Laboratory Manual for

EL405 - Applied Mathematics

B. Tech.

SEM. IV (EC)

Department of Electronics & Communication
Faculty of Technology
Dharmsinh Desai University
Nadiad

TABLE OF CONTENT

PART- I LABORATORY MANUAL

P A R T I

LABORATORY MANUAL

EXPERIMENT – 1

FOURIER SERIES OF PERIODIC WAVEFORM

OBJECTIVE: To observe the effect of truncation of Fourier series which comprise of different frequency components of sine wave.

THEORY:

If you listen to music you may have noticed that you can tell what instruments are used in a given song or symphony. In some cases, the melody is sequentially played by different instruments. For example, you may hear it played by violins and a little later repeated by flutes. Although the violins and the flutes may play the same exact notes, you can definitely tell when the violins or the flutes are playing. The reason is that, when two different instruments play the same note, the pitch or frequency of the sound waves are the same, but the shape of the sound wave is different. For example, the sound wave of instrument one may have the shape of a sine function while the sound wave of the instrument two may be a square wave.

Looking at time record of the sound wave of different musical instruments may be interesting on its own right but it is strictly a qualitative exercise. The question is, can we quantitatively characterize such waves or, in other words, write them as mathematical expressions. In case of instrument one, we can write $y = \sin(2\pi ft)$ and in case of instrument two, it is harder to write an expression valid for any time. The best we can do is write an expression for one period as,

$$y = \begin{cases} A, & 0 \le t < T/2 \\ -A, & T/2 \le t < T \end{cases} \quad (1)$$

and then announce it is periodic. Obviously, this approach gets to be hopeless as waves get more complex.

In 1807, Joseph Fourier proposed the first systematic way to answer the question above. He stated that a completely arbitrary periodic function $g(t)$ could be expressed as a series of the form,

$$g(t) = a_0 + \sum_{n=1}^{\infty}[a_n \cos(2\pi n f_0 t) + b_n \sin(2\pi n f_0 t)] \quad (2)$$

Where f_0 is the fundamental frequency of periodic function *g(t)*.

This generalized Trigonometric Fourier series show that just as vector can be represented as a sum of its components in a variety of ways, depending on the choice of a coordinate system (rectangular, spherical, cylindrical etc), a signal can be represented as a sum of its components

using set of trigonometric (sinusoids) signals since they form a set of mutually orthogonal signals. Many other signal sets like exponential functions, Walsh functions, Bessel functions etc also form the orthogonal sets, hence Fourier series can also expressed in above specified functions.

Now the task is just to determine the Fourier coefficients a_0, a_n and b_n . These coefficients in order can be obtained by multiplying eq.(1) by 1, $cos(2\pi nf_0t)$ and $sin(2\pi nf_0t)$ and integrating over one time period T_0 respectively. Hence we obtain

$$a_0 = \frac{1}{T_0}\int_{t_1}^{t_1+T_0} g(t)dt \tag{3}$$

$$a_n = \frac{2}{T_0}\int_{t_1}^{t_1+T_0} g(t)\cos(2\pi nf_0t)dt \tag{4}$$

$$b_n = \frac{2}{T_0}\int_{t_1}^{t_1+T_0} g(t)\sin(2\pi nf_0t)dt \tag{5}$$

for n = 1,2,3……

So any periodic signal can be represented by the combination of sinusoids of different frequencies. Here we take an example of square wave with peak voltage of V having no dc components and is even function of time. Hence $a_0 = 0$, $b_n = 0$ and only a_n will exists which can be found as

$$a_n = \begin{cases} \frac{4V}{n\pi}, & n = 1,5,9,13,\ldots \\ \frac{-4V}{n\pi}, & n = 3,7,11,15,.. \end{cases} \tag{6}$$

Fourier series expansion of square wave can be represented by

$$g(t) = 4V/n\pi(cos\omega_0t - cos\,3\omega_0t + cos\,5\omega_0t - cos\,7\omega_0t + cos\,9\omega_0t)\ldots \tag{7}$$

where,$\omega_0 = 2\pi f_0$ is angular fundamental frequency, V is peak amplitude and *n* is number of harmonics. Therefore square wave can be represented as weighted sum of different cosine waves. Infinite number of cosine wave addition makes ideal square wave.

Imagine the surprise of many with such proposal stating that even discontinuous functions such as square waves could be represented by beautifully/smooth sines and cosines! But if you see closely the waveform obtained, the series doesn't converge when there is a jump discontinuity. At the point of discontinuity the series takes the average value of the left-hand and right-hand limits of the signal at the instant of discontinuity.

SAMPLE PROGRAM:

```
v=1;
w=15;
t=0:0.001:0.99;
```

```
x1=v*(cos(w*t)-cos(3*w*t)/3)/%pi;
x2=v*(cos(w*t)-cos(3*w*t)/3+cos(5*w*t)/5)/%pi;
x3=v*(cos(w*t)-cos(3*w*t)/3+cos(5*w*t)/5-cos(7*w*t)/7)/%pi
x4=v*(cos(w*t)-cos(3*w*t)/3+cos(5*w*t)/5-cos(7*w*t)/7+cos(9*w*t)/9)/%pi;
x5=v*(cos(w*t)-cos(3*w*t)/3+cos(5*w*t)/5-cos(7*w*t)/7+cos(9*w*t)/9-cos(11*w*t)/11)/%pi;
x6=v*(cos(w*t)-cos(3*w*t)/3+cos(5*w*t)/5-cos(7*w*t)/7+cos(9*w*t)/9-
cos(11*w*t)/11+cos(13*w*t)/13)/%pi;
subplot(3,2,1);
plot(x1);
title('Square Wave Contructed with First 2 Components of Fourier Series');
xlabel('Samples');
ylabel('Amplitude (V)');
subplot(3,2,2)
plot(x2);
title('Square Wave Contructed with First 3 Components of Fourier Series');
xlabel('Samples');
ylabel('Amplitude (V)');
 subplot(3,2,3)
plot(x3);
title('Square Wave Contructed with First 4 Components of Fourier Series');
xlabel('Samples');
ylabel('Amplitude (V)');
subplot(3,2,4)
plot(x4);
title('Square Wave Contructed with First 5 Components of Fourier Series');
xlabel('Samples');
ylabel('Amplitude (V)');
 subplot(3,2,5)
plot(x5);
title('Square Wave Contructed with First 6 Components of Fourier Series');
xlabel('Samples');
ylabel('Amplitude (V)');
 subplot(3,2,6);
plot(x6);
title('Square Wave Contructed with First 7 Components of Fourier Series');
xlabel('Samples');
ylabel('Amplitude (V)');
///---Input Parameters---//
//v==> voltage
//w==>Frequency
```

CONCLUSION:

EXPERIMENT – 2

NUMERICAL METHOD

OBJECTIVE: To implement a Newton Raphson Method for finding a root of a function using SCILAB.

THEORY:

The root of a function $f(x)$ is the value of x for which $f(x)$ become zero. Consider the function $f(x)$ shown in Fig.2.1. Suppose x_1 is the initial guess of root. A tangent drawn at x_1 intersects the x axis at x_2. The slope of this tangent line can be written as:

$$slope = \frac{f(x_1)-0}{x_1-x_2} \quad (1)$$

This is equal to $\frac{df(x)}{dx}$ or $f'(x)$ at x_1 .

Thus,

$$f'(x_1) = \frac{f(x_1)-0}{x_1-x_2} \quad \text{or } x_2 = x_1 - \frac{f'(x_1)}{f(x_1)} \quad (2)$$

Thus, by using initial guess x_1 in above equation, we can get better approximationx_2 to the root. And by using x_2 as next guess we can get more close approximation x_3 . This process can be repeated as:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f(x_n)} \quad (3)$$

Until a sufficiently accurate value is reached.

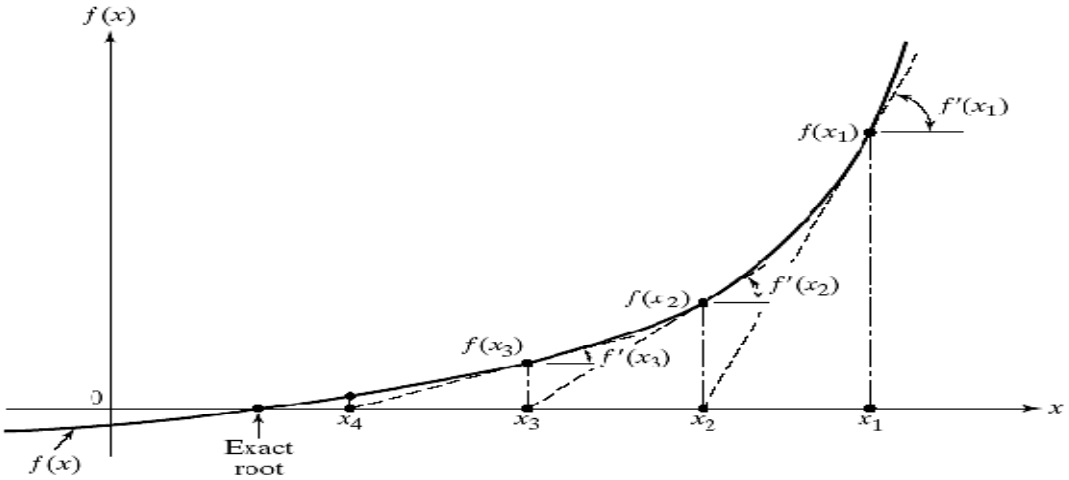

Fig.2.1 Graphical depiction of the *Newton-Raphson* method.

