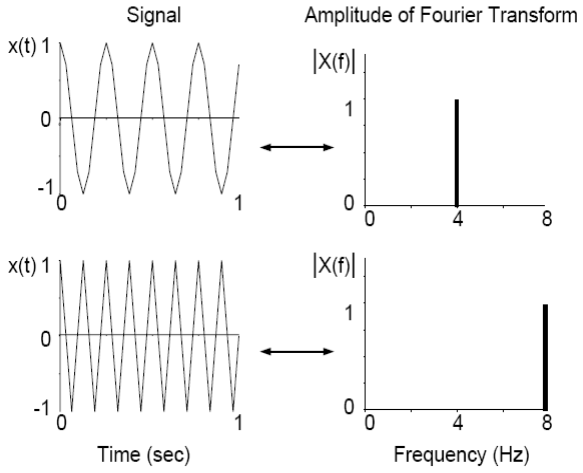
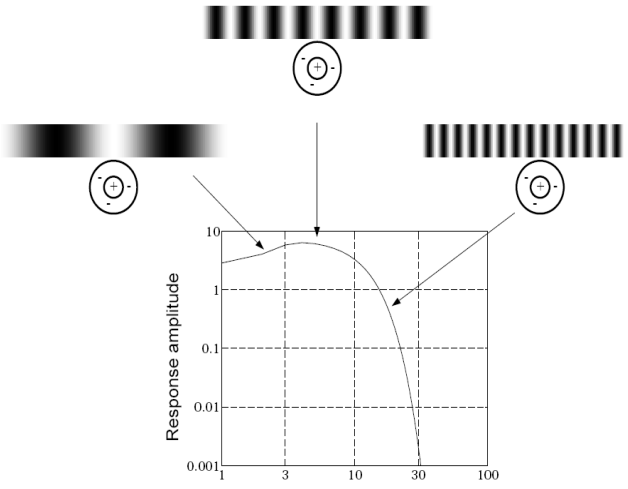
University of Bahrain
College of Engineering
Department of Electrical and Electronics Engineering

EEG 271 Signals and System





*

EEG 271 Signals and Systems Laboratory Classes

Prepared by : Dr. Ebrahim Abdullah Mattar

* : Note : Cover Photo snap taken form Prof. David Heeger

(A) General Guidelines for Lab sessions of EEG 271 :

1) The timetable for laboratory sessions is given separately. You are expected to attend all of these timetable sessions (if, because of illness or some other good reason, you fail to attend a session then you must inform the appropriate laboratory instructor immediately).

2) Remember that experiments cannot be properly executed without prior study as the limited laboratory time must be devoted to experimental investigations.

3) Attendance records are kept, so if you are late please report to the academic supervisor to ensure that presence is recorded. If you are absent form a class you should contact the appropriate organizer at the earliest opportunity.

4) As with most University activities, it is YOUR RESPONSIBLIT to ensure that you receive maximum benefit form the facilities provided. You are MOST STRONGLY advised to sturdy each experiment a few days on ADVANCE. This will involve reading additional materials such as lecture notes and course text books, deriving theoretical formulae quoted in the instructions, making preliminary calculations of expected results and considering appropriate experimental methods. Almost certainly you will encounter problems in understanding the background to an experiment, setting up equipment or interpretation of results.

5) Remember that laboratory time is precious, and must not be wasted because of insufficient preparation (time wasted in this way is also unfair to your partner: it will be regarded as a serious deficiency).

6) You are required to record all experimental work and design work in a personal A4 log book with HARD COVER.

7) WORK ON LOOSE-LEAF PAPERS WILL NOT BE ACCEPTED.

8) Your log work should contain records of trail design, mistakes, repeated work and comments which would not be expected to appear in a formal experimental report. It is essential to record everything you do and your reasons for doing it – the log book should be thought of as a record of a conversation with yourself as the experiment proceeds or the design develops. As a result you should have all the information you need in your log book to write a formal report on a experiment or design several months afterwards.

9) Your log book should always contain significant entries. This is true even if you have spent the day typing your results. A record of the amount of work that you have done may prove invaluable for useful data.

At the end of each laboratory session you should write, in not more than half a page, a brief summary of your conclusions.

(B) Report Writing :

These notes are provided to assist in the preparation of the formal reports required.

Presentation

- A4 Page size.
- Use a word processing system if all possible, together with a printer that produces easily readable characters (sometimes a photocopy of the printed output is easier to read).
- Any report that is difficult to understand, for any reason, will automatically be given a low grading.
- All pages must have a ruled margin (at least 25mm) on the binding edge and be numbered. Only one side of the paper should be used.
- All sections of text must be numbered serially in Arabic numerals.
- All figures and graphs must have a title and be numbered serially in Arabic numerals.
- All tables must have a title and be numbered serially in Roman numerals.

Report Structure

- Title page (front cover).
- Summary (single page – about 50 words).
- Contents (single page – should list content of report with appropriate text sections and page numbers).
- Report (most reports start with an "Introduction" and finish with "Conclusions".
- References (most student reports contain very references).
- Appendices.
- Acknowledgments (if appropriate).

Report Content

- The title of each report is announced at a suitable time, together with a brief specification of items to be included and a suggested maximum length.

- The title should be directly applicable to the report produced. It is especially important that introductions and conclusions be concise and relate specifically to the main topics of the report. A common error with student reports is to begin detailed explanations immediately without any explanation of the overall system or justification for the approach taken.

- Any item selected in the report specification for special consideration must be given appropriate emphasis in the report. Failure to include such items (e.g. "experimental evidence based on laboratory measurements") is a serious omission.

- The report should be exceed the maximum length without good reason as the marking will be partly based on conciseness. A report that appears to be too long will NOT normally achieve a high grade.

- Reference should be sent out the report body, and should be to reputable academic texts (e.g. IEEE publications).

List of EEG 271 experiments

Experiment (1) : Introduction to MATLAB for Signals and Systems.

Experiment (2) : Computations on Analog and Discrete Time Signals.

Experiment (3) : Convolution : Continuous and Discrete - Time Signals

Experiment (4) : Dynamic System Differential Equation (Differential Equation and State Space Description).

Experiment (5) : Experience with MATLAB advanced tools. This will be done in three main parts.

University of Bahrain
College of Engineering
Department of Electrical and Electronics Engineering
EEG 270 : Signals and Systems

Experiment (1) : Introduction to MATLAB for Signals and Systems

Objectives :

- To introduce students to MatLab and relate it to physical signals and systems at the Lab.
- To allow students to appreciate meanings of practical signals and building of physical systems.

Equipment :

- Capacitors, resistors,
- Voltmeter (30-0-30),
- Hand Calculator,
- Storage Oscilloscope.
- Computing Systems with MatLab

1. Warm-up

MATLAB is a high-level programming language that has been used extensively to solve complex engineering problems. The language itself bears some similarities with ANSI C and FORTRAN. MATLAB works with three types of windows on your computer screen. These are the Command window, the Figure window and the Editor window. The Figure window only pops up whenever you plot something. The Editor window is used for writing and editing MATLAB programs (called M-files) and can be invoked in Windows from the pull-down menu after selecting File | New | M-file. In UNIX, the Editor window pops up when you type in the command window: edit filename ('filename' is the name of the file you want to create). The command window is the main window in which you communicate with the MATLAB interpreter. The MATLAB interpreter displays a command

>> indicating that it is ready to accept commands from you.

• View the MATLAB introduction by typing

>> intro at the MATLAB prompt. This short introduction will demonstrate some basic MATLAB commands.

• Explore MATLAB's help capability by trying the following:

```
>> help
>> help plot
>> help ops
>> help arith
```

• Type demo and explore some of the demos of MATLAB commands.
• You can use the command window as a calculator, or you can use it to call other

MATLAB programs (M-files).
Say you want to evaluate the expression a3 +√bd −4c , where a=1.2, b=2.3, c=4.5 and d=4. Then in the command window, type:

```
>> a = 1.2;
>> b=2.3;
>> c=4.5;
>> d=4;
>> a^3+sqrt(b*d)-4*c
```

2. Arithmetic Operations

There are four different arithmetic operators:
+ addition
− subtraction
* multiplication
/ division (for matrices it also means inversion)
There are also three other operators that operate on an element by element basis: .* multiplication of two vectors, element by element
./ division of two vectors, element-wise .^ raise all the elements of a vector to a power.

