be used to generate a causal finite-length sequence with a bandlimited frequency response. For example, we want a sequence with the following spectrum.

fir2() :

- *b = fir2(n,f,m)* returns row vector b containing the n+1 coefficients of an order n FIR filter. The frequency-magnitude characteristics of this filter match those given by vectors *f* and *m*:
- *f* is a vector of frequency points in the range from 0 to 1, where 1 corresponds to the Nyquist frequency. The first point of *f* must be 0 and the last point 1. The frequency points must be in increasing order.
- *m* is a vector containing the desired magnitude response at the points specified in *f*.

f and *m* must be the same length.

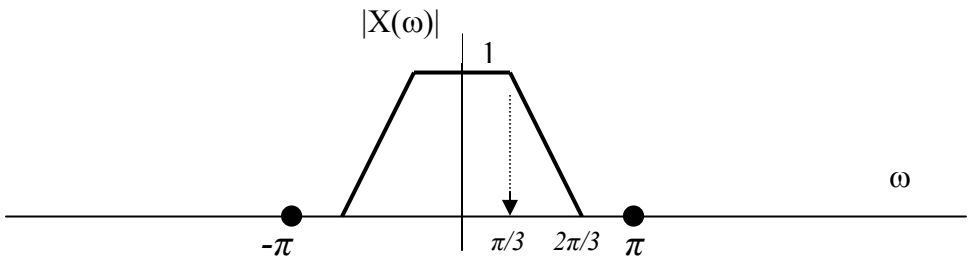


Figure 13: Arbitrary bandlimited magnitude spectra.

```
>> w=[0 pi/3 2*pi/3 pi];% digital angular frequency
>> freq=w/pi; % Normalized angular frequency
>> x=fir2(99,freq,mag); % Generating 100 point sequence
>> [X,w]=freqz(x,1,512); % DTFT
>> plot(w/pi,abs(X),'k','Linewidth',2)
```