This method can find a complex root also, however the initial guess must be complex.
The Newton-Raphson method is not guaranteed to find a root. For example, if the starting point x_1 is sufficiently far away from the root for the function $f(x) = \tan^{-1} x$, the function's small slope tends to drive the x guesses further and further away from the root.

EXAMPLE:

Solve $f(x) = x^3 + 4x^2 - 10$ using the the *Newton-Raphson* method for a root in [1, 2].

SOLUTION:

From the formula , $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

$f(x_n) = x_n^3 + 4x_n^2 - 10 \Rightarrow f'(x_n) = 3x_n^2 + 8x_n$

$x_{n+1} = x_n - \frac{x_n^3 + 4x_n^2 - 10}{3x_n^2 + 8x_n}$

Using the initial guess, $x_n = 1.5$, x_{n+1} is estimated as

$x_{n+1} = 1.5 - \frac{1.5^3 + 4 \times 1.5^2 - 10}{3 \times 1.5^2 + 8 \times 1.5} = 1.3733$ with error rate calculated as 0.0000

SAMPLE PROGRAM:

```
Clear all;
function[y, deriv]=fcn_nr(x)
y=x^3+4*(x)^2-10;
deriv=3*(x)^2+8*x;
endfunction
xp=-15:1:15;
n=length(xp);
for  i=1:n
yp(i)=fcn_nr(xp(i));
end
plot(xp,yp);
xlabel('x');
ylabel('y');
title('Plot of function');
x=input("Enter initial guess of root location: ");
itermax=10;// % max # of iterations
iter=0;
errmax=0.00001;//  % convergence tolerance
error1=1;
while   error1>errmax &  iter<itermax
iter=iter+1;
```

```
[ffprime]=fcn_nr(x);
if ffprime==0
break;
end;
xnew=x-f/fprime;
error1=abs((xnew-x)/xnew)*100;// % find change from previous
x=xnew;// % set up for next iteration
disp('Iteration No.')
disp(iter);
disp('the estimated value of the variable x is=');
disp(x);
end
```

RESULTS:

Enter initial guess of root location: 2

Table.2.1 Estimation of Root

Iteration No	***Estimated Root***	***% Error in Estimation***
1	*1.5*	*33.33*
2	*1.373333*	*9.22*
3	*1.365262*	*0.5911919*
4	*1.36523*	*0.0023440*
5	*1.36500*	*$3.677 X 10^{-08}$*

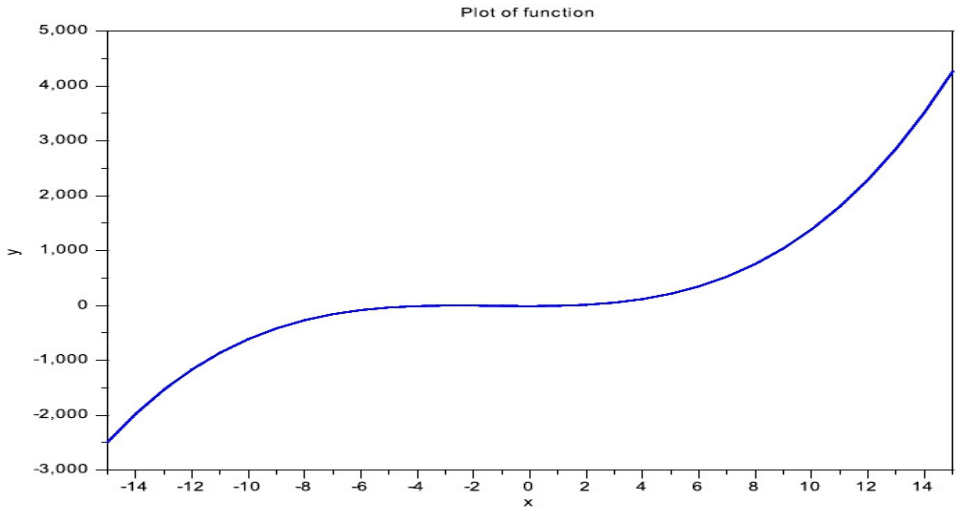

Fig.2.1 Estimation of Root

MODIFICATION:

Repeat the same with other iterative methods :(1) False Position Method (2) Secant method

CONCLUSION:

EXPERIMENT – 3

NODE VOLTAGE & MESH CURRENT ANALYSIS

OBJECTIVE: To implement node voltage (KCL) and mesh current (KVL) analysis of a network.

THEORY:

Node Voltage Analysis:

Nodal analysis employs Kirchhoff's current law (KCL). In a network having N-nodes, one node is selected as a reference or ground node and the voltages at all the remaining N–1 nodes are measured with respect to this reference node. Applying KCL on a network will produce N-1 linearly independent KCL equations. Since variables contained in KCL equations are the unknown node voltages of the network, its solution will provide value of the unknown node voltages. KCL equations can be solved using matrix analysis and it can be implemented in SCILAB.

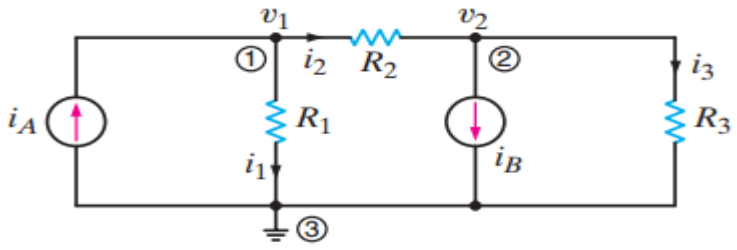

Fig.3.1 Circuit Diagram

KCL equations of the networks in the form of nodal voltages are

$$V_1\left(\frac{1}{R_1}+\frac{1}{R_2}\right)-V_2\left(\frac{1}{R_2}\right)=i_A \quad (1)$$

$$-VI\left(\frac{1}{R_2}\right)+V_2\left(\frac{1}{R_2}+\frac{1}{R_3}\right)=-i_B \quad (2)$$

Equations in the matrix form is

$$V = GI \quad (3)$$

Where,

$$G = \begin{bmatrix} \frac{1}{R_1}+\frac{1}{R_2} & \frac{-1}{R_2} \\ \frac{-1}{R_2} & \frac{1}{R_2}+\frac{1}{R_3} \end{bmatrix}, V = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}, \text{and } I = \begin{bmatrix} i_A \\ i_B \end{bmatrix}$$

The solution of matrix equation is

$$V = G^{-1} I \tag{4}$$

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4K} & \frac{-1}{6K} \\ \frac{-1}{6K} & \frac{1}{3K} \end{bmatrix}^1 \begin{bmatrix} 1\times 10^{-3} \\ 4\times 10^{-3} \end{bmatrix} \tag{5}$$

Task:

To implement node voltage analysis of any network in a SCILAB with **G** and **I** matrices as an input argument and node voltage matrix **V** as a solution of a network.

Mesh current analysis:

Loop analysis employs Kirchhoff's voltage law (KVL). In a network having N-nodes and B-Braches, on applying KCL on a network will produce B+N-1 linearly independent KVL equations. Since variables contained in KVL equations are the unknown mesh currents of the network, its solution provides value of the unknown mesh currents. KVL equations can be solved using matrix analysis and it can be implemented in SCILAB.