Suppose that we have the vectors x = [x1, x2, ..., xn] and y = [y1, y2, ..., yn].

$$\mathrm{x.*y} = [x_1y_1, x_2y_2, ..., x_ny_n]$$
$$\mathrm{x./y} = [\tfrac{x_1}{y_1}, \tfrac{x_2}{y_2}, ..., \tfrac{x_n}{y_n}]$$
$$\mathrm{x.\hat{}p} = [x_1^p, x_2^p, ..., x_n^p]$$

The arithmetic operators + and — can be used to add or subtract matrices, scalars or vectors. By vectors we mean one-dimensional arrays and by matrices we mean multi-dimensional arrays. This terminology of vectors and matrices comes from Linear Algebra. For example:

```
>> X=[1,3,4]
>> Y=[4,5,6]
>> X+Y                    ans=  5  8  10
```

For the vectors X and Y the operator + adds the elements of the vectors, one by one, assuming that the two vectors have the same dimension. In the above example, both vectors had the dimension 1 × 3, i.e., one row with three columns. An error will occur if you try to add a 1 × 3 vector to a 3 × 1 vector. The same applies for matrices. To compute the dot product of two vectors (i.e. Pi xiyi), you can use the multiplication operator * . For the above example, it is:

```
>> X*Y'                   ans = 43
```

Note the single quote after Y. The single quote denotes the transpose of a matrix or a vector. To compute an element by element multiplication of two vectors (or two arrays), you can use the .* operator:

```
>> X .* Y                 ans = 4 15 24
```

That is, X.*Y means [1×4, 3×5, 4×6] = [4 15 24]. The '.*' operator is used very often (and is highly recommended) because it is executed much faster compared to the code that uses for loops.

3. Complex numbers

MATLAB also supports complex numbers. The imaginary number is denoted with the symbol i or j, assuming that you did not use these symbols anywhere in your program (that is very important!). Try the following:

```
>> z=3 + 4i % note that you do not need the '*' after 4
>> conj(z) % computes the conjugate of z
>> angle(z) % computes the phase of z
>> real(z) % computes the real part of z
>> imag(z) % computes the imaginary part of z
>> abs(z) % computes the magnitude of z
```

You can also define the imaginary number with any other variables you like. Try the following:

```
>> img=sqrt(-1)
>> z=3+4*img
>> exp(pi*img)
```

4. Array indexing

In MATLAB, all arrays (vectors) are indexed starting with 1, i.e., y(1) is the first element of the array y. Note that the arrays are indexed using parenthesis (.) and not square brackets [.] as in C/C++. To create an array having as elements the integers 1 through 6, just enter:

```
>> x=[1,2,3,4,5,6]
```

Alternatively, you can use the : notation,

```
>> x=1:6
```

The : notation above creates a vector starting from 1 to 6, in steps of 1. If you want to create a vector from 1 to 6 in steps of say 2, then type:

```
>> x=1:2:6                Ans =135
```

Try the following code:

```
>> ii=2:4:17
>> jj=20:-2:0
>> ii=2:(1/10):4
```

Extracting or inserting numbers in a vector can be done very easily. To concatenate an array, you can use the [ ] operator, as shown in the example below:

```
>> x=[1:3 4 6 100:110]
```

To access a subset of the array, try the following:

```
>> x(3:7)
>> length(x) % gives the size of the array or vector
>> x(2:2:length(x))
```

5. Allocating memory

You can allocate memory for one-dimensional arrays (vectors) using the zeros command. The following command allocates memory for a 100-dimensional array:

```
>> Y=zeros(100,1);
>> Y(30)                  Ans = 0
```

Similarly, you can allocate memory for two-dimensional arrays (matrices). The command

```
>> Y=zeros(4,5)
```

defines a 4 by 5 matrix. Similar to the zeros command, you can use the command ones to define a vector containing all ones,

```
>> Y=ones(1,5)            Ans=  1 1 1 1 1
```

6. Special characters and functions

Some common special characters used in MATLAB are given below:

Symbol	Meaning
pi	$\pi(3.14...)$
sqrt	indicates square root e.g., sqrt(4)=2
^	indicates power(e.g., 3^2=9)
abs	Absolute value | .| e.g., abs(-3)=3
NaN	Not-a-number, obtained when comparing mathematically undefined operations, such as $0/0$
Inf	Represents $+\infty$
;	Indicates the end of a row in a matrix. It is also used to suppress printing on the screen (echo off)
%	Denotes a comment. Anything to the right of % is ignored by the MATLAB interpreter and is considered as comments
'	Denotes transpose of a vector or matrix. It's also used to define strings, e.g.,str1='DSP';

Some special functions are given below: length(x) - gives the dimension of the array x find - Finds indices of nonzero elements. Examples :

```
>> x=1:10;
>> length(x)        Ans =10
```

The function find returns the indices of the vector X that are non-zero. For example, I = find(X>100), finds all the indices of X when X is greater than 100. So for the above example:

```
>> find(x> 4)       Ans =  5  6  7  8  9  10
```

7. Control flow

MATLAB has the following flow control constructs:

- if statements
- switch statements
- for loops
- while loops
- break statements

The if, for, switch and while statements need to terminate with an end statement. Examples:

```
IF:     x=-3;
        if x>0
        str='positive';
                elseif x<0
                        str='negative';
                        elseif x= = 0
                str='zero';
        else
        str='error';
end
```

What is the value of 'str' after execution of the above code?

```
WHILE:  x=-10;
                while x<0
                x=x+1;
                end
```

What is the value of x after execution of the above loop?

```
FOR loop:
        X=0;
        for      i=1:10
                 X=X+1;
        end
```

The above code computes the sum of all numbers from 1 to 10.

BREAK:

The break statement lets you exit early from a for or a while loop:

```
x=-10;
while x<0
        x=x+2;
                if x = = -2
        break;
                end;  end
```

MATLAB supports the following relational and logical operators:

Relational Operators Symbol Meaning

<=	Less than equal
<	Less than
>=	Greater than equal
>	Greater than
==	Equal
~ =	Not equal

Logical Operators	Symbol Meaning
&	AND
|	OR
~	NOT

8. Plotting

You can plot arrays using MATLAB's function plot. The function plot (.) is used to generate line plots. The function stem(.) is used to generate "picket-fence" type of plots. For example:

```
>> x=1:20;
>> plot(x) %see Figure 1
>> stem(x) % see Figure 2
```

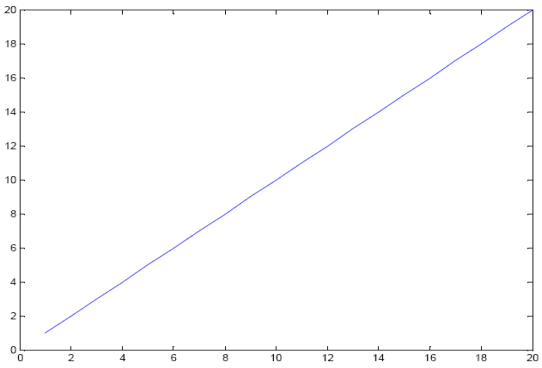


Figure 1: Plot obtained using the plot command.

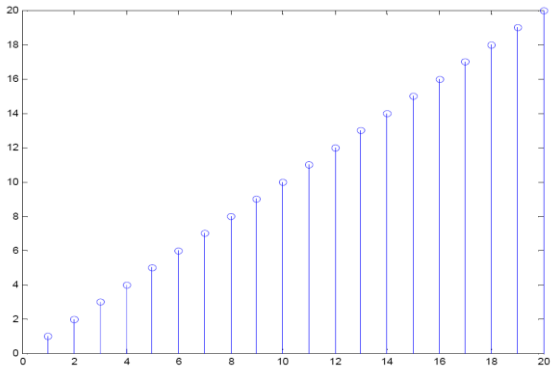


Figure 2: Plot obtained using the stem function

Example of a plot generated using the plot command is shown in Figure 1, and example of a plot generated using the stem function is shown in Figure 2. More generally, plot(X,Y) plots vector Y versus vector X. Various line types, plot symbols and colors may be obtained

using plot(X,Y,S) where S is a character string indicating the color of the line, and the type of line (e.g., dashed, solid, dotted, etc.). Examples for the string S include:

r	red	+	plus	−−	dashed
g	green	*	star		
b	blue	s	square		

You can insert x-labels, y-labels and title to the plots, using the functions xlabel(.), ylabel(.) and title(.) respectively. To plot two or more graphs on the same figure, use the command subplot. For instance, to show the above two plots in the same figure, type:

```
>> subplot(2,1,1), plot(x)
>> subplot(2,1,2), stem(x)
```

The (m,n,p) argument in the subplot command indicates that the figure will be split in m rows and n columns. The 'p' argument takes the values 1, 2, . . . , m× n. In the example above, m = 2, n = 1,and, p = 1 for the top figure and p = 2 for the bottom figure. To get more help on plotting, type: help plot or help subplot.