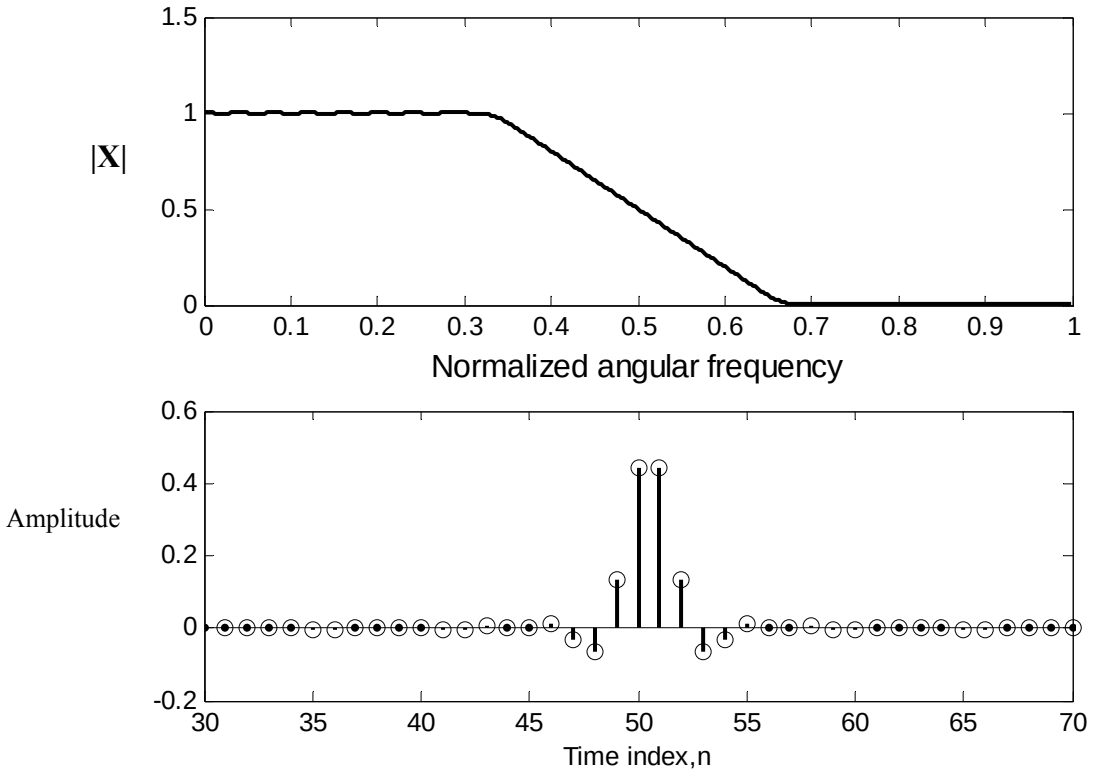


Figure 14: Magnitude spectra upto π (Nyquist frequency). and time sequence for **30:70** shown.

FAQs :

1. **How can we be certain that most of the frequency contents of x(t) are in the DFT?**

Ans : To get X(k) from x(t), we need first to sample x(t) making sure that f_s , the sampling frequency used to produce x(n), is at least twice f_m , the maximum frequency in x(t). This is necessary to avoid aliasing. Next, N samples should be collected from x(t) in NT_s sec, where $T_s = 2\pi f_s$. The number of samples N is inversely related to the frequency resolution, Δf according to

$$\Delta f = \frac{f_s}{N} \quad Hz$$

The smaller the frequency resolution is, the better is the process of detecting most of the frequency components in x(t).

2. **Is the Circular Convolution the same as the Linear Convolution ?**

Ans: The circular convolution of two signals $x_1(n)$ and $x_2(n)$ will be the same as the linear convolution if we append zeros to the end of $x_1(n)$ and $x_2(n)$ so that the number of samples in $x_1(n)$ and $x_2(n)$ will be $N_1 + N_2 - 1$ where N_1 is the number of samples in $x_1(n)$ and N_2 is the number of samples in $x_2(n)$.

3. **Is** $|X(\omega)| \cong |X(k)|$?

Ans: Due to the inherent factor $1/T_s$ in calculating the Fourier transform of discrete signals(due to sampling), the magnitude of the DFT, $|X(k)|$ should be multiplied by T_s to get the approximation to $|X(w)|$, continuous time Fourier transform .

4. **Convolution can be performed in time domain or can be obtained by DFT-IDFT relations. Which is faster?**

Ans: Create two signals x1 and x2 that each have 5,000 random elements and compare the length of time it takes to circularly convolve the two functions by matrix relation(time domain approach) and the DFT method given.

```
%by conv
tic
x1=rand(1,5000);
x2=rand(1,5000);
y=gallery('circul',x2)'*x1';
toc
```

```
%by DFT

tic
y=ifft( fft(x1) .* fft(x2) );
toc
```

Tic saves the system clock into a hidden global variable and **toc** reads that variable, compares it with the current time, and displays the difference. Record how long it takes with each method, and be prepared to be surprised.

Sample Result:
Elapsed time is 833.422000 seconds.(by conv)
Elapsed time is 1.516000 seconds.(by DFT)

Insightful Exercises:

1. What do you know about the nature of PSD of a white noise sequence? Flat !! .Lets check it out. First, obtain a 1000 point random white noise sequence, w(n), obtain R_w (autocorrelation) and then perform DTFT. It is better to plot PSD curve in 10log(PSD). Vs ω format. Comment on your result.

2. Present a technique to obtain N-point DFTs of two real sequences using a single N-point DFT .

   Say, x(n) = {1, 2, 0, 1} and y(n) = {2, 2, 1, 1}

   Obtain X(k) and Y(k) by using 4-point DFT only once.

