**PRACTICAL WORK BOOK**

**For Academic Session Fall 2018**

**Digital Signal Processing (EE-394)**

**For**

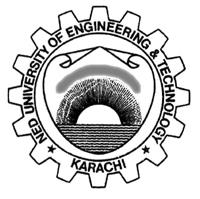
**TE Electrical**

Name: **Muhammad Umer**

Roll Number: **EE-17181** Year: **TE**

Section: **D**

Batch: **2017-18**



**Department of Electrical Engineering**

**N.E.D. University of Engineering & Technology, Karachi**

**CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Lab.**  **No.** | **List of Experiments** | **Page No.** | **Remarks** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |
| 11 |  |  |  |

**Lab Rubrics:**

1. Properly formatted lab document (report writing skill)

2. Understanding of the concepts delivered.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No.** | **Rubric No.** | **Grading** | | | **Score** |
| 1 | 1 | **Exemplary (100%):**  Lab report was well written and properly formatted with good delivery of the required concepts. | **Adequate (70%):**  Lab report was written with some needed improvements | **Poor (40%):**  Poorly written lab content. |  |
| 2 | 2 | **Exemplary (100%):**  Clearly describes the objectives of the lab as well as the concepts learned. | **Adequate (70%):**  Describes the objective and concepts related to lab with some deficiency. | **Poor (40%):**  Inaccurate understanding of the concepts related to lab session. |  |

|  |
| --- |
| **Laboratory Session No. 01** |

**Objective:**

***To get introduced with fundamentals of Digital Signal Processing***

**Post Lab Exercises:**

**Question 1:**

What do you mean by the term digital? Explain it briefly.

**Answer:**

A **digital** signal can only have a set of finite values. If the **digital** signal is \*binary\*, it can only have 2 values, 0 or 1. A PWM signal is an example of a continuous signal (NOT **discrete**), and binary **digital** having just 2 values. A **discrete** signal is one that is defined only at specific time intervals.

**Question 2:**

Write some (at least three) applications of DSP related to electrical (power) engineering.

**Answer:**

* Digital Signal Processing in Power System Protection and Control bridges the gap between the theory of protection and control and the practical applications of protection equipment. Digital Signal Processing in Power System Protection and Control can be useful for protection engineers working in utilities at various levels of the electricity network
* Power conversion systems are composed of different electrical and electronic components that need to be managed. Moreover, such systems are designed to work under different conditions and states; thus, several control algorithms can be found in a single DSP
* Sonar is an application of digital signal processing (DSP); sonar uses sound propagation to navigate and communicate with or detect an object under the surface of the water. Generally, two types of technologies are used in sonar: passive sonar and active sonar technologies. Passive sonar is used to listen to the sound of the vessels; active sonar is used to release pulses of sounds and to listen to echoes. Sonar may also be used for acoustic measurements.

|  |
| --- |
| **Laboratory Session No. 02** |

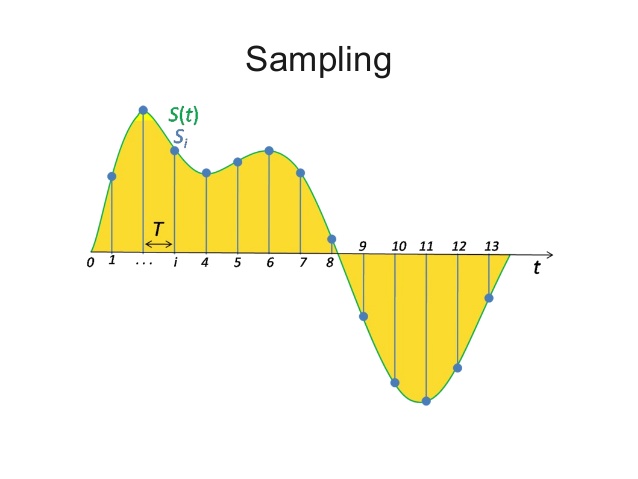
**Objective:**

***An Introduction to Analog to Digital Conversion (Sampling and Aliasing).***

**Post Lab Exercises:**

**Question 1:**

What do you mean by the term “Sampling”? Discuss it briefly with the help of figure.

****

**Answer:**

In digital signal processing, sampling is the reduction of a continuous-time signal to a discrete-time signal. A common example is the conversion of a sound wave (a continuous signal) to a sequence of samples (a discrete-time signal). A sample is a value or set of values at a point in time and/or space.

**Question 2:**

What is Sampling theorem? What do you mean by the term Aliasing?

**Answer:**

A continuous time signal can be represented in its samples and can be recovered back when sampling frequency fs is greater than or equal to the twice the highest frequency component of message signal. i. e.

fs≥2fm.

Aliasing is a phenomenon in which higher frequency signals are mapped into some lower frequency signals. Aliasing can be removed by following Nyquist criteria (Sampling Theorem).The sampling frequency is given by :

Fm – nfs = falias where n is the no. of aliasing

**Question 3:**

Human audible frequency ranges from **20 Hz** to **20 KHz**. Human voice frequency ranges from **4Hz** to **4 KHz**.

**Question 4:**

Record audio for 10sec and complete the following table. Show and verify the output file size through mathematical calculations.

(Hint: Check the Microphone ADC bits and use sampling frequency and the audio record time to evaluate the file size)

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Sampling Frequency(Hz) | File Size  (MB) | Quality  Comment |
| 1 | 44100 | 1.68 | Quality of Sampling is Excellent to hear and there is no Aliasing in this Frequency |
| 2 | 22050 | 0.84 | Quality of Sampling is lesser than above Frequency and there is no Aliasing in this Frequency |
| 3 | 10000 | 0.38 | Quality of Sampling is good and easy to hear voice |
| 4 | 6000 | 0.22 | Quality of Sampling is little good and easy to understand voice |
| 5 | 4000 | 0.152 | Quality of Sampling is little poor |
| 6 | 2000 | 0.076 | Quality of Sampling is quite poor, aliasing is present and difficult to understand voice |
| 7 | 1000 | 0.038 | Quality of Sampling is very poor, aliasing is present and unable to understand voice |

**Answer:**

As we observe in above table that when we decrease the sample rate or frequency, we are difficult or unable to understand the voice. We are requiring lots of focus to understand the voice by decreasing the frequency. When we are unable to understand the voice that’s means that Aliasing is present.

**Calculation of File Size:**

File size can be calculated by the following formula:

𝐵𝑖𝑡𝑠 = 𝑆𝑎𝑚𝑝𝑙𝑖𝑛𝑔 𝐹𝑟𝑒𝑞𝑢𝑒𝑛𝑐𝑦 × 𝑆𝑎𝑚𝑝𝑙𝑒 𝑆𝑖𝑧𝑒 × 𝑇𝑖𝑚𝑒 × 𝐶ℎ𝑎𝑛𝑛𝑒l

|  |  |  |
| --- | --- | --- |
| 𝑭𝒔 = 𝟒𝟒𝟏𝟎𝟎𝑯𝒛  = 44100 × 16 × 10 × 2  = 14112000 𝑏𝑖𝑡𝑠  = 14112000 /8  = 1764000 𝐵  = 1764000/1024  = 1722.656 𝐾𝐵  = 1722.656 /1024  = 1.68 𝑀𝐵 | 𝑭𝒔 = 𝟐𝟐𝟎𝟓𝟎𝑯𝒛  = 22050 × 16 × 10 × 2  = 7056000 𝑏𝑖𝑡𝑠  = 7056000 /8  = 882000 𝐵  = 882000/1024  = 861.328 𝐾𝐵  = 861.328 /1024  = 0.8411 𝑀𝐵 | 𝑭𝒔 = 𝟏𝟎𝟎𝟎𝟎𝑯𝒛  = 10000 × 16 × 10 × 2  = 3200000 𝑏𝑖𝑡𝑠  = 3200000 /8  = 400000 𝐵  = 400000/1024  = 390.625 𝐾𝐵  = 390.625 /1024  = 0.3814 𝑀𝐵 |
| 𝑭𝒔 = 𝟔𝟎𝟎𝟎𝑯𝒛  = 6000 × 16 × 10 × 2  = 1920000 𝑏𝑖𝑡𝑠  = 1920000 /8  = 240000 𝐵  = 240000/1024  = 234.375 𝐾𝐵  = 234.375 /1024  = 0.2288 𝑀𝐵 | 𝑭𝒔 = 𝟒𝟎𝟎𝟎𝑯𝒛  = 4000 × 16 × 10 × 2  = 1280000 𝑏𝑖𝑡𝑠  = 1280000 /8  = 160000 𝐵  = 160000/1024  = 156.25 𝐾𝐵  = 156.25 /1024  = 0.1525𝑀𝐵 | 𝑭𝒔 = 𝟐𝟎𝟎𝟎𝑯𝒛  = 2000 × 16 × 10 × 2  = 640000 𝑏𝑖𝑡𝑠  = 640000 /8  = 80000 𝐵  = 80000/1024  = 78.125 𝐾𝐵  = 78.125 /1024  = 0.0762 𝑀𝐵 |

|  |
| --- |
| 𝑭𝒔 = 𝟏𝟎𝟎𝟎𝑯𝒛 |
| = 1000 × 16 × 10 × 2 |
| = 320000 𝑏𝑖𝑡𝑠 |
| = 320000 /8 |
| = 40000 𝐵 |
| = 40000/1024 |
| = 39.0625 𝐾𝐵 |
| = 39.0625/1024 |
| = 0.03814 𝑀𝐵 |

**Question 5:**

If an ADC has sampling frequency =1000 Hz and receive analog signals of the following frequencies what will be the frequency of a signal which is converted back to analog by a DAC converter?

|  |  |  |  |
| --- | --- | --- | --- |
| **S No** |  | **Frequency (Hz)** | **Frequency of output Signals (Hz)** |
|  | 1 | 100 | 100 |
|  | 2 | 750 | -250 |
|  | 3 | 1250 | 250 |
|  | 4 | 1900 | -100 |
|  | 5 | 2000 | 0 |
|  | 6 | 2500 | 500 |

**Answer:**

## Since 𝑭𝒔 ≥ 𝟐𝒇 OR 𝑭𝒔/𝟐 ≥ 𝒇

Frequencies less than equal to 500hz will remain unaliased while the rest of the signals which are greater than 500hz (750hz,1250hz,1900hz,2000hz,2500hz) will be aliased according Nyquist theorem.

𝑭𝒂𝒍𝒊𝒂𝒔 = 𝑭𝒎𝒂𝒙 − 𝒏𝑭𝒔𝒂𝒎𝒑𝒍𝒊𝒏𝒈

**Calculations: -**

1. No Aliasing in case of 100 Hz signal because 100 Hz < 500 Hz which is the maximum allowable frequency.
2. Falias = 750 − 1000 = −250 Hz
3. Falias = 1250 − 1000 = 250 Hz
4. Falias = 1900 − (2 ∗ 1000) = −100 Hz
5. Falias = 2000 − (2 ∗ 1000) = 0 Hz
6. Falias = 2500 − (2 ∗ 1000) = 500 Hz

|  |
| --- |
| **Laboratory Session No. 03** |

**Objective:**

***An Introduction to Analog to Digital Conversion (Quantization and Coding).***

**Matlab Code for the Analysis of Quantization:**

%% lab3

fprintf('\Lab #03, Quantization');

b=3; % Number of bits

N=120 ; % Number of samples in final signal

n=0:(N-1); %Index

%choose the input type

choice=questdlg('Choose input','Input','Sawtooth','Sine','Random','Random' );

fprintf('Bits = %g , levels = %g , signal= %s.\n',b,2^b,choice);

%Create the input data sequence

switch choice

case 'Sine'

x=sin(2\*pi\*n/N);

case 'Sawtooth'

x=sawtooth(2\*pi\*n/N);

case 'Random'

x=randn(1,N); %Random data

x=x/max(abs(x)); %scale to +/- 1

end

%signal is restricted to between -1 and +1

x(x>=+1)=(1-eps); %make signal from -1 to just less than 1

x(x<-1)=-1;

%Quantize a signal to "b" bits

xq=floor((x+1)\*2^(b-1)); %signal is one of 2^n int values (0 to 2^n-1)

xq=xq/(2^(b-1)); %signal is from 0 to 2 (quantized)

xq=xq-(2^(b)-1)/2^(b); %shift signal down (rounding)

xe=x-xq; %quantization error

stem(x,'b');

hold on;

stem(xq,'r');

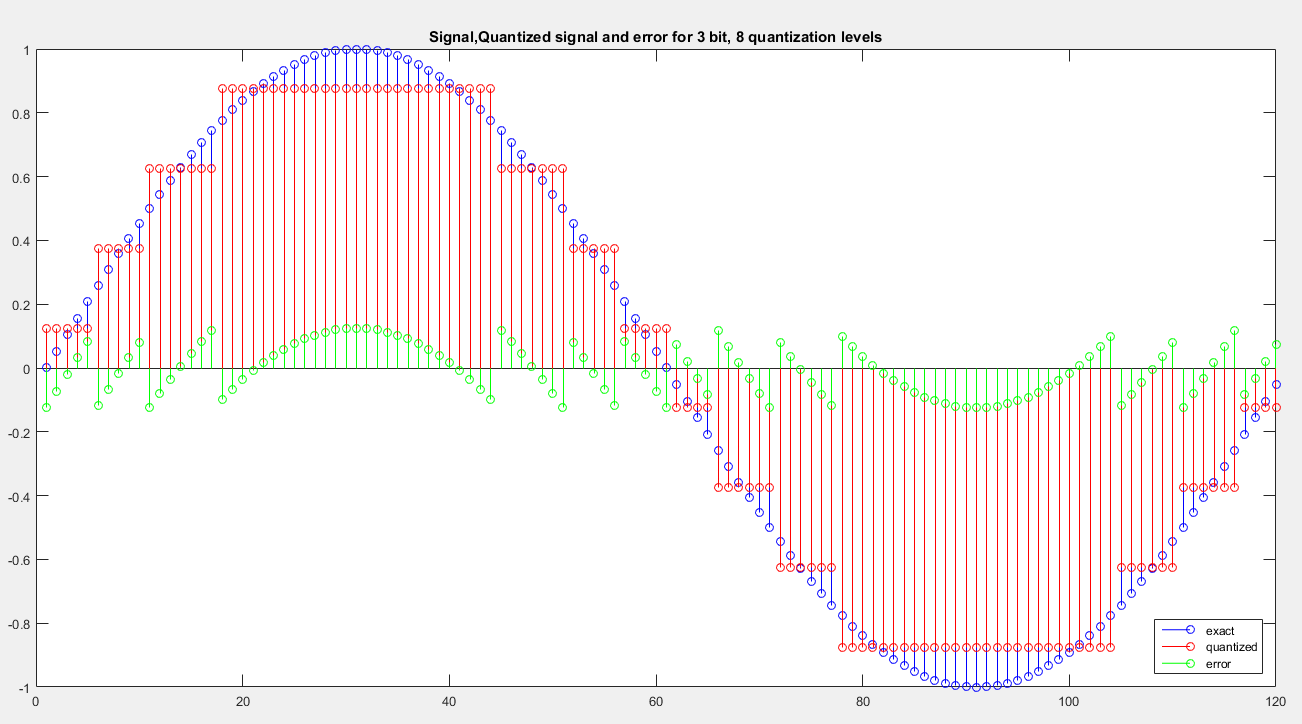
hold on

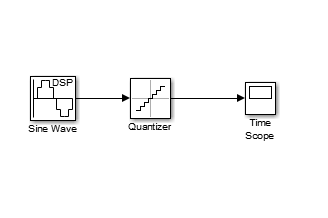
stem(xe,'g');