9. Programming in MATLAB (M-files)

MATLAB programming is done using M-files, i.e., files that have the extension .m. These files are created using a text editor. To open the text editor, go to the File pull-down menu, choose New, then M-file. After you type in the program, save it, and then call it from the command window to execute it.

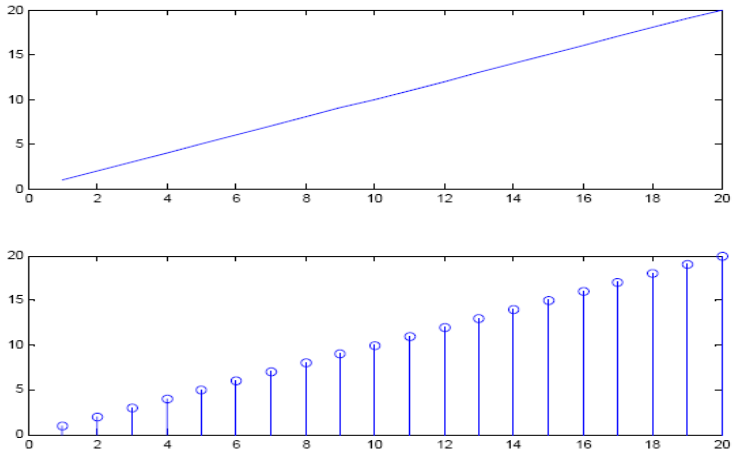


Figure 3: Output of the subplot(2,1,p) command.

Say for instance that you want to write a program to compute the average (mean) of a vector x. The program should take as input the vector x and return the average of the vector.

Steps:
1. You need to create a new file, called "average.m". If you are in Windows, open the text editor by going to the File pull-down menu, choose New, then M-file. If you are in UNIX, then type in the command window: edit average.m. Type the following in the empty file

```
function y=average(x)
        L=length(x);
        sum=0;
                for i=1:L
                        sum=sum+x(i);
                end
        y=sum/L; % the average of x
```

Remarks:
y — is the output of the function "average", x — is the input array to the function "average" average — is the name of the function. It's best if it has the same name as the filename. MATLAB files always need to have the extension .m
2. From the Editor pull-down menu, go to File | Save, and enter: average.m for the filename.
3. Go to the Command window to execute the program by typing:

```
>> x=1:100;
>> y=average(x)
ans = 50.5000
```

Note that the function average takes as input the array x and returns one number, the average of the array. In general, you can pass more than one input argument and can have the function return multiple values. You can declare the function average, for instance, to return 3 variables while taking 4 variables as input with the following statement:

```
function [y1, y2, y3]=average(x1,x2,x3,x4)
```

In the command window it has to be invoked as:

```
>> [y1, y2, y3]=average(x1,x2,x3,x4)
```

10. MATLAB sound
If your PC has a sound card, then you can use the function soundsc to play back speech or audio files through the PC's speakers. The usage of this function is given below: soundsc(Y,FS) sends the signal in vector Y (with sample frequency FS) out to the speaker on platforms that support sound. Stereo sounds are played, on platforms that support it, when Y is an N-by-2 matrix. Try the following code, and listen to a 400-Hz tone:

```
>> t=0:1/8192:1;
>> x=cos(2*pi*400*t);
>> soundsc(x,8192);
```

Now, try listening to noise:

```
>> noise=randn(8192,1); % generate 8192 samples of noise
>> soundsc(noise,8192);
```

The function randn generates Gaussian noise.

11. Loading and saving data

You can load or save data using the commands load and save. To save the variable x of the above code in the file data.mat, type:

```
>> save data.mat x
```

Note that MATLAB's data files have the extension .mat. To retrieve the data that was saved in the vector x, type:

```
>> load data.mat
```

The vector x is loaded in memory. To see the contents of memory use the command whos:

```
>> whos
```

```
Name      Size        Bytes     Class
x         1x8193      65544     double array
Grand total is 8193 elements using 65544 bytes.
```

The command whos gives a list of all the variables currently in memory, along with their dimension. In our case, x contained 8193 samples. To clear up memory after loading a file, you may type clear all when done. That is very important, because if you do not clear all the variables in memory, you may run into problems with other programs that you will write that use the same variables.

Reference :

Dr. Charles P. Bernardin, University of Texas at Dallas. For ALL MatLab materials.

University of Bahrain
College of Engineering
Department of Electrical and Electronics Engineering
EEG 271 : Signals and Systems

Lab (2) : Computations on Analog and Discrete Time Signals

Laboratory Objectives :

To learn some basics of analog and discrete signals computations and analysis..

Needed Equipment :

- Variable Capacitors, Resistors.
- Analog and digital Voltmeters (30-0-30).
- Hand Calculator.
- Storage Oscilloscope.
- Op-Amps and some Feedback Tools.

Introduction to the Experiment:

Analog and discrete-time signals are processed in accordance to certain applications on such signals. Op-Amps (known as Operational Amplifiers, as shown in Fig (1)) are considered as the main tools for computation continuous and discrete-time signals. The 741 operational amplifier (741 Op-Amp) is the basic kind for a wide range of commercial devices available on the market one a day. With standard op-amps, we can perform : signal scaling, integration, differnation, log calculations, and conversions. Op-amps have typically two inputs

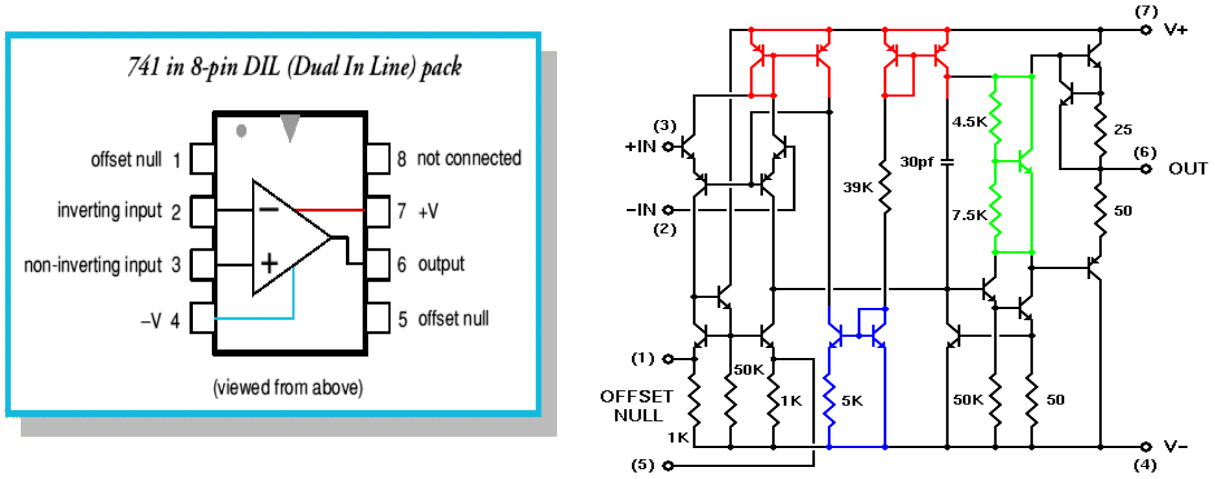


Fig (1) : (A) Op-Amp 741 Chip Layout. (B)The internal circuitry of the 741 op amp. (1)

NOTE : (1)

ALL figure are the @ copyright of : - (Dr. Ken Bigelow**)** , for Interactive Demonstrations for Education.
- (Prof. Connie Chang-Hasnain).