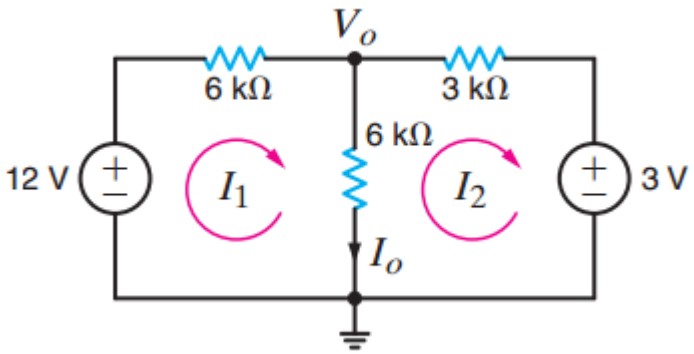

Fig.3.2 Circuit Diagram

KVL equations of the networks in the form of mesh currents are

$$i_1(R_1+R_2) - i_2(R_2) = v_{s1} \tag{6}$$

$$-i_1(R_2) + i_2(R_3+R_2) = v_{s2} \tag{7}$$

Equations in the matrix form is

$$IR = V \tag{8}$$

where,

$$I = \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}, \quad R = \begin{bmatrix} R_1 + R_2 & -R_2 \\ -R_2 & R_3 + R_2 \end{bmatrix}, \quad V = \begin{bmatrix} v_{s1} \\ v_{s2} \end{bmatrix}$$

The solution of matrix equation is ,

$$I = R^{-1} V \tag{9}$$

$$\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} R_1 + R_2 & -R_2 \\ -R_2 & R_3 + R_2 \end{bmatrix}^{1} \begin{bmatrix} v_{s1} \\ v_{s2} \end{bmatrix} \tag{10}$$

With values specified in figure, solution becomes

$$\begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} 1.25 \\ 0.5 \end{bmatrix} \tag{11}$$

Task:

To implement mesh current analysis of any network in a SCILAB with **R** and **V** matrices as an input argument and node current matrix **I** as a solution of a network.

SAMPLE PROGRAM (KCL):

```
v1=input('Enter the voltage in loop-1 (in V):');
i2=input('Enter the current through loop-2 (in A):');
r2=input('Enter the resistance in series with V1 in loop-1 (in Ohm):');
r1=input('Enter the resistance between v2 & GND. (a common element between loop-1 & 2) (in Ohm):');
//The node voltage at node-2 can be solved as
temp1=(v2-v1)/r2;
temp2=(v2/r1)'
temp3=i2*r1*r2;
temp4=v1/r2;
temp5=temp4-temp3;
temp6=temp5*r1*r2;
v2=temp6/(r1+r2);
disp('Voltage at Node-2 V2=');
disp(v2)
//---Input Variables---//
//v1==>equivalent voltage in loop 1 --constant
//i2==>value of the current source in loop 2--constant
//r2==>value of the resistance in series with V1 in loop 1-- constant
//r1==>value of the resistance in the common branch in loop 1 & loop 2—constant
```

SAMPLE PROGRAM (KVL):

```
//number of loops in the networks=2
v1=input('Enter the voltage in loop-1 (in V):');
v2=input('Enter the voltage in loop-2 (in V):');
r1=input('Enter the resistance in series with V1 in loop-1 (in Ohm):');
r2=input('Enter the resistance between v2 & GND. (a common element between loop-1 & 2) (in Ohm):');
r3=input('Enter the resistance in series with V2 in loop-2 (in Ohm):');
//The mesh current can be solved as
//v1=(r1+r2)*i1-r2*i2;
//-v2=-r2*i1+(r2+r3)*i2;
//[v]=[r][i];
//[i]=[v][R]; where [R]=inv([r]);
v=[v1;v2];
r=[r1+r2 -r2 ; -r2 r2+r3];
//i=[i1;i2];
R=inv(r);
i=R*v;
disp('the current through the loops=');
disp(i);
//--Input variables--//
//v1==>voltage in loop 1--constant
//v2==>voltage in loop 2--constant
//r1==>resitance in loop 1 -- constant
//r2==>resitance in the common branch of loop1 & loop 2 --constant
//r3==>resitance in loop 2 – constant
```

CONCLUSION:

EXPERIMENT – 4

FILTER DESIGN USING LAPLACE TRANSFORM

OBJECTIVE: Design and analysis of analog filter using Laplace transform.

THEORY:

The filter circuits are passing or stop the different range of frequency in circuit or system. Electronic filters are electronic circuits which remove unwanted frequency components from the signal and select the wanted signal like in radio receiver. Implementations of linear filters are based on combinations of resistors (R), inductors (L) and capacitors (C).

The purpose of this experiment is to study the frequency response of analog filter. The frequency response is a graph of frequency versus output voltage or voltage gain which indicates characteristics of circuit or system.In RC circuit (Fig. 1(a)), when frequency is zero then capacitive reactance is infinite and the voltage maximum and equal to supply voltage as per the equation,

$$Xc = \frac{1}{2\pi fc} \tag{1}$$

As frequency is increased capacitive reactance decreases hence voltage across capacitor gradually decreases. For very high frequency the voltage across capacitor becomes to zero. The frequency response of RC circuit is shown in Fig. 1(b).

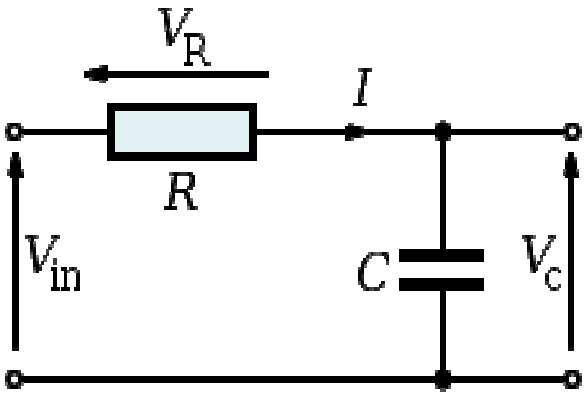

Fig.4.1 RC Circuit

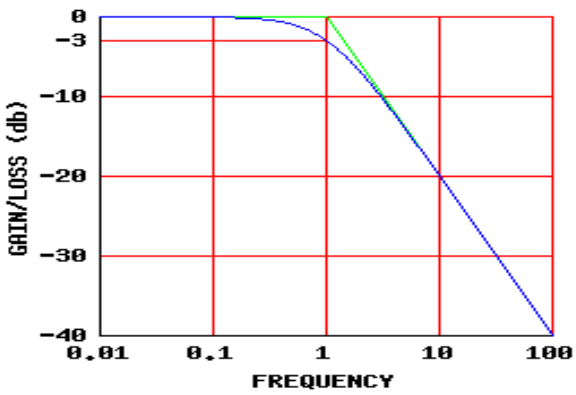

Fig.4.2 Frequency Response

Calculations:

RC Circuit:

By viewing the circuit as a voltage divider as shown in Fig.4.1, the voltage across the capacitor is:

$$Vc = \frac{\frac{1}{jwC}}{R + \frac{1}{jwC}} \times Vin = \frac{1}{1 + jwRC} \times Vin \quad (2)$$

$$Amplitude = \frac{1}{\sqrt{(1 + (wRC)^2}} \quad (3)$$

$$Phase = -\tan^{-1}(wRC)$$

$$A_v = \frac{V_c}{V_{in}} = \frac{1}{1 + sRC} = \frac{1/RC}{s + 1/RC} \quad (4)$$

Hence cut off frequency,

$$w = \frac{1}{RC} \ \& f = \frac{1}{2\pi RC} \quad (5)$$

Task:

To implement different kind of passive filter using R, L and C of different order in a Scilab with the coefficient of the numerator and denominator polynomial in decreasing power order as an input argument and output signal *y* as the filtered signal.

Useful Function in SCILAB

(1) *filter* – Filter a Data Sequence
(2) *analpf* – Analog Low Pass Filter
(3) *freq & repfreq* – Frequency Response
(4) bode – Frequency Response using Bode Plot
(5) dbphi – Frequency Response to Phase & Magnitude Representation
(6) buttmag – Butterworth Filter using Filter Order & Cut-off Angular Frequency
(7) buttmag – Butterworth Filter using Filter Order & Cut-off Angular Frequency

SAMPLE PROGRAM:

```
x=input('Enter the input data sequence');
b=input('Enter the co-effitients for the numerator polynomial:');
a=input('Enter the co-effitients for the denominator polynomial:');
y=filter(b,a,x);
disp('The input signal is=');
disp(x);
disp('The filtered signal is=');
disp(y);
```

CONCLUSION:

EXPERIMENT – 5

CORRELATION

OBJECTIVE: To perform correlation of two discrete time signal.

THEORY:

(1) To understand operation of correlation.

Correlation is a measure of similarity between two signals and is found using a process similar to convolution. The discrete correlation (denoted **) of x[n] and h[n] is defined by ,

$$r_{xy}[n] = x[n] ** h[n] = \sum_{k=-\infty}^{\infty} x[k]h[k-n] \quad (1)$$

Correlation is equivalent to performing the convolution of x[n] with flipped signal h[-n].

SAMPLE PROGRAM:

```
Seq_x=[1 2 3];
Seq_y=[1 2 3];
New_m=length(x);
New_n=length(y);
for k =1: (New_m + New_n-1)
   Final_w(k)=0;
      for j=max(1,k+1-New_n):min(k,New_m)
         Final_w(k)=Final_w(k)+(Seq_x(j)*Seq_y(New_n-k+j));
      end
end
bar(Final_w);
title('correlation');
xlabel('Discrete Time (n)');
ylabel('amplitude');
```

(2) Application of correlation in radar signal.