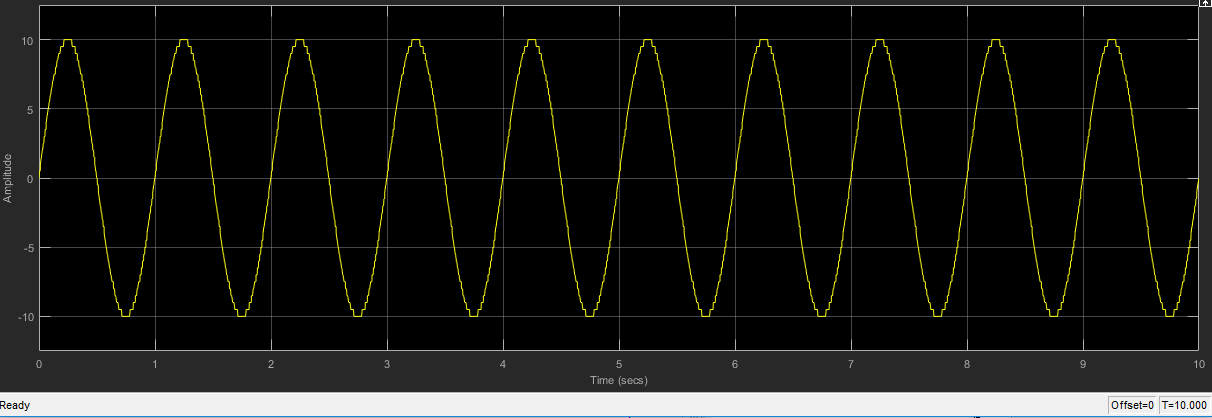
legend('exact','quantized','error','Location','Southeast')

title(sprintf('Signal,Quantized signal and error for %g bit, %g quantization levels',b,2^b));

hold off

**Output**

**Analysis of Quantization with DSP System toolbox in Simulink**



**Post Lab Exercises:**

**Question 1:**

Why Quantization is needed in Digital Signal Processing?

**Answer:**

Quantization, in mathematics and [digital signal processing](https://en.wikipedia.org/wiki/Digital_signal_processing), is the process of mapping input values from a large set (often a continuous set) to output values in a (countable) smaller set, often with a finite [number of elements](https://en.wikipedia.org/wiki/Cardinality).

The process of converting a discrete time continuous amplitude signal into a digital signal by expressing each sample value as a finite (instead of an infinite) number of digits is called quantization, The error induced in representing the continuous valued signal by a finite set of discrete value levels is called quantization error or quantization noise.

**Question 2:**

What is Anti-Aliasing Filter? Discuss it with some example.

**Answer:**

If we do not follow Nyquist criteria, our signal contains some aliased frequencies. These aliasing components are occurred due to the passing of unnecessary high frequency signal from a sampler. In order to avoid those high frequency signals we placed an anti-aliasing filter before the sampler which blocks the high frequency components.

For Example: This anti-aliasing filter is generally a low-pass filter which allows low frequency components and block the higher one.

**Question 3:**

Discuss the specifications of Arduino Uno ADC like number of bits, number of quantization levels, etc. Also, calculate the default Arduino Uno ADC resolution.

Link: <https://store.arduino.cc/usa/arduino-uno-rev3>

**Answer:**

The **Arduino Uno ADC** is of 10 bit **resolution** (so the integer values from (0-(2^10) 1023)) which means total quantization levels are 1024. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. So for every (5/1023= 4.9mV) per unit.

**Question 4:**

A 12-bit ADC has input values in the range of 0 – 1 V. Calculate the resolution of ADC.

**Answer:**

Since No of levels (L) = 212 = 4096 levels

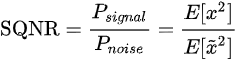
So resolution = 1-0/4095 = 0.24 mV

**Question 5:**

What do you mean by the term SQNR and ENOB? Discuss it briefly.

**Answer**

Signal-to-Quantization-Noise Ratio (SQNR or SNqR) is widely used quality measure in analysing [digitizing](https://en.wikipedia.org/wiki/Digitizing) schemes such as PCM ([pulse code modulation](https://en.wikipedia.org/wiki/Pulse-code_modulation)) and [multimedia codecs](https://en.wikipedia.org/wiki/Multimedia_codec). The SQNR reflects the relationship between the maximum nominal [signal strength](https://en.wikipedia.org/wiki/Signal_strength) and the [quantization error](https://en.wikipedia.org/wiki/Quantization_error) (also known as quantization noise) introduced in the [analog-to-digital conversion](https://en.wikipedia.org/wiki/Analog-to-digital_conversion). As SQNR, like SNR, is a ratio of signal power to some noise power, it can be calculated as:

{\displaystyle \mathrm {SQNR} ={\frac {P\_{signal}}{P\_{noise}}}={\frac {E[x^{2}]}{E[{\tilde {x}}^{2}]}}}

Effective number of bits (ENOB) is a measure of the [dynamic range](https://en.wikipedia.org/wiki/Dynamic_range) of an [analog-to-digital converter](https://en.wikipedia.org/wiki/Analog-to-digital_converter) (ADC), [digital-to-analog converter](https://en.wikipedia.org/wiki/Digital-to-analog_converter), or their associated circuitry. The resolution of an ADC is specified by the number of [bits](https://en.wikipedia.org/wiki/Bit) used to represent the analog value. Ideally, a 12-bit ADC will have an effective number of bits of almost 12. However, real signals have noise, and real circuits are imperfect and introduce additional [noise](https://en.wikipedia.org/wiki/Noise_(electronics)) and [distortion](https://en.wikipedia.org/wiki/Distortion). Those imperfections reduce the number of bits of accuracy in the ADC. The ENOB describes the effective resolution of the system in bits. An ADC may have 12-bit resolution, but the effective number of bits when used in a system may be 9.5.

|  |
| --- |
| **Laboratory Session No. 04** |

**Objective:**

***An Introduction to Time shifting, Reversal and scaling.***

**Post Lab Exercises:**

**Question 1:**

Let x(n)=[**1** 1 2 3 -1 -1 2 5 6 ]

Using Matlab plot the following signals.

1. x(n-3)
2. x(n+1)
3. x(-n)
4. x(2n)
5. x(n/3)
6. x(-n+1)
7. x(-n-1)

also show every step and mention operations i.e. time shifting, time scaling etc.

**Answer:**

%% lab3 shifting,flipping,scaling

x=[1 1 2 3 -1 -1 2 5 6 ];

l=length(x);

n=0:l-1;

subplot(4,2,1);

stem(n,x); %original signal

title('x(n)')

subplot(4,2,2);

stem(n+3,x); %shifting

title('x(n-3)')

subplot(4,2,3);

stem(n-1,x); %shifting

title('x(n+1)')

subplot(4,2,4);

xf1=flip(x); %flipping

stem(-flip(n),xf1);

title('x(-n)')

subplot(4,2,5);

n1=(0/2):(1/2):(l-1)/2; %scaling

stem(n1,x);

title('x(2n)')

subplot(4,2,6);

n2=(0\*3):1\*3:(l-1)\*3; %scaling

stem(n2,x);

title('x(n/3)')

subplot(4,2,7);

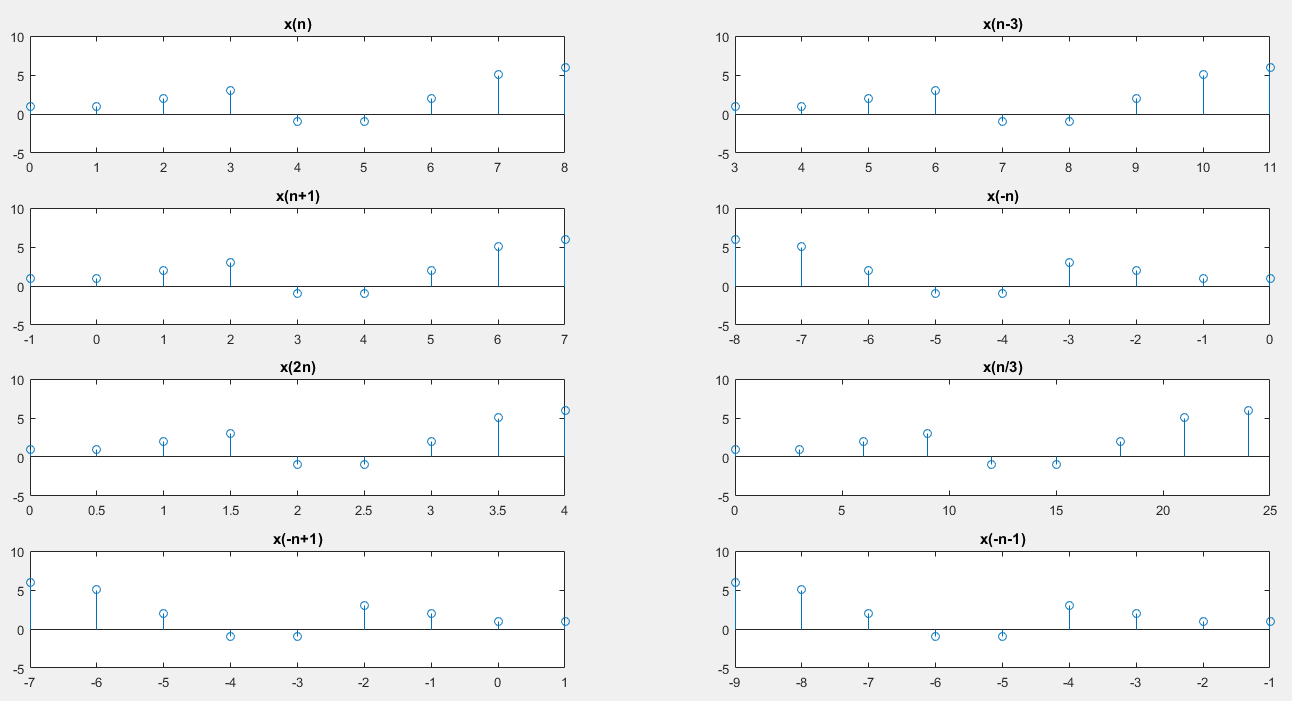
nf=-flip(n); %x(-n)

stem(nf+1,xf1); %x(-n+1)

title('x(-n+1)')

subplot(4,2,8);

stem(nf-1,xf1); %x(-n-1)

title('x(-n-1)');

**Question 2:**

let x(n)= [-2 2 1 -1 **3** 2 +2 -3 -1 5 0 1]

using Matlab plot the following signals.

1. x(n+2)
2. x(-n+2)
3. x(2n)
4. x(2n+3)

also show every step and mention operations i.e. time shifting, time scaling etc.

(Attach codes at the end of this lab)

**Answer:**

%% lab3 task2

x=[-2 2 1 -1 3 2 +2 -3 -1 5 0 1];

l=length(x);

n=-4:7;

subplot(2,3,1);

stem(n,x); grid on % original signal x(n)

title('x(n)')

subplot(2,3,2);

stem(n-2,x); grid on %x(n+2)

title('x(n+2)')

subplot(2,3,3);

xf=flip(x);

nf=-flip(n);

stem(nf,xf); grid on %x(-n)

title('x(-n)')

subplot(2,3,4);

stem(nf+2,xf); grid on %x(-n+2)

title('x(-n+2)')

subplot(2,3,5);

ns=n/2; %scaling by factor of 2

stem(ns,x); grid on %x(2n)

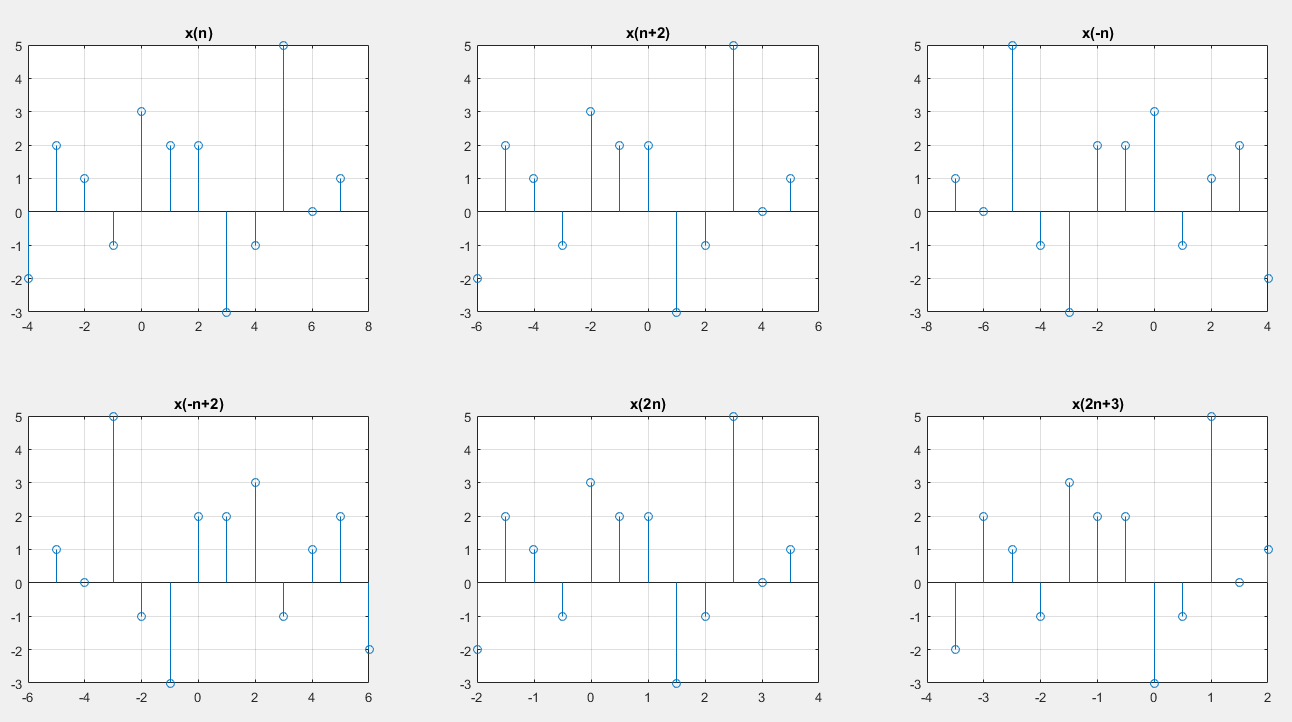
title('x(2n)')

subplot(2,3,6);

ns1=(n-3)/2;

stem(ns1,x); grid on %x(2n+3)

title('x(2n+3)')



|  |
| --- |
| **Laboratory Session No. 05**  **(Open Ended Lab)** |

**Objective:**

***To convert an analog (voltage and current) signal into digital signal using ADC (audio card) and display it on MATLAB Simulink environment.***