A very typical commercial IC op amp circuit is the 741. This IC has been available for many years, and a number of variations have been developed to help minimize the errors inherent in its construction and operation. Nevertheless, the analysis we will perform here using the 741 will apply to any other IC op amp, if you take into account the actual parameters of the device you are actually using. Therefore, we will use the 741 as our example IC op amp. A problem with any op amp is a limited frequency response. The higher the gain of the complete circuit, the lower the working frequency response. This is one reason an overall gain of 20 is a practical limit. (Another reason is that the input and feedback resistors become too different from each other.) Also, the standard 741 has a *slew rate* of 0.5 v/μs. This means that the output voltage cannot change any faster than this. The newer generation of op amps, such as the 741S, have a slew rate more like 5 v/μs, and hence can operate over the entire audio range of frequencies without serious problems.

Laboratory Procedures :

(1) Gain Coefficient (Amplification and Attenuation) :

The scaling of the amplifier, and therefore the constant coefficient, is set by the input and feedback resistors. The gain of the circuit is given by ($-R_f/R_{in}$). Therefore, the gain of this particular circuit is (20k/10k = 2). We could equally well invert the incoming X signal before applying it to the circuit. Hence, construct the shown below circuits. It is an inverting amplifier (attenuator circuit). Apply a sin wave of (5 V_{pp} and 1 KHz frequency). Make the setting of R_1 and R_2 as a ratio of 3, 2, and 0.8. Record your results. Also apply a d.c Volt of variable value and watch your results.

Fig. (2) : Inverting (Amplifier) or (Attenuator)

(2) Analog Addition, Subtraction and Logarithmic Circuit:

The circuit shown in Fig (3-A) is designed to solve the equation (Z = 2X – Y). Input resistors are not set as the same value. Each input signal has its own separate coefficient. Since R_f is necessarily common to both inputs, the coefficients must be set by selecting different input resistors for the input signals, according to the desired coefficients. Each input signal uses its own input resistor, R_{in}, and its own separate value of R_f/R_{in} to determine its coefficient. There is no interaction between input signals or resistors. The circuit in Fig (3-B) combines features of the normal inverting amplifier and the non-inverting amplifier. Note that there are two resistors R_f as well as two R_{in} resistors. For correct circuit operation, it is important that the two pairs be matched. With the circuit shown, the equation for the output voltage is :

$$V_o = (R_f / R_{in})(V_2 - V_1)$$

Finally for LOG calculation of a signal, connect the circuit of Fig (3-C). Apply a variable signal (dc.) and watch the output. Apply different signals (like trinaglur wave, square wave), and record your results.

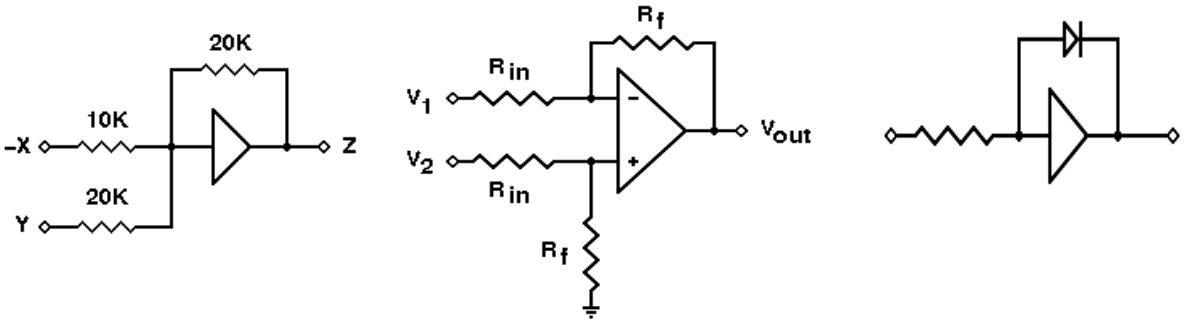


Fig. (3) : Addition, Subtraction of Two Signals, Log of a Signal.

(3) The Integrator and Differentiator

In Fig (4-A) is an integrator. If the input voltage is zero, no input current will flow. No feedback current can flow and the output voltage will remain constant. If the input is a non-zero value, equation for the output voltage becomes $V_{out} = -V_{in}/RC + K$, where R is the input resistance in ohms, C is the feedback capacitance in farads, and K is a fixed constant representing the accumulated voltage from the past. Still you can calculate the integration time contact.

Fig. (4) : (A) Integration : (B) : Differentiator of Signals

In Fig (3-B) we have a differnation circuit. Since the output voltage will reflect the input rate of change, this circuit will indeed do differentiation. The general equation for the output voltage is: Vo = (–RC) dV_{in}/dt The (d/dt) notation indicates differentiation with respect to time. Apply a suitable signal (AC signal) like square and triangular wave, and observe the effect of that.

(4) Discrete –Time Signal Generation : (Analog to Digital Conversion)
Continuous-Time Generation (Digital to Analog Conversion)

Consider the need to generate discrete time values of analog signals. The result would be stored as a two-bit binary number. The first step in making this determination might be a set of three comparators, connected as shown to the right. As the analog voltage increases, the comparators will, one by one from the bottom up, change state from false to true. Of course, additional digital circuitry will be required to encode these signals into the corresponding digital number. Apply an analog signal (say 5V, hence look at the output of the converter). The circuit in Fig (5-B) is a basic digital-to-analog (D to A) converter. It assumes a 4-bit binary number in Binary-Coded Decimal (BCD) format, using +5 volts as a logic 1 and 0 volts as a logic 0. It will convert the applied BCD number to a matching (inverted) output voltage. The digits 1, 2, 4, and 8 refer to the relative weights assigned to each input. Thus, 1 is the Least Significant Bit (LSB) of the input binary number, and 8 is the Most Significant Bit (MSB). Apply the right digital signals and observe the effect of that at the output of Fig (4-B).

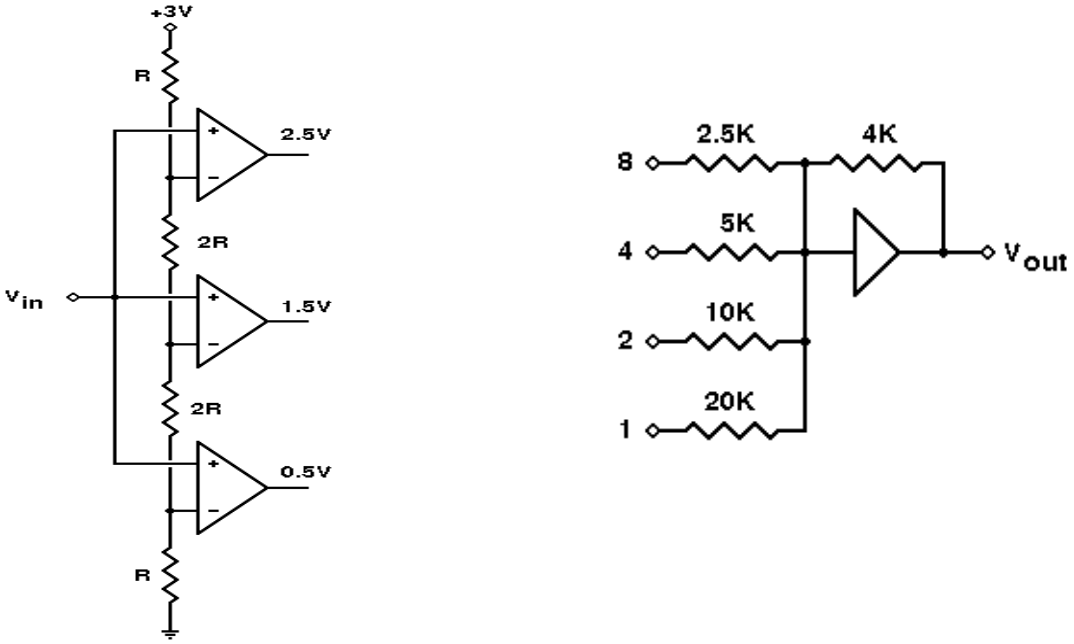


Fig. (5) : (A) : Signals Conversions : (B) : A/D and D/A converters.

Prepared by : Dr. Ebrahim Mattar

University of Bahrain
College of Engineering
Department of Electrical and Electronics Engineering
EEG 271 : Signals and Systems
Lab (3) : Convolution : Continuous and Discrete - Time Signals

Laboratory Objectives :
To learn the Convolution theory, hence do real-time measurement and simulation.