3. Observe the output spectrum of an up-sampler. Consider the bandlimited input sequence used in Appendix. If L is the upsampling factor then

```
>> y=zeros(1,L*length(x));
>> y(1:L:length(y))=x; % up-sampled sequence
```

   Explain your answer on mathematical basis.[Hint : Find z-transform expression of upsampler output. Take help from Experiment 2 –Exercise 1.3]

Reference :

1) Proakis & Manolakis,"Digital Signal Processing:Principles,Algorithms and Applications.",Prentice Hall Ltd.
2) Mitra," Digital Signal Processing: A Computer Based Approach" , Edition 1998, Tata McGraw -Hill Co. Ltd.
3) Denbigh, "System Analysis and Signal Processing" , Edition 1998, Addison-Wesley.
4) Elali, "Discrete Systems and Digital Signal Processing with MATLAB® " ,Edition 2004, CRC Press
5) Ingle & Proakis, "Digital Signal Processing using *MATLAB*® " ,Edition 2000 Thomson-Brooks/Cole Co .
6) Salehi & Proakis, "Contemporary Communication Systems using *MATLAB*®",Thomson-Brooks/Cole Co

---

This laboratory manual is prepared by

Toufiqul Islam

and supervised by

Prof. Md. Kamrul Hasan **July 28, 2007**

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING,
DHAKA-1000, BANGLADESH.

COURSE NO.: EEE 312

Experiment No. – 5

FIR Filter Design

Objective: This lab introduces the concepts of digital filters and covers the systematic methods of finite impulse response (FIR) filter design. The standard filter design techniques used in this experiment include Filter Design using Standard Windows and Parks-McClellan algorithm. When the students will complete the lab, they will able to design FIR filters to meet the specifications and compare different filter design techniques.

Introduction:

In digital signal processing applications, it is often necessary to change the relative amplitudes of frequency components or remove undesired frequencies of a signal. This process is called filtering.

A linear and time-invariant causal digital filter with input x(n) and output y(n) may be specified by its difference equation

$$y(n) = \sum_{i=0}^{N-1} b_i x(n-i) - \sum_{k=1}^{M} a_k y(n-k) \qquad (1)$$

where b_i and a_k are coefficients which parameterize the filter. This filter is said to have N zeros and M poles. Each new value of the output signal, y(n), is determined by past values of the output, and by present and past values of the input. The impulse response, h(n), is the response of the filter to an input of δ(n), and is therefore the solution to the recursive difference equation

$$h(n) = \sum_{i=0}^{N-1} b_i \delta(n-i) - \sum_{k=1}^{M} a_k h(n-k) \qquad (2)$$

There are two general classes of digital filters: infinite impulse response (IIR) and finite impulse response (FIR). The FIR case occurs when $a_k = 0$, for all k. Such a filter is said to have no poles, only zeros. In this case, the difference equation (2) becomes

$$h(n) = \sum_{i=0}^{N-1} b_i \delta(n-i) \qquad (3)$$

Since (3) is no longer recursive, the impulse response has finite duration N. In the case where $a_k \neq 0$, the difference equation usually represents an IIR filter. In this case, (2) will usually generate an impulse response which has non-zero values as $n \rightarrow \infty$.

By taking Z-transforms of both sides of equation (1), we obtain

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{N-1} b_i z^{-i}}{1 + \sum_{k=1}^{M} a_k z^{-k}} \qquad (4)$$

From this formula, we see that any filter which can be represented by a linear difference equation with constant coefficients has a rational transfer function (i.e. a transfer function which is a ratio of polynomials). There are many different methods for implementing a general recursive difference equation of the form (1). Depending on the application, some methods may be more robust to quantization error, require fewer multiplies or adds, or require less memory. Fig. 1 shows a system diagram known as the direct form implementation; it works for any discrete-time filter described by difference equation (1). Note that the boxes containing the symbol z^{-1} represent unit delays, while a parameter written next to a signal path represents multiplication by that parameter.