**Required Components:**

1. Audio Card
2. Transformer (220V/12V)
3. Resistors (for VDR)
4. Veroboard
5. Audio jack
6. PC with MATLAB environment

**Procedure:**

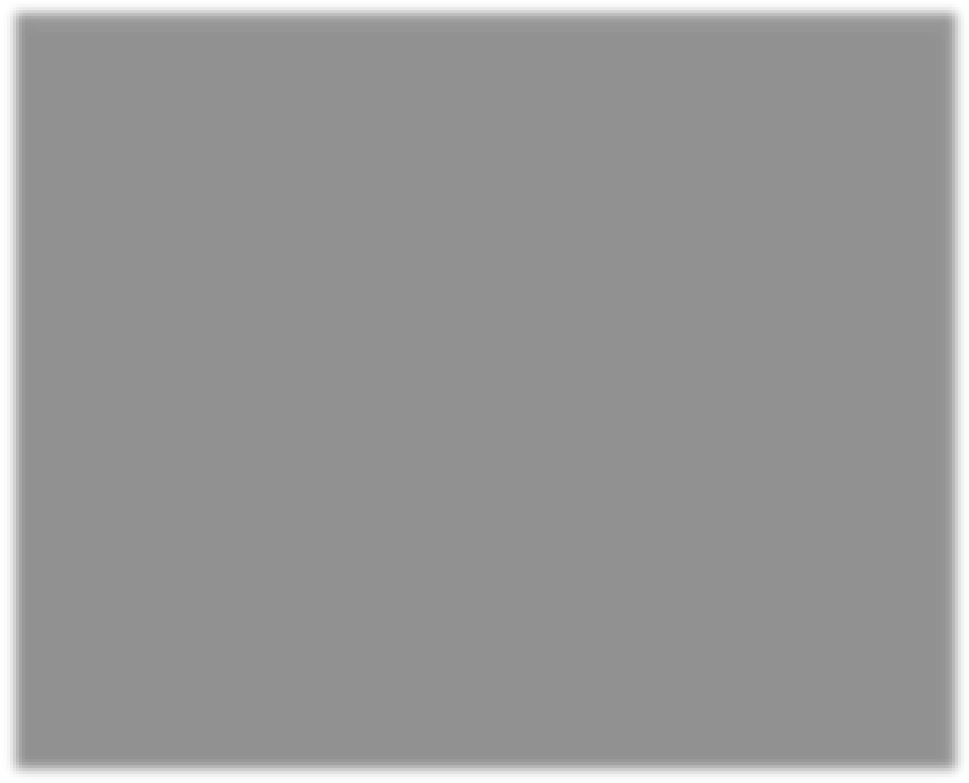
* Using Transformer convert 220VAC from mains into 12VAC.
* Using VDR convert 12VAC to a voltage compatible to audio card (show all the calculations of resistances with their power ratings).
* Set the sampling frequency of the audio card ADC in MATLAB Simulink environment with proper justification
* Plot the acquired voltage waveform to Simulink scope.
* Mention the safe operating range of your equipment.

**Working**

* The basic concept of this lab was to observe and simulate the AC voltage on Matlab.
* For this purpose an external sound card is used.
* The voltage rating of the soundcard is 1V and its current rating is 1mA
* Voltage Measurements have been taken across the load Resistor. In order to do so, secondary of PT is connected to series potentiometer and load. Potentiometer is so adjusted that the voltage across the load will be in 1V range.
* One end of the aux cable is connected in parallel to load resistor and its other end is connected to the microphone input of the external sound card.
* For measuring current CT is placed in series with the load and its secondary is connected to the microphone
* The soundcard come with a built in ADC which converts the analog wave form to discrete wave form using a sampling frequency of 44.1 kHz.
* A sinusoid was observed but with amplitude different than the input applied from AC source.
* The gain block is then added on Simulink and the output is observed with parameters of wave almost similar to that observed practically in oscilloscope.
* A gain factor is applied to scale the wave form to the actual AC (i.e. 220 Vrms)

# SPECIFICATION OF COMPONENTS USED

**DIGITAL AUDIO SOUND CARD:**



* **Model number:** U237-001
* **Operating Temperature Range:** 32 to 104 F( 0 to 40 C)
* **Storage Temperature Range:** 14 to 131 F (0 to 40C)
* **Rated voltage:** 1V
* **Rated Current:** 1mA
* **Power Consumption (Watts) :** 0.5W
* **Ports :** 2
* **Side A- Connector 1:**USB A(MALE)
* **Side B- Connector 2:** (2)3.5mm (FEMALE)
* **Connector Plating:** Nickel

**CURRENT TRANSFORMER:**



* **Model number:** SCT-013-005
* **Application Range:** Measuring
* **Kind:** Current measurement for protection of AC motors
* **Rated voltage:** 1V
* **Package Form:** Plug-in
* **Working Principle:** Electromagnetic Induction
* **Working Temperature:** - 25 – 70 Degree Celsius
* **Quality Factor:** 98
* **Allowable error:** +-20%
* **Inductive XL:** 10 (Ω)

 **Rated Current:** 5Amps

 **Distributed capacitance:** 10 (F)

 **Input Voltage:** 220V (50/60Hz) (V)

**CALCULATIONS**

**V.D.R Used :**

**Series resitance = 65k ohms ; Load resitance = 1K ohms**

**No of bits** = 16 (**bits of sound card set in P.C )**

**Sampling Frequency** = 𝐹𝑠 = 44,100𝐻𝑧/44.1𝐾 𝐻𝑧

𝐍𝐎 𝐎𝐅 𝑳𝒆𝒗𝒆𝒍 = 𝑙 = 2# 𝑜𝑓 𝑏𝑖𝑡𝑠 = 216 = 65,536 𝑙𝑒𝑣𝑒𝑙𝑠 (𝑸𝒖𝒂𝒏𝒕𝒊𝒛𝒂𝒕𝒊𝒐𝒏 𝒍𝒆𝒗𝒆𝒍)

Total memory consumed: 𝑆𝑎𝑚𝑝𝑙𝑖𝑛𝑔 𝐹𝑟𝑒𝑞𝑢𝑒𝑛𝑐𝑦 × 𝑆𝑎𝑚𝑝𝑙𝑒 𝑆𝑖𝑧𝑒 × 𝑇𝑖𝑚𝑒×𝐶ℎ𝑎𝑛𝑛𝑒l

44100 × 16 × 10 × 2

= 14112000 𝑏𝑖𝑡𝑠

= 14112000 /8

= 1764000 𝐵

= 1764000/1024

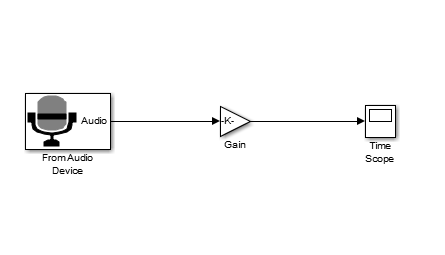
= 1722.656 𝐾𝐵

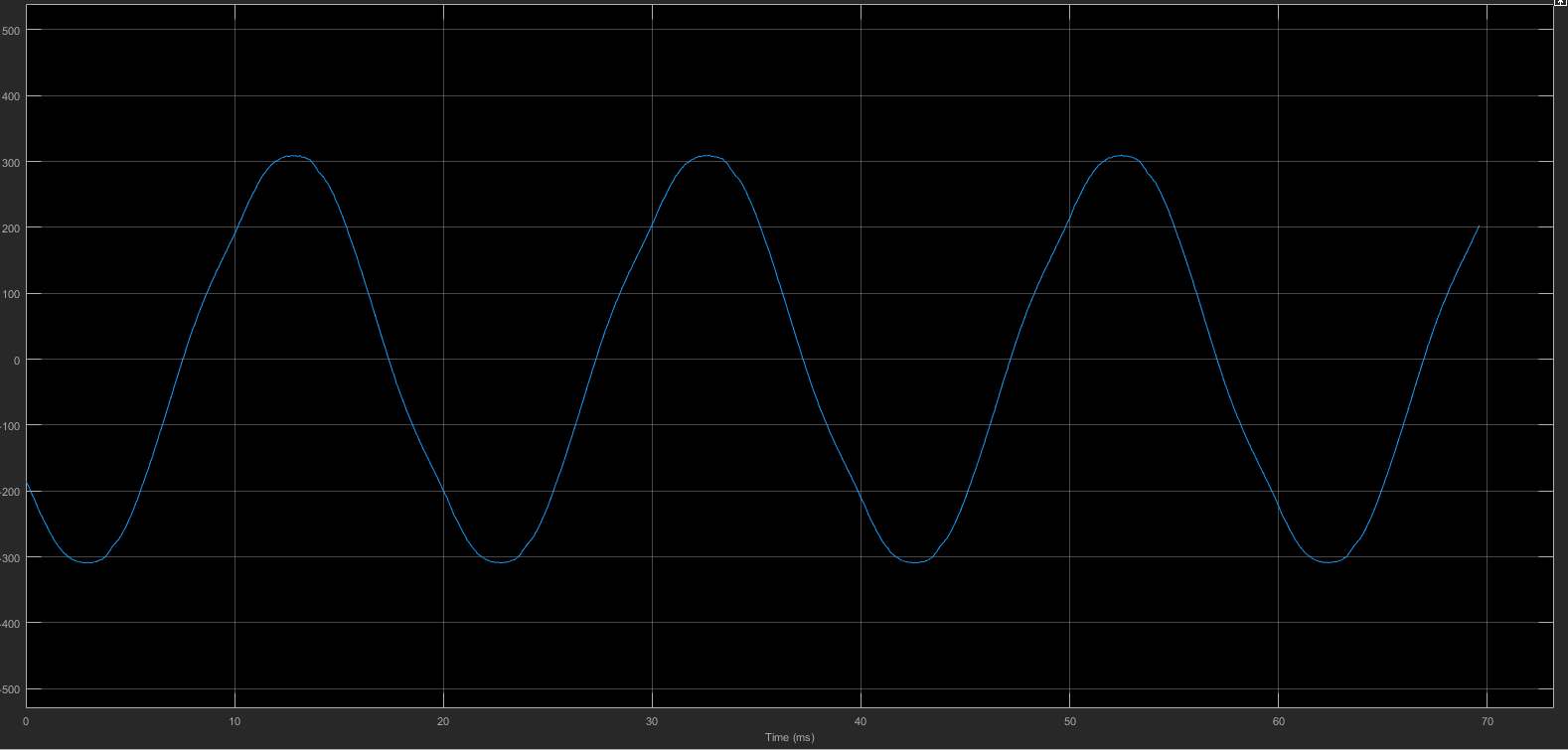
= 1722.656 /1024

= 1.68 𝑀𝐵

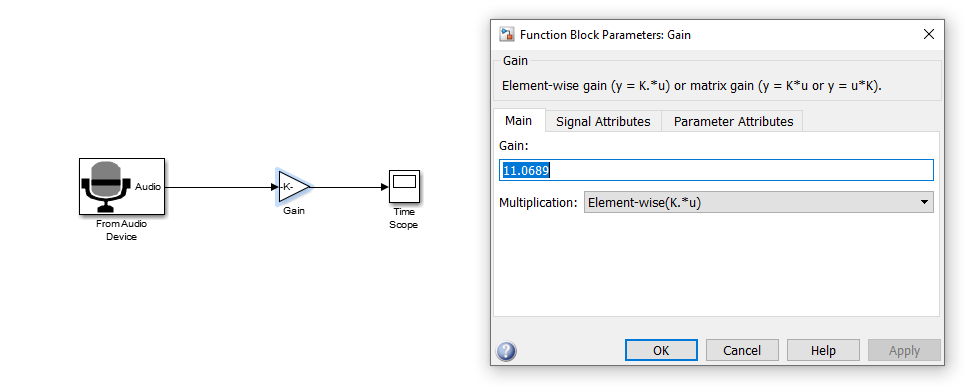
**MATLAB SIMULATION:**

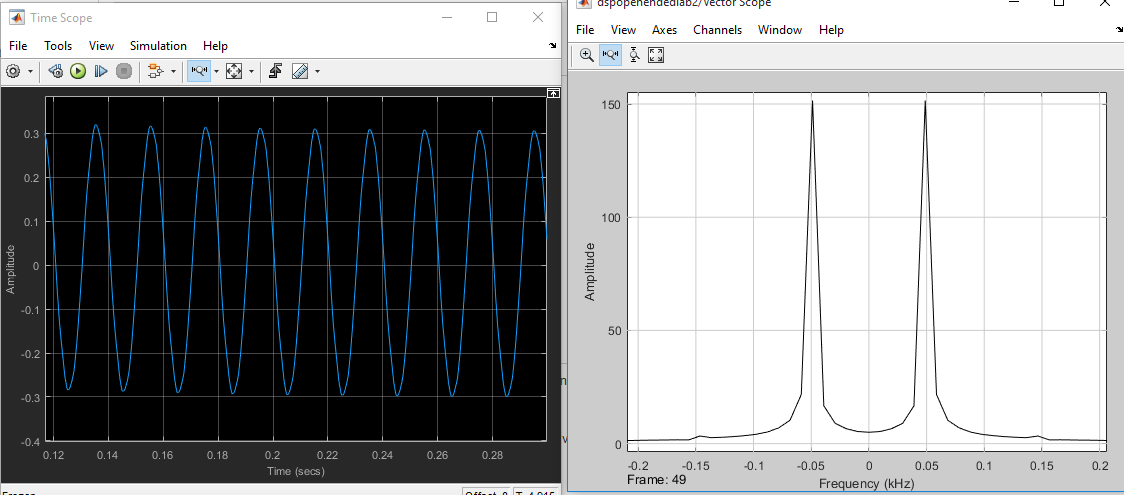
**Voltage waveform**

****

****

The input sinusoidal voltage signal having PK-PK value of 311V was simulated and digitized through an ADC (sound card) and observed on Matlab.