Needed Equipment :

- Variable Capacitors, Resistors, and
- Analog and digital Voltmeters (–30 to + 30) scale.
- Hand Calculator.
- Storage Oscilloscope.

Introduction to the Experiment:

Convolution is regard as a time-scale approach to calculate an output of a dynamic system y(t) due to an external input x(t). Homogeneity, additivity, and shift invariance (as properties of LTI) may sound abstract but they are very useful terms. To characterize a shift-invariant linear system, we need to measure merely one issue: the way the system responds to a unit impulse. This response is called the impulse response function of the system. Once we've measured this function, we can hence, predict how a system will react to any other possible inputs.

The manner we use an impulse response function is rather illustrated in (Fig. 1). We envision of the input stimulus, (a sinusoid in this case), as if it were the sum of a collection of impulses. Since we recognize the responses we would get if each impulse was presented separately (i.e., scaled and shifted copies of the impulse response), hence, simply add together all of the (scaled and shifted) impulse responses to predict how the system will respond to the complete stimulus. To accomplish that in a mathematical notation, our target is to show that the response (e.g., membrane potential fluctuation) of a shift-invariant linear system (e.g., passive neural membrane) is written as a sum of scaled and shifted copies of the system's impulse response function.

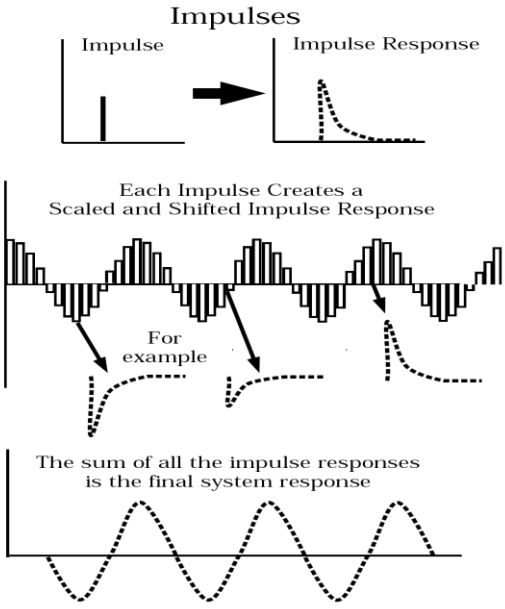


Fig (1) : Convolution theory Summation of Impulses

$$y(t) = \int_{-\infty}^{\infty} x(s)\, h(t-s)\, ds.$$

$$f_1[k] * f_2[k] = \sum_{m=-\infty}^{m=\infty} f_1[m] f_2[k-m], \qquad -\infty < k < \infty$$

(a) (b)

Fig (2) : RC circuit and assumed input signal x(t).

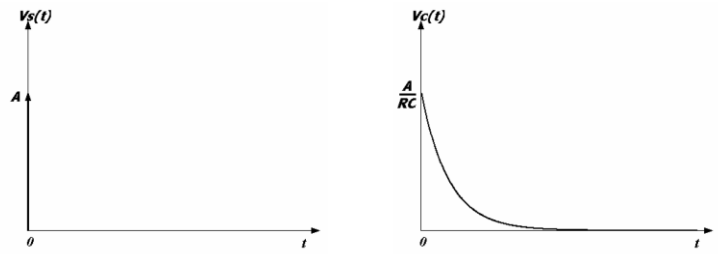


Fig (3) : Impulse response of an RC circuit.

Experiment Steps :

1) Connect the RC circuit shown in Fig (2-a). Chose reasonable values for R and C accordingly.
2) Apply and impulse input of strength (A). If you try to constrain the area of the impulse to a constant $A=V_{ptp}$, then as the pulse becomes narrower, the amplitude V_p increases, resulting in an impulse of strength (A). The response of an impulse of strength (A) is then given by :

$$vc = \frac{A}{RC} e^{\frac{-t}{RC}}$$

3) Using the storage oscilloscope, try to identify the measured signal of the impulse response (shape and time scale).
4) You should get a response identical to the one shown in Fig (3). For various values of R and C, the shape changes.
5) Now apply the signal defined and shown in Fig (2-b). This is a step signal with a delay. Tp to be decided by you as a designer of the problem.
6) Record your results using the storage oscilloscope accordingly. You should get a result similar to the one shown in Fig (4).

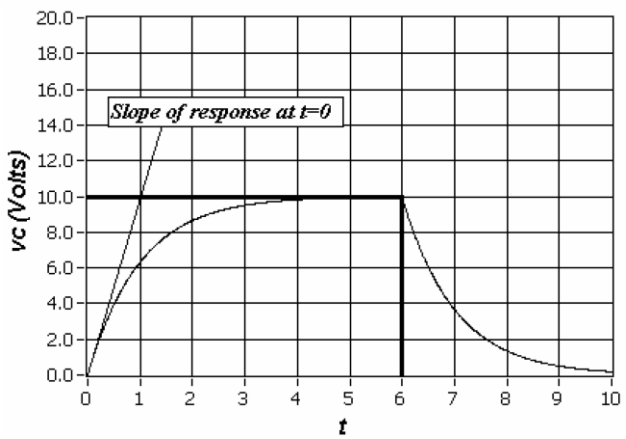


Fig (4) : Real-time response of an RC circuit due to the step signal shown in Fig (2-B)

7) Verify your results via the convolution approach. Use x(t) as : (show the steps of the computation),

$$x(t) = u(t) - u(t-2) \quad \text{and} \quad h(t) = e^{-t}\, u(t)$$

8) Verify your results via MATLAB. Hence, write a piece of code to do the compute the convolution for the shown RC circuit .
9) Use the ready-written Matlab (function) or Command as : y =conv(x, h). Show that the real-time measurement is just similar to the MATLAB simulation.
10) Finally, run the code list shown below. Do think that it does the convolution ? Compare it to the ready made function of cov in step (9). Show your results.
11) Can you make the discrete-time convolution version the continuous-time ? Which command you will use? Verify that via a simulation !!

% CONVOLUTION by code

```
fig_size = [232 84 774 624];
        t = 0:0.01:10;          % this is the time vector for simulation
        nt = length(t);
        dt = t(2);
                h = 2*t.*exp(-t) + exp(-2*t);          % this is an example of an impulse response function (try yours)
                x = 1 - exp(-1.5*t);                   % this is an example input signal,  use the shifted step function

                % plotting of the impulse and input signals
```

```
figure(1),clf,plot(t,x,t,h),grid,xlabel('Time (s)'),ylabel('Amplitude'),...
title('Input x(t) and Impulse Response h(t)'),...
text(2.5,0.95,'x(t)'),text(2.5,0.45,'h(t)'),set(gcf,' Position', fig_size)


figure(2),clf,plot(t,x,2.5-t,h),grid,xlabel('Time Tau (s)'),ylabel('Amplitude'),...
title('Input x(tau) and Impulse Response h(t-tau) plots, t = 2.5 s'),...
text(2.5,0.95,'x(tau)'),text(2.25,0.65,'h(t-tau)'),set(gcf,'Position',fig_size)


% Now computing the calculation of the product of  x(tau) & h(t-tau)  for t = 2.5 seconds only
%
tau = t(1:251);
  tmtau = 2.5 - tau;
    hx = (1 - exp(-1.5*tau)) .* ( 2*tmtau.*exp(-tmtau) + exp(-2*tmtau) - exp(-3*tmtau) );
%
figure(3),clf,plot(t(1:251),hx),grid,xlabel('Time Tau (s)'),ylabel('Amplitude'),...
title('Product of the Input x(tau) and Impulse Response h(t-tau), t = 2.5 s'),...
text(0.75,0.25,  'Value of y(t)  at  t = 2.5 s equals the area under this curve'),...
set(gcf,'Position',fig_size)



%  Still you can get similar results using the Transfer function con concepts ...

[numh,denh] = residue([0;2;1;-1;],[-1;-1;-2;-3],0); % calculation of transfer function
  [numx,denx] = residue([1; -1],[0; -1.5],0); % calculation of input transform
    [numy,deny] = series(numx,denx,numh,denh); % calculation of output transform
%

[resy,poly,ky] = residue(numy,deny); % partial fraction expansion for output
%
y1 = lsim(numh,denh,x,t); % calculation of output using "lsim"
figure(4),clf,plot(t,y1,t,x,t,h,t(251),y1(251),'r*'),grid,xlabel('Time (s)'),ylabel('Amplitude'),...
 title('Input x(t),Impulse Response h(t), and Output y(t)'),...
text(2.5,0.9,'x(t)'),text(3.5,0.3,'h(t)'),text(3.5,1.6,'y(t)'),...
text(2.7,y1(251),sprintf('y(t=2.5) = %g',y1(251))),set(gcf,'Position',fig_size)



y2 = conv(x,h) * dt; % calculation of output using "conv"
y2 = y2(1:nt);
y3 = 0;


figure(5),clf,plot(t,y1,'g-',t,y2,'r-.'),grid,xlabel('Time (s)'),ylabel('Amplitude'),...
title('Output y(t) from LSIM and from CONV'),set(gcf,'Position',fig_size)
```