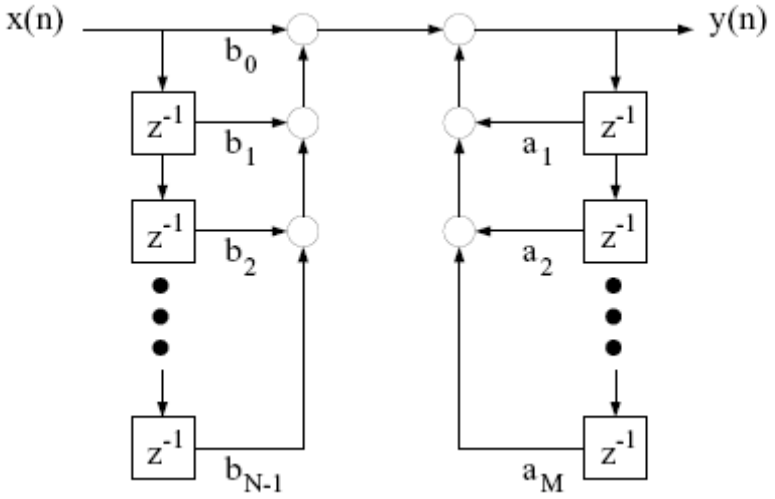


Fig. 1: Direct form implementation for a discrete-time filter described by a general difference equation of the form (1).

Filter design techniques:

1. Filter Design Using Truncation

Ideally, a low-pass filter with cutoff frequency ω_c should have a frequency response of

$$H_d(e^{jw}) = \begin{cases} 1, & |\omega| \le \omega_c \\ 0, & \omega_c < |\omega| \le \pi \end{cases}$$

and a corresponding impulse response of

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})e^{j\omega n}d\omega = \frac{\omega_c}{\pi}\text{sinc}\left(\frac{\omega_c n}{\pi}\right), -\infty < n < \infty$$

However, no real filter can have this frequency response because $h_d(n)$ is both noncausal and infinite in duration.

One method for creating a realizable approximation to an ideal filter is to truncate this impulse response outside of $n \in [-M, M]$.

$$h(n) = \begin{cases} \frac{\omega_c}{\pi}\text{sinc}\left(\frac{\omega_c n}{\pi}\right), & n = -M, \ldots, 0, 1, \ldots M \\ 0, & otherwise \end{cases}$$

In order to visualize the effect of truncation, we express the truncated impulse response as

$$h(n) = h_d(n)w(n)$$

where, $w(n)$ is a finite duration rectangular window defined as

$$w(n) = \begin{cases} 1, & |n| \le M \\ 0, & otherwise \end{cases}$$

Since the multiplication by the window function in time domain corresponds to a convolution in the DTFT domain, we can easily visualize the spectrum of the designed FIR filter as shown below:

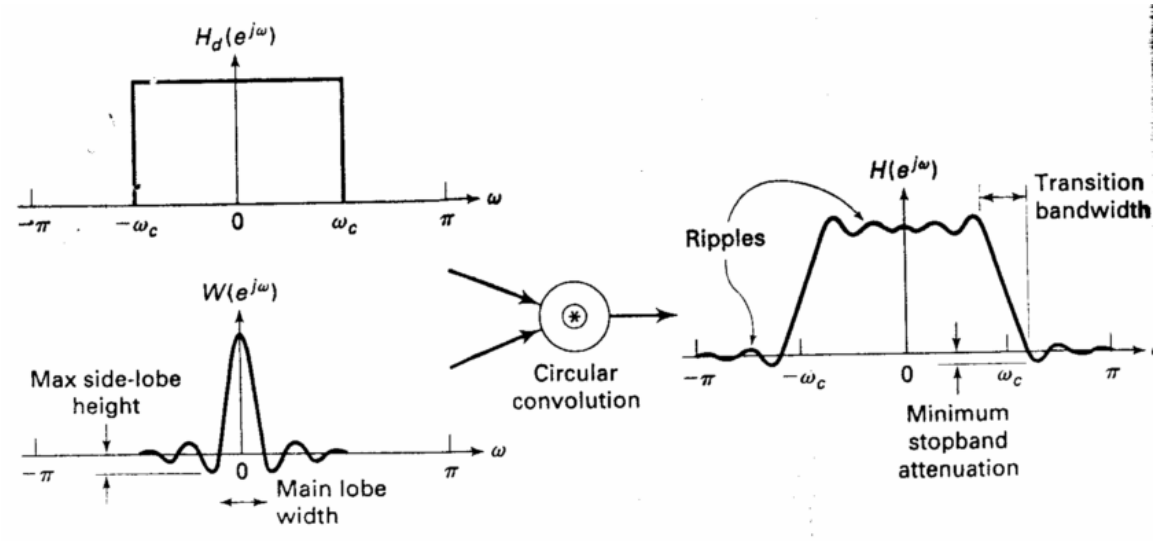


Fig. 2: The periodic convolution produces a smeared version of the ideal impulse response.

A truncated impulse response is of finite duration, yet the filter is still noncausal. In order to make the FIR filter causal, it must be shifted to the right by M units. For a filter of size N = 2M + 1 this shifted and truncated filter is given by

$$h(n) = \begin{cases} \frac{\omega_c}{\pi}\text{sinc}\left(\frac{\omega_c}{\pi}(n - \frac{N-1}{2})\right) & n = 0, 1, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

This time shift of (N − 1)/2 units to the right corresponds to multiplying the frequency response by $e^{-j\omega(N-1)/2}$. It does not affect the magnitude response of the filter, but adds a factor of $-j\omega(N-1)/2$ to the phase response. Such a filter is called linear phase because the phase is a linear function of ω.

It is interesting to see that the filter formula of (5) is valid for N both even and odd. While both of these filters are linear phase, they have different characteristics in the time domain. When N is odd, then the value at n = (N −1)/2 is sampled at the peak of the sinc function, but when N is even, then the two values at n = N/2 and n = (N/2) − 1 straddle the peak.

Filter Design Using Standard Windows

We may generalize the idea of truncation by using different windowing functions to truncate an ideal filter's impulse response. More desirable frequency characteristics can be obtained by making a better selection for the window, w(n). In fact, a variety of raised cosine windows are widely used for this purpose. Some popular windows are listed below.

1. The rectangular window

$$w(n) = \begin{cases} 1, & 0 \le n \le M-1 \\ 0, & otherwise \end{cases}$$

2. Bartlett window

$$w(n) = \begin{cases} \dfrac{2n}{M-1}, & 0 \le n \le \dfrac{M-1}{2} \\ 2 - \dfrac{2n}{M-1}, & \dfrac{M-1}{2} \le n \le M-1 \\ 0, & otherwise \end{cases}$$

3. Hanning window

$$w(n) = \begin{cases} 0.5\left[1 - \cos\left(\dfrac{2\pi n}{M-1}\right)\right], & 0 \le n \le M-1 \\ 0, & otherwise \end{cases}$$

4. Hamming window

$$w(n) = \begin{cases} 0.54 - 0.46\cos\left(\dfrac{2\pi n}{M-1}\right), & 0 \le n \le M-1 \\ 0, & otherwise \end{cases}$$

5. Blackman window

$$w(n) = \begin{cases} 0.42 - 0.5\cos\left(\dfrac{2\pi n}{M-1}\right) + 0.08\cos\left(\dfrac{4\pi n}{M-1}\right), & 0 \le n \le M-1 \\ 0, & otherwise \end{cases}$$

6. Kaiser window

$$w(n) = \begin{cases} \dfrac{I_o\left[\beta\sqrt{1-\left(1-\dfrac{2n}{M-1}\right)^2}\right]}{I_o[\beta]}, & 0 \le n \le M-1 \\ 0, & otherwise \end{cases}$$

where, $I_o[.]$ is the modified zero-order Bessel function, and β controls the shape of the window. Large values of β reduce the window sidelobes and therefore result in reduced passband and stopband ripple.