**Current Waveform**

****

**Conclusion:**

This lab demonstrate the conversion of Analog signal into Digital signal in the form of voltage and current, in simple words, we are converting Analog Electrical quantities into digital Electrical quantities because if we have to perform any processing techniques or want to do any mathematical calculations, we can do it in the form of software (digitized way) more simply than in the analog form by making our own hardware circuit which is quite expensive also as well as we can do data acquisition also which cannot be achieved in Analog domain

|  |
| --- |
| **Laboratory Session No. 06** |

**Objective:**

***To generate a square wave in time domain of specific time period and pulse width and to apply CTFS equation to compute the spectral coefficients using MATLAB.***

**Pre Lab Exercises:**

**Types of signals:**

• Continuous-Time Periodic Signals

• Discrete-Time Periodic Signals

• Continuous-Time aperiodic Signals

• Discrete-Time aperiodic Signals

**Mathematical tool for the analysis of Signals and Systems:**

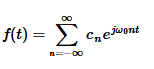
• Fourier series: *preferred for Signal Analysis*

• Laplace transform: *preferred for System Analysis*

• Z transform: *preferred for converting discrete time domain signal into discrete frequency domain signal.*

**Continuous Time Fourier series**

In this lab our objective is to study continous time fourier series. The continuous-time Fourier series expresses a periodic signal as a linear combination of harmonically related complex exponentials. Alternatively, it can be expressed in the form of a linear combination of sines and cosines or sinusoids of different phase angles.The synthesis equation fourier can be written as



Where as the analysis equation is given as



**In Lab Exercises:**

Continuous time Fourier series Analysis of a Sinusoidal signal

Ts=0.001;

Tp=2;

t=[0:Ts:Tp-Ts];

x=2+sin(2\*pi\*1/Tp\*t)+sin(2\*pi\*5/Tp\*t);

subplot(2,1,1), plot(t,x);

xlabel('time(EE-181)');

Fo=1/Tp;

for k=1:20

B=exp(-j\*2\*pi\*(k-1)\*Fo.\*t);

C(k)=sum(x.\*B)/(length(x));

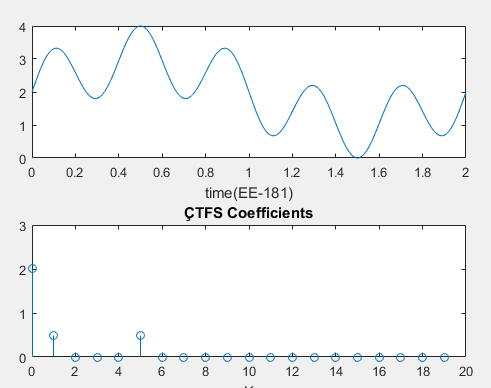
end

k=[0:k-1];

subplot(2,1,2), stem(k,abs(C));

title('ÇTFS Coefficients');

xlabel('K');

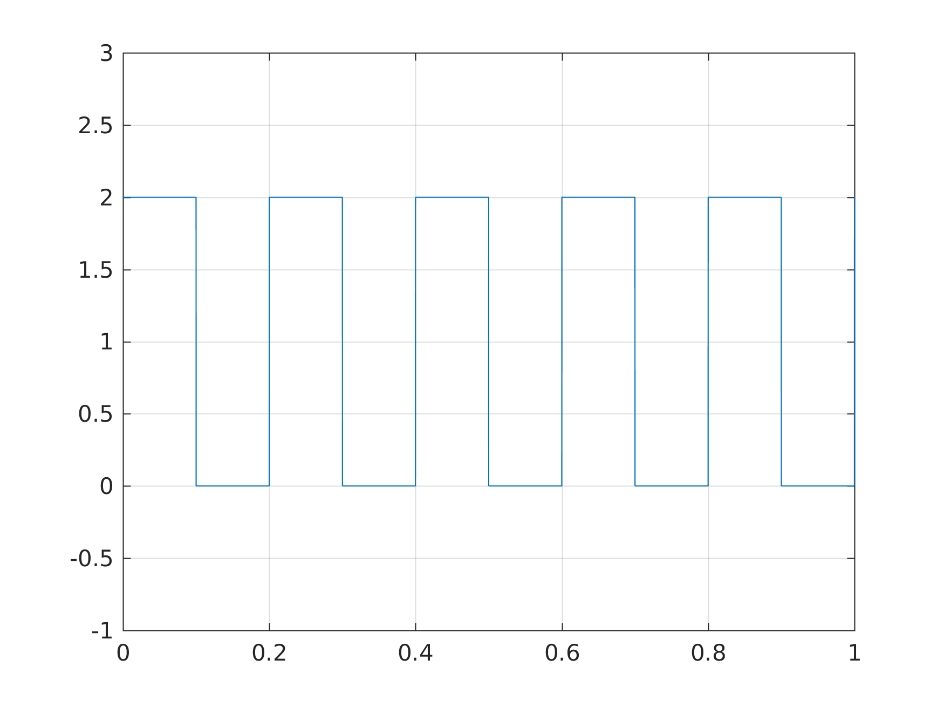


**Post Lab Exercises:**

**Question 1:**

Generate a square wave of magnitude 2 units with duty cycle of 50 % and have a frequency of 5 Hz using MATLAB *square()* function and then plot the frequency spectrum (magnitude vs frequency plot) of the resulting wave using MATLAB script (Evaluate five spectral coefficients). Also, verify your MATLAB results using hand calculations.

Note: Use MATLAB help to get the syntax of the *square()* function.



Ts=0.001;

Tp=1/5;

t=[0:Ts:Tp-Ts];

x=2\*square(2\*pi\*1/Tp\*t);

subplot(2,1,1), plot(t,x) ;

xlabel('time(EE-181)');

Fo=1/Tp;

for k=1:20

B=exp(-j\*2\*pi\*(k-1)\*Fo.\*t);

C(k)=sum(x.\*B)/(length(x));

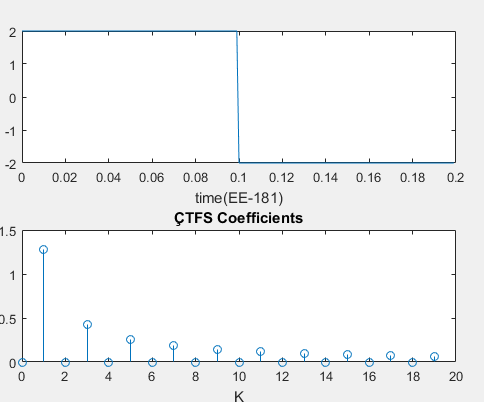
end

k=[0:k-1];

subplot(2,1,2), stem(k,abs(C));

title('ÇTFS Coefficients');

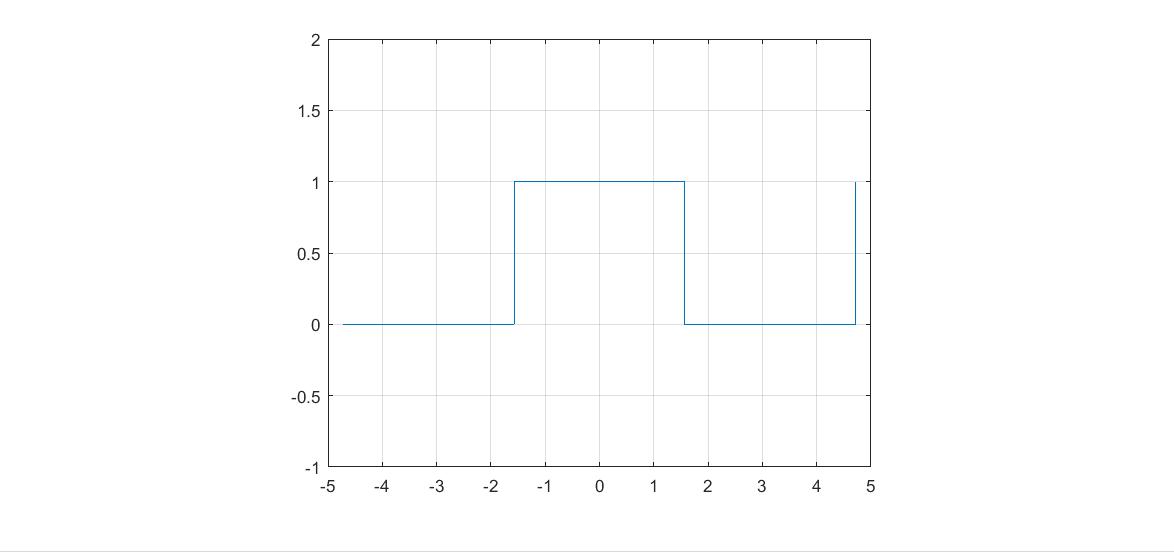
xlabel('K');



**Question 2:**

Verify Fourier series by adding harmonically related cosines and make square wave

Consider the given signal



This signal has time period of To=2 sec

And fundamental frequency is 2 rad/sec

After analyzing the above signal for one period , fourier analysis yields following result



Let's add individual component for one-period of time and see whether they are reproducing our actual time domain plot or not !

t=0:0.01:10;

X\_0=1/2;

X\_X=X\_0;

start=1;

for n=2:1000

if rem(n,2)==0

sign=1;

else

sign=-1;

end

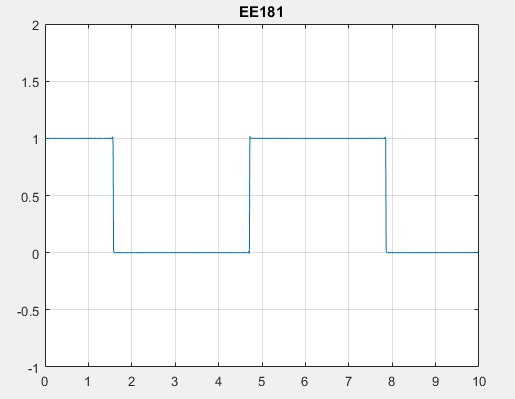
X\_X=X\_X+(2/pi)\*sign\*(1/start)\*cos(start\*t);

start=start+2;

plot(t,X\_X) ,ylim([-1,2]) , grid on;

title('EE181')

end



**Difference between Single Sided and Double Sided Spectrum**

Most frequency analysis instruments display only the positive half of the frequency spectrum because the spectrum of a real-world signal is symmetrical around DC. Thus, the negative frequency information is redundant. The two-sided results from the analysis functions include the positive half of the spectrum followed by the negative half of the spectrum.

A two-sided power spectrum displays half the energy at the positive frequency and half the energy at the negative frequency. Therefore, to convert a two-sided spectrum to a single-sided spectrum, you discard the second half of the array and multiply every point except for DC by two

|  |
| --- |
| **Laboratory Session No. 07** |

**Objective:**

***Analysis and Synthesis of Signals through Discrete Fourier Transform (DFT)***

**PRE LAB TASK:-**

***DISCRETE FOURIER TRANSFORM:-***

In [mathematics](https://en.wikipedia.org/wiki/Mathematics), the discrete Fourier transform (DFT) converts a finite sequence of equally-spaced [samples](https://en.wikipedia.org/wiki/Sampling_(signal_processing)) of a [function](https://en.wikipedia.org/wiki/Function_(mathematics)) into a same-length sequence of equally-spaced samples of the [discrete-time Fourier transform](https://en.wikipedia.org/wiki/Discrete-time_Fourier_transform) (DTFT), which is a [complex-valued](https://en.wikipedia.org/wiki/Complex_number) function of frequency. The interval at which the DTFT is sampled is the reciprocal of the duration of the input sequence. An inverse DFT is a [Fourier series](https://en.wikipedia.org/wiki/Fourier_series), using the DTFT samples as coefficients of [complex](https://en.wikipedia.org/wiki/Complex_number) [sinusoids](https://en.wikipedia.org/wiki/Sine_wave) at the corresponding DTFT frequencies. It has the same sample-values as the original input sequence. The DFT is therefore said to be a [frequency domain](https://en.wikipedia.org/wiki/Frequency_domain) representation of the original input sequence. If the original sequence spans all the non-zero values of a function, its DTFT is continuous (and periodic), and the DFT provides discrete samples of one cycle. If the original sequence is one cycle of a periodic function, the DFT provides all the non-zero values of one DTFT cycle.

The DFT is the most important [discrete transform](https://en.wikipedia.org/wiki/Discrete_transform), used to perform [Fourier analysis](https://en.wikipedia.org/wiki/Fourier_analysis) in many practical applications. In [digital signal processing](https://en.wikipedia.org/wiki/Digital_signal_processing), the function is any quantity or [signal](https://en.wikipedia.org/wiki/Signal_(information_theory)) that varies over time, such as the pressure of a [sound wave](https://en.wikipedia.org/wiki/Sound_wave), a [radio](https://en.wikipedia.org/wiki/Radio) signal, or daily [temperature](https://en.wikipedia.org/wiki/Temperature) readings, sampled over a finite time interval (often defined by a [window function](https://en.wikipedia.org/wiki/Window_function)). In [image processing](https://en.wikipedia.org/wiki/Image_processing), the samples can be the values of [pixels](https://en.wikipedia.org/wiki/Pixel) along a row or column of a [raster image](https://en.wikipedia.org/wiki/Raster_image). The DFT is also used to efficiently solve [partial differential equations](https://en.wikipedia.org/wiki/Partial_differential_equations), and to perform other operations such as [convolutions](https://en.wikipedia.org/wiki/Convolution) or multiplying large integers.

The Analysis Equation of DTFT is given below:-

The Synthesis Equation of DTFT is given below:-

Where,

= kth DFT output component

= index of DFT output in domain

= input sequence

= time domain index of input sequence

= no. of points in DFT output

**In Lab Exercise**

**MATLAB Code (DFT): -**

%% DFT

t=0:.00001:10e-3;

xt=sin(2\*pi\*1000\*t)+0.5\*sin(2\*pi\*2000\*t+(3\*pi/4));

N=8; %length of dft

n=0:N-1;

Fs=8000;

Ts=1/Fs;

xn=sin(2\*pi\*1000\*n\*Ts)+0.5\*sin(2\*pi\*2000\*n\*Ts+(3\*pi/4));

t1=n.\*Ts;

%code block to plot the input sequence

subplot(311);

plot(t,xt,'r');

hold on

stem(t1,xn,'filled');

xlabel('Time(sec)')

ylabel('amplitude')

grid

xk=zeros(1,N);% initialize an array of same size as that of input

%code block to find DFT of the sequence

for k=0:N-1

for n=0:N-1

xk(k+1)=xk(k+1)+(xn(n+1)\*exp((-j)\*2\*pi\*k\*n/N));

end

end

%tic

%ft=fft(xn);

%toc

magnitude=abs(xk); %find magnitude of individual dft points

%code block to plot magnitude spectrum

k=0:N-1;

Df=Fs/N;

Fk=k.\*Df; %analysis frequencies

subplot(312);

stem(Fk,magnitude,'filled','r');

xlabel('frequency(Hz)');

ylabel('magnitude');

xlim([0 Fs]);

title('magnitude spectrum');

grid

phase=angle(round(xk)); %find the phases of individual DFT point

phase=(phase.\*180)/pi;

subplot(313)

stem(Fk,phase,'filled','r');

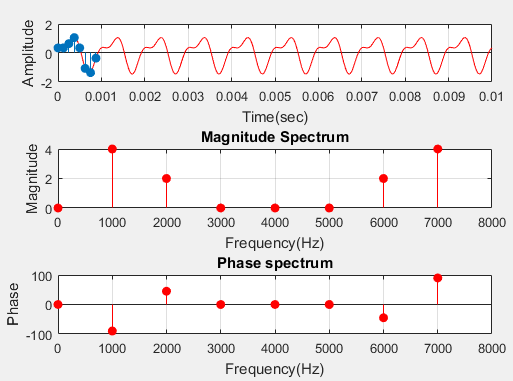
xlabel('Frequency(Hz)');

ylabel('Phase');

xlim([0 Fs]);

title('phase spectrum');

grid

**OUTPUT: -**

**Post Lab Exercises:**

**Question 1:**

Discuss significance of “N” in N point DFT.