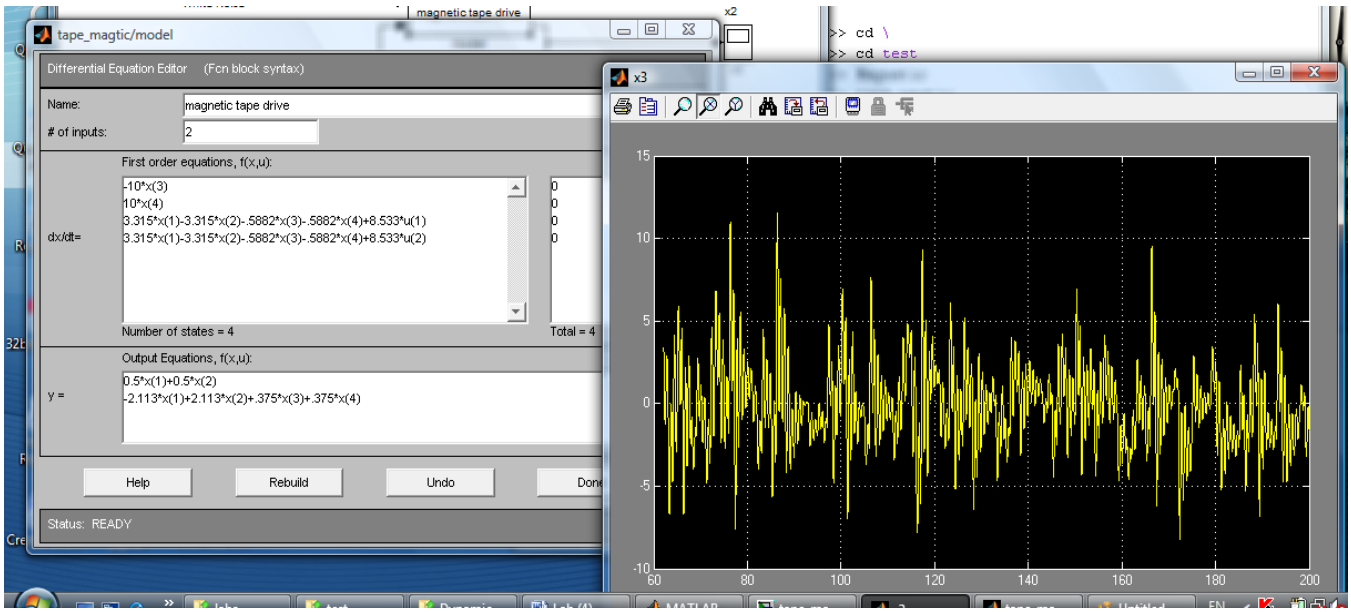
Prepared by : Dr. Ebrahim Mattar

University of Bahrain
College of Engineering
Department of Electrical and Electronics Engineering
EEG 271

Lab No. : [4] Dynamic System Differential Equation (Differential Equation and State Space Description)

Experiment Objectives :

- To learn how to simulate a linear and nonlinear system suing the DEE.
- To study in depth the time domain behavior of dynamic systems and relate that to differntional equation behavior.



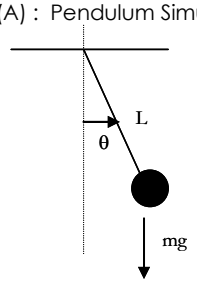
]

Equipment Needed :

- Laptop or Desktop Personal Computer.
- Printer.
- Matlab as a simulation environment.

Steps :

Part (A) : Pendulum Simulation



The motion of a frictionless pendulum of length L is governed by the differential equation, (where τ_{in} is the input toque to the joint) :

$$(\partial^2\theta/\partial t^2) = (-g/L)\sin(\theta) + \tau_{in}$$

(Can you verify that !!). Refer to the basic physics for that. Here, θ is the angle (in radians) that the pendulum makes with the vertical, g = 32.2ft/sec2, and L is the length of the pendulum. Although this a non-linear system, as you will see below, linear systems are very important in its analysis. For undamped dynamic system, assume that (L = g) so that the equation simplies to :

$$(\partial\theta/\partial t) = -\sin(\theta) + \tau_{in}$$

This equation is equivalent to the following state space description : (let $x_1 \rightarrow \theta$, and $(\partial x_1/\partial t) = \theta$), then we have :

$$(\partial x_1/\partial t) = x_2$$
$$(\partial x_2/\partial t) = -\sin(x_1) + \tau_{in}$$

(1)

(2) Verify the differntioanl equation of the shown above system (in state space). What is the order of the system. Is this a damped dynamic system ?

(3) Now using the DEE, simulate the system behavior. Show that the only equilibrium points for this system are (y; v) = (k_; 0) where k runs over the set of integers. You need some help in that !!

(2) Construct the phase plane portrait for system (2) using the range : $(-4 \le \theta \le 10)$, $(-6 \le (\partial\theta/\partial t) \le 6)$.

(4) For small θ, siny _ y. Thus, we expect that for small θ, the solutions to system should have similar behavior to the solutions to the following linear system.

(5) Repeat the DEE simulation for the damped pendulum system as :

$$(\partial x_1/\partial t) = x_2$$
$$(\partial x_2/\partial t) = -\sin(x_1) + (x_2) + \tau_{in}$$

(6) Observe the two different behavior for the damped and undamped motion.

(7) Compute the eigenvalues and eigenvectors of the A matrix. Can you do that ?

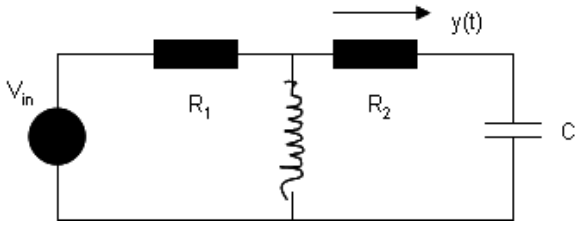
(8) You can get MATLAB to compute the eigenvalues and eigenvectors. Type \help eig" in the MATLAB Command widow for instructions.

(9) There is also an equilibrium at $(\pi; 0)$. To analyze this equilibrium, let $x_1 = (\theta - \pi)$. Then, $x_{10} = \theta_0$ so system is equivalent with :

$$(\partial x_1/\partial t) = x_2$$
$$(\partial x_2/\partial t) = -\sin(x_1 + \pi) = \sin(x_1)$$

Part (B) : Electrical System Differential Equation Simulation

Write the differntional equations (in state space) for the following RLC circuit system expressing the relation between y(t) and the input $V_{in}(t)$. You should get the followings :