Correlation finds application in RADAR Signal Processing wherer objective is to find whether the object is presence or absent. It is also required to find distance between object and radar. Following is explanation for the same.

Let, x[n] Transmitted discrete signal from RADAR,
D Received Signal Delay (if object is found)
x[n-D] Delayed received signal
a Attenuation factor(represents attenuation in signal due to environment)
noise Additive noise of medium

Then,
Final received signal(if object is found),
R[n]=a*X[n-D]+noise (2)
If we find the correlation between received signal and transmitted signal, than the correlation value is maximum at the delay index.

Final received signal(if object is not found),
In the Absence of object the received signal is only the Noise. R(n) = Noise.
if we find the correlation between received signal and transmitted signal, than the correlation value is not higher at the delay index the way we get in presence of object.

SAMPLE PROGRAM:

```
clc;
clear;
xdel(winsid());
x=[0 1 2 3 2 1 0];              //Triangle pulse transmitted by radar
n=[-3 -2 -1 0 1 2 3];           //Time  Index of Triangular Pulse
D=10;                           // Delay amount
nd=n+D;                         // Index of Delayed Signal
y=x;                            // Delayed Signal
scf();
subplot(2,1,1);
bar(n,x,0.1,'red');
title('Original Transmitted Signal','color','red','fontsize', 4);
xlabel("Index", "fontsize", 2,"color", "blue");
ylabel("Amplitude", "fontsize", 2, "color", "blue");
subplot(2,1,2);
bar(nd,y,0.1,'yellow');
title('Delayed Signal','color','red','fontsize', 4);
xlabel("Index", "fontsize", 2,"color", "blue");
ylabel("Amplitude", "fontsize", 2, "color", "blue");
w=rand(1,length(x));                    // Noise Generation
nw=nd;                                  //Time Index of noise signal
scf();
bar(nw,w,0.1,'red');
title('Noisy Signal','color','red','fontsize', 4);
xlabel("Index", "fontsize", 2,"color", "blue");
ylabel("Amplitude", "fontsize", 2, "color", "blue");
a = 0.6;              // Attenuation factor of medium 0(max. attenuation)<=a<=1(no attenuation)
// If object is present we receive the signal R(n) =a* x(n-D) + Noise
R=a.*y+w;                       // Original Signal+Noise
nr=nw;                          // Index of received signal at RADAR
```

```
nr_fold=mtlb_fliplr(nr);                //folding time index
R_fold=mtlb_fliplr(R);                  //folding of signal
nmin=min(n)+min(nr_fold);          // Lowest index of y(n)
nmax=max(n)+max(nr_fold);        // Highest index of y(n)
n_received=nmin:nmax;
Received_Presence=convol(x,R_fold);
// Convolution of Original signal and Received Signal in the Presence of Object
scf();subplot(2,1,1);
bar(n_received,Received_Presence,0.1,'red');
title('Correlation in the Presence of Object','color','red','fontsize', 4);
xlabel("Index", "fontsize", 2,"color", "blue");
ylabel("Correlation Value", "fontsize", 2, "color", "blue");
// If object is not present we receive the signal R(n) = Noise
R=w;   nr=nw;                                  // only Noise Signal
nr_fold=mtlb_fliplr(nr);
R_fold=mtlb_fliplr(R);
nmin=min(n)+min(nr_fold);          // Lowest index of y(n)
nmax=max(n)+max(nr_fold);        // Highest index of y(n)
n_received=nmin:nmax;
Received_Absence=convol(x,R_fold);
// Convolution of Original transmitted signal and Received Signal in the Absence of Object
subplot(2,1,2);
bar(n_received,Received_Absence,0.1,'Green');
title('Correlation in the Absence of Object','color','red','fontsize', 4);
xlabel("Index", "fontsize", 2,"color", "blue");
ylabel("Correlation Value", "fontsize", 2, "color", "blue");
```

RESULTS:

Correlation output of two sequences (Seq_x=[1 2 3],Seq_y = [1 2 3])