**Answer:**

The number of samples in the time domain is called as “N”

* 1. The significance of **N** are as follow
  2. The spectral content will be decided according to **N**.
  3. **N** determines the number of samples in an output.

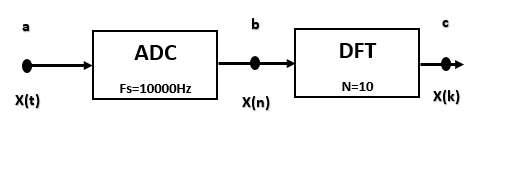
**Question 2:**

DFT can deal both **Periodic**  and **Aperiodic** signals.

**Question 3:**

Consider the following figure, if this system receives input then, plot outputs at point *“b”* and *“c”*.

(Note: Both magnitude and phase plot is required at point *“c”*)



**Answer:**

t=0:.00001:10e-3;

xt = sin(2\*pi\*1000\*t)+sin(2\*pi\*3000\*t)+cos(2\*pi\*0\*t)+cos(2\*pi\*2000\*t);

N=8; %length of dft

n=0:N-1;

Fs=8000;

Ts=1/Fs;

xn = sin(2\*pi\*1000\*n\*Ts)+sin(2\*pi\*3000\*n\*Ts)+cos(2\*pi\*0\*n\*Ts)+cos(2\*pi\*2000\*n\*Ts);

t1=n.\*Ts;

%code block to plot the input sequence

subplot(311);

plot(t,xt,'r');

hold on

stem(t1,xn,'filled');

xlabel('Time(sec)')

ylabel('amplitude')

grid

xk=zeros(1,N);% initialize an array of same size as that of input

%code block to find DFT of the sequence

for k=0:N-1

for n=0:N-1

xk(k+1)=xk(k+1)+(xn(n+1)\*exp((-j)\*2\*pi\*k\*n/N));

end

end

%tic

%ft=fft(xn);

%toc

magnitude=abs(xk); %find magnitude of individual dft points

%code block to plot magnitude spectrum

k=0:N-1;

Df=Fs/N;

Fk=k.\*Df; %analysis frequencies

subplot(312);

stem(Fk,magnitude,'filled','r');

xlabel('frequency(Hz)');

ylabel('magnitude');

xlim([0 Fs]);

title('magnitude spectrum');

grid

phase=angle(round(xk)); %find the phases of individual DFT point

phase=(phase.\*180)/pi;

subplot(313)

stem(Fk,phase,'filled','r');

xlabel('Frequency(Hz)');

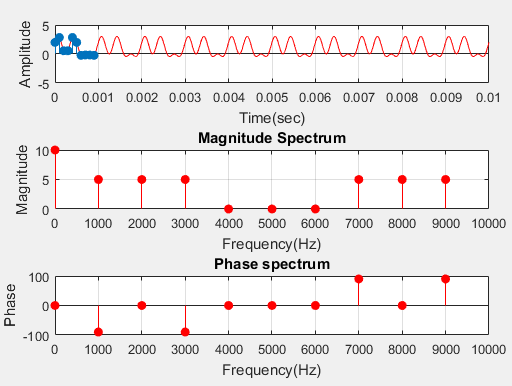
ylabel('Phase');

xlim([0 Fs]);

title('phase spectrum');

grid

**Output**

****

|  |
| --- |
| **Laboratory Session No. 08** |

**Objective:**

***Analysis and Synthesis of Signals through Fast Fourier Transform (FFT)***

***FAST FOURIER TRANSFORM:-***

A fast Fourier transform (FFT) is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) that computes the [discrete Fourier transform](https://en.wikipedia.org/wiki/Discrete_Fourier_transform) (DFT) of a sequence, or its inverse (IDFT). [Fourier analysis](https://en.wikipedia.org/wiki/Fourier_analysis) converts a signal from its original domain (often time or space) to a representation in the [frequency domain](https://en.wikipedia.org/wiki/Frequency_domain) and vice versa. The DFT is obtained by decomposing a [sequence](https://en.wikipedia.org/wiki/Sequence) of values into components of different frequencies.[[1]](https://en.wikipedia.org/wiki/Fast_Fourier_transform#cite_note-Heideman_Johnson_Burrus_1984-1) This operation is useful in many fields, but computing it directly from the definition is often too slow to be practical. An FFT rapidly computes such transformations by [factorizing](https://en.wikipedia.org/wiki/Matrix_decomposition) the [DFT matrix](https://en.wikipedia.org/wiki/DFT_matrix) into a product of [sparse](https://en.wikipedia.org/wiki/Sparse_matrix) (mostly zero) factors.[[2]](https://en.wikipedia.org/wiki/Fast_Fourier_transform#cite_note-Loan_1992-2) As a result, it manages to reduce the [complexity](https://en.wikipedia.org/wiki/Computational_complexity_theory) of computing the DFT from {\displaystyle O\left(N^{2}\right)}, which arises if one simply applies the definition of DFT, to {\displaystyle O(N\log N)}where {\displaystyle N} is the data size. The difference in speed can be enormous, especially for long data sets where *N* may be in the thousands or millions. In the presence of [round-off error](https://en.wikipedia.org/wiki/Round-off_error), many FFT algorithms are much more accurate than evaluating the DFT definition directly or indirectly. There are many different FFT algorithms based on a wide range of published theories, from simple [complex-number arithmetic](https://en.wikipedia.org/wiki/Complex_number) to [group theory](https://en.wikipedia.org/wiki/Group_theory) and [number theory](https://en.wikipedia.org/wiki/Number_theory).

**In Lab Exercise**

%% fft

t=0:.00001:10e-3;

xt=sin(2\*pi\*1000\*t)+0.5\*sin(2\*pi\*2000\*t+(3\*pi/4));

N=8; %length of dft

n=0:N-1;

Fs=8000;

Ts=1/Fs;

xn=sin(2\*pi\*1000\*n\*Ts)+0.5\*sin(2\*pi\*2000\*n\*Ts+(3\*pi/4));

t1=n.\*Ts;

%code block to plot the input sequence

subplot(311);

plot(t,xt,'r');

hold on

stem(t1,xn,'filled');

xlabel('Time(sec)')

ylabel('amplitude')

grid

%code block to find FFT of the sequence

tic

ft=fft(xn);

toc

magnitude=abs(ft); %find magnitude of individual dft points

%code block to plot magnitude spectrum

k=0:N-1;

Df=Fs/N;

Fk=k.\*Df; %analysis frequencies

subplot(312);

stem(Fk,magnitude,'filled','r');

xlabel('frequency(Hz)');

ylabel('magnitude');

xlim([0 Fs]);

title('magnitude spectrum');

grid

phase=angle(round(ft)); %find the phases of individual DFT point

phase=(phase.\*180)/pi;

subplot(313)

stem(Fk,phase,'filled','r');

xlabel('Frequency(Hz)');

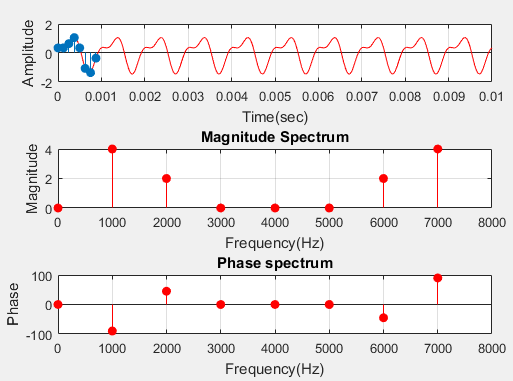
ylabel('Phase');

xlim([0 Fs]);

title('phase spectrum');

grid

**Output**

****

**Post Lab Exercises:**

**Question 1:**

Justify FFT computation time is less than DFT computation time:

**Answer:**

Computation time can be find by using tic and toc functions of MATLAB

Computation time of FFT is **Elapsed time is 0.000036 seconds.**

Computation time of DFT is **Elapsed time is 0.005318 seconds**.

**Question 2:**

Discuss at least three applications of FFT in Electrical power system Engineering.

**Answer:**

The applications related to FFT in electrical power system are as follows.

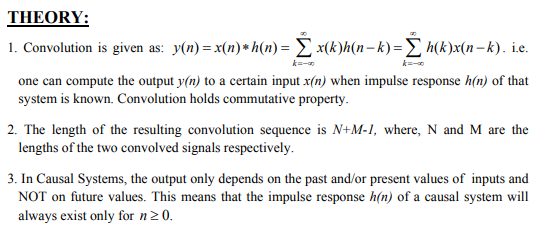
* The Fast Fourier Transform (FFT) is the most commonly used technique for harmonic spectral analysis. Mostly digital relays adopt FFT-Based algorithm to characterize harmonics of the measured signals.
* FFT is also used in power meters. The critical task in a metering application is an accurate computation of energies, which are sometimes referred to as billing quantities. Their computation must be compliant with the international standard for electronic meters.
* FFT also apply on LOLP (Loss Of Load Probability) and LOLE (Loss Of Load Expected). The most important consideration in reliability analysis of power system is its reliability to meet peak load requirement. This is measured by the LOLP.

|  |
| --- |
| **Laboratory Session No. 09** |

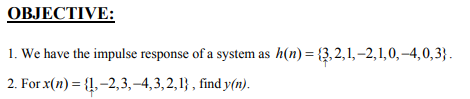
**Objective:**

***Convolution for Discrete Time Signals***

**Discrete Time Convolution**

Convolution, one of the most important concepts in electrical engineering, can be used to determine the output a system produces for a given input signal. It can be shown that a linear time invariant system is completely characterized by its impulse response. The sifting property of the discrete time impulse function tells us that the input signal to a system can be represented as a sum of scaled and shifted unit impulses. Thus, by linearity, it would seem reasonable to compute of the output signal as the sum of scaled and shifted unit impulse responses. That is exactly what the operation of convolution accomplishes. Hence, convolution can be used to determine a linear time invariant system's output from knowledge of the input and the impulse response.

**In Lab Exercise**



clear all;

close all;

clc;

h = [3 2 1 -2 1 0 -4 0 3]; % impulse response

org\_h = 1; % Sample number where origin exists

nh = [0 : length(h)-1]- org\_h + 1;

x = [1 -2 3 -4 3 2 1]; % input sequence

org\_x = 1; % Sample number where origin exists

nx = [0 : length(x)-1]- org\_x + 1;

y = conv(h,x);

ny = [nh(1)+ nx(1) : nh(end)+nx(end)];

figure,

subplot(3,1,1),

stem(nh,h);

xlabel('Time index n'); ylabel('Amplitude');

xlim([nh(1)-1 nh(end)+1]);

title('Impulse Response h(n)'); grid;

subplot(3,1,2),

stem(nx,x);

xlabel('Time index n'); ylabel('Amplitude');

xlim([nx(1)-1 nx(end)+1]);

title('Input Signal x(n)'); grid;

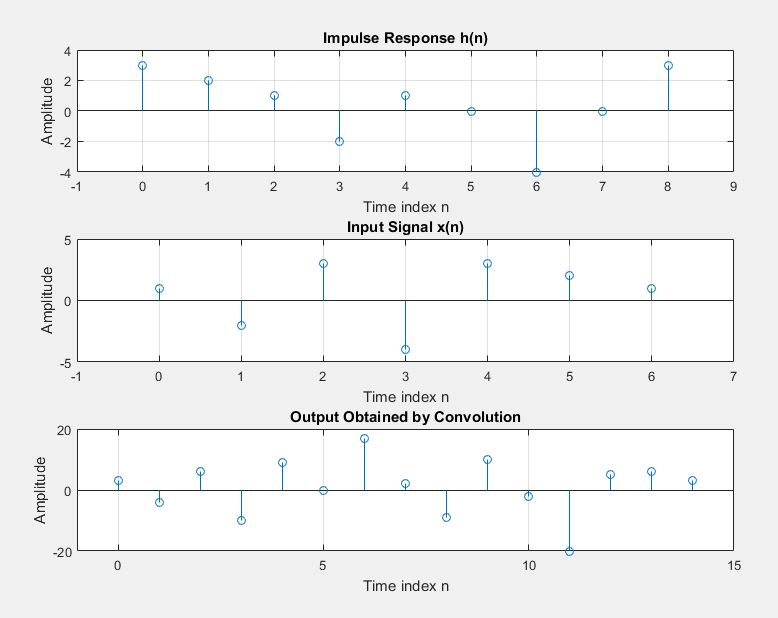
subplot(3,1,3)

stem(ny,y);

xlabel('Time index n'); ylabel('Amplitude');

xlim([ny(1)-1 ny(end)+1]);

title('Output Obtained by Convolution'); grid;



**Post Lab Exercises:**

**Question 1**

%% task1

clear all;

close all;

clc;

h = [3 2 1 -2 1 0 -4 0 3]; % impulse response

org\_h = 3; % Sample number where origin exists

nh = [0 : length(h)-1]- org\_h + 1;

x = [1 -2 3 -4 3 2 1]; % input sequence

org\_x = 1; % Sample number where origin exists

nx = [0 : length(x)-1]- org\_x + 1;

y = conv(h,x);

ny = [nh(1)+ nx(1) : nh(end)+nx(end)];

figure,

subplot(3,1,1),

stem(nh,h);

xlabel('Time index n'); ylabel('Amplitude');

xlim([nh(1)-1 nh(end)+1]);

title('Impulse Response h(n)'); grid;

subplot(3,1,2),

stem(nx,x);

xlabel('Time index n'); ylabel('Amplitude');

xlim([nx(1)-1 nx(end)+1]);

title('Input Signal x(n)'); grid;