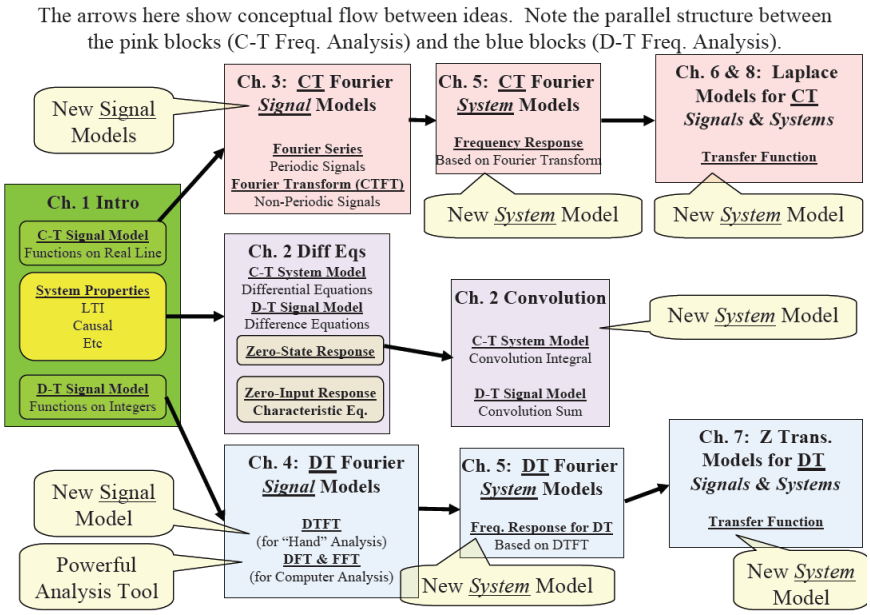


- What is the order of this dynamic system ? verify your justification accordingly !!
- Write the Differential equation for the shown above dynamic system. This will be a relation between y(t) and Vin(t).
- For a selected value of parameters (refer to the Lab instructor), now using the DEE toolbox (via Matlab), verify your results of simulations through exciting the system with a : Step input, Ramp input, a Unit impulse, and a Sinusoidal signal with a given periodic frequency.
- Verify and record yours results. What are the equilibrium states ? Show your hand calculations .
- Verify the results via hand calculation. Do think DTT results coincides with your hand calculations for such different excitations ?
- Do think the above RLC system is stable ? Can you get such an information from the Differential equation ?

University of Bahrain
College of Engineering
Department of Electrical and Electronics Engineering
EEG 270 : Signals and Systems

Lab (5) : MATLAB Convolution for Signals and Systems

Experiment Objectives : Gain Experience with MATLAB advanced tools. This will be done in three main parts.



(Part No. 1) : Linear Convolution of Signals :

You are given a pair of sequences, use discrete convolution to find the response to the input x[n] of the linear time-invariant system with impulse response h[n].

x[n]= square wave with amplitude 1.
y[n]= triangle wave with amplitude 2.

- For both signals, use PLOT command to plot the input and impulse response.
- Compute the convolution by hand, use MATLAB to plot the results.
- Write a MATLAB function to compute the convolution of the two finite-length sequences and plot the results.
- For length N input vector x, the DFT is a length N vector X, with elements N
- X(k) = sum x(n)*exp(-j*2*pi*(k-1)*(n-1)/N), 1 <= k <= N.
- n=1
- Use CONV command to verify the results from b).

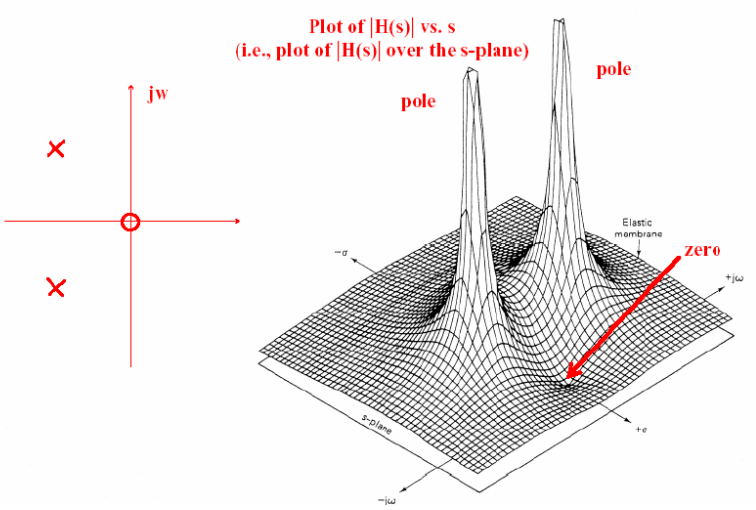
(Part No. 2): Low-pass Filters Design :

In this section of the tutorial, you will create a signal with added high frequency noise. Later in the tutorial, you will use a low-pass filter to eliminate high frequency noise.

- Load Matlab.
- You can hear a voice say "MATLAB." This is the signal to which you will add noise soundsc (mtlb,Fs).
- Create a noise signal noise = cos(2*pi*3*Fs/8*(0:length(mtlb)-1)/Fs)'; (You can hear the noise signal by typing soundsc(noise,Fs)). (You can also use random function to introduce noise).
- Add the noise to the original signal u = mtlb + noise.
- Scale the signal with noise ….. u = u/max(abs(u)); (You scale the signal to try to avoid overflows later on. You can hear the scaled signal with noise by typing soundsc(u,Fs)).
- Display the frequency spectrum using FFT.
- View the scaled signal with noise specgram(u,256,Fs);colorbar (In the spectrogram, you can see the noise signal as a horizontal line at about 2800 Hz, which is equal to 3*Fs/8).
- Use low-pass filters to eliminate high frequency . Just implement the below shown program.

```
b = ones(1,10)/10;%
10 point averaging filter fy = filtfilt(b,1,x);  %
Noncausal filtering fyy = filter(b,1,x);  %
Normal filtering %
```

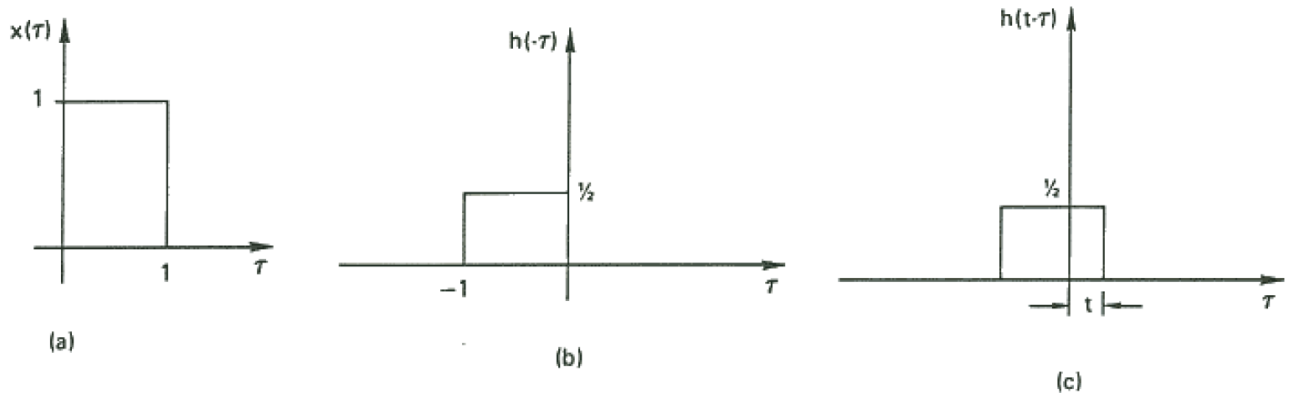
Use FFT to plot the power spectrum of the filter signal and compare it to both, the original and the corrupted signals.

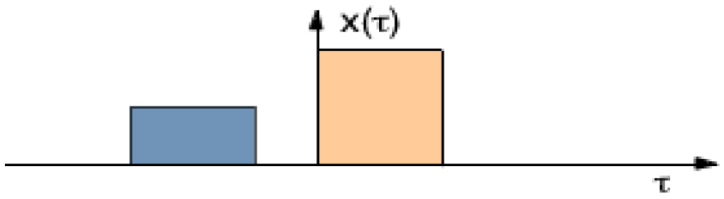


(Part No. 3): Real World Practical Application

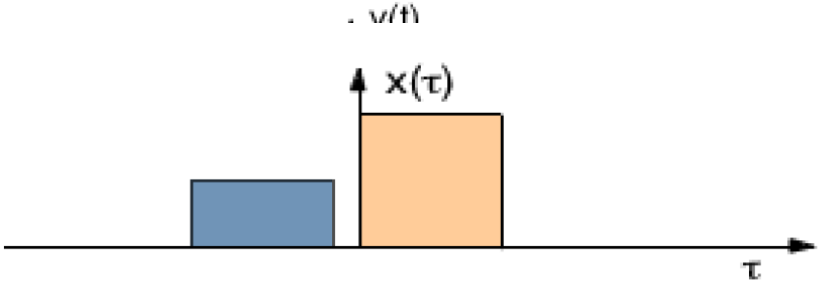
In this section, you will be provided an ECG (Electrocardiogram, as taken from Dr. Sami Nassimi) signal with noise. Your task, here, is to create a Matlab program to determine the frequency of the noise and eliminate the noise signal.