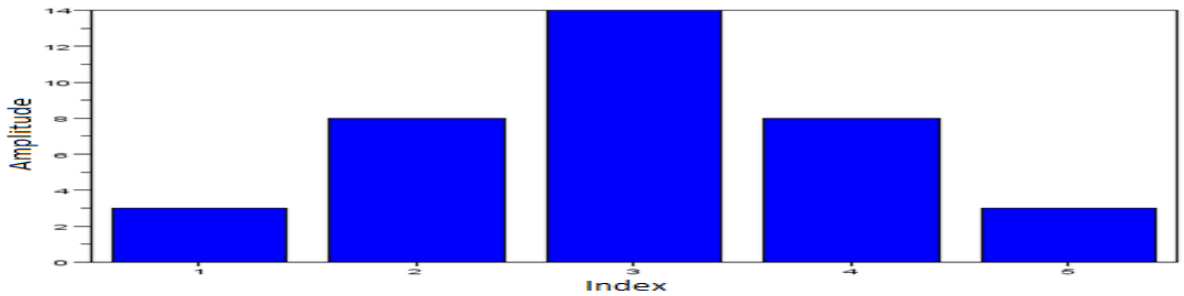

Fig.5.1 Correlation Output

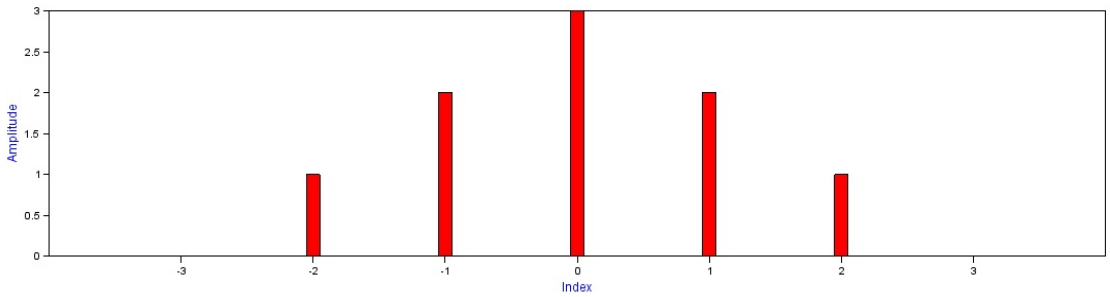

Fig.5.2 Transmitted Signal

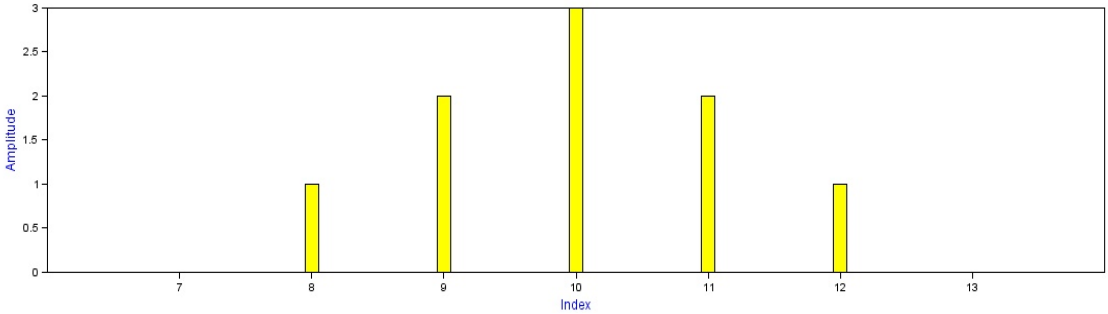

Fig.5.3 Delayed Signal

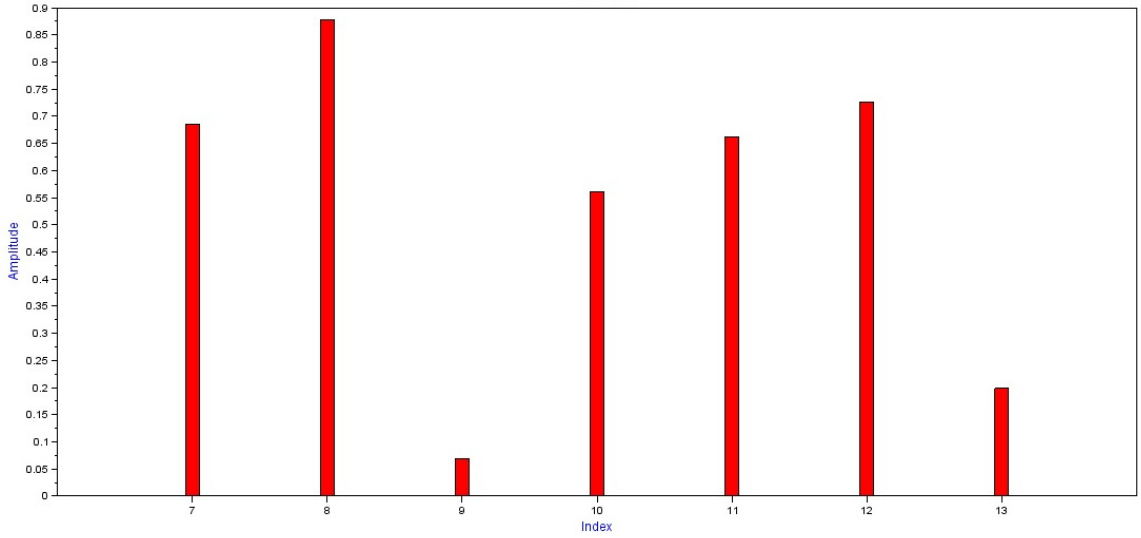

Fig.5.4 Noisy Signal

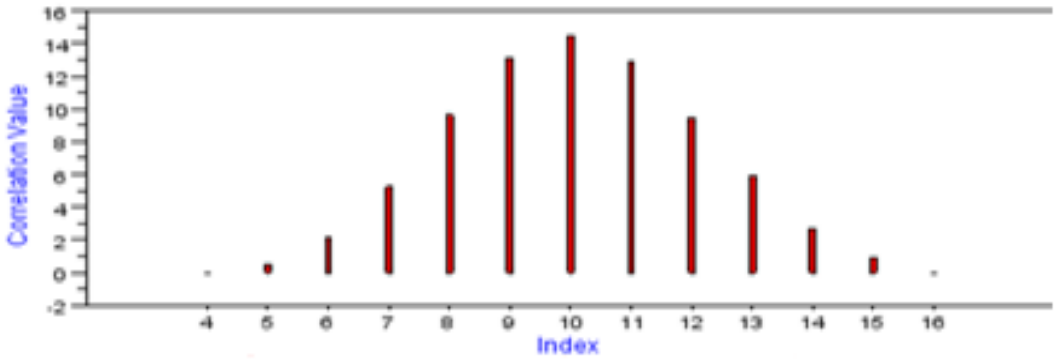

Fig.5.5 Correlation in the Presence of Object

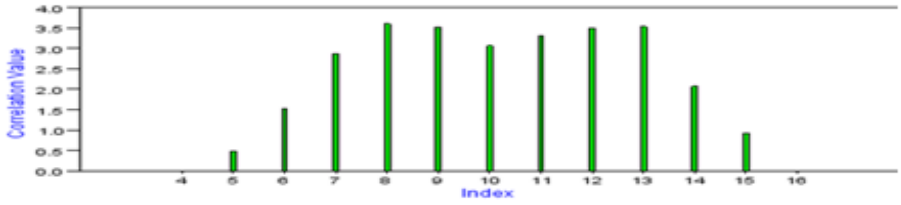

Fig.5.6 Correlation in the Absence of Object

CONCLUSION:

EXPERIMENT –6

ORDINARY DIFFERENTIAL EQUATION

OBJECTIVE: Solving Ordinary Differential Equation of First order and using it in R-L and R-C circuit.

THEORY:

Linear Differential equation of order one,

If $P = P(x)$ and $Q = Q(x)$ are functions of x only, then

$$\frac{dy}{dx} = Px + Q \tag{1}$$

is called linear differential equation of order 1.
With respect to time above equation can be written as follows.

$$ao\frac{di}{dt} + a_1 i = v(t) \tag{2}$$

Solution to differential equation in eq.(2) can be solved using integrating factor. The above equation is called first order linear ordinary non-homogeneous differential equation.
Which is

$$i = e^{-Pt}\int Qe^{Pt}\,dt + ke^{Pt} \tag{3}$$

$$i = i_P + i_C$$

Where i_P is called Particular Integral and
i_C is called Complementary function

The application of Kirchhoff's laws to a circuit containing R-L or R-C elements results in a first order linear differential equation. When a circuit is switched from one condition to another either by a change in the applied voltage or a change in one of the elements, there is a transitional period during which branch currents and voltage change from their former values to new ones. After this transition period called the 'transient', the circuit is said to be in the steady state.

The application of Kirchhoff's voltage law or Kirchhoff's current law to a circuit containing Energy elements results in a differential Equation which can be solved by any of several available methods. This solution consists of two parts-the 'complementary function' and the 'Particular integral'. For equations in circuit analysis, the Complementary function always goes to zero in a relatively short time and hence is the transient part of the solution. The Particular integral is the steady state response. Total solution is called Particular Solution.

SCILAB Function to solve differential equation :
ode(); *//ordinary differential equation solver*

Calling Sequence:
y=ode(y0,t0,t,f)
[y,w,iw]=ode([type,]y0,t0,t [,rtol [,atol]],f [,jac] [,w,iw])
Arguments :
y0 : real vector or matrix, the initial conditions.
t0 : real scalar, the initial time.
t : real vector, the times at which the solution is computed.
f : function, external, string or list, the right hand side of the differential equation.
In the following example, we solve the Ordinary Differential Equation dy/dt=y^2-y sin(t)+cos(t) with the initial condition y(0)=0. We use the default solver.

SAMPLE PROGRAM:

```
function ydot=f(t, y)
   ydot=y^2-y*sin(t)+cos(t);
endfunction;
y0=0;
t0=0;
t=0:0.1:%pi;
y=ode(y0,t0,t,f);
plot(t,y);
```

Application to R-L circuit with Constant Voltage Source

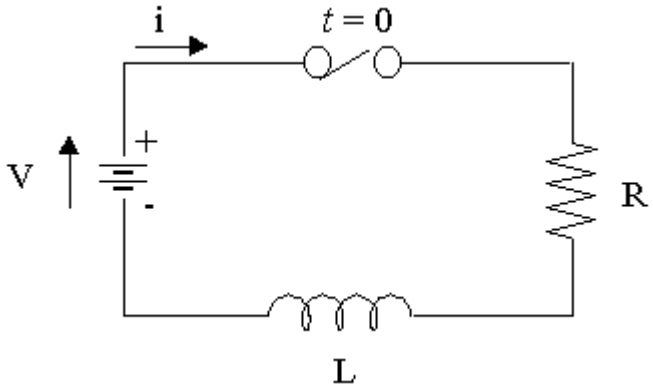

Fig.6.1 Series RL Circuit

The R-L circuit has a Resistor and an Inductor connected in series. A constant voltage V is applied when the switch is closed.

The (variable) voltage across the resistor is given by:

$$V_R = iR \tag{4}$$

The (variable) voltage across the Inductor is given by:

$$V_L = L\frac{di}{dt} \tag{5}$$

Kirchhoff's Voltage law says that the directed sum of the voltages around a circuit must be Zero. This results in the e following differential Equation:

$$Ri + L\frac{di}{dt} = V \tag{6}$$

Once the switch is closed, the current in the circuit is not constant .Instead it will build up from zero to some steady state.

$$\frac{di}{dt} = \frac{V - Ri}{L} \tag{7}$$

SAMPLE PROGRAM:

here, R = 10Ω, L = 1H and (initial voltage)V = 10V.

```
//Differential Eq. of Series RL circuit with constant Input Voltage
function ydot=f(t, i)
 ydot= (V - R*i)/L;
endfunction
V = 10;
L = 1;
R = 10;
i0=10;
t0=0;
t=0:0.001:5;
y=ode(i0,t0,t,f);
clf();
plot(t,y);
```

Application to R-C circuit with Constant Voltage Source

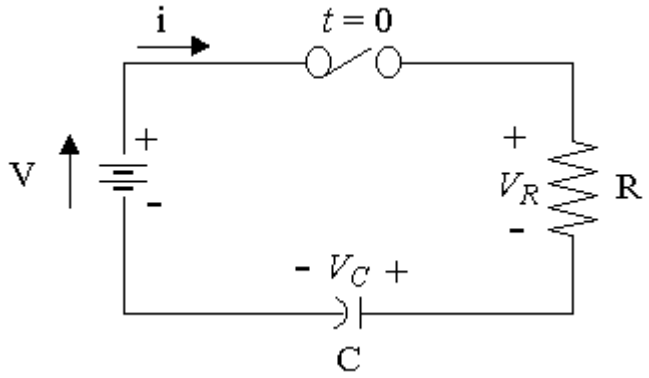

Fig.6.1 Series RC Circuit

The R-C circuit has a Resistor and a Capacitor connected in series. A constant voltage V is applied when the switch is closed.
The voltage across the resistor and capacitor are as follows

$$V_R = iR \tag{8}$$

$$Vc = \frac{1}{C}\int i\,dt \tag{9}$$

Kirchhoff's voltage law says that the total voltage must be zero. So applying this law to a series.
R-C circuit results in the equation:

$$Ri + \frac{1}{C}\int idt = V \tag{10}$$

One way to solve this equation is to turn it into a differential equation with respect to t:

$$R\frac{di}{dt} + \frac{i}{C} = 0 \tag{11}$$

Solving the equation give us:

$$i = \frac{V}{R}e^{-t/RC} \tag{12}$$

SAMPLE PROGRAM:

here, R = 6Ω, C = 1H and (initial condition) I = 1A.

```
//Differential Eq. of Series RC Curcuit with constant Input Voltage
function  ydot=f(t, i)
ydot=-i/(R*C);
endfunction
R = 6;
C = 0.1;
y0=1;    //initial value of current
t0=0;
t=0:0.0001:1;
y=ode(y0,t0,t,f);
plot(t,y);
```

CONCLUSION:

EXPERIMENT –7

STEP RESPONSE OF THE SYSTEM USING XCOS

OBJECTIVE: Verify step response of first and second order system using XCOS.