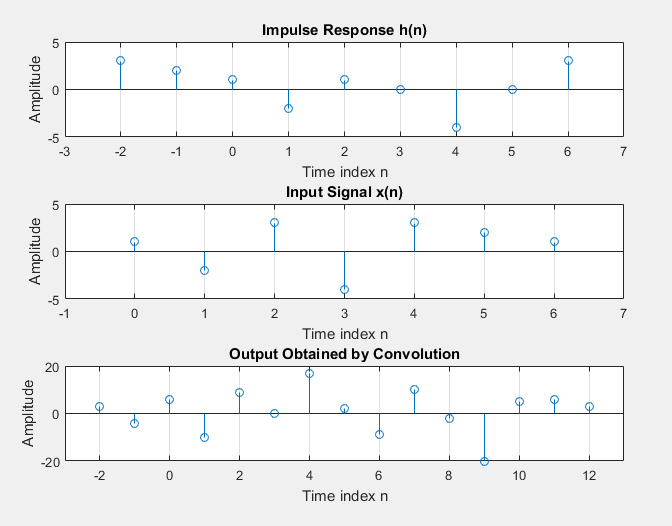
subplot(3,1,3)

stem(ny,y);

xlabel('Time index n'); ylabel('Amplitude');

xlim([ny(1)-1 ny(end)+1]);

title('Output Obtained by Convolution'); grid;



**Question 2**



%% task2

clear all;

close all;

clc;

h = [3 2 1 -2 1 0 -4 0 3]; % impulse response

org\_h = 3; % Sample number where origin exists

nh = [0 : length(h)-1]- org\_h + 1;

x = [0 0 1 0 0]; % input sequence

org\_x = 3; % Sample number where origin exists

nx = [0 : length(x)-1]- org\_x + 1;

y = conv(h,x);

ny = [nh(1)+ nx(1) : nh(end)+nx(end)];

figure,

subplot(3,1,1),

stem(nh,h);

xlabel('Time index n'); ylabel('Amplitude');

xlim([nh(1)-1 nh(end)+1]);

title('Impulse Response h(n)'); grid;

subplot(3,1,2),

stem(nx,x);

xlabel('Time index n'); ylabel('Amplitude');

xlim([nx(1)-1 nx(end)+1]);

title('Input Signal x(n)'); grid;

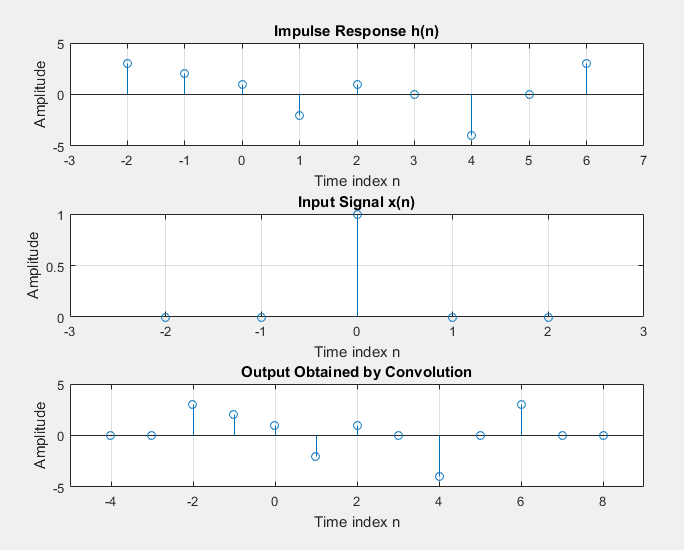
subplot(3,1,3)

stem(ny,y);

xlabel('Time index n'); ylabel('Amplitude');

xlim([ny(1)-1 ny(end)+1]);

title('Output Obtained by Convolution'); grid;



|  |
| --- |
| **Laboratory Session No. 10** |

**Objective:**

***To become familiar with practical constraints in calculating Fourier Transform of any real world signal and to understand the concept of windowing.***

**Spectral Leakage**

The [Fourier transform](https://en.wikipedia.org/wiki/Fourier_transform) of a function of time, s(t), is a complex-valued function of frequency, S(f), often referred to as a [frequency spectrum](https://en.wikipedia.org/wiki/Frequency_spectrum). Any [linear time-invariant](https://en.wikipedia.org/wiki/LTI_system_theory) operation on s(t) produces a new spectrum of the form H(f)•S(f), which changes the relative magnitudes and/or angles ([phase](https://en.wikipedia.org/wiki/Phase_(waves))) of the non-zero values of S(f). Any other type of operation creates new frequency components that may be referred to as spectral leakage in the broadest sense. [Sampling](https://en.wikipedia.org/wiki/Sampling_(signal_processing)), for instance, produces leakage, which we call [aliases](https://en.wikipedia.org/wiki/Aliasing) of the original spectral component. But the term 'leakage' usually refers to the effect of *windowing*, which is the product of s(t) with a different kind of function, the [window function](https://en.wikipedia.org/wiki/Window_function). Window functions happen to have finite duration, but that is not necessary to create leakage. Multiplication by a time-variant function is sufficient. Leakage caused by a window function is most easily characterized by its effect on a sinusoidal s(t) function, whose unwindowed Fourier transform is zero for all but one frequency.

**Windowing**

In [signal processing](https://en.wikipedia.org/wiki/Signal_processing) and [statistics](https://en.wikipedia.org/wiki/Statistics), a window function (also known as an apodization function or tapering function) is a [mathematical function](https://en.wikipedia.org/wiki/Function_(mathematics)) that is zero-valued outside of some chosen [interval](https://en.wikipedia.org/wiki/Interval_(mathematics)), normally symmetric around the middle of the interval, usually near a maximum in the middle, and usually tapering away from the middle. Mathematically, when another function or waveform/data-sequence is "multiplied" by a window function, the product is also zero-valued outside the interval: all that is left is the part where they overlap, the "view through the window". Equivalently, and in actual practice, the segment of data within the window is first isolated, and then only that data is multiplied by the window function values.

**In lab Task**

To understand the concept of windowing consider the MATLAB code

%% windowing

fs=10000;

f=553;

n=0:1:99;

x=cos(2\*pi\*f\*n/1000);

subplot(3,2,1);

stem(x);

title('Original Signal')

subplot(3,2,2);

fx=0:fs/length(x):fs-fs/length(x);

stem(fx,abs(fft(x)));

title('Spectrum of original signal')

w=blackman(100);

subplot(3,2,4);

stem((abs(fft(w,128))));

title('Spectrum of windowing function')

subplot(3,2,3);

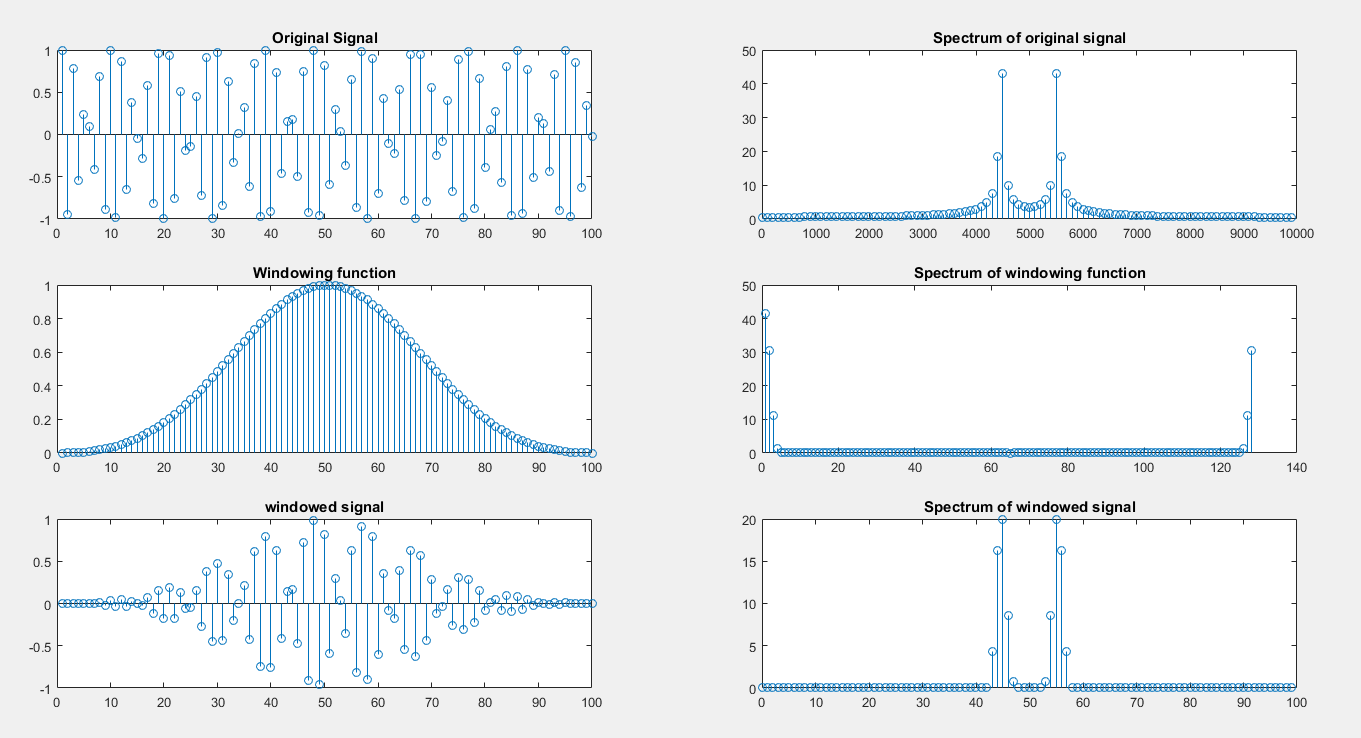
stem(w);

title('Windowing function')

wt=w';

wx=wt.\*x;

subplot(3,2,5);

stem(wx);

**Post lab Exercise**

Discuss some of the windowing functions and their application.

**Hamming Windowing function:** Computers can't do computations with an infinite number of data points, so all signals are "cut off" at either end. This causes the ripple on either side of the peak that you see. The hamming window reduces this ripple, giving you a more accurate idea of the original signal's frequency spectrum

fs=10000;

f=553;

n=0:1:99;

x=cos(2\*pi\*f\*n/1000);

subplot(3,2,1);

stem(x);

title('Original Signal')

subplot(3,2,2);

fx=0:fs/length(x):fs-fs/length(x);

stem(fx,abs(fft(x)));

title('Spectrum of original signal')

w=hamming(100);

subplot(3,2,4);

stem((abs(fft(w,128))));

title('Spectrum of windowing function')

subplot(3,2,3);

stem(w);

title('Windowing function')

wt=w';

wx=wt.\*x;

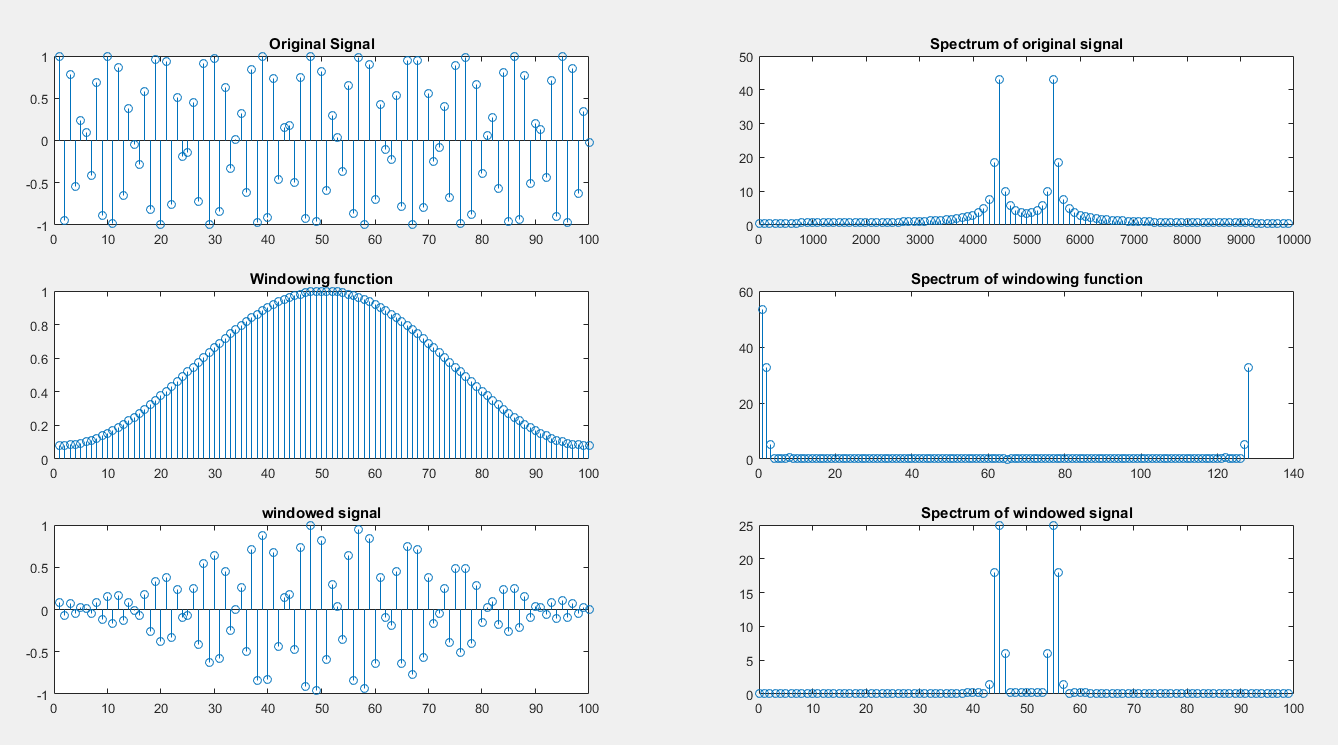
subplot(3,2,5);

stem(wx);

title('windowed signal')

subplot(3,2,6);

stem(n,abs(fft(wx)));

title('Spectrum of windowed signal')

**Taylor Windowing function:** A Taylor window allows you to make tradeoffs between the mainlobe width and the sidelobe level. The Taylor distribution avoids edge discontinuities, so Taylor window sidelobes decrease monotonically.

fs=10000;

f=553;

n=0:1:99;

x=cos(2\*pi\*f\*n/1000);

subplot(3,2,1);

stem(x);

title('Original Signal')

subplot(3,2,2);

fx=0:fs/length(x):fs-fs/length(x);

stem(fx,abs(fft(x)));

title('Spectrum of original signal')

w=taylorwin(100);

subplot(3,2,4);

stem((abs(fft(w,128))));

title('Spectrum of windowing function')

subplot(3,2,3);

stem(w);

title('Windowing function')

wt=w';

wx=wt.\*x;