Continuous Convolution

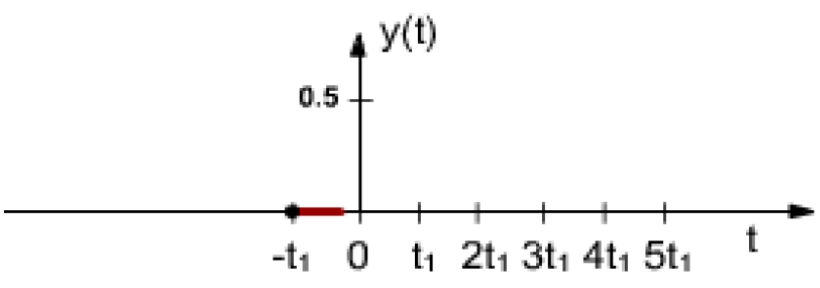


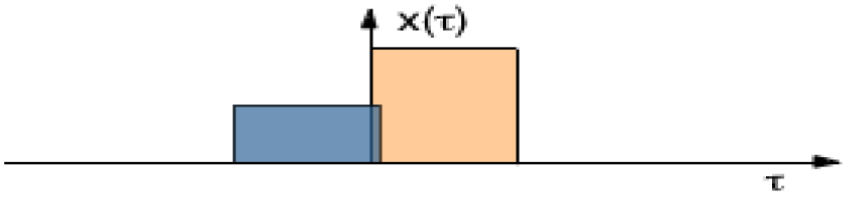


1

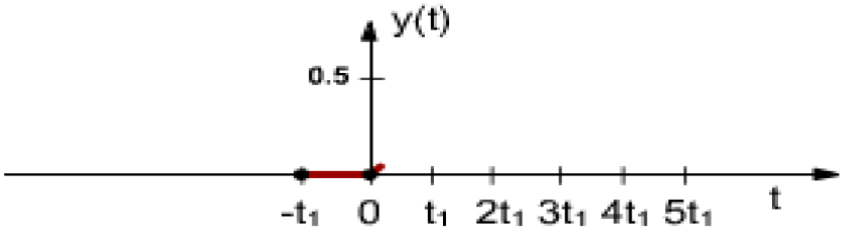


2



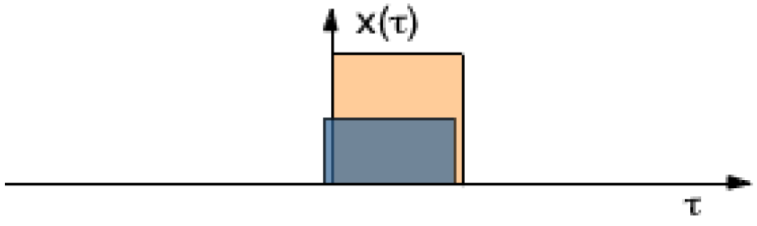

x(τ)
τ

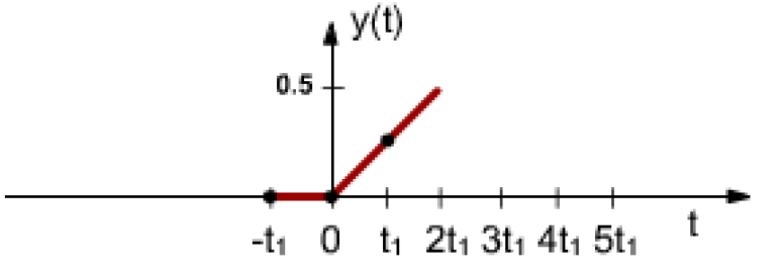
3


y(t)
0.5
-t1
0
t1
2t1
3t1
4t1
5t1
t


x(τ)

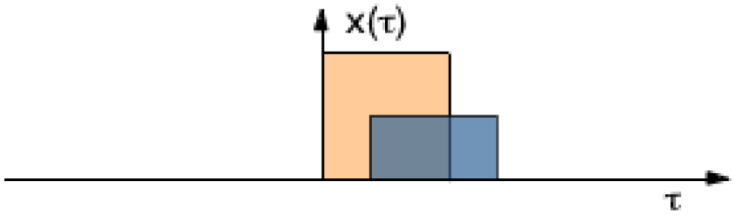
4

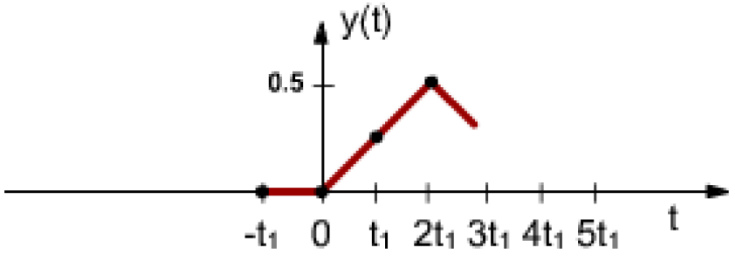

x(τ)
τ


y(t)
0.5
-t1
0
t1
2t1
3t1
4t1
5t1
t

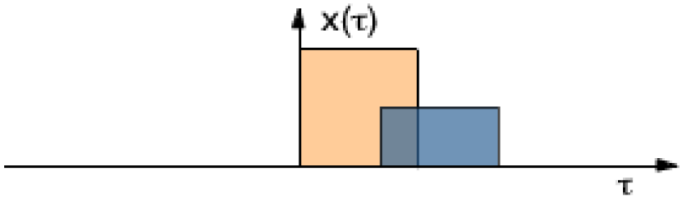
5

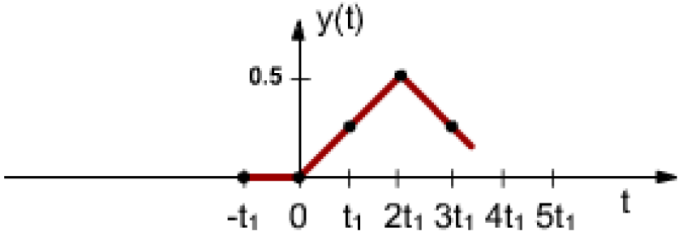
6





7





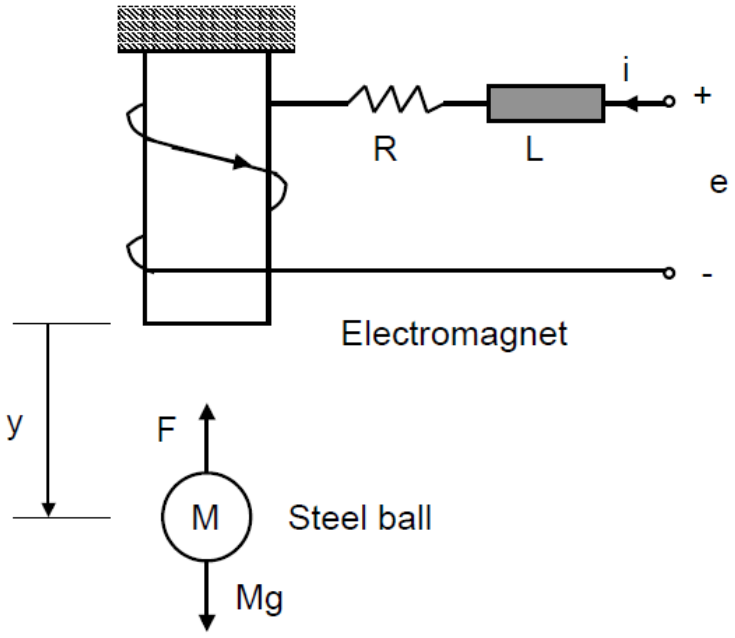
University of Bahrain
College of Engineering
Department of Electrical and Electronics Engineering
EEG 271: Signals and Systems

Assignment No. [6] with Computer

(10 Marks)

Use DEE in Matlab or any other differential equation solver in Matalb or (C++) or any other programming language to simulate the following:

The Figure below shows the diagram of a magnetic ball-suspension system, where the objective of the system is to observe the position of the steel ball by adjusting the current.



- Write the differential equation for the system, relating the input e(t) with the ball position y(t).
- What is the order of the system ?
- Simulate the system using the DEE . Show how the ball is moving with time for s unit step input.

Prepared by
Dr. Ebrahim Mattar