THEORY:

XCOS is a graphical editor to design hybrid dynamical systems models. Models can be designed, loaded, saved, compiled and simulated. Time domain behavior of a system is an important aspect in designing or describing a system. How quickly a system responds is important. If you have a control system that's controlling a temperature of a heater, so how long it takes the temperature to reach a new steady state is important.

Ability to extrapolate the details of how a system or circuit responds to specific input is important when you design systems. First, we will apply step as a test signal and observe the time domain plot. The shape of the step response helps the designer to conclude how fast it occurs, how much it oscillates, etc. We will observe the time domain behavior of a first order system and second order system respectively.

First Order System:

It is described by,

$$\tau \frac{dy(t)}{dt} + y(t) = G.u(t) \quad (1)$$

$y(t)$ = Response of the system,

$u(t)$ = Unit step Input to the system,

τ = *Time constant* of the system,

G = *DC Gain* of the system.

Systems those are characterized by differential equation (1), have a transfer function of the form:

$$\frac{Y(s)}{U(s)} = \frac{G}{\tau s + 1} \quad (2)$$

A block diagram representation of such a system is provided in Fig.7.1.

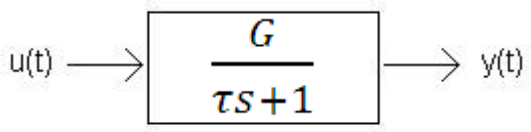

Fig.7.1 First order system

Any system which can be described by a transfer function of the above form,

Variation in *Time constant* (τ) determine how quickly the system moves toward steady state and Variation in *DC gain* (G) of the system determine the amplitude of steady state response when the input settles out to a constant value.

EXAMPLE:

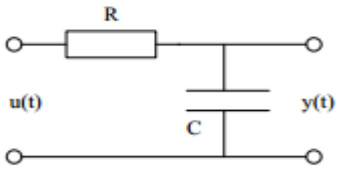

Fig.7.2 RC Circuit

Transfer function of the circuit provided in Fig.7.2 is given by,

$$\frac{Y(s)}{U(s)} = \frac{1}{(RC)s+1} \tag{3}$$

Comparing above equation with eq.(2), *DC gain* (G) of the system = 1, and *time constant* (τ) = RC.

First Order System Model In Xcos:

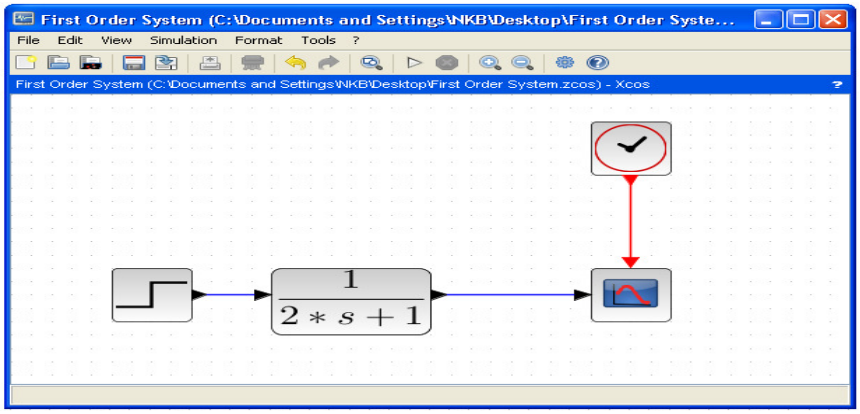

Fig.7.3 First order system model in XCOS

Construct first order system model described by eq.(3) in XCOS using the procedure described below and observe time domain response of the system by varying τ.

Second Order System:

A second order system is very much important. They are the systems that exhibit oscillations. The simplest second order system satisfies a differential equation of this form:

$$\frac{d^2 y(t)}{dt^2} + 2.\xi.\omega_n.\frac{dy(t)}{dt} + \omega_{n^2}.y(t) = G.\omega_{n^2}.u(t) \tag{4}$$

Where,

$y(t)$ = Response of the system,
$u(t)$ = Input to the system,
ξ = *Damping ratio*,
ω_n= *Undamped natural frequency*,
G = *DC gain of the system*.
Systems that can be represented in the form of eq.(5) have a transfer function of the form:

$$\frac{G.\omega_{n^2}}{s^2+2.\xi.\omega_n+\omega_{n^2}} \tag{5}$$

A block diagram representation of such a system is provided in Fig.7.4.

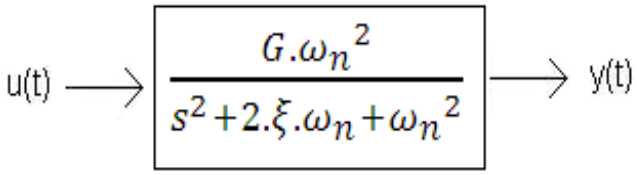

Fig.7.4 Second order system

By varying the parameters you find in a second order system determine various kinds of responses.

(1) *DC gain* (G) of the system, will determine the size of steady state response when the input settles out to a constant value.

(2) *Undamped natural frequency* (ω_n), will determine how fast the system oscillates during any transient response.

(3) *Damping ratio* (ξ), will determine how much the system oscillates as the response decays toward steady state.

If the *Damping ratio* is less than one (ξ <1), then the step response contains a decaying sinusoid with a *damped natural frequency* of $\omega_n\sqrt{(1-\xi)^2}$ and the exponential envelope of the decay is $e^{-\xi\omega_n t}$. This system is called under-damped. Most of the control systems with the exception of robotic control systems are designed with damping ratio less than one. As ξ increases, the response becomes progressively less oscillatory till it becomes unity.

If the *Damping ratio* is equal to unity ($\xi = 1$), then the step response become non-oscillatory and system is called critically-damped. If the *Damping ratio* is greater then unity ($\xi > 1$), then the step response will contains two decaying exponentials and system is called over-damped. If the *Damping ratio* is equal to zero ($\xi = 0$), then response of the system will oscillate forever with its undamped natural frequency (ω_n) and system is called undamped.

Example:

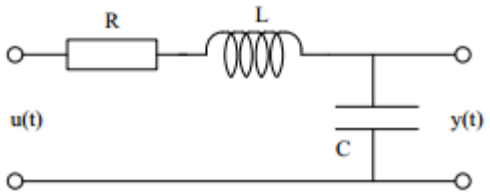

Fig.7.5 RLC Circuit

Transfer function of a circuit provided in figure (4) is given by

$$\frac{Y(s)}{U(s)} = \frac{1}{LCs^2+RCs+1} \quad (6)$$

Comparing above equation with eq.(5), we get

DC Gain of the system (G) = 1

Undamped natural frequency $(\omega_n) = \frac{1}{\sqrt{LC}}$

Damping ratio $(\xi) = \frac{R}{2}\sqrt{\frac{C}{L}}$

Second Order System Model In XCOS:

Construct second order system model described by eq.(6) in XCOS using the procedure described in the following section and observe time domain response of the system by varying ω_n , and ξ .

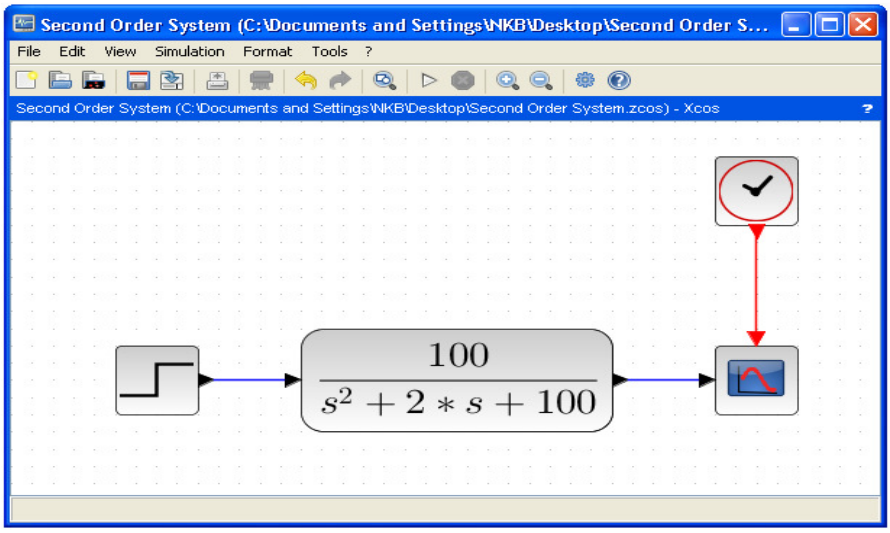

Fig.7.6 Second order system model in XCOS

L = 1H, C = 0.01F: ω_n= 10, $\xi = \frac{R}{2}(0.1)$

By varying the value of R, observe its effect on time response of the system.