In filter design using different truncation windows, there are two key frequency domain effects that are important to the design: the transition band roll-off, and the passband and stopband ripple (see Fig. 2). There are two corresponding parameters in the spectrum of each type of window that influence these filter parameters.

- The filter's roll-off is related to the width of center lobe of the window's magnitude spectrum.
- The ripple is influenced by the ratio of the mainlobe amplitude to the first sidelobe amplitude (or difference if using a dB scale).

These two window spectrum parameters are not independent, and one can easily find a trend by examining the spectra for different windows.

The theoretical values for the mainlobe width and the peak-to-sidelobe amplitude are shown in Table 1.

Table 1

Window name	Main lobe width	Peak-to-side lobe amplitude (dB)
Rectangular	$\frac{4\pi}{M}$	-13 dB
Bartlett	$\frac{8\pi}{M}$	-27 dB

Hanning	$\frac{8\pi}{M}$	-32 dB
Hamming	$\frac{8\pi}{M}$	-43 dB
Blackman	$\frac{12\pi}{M}$	-58 dB

Filter Design Using the Kaiser Window

The Kaiser window function of length M is defined as

$$w(n) = \begin{cases} \dfrac{I_o\left[\beta\sqrt{1-\left(1-\dfrac{2n}{M-1}\right)^2}\right]}{I_o[\beta]}, & 0 \le n \le M-1 \\ 0, & otherwise \end{cases}$$

- The Kaiser filter must be designed to meet the smaller of the two ripple constraints:

$$\delta = \min\{\delta_p, \delta_s\}$$

- The values of β and M could be chosen to meet any set of design parameters, $(\delta, \omega_p, \omega_s)$, by defining A = −20 log10 δ and using the following two equations:

$$\beta = \begin{cases} 0.1102(A-8.7) & A > 50 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21) & 21 \le A \le 50 \\ 0.0 & A < 21 \end{cases}$$

$$M = \left\lceil 1 + \frac{A-8}{2.285(\omega_s - \omega_p)} \right\rceil$$

where, ⌈.⌉ is the ceiling function, i.e. ⌈x⌉ is the smallest integer which is greater than or equal to x.

Filter Design Using Parks-McClellan Algorithm

Kaiser windows are versatile since they allow the design of arbitrary filters which meet specific design constraints. However, filters designed with Kaiser windows still have a number of disadvantages. For example,

- Kaiser filters are not guaranteed to be the minimum length filter which meets the design constraints.
- Kaiser filters do not allow passband and stopband ripple to be varied independently.

In 1972, Parks and McClellan devised a methodology for designing symmetric filters that minimize filter length for a particular set of design constraints $\{\omega_p, \omega_s, \delta_p$, and $\delta_s\}$. The resulting filters minimize the maximum error between the desired frequency response and the actual frequency response by spreading the approximation error uniformly over each band. The Parks and McClellan algorithm makes use of the Remez exchange algorithm and Chebyshev approximation theory. Such filters that exhibit equiripple behavior in both the passband and the stopband, and are sometimes called equiripple filters.

Designing a filter with the Parks and McClellan algorithm is a two step process. First the length (i.e. order) of the filter must be computed based on the design constraints. Then the optimal filter for a specified length can be determined. As with Kaiser windows, the filter length computation is approximate so the resulting filter may exceed or violate the design constraints. This is generally not a problem since the filter can be redesigned for different lengths until the constraints are just met. The Matlab command for computing the approximate filter length is

[n, fo, mo, w] = remezord (f, m, ripple, 2*pi)

where the inputs are:

f - vector containing an even number of band edge frequencies. For a simple low pass filter, f = [wp ws], where wp and ws are the passband and stopband frequencies, respectively.

m - vector containing the ideal filter magnitudes of the filter in each band. For a simple low pass filter m = [1 0].

ripple - vector containing the allowed ripple in each band. For a simple low pass filter ripple = [delta_p delta_s], where delta_p and delta_s are the passband and stopband ripples, respectively.