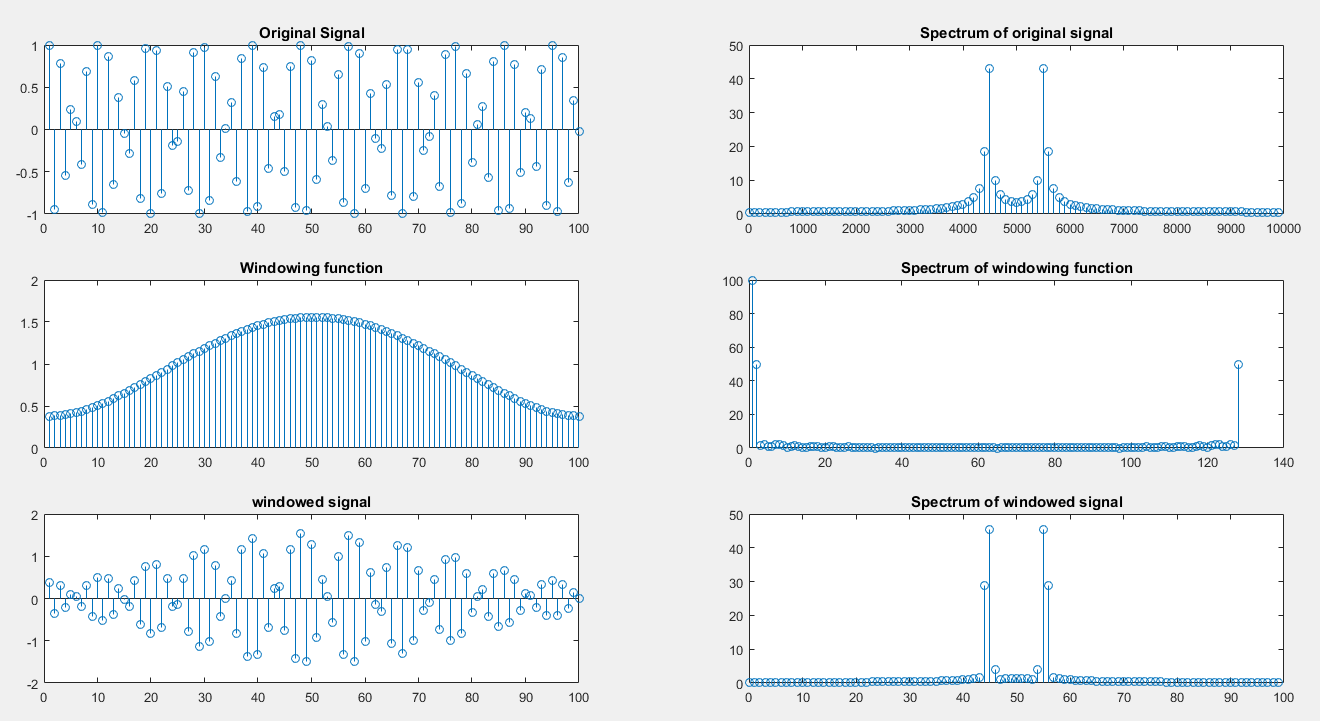
subplot(3,2,5);

stem(wx);

title('windowed signal')

subplot(3,2,6);

stem(n,abs(fft(wx)));

title('Spectrum of windowed signal')

**Flat top windowing function**A flat top window is a partially negative-valued window that has minimal [scalloping loss](https://en.wikipedia.org/wiki/Window_function#Discrete-time_signals) in the frequency domain. That property is desirable for the measurement of amplitudes of sinusoidal frequency components

fs=10000;

f=553;

n=0:1:99;

x=cos(2\*pi\*f\*n/1000);

subplot(3,2,1);

stem(x);

title('Original Signal')

subplot(3,2,2);

fx=0:fs/length(x):fs-fs/length(x);

stem(fx,abs(fft(x)));

title('Spectrum of original signal')

w=flattopwin(100);

subplot(3,2,4);

stem((abs(fft(w,128))));

title('Spectrum of windowing function')

subplot(3,2,3);

stem(w);

title('Windowing function')

wt=w';

wx=wt.\*x;

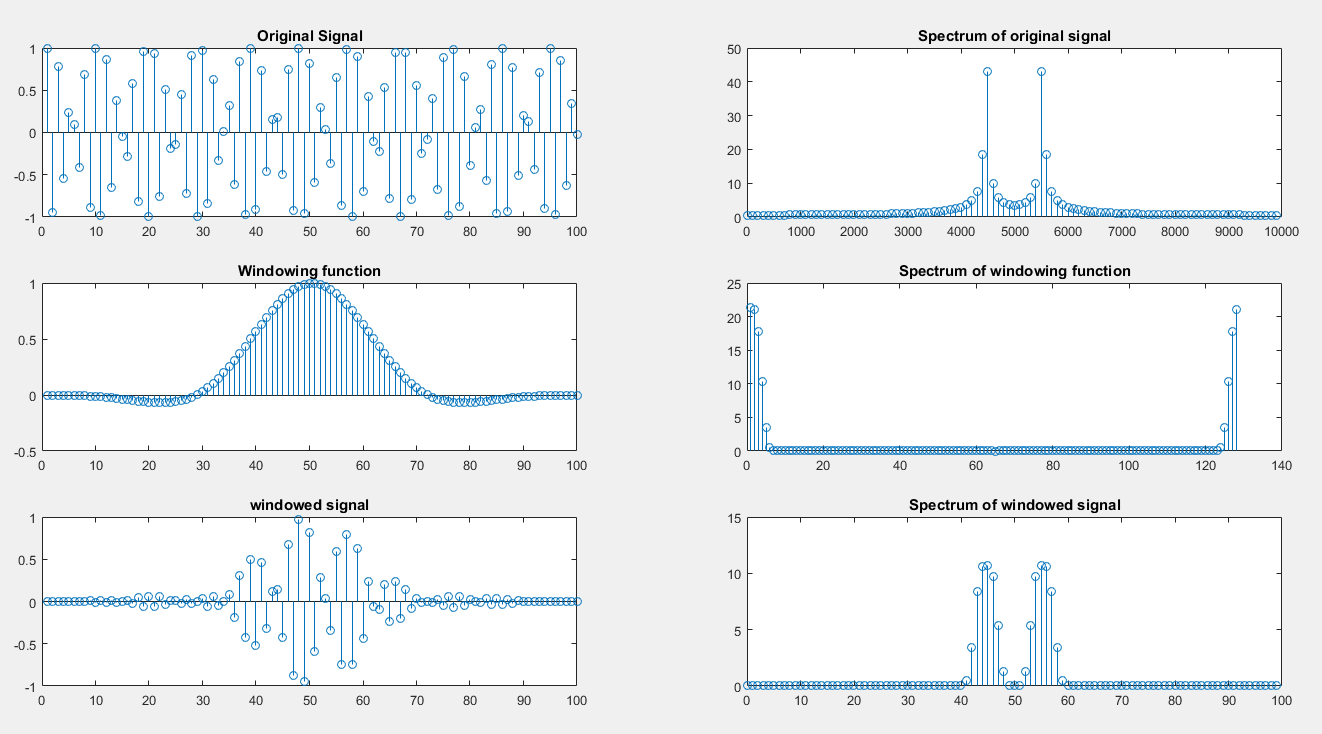
subplot(3,2,5);

stem(wx);

title('windowed signal')

subplot(3,2,6);

stem(n,abs(fft(wx)));

title('Spectrum of windowed signal')

Thus by applying suitable window function spectral leakage will decrease and vice versa.

|  |
| --- |
| **Laboratory Session No. 11**  **(Open Ended Lab 02)** |

**Objective:**

***To convert an analog (current) signal into digital signal using ADC (audio card). Display it on MATLAB Simulink environment and perform FFT of the resulting current signal.***

**Required Components:**

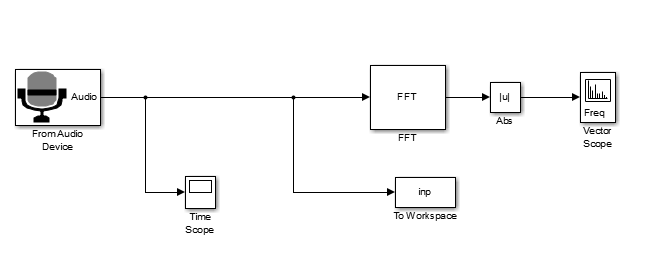
1. Audio Card
2. Current Sensor (current sensing resistor / hall effect sensor / CT)
3. Vero board
4. Audio jack
5. PC with MATLAB environment

**Procedure:**

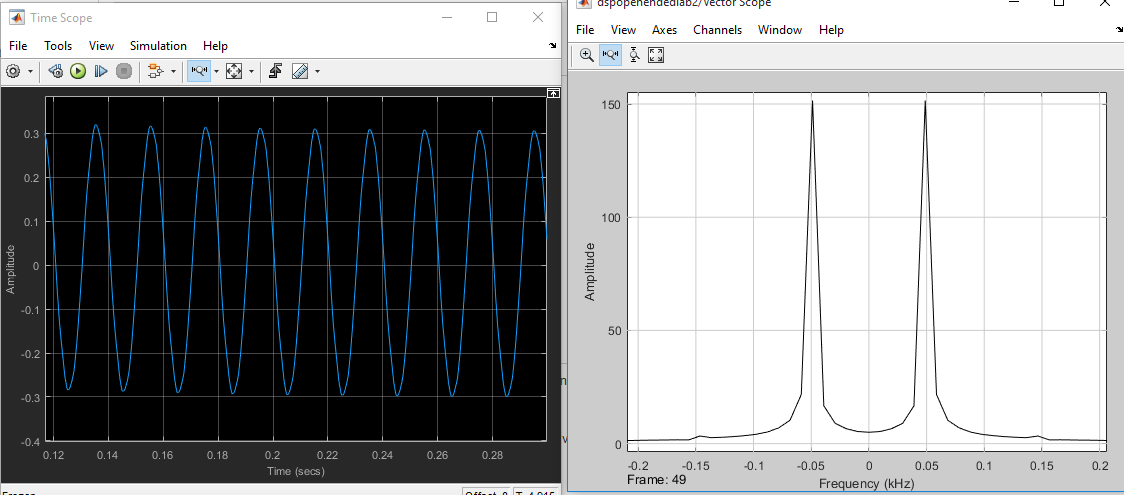
* Using current sensor, convert the current flowing through load into an equivalent voltage.
* If required, using VDR to convert the voltage as obtained from current sensor to a voltage compatible to audio card (show all the calculations of resistances with their power ratings).
* Set the sampling frequency of the audio card ADC in MATLAB Simulink environment with proper justification
* Plot the acquired current waveform to Simulink scope.
* Mention the safe operating range of your equipment.
* Plot the frequency spectrum of the obtained current waveform. Use windowing function to reduce DFT leakage if required.
* Also, plot the frequency spectrum of the line voltage as obtained in open ended lab 01.

# Procedure for Current Waveform

The process of viewing the characteristics of current waveform in frequency domain by applying fft is as follows,

The ‘From audio device’ block will read the waveform from the sound card which is fed through an audio jack and will record it in 10000 samples per second and then that waveform will be stored using ‘Simout Workspace block’ in the variable **inp** and will also plot the waveform using time scope.

FFT can be done in simulink as well by using the block of FFT. The frequency analysis of the signal is plotted on vector scope which is given below



The waveform which came into the work space was plotted in order to verify the output and then the samples were reduced from 10000 to around 1000 as the cycle was repeating itself; therefore the information which was to be communicated was sufficient in one cycle. Now the signal which has come to the workspace of MATLAB can go through the process of the frequency analysis. The MATLAB code for fft and the spectrum is given by

fs=10000;

z=inp(1:1000);

subplot(311) , plot(z) ,grid;

xlabel('Time(sec)')

ylabel('amplitude')

title('Current Wave')

N=length(z);

%n=0:N-1;

f=abs(fft(z/N));

k=0:N/2;

df=fs/N;

p1=f(1:N/2+1);

fk=k.\*df;

subplot(312) , stem(fk,p1\*2), grid , xlim([0 500]);

xlabel('frequency(Hz)');

ylabel('magnitude');

title('Amplitude Spectrum')

phase=angle(round(p1)); %find the phases of individual DFT point

phase=(phase.\*180)/pi;

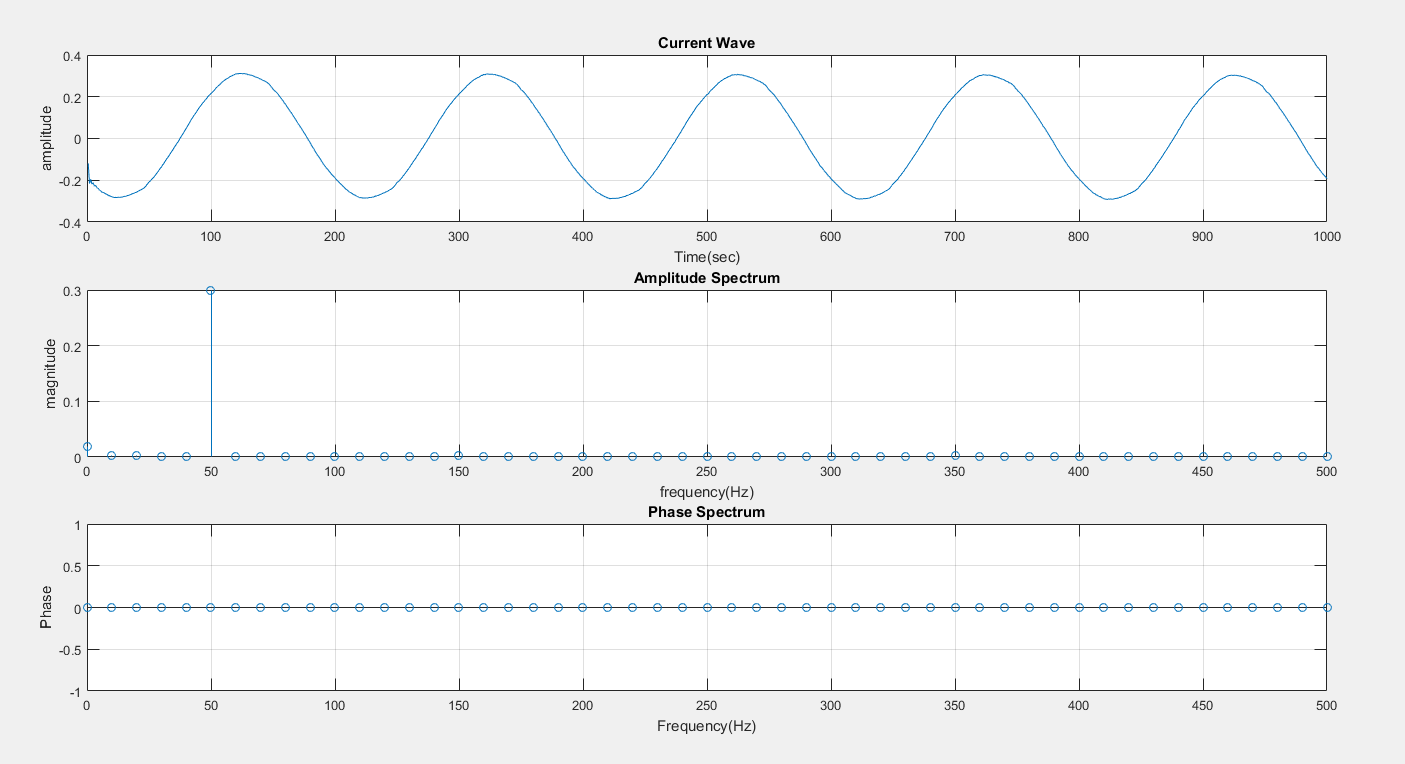
subplot(313)

stem(fk,phase) ,grid, xlim([0 500]);

xlabel('Frequency(Hz)');

ylabel('Phase');