Table.7.1 Effect of R on Time Response

R (Ω)	2	8	16	20	40	60
ξ	0.1	0.4	0.8	1.0	2.0	3.0
2. ξ. ω_n	2	8	16	20	40	60

L = 1H, C = 0.1uF: ω_n= 10, $\xi = \frac{R}{2}(0.01)$

By varying the value of R, observe its effect on time response of the system.

Table.7.2 Effect of R on Time Response

R (Ω)	20	80	160	200	400	600
ξ	0.1	0.4	0.8	1.0	2.0	3.0
2. ξ. ω_n	20	80	160	200	400	600

PROCEDURE:

(1) Run XCOS from menu of SCILAB. Create a new XCOS file by selecting 'File/New Diagram'. You will see a blank programming window.

(2) From pallete, select a 'Sources/Step_function' block and drag it to the blank window. This will be a step input to the system. By default it is a unit step that starts at t=0s. This can be changed by right clicking, or double clicking, on the icon and selecting new parameters.

(3) Add a transfer function block (system model) using 'Continuous time systems/CLR'. Default value of the transfer function is '1/(s+1)'. Modify transfer function according to your requirement by double clicking on transfer function block.

(4) Connect the step input to the transfer function block by clicking on a small arrow at input of the transfer function block and dragging it to the step input.

(5) Create an output display using 'Sinks/Cscope' and connect the output of the transfer function block to the scope.

(6) For event handling of scope add Clock_c from event handeling and connect to scope. At this point the system diagram should look like as shown in Figure.

(7) All the blocks can be configured as per requirement of observations.

(8) Implement the first and second order system transfer function provided in examples and observe the time response of a system by changing parameters of transfer function. Print and attach the output graph in your lab book.

CONCLUSION:

EXPERIMENT –8

CONVOLUTION

OBJECTIVE: To find response of the system using convolution.

THEORY:

The definition of convolution is,

$$\boldsymbol{y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\lambda)\, h(t-\lambda) d\lambda} \quad (1)$$

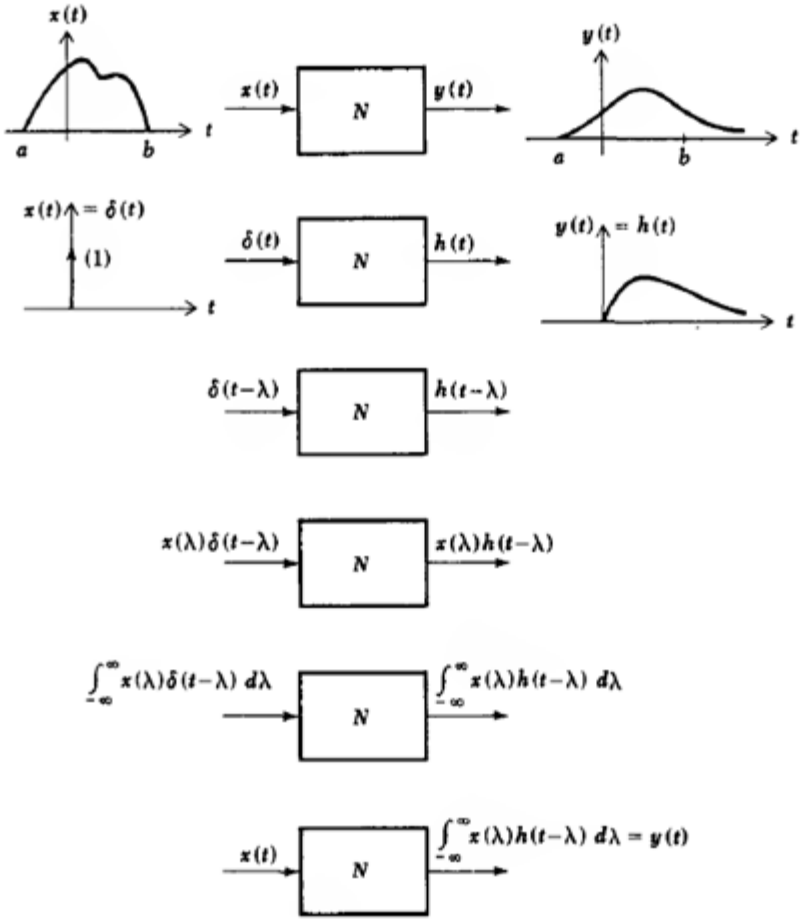

Fig.8.1 Convolution

(1) Consider an electric network N without initial stored energy to which a forcing function x(t) is applied. At some point in this circuit, a response function y(t) is present. It is shown in the Fig.8.1.

(2) The forcing function is arbitrarily shown to exist only over the interval a < t < b. Thus, y(t) can exist only for t > a.

(3) For known form of x(t), how is y(t) described? To answer this question, it is obvious that we need to know something about N. Our knowledge of N is the way it responds when the forcing function is a unit impulse.

(4) That is, we are assuming that we know h(i), the response function resulting from a unit impulse being supplied as the forcing function at t = 0, as shown in above Fig.(1). The function h(i) is commonly called the unit-impulse response function or the impulse response. This is a very important descriptive property for an electric circuit.

(5) Instead of applying the unit impulse at time t = 0, suppose that it were applied at time t = λ. It is evident that the only change in the output would be a time delay. Thus, the output becomes h(t - λ) when the input is δ(t - λ), as shown in Fig.(c).

(6) Next, suppose that the input impulse were to have some strength other than unity. Specifically, let the strength of the impulse be numerically equal to the value of x(t) when t = λ. This value x(λ) is a constant; we know that the multiplication of a single forcing function in a linear circuit by a constant simply causes the response to change proportionately.

(7) Thus, if the input is changed to x(λ)δ(t - λ), then the response becomes x(λ)h(t - λ), as shown in Fig.(d).

(8) Now let us sum this latest input over all possible values of λ and use the result as a forcing function for N.

(9) Linearity decrees that the output must be equal to the sum of the responses resulting from the use of all possible values of λ. Loosely speaking, the integral of the input produces the integral of the output, as shown in Fig.(e). But what is the input now? Using the shifting property of the unit impulse, we see that the input is simply x(t), the original input.

(10) Our question is now answered. When x(t), the input to N, is known, and when h(t), the impulse response of N, is known, then y(t), the output or response function, is expressed by, $y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\lambda)\, h(t-\lambda) d\lambda$

Graphical Method to Compute Convolution:

(1) Plot the given discrete signals x(n) and y(n).
x(n)=[1,1,1] y(n)=[1,1,1]
Draw folded version of y(n) ,i.e y(-n).

(2) Multiply it with x(n) for corresponding index. Add the values and save it to z(n) for Corresponding index.

(3) Shift the y(-n) towards right and multiply it with x(n) for corresponding index. Add the values and save it to z(n) for corresponding index.

(4) Repeat the process till you get nonzero value.

SAMPLE PROGRAM:

```
clc;                //clears data of scilab console
clear all;          //clears all variable data
xdel(winsid());     //closes all previously opened figures
t=10;               //length of rect pulse
w=1;                //amplitude for the rectangular function
y=w*ones(t,1);      //Pulse generation
figure();
plot(y);
xlabel('Time');
ylabel('Amplitude');
title('pulse function');
//%%% exponential %%%%%%
t1=0:1:100;
a=0.1;
x3=exp(-(a*t1));
figure();
subplot(2,2,1);
plot(x3);
title ('Exponential Sequence (a=0.1)');
xlabel('Time Index n');
ylabel('Amplitude');
//%%%%%% convolution by function %%%%%%%
z=conv(y,x3);
subplot(2,2,2);
plot(z);
xlabel('Time');
ylabel('Amplitude');
title('Convolution');
//%%%%%%% exponential %%%%%%%%%%%%
t1=0:1:100;
a=1;
x3=exp(-(a*t1));
subplot(2,2,3);
plot(x3);
title ('Exponential Sequence (a=1)');
xlabel('Time Index n');
ylabel('Amplitude');
//%%%%%%%%% convolution by function %%%%%
z=conv(y,x3);
subplot(2,2,4);
plot(z);
```

xlabel('Time');ylabel('Amplitude');
title('Convolution');

RESULT & ANALYSIS:

Here we can observe that if pulse duration is higher its frequency constants are less. So low pass filter distorts less. If we decrease cut off frequency distortion increases.