2*pi - value, in radians, that corresponds to the sampling frequency.

The outputs of the command are n = filter length - 1, and the vectors fo, mo, and w which are intermediate filter parameters.

Once the filter length, n, is obtained, the Matlab command for designing a Parks-McClellan filter is b = remez (n, fo, mo, w). The inputs n, fo, mo, and w are the corresponding outputs of remezord, and the output b is a vector of FIR filter coefficients such that

$$H(z) = b(1) + b(2)z^{-1} + \cdots + b(n+1)z^{-n}$$

Labwork

1. Plot the rectangular, Bartlett, Hanning, Hamming, and Blackman window functions of length 21 on a single figure using the subplot command. Then compute and plot the DFT magnitude of each of the five windows. Plot the magnitudes on a decibel scale (i.e., plot 20 log10 $|W(e^{j\omega})|$).

   Hint: Use at least 512 sample points in computing the DFT.

   Measure the null-to-null main lobe width (in rad/sample) and the peak-to-side lobe amplitude (in dB) from the logarithmic magnitude response plot for each window type. The Matlab command zoom is helpful for this. Make a table with these values and the theoretical ones.

2. Plot the Kaiser windows and their DFT magnitudes (in dB) for M = 21 and the following values of β

   β = 0, 1, 5.

3. Write a Matlab program that computes the truncated and shifted impulse response of size N for a low pass filter with a cutoff frequency of ω_c = 2.0. For each of the following filter sizes, plot the magnitude of the filter's DFT in decibels:

   - N = 21
   - N = 101

   Report:
   (a) Submit the plots of the magnitude response for the two filters (not in decibels). On each of the plots, mark the passband, the transition band and the stopband.
   (b) Submit the plots of the magnitude response in decibels for the two filters.
   (c) Explain how the filter size effects the stopband ripple. Why does it have this effect?

4. Design a low pass filter, h(n) using Kaiser window with the following design specifications:

$$\begin{aligned} \omega_p &= 1.8 \\ \omega_c &= 2.0 \\ \omega_s &= 2.2 \\ \delta_p &= 0.05 \\ \delta_s &= 0.005 \end{aligned}$$

Plot the magnitude of the DFT of h(n) for $|\omega| < \pi$. Create three plots in the same figure: one that shows the entire frequency response, and ones that zoom in on the passband and stopband ripple,

respectively. Mark $\delta_p, \delta_s, \omega_p, \omega_s$ on this plot where appropriate. (Do not use a decibel scale on this set of plots.)

Report:

(a) Submit the values of β and M that you computed.
(b) Submit the plot of the filter's magnitude response. Make sure the plot is labeled.
(c) Submit the values of the passband and stopband ripple. Does this filter meet the design specifications?

5. Now design a symmetric FIR filter using remezord and remez in Matlab to meet the design specifications given in step 2.

   Compute the DFT of the filter's response for at least 512 points, and use this result to compute the passband and stopband ripple of the filter that was designed.

   Adjust the filter length until the minimum order which meets the design constraints is found. Plot the magnitude of the DFT in dB of the final filter design.

   Report:

   (a) Submit the final measured values of filter length, passband ripple, and stopband ripple. How accurate was the filter order computation using Matlab's remezord? How does the length of this filter compare to the filter designed using a Kaiser window?

   (b) Submit the plot of the filter's DFT. How does the frequency response of the Parks-McClellan filter compare to the filter designed using the Kaiser window? Comment on the shape of both the passband and stopband.

This laboratory manual is prepared by

Mohammad Ariful Haque

and supervised by

Prof. Md. Kamrul Hasan **August 28, 2007**