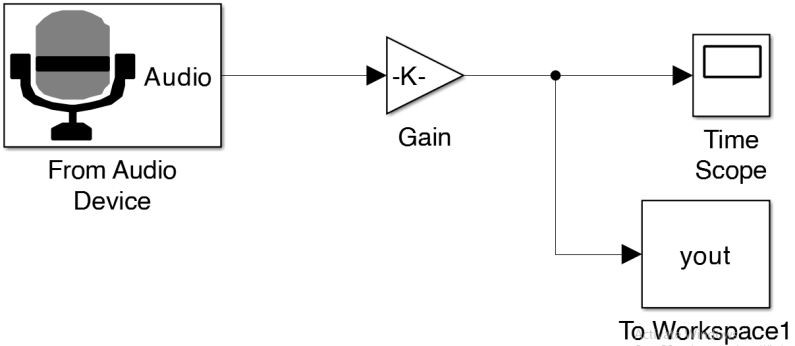
title('Phase Spectrum')

**** **Output**

# Procedure for Voltage Waveform

The process of viewing the characteristics of voltage waveform in frequency domain by applying fft is same as the current waveform. The Matlab based simulation is as follows

The ‘From audio device’ block will read the waveform from the sound card which is fed through an audio jack and will record it in 10000 samples per second and then that waveform will be stored using ‘Simout Workspace block’ in the variable **yout** and will also plot the waveform using time scope.



The waveform which came into the work space was plotted in order to verify the output and then the samples were reduced from 10000 to around 1000 as the cycle was repeating itself; therefore the information which was to be communicated was sufficient in one cycle. Now the signal which has come to the workspace of MATLAB can go through the process of the frequency analysis. The MATLAB code for fft and the spectrum is given by

fs=10000;

z=yout(1:1000);

subplot(311) , plot(z) ,grid;

xlabel('Time(sec)')

ylabel('amplitude')

title('Voltage Wave')

N=length(z);

%n=0:N-1;

f=abs(fft(z/N));

k=0:N/2;

df=fs/N;

p1=f(1:N/2+1);

fk=k.\*df;

subplot(312) , stem(fk,p1\*2), grid , xlim([0 500]);

title('Magnitude Spectrum')

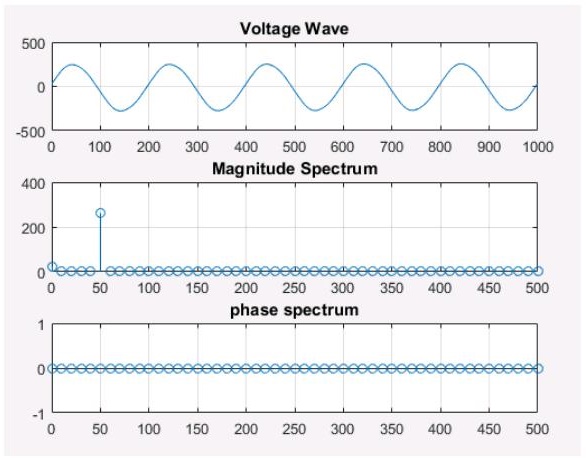
phase=angle(round(p1)); %find the phases of individual DFT point

phase=(phase.\*180)/pi;

subplot(313)

stem(fk,phase) ,grid, xlim([0 500]);

title('phase Spectrum')



**Results:**

In this lab it is observed that the voltage and current signal coming from the sound card only contain a single frequency component at supply frequency equal to 50 Hz. The frequency and phase spectrum is analyzed and thus proved our intuition.

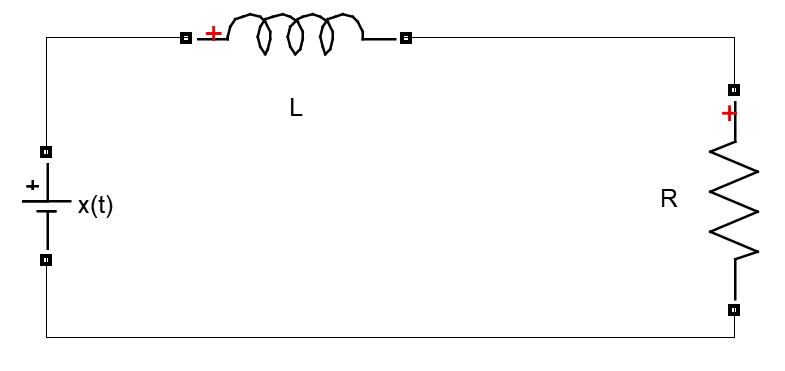
|  |
| --- |
| **Laboratory Session No. 06** |

**Objective:**

***To model a source free and sourced RL circuit in a discrete time domain and to verify it using MATLAB code and Simulink environment.***

**Theory:**

Consider a series RL circuit excited by a voltage source as shown below,



Applying KVL we get,

(1)

For a source a source free RL circuit the above expression will become,

(2)

By using first principle, the derivate of inductor current in a discrete time domain can be evaluated as,

(3)

or,

(4)

From eq. (2), the derivative at the instant can be evaluated as,

(5)

Equating eq. (4) and (5) we get,

(6)

By choosing sampling *T* close to zero we can evaluate the difference equation as,

(7)

Where, and are replaced by and respectively.

**Post Lab Exercise:**

**Question 1:**

For a series RL circuit excited by a voltage source , derive an expression for inductor current , in discrete time domain. Using the derived expression, develop a MATLAB code to plot the inductor current under following system parameters:

**Answer:**

**Question 1:**

We have an impulse response of a system h(n)= {**3**,2,1, 2,1,0, 4,0,3}, provided with an input

*x(n)*={**1** 2 3 5 4 6}, find and plot *y(n)*. Also check and comment on the causality property of the system.

**Answer:**

**Question 2:**

We have an impulse response of a system h(n)= {-3,1,2, 4,**1**,0, 4,0,3}, provided with an input

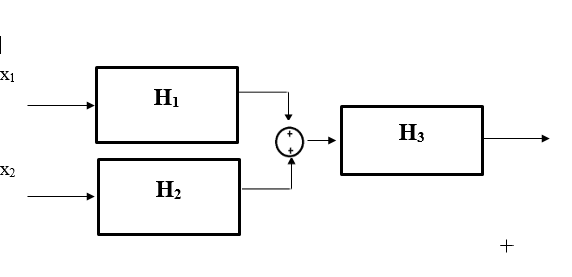
x(n)={**1** 2 3 5 4 6}, find and plot y(n). Also check and comment on the causality property of the system.

**Answer:**

**Question 3:**

If impulse response of the system h1(n)= {**1** 1 1}, h2(n)= {**0**,0, 1} and h3(n)= {1,**2**,0, 1} , Input

*x1(n)*={**1** 2 3 5 4 6} and *x2(n)*={**1** 2 3 5 4 6}, find and plot *y(n)*. Also check and comment on the causality property of the system.



**Answer:**

|  |
| --- |
| **Laboratory Session No. 07** |

**Objective:**

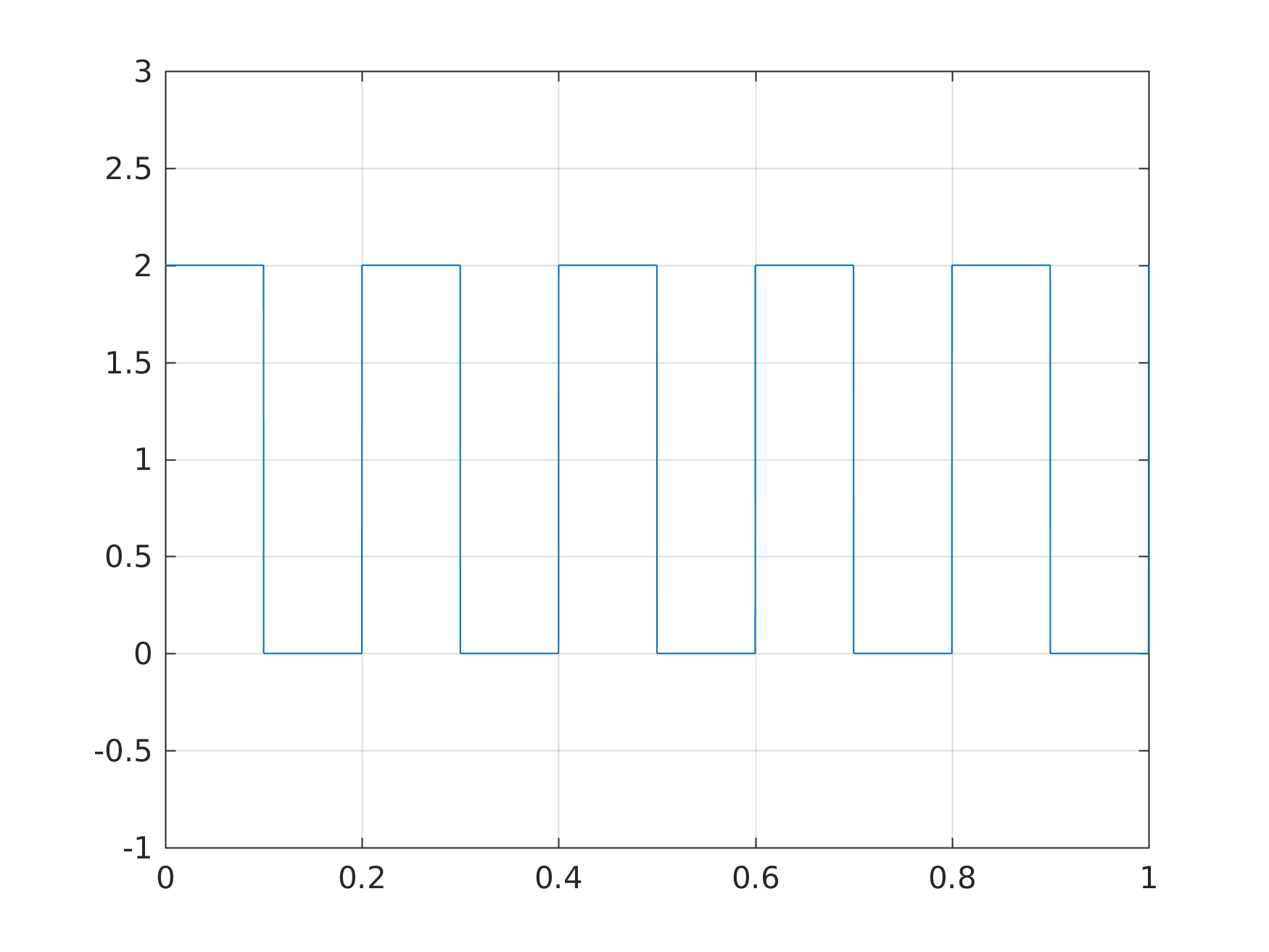
***To generate a square wave in time domain of specific time period and pulse width and to apply CTFS equation to compute the spectral coefficients using MATLAB.***

**Post Lab Exercises:**

**Question 1:**

Generate a square wave of magnitude 2 units with duty cycle of 50 % and have a frequency of 5 Hz using MATLAB *square()* function and then plot the frequency spectrum (magnitude vs frequency plot) of the resulting wave using MATLAB script (Evaluate five spectral coefficients). Also, verify your MATLAB results using hand calculations.

Note: Use MATLAB help to get the syntax of the *square()* function.



**Answer:**

|  |
| --- |
| **Laboratory Session No. 09** |

**Objective:**

***To become familiar with practical constraints in calculating Fourier Transform of any real world signal****.*

**Post Lab Exercises:**

**Question 1:**

Briefly discuss the concept of spectral leakage.

**Answer:**

**Question 2:**

Discuss the solution of spectral leakage.

**Answer:**

**Question 3:**

Discuss the importance of window size (value of N) on frequency spectrum.

**Answer:**

**Question 4:**

Run the following MATLAB script, observe and discuss the resulting plots.

|  |
| --- |
| clear; close;  fs=4000;  N=400;  n=0:N-1;  y=10\*cos(2\*pi\*1500\*n/fs);  f=abs(fft(y));  k=0:N-1;  df=fs/N;  kf=k.\*df;  subplot(411);  stem(kf,f);  xlabel('Frequency Hz');  title('Magnitude Spectrum Without Spectral Leakage');  % If frequency bins are not available (Spectral Leakage) %  N=50;  n=0:N-1;  nfft=1024;  y=100\*cos(2\*pi\*1500\*n/fs);  f=abs(fft(y,nfft));  k=0:nfft-1;  df=fs/nfft;  kf=k.\*df;  subplot(412);  plot(kf,f);  xlabel('Frequency Hz');  title('Magnitude Spectrum With spectral leakage Leakage');    % Solution of Spectral leakage (Windowing)%  win=window(@hamming,N);  subplot(413);  stem(n,win);  title('Window');  y\_win=y.\*win';  f\_win=abs(fft(y\_win,nfft));  k=0:nfft-1;  df=fs/nfft;  kf=k.\*df;  subplot(414);  plot(kf,f\_win);  xlabel('Frequency Hz');  title('Magnitude Spectrum With Windowing'); |

**Answer:**

|  |
| --- |
| **Laboratory Session No. 11** |

**Objective:**

***To understand the concept of window overlapping.***

**Post Lab Exercises:**

**Question 1:**

Run the following MATLAB script, attach and discuss the resulting plot.

|  |
| --- |
| clear all;  clc;  [y fs] = audioread('wavTones.com.unregistred.warble\_1000-2000Hz\_-6dBFS\_3s.wav');  n=0:5000;  win\_size = 40e-3\*fs;  frame\_rate = 20e-3\*fs;  Y = buffer(y,win\_size,frame\_rate);  [m n] = size(Y);  win = window(@hamming,win\_size);  YYF = [];  for i = 1:n  YY = Y(:,i).\*win;  YF = abs(fft(YY,fs));  YYF = [YYF YF];  end  figure;  for i = 1:n  clf;  plot(YYF(:,i));  drawnow;  end |

**Answer:**