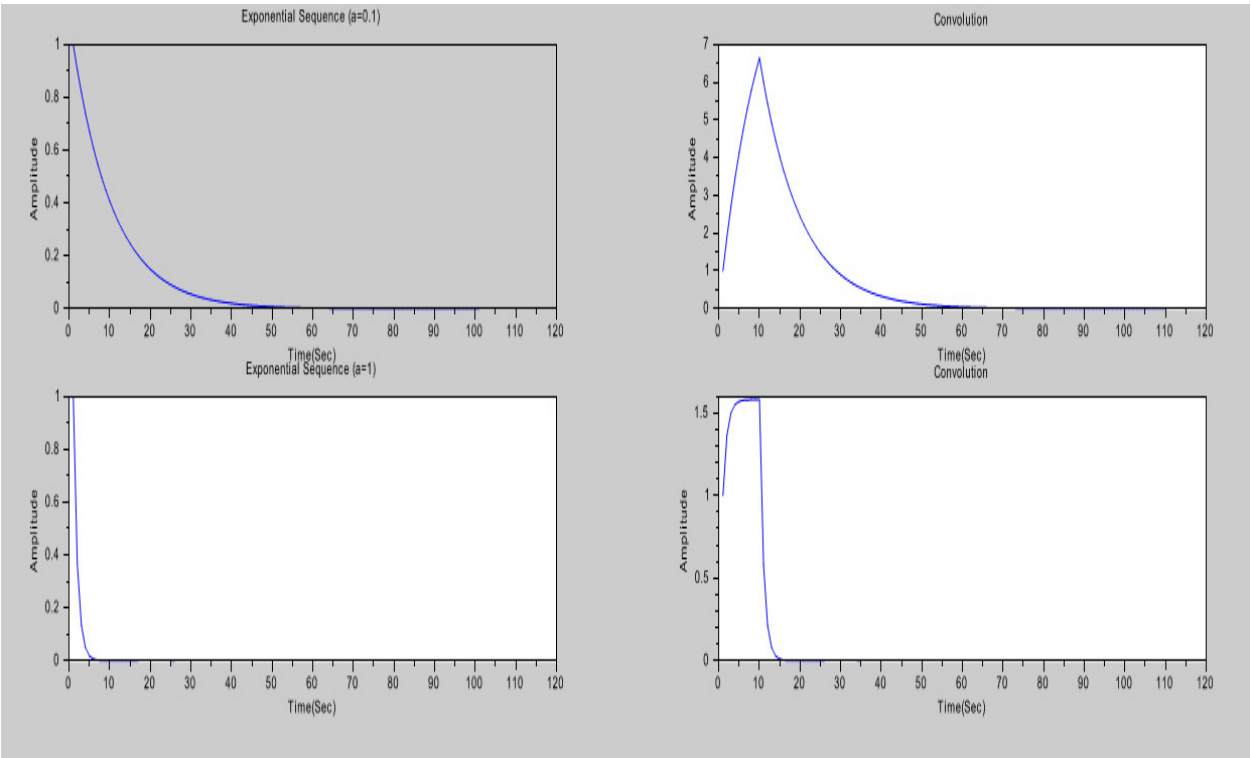

Fig.8.2 Convolution

CONCLUSION:

EXPERIMENT –9

DISCRETE FOURIER TRANSFORM

OBJECTIVE : To Perform operation of DFT & IDFT with and without SCILAB inbuilt function.

THEORY:

Discrete Fourier Transform is a tool for frequency analysis of discrete time signals. Frequency Domain Representation based on type of the Signal:

Table.9.1 Frequency Domain Representation

	Time Domain	**Freq. Domain**
Fourier Series	Continuous & Periodic	Discrete & Aperiodic
Fourier Transform	Continuous & Aperiodic	Continuous & Aperiodic
DTFS(Discrete Time Fourier Series)	Discrete & Periodic	Discrete & Periodic
DTFT(Discrete Time Fourier Transform)	Discrete & Aperiodic	Continuous & Periodic

Difference Between Continuous Time And Discrete Time Signal:

A continuous time signal denoted by X(t) as a continuous function of time, while a signal that is specified only at a discrete value of time is called discrete time signal.In discrete time signal the time interval between two consecutive samples is constant and is called sampling time(duration) T_s. That means continuous time signal X(t) is sampled at a constant rate $F_s=1/T_s$, it becomes a discrete time signal. To emphasize the discrete nature of a signal, it is denoted by X(n) instead of X(t).i.e $t=n\ T_s$.
$X(n)=X(\ n\ T_s)=X(t)$

The relation and the range of Analog and digital frequency:
The Digital frequency (f) is defined by = Analog frequency (F)/ Sampling frequency (F_s)
The unit of Analog frequency (F) is Number of cycles/second.
The unit of sampling frequency (F_s) is Number of samples/second.
So the unit of Digital frequency (f) is Number of cycles/ Number of samples.
The minimum sampling frequency is 2 * Analog frequency (F).
Therefore, The range of digital frequency= $-1/2 \le f \le 1/2$ [$-\pi \le \omega \le \pi$], while the range of Analog frequency is from $-\infty \le F \le \infty$.

Discrete Time Fourier Transform (DTFT):

The DTFT of a discrete time signal is used to represent discrete time aperiodic signal in terms of the complex exponential sequence $e^{-j\omega n}$ where ω is the real frequency variable. The frequency ω of discrete time signal always have range [-π, π] or [0,2π].The DTFT is useful for analyzing discrete time signal in theory. But they are not suitable for implementation on digital hardware or software because X(w) is a continuous function of frequency. They are not numerically computable by any software.

A numerically computable form of the DTFT is called the Discrete Fourier Transform (DFT). It represents the finite length sequence by samples of continuous transform. The DFT is itself a sequence rather than a continuous function.

Discrete Fourier Transform (DFT):

DFT represent sequence x(n) by samples of its spectrum X(w).

$X(k)=\sum_{n-0}^{N-1} x(n)W_N^{nk}$, 0<=k<=N-1 (1)

Where k is the nuber of samples taken from the one period of the spectrum obtained by DTFT. The Inverse Discrete fourier transform (IDFT) is defined by,

$x(n)=\frac{1}{N}\sum_{n-0}^{N-1} X(k)W_N^{-nk}$, 0<=n<=N-1 (2)

DFT computes N equally spaced frequency samples of the DTFT. Both the indices n and K are ranging from 0 to N-1.The integer n is known as time index since it denotes the time index. The integer K denotes discrete frequency and is called frequency index.

SAMPLE PROGRAM :

```
clc;
close;
clear;
x=[1 2 3 4];
N=length(x);
for k=1:N
y(k)=0;
for n=1:N
y(k)=y(k)+(x(n)*exp((-2*%pi*(k-1)*(n-1)*%i)/N));
end
end
disp('Original Sequence');
disp(x);
disp('DFT of Original Sequence');
```

```
disp(y);
for n=1:N
p(n)=0;
for  k=1:N
p(n)=p(n)+((y(k)*exp((%i*2*%pi*(k-1)*(n-1))/N))/N);
end
end
disp('IDFT of Original Sequence');
disp(abs(p));
```

DFT and IDFT using SCILAB inbuilt function:

```
x= [1 2 3 4];
y=fft(x,4);      // y will contain frequency domain samples of x
z=ifft(y,4)       //z will contain time domain samples of
```

RESULTS:

DFT of Original Sequence :

```
    10.        - 2. + 2.i         - 2. - 9.797D-16i    - 2. - 2.i
```

IDFT of Original Sequence :

```
   1.         2.         3.         4.
```

CONCLUSION:

EXCERCISE

(1) State the advantages and drawbacks of using iterative methods in determination of roots of a non-linear equation. State the applications of using iterative met.
(2) Implement five loop ladder network using resistors and find mesh current using SCILAB.
(3) Make a generalized program for Loop and Node Variable Analysis of solution for mesh current and node voltage.
(4) Plot the magnitude and phase response for RC circuit used in this experiment and also plot frequency response by increasing the order of the filter.
(5) Design Butterworth BPF with pass band of 1000 Hz to 2000 Hz, with less than 3 dB of ripple in the pass band, and 80 dB attenuation in the stop bands that are 100 Hz wide on both sides of the pass band. Also in the same filter apply input signal frequency 1500 Hz and plot input and output signal on same figure window.
(6) Design a band stop filter with stop band of 500 Hz to 1600 Hz, with less than 3 dB of ripple in the pass band, and 60 dB attenuation in the stop bands that are 400 Hz wide on both sides of the pass band.
(7) Change values of Resistance or Capacitance to see the effect of time constant.